MICROCHIP

SparX-5

SparX-5 Family of L2/L3 Enterprise 10G Ethernet Switches

Introduction

The SparX-5 Enterprise Ethernet switch family provides a rich set of Enterprise switching features such as advanced TCAM-based VLAN and QoS processing enabling delivery of differentiated services, and security through TCAM-based frame processing using versatile content aware processor (VCAP). IPv4/IPv6 Layer 3 (L3) unicast and multicast routing is supported with up to 18K IPv4/9K IPv6 unicast LPM entries and up to 9K IPv4/3K IPv6 (S,G) multicast groups. L3 security features include source guard and reverse path forwarding (uRPF) tasks. Additional L3 features include VRF-Lite and IP tunnels (IP over GRE/IP).

The SparX-5 switch family features a highly flexible set of Ethernet ports with support for 10G aggregation links, QSGMII, USGMII, and USXGMII.

The device integrates a powerful 1 GHz dual-core ARM® Cortex®-A53 CPU enabling full management of the switch and advanced Enterprise applications.

The SparX-5 switch family targets managed Layer 2 and Layer 3 equipment in SMB, SME, and Enterprise where high port count 1G/2.5G/5G/10G switching with 10G aggregation links is required.

The SparX-5 switch family consists of following SKUs.

VSC7546 SparX-5-64 supports up to 64 Gbps of bandwidth with the following primary port configurations.

- 6 ×10G
- 16 × 2.5G + 2 × 10G
- 24 × 1G + 4 × 10G

VSC7549 SparX-5-90 supports up to 90 Gbps of bandwidth with the following primary port configurations.

- 9 × 10G
- 16 × 2.5G + 4 × 10G
- 48 × 1G + 4 × 10G

In addition, the device supports one 10/100/1000/2500/5000 Mbps SGMII/SerDes node processor interface (NPI) Ethernet port.

Features

This section lists the key device features and benefits.

- Layer 2 and Layer 3 Forwarding
  - IEEE®802.1Q switch with 4K VLANs and 32K MAC table entries
  - Push/pop/translate up to three VLAN tags on ingress and egress
  - RSTP and MSTP support
  - Fully nonblocking wire-speed switching performance for all frame sizes
  - Link aggregation and DRNI per IEEE 802.1AX
  - External bridge port extender role per IEEE 802.1BR
  - IPv4/IPv6 unicast and multicast Layer 2 switching with up to 32K groups and 2K port masks
  - IPv4/IPv6 unicast and multicast Layer 3 forwarding (routing) with reverse path forwarding (RPF) support
  - IGMPv2, IGMPv3, MLDv1, and MLDv2 support

  - IPv4 tunnels including GRE, 6to4, 6rd, 6over4, ISATAP, and 6in4
  - Energy Efficient Ethernet (EEE) (IEEE 802.3az)
- Quality of Service
  - Four megabytes of integrated shared packet memory
  - Two megabytes of integrated shared packet memory
  - Eight QoS classes with a pool of up to 32K queues
  - TCAM-based classification with pattern matching against Layer 2 through Layer 4 information
  - Dual-rate policers selected by VCAP IS2, eight dual-rate priority policers per port, and four single-rate port policers for each port
  - Flexible 4K ingress QoS mappings and 8K egress QoS mappings for VLAN tags and DSCP values
  - 4K egress VLAN tag operations
  - Low latency cut-through forwarding mode
  - Priority-based flow control (PFC) (IEEE 802.1Qbb)
- Security
  - Versatile content aware processor (VCAP™) packet filtering engine using ACLs for ingress and egress packet inspection with four ingress lookups per frame and two egress lookups per egress frame copy
  - Hierarchical VLAN ACLs and router ACLs
  - Storm controllers for flooded broadcast, flooded multicast, and flooded unicast traffic
  - Per-port, per-address registration for copying/redirecting/discarding
  - 64 single-rate policers for ingress ACLs
  - 64 single-rate policers for egress ACLs
- Management
  - VCore-IV™ CPU system with integrated dual-core 1 GHz ARM Cortex-A53 CPU with MMU and DDR3/DDR4 SDRAM controller
  - Integrated ARM Cortex-M3 CPU core for dedicated PCIe bootup and POE management.
  - PCIe 1.x/2.0/3.0 CPU interface
  - CPU frame extraction (eight queues) and injection (two queues) through DMA, which enables efficient data transfer between Ethernet ports and CPU/PCIe
  - JTAG CPU debug interface
  - Configurable 32-bit data plus 8-bit ECC-capable DDR3/DDR4 SDRAM interface supporting up to eight gigabytes (GB) of memory
  - eMMC flash interface
  - Sixty-four pin-shared general-purpose I/Os:
    - Serial GPIO and two LED controllers controlling up to 32 ports with four LEDs each
    - Triple PHY management controller (MIIM)
    - Dual UART
    - Dual built-in two wire serial interface multiplexer
    - External interrupts
    - SFP loss of signal inputs
  - External access to registers through PCIe, SPI, MIIM, or through an Ethernet port with inline versatile register access protocol (VRAP)
  - Per-port counter set with support for the RMON statistics group (RFC 2819) and SNMP interfaces group (RFC 2863)
  - Support for CPU modes with internal CPU only, external CPU only, or dual CPU
- Applications
  - Enterprise L2 managed and L2/L3 (L3-Lite) managed
  - Enterprise edge
  - WiFi aggregation
  - High-end SMB/net café
  - Embedded and control plane switches

- Security appliances
- Base stations and baseband processor interconnect
- Service provider CPEs

Table of Contents

1. Port Configurations

The SparX-5 device supports the following main port configurations.

Table 1-1. Main Port Configurations

Port Configuration	SparX-5-64 VSC7546	SparX-5-90 VSC7549
24 x 1G + 4 x 10G	x	x
48 x 1G + 4 x 10G		x
8 x 2.5G + 2 x 10G	x	x
8 x 2.5G + 4 x 10G	x	x
16 x 2.5G + 4 x 10G		x
8 x 5G + 4 x 10G		x
48 x 1G + 8 x 10G		
8 x 10G		x

2. Product Parameters

All SerDes, packet memory, and configuration tables necessary to support network applications are integrated. The following table lists the primary parameters for the device.

Table 2-1. Product Parameters

Features and Port Configurations	VSC7546	VSC7549
Maximum bandwidth	64 Gbps	90 Gbps
Maximum number of ports	64	64
Max number of QSGMII ports	12	12
Max number of USGMII ports	6	6
Max number of 10GUSXGMII ports	6	9
Max number of 5GUSXGMII ports	12	18
Max number of 1G ports	64	64
Max number of 1G SGMII ports	32	32
Max number of 100FX ports	24	24
Max number of 2.5G ports	24	36
Max number of 5G ports	12	18
Max number of 10G ports	6	9
Max number of 25G ports	0	0
SerDes5G lanes	12	12
SerDes10G lanes	12	12
SerDes25G lanes	8	8
NPI port (5G)	1	1
Layer 2 Switching		
Packet buffer	32 Mb	32 Mb
MAC table size	32K	32K
Layer 2 multicast port masks	2K	2K
Super VCAP blocks (3K x 52 bits per block)	10	10
VCAP CLM entries (52 bits) per super VCAP block	3K	3K
VCAP LPM entries (52 bits) per super VCAP block	3K	3K
VCAP IS2 entries (312 bits) per super VCAP block	512	512
VCAP ES0 entries (52 bits)	4K	4K
VCAP ES2 entries (312 bits)	1K	1K

.....continued

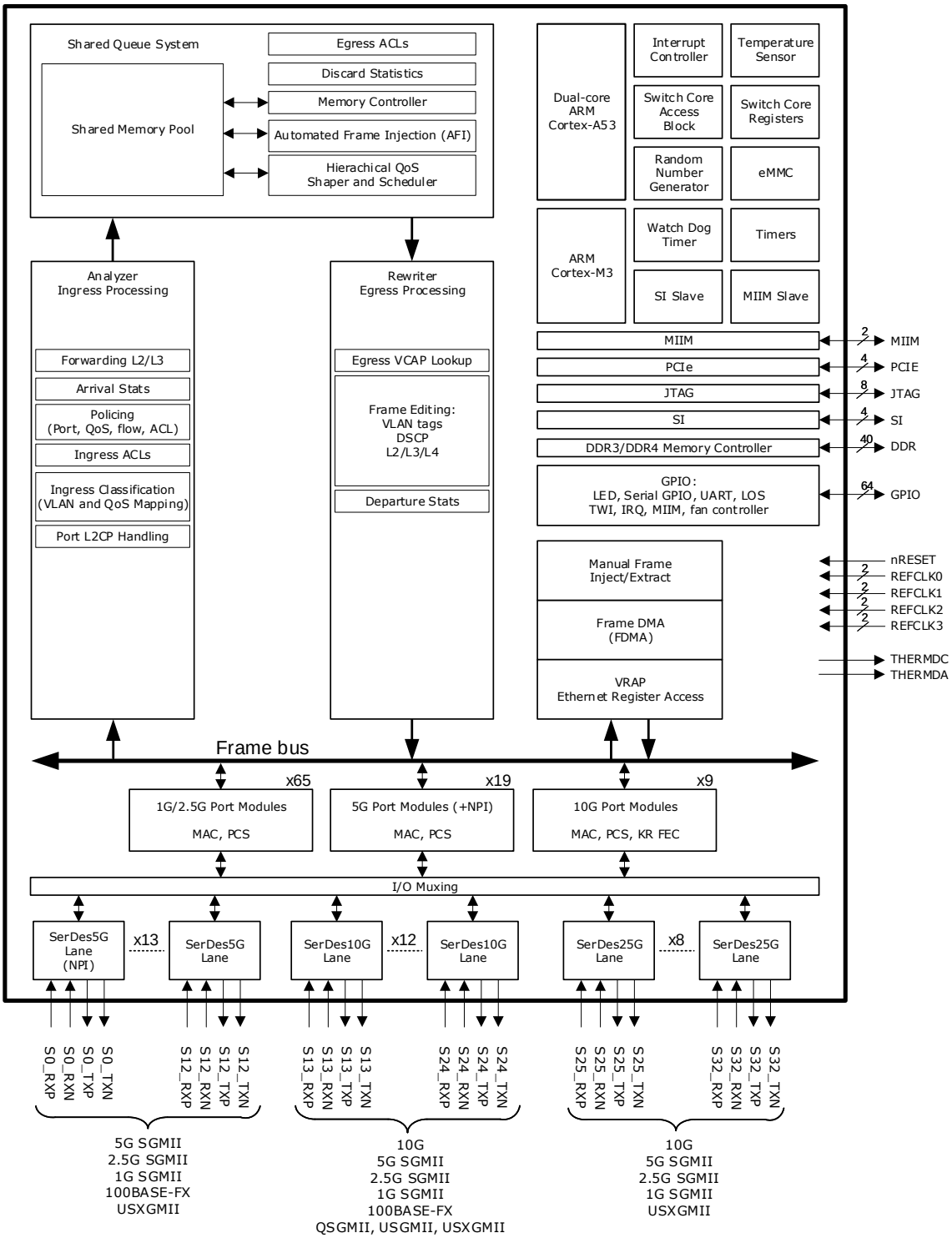
Features and Port Configurations	VSC7546	VSC7549
Layer 3 Routing		
Router legs	511	511
IPv6 prefix entries	512	512
IP unicast routes/hosts per super VCAP block allocated to VCAP LPM for unicast routing (64-bit DIP for IPv6)	IPv4: 3K IPv6: 1.5K	IPv4: 3K IPv6: 1.5K
Next-hop/ARP table entries	2K	2K
IP (S,G) or (*, G) multicast groups per super VCAP block allocated to VCAP LPM for multicast routing	IPv4: 1.5 IPv6: 512	IPv4: 1.5 IPv6: 512
Multicast router leg masks	2K	2K
ECMPs	16	16
IPv4 tunnels (GRE, 6to4, 6rd, 6over4, ISATAP, 6in4)	1K	1K
VRF-Lite instances	64	64
Quality of Service and Security		
Queues	32K	32K
Ingress QoS mapping table entries	4K	4K
Egress QoS mapping table entries	8K	8K
Ingress security enforcement (ACL) rules (312 bits) per super VCAP block allocated to VCAP IS2	512	512
Egress security enforcement (ACL) rules (312 bits)	2K	2K

3. Functional Overview

This section provides an overview of all the major blocks and functions involved in the forwarding operation in the same order as a frame traverses through the device. It also outlines other major functionality of the device, such as the CPU subsystem.

The following figure shows the VSC7549 block diagram. The block diagrams for VSC7546 is identical, except for the port module configurations.

Figure 3-1. Block Diagram



3.1 Frame Arrival in Ports and Port Modules

The Ethernet interfaces receive incoming frames and forward these to the port modules.

Each port module contains a media access controller (MAC) that performs a full suite of checks, such as VLAN tag-aware frame size checking, frame check sequence (FCS) checking, and pause frame identification.

Each port module connects to a SerDes macro and contains a physical coding sublayer (PCS), which performs 4b/5b, 8b/10b, or 64b/66b encoding, auto-negotiation of link speed and duplex mode, and monitoring of the link status.

Symmetric and asymmetric pause flow control are both supported as well as priority-based flow control (IEEE 802.1Qbb).

All Ethernet ports support Energy Efficient Ethernet (EEE) according to IEEE 802.3az. The shared queue system is capable of controlling the PCS operating states, which are active or low power. The PCS understands the line signaling as required for EEE. This includes signaling of active, sleep, quiet, refresh, and wake.

3.1.1 1G Line Ports

1G line ports operate in one of the following modes.

- 1000BASE-X. The 1000BASE-X interface connects directly to a 1000BASE-X SFP optical module. Autonegotiation is supported for pause settings and remote fault signaling. Operation is always 1000 Mbps and full duplex. 1G backplane Ethernet, 1000BASE-KX, is fully supported, including autonegotiation.
- 100BASE-FX. The 100BASE-FX interface connects directly to a 100BASE-FX SFP optical module. Autonegotiation is not specified for 100BASE-FX and is not supported. Operation is always 100 Mbps and full duplex.

3.1.2 1G SGMII Line Ports

1G SGMII line ports operate in 10/100/1000 Mbps SGMII. The SGMII interface connects to an external copper PHY device or copper SFP optical module with SGMII interface. In this mode, autonegotiation is supported for link speed, duplex settings, pause settings, and remote fault signaling. Full-duplex is supported for all speeds, while half-duplex is supported for 10/100 Mbps.

3.1.3 2.5G Line Ports

The 2.5G interface connects directly to an SFP optical module. Operation is always 2500 Mbps and full duplex. 2.5G backplane Ethernet is also supported.

3.1.4 1G QSGMII Line Ports

1G QSGMII line ports operate in quad 10/100/1000 Mbps QSGMII. The QSGMII interface connects to an external copper PHY device. In this mode, autonegotiation is supported for link speed, duplex settings, pause settings, and remote fault signaling. Full-duplex is supported for all speeds, while half-duplex is supported for 10/100 Mbps.

3.1.5 5G Line Ports

The 5G interface connects directly to an SFP optical module or a backplane. Operation is always 5 Gbps and full duplex. 5G backplane Ethernet (5GBASE-KR) is supported.

3.1.6 5G USXGMII Line Ports

5G USXGMII line ports operate in dual 10/100/1000/2500 Mbps USXGMII mode. The USXGMII interface connects to an external copper PHY device. In this mode, autonegotiation is supported for link speed, duplex settings, pause settings. Full-duplex is supported for all speeds, while half-duplex is supported for 10/100 Mbps.

3.1.7 10G Line Ports

10G line ports can operate in one of the following modes.

- 10 Gbps SFI. The interface connects directly to a 10GBASE-R SFP+ optical module or an external PHY device. Auto-negotiation is supported for pause settings, remote fault signaling, and backplane Ethernet functions. The 10G, 5G. 2.5G, and 1G operation is supported for SFP+ optical modules, and operation is always full duplex.

- 10G backplane Ethernet 10GBASE-KR (serial backplane) is fully supported, including autonegotiation, training, and 10GBASE-KR FEC.
- When used with external devices capable of WAN-PHY (10GBASE-W) operation, the 10G line ports support pacing operation as specified for 10GBASE-W operation.

3.1.8 10G USXGMII Line Ports

10G USXGMII line ports operate in one of the following modes.

- Quad port 10/100/1000/2500 Mbps mode.
- Dual port 10/100/1000/2500/5000 Mbps mode

The USXGMII interface connects to an external copper PHY device. In this mode, autonegotiation is supported for link speed, duplex settings, pause settings. Full-duplex is supported for all speeds, while half-duplex is supported for 10/100 Mbps.

3.1.9 10G USGMII Line Ports

10G USGMII line ports operate in octal 10/100/1000 Mbps port mode. The USGMII interface connects to an external octal copper PHY device. In this mode, autonegotiation is supported for link speed, du-plex settings, pause settings, and remote fault signaling. Full-duplex is supported for all speeds, while half-duplex is supported for 10/100 Mbps.

3.2 Basic Classification

Basic frame classification in the ingress processing module (the analyzer) receives all frames before further classification processing is performed. The basic classifier uses configurable logic to classify each frame to a VLAN, QoS class, drop precedence (DP) level, DSCP value, and an aggregation code. This information is carried through the switch together with the frame and affects policing, drop precedence marking, statistics collecting, security enforcement, Layer 2 and Layer 3 forwarding, and rewriting.

The basic classifier also performs a general frame acceptance check. The output from the basic classification may be overwritten or changed by the more intelligent advanced classification using the TCAM-based VCAP.

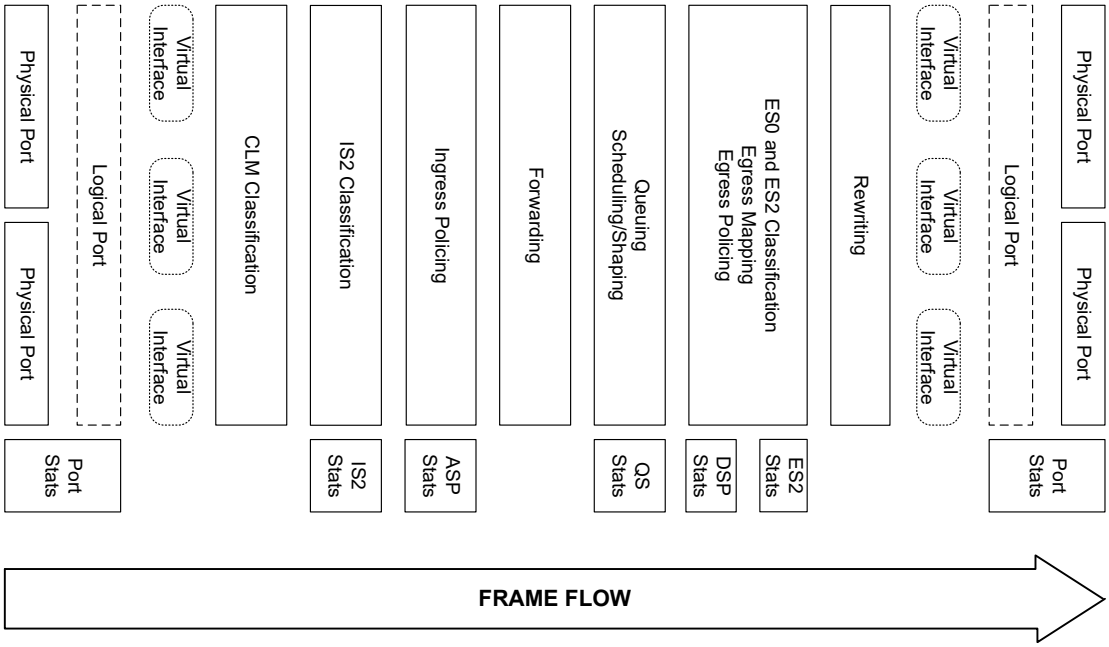
3.3 Enterprise Ethernet Flows

The Microchip flow-aware architecture enables a rich set of Enterprise Ethernet features using multiple lookups into VCAP classification matching (CLM). Up to 64 ingress DSCP mapping tables (64 entries each) can be used for more efficient TCAM utilization. Enterprise Ethernet flow policy attributes include:

- Mapping to a dual-rate policer. Multiple flows can map to one policer. Up to eight policers can be assigned per flow to implement the per-COS per-flow bandwidth profiles.
- Mapping to a VLAN (MAC table) resource and a flow-specific port map.
- Mapping to flow-specific arrival and departure frame modification instructions.
- Flow-specific QoS handling instructions including remarking and hierarchical QoS treatment.
- Per-flow arrival, departure, and discard statistics. Multiple flows can map to one statistics set. Up to eight statistics sets can be assigned to a flow to implement the per-CoS per-flow bandwidth profiles.

The overall processing architecture is shown in the following figure.

Figure 3-2. Enterprise Ethernet Flow Processing



3.4 Security and Control Protocol Classification

Before being passed on to the Layer 2 forwarding, all frames are inspected by the VCAP IS2 for security enforcement and control protocol actions.

The action associated with each IS2 entry (programmed into the IS2 action RAM) includes frame filtering, single leaky bucket rate limiting (frame or byte based), snooping to CPU, redirecting to CPU, mirroring, and accounting.

The VCAP IS2 engine embeds powerful protocol awareness for well-known protocols such as LLC, SNAP, ARP, IPv4, IPv6, and UDP/TCP. IPv6 is supported with full matching against both the source and the destination IP addresses.

For each frame, up to four lookup keys are generated and matched against the VCAP entries. It is possible to build match hierarchies of port, VLAN, and ingress/egress router ACLs.

Although VCAP IS2 allows actions per egress port, true egress ACLs with lookups per egress frame copy can be achieved through VCAP ES2.

3.5 Policing

Each frame is subject to one or more of the following policing operations.

- Dual-rate flow policers: Flow classification selects which flow policer to use.
- Dual-rate bundle policers: Flow classification selects which bundle policer to use. Bundle policers are placed immediately after the flow policers and can police one or more flows.
- Single-rate priority policers: Ingress port number and QoS class determine which policer to use.
- Single-rate port policers: Ingress port number determines which policer to use.
- VCAP single-rate policers: IS2 action selects which VCAP single-rate policer to use.
- Single-rate global storm policers: Frame characteristics, such as broadcast, unicast, multicast (BUM), or learn-frame, select which policer to use.
- Single-rate per-flow broadcast, unicast, and multicast (BUM) policers for flooded frames. Flow classification and destination MAC address select which policer to use.

Policers can measure frame rates (frames per second) or bit rates (bits per second). Frames classified to a flow can trigger the following policers.

- One single-rate BUM policer
- One dual-rate flow policer
- One dual-rate bundle policer
- One single-rate VCAP policer
- Four single-rate port policers
- Eight single-rate storm policers

Frames not classified to a flow can trigger the following policers:

- One single-rate BUM policer
- One dual-rate priority policer or one dual-rate VCAP policer
- One dual-rate port policer
- One single-rate VCAP policer
- Four single-rate port policers
- Eight single-rate storm policers

Dual-rate policers support both color-blind and color-aware operation. The initial frame color is derived from the drop precedence level from the frame classification. The MEF coupling mode is also configurable for each policer.

Dual-rate policers ensure service level agreement (SLA) compliance. The outcome of this policing operation is to mark each accepted frame as in-profile (green) or out-of-profile (yellow). Yellow frames are treated as excess or discard-eligible and green frames are committed. Frames that exceed the yellow/excess limits are discarded (red).

Each frame is counted in associated statistics counters reflecting the flow and the frame's color (green, yellow, red). The statistics counters count bytes and frames.

The four single-rate port policers and eight single-rate storm policers can be programmed to limit different types of traffic such as CPU-forwarded frames, learn frames, known or unknown unicast, multicast, or broadcast.

3.6 Layer 2 Forwarding

After the policers, the Layer 2 forwarding block of the analyzer handles all fundamental Layer 2 forwarding operations and maintains the associated MAC table, the VLAN table, and the aggregation table. The device implements a 32K MAC table and a 4K VLAN table.

The main task of the analyzer is to determine the destination port set of each frame. The Layer 2 forwarding decision is based on various information such as the frame's ingress port, source MAC address, destination MAC address, and the VLAN identifier, as well as the frame's VCAP actions, mirroring, and the destination port's link aggregation configuration.

Layer 2 forwarding of unicast and Layer 2 multicast frames is based on the destination MAC address and the VLAN.

The switch can also L2-forward IPv4 and IPv6 multicast frames using additional Layer-3 information, such as the source IP address. The latter enables source-specific IPv4 multicast forwarding (IGMPv3) and source-specific IPv6 multicast forwarding (MLDv2). This process of L2-forwarding multicast frames using Layer 3 information is called "snooping", which is different from L3-forwarding (routing) of multicast frames.

The following describes some of the contributions to the Layer 2 forwarding.

- VLAN Classification—VLAN-based forward filtering includes source port filtering, destination port filtering, VLAN mirroring, and asymmetric VLANs.
- Flow Classification—flow-based forward filtering includes destination port filtering and forced forwarding.
- Security Enforcement—the security decision made by the VCAP IS2 can, for example, redirect the frame to the CPU based on security filters.
- MSTP —the VLAN identifier maps to a Multiple Spanning Tree instance, which determines MSTP-based destination port filtering.
- MAC Addresses —destination and source MAC address lookups in the MAC table determine if a frame is a learn frame, a flood frame, a multicast frame, or a unicast frame.

- Learning—by default, the device performs hardware learning on all ports. However, certain ports could be configured with secure learning enabled, where an incoming frame with unknown source MAC address is classified as a “learn frame” and is redirected to the CPU. The CPU performs the learning decision and also decides whether the frame is forwarded.
- Learning can also be disabled. If learning is disabled, it does not matter if the source MAC address is in the MAC table.
- Link Aggregation—a frame targeted to a link aggregate is processed to determine which physical port of the link aggregate group the frame must be forwarded to. The device supports hash-based load-balancing based on Layer 2 - 4 frame protocol fields as well as conversation sensitive distribution based on the frame’s outer VID.
- Mirroring—mirror probes may be set up in different places in the forwarding path for monitoring purposes. As part mirroring, a copy of the frame is sent either to the CPU or to another port.

3.7 Layer 3 Forwarding

The device supports Layer 3 forwarding (routing), which is performed by the analyzer. With Layer 3 forwarding, the IPv4 or IPv6 addresses determine the destination port set and Layer 3 processing is performed on IP header. The Layer 3 forwarding process also replaces the arrival Layer-2 Ethernet header (including MAC addresses and VLAN tags) with a new Layer 2 Ethernet header after departure from the switch.

The analyzer supports 511 router legs, and supports virtual router redundancy protocol (VRRP). Ingress router legs are addressed using classified arrival VLAN and DMAC address.

If an arrival frame is determined to belong to an ingress router leg and routing is enabled, Layer 3 forwarding is applied. Note that this is in addition to any Layer 2 forwarding, so for example, an arrival multicast frame may be Layer 2 forwarded on the classified arrival VLAN and also Layer 3 forwarded to one or more additional VLANs. A Layer 3 forwarded frame is always Layer 2 forwarded in the departure VLAN.

Layer 3 forwarding first checks the IP header for validity. IP header checks include IP version, header length, and header checksum. The time-to-live (TTL) or Hop Limit values are also checked for validity and decremented if the packet is forwarded.

The analyzer then searches the longest prefix match (LPM) table for an exact match of the destination IP address. If there is not an exact match, the table is searched for the best partial match. If there is no partial match in the LPM table, the frame is Layer 3 forwarded to the location of the default gateway.

To improve the scale of the LPM table for IPv6 unicast entries, Layer 3 forwarded IPv6 unicast frames are subject to an IPv6 prefix match prior to looking up the LPM table. The IPv6 prefix match uses the 64 most significant bits of the IPv6 destination address and passes a pointer to the LPM table. As a result, an IPv6 unicast entry is only twice the size of an IPv4 entry.

With Layer 3 forwarding, each egress frame copy uses an egress router leg to determine the per-copy egress encapsulation. There can be up to 16 egress router legs associated with one forwarding entry in support of equal cost multipath (ECMP) functionality, where multiple forwarding paths are used between adjacent routers for increased forwarding bandwidth.

Reverse path forwarding (RPF) is optionally performed on multicast and unicast (uRPF) packets as a security check. This source IP address check helps detect and prevent address spoofing.

Multicast frames can be Layer 2 forwarded on the arrival VLAN as well as Layer 3 forwarded to different VLANs. Layer 3 forwarding of IP multicast is different from Layer 2 forwarding of IP multicast using IGMP/MLD snooping in the following ways.

- IP header checks and TTL processing are performed
- The Ethernet header is swapped
- The departure VLANs may be different from the arrival VLANs
- Multiple frame copies can be generated per physical port

Network control such as ICMP and ARP are redirected to the CPU, along with malformed packets, packets with expired TTL, and packets with options.

The Layer 3 forwarding engine supports various tunneling schemes where IP frames can be carried in IPv4 tunnels. The following tunnels are supported for transparent operation, encapsulation, and decapsulation.

- GRE/mGRE tunnel
- 6to4 (RFC3056)
- 6rd (RFC5569)
- 6over4 (RFC2529)
- ISATAP (RFC5214)
- 6in4 (RFC4213)

GRE tunnels can be used with or without GRE checksums. The GRE checksum is checked for arrival frames prior to transparent operation or tunnel decapsulation. GRE tunnel encapsulations performed by the device do not use GRE checksums.

Virtual routing over Ethernet (VRF-Lite) is supported with up to 64 virtual routers. Each virtual router uses its own set of LPM entries for forwarding. Classification to a virtual router instance is performed in VCAP CLM.

3.8 Shared Queue System and Hierarchical Scheduler

The analyzer provides the destination port set of a frame to the shared queue system. It is the queue system's task to control the frame forwarding to all destination ports.

The shared queue system embeds memory that can be shared between all queues and ports. Frames are mapped to queues through a programmable mapping function allowing ingress ports, egress ports, QoS classes, and flows to be taken into account. The sharing of resources between queues and ports is controlled by an extensive set of thresholds.

Each egress port implements default schedulers and shapers as shown in the following illustration. Per egress port, the scheduler sees the outcome of aggregating the egress queues (one per ingress port per QoS class) into eight QoS classes.

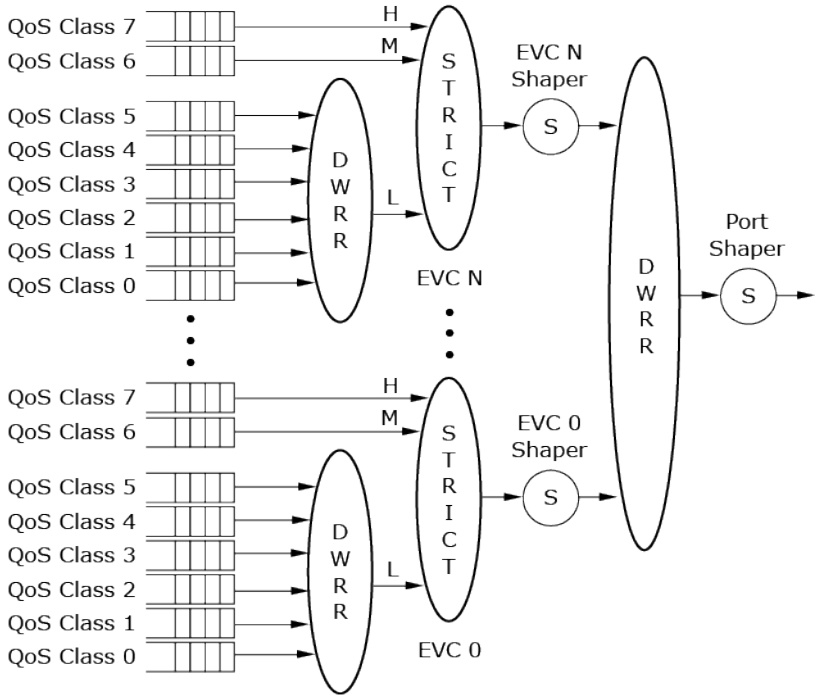
The shared queue system supports a cut-through mode of operation enabled per ingress port per QoS class. Cut-through is active when a frame is destined to a port with the same or lower port speed than the ingress port and with no other frames queued for transmission. Due frame analysis being performed prior to cut-through, frames less than 148 bytes will not benefit from cut-through.

3.8.1 Class-Based Queuing (Default Configuration)

By default, aggregation is done in a deficit weighted round robin fashion (DWRR) with equal weights to all ingress ports within a QoS class, so byte-accurate weighted round robin scheduling between ingress ports is performed for each QoS class.

The following figure shows the default scheduler-shaper configuration. Hierarchical scheduling-shaping is also possible. All shapers shown are dual leaky bucket shapers.

Figure 3-3. Default Scheduler-Shaper Configuration



Scheduling between QoS classes within the port can use one of the three following methods.

- Strict priority scheduling—frames with the highest QoS class are always transmitted before frames with lower QoS class.
- Deficit weighted round robin (DWRR) scheduling—each QoS class serviced using DWRR sets a DWRR weight ranging from 0 to 31.
- Combination of strict priority and DWRR scheduling—the default configuration is shown, where QoS classes 7 and 6 use strict priority while QoS classes 5 through 0 use DWRR. But any split between strict and DWRR QoS classes can be configured.

3.9 Egress Security Classification

The shared queue system generates a frame copy per egress destination port. Before being passed on to the egress queues, each frame copy is inspected the VCAP ES2 for security enforcement actions.

The VCAP ES2 engine is similar to the VCAP IS2 engine and embeds powerful protocol awareness for well-known protocols such as LLC, SNAP, ARP, IPv4, IPv6, and UDP/TCP. IPv6 is supported with full matching against both the source and the destination IP addresses.

For each frame, up to two lookup keys are generated and matched against the VCAP entries. Since VCAP ES2 is located after frame replication, it uses the departure VLAN for Layer 3 forwarded frames in its lookup keys.

The action retrieved from VCAP ES2 is applied to the frame prior to storing the frame copy in the egress queues.

3.10 Rewriter and Frame Departure

Before transmitting the frame on the egress line, the rewriter can modify selected fields in the frame such as Ethernet headers and VLAN tags, IPv4/IPv6 fields, IP tunnels, and FCS. The rewriter also updates the frame's COS and color indications such as DEI, PCP, and DSCP. Departure statistics are also kept based on the classified VLAN.

By default, the egress VLAN actions are determined by the egress port settings and classified VLAN. These include basic VLAN operations such as pushing a VLAN tag, untagging for specific VLANs, and simple translations of DEI, PCP, and DSCP.

By using the egress VCAP ES0, significantly more advanced VLAN tagging operations can be achieved. ES0 enables pushing up to three VLAN tags and flexible VLAN tag translation for per-flow VLAN tag/TPID, PCP, DEI, and DSCP control. The lookup by VCAP ES0 includes classified VLAN, flow identifier, egress port, and COS/color indications.

Supports per-router-leg control of Ethernet link layer encapsulation and VID, as well as modification of other Layer 3 fields such as IPv4 TTL, IPv4 checksum, and IPv6 hop limit. IP tunnels may be pushed or popped.

The rewriter pads frames to minimum legal size if needed, and updates the FCS if the frame was modified before the frame is transmitted.

The egress port module controls the flow control exchange of pause frames with a neighboring device when the interconnection link operates in full-duplex flow control mode.

3.11 CPU Port Module

The CPU port module (DEVCPU) contains eight CPU extraction queues and two CPU injection queues. These queues provide an interface for exchanging frames between the internal CPU system and the switch core. An external CPU using the serial interface can also inject and extract frames to and from the switch core by using the CPU port module.

In addition, any Ethernet interface on the device can be used for extracting and injecting frames. The Ethernet interface used in this way is called the node processor interface (NPI) and is typically connected to an external CPU.

Injected frames may be prepended with an injection header to control processing and forwarding of these frames. Extracted frames may be prepended with an extraction header to supply frame arrival and classification information associated with each frame. These headers may be used by internal CPU or external CPU.

The switch core can intercept a variety of different frame types and copy or redirect these to the CPU extraction queues. The classifier can identify a set of well-known frames, such as IEEE reserved destination MAC addresses (BPDUs, GARPs, CCM/Link trace), as well as IP-specific frames (IGMP, MLD). Security VCAP IS2 provides another flexible way of intercepting all kinds of frames, such as specific OAM frames, ARP frames or explicit applications based on TCP/UDP port numbers. In addition, frames can be intercepted based on the MAC table, the VLAN table, or the learning process.

Whenever a frame is copied or redirected to the CPU, a CPU extraction queue number is associated with the frame and used by the CPU port module when enqueuing the frame into the eight CPU extraction queues. The CPU extraction queue number is programmable for every interception option in the switch core.

3.12 CPU Subsystem

The device contains a powerful dual-core 1 GHz ARM Cortex-A53 microprocessor, a high bandwidth Ethernet Frame DMA engine, and a DDR3/DDR4 controller supporting up to 8 gigabytes (GB) of memory. This complete system-on-chip supports Linux or embedded operating systems, enabling full management of the switch and advanced software applications. In addition to the Cortex-A53 microprocessor, the CPU system contains an ARM Cortex-M3 microprocessor dedicated to handling power-over-Ethernet (POE) management and PCIe bootup.

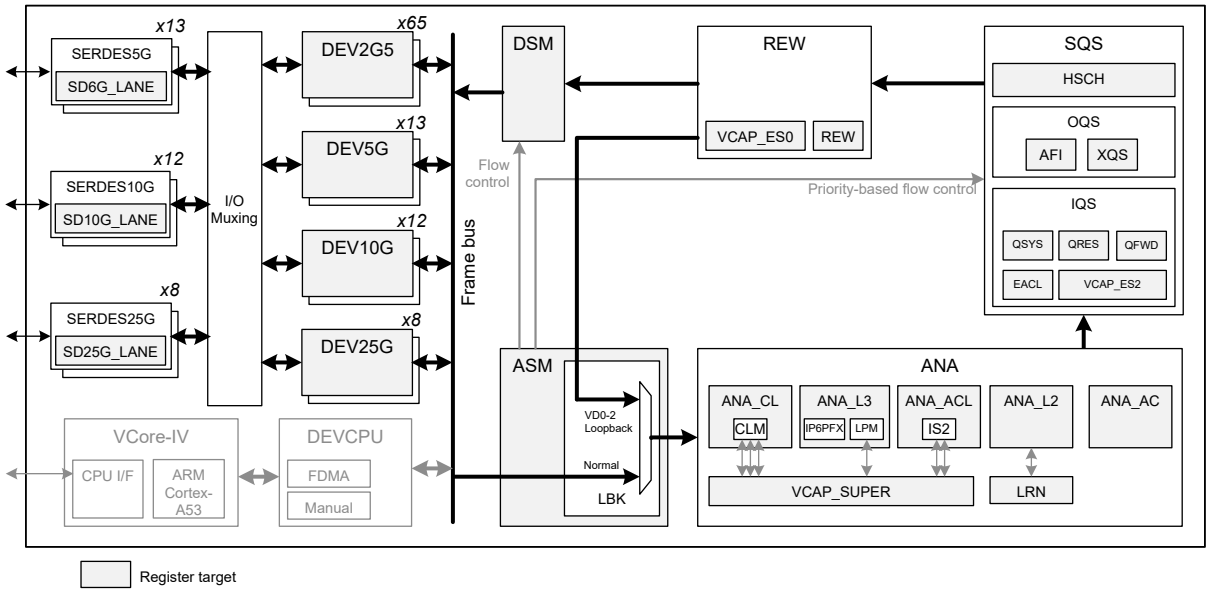
The device supports external CPU register access by the on-chip PCIe 1.x/2.0/3.0 endpoint controller, by specially formatted Ethernet frames on the NPI port (Versatile Register Access Protocol), or by register access interface using SPI protocol. External CPUs can inject or extract Ethernet frames by the NPI port, by PCIe DMA access, or by register read/writes (using any register-access interface).

4. Functional Descriptions

This section describes the functional aspects of the device, including available configurations, operational features, and testing functionality. It also defines the device setup parameters that configure the device for a particular application.

The following figure shows an RTL block diagram of the physical blocks in the device and how the blocks interconnect. The functional aspects of each block is provided in following sections. The grayed-out blocks represent the VCore-IV and DEVCPU blocks. For more information about the VCore-IV and DEVCPU blocks, see 5. VCore System and CPU Interfaces.

Figure 4-1. Physical Block Diagram



4.1 Register Notations

This datasheet uses the following general register notations.

<TARGET>:<REGISTER_GROUP>:<REGISTER>.<FIELD>

<REGISTER_GROUP> is not always present. In that case the following notation is used.

<TARGET>::<REGISTER>.<FIELD>

When a register group does exist, it is always perpended with a target in the notation.

In sections where only one register is discussed or the target (and register group) is known from the context, the <TARGET>:<REGISTER_GROUP>: may be omitted for brevity leading to the following notation.

<REGISTER>.<FIELD>

When a register contains only one field, the .<FIELD> is not included in the notation.

When referring to a specific instance of a register group, specific register instance, or a specific bit in a field, square brackets are used, for example:

<TARGET>:<REGISTER_GROUP>[<register group instance>]:<REGISTER>.<FIELD>

<TARGET>:<REGISTER_GROUP>:<REGISTER>[<register instance>].<FIELD>

<TARGET>:<REGISTER_GROUP>:<REGISTER>.<FIELD>[<bit number>]

4.2 Frame Headers

This section describes the internal header formats used within the device that are visible in frames to and from the CPU, NPI port, and in frames exchanged between devices in multichip configurations.

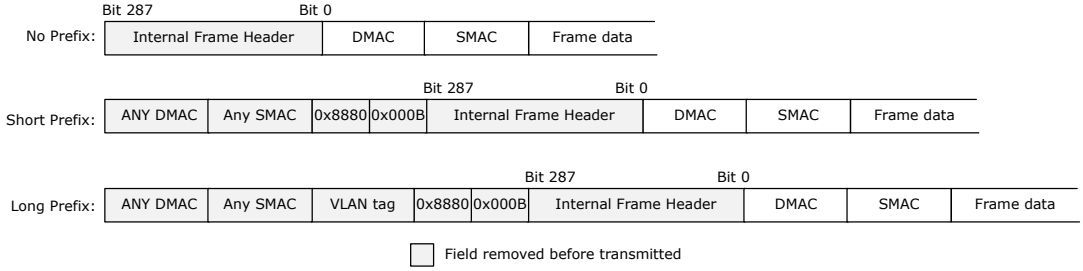
The header formats are internal frame header (IFH) and VStaX header.

- Internal frame header (IFH) IFH is used when extracting frames to CPU and injecting frames from CPU. The IFH can also be inserted into frames transmitted on the NPI port. The IFH includes a VStaX header.
- VStaX header The VStaX header can be used for transmission to and from an NPI port.

4.2.1 Internal Frame Header Placement

The following figure shows internal frame header placement.

Figure 4-2. Frame with Internal Frame Header



Frames are injected without prefix from internal CPU or directly attached CPU.

Frames can be injected with a prefix to accommodate CPU injection from a front port (that may be directly attached), controlled by ASM:CFG:PORT_CFG.INJ_FORMAT_CFG.

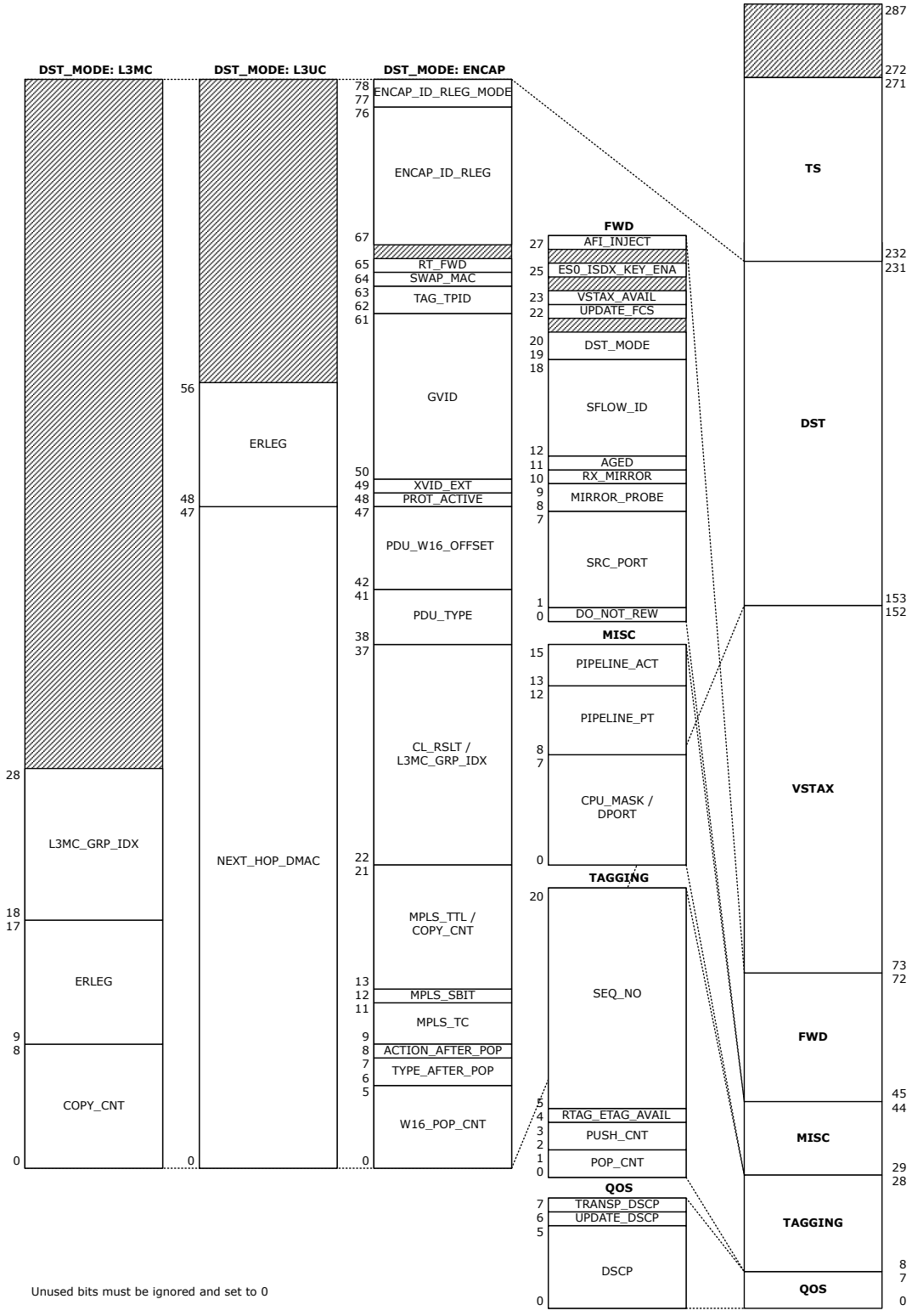
The internal frame header and optional prefix is removed before transmitting the frame on a port. It is only visible to the user when injecting and extracting frames to or from the switch core and optionally when sending/receiving frames using the NPI port.

It is possible to configure a port to transmit frames with internal frame headers using REW:COMMON:PORT_CTRL.KEEP_IFH_SEL. It is also possible to only transmit CPU redirected frames with Internal frame headers using REW:COMMON:GCPU_CFG.GCPU_KEEP_IFH.

4.2.2 Internal Frame Header Layout

The internal frame header is 36 bytes long. The following figure shows the layout.

Figure 4-3. Internal Frame Header



The following table shows the internal frame header fields, including information whether a field is relevant for injection, extraction, or both.

Table 4-1. Internal Frame Header Fields

Field Category	Field Name	Bit	Width	Description
DST when FWD.DST_MODE is ENCAP	ENCAP_ID_RLEG_MODE	231:230	2 bits	Controls value in ENCAP_ID_RLEG. 0: Reserved 1: Contains encapsulation identifier for an IP tunnel, ENCAP_ID 2: Contains ingress router leg, IRLEG 3: Contains egress router leg, ERLEG
	ENCAP_ID_RLEG	229:220	10 bits	Value according to ENCAP_ID_RLEG_MODE.
	RSV	219	1 bit	Reserved field. Must be set to 0.
	RT_FWD	218	1 bit	Signals the frame was routed. Only valid for extraction.
	SWAP_MAC	217	1 bit	Swap MAC addresses.
	TAG_TPID	216:215	2 bits	Tag protocol IDs. 0: Standard TPID as specified in VSTAX.TAG.TAG_TYPE. 1: custom1 stag TPID. 2: custom2 stag TPID. 3: custom3 stag TPID.
	GVID	214:203	12 bits	Generic VLAN identifier.
	XVID_EXT	202	1 bit	Classified VID extension bit. Classified VID = 4096 * XVID_EXT + VSTAX.TAG.VID
	PROT_ACTIVE	201	1 bit	Protection is active.
	PDU_W16_OFFSET	200:195	6 bits	PDU WORD16 (= 2 bytes) offset from W16_POP_CNT to protocol data unit (PDU).
	PDU_TYPE	194:191	4 bits	PDU type used to handle OAM, PTP and SAT. 0: None. 1: OAM_Y1731. 2: OAM_MPLS_TP. 3: OAM_TWAMP. 4: OAM_IP_BFD. 5: PTP. 6: IP4_UDP_PTP. 7: IP6_UDP_PTP. 8: Reserved. 9: SAM_SEQ. 10-15: Reserved.

.....continued

Field Category	Field Name	Bit	Width	Description
	CL_RSLT/ L3MC_GRP_IDX	190:175	16 bits	Classified MATCH_ID combined from VCAP CLM and VCAP IS2. Used if the frame is forwarded to the CPU. For multicast-routed frame: Contains L3_MC.L3_GRP_IDX.
	MPLS_TTL/COPY_CNT	174:166	9 bits	TTL value for possible use in MPLS label. For multicast-routed frame: Contains L3_MC.COPY_CNT.
	MPLS_SBIT	165	1 bit	S-bit of last popped MPLS label for possible use in MPLS label at egress.
	MPLS_TC	164:162	3 bits	TC value for possible use in MPLS label at egress.
	ACTION_AFTER_POP	161	1 bit	0: No action 1: Decrement IPv4 TTL and IPv4 header checksum or IPv6 hop limit.
	TYPE_AFTER_POP	160:159	2 bits	0: ETH - Normal Ethernet header, starting with DMAC. 1: CW. First 4 bits: 4: IPv4 header. 6: IPv6 header. Others: MPLS CW (see RFC 4385) 2: MPLS. MPLS shim header follows.
	W16_POP_CNT	158:153	6 bits	Number of WORD16 (= 2 bytes) to be popped by rewriter, starting from beginning of frame.
DST when FWD.DST_MODE is “L3UC” (only used for extraction)	RSV	231:210	22 bits	Reserved field. Must be set to 0.
	ERLEG	209:201	9 bits	Egress router leg.
	NEXT_HOP_DMAC	200:153	48 bits	Next hop DMAC. Only used for unicast routing.
DST when FWD.DST_MODE is “L3MC” (only used for extraction)	RSV	231:182	50 bits	Reserved field. Must be set to 0.
	L3MC_GRP_IDX	181:171	11 bits	IP multicast group used for L3 multicast copies.
	ERLEG	170:162	9 bits	Egress router leg.
	COPY_CNT	161:153	9 bits	Number of multicast routed copies. Only used for multicast routing.
VSTAX		152:73	80 bits	VStaX. See 4.2.3 VStaX Header.

.....continued

Field Category	Field Name	Bit	Width	Description
FWD	AFI_INJ	72	1 bit	Injected into AFI.
	RSV	71	1 bit	Reserved field. Must be set to 0.
	ES0_ISDX_KEY_ENA	70	1 bit	Controls use of ISDX in ES0 key.
	RSV	69	1 bit	Reserved field. Must be set to 0.
	VSTAX_AVAIL	68	1 bit	Received by ANA with valid VSTAX section. Extract: Frame received by ANA with valid VSTAX section. Inject: Frame to be forwarded based on VSTAX section.
	UPDATE_FCS	67	1 bit	Forces update of FCS. 0: Does not update FCS. FCS is only updated if frame is modified by hardware. 1: Forces unconditional update of FCS.
	RSV	66	1 bit	Reserved field. Must be set to 0.
	DST_MODE	65:64	2 bits	Controls format of IFH.DST. 0: ENCAP 1: L3UC routing 2: L3MC routing Others: Reserved
	SFLOW_ID	63:57	7 bits	SFlow sampling information. 0-64: Tx sampling on port given by SFLOW_ID. 124: For injection: Disable SFlow sampling of frame. 125, 126: Rx sampling on port given by IFH.FWD.SRC_PORT. 127: No SFlow sampling (default value).
	AGED	56	1 bit	Must be set to 0. Set if frame is aged by QSYS. Only valid for extraction.
	RX_MIRROR	55	1 bit	Signals that the frame is Rx mirrored. Only valid for extraction.
	MIRROR_PROBE	54:53	2 bits	Signals mirror probe for mirrored traffic. Only valid for extraction. 0: Not mirrored. 1-3: Mirror probe 0-2.
	SRC_PORT	52:46	7 bits	Physical source port number. May be set by CPU to non-CPU port number to masquerade another port.
	DO_NOT_REW	45	1 bit	Controlled by CPU or ANA_CL:PORT:QOS_CFG.KEEP_ENA and prevents the rewriter from making any changes of frames sent to front ports when set. Only valid for injection.

.....continued

Field Category	Field Name	Bit	Width	Description
MISC	PIPELINE_ACT	44:42	3 bits	Pipeline action specifies if a frame is injected, extracted, or discarded. 0: None 1: INJ 2: INJ_MASQ 3: Reserved 4: XTR 5: XTR_UPMEP 6: LBK_ASM 7: LBK_QS
	PIPELINE_PT	41:37	5 bits	Pipeline point specifies the location where a frame is injected, extracted, or discarded. 0: None 1: ANA_VRAP 2: ANA_PORT_VOE 3: ANA_CL 4: ANA_CLM 5: ANA_IPT_PROT 6: ANA_OU_MIP 7: ANA_OU_SW 8: ANA_OU_PROT 9: ANA_OU_VOE 10: ANA_MID_PROT 11: ANA_IN_VOE 12: ANA_IN_PROT 13: ANA_IN_SW 14: ANA_IN_MIP 15: ANA_VLAN 16: ANA_DONE 17: REW_IN_MIP 18: REW_IN_SW 19: RE_IN_VOE 20: REW_OU_VOE 21: REW_OU_SW 22: REW_OU_MIP 23: REW_SAT 24: REW_PORT_VOE 25: REW_VRAP

.....continued

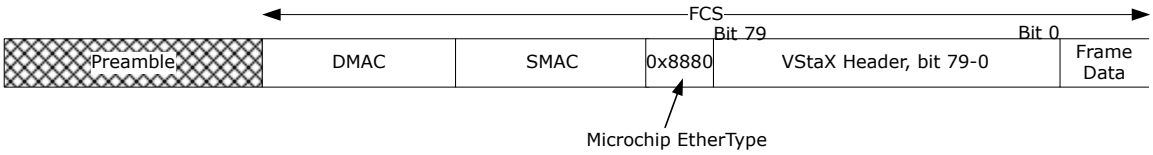
Field Category	Field Name	Bit	Width	Description
	CPU_MASK/DPORT	36:29	8 bits	For extracted frames: CPU extraction queue mask. For injected frames: Destination port number when PIPELINE_ACT == INJ/INJ_MASQ, and PIPELINE_PT > ANA_DONE, and AFI_INJ == 0.
TAGGING	SEQ_NO	28:13	16 bits	IEEE 802.1CB sequence number, either extracted from frame's R-TAG or generated by analyzer.
	RTAG_ETAG_AVAIL	12	1 bit	Set if frame contains either an R-TAG (IEEE 802.1CB) or an E-TAG (IEEE 802.1BR)
	PUSH_CNT	11:10	2 bits	Controlled by CPU or ANA_CL:PORT:VLAN_CTRL_2.VLAN_PUSH_CNT or CLM VLAN_PUSH_CNT_ENA and causes the rewriter to push the signaled number of consecutive VLAN tags. If ENCAP.W16_POP_CNT >0 and TYPE_AFTER_POP = ETH POP_CNT applies to inner Ethernet layer.
	POP_CNT	9:8	2 bits	Controlled by CPU or ANA_CL:PORT:VLAN_CTRL.VLAN_POP_CNT or CLM VLAN_POP_CNT_ENA and causes the rewriter to pop the signaled number of consecutive VLAN tags. If ENCAP.W16_POP_CNT >0 and TYPE_AFTER_POP = ETH POP_CNT applies to inner Ethernet layer.
QOS	TRANSP_DSCP	7	1 bit	Controlled by CPU or ANA_CL:PORT:QOS_CFG. DSCP_KEEP_ENA and prevents the rewriter from remapping DSCP values of frames sent to front ports when set.
	UPDATE_DSCP	6	1 bit	Controlled by CPU, ANA_CL:PORT:QOS_CFG DSCP_REWR_MODE_SEL, CLM DSCP_ENA or ANA_CL:MAP_TBL: SET_CTRL.DSCP_ENA and causes the rewriter to update the frame's DSCP value with IFH.QOS.DSCP for frames sent to front ports when set.
	DSCP	5:0	6 bits	DSCP value.

The IFH is only used to control the rewriter on front ports or send to CPU. It is not transmitted with the frame. For CPU ports, the information in the extraction IFH is for reference purposes only.

4.2.3 VStaX Header

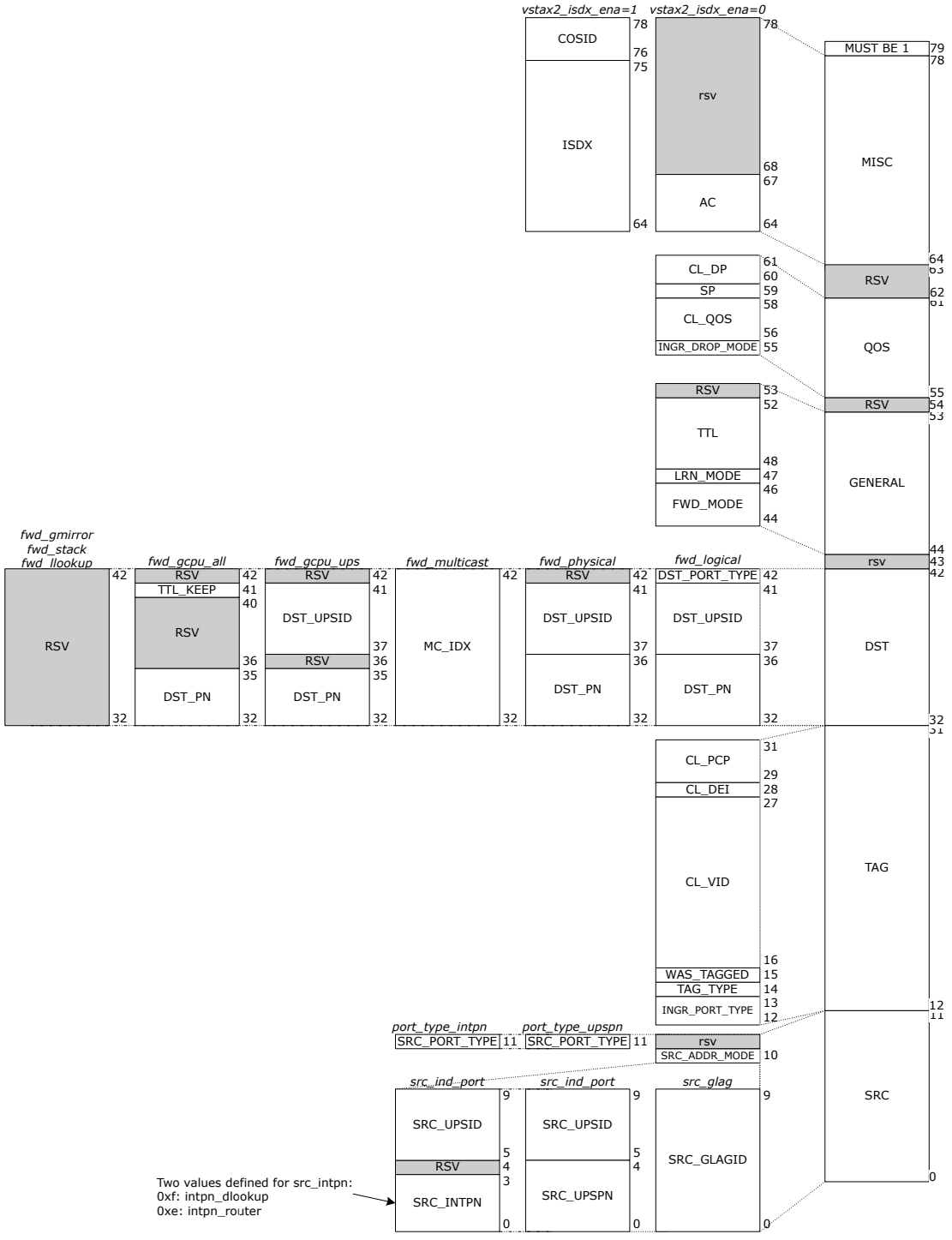
When frames are sent to or from the NPI port, an optional VStaX header is inserted into the frames. The VStaX header consists of a 10-byte payload and is preceded by 2 bytes for the Microchip EtherType (0x8880). That is, a total of 12 bytes are inserted into the frame, as shown in the following figure.

Figure 4-4. Frame With VStaX Header



The layout of the 10-byte VStaX header is shown in the following figure. In VStaX context, each switch in a stack is termed a unit.

Figure 4-5. VStaX Header Layout



The following table describes each field in the VStaX header.

Table 4-2. VStaX Header Fields

Field Category	Field Name	Bit	Width	Description
Reserved	RSV	79	1 bit	Reserved field. Must be set to 1 when injecting frames and must be checked on extraction.
MISC when ANA_AC:PS_COMMON.VSTAX2_MISC_ISDX_ENA=1	COSID	78:76	3 bits	Class of service.
	ISDX	75:64	12 bits	Ingress service index.
MISC when ANA_AC:PS_COMMON.VSTAX2_MISC_ISDX_ENA=0	RSV	78:68	11 bits	Reserved field. Must be set to 0 when injecting frames from CPU and ignored on extraction.
	AC	67:64	4 bits	GLAG aggregation code
Reserved	RSV	63:62	2 bits	Reserved field. Must be set to 0.
QOS	CL_DP	60:61	2 bits	Classified drop precedence level.
	SP	59	1 bit	Super priority. Identifies frames to be forwarded between CPUs of neighboring units, using egress and ingress super priority queues in the assembler and disassembler.
	CL_QOS (IPRIO)	58:56	3 bits	Classified quality of service value (internal priority).
	INGR_DROP_MODE	55	1 bit	Congestion management information. 0: Ingress front port is in flow control mode. 1: Ingress front port is in drop mode.
Reserved	RSV	54	1 bit	Reserved field. Must be set to 0.

.....continued

Field Category	Field Name	Bit	Width	Description
General	RSV	53	1 bit	Reserved field. Must be set to 0.
	TTL	52:48	5 bits	Time to live.
	LRN_MODE	47	1 bit	0: Normal learning. SMAC and VID of the frame is subject to learning. 1: Skip learning. Do not learn SMAC and VID of the frame.
	FWD_MODE	46:44	3 bits	Forward mode. Encoding: 0x0: FWD_LLOOKUP: Forward using local lookup in every unit. 0x1: FWD_LOGICAL: Forward to logical front port at specific UPS. 0x2: FWD_PHYSICAL: Forward to physical front port at specific UPS. 0x3: FWD_MULTICAST: Forward to ports part of multicast group. 0x4: FWD_GMIRROR: Forward to global mirror port. 0x5: FWD_GCPU_UPS: Forward to GCPU of specific UPS. 0x6: FWD_GCPU_ALL: Forward to all GCPUs. 0x7: Reserved.
Reserved	RSV	43	1 bit	Reserved field. Must be set to 0.
VSTAX.DST when VSTAX.FWD_MODE is FWD_LLOOKUP or FWD_GMIRROR	REW_CMD	42:32	11 bits	VCAP IS2 action REW_CMD.
VSTAX.DST when VSTAX.FWD_MODE is FWD_LOGICAL	DST_PORT_TYPE	42	1 bit	Destination port type. Encoding: 0: Front port 1: Internal port
	DST_UPSID	41:37	5 bits	Destination unit port set ID.
	DST_PN	36:32	5 bits	Logical destination port at unit identified by dst_upsid.
VSTAX.DST when VSTAX.FWD_MODE is FWD_PHYSICAL	RSV	42	1 bit	Reserved field. Must be set to 0.
	DST_UPSID	41:37	5 bits	Destination unit port set ID.
	DST_PN	36:32	5 bits	Physical destination port at unit identified by dst_upsid.
VSTAX.DST when VSTAX.FWD_MODE is FWD_MULTICAST	MC_IDX	41:32	11 bits	Forward to ports part of this multicast group index.

.....continued

Field Category	Field Name	Bit	Width	Description
VSTAX.DST when VSTAX.FWD_MODE is FWD_GCPU_UPS	RSV	42	1 bit	Reserved field. Must be set to 0.
	DST_UPSID	41:37	5 bits	Destination unit port set ID.
	RSV	36	1 bit	Reserved field. Must be set to 0.
	DST_PN	35:32	4 bits	CPU destination port at unit identified by dst_upsid.
VSTAX.DST when VSTAX.FWD_MODE is FWD_GCPU_ALL	RSV	42	1 bit	Reserved field. Must be set to 0.
	TTL_KEEP	41	1 bit	Special TTL handling used for neighbor discovery.
	RSV	40:36	5 bits	Reserved field. Must be set to 0.
	DST_PN	35:32	4 bits	CPU destination port at unit identified by dst_upsid.
TAG	CL_PCP	31:29	3 bits	Classified priority code point value.
	CL_DEI	28	1 bit	Classified drop eligible indicator value.
	CL_VID	27:16	12 bits	Classified VID.
	WAS_TAGGED	15	1 bit	If set, frame was VLAN-tagged at reception.
	TAG_TYPE	14	1 bit	Tag type. 0: C-tag (EtherType 0x8100). 1: S-tag (EtherType 0x88A8).
	INGR_PORT_TYPE	13:12	2 bits	Ingress ports type for private VLANs/ Asymmetric VLANs. 00: Promiscuous port. 01: Community port. 10: Isolated port.

.....continued				
Field Category	**Field Name**	**Bit**	**Width**	**Description**
SRC	SRC_ADDR_MODE	10	1 bit	0: src_ind_port: Individual port. Source consists of src_upsid and src_upspn. 1: src_glag: Source consists of src_glag.
	SRC_PORT_TYPE	11	1 bit	Only applicable if src_addr_mode==src_ind_port. Reserved and must be set to 0 if src_addr_mode==src_glag. 0: port_type_upspn. 1: port_type_intpn.
	SRC_UPSID	9:5	5 bits	If src_addr_mode = src_ind_port: ID of stack unit port set, where the frame was initially received.
	SRC_UPSPN	4:0	5 bits	If src_addr_mode = src_ind_port and src_port_type = port_type_upspn: Logical port number of the port (on source ups), where the frame was initially received.
	SRC_INTPN	3:0	4 bits	If src_addr_mode = src_ind_port and src_port_type = port_type_intpn: Internal port number.
	SRC_GLAGID	0	5 bits	If src_addr_mode = src_glag: ID of the GLAG.

4.3 Port Numbering and Mappings

The device has in total 65 logical ports (denoted D0 to D64) and 33 physical SERDES connections (S0 to S32).

The 65th port (D64) is fixed allocated to SERDES0 (S0). The remaining 64 ports can in various multiplexing scenarios be connected to the remaining 32 SERDES using QSGMII, or USGMII or USXGMII extenders. 32 of the ports can have a 1:1 mapping to the 32 SERDES.

Some additional ports (D65 to D69) are internal to the device and do not connect to port modules or SERDES macros. For example, internal ports are used for frame injection and extraction to the CPU queues.

The 65 logical ports are split up into the following blocks.

- 13 x 5G ports (D0-D11, D64)
- 32 x 2G5 ports (D16-D47)
- 12 x 10G ports (D12-D15, D48-D55)
- 8 x 25G ports, limited to maximum speed of 10Gbps (D56-D63)

Each logical port supports different line speeds, and depending on the speeds supported, different port modules (MAC+PCS) are needed. A port supporting 5 Gbps or 10 Gbps as maximum line speed, will have a DEV5G or DEV10G/DEV25G module to support the 5 Gbps or 10 Gbps (incl 5 Gbps) speeds. As well as, it will have a shadow DEV2G5 port module to support the lower speeds (10/100/1000/2500Mbps). When a port needs to operate at lower speed and the shadow DEV2G5 needs to be connected to its corresponding SERDES, this is achieved by programming field PORT_CONF:HW_CFG:DEVx_MODES.DEVx_Dn_MODE = 1, x=5G, 10G, 25G, and n = 0 to 64. The only exception to this is the 10G-DXGMII mode, operating at lower speeds(<=2.5 Gbps), where correct operation is achieved through USXGMII replication.

The 33 SERDES comes in three flavors: SERDES6G (S0-S12) with a maximum speed of 5 Gbps, SERDES10G (S13-S24 with a maximum speed of 10 Gbps, and SERDES25G (S25-S32) with a maximum speed of 10 Gbps.
Note: The 100FX is only supported on S1-S24 for the associated direct port.

The following table lists the logical port mapping to port module numbers (DEVxx), as well as SERDES number to be used for register configuration.

Table 4-3. Port, Port Module, SERDES Mapping for Ports with Direct SERDES Connection.

Port Numbers	Port Modules	SERDES Number	SERDES Macro
D0	DEV2G5_0, DEV5G_0	S1	SERDES6G_1
D1	DEV2G5_1, DEV5G_1	S2	SERDES6G_2
D2	DEV2G5_2, DEV5G_2	S3	SERDES6G_3
D3	DEV2G5_3, DEV5G_3	S4	SERDES6G_4
D4	DEV2G5_4, DEV5G_4	S5	SERDES6G_5
D5	DEV2G5_5, DEV5G_5	S6	SERDES6G_6
D6	DEV2G5_6, DEV5G_6	S7	SERDES6G_7
D7	DEV2G5_7, DEV5G_7	S8	SERDES6G_8
D8	DEV2G5_8, DEV5G_8	S9	SERDES6G_9
D9	DEV2G5_9, DEV5G_9	S10	SERDES6G_10
D10	DEV2G5_10, DEV5G_10	S11	SERDES6G_11
D11	DEV2G5_11, DEV5G_11	S12	SERDES6G_12
D12	DEV2G5_12, DEV10G_0	S13	SERDES10G_0
D13	DEV2G5_13, DEV10G_1	S14	SERDES10G_1
D14	DEV2G5_14, DEV10G_2	S15	SERDES10G_2
D15	DEV2G5_15, DEV10G_3	S16	SERDES10G_3
D16-D47	DEV2G5_16 – DEV2G5_47	-	No direct SERDES
D48	DEV2G5_48, DEV10G_4	S17	SERDES10G_4
D49	DEV2G5_49, DEV10G_5	S18	SERDES10G_5
D50	DEV2G5_50, DEV10G_6	S19	SERDES10G_6
D51	DEV2G5_51, DEV10G_7	S20	SERDES10G_7
D52	DEV2G5_52, DEV10G_8	S21	SERDES10G_8
D53	DEV2G5_53, DEV10G_9	S22	SERDES10G_9
D54	DEV2G5_54, DEV10G_10	S23	SERDES10G_10
D55	DEV2G5_55, DEV10G_11	S24	SERDES10G_11
D56	DEV2G5_56, DEV25G_0	S25	SERDES25G_0
D57	DEV2G5_57, DEV25G_1	S26	SERDES25G_1

.....continued			
Port Numbers	**Port Modules**	**SERDES Number**	**SERDES Macro**
D58	DEV2G5_58, DEV25G_2	S27	SERDES25G_2
D59	DEV2G5_59, DEV25G_3	S28	SERDES25G_3
D60	DEV2G5_60, DEV25G_4	S29	SERDES25G_4
D61	DEV2G5_61, DEV25G_5	S30	SERDES25G_5
D62	DEV2G5_62, DEV25G_6	S31	SERDES25G_6
D63	DEV2G5_63, DEV25G_7	S32	SERDES25G_7
D64	DEV2G5_64, DEV5G_12	S0	SERDES6G_0
D65(CPU0)	No Port Module	No SERDES	No SERDES
D66(CPU1)	No Port Module	No SERDES	No SERDES
D67(VD0)	No Port Module	No SERDES	No SERDES
D68(VD1)	No Port Module	No SERDES	No SERDES
D69(VD2)	No Port Module	No SERDES	No SERDES

Ports 16 through 47 do not connect to SERDES macros in the default configuration. They are only connected when QSGMII, USGMII or USXGMII is enabled.

Ports 65 through 69 are internal ports and are defined as follows:

- Port 65 and port 66: CPU port 0 and 1 (CPU0, CPU1) are used for injection and extraction of frames.
- Port 67: Virtual device 0 (VD0) is used for IP multicast routing.
- Port 68: Virtual device 1 (VD1) is used for AFI frame injections.
- Port 69: Virtual device 2 (VD2) is used for IP-in-IPv4 tunneling.

The internal ports do not have associated port modules and interface macros.

In the following sections, any reference to DEV5G, DEV10G, or DEV25G covering lower speeds as well includes reference to its shadow DEV2G5 port module.

4.3.1 SERDES Macro to I/O Pin Mapping

The following table shows the fixed mapping from SERDES macros to the device I/O pins.

Table 4-4. SERDES Macro to I/O Pin Mapping

SERDES No	SERDES Macros	I/O Pin Names
S0–S12	SERDES6G_0 to SERDES6G_12	S0_RXN, S0_RXP, S0_TXN, S0_TXP - S12_RXN, S12_RXP, S12_TXN, S12_TXP
S13–S24	SERDES10G_0 to SERDES10G_11	S13_RXN, S13_RXP, S13_TXN, S13_TXP - S24_RXN, S24_RXP, S24_TXN, S24_TXP
S25–S32	SERDES25G_0 to SERDES25G_7	S25_RXN, S25_RXP, S25_TXN, S25_TXP - S32_RXN, S32_RXP, S32_TXN, S32_TXP

4.3.2 Supported Port Interfaces

The device supports a range of physical port interfaces. The following table lists the interfaces supported by the SERDES macros, as well as standards, data rates, connectors, medium, and coding for each port interface.

Table 4-5. Supported Port Interface Standards.

Port Interface	Specification	Port Speed	Data Rate	Connector	Medium	Coding
100BASE-FX	IEEE 802.3, Clause 24	100M	125 Mbps	SFP	PCB	4B5B
SGMII	Cisco	1G	1.25 Gbps		PCB	8B10B
SFP	SFP-MSA, SFF INF-8074i	1G	1.25 Gbps	SFP	Optical	8B10B
1000BASE-KX	IEEE802.3, Clause 70	1G	1.25 Gbps		PCB, Backplane	8B10B
2.5GBASE-KX	IEEE802.3, Clause 128	2.5G	3.125 Gbps		PCB, Backplane	8B10B
QSGMII	Cisco	4 x 1G	5 Gbps		PCB, Backplane	8B10B
USGMII	Cisco	8 x 1G	10 Gbps		PCB, Backplane	8B10B
USXGMII (10G-QXGMII)	Cisco	4 x 2.5G	10.3125 Gbps		PCB, Backplane	64B66B
USXGMII (10G-DXGMII)	Cisco	2 x 5G	10.3125 Gbps		PCB, Backplane	64B66B
USXGMII (5G-DXGMII)	Cisco	2 x 2.5G	10.3125 Gbps		PCB, Backplane	64B66B
5GBASE-KR	IEEE 802.3, Clause 130	5G	5.156 Gbps		PCB, Backplane	64B66B
10GBASE-KR	IEEE 802.3, Clause 172	10G	10.3125 Gbps		PCB, Backplane	64B66B
SFP+(SFI)	SFF-8431: Direct Attached Cu IEEE 802.3, Clause 52	10G	10.3125 Gbps	SFP+	PCB, Cable, Optical	64B66B

4.3.3 QSGMII

Up to 12 of the SERDES macros can be configured QSGMII. This is enabled in PORT_CONF::HW_CFG.QSGMII_ENA per macro. When QSGMII is enabled for a SERDES macro, four port modules are associated with the SERDES macro to form a QSGMII port with four 1G ports.

To configure QSGMII[n] set PORT_CONF:HW_CFG:QSGMII_ENA[n] = 1, where n = 0 to 11, selects the corresponding QSGMII extender.

The following table lists which ports, port modules, and SERDES macros are associated with each QSGMII instance for the device.

Table 4-6. Related Ports, Port modules, and SERDES Macros with each QSGMII Instance

QSGMII Number	Port Numbers	SERDES Number	SERDES Macro
0	D0–D3	S13	SERDES10G_0
1	D4–D7	S14	SERDES10G_1
2	D8–D11	S15	SERDES10G_2
3	D12–D15	S16	SERDES10G_3
4	D16–D19	S17	SERDES10G_4
5	D20–D23	S18	SERDES10G_5
6	D24–D27	S19	SERDES10G_6
7	D28–D31	S20	SERDES10G_7
8	D32–D35	S21	SERDES10G_8
9	D36–D39	S22	SERDES10G_9
10	D40–D43	S23	SERDES10G_10
11	D44–D47	S24	SERDES10G_11

Note that when a SERDES block is enabled for QSGMII, the default port connections are removed. For example, if S13 (SERDES10G_0) is enabled for QSGMII, then port numbers D0, D1, D2 and D3 connect to S13, and the default connection from D12 is not present.

4.3.4 USGMII

Up to 6 of the SERDES10G macros can be configured for USGMII. This is enabled in PORT_CONF::HW_CFG.USGMII_ENA per macro. When USGMII is enabled for a SERDES macro, eight port modules are associated with the SERDES10G macro to form a USGMII port with eight 1G ports.

To configure USGMII[n] set PORT_CONF:HW_CFG:USGMII_ENA[n] = 1, where n = 0 to 5, selects the corresponding USGMII extender.

The following table lists which ports, port modules, and SERDES10G macros are associated with each USGMII instance for the device.

Table 4-7. Related Ports, Port modules, and SERDES10G Macros with each USGMII Instance

USGMII Number	Port Numbers	SERDES Number	SERDES Macro
0	D0–D7	S17	SERDES10G_4
1	D8–D15	S18	SERDES10G_5
2	D16–D23	S19	SERDES10G_6
3	D24–D31	S20	SERDES10G_7
4	D32–D39	S21	SERDES10G_8
5	D40–D47	S22	SERDES10G_9

Note that when a SERDES macro is enabled for USGMII, the default port connection is not available. For example, if S17 (SERDES10G_4) is enabled for USGMII, then port numbers D0, D1, D2, D3, D4, D5, D6, and D7 connect to S17 and the default connection from D48 is not present. Also configuring a SERDES macro for USGMII takes priority over QSGMII mode.

4.3.5 USXGMII

All SerDes macros can be configured USXGMII. Supported modes with involved ports are described in the following chapters. Note that configuring a SERDES block for USXGMII takes priority over QSGMII and USGMII mode.

4.3.5.1 10G-QXGMII

Up to 16 of the SERDES macros can be configured 10G-QXGMII. This is enabled in PORT_CONF:HW_CFG:USXGMII_ENA and configured in PORT_CONF:USXGMII_CFG[16-31]:USXGMII_CFG per macro. When 10G-QXGMII is enabled for a SERDES macro running at 10G speed, four port modules are associated with the SERDES macro to form a 10G-QXGMII port with four 2.5G ports.

To configure USXGMII extender “n” in 10G-QXGMII mode, set PORT_CONF:HW_CFG:USXGMII_ENA[n+16], PORT_CONF:USXGMII_CFG[n+16]:USXGMII_CFG.NUM_PORTS=2, n = 0 – 15.

The following table lists which ports, port modules, and SERDES10G macros are associated with each 10G-QXGMII instance for the device.

Table 4-8. Related Ports, Port modules, and SERDES10G Macros with each 10G-QXGMII Instance

USXGMII Number	Port Numbers (P0, P1, P2 and P3)	SERDES Number	SERDES Macro
16	D48, D0, D16, D32	S17	SERDES10G_4
17	D49, D1, D17, D33	S18	SERDES10G_5
18	D50, D2, D18, D34	S19	SERDES10G_6
19	D51, D3, D19, D35	S20	SERDES10G_7
20	D52, D4, D20, D36	S21	SERDES10G_8
21	D53, D5, D21, D37	S22	SERDES10G_9
22	D54, D6, D22, D38	S23	SERDES10G_10
23	D55, D7, D23, D39	S24	SERDES10G_11
24	D56, D8, D24, D40	S25	SERDES25G_0
25	D57, D9, D25, D41	S26	SERDES25G_1
26	D58, D10, D26, D42	S27	SERDES25G_2
27	D59, D11, D27, D43	S28	SERDES25G_3
28	D60, D12, D28, D44	S29	SERDES25G_4
29	D61, D13, D29, D45	S30	SERDES25G_5
30	D62, D14, D30, D46	S31	SERDES25G_6
31	D63, D15, D31, D47	S32	SERDES25G_7

Note that when a SERDES block is enabled for 10G-QXGMII, the default port connection is not available. For example, if S17 (SERDES10G_4) is enabled for 10G-QXGMII, then port number D48, D0, D16, D32 are connected to S17, and the default direct connection from D48 to S17 is not present. 10G-QXGMII mode is configured by setting PORT_CONF:USXGMII_CFG[*n*]:USXGMII_CFG.NUM_PORTS to 2 (quad port mode).

4.3.5.2 10G-DXGMII

Up to 16 of the SERDES macros can be configured 10G-DXGMII. This is enabled in PORT_CONF:HW_CFG:USXGMII_ENA and configured in PORT_CONF:USXGMII_CFG[16-31]:USXGMII_CFG per macro. When 10G-DXGMII is enabled for a SERDES macro running at 10G speed, two port modules are associated with the SERDES macro to form a 10G-DXGMII port with two 5G ports.

To configure USXGMII extender “n” in 10G-DXGMII mode, set : PORT_CONF:HW_CFG:USXGMII_ENA[n+16], PORT_CONF:USXGMII_CFG[n+16]:USXGMII_CFG.NUM_PORTS=1, where n = 0 – 15.

The following table lists which ports, port modules, and SERDES macros are associated with each 10G-DXGMII instance for the device.

Table 4-9. Related Ports, Port modules, and SERDES Macros with each 10G-DXGMII Instance

USXGMII Number	Port Numbers (P0 & P1)	SERDES Number	SERDES Macro
16	D48, D0	S17	SERDES10G_4
17	D49, D1	S18	SERDES10G_5
18	D50, D2	S19	SERDES10G_6
19	D51, D3	S20	SERDES10G_7
20	D52, D4	S21	SERDES10G_8
21	D53, D5	S22	SERDES10G_9
22	D54, D6	S23	SERDES10G_10
23	D55, D7	S24	SERDES10G_11
24	D56, D8	S25	SERDES25G_0
25	D57, D9	S26	SERDES25G_1
26	D58, D10	S27	SERDES25G_2
27	D59, D11	S28	SERDES25G_3
28	D60, D12	S29	SERDES25G_4
29	D61, D13	S30	SERDES25G_5
30	D62, D14	S31	SERDES25G_6
31	D63, D15	S32	SERDES25G_7

Note that when a SERDES macro is enabled for 10G-DXGMII, the default port connection is not available. For example, if S13 (SERDES10G_4) is enabled for 10G-DXSGMII, then port numbers D48 and D0 connect to S13, and the default direct connection from D48 to S13 is not present. 10G-DXGMII mode is configured by setting PORT_CONF:USXGMII_CFG[*n*]:USXGMII_CFG.NUM_PORTS to 1 (dual port mode).

4.3.5.3 5G-DXGMII

Up to 32 of the SERDES macros can be configured 5G-DXGMII. This is enabled in PORT_CONF:HW_CFG:USXGMII_ENA and configured in PORT_CONF:USXGMII_CFG[0-31]:USXGMII_CFG in per macro. When 5G-DXGMII is enabled for a SERDES macro, two port modules are associated with the SERDES macro to form a 5G-DXGMII port with two 2.5G ports.

To configure USXGMII extender “n” in 10G-DXGMII mode, set : Fn: PORT_CONF:HW_CFG:USXGMII_ENA[n], PORT_CONF:USXGMII_CFG[n]:USXGMII_CFG.NUM_PORTS=1, n = 0 – 31. Additionally for USXGMII extenders 16 to 31 PORT_CONF:USXGMII_CFG[n]:USXGMII_CFG. FLIP_PORT_12 must be set to 1.

The following table lists which ports, port modules, and SERDES macros are associated with each 5G-DXGMII instance for the device.

Table 4-10. Related Ports, Port modules, and SERDES Macros with each 5G-DXGMII Instance

USXGMII Number	Port Numbers (P0 & P1)	SERDES Number	SERDES Macro
0	D0, D32	S1	SERDES6G_1
1	D1, D33	S2	SERDES6G_2
2	D2, D34	S3	SERDES6G_3
3	D3, D35	S4	SERDES6G_4
4	D4, D36	S5	SERDES6G_5
5	D5, D37	S6	SERDES6G_6
6	D6, D38	S7	SERDES6G_7
7	D7, D39	S8	SERDES6G_8
8	D8, D40	S9	SERDES6G_9
9	D9, D41	S10	SERDES6G_10
10	D10, D42	S11	SERDES6G_11
11	D11, D43	S12	SERDES6G_12
12	D12, D44	S13	SERDES10G_0
13	D13, D45	S14	SERDES10G_1
14	D14, D46	S15	SERDES10G_2
15	D15, D47	S16	SERDES10G_3
16	D48, D16	S17	SERDES10G_4
17	D49, D17	S18	SERDES10G_5
18	D50, D18	S19	SERDES10G_6
19	D51, D19	S20	SERDES10G_7
20	D52, D20	S21	SERDES10G_8
21	D53, D21	S22	SERDES10G_9
22	D54, D22	S23	SERDES10G_10
23	D55, D23	S24	SERDES10G_11
24	D56, D24	S25	SERDES25G_0
25	D57, D25	S26	SERDES25G_1
26	D58, D26	S27	SERDES25G_2
27	D59, D27	S28	SERDES25G_3
28	D60, D28	S29	SERDES25G_4

.....continued			
USXGMII Number	**Port Numbers (P0 & P1)**	**SERDES Number**	**SERDES Macro**
29	D61, D29	S30	SERDES25G_5
30	D62, D30	S31	SERDES25G_6
31	D63, D31	S32	SERDES25G_7

Note that when a SERDES macro is enabled for 5G-DXGMII, the default port connection is not available. For example, if S1 is enabled for 5G-DXSGMII, then port numbers D0 and D32 connect to S1, and the default direct connection from D0 to S1 is not present. 5G-DXGMII mode is configured by setting PORT_CONF:USXGMII_CFG[*n*]:USXGMII_CFG.NUM_PORTS to 1 (dual port mode).

4.3.6 Port Multiplexing Examples

The SparX-5 multiplexing ability is significant and provides a very large number of possible permutations.

This section will show and explain a number of configuration/port multiplexing examples according to the multiplexing tables.

In following tables shows a number of SparX-5 multiplexing examples. Top rows show SERDES number and type. The following rows show the multiplexing examples. The following multiplexing notation is used.

- Q0–Q11 are QSGMII number 0–11
- X0–X5 are USGMII number 0–5
- R0–R8 are 10G-USXGMII number 0–8
- U0–8 are 10G-DXGMII number 0–8
- F0–17 are 5G-USXGMII number 0–17

The following tables show how the port modules (D0–D64) map to SERDES in various port configurations.

Table 4-11. Mapping Between Port Modules and SERDES for Different Port Configuration Examples—0 to 16

Example	SerDes Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	**SerDes Type**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**6G**	**10G**	**10G**	**10G**	**10G**
		NPI																
1	48x1G+4x10G (QSGMII)	D64	-	-	-	-	-	-	-	-	-	-	-	-	Q0	Q1	Q2	Q3
2	48x1G+4x10G	D64	D0	D1	D2	D3	-	-	-	-	-	-	-	-	-	Q1	Q2	Q3
3	48x1G+4x10G	D64	D0	D1	D2	D3	D4	D5	D6	D7	-	-	-	-	-	-	Q2	Q3
4	48x1G+4x10G	D64	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	-	-	-	Q3
5	48x1G+4x10G	D64	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
6	24x1G+6x10G (24 QSGMII)	D64	-	-	-	-	-	-	-	-	-	-	-	-	Q0	Q1	Q2	Q3
7	9x10G	D64	-	-	-	-	-	-	-	-	-	-	-	-	D12	D13	D14	D15
8	18x5G	D64	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
9	36x2.5G (10USX)	D64	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
10	18x5G (10USX)	D64	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	48x1G (USGMII) + 4x10G	D64	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
12	36x2.5G (5USX)	D64	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15

Table 4-12. Mapping Between Port Modules and SERDES for Different Port Configuration Examples—17 to 32

Example	SerDes Number	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Number	SerDes Type	10G	10G	10G	10G	10G	10G	10G	10G	25G	25G	25G	25G	25G	25G	25G	25G
1	48x1G+4x 10G (QSGMII)	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	D56	D57	D58	D59	-	-	-	-
2	48x1G+4x 10G	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	D56	D57	D58	D59	-	-	-	-
3	48x1G+4x 10G	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	D56	D57	D58	D59	-	-	-	-
4	48x1G+4x 10G	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	D56	D57	D58	D59	-	-	-	-
5	48x1G+4x 10G	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	D56	D57	D58	D59	-	-	-	-
6	24x1G+6x 10G (24 QSGMII)	Q4	Q5	-	-	D52	D53	D54	D55	D56	D57	-	-	-	-	-	-
7	9x10G	D48	D49	D50	D51	D52	-	-	-	-	-	-	-	-	-	-	-
8	18x5G	D48	D49	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9	36x2.5G (10USX)	R0	R1	R2	R3	R4	R5	R6	R7	R8	-	-	-	-	-	-	-
10	18x5G (10USX)	U0	U1	U2	U3	U4	U5	U6	U7	U8	-	-	-	-	-	-	-
11	48x1G (USGMII) + 4x10G	X0	X1	X2	X3	X4	X5	D54	D55	D56	D57	-	-	-	-	-	-
12	36x2.5G (5USX)	F16	F17	-	-	-	-	-	-	-	-	-	-	-	-	-	-

The different examples from Figure xx will be explained in the following. In all examples the NPI port D64 is directly connected to its own SERDES S0 and will not be discussed further.

Example 1:

48 x 1G + 4 x 10G port configuration, where all 48 x 1G ports are established using 12 x QSGMII (Q0–Q11). In this configuration SERDES S13 – S28 are used.

Example 2:

48 x 1G + 4 x 10G port configuration. The first 4 x 1G port modules (D0–D3) are using direct SERDES connections (S1–S4) whereas the remaining 44 x 1G are using 11 x QSGMII (Q1–Q11) which are connected to SERDES S14–S28.

Example 3:

48 x 1G + 4 x 10G port configuration. The first 8 x 1G port modules (D0–D7) are using direct SERDES connections (S1–S4) whereas the remaining 40 x 1G are using 10 x QSGMII (Q2–Q11) which are connected to SERDES S15–S28.

Example 4:

48 x 1G + 4 x 10G port configuration. The first 12 x 1G port modules (D0–D11) are using direct SERDES connections (S1–S12) whereas the remaining 36 x 1G are using 9 x QSGMII (Q3–Q11) which are connected to SERDES S16–S28.

Example 5:

48 x 1G + 4 x 10G port configuration. The first 16 x 1G port modules (D0–D15) are using direct SERDES connections (S1–S16) whereas the remaining 32 x 1G are using 8 x QSGMII (Q4–Q11) which are connected to SERDES S17–S28. As can be seen from the SERDES usage it is not possible in a 48 x 1G + 4 x 10G configuration to have more 1G ports connected directly to SERDES's.

Example 6:

24 x 1G + 6 x 10G port configuration. In this example (like Example 1) all 1G ports are established using 6 x QSGMII (Q0–Q5), the last 6 capable 10G ports are used for 10 Gbps. In this configuration SERDES S13–S18, S21–S26 are used.

Example 7:

9 x 10G port configuration. No multiplexing with all native ports capable to run 10 Gbps (D12–D15, D48–D52) connecting directly to their respective SERDES S13–S21.

Example 8:

18 x 5G port configuration. No multiplexing is used with all native ports connect directly to their respective SERDES S1–S18.

Example 9:

36 x 2.5G port configuration. All ports are connected to SERDES S17–S25 through 10G-USXGMII (R0–R8).

Example 10:

18 x 5G port configuration. All ports are connected to SERDES S17–S25 through 10G-DXGMII (U0–U8).

Example 11:

48 x 1G + 4 x 10G port configuration, where all 48 x 1G ports are established using 6 x USGMII (X0–X5). In this configuration SERDES S17 – S26 are used.

Example 12:

36 x 2.5G port configuration, where all 36 ports are established using 18 x 5G-USGMII (F0–F17). In this configuration SERDES S1 – S18 are used.

The total switching bandwidth of the device depends on the system clock frequency. The following table shows how the port modules are allocated to 7 groups of ports where the total bandwidth within a group must not be exceeded. The total switching bandwidth within a group depends on the system clock frequency.

Table 4-13. Overview of Port Modules in the Seven Allocated Port Groups

Port Group	Port Modules												
1	D57	D12	D0	D1	D2	D16	D17	D18	D19	D20	D21	D22	D23
2	D58	D13	D3	D4	D5	D24	D25	D26	D27	D28	D29	D30	D31
3	D59	D14	D6	D7	D8	D32	D33	D34	D35	D36	D37	D38	D39
4	D60	D15	D9	D10	D11	D40	D41	D42	D43	D44	D45	D46	D47
5	D61	D48	D49	D50									
6	D62	D51	D52	D53									
7	D56	D63	D54	D55									

The following table shows the maximum bandwidth within a pool of port modules at the different operation frequencies.

Table 4-14. Port Group Bandwidth in Gbps for Different System Clock Frequencies

System Clock Frequency (MHz)	Max Bandwidth (Gbps)
500	60
250	30

4.4 SERDES

All the Ethernet SERDES interfaces share the same basic functionality, but support different operating modes and line rates.

The following list lists the SERDES features.

- RX Adaptive Decision Feedback Equalizer (DFE)
- Programmable continuous time linear equalizer (CTLE)
- Rx variable gain control
- Rx built-in fault detector (loss-of-lock/loss-of-signal)
- Adjustable tx de-emphasis (FFE)
- Tx output amplitude control
- Supports rx eye monitor
- Multiple loopback modes
- Prbs generator and checker
- Polarity inversion control

4.4.1 Supported Line Rates

The following sections describe various SERDES supported line rates.

4.4.1.1 SERDES6G

The SERDES6G is a high-speed SERDES interface, which can operate at the following data rates.

- 100 Mbps (100BASE-FX)
- 1.25 Gbps (SGMII/1000BASE-X/1000BASE-KX)
- 3.125 Gbps (2.5GBASE-X/2.5GBASE-KX)
- 5.15625 Gbps (5GBASE-KR/5G-USXGMII)

4.4.1.2 SERDES10G

The SERDES10G is a high-speed SERDES interface, which can operate at the following data rates.

- 100 Mbps (100BASE-FX)
- 1.25 Gbps (SGMII/1000BASE-X/1000BASE-KX)
- 3.125 Gbps (2.5GBASE-X/2.5GBASE-KX)
- 5 Gbps (QSGMII/USGMII)
- 5.15625 Gbps (5GBASE-KR/5G-USXGMII)
- 10 Gbps (10G-USGMII)
- 10.3125 Gbps (10GBASE-R/10GBASE-KR/USXGMII)

4.4.1.3 SERDES25G

The SERDES25G is a high-speed SERDES interface, which can operate at the following data rates.

- 1.25 Gbps (SGMII/1000BASE-X/1000BASE-KX)
- 3.125 Gbps (2.5GBASE-X/2.5GBASE-KX)
- 5 Gbps (QSGMII/USGMII)
- 5.15625 Gbps (5GBASE-KR/5G-USXGMII)
- 10 Gbps (10G-USGMII)

- 10.3125 Gbps (10GBASE-R/10GBASE-KR/USXGMII)

4.4.2 RX and TX Equalization

The SERDES contains advanced equalization features. RX equalization features a 5-tap DFE and CTLE. TX equalization features a 3-tap FIR.

4.4.2.1 RX Equalization

The differential receiver is comprised of a variable gain amplifier (VGA) followed by a continuous-time linear equalizer (CTLE). The VGA amplifies/attenuates incoming signals while maintaining the signal linearity. The CTLE is a programmable high-pass filter that compensates for high frequency loss in the electrical channels. The settings for VGA and CTLE are set through programmable registers. Additionally, a 5-tap decision feedback equalizer (DFE) is incorporated to compensate channels with the significant post-cursor ISI. The DFE can be set to adaptive mode or manual mode. These equalizers allow the receiver to recover data transmitted over lossy backplane channels.

4.4.2.2 TX Equalization

The transmitter includes a programmable equalization feature to compensate for the frequency-dependent loss in the channel. This is accomplished by a 3-tap finite impulse response (FIR).

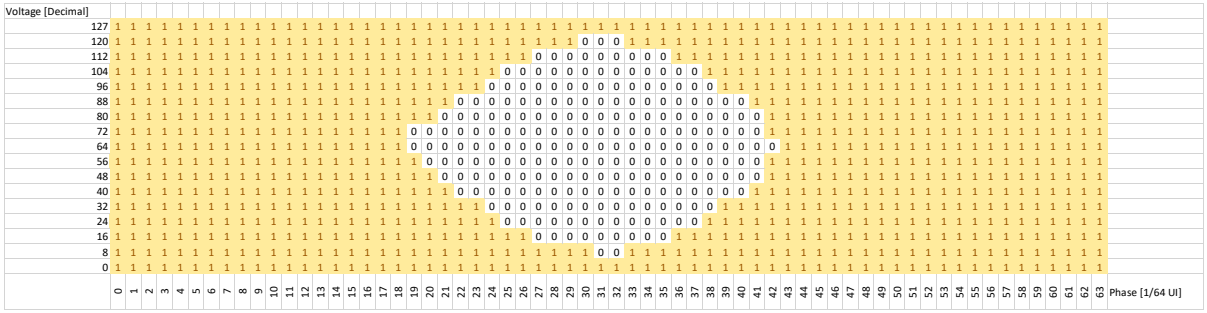
4.4.3 RX Eye Monitor

The receive side includes an Rx Eye monitor that allows the user to get a representation of the receive eye after the RX equalization. This is especially useful when debugging and qualifying links with a speed of 10 Gbps and above where the receive equalizer is required to ensure a link with low bit error rate.

The eye monitor can run in two modes with different resolution: Normal mode and Fast mode.

In Normal mode, the eye scan receiver reference voltage is set to 0 and the sample point is swept across the phase in UI. The result is read back, the reference voltage is increased, and the sweep is repeated. This produces a pass/fail eye with high resolution. For repeatable results, a fixed test patterns like PRBS should be used.

Figure 4-6. Normal Eye Scan Plot



The SERDES also supports a Fast mode where the eye hight can be read back at the center phase. This allows a much faster readback of the general eye closure, but does not provide the same detail level as the normal mode.

Note:
The RX equalizers are frozen during the eye monitor cycle, so it is not recommended to keep the eye monitor function enabled during normal operation.

4.4.4 Loopback Test Mode

Loopback master and slave modes are supported by the SERDES. The description and signal flow diagram of these loopback modes are described in the following sections.

4.4.4.1 Loopback Master Mode

There is only one way to do loopback master mode.

- LM1: this path is a loopback inside the serial of the SERDES with the loopback path from TX to RX.

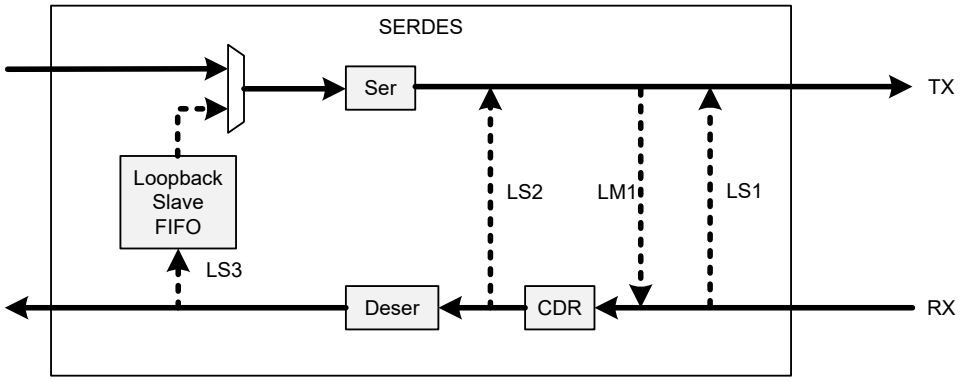
4.4.4.2 Loopback Slave Mode

There are three ways to do loopback slave mode.

- LS1: This path is loopback inside the serial part of the SERDES from RX to TX and the loop-back path is before CDR.

- LS2: This path is also loopback inside the serial part of the SERDES from RX to TX and the loopback path is after CDR.
- LS3: This path is loopback on the parallel side of the SERDES from RX to TX and the loopback path is through a loopback slave FIFO.

Figure 4-7. Serdes Loopback Options



4.4.5 PRBS Function

The SERDES blocks have integrated PRBS generator and checker, which allows fast channel testing with standard compliant test patterns. The PRBS generator/checker supports PRBS-7, 9, 11, 15, 23, and 31. It also supports user-defined patterns.

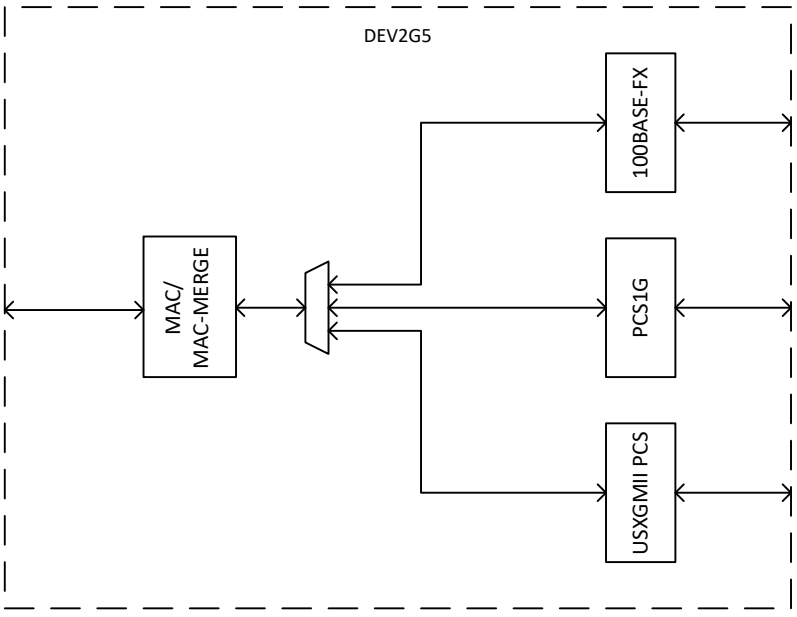
4.4.6 Polarity Inversion

The SERDES supports inversion of the TX and RX differential pairs to allow the PCB signal routing to be op-timized to a given PCB layout.

4.5 DEV2G5 Port Module

DEV2G5 port module supports data speed of 2.5 Gbps and lower speeds, that is, 2.5 Gbps, 1 Gpbs, 100 Mbps, and 10 Mbps. DEV2G5 port module consists of MAC, which also handles MAC Merge sub-layer function, along with PCS (SGMII, 1000BASE-X PCS), 100BASE-FX PCS, and USXGMII PCS.

Figure 4-8. DEV2G5 Port Module



Each of the DEV5G, DEV10G, and DEV25G Port modules have an associated DEV2G5 port module as a shadow device to handle speeds less than or equal to 2.5G.

4.5.1 MAC

This section describes the high level functions and the configuration options of the Media Access Controller (MAC) that is used in the DEV2G5 port modules.

DEV2G5 MAC supports the following speeds and duplex modes depending on the associated mode.

- SERDES: 10/100/1000/2500 Mbps in full-duplex mode and 10/100 Mbps in half-duplex mode.
- QSGMII MODE: 10/100/1000/2500 Mbps in full-duplex mode and 10/100 Mbps in half-duplex mode.
- USGMII MODE: 10/100/1000/2500 Mbps in full-duplex mode and 10/100 Mbps in half-duplex mode.
- USXGMII MODE: 10/100/1000/2500 Mbps in full-duplex mode.

The following table lists the registers associated with configuring the MAC.

Table 4-15. DEV2G5 MAC Configuration Registers

Register	Description	Replication
MAC_ENA_CFG	Enabling of Rx and Tx data paths	Per port module
MAC_MODE_CFG	Port mode configuration	Per port module
MAC_MAXLEN_CFG	Maximum length configuration	Per port module
MAC_TAGS_CFG	VLAN/Service tag configuration	Per port module
MAC_TAGS_CFG2	VLAN / Service tag configuration 2	Per port module
MAC_ADV_CHK_CFG	Configuration to enable dropping of Type/Length error frames	Per port module
MAC_IFG_CFG	Inter frame gap configuration	Per port module
MAC_HDX_CFG	Half-duplex configuration	Per port module
MAC_STICKY	Sticky bit recordings	Per port module

4.5.1.1 Clock and Reset

There are a number of resets in the port module. All of the resets can be set and cleared simultaneously. By default, all blocks are in the reset state. With reference to register DEV_RST_CTRL, the resets are as follows.

Table 4-16. DEV2G5 Resets

Register.Field	Description
DEV_RST_CTRL.MAC_RX_RST	Reset MAC receiver
DEV_RST_CTRL.MAC_TX_RST	Reset MAC transmitter
DEV_RST_CTRL.PCS_RX_RST	Reset PCS receiver
DEV_RST_CTRL.PCS_TX_RST	Reset PCS transmitter
DEV_RST_CTRL.USX_PCS_RX_RST	Reset USXGMI PCS receiver
DEV_RST_CTRL.USX_PCS_TX_RST	Reset USXGMI PCS transmitter

When changing the MAC configuration, the port must go through a reset cycle. This is done by writing register DEV_RST_CTRL twice. On the first write, the reset bits are set. On the second write, the reset bits are cleared. The non-reset field DEV_RST_CTRL.SPEED_SEL must keep its new value for both writes.

4.5.1.2 Interrupts

There are 8 interrupt sources defined for DEV2G5 port module, which can be found in the following table. General interrupt handling is more detailed described in 5.4.2 VCore CPU Interrupt Controller.

Table 4-17. DEV2G5 Interrupt Sources

Register.Field	Description
DEV2G5_INTR.FEF_FOUND_INTR_STICKY	Far-end-fault indication found
DEV2G5_INTR.TX_LPI_INTR_STICKY	Low Power Idle Transmit interrupt
DEV2G5_INTR.RX_LPI_INTR_STICKY	Low Power Idle Receive interrupt
DEV2G5_INTR.AN_PAGE_RX_INTR_STICKY	New Page Event received by ANEG
DEV2G5_INTR.AN_LINK_UP_INTR_STICKY	ANEG - Link status has changed to up
DEV2G5_INTR.AN_LINK_DWN_INTR_STICKY	ANEG - Link status has changed to down
DEV2G5_INTR.LINK_UP_INTR_STICKY	Link status is up
DEV2G5_INTR. LINK_DWN_INTR_STICKY	Link status is down

4.5.1.3 Port Mode Configuration

The MAC provides a number of handles for configuring the port mode. With reference to the MAC_MODE_CFG, MAC_IFG_CFG, and MAC_ENA_CFG registers, the handles are as follows:

Table 4-18. DEV2G5 Port Mode Configurations

Register.Field	Description
MAC_MODE_CFG.FDX_ENA	Enables full-duplex mode. 1: Full Duplex - Default Mode 0: Half Duplex
MAC_MODE_CFG.GIGA_MODE_ENA	1: Enables 1 Gbps or 2.5 Gbps mode (FDX_ENA must be 1), 0: Enables 10 Mbps or 100 Mbps mode
MAC_ENA_CFG.RX_ENA	Enable MAC receiver module
MAC_ENA_CFG.TX_ENA	Enable MAC transmitter module
MAC_IFG_CFG.TX_IFG	Adjust inter frame gap in TX direction
DEV_RST_CTRL.SPEED_SEL	Configure port speed and data rate 0: 10 Mbps 1: 100 Mbps 2: 1000 Mbps 3: No clock. 4–5: Reserved. 6: USXGMII 2500 Mbps

Clearing MAC_ENA_CFG.RX_ENA stops the reception of frames and further frames are discarded. An ongoing frame reception is interrupted.

Clearing MAC_ENA_CFG.TX_ENA stops the dequeueing of frames from the egress queues, which means that frames are held back in the egress queues. An ongoing frame transmission is completed.

TX inter frame gap (MAC_IFG_CFG.TX_IFG) must be set according to the selected speed and mode.

4.5.1.4 Half-Duplex Mode

A number of special configuration options are available for half-duplex (HDX) mode:

- Seed for back-off randomizer: Field MAC_HDX_CFG.SEED seeds the randomizer used by the back-off algorithm. Use MAC_HDX_CFG.SEED_LOAD to load a new seed value.
- Retransmission of frame after excessive collision: Field MAC_HDX_CFG.RETRY_AFTER_EXC_COL_ENA determines whether the MAC retransmits frames after an excessive collision has occurred. If set, a frame is not dropped after excessive collisions, but the back-off sequence is restarted. This is a violation of IEEE 802.3, but is useful in non-dropping half-duplex flow control operation.
- Late collision timing**:** Field MAC_HDX_CFG.LATE_COL_POS adjusts the border between a collision and a late collision in steps of 1 byte. According to IEEE 802.3, section 21.3, this border is permitted to be on data byte 56 (counting frame data from 1); that is, a frame experiencing a collision on data byte 55 is always retransmitted, but it is never retransmitted when the collision is on byte 57. For each higher LATE_COL_POS value, the border is moved 1 byte higher.
- Rx-to-Tx inter-frame gap**:** The sum of Fields MAC_IFG_CFG.RX_IFG1 and MAC_IFG_CFG.RX_IFG2 establishes the time for the Rx-to-Tx inter-frame gap. RX_IFG1 is the first part of half-duplex Rx-to-Tx inter-frame gap. Within RX_IFG1, this timing is restarted if carrier sense (CRS) has multiple high-low transitions (due to noise). RX_IFG2 is the second part of half-duplex Rx-to-Tx inter-frame gap. Within RX_IFG2, transitions on CRS are ignored.

4.5.1.5 Type/Length Check

The MAC supports frame lengths of up to 14,000 bytes. The maximum frame accepted by the MAC is configurable and defined in MAC_MAXLEN_CFG.MAX_LEN.

The MAC allows 1/2/3 tagged frames to be 4/8/12 bytes longer respectively, than the specified maximum length, with MAC_TAGS_CFG.VLAN_LEN_AWR_ENA. The MAC must be configured to look for VLAN tags (MAC_TAGS_CFG.VLAN_AWR_ENA). By default, EtherType 0x8100 is identified a VLAN tag. In addition, three custom EtherTypes can be configured by MAC_TAGS_CFG.TAG_ID, MAC_TAGS_CFG2.TAG_ID2, and MAC_TAGS_CFG2.TAG_ID3.

If a received frame exceeds the maximum length, the frame is truncated and marked as aborted.

The MAC can drop frames with in-range and out-of-range length errors by enabling MAC_ADV_CHK_CFG.LEN_DROP_ENA.

4.5.2 PCS

This section describes the Physical Coding Sublayer (PCS) block where the auto-negotiation process establishes mode of operation for a link. The PCS supports SGMII mode and two SERDES modes: 1000BASE-X and 100BASE-FX.

The following table lists the registers associated with the PCS.

Table 4-19. DEV2G5 PCS Configuration Registers

Register	Description	Replication
PCS1G_CFG	PCS configuration	Per PCS
PCS1G_MODE_CFG	PCS mode configuration	Per PCS
PCS1G_SD_CFG	Signal Detect configuration	Per PCS
PCS1G_ANEG_CFG	Auto-negotiation configuration register	Per PCS
PCS1G_ANEG_NP_CFG	Auto-negotiation next page configuration	Per PCS
PCS1G_LB_CFG	Loopback configuration	Per PCS

.....continued		
Register	**Description**	**Replication**
PCS1G_ANEG_STATUS	Status signaling of PCS auto-negotiation process	Per PCS
PCS1G_ANEG_NP_STATUS	Status signaling of the PCS auto-negotiation next page process	Per PCS
PCS1G_LINK_STATUS	Link Status	Per PCS
PCS1G_LINK_DOWN_CNT	Link Down Counter	Per PCS
PCS1G_STICKY	Sticky bit register	Per PCS

The PCS is enabled in PCS1G_CFG.PCS_ENA and PCS1G_MODE_CFG.SGMII_MODE_ENA selects between the SGMII and SERDES mode. For more information about enabling 100BASE-FX, see 4.5.2.7 100BASE-FX.

The PCS supports the IEEE 802.3, Clause 66 unidirectional mode where the transmission of data is independent of the state of the receive link (PCS1G_MODE_CFG.UNIDIR_MODE_ENA).

4.5.2.1 Auto-Negotiation

Auto-negotiation is enabled with PCS1G_ANEG_CFG.ANEG_ENA. To restart the auto-negotiation process, PCS1G_ANEG_CFG.ANEG_RESTART_ONE_SHOT must be set.

In SGMII mode (PCS1G_MODE_CFG.SGMII_MODE_ENA=1), matching the duplex mode with the link partner must be ignored (PCS1G_ANEG_CFG.SW_RESOLVE_ENA). Otherwise the link is kept down when the auto-negotiation process fails.

The advertised word for the auto-negotiation process (base page) is configured in PCS1G_ANEG_CFG.ADV_ABILITY. The next page information is configured in PCS1G_ANEG_NP_CFG.NP_TX.

When the auto-negotiation state machine has exchanged base page abilities, the PCS1G_ANEG_STATUS.PAGE_RX_STICKY is asserted indicating that the link partner's abilities were received (PCS1G_ANEG_STATUS.LP_ADV_ABILITY).

If next page information need to be exchanged (only available in SERDES mode, that is. PCS1G_MODE_CFG.SGMII_MODE_ENA=0), PAGE_RX_STICKY must be cleared, next page abilities must be written to PCS1G_ANEG_NP_CFG.NP_TX, and PCS1G_ANEG_NP_CFG.NP_LOADED_ONE_SHOT must be set. When the auto-negotiation state machine has exchanged the next page abilities, the PCS1G_ANEG_STATUS.PAGE_RX_STICKY is asserted again indicating that the link partner's next page abilities were received (PCS1G_ANEG_STATUS.LP_NP_RX). Additional exchanges of next page information are possible using the same procedure.

Next page engagement is coded in bit 15 of Base Page of Next page information.

After the last next page has been received, the auto-negotiation state machine enters the IDLE_DETECT state and the PCS1G_ANEG_STATUS.PR bit is set indicating that ability information exchange (base page and possible next pages) has finished and software can resolve priority now. Appropriate actions, such as Rx or Tx reset, or auto-negotiation restart, can then be taken, based on the negotiated abilities. The LINK_OK state is reached one link timer period later.

When the auto-negotiation process reached the LINK_OK state, PCS1G_ANEG_STATUS.ANEG_COMPLETE is asserted.

4.5.2.2 Link Surveillance

The current link status can be observed through PCS1G_LINK_STATUS.LINK_STATUS. The LINK_STATUS is defined as either the PCS synchronization state or as bit 15 of PCS1G_ANEG_STATUS.LP_ADV_ABILITY, which carries information about the link status in SGMII mode.

Link down is defined as the auto-negotiation state machine being in neither the AN_DISABLE_LINK_OK state nor the LINK_OK state for one link timer period. If a link down event occurred PCS1G_STICKY.LINK_DOWN_STICKY is set and additionally PCS1G_LINK_DOWN_CNT is incremented. In SGMII mode, the link timer period is 1.6 ms, whereas in SERDES mode, the link timer period is 10 ms.

The PCS synchronization state can be observed through PCS1G_LINK_STATUS.SYNC_STATUS. Synchronization is lost when the PCS is not able to recover and decode data received from the attached serial link.

4.5.2.3 Signal Detect

The PCS can be enabled to react to loss of signal through signal detect (PCS1G_SD_CFG.SD_ENA). Upon loss of signal, the PCS Rx state machine is restarted, and frame reception stops. If signal detect is disabled, no action is taken upon loss of signal. The polarity of signal detect is configurable in PCS1G_SD_CFG.SD_POL

The source of signal detect is selected in PCS1G_SD_CFG.SD_SEL to either the SERDES PMA or the PMD receiver. If the SERDES PMA is used as source, the SERDES macro provides the signal detect. If the PMD receiver is used as source, signal detect is sampled externally through one of the GPIO pins on the device. For more information about the configuration of the GPIOs and signal detect, see 5.10.8 GPIO Controller.

PCS1G_LINK_STATUS.SIGNAL_DETECT contains the current value of the signal detect input.

4.5.2.4 Tx Loopback

For debug purposes, the Tx data path in the PCS can be looped back into the Rx data path. This is enabled through PCS1G_LB_CFG.TBI_HOST_LB_ENA.

4.5.2.5 Test Patterns

The following table lists the registers associated with configuring test patterns.

Table 4-20. DEV2G5 PCS Test Pattern Configuration Registers

Register	Description	Replication
PCS1G_TSTPAT_MODE_CFG	Test pattern configuration	Per PCS
PCS1G_TSTPAT_STATUS	Test pattern status	Per PCS

PCS1G_TSTPAT_MODE_CFG.JTP_SEL overwrites normal operation of the PCS and enables generation of jitter test patterns for debug. The jitter test patterns are defined in IEEE Std 802.3, Annex 36A, and the following patterns are supported.

- High frequency test pattern
- Low frequency test pattern
- Mixed frequency test pattern
- Continuous random test pattern with long frames
- Continuous random test pattern with short frames

PCS1G_TSTPAT_MODE_STATUS register holds information about error and lock conditions while running the jitter test patterns.

4.5.2.6 Low-Power Idle

The PCS supports Energy Efficient Ethernet (EEE) as defined by IEEE 802.3az. The PCS converts Low Power Idle (LPI) encoding between the MAC and the serial interface transparently.

4.5.2.7 100BASE-FX

The applicable registers are listed in the following table.

Table 4-21. DEV2G5 100BASE-FX Configuration Registers

Register	Description	Replication
PCS_FX100_CFG	Configuration of the PCS 100BASE-FX mode	Per PCS
PCS_FX100_STATUS	Status of the PCS 100BASE-FX mode	Per PCS

The PCS supports a 100BASE-FX mode in addition to the SGMII and 1000BASE-X SERDES modes. The 100BASE-X mode uses 4bit/5bit coding as specified in IEEE802.3 Clause 24 for fiber connections. The 100BASE-FX mode is enabled through PCS_FX100_CFG.PCS_ENA, which masks out all PCS1G related registers.

The following options are available.

- Far-End Fault facility: In 100BASE-FX, the PCS supports the optional Far-End Fault facility. Both Far-End Fault Generation (PCS_FX100_CFG.FEFGEN_ENA) and Far-End Fault Detection (PCS_FX100_CFG.FEFCHK_ENA) are supported. A Far-End Fault incident is recorded in PCS_FX100_STATUS.FEF_FOUND.
- Signal Detect: 100BASE-FX has a similar signal detect scheme as of the SGMII and SERDES modes. For 100BASE-FX, PCS_FX100_CFG.SD_ENA enables signal detect, and PCS_FX100_CFG.SD_SEL selects the input source. The current status of the signal detect input can be observed through PCS_FX100_STATUS.SIGNAL_DETECT. For more information about signal detect, see 4.5.3.1 Signal Detect.
- Link Surveillance: The PCS synchronization status can be observed through PCS_FX100_STATUS.SYNC_STATUS. When synchronization is lost the link breaks and PCS_FX100_STATUS.SYNC_LOST_STICKY is set. The PCS continuously tries to recover the link.
- Unidirectional mode: 100BASE-FX has a similar unidirectional mode as for the SGMII and SERDES modes. This is enabled through PCS_FX100_CFG.UNIDIR_MODE_ENA.

4.5.3 USXGMII PCS

This section describes the Device-level functions and configuration options of the USXGMII PCS.

USXGMII PCS supports 10/100/1000/2500 Mbps in full-duplex mode.

The following tables lists the registers associated with configuration of USXGMII PCS.

Table 4-22. USXGMII PCS Configuration Registers

Register	Description	Replication
USXGMII_PCS_SD_CFG.SD_POL	Signal detect polarity. The signal level on signal_detect input pin must be equal to SD_POL to indicate signal detection	Per USXGMII PCS
USXGMII_PCS_SD_CFG.SD_ENA	Signal Detect Enable. 0: The Signal Detect input pin is ignored. 1: The Signal Detect input pin is used to determine if a signal is detected	Per USXGMII PCS
USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK	Indicates if USXGMII PCS has reached block lock state 0: USXGMII PCS is not in block lock state 1: USXGMII PCS is in block lock state	Per USXGMII PCS
DEV_RST_CTRL.USXGMII_OSET_FILTER_DIS	The USXGMII replicator block samples only the first XGMII word. The other replications are ignored. Only if they contain a Local Fault or remote Fault ordered set the next data will be overwritten. This bit will disable overwrite wit ordered sets. 0: Non-SampleId ordered sets will overwrite the following XGMII words 1: Overwrite of XGMII words with ordered sets is disabled.	Per USXGMII PCS

Table 4-23. USXGMII PCS Rate Adapt Configuration Registers

Register	Description	Replication
USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL	Set the add threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.TX_RADAPT_DROP_LVL	Set the drop threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.TX_LF_GEN_DIS	Disable TX local fault generation when no alignment has been reached	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS
USXGMII_RX_RADAPT_CFG.RX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 Idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS
USXGMII_RX_RADAPT_CFG.RX_RADAPT_ADD_LVL	Set the add threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_RX_RADAPT_CFG.RX_RADAPT_DROP_LVL	Set the drop threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_RX_RADAPT_CFG.RX_LF_GEN_DIS	Disable RX local fault generation when no alignment has been reached	Per USXGMII PCS
USXGMII_RX_RADAPT_CFG.RX_RADAPT_FIFO_FLUSH	One-shot that causes the Rx rate adaptation FIFO to be cleared and reset. Bit is reset by hardware	Per USXGMII PCS

For 10G-QXGMII modes, the USXGMII Rate adaptation register fields should be set to the following values.

USXGMII_RX_RADAPT_CFG.RX_RADAPT_ADD_LVL = 3

USXGMII_RX_RADAPT_CFG.RX_RADAPT_DROP_LVL = 7

USXGMII_RX_RADAPT_CFG.RX_RADAPT_MIN_IFG = 1

USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL = 2

USXGMII_TX_RADAPT_CFG.TX_RADAPT_DROP_LVL = 6

USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG = 1

For 5G-DXGMII modes, the USXGMII Rate adaptation register fields should be set to the following values.

USXGMII_RX_RADAPT_CFG.RX_RADAPT_ADD_LVL = 3

USXGMII_RX_RADAPT_CFG.RX_RADAPT_DROP_LVL = 7

USXGMII_RX_RADAPT_CFG.RX_RADAPT_MIN_IFG = 1

USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL = 1

USXGMII_TX_RADAPT_CFG.TX_RADAPT_DROP_LVL = 5

USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG = 1

4.5.3.1 Signal Detect

The USXGMII_PCS can be enabled to react to loss of signal through signal detect (USXGMII_PCS_SD_CFG.SD_ENA). The polarity of signal detect is configurable USXGMII_PCS_SD_CFG.SD_ENA.SD_POL.

USXGMII_PCS_SD_CFG.SD_ENA is used to determine if a signal is detected.

0: The Signal Detect input pin is ignored

1: The Signal Detect input pin is used to determine if a signal is detected

If signal detect status is not detected then USXGMII ANEG is restarted.

USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK shows USXGMII PCS has reached block lock state.

0: USXGMII PCS is not in block lock state

1: USXGMII PCS is in block lock state

4.5.3.2 USXGMII ANEG - Auto Negotiation

Hardware Assisted Auto-Negotiation described in Section 2.7.6 of USXGMII- Multiport Copper PHY spec is enabled by configuring USXGMII_ANEG_CFG.ANEG_ENA to 1. To restart the auto-negotiation process, USXGMII_ANEG_CFG.ANEG_RESTART_ONE_SHOT must be set.

Link timer for the Auto-Negotiation Process can be set by configuring the register USXGMII_ANEG_CFG.LINK_TIMER appropriately.

USXGMII_ANEG_CFG.SW_RESOLVE_ENA should always be set to 1 to enable Software resolution.

The advertised word for the auto-negotiation process is configured in USXGMII_ANEG_CFG.ADV_ABILITY.

When the auto-negotiation state machine has exchanged abilities, the USXGMII_ANEG_STATUS.PAGE_RX_STICKY is asserted indicating that the link partner’s abilities were received (in USXGMII_ANEG_STATUS.LP_ADV_ABILITY).

After the abilities are exchanged between Local Device and Link Partner, the auto-negotiation state machine enters the IDLE_DETECT state and the USXGMII_ANEG_STATUS.PR bit is set indicating that ability information exchange has finished, and software can resolve priority now. Appropriate actions, such as Rx or Tx reset, Device/MAC Speed settings etc., can then be taken, based on the negotiated abilities. The LINK_OK state is reached one link timer period later.

When the auto-negotiation process reached the LINK_OK state, USXGMII_ANEG_STATUS.ANEG_COMPLETE is asserted.

Typically loss of block_lock, would cause the an_sync_status to be set to FAIL and the Auto-Negotiation shall restart. But by setting USXGMII_ANEG_CFG.BLK_LCK_HYS_ENA to 1, an_sync_status shall be set to FAIL only after block_lock is set to 0 for at least LINK_TIMER duration. This is provided as a configuration to support any such interpretations of the specification. But, by default it is set to 0 and is not expected to set to 1.

4.5.4 Port statistics

Port statistics for DEV2G5 port modules are collected in the assembler and is accessed through registers in the assembler.

4.6 DEV5G Port Module

The DEV5G port module can be used in the following full-duplex operation modes.

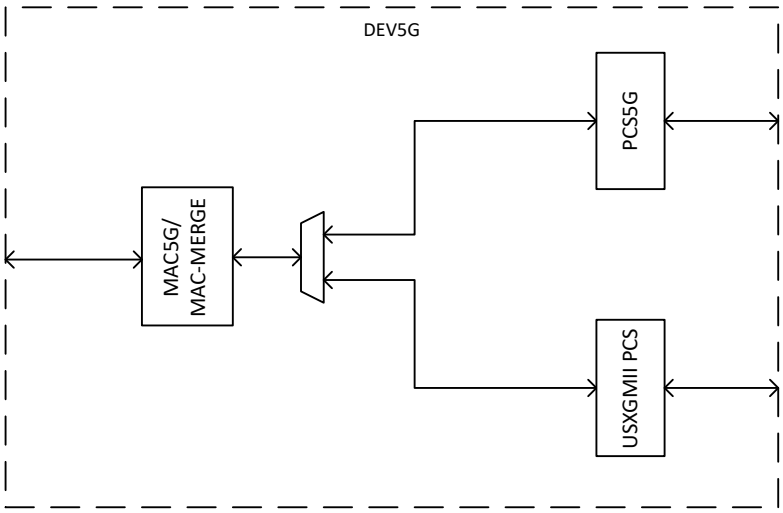
Table 4-24. DEV5G Operation Modes

Mode	5GBASE-R
Lane speed	5.156
No. of Lanes	1
Encoding	64b/66b
PCS used	CLAUSE 129
IPG shrink (average)	No
Preamble shrink	No
Data bandwidth	5 Gbps

Apart from above the DEV5G block also can also operate in 10G-DXGMII mode of operation as per Cisco USXGMII spec.

DEV5G port module consists of MAC which also handles MAC Merge sub-layer function, along with PCS5G, USXGMII PCS.

Figure 4-9. DEV10G Port Module



4.6.1 MAC

This section describes the high level functions and the configuration options of the Media Access Controller (MAC) that is used in the DEV5G port modules.

The applicable registers are listed in the following table.

Table 4-25. DEV5G MAC Configuration Registers

Register	Description	Replication
MAC_ENA_CFG	Enabling of Rx and Tx data paths	Per port
MAC_MODE_CFG	Port mode configuration	Per port
MAC_MAXLEN_CFG	Maximum length configuration	Per port
MAC_NUM_TAGS_CFG	Number of tags configuration	Per port

.....continued		
Register	**Description**	**Replication**
MAC_TAGS_CFG	VLAN/service tag configuration	Three per port
MAC_ADV_CHK_CFG	Advance check feature configuration	Per port
MAC_LFS_CFG	Link fault signaling configuration	Per port
MAC_LB_CFG	Loopback configuration	Per port
MAC_STICKY	Sticky bit recordings	Per port

4.6.1.1 Clock and Reset

There are a number of synchronous resets in the DEV5G port module. All of the resets can be set and cleared simultaneously. By default, all blocks are in the reset state (reset active). The following table lists all available resets.

Table 4-26. DEV5G Resets

Target::Register.Field	Description
DEV5G::DEV_RST_CTRL.MAC_RX_RST	Reset MAC receiver
DEV5G::DEV_RST_CTRL.MAC_TX_RST	Reset MAC transmitter
DEV5G::DEV_RST_CTRL.PCS_RX_RST	Reset PCS receiver
DEV5G::DEV_RST_CTRL.PCS_TX_RST	Reset PCS transmitter

When changing the MAC configuration, the port must go through a reset cycle. This is done by writing the register DEV5G::DEV_RST_CTRL twice. On first write all reset bits must be set to active (0x1). On the second write resets bits are cleared (0x0). Non-reset bits in DEV5G::DEV_RST_CTRL must keep their value for both writes.

4.6.1.2 Interrupts

There are 5 interrupt sources defined for DEV5G port module, which can be found below. General interrupt handling is described in detail in 5.10.13 Outbound Interrupt Controller.

Table 4-27. DEV5G Interrupt Sources

Target::Register.Field	Description
DEV5G::INTR. PCS_BR_INTR	One of the Base-R PCS interrupt indications is active
DEV5G::INTR.TX_LPI_INTR	TX Low Power Idle mode has changed
DEV5G::INTR.RX_LPI_INTR	RX Low Power Idle mode has changed
DEV5G::INTR.LINK_UP_INTR	Link status is up
DEV5G::INTR. LINK_DWN_INTR	Link status is down

All the sticky events in XFI PCS are capable of generating an interrupt if their corresponding interrupt mask is set to 1. The following are the interrupt registers that are used for interrupt generation if corresponding event bit is set from 5G Base-R PCS (or XFI PCS).

Table 4-28. DEV5G Base-R PCS Interrupt Sources

Register	Sticky Bits that can be Enabled for Interrupt Generation	Replication
PCS_INTR_MASK	XFI PCS sticky bits	Per PCS

.....continued		
Register	**Sticky Bits that can be Enabled for Interrupt Generation**	**Replication**
KR_FEC_STICKY_MASK	KR FEC sticky bits	Per PCS
EEE_INTR_MASK	XFI PCS EEE sticky bits	Per PCS

4.6.1.3 Port Mode Configuration

The MAC provides a number of handles for configuring the port mode.

Table 4-29. DEV5G Port Mode Configurations

Target::Register.Field	Description
DEV5G::DEV_RST_CTRL.SPEED_SEL	Configuration of MAC and PCS Rx/Tx clock frequencies 1: MAC works at 5 Gbps Unused values are reserved.
DEV5G::MAC_ENA_CFG.RX_ENA	Enable MAC Receiver Module 0: Disabled 1: Enabled
DEV5G::MAC_ENA_CFG.TX_ENA	Enable MAC Transmitter Module 0: Disabled 1: Enabled
DEV5G::MAC_MODE_CFG.TUNNEL_PAUSE_FRAMES	Enable tunnelling of Pause Frames on a link 0: Pause frames are expected to be in Link Pause Frames
DEV5G::MAC_MODE_CFG.MAC_PREAMBLE_CFG	Preamble format configuration 0: Standard Preamble
DEV5G::MAC_MODE_CFG.MAC_IPG_CFG	IPG format configuration 0: Normal IPG
DEV5G::MAC_MODE_CFG. HIH_CRC_CHECK	Enable check for HiH checksum 0: MAC5G should disregard the HIH checksum

4.6.1.4 Type/Length Check

The MAC supports frame lengths of up to 14,000 bytes. The maximum frame accepted by the MAC is configurable and defined in DEV5G::MAC_MAXLEN_CFG.MAX_LEN. Maximum length is automatically adjusted if one or more tags are included in the current frame. The MAC allows 1/2/3 tagged frames to be 4/8/12 bytes longer respectively, than the specified maximum length. DEV5G::MAC_MAXLEN_CFG.MAX_LEN_TAG_CHK configures whether the max Length check takes the number of tags into consideration.

Number of tags can be selected by DEV5G::MAC_NUM_TAGS_CFG.NUM_TAGS. By default, EtherType 0x8100 is identified a VLAN tag. In addition, three custom EtherTypes can be configured by MAC_TAGS_CFG.TAG_ID.

If a received frame exceeds the maximum length the frame is truncated and marked as aborted.

The MAC can drop the frames with length (in-range and out-of-range) errors by enabling MAC_ADV_CHK_CFG.INR_ERR_ENA and MAC_ADV_CHK_CFG.OOR_ERR_ENA.

There are also other checks available which can be enabled based on following configurations.

Table 4-30. DEV5G Port Module Advance Check Feature Configuration

Target::Register.Field	Description
DEV5G::MAC_ADV_CHK_CFG.EXT_EOP_CHK_ENA	Enable extended end of packet check
DEV5G::MAC_ADV_CHK_CFG.EXT_SOP_CHK_ENA	Enable extended start of packet check
DEV5G::MAC_ADV_CHK_CFG.SFD_CHK_ENA	Enable start-of-frame-delimiter check
DEV5G::MAC_ADV_CHK_CFG.PRM_SHK_CHK_DIS	Disable preamble shrink check
DEV5G::MAC_ADV_CHK_CFG.PRM_CHK_ENA	Enable preamble check
DEV5G::MAC_ADV_CHK_CFG.OOR_ERR_ENA	Enable out-of-range error check
DEV5G::MAC_ADV_CHK_CFG.INR_ERR_ENA	Enable in-range error check

4.6.1.5 Local and Remote Fault Signaling

By default the MAC is configured to detect a Link Fault indication and react by transmitting the appropriate sequence ordered set. The feature can be disabled completely using DEV5G::MAC_LFS_CFG.LFS_MODE_ENA. It is also possible to configure MAC into unidirectional mode according to IEEE 802.3, Clause 66, where frames are transmitted while receiving link fault indication but interleaved by appropriate ordered sets in between (DEV5G::MAC_LFS_CFG.LFS_UNIDIR_ENA).

4.6.2 PCS5G

This section describes the Physical Coding Sublayer (PCS) blocks included in the DEV5G port module. Depending on the DEV5G mode configuration, the corresponding PCS block must be selected and configured accordingly. Status information such as link status is available per PCS block and the right register bit, depending on mode must be checked. Interrupt source signal are selected from the current selected PCS block depending on mode selection.

Note:
Each DEV5G port module must only enable one PCS block at the same time.

Applicable registers for DEV5G PCS configuration and status can be found in the following tables.

Table 4-31. DEV5G PCS Configuration Registers

Register	Description	Replication
PCS_CFG	XFI-PCS configuration	Per PCS
PCS_SD_CFG	XFI-PCS signal detect configuration	Per PCS

Table 4-32. DEV5G PCS Status Registers

Register	Description	Replication
PCS_STATUS	XFI-PCS status	Per PCS
TX_ERRBLK_CNT	XFI-PCS counter for number of times Transmit FSM enters TX_E state.	Per PCS

.....continued		
Register	**Description**	**Replication**
TX_CHARERR_CNT	XFI-PCS counter for number of invalid control characters transmitter detected from MAC	Per PCS
RX_BER_CNT	XFI-PCS counter for number of times Receive BER FSM enters BER_BAD_SH state (ber_count).	Per PCS
RX_ERRBLK_CNT	XFI-PCS counter for number of times Receive FSM enters RX_E state.	Per PCS
RX_CHARERR_CNT	XFI-PCS counter for Receive number of invalid control characters after decoding.	Per PCS
RX_OSET_FIFO_STAT	XFI-PCS receive ordered set FIFO status.	Per PCS
RX_OSET_FIFO_DATA	XFI-PCS receive ordered set FIFO data.	Per PCS
RX_FSET_FIFO_STAT	XFI-PCS receive F-Sig FIFO status.	Per PCS
RX_FSET_FIFO_DATA	XFI-PCS receive F-Sig FIFO data.	Per PCS

4.6.2.1 Backplane Auto-Negotiation

Backplane Auto-Negotiation must be used when normal Auto-Negotiation is disabled. For the DEV5G port module there is no specific function or setup available.

4.6.2.2 Link Surveillance

For XFI mode of operation, current link status can be observed through PCS5G_BR::PCS_STATUS as mentioned in the following paragraph:

The XFI PCS must achieve block lock (PCS5G_BR::PCS_STATUS.RX_BLOCK_LOCK) to be able to receive correct data. This block lock is achieved when PCS detects a 66-bit boundary. Whenever the block lock is lost or achieved, a sticky bit is set (PCS5G_BR::PCS_INTR_STAT.LOCK_CHANGED_STICKY). Whenever a large number of false sync headers is detected HI_BER (PCS5G_BR::PCS_STATUS.RX_HI_BER) is asserted and a sticky bit is set (PCS5G_BR::PCS_INTR_STAT.RX_HI_BER_STICKY). If a local fault or remote fault is received then it is flagged in PCS5G_BR::PCS_INTR_STAT.RX_OSET_STICKY and the sequence ordered set received can be read from PCS5G_BR::RX_OSET_FIFO_DATA.RX_OSET_FIFO_DATA. PCS5G_BR::RX_FSET_FIFO_STAT can be used to know how many ordered sets are captured and FIFO full status.

4.6.2.3 Signal Detect

The PCS can be enabled to react to loss of signal through signal detect (x_SD_CFG.SD_ENA). Upon loss of signal, the PCS Rx state machine is restarted, and frame reception stops. If signal detect is disabled, no action is taken upon loss of signal. The polarity of signal detect is configurable by x_SD_CFG.SD_POL, where value of this bit indicates the active ‘Signal Detect’ state of the selected input line.

The source of signal detect is selected in x_SD_CFG.SD_SEL to either the SERDES PMA or the PMD receiver. If the SERDES PMA is used as source, the SERDES macro provides the signal detect. If the PMD receiver is used as source, signal detect is sampled externally through one of the GPIO pins on the device. For more information about the configuration of the GPIOs and signal detect, see 5.10.8 GPIO Controller.

4.6.2.4 Tx Loopback

For debug purposes there is a loop back path defined within all PCS blocks, which loops back Tx data path into the Rx data path. This is enabled through PCS5G_BR::PCS_CFG.PMA_LOOPBACK_ENA for PCS XFI.

4.6.2.5 Test Patterns

For XFI PCS applicable registers are listed in following table.

Table 4-33. DEV5G Base-R PCS Test Pattern Configuration Registers

Target::Register.Field	Description	Replication
PCS5G_BR::PCS_CFG.TX_TEST_MODE	Enables test pattern mode in XFI PCS transmitter.	Per PCS
PCS5G_BR::PCS_CFG.TX_REST_MODE	Enables test pattern mode in XFI PCS receiver.	Per PCS
PCS5G_BR::TEST_CFG	Test pattern configuration register when test mode is enabled for XFI PCS	Per PCS
PCS5G_BR::TX_SEEDA_MSB.TX_SEEDA_MSB	Most significant 26 bits of seed A used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS5G_BR::TX_SEEDA_LSB.TX_SEEDA_LSB	Least significant 32 bits of seed A used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS5G_BR::TX_SEEDB_MSB.TX_SEEDB_MSB	Most significant 26 bits of seed B used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS5G_BR::TX_SEEDB_LSB.TX_SEEDB_LSB	Least significant 32 bits of seed B used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS5G_BR::TX_DATAPAT_MSB.TX_DATAPAT_MSB	Most significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for transmitter of XFI PCS.	Per PCS
PCS5G_BR::TX_DATAPAT_LSB.TX_DATAPAT_LSB	Least significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for transmitter of XFI PCS.	Per PCS
PCS5G_BR::RX_DATAPAT_MSB.RX_DATAPAT_MSB	Most significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for receiver of XFI PCS.	Per PCS
PCS5G_BR::RX_DATAPAT_LSB.RX_DATAPAT_LSB	Least significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for receiver of XFI PCS.	Per PCS
PCS5G_BR::PCS_STATUS.TESTPAT_MATCH	When in test pattern check mode, this bit will read 1 if the test pattern checker detects a match. When 0, the test pattern does not match. The test pattern error counts should still be used along with this register bit to determine proper test match status. The bit will read back 1 only when the test pattern is matching. This may happen even while test pattern errors are counted on other clock cycles.	Per PCS
PCS5G_BR::TEST_ERR_CNT.TEST_ERR_CNT	Count of detected test pattern errors in Rx test pattern checker. Write 0 to clear.	Per PCS

By setting register bits PCS5G_BR::PCS_CFG.TX_TEST_MODE and PCS5G_BR::PCS_CFG.RX_TEST_MODE to 1, test pattern mode is enabled in XFI PCS. Below test patterns are supported as per IEEE Std 802.3, clause 49.2.8 and 49.2.12.

- Square wave
- Pseudo random
- PRBS31
- User defined

User defined mode is a slightly modified test pattern in which scrambler will not be initialized with any seed value and so there will not be any errors after every 128-block window.

Different types of test pattern generation can be chosen by configuring PCS5G_BR::TEST_CFG.TX_TESTPAT_SEL and that of checkers by configuring PCS5G_BR::TEST_CFG.RX_TESTPAT_SEL. For square wave test pattern generation, period can be configured using PCS5G_BR::TEST_CFG.TX_SQPW_4B. For pseudo-random, test pattern inversion of seeds and data is enabled by PCS5G_BR::TEST_CFG.TX_DSBL_INV and PCS5G_BR::TEST_CFG.RX_DSBL_INV.

4.6.3 USXGMII PCS

This section describes Device level functions and configuration options of the USXGMII PCS.

USXGMII PCS supports 10/100/1000/2500/5000 Mbps in full-duplex mode.

The following tables list the registers associated with configuration of USXGMII PCS.

Table 4-34. USXGMII PCS Configuration Registers

Register	Description	Replication
USXGMII_PCS_SD_CFG.SD_POL	Signal detect polarity. The signal level on signal_detect input pin must be equal to SD_POL to indicate signal detection	Per PCS
USXGMII_PCS_SD_CFG.SD_ENA	Signal Detect Enable. 0: The Signal Detect input pin is ignored. 1: The Signal Detect input pin is used to determine if a signal is detected	Per PCS
USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK	Indicates if USXGMII PCS has reached block lock state 0: USXGMII PCS is not in block lock state, 1: USXGMII PCS is in block lock state	Per PCS
DEV_RST_CTRL.USXGMII_OSET_FILTER_DIS	The USXGMII replicator block samples only the first XGMII word. The other replications are ignored. Only if they contain a Local Fault or remote Fault ordered set the next data will be overwritten. This bit will disable overwrite wit ordered sets. 0': Non-SampleId ordered sets will overwrite the following XGMII words '1': Overwrite of XGMII words with ordered sets is disabled.	Per PCS

Table 4-35. USXGMII PCS Rate Adapt Configuration Registers

Register	Description	Replication
USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL	Set the add threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS

USXGMII_TX_RADAPT_CFG. TX_RADAPT_DROP_LVL	Set the drop threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG. TX_LF_GEN_DIS	Disable TX local fault generation when no alignment has been reached	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG. TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS

For 10G-DXGMII modes, the USXGMII Rate adaptation register fields should be set to the following values.

USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL = 4

USXGMII_TX_RADAPT_CFG.TX_RADAPT_DROP_LVL = 8

USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG = 1

4.6.3.1 Signal Detect

The USXGMII_PCS can be enabled to react to loss of signal through signal detect (USXGMII_PCS_SD_CFG.SD_ENA). The polarity of signal detect is configurable USXGMII_PCS_SD_CFG.SD_ENA.SD_POL.

USXGMII_PCS_SD_CFG.SD_ENA is used to determine if a signal is detected.

0: The Signal Detect input pin is ignored.

1: The Signal Detect input pin is used to determine if a signal is detected

If signal detect status is not detected, then USXGMII ANEG is restarted.

USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK shows USXGMII PCS has reached block lock state.

0: USXGMII PCS is not in block lock state

1: USXGMII PCS is in block lock state

4.6.3.2 USXGMII ANEG - Auto Negotiation

The DEV5G USXGMII Auto-negotiation is similar to DEV2G5 USXGMII ANEG. Refer to 4.5.3.2 USXGMII ANEG - Auto Negotiation.

4.6.4 Port Statistics

The DEV5G port module contains a set of port statistics. Packets counters are 32 bits wide and byte counters are 40 bits wide. The following table lists the counters.

Table 4-36. DEV5G Statistics

Register	Description
RX_SYMBOL_ERR_CNT	Rx Symbol Carrier Error Counter
RX_PAUSE_CNT	Rx Pause Frame Counter
RX_UNSUP_OPCODE_CNT	Rx Control Frame Counter
RX_UC_CNT	Rx Unicast Frame Counter
RX_MC_CNT	Rx Multicast Frame Counter
RX_BC_CNT	Rx Broadcast Frame Counter

.....continued

Register	Description
RX_CRC_ERR_CNT	Rx CRC Error Counter
RX_UNDERSIZE_CNT	Rx Undersize Counter (valid frame format)
RX_FRAGMENTS_CNT	Rx Undersize Counter (CRC error)
RX_IN_RANGE_LEN_ERR_CNT	Rx In-range Length Error Counter
RX_OUT_OF_RANGE_LEN_ERR_CNT	Rx Out-Of-Range Length Error Counter
RX_OVERSIZE_CNT	Rx Oversize Counter (valid frame format)
RX_JABBERS_CNT	Rx Jabbers Counter
RX_SIZE64_CNT	Rx 64 Byte Frame Counter
RX_SIZE65TO127_CNT	Rx 65-127 Byte Frame Counter
RX_SIZE128TO255_CNT	Rx 128-255 Byte Frame Counter
RX_SIZE256TO511_CNT	Rx 256-511 Byte Frame Counter
RX_SIZE512TO1023_CNT	Rx 512-1023 Byte Frame Counter
RX_SIZE1024TO1518_CNT	Rx 1024-1518 Byte Frame Counter
RX_SIZE1519TOMAX_CNT	Rx 1519 To Max. Length Byte Frame Counter
RX_IPG_SHRINK_CNT	Rx Inter Packet Gap Shrink Counter
TX_PAUSE_CNT	Tx Pause Frame Counter
TX_UC_CNT	Tx Unicast Frame Counter
TX_MC_CNT	Tx Multicast Frame Counter
TX_BC_CNT	Tx Broadcast Frame Counter
TX_SIZE64_CNT	Tx 64 Byte Frame Counter
TX_SIZE65TO127_CNT	Tx 65-127 Byte Frame Counter
TX_SIZE128TO255_CNT	Tx 128-255 Byte Frame Counter
TX_SIZE256TO511_CNT	Tx 256-511 Byte Frame Counter
TX_SIZE512TO1023_CNT	Tx 512-1023 Byte Frame Counter
TX_SIZE1024TO1518_CNT	Tx 1024-1518 Byte Frame Counter
TX_SIZE1519TOMAX_CNT	Tx 1519 To Max. Length Byte Frame Counter
RX_ALIGNMENT_LOST_CNT	Counter to track the dribble-nibble (extra nibble) errors in frames.
RX_TAGGED_FRMS_CNT	Counts frames that are tagged (C-Tagged or S-Tagged).
RX_UNTAGGED_FRMS_CNT	Counts frames that are Not tagged (neither C-Tagged nor S-Tagged).

.....continued

Register	Description
TX_TAGGED_FRMS_CNT	Counts frames that are tagged (C-Tagged or S-Tagged).
TX_UNTAGGED_FRMS_CNT	Counts frames that are Not tagged (neither C-Tagged nor S-Tagged).
RX_HIH_CKSM_ERR_CNT	Rx HiH Checksum Error Counter
RX_XGMII_PROT_ERR_CNT	Rx XGMII Protocol Error Counter
PMAC_RX_SYMBOL_ERR_CNT	PMAC Rx Symbol Carrier Error Counter
PMAC_RX_PAUSE_CNT	PMAC Rx Pause Frame Counter
PMAC_RX_UNSUP_OPCODE_CNT	PMAC Rx Control Frame Counter
PMAC_RX_UC_CNT	PMAC Rx Unicast Frame Counter
PMAC_RX_MC_CNT	PMAC Rx Multicast Frame Counter
PMAC_RX_BC_CNT	PMAC Rx Broadcast Frame Counter
PMAC_RX_CRC_ERR_CNT	PMAC Rx CRC Error Counter
PMAC_RX_UNDERSIZE_CNT	PMAC Rx Undersize Counter (valid frame format)
PMAC_RX_FRAGMENTS_CNT	PMAC Rx Undersize Counter (CRC error)
PMAC_RX_IN_RANGE_LEN_ERR_CNT	PMAC Rx In-range Length Error Counter
PMAC_RX_OUT_OF_RANGE_LEN_ERR_CNT	PMAC Rx Out-Of-Range Length Error Counter
PMAC_RX_OVERSIZE_CNT	PMAC Rx Oversize Counter (valid frame format)
PMAC_RX_JABBERS_CNT	PMAC Rx Jabbers Counter
PMAC_RX_SIZE64_CNT	PMAC Rx 64 Byte Frame Counter
PMAC_RX_SIZE65TO127_CNT	PMAC Rx 65-127 Byte Frame Counter
PMAC_RX_SIZE128TO255_CNT	PMAC Rx 128-255 Byte Frame Counter
PMAC_RX_SIZE256TO511_CNT	PMAC Rx 256-511 Byte Frame Counter
PMAC_RX_SIZE512TO1023_CNT	PMAC Rx 512-1023 Byte Frame Counter
PMAC_RX_SIZE1024TO1518_CNT	PMAC Rx 1024-1518 Byte Frame Counter
PMAC_RX_SIZE1519TOMAX_CNT	PMAC Rx 1519 To Max. Length Byte Frame Counter
PMAC_TX_PAUSE_CNT	PMAC Tx Pause Frame Counter
PMAC_TX_UC_CNT	PMAC Tx Unicast Frame Counter
PMAC_TX_MC_CNT	PMAC Tx Multicast Frame Counter
PMAC_TX_BC_CNT	PMAC Tx Broadcast Frame Counter
PMAC_TX_SIZE64_CNT	PMAC Tx 64 Byte Frame Counter
PMAC_TX_SIZE65TO127_CNT	PMAC Tx 65-127 Byte Frame Counter

.....continued

Register	Description
PMAC_TX_SIZE128TO255_CNT	PMAC Tx 128-255 Byte Frame Counter
PMAC_TX_SIZE256TO511_CNT	PMAC Tx 256-511 Byte Frame Counter
PMAC_TX_SIZE512TO1023_CNT	PMAC Tx 512-1023 Byte Frame Counter
PMAC_TX_SIZE1024TO1518_CNT	PMAC Tx 1024-1518 Byte Frame Counter
PMAC_TX_SIZE1519TOMAX_CNT	PMAC Tx 1519 To Max. Length Byte Frame Counter
PMAC_RX_ALIGNMENT_LOST_CNT	PMAC Counter to track the dribble-nibble (extra nibble) errors in frames.
MM_RX_ASSEMBLY_ERR_CNT	MAC MERGE - The number of Pframes received with aseembly error
MM_RX_SMD_ERR_CNT	MAC MERGE - The number of Pframe fragaments rejected die to unknown SMD values or arrving with SMD-C when no pframe in progress
MM_RX_ASSEMBLY_OK_CNT	MAC MERGE - The number of Pframes received whitch are successfully reassembled
MM_RX_MERGE_FRAG_CNT	MAC MERGE - The number of Pframes received whitch are mPackets (Fragments) due to preemption
MM_TX_PFRAGMENT_CNT	MAC MERGE - The number of Pframe fragments sent when preemption enabled
PMAC_RX_HIH_CKSM_ERR_CNT	PMAC Rx HiH Checksum Error Counter
PMAC_RX_XGMII_PROT_ERR_CNT	PMAC Rx XGMII Protocol Error Counter
RX_IN_BYTES_CNT	The number of bytes received (good, bad, and framing).
RX_IN_BYTES_MSB_CNT	The number of bytes received (good, bad, and framing) - MSBs only
RX_OK_BYTES_CNT	The number of received bytes in good frames.
RX_OK_BYTES_MSB_CNT	The number of received bytes in good frames - MSBs only.
RX_BAD_BYTES_CNT	The number of received bytes in bad frames.
RX_BAD_BYTES_MSB_CNT	The number of received bytes in bad frames - MSBs only
TX_OUT_BYTES_CNT	The number of bytes transmitted (good, bad and framing)
TX_OUT_BYTES_MSB_CNT	The number of bytes transmitted (good, bad, framing) - MSBs only
TX_OK_BYTES_CNT	The number of bytes transmitted successfully.
TX_OK_BYTES_MSB_CNT	The number of transmitted bytes transmitted successfully - MSBs only
PMAC_RX_OK_BYTES_CNT	PMAC - The number of received bytes in good frames.

.....continued	
Register	**Description**
PMAC_RX_OK_BYTES_MSB_CNT	PMAC - The number of received bytes in good frames - MSBs only.
PMAC_RX_BAD_BYTES_CNT	PMAC - The number of received bytes in bad frames
PMAC_RX_BAD_BYTES_MSB_CNT	PMAC - The number of received bytes in bad frames - MSBs only
PMAC_TX_OK_BYTES_CNT	PMAC - The number of bytes transmitted successfully.
PMAC_TX_OK_BYTES_MSB_CNT	PMAC - The number of transmitted bytes transmitted successfully - MSBs only.

The counters are writable. In particular, the counters are cleared by writing a 0 to each counter.

4.7 DEV10G Port Module

The DEV10G port module can be used in the following full-duplex operation modes.

Table 4-37. DEV10G Operation Modes – 10G speed

Mode	10GBASE-R
Lane speed	10.3125
No. of Lanes	1
Encoding	64b/66b
PCS used	CLAUSE 49
IPG shrink (average)	No
Preamble shrink	No
Data bandwidth	10 Gbps
Payload bandwidth (compensated for stacking header)	10 Gbps

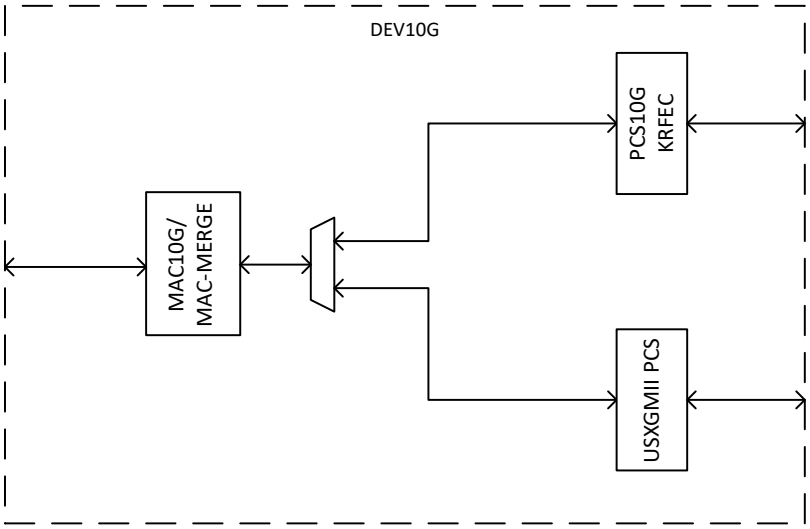
Table 4-38. Table 10 DEV10G Operation Modes – 5G speed

Mode	5GBASE-R
Lane speed	5.156
No. of Lanes	1
Encoding	64b/66b
PCS used	CLAUSE 129
IPG shrink (average)	No
Preamble shrink	No
Data bandwidth	5 Gbps
Payload bandwidth (compensated for stacking header)	5 Gbps

Apart from above the DEV10G block also can also operate in 10G-DXGMII mode of operation as per Cisco USXGMII spec.

DEV10G port module consists of MAC which also handles MAC Merge sub-layer function, along with PCS10G (10G speed with optional KRFEC, 5G speeds NO FEC), USXGMII PCS.

Figure 4-10. DEV10G Port Module



4.7.1 MAC

This section describes the high level functions and the configuration options of the MAC that is used in the DEV10G port modules.

The applicable registers are listed in the following table.

Table 4-39. DEV10G MAC Configuration Registers

Register	Description	Replication
MAC_ENA_CFG	Enabling of Rx and Tx data paths	Per port
MAC_MODE_CFG	Port mode configuration	Per port
MAC_MAXLEN_CFG	Maximum length configuration	Per port
MAC_NUM_TAGS_CFG	Number of tags configuration	Per port
MAC_TAGS_CFG	VLAN/service tag configuration	3 per port
MAC_ADV_CHK_CFG	Advance check feature configuration	Per port
MAC_LFS_CFG	Link fault signaling configuration	Per port
MAC_LB_CFG	Loopback configuration	Per port
MAC_STICKY	Sticky bit recordings	Per port

4.7.1.1 Clock and Reset

There are a number of synchronous resets in the DEV10G port module. All of the resets can be set and cleared simultaneously. By default, all blocks are in the reset state (reset active). The following table lists all the available resets.

Table 4-40. DEV10G Resets

Target::Register.Field	Description
DEV10G::DEV_RST_CTRL.MAC_RX_RST	Reset MAC receiver
DEV10G::DEV_RST_CTRL.MAC_TX_RST	Reset MAC transmitter
DEV10G::DEV_RST_CTRL.PCS_RX_RST	Reset PCS receiver
DEV10G::DEV_RST_CTRL.PCS_TX_RST	Reset PCS transmitter

When changing the MAC configuration, the port must go through a reset cycle. This is done by writing the register DEV10G::DEV_RST_CTRL twice. On first write all reset bits must be set to active (0x1). On the second write resets bits are cleared (0x0). Non-reset bits in DEV10G::DEV_RST_CTRL must keep their value for both writes.

4.7.1.2 Interrupts

There are five interrupt sources defined for DEV10G port module, which can be found below. For more detail description of general interrupt handling, see 5.10.13 Outbound Interrupt Controller.

Table 4-41. DEV10G Interrupt Sources

Target::Register.Field	Description
DEV10G::INTR.PCS_BR_INTR	One of the Base-R PCS interrupt indications is active
DEV10G::INTR.TX_LPI_INTR	TX Low Power Idle mode has changed
DEV10G::INTR.RX_LPI_INTR	RX Low Power Idle mode has changed
DEV10G::INTR.LINK_UP_INTR	Link status is up
DEV10G::INTR.LINK_DWN_INTR	Link status is down

All the sticky events in XFI PCS are capable of generating an interrupt if their corresponding interrupt mask is set to 1. Below are the interrupt registers that are used for interrupt generation if corresponding event bit is set from 10G Base-R PCS (or XFI PCS).

Table 4-42. Dev10G Base-R PCS Interrupt Sources

Register	Sticky Bits that can be Enabled for Interrupt Generation	Replication
PCS_INTR_MASK	XFI PCS sticky bits	Per PCS
KR_FEC_STICKY_MASK	KR FEC sticky bits	Per PCS
EEE_INTR_MASK	XFI PCS EEE sticky bits	Per PCS

4.7.1.3 Port Mode Configuration

The MAC provides a number of handles for configuring the port mode, which are listed in the following table.

Table 4-43. DEV10G Port Mode Configurations

Target::Register.Field	Description
DEV10G::DEV_RST_CTRL.SPEED_SEL	Configuration of MAC and PCS Rx/Tx clock frequencies 0: MAC works at 10 Gbps 1: MAC works at 5 Gbps Unused values are reserved.
DEV10G::MAC_ENA_CFG.RX_ENA	Enable MAC Receiver Module 0: Disabled 1: Enabled
DEV10G::MAC_ENA_CFG.TX_ENA	Enable MAC Transmitter Module 0: Disabled 1: Enabled
DEV10G::MAC_MODE_CFG.TUNNEL_PAUSE_FRAMES	Enable tunnelling of Pause Frames on a link 0: Pause frames are expected to be in Link Pause Frames
DEV10G::MAC_MODE_CFG.MAC_PREAMBLE_CFG	Preamble format configuration 0: Standard Preamble
DEV10G::MAC_MODE_CFG.MAC_IPG_CFG	IPG format configuration 0: Normal IPG
DEV10G::MAC_MODE_CFG.HIH_CRC_CHECK	Enable check for HiH checksum 0: MAC10G should disregard the HIH checksum

4.7.1.4 Type/Length Check

The MAC supports frame lengths of up to 14,000 bytes. The maximum frame accepted by the MAC is configurable and defined in DEV10G::MAC_MAXLEN_CFG.MAX_LEN. Maximum length is automatically adjusted if one or more tags are included in the current frame. The MAC allows 1/2/3 tagged frames to be 4/8/12 bytes longer respectively, than the specified maximum length. DEV10G::MAC_MAXLEN_CFG.MAX_LEN_TAG_CHK configures whether the max Length check takes the number of tags into consideration.

Number of tags can be selected by DEV10G::MAC_NUM_TAGS_CFG.NUM_TAGS. By default, EtherType 0x8100 is identified a VLAN tag. In addition, three custom EtherTypes can be configured by MAC_TAGS_CFG.TAG_ID.

If a received frame exceeds the maximum length the frame is truncated and marked as aborted.

The MAC can drop the frames with length (in-range and out-of-range) errors by enabling MAC_ADV_CHK_CFG.INR_ERR_ENA and MAC_ADV_CHK_CFG.OOR_ERR_ENA.

There are also other checks available which can be enabled based on following configurations.

Table 4-44. DEV10G Port Module Advance Check Feature Configuration

Target::Register.Field	Description
DEV10G::MAC_ADV_CHK_CFG.EXT_EOP_CHK_ENA	Enable extended end Of packet check.

.....continued	
Target::Register.Field	**Description**
DEV10G::MAC_ADV_CHK_CFG.EXT_SOP_CHK_ENA	Enable extended start of packet check.
DEV10G::MAC_ADV_CHK_CFG.SFD_CHK_ENA	Enable start-of-frame-delimiter check.
DEV10G::MAC_ADV_CHK_CFG.PRM_SHK_CHK_DIS	Disable preamble shrink check.
DEV10G::MAC_ADV_CHK_CFG.PRM_CHK_ENA	Enable preamble check.
DEV10G::MAC_ADV_CHK_CFG.OOR_ERR_ENA	Enable out-of-range error check.
DEV10G::MAC_ADV_CHK_CFG.INR_ERR_ENA	Enable in-range error check.

4.7.1.5 Local and Remote Fault Signaling

By default the MAC is configured to detect a Link Fault indication and react by transmitting the appropriate sequence ordered set. The feature can be disabled completely using DEV10G::MAC_LFS_CFG.LFS_MODE_ENA. It is also possible to configure MAC into unidirectional mode according to IEEE 802.3, Clause 66, where frames are transmitted while receiving link fault indication but interleaved by appropriate ordered sets in between (DEV10G::MAC_LFS_CFG.LFS_UNIDIR_ENA).

4.7.2 PCS10G

This section describes the Physical Coding Sublayer (PCS) blocks included in the DEV10G port module. Depending on the DEV10G mode configuration, the corresponding PCS block must be selected and configured accordingly. Status information such as link status is available per PCS block and the right register bit, depending on mode must be checked. Interrupt source signal are selected from the current selected PCS block depending on mode selection.

Note:
Each DEV10G port module must only enable one PCS block at the same time.

The applicable registers for DEV10G PCS configuration and status can be found in the following tables.

Table 4-45. DEV10G PCS Configuration Registers

Register	Description	Replication
PCS_CFG	XFI-PCS configuration	Per PCS
PCS_SD_CFG	XFI-PCS signal detect configuration	Per PCS

Table 4-46. DEV10G PCS Status Registers

Register	Description	Replication
PCS_STATUS	XFI-PCS status	Per PCS
TX_ERRBLK_CNT	XFI-PCS counter for number of times Transmit FSM enters TX_E state.	Per PCS
TX_CHARERR_CNT	XFI-PCS counter for number of invalid control characters transmitter detected from MAC	Per PCS
RX_BER_CNT	XFI-PCS counter for number of times receive BER FSM enters BER_BAD_SH state (ber_count).	Per PCS

.....continued		
Register	**Description**	**Replication**
RX_ERRBLK_CNT	XFI-PCS counter for number of times receive FSM enters RX_E state.	Per PCS
RX_CHARERR_CNT	XFI-PCS counter for receive number of invalid control characters after decoding.	Per PCS
RX_OSET_FIFO_STAT	XFI-PCS receive ordered set FIFO status.	Per PCS
RX_OSET_FIFO_DATA	XFI-PCS receive ordered set FIFO data.	Per PCS
RX_FSET_FIFO_STAT	XFI-PCS receive F-Sig FIFO status.	Per PCS
RX_FSET_FIFO_DATA	XFI-PCS receive F-Sig FIFO data.	Per PCS

4.7.2.1 Backplane Auto-Negotiation

Backplane Auto-Negotiation must be used when normal Auto-Negotiation is disabled. For the DEV10G port module there is no specific function or setup available.

4.7.2.2 Link Surveillance

For XFI mode of operation, current link status can be observed through PCS_10GBR::PCS_STATUS as mentioned in the following section.

The XFI PCS must achieve block lock (PCS10G_BR::PCS_STATUS.RX_BLOCK_LOCK) to be able to receive correct data. This block lock is achieved when PCS detects a 66-bit boundary. Whenever the block lock is lost or achieved, a sticky bit is set (PCS10G_BR::PCS_INTR_STAT.LOCK_CHANGED_STICKY). Whenever a large number of false sync headers is detected HI_BER (PCS10G_BR::PCS_STATUS.RX_HI_BER) is asserted and a sticky bit is set (PCS10G_BR::PCS_INTR_STAT.RX_HI_BER_STICKY). If a local fault or remote fault is received then it is flagged in PCS10G_BR::PCS_INTR_STAT.RX_OSET_STICKY and the sequence ordered set received can be read from PCS10G_BR::RX_OSET_FIFO_DATA.RX_OSET_FIFO_DATA. PCS10G_BR::RX_FSET_FIFO_STAT can be used to know how many ordered sets are captured and FIFO full status.

4.7.2.3 Signal Detect

The PCS can be enabled to react to loss of signal through signal detect (x_SD_CFG.SD_ENA). Upon loss of signal, the PCS Rx state machine is restarted, and frame reception stops. If signal detect is disabled, no action is taken upon loss of signal. The polarity of signal detect is configurable by x_SD_CFG.SD_POL, where value of this bit indicates the active ‘Signal Detect’ state of the selected input line.

The source of signal detect is selected in x_SD_CFG.SD_SEL to either the SERDES PMA or the PMD receiver. If the SERDES PMA is used as source, the SERDES macro provides the signal detect. If the PMD receiver is used as source, signal detect is sampled externally through one of the GPIO pins on the device. For more information about the configuration of the GPIOs and signal detect, see 5.10.8 GPIO Controller.

4.7.2.4 Tx Loopback

For debug purposes there is a loop back path defined within all PCS blocks, which loops back Tx data path into the Rx data path. This is enabled through PCS_10GBR::PCS_CFG.PMA_LOOPBACK_ENA for PCS XFI.

4.7.2.5 Test Patterns

The XFI PCS applicable registers are listed in following table.

Table 4-47. DEV10G Base-R PCS Test Pattern Configuration Registers

Target::Register.Field	Description	Replication
PCS10G_BR::PCS_CFG.TX_TEST_MODE	Enables test pattern mode in XFI PCS transmitter.	Per PCS

.....continued

Target::Register.Field	Description	Replication
PCS10G_BR::PCS_CFG.TX_REST_MODE	Enables test pattern mode in XFI PCS receiver.	Per PCS
PCS10G_BR::TEST_CFG	Test pattern configuration register when test mode is enabled for XFI PCS	Per PCS
PCS10G_BR::TX_SEEDA_MSB.TX_SEEDA_MSB	Most significant 26 bits of seed A used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_SEEDA_LSB.TX_SEEDA_LSB	Least significant 32 bits of seed A used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_SEEDB_MSB.TX_SEEDB_MSB	Most significant 26 bits of seed B used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_SEEDB_LSB.TX_SEEDB_LSB	Least significant 32 bits of seed B used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_DATAPAT_MSB.TX_DATAPAT_MSB	Most significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for transmitter of XFI PCS.	Per PCS
PCS10G_BR::TX_DATAPAT_LSB.TX_DATAPAT_LSB	Least significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for transmitter of XFI PCS.	Per PCS
PCS10G_BR::RX_DATAPAT_MSB.RX_DATAPAT_MSB	Most significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for receiver of XFI PCS.	Per PCS
PCS10G_BR::RX_DATAPAT_LSB.RX_DATAPAT_LSB	Least significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for receiver of XFI PCS.	Per PCS
PCS10G_BR::PCS_STATUS.TESTPAT_MATCH	When in test pattern check mode, this bit will read 1 if the test pattern checker detects a match. When 0, the test pattern does not match. The test pattern error counts should still be used along with this register bit to determine proper test match status. The bit will read back 1 only when the test pattern is matching. This may happen even while test pattern errors are counted on other clock cycles.	Per PCS
PCS10G_BR::TEST_ERR_CNT.TEST_ERR_CNT	Count of detected test pattern errors in Rx test pattern checker. Write 0 to clear.	Per PCS

By setting register bits PCS10G_BR::PCS_CFG.TX_TEST_MODE and PCS10G_BR::PCS_CFG.RX_TEST_MODE to 1, test pattern mode is enabled in XFI PCS. Below test patterns are supported as per IEEE Std 802.3, clause 49.2.8 and 49.2.12.

- Square wave
- Pseudo random
- PRBS31
- User defined

User defined mode is a slightly modified test pattern in which scrambler will not be initialized with any seed value and so there will not be any errors after every 128-block window.

Different types of test pattern generation can be chosen by configuring PCS10G_BR::TEST_CFG.TX_TESTPAT_SEL and that of checkers by configuring PCS10G_BR::TEST_CFG.RX_TESTPAT_SEL. For square wave test pattern generation, period can be configured using PCS10G_BR::TEST_CFG.TX_SQPW_4B. For pseudo-random test pattern inversion of seeds and data is enabled by PCS10G_BR::TEST_CFG.TX_DSBL_INV and PCS10G_BR::TEST_CFG.RX_DSBL_INV.

4.7.2.6 KR FEC

XFI PCS also supports KR FEC. Applicable registers can be found in following table.

Table 4-48. DEV10G KR FEC Configuration and Status Registers

Register	Description	Replication
KR_FEC_CFG	KR FEC configuration register	Per PCS
KR_FEC_STATUS	KR FEC Status	Per PCS
KR_FEC_STICKY	KR FEC sticky bits	Per PCS
KR_FEC_CORRECTED	KR FEC corrected error blocks counter	Per PCS
KR_FEC_UNCORRECTED	KR FEC uncorrected error blocks counter	Per PCS

To enable FEC, PCS10G_BR::KR_FEC_CFG.FEC_ENA must be set. In addition, data bits must be flipped at KR FEC interface with PCS. This is achieved by setting both PCS10G_BR::KR_FEC_CFG.TX_DATA_FLIP and PCS10G_BR::KR_FEC_CFG.RX_DATA_FLIP to 1.

For enabling FEC error indication in the PCS register bit, set PCS10G_BR::KR_FEC_CFG.ENABLE_ERROR_INDICATION to 1. This ensures that XFI PCS decodes the block correctly, and hence correct data is presented to MAC. When FEC looses frame lock, it is reflected in the PCS10G_BR::KR_FEC_STICKY.FEC_FRAME_LOCK_STICKY sticky bit.

There are two counters in FEC for counting corrected and uncorrected error blocks. These are cleared by writing a 1 to PCS10G_BR::KR_FEC_CFG.RESET_MONITOR_COUNTERS, followed by a 0.

When the counters cross a threshold value configurable by PCS10G_BR::FIXED_ERROR_COUNT_THRESHOLD and PCS10G_BR::UNFIXABLE_ERROR_COUNT_THRESHOLD, two corresponding sticky bits are set (PCS10G_BR::KR_FEC_STICKY.FEC_FIXED_ERROR_COUNT_STICKY and PCS10G_BR::KR_FEC_STICKY.FEC_UNFIXABLE_ERROR_COUNT_STICKY).

Note that KR FEC must be disabled when EEE is enabled.

4.7.3 USXGMII PCS

This section describes Device level functions and configuration options of the USXGMII PCS.

USXGMII PCS supports 10/100/1000/2500/5000 Mbps in full-duplex mode.

The following tables list the registers associated with the configuration of USXGMII PCS.

Table 4-49. USXGMII PCS Configuration Registers

Register	Description	Replication
USXGMII_PCS_SD_CFG.SD_POL	Signal detect polarity. The signal level on signal_detect input pin must be equal to SD_POL to indicate signal detection	Per PCS
USXGMII_PCS_SD_CFG.SD_ENA	Signal Detect Enable. 0: The Signal Detect input pin is ignored. 1: The Signal Detect input pin is used to determine if a signal is detected	Per PCS

.....continued		
Register	**Description**	**Replication**
USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK	Indicates if USXGMII PCS has reached block lock state 0: USXGMII PCS is not in block lock state 1: USXGMII PCS is in block lock state	Per PCS
DEV_RST_CTRL.USXGMII_OSET_FILTER_DIS	The USXGMII replicator block samples only the first XGMII word. The other replications are ignored. Only if they contain a Local Fault or remote Fault ordered set the next data will be overwritten. This bit will disable overwrite wit ordered sets. 0: Non-SampleId ordered sets will overwrite the following XGMII words 1: Overwrite of XGMII words with ordered sets is disabled.	Per PCS

Table 4-50. USXGMII PCS Rate Adapt Configuration Registers

Register	Description	Replication
USXGMII_TX_RADAPT_CFG. TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG. TX_RADAPT_ADD_LVL	Set the add threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG. TX_RADAPT_DROP_LVL	Set the drop threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG. TX_LF_GEN_DIS	Disable TX local fault generation when no alignment has been reached	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.s TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS

For 10G-DXGMII modes, the USXGMII Rate adaptation register fields should be set to the following values.

USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL = 4

USXGMII_TX_RADAPT_CFG.TX_RADAPT_DROP_LVL = 8

USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG = 1

4.7.3.1 Signal Detect

The USXGMII_PCS can be enabled to react to loss of signal through signal detect (USXGMII_PCS_SD_CFG.SD_ENA). The polarity of signal detect is configurable USXGMII_PCS_SD_CFG.SD_ENA.SD_POL.

USXGMII_PCS_SD_CFG.SD_ENA is used to determine if a signal is detected.

0: The Signal Detect input pin is ignored.

1: The Signal Detect input pin is used to determine if a signal is detected

If signal detect status is not detected, then USXGMII ANEG is restarted.

USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK shows USXGMII PCS has reached block lock state.

0: USXGMII PCS is not in block lock state

1: USXGMII PCS is in block lock state

4.7.3.2 USXGMII ANEG - Auto Negotiation

The DEV10G USXGMII Auto-negotiation is similar to DEV2G5 USXGMII ANEG. Refer to 4.5.3.2 USXGMII ANEG - Auto Negotiation.

4.7.4 Port Statistics

The DEV10G port module contains a set of port statistics. Packets counters are 32 bits wide and byte counters are 40 bits wide. The following table lists the counters.

Table 4-51. DEV10G Statistics

Register	Description
RX_SYMBOL_ERR_CNT	Rx Symbol Carrier Error Counter
RX_PAUSE_CNT	Rx Pause Frame Counter
RX_UNSUP_OPCODE_CNT	Rx Control Frame Counter
RX_UC_CNT	Rx Unicast Frame Counter
RX_MC_CNT	Rx Multicast Frame Counter
RX_BC_CNT	Rx Broadcast Frame Counter
RX_CRC_ERR_CNT	Rx CRC Error Counter
RX_UNDERSIZE_CNT	Rx Undersize Counter (valid frame format)
RX_FRAGMENTS_CNT	Rx Undersize Counter (CRC error)
RX_IN_RANGE_LEN_ERR_CNT	Rx In-range Length Error Counter
RX_OUT_OF_RANGE_LEN_ERR_CNT	Rx Out-Of-Range Length Error Counter
RX_OVERSIZE_CNT	Rx Oversize Counter (valid frame format)
RX_JABBERS_CNT	Rx Jabbers Counter
RX_SIZE64_CNT	Rx 64 Byte Frame Counter
RX_SIZE65TO127_CNT	Rx 65-127 Byte Frame Counter
RX_SIZE128TO255_CNT	Rx 128-255 Byte Frame Counter
RX_SIZE256TO511_CNT	Rx 256-511 Byte Frame Counter

.....continued

Register	Description
RX_SIZE512TO1023_CNT	Rx 512-1023 Byte Frame Counter
RX_SIZE1024TO1518_CNT	Rx 1024-1518 Byte Frame Counter
RX_SIZE1519TOMAX_CNT	Rx 1519 To Max. Length Byte Frame Counter
RX_IPG_SHRINK_CNT	Rx Inter Packet Gap Shrink Counter
TX_PAUSE_CNT	Tx Pause Frame Counter
TX_UC_CNT	Tx Unicast Frame Counter
TX_MC_CNT	Tx Multicast Frame Counter
TX_BC_CNT	Tx Broadcast Frame Counter
TX_SIZE64_CNT	Tx 64 Byte Frame Counter
TX_SIZE65TO127_CNT	Tx 65-127 Byte Frame Counter
TX_SIZE128TO255_CNT	Tx 128-255 Byte Frame Counter
TX_SIZE256TO511_CNT	Tx 256-511 Byte Frame Counter
TX_SIZE512TO1023_CNT	Tx 512-1023 Byte Frame Counter
TX_SIZE1024TO1518_CNT	Tx 1024-1518 Byte Frame Counter
TX_SIZE1519TOMAX_CNT	Tx 1519 To Max. Length Byte Frame Counter
RX_ALIGNMENT_LOST_CNT	Counter to track the dribble-nibble (extra nibble) errors in frames.
RX_TAGGED_FRMS_CNT	Counts frames that are tagged (C-Tagged or S-Tagged).
RX_UNTAGGED_FRMS_CNT	Counts frames that are Not tagged (neither C-Tagged nor S-Tagged).
TX_TAGGED_FRMS_CNT	Counts frames that are tagged (C-Tagged or S-Tagged).
TX_UNTAGGED_FRMS_CNT	Counts frames that are Not tagged (neither C-Tagged nor S-Tagged).
RX_HIH_CKSM_ERR_CNT	Rx HiH Checksum Error Counter
RX_XGMII_PROT_ERR_CNT	Rx XGMII Protocol Error Counter
PMAC_RX_SYMBOL_ERR_CNT	PMAC Rx Symbol Carrier Error Counter
PMAC_RX_PAUSE_CNT	PMAC Rx Pause Frame Counter
PMAC_RX_UNSUP_OPCODE_CNT	PMAC Rx Control Frame Counter
PMAC_RX_UC_CNT	PMAC Rx Unicast Frame Counter
PMAC_RX_MC_CNT	PMAC Rx Multicast Frame Counter
PMAC_RX_BC_CNT	PMAC Rx Broadcast Frame Counter
PMAC_RX_CRC_ERR_CNT	PMAC Rx CRC Error Counter

.....continued

Register	Description
PMAC_RX_UNDERSIZE_CNT	PMAC Rx Undersize Counter (valid frame format)
PMAC_RX_FRAGMENTS_CNT	PMAC Rx Undersize Counter (CRC error)
PMAC_RX_IN_RANGE_LEN_ERR_CNT	PMAC Rx In-range Length Error Counter
PMAC_RX_OUT_OF_RANGE_LEN_ERR_CNT	PMAC Rx Out-Of-Range Length Error Counter
PMAC_RX_OVERSIZE_CNT	PMAC Rx Oversize Counter (valid frame format)
PMAC_RX_JABBERS_CNT	PMAC Rx Jabbers Counter
PMAC_RX_SIZE64_CNT	PMAC Rx 64 Byte Frame Counter
PMAC_RX_SIZE65TO127_CNT	PMAC Rx 65-127 Byte Frame Counter
PMAC_RX_SIZE128TO255_CNT	PMAC Rx 128-255 Byte Frame Counter
PMAC_RX_SIZE256TO511_CNT	PMAC Rx 256-511 Byte Frame Counter
PMAC_RX_SIZE512TO1023_CNT	PMAC Rx 512-1023 Byte Frame Counter
PMAC_RX_SIZE1024TO1518_CNT	PMAC Rx 1024-1518 Byte Frame Counter
PMAC_RX_SIZE1519TOMAX_CNT	PMAC Rx 1519 To Max. Length Byte Frame Counter
PMAC_TX_PAUSE_CNT	PMAC Tx Pause Frame Counter
PMAC_TX_UC_CNT	PMAC Tx Unicast Frame Counter
PMAC_TX_MC_CNT	PMAC Tx Multicast Frame Counter
PMAC_TX_BC_CNT	PMAC Tx Broadcast Frame Counter
PMAC_TX_SIZE64_CNT	PMAC Tx 64 Byte Frame Counter
PMAC_TX_SIZE65TO127_CNT	PMAC Tx 65-127 Byte Frame Counter
PMAC_TX_SIZE128TO255_CNT	PMAC Tx 128-255 Byte Frame Counter
PMAC_TX_SIZE256TO511_CNT	PMAC Tx 256-511 Byte Frame Counter
PMAC_TX_SIZE512TO1023_CNT	PMAC Tx 512-1023 Byte Frame Counter
PMAC_TX_SIZE1024TO1518_CNT	PMAC Tx 1024-1518 Byte Frame Counter
PMAC_TX_SIZE1519TOMAX_CNT	PMAC Tx 1519 To Max. Length Byte Frame Counter
PMAC_RX_ALIGNMENT_LOST_CNT	PMAC Counter to track the dribble-nibble (extra nibble) errors in frames.
MM_RX_ASSEMBLY_ERR_CNT	MAC MERGE - The number of Pframes received with aseembly error
MM_RX_SMD_ERR_CNT	MAC MERGE - The number of Pframe fragaments rejected die to unknown SMD values or arrving with SMD-C when no pframe in progress
MM_RX_ASSEMBLY_OK_CNT	MAC MERGE - The number of Pframes received whitch are successfully reassembled

.....continued	
Register	**Description**
MM_RX_MERGE_FRAG_CNT	MAC MERGE - The number of Pframes received whitch are mPackets (Fragments) due to preemption
MM_TX_PFRAGMENT_CNT	MAC MERGE - The number of Pframe fragments sent when preemption enabled
PMAC_RX_HIH_CKSM_ERR_CNT	PMAC Rx HiH Checksum Error Counter
PMAC_RX_XGMII_PROT_ERR_CNT	PMAC Rx XGMII Protocol Error Counter
RX_IN_BYTES_CNT	The number of bytes received (good, bad, and framing).
RX_IN_BYTES_MSB_CNT	The number of bytes received (good, bad, and framing) - MSBs only
RX_OK_BYTES_CNT	The number of received bytes in good frames.
RX_OK_BYTES_MSB_CNT	The number of received bytes in good frames - MSBs only.
RX_BAD_BYTES_CNT	The number of received bytes in bad frames.
RX_BAD_BYTES_MSB_CNT	The number of received bytes in bad frames - MSBs only
TX_OUT_BYTES_CNT	The number of bytes transmitted (good, bad and framing)
TX_OUT_BYTES_MSB_CNT	The number of bytes transmitted (good, bad, framing) - MSBs only
TX_OK_BYTES_CNT	The number of bytes transmitted successfully.
TX_OK_BYTES_MSB_CNT	The number of transmitted bytes transmitted successfully - MSBs only
PMAC_RX_OK_BYTES_CNT	PMAC - The number of received bytes in good frames.
PMAC_RX_OK_BYTES_MSB_CNT	PMAC - The number of received bytes in good frames - MSBs only.
PMAC_RX_BAD_BYTES_CNT	PMAC - The number of received bytes in bad frames
PMAC_RX_BAD_BYTES_MSB_CNT	PMAC - The number of received bytes in bad frames - MSBs only
PMAC_TX_OK_BYTES_CNT	PMAC - The number of bytes transmitted successfully.
PMAC_TX_OK_BYTES_MSB_CNT	PMAC - The number of transmitted bytes transmitted successfully - MSBs only.

The counters are writable. In particular, the counters are cleared by writing a 0 to each counter.

4.8 DEV25G Port Module

The DEV25G port module can be used in the following full-duplex operation modes.

Table 4-52. DEV25G Operation Modes – 10G speed

Mode	10GBASE-R
Lane speed	10.3125
No. of Lanes	1
Encoding	64b/66b
PCS used	CLAUSE 49
IPG shrink (average)	No
Preamble shrink	No
Data bandwidth	10 Gbps
Payload bandwidth (compensated for stacking header)	10 Gbps

Table 4-53. DEV25G Operation Modes – 5G speed

Mode	5GBASE-R
Lane speed	5.156
No. of Lanes	1
Encoding	64b/66b
PCS used	CLAUSE 129
IPG shrink (average)	No
Preamble shrink	No
Data bandwidth	5 Gbps
Payload bandwidth (compensated for stacking header)	5 Gbps

4.8.1 MAC

This section describes the high level functions and the configuration options of the MAC that is used in the DEV25G port modules.

The applicable registers are listed in the following table.

Table 4-54. DEV25G MAC Configuration Registers

Register	Description	Replication
MAC_ENA_CFG	Enabling of Rx and Tx data paths	Per port
MAC_MODE_CFG	Port Mode Configuration	Per port
MAC_MAXLEN_CFG	Maximum length configuration	Per port
MAC_NUM_TAGS_CFG	Number of tags configuration	Per port
MAC_TAGS_CFG	VLAN/Service tag configuration	3 per port

.....continued		
Register	**Description**	**Replication**
MAC_ADV_CHK_CFG	Advance Check Feature configuration	Per port
MAC_LFS_CFG	Link Fault Signaling configuration	Per port
MAC_LB_CFG	Loopback configuration	Per port
MAC_STICKY	Sticky bit recordings	Per port

4.8.1.1 Clock and Reset

There are a number of synchronous resets in the DEV25G port module. All of the resets can be set and cleared simultaneously. By default, all blocks are in the reset state (reset active). The following table lists all the available resets.

Table 4-55. DEV25G Resets

Target::Register.Field	Description
DEV25G::DEV_RST_CTRL.MAC_RX_RST	Reset MAC receiver
DEV25G::DEV_RST_CTRL.MAC_TX_RST	Reset MAC transmitter
DEV25G::DEV_RST_CTRL.PCS_RX_RST	Reset PCS receiver
DEV25G::DEV_RST_CTRL.PCS_TX_RST	Reset PCS transmitter

When changing the MAC configuration, the port must go through a reset cycle. This is done by writing the register DEV25G::DEV_RST_CTRL twice. On first write all reset bits must be set to active (0x1). On the second write resets bits are cleared (0x0). Non-reset bits in DEV25G::DEV_RST_CTRL must keep their value for both writes.

4.8.1.2 Interrupts

There are 5 interrupt sources defined for DEV25G port module, which can be found below. For detailed descriptions of the general interrupt handling, see 5.4.2 VCore CPU Interrupt Controller.

Table 4-56. DEV25G Interrupt Sources

Target::Register.Field	Description
DEV25G::INTR. PCS_BR_INTR	One of the Base-R PCS interrupt indications is active
DEV25G::INTR.TX_LPI_INTR	TX Low Power Idle mode has changed
DEV25G::INTR.RX_LPI_INTR	RX Low Power Idle mode has changed
DEV25G::INTR.LINK_UP_INTR	Link status is up
DEV25G::INTR. LINK_DWN_INTR	Link status is down

All the sticky events in XFI PCS are capable of generating an interrupt if their corresponding interrupt mask is set to 1. Below are the interrupt registers that are used for interrupt generation if corresponding event bit is set from 10G Base-R PCS (or XFI PCS).

Table 4-57. Dev25G Base-R PCS Interrupt Sources

Register	Sticky Bits that can be Enabled for Interrupt Generation	Replication
PCS_INTR_MASK	XFI PCS sticky bits	Per PCS
KR_FEC_STICKY_MASK	KR FEC sticky bits	Per PCS
EEE_INTR_MASK	XFI PCS EEE sticky bits	Per PCS

4.8.1.3 Port Mode Configuration

The MAC provides a number of handles for configuring the port mode:

Table 4-58. DEV25G Port Mode Configurations

Target::Register.Field	Description
DEV25G::DEV_RST_CTRL.SPEED_SEL	Configuration of MAC and PCS Rx/Tx clock frequencies 0: MAC works at 10 Gbps Unused values are reserved.
DEV25G::MAC_ENA_CFG.RX_ENA	Enable MAC Receiver Module 0: Disabled 1: Enabled
DEV25G::MAC_ENA_CFG.TX_ENA	Enable MAC Transmitter Module 0: Disabled 1: Enabled
DEV25G::MAC_MODE_CFG.TUNNEL_PAUSE_FRAMES	Enable tunnelling of Pause Frames on a link 0: Pause frames are expected to be in Link Pause Frames
DEV25G::MAC_MODE_CFG.MAC_PREAMBLE_CFG	Preamble format configuration 0: Standard Preamble
DEV25G::MAC_MODE_CFG.MAC_IPG_CFG	IPG format configuration 0: Normal IPG
DEV25G::MAC_MODE_CFG. HIH_CRC_CHECK	Enable check for HiH checksum 0: MAC10G should disregard the HIH checksum

4.8.1.4 Type/Length Check

The MAC supports frame lengths of up to 14,000 bytes. The maximum frame accepted by the MAC is configurable and defined in DEV25G::MAC_MAXLEN_CFG.MAX_LEN. Maximum length is automatically adjusted if one or more tags are included in the current frame. The MAC allows 1/2/3 tagged frames to be 4/8/12 bytes longer respectively, than the specified maximum length. DEV25G::MAC_MAXLEN_CFG.MAX_LEN_TAG_CHK configures whether the max Length check takes the number of tags into consideration.

Number of tags can be selected by DEV25G::MAC_NUM_TAGS_CFG.NUM_TAGS. By default, EtherType 0x8100 is identified a VLAN tag. In addition, three custom EtherTypes can be configured by MAC_TAGS_CFG.TAG_ID.

If a received frame exceeds the maximum length the frame is truncated and marked as aborted.

The MAC can drop the frames with length (in-range and out-of-range) errors by enabling MAC_ADV_CHK_CFG.INR_ERR_ENA and MAC_ADV_CHK_CFG.OOR_ERR_ENA.

There are also other checks available which can be enabled based on following configurations.

Table 4-59. DEV25G Port Module Advance Check Feature Configuration

Target::Register.Field	Description
DEV25G::MAC_ADV_CHK_CFG.EXT_EOP_CHK_ENA	Enable extended end of packet check.
DEV25G::MAC_ADV_CHK_CFG.EXT_SOP_CHK_ENA	Enable extended start of packet check.
DEV25G::MAC_ADV_CHK_CFG.SFD_CHK_ENA	Enable start-of-frame-delimiter check.
DEV25G::MAC_ADV_CHK_CFG.PRM_SHK_CHK_DIS	Disable preamble shrink check.
DEV25G::MAC_ADV_CHK_CFG.PRM_CHK_ENA	Enable preamble check.
DEV25G::MAC_ADV_CHK_CFG.OOR_ERR_ENA	Enable out-of-range error check.
DEV25G::MAC_ADV_CHK_CFG.INR_ERR_ENA	Enable in-range error check.

4.8.1.5 Local and Remote Fault Signaling

By default the MAC is configured to detect a Link Fault indication and react by transmitting the appropriate sequence ordered set. The feature can be disabled completely using DEV25G::MAC_LFS_CFG.LFS_MODE_ENA. It is also possible to configure MAC into unidirectional mode according to IEEE 802.3, Clause 66, where frames are transmitted while receiving link fault indication but interleaved by appropriate ordered sets in between (DEV25G::MAC_LFS_CFG.LFS_UNIDIR_ENA).

4.8.2 PCS10G

This section describes the PCS10G block included in the DEV25G port module. Depending on the DEV25G mode configuration, the corresponding PCS block must be selected and configured accordingly. Status information such as link status is available per PCS block and the right register bit, depending on mode must be checked. Interrupt source signal are selected from the current selected PCS block depending on mode selection.

Applicable registers for DEV25G PCS10G configuration and status can be found in the following tables.

Table 4-60. DEV25G PCS10G Configuration Registers

Register	Description	Replication
PCS_CFG	XFI-PCS configuration	Per PCS
PCS_SD_CFG	XFI-PCS signal detect configuration	Per PCS

Table 4-61. DEV25G PCS10G Status Registers

Register	Description	Replication
PCS_STATUS	XFI-PCS Status	Per PCS
TX_ERRBLK_CNT	XFI-PCS Counter for number of times Transmit FSM enters TX_E state.	Per PCS
TX_CHARERR_CNT	XFI-PCS Counter for number of invalid control characters transmitter detected from MAC	Per PCS
RX_BER_CNT	XFI-PCS Counter for number of times Receive BER FSM enters BER_BAD_SH state (ber_count).	Per PCS
RX_ERRBLK_CNT	XFI-PCS Counter for number of times Receive FSM enters RX_E state.	Per PCS

.....continued		
Register	**Description**	**Replication**
RX_CHARERR_CNT	XFI-PCS Counter for Receive number of invalid control characters after decoding.	Per PCS
RX_OSET_FIFO_STAT	XFI-PCS Receive Ordered set FIFO status.	Per PCS
RX_OSET_FIFO_DATA	XFI-PCS Receive Ordered set FIFO data.	Per PCS
RX_FSET_FIFO_STAT	XFI-PCS Receive F-Sig FIFO status.	Per PCS
RX_FSET_FIFO_DATA	XFI-PCS Receive F-Sig FIFO data.	Per PCS

4.8.2.1 Backplane Auto-Negotiation

Backplane Auto-Negotiation must be used when normal Auto-Negotiation is disabled. For the DEV25G port module there is no specific function or setup available.

4.8.2.2 Link Surveillance

For XFI mode of operation, current link status can be observed through PCS_10GBR::PCS_STATUS as mentioned the following section.

The XFI PCS must achieve block lock (PCS10G_BR::PCS_STATUS.RX_BLOCK_LOCK) to be able to receive correct data. This block lock is achieved when PCS detects a 66-bit boundary. Whenever the block lock is lost or achieved, a sticky bit is set (PCS10G_BR::PCS_INTR_STAT.LOCK_CHANGED_STICKY). Whenever a large number of false sync headers is detected HI_BER (PCS10G_BR::PCS_STATUS.RX_HI_BER) is asserted and a sticky bit is set (PCS10G_BR::PCS_INTR_STAT.RX_HI_BER_STICKY). If a local fault or remote fault is received then it is flagged in PCS10G_BR::PCS_INTR_STAT.RX_OSET_STICKY and the sequence ordered set received can be read from PCS10G_BR::RX_OSET_FIFO_DATA.RX_OSET_FIFO_DATA. PCS10G_BR::RX_FSET_FIFO_STAT can be used to know how many ordered sets are captured and FIFO full status.

4.8.2.3 Signal Detect

The PCS can be enabled to react to loss of signal through signal detect (x_SD_CFG.SD_ENA). Upon loss of signal, the PCS Rx state machine is restarted, and frame reception stops. If signal detect is disabled, no action is taken upon loss of signal. The polarity of signal detect is configurable by x_SD_CFG.SD_POL, where value of this bit indicates the active ‘Signal Detect’ state of the selected input line.

The source of signal detect is selected in x_SD_CFG.SD_SEL to either the SERDES PMA or the PMD receiver. If the SERDES PMA is used as source, the SERDES macro provides the signal detect. If the PMD receiver is used as source, signal detect is sampled externally through one of the GPIO pins on the device. For more information about the configuration of the GPIOs and signal detect, 5.10.8 GPIO Controller.

4.8.2.4 Tx Loopback

For debug purposes there is a loop back path defined within all PCS blocks, which loops back Tx data path into the Rx data path. This is enabled through PCS_10GBR::PCS_CFG.PMA_LOOPBACK_ENA for PCS XFI.

4.8.2.5 Test Patterns

For XFI PCS, the applicable registers are listed in the following table.

Table 4-62. DEV25G 10GBase-R PCS Test Pattern Configuration Registers

Target::Register.Field	Description	Replication
PCS10G_BR::PCS_CFG.TX_TEST_MODE	Enables test pattern mode in XFI PCS transmitter.	Per PCS

.....continued

Target::Register.Field	Description	Replication
PCS10G_BR::PCS_CFG.TX_REST_MODE	Enables test pattern mode in XFI PCS receiver.	Per PCS
PCS10G_BR::TEST_CFG	Test pattern configuration register when test mode is enabled for XFI PCS	Per PCS
PCS10G_BR::TX_SEEDA_MSB.TX_SEEDA_MSB	Most significant 26 bits of seed A used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_SEEDA_LSB.TX_SEEDA_LSB	Least significant 32 bits of seed A used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_SEEDB_MSB.TX_SEEDB_MSB	Most significant 26 bits of seed B used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_SEEDB_LSB.TX_SEEDB_LSB	Least significant 32 bits of seed B used to initialization XFI PCS scrambler in test mode.	Per PCS
PCS10G_BR::TX_DATAPAT_MSB.TX_DATAPAT_MSB	Most significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for transmitter of XFI PCS.	Per PCS
PCS10G_BR::TX_DATAPAT_LSB.TX_DATAPAT_LSB	Least significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for transmitter of XFI PCS.	Per PCS
PCS10G_BR::RX_DATAPAT_MSB.RX_DATAPAT_MSB	Most significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for receiver of XFI PCS.	Per PCS
PCS10G_BR::RX_DATAPAT_LSB.RX_DATAPAT_LSB	Least significant 32 bits of 64-bit data pattern used in pseudo-random and user-defined test pattern mode for receiver of XFI PCS.	Per PCS
PCS10G_BR::PCS_STATUS.TESTPAT_MATCH	When in test pattern check mode, this bit will read 1 if the test pattern checker detects a match. When 0, the test pattern does not match. The test pattern error counts should still be used along with this register bit to determine proper test match status. The bit will read back 1 only when the test pattern is matching. This may happen even while test pattern errors are counted on other clock cycles.	Per PCS
PCS10G_BR::TEST_ERR_CNT.TEST_ERR_CNT	Count of detected test pattern errors in Rx test pattern checker. Write 0 to clear.	Per PCS

By setting register bits PCS10G_BR::PCS_CFG.TX_TEST_MODE and PCS10G_BR::PCS_CFG.RX_TEST_MODE to 1, test pattern mode is enabled in XFI PCS. Below test patterns are supported as per IEEE Std 802.3, clause 49.2.8 and 49.2.12.

- Square wave
- Pseudo random
- PRBS31
- User defined

User defined mode is a slightly modified test pattern in which scrambler will not be initialized with any seed value and so there will not be any errors after every 128-block window.

Different types of test pattern generation can be chosen by configuring PCS10G_BR::TEST_CFG.TX_TESTPAT_SEL and that of checkers by configuring PCS10G_BR::TEST_CFG.RX_TESTPAT_SEL. For square wave test pattern generation, period can be configured using PCS10G_BR::TEST_CFG.TX_SQPW_4B. For pseudo-random test pattern inversion of seeds and data is enabled by PCS10G_BR::TEST_CFG.TX_DSBL_INV and PCS10G_BR::TEST_CFG.RX_DSBL_INV.

4.8.2.6 KR FEC

XFI PCS also supports KR FEC. Applicable registers can be found in following table.

Table 4-63. DEV25G 10G-KR FEC Configuration and Status Registers

Register	Description	Replication
KR_FEC_CFG	KR FEC configuration register	Per PCS
KR_FEC_STATUS	KR FEC Status	Per PCS
KR_FEC_STICKY	KR FEC sticky bits	Per PCS
KR_FEC_CORRECTED	KR FEC corrected error blocks counter	Per PCS
KR_FEC_UNCORRECTED	KR FEC uncorrected error blocks counter	Per PCS

To enable FEC, PCS10G_BR::KR_FEC_CFG.FEC_ENA must be set. In addition, data bits must be flipped at KR FEC interface with PCS. This is achieved by setting both PCS10G_BR::KR_FEC_CFG.TX_DATA_FLIP and PCS10G_BR::KR_FEC_CFG.RX_DATA_FLIP to 1.

For enabling FEC error indication in the PCS register bit, set PCS10G_BR::KR_FEC_CFG.ENABLE_ERROR_INDICATION to 1. This ensures that XFI PCS decodes the block correctly, and hence correct data is presented to MAC. When FEC looses frame lock, it is reflected in the PCS10G_BR::KR_FEC_STICKY.FEC_FRAME_LOCK_STICKY sticky bit.

There are two counters in FEC for counting corrected and uncorrected error blocks. These are cleared by writing a 1 to PCS10G_BR::KR_FEC_CFG.RESET_MONITOR_COUNTERS, followed by a 0.

When the counters cross a threshold value configurable by PCS10G_BR::FIXED_ERROR_COUNT_THRESHOLD and PCS10G_BR::UNFIXABLE_ERROR_COUNT_THRESHOLD, two corresponding sticky bits are set (PCS10G_BR::KR_FEC_STICKY.FEC_FIXED_ERROR_COUNT_STICKY and PCS10G_BR::KR_FEC_STICKY.FEC_UNFIXABLE_ERROR_COUNT_STICKY).

Note that KR FEC must be disabled when EEE is enabled.

4.8.3 USXGMII PCS

This section describes Device level functions and configuration options of the USXGMII PCS.

USXGMII PCS supports 10/100/1000/2500/5000 Mbps in full-duplex mode.

The following tables list the registers associated with configuration of USXGMII PCS.

Table 4-64. USXGMII PCS Configuration Registers

Register	Description	Replication
USXGMII_PCS_SD_CFG.SD_POL	Signal detect polarity. The signal level on signal_detect input pin must be equal to SD_POL to indicate signal detection	Per PCS
USXGMII_PCS_SD_CFG.SD_ENA	Signal Detect Enable. 0: The Signal Detect input pin is ignored. 1: The Signal Detect input pin is used to determine if a signal is detected	Per PCS

.....continued		
Register	**Description**	**Replication**
USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK	Indicates if USXGMII PCS has reached block lock state 0: USXGMII PCS is not in block lock state 1: USXGMII PCS is in block lock state	Per PCS
DEV_RST_CTRL.USXGMII_OSET_FILTER_DIS	The USXGMII replicator block samples only the first XGMII word. The other replications are ignored. Only if they contain a Local Fault or remote Fault ordered set the next data will be overwritten. This bit will disable overwrite wit ordered sets. 0: Non-SampleId ordered sets will overwrite the following XGMII words 1: Overwrite of XGMII words with ordered sets is disabled.	Per PCS

Table 4-65. USXGMII PCS Rate Adapt Configuration Registers

Register	Description	Replication
USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL	Set the add threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG.TX_RADAPT_DROP_LVL	Set the drop threshold in the FIFO. Level is in 72-bit words.	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG. TX_LF_GEN_DIS	Disable TX local fault generation when no alignment has been reached	Per USXGMII PCS
USXGMII_TX_RADAPT_CFG. TX_RADAPT_MIN_IFG	Set the minimum inter-frame gap to maintain during rate adaptation when dropping transactions. Setting the value to 1 means that 1 idle or ordered set is kept after a terminate. When set to 2 or 3, that many idle or ordered set transactions are kept. Each transaction is 4 bytes.	Per USXGMII PCS

For the 10G-DXGMII modes, the USXGMII Rate adaptation register fields should be set to the following values.

- USXGMII_TX_RADAPT_CFG.TX_RADAPT_ADD_LVL = 4
- USXGMII_TX_RADAPT_CFG.TX_RADAPT_DROP_LVL = 8
- USXGMII_TX_RADAPT_CFG.TX_RADAPT_MIN_IFG = 1

4.8.3.1 Signal Detect

The USXGMII_PCS can be enabled to react to loss of signal through signal detect (USXGMII_PCS_SD_CFG.SD_ENA). The polarity of signal detect is configurable USXGMII_PCS_SD_CFG.SD_ENA.SD_POL.

USXGMII_PCS_SD_CFG.SD_ENA is used to determine if a signal is detected.

0: The Signal Detect input pin is ignored.

1: The Signal Detect input pin is used to determine if a signal is detected

If signal detect status is not detected, then USXGMII ANEG is restarted.

USXGMII_PCS_STATUS.USXGMII_BLOCK_LOCK shows USXGMII PCS has reached block lock state.

0: USXGMII PCS is not in block lock state

1: USXGMII PCS is in block lock state

4.8.3.2 USXGMII ANEG - Auto Negotiation

The DEV25G USXGMII Auto-negotiation is similar to DEV2G5 USXGMII ANEG. Refer to 4.5.3.2 USXGMII ANEG - Auto Negotiation.

4.8.4 Port Statistics

The DEV25G port module contains a set of port statistics. Packets counters are 32 bits wide and byte counters are 40 bits wide. The following table lists the counters.

Table 4-66. DEV25G Statistics

Register	Description
RX_SYMBOL_ERR_CNT	Rx Symbol Carrier Error Counter
RX_PAUSE_CNT	Rx Pause Frame Counter
RX_UNSUP_OPCODE_CNT	Rx Control Frame Counter
RX_UC_CNT	Rx Unicast Frame Counter
RX_MC_CNT	Rx Multicast Frame Counter
RX_BC_CNT	Rx Broadcast Frame Counter
RX_CRC_ERR_CNT	Rx CRC Error Counter
RX_UNDERSIZE_CNT	Rx Undersize Counter (valid frame format)
RX_FRAGMENTS_CNT	Rx Undersize Counter (CRC error)
RX_IN_RANGE_LEN_ERR_CNT	Rx In-range Length Error Counter
RX_OUT_OF_RANGE_LEN_ERR_CNT	Rx Out-Of-Range Length Error Counter
RX_OVERSIZE_CNT	Rx Oversize Counter (valid frame format)
RX_JABBERS_CNT	Rx Jabbers Counter
RX_SIZE64_CNT	Rx 64 Byte Frame Counter
RX_SIZE65TO127_CNT	Rx 65-127 Byte Frame Counter
RX_SIZE128TO255_CNT	Rx 128-255 Byte Frame Counter
RX_SIZE256TO511_CNT	Rx 256-511 Byte Frame Counter
RX_SIZE512TO1023_CNT	Rx 512-1023 Byte Frame Counter
RX_SIZE1024TO1518_CNT	Rx 1024-1518 Byte Frame Counter
RX_SIZE1519TOMAX_CNT	Rx 1519 To Max. Length Byte Frame Counter

.....continued

Register	Description
RX_IPG_SHRINK_CNT	Rx Inter Packet Gap Shrink Counter
TX_PAUSE_CNT	Tx Pause Frame Counter
TX_UC_CNT	Tx Unicast Frame Counter
TX_MC_CNT	Tx Multicast Frame Counter
TX_BC_CNT	Tx Broadcast Frame Counter
TX_SIZE64_CNT	Tx 64 Byte Frame Counter
TX_SIZE65TO127_CNT	Tx 65-127 Byte Frame Counter
TX_SIZE128TO255_CNT	Tx 128-255 Byte Frame Counter
TX_SIZE256TO511_CNT	Tx 256-511 Byte Frame Counter
TX_SIZE512TO1023_CNT	Tx 512-1023 Byte Frame Counter
TX_SIZE1024TO1518_CNT	Tx 1024-1518 Byte Frame Counter
TX_SIZE1519TOMAX_CNT	Tx 1519 To Max. Length Byte Frame Counter
RX_ALIGNMENT_LOST_CNT	Counter to track the dribble-nibble (extra nibble) errors in frames.
RX_TAGGED_FRMS_CNT	Counts frames that are tagged (C-Tagged or S-Tagged).
RX_UNTAGGED_FRMS_CNT	Counts frames that are Not tagged (neither C-Tagged nor S-Tagged).
TX_TAGGED_FRMS_CNT	Counts frames that are tagged (C-Tagged or S-Tagged).
TX_UNTAGGED_FRMS_CNT	Counts frames that are Not tagged (neither C-Tagged nor S-Tagged).
RX_HIH_CKSM_ERR_CNT	Rx HiH Checksum Error Counter
RX_XGMII_PROT_ERR_CNT	Rx XGMII Protocol Error Counter
PMAC_RX_SYMBOL_ERR_CNT	PMAC Rx Symbol Carrier Error Counter
PMAC_RX_PAUSE_CNT	PMAC Rx Pause Frame Counter
PMAC_RX_UNSUP_OPCODE_CNT	PMAC Rx Control Frame Counter
PMAC_RX_UC_CNT	PMAC Rx Unicast Frame Counter
PMAC_RX_MC_CNT	PMAC Rx Multicast Frame Counter
PMAC_RX_BC_CNT	PMAC Rx Broadcast Frame Counter
PMAC_RX_CRC_ERR_CNT	PMAC Rx CRC Error Counter
PMAC_RX_UNDERSIZE_CNT	PMAC Rx Undersize Counter (valid frame format)
PMAC_RX_FRAGMENTS_CNT	PMAC Rx Undersize Counter (CRC error)
PMAC_RX_IN_RANGE_LEN_ERR_CNT	PMAC Rx In-range Length Error Counter

.....continued

Register	Description
PMAC_RX_OUT_OF_RANGE_LEN_ERR_CNT	PMAC Rx Out-Of-Range Length Error Counter
PMAC_RX_OVERSIZE_CNT	PMAC Rx Oversize Counter (valid frame format)
PMAC_RX_JABBERS_CNT	PMAC Rx Jabbers Counter
PMAC_RX_SIZE64_CNT	PMAC Rx 64 Byte Frame Counter
PMAC_RX_SIZE65TO127_CNT	PMAC Rx 65-127 Byte Frame Counter
PMAC_RX_SIZE128TO255_CNT	PMAC Rx 128-255 Byte Frame Counter
PMAC_RX_SIZE256TO511_CNT	PMAC Rx 256-511 Byte Frame Counter
PMAC_RX_SIZE512TO1023_CNT	PMAC Rx 512-1023 Byte Frame Counter
PMAC_RX_SIZE1024TO1518_CNT	PMAC Rx 1024-1518 Byte Frame Counter
PMAC_RX_SIZE1519TOMAX_CNT	PMAC Rx 1519 To Max. Length Byte Frame Counter
PMAC_TX_PAUSE_CNT	PMAC Tx Pause Frame Counter
PMAC_TX_UC_CNT	PMAC Tx Unicast Frame Counter
PMAC_TX_MC_CNT	PMAC Tx Multicast Frame Counter
PMAC_TX_BC_CNT	PMAC Tx Broadcast Frame Counter
PMAC_TX_SIZE64_CNT	PMAC Tx 64 Byte Frame Counter
PMAC_TX_SIZE65TO127_CNT	PMAC Tx 65-127 Byte Frame Counter
PMAC_TX_SIZE128TO255_CNT	PMAC Tx 128-255 Byte Frame Counter
PMAC_TX_SIZE256TO511_CNT	PMAC Tx 256-511 Byte Frame Counter
PMAC_TX_SIZE512TO1023_CNT	PMAC Tx 512-1023 Byte Frame Counter
PMAC_TX_SIZE1024TO1518_CNT	PMAC Tx 1024-1518 Byte Frame Counter
PMAC_TX_SIZE1519TOMAX_CNT	PMAC Tx 1519 To Max. Length Byte Frame Counter
PMAC_RX_ALIGNMENT_LOST_CNT	PMAC Counter to track the dribble-nibble (extra nibble) errors in frames.
MM_RX_ASSEMBLY_ERR_CNT	MAC MERGE - The number of Pframes received with aseembly error
MM_RX_SMD_ERR_CNT	MAC MERGE - The number of Pframe fragaments rejected die to unknown SMD values or arrving with SMD-C when no pframe in progress
MM_RX_ASSEMBLY_OK_CNT	MAC MERGE - The number of Pframes received whitch are successfully reassembled
MM_RX_MERGE_FRAG_CNT	MAC MERGE - The number of Pframes received whitch are mPackets (Fragments) due to preemption

.....continued

Register	Description
MM_TX_PFRAGMENT_CNT	MAC MERGE - The number of Pframe fragments sent when preemption enabled
PMAC_RX_HIH_CKSM_ERR_CNT	PMAC Rx HiH Checksum Error Counter
PMAC_RX_XGMII_PROT_ERR_CNT	PMAC Rx XGMII Protocol Error Counter
RX_IN_BYTES_CNT	The number of bytes received (good, bad, and framing).
RX_IN_BYTES_MSB_CNT	The number of bytes received (good, bad, and framing) - MSBs only
RX_OK_BYTES_CNT	The number of received bytes in good frames.
RX_OK_BYTES_MSB_CNT	The number of received bytes in good frames - MSBs only.
RX_BAD_BYTES_CNT	The number of received bytes in bad frames.
RX_BAD_BYTES_MSB_CNT	The number of received bytes in bad frames - MSBs only
TX_OUT_BYTES_CNT	The number of bytes transmitted (good, bad and framing)
TX_OUT_BYTES_MSB_CNT	The number of bytes transmitted (good, bad, framing) - MSBs only
TX_OK_BYTES_CNT	The number of bytes transmitted successfully.
TX_OK_BYTES_MSB_CNT	The number of transmitted bytes transmitted successfully - MSBs only
PMAC_RX_OK_BYTES_CNT	PMAC - The number of received bytes in good frames.
PMAC_RX_OK_BYTES_MSB_CNT	PMAC - The number of received bytes in good frames - MSBs only.
PMAC_RX_BAD_BYTES_CNT	PMAC - The number of received bytes in bad frames
PMAC_RX_BAD_BYTES_MSB_CNT	PMAC - The number of received bytes in bad frames - MSBs only
PMAC_TX_OK_BYTES_CNT	PMAC - The number of bytes transmitted successfully.
PMAC_TX_OK_BYTES_MSB_CNT	PMAC - The number of transmitted bytes transmitted successfully - MSBs only.

The counters are writable. In particular, the counters are cleared by writing a 0 to each counter.

4.9 Assembler

The assembler (ASM) block is responsible for collecting words from the smaller taxi bus and assembling them into cells. It is also responsible for the loopback path between the rewriter and the analyzer.

For the first cell of a frame, which is the SOF cell, the assembler adds room for a 36-byte internal frame header (IFH). On a stacking port, the stacking tag is extracted from the frame data and placed in the respective part of the IFH.

The assembler receives a calendar sequence from the queue system defining in the sequence the ports should be served on the outgoing cell bus.

The assembler also detects PAUSE frames and forwards the extracted PAUSE information to the disassembler. PFC pause information extracted from PFC PAUSE frames are forwarded to the queue system.

Finally, the assembler collects the port statistics for all lower speed ports, which are the DEV2G5 ports. Statistics for the high-speed DEV5G, DEV10G, and DEV25G ports are handled locally in the port module.

4.9.1 Setting Up a Port in the Assembler

The following table lists the port configuration registers within the assembler. Ethernet ports and CPU ports are configured using the same registers. Some of the fields are only relevant for setting up a CPU port (internal or external) and they will be covered in a later section.

Table 4-67. Port Configuration Register Overview

Target::Register.Field	Description	Replication
ASM::PORT_CFG.NO_PREAMBLE_ENA	Preamble configuration of incoming frames	Per port
ASM::PORT_CFG.SKIP_PREAMBLE_ENA	Preamble configuration of incoming frames	Per port
ASM::PORT_CFG.PAD_ENA	Enable padding	Per port
ASM::PORT_CFG.INJ_DISCARD_CFG	Configures discard behavior for injected frames	Per port
ASM::PORT_CFG.INJ_FORMAT_CFG	Configure format of injected frames	Per port
ASM::PORT_CFG.VSTAX2_AWR_ENA	Enable VStaX stacking header awareness	Per port

By default, an Ethernet port does not need configuration in the assembler as long as special settings are not required. However, the following exceptions may apply:

If the port is used as a stacking port, set ASM::PORT_CFG.VSTAX2_AWR_ENA, which enables detection of the VStaX stacking header. If a VStaX stacking header is found, the assembler will remove the header from the frame and put it into the internal frame header.

Frames received from the port modules are preamble prepended by default. If a port module is configured for preamble shrink mode, the ASM::PORT_CFG.NO_PREAMBLE_ENA must be set to 1, because the port module does not prepend a preamble in this mode.

When ASM::PORT_CFG.PAD_ENA is set, frames that are smaller than 64 bytes are padded to reach the minimum frame size.

The assembler has a built-in frame fragment detection mechanism for incoming frames that were started but never received their EOF (because the port module was taken down, for example). These frames are normally finalized by creating an EOF abort marked cell. If a frame has been discarded, it is noted in ASM::PORT_STICKY.FRM_AGING_STICKY. The following table lists the port status register.

Table 4-68. Port Status Register Overview

Target::Register.Field	Description	Replication
ASM::PORT_STICKY.IFH_PREFIX_ERR_STICKY	Injection format mismatch sticky bit	Per port
ASM::PORT_STICKY.FRM_AGING_STICKY	Frame aging sticky bit	Per port

4.9.2 Setting Up a Port for Frame Injection

Any front port and the internal CPU ports (ports 64-65) can be configured for frame injection. Frames that are to be injected must have an IFH perpended the frame data. Optionally, the frame can further be perpended an SMAC, a DMAC, and a VLAN tag. This results in the following three prefix formats for injection frames.

- Injection with long prefix
- Injection with short prefix
- Injection with no prefix

The injection format is selected in ASM::PORT_CFG.INJ_FORMAT_CFG. The prefix formats are shown in the following figures.

Figure 4-11. Frame Injection Formats

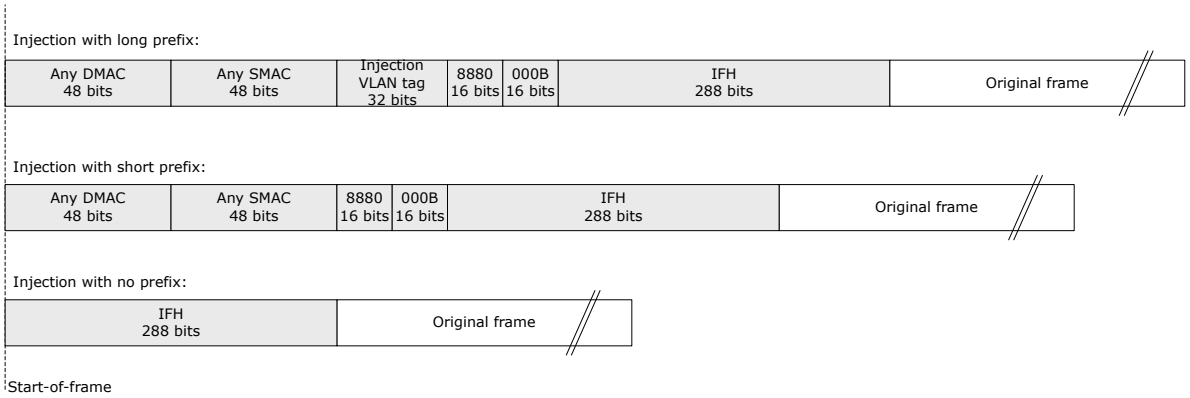


Table 4-69. Injection VLAN Register Overview

Target::Register.Field	Description	Replication
ASM::INJ_VLAN_CFG.INJ_VID_CFG	Injection tag VID	1
ASM::INJ_VLAN_CFG.INJ_TPID_CFG	Injection tag TPID	1

When a port is setup to receive frames with either a short prefix or a long prefix, the incoming frames are checked against the selected prefix. If the prefix does not match, an error indication is set in ASM::PORT_STICKY.IFH_PREFIX_ERR_STICKY. Depending on the setting of ASM::PORT_CFG.INJ_DISCARD_CFG, the frames are processed in one of the three following modes.

- None. Both compliant and non-compliant frames are forwarded. Compliant frames are forwarded based on the IFH, and non-compliant frames are forwarded as regular frames.
- Drop non-compliant. Compliant frames are forwarded based on the IFH, and non-compliant frames are discarded.
- Drop compliant. Compliant frame are discarded, and non-compliant frames are forwarded as normal frames.

If a CPU port is used for frame injection, ASM::PORT_CFG.NO_PREAMBLE_ENA must be set to 1 so that the assembler does not expect frame data to be perpended with a preamble.

If a front port is used for frame injection, ASM::PORT_CFG.SKIP_PREAMBLE_ENA must be set to 1 to discard the preamble data before processing of the perpended injection header.

4.9.3 Setting Up MAC Control Sublayer PAUSE Frame Detection

The following table lists the PAUSE frame detection configuration registers in the assembler.

Table 4-70. PAUSE Frame Detection Configuration Register Overview

Target::Register.Field	Description	Replication
ASM::PAUSE_CFG.ABORT_PAUSE_ENA	Stop forwarding of PAUSE frames	Per port
ASM::PAUSE_CFG.ABORT_CTRL_ENA	Stop forwarding of MAC control frames	Per port
ASM::MAC_ADDR_HIGH_CFG.MAC_ADDR_HIGH	Bits 47-24 of DMAC address checked in MAC_Control frames	Per port
ASM::MAC_ADDR_LOW_CFG.MAC_ADDR_LOW	Bits 23-0 of DMAC address checked in MAC_Control frames	Per port

The assembler is responsible for detecting PAUSE frames and forwarding the PAUSE value to the disassembler. Because PAUSE frames belong to the MAC control sublayer, they should in general be filtered and not forwarded to

a higher layer; that is, forwarded on the cell bus so they reach the analyzer. The filtering is done by abort marking the frame.

The PAUSE frame and other MAC control frames can be forwarded to the analyzer. For PAUSE frame forwarding, ASM::PAUSE_CFG.ABORT_PAUSE_ENA must be set to 0. For other MAC control frame forwarding, ASM::PAUSE_CFG.ABORT_CTRL_ENA must be set to 0.

When PAUSE frames are received, the DMAC address is checked. The DMAC address must be either the 48-bit multicast address 01-80-C2-00-00-01 as mentioned by IEEE802.3 Annex 31B.1 or the MAC address of the port, which is configured in ASM::MAC_ADDR_HIGH_CFG.MAC_ADDR_HIGH and ASM::MAC_ADDR_LOW_CFG.MAC_ADDR_LOW.

Note:
The values configured in ASM::MAC_ADDR_HIGH_CFG.MAC_ADDR_HIGH and ASM::MAC_ADDR_LOW_CFG.MAC_ADDR_LOW must match the value configured in DSM::MAC_ADDR_BASE_HIGH_CFG.MAC_ADDR_HIGH and DSM::MAC_ADDR_BASE_LOW_CFG.MAC_ADDR_LOW.

The number of received PAUSE frames is tracked as part of the port statistics. For more information, see 4.9.5 Setting Up Assembler Port Statistics.

4.9.4 Setting Up PFC

The PFC module of the assembler identifies PFC PAUSE frames by checking if the DMAC matches the reserved multicast address 01-80-c2-00-00-01, the Type/Length field matches a MAC control frame (0x8808), and the opcode is indicating a PFC frame (0x0101). Upon receiving a PFC PAUSE frame, the assembler extracts and stores the timer information for all enabled priorities in the assembler PFC timers. The PFC module generates a stop signal towards the queue system based on the current timer values.

Detection of PFC frames is enabled by setting ASM::PFC_CFG.RX_PFC_ENA to 1 for the respective (port,priority). When enabling PFC the current link speed for the port must be configured in ASM::PFC_CFG.FC_LINK_SPEED for timing information to be evaluated correctly. The PFC configuration registers are shown in the following table.

Table 4-71. PFC Configuration Register Overview

Target::Register.Field	Description	Replication
ASM::PFC_CFG.RX_PFC_ENA	Enable PFC per priority	Per port
ASM::PFC_CFG.FC_LINK_SPEED	Configure the link speed	Per port

4.9.5 Setting Up Assembler Port Statistics

The assembler collects port statistics for all DEV2G5s. Statistics are also collected from the disassembler and the assembler. Port statistics for DEV5G, DEV10Gs, and DEV25Gs are not collected in the assembler and must be looked up in each DEV separately.

The following table lists the port statistics configuration register in the assembler.

Table 4-72. Port Statistics Configuration Register Overview

Target::Register.Field	Description	Replication
ASM::STAT_CFG.STAT_CNT_CLR_SHOT	Statistics counter initialization.	1
ASM::PORT_CFG.CSC_STAT_DIS	Disables collection of statistics in the ASM.	Per port

Port statistics inside the ASM are stored in RAMs. Before they can be used, they must be initialized to 0 by setting ASM::STAT_CFG.STAT_CNT_CLR_SHOT to 1.

The statistic counters start counting as soon as the hardware has reset ASM::STAT_CFG.STAT_CNT_CLR_SHOT to 0.

For information about the lists of port statistics registers available per port, see ASM:DEV_STATISTICS register group in the Register List.

All statistic counters have a width of 32 bits, except for the five byte-counters; RX_IN_BYTES_CNT, RX_OK_BYTES_CNT, RX_BAD_BYTES_CNT, TX_OUT_BYTES_CNT, and TX_OK_BYTES_CNT. Those have an

additional 4-bit MSB counter. When reading from counters with more than 32 bits, the non-MSB part has to be read first in order to latch the MSB part into a shadow register, where the value can be read afterward. For writing, the MSB part must be written first.

4.9.6 Setting Up the Loopback Path

Cells to be looped back by the assembler are received on a separate loopback cell bus interface from the rewriter. The assembler merges the loopback data with data received from the ports onto the outgoing cell bus towards the analyzer. Loopback data belongs to one of the three following types.

- Virtual device 0 (VD0)
- Virtual device 1 (VD1)
- Virtual device 2 (VD2)
- Front port loopback (LBK)

Each of the three loopback traffic types has its own FIFO. VD0 is accessed at register replication 0, VD1 at register replication 1, VD2 at register replication 2, and LBK at register replication 3. The following table lists the loopback configuration registers.

Table 4-73. Loopback FIFO Configuration Register Overview

Target::Register.Field	Description	Replication
ASM::LBK_FIFO_CFG.FIFO_FLUSH	Flush all data in the FIFO.	Per FIFO
ASM::VD_FC_WM.VD_FC_WM	Watermark for flow control towards the queue system.	Per VD FIFO

A FIFO can be flushed using the ASM::LBK_FIFO_CFG.FIFO_FLUSH register. The register field clears itself when the flush is completed.

Cells in the FIFO can be discarded due to aging. If a frame is discarded due to aging, a per port sticky bit is set in ASM::LBK_AGING_STICKY. The following table lists the loopback FIFO sticky bit registers.

Table 4-74. Loopback FIFO Sticky-Bit Registers Overview

Target::Register.Field	Description	Replication
ASM::LBK_AGING_STICKY.LBK_AGING_STICKY	Set if a frame has been discarded due to aging. Per port bitmask.	1
ASM::LBK_OVFLW_STICKY.LBK_OVFLW_STICKY	Set if a frame have been discarded due to overflow.	1

For VD0, VD1, and VD2, the loopback block pushes back towards the queue system to avoid FIFO overflows. The watermark for configuring the FIFO level at which push back is active can be set in ASM::VD_FC_WM.VD_FC_WM. The watermark can be configured individually for VD0, VD1, and VD2.

No flow control handling exists for the LBK FIFO. In case of overflow, all new cells received from the rewriter are discarded, and a bit in the ASM::LBK_OVFLW_STICKY.LBK_OVFLW_STICKY sticky-bit register is set.

4.10 Versatile Content-Aware Processor (VCAP™)

The Versatile Content-Aware Processor (VCAP™) is a content-aware packet processor that allows wirespeed packet inspection for rich implementation of, for example, advanced VLAN and QoS classification and manipulations, IP source guarding, longest prefix matching for Layer-3 routing, and security features for wireline and wireless applications. This is achieved by programming rules into the VCAP.

When a VCAP is enabled, every frame passing though the switch is analyzed and multiple keys are created based on the contents of the frame. The frame is examined to determine the frame type (for example, IPv4 TCP frame), so that the frame information is extracted according to the frame type, portspecific configuration, and classification results from the basic classification. Keys are applied to the VCAP and when there is a match between a key and a rule in the VCAP, the rule is then applied to the frame from which the key was extracted.

A rule consists of an entry, an action, and a counter. Keys are matched against the entry of a rule. When a rule is matched, the action is returned by the VCAP, and the rule's counter is incremented.

One and only one rule will match a key. If a key matches more than one rule, the rule at the highest address is selected. This means that a rule at high address takes priority over rules at lower addresses. When a key does not match a rule, a default rule is triggered. A default rule's action is programmable just like regular rules. Some VCAPs have more than one default rule; which default rule to use is determined at the same time as the key is created and applied to the VCAP.

This section provides information about entries and actions in general terms. When discussing an entry or an action, it is considered a vector of bits that is provided by software. For information about building classification matching (CLM) keys and actions, see 4.13 VCAP CLM Keys and Actions. For information about building ingress stage 2 (IS2) keys, see 4.19 VCAP IS2 Keys and Actions. For information about building IPv6 prefix keys and actions, see 4.10.3.2 VCAP IP6PFX. For information about building longest prefix matching (LPM) keys, see 4.13 VCAP CLM Keys and Actions. For information about building egress stage 0 (ES0) keys, see 4.28.5 VCAP_ES0 Lookup. For information about building egress stage 2 (ES2) keys and actions, see 4.10.3.6 VCAP ES2.

The following sections describe how to program rules into the VCAPs. First, a general description of configuration and VCAP operations is provided, including a description of wide VCAP rules. Second, detailed information about individual VCAPs is provided. Finally, a few practical examples illustrate how everything fits together when programming rules into VCAPs.

4.10.1 Configuring VCAP

Registers are available in four targets: VCAP_IP6PFX, VCAP_ES0, VCAP_ES2, and VCAP_SUPER. Use the VCAP_IP6PFX target to configure the VCAP IP6PFX, the VCAP_ES0 target to configure the VCAP ES0, the VCAP_ES2 target to configure the VCAP ES2, and the VCAP_SUPER target for all other VCAPs.

The following table lists the registers associated with the targets.

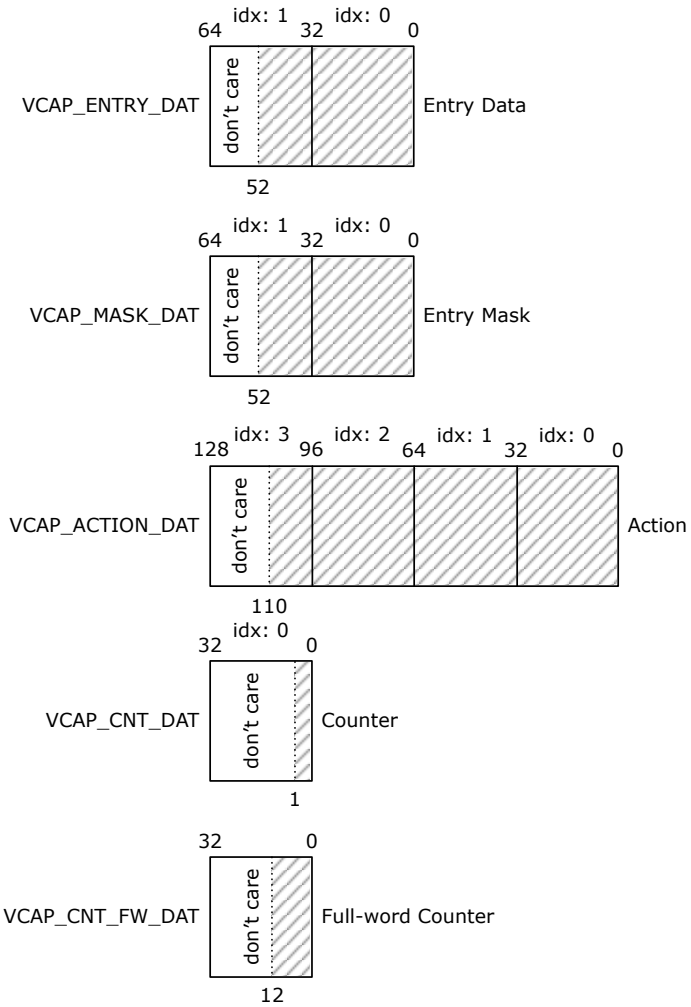
Table 4-75. VCAP_IP6PFX, VCAP_ES0, VCAP_ES2, and VCAP_SUPER Registers

Register	Description
VCAP_UPDATE_CTRL	Initiates of read/write/move/initialization operations
VCAP_MV_CFG	Configures move/initialization operations
VCAP_CORE_IDX	Resource allocation, core index
VCAP_CORE_MAP	Resource allocation, mapping of cores
VCAP_ENTRY_DAT	Cache: Entry data
VCAP_MASK_DAT	Cache: Entry mask
VCAP_ACTION_DAT	Cache: Action
VCAP_CNT_DAT	Cache: Sticky-counter
VCAP_CNT_FW_DAT	Cache: Full-word counter, 1 bit per address offset

A rule in the VCAP consists of an entry, action, and counter showing if the entry was hit. Rules are accessed indirectly though a cache. The cache corresponds to one address in VCAP memory. Software access the cache using the VCAP configuration registers VCAP_ENTRY_DAT, VCAP_MASK_DAT, VCAP_ACTION_DAT, VCAP_CNT_DAT, and VCAP_CNT_FW_DAT.

The following illustration shows an example cache register mapping for a VCAP with 52-bit entry data/mask, 110-bit action, and a 1-bit sticky counter. Cache registers are replicated to accommodate fields that are wider than 32 bits. For example, if the entry size is 52 bits, bits [31:0] are accessed using VCAP_ENTRY_DAT[0][31:0], and bits [51:32] are accessed using VCAP_ENTRY_DAT[1][19:0].

Figure 4-12. VCAP Cache Layout Example



Entries have both entry data and entry mask fields. This allows a don't-care of any bit position in the key when matching a key against entries.

Table 4-76. Entry Bit Encoding

VCAP_ENTRY_DAT[n]/ VCAP_MASK_DAT[n]	Description
0/0	Bit n is encoded for the match-0 operation. When a key is applied to the VCAP, this bit in the entry will match a 0 at position n in the key.
0/1	Bit n is encoded for match-any operation. When a key is applied to the VCAP, this bit in the entry will match both 0 and 1 at position n in the key. This encoding is sometimes referred to as don't care.
1/0	Bit n is encoded for match-1 operation. When a key is applied to the VCAP, this bit in the entry will match a 1 at position n in the key.

.....continued	
VCAP_ENTRY_DAT[n]/ VCAP_MASK_DAT[n]	**Description**
1/1	Bit n is encoded for match-off operation. The entry will never match any keys. This encoding is not available in the VCAP ES0. Instead, it is treated as match-any.

When programming a rule that does not use all the available bits in the cache, remaining (undefined) entry bits must be set to match-0, and action bits must be set to zeros.

The counter for a rule is incremented when the entry of the rule is matched by a key. Counters that are 1-bit wide do not wrap, but instead saturate at 1.

4.10.1.1 Writing, Reading, and Initializing Rules

All access to and from the VCAP memory goes through the cache.

To write a rule in the VCAP memory at address a: Entry data, entry action, and counter is first written to the cache registers and then transferred into VCAP memory by triggering a write operation on the cache by setting VCAP_UPDATE_CTRL.UPDATE_CMD = 0, VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATE_CTRL.UPDATE_SHOT = 1. The UPDATE_SHOT field is cleared by hardware when the rule has put into VCAP memory.

To read a rule from address a in the VCAP memory: Trigger a read operation on the cache by setting VCAP_UPDATE_CTRL.UPDATE_CMD = 1, VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATA_CTRL.UPDATE_SHOT = 1. The UPDATE_SHOT field is cleared by hardware when a copy of the rule is available for reading via cache registers.

Active rules must not be overwritten by other rules (except when writing disabled rules). Software must make sure that addresses are initialized before they are written.

It is possible to write the contents of the cache can be written to a range of addresses inside VCAP memory. This is used during initialization of VCAP memory.

To write the contents of the cache to addresses a through b in VCAP memory, where a < b: Trigger an initialization-operation by setting VCAP_MV_CFG.MV_NUM_POS = b-a. And VCAP_UPDATE_CTRL.UPDATE_CMD = 4, VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATE_CTRL.UPDATA_SHOT = 1. When the UPDATE_SHOT field is cleared, the addresses have been overwritten with the contents of the cache.

The cache can be cleared by writing VCAP_UPDATE_CTRL.CLEAR_CACHE = 1. This immediately sets the contents of the entry to disabled, action and counter is set to all-zeros. VCAP_UPDATE_CTRL.CLEAR_CACHE can be set at the same time as triggering an write or initialization operation, the cache will be cleared before writing to the VCAP memory.

It is possible to selectively disable access to entry, action, and/or counter during operations by setting VCAP_UPDATE_CTRL.UPDATE_ENTRY_DIS, VCAP_UPDATE_CTRL.UPDATE_ACTION_DIS, or VCAP_UPDATE_CTRL.UPDATE_CNT_DIS. These fields allow specialized operations such as clearing counter values by performing initialization or write operations with disabled entry and action updating.

4.10.1.2 Moving a Block of Rules

The VCAP supports move operation for efficient and safe moving of rules inside VCAP memory. Moving may be required to make room for new rules, because rules are prioritized by address.

To move n addresses from address a to address b in VCAP memory: Trigger a move-operation by setting VCAP_MV_CFG.MV_NUM_POS = (a>b ? a-b : b-a), VCAP_MV_CFG.MV_SIZE = (n-1). And setting VCAP_UPDATE_CTRL.UPDATE_CMD = (a>b ? 2 : 3), VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATE_CTRL.UPDATE_SHOT = 1. When the UPDATE_SHOT field is cleared, the contents of the addresses have been moved inside VCAP memory. The cache is overwritten during the move-operation.

Rules must not be moved to a region that contains active rules. Software must make sure that destination addresses are initialized before performing a move operation. After moving rules the move operation leaves VCAP memory in initialized state, so overlapping source and destination regions is not a problem during move operations.

4.10.1.3 Sharing Resources

The device implements a shared pool of 10 blocks that can be distributed freely among the CLM, IS2, and LPM VCAPs. Each block in the pool has 3,072 addresses with entries, actions, and counters.

Blocks are located back-to-back in the VCAP memory space. Block 0 is assigned addresses 0 through 3,071; block 1 is assigned addresses 3,072 though 6,143; and so on. Prioritizing of rules based on address still applies when multiple blocks are assigned to the same VCAP.

To simplify software development, all blocks assigned to the same VCAP should be allocated as one consecutive region of addresses in memory. After a block of VCAP resources is allocated to a VCAP, it cannot be reallocated. Resource assignment must be done during startup before the VCAPs are enabled.

To allocate a block from the shared pool, write block index to VCAP_CORE_IDX.CORE_IDX, then map it to a specific interface by writing VCAP_CORE_MAP.CORE_MAP.

4.10.1.4 Bringing Up VCAP

After reset, all VCAP resources default to unallocated power-down mode. To allocate a block, write block index to VCAP_CORE_IDX.CORE_IDX, then map it to a specific interface by writing VCAP_CORE_MAP.CORE_MAP. For more information, see the register description for encoding of owners. This applies to all VCAP targets.

The contents of the VCAP are unknown after reset. Software must initialize all addresses of the VCAP to disabled entry and all-zeros in action and counter before enabling lookups.

The default rules of the VCAPs must also be initialized. After lookup is enabled for the VCAPs, default rules are triggered and applied to frames.

Some VCAPs implement resource sharing. As a result, software must allocate resources as part of VCAP bring up.

For more information about default addresses rules and resource allocation, see 4.10.3 Individual VCAPs.

4.10.2 Wide VCAP Entries and Actions

Some VCAPs support entries and actions that are wider than a single address in the VCAP memory. The size of an entry or action is described as Xn, where n is the number of addresses that is taken up in memory. When programming an X2 or larger entry or action, addresses are written one by one until the entire entry or action is written to VCAP memory.

A rule consists of one entry and one action. The size of the rule is described as Xn where n is the largest of entry or action widths. For example, a rule consisting of an X2 entry and an X3 action is considered to be an X3 rule. The entry and action must start at the same address, so that address offset 0 for both entry and action is located at the same address inside the VCAP memory.

When programming a rule where the action is wider than the entry, entry addresses exceeding the size of the entry must be disabled. When the entry is wider than the action, action addresses exceeding the size of the action must be set to zeros.

In general, the starting address of an Xn rule must be divisible by n. As examples, an X1 rule can be placed at any address and an X2 rule can be placed at even addresses. However, an X3 rule consisting of a mix of either an X2 entry and an X3 action or vice versa must be placed at addresses divisible by 6 since both the entry and the action must be placed at a valid address.

When performing move-operation on a region of VCAP memory that contains rules of X2 or larger size, the rules must still be at legal addresses after the move.

When a rule is matched by a key, the counter at address offset 0 is incremented.

When writing an X2 or larger rule, software must start with the highest address and end with the lowest address. This is needed so that rules are not triggered prematurely during writing. When disabling rules, software must start by disabling the lowest address. This is automatically handled when using the initialization operation for disabling rules.

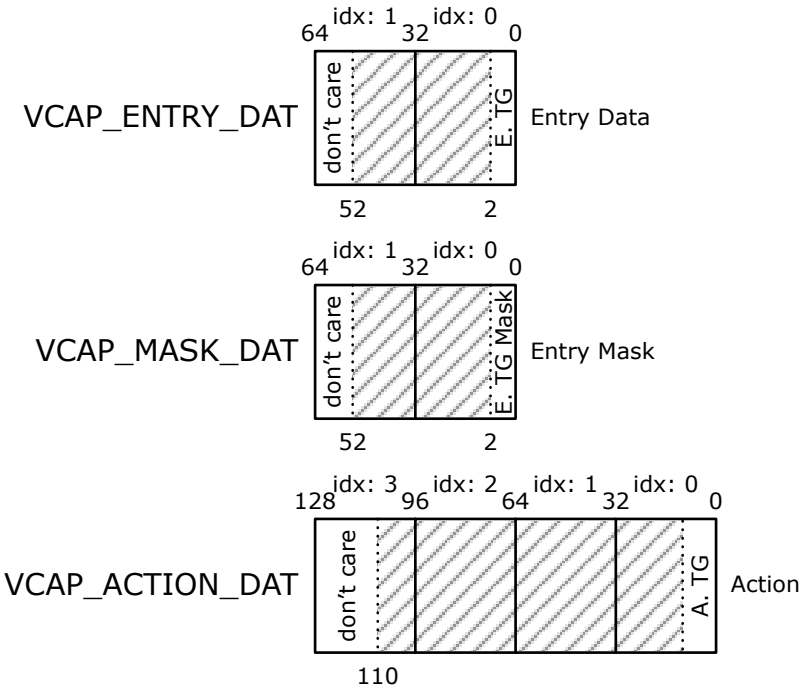
4.10.2.1 Type-Group Fields

A special type-group field is required to differentiate rules when a VCAP supports different sized entries or actions. The type-group is not part of the entry/action description. Software has to manually lookup and insert type-group field into entry/action data when writing rules to VCAP memory.

The type-group field is placed at the low bits of the entry and/or action. The value and width of the typeegroup field differs between VCAPs and depends on the entry/action size of and the address offset into the rule. For more information about type-group fields for individual VCAPs, see 4.10.3 Individual VCAPs.

The following figure shows an example of inserting a 2-bit entry type-group field and a 3-bit action type-group field for a VCAP with 52-bit entry data/mask and a 110-bit action. After inserting of typegroup, there is room for 50 additional entry/mask bits and 107 action bits in the cache.

Figure 4-13. VCAP Cache Type-Group Example



When programming the entry's type-group field, the mask bits must be set to zeros, so that the typegroup bits consist of match-1 or match-0, or both.

Use the following procedure for each address that is part of an entry.

1. Use the entry's current address offset and size to find the appropriate type-group field and associated field-width by using the entry type-group table for the VCAP. Skip step 2 if entry typegroup is not applicable for the VCAP.
2. Insert the entry type-group field into the cache for a type-group field width of n. Write the value of the type-group field to bit-positions [n-1:0] in VCAP_ENTRY_DAT and write zeros to bit-positions [n-1:0] in VCAP_MASK_DAT. In this way, type-group field value 1 becomes match-1 and 0 becomes match-0.
3. Fill the remainder of VCAP_ENTRY_DAT and VCAP_MASK_DAT with appropriate entry data. There will be room for (entry width – type-group width) additional bits of entry data.

Use the following procedure for each address is that is part of an action.

1. Use the action's current address offset and size to find the appropriate type-group field and associated field-width by using the action type-group table for the VCAP. Skip step 2 if action typegroup is not applicable for the VCAP.
2. Insert the action type-group field into the cache. For a type-group field width of n. Write the value of the type-group field to bit positions [n-1:0] in VCAP_ACTION_DAT.

3. Fill the remainder of VCAP_ACTION_DAT with appropriate action data. There will be room for (action width – type-group width) additional bits of action data.

Counters never use type-group encoding.

4.10.3 Individual VCAPs

This section provides detailed information about the individual VCAPs; VCAP CLM, VCAP IS2, VCAP IP6PFX, VCAP LPM, VCAP ES0, and VCAP ES2. The CLM, IS2, and LPM VCAPs share resources. For more information, see 4.10.1.3 Sharing Resources.

4.10.3.1 VCAP CLM

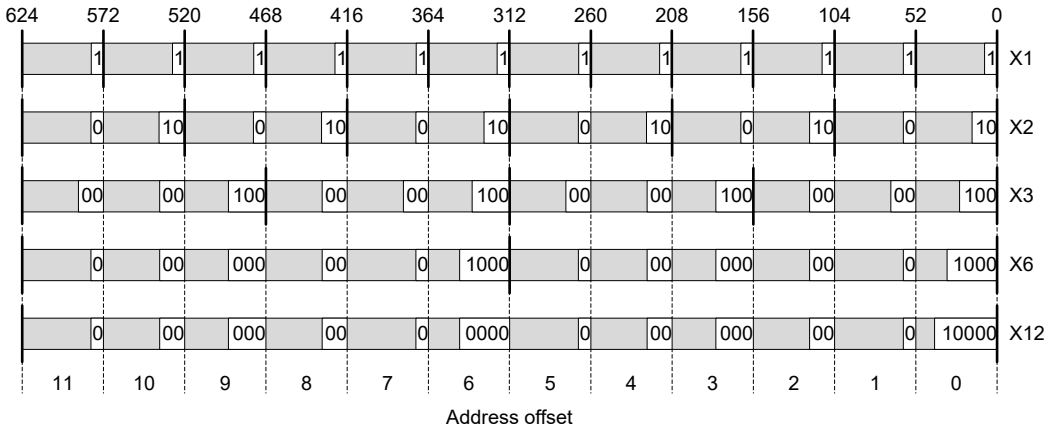
The following table lists the parameters that apply to the VCAP CLM.

Table 4-77. VCAP CLM Parameters

Parameter	Description
Number of VCAP addresses	3,072 per block that is allocated to this VCAP
Width of entry	52 bits per address
Width of action	110 bits per address
Counter type	1-bit saturating counter
Supported entry sizes	X1, X2, X3, X6, and X12
Supported action sizes	X1, X2, X3
Associated register target	VCAP_SUPER
Number of VCAP lookups per frame	6

The type-group field must be inserted into entries when programming rules, because the VCAP CLM supports more than one entry size. The following figure shows the binary type-group field values for each entry size and address offset.

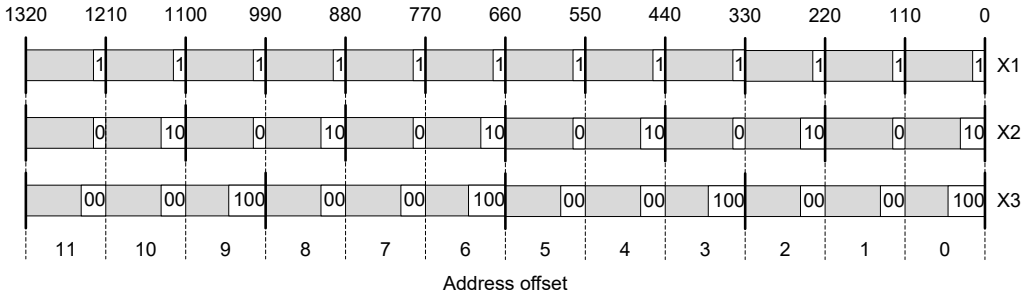
Figure 4-14. VCAP CLM Entry Type-Group Fields



In total, X1 entries require 1 type-group bit per entry, X2 entries require 3 type-group bits per entry, X3 entries require 7 type-group bits per entry, X6 entries require 13 type-group bits per entry, and X12 entries require 27 type-group bits per entry.

The type-group field must be inserted into actions when programming rules, because the VCAP CLM supports more than one action size. The following figure shows the binary type-group field values for each action size and address offset.

Figure 4-15. VCAP CLM Action Type-Group Fields



In total, X1 actions require 1 type-group bit per action, X2 actions require 3 type-group bits per action, and X3 actions require 7 type-group bits per action.

The VCAP CLM implements default rules. When submitting a key to the VCAP CLM, the analyzer simultaneously decides which default rule to hit if the key does not match any entries in the VCAP. If no resources (blocks) are assigned to VCAP CLM, there can be no matches. As a result, default rules will be hit.

Table 4-78. VCAP CLM Default Rule Addresses

VCAP	Lookup	Number of Default Rules	Address
CLM	0	70	Starting address of CLM default rule number n is (30,720 + n × 12).
CLM	1	70	Starting address of CLM default rule number n is (31,560 + n × 12).
CLM	2	70	Starting address of CLM default rule number n is (32,400 + n × 12).
CLM	3	70	Starting address of CLM default rule number n is (33,240 + n × 12).
CLM	4	70	Starting address of CLM default rule number n is (34,080 + n × 12).
CLM	5	70	Starting address of CLM default rule number n is (34,920 + n × 12).

4.10.3.2 VCAP IP6PFX

The following table lists the parameters that apply to the VCAP IP6PFX.

Table 4-79. VCAP IP6PFX Parameters

Parameter	Description
Number of VCAP addresses	1024
Width of entry	52 bits per address
Width of action	10 bits per address
Counter-type	1-bit saturating counter
Supported entry sizes	X2
Supported action sizes	X1
Associated register target	VCAP_IP6PFX
Number of VCAP lookups per frame	1

Although VCAP IP6PFX only supports one entry size, a type-group field must be inserted into entries when programming rules. The following figure shows the binary type-group field values for each address offset.

Figure 4-16. VCAP IP6PFX Entry Type-Group Fields

In total, X2 entries require 3 type-group bits per entry. Although VCAP IP6PFX only supports one action size, a type-group field must be inserted into actions when programming rules. The following figure shows the binary type-group field values.

Figure 4-17. VCAP IP6PFX Action Type-Group Fields

In total, X1 actions require 1 type-group bit per action. The VCAP IP6PFX does not implement default rules. If the key does not match any entries, lookups are treated as misses.

4.10.3.3 VCAP LPM

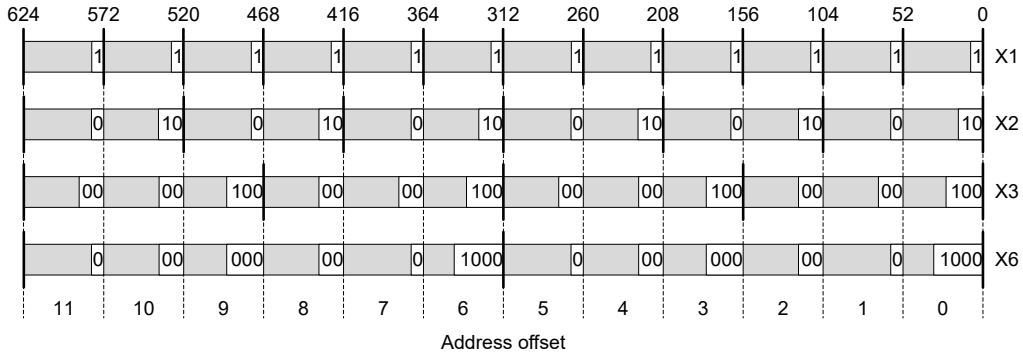
The following table lists the parameters that apply to the VCAP LPM.

Table 4-80. VCAP LPM Parameters

Parameter	Description
Number of VCAP addresses	3,072 per block that is allocated to this VCAP
Width of entry	52 bit per address
Width of action	110 bit per address
Counter-type	1 bit saturating counter
Supported entry sizes	X1, X2, X3, X6
Supported action sizes	X1
Associated register target	VCAP_SUPER
Number of VCAP lookups per frame	2

The type-group field must be inserted into entries when programming rules, because the VCAP LPM supports more than one entry size. The following figure shows the binary type-group field values for each entry size and address offset.

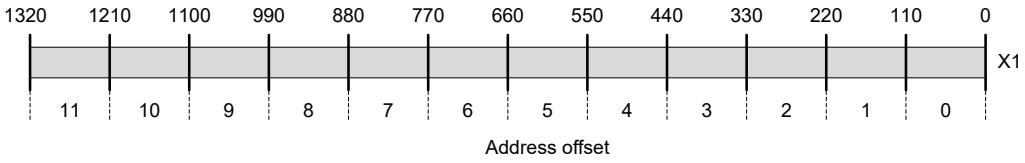
Figure 4-18. VCAP LPM Entry Type-Group Fields



In total, X1 entries require 1 type-group bit per entry, X2 entries require 3 type-group bits per entry, X3 entries require 7 type-group bits per entry, and X6 entries require 13 type-group bits per entry.

VCAP LPM only supports one action size and no type-group fields are required. The following figure shows the binary type-group field values.

Figure 4-19. VCAP LPM Action Type-Group Fields



The VCAP LPM does not implement default rules. If the key does not match any entries, or if no resources (blocks) are assigned to the VCAP LPM, routing lookups are treated as misses.

4.10.3.4 VCAP IS2

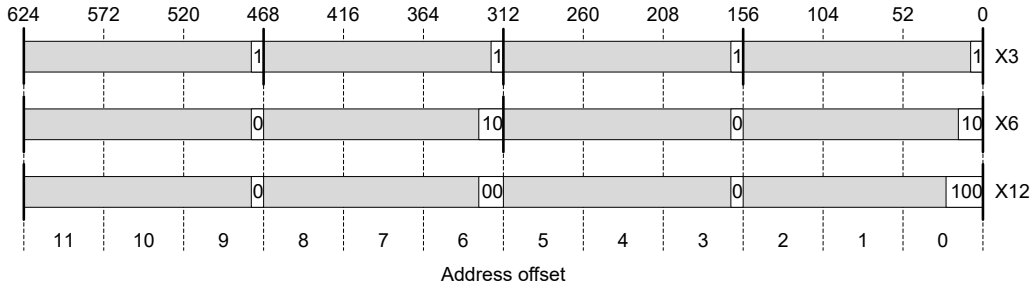
The following table lists the parameters that apply to the VCAP IS2.

Table 4-81. VCAP IS2 Parameters

Parameter	Description
Number of VCAP addresses	3,072 per block that is allocated to this VCAP
Width of entry	52 bits per address
Width of action	110 bits per address
Counter-type	1-bit saturating counter
Supported entry sizes	X3, X6, X12
Supported action sizes	X3
Associated register target	VCAP_SUPER

The type-group field must be inserted into entries when programming rules, because the VCAP IS2 supports more than one entry size. The following figure shows the binary type-group field values for each entry size and address offset.

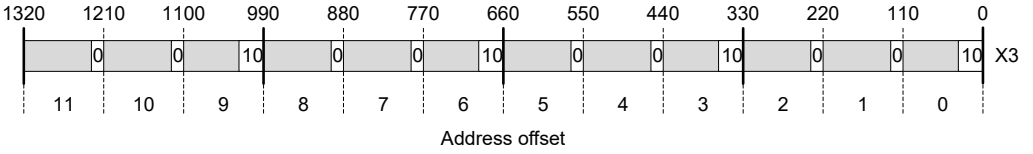
Figure 4-20. VCAP IS2 Entry Type-Group Fields



In total, X3 entries require one type-group bits per entry, X6 entries require three type-group bits per entry, and X12 entries require 7 type-group bits per entry.

Although VCAP IS2 only supports one action size, a type-group field must be inserted into actions when programming rules. The following figure shows the binary type-group field values for each address offset.

Figure 4-21. VCAP IS2 Action Type-Group Fields



In total, X3 actions require 4 type-group bits per action.

The VCAP IS2 implements default rules. When submitting a key to the VCAP IS2, the analyzer simultaneously decides which default rule to hit if the key does not match any entries in the VCAP. If no resources (blocks) are assigned to VCAP IS2, there can be no matches. As a result, default rules will be hit.

Table 4-82. VCAP IS2 Default Rule Addresses

VCAP	Lookup	Number of Default Rules	Address
IS2	0	70	Starting address of IS2 default rule number n is (35,760 + n × 12)
IS2	1	1	Starting address of IS2 default rule is 36,600
IS2	2	1	Starting address of IS2 default rule is 36,612
IS2	3	1	Starting address of IS2 default rule is 36,624

4.10.3.5 VCAP ES0

The following parameters apply to the VCAP ES0.

Table 4-83. VCAP ES0 Parameters

Parameter	Description
Number of VCAP addresses	4,096
Width of entry	51 bits per address
Width of action	489 bits per address
Counter-type	1-bit saturating counter

.....continued	
Parameter	**Description**
Supported entry sizes	X1
Supported action sizes	X1
Associated register target	VCAP_ES0
Number of VCAP lookups per frame	1

The type-group field is not used when programming entries and actions, because VCAP ES0 supports only one entry size and one action size of same size.

The VCAP ES0 has 70 default rules. The address of default rule number n is (4,096 + n). When submitting a key to the VCAP ES0, the rewriter simultaneously decides which default rule to hit if the key does not match any entries in the VCAP.

4.10.3.6 VCAP ES2

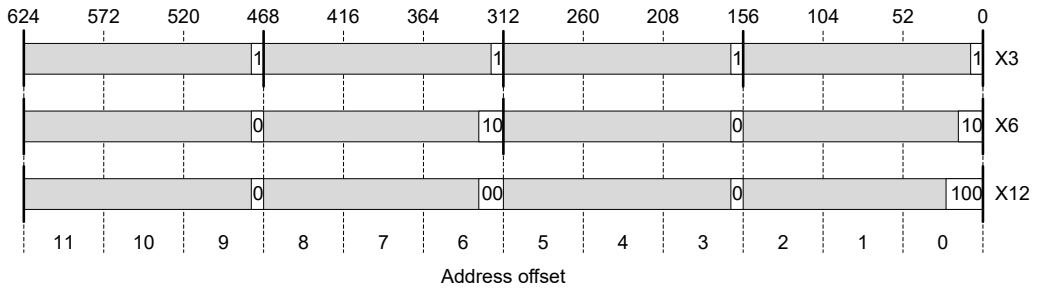
The following table lists the parameters that apply to the VCAP ES2.

Table 4-84. VCAP ES2 Parameters

Parameter	Description
Number of VCAP addresses	12,288 per block that is allocated to this VCAP
Width of entry	52 bits per address
Width of action	21 bits per address
Counter-type	1-bit saturating counter
Supported entry sizes	X3, X6, X12
Supported action sizes	X3
Associated register target	VCAP_SUPER
Number of VCAP lookups per frame	2

The type-group field must be inserted into entries when programming rules, because the VCAP ES2 supports more than one entry size. The following figure shows the binary type-group field values for each entry size and address offset.

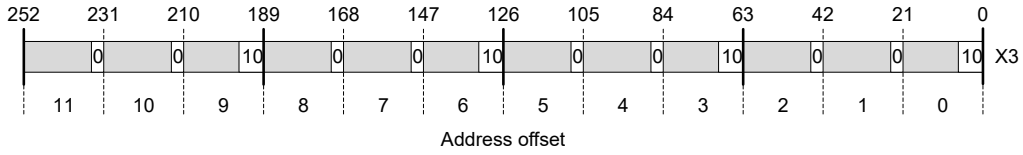
Figure 4-22. VCAP ES2 Entry Type-Group Fields



In total, X3 entries require 1 type-group bits per entry, X6 entries require 3 type-group bits per entry, and X12 entries require 7 type-group bits per entry.

Although VCAP ES2 only supports one action size, a type-group field must be inserted into actions when programming rules. The following figure shows the binary type-group field values for each address offset.

Figure 4-23. VCAP ES2 Action Type-Group Fields



In total, X3 actions require 4 type-group bits per action.

The VCAP ES2 implements default rules. When submitting a key to the VCAP ES2, the analyzer simultaneously decides which default rule to hit if the key does not match any entries in the VCAP. If no resources (blocks) are assigned to VCAP ES2, there can be no matches. As a result, default rules will be hit.

Table 4-85. VCAP ES2 Default Rule Addresses

VCAP	Lookup	Number of Default Rules	Address
ES2	0	73	Starting address of ES2 default rule number n is (12,228 + n × 12)
ES2	1	1	Starting address of ES2 default rule is 13,164

4.10.4 VCAP Programming Examples

This section provides examples of programming VCAP CLM and VCAP ES0.

4.10.4.1 Writing a Wide CLM Rule

This example shows how to write a CLM X3 rule, consisting of an X2 TRI_VID entry and an X3 FULL action, to the CLM1 VCAP. In this example, the second and third blocks of VCAP resources have already been mapped to CLM1. When the second and third resource blocks are mapped to CLM1, it then owns VCAP address range 3,072-9,216.

When programming a rule where entry and action have different sizes, the largest of the two decides the rule size. The rule's starting address must be divisible by both the entry size and the action size. In the current example, the X3 action determines the rule size and the starting address must be divisible with six to comply with the X2 entry and the X3 action. Addresses 4,482-4,484 are chosen for the X3 rule.

Software is expected to track memory usage so that it only writes to VCAP memory that does not already contain active rules. Deleting of rules is done by initializing the associated addresses.

The X2 TRI_VID entry is 100 bits wide. The X3 FULL action is 317 bits wide. For information about how to build a TRI_VID entry, see 4.13.3 VCAP CLM X2 Key Details. For information about how to build a FULL action, see 4.13.7 VCAP CLM Actions.

Each entry address holds 52 bits of entry data/mask, but some of them are used for type-group fields. For more information about type-group fields for an X2 entry, see Figure 4-14. The resulting programming is shown in the following table.

Table 4-86. CLM X2 Entry Type-Group and Distribution Summary

Address	Type-Group Field	Entry Data/Mask Range
4,482	0b10	[49:0]
4,483	0b0	[100:50]
4,484	Unused by rule but reserved	

The TRI_VID entry is only 100 bits wide, so bit 100 of address offset 1 must be treated as Match-0.

Each action address holds 110 bits of action, but some of them are used for type-group fields. For more information the type-group fields for an X3 action, see Figure 4-15. The resulting programming is shown in the following table.

Table 4-87. CLM X3 Action Type-Group and Distribution Summary

Address	Type-Group Field	Action Data
4,482	0b100	[106:0]
4,483	0b00	[214:107]
4,484	0b00	[322:215]

The FULL action is only 317 bits wide, so bits [322:317] of address offset 2 must be treated as zeros.

Software must write the highest address offset first and work down to the lowest address offset.

4.10.4.2 Writing Address Offset 2 of X3 Rule

Software is not allowed to modify the cache while VCAP operation is ongoing. Always check if VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared before starting to modify the cache. If a previous operation is still ongoing, wait for the UPDATE_SHOT to clear.

When a new rule is started, first clear cache's entry, action, and counter by setting VCAP_UPDATE_CTRL.CLEAR_CACHE = 1.

The TRI_VID entry is an X2 entry and not part of address offset 3. The entry at this address must be set to Match-off. This has already been achieved by clearing of the cache.

The FULL action is not so wide that it needs a fifth VCAP_ACTION_DAT replication, so the fifth VCAP_ACTION_DAT replication must be set to zero, which was already achieved by clearing the cache.

Write action to cache:

- VCAP_ACTION_DAT[3][13:0] = action[322:309]
- VCAP_ACTION_DAT[2][31:0] = action[308:277]
- VCAP_ACTION_DAT[1][31:0] = action[276:245]
- VCAP_ACTION_DAT[0][31:2] = action[244:215]
- VCAP_ACTION_DAT[0][1:0] = 0, because type-group field for the X3 CLM action at address offset 3 is 0b00.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(4,484<<3)).

4.10.4.3 Writing Address Offset 1 of X3 Rule

Wait for VCAP_UPDATE_CTRL.UPDATE_SHOT to clear.

Write entry data to cache:

- VCAP_ENTRY_DAT[1][19:0] = entry data[100:81]
- VCAP_ENTRY_DAT[0][31:1] = entry data[80:50]
- VCAP_ENTRY_DAT[0][0] = 0, because type-group field for the X2 CLM entry at address offset 1 is 0b0.

Write entry mask to cache:

- VCAP_MASK_DAT[1][19:0] = entry mask[100:81]
- VCAP_MASK_DAT[0][31:1] = entry mask[80:50]
- VCAP_MASK_DAT[0][0] = 0, because type-group field must not be don't-cared.

Write action to cache:

- VCAP_ACTION_DAT[3][13:0] = action[214:201]
- VCAP_ACTION_DAT[2][31:0] = action[200:169]
- VCAP_ACTION_DAT[1][31:0] = action[168:137]
- VCAP_ACTION_DAT[0][31:2] = action[136:107]
- VCAP_ACTION_DAT[0][1:0] = 0, because type-group field for the X3 CLM action at address offset 1 is 0b00.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(4,483<<3)).

4.10.4.4 Writing Address Offset 0 of X3 Rule

Wait for VCAP_UPDATE_CTRL.UPDATE_SHOT to clear.

Write entry data to cache:

- VCAP_ENTRY_DAT[1][19:0] = entry data[49:30]
- VCAP_ENTRY_DAT[0][31:2] = entry data[29:0]
- VCAP_ENTRY_DAT[0][1:0] = 2, because type-group field for the X2 CLM entry at address offset 0 is 0b10.

Write entry mask to cache:

- VCAP_MASK_DAT[1][19:0] = entry mask[49:30]
- VCAP_MASK_DAT[0][31:2] = entry mask[29:0]
- VCAP_MASK_DAT[0][1:0] = 0, because type-group field must not be don't-cared.

Write action to cache:

- VCAP_ACTION_DAT[3][13:0] = action[106:93]
- VCAP_ACTION_DAT[2][31:0] = action[92:61]
- VCAP_ACTION_DAT[1][31:0] = action[60:29]
- VCAP_ACTION_DAT[0][31:3] = action[28:0]
- VCAP_ACTION_DAT[0][2:0] = 4, because type-group field for the X3 CLM action at address offset 0 is 0b100.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(4,482<<3)).

Once VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared, then the rule has been completely written to memory and will be able to match TRI_VID keys applied to the CLM1 VCAP.

4.10.4.5 Writing an ES0 Rule

This example shows how to write an ES0 rule, consisting of ISDX entry and ES0 action, to the VCAP ES0.

The ES0 is not part of any resource sharing so it owns the entire address range 0-4095. In this example, the rule is written to addresses 1215.

Software is expected to track memory usage so that it only writes to VCAP memory which does not already contain active rules.

The ISDX entry is 51 bits wide and the action is 489 bits wide. For information about how to build a ISDX entry and ES0 action, see 4.28.5 VCAP_ES0 Lookup.

4.10.4.6 Writing a Rule to ES0

Software is not allowed to modify the cache while VCAP operation is ongoing. Always check if VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared before starting to modify the cache. If a previous operation is still ongoing, then wait for the UPDATE_SHOT to clear.

When a new rule is started, first clear cache's entry, action, and counter by setting VCAP_UPDATE_CTRL.CLEAR_CACHE = 1.

Write entry data to cache:

- VCAP_ENTRY_DAT[1][18:0] = entry data[50:32]
- VCAP_ENTRY_DAT[0][31:0] = entry data[31:0]

Write entry mask to cache:

- VCAP_MASK_DAT[1][18:0] = entry mask[50:32]
- VCAP_MASK_DAT[0][31:0] = entry mask[31:0]

Write action to cache:

- VCAP_ACTION_DAT[15][8:0] = action[488:480]
- VCAP_ACTION_DAT[14][31:0] = action[479:448]
- VCAP_ACTION_DAT[13][31:0] = action[447:416]
- VCAP_ACTION_DAT[12][31:0] = action[415:384]
- VCAP_ACTION_DAT[11][31:0] = action[383:352]
- VCAP_ACTION_DAT[10][31:0] = action[351:320]

- VCAP_ACTION_DAT[9][31:0] = action[319:288]
- VCAP_ACTION_DAT[8][31:0] = action[284:256]
- VCAP_ACTION_DAT[7][31:0] = action[255:224]
- VCAP_ACTION_DAT[6][31:0] = action[223:192]
- VCAP_ACTION_DAT[5][31:0] = action[191:160]
- VCAP_ACTION_DAT[4][31:0] = action[159:128]
- VCAP_ACTION_DAT[3][31:0] = action[127:96]
- VCAP_ACTION_DAT[2][31:0] = action[95:64]
- VCAP_ACTION_DAT[1][31:0] = action[63:32]
- VCAP_ACTION_DAT[0][31:0] = action[31:0]

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(1215<<3)).

After VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared, the rule is completely written to memory and will be able to match ISDX keys applied to the VCAP ES0.

4.11 Pipeline Points

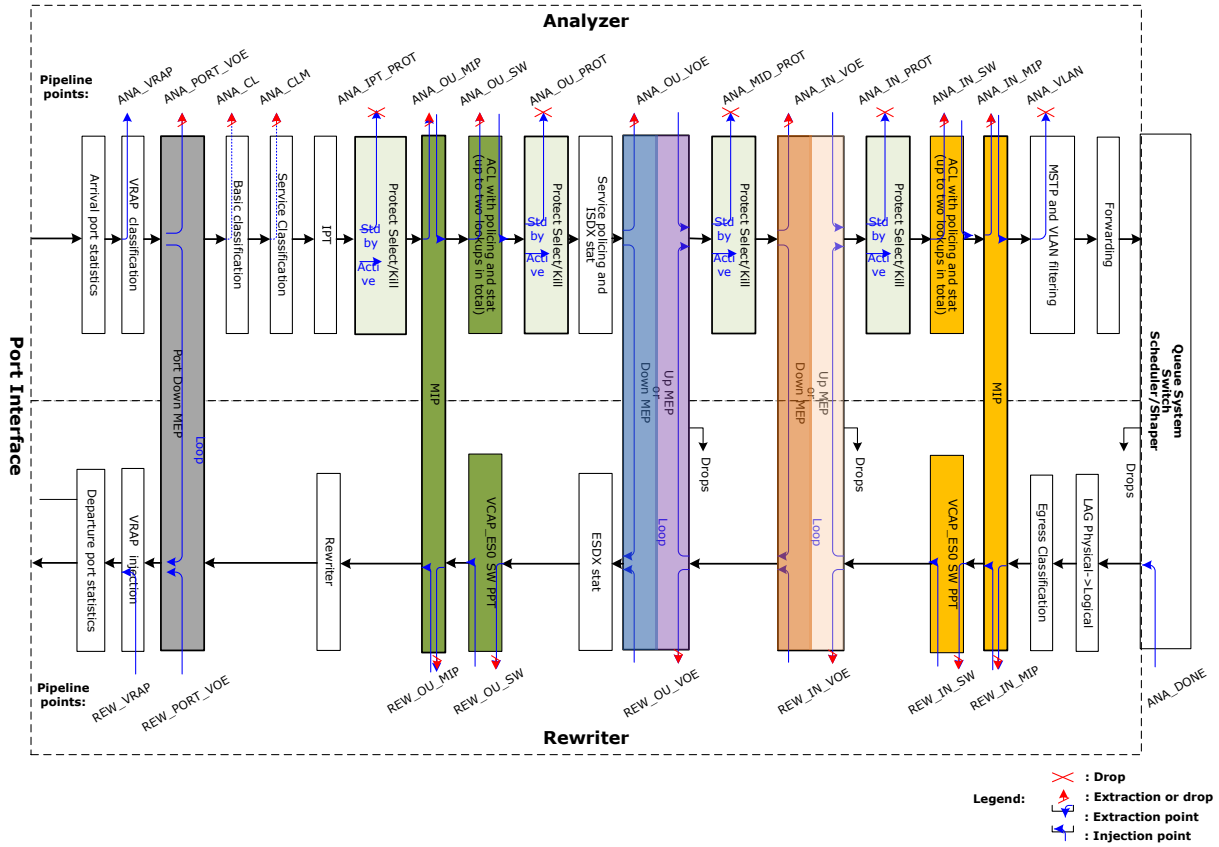
This section describes the use of pipeline points in the processing flow through the switch core. A pipeline point defines a specific position in the processing flow where frames can be injected, extracted, discarded, or looped. These actions are special, because they define either a starting point (inject) or an ending point (extract, discard, loop) in the processing flow. A starting point identifies the point in the processing flow where the processing of the frame begins. Similarly, an ending point identifies the point in the processing flow where the processing of the frame ends.

Frames subject to one of the actions injection, extraction, discarding, or looping, are assigned the associated pipeline point as well as a pipeline action defining what caused the pipeline point to be triggered. All frames are physically passed through all blocks of the processing flow, but each frame's pipeline point control determines whether it is actively processed in a given block or passed transparently to the next block; this includes which counters and policers are active.

The logical processing flow for a frame is shown in the following illustration. As an example of how the pipeline points affect the processing flow, consider a frame being discarded by the VLAN acceptance filter in the basic classification. The frame is assigned the ANA_CL pipeline point, and it is therefore not service classified and hence not service policed nor counted in the service statistics. For information about how the pipeline points affect policers and statistics, see 4.22.2 Policing.

The processing flow illustrates the different functional blocks and their pipeline points.

Figure 4-24. Processing Flow



4.11.1 Pipeline Definitions

The IFH fields IFH.MISC.PIPELINE_PT and IFH.MISC.PIPELINE_ACT carry the frame's pipeline information (point and action). By default, no pipeline points are triggered and both fields are set to 0. This implies that there are no restrictions on the processing of the frame.

The following table defines the available pipeline points in the analyzer and the rewriter.

Table 4-88. Pipeline Definitions

Pipeline Point	Block/Mechanism That Uses Pipeline Point	Application
ANA_VRAP	VRAP	VRAP extraction point.
ANA_PORT_VOE	Port VOE	Port VOE extraction/discard point.
ANA_CL	Basic classifier	Filter discard point/CPU injection point with software configured IFH.
ANA_CLM	VCAP CLM/service classification	CPU injection point with software configured IFH including service classification. Extraction point related to services.
ANA_IPT_PROT	IPT protection point	Protection discard point for e.g. LAG protection (where protection discard is done before Down MEPs) controlled by port VOE.

.....continued

Pipeline Point	Block/Mechanism That Uses Pipeline Point	Application
ANA_OU_MIP	Outer OAM half-MIP located in analyzer	Extraction/copy/injection point for analyzer half-MIP for: • UNI-N: Subscriber MIP • E-NNI: EVC MIP
ANA_OU_SW	Outer software MEP with extraction controlled by VCAP IS2	SW MEP located between the VOEs and the port for: • Extraction point for SW Down-MEP • Injection point for SW Up-MEP
ANA_OU_PROT	Outer protection point controlled by IPT	Protection discard point controlled by IPT for NNI protection
ANA_OU_VOE	Outer OAM VOE located in analyzer/VOP	Outer VOE pipeline point for: • UNI: EVC OAM injection • E-NNI: OVC OAM Injection • NNI: Path OAM Extraction
ANA_MID_PROT	Middle protection point located between outer and inner VOE controlled by IPT	Protection discard point for path protection controlled by path MEP.
ANA_IN_VOE	Inner OAM VOE located in analyzer/VOP	Inner VOE pipeline point for: • UNI: OVC OAM injection • NNI: EVC/OVC OAM segment extraction
ANA_IN_PROT	Inner protection point located after inner VOE controlled by IPT	Protection discard point. Frames are discarded after the VOEs used for discarding which exit an inactive EVC Segment
ANA_IN_SW	Inner software MEP with extraction controlled by VCAP IS2	SW MEP located between the VOEs and shared queue system for: • Extraction point for SW Down-MEP • Injection point for SW Up-MEP
ANA_IN_MIP	Inner OAM half-MIP located in analyzer	Extraction/copy/injection point for analyzer half MIP for NNI: EVC/OVC MIP.
ANA_VLAN	Pipeline point associated with VLAN handling and forwarding	Pipeline point for VLAN discard.
ANA_DONE	Pipeline point after analyzer	Injection point where analyzer is bypassed. Used for injection directly into rewriter.
REW_IN_MIP	Inner OAM half-MIP located in rewriter	Extraction/copy/injection point for rewriter half MIP function for NNI: EVC/OVC MIP.
REW_IN_SW	Inner software MEP located in rewriter controlled by VCAP_ES0	SW MEP located between the shared queue system and the VOEs for: • Injection point for SW Down-MEP • Extraction point for SW Up-MEP
REW_IN_VOE	Inner OAM VOE located in rewriter/VOP	Inner VOE pipeline point for: • UNI: OVC OAM extraction • NNI: EVC/OVC OAM segment injection

.....continued		
Pipeline Point	**Block/Mechanism That Uses Pipeline Point**	**Application**
REW_OU_VOE	Outer OAM VOE in rewriter/VOP	Outer VOE pipeline point for: • UNI: EVC OAM extraction • E-NNI: OVC OAM extraction • NNI: Path OAM injection
REW_OU_SW	Outer software MEP located in rewriter controlled by VCAP_ES0	SW MEP located between the VOEs and the port for: • Injection point for SW Down-MEP • Extraction point for SW Up-MEP
REW_OU_MIP	Outer OAM half-MIP located in rewriter	Extraction/copy/injection point for rewriter half MIP function for: • UNI-N: Subscriber MIP • E-NNI: EVC MIP
REW_SAT	Service activation testing	SAT loop point.
REW_PORT_VOE	Port VOE in rewriter VOP	Port VOE injection point.
REW_VRAP	VRAP injection point	VRAP injection point.

The following table defines the pipeline actions associated with the pipeline point.

Table 4-89. Pipeline Actions

Pipeline Action	Description
INJ	Set when CPU-injecting a frame into the processing flow at the associated pipeline point. The processing starts at the associated pipeline point.
INJ_MASQ	Set when CPU-injecting a frame masqueraded into the processing flow at the associated pipeline point. The processing starts at the associated pipeline point.
XTR	Assigned to frames being redirected to the CPU. The processing ends at the associated pipeline point. Frames copied to the CPU are not assigned a pipeline point as the processing continues.
XTR_UPMEP	Assigned to frames being extracted to the CPU by an Up MEP (VOE). The processing ends at the associated pipeline point. Frames copied to the CPU are not assigned a pipeline point as the processing continues.
XTR_LATE_REW	Assigned to MPLS OAM frames by VCAP CLM to instruct the rewriter to terminate the processing of the frame. The processing ends at the associated pipeline point.
LBK_ASM	Assigned by Up MEPs (VOE) requiring the frame to be looped. This action implies that the processing in the rewriter is ended at the associated pipeline point and that the processing in the analyzer after looping the frame starts at the corresponding analyzer pipeline point.
LBK_QS	Assigned by Down MEPs (VOE) requiring the frame to be looped. This action implies that the processing in the analyzer is ended at the associated pipeline point and that the processing in the rewriter after looping the frame starts at the corresponding rewriter pipeline point.

Note:
A frame cannot be assigned both a starting point and an ending point at the same side of the queue system. For instance, a frame cannot be injected in the analyzer at ANA_CLM and then extracted at ANA_IN_SW. However, a frame can be injected in the analyzer and then extracted in the rewriter.

4.12 Analyzer

The following sections provides information about the functional aspects of the analyzer (ANA). The analyzer evokes different actions based on the contents of the frame fields. The analyzer is organized into the following blocks.

- VCAP CLM: VCAP Classification matching—keys and actions
- ANA_CL: Analyzer classifier
- ANA_L3: VLAN and MSTP
- VCAP LPM: VCAP longest prefix matching—keys and actions
- ANA_L3: IP routing
- VCAP IS2: VCAP Ingress Stage 2—keys and actions
- ANA_ACL: Access Control Lists
- ANA_L2: Analyzer Layer 2 forwarding and learning
- ANA_AC: Analyzer Actions—forwarding, policing, statistics

4.12.1 Initializing the Analyzer

Before the analyzer can be configured, the RAM-based configuration registers must be initialized by setting ANA_AC:RAM_CTRL:RAM_INIT.RAM_ENA and ANA_AC:RAM_CTRL:RAM_INIT.RAM_INIT to 1.

The ANA_AC:RAM_CTRL:RAM_INIT.RAM_INIT register is reset by hardware when initialization is complete.

For information about initializing VCAP CLM, VCAP IS2, VCAP IP6PFX, and VCAP LPM, see 4.10.1.4 Bringing Up VCAP.

4.13 VCAP CLM Keys and Actions

The VCAP CLM is part of the analyzer and enables frame classification using VCAP functionality. This section provides an overview of all available VCAP CLM keys, followed by information about each key and action.

For information about how to select which key to use and how the VCAP CLM interacts with other parts of the analyzer classifier, see 4.14.2 VCAP CLM Processing.

VCAP CLM supports a number of different keys that can be used for different purposes. The keys are of type X1, X2, X3, X6, or X12, depending on the number of words each key uses. The keys are grouped after size as shown in the following table.

Table 4-90. VCAP CLM Keys and Sizes

Key Name	Number of Words	Key Type
SGL_MLBS	1 word	X1 type
DBL_VID		
TRI_VID	2 words	X2 type
DBL_MLBS		
ETAG		

.....continued		
Key Name	**Number of Words**	**Key Type**
MLL	3 words	X3 type
TRI_MLBS		
PURE_5TUPLE_IP4		
CUSTOM_4		
LL_FULL	6 words	X6 type
NORMAL		
NORMAL_5TUPLE_IP4		
CUSTOM_2		
NORMAL_7TUPLE	12 words	X12 type
CUSTOM_1		

4.13.1 Keys Overview

The following table lists all VCAP CLM keys and fields and provides a quick overview of which fields are available in which keys. When programming an entry in VCAP CLM, the associated key fields listed must be programmed in the listed order with the first field in the table starting at bit 0 in the entry.

Table 4-91. VCAP CLM Key Overview

Field Name	Short Description	Size	SGL_MLBS	DBL_VID	TRI_VID	DBL_MLBS	ETAG	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2	NORMAL_7TUPLE	CUSTOM_1
X1_TYPE	X1 type (each entry uses 1 word)	1	x	x													
X2_TYPE	X2 type (each entry uses 2 words)	2			x	x	x										
X3_TYPE	X3 type (each entry uses 3 words)	2						x	x	x	x						
X6_TYPE	X6 type (each entry uses 6 words)	2										x	x	x	x		
X12_TYPE	X12 type (each entry uses 12 words)	1														x	x
FIRST	Set for first lookup	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
IGR_PORT	Ingress port number	7			x		x	x				x					
G_IDX_SEL	G_IDX selector	2	x	x	x	x			x	x	x		x	x	x	x	x
G_IDX	Generic index	12	x	x	x	x			x	x	x		x	x	x	x	x
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK	2											x	x		x	

.....continued

Field Name	Short Description	Size	SGL_MLBS	DBL_VID	TRI_VID	DBL_MLBS	ETAG	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2	NORMAL_7TUPLE	CUSTOM_1
IGR_PORT_MASK	Ingress port mask	65											x	x		x	
L2_MC	Multicast DMAC address	1											x	x		x	
L2_BC	Broadcast DMAC address	1											x	x		x	
ETAGGED	Set for frames containing an E-TAG	1					x										
GRP	Group bits	2					x										
ECID_EXT	ECID extension	8					x										
ECID_BASE	ECID base	12					x										
IGR_ECID_EXT	Ingress ECID extension	8					x										
IGR_ECID_BASE	Ingress ECID base	12					x										
VLAN_TAGS	Number of VLAN tags	3			x			x				x	x	x		x	
VLAN_TAG_TPID	VLAN TPID type	3		x													
TPID0	TPID identifier from first tag	3			x			x				x	x	x		x	
PCP0	PCP from first tag	3			x							x	x	x		x	
DEI0	DEI from first tag	1			x							x	x	x		x	
VID0	VID from first tag	12		x	x			x				x	x	x		x	
TPID1	TPID identifier from second tag	3			x			x				x	x	x		x	
PCP1	PCP from second tag	3			x							x	x	x		x	
DEI1	DEI from second tag	1			x							x	x	x		x	
VID1	VID from second tag	12		x	x			x				x	x	x		x	
TPID2	Tag protocol identifier from third tag	3			x							x	x	x		x	
PCP2	PCP from third tag	3			x							x	x	x		x	
DEI2	DEI from third tag	1			x							x	x	x		x	
VID2	VID from third tag	12			x							x	x	x		x	
DST_ENTRY	Set if destination entry	1											x				
L2_DMAC	Destination MAC address	48						x				x				x	

.....continued

Field Name	Short Description	Size	SGL_MLBS	DBL_VID	TRI_VID	DBL_MLBS	ETAG	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2	NORMAL_7TUPLE	CUSTOM_1
L2_SMAC	Source MAC address	48						x				x	x			x	
ETYPE_MPLS	EtherType identifier	2						x									
IP_MC	Multicast DIP address	1											x	x		x	
ETYPE_LEN	ETYPE encoded frame	1										x	x			x	
ETYPE	EtherType, overloaded for other frame types	16										x	x			x	
IP_SNAP	IP or SNAP frame	1										x	x			x	
IP4	IPv4 frame	1										x	x	x		x	
LBL0	Label from MPLS Label Stack entry 0	20	x			x			x								
TC0	TC bits from MPLS Label Stack entry 0	3	x			x			x								
SBIT0	S-bit from MPLS Label Stack entry 0	1	x			x			x								
TTL0_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 0	1	x			x			x								
LBL1	Label from entry 1	20				x			x								
TC1	TC bits from entry 1	3				x			x								
SBIT1	S-bit from entry 1	1				x			x								
TTL1_EXPIRY	Set if TTL<=1 for entry 1	1				x			x								
LBL2	Label from entry 2	20							x								
TC2	TC bits from entry 2	3							x								
SBIT2	S-bit from entry 2	1							x								
TTL2_EXPIRY	Set if TTL<=1 for entry 2	1							x								
RSV_LBL_VAL	Reserved label value	4	x			x			x								
CW_ACH	CW/ACH after label with S-bit set	32							x								
RSV_LBL_POS	Reserved label position	3	x			x			x								

.....continued

Field Name	Short Description	Size	SGL_MLBS	DBL_VID	TRI_VID	DBL_MLBS	ETAG	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2	NORMAL_7TUPLE	CUSTOM_1
L3_FRAGMENT_TYPE	Fragment type IPv4 frame	2								x		x	x	x		x	
L3_FRAG_INVLD_L4_LEN	L4 length is invalid for IPv4 fragment	1								x		x	x	x		x	
L3_OPTIONS	IPv4 frame with options	1								x		x	x	x		x	
L3_DSCP	Frame's DSCP value	6								x		x	x	x		x	
L3_IP4_DIP	Destination IP address	32								x		x		x			
L3_IP4_SIP	SIP address, overloaded for other frame types	32								x		x	x	x			
L3_IP6_DIP	Destination IP address	128														x	
L3_IP6_SIP	Source IP address	128														x	
L3_IP_PROTO	IP protocol / next header	8								x				x			
TCP_UDP	TCP/UDP frame	1										x	x	x		x	
TCP	TCP frame	1										x	x	x		x	
L4_SPORT	TCP/UDP source port	16										x	x			x	
L4_RNG	Range checker mask	8			x			x		x		x	x	x		x	
IP_PAYLOAD_5TUPLE	Payload bytes after IP header	32								x				x			
OAM_Y1731	Set if frame's EtherType = 0x8902	1		x	x												
OAM_MEL_FLAGS	Encoding of MD level/MEG level (MEL)	7		x	x												
CUSTOM1	64 bytes payload	512															x
CUSTOM2	32 bytes payload	256													x		
CUSTOM4	16 bytes payload	128									x						

4.13.2 VCAP CLM X1 Key Details

The X1 keys include an X1_TYPE field, which is used to tell the keys apart. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 4-92. VCAP CLM X1 Key Details

Field Name	Description	Size	SGL_MLBS	DBL_VID
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x
G_IDX_SEL	Mode selector for G_IDX: 0: Common default (0, LPORT, or MASQ port) 1: G_IDX 2: ISDX - Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX. 3: IRLEG - Set for frames from VD2.	2	x	x
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number. For frames from VD2, G_IDX is set to the frame's IRLEG, independently of CLM_GIDX_DEF_SEL.	12	x	x
VLAN_TAG_TPID	Tag types: 0: Untagged 2: Single C-tag 3: Single S-tag 4: Outer C-tag, inner C-tag 5: Outer C-tag, inner S-tag 6: Outer S-tag, inner C-tag 7: Outer S-tag, inner S-tag S-tags can use any S-tag TPID (0x88A8 or custom values).	3		x
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12		x
VID1	Frame's VID from second tag.	12		x
LBL0	MPLS Label Stack entry 0 - Label (Top Label).	20	x	
TC0	MPLS Label Stack entry 0 - TC bits (Top Label).	3	x	

.....continued

Field Name	Description	Size	SGL_MLBS	DBL_VID
SBIT0	MPLS Label Stack entry 0 - S-bit (Top Label).	1	x	
TTL0_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 0.	1	x	
RSV_LBL_VAL	Reserved label value. Only valid if RSV_LBL_POS > 0.	4	x	
RSV_LBL_POS	Reserved label position: 0: Reserved label at position 0 seen or skipped. 1: Reserved label at position 1 seen or skipped. 2: Reserved label at position 2 seen or skipped. 3: Reserved label at position 3 seen. 4: Reserved. 5: No reserved label seen. Label at top of stack is position 0, followed by 1, 2, and 3. In order for a reserved label to be skipped, either ANA_CL::MPLS_RSV_LBL_CFG[<label>].RSVD_LBL_SKIP_ENA or ANA_CL::MPLS_MISC_CFG.CLM_RSVD_LBL_SKIP_ENA[<clm idx>] must be set.	3	x	
OAM_Y1731	Set if frame's EtherType = 0x8902.	1		x

.....continued

Field Name	Description	Size	SGL_MLBS	DBL_VID
OAM_MEL_FLAGS	Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0x0000000 MEL=1: 0x0000001 MEL=2: 0x0000011 MEL=3: 0x0000111 MEL=4: 0x0001111 MEL=5: 0x0011111 MEL=6: 0x0111111 MEL=7: 0x1111111 Together with the mask, the following kinds of rules may be created: • Exact match. Fx. MEL=2: 0x0000011 • Below. Fx. MEL<=4: 0x000XXXX • Below. Fx. MEL<=4: 0x000XXXX • Above. Fx. MEL>=5: 0xXX11111 • Between. Fx. 3<= MEL<=5: 0x00XX111 Where, 'X' means don't care. For non-Y1731 OAM frames (OAM_Y1731 = 0), OAM_MEL_FLAGS encodes an EtherType identifier: 0: IPv4frame (EtherType = 0x0800) 1: IPv6 frame (EtherType = 0x86DD) 2: Downstream assigned label (EtherType = 0x8847) 3: Upstream assigned label (EtherType = 0x8848) 4: LCC/SNAP (EtherType < 0x0600) 5: (R)ARP (EtherType = 0x0806 or 0x8035) 6: Reserved 7: Other	6		x

4.13.3 VCAP CLM X2 Key Details

The X2 keys include an X2_TYPE field, which is used to tell the keys apart. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 4-93. VCAP CLM X2 Key Details

Field Name	Description	Size	TRI_VID	DBL_MLBS	ETAG
X2_TYPE	X2 type: 0: TRI_VID 1: DBL_MLBS 2: Reserved 3: ETAG	2	x	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x
IGR_PORT	Logical ingress port number retrieved from ANA_CL::PORT_ID_CFG.LPORT_NUM.	7	x		x
G_IDX_SEL	Mode selector for G_IDX: 0: Common default (0, LPORT, or MASQ port) 1: G_IDX 2: ISDX - Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX. 3: IRLEG - Set for frames from VD2.	2	x	x	
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL NXT_IDX Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12	x	x	
ETAGGED	Set for frames containing an E-TAG	1			x
GRP	Group bits	2			x
ECID_EXT	ECID extension	8			x
ECID_BASE	ECID base	12			x
IGR_ECID_EXT	Ingress ECID extension	8			x
IGR_ECID_BASE	Ingress ECID base	12			x

.....continued

Field Name	Description	Size	TRI_VID	DBL_MLBS	ETAG
VLAN_TAGS	Number of VLAN tags in frame: 0: Untagged 1: Single tagged 3: Double tagged 7: Triple tagged	3	x		
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x		
PCP0	By default PCP from frame but it is selectable per lookup in VCAP CLM whether to use the current classified PCP instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	3	x		
DEI0	By default DEI from frame but it is selectable per lookup in VCAP CLM whether to use the current classified DEI instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	1	x		
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x		
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x		
PCP1	Frame's PCP from second tag.	3	x		
DEI1	Frame's DEI from second tag.	1	x		
VID1	Frame's VID from second tag.	12	x		

.....continued

Field Name	Description	Size	TRI_VID	DBL_MLBS	ETAG
TPID2	Tag protocol identifier of the frame's third tag: 0: No third tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x		
PCP2	Frame's PCP from third tag.	3	x		
DEI2	Frame's DEI from third tag.	1	x		
VID2	Frame's VID from third tag.	12	x		
LBL0	MPLS Label Stack entry 0 - Label (Top Label).	20		x	
TC0	MPLS Label Stack entry 0 - TC bits (Top Label).	3		x	
SBIT0	MPLS Label Stack entry 0 - S-bit (Top Label).	1		x	
TTL0_EXPIRY	Set if TTL< = 1 for MPLS Label Stack entry 0.	1		x	
LBL1	MPLS Label Stack entry 1 - Label.	20		x	
TC1	MPLS Label Stack entry 1 - TC bits.	3		x	
SBIT1	MPLS Label Stack entry 1 - S-bit.	1		x	
TTL1_EXPIRY	Set if TTL< = 1 for MPLS Label Stack entry 1.	1		x	
RSV_LBL_VAL	Reserved label value. Only valid if RSV_LBL_POS > 0.	4		x	
RSV_LBL_POS	Reserved label position: 0: Reserved label at position 0 seen or skipped. 1: Reserved label at position 1 seen or skipped. 2: Reserved label at position 2 seen or skipped. 3: Reserved label at position 3 seen. 4: Reserved. 5: No reserved label seen. Label at top of stack is position 0, followed by 1, 2, and 3. In order for a reserved label to be skipped, either ANA_CL::MPLS_RSV_LBL_CFG[<label>].RSVD_LBL_SKIP_ENA or ANA_CL::MPLS_MISC_CFG.CLM_RSVD_LBL_SKIP_ENA[<clm idx>] must be set.	3		x	

.....continued

Field Name	Description	Size	TRI_VID	DBL_MLBS	ETAG
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, ETYPE, outer VID, or inner VID. Input into range checkers is taken from classified results (VID, DSCP) and frame (SPORT, DPORT, ETYPE, outer VID, inner VID). Range checkers are configurable per port.	8	x		
OAM_Y1731	Set if frame's EtherType = 0x8902.	1	x		
OAM_MEL_FLAGS	Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated. MEL=0: 0x0000000 MEL=1: 0x0000001 MEL=2: 0x0000011 MEL=3: 0x0000111 MEL=4: 0x0001111 MEL=5: 0x0011111 MEL=6: 0x0111111 MEL=7: 0x1111111 Together with the mask, the following kinds of rules may be created: • Exact match. Fx. MEL=2: 0x0000011 • Below. Fx. MEL<=4: 0x000XXXX • Above. Fx. MEL>=5: 0xXX11111 • Between. Fx. 3<= MEL<=5: 0x00XX111 Where, 'X' means do not care. For non-Y1731 OAM frames (OAM_Y1731 = 0), OAM_MEL_FLAGS encodes an EtherType identifier: 0: IPv4frame (EtherType = 0x0800) 1: IPv6 frame (EtherType = 0x86DD) 2: Downstream assigned label (EtherType = 0x8847) 3: Upstream assigned label (EtherType = 0x8848) 4: LCC/SNAP (EtherType < 0x0600) 5: (R)ARP (EtherType = 0x0806 or 0x8035) 6: Reserved 7: Other	7	x		

4.13.4 VCAP CLM X3 Key Details

The X3 keys include an X3_TYPE field, which is used to tell the keys apart. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 4-94. VCAP CLM X3 Key Details

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
X3_TYPE	X3 type: 0: MLL. 1: TRI_MLBS. 2: PURE_5TUPLE_IP4. 3: CUSTOM_4.	2	x	x	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x	x
IGR_PORT	Logical ingress port number retrieved from ANA_CL::PORT_ID_CFG.LPORT_NUM.	7	x			
G_IDX_SEL	Mode selector for G_IDX: 0: Common default (0, LPORT, or MASQ port) 1: G_IDX 2: ISDX - Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX. 3: IRLEG - Set for frames from VD2.	2		x	x	x
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12		x	x	x
VLAN_TAGS	Number of VLAN tags in frame: 0: Untagged 1: Single tagged 3: Double tagged 7: Triple tagged	3	x			

.....continued

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x			
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x			
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x			
VID1	Frame's VID from second tag.	12	x			
L2_DMAC	Destination MAC address.	48	x			
L2_SMAC	Source MAC address.	48	x			
ETYPE_MPLS	EtherType identifier: 0: Non-MPLS. 1: Downstream Assigned Label (EtherType = 0x8847). 2: Upstream Assigned Label (EtherType = 0x8848).	2	x			
LBL0	MPLS Label Stack entry 0 - Label (Top Label).	20		x		
TC0	MPLS Label Stack entry 0 - TC bits (Top Label).	3		x		
SBIT0	MPLS Label Stack entry 0 - S-bit (Top Label).	1		x		
TTL0_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 0.	1		x		
LBL1	MPLS Label Stack entry 1 - Label.	20		x		

.....continued

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
TC1	MPLS Label Stack entry 1 - TC bits.	3		x		
SBIT1	MPLS Label Stack entry 1 - S-bit.	1		x		
TTL1_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 1.	1		x		
LBL2	MPLS Label Stack entry 2 - Label.	20		x		
TC2	MPLS Label Stack entry 2 - TC bits.	3		x		
SBIT2	MPLS Label Stack entry 2 - S-bit.	1		x		
TTL2_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 2.	1		x		
RSV_LBL_VAL	Reserved label value. Only valid if RSV_LBL_POS > 0.	4		x		
CW_ACH	The 32 bits following the label with S-bit set.	32		x		
RSV_LBL_POS	Reserved label position: 0: Reserved label at position 0 seen or skipped. 1: Reserved label at position 1 seen or skipped. 2: Reserved label at position 2 seen or skipped. 3: Reserved label at position 3 seen. 4: Reserved. 5: No reserved label seen. Label at top of stack is position 0, followed by 1, 2, and 3. In order for a reserved label to be skipped, either ANA_CL::MPLS_RSV_LBL_CFG[<label>].RSVD_LBL_SKIP_ENA or ANA_CL::MPLS_MISC_CFG.CLM_RSVD_LBL_SKIP_ENA[<clm idx>] must be set.	3		x		

.....continued

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
L3_FRAGMENT_TYPE	L3 Fragmentation type: 0: No Fragments: MF==0 && FO==0 1: Initial Fragments: MF==1 && FO == 0 2: Suspicious Fragment: FO!=0 && FO <= FRAGMENT_OFFSET_THRES 3: Valid Follow-up Fragment: FO > FRAGMENT_OFFSET_THRES Where, FRAGMENT_OFFSET_THRES is programmed in ANA_CL::CLM_FRAGMENT_CFG.FRAGMENT_OFFSET_THRES. MF: More Fragments flag in IPv4 header FO: Fragments Offset in IPv4 header	2			x	
L3_FRAG_INVLD_L4_LEN	Set if frame's L4 length is less than ANA_CL:COMMON:CLM_FRAGMENT_CFG.L4_MIN_LEN.	1			x	
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1			x	
L3_DSCP	By default DSCP from frame. Per match, selectable to be current classified DSCP (ANA_CL::ADV_CL_CFG.USE_CL_DSCP_ENA).	6			x	
L3_IP4_DIP	IPv4 frames: Destination IPv4 address. IPv6 frames: Destination IPv6 address, bits 31:0.	32			x	

.....continued

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
L3_IP4_SIP	Overloaded field for different frame types: LLC frames (ETYPE_LEN = 0 and IP_SNAP = 0): L3_IP4_SIP = [CTRL, PAYLOAD[0:2]] SNAP frames (ETYPE_LEN = 0 and IP_SNAP=1): L3_IP4_SIP = [PID[2:0], PAYLOAD[0]] IPv4 frames (ETYPE_LEN=1, IP_SNAP=1, and IP4=1): L3_IP4_SIP = source IPv4 address IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, IP4=0): L3_IP4_SIP = source IPv6 address, bits 31:0 Other frames (ETYPE_LEN=1, IP_SNAP=0): L3_IP4_SIP = PAYLOAD[0:3].	32			x	
L3_IP_PROTO	IPv4 frames (IP4=1): IP protocol. IPv6 frames (IP4=0): Next header.	8			x	
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, ETYPE, outer VID, or inner VID. Input into range checkers is taken from classified results (VID, DSCP) and frame (SPORT, DPORT, ETYPE, outer VID, inner VID). Range checkers are configurable per port.	8	x		x	
IP_PAYLOAD_5TUPLE	Payload bytes after IP header. IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header.	32			x	
CUSTOM4	16 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, then payload is retrieved from a position following the Ethernet layer, for example, after DMAC, SMAC, 0-3 VLAN tags and EtherType.	128				x

4.13.5 VCAP CLM X6 Key Details

The X6 keys include an X6_TYPE field, which is used to tell the keys apart. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 4-95. VCAP CLM X6 Key Details

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
X6_TYPE	X6 type: 0: LL_FULL. 1: NORMAL. 2: NORMAL_5TUPLE_IP4. 3: CUSTOM_2.	2	x	x	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x	x
IGR_PORT	Logical ingress port number retrieved from ANA_CL::PORT_ID_CFG.LPORT_NUM.	7	x			
G_IDX_SEL	Mode selector for G_IDX: 0: Common default (0, LPORT, or MASQ port) 1: G_IDX 2: ISDX - Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX. 3: IRLEG - Set for frames from VD2.	2		x	x	x
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12		x	x	x

.....continued

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK. 0: DEFAULT. 1: LOOPBACK. 2: MASQUERADE. 3: CPU_VD. If a frame fulfills multiple of below criteria, then higher value of IGR_PORT_MASK_SEL takes precedence. DEFAULT: Each bit in the mask correspond to physical ingress port. LOOPBACK: Set for frames received from a loopback device. Each bit in the mask correspond to the physical port which the frame was looped on. MASQUERADE: Set for masqueraded frames if ANA_CL::CLM_MISC_CTRL.MASQ_IGR_MASK_ENA is set. A frame is masqueraded when IFH.MISC.PIPELINE_ACT == INJ_MASQ. The bit corresponding to the masqueraded port is set. CPU_VD: Set for the following frame types: 1: CPU injected frames and ANA_CL::CLM_MISC_CTRL.CPU_IGR_MASK_ENA == 1 2: Frames received from VD0 or VD1 and ANA_CL::CLM_MISC_CTRL.VD_IGR_MASK_ENA == 1 Bits 4:0: Bit set if physical src port is VD2, VD1, VD0, CPU1, or CPU0 Bits 5:11: Src port (possibly masqueraded) Bits 12:16: ifh.misc.pipeline_pt, bits 17:19: ifh.misc.pipeline_act Bits 20: ifh.fwd.afi_inj	2		x	x	
IGR_PORT_MASK	Ingress port mask. See IGR_PORT_MASK_SEL for details.	65		x	x	

.....continued

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
L2_MC	Set if frame's destination MAC address is a multicast address (bit 40 = 1).	1		x	x	
L2_BC	Set if frame's destination MAC address is the broadcast address (FF-FF-FF-FF-FF-FF).	1		x	x	
VLAN_TAGS	Number of VLAN tags in frame: 0: Untagged 1: Single tagged 3: Double tagged 7: Triple tagged	3	x	x	x	
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	x	x	
PCP0	By default PCP from frame but it is selectable per lookup in VCAP CLM whether to use the current classified PCP instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	3	x	x	x	
DEI0	By default DEI from frame but it is selectable per lookup in VCAP CLM whether to use the current classified DEI instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	1	x	x	x	
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x	x	x	

.....continued

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	x	x	
PCP1	Frame's PCP from second tag.	3	x	x	x	
DEI1	Frame's DEI from second tag.	1	x	x	x	
VID1	Frame's VID from second tag.	12	x	x	x	
TPID2	Tag protocol identifier of the frame's third tag: 0: No third tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	x	x	
PCP2	Frame's PCP from third tag.	3	x	x	x	
DEI2	Frame's DEI from third tag.	1	x	x	x	
VID2	Frame's VID from third tag.	12	x	x	x	
DST_ENTRY	Selects whether the frame's destination or source information is used for fields L2_SMAC and L3_IP4_SIP. If set, L2_SMAC contains the frame's destination MAC address and L3_IP4_SIP contains the frame's destination IP address. The setting is controlled by ANA_CL::ADV_CL_CFG and VCPA_CLM_action.NXT_KEY_TYPE.	1		x		
L2_DMAC	Destination MAC address.	48	x			
L2_SMAC	Source MAC address. Note that - optionally - L2_SMAC may contain the destination MAC address, see DST_ENTRY.	48	x	x		

.....continued

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
IP_MC	IPv4 frames: Set if frame's destination IP address is an IPv4 multicast address (0xE /4). IPv6 frames: Set if frame's destination IP address is an IPv6 multicast address (0xFF /8).	1		x	x	
ETYPE_LEN	Frame type flag indicating that the frame is EtherType encoded. Set if frame has EtherType >= 0x600.	1	x	x		
ETYPE	Overloaded field for different frame types: LLC frames (ETYPE_LEN=0 and IP_SNAP=0): ETYPE = [DSAP, SSAP] SNAP frames (ETYPE_LEN=0 and IP_SNAP=1): ETYPE = PID[4:3] from SNAP header TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, and TCP_UDP=1): ETYPE = TCP/UDP destination port Non-TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, and TCP_UDP=0): ETYPE = IP protocol Other frames (ETYPE_LEN=1 and IP_SNAP=0): ETYPE = Frame's EtherType.	16	x	x		
IP_SNAP	Frame type flag indicating that the frame is either an IP frame or a SNAP frame. IP frames (ETYPE_LEN=1): Set if (EtherType= 0x0800 and IP version = 4) or (ETYPE = 0x86DD and IP version = 6) SNAP frames (ETYPE_LEN=0): Set if LLC header contains AA-AA-03.	1	x	x		
IP4	Frame type flag indicating the frame is an IPv4 frame. Set if frame is IPv4 frame (EtherType = 0x800 and IP version = 4).	1	x	x	x	

.....continued

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
L3_FRAGMENT_TYPE	L3 Fragmentation type: 0: No Fragments: MF==0 && FO==0 1: Initial Fragments: MF==1 && FO == 0 2: Suspicious Fragment: FO!=0 && FO <= FRAGMENT_OFFSET_THRES 3: Valid Follow-up Fragment: FO > FRAGMENT_OFFSET_THRES where FRAGMENT_OFFSET_THRES is programmed in ANA_CL::CLM_FRAGMENT_CFG.FRAGMENT_OFFSET_THRES. MF: More Fragments flag in IPv4 header FO: Fragments Offset in IPv4 header	2	x	x	x	
L3_FRAG_INVLD_L4_LEN	Set if frame's L4 length is less than ANA_CL:COMMON:CLM_FRAGMENT_CFG.L4_MIN_LEN.	1	x	x	x	
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1	x	x	x	
L3_DSCP	By default DSCP from frame. Per match, selectable to be current classified DSCP (ANA_CL::ADV_CL_CFG.USE_CL_DSCP_ENA).	6	x	x	x	
L3_IP4_DIP	IPv4 frames: Destination IPv4 address. IPv6 frames: Destination IPv6 address, bits 31:0.	32	x		x	

.....continued

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
L3_IP4_SIP	Overloaded field for different frame types: LLC frames (ETYPE_LEN=0 and IP_SNAP=0): L3_IP4_SIP = [CTRL, PAYLOAD[0:2]] SNAP frames (ETYPE_LEN=0 and IP_SNAP=1): L3_IP4_SIP = [PID[2:0], PAYLOAD[0]] IPv4 frames (ETYPE_LEN=1, IP_SNAP=1, and IP4=1): L3_IP4_SIP = source IPv4 address IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, IP4=0): L3_IP4_SIP = source IPv6 address, bits 31:0 Other frames (ETYPE_LEN=1, IP_SNAP=0): L3_IP4_SIP = PAYLOAD[0:3] Note that, optionally, L3_IP4_SIP may contain the destination IP address for IP frames, see DST_ENTRY.	32	x	x	x	
L3_IP_PROTO	IPv4 frames (IP4=1): IP protocol. IPv6 frames (IP4=0): Next header.	8			x	
TCP_UDP	Frame type flag indicating the frame is a TCP or UDP frame. Set if frame is IPv4/IPv6 TCP or UDP frame (IP protocol/ next header equals 6 or 17). Overloading: Set to 1 for OAM Y.1731 frames.	1	x	x	x	
TCP	Frame type flag indicating the frame is a TCP frame. Set if frame is IPv4 TCP frame (IP protocol = 6) or IPv6 TCP frames (Next header = 6).	1	x	x	x	

.....continued

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
L4_SPORT	TCP/UDP source port. Overloading: OAM Y.1731 frames: L4_SPORT = OAM MEL flags, see the following. TCP_UDP is at the same time set to 1 to indicate the overloading. OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0b0000000. MEL=1: 0b0000001. MEL=2: 0b0000011. MEL=3: 0b0000111. MEL=4: 0b0001111. MEL=5: 0b0011111. MEL=6: 0b0111111. MEL=7: 0b1111111. Together with the mask, the following kinds of rules may be created: Exact match. Fx. MEL=2: 0b0000011. Below. Fx. MEL<=4: 0b000XXXX. Above. Fx. MEL>=5: 0bXX11111. Between. Fx. 3<= MEL<=5: 0b00XX111, Where, ‘X’ means don’t care.	16	x	x		
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, ETYPE, outer VID, or inner VID. Input into range checkers is taken from classified results (VID, DSCP) and frame (SPORT, DPORT, ETYPE, outer VID, inner VID). Range checkers are configurable per port.	8	x	x	x	
IP_PAYLOAD_5TUPLE	Payload bytes after IP header. IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header.	32			x	

.....continued						
Field Name	**Description**	**Size**	**LL_FULL**	**NORMAL**	**NORMAL_5TUPLE_IP4**	**CUSTOM_2**
CUSTOM2	32 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, then payload is retrieved from a position following the Ethernet layer; for example, after DMAC, SMAC, 0–3 VLAN tags and EtherType.	256				x

4.13.6 VCAP CLM X12 Key Details

The X12 keys include an X12_TYPE field, which is used to tell the keys apart. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 4-96. VCAP CLM X12 Key Details

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
X12_TYPE	X12 type. 0: NORMAL_7TUPLE. 1: CUSTOM_1.	1	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x
G_IDX_SEL	Mode selector for G_IDX: 0: Common default (0, LPORT, or MASQ port) 1: G_IDX 2: ISDX - Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX. 3: IRLEG - Set for frames from VD2.	2	x	x

.....continued

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12	x	x

.....continued

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK. 0: DEFAULT. 1: LOOPBACK. 2: MASQUERADE. 3: CPU_VD. If a frame fulfills multiple of below criteria, then higher value of IGR_PORT_MASK_SEL takes precedence. DEFAULT: Each bit in the mask correspond to physical ingress port. LOOPBACK: Set for frames received from a loopback device. Each bit in the mask correspond to the physical port which the frame was looped on. MASQUERADE: Set for masqueraded frames if ANA_CL::CLM_MISC_CTRL.MASQ_IGR_MASK_ENA is set. A frame is masqueraded when IFH.MISC.PIPELINE_ACT == INJ_MASQ. The bit corresponding to the masqueraded port is set. CPU_VD: Set for the following frame types: 1: CPU injected frames and ANA_CL::CLM_MISC_CTRL.CPU_IGR_MASK_ENA == 1 2: Frames received from VD0 or VD1 and ANA_CL::CLM_MISC_CTRL.VD_IGR_MASK_ENA == 1 Bits 4:0: Bit set if physical src port is VD2, VD1, VD0, CPU1, or CPU0 Bits 5:11: Src port (possibly masqueraded) Bits 12:16: ifh.misc.pipeline_pt, bits 17:19: ifh.misc.pipeline_act Bits 20: ifh.fwd.afi_inj	2	x	
IGR_PORT_MASK	Ingress port mask. See IGR_PORT_MASK_SEL for details.	65	x	
L2_MC	Set if frame's destination MAC address is a multicast address (bit 40 = 1).	1	x	

.....continued

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
L2_BC	Set if frame's destination MAC address is the broadcast address (FF-FF-FF-FF-FF-FF).	1	x	
VLAN_TAGS	Number of VLAN tags in frame: 0: Untagged 1: Single tagged 3: Double tagged 7: Triple tagged	3	x	
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	
PCP0	By default PCP from frame but it is selectable per lookup in VCAP CLM whether to use the current classified PCP instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	3	x	
DEI0	By default DEI from frame but it is selectable per lookup in VCAP CLM whether to use the current classified DEI instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	1	x	
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x	
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	
PCP1	Frame's PCP from second tag.	3	x	
DEI1	Frame's DEI from second tag.	1	x	

.....continued

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
VID1	Frame's VID from second tag.	12	x	
TPID2	Tag protocol identifier of the frame's third tag: 0: No third tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3	3	x	
PCP2	Frame's PCP from third tag.	3	x	
DEI2	Frame's DEI from third tag.	1	x	
VID2	Frame's VID from third tag.	12	x	
L2_DMAC	Destination MAC address.	48	x	
L2_SMAC	Source MAC address.	48	x	
IP_MC	IPv4 frames: Set if frame's destination IP address is an IPv4 multicast address (0xE /4). IPv6 frames: Set if frame's destination IP address is an IPv6 multicast address (0xFF /8).	1	x	
ETYPE_LEN	Frame type flag indicating that the frame is EtherType encoded. Set if frame has EtherType >= 0x600.	1	x	

.....continued

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
ETYPE	Overloaded field for different frame types: LLC frames (ETYPE_LEN=0 and IP_SNAP=0): ETYPE = [DSAP, SSAP] SNAP frames (ETYPE_LEN=0 and IP_SNAP=1): ETYPE = PID[4:3] from SNAP header. TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, and TCP_UDP=1): ETYPE = TCP/UDP destination port. Non-TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, and TCP_UDP=0): ETYPE = IP protocol. Other frames (ETYPE_LEN=1 and IP_SNAP=0): ETYPE = Frame's EtherType.	16	x	
IP_SNAP	Frame type flag indicating that the frame is either an IP frame or a SNAP frame. IP frames (ETYPE_LEN=1): Set if (EtherType= 0x0800 and IP version = 4) or (ETYPE = 0x86DD and IP version = 6). SNAP frames (ETYPE_LEN=0): Set if LLC header contains AA-AA-03.	1	x	
IP4	Frame type flag indicating the frame is an IPv4 frame. Set if frame is IPv4 frame (EtherType = 0x800 and IP version = 4).	1	x	
L3_FRAGMENT_TYPE	L3 Fragmentation type: 0: No Fragments: MF==0 && FO==0 1: Initial Fragments: MF==1 && FO == 0 2: Suspicious Fragment: FO!=0 && FO <= FRAGMENT_OFFSET_THRES 3: Valid Follow-up Fragment: FO > FRAGMENT_OFFSET_THRES Where, FRAGMENT_OFFSET_THRES is programmed in ANA_CL::CLM_FRAGMENT_CFG.FRAGMENT_OFFSET_THRES . MF: More Fragments flag in IPv4 header FO: Fragments Offset in IPv4 header	2	x	

.....continued

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
L3_FRAG_INVLD_L4_LEN	Set if frame's L4 length is less than ANA_CL:COMMON:CLM_FRAGMENT_CFG.L4_MIN_LEN.	1	x	
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1	x	
L3_DSCP	By default DSCP from frame. Per match, selectable to be current classified DSCP (ANA_CL::ADV_CL_CFG.USE_CL_DSCP_ENA).	6	x	
L3_IP6_DIP	Destination IP address. IPv4 frames (IP4=1): Bits 31:0: Destination IPv4 address.	128	x	
L3_IP6_SIP	Source IP address. IPv4 frames (IP4=1): Bits 31:0: Source IPv4 address.	128	x	
TCP_UDP	Frame type flag indicating the frame is a TCP or UDP frame. Set if frame is IPv4/IPv6 TCP or UDP frame (IP protocol/next header equals 6 or 17). Overloading: Set to 1 for OAM Y.1731 frames.	1	x	
TCP	Frame type flag indicating the frame is a TCP frame. Set if frame is IPv4 TCP frame (IP protocol = 6) or IPv6 TCP frames (Next header = 6).	1	x	

.....continued				
Field Name	**Description**	**Size**	**NORMAL_7TUPLE**	**CUSTOM_1**
L4_SPORT	TCP/UDP source port. Overloading: OAM Y.1731 frames: L4_SPORT = OAM MEL flags. TCP_UDP is at the same time set to 1 to indicate the overloading. OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0b0000000 MEL=1: 0b0000001 MEL=2: 0b0000011 MEL=3: 0b0000111 MEL=4: 0b0001111 MEL=5: 0b0011111 MEL=6: 0b0111111 MEL=7: 0b1111111 Together with the mask, the following kinds of rules may be created: Exact match. Fx. MEL=2: 0b0000011 Below. Fx. MEL<=4: 0b000XXXX Above. Fx. MEL>=5: 0bXX11111 Between. Fx. 3<= MEL<=5: 0b00XX111, Where, 'X' means don't care.	16	x	
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, ETYPE, outer VID, or inner VID. Input into range checkers is taken from classified results (VID, DSCP) and frame (SPORT, DPORT, ETYPE, outer VID, inner VID). Range checkers are configurable per port.	8	x	
CUSTOM1	64 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, payload is retrieved from a position following the Ethernet layer. For example, after DMAC, SMAC, 0-3 VLAN tags, and EtherType.	512		x

4.13.7 VCAP CLM Actions

VCAP CLM supports five different actions, which can be used for different purposes. The actions have different sizes and must be paired with the keys as listed in the following table.

Table 4-97. VCAP CLM Action Selection

Action Name	Number of Words	Action Type
MLBS_REDUCED	1 word	X1 type
CLASS_REDUCED		
CLASSIFICATION	2 words	X2 type
MLBS		
FULL	3 words	X3 type

Any VCAP CLM action can be used with any of VCAP CLM keys. However, if the action's type is larger than the associated key's type (for instance FULL action of type X3 with TRI_VID key of type X2), then the entry row in the VCAP cannot be fully utilized.

The following table provides details for all VCAP CLM actions. When programming an action in VCAP CLM, the associated action fields listed must be programmed in the listed order with the first field in the table starting at bit 0 in the action.

Table 4-98. VCAP CLM Actions

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
X1_TYPE	X1 type. 0: Action MLBS_REDUCED. 1: Action CLASS_REDUCED.	1	x	x			
X2_TYPE	X2 type. 0: Action MLBS. 1: Action CLASSIFICATION.	1			x	x	
DSCP_ENA	If set, use DSCP_VAL as classified DSCP value.	1				x	x
DSCP_VAL	See DSCP_ENA.	6				x	x
COSID_ENA	If set, use COSID_VAL as classified COSID value.	1	x	x	x	x	x
COSID_VAL	See COSID_ENA.	3	x	x	x	x	x
QOS_ENA	If set, use QOS_VAL as classified QoS class.	1	x	x	x	x	x
QOS_VAL	See QOS_ENA.	3	x	x	x	x	x
DP_ENA	If set, use DP_VAL as classified drop precedence level.	1	x	x	x	x	x
DP_VAL	See DP_ENA.	2	x	x	x	x	x
DEI_ENA	If set, use DEI_VAL as classified DEI value.	1				x	x
DEI_VAL	See DEI_ENA.	1				x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
PCP_ENA	If set, use PCP_VAL as classified PCP value.	1				x	x
PCP_VAL	See PCP_ENA.	3				x	x
MAP_LOOKUP_SEL	Selects which of the two QoS Mapping Table lookups that MAP_KEY and MAP_IDX are applied to. 0: No changes to the QoS Mapping Table lookup. That is, MAP_KEY and MAP_IDX are not used. 1: Update key type and index for QoS Mapping Table lookup #0. 2: Update key type and index for QoS Mapping Table lookup #1. 3: Reserved. MLBS_REDUCED: Use 8 bits from COSID_ENA (LSB), COSID_VAL, QOS_ENA, and QOS_VAL (MSB) as MAP_IDX. COSID and QOS class cannot be assigned at the same time. MAP_KEY is always 3 (TC).	2	x	x	x	x	x
MAP_KEY	Key type for QoS mapping table lookup. 0: DEI0, PCP0 (outer tag). 1: DEI1, PCP1 (middle tag). 2: DEI2, PCP2 (inner tag). 3: TC. If ANA_CL:MPLS_CFG.MPLS_SEL_TC_ONLY_ENA is set, TC bits are extracted from the label stack by the current CLM lookup to perform the QoS mapping table lookup. If cleared, the QoS mapping table lookup is done using IFH.ENCAP.MPLS_TC. 4: PCP0 (outer tag). 5: E-DEI, E-PCP (E-TAG). 6: DSCP if available, otherwise none (64 entries). 7: DSCP if available, otherwise DEI0, PCP0 (outer tag) if available using MAP_IDX+8, otherwise none (80 entries).	3		x	x	x	x
MAP_IDX	Index for QoS mapping table lookup. Index bits 11:3 into 2K mapping table. Bits 2:0 are always 0.	9			x	x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
XVID_ADD_REPLACE_SEL	Controls the classified VID: 0: VID_NONE: No action. 1: VID_ADD: New VID = old VID + VID_VAL. 2: VID_REPLACE: New VID = VID_VAL. 3: VID_FIRST_TAG: New VID = VID from frame's first tag (outer tag) if available, otherwise VID_VAL. 4: VID_SECOND_TAG: New VID = VID from frame's second tag (middle tag) if available, otherwise VID_VAL. 5: VID_THIRD_TAG: New VID = VID from frame's third tag (inner tag) if available, otherwise VID_VAL.	3		x	x	x	x
GVID_ADD_REPLACE_SEL	Controls the classified GVID: 0: VID_NONE: No action. 1: VID_ADD: New GVID = old GVID + VID_VAL[11:0]. 2: VID_REPLACE: New GVID = VID_VAL[11:0]. 3: VID_FIRST_TAG: New GVID = VID from frame's first tag (outer tag) if available, otherwise VID_VAL[11:0]. 4: VID_SECOND_TAG: New GVID = VID from frame's second tag (middle tag) if available, otherwise VID_VAL[11:0]. 5: VID_THIRD_TAG: New GVID = VID from frame's third tag (inner tag) if available, otherwise VID_VAL[11:0].	3		x	x	x	x
VID_VAL	Value used by XVID_ADD_REPLACE_SEL and GIVD_ADD_REPLACE_SEL.	13		x	x	x	x
VLAN_POP_CNT_ENA	If set, VLAN_POP_CNT is used.	1		x		x	x
VLAN_POP_CNT	VLAN pop count: 0, 1, 2, or 3 tags. Post-processing of VLAN_POP_CNT: If VLAN_POP_CNT exceeds the actual number of tags in the frame, it is reduced to match the number of VLAN tags in the frame.	2		x		x	x
VLAN_PUSH_CNT_ENA	If set, VLAN_PUSH_CNT is used.	1		x		x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
VLAN_PUSH_CNT	VLAN push count: 0, 1, 2, or 3 tags.	2		x		x	x
TPID_SEL	Controls which tag provides the TPID carried in the IFH: 0: No action. 1: Use TPID from outer tag. 2: Use TPID from middle tag. 3: Use TPID from inner tag.	2		x		x	x
VLAN_WAS_TAGGED	Controls the WAS_TAGGED setting forwarded to the rewriter. 0: No changes to WAS_TAGGED. 1: Set WAS_TAGGED to 0. 2: Set WAS_TAGGED to 1. 3: Reserved.	2		x		x	x
ISDX_ADD_REPLACE_SEL	Controls the classified ISDX. 0: New ISDX = old ISDX + ISDX_VAL. 1: New ISDX = ISDX_VAL.	1	x	x	x	x	x
ISDX_VAL	See ISDX_ADD_REPLACE_SEL.	12	x	x	x	x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
MASK_MODE	Controls the PORT_MASK use. 0: OR_DSTMASK: Or PORT_MASK with destination mask. 1: AND_VLANMASK: And PORT_MASK to VLAN mask. The actual ANDing with the VLAN mask is performed after the ANA_L3 block has determined the VLAN mask. 2: REPLACE_PGID: Replace PGID port mask from MAC table lookup by PORT_MASK. 3: REPLACE_ALL: Use PORT_MASK as final destination set replacing all other port masks. 4: REDIR_PGID: Redirect using PORT_MASK[7:0] as PGID table index. See PORT_MASK for extra configuration options. 5: OR_PGID_MASK: Or PORT_MASK with PGID port mask from MAC table lookup. 6: VSTAX: Allows control over VSTAX forwarding. 7: Not applicable. Note that for REDIR_PGID, REPLACE_ALL and VSTAX, the port mask becomes “sticky” and cannot be modified by subsequent processing steps. The CPU port is untouched by MASK_MODE.	3					x
PORT_MASK	Port mask. MASK_MODE=4 (REDIR_PGID): PORT_MASK[53]: CSD_PORT_MASK_ENA. If set, CSD_PORT_MASK is AND'ed with destination result. PORT_MASK[52]: SRC_PORT_MASK_ENA. If set, SRC_PORT_MASK is AND'ed with destination result. PORT_MASK[51]: AGGR_PORT_MASK_ENA. If set, AGGR_PORT_MASK is AND'ed with destination result. PORT_MASK[50]: VLAN_PORT_MASK_ENA. If set, VLAN_PORT_MASK is AND'ed with destination result. PORT_MASK[11:0]: PGID table index.	65					x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
RT_SEL	Controls routing. 0: No change to routing. 1: Enable routing. 2: Disable routing.	2				x	x
LPM_AFFIX_ENA	If set, LPM_AFFIX_VAL is used as affix in LPM table. Note that frames from VD2 carrying an IRLEG in their IFH (ENCAP_ID_RLEG_MODE = 2) are assigned a default affix by ANA_CL:VMID:LPM_AFFIX.	1				x	x
LPM_AFFIX_VAL	Index into LPM table. Zero means no AFFIX used in LPM lookup.	10				x	x
RLEG_DMAC_CHK_DIS	If set, the router leg DMAC check in ANA_L3 is skipped.	1				x	x
TTL_DECR_DIS	If set, TTL decrement is disabled for routed frames.	1				x	x
L3_MAC_UPDATE_DIS	If set, DMAC and SMAC updates are disabled for routed frames.	1				x	x
FWD_DIS	If set, forwarding of the frame to front ports is disabled. CPU extraction is still possible.	1	x	x	x	x	
CPU_ENA	Copy frame to specified CPU extraction queue.	1	x	x	x	x	x
CPU_Q	CPU extraction queue when frame is forwarded to CPU.	3	x	x	x	x	x
MIP_SEL	Controls the MIP selection. 0: No change in MIP selection. 1: Enable MIP. 2: Disable MIP. If enabled, the MIP_IDX is given by the IPT table. OAM_Y1731_SEL determines the number of VLAN tags required for processing a frame in the MIP.	2		x		x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
OAM_Y1731_SEL	Selects which frames are detected as OAM frames by MEP and MIP. 0: No change in detection. 1: Disable OAM. 2: Detect untagged OAM. 3: Detect single tagged OAM. 4: Detect double tagged OAM. 5: Detect triple tagged OAM. 6: Enables Y1731 OAM unconditionally. 7: Enables any tags and EtherType as OAM unconditionally.	3		x	x	x	x
OAM_TWAMP_ENA	Enable frame as a TWAMP OAM frame.	1			x	x	x
OAM_IP_BFD_ENA	Enable frame as a IP BFD OAM frame.	1			x	x	x
RSVD_LBL_VAL	Use when MPLS_OAM_TYPE = 2, 4, or 5. Configures the reserved label used as OAM label for this MEP or MIP. Only the lower 4 bits of the label value is configured. Normally this field should be set to 13 (GAL label value). If MPLS_OAM_TYPE is set to 2, 4, or 5 and action record does not include RSVD_LBL_VAL, then a value of 13 (GAL label value) is used.	4					x
TC_ENA	Use the label's TC as classified TC.	1	x				
TTL_ENA	Use the label's TTL as classified TTL.	1	x				
TC_LABEL	Selects which label provides the classified TC. 0: Use TC from LSE #0 (if available). 1: Use TC from LSE #1 (if available). 2: Use TC from LSE #2 (if available). 3: Use TC from LSE #3 (if available). 7: Do not update TC. LSE #0 is top LSE, followed by LSE #1, and so on.	3			x		x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
TTL_LABEL	Selects which label provides the classified TTL. 0: Use TTL from LSE #0 (if available). 1: Use TTL from LSE #1 (if available). 2: Use TTL from LSE #2 (if available). 3: Use TTL from LSE #3 (if available). 7: Do not update TTL. LSE #0 is top LSE, followed by LSE #1, and so on.	3			x		x
NUM_VLD_LABELS	Number of valid labels, without reserved labels. Position of “deepest” label to be processed. Numbering starts with 0 at top LSE. If reserved labels are “skipped”, then these are not counted in NUM_VLD_LABELS. Valid range: 0–-3. Use of NUM_VLD_LABELS is dependent on FWD_TYPE as follows: FWD_TYPE=0: NUM_VLD_LABELS is not used. FWD_TYPE=1: Position of PW label. FWD_TYPE=2: Position of label to be swapped. LSEs above the swap label are popped. FWD_TYPE=3: Position of deepest label, which is to be popped.	2			x		x
FWD_TYPE	Forwarding type. 0: NONE. 1: TERMINATE_PW - Terminate pseudo wire. 2: LBL_SWAP - Swap label. 3: LBL_POP - Pop Labels. 3: CTRL_PDU	3	x		x		x
	FWD_TYPE: NONE No MPLS termination, MPLS SWAP, or MPLS POP						

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
	FWD_TYPE: TERMINATE_PW Terminate pseudo wire. NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET must be set to move frame pointer to CW/ACH if present, and otherwise to the PDU that follows the MPLS label stack. NUM_VLD_LABELS must be set to the position of the PW label. NXT_TYPE_AFTER_OFFSET must be set to CW if PW uses CW, otherwise ETH. NXT_NORMALIZE must be set to 1 to have MPLS Link Layer, LSEs and CW stripped. MPLS_OAM_TYPE can be set to values 0-4 to disable/enable OAM PDU detection.						
	FWD_TYPE: LBL_SWAP Swap label at position NUM_VLD_LABELS. Any label above the swap label are popped. NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET must be set to move frame pointer to swap label. NUM_VLD_LABELS must be set to the position of the swap label. NXT_NORMALIZE must be set to 1 to have MPLS Link Layer and, possibly, labels stripped. NXT_TYPE_AFTER_OFFSET must be set to MPLS. OAM PDUs are not detected, though frames with reserved label values as well as frames with TTL expiry can be redirected to CPU.						

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
	FWD_TYPE: LBL_POP Pop labels. NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET must be set to move frame pointer past popped labels. NUM_VLD_LABELS must be set to the position of the deepest label to be popped. NXT_NORMALIZE must be set to 1. NXT_TYPE_AFTER_OFFSET should normally be set to MPLS. NXT_TYPE_AFTER_OFFSET may be set to CW when terminating LSP with IP frames for CPU processing. MPLS_OAM_TYPE can be set to 0, 5, or 6 to disable/enable OAM PDU detection.						

.....continued							
Field Name	**Description and Encoding**	**Size**	**MLBS_REDUCED**	**CLASS_REDUCED**	**MLBS**	**CLASSIFICATION**	**FULL**
	FWD_TYPE: CTRL_PDU MPLS_OAM_FLAVOR (MSB) and MPLS_OAM_TYPE (LSB) control PDU type (as encoded in IFH.ENCAP.PDU_TYPE): 0: NONE, 1: OAM_Y1731 2: OAM_MPLS_TP 3: OAM_TWAMP 4: OAM_IP_BFD 5: PTP, 6: IP4_UDP_PTP 7: IP6_UDP_PTP 8: Reserved 9: SAM_SEQ OAM_Y1731_SEL controls PDU_OFFSET: 0: No change to PDU offset. 1: ETH_PAYLOAD. 2: IP_PAYLOAD. 3: Non-normalized offset (PDU_OFFSET = differences between normalized and non normalized payload).						

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
MPLS_OAM_TYPE	When either MIP or MEP OAM detection is enabled (MPLS_MIP_ENA or MPLS_MEP_ENA), MPLS_OAM_TYPE configures in which control channel to look for OAM. For FWD_TYPE = 1 (TERMINATE_PW), MPLS_OAM_TYPE configures which Control Channel Type is used for forwarding PW OAM: 1: Detect Vccv1 OAM 2: Detect Vccv2 OAM 3: Detect Vccv3 OAM 4: Detect Vccv4 OAM For FWD_TYPE = 3 (LSP_POP), MPLS_OAM_TYPE configures which Control Channel Type is used for LSP OAM: 5: LSP OAM under GAL found at pos NUM_VLD_LABELS + 1 6: LSP OAM under GAL found at pos NUM_VLD_LABELS For FWD_TYPE = 4 (CTRL_PDU), MPLS_OAM_FLAVOR (MSB) and MPLS_OAM_TYPE (LSB) configure directly the control type and PDU position (with OAM_Y1731_SEL and NXT_NORM_W16_OFFSET/ NXT_NORM_W32_OFFSET) and bypasses post-processing: 1: OAM_Y1731 2: OAM_MPLS_TP 3: OAM_TWAMP 4: OAM_TWAMP 5: PTP 6: IP4_UDP_PTP 7: IP6_UDP_PTP 8: Reserved 9: SAM_SEQ	3	x		x		x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
MPLS_MEP_ENA	Enables detection of MPLS MEP. If enabled and if frame is not selected for MIP extraction, it is determined whether it is a valid frame OAM PDU according to the MPLS_OAM_TYPE.	1	x		x		x
MPLS_MIP_ENA	Enables detection of MPLS MIP. If enabled, the iTTL will be checked for expiry. In case of iTTL expiry, the frame is checked if it is a valid OAM PDU according to the MPLS_OAM_TYPE.	1	x		x		x
MPLS_OAM_FLAVOR	If OAM detection is enabled, this setting determines if the G-ACH/ACH selected by MPLS_OAM_TYPE is checked for Channel Type = 0x8902. This configuration applies only to MEP OAM detection. OAM MIP detection is extracted to SW based on from the configured Control Channel Type, regardless of the channel type, because all OAM MIP processing is done in software. 0: G8113_2 (BFD and so on). 1: G8113_1 (BHH).	1	x		x		x
MPLS_IP_CTRL_ENA	Enables IP directly under MPLS based on IP profiles.	1	x		x		x
CUSTOM_ACE_ENA	Controls S2 custom rule selection: Bit 0: Selects custom key to use (0: CUSTOM_1, 1: CUSTOM_2). Bit 1: Enables custom key for first lookup. Bit 2: Enables custom key for second lookup. Bit 3: Enables custom key for third lookup. Bit 4: Enables custom key for fourth lookup.	5					x
CUSTOM_ACE_OFFSET	Selects custom data extraction point: 0: Current position (after IFH.WORD16_POP_CNT). Only applicable to frame type ETH. 1: Link layer payload (after VLAN tags and EtherType for ETH, after control word for CW). 2: Link layer payload + 20 bytes. 3: Link layer payload + 40 bytes.	2					x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
PAG_OVERRIDE_MASK	Bits set in this mask override the previous PAG result with the PAG_VAL value from this action.	8			x	x	x
PAG_VAL	PAG is updated using the following bit mask operation: PAG = (PAG and ~PAG_OVERRIDE_MASK) | (PAG_VAL and PAG_OVERRIDE_MASK). PAG can be used to tie VCAP CLM lookups with VCAP IS2 lookups. The PAG default value is configurable per port in ANA_CL:PORT:PORT_ID_CFG.PAG_VAL.	8			x	x	x
S2_KEY_SEL_ENA	If set, use S2_KEY_SEL to select VCAP_IS2 keys. Overrules previous selections by CLM.	1			x	x	x
S2_KEY_SEL	Determines key selection in VCAP_IS2 (ANA_ACL:KEY_SEL).	6			x	x	x
INJ_MASQ_ENA	If set, INJ_MASQ_PORT is used as physical ingress port. Learning is disabled. Pipeline action is set to INJ_MASQ.	1				x	x
INJ_MASQ_PORT	Use this value for physical ingress port when INJ_MASQ_ENA is set.	7				x	x
LPORT_ENA	Controls if LPORT_NUM updates LPORT and IFH.SRC_PORT.	1		x		x	x
LPORT_NUM	Use this value for LPORT and IFH.SRC_PORT when LPORT_ENA is set.	7		x		x	x
MATCH_ID	Logical ID for the entry. If the frame is forwarded to the CPU, the MATCH_ID is extracted together with the frame. MATCH_ID is used together with MATCH_ID_MASK as follows: MATCH_ID = (MATCH_ID and ~MATCH_ID_MASK) | (MATCH_ID and MATCH_ID_MASK). The result is placed in IFH.CL_RSLT.	16					x
MATCH_ID_MASK	Bits set in this mask overrides the previous MATCH_ID with the MATCH_ID value from this action.	16					x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
PIPELINE_FORCE_ENA	If set, use PIPELINE_PT unconditionally. Overrule previous settings of pipeline point. Use the following pipeline actions. 0: No change to PIPELINE_PT or PIPELINE_ACT. 1: Change PIPELINE_PT and set PIPELINE_ACT=XTR. Change PIPELINE_PT and set PIPELINE_ACT=INJ (PIPELINE_ACT_SEL=0) or INJ_MASQ (PIPELINE_ACT_SEL=1) 3: Change PIPELINE_PT and set PIPELINE_ACT=LBK_QS (PIPELINE_ACT_SEL=0) or LBK_ASM (PIPELINE_ACT_SEL=1)	2	x	x	x	x	x
PIPELINE_ACT_SEL	Used by PIPELINE_FORCE_ENA to control PIPELINE_ACT, see PIPELINE_FORCE_ENA.	1	x	x	x	x	x
PIPELINE_PT_REDUCED	Pipeline point used if PIPELINE_FORCE_ENA is set. 0: ANA_CLM 1: ANA_OU_MIP 2: ANA_IN_MIP 3: ANA_PORT_VOE 4: ANA_OU_VOE 5: ANA_IN_VOE 6: REW_IN_VOE 7: REW_OU_VOE	3	x				

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
PIPELINE_PT	Pipeline point used if PIPELINE_FORCE_ENA is set: 0: NONE 1: ANA_VRAP 2: ANA_PORT_VOE 3: ANA_CL 4: ANA_CLM 5: ANA_IPT_PROT 6: ANA_OU_MIP 7: ANA_OU_SW 8: ANA_OU_PROT 9: ANA_OU_VOE 10: ANA_MID_PROT 11: ANA_IN_VOE 12: ANA_IN_PROT 13: ANA_IN_SW 14: ANA_IN_MIP 15: ANA_VLAN 16: ANA_DONE	5		x	x	x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
NXT_KEY_TYPE	Enforces specific key type for next VCAP CLM lookup. 0: No overruling of default key type selection. 1: MLL. 2: SGL_MLBS. 3: DBL_MLBS. 4: TRI_MLBS. 5: TRI_VID or TRI_VID_IDX, depending on configuration per VCAP CLM lookup in ANA_CL::CLM_KEY_CFG.CLM_TRI_VID_SEL. 6: LL_FULL 7: NORMAL (with normal SRC information). 8: NORMAL with DST information. This is a modified version of NORMAL key type, where: the frame's DMAC is used in the L2_SMAC key field. the frame's IPv4 DIP is used in the L3_IP4_SIP key field. 9: NORMAL_7TUPLE 10: NORMAL_5TUPLE_IP4 11: PURE_5TUPLE_IP4 12: CUSTOM1 13: CUSTOM2 14: CUSTOM4 15: DBL_VID_IDX 16: ETAG 17: CLMS_DONE Complete - disable all remaining VCAP CLM lookups.	5	x	x	x	x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
NXT_NORM_W32_ OFFSET	4-byte value to be added to frame pointer. If any reserved MPLS labels were “skipped” during key generation before VCAP CLM lookup, then these are not counted in NXT_NORM_W32_OFFSET, because frame pointer is automatically moved past “skipped” labels. If both NXT_NORM_W32_OFFSET and NXT_OFFSET_FROM_TYPE are specified, then frame pointer is first modified based on NXT_OFFSET_FROM_TYPE and then moved further based on NXT_NORM_W32_OFFSET..	2	x				
NXT_NORM_W16_ OFFSET	2-byte value to be added to frame pointer. If any reserved MPLS labels were skipped during key generation before VCAP CLM lookup, then these are not be counted in NXT_NORM_W16_OFFSET, because frame pointer is automatically moved past “skipped” labels. If both NXT_NORM_W16_OFFSET and NXT_OFFSET_FROM_TYPE are specified, then frame pointer is first modified based on NXT_OFFSET_FROM_TYPE and then moved further based on NXT_NORM_W16_OFFSET.	5			x	x	x
NXT_OFFSET_FRO M_ TYPE	Moves frame pointer depending on type of current protocol layer. 0: NONE. No frame pointer movement. 1: SKIP_ETH. Skip Ethernet layer. If frame type is ETH, then move frame pointer past DMAC, SMAC and 0-3 VLAN tags. 2: SKIP_IP_ETH - Skip IP layer and preceding Ethernet layer (if present). If frame type is ETH: Move frame pointer past DMAC, SMAC, 0–3 VLAN tags and IPv4/IPv6 header. If frame type is CW and IP version is 4 or 6: Move frame pointer past IPv4/IPv6 header. For IPv4 header frame pointer is also moved past any IPv4 header options. 3: RSV. Reserved.	2			x	x	x

.....continued

Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
NXT_TYPE_AFTER_OFFSET	Protocol layer (frame_type) at frame pointer position after update based on NXT_OFFSET_FROM_TYPE and NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET. Frame type is only changed if one of the following conditions is true: • Frame pointer is moved. • Frame_type going into VCAP CLM was DATA and NXT_KEY_TYPE != 0. 0: ETH. Frame pointer points to start of DMAC. 1: CW (IP/MPLS PW CW/ MPLS ACH). Frame pointer points to MPLS CW/ACH or IP version. 2: MPLS. Frame pointer points to MPLS label. 3: DATA. “Raw” data, such as unknown protocol type.	2	x		x	x	x
NXT_NORMALIZE	Instruct rewriter to strip all frame data up to current position of frame pointer. The stripped data is not used for forwarding. When a VCAP CLM action is processed, frame pointer is updated before NXT_NORMALIZE is applied. **Note:** The rewriter can strip a maximum of 42 bytes.	1	x		x	x	x

.....continued							
Field Name	Description and Encoding	Size	MLBS_REDUCED	CLASS_REDUCED	MLBS	CLASSIFICATION	FULL
NXT_IDX_CTRL	Controls the generation of the G_IDX used in the VCAP CLM next lookup: 0: TRANSPARENT. G_IDX of previous parser step is carried forward to next parser step. 1: REPLACE. Replace previous G_IDX value with NXT_IDX of this result. 2: ADD. Add NXT_IDX of this result to G_IDX of previous parser step. 3: ISDX.Use ISDX as G_IDX. 4: RPL_COND_ISDX. Replace previous G_IDX value with NXT_IDX of this result. Assign ISDX to ISDX_VAL for terminated data. 5: Use ISDX with NXT_IDX for non terminated data (not CPU redir and MEP traffic). 6: Replace ISDX with NXT_IDX. Use ISDX as G_IDX in next key for non terminated data. 7: Reserved.	3	x	x	x	x	x
NXT_IDX	Index used as part of key (field G_IDX) in the next lookup.	12	x	x	x	x	x

4.14 Analyzer Classifier

The analyzer classifier (ANA_CL) handles frame classification. This section provides information about the following tasks performed by ANA_CL.

- Basic classification. Initial classification including frame filtering, tag, and QoS classification.
- VCAP CLM processing. Advanced TCAM-based classification.
- QoS mappings. Mapping and translations of incoming QoS parameters.
- Conversation-sensitive frame collection for link aggregation (IEEE 802.1AX).

4.14.1 Basic Classifier

The analyzer classifier includes a common basic classifier that determines the following basic properties that affect the forwarding of each frame through the switch.

- VLAN tag extraction. Parses the frame's VLAN tags in accordance with known tag protocol identifier values.
- Redundancy tag extraction.
- E-TAG extraction.
- Pipeline point evaluation. For injected and looped frames, ensures ANA_CL is part of the frame's processing flow.
- Routing control. Evaluates if a frame can be routed in terms of VLAN tags.
- Frame acceptance filtering. Drops illegal frame types.
- QoS classification. Assigns one of eight QoS classes to the frame.

- Drop precedence (DP) classification. Assigns one of four drop precedence levels to the frame.
- DSCP classification. Assigns one of 64 DSCP values to the frame.
- Default COSID classification. Assigns one of eight COSID values to the frame.
- VLAN classification. Extracts tag information from the frame or use the port VLAN.
- Link aggregation code generation. Generates the link aggregation code.
- Layer 2 and Layer 3 control protocol handling. Determines CPU forwarding, CPU extraction queue number, and dedicated protocol QoS handling.
- Versatile Register Access Protocol (VRAP) handling. Extracts VRAP frames to VRAP engine.
- Logical port mapping. Maps physical ingress port to logical port used by analyzer.
- GRE checksum validation for IP-in-IP encapsulated frames with GRE header.

The outcome of the classifier is the basic classification result, which can be overruled by more intelligent frame processing with VCAP classification matching (VCAP CLM).

4.14.1.1 VLAN Tag Extraction

The following table lists the register associated with VLAN tag extraction.

Table 4-99. VLAN Tag Extraction Register

Register	Description	Replication
ANA_CL::VLAN_STAG_CFG	Ethertype for S-tags in addition to default value 0x88A8.	3

Up to five different VLAN tags are recognized by the device.

- Customer tags (C-tags), which use TPID 0x8100.
- Service tags (S-tags), which use TPID 0x88A8 (IEEE 802.1ad).
- Service tags (S-tags), which use a custom TPID programmed in ANA_CL::VLAN_STAG_CFG. Three different custom TPIDs are supported.

The device can parse and use information from up to three VLAN tags of any of the types previously described.

Various blocks in the device uses Layer 3 and Layer 4 information for classification and forwarding. Layer 3 and Layer 4 information can be extracted from a frame with up to three VLAN tags. Frames with more than three VLAN tags are always considered non-IP frames.

4.14.1.2 E-TAG Extraction

The following table lists the register associated with E-TAG extraction.

Table 4-100. E-Tag Extraction Register

Register	Description	Replication
ANA_CL::ETAG_CFG.ETAG_TPID_ENA	Enable parsing for E-TAGs (EtherType 0x893F).	None

The device can parse and use information from up to one E-TAG. The E-TAG must be the outer-most tag if present, following immediately after the source MAC address. The device can further parse up to three VLAN tags following the E-TAG.

When parsing an E-TAG, IFH.TAGGING.RTAG_ETAG_AVAIL is set. The contents of the E-TAG is available for matching against the ETAG key in VCAP CLM. See 4.13.1 Keys Overview for more details.

When E-TAG extraction is disabled, E-TAG frames are considered untagged and any VLAN tags following the E-TAG are not parsed.

Various blocks in the device uses Layer 3 and Layer 4 information for classification and forwarding. Layer 3 and Layer 4 information can be extracted from a frame with up to one E-TAG. Frames with more than one E-TAG are always considered non-IP frames.

E-TAG extraction cannot be enabled while redundancy tag extraction is enabled.

4.14.1.3 Redundancy Tag Extraction

The following table lists the register associated with redundancy tag extraction.

Table 4-101. Redundancy Tag Extraction Register

Register	Description	Replication
ANA_CL::RTAG_CFG.RTAG_TPID_ENA	Enable parsing for redundancy tags (EtherType 0xF1C1).	None

The device can parse and use information from up to one redundancy tag (R-TAG). The redundancy tag can be placed in any location replacing one of the up to three VLAN tags. The following redundancy and VLAN tag combinations are thus possible:

- Triple tagged frames:

  DMAC, SMAC, R-TAG, VLAN tag, VLAN tag, EtherType.

  DMAC, SMAC, VLAN tag, R-TAG, VLAN tag, EtherType.

  DMAC, SMAC, VLAN tag, VLAN tag, R-TAG, EtherType.

  DMAC, SMAC, VLAN tag, VLAN tag, VLAN tag, and EtherType.
- Double tagged frames:

  DMAC, SMAC, R-TAG, VLAN tag, EtherType.

  DMAC, SMAC, VLAN tag, R-TAG, EtherType.

  DMAC, SMAC, VLAN tag, and VLAN tag, EtherType.
- Single tagged frames:

  DMAC, SMAC, R-TAG, EtherType.

  DMAC, SMAC, VLAN tag, and EtherType.
- Untagged.

Note that if a frame contains an redundancy tag in addition to zero or more VLAN tags, then any VLAN tags are referenced as if the frame had no redundancy tag. For instance, the frame's outer VLAN tag is always the first VLAN tag encountered, also when it is behind a redundancy tag. This applies among other places to the VLAN acceptance filter, the VLAN classification, QoS mappings, and VCAP CLM keys.

When parsing a redundancy tag, IFH.TAGGING.RTAG_ETAG_AVAIL is set and IFH.TAGGING.SEQ_NO is to the value of the sequence number in the redundancy tag.

When redundancy tag extraction is disabled, the redundancy tag's TPID is considered a normal EtherType and any VLAN tags following the redundancy tag are not parsed.

Various blocks in the device uses Layer 3 and Layer 4 information for classification and forwarding. Layer 3 and Layer 4 information can be extracted from a frame with one redundancy tag and additional two VLAN tags. Frames with more than one redundancy tag are always considered non-IP frames.

Redundancy tag extraction cannot be enabled while E-TAG extraction is enabled.

4.14.1.4 Pipeline Point Evaluation

The basic classifier is active for all frames with no active pipeline information for frames injected before or at pipeline point ANA_CL and for frames extracted at or after pipeline point ANA_CL.

Frames looped by the rewriter (PIPELINE_ACT = LBK_ASM) have their REW pipeline point changed to an ANA pipeline point, which indicates the point in the analyzer flow in ANA where they appear again after being looped. The following pipeline point translation are handled.

- PIPELINE_PT = REW_PORT_VOE is changed to ANA_PORT_VOE
- PIPELINE_PT = REW_OU_VOE is changed to ANA_OU_VOE
- PIPELINE_PT = REW_IN_VOE is changed to ANA_IN_VOE
- PIPELINE_PT = REW_SAT is changed to ANA_CLM
- PIPELINE_PT = REW_OU_VOE is changed to ANA_OU_SW.

- PIPELINE_PT = REW_IN_SW is changed to ANA_IN_SW.

4.14.1.5 Routing Tag Control

The basic classifier determines which IP frames can be routed in terms of the VLAN tag types. If a frame can be routed, it is signaled to ANA_L3, where the routing decision is made. The following table lists the register associated with routing control.

Table 4-102. Routing Tag Control Register

Register	Description	Replication
ANA_CL:PORT:VLAN_TPID_CTRL.BASIC_TPID_AWARE_DIS	Configures valid TPIDs for routing. This configuration is shared with VLAN classification.	Per port
ANA_CL:PORT:VLAN_TPID_CTRL.RT_TAG_CTRL	Configures number of valid VLAN tags in routable frames.	Per port

In order for a frame to be routed, the frame's VLAN tags must all be valid, and routing must be enabled for the number of VLAN tags in the frame.

The settings in BASIC_TPID_AWARE_DIS define each of three processed VLAN tags of which TPID values are valid for routing. The settings in RT_TAG_CTRL define the number of valid VLAN tags that must be present in routable frames.

Example: If routing is enabled for single-tagged frames with TPID=0x8100, configure the following:

- BASIC_TPID_AWARE_DIS = 0x7FFE. This invalidates all TPIDs except TPID = 0x8100 for outer most VLAN tag.
- RT_TAG_CTRL = 0x2. This enables routing of frames with one accepted tag only.

In this example, if an untagged frame is received, it is not routable because routing is not enabled for untagged frames. If a single-tagged frame is received with for instance TPID=0x88A8 or a double-tagged frame is received with for instance TPID=0x8100 for outer tag and TPID=0x8100 for inner tag, they are not routable because not all their VLAN tags are valid.

Example: To configure routing on a VLAN unaware port where routing is enabled for untagged frames only, set RT_TAG_CTRL = 0x1 to enable routing of untagged frames only.

4.14.1.6 Frame Acceptance Filtering

The following table lists the registers associated with frame acceptance filtering.

Table 4-103. Frame Acceptance Filtering Registers

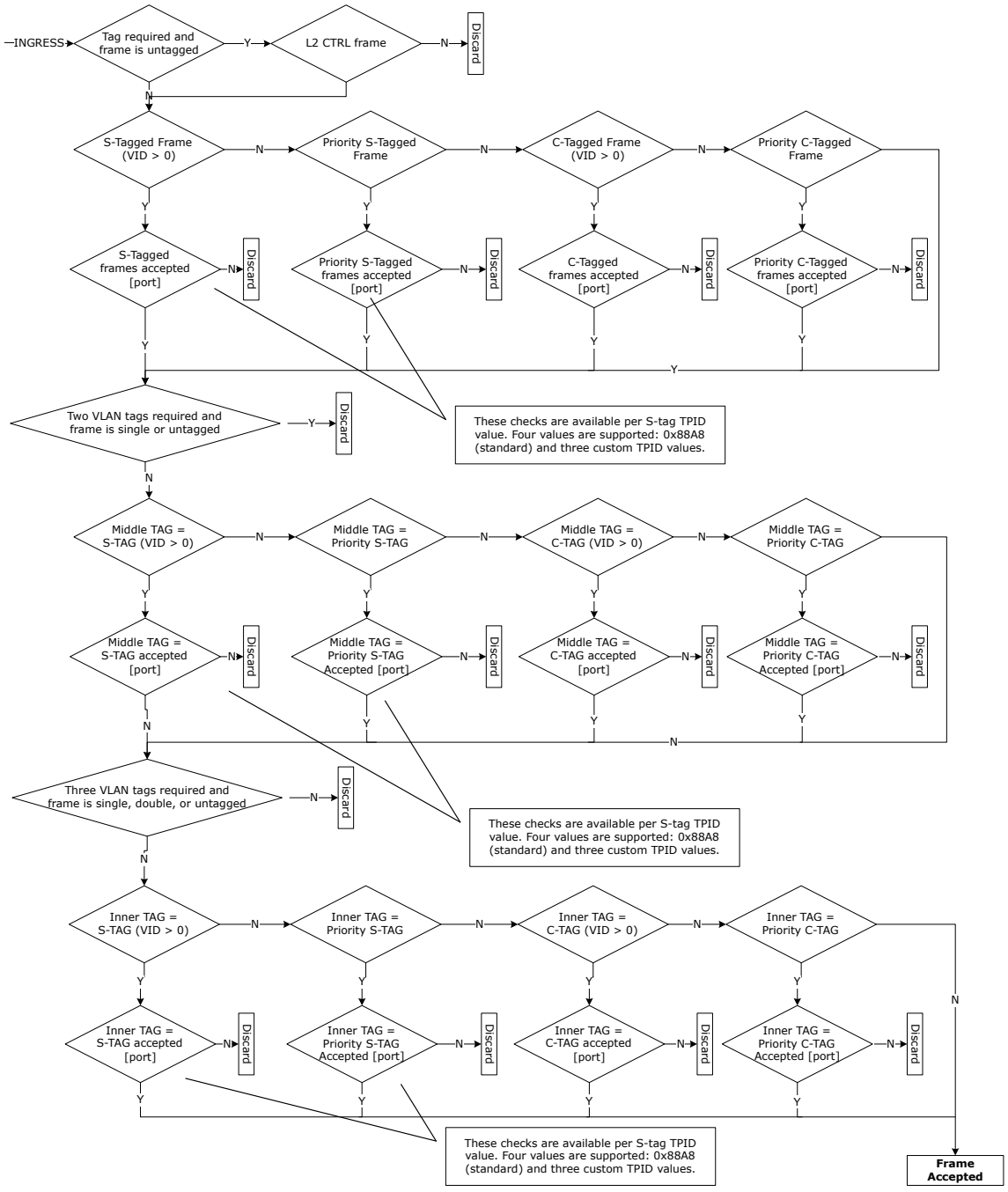
Register	Description	Replication
ANA_CL::FILTER_CTRL	Configures filtering of special frames	Per port
ANA_CL::VLAN_FILTER_CTRL	Configures VLAN acceptance filter	Three per port
ANA_CL::ETAG_FILTER_CTRL	Configures E-TAG acceptance filter	Per port

Based on the configurations in the FILTER_CTRL register, the classifier can discard certain frame types that are normally not allowed to enter the network, such as the following.

- Frames with a multicast source MAC address (bit 40 in address is set)
- Frames with a null source or null destination MAC address (address = 0x000000000000)

The VLAN acceptance filter decides whether a frame's VLAN tagging is allowed on the port. It has three stages where each stage checks one of the up to three processed VLAN tags. Each stage is programmable independently of the other stages in ANA_CL:PORT:VLAN_FILTER_CTRL. The following illustration shows the flowchart for the VLAN acceptance filter.

Figure 4-25. VLAN Acceptance Filter



The E-TAG acceptance filter decides whether a frame's E-TAG is allowed on the port. The following filter settings are possible.

- Accept all frames.
- Accept only frames without E-TAGs.
- Accept only frames with E-TAGs.

The classifier discards frames violating the E-TAG filter settings.

The VLAN and E-TAG acceptance filter do not apply to untagged Layer 2 control protocol frames. They are always accepted even when tags are required for other frames. The following frames are recognized as Layer 2 control protocol frames.

- Frames with DMAC = 0x0180C2000000 through 0x0180C200000F
- Frames with DMAC = 0x0180C2000020 through 0x0180C200002F
- Frames with DMAC = 0x0180C2000030 through 0x0180C200003F

Tagged Layer 2 control protocol frames are handled by the VLAN and E-TAG acceptance filters like normal tagged frames and they must comply with the filter settings to be accepted.

Frames discarded by the frame acceptance filters are assigned PIPELINE_PT = ANA_CL and learning of the frames' source MAC addresses are at the same time disabled.

4.14.1.7 Quality of Service Classification

This section provides information about the functions in the quality of service classification.

The following table lists the registers associated with quality of service classification.

Table 4-104. Quality of Service Classification Registers

Register	Description	Replication
ANA_CL:PORT:QOS_CFG	Configuration of the overall classification flow for QoS, DP, and DSCP.	Per port
ANA_CL:PORT:PCP_DEI_MAP_CFG	Port-based mapping of (DEI, PCP) to (DP, QoS).	Per port per DEI per PCP (16 per port)
ANA_CL::DSCP_CFG	DSCP configuration per DSCP value.	Per DSCP (64)
ANA_CL::QOS_MAP_CFG. DSCP_REWR_VAL	DSCP rewrite values per QoS class.	Per QoS class (8)
ANA_CL::CPU_8021_QOS_CFG	Configuration of QoS class for Layer 2 control protocol frames redirected to the CPU.	Per Layer 2 protocol address (32)
ANA_CL:VMID:VD2_QOS_CFG	Configuration of the overall classification flow for QoS, DP, and DSCP for frames from VD2.	Per IRLEG

The basic classification provides the user with control of the quality of service classification algorithms. The result of the basic classification are the following frame properties.

- The frame's QoS class:. This class is encoded in a 3-bit field, where 7 is the highest priority QoS class and 0 is the lowest priority QoS class. The QoS class is used by the queue system when enqueuing frames and when evaluating resource consumptions, for policing, statistics, and rewriter actions. The QoS class is carried internally in the IFH field IFH.VSTAX.QOS.CL_QOS.
- The frame's DP level: This level is encoded in a 2-bit field, where frames with DP = 3 have the highest probability of being dropped and frames with DP = 0 have the lowest probability. The DP level is used by the MEF compliant policers for measuring committed and peak information rates, for restricting memory consumptions in the queue system, for collecting statistics, and for rewriting priority information in the rewriter. The DP level is incremented by the policers if a frame is exceeding a programmed committed information rate. The DP level is carried internally in the IFH field IFH.VSTAX.QOS.CL_DP.
- The frame's DSCP: This value is encoded in a 6-bit fields. The DSCP value is forwarded with the frame to the rewriter where it is translated and rewritten into the frame. The DSCP value is only applicable to IPv4 and IPv6 frames. The DSCP is carried internally in the IFH field IFH.QOS.DSCP.
- The frame's COSID: This value is encoded in a 3-bit field. The COSID value is used as class of service identifier for services. It is used by the queue system, for policing, statistics, OAM, and rewriter actions. The COSID value is carried internally in the IFH field IFH.VSTAX.MISC.COSID.

The classifier looks for the following fields in the incoming frame to determine the QoS, DP, and DSCP classification.

- Priority Code Point (PCP) when the frame is VLAN tagged or priority tagged: There is an option to use the inner tag for double tagged frames (VLAN_CTRL.VLAN_TAG_SEL).

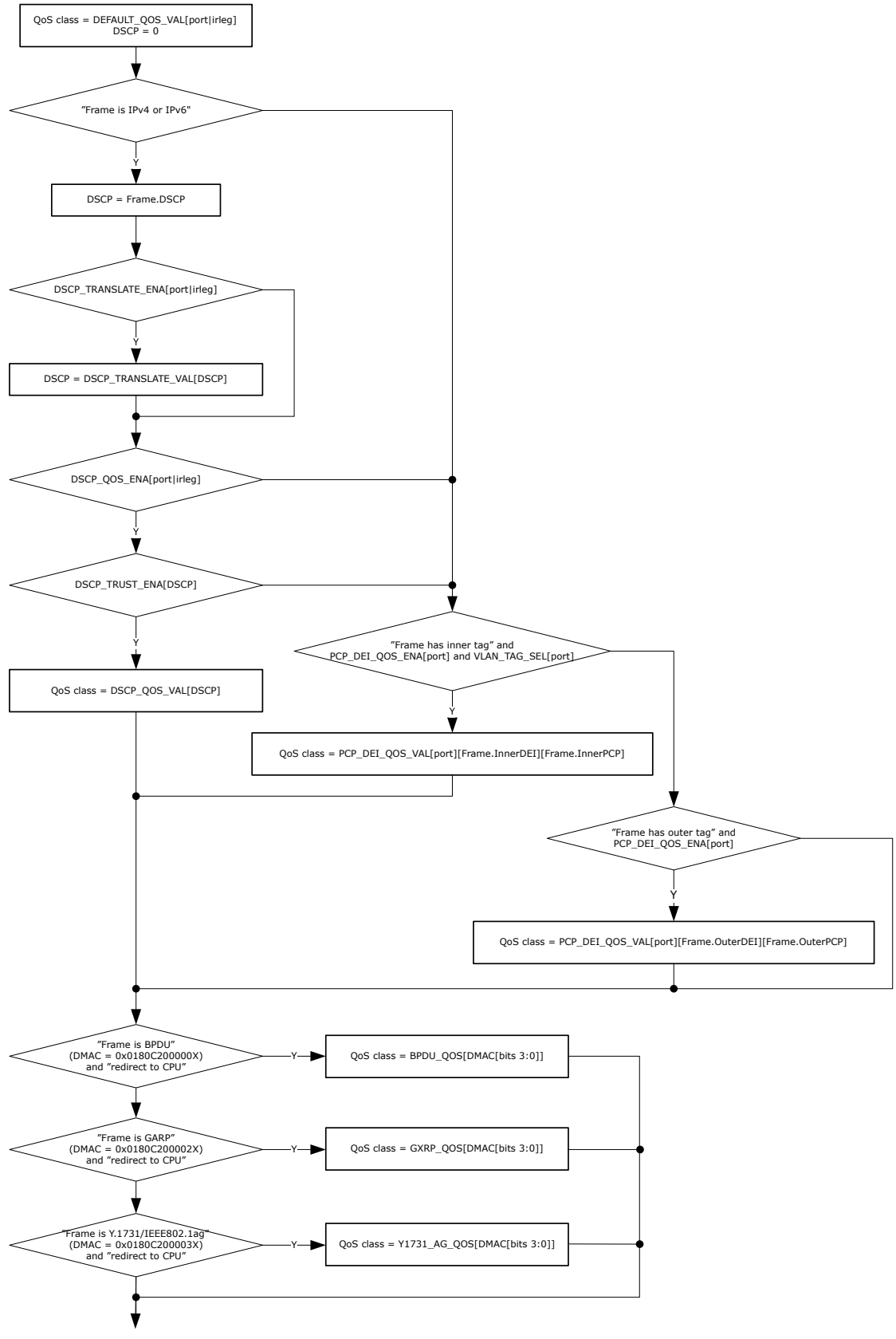
- Drop Eligible Indicator (DEI) when the frame is VLAN tagged or priority tagged: There is an option to use the inner tag for double tagged frames (VLAN_CTRL.VLAN_TAG_SEL).
- DSCP (all 6 bits, both for IPv4 and IPv6 packets): The classifier can look for the DSCP value behind up to three VLAN tags. For non-IP frames, the DSCP is 0 and it is not used elsewhere in the switch.
- Destination MAC address (DMAC) for L2CP frames: Layer 2 control protocol frames that are redirected to the CPU are given a programmable QoS class independently of other frame properties.

By default, frames from VD2 are subject to a normal quality of service classification using the port configuration associated with VD2. However, when a frame from VD2 carries an IRLEG in the IFH (IFH.ENCAP.ENCAP_ID_RLEG), then its IRLEG selects the classification parameters controlling the IPrelated parts of the classification algorithms instead of by the port. These parameters are controlled by the IRLEG:

- Default QoS class and QoS classification based on DSCP value
- Default drop precendence level and drop precedence classification based on DSCP value.
- DSCP translation.

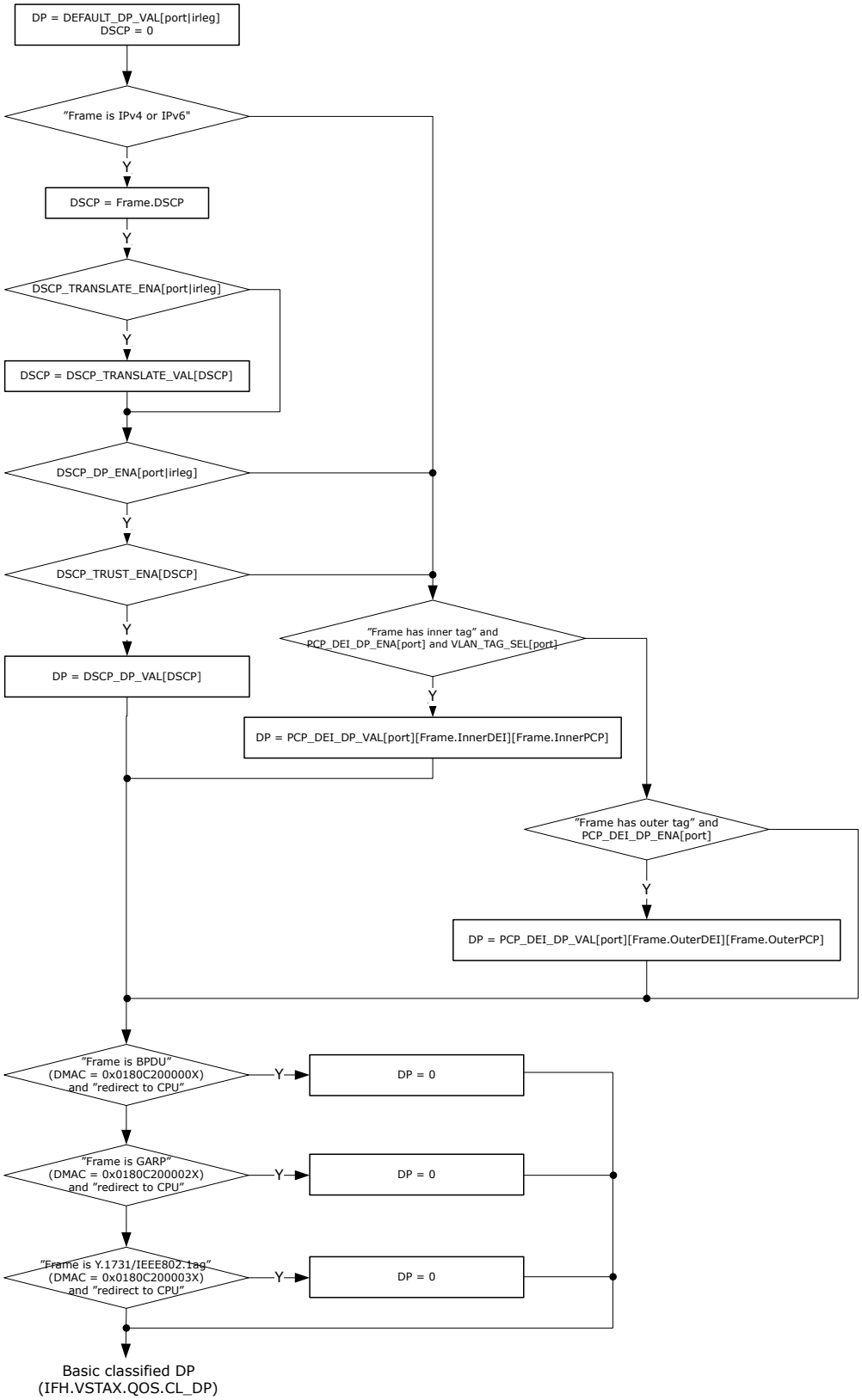
The following figure shows the flow chart of basic QoS classification.

Figure 4-26. Basic QoS Classification Flow Chart



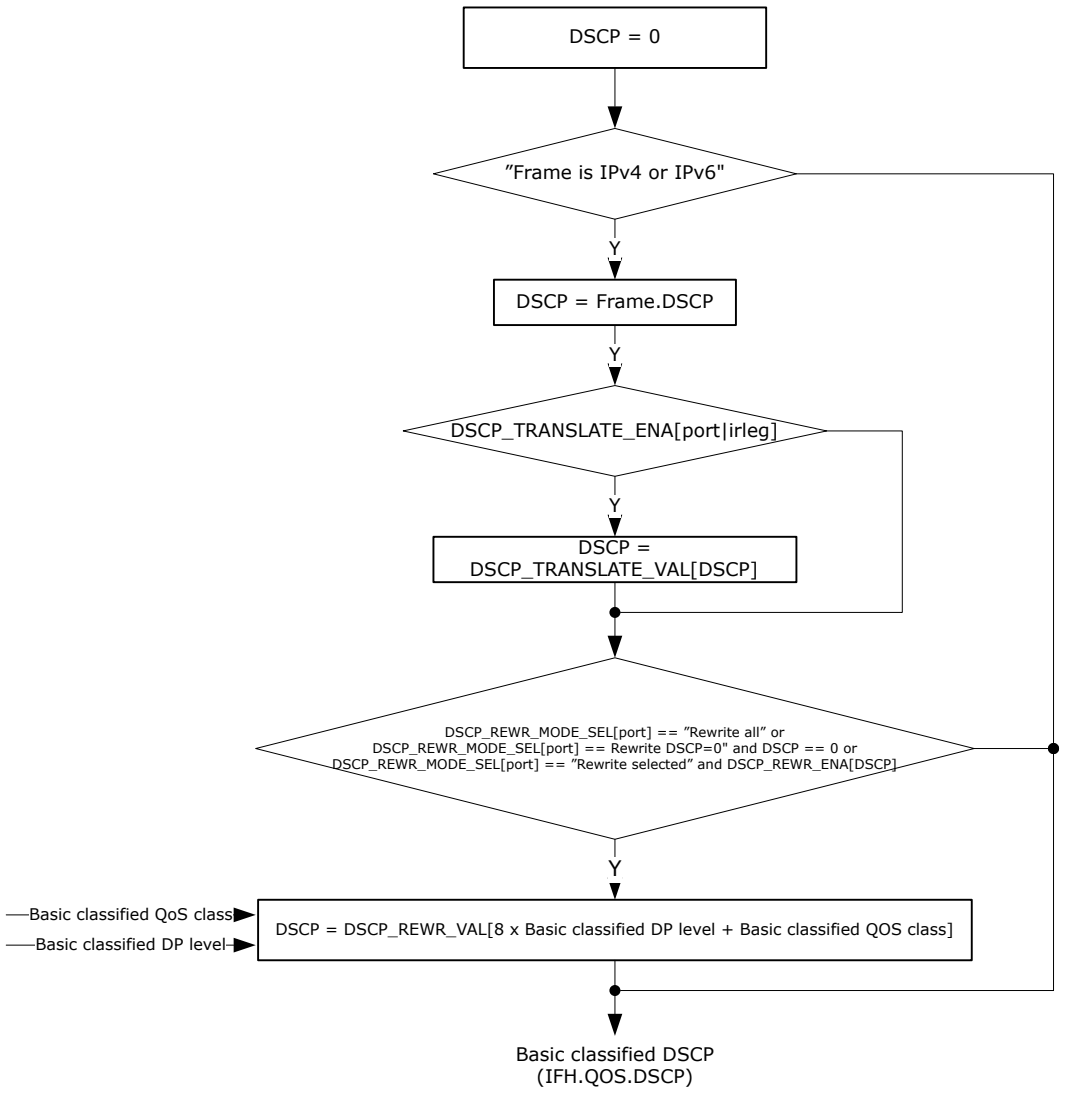
The following figure shows the flow chart for basic DP classification.

Figure 4-27. Basic DP Classification Flow Chart



The following figure shows the flow chart for basic DSCP classification.

Figure 4-28. Basic DSCP Classification Flow Chart



Note that the DSCP translation part is common for both QoS, DP, and DSCP classification, and the DSCP trust part is common for both QoS and DP classification.

The basic classifier has the option to overrule rewriter configuration (REW:PORT:DSCP_MAP) that remaps the DSCP value at egress. When ANA_CL:PORT:QOS_CFG.DSCP_KEEP_ENA is set, the rewriter is instructed not to modify the frame's DSCP value (IFH.QOS.TRANSP_DSCP is set to 1).

The basic COSID classification is by default zero but can optinally be assigned a different default value (ANA_CL:PORT:QOS_CFG.DEFAULT_COSID_VAL).

The basic classified QoS, DP, DSCP, and COSID can be overwritten by more intelligent decisions made in the VCAP CLM.

4.14.1.8 VLAN Classification

The following table lists the registers associated with VLAN classification.

Table 4-105. VLAN Configuration Registers

Register	Description	Replication
ANA_CL:PORT:VLAN_CTRL	Configures the port's processing of VLAN information in VLAN-tagged and priority-tagged frames. Configures the port-based VLAN.	Per port
ANA_CL:PORT:VLAN_CTRL2	Configures the port's VLAN push count.	Per port
ANA_CL::VLAN_STAG_CFG	Configures custom S-tag TPID value.	3
ANA_CL:PORT:PCP_DEI_TRANS_CFG	Port-based translation of (DEI, PCP) to basic classified (DEI, PCP).	Per port per DEI per PCP (16 per port)
ANA_CL:PORT:VLAN_TPID_CTRL.BASIC_TPID_AWARE_DIS	Configures valid TPIDs for VLAN classification. This configuration is shared with routing tag control.	Per port

The VLAN classification determines the following set of VLAN parameters for all frames.

- Priority Code Point (PCP). The PCP is carried internally in the IFH field IFH.VSTAX.TAG.CL_PCP.
- Drop Eligible Indicator (DEI). The DEI is carried internally in the IFH field IFH.VSTAX.TAG.CL_DEI.
- VLAN Identifier (VID). The VID is carried internally in the IFH field IFH.VSTAX.TAG.CL_VID.
- Tag Protocol Identifier (TPID) type, which holds information about the TPID of the tag used for classification. The TPID type is carried internally in IFH field IFH.VSTAX.TAG.TAG_TYPE. Extended information about the tag type is carried in IFH.DST.TAG_TPID.

The device recognizes five types of tags based on the TPID, which is the EtherType in front of the tag.

- Customer tags (C-tags), which use TPID 0x8100.
- Service tags (S-tags), which use TPID 0x88A8 (IEEE 802.1ad).
- Custom service tags (S-tags), which use one of three custom TPIDs programmed in ANA_CL::VLAN_STAG_CFG. Three different values are supported.

For customer tags and service tags, both VLAN tags (tags with nonzero VID) and priority tags (tags with VID = 0) are processed.

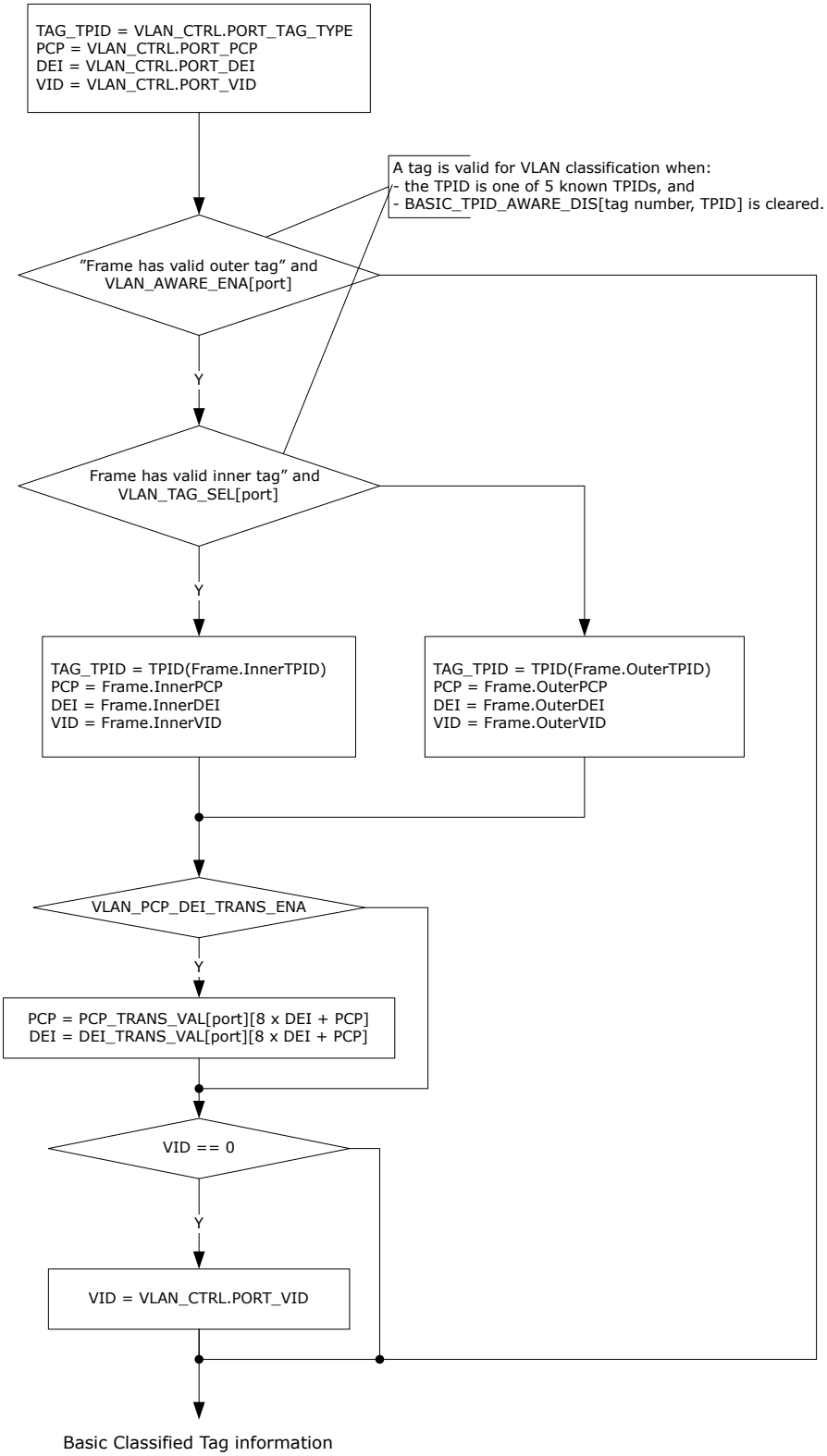
It is configurable which of the five recognized TPIDs are valid for the VLAN classification (VLAN_TPID_CTRL.BASIC_TPID_AWARE_DIS). Only VLAN tags with valid TPIDs are used by the VLAN classification. The parsing of VLAN tags stops when a VLAN tag with an invalid TPID is seen.

The VLAN tag information is either retrieved from a tag in the incoming frame or from default port-based configuration. The port-based VLAN tag information is configured in ANA_CL:PORT:VLAN_CTRL.

For double tagged frames (at least two VLAN tags with valid TPIDs), there is an option to use the inner tag instead of the outer tag (VLAN_CTRL.VLAN_TAG_SEL). Note that if the frame has more than two valid VLAN tags, the inner tag refers to the tag immediately after the outer tag.

The following figure shows the flow chart for basic VLAN classification.

Figure 4-29. Basic VLAN Classification Flow Chart



In addition to the classification shown, the port decides the number of VLAN tags to pop at egress (VLAN_CTRL2.VLAN_POP_CNT) and to push at egress (VLAN_CTRL2.VLAN_PUSH_CNT).

If the configured number of tags to pop is greater than the actual number of valid tags in the frame, the number is reduced to the number of actual valid tags in the frame. A maximum of three VLAN tags can be popped. The VLAN pop count is carried in IFH field IFH.TAGGING.POP_CNT.

A maximum of three VLAN tags can be pushed. The VLAN pop count is carried in IFH field IFH.TAGGING.PUSH_CNT.

Finally, the VLAN classification sets IFH field IFH.VSTAX.TAG.WAS_TAGGED if the port is VLAN aware and the frame has one or more valid VLAN tags. WAS_TAGGED is used the rewriter to make decision on whether tag information from existing tags can be used for rewriting.

The basic classified tag information can be overwritten by more intelligent decisions made in the VCAP CLM.

4.14.1.9 Link Aggregation Code Generation

This section provides information about the functions in link aggregation code generation.

The following table lists the registers associated with aggregation code generation.

Table 4-106. Aggregation Code Generation Registers

Register	Description	Replication
ANA_CL::AGGR_CFG	Configures use of Layer 2 through Layer 4 flow information for link aggregation code generation.	Common

The classifier generates a link aggregation code, which is used in the analyzer when selecting to which port in a link aggregation group a frame is forwarded.

The following contributions to the link aggregation code are configured in the AGGR_CFG register.

- Destination MAC address—use the 48 bits of the DMAC.
- Reversed destination MAC address—use the 48 bits of the DMAC in reverse order (bit 47 becomes bit 0, bit 46 becomes bit 1, and so on).
- Source MAC address—use the 48 bits of the SMAC.
- IPv6 flow label—use the 20 bits of the flow label.
- Source and destination IP addresses for IPv4 and IPv6 frames —use all bits of both the SIP and DIP.
- TCP/UDP source and destination ports for IPv4 and IPv6 frames use the 16 bits of both the SPORT and DPORT.
- Random aggregation code—use a 4-bit pseudo-random number.

All contributions that are enabled are 4-bit XOR'ed, and the resulting code is used by the Layer 2 forwarding function (ANA_AC) to select 1 out of 16 link aggregation port masks. The 16 link aggregation port masks are configured in ANA_AC:AGGR. For more information see 4.22.1.3 Aggregation Group Port Selection.

If AGGR_CFG.AGGR_USE_VSTAX_AC_ENA is enabled and the frame has a VStaX header, all other contributions to link aggregation code calculation are ignored, and the AC field of the VStaX-header is used directly. Otherwise, link aggregation code generation operates as previously described.

Note that AGGR_CFG.AGGR_DMAC_REVERSED_ENA and AGGR_CFG.AGGR_DMAC_ENA are independent. If both bits are enabled, the frame's DMAC contributes both normally and reversed in the link aggregation code calculation.

4.14.1.10 ECMP Aggregation Code Generation

The classifier generates an ECMP aggregation code for IPv4 and IPv6 frames. The following protocol fields contribute to the ECMP aggregation code.

- Source IP address for IPv4 and IPv6 frames—all bits contribute.
- IPv6 flow label—all bits contribute.
- TCP/UDP source and destination ports for TCP/UDP frames with no fragments and no IPv4 options—all bits contribute.

All contributions are 4-bit XOR'ed, and the resulting code is used by the Layer 3 routing function (ANA_L3).

4.14.1.11 CPU Forwarding Determination

The following table lists the registers associated with CPU forwarding in the basic classifier.

Table 4-107. CPU Forwarding Registers

Register	Description	Replication
ANA_CL:PORT:CAPTURE_CFG	Enables CPU forwarding for various frame types. Configures valid TPIDs for CPU forwarding.	Per port
ANA_CL:PORT:CAPTURE_BPDU_CFG	Enables CPU forwarding per BPDU address	Per port
ANA_CL:PORT:CAPTURE_GXRP_CFG	Enables CPU forwarding per GxRP address	Per port
ANA_CL:PORT:CAPTURE_Y1731_AG_CFG	Enables CPU forwarding per Y.1731/ IEEE802.1ag address	Per port
ANA_CL::CPU_PROTO_QU_CFG	CPU extraction queues for various frame types	None
ANA_CL::CPU_8021_QU_CFG	CPU extraction queues for BPDU, GARP, and Y.1731/IEEE802.1ag addresses	None
ANA_CL::VRAP_CFG	VLAN configuration of VRAP filter	None
ANA_CL::VRAP_HDR_DATA	Data match against VRAP header	None
ANA_CL::VRAP_HDR_MASK	Mask used to don’t care bits in the VRAP header	None
ANA_CL:VMID:VD2_CAPTURE_CFG	Enables CPU forwarding for various frame types for frames from VD2.	Per IRLEG

The basic classifier has support for determining whether certain frames must be forwarded to the CPU extraction queues. Other parts of the device can also determine CPU forwarding, for example, the VCAPs or the VOEs. All events leading to CPU forwarding are OR’ed together, and the final CPU extraction queue mask, which is available to the user, contains the sum of all events leading to CPU extraction.

Upon CPU forwarding by the basic classifier, the frame type and configuration determine whether the frame is redirected or copied to the CPU. Any frame type or event causing a redirection to the CPU causes all front ports to be removed from the forwarding decision—only the CPU receives the frame. When copying a frame to the CPU, the normal forwarding of the frame to front ports is unaffected.

The following table lists the standard frame types recognized by the basic classifier, with respect to CPU forwarding.

Table 4-108. Frame Type Definitions for CPU Forwarding

Frame	Condition	Copy/Redirect
BPDU Reserved Addresses (IEEE 802.1D 7.12.6)	DMAC = 0x0180C2000000 to 0x0180C20000F	Redirect/Copy/ Discard
GARP Application Addresses (IEEE 802.1D 12.5)	DMAC = 0x0180C2000020 to 0x0180C200002F	Redirect/Copy/ Discard
Y.1731/IEEE802.1ag Addresses	DMAC = 0x0180C2000030 to 0x0180C200003F	Redirect/Copy/ Discard
IGMP	DMAC = 0x01005E000000 to 0x01005E7FFFFF EtherType = IPv4 IP Protocol = IGMP	Redirect

.....continued

Frame	Condition	Copy/Redirect
MLD	DMAC = 0x333300000000 to 0x3333FFFFFFFF EtherType = IPv6 IPv6 Next Header = 0 (Hop-by-hop options header) Hop-by-hop options header with the first option being a Router Alert option with the MLD message (Option Type = 5, Opt Data Len = 2, Option Data = 0).	Redirect
Hop-by-hop	EtherType = IPv6 IPv6 Next Header = 0 (Hop-by-hop options header)	Redirect
ICMPv6	EtherType = IPv6 IPv6 Next Header = 58 (ICMPv6)	Redirect
IPv4 Multicast Ctrl	DMAC = 0x01005E000000 to 0x01005E7FFFFF EtherType = IPv4 IP protocol is not IGMP IPv4 DIP inside 224.0.0.x	Copy
IPv6 Multicast Ctrl	DMAC = 0x333300000000 to 0x3333FFFFFFFF EtherType = IPv6 IPv6 header is not hop-by-hop or ICMPv6 IPv6 DIP inside FF02::/16	Copy

Each frame type has a corresponding CPU extraction queue. Note that hop-by-hop and ICMPv6 frames share the same CPU extraction queue.

Prior to deciding whether to CPU forward a frame, the frame's VLAN tagging must comply with a configurable filter (ANA_CL::CAPTURE_CFG.CAPTURE_TPID_AWARE_DIS). For all frame types listed in Table 4-108, only untagged frames or VLAN-tagged frames where the outer VLAN tag's TPID is not disabled are considered. The VLAN TPID filter applies to all frames except frames received from VD2.

Each port controls the CPU forwarding of the frame types listed in Table 4-108. However, when a frame from VD2 carries an IRLEG in the IFH (IFH.ENCAP.ENCAP_ID_RLEG), then its IRLEG selects the parameters controlling the CPU forwarding of IP frames.

The basic classifier can recognize VRAP frames and redirect them to the CPU. This is a proprietary frame format, which is used for reading and writing switch configuration registers through Ethernet frames. For more information, see 4.31 VRAP Engine.

The VRAP filter in the classifier performs three checks in order to determine whether a frame is a VRAP frame:

1. VLAN check. The filter can be either VLAN unaware or VLAN aware (ANA_CL::VRAP_CFG.VRAP_VLAN_AWARE_ENA). If VLAN unaware, VRAP frames must be untagged. If VLAN aware, VRAP frames must be VLAN tagged and the frame's VID must match a configured value (ANA_CL::VRAP_CFG.VRAP_VID). Double VLAN tagged frames always fail this check.

2. EtherType and EPID check. The EtherType must be 0x8880 and the EPID (bytes 0 and 1 after the EtherType) must be 0x0004.
3. VRAP header check. The VRAP header (bytes 0, 1, 2, and 3 after the EPID) must match a 32-bit configured value (ANA_CL::VRAP_HDR_DATA) where any bits can be don't cared by a mask (ANA_CL::VRAP_HDR_MASK).

If all three checks are fulfilled, frames are redirected to CPU extraction queue ANA_CL::CPU_PROTO_QU_CFG.CPU_VRAP_QU. The VRAP filter is enabled in ANA_CL:PORT:CAPTURE_CFG.CPU_VRAP_REDIR_ENA.

4.14.1.12 Logical Port Mapping

The basic classifier works on the physical port number given by the interface where the frame was received. Other device blocks work on a logical port number that defines the link aggregation instance rather than the physical port. The logical port number is defined in ANA_CL:PORT:PORT_ID_CFG.LPORT_NUM. It is, for instance, used for learning and forwarding in ANA_L2.

4.14.1.13 GRE Checksum Validation

This section provides information about the functions in GRE checksum validation.

The following table lists the registers associated with GRE checksum validation.

Table 4-109. GRE Checksum Validation Registers

Register	Description	Replication
ANA_CL::GRE_MISC_CFG.GRE_CHKSUM_SKIP	Disables GRE checksum validation.	Common

The classifier identifies whether incoming frames include a GRE header (EtherType = IPv4 and IP protocol is GRE). If the GRE header includes a GRE checksum (Checksum Present bit set), then the frame data is validated against the GRE checksum. The result of the GRE checksum validation is forwarded to ANA_L3 for further action, see 4.18.5 IP-in-IPv4 for more details.

The GRE checksum validation can optionally be disabled.

4.14.2 VCAP CLM Processing

Each frame is matched six times against entries in the VCAP CLM. The matching is done in serial implying results from one match can influence the keys used in the next match.

The VCAP CLM returns an action for each enabled VCAP CLM lookup. If the lookup matches an entry in VCAP CLM, the associated action is returned. If an entry is not matched, a per-port default action is returned. There is no difference between an action from a match and a default action.

4.14.2.1 VCAP CLM Port Configuration and Key Selection

This section provides information about special port configurations that control the key generation for VCAP CLM.

The following table lists the registers associated with port configuration for VCAP CLM.

Table 4-110. Port Configuration of VCAP CLM

Register	Description	Replication
ANA_CL:PORT:ADV_CL_CFG	Configuration of the key selection for the VCAP CLM. Enables VCAP CLM lookups.	Per port per lookup
ANA_CL::CLM_MISC_CTRL	Force lookup in VCAP CLM for looped frames or frames where processing is terminated by a pipeline point.	None

VCAP CLM is looked up if the lookup is enabled by the port configuration (ADV_CL_CFG.LOOKUP_ENA) or if the previous VCAP CLM lookup has returned a key type for the next lookup through VCAP CLM action NXT_KEY_TYPE. However, if the previous lookup returns NXT_KEY_TYPE = CLMS_DONE, then no further lookups are done, even if the port has enabled the lookup.

Prior to each VCAP CLM lookup, the type of key to be used in the lookup is selected. A number of parameters control the key type selection:

- Configuration parameters (ADV_CL_CFG).
  For each (port, VCAP CLM lookup) a key type can be specified for different frame types. Although any key type can be specified for a given frame type, not all key types are in reality applicable to a given frame type. The applicable key types are listed in the ANA_CL:PORT:ADV_CL_CFG register description. The following frame types are supported.
  - IPv6 frames (EtherType 0x86DD and IP version 6)
  - IPv4 frames (EtherType 0x0800 and IP version 4)
  - MPLS unicast frames (EtherType 0x8847)
  - MPLS multicast frames (EtherType 0x8848)
  - MPLS label stack processing (frame type after MPLS link layer is processed)
  - Other frames (non-MPLS, non-IP)
- The type of the current protocol layer.
  For each VCAP CLM lookup, a frame pointer can optionally be moved to the next protocol layer, and the type of the next protocol layer can be specified. This influences key type selection for the next VCAP CLM lookup. The variables frame_ptr and frame_type holds information about the current protocol layer. For more information about how frame_ptr and frame_type is updated after each VCAP CLM lookup, see 4.14.2.2 Frame Pointer and Frame Type Update.

Key type is specified in action in the previous VCAP CLM lookup.
The action of a VCAP CLM lookup can optionally specify a specific key type to be used in the following VCAP CLM lookup. The full algorithm used for selecting VCAP CLM key type is shown in the following pseudo code.

```
// ================================================================
// ANA_CL: VCAP CLM Key Type Selection
//
// Determine which key type to use in VCAP CLM lookup.
// The algorithm is run prior to each VCAP CLM lookup.
// ----------------------------------------------------------------
// Frame position (byte pointer) for use as base for next VCAP CLM key
// Updated by UpdateFramePtrAndType() upon VCAP CLM lookup
int frame_ptr = 0;

// Frame type at frame_ptr position.
// Updated by UpdateFramePtrAndType() upon VCAP CLM lookup
//
// Supported values:
//   ETH:  frame_ptr points to DMAC
//   CW:   frame_ptr points to start of IP header or CW/ACH of MPLS frame
//   MPLS: frame_ptr points to an MPLS label.
frame_type_t frame_type = ETH;

int ClmKeyType(
               // VCAP CLM index. 0=First VCAP CLM lookup, 5=Last lookup
               clm_idx,

               // Action from previous VCAP CLM lookup (if clm_idx > 0)
               //
               // clm_action.vld is only set if VCAP CLM lookup
               // was enabled.
               clm_action,
               )
{
  int        key_type = 0;
  port_cfg_t port_clm_cfg;
  port_clm_cfg = csr.port[port_num].adv_cl_cfg[clm_idx];

  if (clm_action.vld && clm_action.nxt_key_type != 0) {
    // Previous VCAP CLM lookup specifies next key type
    return clm_action.nxt_key_type;
  }

  // Determine key type based on port parameters and frame_ptr/frame_type
  switch (frame_type) {
```

```
  case ETH:
    // Choose key type based on EtherType
    // The EtherType() function skips any VLAN tags. A maximum of 3 VLAN
    // tags can be skipped.
    switch (EtherType(frame_ptr)) {
    case ETYPE_IP4: // 0x0800
      key_type = port_clm_cfg.ip4_clm_key_sel;
      break;

    case ETYPE_IP6: // 0x86DD
      key_type = port_clm_cfg.ip6_clm_key_sel;
      break;

    case ETYPE_MPLS_UC: // 0x8847
      key_type = port_clm_cfg.mpls_uc_clm_key_sel;
      break;
    case ETYPE_MPLS_MC: // 0x8848
      key_type = port_clm_cfg.mpls_mc_clm_key_sel;
      break;
    }
    if (key_type = 0) {
      key_type = port_clm_cfg.etype_clm_key_sel;
    }
    break;

  case CW:
    switch (IPVersion(frame_ptr)) {
    case 4 : key_type = port_clm_cfg.ip4_clm_key_sel;   break;
    case 6 : key_type = port_clm_cfg.ip6_clm_key_sel;   break;
    default: key_type = port_clm_cfg.etype_clm_key_sel; break;
    }
    break;
  case MPLS:
    key_type = port_clm_cfg.mlbs_clm_key_sel;
    break;
  }
  return key_type;
} // ClmKeyType

// ------------------------------------------------------------------------
// ANA_CL: VCAP CLM Key Type Selection
// ========================================================================
```

By default, frames looped by the rewriter or frames already discarded or CPU-redirected, for instance the basic classifier, are not subject to VCAP CLM lookups. Optionally, VCAP CLM lookups can be enforced through configuration ANA_CL::CLM_MISC_CTRL.LBK_CLM_FORCE_ENA for looped frames and through ANA_CL::CLM_MISC_CTRL.IGR_PORT_CLM_FORCE_ENA for frames already discarded or CPU-redirected. When forcing a VCAP CLM lookup, the VCAP CLM key is chosen from the configuration in ANA_CL::CLM_MISC_CTRL.FORCED_KEY_SEL. Only selected keys are available.

4.14.2.2 Frame Pointer and Frame Type Update

After each VCAP CLM lookup, the information in the associated action is used to update frame_ptr and frame_type. This in turn influences the key type and key field values used for the next VCAP CLM lookup.

- NXT_OFFSET_FROM_TYPE:
  Move frame pointer past current protocol layer (Ethernet and/or IP). The actual number of bytes the frame pointer is moved may vary depending on content of the current protocol layer, such as the number of VLAN tags or IPv4 options.
- NXT_NORM_W16_OFFSET:

  Move frame pointer a number of 16 bit words. If both NXT_OFFSET_FROM_TYPE and NXT_NORM_W16_OFFSET are specified, then frame pointer is first moved based on NXT_OFFSET_FROM_TYPE and afterwards moved further into the frame based on NXT_NORM_W16_OFFSET.
- NXT_TYPE_AFTER_OFFSET:

Specify protocol layer at new frame pointer position.

Each VCAP CLM lookup can at most move the frame pointer 66 bytes. The following pseudo code shows the algorithm used to update frame_ptr and frame_type upon each VCPA_CLM lookup.

```
// ========================================================================
// ANA_CL: VCAP CLM Frame Pointer and Frame Type Update
//
// Update frame_ptr and frame_type upon VCAP CLM lookup.
// Prior to first VCAP CLM lookup frame_ptr is initialized to 0 and
// frame_type is set to ETH.
// The algorithm is run after each VCAP CLM lookup.
// ------------------------------------------------------------------------

// Function for updating
//   frame_ptr
//   frame_type
// based on action from VCAP CLM lookup
void UpdateFramePtrAndType(
                           // VCAP CLM index. 0=First VCAP CLM lookup,
                           // 5=Last VCAP CLM lookup
                           clm_idx,

                           // Action from VCAP CLM lookup
                           // clm_action.vld is only set if
                           // VCAP CLM lookup was enabled.
                           clm_action,

                           // From ClmKeyFieldsMpls()
                           clm_key_fields_mpls,
                           )
{
  int frame_ptr_current  = frame_ptr;
  int frame_type_current = frame_type;

  if (!clm_action.vld) return;

  if (clm_action.nxt_offset_from_type > 0) {
    // Move frame_ptr based on frame type
    switch (clm_action.nxt_offset_from_type) {
    case SKIP_ETH:
      if (frame_type_current == ETH) {
        // Move frame_ptr past DMAC, SMAC, 0-3 VLAN tags and EtherType
        SkipLinkLayer(&frame_ptr);
      }
      break;

    case SKIP_IP_ETH:
      switch (frame_type_current) {
      case ETH:
        // Move frame_ptr past DMAC, SMAC, 0-3 VLAN tags and EtherType
        if (EtherType(frame_ptr) == ETYPE_IP4) || // 0x0800
           (EtherType(frame_ptr) == ETYPE_IP6) {  // 0x86DD
          SkipLinkLayer(&frame_ptr);

          // Move frame_ptr past IPv4 header (including options) or
          // IPv6 header
          SkipIPHeader(&frame_ptr);
        }
        break;

      case CW:
        switch (IPVersion(frame_ptr)) {
        case 4:
        case 6:
          // Move frame_ptr past IPv4 header (including options) or
          // IPv6 header
          SkipIPHeader(&frame_ptr);
          break;
        }
        break;
      }
    }

    frame_type = clm_action.nxt_type_after_offset;
  } // clm_action.nxt_offset_from_type > 0

  if (clm_action.nxt_norm_w16_offset > 0) {
    // Move frame_ptr a specific number of bytes
    frame_ptr  = frame_ptr + 2*clm_action.nxt_norm_w16_offset;
    frame_type = clm_action.nxt_type_after_offset;
```

```
  }

  if (frame_type_current == DATA &&
      clm_action.nxt_key_type != 0) {
    // Allow frame_type change, regardsless of whether frame_ptr
    // has been moved
    frame_type = clm_action.nxt_type_after_offset;
  }

  // PW termination of MPLS frame
  if (clm_action.fwd_type == TERMINATE_PW) {
    if IsOam(frame_ptr, csr) && (clm_key_fields_mpls.rsv_lbl_pos) {
      // Reserved label used to identify OAM
      // Move frame_ptr if reserved MPLS label was skipped by
      // ClmKeyFieldsMpls() in key generation prior to VCAP CLM lookup
      frame_ptr += 4;
    } elsif (frame_type == CW && !IsOam(frame_ptr, csr)) {
      // PW termination of MPLS frame with CW => Skip CW and
      // change frame_type
      frame_ptr += 4;
      frame_type = ETH;
    }
  }

  // OAM LSP
  if (clm_action.fwd_type == POP_LBL && IsOam(frame_ptr, csr) &&
      (clm_key_fields_mpls.rsv_lbl_pos)) {
    // Reserved label used to identify OAM
    // Move frame_ptr if reserved MPLS label was skipped by
    // ClmKeyFieldsMpls() in key generation prior to VCAP CLM lookup
    frame_ptr += 4;
  }

  if (frame_ptr - frame_ptr_current > 66) {
    // Cannot skip >66 bytes per VCAP CLM.
    // Set sticky bit and don't skip anything
    csr.adv_cl_max_w16_offset_fail_sticky = 1;
    frame_ptr  = frame_ptr_current;
    frame_type = frame_type_current;
  }

  if (frame_ptr > 126 ) {
    // Cannot skip >126 bytes total.
    // Set sticky bit and discard frame
    csr.adv_cl_nxt_offset_too_big_sticky = 1;
    frame.abort  = true;
  }
} // UpdateFramePtrAndType

// ------------------------------------------------------------------------
// ANA_CL: VCAP CLM Frame Pointer and Frame Type Update
// ========================================================================
```

4.14.2.3 Removing Protocol Layers

The analyzer classifier can instruct the rewriter to remove protocol layers from the frame, that is, to remove a number of bytes from the start of the frame.

This is achieved by setting the NXT_NORMALIZE action field in the VCAP CLM action. When NXT_NORMALIZE is set in a VCAP CLM action, then the frame pointer for the next VCAP CLM lookup (if any) is calculated, and the rewriter is then instructed to remove all bytes up to, but not including, the position pointed to by the frame pointer. For more information about the calculation, see 4.14.2.2 Frame Pointer and Frame Type Update.

If multiple VCAP CLM actions have the NXT_NORMALIZE bit set, the last such action controls the number of bytes removed by the rewriter.

Note that the rewriter can skip a maximum of 42 bytes, starting at the DMAC of the frame. This corresponds to an Ethernet header with three VLAN tags and an MPLS stack with three LSEs and a control word (CW).

4.14.2.4 Miscellaneous VCAP CLM Key Configuration

The analyzer classifier can control some specific fields when generating the VCAP CLM keys. The following table lists the registers that control the parameters.

Table 4-111. Miscellaneous VCAP CLM Key Configuration

Register	Description	Replication
ANA_CL:PORT:ADV_CL_CFG	Configures L3_DSCP and TCI0 source.	Per port per VCAP CLM lookup
ANA_CL::CLM_MISC_CTRL	Configures IGR_PORT_MASK_SEL and IGR_PORT_MASK.	None

Each port can control specific fields in the generated keys:

- Selects source of DSCP in L3_DSCP key field (ANA_CL:PORT:ADV_CL_CFG.USE_CL_DSCP_ENA). By default, the DSCP value is taken from the frame. Another option is to use the current classified DSCP value as input to the key. The current classified DSCP value is the result from either the previous VCAM_CLM lookup or the basic classifier if this is the first lookup.
- Selects source of VID, PCP, and DEI in the VID0, PCP0, and DEI0 key fields (ANA_CL:PORT:ADV_CL_CFG.USE_CL_TCI0_ENA). By default, the values are taken from the outer VLAN tag in the frame (or port's VLAN if frame is untagged). Another option is to use the current classified values as input to the key. The current classified values are the result from either the previous VCAM_CLM lookup or the basic classifier if this is the first lookup.

Note that these configurations are only applicable to keys containing the relevant key fields.

VCAP CLM keys NORMAL, NORMAL_7TUPLE, and NORMAL_5TUPLE_IP4 contain the key fields IGR_PORT_MASK_SEL and IGR_PORT_MASK. For frames received on a front port, IGR_PORT_MASK_SEL is set to 0 and the bit corresponding to the frame's physical ingress port number is set in IGR_PORT_MASK. The following lists some settings for frames received on other interfaces and how this can be configured through register CLM_MISC_CTRL.

- Loopback frames:
  By default, IGR_PORT_MASK_SEL=1 and IGR_PORT_MASK has one bit set corresponding to the physical port which the frame was looped on. Optionally use IGR_PORT_MASK_SEL = 3.
- Masqueraded frames:
  By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has one bit set corresponding to the masquerade port. Optionally, use IGR_PORT_MASK_SEL = 2.
- VStaX frames:
  By default, frames received with a VStaX header on a front port use IGR_PORT_MASK_SEL = 0. Optionally, use IGR_PORT_MASK_SEL = 3.
- Virtual device frames:

  By default, IGR_PORT_MASK_SEL = 0 and IGR_PORT_MASK has one bit set corresponding to the original physical ingress port. Optionally, use IGR_PORT_MASK_SEL = 3.
- CPU injected frames:

  By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has none bits set. Optionally use IGR_PORT_MASK_SEL=3.

For more information about the contents of IGR_PORT_MASK_SEL and IGR_PORT_MASK, see 4.13 VCAP CLM Keys and Actions.

4.14.2.5 VCAP CLM Range Checkers

The following table lists the registers associated with configuring VCAP CLM range checkers.

Table 4-112. VCAP CLM Range Checker Configuration Registers Overview

Register	Description	Replication
ANA_CL::ADV_RNG_CTRL	Configures range checker types	Per range checker
ANA_CL::ADV_RNG_VALUE_CFG	Configures range start and end points	Per range checker

.....continued		
Register	**Description**	**Replication**
ANA_CL:PORT:ADV_PORT_RNG_CTRL	Configures range checker types	Per port per range checker
ANA_CL:PORT:ADV_PORT_RNG_VALUE_CFG	Configures range start and end points	Per port per range checker

VCAP CLM supports a combination of eight global range checkers and eight per-port range checkers in VCAP CLM keys MLL, LL_FULL, TRI_VID, NORMAL, NORMAL_7TUPLE, NORMAL_5TUPLE_IP4, and PURE_5TUPLE_IP4. All frames using these keys are compared against the global range checkers and the per-port range checkers and a combined 1-bit range “match/no match” flag is returned. The eight match/no match flags are used in the L4_RNG key field. The results of global range checker n and perport range checker n map to bit n in L4_RNG. The result of the per-port range checker takes precedence over the global range checker.

VCAP IS2 also supports range checkers, however, they are independent of the VCAP CLM range checkers.

The range key is generated for each frame based on extracted frame data and the configurations in ANA_CL::ADV_RNG_CTRL and ANA_CL:PORT:ADV_PORT_RNG_CTRL. Each range checkers can be configured to one of the following range types.

- TCP/UDP destination port range:

  Input to the range is the frame’s TCP/UDP destination port number.
  Range is only applicable to TCP/UDP frames.
- TCP/UDP source port range:

  Input to the range is the frame’s TCP/UDP source port number.
  Range is only applicable to TCP/UDP frames.
- TCP/UDP source and destination ports range:
  Range is matched if either source or destination port is within range.
  Input to the range are the frame’s TCP/UDP source and destination port numbers.
  Range is only applicable to TCP/UDP frames.
- VID range:
  Input to the range is the classified VID.
  Range is applicable to all frames.
- DSCP range:

  Input to the range is the classified DSCP value.
  Range is applicable to IPv4 and IPv6 frames.
- EtherType range:

  Input to the range is the frame’s EtherType.
  Range is applicable to all frames
- Outer VID range:
  Input to the range is the frame’s outer VID. Range is applicable to all frames with one VLAN tag or more.
- Inner VID range:
  Input to the range is the frame’s inner VID. Range is applicable to all frames with two VLAN tags or more.

Range start points and range end points are configured in ANA_CL::ADV_RNG_VALUE_CFG with both the start point and the end point being included in the range. A range matches if the input value to the range (for instance the frame’s TCP/UDP destination port) is within the range defined by the start point and the end point.

4.14.2.6 Cascading VCAP CLM Lookups

Each of the six VCAP CLM lookups can be “cascaded” such that a frame only matches a VCAP CLM entry in lookup *n* + 1 if it also matched a specific entry in VCAP CLM lookup *n*. In other words, VCAP CLM entries in different VCAP CLM lookups are bound together.

The following parameters control cascading of VCAP CLM entries.

- Key field:
  G_IDX
  The generic index, G_IDX, is available in most VCAP CLM key types. The value of this field can be specified in the action of the preceding VCAP CLM lookup.
- Action fields:
  NXT_IDX_CTRL and NXT_IDX
  Controls how to calculate G_IDX for next VCAP CLM lookup.

For information about detailed encoding of NXT_IDX_CTRL, see Table 4-124.

In addition to cascading VCAP CLM lookups, it is also possible for each VCAP CLM lookup to move to the next protocol layer of the frame, such that key values for lookup n + 1 are retrieved from a different offset in the frame than for lookup n. For information about the algorithm, see 4.14.2.2 Frame Pointer and Frame Type Update.

4.14.3 QoS Mapping Table

The following table lists the registers associated with the QoS mapping table.

Table 4-113. VCAP CLM QoS Mapping Table Registers Overview

Register	Description	Replication
ANA_CL:PORT:MAP_CFG	Configuration of default QoS mappings.	Per port per QoS mapping lookup (x2)
ANA_CL:MAP_TBL:SET_CTRL	Controls which mapping results to use.	512
ANA_CL:MAP_TBL:MAP_ENTRY	Configuration of the QoS mapping values.	4,096

The classifier contains a shared QoS mapping table with 4,096 entries (ANA_CL:MAP_TBL:MAP_ENTRY) that maps incoming QoS information to classified internal QoS information. Inputs to a mapping can be either DEI and PCP from VLAN tags or E-TAGs, DSCP value, or MPLS TC bits. Outputs from a mapping are new classified values for COSID, DEI, PCP, DSCP, QoS class, DP level, and TC bits. The table is looked up twice for each frame with two different keys.

The QoS mapping table is laid out with 512 rows with each eight mapping entries. Each specific QoS mapping only uses a subset of the 4,096 entries and thus many different QoS mappings can be active at the same time. As examples, a table may hold 64 unique mappings from DSCP to DEI/PCP or 256 unique translations of DEI/PCP to DEI/PCP.

A lookup is defined by a base index and a key. The base index defines which of the rows in the QoS mapping table to start at and the key defines which parameters from the frame to use as offset from the base index. By default, the port defines the two lookups in the QoS mapping table. However, VCAP CLM can override the port-based configuration through VCAP CLM actions MAP_IDX and MAP_KEY. Each match in VCAP CLM may redefine one of the two lookups.

The following lists the available keys and how the resulting row and entry number in the mapping table is derived.

- **PCP**: Use PCP from the frame's outer VLAN tag. If the frame is untagged, the port's VLAN tag is used. Row and entry number is derived the following way:
  Row = MAP_IDX

  Entry = PCP

  Number of entries per mapping: 8 entries.
- **DEI and PCP**: Use DEI and PCP from one of the frame's VLAN tags. The MAP_KEY can select between using either outer, middle, or inner tag. If the frame is untagged, the port's VLAN tag is used. Row and entry number is derived the following way:

  Row = (8x DEI + PCP) DIV 8 + MAP_IDX

Entry = (8x DEI + PCP) MOD 8

Number of entries per mapping: 16 entries.

- **DSCP (IP frames only)**: Use the frame's DSCP value. For non-IP frames, no mapping is done. Row and entry number is derived the following way:

  Row = DSCP DIV 8 + MAP_IDX

  Entry = DSCP MOD 8

  Number of entries per mapping: 64 entries.
- **DSCP (all frames)**: Use the frame's DSCP value. For non-IP VLAN tagged frames, use DEI and PCP from the frame's outer VLAN tag. For non-IP untagged frames, use DEI and PCP from port VLAN. Row and entry number is derived the following way:

  IP frames:

  Row = DSCP DIV 8 + MAP_IDX

  Entry = DSCP MOD 8.

  Non-IP frames:

  Row = (8x DEI + PCP) DIV 8 + MAP_IDX + 8

  Entry = (8x DEI + PCP) MOD 8

  Number of entries per mapping: 80 entries.
- **TC**: Use the frame's classified TC bits from the extracted MPLS label (label selected by VCAP CLM actions TC_ENA or TC_LABEL). Row and entry number is derived the following way:

  Row = MAP_IDX,

  Entry = TC

  Number of entries per mapping: 8 entries.
- **E-DEI and E-PCP**: Use E-DEI and E-PCP from the frame's E-TAG. For frames with no E-TAGs, no mapping is done. Row and entry number is derived the following way:

  Row = (8x E-DEI + E-PCP) DIV 8 + MAP_IDX

  Entry = (8x E-DEI + E-PCP) MOD 8.

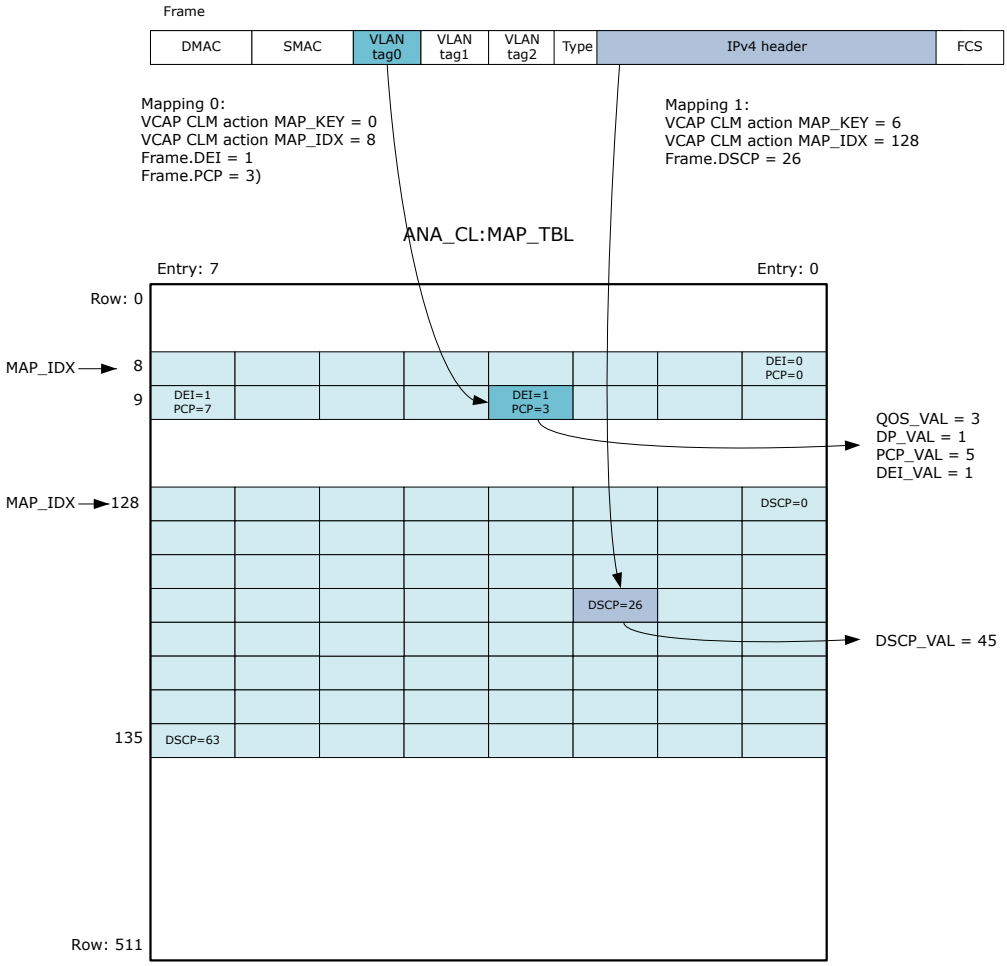
  Number of entries per mapping: 16 entries.

QoS mappings can be shared between ports and VCAP CLM entries by defining the same mapping and using the same entries in the mapping table. It is up to the user of the QoS mapping table to make sure that entries are allocated so that they do not overlap unintentionally.

Each entry in the QoS mapping table (ANA_CL:MAP_TBL:MAP_ENTRY) contains a full set of new QoS values (COSID, DEI, PCP, QoS class, DP level, DSCP, TC) to be used as new classified values instead of the currently classified values from basic classification and VCAP CLM. Each of the QoS values is gated by an enable bit (ANA_CL:MAP_TBL:SET_CTRL) so that a mapping can for example define a complete new set of QoS values by enabling all values or it can override selected values only. Note that ANA_CL:MAP_TBL:SET_CTRL must be configured for each row used in the QoS mapping table.

The following figure shows an example of a frame for which a DSCP and a DEI/PCP mapping are defined.

Figure 4-30. Example of QoS Mappings



4.14.4 Conversation-Sensitive Frame Collection (CSC)

The following table lists the registers associated with the conversation-sensitive frame collection for link aggregation.

Table 4-114. Conversation-Sensitive Frame Collection Registers Overview

Register	Description	Replication
ANA_CL:PORT:CSC_CFG	Enables port's link aggregation conversation identifier classification and configures valid TPIDs.	Per port
ANA_CL:CSC	Discard configuration for LAG conversation identifiers not accepted by port.	Per port per conversation identifier
ANA_CL::MISC_CFG.CSC_PIPELINE_PT	Pipeline point for frames discarded by CSC.	Common

As part of IEEE 802.1AX Link Aggregation, conversation-sensitive frame collection determines a conversation identifier for all frames based on outer VLAN tags. The conversation identifier is a value in the range from 0 through 4095. The device recognizes following types of tags based on the TPID.

- Customer tags (C-tags), which use TPID 0x8100.
- Service tags (S-tags), which use TPID 0x88A8 (IEEE 802.1ad).
- Custom service tags (S-tags), which use one of three custom TPIDs programmed in ANA_CL::VLAN_STAG_CFG. Three different values are supported.

It is configurable which of the five recognized TPIDs are valid for the conversation identifier classification (CSC_CFG.CSC_TPID_AWARE_DIS). Only outer VLAN tags with valid TPIDs are used by the conversation identifier classification but frames may still be carrying an E-TAG or a redundancy tag as the outermost tag.

The conversation identifier is determined the following way.

- Untagged and priority-tagged frames are assigned the port's VID (VLAN_CTRL.PORT_VID).
- VLAN tagged frames for which the TPID is allowed are assigned the VLAN tag's VID.
- VLAN tagged frames for which the TPID is disallowed are assigned the value zero.

The CSC table is looked up using the conversation identifier as index. The conversation-sensitive frame collection algorithm then discards frames received on an ingress port marked in the port mask retrieved from the CSC table.

Frames discarded by the conversation-sensitive frame collection are assigned a configurable pipeline point.

4.14.5 Analyzer Classifier Diagnostics

The analyzer classifier contains a large set of sticky bits that can inform about the frame processing and decisions made by ANA_CL. The following categories of sticky bits are available.

- Frame acceptance filtering (ANA_CL::FILTER_STICKY, ANA_CL::VLAN_FILTER_STICKY, ANA_CL::ETAG_FILTER_STICKY).
- VLAN and QoS classification (ANA_CL::CLASS_STICKY)
- CPU forwarding (ANA_CL::CAT_STICKY)
- VCAP CLM lookups and actions, QoS mapping tables (ANA_CL::ADV_CL_STICKY)
- IP header checks (ANA_CL::IP_HDR_CHK_STICKY)
- GRE header checks (ANA_CL::GRE_HDR_CHK_STICKY)
- Conversation-sensitive collection and classification (ANA_CL::CLASS_STICKY)

The sticky bits are common for all ports in the analyzer classifier. All sticky bits, except for VCAP CLM sticky bits, have four corresponding sticky mask bits (ANA_CL:STICKY_MASK) that allow any sticky bit to be counted by one of the four per-port analyzer access control counters. For more information, see 4.22.3 Analyzer Statistics.

4.15 VLAN and MSTP

The VLAN table and the MSTP table are the two main tables that control VLAN and Spanning Tree frame processing.

The following table shows the configuration parameters (located within the ANA_L3 configuration target) for each entry in the VLAN table.

Table 4-115. VLAN Table (5120 Entries)

Field in ANA_L3:VLAN	Bits	Description
VMID	9	VMID, identifying VLAN's router leg.
VLAN_MSTP_PTR	7	Pointer to STP instance associated with VLAN.
VLAN_FID	13	FID to use for learning and forwarding.
VLAN_SEC_FWD_ENA	1	Enables secure forwarding on a per VLAN basis.When secure forwarding is enabled, only frames with known SMAC are forwarded.
VLAN_FLOOD_DIS	1	Disables flooding of frames with unknown DMAC on a per VLAN basis.
VLAN_LRN_DIS	1	Disables learning of SMAC of frames received on this VLAN.

.....continued		
Field in ANA_L3:VLAN	**Bits**	**Description**
VLAN_RLEG_ENA	1	Enables router leg in VLAN.
VLAN_PRIVATE_ENA	1	Enables/disables this VLAN as a Private VLAN (PVLAN).
VLAN_MIRROR_ENA	1	VLAN mirror enable flag. If this field is set, frames classified to this ingress VLAN are mirrored. Note that a mirroring probe must also be configured to enable VLAN based mirroring. see 4.22.5 Mirroring.
VLAN_PORT_MASK	65	Specifies mask of ports belonging to VLAN.
TUPE_CTRL	16	Controls value for Table Update Engine (TUPE).

The following table shows the configuration parameters for each entry in the MSTP table.

Table 4-116. MSTP Table (66 Entries)

Field in ANA_L3:MSTP	Bits	Description
MSTP_FWD_MASK	1 per port	Enables/disables forwarding per port
MSTP_LRN_MASK	1 per port	Enables/disables learning per port

The following table lists common parameters that also affect the VLAN and MSTP processing of frames.

Table 4-117. Common VLAN and MSTP Parameters

Register/Field in ANA_L3_COMMON	Bits	Description
VLAN_CTRL.VLAN_ENA	1	Enables/disables VLAN lookup
PORT_FWD_CTRL	1 per port	Configures forwarding state per port
PORT_LRN_CTRL	1 per port	Configures learning state per port
VLAN_FILTER_CTRL	1 per port	Configures VLAN ingress filtering per port
VLAN_ISOLATED_CFG	1 per port	Configures isolated port mask
VLAN_COMMUNITY_CFG	1 per port	Configures community port mask

If VLAN support is enabled (in VLAN_CTRL.VLAN_ENA), the classified VID is used to look up the VLAN information in the VLAN table. The VLAN table in turn provides an address (VLAN_MSTP_PTR) into the MSTP table. Learn, mirror, and forwarding results are calculated by combining information from the VLAN and MSTP tables.

4.15.1 Private VLAN

In a Private VLAN (PVLAN), ports are configured to be one of three different types: promiscuous, isolated, or community.

- Promiscuous Ports: A promiscuous port can communicate with all ports in the PVLAN, including the isolated and community ports.
- Isolated Ports: An isolated port has complete Layer 2 separation from the other ports within the same PVLAN, but not from the promiscuous ports. PVLANs block all traffic to isolated ports except traffic from promiscuous ports. Traffic from isolated port is forwarded only to promiscuous ports.
- Community Ports: Community ports communicate among themselves and with the PVLAN's promiscuous ports. Community ports cannot communicate with isolated ports.

PVLAN can be enabled per VLAN, and port type can be configured per physical port.

4.15.2 VLAN Pseudo Code

The pseudo code for the ingress VLAN processing is shown below. For routed frames, an egress VLAN processing also takes place. This is not part of the following pseudo code. In the pseudo code, "csr." is used to denote

configuration parameters, “req.” is used to denote input from the previous processing steps in the analyzer, as well as the information provided for the following steps in the analyzer.

```
if (req.cpu_inject != 0) {
  // cpu_inject => Bypass!
  return;
}

if (csr.vlan_ena) {
  // Get VLAN entry based on Classified VID
  igr_vlan_entry = csr.vlan_tbl[req.ivid + req.xvid_ext*4096];
} else {
  // Default VLAN entry
  igr_vlan_entry.vlan_mstp_ptr    = 0;
  igr_vlan_entry.vlan_fid         = 0;
  igr_vlan_entry.vlan_sec_fwd_ena = 0;
  igr_vlan_entry.vlan_flood_dis   = 0;
  igr_vlan_entry.vlan_lrn_dis     = 0;
  igr_vlan_entry.vlan_rleg_ena    = 0;
  igr_vlan_entry.vlan_private_ena = 0;
  igr_vlan_entry.vlan_mirror_ena  = 0;
  igr_vlan_entry.vlan_port_mask   = -1; // All-ones
  igr_vlan_entry.vmid             = 0;
}

// Retrieve MSTP state
if (csr.vlan_ena) {
  igr_mstp_entry = csr.mstp_tbl[igr_vlan_entry.vlan_mstp_ptr];
} else {
  igr_mstp_entry.mstp_fwd_mask = -1; // All-ones
  igr_mstp_entry.mstp_lrn_mask = -1; // All-ones
}

if (IsCpuPort(req.port_num) ||
    IsVD0(req.port_num) ||
    IsVD1(req.port_num)) {
  // Received from CPU or VD0/VD1 =>
  // * Do not learn
  // * Do not perform ingress filtering
  //   (these ports are not in ingress masks)
  req.l2_lrn = 0;
} else {
  // ------------------------------------------
  // Perform ingress filtering and learning disable
  // ------------------------------------------

  // Port forwarding state
  if (csr.common.port_fwd_ena[req.port_num] == 0) {
    // Ingress port not enabled for forwarding
    req.l2_fwd = 0;
    csr.port_fwd_deny_sticky =1;
  }

  // Port learning state
  if (csr.port_lrn_ena[req.port_num] == 0) {
    req.l2_lrn = 0;
    csr.port_lrn_deny_sticky = 1;
  }

  if (csr.vlan_ena) {
    // Empty VLAN?
    if (igr_vlan_entry.vlan_port_mask == 0) {
      csr.vlan_lookup_invld_sticky = 1;
    }

    // VLAN ingress filtering
```

```
    if (csr.vlan_igr_filter_ena[req.port_num] &&
        igr_vlan_entry.vlan_port_mask[req.port_num] == 0) {
      req.l2_fwd = 0;
      req.l2_lrn = 0;
      csr.vlan_igr_filter_sticky = 1;
    } else {
      // MSTP forwarding state
      if (igr_mstp_entry.mstp_fwd_mask[req.port_num] == 0) {
        req.l2_fwd = 0;
        csr.mstp_discard_sticky = 1;
      } else {
        csr.mstp_fwd_allowed_sticky = 1;
      }

      // Learning enabled for VLAN?
      if (igr_vlan_entry.vlan_lrn_dis) {
        req.l2_lrn = 0;
        csr.vlan_lrn_deny_sticky = 1;

        // MSTP learning state
      } else if (igr_mstp_entry.mstp_lrn_mask[req.port_num] == 0) {
        req.l2_lrn = 0;
        csr.mstp_lrn_deny_sticky = 1;
      } else {
        csr.mstp_lrn_allowed_sticky = 1;
      }

      // Private VLAN handling
      if (igr_vlan_entry.vlan_private_ena) {
        if (req.vs2_avail == 0) {
          // Not received from stack port, so determine port type and
          // update req accordingly.

          // 0b00: Promiscuous port
          // 0b01: Community port
          // 0b10: Isolated port
          req.ingr_port_type = 0b00;
          if (csr.vlan_community_mask[req.port_num]) {
            req.ingr_port_type = 0b01;
          }
          if (csr.vlan_isolated_mask[req.port_num]) {
            req.ingr_port_type = 0b10;
          }
        }

        if (req.ingr_port_type == 0b10) {
          // Isolated port
          // Only allow communication with promiscuous ports
          igr_vlan_entry.vlan_port_mask &=
            (~csr.vlan_isolated_mask & ~csr.vlan_community_mask);
        } else if (req.ingr_port_type == 0b01) {
          // Community port
          // Allow communication with promiscuous ports and other community ports,
          // but not with isolated ports
          igr_vlan_entry.vlan_port_mask &= ~csr.vlan_isolated_mask;
        }

      }
    } // vlan_igr_filter_ena
  } // csr.vlan_ena
}

// Only allow forwarding to enabled ports
igr_vlan_entry.vlan_port_mask &= csr.port_fwd_ena;

// Update req with results
req.vlan_mask = igr_vlan_entry.vlan_port_mask & igr_mstp_entry.mstp_fwd_mask;
```

```
req.ifid             = igr_vlan_entry.vlan_fid;
req.vlan_mirror      = igr_vlan_entry.vlan_mirror_ena;
req.vlan_flood_dis   = igr_vlan_entry.vlan_flood_dis;
req.vlan_sec_fwd_ena = igr_vlan_entry.vlan_sec_fwd_ena;

// Identical ingress and egress FID/VID.
// May be overruled later by egress VLAN handling.
// Note: ANA_L2 DMAC lookup always use req.efid, even when req.l3_fwd=0.
req.evid = req.ivid;
if (req.dmac[40]) {
  // If MC, DA lookup must use EVID
  req.efid = req.evid;
} else {
  // If UC, DA lookup must use IFID
  req.efid = req.ifid;
}

// Store vmid for later use (e.g. in rleg detection)
req.irleg = igr_vlan_entry.vmid;
req.erleg = req.irleg;
```

4.15.2.1 VLAN Table Update Engine

The VLAN table update engine (TUPE) can be used to quickly update a large number of port masks in the VLAN Table. For example, to support fast fail-over in scenarios with redundant forwarding paths. The following table lists the TUPE related parameters that are outside the VLAN Table.

Table 4-118. VLAN Table Update Engine (TUPE) Parameters

Register/Field in ANA_L3:TUPE	Bits	Description
TUPE_MISC.TUPE_START	1	Start TUPE.
TUPE_MISC.TUPE_CTRL_VAL_ENA	1	Enable use of TUPE_CTRL_VAL and TUPE_CTRL_VAL_MASK.
TUPE_CTRL_BIT_ENA	1	Enable use of TUPE_CTRL_BIT_MASK.
TUPE_PORT_MASK_A_ENA	1	Enable use of TUPE_PORT_MASK_A.
TUPE_PORT_MASK_B_ENA	1	Enable use of TUPE_PORT_MASK_B
TUPE_COMB_MASK_ENA	1	Enable combined use of TUPE_CTRL_BIT_MASK and TUPE_PORT_MASK_A.
TUPE_ADDR.TUPE_START_ADDR	13	First address in VLAN table for TUPE to process.
TUPE_ADDR.TUPE_END_ADDR	13	Last address in VLAN table for TUPE to process.
TUPE_CMD_PORT_MASK_CLR	65	TUPE command: Port mask bits to clear.
TUPE_CMD_PORT_MASK_SET	65	TUPE command: Port mask bits to set.
TUPE_CTRL_VAL	16	TUPE parameter controlling which VLAN table entries to update.
TUPE_CTRL_VAL_MASK	16	TUPE parameter controlling which VLAN table entries to update.
TUPE_CTRL_BIT_MASK	16	TUPE parameter controlling which VLAN table entries to update.
TUPE_PORT_MASK_A	65	TUPE parameter controlling which VLAN table entries to update.
TUPE_PORT_MASK_B	65	TUPE parameter controlling which VLAN table entries to update.

The TUPE_CTRL field in the VLAN table can be used to classify VLANs into different groups, and the VLAN_PORT_MASK for such groups of VLANs are quickly updated using TUPE. Using the parameters listed, the TUPE_CTRL field in the VLAN table can be processed as a field of individual bits, a field with one value, or a combination of the two.

For example, if a command for TUPE uses the following, the TUPE_CTRL field is treated as an 8-bit value field and eight individual bits.

- TUPE_CTRL_BIT_ENA = 1
- TUPE_CTRL_BIT_MASK = 0x00ff
- TUPE_CTRL_VAL_MASK = 0xff00

In order for TUPE to update a VLAN entry's VLAN_PORT_MASK, the value part of TUPE_CTRL must match the required value, and one or more bits of the bit part of TUPE_CTRL must be set.

If all bits in TUPE_CTRL are used as a value field, then a total of 216 groups can be created, but each VLAN can only be member of one such group.

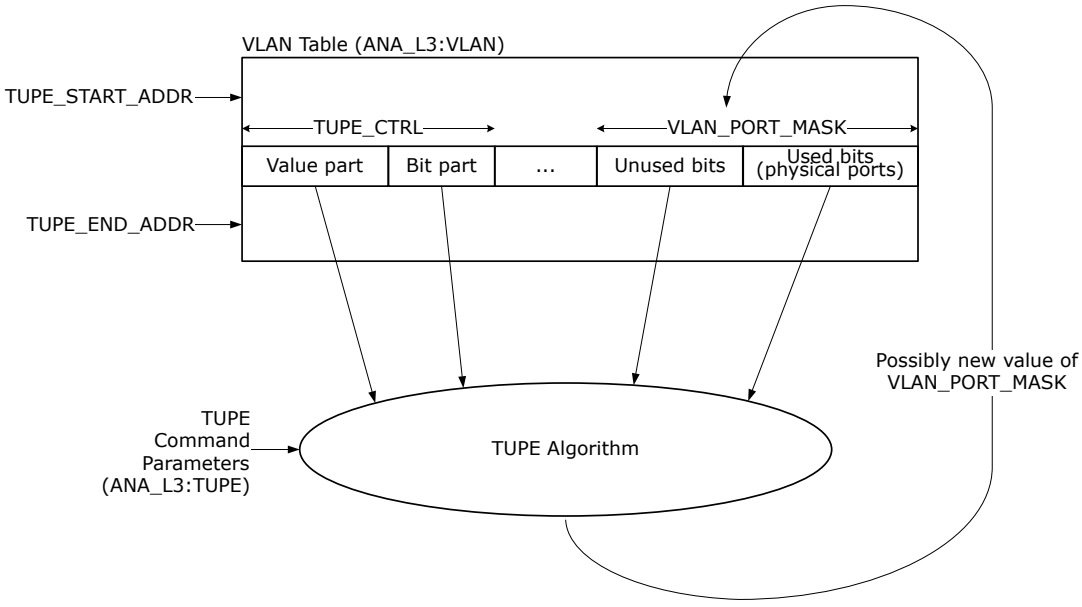
On the other end of the spectrum, if all bits in TUPE_CTRL are treated as individual bits, then only 16 groups can be created, but every VLAN can be member of any number of these groups.

In configurations that have fewer physical ports than the number of bits in the VLAN_PORT_MASK, such bits can be used as an extension to the bit part of the TUPE_CTRL field.

Apart from the VLAN's TUPE_CTRL, the value of the VLAN's VLAN_PORT_MASK is also used to decide whether to update a given VLAN table entry.

The following figure depicts the TUPE functionality.

Figure 4-31. TUPE



The following pseudo code shows the full algorithm used by TUPE.

```
for (addr = csr.tupe_start_addr; addr < csr.tupe_end_addr; addr++) {
  vlan_entry = vlan_tbl[addr];
  if (
      (// If enabled, check matching value
```

```
        !csr.tupe_ctrl_val_ena
        ||
        ((vlan_entry.tupe_ctrl & csr.tupe_ctrl_val_mask) == csr.tupe_ctrl_val))
       &&
       (// If enabled, check if VLAN's tupe_ctrl has any bits overlapping
        // with csr.tupe_CTRL_BIT_MASK
        !csr.tupe_ctrl_bit_ena
        ||
        ((vlan_entry.tupe_ctrl & csr.tupe_ctrl_bit_mask) != 0))
       &&
       (// If enabled, check if VLAN's vlan_port_mask has any bits overlapping
        // with csr.tupe_PORT_MASK_A
        !csr.tupe_port_mask_a_ena
        ||
        ((vlan_entry.vlan_port_mask & csr.tupe_port_mask_a) != 0))
       &&
       (// Combined mode
        // If enabled, check if VLAN's tupe_ctrl has any bits overlapping
        // with TUPE_CTRL_BIT_MASK or VLAN's vlan_port_mask has any bits
        // overlapping with TUPE_PORT_MASK_A
        // I.e. the bit mask part of TUPE_CTRL is extended with bits from
        // vlan_port_maks.
        !csr.tupe_comb_mask_ena
        ||
        ((vlan_entry.tupe_ctrl & csr.tupe_ctrl_bit_mask) != 0)
        ||
        ((vlan_entry.vlan_port_mask & csr.tupe_port_mask_a) != 0))
       &&
       (// If enabled, check if VLAN's vlan_port_mask has any bits overlapping
        // with csr.tupe_PORT_MASK_B
        !csr.tupe_port_mask_b_ena
        ||
        ((vlan_entry.vlan_port_mask & csr.tupe_port_mask_b) != 0))
       ) {
     // vlan_port_mask must be updated for this addr
     for (p = 0; p < port_count; p++) {
       if (csr.tupe_cmd_port_mask_clr[p] == 1 &&
           csr.tupe_cmd_port_mask_set[p] == 1) {
         // clr+set => toggle
         vlan_entry.vlan_port_mask[p] = !vlan_entry.vlan_port_mask[p];
       } else if (csr.tupe_cmd_port_mask_clr[p] == 1) {
         vlan_entry.vlan_port_mask[p] = 0;
       } else if (csr.tupe_cmd_port_mask_set[p] == 1) {
         vlan_entry.vlan_port_mask[p] = 1;
       }
     }
     // Write back to VLAN table
     vlan_tbl[addr] = vlan_entry;
   }
```

4.16 VCAP IP6PFX: Keys and Action

For IPv6 frames, the most significant 64 bits can - optionally - be identified using VCAP IP6PFX and the resulting IP6PFX_ID can then be used in LPM lookup in order to reduce the width of the IPv6 LPM key and thus increase the supported IPv6 routing table size.

The intended use is to configure the local IPv6 prefixes in VCAP IP6PFX together with non-zero IP6PFX_ID action values identifying the local prefix. If no match is found in VCAP IP6PFX, IP6PFX_ID=0 is returned and thus identifies “any remote prefix”.

For IP6PFX_ID=0 (i.e. remote prefix), the most significant 64 bits of the IPv6 address together with IP6PFX_ID is used in the VCAP LPM key.

For IP6PFX_ID!=0 (i.e. local prefix), the least significant 64 bits of the IPv6 address together with IP6PFX_ID is used in the VCAP LPM key.

For SIP lookup, VCAP IP6PFX lookup is performed if the frame is IPv6 and:

```
SIP_IP6PFX_ENA==1 && (SIP_RPF_ENA==1 || SIP_CMP_ENA==1).
```

For DIP lookup, VCAP IP6PFX lookup is performed if the frame is IPv6 and:

```
DIP_IP6PFX_ENA==1 && (IsUnicast(frame) || IP6_MC_DIP_FWD_ENA==1).
```

The following table shows the different key types supported by the VCAP IP6PFX.

Table 4-119. VCAP IP6PFX Keys and Sizes

Key Name	Number of Words	Key Type
IP6PFX	2 words	X2 type

The following table provides an overview of the VCAP IP6PFX key. When programming an entry in VCAP IP6PFX, the associated key fields listed must be programmed in the listed order with the first field in the table starting at bit 0 of the entry. For more information, see 4.10 Versatile Content-Aware Processor (VCAP™).

Table 4-120. VCAP IP6PFX Key Overview

Field Name	Short Description	Size	IP6PFX
DST_FLAG	0=SIP, 1=DIP	1	x
AFFIX	LPM affix	10	x
IP6_XIP64	64 most significant bits of SIP/DIP	64	x

4.16.1 VCAP IP6PFX Key Details

Table 4-121. VCAP IP6PFX Key Details

Field Name	Description	Size
DST_FLAG	0: IP4_XIP is only to be used for SIP matching 1: IP4_XIP is only to be used for DIP matching	1
AFFIX	LPM affix	10
IP6_XIP64	64 most significant bits of SIP/DIP	64

4.16.2 VCAP IP6PFX Actions

VCAP IP6PFX supports one action: IP6_PFX_ID.

The following table provides information about the VCAP IP6PFX action. When programming an action in VCAP IP6PFX, the associated action fields listed must be programmed in the listed order, with the first field in the table starting at bit 0 of the action.

Table 4-122. VCAP IP6PFX Actions

Field Name	Description and Encoding	Size	IP6PFX
IP6PFX_ID	IP6PFX_ID to be used in VCAP LPM lookup.	9	x

4.17 VCAP LPM: Keys and Action

The IP addresses used for IP source/destination guard and for IP routing are stored in a VCAP in the VCAP_SUPER block. The part of the VCAP_SUPER block allocated for this purpose is VCAP LPM (Longest Path Match).

VCAP LPM encodes both the IP addresses (VCAP keys) and the action information associated with each VCAP key.

The following table shows the different key types supported by the VCAP LPM.

Table 4-123. VCAP LPM Keys and Sizes

Key Name	Number of Words	Key Type
SGL_IP4	1 word	X1 type
DBL_IP4	2 words	X2 type
IP6PFX_ID		
SGL_IP6	3 words	X3 type
DBL_IP6	6 words	X6 type

The following table provides an overview of the VCAP LPM key. When programming an entry in VCAP LPM, the associated key fields listed must be programmed in the listed order with the first field in the table starting at bit 0 of the entry. For more information, see 4.10 Versatile Content-Aware Processor (VCAP™).

Table 4-124. VCAP LPM Key Overview

Field Name	Short Description	Size	SGL_IP4	DBL_IP4	SGL_IP6	DBL_IP6	IP6PFX_ID
X2_TYPE	X2 type (each entry uses 2 words)	1		x			x
DST_FLAG	0=SIP, 1=DIP	1	x		x		x
AFFIX	LPM affix	10	x	x	x	x	x
IP4_XIP	IPv4 SIP/DIP	32	x				
IP4_SIP	IPv4 SIP	32		x			
IP4_DIP	IPv4 DIP	32		x			
IP6PFX_ID	IP6PFX_ID from VCAP IP6PFX lookup	9					x
IP6_XIP64	64 bits of SIP/DIP	64					x
IP6_XIP	IPv6 SIP/DIP	128			x		
IP6_SIP	IPv6 SIP	128				x	
IP6_DIP	IPv6 DIP	128				x	

4.17.1 VCAP LPM SGL_IP4 Key Details

The SGL_IP4 key is to be used for IPv4 UC routing as well as IPv4 Source/Destination Guard.

Table 4-125. VCAP LPM SGL_IP4 Key Details

Field Name	Description	Size
DST_FLAG	0: IP4_XIP is only to be used for SIP matching. 1: IP4_XIP is only to be used for DIP matching.	1
AFFIX	LPM affix.	10
IP4_XIP	IPv4 address.	32

4.17.2 VCAP LPM DBL_IP4 Key Details

The DBL_IP4 key is to be used for IPv4 MC routing.

Table 4-126. VCAP LPM DBL_IP4 Key Details

Field Name	Description	Size
X2_TYPE	Must be set to 0.	1
AFFIX	LPM affix.	10
IP4_SIP	IPv4 source address.	32
IP4_DIP	IPv4 destination address.	32

4.17.3 VCAP LPM SGL_IP6 Key Details

The SGL_IP6 key is to be used for IPv6 UC routing as well as IPv6 Source/Destination Guard.

Table 4-127. VCAP LPM SGL_IP6 Key Details

Field Name	Description	Size
DST_FLAG	0: IP6_XIP is only to be used for SIP matching. 1: IP6_XIP is only to be used for DIP matching.	1
AFFIX	LPM affix.	10
IP6_XIP	IPv6 address.	128

4.17.4 VCAP LPM DBL_IP6 Key Details

The DBL_IP6 key is to be used for IPv6 MC routing.

Table 4-128. VCAP LPM DBL_IP6 Key Details

Field Name	Description	Size
AFFIX	LPM affix.	10
IP6_SIP	IPv6 source address.	128
IP6_DIP	IPv6 destination address.	128

4.17.5 VCAP LPM IP6PFX_ID Key Details

The IP6PFX_ID key is to be used for IPv6 UC Routing as well as IPv6 Source/Destination Guard. This key only applies when VCAP IP6PFX is also used.

Table 4-129. VCAP LPM IP6PFX_ID Key Details

Field Name	Description	Size
X2_TYPE	Must be set to 1.	1
DST_FLAG	0: IP6_XIP64 is only to be used for SIP matching. 1: IP4_XIP64 is only to be used for DIP matching.	1
AFFIX	LPM affix.	10
IP6PFX_ID	IP6PFX_ID from VCAP IP6PFX_ID lookup.	9
IP6_XIP64	IP6PFX_ID=0: Most significant 64 bits of IPv6 address. IP6PFX_ID>0: Least significant 64 bits of IPv6 address.	64

4.17.6 VCAP LPM Actions

VCAP LPM supports three different X1 actions:

- ARP_PTR
- L3MC_PTR
- ARP_ENTRY

The following table lists the actions and the key types for which they are intended to be used.

Table 4-130. VCAP LPM Action Selection

Action Name	Description and Appropriate VCAP LPM Key Types
ARP_PTR	Provides pointer to ARP entry in ARP Table (ANA_L3:ARP). LPM key types: SGL_IP4, SGL_IP6, IP6PFX_ID.
L3MC_PTR	Provides pointer to L3MC Table (ANA_L3:L3MC). LPM key types: DBL_IP4, DBL_IP6.
ARP_ENTRY	ARP entry. LPM key types: SGL_IP4, SGL_IP6, IP6PFX_ID.

The following table provides information about the VCAP LPM actions. When programming an action in VCAP LPM, the associated action fields listed must be programmed in the listed order, with the first field in the table starting at bit 0 of the action.

Table 4-131. VCAP LPM Actions

Field Name	Description and Encoding	Size	ARP_PTR	L3MC_PTR	ARP_ENTRY
Action Type	0: ARP_PTR. 1: L3MC_PTR. 2: ARP_ENTRY. 3: Reserved.	2	x	x	x
ARP_PTR	Pointer to entry in ARP Table (ANA_L3:ARP).	11	x		
ARP_PTR_REMAP_ENA	If this bit is set, ARP_PTR is used to point to an entry in the ARP pointer remap table (ANA_L3: ARP_PTR_REMAP).	1	x		

.....continued

Field Name	Description and Encoding	Size	ARP_PTR	L3MC_PTR	ARP_ENTRY
ECMP_CNT	Number of equal cost, multiple paths routes to DIP. See 4.18.2.1.4 IP Multicast Routing.	4	x		
RGID	Route Group ID. Used for SIP RPF check. See 4.18.2.1.5 SIP RPF Check.	3	x		
L3MC_PTR	Pointer to entry in L3MC Table (ANA_L3:L3MC).	11		x	
MAC_MSB	See ANA_L3:ARP:ARP_CFG_0.MAC_MSB.	16			x
MAC_LSB	See ANA_L3:ARP:ARP_CFG_1.MAC_LSB.	32			x
ARP_VMID	See ANA_L3:ARP:ARP_CFG_0.ARP_VMID.	9			x
ZERO_DMAC_CPU_QU	See ANA_L3:ARP:ARP_CFG_0.ZERO_DMAC_CPU_QU.	3			x
SIP_RPF_ENA	See ANA_L3:ARP:ARP_CFG_0.SIP_RPF_ENA.	1			x
SECUR_MATCH_VMID_ENA	See ANA_L3:ARP:ARP_CFG_0.SECUR_MATCH_VMID_ENA.	1			x
SECUR_MATCH_MAC_ENA	See ANA_L3:ARP:ARP_CFG_0.SECUR_MATCH_MAC_ENA.	1			x
ARP_ENA	See ANA_L3:ARP:ARP_CFG_0.ARP_ENA.	1			x
ENCAP_ID	See ANA_L3:ARP:ARP_ENCAP.ENCAP_ID.	10			x
RSDX	See ANA_L3:ARP:ARP_MISC.RSDX.	12			x

4.18 IP Processing

This section provides information about IP routing and IP security checks. The configuration parameters for IP routing are located within the ANA_L3 configuration target.

Each frame is subject to two lookups in VCAP LPM. One lookup is used for looking up SIP, and the other lookup is used for looking up DIP or DIP + SIP (for IP multicast frames).

4.18.1 IP Source/Destination Guard

For security purposes, the device can be configured to identify specific combinations of the following information and apply security rules based on that information.

- (SMAC, SIP) or (VMID, SIP)
- (DMAC, DIP) or (VMID, DIP)

The VCAP LPM and ARP table in ANA_L3 are used to perform matching. The result of the matching is available for security rules in ANA_ACL through the following VCAP fields.

- L3_SMAC_SIP_MATCH. Set to 1 if (VMID, SIP) and/or (SMAC, SIP) match was found.
- L3_DMAC_DIP_MATCH. Set to 1 if (VMID, DIP) and/or (DMAC, DIP) match was found.

IP source/destination guard check can be enabled per port using COMMON:SIP_SECURE_ENA and COMMON:DIP_SECURE_ENA.

When enabled, the frame's DIP and the frame's SIP are each looked up in VCAP LPM. The associated VCAP action provides an index to an ARP Table entry in which the required SMAC/DMAC and/or VMID is configured.

The following pseudo code specifies the behavior of the IP Source Guard checks.

```
// Determine value of req.l3_sip_match (available in ANA_ACL as L3_SMAC_SIP_MATCH)
if (!req.ip4_avail && !req.ip6_avail) {
  req.l3_sip_match = 1;
  return;
}
if (csr.sip_cmp_ena(req.port_num)) {
  req.l3_sip_match = 0;
} else {
  req.l3_sip_match = 1;
}
if (!LpmHit()) {
  return;
}
if (req.ip4_avail) {
  csr.secur_ip4_lpm_found_sticky = 1;
}
if (req.ip6_avail) {
  csr.secur_ip6_lpm_found_sticky = 1;
}
sip_arp_entry = csr.arp_tbl[sip_lpm_entry.base_ptr +
                            (ecmp_ac % (sip_lpm_entry.ecmp_cnt+1))];
if ((sip_arp_entry.secur_match_vmid_ena == 0 ||
     igr_vlan_entry.vmid == sip_arp_entry.arp_vmid)
    &&
    (sip_arp_entry.secur_match_mac_ena == 0 ||
     req.smac == sip_arp_entry.mac)) {
  req.l3_sip_match = 1;
  if (req.ip4_avail) {
    csr.secur_ip4_sip_match_sticky = 1;
  } else {
    csr.secur_ip6_sip_match_sticky = 1;
  }
}
if (req.l3_sip_match == 0) {
  csr.secur_sip_fail_sticky = 1;
}
```

The IP Destination Guard check works in the same way as the SIP check, except for the following:

- DIP check is not performed if frame has a multicast DMAC.
- DIP check is not performed if router leg has been matched.

The following pseudo code specifies the behavior of the IP Destination Guard checks.

```
// Determine value of req.l3_sip_match (available in ANA_ACL as L3_SMAC_SIP_MATCH)
if (!req.ip4_avail && !req.ip6_avail) {
  req.l3_sip_match = 1;
  return;
}
if (csr.sip_cmp_ena(req.port_num)) {
  req.l3_sip_match = 0;
} else {
  req.l3_sip_match = 1;
}
if (!LpmHit()) {
  return;
}
if (req.ip4_avail) {
  csr.secur_ip4_lpm_found_sticky = 1;
}
if (req.ip6_avail) {
  csr.secur_ip6_lpm_found_sticky = 1;
}
sip_arp_entry = csr.arp_tbl[sip_lpm_entry.arp_ptr];
if ((sip_arp_entry.secur_match_vmid_ena == 0 ||
```

```
     igr_vlan_entry.vmid == sip_arp_entry.arp_vmid)
    &&
    (sip_arp_entry.secur_match_mac_ena == 0 ||
     req.smac == sip_arp_entry.mac)) {
  req.l3_sip_match = 1;
  if (req.ip4_avail) {
    csr.secur_ip4_sip_match_sticky = 1;
  } else {
    csr.secur_ip6_sip_match_sticky = 1;
  }
}
if (req.l3_sip_match == 0) {
  csr.secur_sip_fail_sticky = 1;
}
```

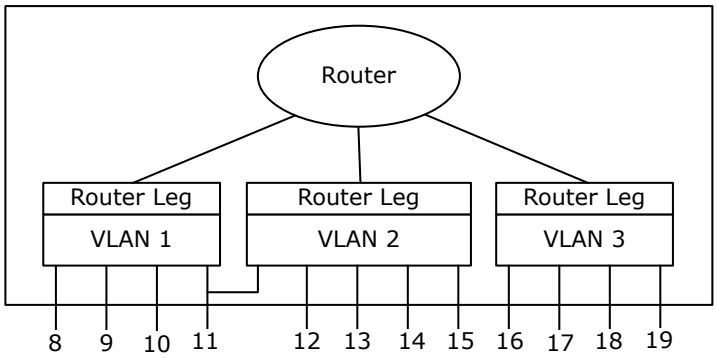
4.18.2 IP Routing

The device supports routing of IPv4 and IPv6 unicast and multicast packets through the 511 router interfaces, also known as “router legs”. Each router leg is attached to a VLAN.

The following illustration shows a configuration with three VLANs, each with an attached router leg for routing IP packets between the VLANs. Port 11 is member of both VLAN 1 and VLAN 2.

When a packet is being routed from one VLAN (the ingress VLAN) to another VLAN (the egress VLAN), the VID of the ingress VLAN is termed IVID, and the VID of the egress VLAN is termed the EVID.

Figure 4-32. Router Model



Within the device, a router leg is identified by a 9-bit VMID value. When a packet is being routed, it is then received by the router entity using a router leg identified by an ingress VMID (or IVMID) and transmitted on an egress router leg identified by an egress VMID (or EVMID).

IP routing, also known as L3 forwarding, is only supported in VLAN-aware mode.

4.18.2.1 Frame Types for IP Routing

In order for IP packets to be routed by the device, the following conditions must be met.

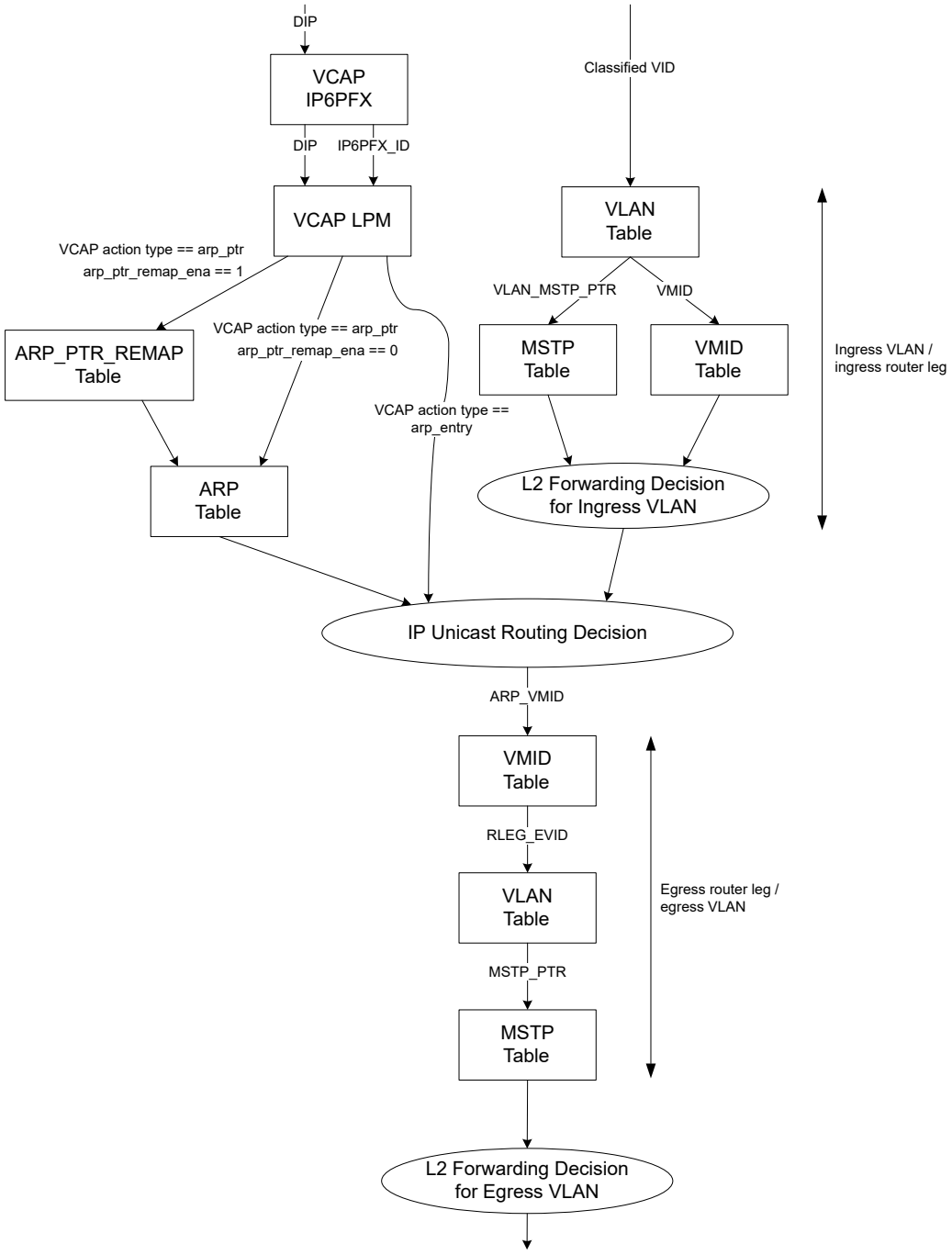
- IPv4 packets must not contain options.
- IPv6 packets must not contain hop-by-hop options.
- Optionally packets with some illegal SIP and DIP addresses can be filtered. For more information, see Illegal IPv4 Addresses and Illegal IPv6 Addresses.
- IPv4 header checksum must be correct.

4.18.2.1.1 Routing Table Overview

The following figure provides an overview of the table lookups involved in IP unicast routing within the L3 part of the analyzer.

The VCAP and ARP table lookups are performed in parallel with the VLAN/MSTP/VMID table lookups. If the frame does not match a router leg, the result of VCAP and ARP table lookups are not used for routing.

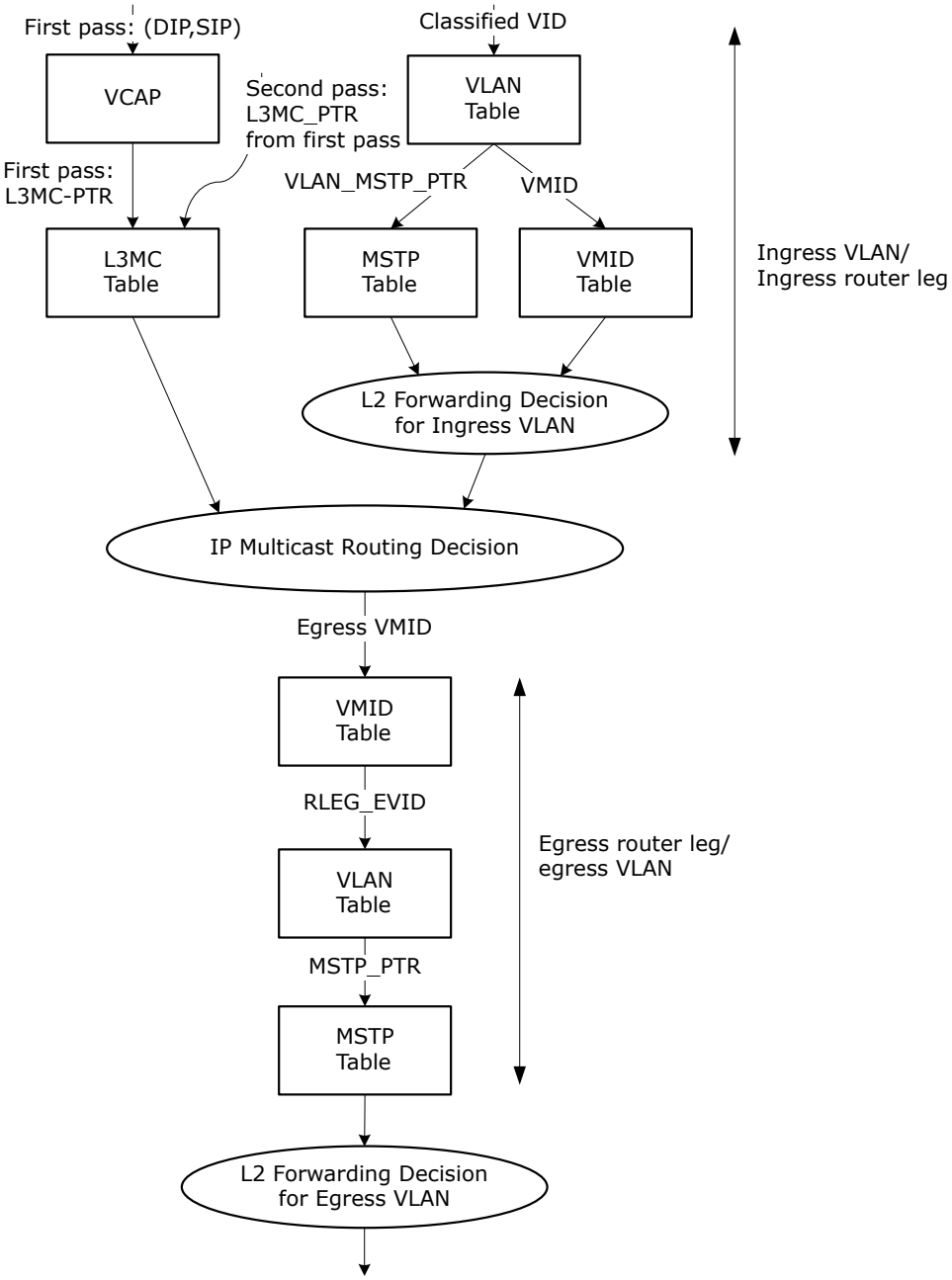
Figure 4-33. Unicast Routing Table Overview



The following figure provides an overview of the table lookups involved in IP multicast routing within the L3 part of the analyzer.

The VCAP and L3MC table lookups are performed in parallel with the VLAN/MSTP/VMID table lookups. If the frame does not match a router leg, the result of VCAP and L3MC table lookups are not used for routing.

Figure 4-34. Multicast Routing Table Overview



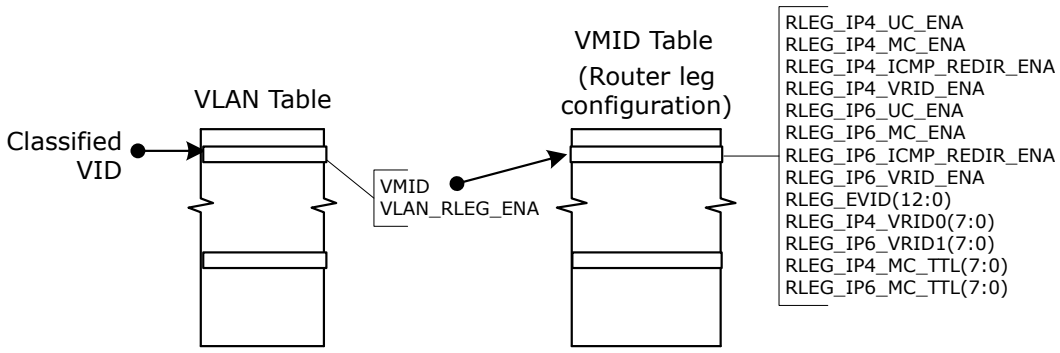
4.18.2.1.2 Ingress Router Leg Lookup

The VID used to determine the ingress router leg is the classified VID. The classified VID can be deducted based on a variety of parameters, such as VLAN tag or ingress port. For more information, see 4.14.1.8 VLAN Classification.

A total of 511 router legs are supported. Each router leg can be used for both IPv4 and IPv6 unicast and multicast routing. A maximum of one router leg per VLAN is supported.

When processing incoming frames, the classified VID is used to index the VLAN table. The RLEG_ENA parameter determines if the VLAN is enabled for routing of packets, and if so, the VMID is used to index the VMID table. The flow is depicted in the following illustration.

Figure 4-35. Ingress Router Leg Lookup Flow



Unicast Router Leg Detection

On L2, up to three different MAC addresses are used to identify the router leg:

- The normal router leg MAC address: This MAC address consists of a base address that is configured using COMMON:RLEG_CFG_0.RLEG_MAC_LSB and COMMON:RLEG_CFG_0.RLEG_MAC_MSB.

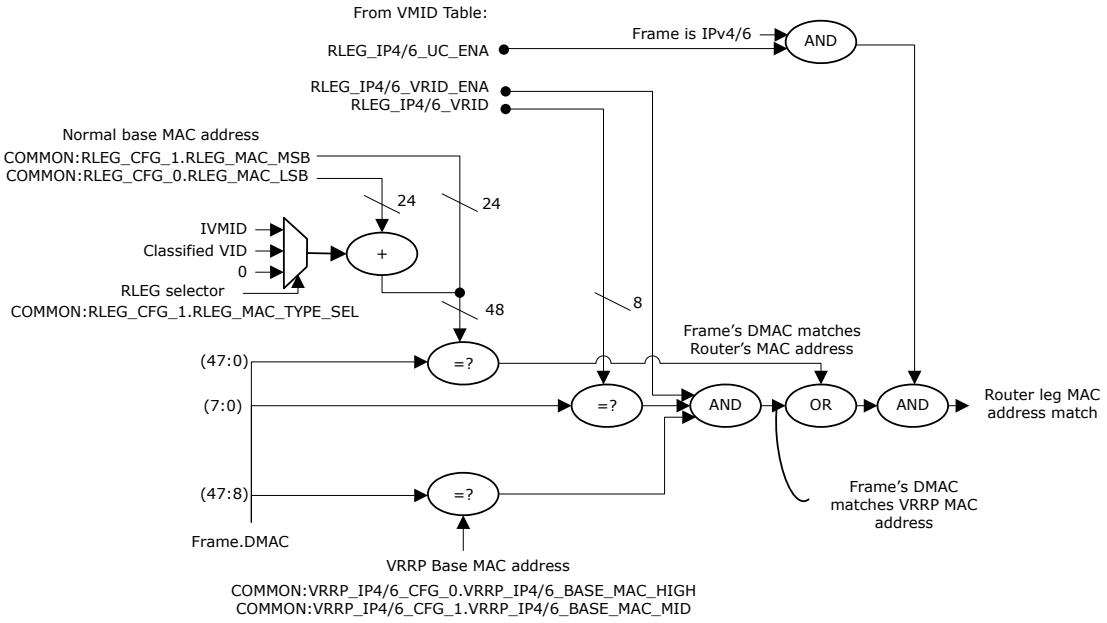
  To have different MAC addresses for each router leg, the three least significant bytes of the base MAC address can optionally be incremented by the Classified VID or the VMID. This is configured using COMMON:RLEG_CFG_1.RLEG_MAC_TYPE_SEL.
- VRRP MAC address for IPv4: This MAC address consists of a base address that is configured using COMMON:VRRP_IP4_CFG_0.VRRP_IP4_BASE_MAC_MID and COMMON:VRRP_IP4_CFG_0.VRRP_IP4_BASE_MAC_HIGH.

  For each router leg, the least signficant byte is configured per router leg using VMID:VRRP_CFG.RLEG_IP4_VRID. Up to two VRID values are supported per router leg.

  Use of VRRP for IPv4 is enabled using VMID:RLEG_CTRL.RLEG_IP4_VRID_ENA.
- VRRP MAC address for IPv6: This is configured in the same manner as VRRP MAC address for IPv4.

The matching of router leg MAC address for unicast packets is shown in the following figure.

Figure 4-36. Ingress Router Leg MAC Address Matching for Unicast Packets



The packet is a candidate for unicast routing if a frame's DMAC matches one of the router leg MAC addresses. Packets sent to the router itself will also match the router leg MAC address, but these will be caught through entries in the ARP table.

Despite matching a router leg MAC address, routing is not performed for any of the following reasons. If any of following criteria are met, the frame may be redirected to the CPU, depending on configuration parameters.

- It is not an IP packet (for example, an ARP reply).
- There are IP header errors.
- DIP or SIP address is illegal (optional).
- Packet contains IPv4 options/IPv6 hop-by-hop options.
- IPv4 TTL or IPv6 HL is less than two.

Multicast Router Leg Detection

For each router leg, IP multicast routing can be enabled using VMID:RLEG_CTRL.RLEG_IP4_MC_ENA and VMID:RLEG_CTRL.RLEG_IP6_MC_ENA.

In addition to having multicast routing enabled for the router leg configured for the VLAN, the frame's DMAC must be within a specific range in order for the packet to be candidate for IPv4/6 routing.

- IPv4 DMAC range: 01-00-5E-00-00-00 to 01-00-5E-7F-FF-FF
- IPv6 DMAC range: 33-33-00-00-00-00 to 33-33-FF-7F-FF-FF
  Despite having a DMAC address within the required range, routing is not performed for any of the following reasons:

Despite having a DMAC address within the required range, routing is not performed for any of the following reasons.

- There are IP header errors.
- DIP or SIP address is illegal (optional).
- Packet contains IPv4 options/IPv6 hop-by-hop options
- IPv4 TTL or IPv6 HL is less than two.

If any of above criteria are met, then the frame may be redirected to CPU, depending on configuration parameters.

The frame is still L2 forwarded if no router leg match is found.

Illegal IPv4 Addresses

Use the ROUTING_CFG.IP4_SIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following SIP address ranges to be considered illegal.

- 0.0.0.0 to 0.255.255.255 (IP network zero)
- 127.0.0.0 to 127.255.255.255 (IP loopback network)
- 224.0.0.0 to 255.255.255.255 (IP multicast/experimental/broadcast)

Frames with an illegal SIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

Use the ROUTING_CFG.IP4_DIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following DIP address ranges to be considered illegal.

- 0.0.0.0 to 0.255.255.255 (IP network zero)
- 127.0.0.0 to 127.255.255.255 (IP loopback network)
- 240.0.0.0 to 255.255.255.254 (IP experimental)

Frames with an illegal DIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

Illegal IPv6 Addresses

Use the ROUTING_CFG.IP6_SIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following SIP address ranges to be considered illegal.

- ::/128 (unspecified address)
- ::1/128 (loopback address)
- ff00::/8 (IPv6 multicast addresses)

Frames with an illegal SIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

Use the ROUTING_CFG.IP6_DIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following DIP address ranges to be considered illegal.

- ::/128 (unspecified address)
- ::1/128 (loopback address)

Frames with an illegal DIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

4.18.2.1.3 Unicast Routing

The IPv4/IPv6 unicast forwarding table in the analyzer is configured with known routes. There are typically two types of routes: Local IP and Remote IP.

- Local IP networks directly accessible through the router. These entries return the DMAC address of the destination host and correspond to an ARP lookup.
- Remote IP networks accessible through a router. These entries return the DMAC address of the next-hop router.

If frames are deemed suitable for IP unicast routing, a DIP lookup is performed in the Longest Prefix Match (LPM) table. The lookup is a Classless Inter-Domain Routing (CIDR) lookup. That is, all network sizes are supported.

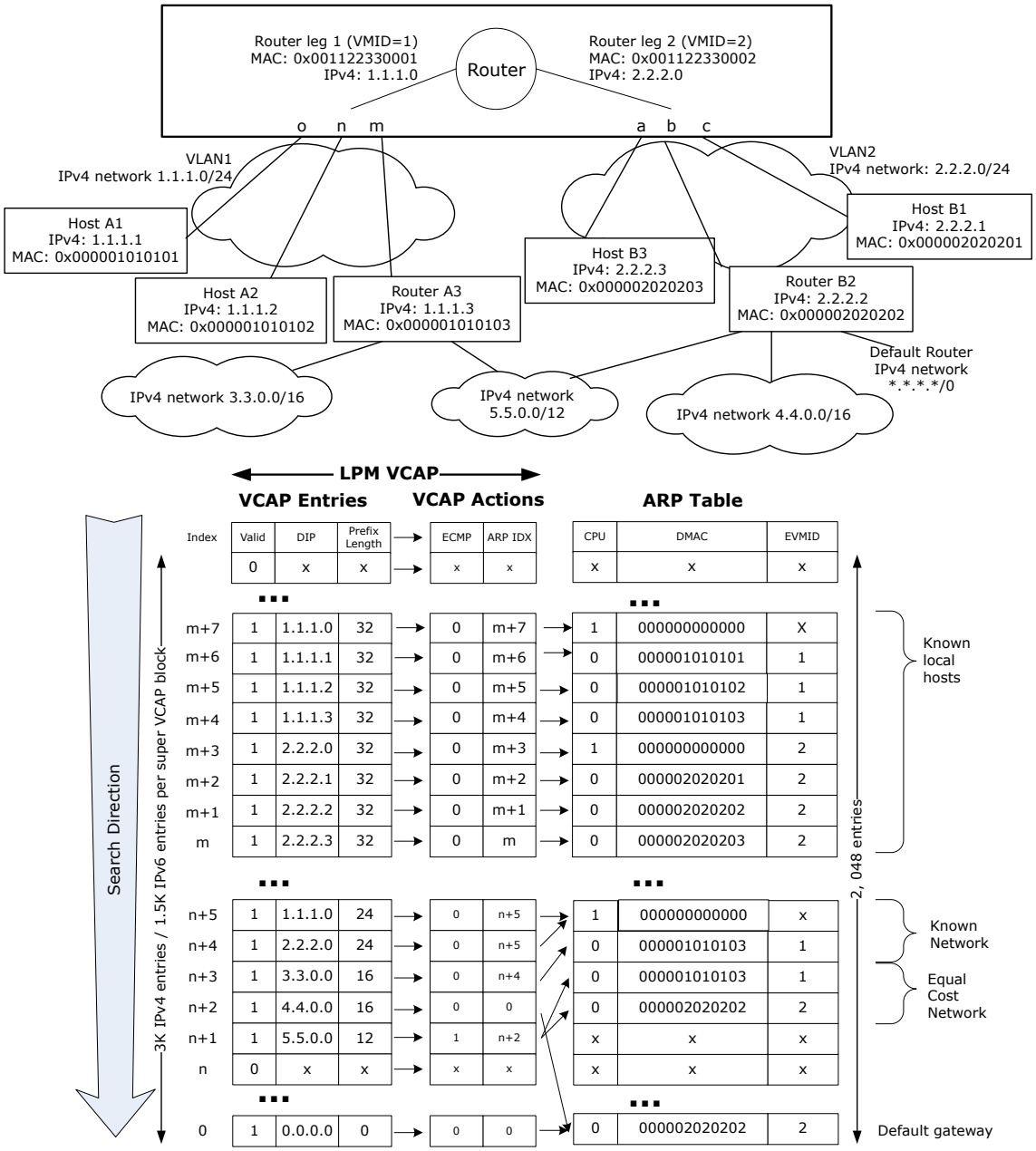
For IP addresses in local IP networks without corresponding MAC addresses, LPM entry with a corresponding ARP entry with a zero MAC address should be configured. This results in redirecting the matching frames to the CPU and thus allows the CPU to exchange ARP/NDP messages with the stations on the local IP networks to request their MAC address and afterwards use this information to update the ARP table.

Packets with IVMID = EVMID (packets that are to be routed back to the router leg on which they were received) can optionally be redirected to the CPU such that the CPU can generate an ICMP Redirect message. This is configurable using VMID:RLEG_CTRL.RLEG_IP4_ICMP_REDIR_ENA and VMID:RLEG_CTRL.RLEG_IP6_ICMP_REDIR_ENA. If such packets are not redirected to the CPU, they are routed.

IVMID and EVMID are configured using VLAN:VMID_CFG.VMID and ARP:ARP_CFG_0.ARP_VMID.

The following figure shows an IPv4 unicast routing example. IPv6 unicast routing works identically to IPv4 unicast routing, with the exception that each of the IPv6 addresses occupies twice or four times as much space in VCAP LPM, depending on whether IP6PFX VCAP is used.

Figure 4-37. IP Unicast Routing Example



Ports o, n, and m are connected to Router A3, Host A1, and Host A2. These stations are included in an IP subnet associated with VLAN 1 and connected to the router by router leg 1. Router leg 1's MAC address is shown.

Ports a, b, and c are connected to Router B2, Host B1, and Host B3. These stations are included in an IP subnet associated with VLAN 2 and connected to the router by router leg 2.

The router leg MAC addresses in the example offsets the base MAC address by the respective router leg's VMID.

Traffic destined within a subnet is L2-forwarded. For example, traffic from Host A1 to Host A2 is L2 forwarded.

Traffic destined outside a subnet is sent to the router using the router leg's MAC address. The router performs the routing decision as follows:

1. Known local host: If the destination IP address is known to be a local host (with the full IP address installed in the VCAP LPM table), the router takes the local host DMAC address and egress router leg VMID (EVMID) from the ARP table entry pointed to by the ARP_IDX of the VCAP Action. For example, if traffic sent from Host A1 to B1 is routed using LPM entry m+2 (2.2.2.1/32) and ARP entry m + 2, the local host's DMAC address is found to be 0x000002020201 and egress router leg to be RLEG2 (that is, VMID = 2).
2. Known longest prefix: If the full destination IP address is not known, the router finds the longest matching IP prefix and uses it to determine the next hop DMAC address and egress router leg. For example, traffic sent from host A1 to host 4.4.4.2 is routed using LPM entry n + 2 (the 4.4.0.0 subnet is reached through Router B2) and using ARP entry 0, the next hop's DMAC address is found to be 0x000002020202 and egress router leg to be RLEG2 (for example, VMID = 2).
3. Known equal cost multiple paths (ECMP): If the full destination IP address is not known by the router and the longest matching IP prefix is an equal cost path, the next hop DMAC address and egress router leg is derived from one of up to 16 ARP entries. For example, traffic sent from host A1 to host 5.5.5.13 is routed using VCAP LPM entry n + 1. The corresponding VCAP action has ECMP = 1, so the ARP table entry is chosen to be either n + 2 or n + 3, based on an ECMP aggregation code, see 4.14.1.10 ECMP Aggregation Code Generation. If entry n + 2 is chosen, the next hop's DMAC address is found to be 0x000001010103 (Router A3) and egress router leg to be RLEG1. If entry n + 3 is chosen, the next hop's DMAC address is found to be 0x000002020202 (Router B2) and egress router leg to be RLEG2.
4. Default gateway: In order to forward packets destined to unknown subnets to a default router, a default rule can be configured for the VCAP LPM. This is illustrated in the IP Unicast Routing example as a "0.0.0.0" rule. For example, traffic sent from host A1 to host 8.9.4.2 is routed using LPM entry 0 (the default router is reached through Router B2) and ARP entry 0, the next hop's DMAC address is found to be 0x000002020202 and egress Router Leg to be RLEG2.

When routing packets, the router replaces the frame's DMAC with the ARP table's DMAC and replaces the SMAC with MAC address associated with the egress router leg. The classified VID is replaced by the egress VID (VMID:RLEG_CTRL.RLEG_EVID). The TTL/HL is decremented by 1. Note that the VMID tables (ANA_L3:VMID and REW:VMID) and the router leg MAC address must be configured consistently in the analyzer Layer 3 and rewriter.

If host A2 wants to send traffic to a host in subnet 3.3.0.0/16, the host issues an ARP request and gets a response to send traffic to router A3. Host A2 can choose to ignore this request, in which case, the router routes to router A3 in the same VLAN.

Local networks 1.1.1.0/24 and 2.2.2.0/24 are also configured. Hosts and routers located in these networks can be directly L2 accessed. This includes IP addresses 1.1.1.0 and 2.2.2.0, which are the IP addresses of the router in the respective subnets to be used for IP management traffic, for example.

Encoding ARP Entry in VCAP Action

Instead of using an entry in the ARP table, the ARP information can alternatively be written into the VCAP action of the LPM entry so as to not be limited by the size of the ARP table.

ARP entries written into a VCAP actions have the following limitations compared to ARP entries in the ARP table.

- ECMP cannot be supported. If there are multiple paths to a given subnet, then ARP table entries must be used.
- RGID cannot be configured. If RGIDs of SIP RPF are used, then ARP table entries must be used for such routes.
- Alternative Next Hop configuration cannot be supported. For example, if Alternative Next Hop configuration is required, then ARP table entries must be used.

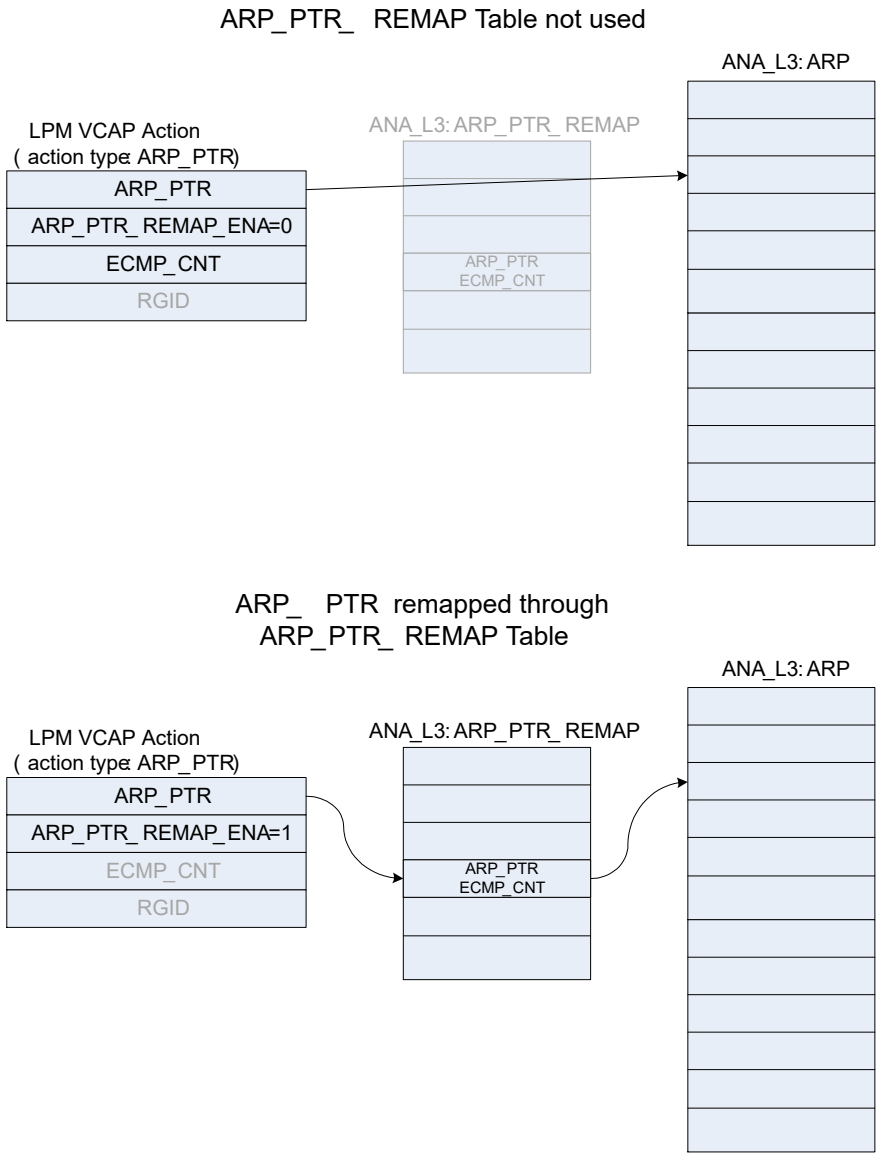
ARP Pointer Remap

An ARP Pointer Remap table is available to support fast fail-over when the next hop changes for a large group of LPM VCAP entries.

By setting ARP_PTR_REMAP_ENA=1 in LPM VCAP action, the ARP_PTR in the VCAP action is used to address an entry in the ARP_PTR_REMAP table and through this, new ARP_PTR and ECMP_CNT values are looked up and used for lookup in the ARP table. Thus instead of updating ARP_PTR and ECMP_CNT for many LPM VCAP entries, only the corresponding entry in ARP_PTR_REMAP needs to be updated.

The following illustration depicts both where ARP Pointer remapping is not used, as well as where it is used; that is, ARP_PTR_REMAP_ENA = 1.

Figure 4-38. ARP Pointer Remapping



4.18.2.1.4 IP Multicast Routing

The multicast system is split into two stages, stage 1 and stage 2.

In stage 1, the frame is received on a front port and the IP multicast system in ANA_L3 is activated. This stage is used to ensure L2 forwarding to all relevant ports in the ingress VLAN as well as ingress mirroring. If the frame needs to be L3 forwarded to other VLANs, an internal port module called virtual device 0 (VD0) also receives an L2 copy, together with information about the number of VLANs that receive a routed copy of the frame.

In stage 2, the analyzer processes each frame copy from VD0, and L3 forwards each frame copy to its egress VLAN.

In stage 1, ANA_L3 uses the VCAP LPM to determine the forwarding of the frame. Typically, two types of VCAP LPM multicast entries exist: source-specific and source-independent.

- Source specific IP multicast groups, where the same group IP address can be used by multiple sources. Such groups are denoted (S,G).

- Source independent IP multicast groups, where the group IP address uniquely identifies the IP multicast flow. Such groups are denoted (*,G).

IPv4 multicast groups occupies two entries in the VCAP LPM, and IPv6 multicast groups occupies eight entries. Source-specific multicast groups must be placed with higher precedence than source-independent multicast groups.

A matching entry in the VCAP LPM results in an index (L3MC_IDX) to an entry in the L3MC Table.

Each L3MC table entry contains a list of egress router legs to which frames are routed. Upon removing the ingress router leg from the list, the required number of routed multicast copies is calculated and this number (L3MC_COPY_CNT) is sent to VD0.

Stage 1 L2 forwards the frame. ANA_ACL can be used to limit the L2 forwarding, for example, to support IGMP/MLD snooping.

For each L3MC entry, it can optionally be configured that routing is only performed if the frame was received by a specific ingress router leg. This is termed reverse path forwarding check and is configured using L3MC:L3MC_CTRL.RPF_CHK_ENA and L3MC:L3MC_CTRL.RPF_VMID.

Note that this RPF check is unrelated to the SIP RPF check. For more information about the SIP RPF check, see 4.18.2.1.5 SIP RPF Check.

Reverse path forwarding check does not affect the L2 forwarding decision.

In stage 2, VD0 replicates the frame according to L3MC_COPY_CNT, and L3MC_IDX is written into the frame's IFH.

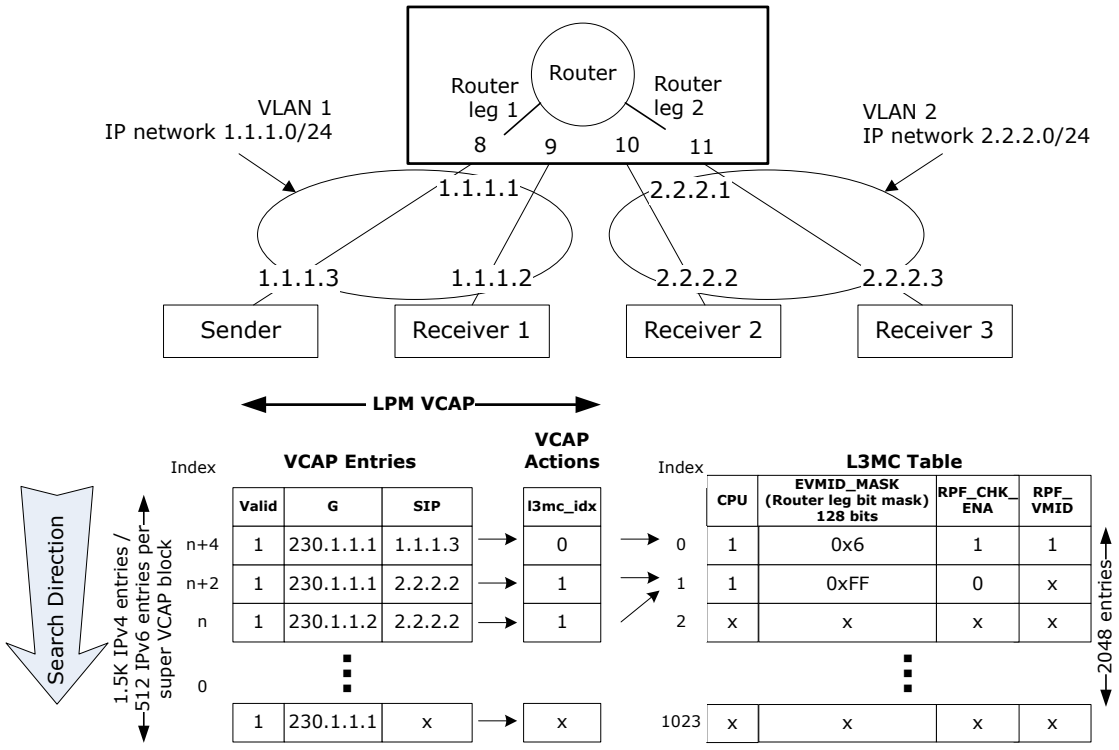
For each copy, which ANA_L3 receives from VD0 during stage 2, L3MC_IDX from IFH is used to lookup the L3MC table entry and thus find the list of egress router legs that require a routed copy. The egress router leg is used to determine the egress VID which in turn is used to perform (VID, DMAC) lookup in the MAC table for L2 forwarding within the egress VLAN.

For each egress router leg, it can be configured that frames are only L3 forwarded to the router leg if the TTL/HL of the packet is above a certain value. This is configured using VMID:VMID_MC.RLEG_IP4_MC_TTL and VMID:VMID_MC.RLEG_IP6_MC_TTL.

In stage 2, both learning and ingress filtering is disabled.

The following figure shows an IPv4 multicast routing example. IPv6 multicast routing works identically to IPv4 multicast routing, with the exception that the IPv6 entries occupies four times as much space in VCAP LPM.

Figure 4-39. IP Multicast Routing Example



A sender host with IPv4 address 1.1.1.3 is transmitting IPv4 multicast frames to the group address 230.1.1.1 and DMAC = 0x01005E010101. The frames must be L2 forwarded to receivers in VLAN 1 and L3 forwarded to receivers in VLAN 2. For the latter, an entry for the (1.1.1.3, 230.1.1.1) pair is added to the VCAP LPM.

Stage 1

- Lookup of (SIP,G) = (1.1.1.3, 230.1.1.1) in VCAP LPM returns L3MC_IDX = 0 and a corresponding entry in L3MC table. L3MC_IDX = 0 is inserted into the IFH.
- The L3MC entry has RPF check enabled (RPF_ENA = 1). Because the frame has been received through the specified router leg, L3 forwarding is allowed.
- The L3MC entry specifies that frames are forwarded to router leg 1 and 2. Because the frame has been received on router leg 1, only router leg 2 needs a routed copy. In other words, the number of L3 frame copies required is set to 1.
- If the frame's TTL is <2, then the frame is not L3 forwarded.
- Lookup of (DMAC, VID) = (0x01005E010101, 1) in the MAC table returns a destination set with port 8 and 9. Port 8 is source filtered due to normal L2 source port filtering, so the resulting destination set consists only of port 9.
- As a result of the first stage, the frame has been L2 forwarded to port 9 and a copy with L3MC_IDX = 0 and L3 copies required = 1 has been forwarded to the VD0.

Stage 2

- A frame is received once from VD0 with L3MC_IDX = 0 and L3MC_COPY_CNT = 1.
- Lookup of L3MC_IDX = 0 in the L3MC table returns the same entry as used in stage 1. Because this is the first routed copy, the new EVMID is 2, and through lookup in the VMID table, the corresponding EVID is 2.
- If Frame.IP.TTL < VMID.VMID_MC.RLEG_IP4_MC_TTL, the frame is not L3 forwarded.
- The frame's TTL is decremented.
- Lookup of (DMAC, VID) = (0x01005E010101, 2) in the MAC table returns a destination set with port 10 and 11. Because this is a routed copy, no source port filtering is applied.

- The result is a routed copy to ports 10 and 11 with replaced SMAC and VID from VMID = 2 and TTL decremented.

L3MC: Mask Mode and List Mode

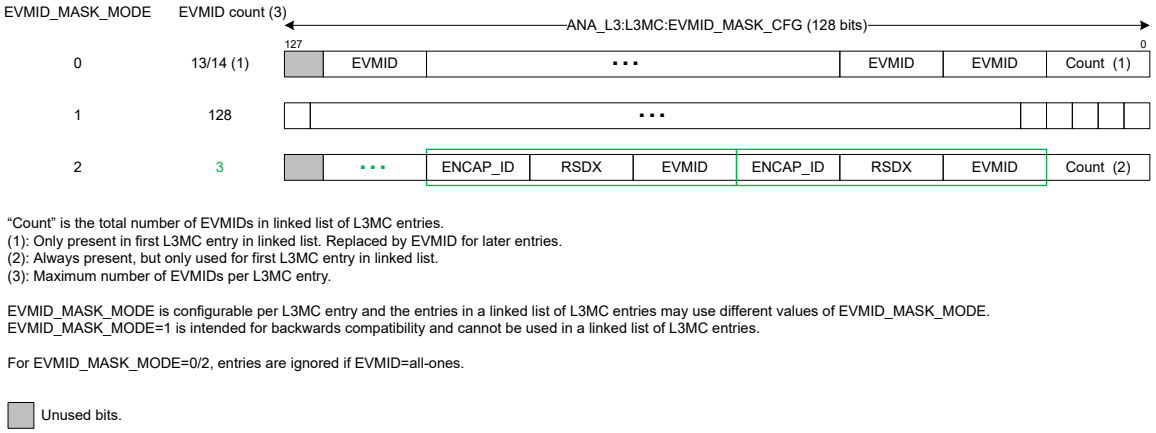
The set of router legs in ANA_L3:L3MC can be specified in three ways:

- Mask Mode
- List Simple Mode
- List Encap Mode

When using Mask Mode, the set of router legs are specified as a bit mask with one bit per router leg. Router legs 0-127 can be supported.

When using List Simple Mode or List Encap Mode, the set of router legs are specified as router leg numbers. For the two List modes, each L3MC entry can hold a number of router leg numbers and multiple L3MC entries can be linked together to enable forwarding to more router legs. The full range of router legs can be supported.

Figure 4-40. EVMID_Mask Encoding



"Count" is the total number of EVMIDs in linked list of L3MC entries.
(1): Only present in first L3MC entry in linked list. Replaced by EVMID for later entries.
(2): Always present, but only used for first L3MC entry in linked list.
(3): Maximum number of EVMIDs per L3MC entry.

EVMID_MASK_MODE is configurable per L3MC entry and the entries in a linked list of L3MC entries may use different values of EVMID_MASK_MODE.
EVMID_MASK_MODE=1 is intended for backwards compatibility and cannot be used in a linked list of L3MC entries.

For EVMID_MASK_MODE=0/2, entries are ignored if EVMID=all-ones.

Unused bits.

4.18.2.1.5 SIP RPF Check

As a security measure, the device can be configured to only L3 forward frames if the SIP belongs to a known subnet and is received by one (or more) specific router legs, for example, the router leg through which the subnet is reached. This configuration can be used to filter IP address spoofing attempts.

To support SIP RPF check, each route in VCAP LPM is assigned a Route Group ID (RGID); that is, any route in the VCAP LPM belongs to exactly one Route Group. Eight RGID values are supported.

In the VMID table the set of acceptable RGIDs for each router leg is configured using RGID_MASK.

Each router leg can be configured to operate in the following SIP RPF modes.

- Disabled: No SIP RPF check is performed.
- RGID mode: Frames are only L3 forwarded if the SIP's RGID is enabled in the router leg's RGID_MASK. If the ARP information is encoded directly in the VCAP action and SIP RPF mode is set to RGID mode, then an Rleg mode check is performed instead, because an ARP entry encoded in the VCAP Action contains to RGID.
- Rleg mode: Frames are only L3 forwarded if the VMID of the ARP table entry resulting from SIP LPM lookup is identical to the ingress VMID. For example, the router leg, which the frame was received on, is also the router leg used for forwarding frames to the SIP.

  Rleg mode cannot be used for ECMP LPM entries, because each route is only accepted by one router leg. In Rleg mode, no SIP RPF check is performed if SIP is reachable using an ECMP path.
- Combined mode: This is a combination of RGID mode and Rleg mode. If the SIP LPM lookup results in an LPM entry with ECMP_CNT = 0, an Rleg mode check is performed, otherwise a RGID Mode check is performed.

RFC3704 defines a number of different SIP RPF modes, which can be supported by the device as follows:

- Strict RPF: Use Rleg mode or Combined mode to also support ECMP.
- Loose RPF: Use RGID mode and configure RGID_MASK to all-ones. For example, any RGID value is acceptable.
- Loose RPF ignoring default routes: Use RGID mode and configure a separate RGID value for the default route. Exclude this value in router leg's RGID_MASK.
- Feasible RPF: Use RGID mode, group LPM entries using RGID and enable only the accepted groups in the router leg's RGID_MASK.

Note that SIP RPF is unrelated to the L3MC RPF check enabled in L3MC:L3MC_CTRL.RPF_CHK_ENA, which validates that a given IP multicast flow (identified by (S,G) or (*,G) was received by the expected router leg.

Frames that are not routed due to SIP RPF check can optionally be redirected to CPU by configuring COMMON:ROUTING_CFG.RLEG_IP4_SIP_RPF_REDIR_ENA and COMMON:ROUTING_CFG.RLEG_IP6_SIP_RPF_REDIR_ENA.

4.18.2.1.6 LPM Affix

When IP addresses are being looked up in the VCAP LPM, an LPM affix is part of the key. This affix defaults to 0, but can be set to other values using VCAP CLM.

Using the LPM affix, the VCAP LPM can be split into multiple logically separate routing tables, e.g. to support features such as VRF-lite.

4.18.2.1.7 CPU Redirection of Filtered Frames

The following table provides an overview of the available CPU queues used when copying/redirecting frames to CPU.

Table 4-132. CPU Queue Overview

CPU Queue	Description
COMMON:CPU_QU_CFG.CPU_RLEG_QU	Non-IP unicast frames matching an ingress router leg. For example, ARP PDUs. CPU queue for IP frames with L2 broadcast DMAC, received by router leg.
COMMON:CPU_QU_CFG.CPU_RLEG_IP_OPT_QU	IPv4 frames with options and IPv6 frames with hop-by-hop option.
COMMON:CPU_QU_CFG.CPU_RLEG_IP_HDR_FAIL_QU	IPv4 frames with IP header errors.
COMMON:CPU_QU_CFG.CPU_IP_LEN_QU	CPU queue for IPv4/IPv6 frames failing MTU check.
COMMON:CPU_QU_CFG.CPU_MC_FAIL_QU	Failed IP multicast lookup or failed RPF check.
COMMON:CPU_QU_CFG.CPU_UC_FAIL_QU	Failed IPv4/IPv6 unicast LPM lookup, invalid ARP entry (ARP_ENA=0) or failed ICMP redirect check.
COMMON:CPU_QU_CFG.CPU_IP_TTL_FAIL_QU	Frames with TTL/HL <2.
COMMON:CPU_QU_CFG.CPU_SIP_RPF_QU	Frames failing SIP RPF check.
ARP:ARP_CFG_0.DMAC0_CPU_QU	MAC address in ARP entry is all-zeros.
L3MC:L3MC_CTRL.CPU_QU	L3MC.L3MC_CTRL.CPU_REDIR_ENA = 1.

4.18.3 Statistics

For each frame, ANA_L3 generates a number of events that can be counted in ANA_AC's statistics block. Events are generated separately for ingress and egress router legs, and in ANA_AC, event counting is further split into IPv4 and IPv6.

4.18.3.1 Ingress Router Leg Statistics

The ingress router leg statistics events generated by ANA_L3 are listed in the following table.

Table 4-133. Ingress Router Leg Events

Event	Description
ivmid_ip_uc_received	IP unicast frame received by router.
ivmid_ip_mc_received	IP multicast frame received by router.
ivmid_ip_uc_routed	IP unicast frame received and L3 forwarded by router.
ivmid_ip_mc_routed	IP multicast frame received and L3 forwarded by router.
ivmid_ip_mc_rpf_discarded	IP multicast frame received by router, but discarded due to MC RPF check.
ivmid_ip_ttl_discarded	IP multicast frame received by router, but discarded due to TTL <2.
ivmid_ip_acl_discarded	IP frame received by router, but discarded by ACL rules in ANA_AC.

For counting of ingress router leg events, eight counter pairs are provided per router leg. Each pair consists of a counter for IPv4 and a counter for IPv6, for a total of 16 counters that are available per router leg.

For each of the eight counter pairs, an event mask can be configured, specifying which of the Ingress router leg events listed will trigger the counter. Each counter pair can be configured to count either bytes or frames.

An example usage of the counters is shown in the following table.

Table 4-134. Ingress Router Leg Statistics Example

Reference	Counter	Byte (b/) or Frame (f)	Event						
			ivmid_ip_uc_received	ivmid_ip_mc_received	ivmid_ip_uc_routed	ivmid_ip_mc_routed	ivmid_ip_mc_rpf_discarded	ivmid_ip_ttl_discarded	ivmid_ip_acl_discarded
RFC 4293	ipIfStatsInReceives	f	x	x					
RFC 4293	ipIfStatsInOctets	b	x	x					
RFC 4293	ipIfStatsInForwDatagrams	f			x	x			
RFC 4293	ipIfStatsInMcastPkts	f		x					
RFC 4293	ipIfStatsInMcastOctets	b		x					
RFC 4293	ipIfStatsInHdrErrors	f						x	
Microchip	MC RPF discards	f					x		
Microchip	ACL discards	f							x

4.18.3.2 Egress Router Leg Statistics

The egress router leg statistics events generated by ANA_L3 are listed in the following table.

Table 4-135. Egress Router Leg Events

Event	Description
evmid_ip_uc_routed	IP unicast frame L3 forwarded by router.
evmid_ip_mc_routed	IP multicast frame L3 forwarded by router.
evmid_ip_mc_switched	IP multicast frame L2 forwarded by switch.
evmid_ip_mc_ttl_discarded	IP multicast frame discarded by router due to TTL value configured for egress router leg. See VMID:VMID_MC:RLEG_IP4_MC_TTL and VMID:VMID_MC:RLEG_IP6_MC_TTL.
evmid_ip_acl_discarded	IP frame discarded by ACL rules in ANA_AC.

For counting of egress router leg events, eight counter pairs are provided per router leg. Each pair consists of a counter for IPv4 and a counter for IPv6, for a total of 16 counters available per router leg.

For each of the eight counter pairs, an event mask can be configured to specify which of the events listed in the following table will trigger the counter. Each counter pair can be configured to count either bytes or frames.

An example usage of the counters is shown in the following table.

Table 4-136. Egress Router Leg Statistics Example

Reference	Counter	Byte (b/) or Frame (f)	Event				
			evmid_ip_uc_routed	evmid_ip_mc_routed	evmid_ip_mc_switched	evmid_ip_ttl_discarded	evmid_ip_acl_discarded
RFC 4293	ipIfStatsOutForwDatagrams	f	x	x			
RFC 4293	ipIfStatsOutOctets	b	x	x			
RFC 4293	ipIfStatsOutMcastPkts	f		x			
RFC 4293	ipIfStatsOutMcastOctets	b		x			
Microchip	IP MC TTL discards	f				x	
Microchip	ACL discards	f					x

4.18.3.3 LPM Statistics

For debugging purposes and usage monitoring, the VCAP includes a sticky bit per entry.

4.18.4 IGMP/MLD Snooping Switch

RFC4541 describes recommendations on how an IGMP/MLD snooping switch forwards IP Multicast packets.

To implement the required data forwarding rules for an IGMP/MLD snooping switch, the VCAP IS2 lookup in ANA_ACL must be used.

For each IP multicast flow, the VCAP IS2 lookup can be used to limit the forwarding within each VLAN using the following key types.

- IGMP snooping switch: IP4_VID
- MLD snooping switch: IP6_VID

Both key types include SIP, DIP, and VID such that snooping switches can be supported for IGMPv3 and MLDv2.

In the action of the VCAP rules, the following fields can be used to limit the forwarding.

- PORT_MASK. The ports to which the frame is forwarded within the VLAN.
- MASK_MODE. Should normally be set to 1 (AND_VLANMASK).

When IP multicast routing is combined with snooping, a lookup in VCAP IS2 is made for each copy of the IP multicast packet. The VID in each such lookup is the VID of the egress VLAN (of that copy). In order for the VCAP rules to use the EVID in the IP4_VID/IP6_VID key, the second VCAP lookup must be used, and ANA_ACL::VCAP_S2_CFG.SEC_ROUTE_HANDLING_ENA must be set to 1.

4.18.5 IP-in-IPv4

This section discusses IP-in-IPv4.

4.18.5.1 Overview

When IP-in-IPv4 is being used, the device is situated between two routing domains and IP-in-IP is used to forward frames from one routing domain through the other routing domain.

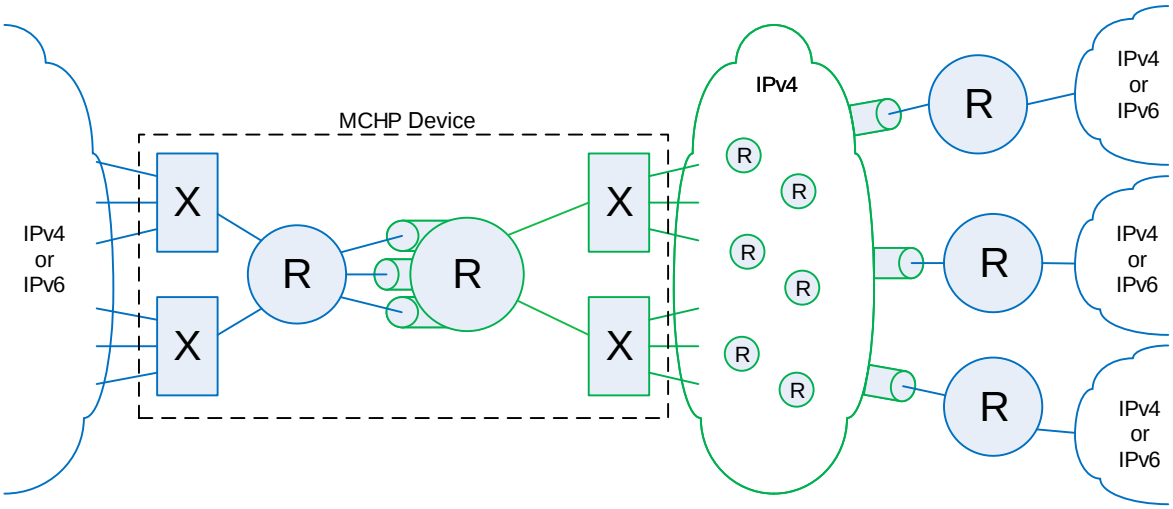
The inner IP layer may be either IPv4 or IPv6. The outer IP layer must be IPv4.

The following shows a configuration, where IP-in-IPv4 "tunnels" are used to tunnel IP frames from a blue IP domain, through a green IPv4 domain to other "islands" of the blue IP domain.

When entering a tunnel, frames are being routed in the blue IP domain, encapsulated in an IP layer of the green domain and afterwards the frame is being routed in the green domain. When exiting a tunnel, frames are being routed in the green domain, decapsulated and afterwards the frame is routed in the blue domain.

Before and after routing in the two domains, frames are bridged in the ingress and egress VLANs. Each of the bridges (marked with "X") in the following represent bridging within a VLAN.

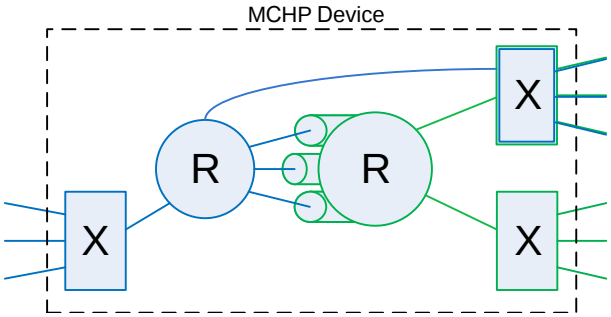
Figure 4-41. Router Leg-Based IP-in-IP Tunneling



The frames forwarded between the blue and the green routers are pure IP packets, i.e. they have no associated Ethernet protocol layer including no associated VLAN or ingress port number.

The blue and the green IP domains may co-exist in the same VLAN, in which case a configuration as shown in the following may result.

Figure 4-42. IP-in-IP Tunneling and Two IP Domains in One VLAN



In order to have the device route the frame twice (once for the blue IP domain and once for the green IP domain), the frame must be processed twice by ANA-QSYS-REW.

This is done by forwarding the frame to Virtual Device 2 (VD2) after the first processing round together with information on which router leg the frame is to be received on for the second processing round.

In the first round:

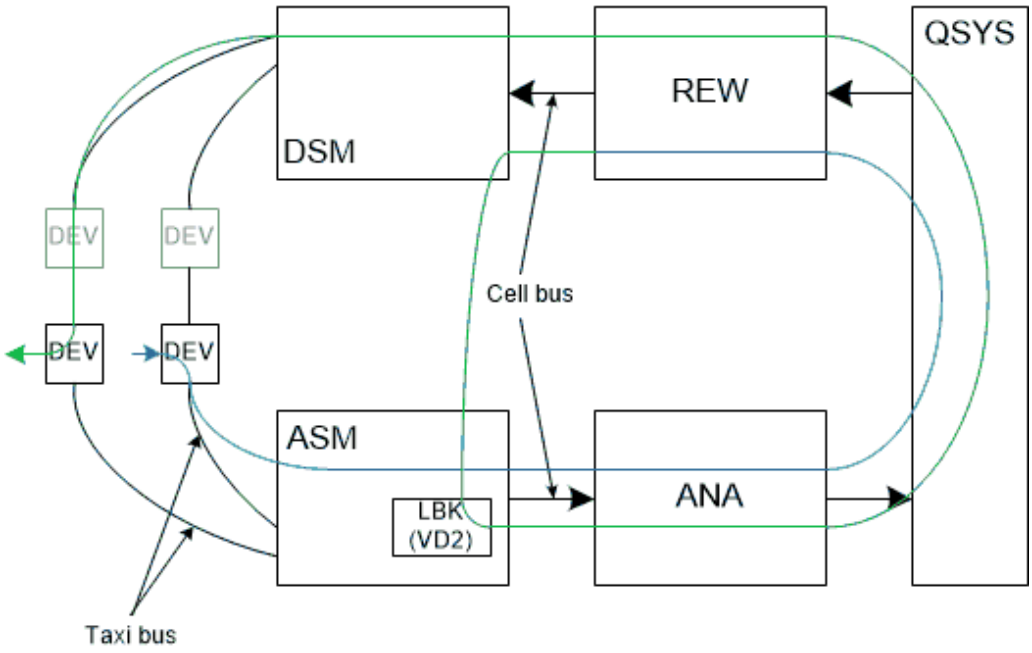
1. L2 forward the frame within the ingress VLAN.
2. When encapsulating:
   - Route within the inner IP domain.
   - Pop the Ethernet layer.
   - Push an IP layer as well as a possibe GRE layer for the outer IP domain.
3. When decapsulating:
   - Route within the outer IP domain.
   - Pop the Ethernet layer.
   - Verify validity of the outer IP layer.
   - Pop the outer IP layer as well as a possible GRE layer.
4. Forward the frame to VD2 for second round processing.

In the second round:

1. When encapsulating: Route within the outer IP domain.
2. When decapsulating: Route within the inner IP domain.
3. L2 forward the frame within the egress VLAN.

The following figure shows the forwarding flow for an IP unicast packet being forwarded from the blue domain to the green domain.

Figure 4-43. Ip-In-Ipv4 Encapsulation Unicast Forwarding Flow



For IP multicast packets, each round consists of two passes: One pass for possibly L2 forwarding and one pass (through VD0) for L3 forwarding into each of the egress VLAN(s).

4.18.5.2 Encapsulation Control

When a packet is encapsulated with an additional IP layer, the encapsulating IP layer - the "outer IP layer" - is controlled by an "ENCAP_ID" resulting from processing in ANA_L3.

When routing in ANA_L3 results in ENCAP_ID!=0, then REW will encapsulate the IP packet in an additional IP layer (outer IP layer) and forward the frame back to ANA for routing in the outer IP domain.

For IP UC (inner IP layer), ENCAP_ID can be configured using the parameters listed by order of preference below. ENCAP_ID=0 means unspecified. The first non-zero ENCAP_ID takes precedence.

1. ENCAP_ID in LPM action of type "ARP_ENTRY".
2. ANA_L3:ARP:ARP_ENCAP.ENCAP_ID
3. ANA_L3:VMID:VMID_ENCAP.ENCAP_ID (egress router leg)

If none of above parameters result in a non-zero ENCAP_ID, then no IP encapsulation will be added to the frame.

For IP MC (inner IP layer), ENCAP_ID can be configured using the parameters listed by order of preference below. ENCAP_ID=0 means unspecified. The first non-zero ENCAP_ID takes precedence.

1. ANA_L3:L3MC:EVMID_MASK_CFG (per egress router leg) by setting EVMID_MASK_MODE=2
2. ANA_L3:VMID:ARP_ENCAP.ENCAP_ID (egress router leg)

If none of above parameters result in a non-zero ENCAP_ID, then no IP encapsulation will be added to the frame.

For non-zero ENCAP_ID, ENCAP_ID is used by REW to lookup encapsulation in REW:ENCAP_IP4.

4.18.5.3 Decapsulation Control

In order to enable ANA_L3 to decapsulate IP-in-IP frames, ANA_CL analyzes the frame to determine the IP-in-IP encapsulation type (if any).

To specify that frames must be decapsulated, the following parameter is used.

ANA_L3:VMID:VMID_ENCAP.DECAP_ENA (egress router leg)

To verify that the outer IP layer fulfills requirements for SIP and/or encapsulation type the following parameters can be used:

ANA_L3:VMID:VMID_ENCAP.DECAP_SIP_ID

ANA_L3:VMID:VMID_ENCAP.DECAP_ENCAP_TYPE

When an IP MC frame is decapsulated, the DMAC for the new Ethernet layer is generated (in the second round) based on and RFC 1112 and RFC 2464.

4.18.5.4 Simple Tunnel

For "simple tunnels", the egress router leg of the encapsulating (blue) router maps 1:1 to a tunnel which forwards all frames to a given remote (blue) router.

In this scenario, encapsulation is selected using the following encapsulation parameters.

ANA_L3:VMID:VMID_ENCAP:ENCAP_ID

REW:ENCAP:ENCAP_IP4 (indexed by ENCAP_ID)

In order to verify the SIP and encapsulation type before decapsulating frames, the following parameters are used.

ANA_L3:VMID:VMID_ENCAP.DECAP_SIP_ID

ANA_L3:VMID:VMID_ENCAP.DECAP_ENCAP_TYPE

4.18.5.5 Support for IPv6 over IPv4 Standards

A number of RFCs cover different ways of forwarding IPv6 over IPv4.

Most of these RFCs either view the IPv4 network as a link layer below IPv6 or specify various algorithms for mapping IPv6 addresses to IPv4 addresses.

By selecting encapsulation in LPM action or in ANA_L3:ARP it is possible to map IPv6 UC addresses to corresponding IPv4 addresses.

By selecting encapsulation in ANA_L3:L3MC it is possible to map IPv6 MC addresses to IPv4 addresses.

In order to monitor which LPM entries are being actively used, the VCAP provides an activity bit per entry.

4.18.5.6 Encapsulation Formats

This section describes the IP-in-IP encapsulations supported. The Ethernet layer is not shown.

The IP layers for the green and blue IP domains in Figure 4-41 are shown in green and blue, and encapsulation specific information - the "glue" - is show in brown.

4.18.5.6.1 IP-over-GRE-over-IPv4 Encapsulation Format

The following code-block shows an example of the IPv4/IPv6 over GRE (without checksum) over IPv4 (without options) encapsulation format.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version|  IHL  |Type of Service|          Total Length         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Identification        |Flags|      Fragment Offset    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Time to Live |  Protocol=47  |         Header Checksum       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Source Address                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Destination Address                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |C|       Reserved0       | Ver |         Protocol Type         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version|  ...                                                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

C=0 (no checksum present)
Ver=0
Protocol Type=0x0800/0x86DD for IPv4/IPv6
Version=4/6
```

The following code-block shows an example of IPv4/IPv6 over GRE (without checksum) over IPv4 (without options) encapsulation format for which the frame reception GRE checksum is supported.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version|  IHL  |Type of Service|          Total Length         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Identification        |Flags|      Fragment Offset    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Time to Live |  Protocol=47  |         Header Checksum       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Source Address                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Destination Address                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |C|       Reserved0       | Ver |         Protocol Type         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      Checksum                 |       Reserved1               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version|  ...                                                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

C=1 (checksum present)
Ver=0
Protocol Type=0x0800/0x86DD for IPv4/IPv6
Version=4/6
```

4.18.5.6.2 IP-over-IPv4 Encapsulation Format

Most IPv6-over-IPv4 RFCs use Protocol=41 to specify that the inner protocol layer is IPv6.

uses Protocol=4 to specify that the inner protocol layer is IPv4.

The following code-block shows an example of the IP-over-IPv4 encapsulation format.

```
0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version|  IHL  |Type of Service|          Total Length         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Identification        |Flags|      Fragment Offset    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Time to Live | Protocol=4/41 |         Header Checksum       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Source Address                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Destination Address                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version|  ...                                                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Version=4/6
```

4.19 VCAP IS2 Keys and Actions

VCAP IS2 is part of ANA_ACL and enables access control lists using VCAP functionality. This section provides detailed information about all available VCAP IS2 keys and actions.

For information about how to select which key to use and how the VCAP IS2 action is applied, see 4.20.1 VCAP IS2.

4.19.1 VCAP IS2 Keys

VCAP IS2 supports a number of different keys that can be used for different purposes and frame types. The keys are of type X3, X6, or X12 depending on the number of words each key uses. The keys are grouped after size as shown in the following table.

Table 4-137. VCAP IS2 Keys and Sizes

Key Name	Key Size	Number of Words	Key Type
IP4_VID	104 bits	3 words	X3 type
MAC_ETYPE	288 bits	6 words	X6 type
ARP	226 bits		
IP4_TCP_UDP	293 bits		
IP4_OTHER	292 bits		
CUSTOM_2	285 bits		
IP6_VID	302 bits		
IP_7TUPLE	613 bits	12 words	X12 type
CUSTOM_1	608 bits		

The following table lists details for all VCAP IS2 keys and fields. When programming an entry in VCAP IS2, the associated key fields must be programmed in the listed order with the first field in the table starting at bit 0 in the entry. As an example, for a MAC_ETYPE entry, the first field X6_TYPE must be programmed at bits 3:0 of the entry, the second field FIRST must be programmed at bit 4, and so on.

Table 4-138. VCAP IS2 Key Overview

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
X6_TYPE	X6 type. 0: MAC_ETYPE 3: ARP 4: IP4_TCPUDP 5: IP4_OTHER 8: CUSTOM_2 9: IP6_VID	4		x	x	x	x	x	x		
X12_TYPE	X12 type. 1: IP7_TUPLE 2: CUSTOM_1	2								x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x	x	x	x	x	x	x
PAG	Classified Policy Association Group (PAG). PAG can be used to tie VCAP CLM lookups with VCAP IS2 lookups. The default PAG value is configurable per port in ANA_CL:PORT:PORT_ID_CFG.PAG.VAL.	8	x	x	x	x	x	x	x	x	x

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
IGR_PORT_MASK_L3	If set, IGR_PORT_MASK, IGR_PORT_MASK_RNG, and IGR_PORT_MASK_SEL are used to specify L3 interfaces.	1		x	x	x	x	x		x	x
IGR_PORT_MASK_RN G	Range selector for IGR_PORT_MASK: For X6 keys: IGR_PORT_MASK contains ports/ router legs 32*IGR_PORT_MASK_RNG through 32*IGR_PORT_MASK_RNG+31. IGR_PORT_MASK_RNG can take values 0 through 2 for L2 interfaces and values 0 through 15 for L3 interfaces. For X12 keys: L2 interfaces: IGR_PORT_MASK_RNG always 0. IGR_PORT_MASK contains all ports. L3 interfaces: IGR_PORT_MASK contains router legs 64*IGR_PORT_MASK_RNG through 64*IGR_PORT_MASK_RNG+63. Note bit 64 in IGR_PORT_MASK is not used. IGR_PORT_MASK_RNG can take values 0 through 7.	4		x	x	x	x	x		x	x
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK. Applicable when IGR_PORT_MASK_L3 == 0. 0: DEFAULT 1: LOOPBACK 2: MASQUERADE 3: CPU_VD If a frame fulfills multiple of below criteria, then higher value of IGR_PORT_MASK_SEL takes precedence.	2		x	x	x	x	x		x	x
	IGR_PORT_MASK_SEL: DEFAULT Each bit in the mask correspond to physical ingress port.			x	x	x	x	x		x	x
	IGR_PORT_MASK_SEL: LOOPBACK Set for frames received from a loopback device. Each bit in the mask corresponds to the physical egress port on which the frame was looped.			x	x	x	x	x		x	x

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
	IGR_PORT_MASK_SEL: MASQUERADE Set for masqueraded frames if ANA_ACL::VCAP_S2_MISC_CTRL.MASQ_IGR_MAS K_ENA == 1. A frame is masqueraded when IFH.MISC.PIPELINE_ACT == INJ_MASQ. The bit corresponding to the masqueraded port is set.			x	x	x	x	x		x	x
	IGR_PORT_MASK_SEL: CPU_VD Set for the following frame types: CPU: CPU injected frames and ANA_ACL::VCAP_S2_MISC_CTRL.CPU_IGR_MASK _ENA == 1 VD: Frames received from VD0, VD1, or VD2 and ANA_ACL::VCAP_S2_MISC_CTRL.VD_IGR_MASK_ ENA == 1 The encoding of IGR_PORT_MASK depends on the size of the IGR_PORT_MASK, see the following. The size of IGR_PORT_MASK depends on the key type.			x	x	x	x	x		x	x
	IGR_PORT_MASK_SEL: CPU_VD 32-bit IGR_PORT_MASK For CPU and VD frame types: Bits 4:0: Bit set if physical src port is VD2, VD1, VD0, CPU1, or CPU0 Bits 5:11: Src port (possibly masqueraded) Bits 12:16: ifh.misc.pipeline_pt Bits 17:19: ifh.misc.pipeline_act In addition, IGR_PORT_MASK_RNG is set to 0.			x	x	x	x	x		x	x
IGR_PORT_MASK	Ingress port mask.	32		x	x	x	x	x			
IGR_PORT_MASK	Ingress port mask.	65								x	x
L2_MC	Set if frame's destination MAC address is a multicast address (bit 40 = 1).	1		x	x	x	x	x		x	x
L2_BC	Set if frame's destination MAC address is the broadcast address (FF-FF-FF-FF-FF-FF).	1		x	x	x	x	x		x	x
VLAN_TAGGED	Set if frame was received with a VLAN tag.	1		x	x	x	x	x		x	x

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
SERVICE_FRM	Set if classified ISDX > 0.	1	x	x	x	x	x	x	x	x	x
ISDX	Classified ISDX. RSDX when RSDX > 0 and: L3_DST == 1 or ANA_ACL::VCAP_S2_MISC_CTRL2.EGR_KEY_ENA == 1.	12	x	x	x	x	x	x	x	x	x
XVID	Classified VID. Egress VID when L3_RT == 1 and: L3_DST == 1 or ANA_ACL::VCAP_S2_MISC_CTRL2.EGR_KEY_ENA == 1.	13	x	x	x	x	x	x	x	x	x
DEI	Classified DEI.	1		x	x	x	x	x		x	x
PCP	Classified PCP.	3		x	x	x	x	x		x	x
L2_FWD	Set if the frame is allowed to be forwarded to front ports. L2_FWD can for instance be cleared due to MSTP filtering or frame acceptance.	1		x	x	x	x	x		x	x
L3_SMAC_SIP_MATCH	Match found in SIP security lookup in ANA_L3. See 4.18.1 IP Source/Destination Guard.	1	x	x		x	x	x	x	x	x
L3_DMAC_DIP_MATCH	Match found in DIP security lookup in ANA_L3. See 4.18.1 IP Source/Destination Guard.	1	x	x		x	x	x	x	x	x
L3_RT	Set if frame has hit a router leg.	1	x	x		x	x	x	x	x	x
L3_DST	Set if lookup is done for egress router leg. This may influence information used in fields such as IGR_PORT_MASK, XVID, and L4_RNG.	1	x	x		x	x	x	x	x	x
L2_DMAC	Destination MAC address.	48		x						x	
L2_SMAC	Source MAC address.	48		x	x					x	
ETYPE_LEN	Frame type flag indicating that the frame is EtherType encoded. Set if frame has EtherType >= 0x600.	1		x							

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
ETYPE	Frame's EtherType Overloading: OAM Y.1731 frames: ETYPE[6:0] contains OAM MEL flags, see the following. OAM_Y1731 is set to 1 to indicate the overloading of ETYPE. OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0b0000000 MEL=1: 0b0000001 MEL=2: 0b0000011 MEL=3: 0b0000111 MEL=4: 0b0001111 MEL=5: 0b0011111 MEL=6: 0b0111111 MEL=7: 0b1111111 Together with the mask, the following kinds of rules may be created: • Exact match. Fx. MEL = 2: 0b0000011 • Below. Fx. MEL< = 4: 0b000XXXX • Above. Fx. MEL> = 5: 0bXX11111 • Between. Fx. 3< = MEL< = 5: 0b00XX111 Where, 'X' means don't care.	16		x							
IP4	Frame type flag indicating the frame is an IPv4 frame. Set if frame is IPv4 frame (EtherType = 0x800 and IP version = 4)	1			x	x				x	
L2_PAYLOAD_ETYPE	Byte 0-7 of L2 payload after Type/Len field. Overloading: OAM Y.1731 frames (OAM_Y1731 = 1): Bytes 0–3: OAM PDU bytes 0–3 Bytes 4–5: OAM_MEPID Bytes 6–7: Unused.	64		x							

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
L3_FRAGMENT_TYPE	L3 Fragmentation type: 0: No Fragments: MF==0 && FO==0 1: Initial Fragments: MF==1 && FO == 0 2: Suspicious Fragment: FO!=0 && FO <= FRAGMENT_OFFSET_THRES 3: Valid Follow-up Fragment: FO > FRAGMENT_OFFSET_THRES Where, FRAGMENT_OFFSET_THRES is programmed in ANA_ACL::VCAP_S2_FRAGMENT_CFG.FRAGMEN T_OFFSET_THRES. MF: More Fragments flag in IPv4 header FO: Fragments Offset in IPv4 header	2			x	x					
L3_FRAG_INVLD_L4_L EN	Set if frame's L4 length is less than ANA_ACL:VCAP_S2_FRAGMENT_CFG.L4_MIN_LE N.	1			x	x					
ARP_ADDR_SPACE_O K	Set if hardware address is Ethernet.	1		x							
ARP_PROTO_SPACE_ OK	Set if protocol address space is 0x0800.	1		x							
ARP_LEN_OK	Set if Hardware address length = 6 (Ethernet) and IP address length = 4 (IP).	1		x							
ARP_TARGET_MATCH	Target Hardware Address = SMAC (RARP).	1		x							
ARP_SENDER_MATCH	Sender Hardware Address = SMAC (ARP).	1		x							
ARP_OPCODE_UNKNO WN	Set if ARP opcode is none of the below mentioned.	1		x							
ARP_OPCODE	ARP opcode. Only valid if ARP_OPCODE_UNKNOWN is cleared. 0: ARP request. 1: ARP reply. 2: RARP request. 3: RARP reply.	2		x							
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1				x	x				
L3_TTL_GT0	Set if IPv4 TTL / IPv6 hop limit is greater than 0.	1				x	x			x	

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
L3_TOS	Frame's IPv4/IPv6 DSCP and ECN fields.	8				x	x			x	
L3_IP4_DIP	IPv4 frames: Destination IPv4 address. IPv6 frames: Source IPv6 address, bits 63:32.	32	x		x	x	x				
L3_IP4_SIP	IPv4 frames: Source IPv4 address. IPv6 frames: Source IPv6 address, bits 31:0.	32	x		x	x	x				
L3_IP6_DIP	IPv6 frames: Destination IPv6 address. IPv4 frames: Bits 31:0: Destination IPv4 address.	128							x	x	
L3_IP6_SIP	IPv6 frames: Source IPv6 address. IPv4 frames: Bits 31:0: Source IPv4 address.	128							x	x	
DIP_EQ_SIP	Set if destination IP address is equal to source IP address.	1			x	x	x			x	
L3_IP_PROTO	IPv4 frames (IP4 = 1): IP protocol. IPv6 frames (IP4 = 0): Next header.	8					x				
TCP_UDP	Frame type flag indicating the frame is a TCP or UDP frame. Set if frame is IPv4/IPv6 TCP or UDP frame (IP protocol/next header equals 6 or 17).	1								x	
TCP	Frame type flag indicating the frame is a TCP frame. Set if frame is IPv4 TCP frame (IP protocol = 6) or IPv6 TCP frames (Next header = 6)	1				x				x	
L4_DPORT	TCP/UDP destination port. Overloading for IP_7TUPLE: Non-TCP/UDP IP frames: L4_DPORT = L3_IP_PROTO.	16				x				x	
L4_SPORT	TCP/UDP source port.	16				x				x	
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, custom. Input into range checkers is taken from classified results (VID, DSCP) and frame (SPORT, DPORT, custom).	16		x	x	x	x			x	
SPORT_EQ_DPORT	Set if TCP/UDP source port equals TCP/UDP destination port.	1				x				x	

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
SEQUENCE_EQ0	Set if TCP sequence number is 0.	1				x				x	
L4_FIN	TCP flag FIN.	1				x				x	
L4_SYN	TCP flag SYN.	1				x				x	
L4_RST	TCP flag RST.	1				x				x	
L4_PSH	TCP flag PSH.	1				x				x	
L4_ACK	TCP flag ACK.	1				x				x	
L4_URG	TCP flag URG.	1				x				x	
L3_PAYLOAD	Payload bytes after IP header. IPv4: IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header.	96					x				
L4_PAYLOAD	Payload bytes after TCP/UDP header Overloading for IP_7TUPLE: Non TCP/UDP frames (TCP_UDP = 0): Payload bytes 0–7 after IP header. IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header for non TCP/UDP IPv4 frames.	64				x				x	
OAM_CCM_CNTS_EQ0	Flag indicating if dual-ended loss measurement counters in CCM frames are used. This flag is set if the TxFCf, RxFCb, and TxFCb counters are set to all zeros.	1		x							
OAM_Y1731	Set if frame's EtherType = 0x8902. If set, ETYPE is overloaded with OAM MEL flags, See ETYPE.	1		x							
CUSTOM1	60 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, then payload is retrieved from a position following the Ethernet layer, for example, after DMAC, SMAC, 0–3 VLAN tags, and EtherType.	480									x
CUSTOM2	24 bytes payload from frame. Location in frame is controlled by VCAP CLM actions. Note that if frame_type==ETH, payload is retrieved from a position following the Ethernet layer, for example, after DMAC, SMAC, 0–3 VLAN tags and EtherType.	192						x			

4.19.2 VCAP IS2 Actions

VCAP IS2 supports only one action, which applies to all VCAP IS2 keys. The action is listed in the following table. When programming an action in VCAP IS2, the associated action fields listed must be programmed in the listed order with the first field in the table starting at bit 0 in the action.

Table 4-139. VCAP IS2 Actions

Action Name	Description	Size
IS_INNER_ACL	Set if VCAP IS2 action belongs to ANA_IN_SW pipeline point.	1
PIPELINE_FORCE_ENA	If set, use PIPELINE_PT unconditionally and set PIPELINE_ACT = NONE if PIPELINE_PT == NONE. Overrules previous settings of pipeline point.	1
PIPELINE_PT	Pipeline point used if PIPELINE_FORCE_ENA is set: 0: NONE 1: ANA_VRAP 2: ANA_PORT_VOE 3: ANA_CL 4: ANA_CLM 5: ANA_IPT_PROT 6: ANA_OU_MIP 7: ANA_OU_SW 8: ANA_OU_PROT 9: ANA_OU_VOE 10: ANA_MID_PROT 11: ANA_IN_VOE 12: ANA_IN_PROT 13: ANA_IN_SW 14: ANA_IN_MIP 15: ANA_VLAN 16: ANA_DONE	5
HIT_ME_ONCE	If set, the next frame hitting the rule is copied to CPU using CPU_QU_NUM. CPU_COPY_ENA overrules HIT_ME_ONCE implying that if CPU_COPY_ENA is also set, all frames hitting the rule are copied to the CPU.	1
INTR_ENA	If set, an interrupt is triggered when this rule is hit	1
CPU_COPY_ENA	If set, frame is copied to CPU using CPU_QU_NUM.	1

.....continued

Action Name	Description	Size
CPU_QU_NUM	CPU queue number used when copying to the CPU	3
CPU_DIS	If set, disable extraction to CPU queues from previous forwarding decisions. Action CPU_COPY_ENA is applied after CPU_DIS.	1
LRN_DIS	If set, autolearning is disallowed.	1
RT_DIS	If set, routing is disallowed. Only applies when IS_INNER_ACL is 0. See also IGR_ACL_ENA, EGR_ACL_ENA, and RLEG_STAT_IDX.	1
POLICE_ENA	If set, frame is policed by policer pointed to by POLICE_IDX.	1
POLICE_IDX	Policer index.	6
IGNORE_PIPELINE_CTRL	Ignore ingress pipeline control. This enforces the use of the VCAP IS2 action even when the pipeline control has terminated the frame before VCAP IS2.	1
RESERVED	Must be set to 0.	3
MASK_MODE	Controls how PORT_MASK is applied. 0: OR_DSTMASK: Or PORT_MASK with destination mask. 1: AND_VLANMASK: And PORT_MASK with VLAN mask. 2: REPLACE_PGID: Replace PGID port mask from MAC table lookup by PORT_MASK. 3: REPLACE_ALL: Use PORT_MASK as final destination set replacing all other port masks. Note that with REPLACE_ALL, the port mask becomes sticky and cannot be modified by subsequent processing steps. 4: REDIR_PGID: Redirect using PORT_MASK[11:0] as PGID table index. See PORT_MASK for extra configuration options. Note that the port mask becomes sticky and cannot be modified by subsequent processing steps. 5: OR_PGID_MASK: Or PORT_MASK with PGID port mask from MAC table lookup. 6: Does not apply. 7: Does not apply. The CPU port is untouched by MASK_MODE.	3

.....continued

Action Name	Description	Size
PORT_MASK	Port mask applied to the forwarding decision. MASK_MODE=4: PORT_MASK[53]: CSD_PORT_MASK_ENA. If set, CSD_PORT_MASK is AND'ed with destination result. PORT_MASK[52]: SRC_PORT_MASK_ENA. If set, SRC_PORT_MASK is AND'ed with destination result. PORT_MASK[51]: AGGR_PORT_MASK_ENA. If set, AGGR_PORT_MASK is AND'ed with destination result. PORT_MASK[50]: VLAN_PORT_MASK_ENA. If set, VLAN_PORT_MASK is AND'ed with destination result.	68
RSDX_ENA	If set, RSDX is set to RSDX_VAL.	1
RSDX_VAL	RSDX value.	12
MIRROR_PROBE	Mirroring performed according to configuration of a mirror probe. 0: No mirroring. 1: Mirror probe 0. 2: Mirror probe 1. 3: Mirror probe 2.	2
REW_CMD	Reserved	11
TTL_UPDATE_ENA	If set, IPv4 TTL or IPv6 hop limit is decremented. Overloading: If ACL_RT_MODE[3]=1 (SWAPPING) and ACL_RT_MODE[2:0]=4, 5, 6, or 7, the IPv4 TTL or IPv6 hop limit is preset to value configured in ANA_ACL::SWAP_IP_CTRL.	1
SAM_SEQ_ENA	If set, ACL_RT_MODE only applies to frames classified as SAM_SEQ.	1
TCP_UDP_ENA	If set, frame's TCP/UDP DPORT and SPORT are replaced with TCP_UDP_DPORT and TCP_UDP_SPORT.	1
TCP_UDP_DPORT	New TCP/UDP DPORT when TCP_UDP_ENA is set.	16
TCP_UDP_SPORT	New TCP/UDP SPORT when TCP_UDP_ENA is set.	16
MATCH_ID	Logical ID for the entry. The MATCH_ID is extracted together with the frame if the frame is forwarded to the CPU (CPU_COPY_ENA). The result is placed in IFH.CL_RSLT.	16
MATCH_ID_MASK	Mask used by MATCH_ID.	16
CNT_ID	Counter ID, used per lookup to index the 4K frame counters (ANA_ACL:CNT_TBL). Multiple VCAP IS2 entries can use the same counter.	12
SWAP_MAC_ENA	Swap MAC addresses.	1

.....continued

Action Name	Description	Size
ACL_RT_MODE	Controls routing updates in ANA_ACL and how ACL_MAC is applied: 0: No change. ROUTING mode: ACL_RT_MODE[3] = 0 (ROUTING mode): ACL_RT_MODE[0] = 1: Enable Unicast routing updates in ANA_ACL. ACL_RT_MODE[1] = 1: Enable Multicast routing updates in ANA_ACL. ACL_RT_MODE[2] = 1: Enable overloading of ACL_MAC field. SWAPPING mode: ACL_RT_MODE[3] = 1 (SWAPPING mode): ACL_RT_MODE[2:0] encodes a SWAPPING sub-mode: 0: Replace DMAC. 1: Replace SMAC. 2: Swap MAC addresses. Replace SMAC if new SMAC is multicast. 3: Swap MAC addresses if DMAC is unicast. Replace SMAC if DMAC is multicast. 4: Swap MAC addresses. Replace SMAC if new SMAC is multicast. Swap IP addresses. Replace SIP if new SIP is multicast. 5: Swap MAC addresses if DMAC is unicast. Replace SMAC if DMAC is multicast. Swap IP addresses if DIP is unicast. Replace SIP if DIP is multicast. 6: Swap IP addresses. Replace SIP if new SIP is multicast. 7: Replace SMAC. Swap IP addresses. Replace SIP if new SIP is multicast. Note that when replacing a MAC address, the new MAC address is taken from ACL_MAC. When replacing a SIP, the new SIP is taken from ANA_ACL::SWAP_SIP[POLICE_IDX].	4
ACL_MAC	MAC address used to replace either SMAC or DMAC based on ACL_RT_MODE. Overloading: ACL_RT_MODE[3:2] = 1. ACL_MAC[0]: PORT_PT_CTRL_ENA - Overrule PORT_PT_CTRL. ACL_MAC[7:1]: PORT_PT_CTRL_IDX - into ANA_AC_POL:PORT_PT_CTRL. ACL_MAC[17]: Enable use of ACL_MAC[30:18] as DLB policer index. ACL_MAC[30:18]: DLB policer index. ACL_RT_MODE(3) = 1: MAC address used to replace either SMAC or DMAC based on ACL_RT_MODE.	48

.....continued

Action Name	Description	Size
DMAC_OFFSET_ENA	If set, and ACL_RT_MODE[3:0] = 8: Use ANA_ACL:SWAP_IP_CTRL.DMAC_REPL_OFFSET_VAL to select number of bits to replace in DMAC, counting from MSB.	1
PTP_MASTER_SEL	PTP master selector. PTP_MASTER_SEL indexes the PTP configuration in ANA_ACL:PTP_MASTER_CFG used in Delay_Resp frames. See REW_CMD.	2
LOG_MSG_INTERVAL	Selects logMessageInterval used in multicast Delay_Resp frames Values is interpreted as a signed two's complement with values ranging from -8 to 7: LOG_MSG_INT=0: logMessageInterval = 0 ... LOG_MSG_INT=7: logMessageInterval = 7 LOG_MSG_INT=0: logMessageInterval = –8 ... LOG_MSG_INT=7: logMessageInterval = –1	4
SIP_IDX	Selects source IP address in ANA_ACL::SWAP_SIP when replacing the source IP address (see ACL_RT_MODE).	5
RLEG_STAT_IDX	RLEG statistics index into ANA_ACL::VCAP_S2_RLEG_STAT, which controls how the ACL action affects IRLEG and ERLEG statistics. See also IGR_ACL_ENA and EGR_ACL_ENA. Applies when RT_DIS is set. 0: No RLEG statistics index. 1: RLEG statistics index 0. 2: RLEG statistics index 1. 3: RLEG statistics index 2. 4: RLEG statistics index 3.	3
IGR_ACL_ENA	If set, ACL is acting before the router as an I-RACL in terms of routing statistics. Applies when RT_DIS is set.	1
EGR_ACL_ENA	If set, ACL is acting after the router as an E-RACL in terms of routing statistics. EGR_ACL_ENA takes precedence over IGR_ACL_ENA. Applies when RT_DIS is set.	1

4.20 Analyzer Access Control Lists

The analyzer access control list (ANA_ACL) block performs the following tasks.

- Controls the VCAP IS2 lookups in terms of key selection, evaluation of range checkers, and control of specific VCAP IS2 key fields such as IGR_PORT_MASK_SEL and VID.
- Generates VCAP IS2 keys and execute the four lookups.
- Updates VCAP IS2 statistics based on match results.
- Assigns actions from VCAP IS2 matching or default actions when there is no match.
- Controls routing-related frame rewrites.

The following sections provides specific information about each of these tasks.

4.20.1 VCAP IS2

Each frame is classified to one of nine overall VCAP IS2 frame types. The frame type determines which VCAP IS2 key types are applicable and controls which frame data to extract.

Table 4-140. VCAP IS2 Frame Types

Frame Type	Condition
IPv6 multicast frame	The Type/Len field is equal to 0x86DD. The IP version is 6. The destination IP address is a multicast address (FF00::/8). Special IPv6 frames: • IPv6 TCP frame: Next header is TCP (0x6). • IPv6 UDP frame: Next header is UDP (0x11). • IPv6 Other frame: Next header is neither TCP nor UDP.
IPv6 unicast frame	The Type/Len field is equal to 0x86DD. The IP version is 6. The destination IP address is not a multicast address. Special IPv6 frames: • IPv6 TCP frame: Next header is TCP (0x6). • IPv6 UDP frame: Next header is UDP (0x11). • IPv6 Other frame: Next header is neither TCP nor UDP
IPv4 multicast frame	The Type/Len field is equal to 0x800. The IP version is 4. The destination IP address is a multicast address (224.0.0.0/4). Special IPv4 frames: • IPv4 TCP frame: IP protocol is TCP (0x6). • IPv4 UDP frame: IP protocol is UDP (0x11). • IPv4 Other frame: IP protocol is neither TCP nor UDP.
IPv4 unicast frame	The Type/Len field is equal to 0x800. The IP version is 4. The destination IP address is not a multicast address. Special IPv4 frames: • IPv4 TCP frame: IP protocol is TCP (0x6). • IPv4 UDP frame: IP protocol is UDP (0x11). • IPv4 Other frame: IP protocol is neither TCP nor UDP.
(R)ARP frame	The Type/Len field is equal to 0x0806 (ARP) or 0x8035 (RARP).
OAM Y.1731 frame	The Type/Len field is equal to 0x8902 (ITU-T Y.1731).
SNAP frame	The Type/Len field is less than 0x600. The Destination Service Access Point field, DSAP is equal to 0xAA. The Source Service Access Point field, SSAP is equal to 0xAA. The Control field is equal to 0x3.
LLC frame	The Type/Len field is less than 0x600 The LLC header does not indicate a SNAP frame.

.....continued	
Frame Type	**Condition**
ETYPE frame	The Type/Len field is greater than or equal to 0x600, and the Type field does not indicate any of the previously mentioned frame types, that is, ARP, RARP, OAM, IPv4, or IPv6.

4.20.1.1 Port Configuration and Key Selection

This section provides information about special port configurations that control the key generation for VCAP IS2.

The following table lists the registers associated with port configuration for VCAP IS2.

Table 4-141. Port Configuration of VCAP IS2

Register	Description	Replication
ANA_ACL:PORT:VCAP_S2_KEY_SEL	Configuration of the key selection for the VCAP IS2	Per port per lookup
ANA_ACL::VCAP_S2_CFG	Enabling of VCAP IS2 lookups.	Per port per lookup

Each of the two lookups in VCAP IS2 must be enabled before use (VCAP_S2_CFG.SEC_ENA) for a key to be generated and a match to be used. If a lookup is disabled on a port, frames received by the port are not matched against rules in VCAP IS2 for that lookup.

Each port controls the key selection for each of the two lookups in VCAP IS2 through the VCAP_S2_KEY_SEL register. For some frame type (OAM Y.1731, SNAP, LLC, EYTYPE) the key selection is given as only the MAC_ETYPE key can be used. For others frames types (ARP and IP) multiple keys can be selected from. The following table lists the applicable key types available for each frame type listed in Table 4-140.

Table 4-142. VCAP IS2 Key Selection

Frame Type	Applicable VCAP IS2 keys
IPv6 multicast frames	IP6_VID IP_7TUPLE MAC_ETYPE
IPv6 unicast frames	IP_7TUPLE MAC_ETYPE
IPv4 multicast frames	IP4_VID IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
IPv4 unicast frames	IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
(R)ARP frames	ARP MAC_ETYPE
OAM Y.1731 frames	MAC_ETYPE
SNAP frames	MAC_ETYPE

.....continued	
Frame Type	**Applicable VCAP IS2 keys**
LLC frames	MAC_ETYPE
ETYPE frames	MAC_ETYPE

Note:
The key selection is overruled if VCAP CLM instructs the use of a custom key, where data from the frame is extracted at a VCAP CLM-selected point. For more information, see 4.20.1.3 VCAP IS2 Custom Keys.

For information about VCAP IS2 keys and actions, see 4.19 VCAP IS2 Keys and Actions.

4.20.1.2 Key Selection

This section provides information about special configurations that control the key selection for VCAP IS2. The key selection selects which overall key type to use for a lookup and controls specific key fields and where the input is taken from.

The following table lists the registers associated with key selection for VCAP IS2.

Table 4-143. Key Selection of VCAP IS2

Register	Description	Replication
ANA_ACL::VCAP_S2_CFG	Enabling of VCAP IS2 lookups.	Per port per lookup
ANA_ACL:KEY_SEL:VCAP_S2_KEY_SEL	Configuration of the key selection and key generation for the VCAP IS2	134 per lookup
ANA_L3:VMID:VMID_MISC.IRLEG_S2_KEY_SEL_IDX	Key selection index for routed frames associated with the ingress router leg. Index is offset with 70 into ANA_ACL:KEY_SEL:VCAP_S2_KEY_SEL.	Per VMID
ANA_L3:VMID:VMID_MISC.IRLEG_S2_KEY_SEL_IDX	Key selection index for routed frames associated with the egress router leg. Index is offset with 70 into ANA_ACL:KEY_SEL:VCAP_S2_KEY_SEL.	Per VMID
VCAP CLM action S2_KEY_SEL	Key selection index for VCAP CLM initiated rules. Index is offset with 70 into ANA_ACL:KEY_SEL:VCAP_S2_KEY_SEL.	Per VCAP CLM action

Each of the four lookups in VCAP IS2 must be enabled before use (VCAP_S2_CFG.SEC_ENA) for a key to be generated and a match to be used. If a lookup is disabled on a port, frames received by the port are not matched against rules in VCAP IS2 for that lookup.

The key selection for VCAP IS2 is governed by one of four sources:

- The ingress port (logical port). Using the ingress port is the default behavior and applies to all frames. This is intended for implementing port ACLs (PACL) or VLAN ACLs (VACL).
- The ingress router leg. This applies only to routed frames. This is intended for implementing ingress router ACLs (I-RACL) using the ingress router interface (IRLEG) as port information.
- The egress router leg. This applies only to routed frames.This is intended for implementing egress router ACLs (E-RACL) using the egress router interface (ERLEG) as port information.
- VCAP CLM actions. This applies only to frames matching a VCAP CLM entry with action S2_KEY_SEL_ENA set. This is intended for exception handling and special user-defined ACLs.

For each lookup, the key selection for each of the four sources is retrieved and evaluated in a prioritized order.

Use the key selection configuration indexed by VCAP CLM if applicable and the selected configuration has VCAP_S2_KEY_SEL.KEY_SEL_ENA set.

Use the key selection configuration indexed by the ERLEG if the frame is routed and the selected configuration has VCAP_S2_KEY_SEL.KEY_SEL_ENA set.

Use the key selection configuration indexed by the IRLEG if the frame is routed and the selected configuration has VCAP_S2_KEY_SEL.KEY_SEL_ENA set.

Use the key selection configuration indexed by the ingress port.

The resulting key selection configuration controls the key selection for the lookup in VCAP IS2. For some frame types (OAM Y.1731, SNAP, LLC, EYTYPE) the key selection is given as only the MAC_ETYPE key can be used. For others frame types (ARP and IP) multiple keys can be selected from. The following table lists the applicable key types available for each frame type listed in Table 4-140.

Table 4-144. VCAP IS2 Key Selection

Frame Type	Applicable VCAP IS2 keys
IPv6 multicast frames	IP6_VID IP_7TUPLE MAC_ETYPE
IPv6 unicast frames	IP_7TUPLE MAC_ETYPE
IPv4 multicast frames	IP4_VID IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
IPv4 unicast frames	IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
(R)ARP frames	ARP MAC_ETYPE
OAM Y.1731 frames	MAC_ETYPE
SNAP frames	MAC_ETYPE
LLC frames	MAC_ETYPE
ETYPE frames	MAC_ETYPE

Note: The key selection is overruled if VCAP CLM instructs the use of a custom key, where data from the frame is extracted at a VCAP CLM-selected point. For more information, see 4.20.1.3 VCAP IS2 Custom Keys.

In addition to the selected key type, specific key fields are controlled by the key selection configuration and its source.

Key selection configurations used by ingress or egress router legs must set ANA_ACL:KEY_SEL:VCAP_S2_KEY_SEL.IGR_PORT_MASK_SEL in order to use the router leg as port interface instead of the physical port normally used in VCAP IS2 keys. The following table lists key fields that are encoded differently depending on the source.

Table 4-145. Source-Dependent VCAP IS2 Key Fields

Key Field	Key Selection Source		
	Ingress port / VCAP CLM	IRLEG	ERLEG
IGR_PORT_MASK_L3	0	1	1
IGR_PORT_MASK	Physical port used	IRLEG used	ERLEG used
L3_DST	0	0	1
XVID	Ingress VID, or Egress VID when frame is routed and ANA_ACL::VCAP_S2_MISC_CTRL2.EGR_KEY_ENA is set	Ingress VID	Egress VID
ISDX	ISDX, or RSDX when > 0 and ANA_ACL::VCAP_S2_MISC_CTRL2.EGR_KEY_ENA is set	ISDX	RSDX when > 0, otherwise ISDX

For information about VCAP IS2 keys and actions, see 4.19 VCAP IS2 Keys and Actions.

4.20.1.3 VCAP IS2 Custom Keys

VCAP IS2 supports two custom keys, CUSTOM_1 and CUSTOM_2. Custom keys extract raw data from the frame at one selectable location. The custom keys enable matching against protocol fields not supported by other VCAP IS2 keys. The use of the custom keys must be enabled through a VCAP CLM FULL action using CUSTOM_ACE_ENA and CUSTOM_ACE_OFFSET.

CUSTOM_1 extracts 60 consecutive bytes of data from the frame and CUSTOM_2 extracts 24 consecutive bytes of data from the frame. The data is extracted at one of four different positions in the frame. For frames of frame_type ETH (IFH.TYPE_AFTER_POP), the positions are defined as:

- Link layer. Corresponds to first byte in DMAC.
- Link layer payload. Corresponds to first byte in L3, after VLAN tags and EtherType.
- Link layer payload plus 20 bytes. Corresponds to first byte in L4 for IPv4 frame.
- Link layer payload plus 40 bytes. Corresponds to first byte in L4 for IPv6 frame.

For frames of frame_type CW, the first link layer position is not available. The positions are defined as:

- Link layer payload. Corresponds to first byte in control word for non-IP frames and first byte in IP header for IP frames.
- Link layer payload plus 20 bytes. Corresponds to first byte in L4 for IPv4 frame.
- Link layer payload plus 40 bytes. Corresponds to first byte in L4 for IPv6 frame.

4.20.1.4 VCAP IS2 Range Checkers

The following table lists the registers associated with configuring VCAP IS2 range checkers.

Table 4-146. VCAP IS2 Range Checker Configuration

Register	Description	Replication
ANA_ACL::VCAP_S2_RNG_CTRL	Configuration of the range checker types	Per range checker
ANA_ACL::VCAP_S2_RNG_VALUE_CFG	Configuration of range start and end points	Per range checker
ANA_ACL::VCAP_S2_RNG_OFFSET_CFG	Offset into frame when using a range checker based on selected value from frame	None

Sixteen global VCAP IS2 range checkers are supported by keys MAC_ETYPE, ARP, IP4_TCP_UDP, IP4_OTHER, and IP_7TUPLE. All frames using these keys are compared against the range checkers and a 1-bit range “match/no match” flag is returned for each range checker. The combined 16 match/no match flags are used in the L4_RNG key field. So, it is possible to include for example ranges of DSCP values and/or ranges of TCP source port numbers in the ACLs.

Note, VCAP CLM also supports range checkers but they are independent of the VCAP IS2 range checkers.

The range key is generated for each frame based on extracted frame data and the configuration in ANA_ACL::VCAP_S2_RNG_CTRL. Each of the 16 range checkers can be configured to one of the following range types.

- TCP/UDP destination port range:

  Input to the range is the frame’s TCP/UDP destination port number.
  Range is only applicable to TCP/UDP frames.
- TCP/UDP source port range:

  Input to the range is the frame’s TCP/UDP source port number.
  Range is only applicable to TCP/UDP frames.
- TCP/UDP source and destination ports range:
  Range is matched if either source or destination port is within range.
  Input to the range are the frame’s TCP/UDP source and destination port numbers.
  Range is only applicable to TCP/UDP frames.
- VID range:
  Input to the range is the ingress classified VID.
  Range is applicable to all frames.
- DSCP range:

  Input to the range is the classified DSCP value.
  Range is applicable to IPv4 and IPv6 frames.
- Selected frame field range:

  Input to the range is a selected 16-bit wide field from the frame. The field is selected using the offset value configured in ANA_ACL::VCAP_S2_RNG_OFFSET_CFG. The offset starts at the Type/Len field in the frame after up to 3 VLAN tags have been skipped. Note that only one selected value can be configured for all eight range checkers.
  Range is applicable to all frames.

Range start points and range end points are configured in ANA_ACL::VCAP_S2_RNG_VALUE_CFG with both the start point and the end point being included in the range. A range matches if the input value to the range (for instance the frame’s TCP/UDP destination port) is within the range defined by the start point and the end point.

4.20.1.5 Miscellaneous VCAP IS2 Key Configurations

The following table lists the registers associated with configuration that control the specific fields in the VCAP IS2 keys.

Table 4-147. Other VCAP IS2 Key Configurations

Register	Description	Replication
ANA_ACL:VCAP_S2:VCAP_S2_MISC_CTRL	Configuration of IGR_PORT_MASK_SEL and IGR_PORT_MASK.	None

All VCAP IS2 keys except IP4_VID and IP6_VID contain the field IGR_PORT_MASK_SEL and IGR_PORT_MASK. For frames received on a front port, IGR_PORT_MASK_SEL is set to 0 and the bit corresponding to the frame’s physical ingress port number is set in IGR_PORT_MASK. The following lists some settings for frames received on other interfaces and how this can be configured through register VCAP_S2_MISC_CTRL.

- Loopback frames:

By default, IGR_PORT_MASK_SEL = 1 and IGR_PORT_MASK has one bit set corresponding to the physical port which the frame was looped on. Optionally use IGR_PORT_MASK_SEL = 3.

- Masqueraded frames:
  By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has one bit set corresponding to the masquerade port. Optionally, use IGR_PORT_MASK_SEL = 2.
- VStaX frames:

  By default, frames received with a VStaX header on a front port use IGR_PORT_MASK_SEL = 0. Optionally, use IGR_PORT_MASK_SEL=3.
- Virtual device frames:

  By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has one bit set corresponding to the original physical ingress port. Optionally use IGR_PORT_MASK_SEL = 3.
- CPU injected frames:
  By default, IGR_PORT_MASK_SEL = 0 and IGR_PORT_MASK has none bits set. Optionally use IGR_PORT_MASK_SEL = 3.

For more information about the contents of IGR_PORT_MASK_SEL and IGR_PORT_MASK, seeTable 4-138.

4.20.1.6 VCAP IS2 Statistics and Diagnostics

The following table lists the registers associated with VCAP IS2 statistics and diagnostics.

Table 4-148. VCAP IS2 Statistics and Diagnostics

Register	Description	Replication
ANA_ACL::SEC_LOOKUP_STICKY	Sticky bits for each key type used in VCAP IS2.	Per lookup
ANA_ACL::CNT_A	VCAP IS2 match counters for lookup first and second. Indexed with VCAP IS2 action CNT_ID.	4,096
ANA_ACL::CNT_B	VCAP IS2 match counters for third and fourth lookup. Indexed with VCAP IS2 action CNT_ID.	4,096

Each key type use in a lookup in VCAP IS2 triggers a sticky bit being set in ANA_ACL::SEC_LOOKUP_STICKY. A sticky bit is cleared by writing 1 to it.

Each action returned by VCAP IS2 triggers a counter to be incremented. This applies to both actions from matches and default actions. The index into the counters is provided by VCAP IS2 action CNT_ID. The CNT_ID is fully flexible meaning multiple actions can point to the same counter. However, actions returned by the first or second lookup use counters in ANA_ACL:CNT_A while actions returned by the third or fourth lookup use counters in ANA_ACL:CNT_B. All counters are writable so they can be preset to any value. A counter is cleared by writing 0.

4.20.1.7 Processing of VCAP IS2 Actions

VCAP IS2 returns an action for each enabled VCAP IS2 lookup. If the lookup matches an entry in VCAP IS2 then the associated action is returned. A default action is returned when no entries are matched. The first VCAP IS2 lookup returns a default action per physical ingress port while the following VCAP IS2 lookup return a common default action. There is no difference between an action from a match and a default action.

VCAP IS2 contains 73 default actions that are allocated the following ways.

- 0-64: Per ingress port default actions for first lookup.
- 65, 66: Per CPU port (CPU0 and CPU1) default actions for first lookup.
- 67, 68, 69: Per virtual device (VD0, VD1, and VD2) default actions for first lookup.
- 70: Common default action for second lookup.
- 71: Common default action for third lookup.
- 72: Common default action for fourth lookup.

Each of the four VCAP IS2 actions (from the four lookups) can be processed either at the pipeline point ANA_OU_SW or at pipeline point ANA_IN_SW. This is controlled by VCAP IS2 action IS_INNER_ACL.

The first and the third lookups are done in parallel in separate TCAMs. The same applies to the second and the fourth lookups. Consequently, the actions from the first and the third lookups are processed before the actions from the second and the fourth lookups. The order of processing is given as:

1. Process first action if the first's action IS_INNER_ACL is 0.
2. Process third action if the third's action IS_INNER_ACL is 0.
3. Process second action if the second's action IS_INNER_ACL is 0.
4. Process fourth action if the fourth's action IS_INNER_ACL is 0.
5. Process first action if the first's action IS_INNER_ACL is 1.
6. Process third action if the third's action IS_INNER_ACL is 1.
7. Process second action if the second's action IS_INNER_ACL is 1.
8. Process fourth action if the fourth's action IS_INNER_ACL is 1.

This implies that any action can be processed in ANA_OU_SW pipeline point and in ANA_IN_SW pipeline point. All actions can also be placed at the same pipeline point.

The reason for the third action being processed before the second is due to the first and third lookups being done in parallel in separate TCAMs.

Processing of each action obeys the frame's current pipeline information (pipeline point and pipeline action). For instance if the frame has been redirected at pipeline point ANA_CLM then neither of the VCAP IS2 actions are applied. However, VCAP IS2 action IGNORE_PIPELINE_CTRL can overrule this.

Furthermore, frame processing between pipeline points ANA_OU_SW and ANA_IN_SW such as OAM filtering, protection switching, or routing can influence the processing of the VCAP IS2 actions so that only an action belonging to pipeline point ANA_OU_SW is processed.

The following table lists how the four VCAP IS2 action are combined when being processed. For most cases, the processing is sequential, meaning that if more than one action has settings for the same (for instance MIRROR_PROBE), the last action processed takes precedence. Others are sticky meaning they cannot be undone by a subsequent action if already set by an earlier action.

Table 4-149. Combining VCAP IS2 Actions

Action Name	Combining Actions
IS_INNER_ACL	Processed individually for each action.
PIPELINE_FORCE_ENA	Processed individually for each action. Subsequent processing uses the result from the previous processing which can prevent its processing if the pipeline information from previous processing disables the subsequent processing.
PIPELINE_PT	See PIPELINE_FORCE_ENA.
HIT_ME_ONCE	Sticky.
INTR_ENA	Sticky.
CPU_COPY_ENA	Sticky.
CPU_QU_NUM	Sticky (each action can generate a CPU copy and based on configuration in ANA_AC:PS_COMMON:CPU_CFG.ONE_CPU_COPY_ONLY_MASK, each copy can be sent to CPU).
CPU_DIS	Processed individually for each action. Subsequent processing may clear CPU copy from previous processing.
LRN_DIS	Sticky.
RT_DIS	Sticky.
POLICE_ENA	Sticky.

.....continued

Action Name	Combining Actions
POLICE_IDX	Subsequent actions take precedence when POLICE_ENA is set.
IGNORE_PIPELINE_CTRL	Processed individually for each action.
MASK_MODE	Processed individually for each action. Subsequent processing uses the result from the previous processing. (If the previous processing is sticky, the subsequent processing is not applied). MASK_MODE is sticky for the following values: 3: REPLACE_ALL 4: REDIR_PGID 6: VSTAX
PORT_MASK	See MASK_MODE.
RSDX_ENA	Sticky.
RSDX_VAL	Subsequent actions take precedence when RSDX_ENA is set.
MIRROR_PROBE	Subsequent actions take precedence when MIRROR_PROBE > 0.
REW_CMD	Subsequent actions take precedence same functions are triggered by additional actions. If different functions are used, all are applied.
TTL_UPDATE_ENA	Sticky.
SAM_SEQ_ENA	Sticky.
TCP_UDP_ENA	Sticky.
TCP_UDP_DPORT	Subsequent actions takes precedence when TCP_UDP_ENA is set.
TCP_UDP_SPORT	Subsequent actions takes precedence when TCP_UDP_ENA is set.
MATCH_ID	Processed individually for each action. Subsequent processing uses the result from the previous processing.
MATCH_ID_MASK	Processed individually for each action. Subsequent processing uses the result from the previous processing.
CNT_ID	Processed individually for each action. Four counters are updated, one for each action.
SWAP_MAC_ENA	Sticky.
ACL_RT_MODE	Subsequent actions take precedence when same function is triggered by additional actions. If different functions are used, all are applied.
ACL_MAC	Subsequent actions take precedence when same function is triggered by additional actions. If different functions are used, all are applied.
DMAC_OFFSET_ENA	Sticky
PTP_MASTER_SEL	Subsequent actions take precedence when used by REW_CMD.
LOG_MSG_INTERVAL	Subsequent actions take precedence when used by REW_CMD.
SIP_IDX	Subsequent actions take precedence when SIP replace is triggered ACL_RT_MODE.

.....continued	
Action Name	**Combining Actions**
RLEG_STAT_IDX	Processed individually for each action.
IGR_ACL_ENA	Processed individually for each action.
EGR_ACL_ENA	Processed individually for each action.

4.20.1.8 Routing Statistics

The following table lists the registers associated with routing statistics affected by VCAP IS2.

Table 4-150. VCAP IS2 Routing Statistics

Register	Description	Replication
ANA_ACL::VCAP_S2_RLEG_STAT	Router leg statistics masks.	4

If a routed frame is discarded by a VCAP IS2 rule by setting RT_DIS, it is configurable how the frame is counted in terms of ingress and egress router leg statistics.

VCAP IS2 actions IGR_KEY_ENA and EGR_KEY_ENA select whether the frame is discarded during the ingress router processing or during the egress router processing. VCAP IS2 action RLEG_STAT_IDX selects an ingress and an egress router leg statistics mask (ANA_ACL::VCAP_S2_RLEG_STAT) that program which router leg statistics to clear by the discard. For more details about the router leg statistics, see 4.18.3 Statistics.

Before an ingress discard is applied, it is verified that ANA_L3 has received the IP frame by checking that either ingress router leg events ivmid_ip_uc_received or ivmid_ip_mc_received is set. The following actions are then applied.

- Clear ingress router leg statistics events according to VCAP_S2_RLEG_STAT.IRLEG_STAT_MASK.
- Clear egress router leg statistics events according to VCAP_S2_RLEG_STAT.ERLEG_STAT_MASK.
- Set ivmid_ip_acl_discarded when also set in VCAP_S2_RLEG_STAT.IRLEG_STAT_MASK.

An egress discard is applied to any frame marked as routed by ANA_L3. The following actions are applied.

- Keep ingress router leg statistics events as signaled by ANA_L3 because the frame is discarded after the routing.
- Clear egress router leg statistics events according to VCAP_S2_RLEG_STAT.ERLEG_STAT_MASK.
- Set evmid_ip_acl_discarded when also set in VCAP_S2_RLEG_STAT.ERLEG_STAT_MASK.

4.20.2 Analyzer Access Control List Frame Rewriting

When rewriting in the ANA_ACL block, ingress mirroring with an exact mirror copy of the incoming frame is no longer possible, because any rewrites are also reflected in the mirror copy.

4.20.2.1 Address Swapping and Rewriting

VCAP IS2 actions can trigger frame rewrites in the ANA_ACL block with focus on MAC addresses, IP addresses, TCP/UDP ports as well as IPv4 TTL and IPv6 hop limit.

The following table lists the registers associated with address swapping and rewriting capabilities in ANA_ACL.

Table 4-151. ANA_ACL Address Swapping and Rewriting Registers

Register	Description	Replication
ANA_ACL::SWAP_SIP	IP table with 32 IPv4 addresses or 8 IPv6 addresses. Provides new source IP address when swapping IP addresses in frame with multicast destination IP address.	32
ANA_ACL::SWAP_IP_CTRL	Controls IPv4 TTL and IPv6 hop limit values. Controls number of bits to replace in DMAC.	None

VCAP IS2 action ACL_RT_MODE can enable the following rewrite modes.

- Replace DMAC: The frame's DMAC is replaced with the MAC address specified in VCAP IS2 action ACL_MAC. It is selectable to only replace part of the DMAC (for instance the OUI only). This is enabled with VCAP IS2 action DMAC_OFFSET_ENA, and the number of bits to replace is set in ANA_ACL::SWAP_IP_CTRL.DMAC_REPL_OFFSET_VAL.
- Replace SMAC: The frame's SMAC is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC.
- Swap MAC addresses and replace SMAC if MC: The frame's MAC addresses are swapped. If the new SMAC is a multicast MAC, it is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC.
- Swap MAC addresses if UC else replace SMAC if MC: The frame's MAC addresses are swapped if the frame's DMAC is unicast. Otherwise, the frame's SMAC is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC.
- Swap MAC and IP addresses and replace SIP and SMAC if MC: The frame's MAC addresses are swapped and the frame's IP addresses are swapped. If the new SMAC is a multicast MAC, it is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC. If the new SIP is a multicast, it is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.
- Swap MAC and IP addresses if UC else replace SIP and SMAC if MC: The frame's MAC addresses are swapped if the frame's DMAC is unicast. Otherwise, the frame's SMAC is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC. The frame's IP addresses are swapped if the frame's DIP is unicast. Otherwise, the frame's SIP is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.
- Swap IP addresses and replace SIP if MC: The frame's IP addresses are swapped. If the new SIP is a multicast, it is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.
- Replace SMAC, swap IP addresses, and replace SIP if MC: The frame's SMAC is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC. The frame's IP addresses are swapped. If the new SIP is a multicast, it is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.

The IP address table (ANA_ACL::SWAP_SIP) used by some of the above mentioned modes is indexed using VCAP IS2 action SIP_IDX. The IP address table can contain 32 IPv4 addresses or 8 IPv6 addresses.

When swapping IP addresses or replacing the source IP address, it is also possible to preset the IPv4 TTL or the IPv6 hop limit. This is enabled by VCAP IS2 action TTL_UPDATE_ENA. The new TTL value for IPv4 frames is configured in ANA_ACL::SWAP_IP_CTRL.IP_SWAP_IP4_TTL_VAL and the new hop limit value for IPv6 frames is configured in ANA_ACL::SWAP_IP_CTRL.IP_SWAP_IP4_TTL_VAL.

IPv4 and IPv6 TCP/UDP frames have the option to replace the source and destination port numbers with new values. This is enabled through VCAP IS2 action TCP_UDP_ENA and new port numbers are provided by VCAP IS2 actions TCP_UDP_DPORT and TCP_UDP_SPORT.

The IPv4 header checksum is updated whenever required by above mentioned frame rewrites. The UDP checksum is updated whenever required for IPv6 frames and cleared for IPv4 frames.

4.20.2.2 Routing Rewrites in ANA_ACL

The following table lists the registers associated with routing capabilities in ANA_ACL.

Table 4-152. ANA_ACL Routing Registers

Register	Description	Replication
ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_SEL	Enable routing related frame rewrites in ANA_ACL	None

If PTP and routing is to be supported concurrently, then some routing related frame rewrites must be done in ANA_ACL instead of in the rewriter. This is enabled in ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_SEL or by setting VCAP IS2 action ACL_RT_MODE. When enabled, the frame's DMAC is changed to the next-hop MAC address specified by ANA_L3 and the rewriter is informed not to rewrite the DMAC.

The frame's classified VID is set to the egress VID (ANA_L3:VMID:RLEG_CTRL.RLEG_EVID).

In addition, ANA_L3 must be setup to change the source MAC address with the address belonging to the egress router leg (ANA_L3::ROUTING_CFG.RT_SMAC_UPDATE_ENA). The IPv4 TTL or IPv6 hop limit is still decremented by the rewriter.

4.21 Analyzer Layer 2 Forwarding and Learning

The analyzer Layer 2 (ANA_L2) block performs the following tasks.

- Determining forwarding decisions
- Tracking network stations and their MAC addresses through learning and aging
- Tracking and setting limits for number station learned per service, per FID, or per port
- Scanning and updating the MAC table

The analyzer Layer 2 block makes a forwarding decision based on a destination lookup in the MAC table. The lookup is based on the received Destination MAC address together with either the classified VID for multicast traffic or the classified FID for unicast traffic. If an entry is found in the MAC table lookup, the associated entry address is used for selecting forwarding port or ports. A flood forwarding decision is made if no entry is found and forwarding to unknown destinations is permitted (ANA_L3:VLAN:VLAN_CFG.VLAN_SEC_FWD_ENA).

The analyzer Layer 2 block performs a source lookup in the MAC table to determine if the source station is known or unknown by looking at if there is an entry in the MAC table. If a MAC table entry exists, a port move detection is performed by looking at whether the frame was received at the expected interface specified. The source check can trigger the following associated actions.

- Disabling forwarding from unknown or moved stations
- Triggering automated learning
- Sending copy to CPU

If the source station is known, the entry AGE_FLAG is cleared to indicate that source associated with the entry is active. The AGE_FLAG is part of the aging functionality to remove inactive MAC table entries. For more information, see 4.21.7 Automated Aging (AUTOAGE).

The following subsections further describe the main analyzer Layer 2 blocks.

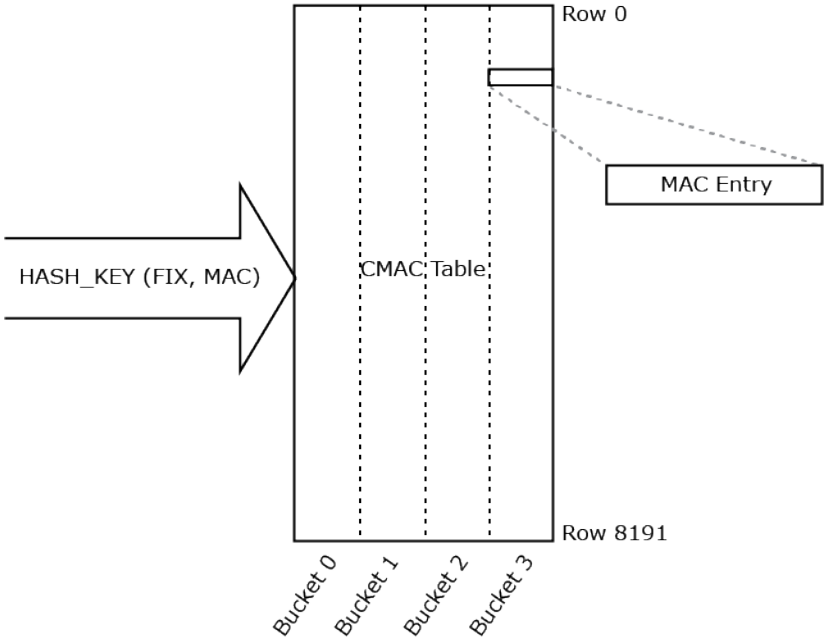
4.21.1 Analyzer MAC Table

The analyzer Layer 2 block contains a MAC table with 32,768 entries containing information about stations learned or known by the device. The table is organized as a hash table with four buckets on each row. Each row is indexed based on a hash value calculated based on the station's (FID, MAC) pair.

The filtering ID (FID) used is either the FID from the VLAN table if a unicast MAC is looked up or else it is the classified VID. It is possible to enforce use of FID for multicast (ANA_L3:COMMON:SERVICE_CFG).

MAC table organization is depicted in the following figure.

Figure 4-44. MAC Table Organization



4.21.1.1 MAC Entry Table

Each entry in the MAC table contains the fields listed in the following table.

Table 4-153. MAC Table Entry

Field	Bits	Description
VLD	1	Entry is valid.
MAC	48	The MAC address of the station. (Part of primary key).
FID	13	VLAN filtering identifier (FID) unicast entries. VLAN identifier (VID) for multicast. (Part of primary key).
LOCKED	1	Lock the entry as static. **Note:** Locking an entry will prevent hardware from changing anything but the AGE_FLAG in the entry.
MIRROR	1	Frames from or to this station are candidate for mirroring.
AGE_FLAG	2	The number of loopbackaging periods that have taken place since the last frame was received from this station. Incremented by hardware during a CPU SCAN and AUTOAGE SCAN for the entry configured AGE_INTERVAL.
AGE_INTERVAL	2	Used to select which age timer is associated with the entry.
NXT_LRN_ALL	1	Used to ensure consistent MAC tables in a stack.
CPU_COPY	1	Frames from or to this station are candidate for CPU copy. Used together with CPU_QU (to specify queue number).
CPU_QU	3	Used together with CPU_COPY.
SRC_KILL_FWD	1	Frames from this station will not be forwarded. This flag is not used for destination lookups.

.....continued		
Field	**Bits**	**Description**
VLAN_IGNORE	1	Used for ignoring the VLAN_PORT_MASK contribution from the VLAN table when forwarding frames to this station. Can optionally be used for source port ignore (ANA_L2::FWD_CFG.FILTER_MODE_SEL).
ADDR	12	Stored entry address used for forwarding and port move detection. The field is encoded according to ADDR_TYPE.
ADDR_TYPE	3	This field encodes how to interpret the ADDR field. Encoded as: ADDR_TYPE= UPSID_PN(0): Unicast entry address. Used to associate the entry with a unique {UPSID, UPSPN} port. ADDR(9:5) = UPSID ADDR(4:0) = UPSPN ADDR_TYPE=GCPU_UPS(1): CPU management entry address. Used to associate the entry with a unique global CPU port or internal port. ADDR(9:5) = UPSID ADDR(11) = 0: ADDR(3:0) = CPU port number. ADDR(11) = 1: ADDR(3:0) = 0xe: Internal port number. ADDR(3:0) = 0xf: Local lookup at destination unit (UPSID). ADDR_TYPE=GLAG(2): Unicast GLAG address. Used to associate the entry with a global aggregated port. ADDR = GLAGID. ADDR_TYPE=MC_IDX(3): Multicast entry address. Used to associate the entry with an entry in the PGID table. Specifies forwarding to the ports found in the PGID table entry indexed by mc_idx. ADDR = mc_idx.
ISDX_LIMIT_IDX	11	Configures the ISDX learn limit index associated with the entry.

4.21.1.2 CAM Row

Under some circumstances, it might be necessary to store more entries than possible on a MAC table row, in which case entries are continuously changed or automatically updated.

To reduce this issue, one additional MAC table row (named CAM row) exist with entries that can be used to extend any hash chain, and thereby reduce the probability for hash depletion where all the buckets on a MAC table row are used. These entries on the CAM row can be used as any other MAC table entries.

CAM row usage is controlled separately (by ANA_L2::LRN_CFG.AUTO_LRN_USE_MAC_CAM_ENA, ANA_L2::LRN_CFG.CPU_LRN_USE_MAC_CAM_ENA and ANA_L2::LRN_CFG.LRN_MOVE_CAM_ENTRY_BACK).

4.21.2 MAC Table Updates

Entries in the MAC table are added, deleted, or updated by the following methods.

- Automatic (hardware-based) learning of source MAC addresses; that is, inserting new (MAC, FID) pairs in the MAC table
- CPU access to MAC table (for example, CPU-based learning and table maintenance)

- Automated age scan.

4.21.3 CPU Access to MAC Table

This section describes CPU access to the MAC table. The following table lists the applicable registers.

Table 4-154. MAC Table Access Registers

Target::Register.Field	Description	Replication
LRN:: COMMON_ACCESS_CTRL	Controls CSR access to MAC table.	1
LRN::MAC_ACCESS_CFG_0	Configures MAC (most significant bits) and FID for MAC table entries.	1
LRN::MAC_ACCESS_CFG_1	Configures MAC address (Least significant bits) for MAC table entries and FID.	1
LRN::MAC_ACCESS_CFG_2	Configures the following MAC entry fields: ADDR ADDR_TYPE SRC_KILL_FWD NXT_LRN_ALL CPU_COPY VLAN_IGNORE AFE_FLAG AGE_INTERNAL MIRROR LOCKED VLD	1
LRN::MAC_ACCESS_CFG_3	Configures ISDX_LIMIT_IDX.	1
LRN::EVENT_STICKY	Sticky status for access performed.	1
LRN::SCAN_NEXT_CFG	MAC table scan filter controls.	1
LRN::SCAN_NEXT_CFG_1	MAC table port move handles when performing MAC table SCAN.	1
ANA_L2::SCAN_FID_CTRL	Controls use of additional FIDs when doing scan.	1
ANA_L2::SCAN_FID_CFG	Configures additional VID/FID filters during scan if enabled in ANA_L2::SCAN_FID_CTRL.	16
LRN::SCAN_LAST_ROW_CFG	Configures a last row applicable for scan.	1
LRN::SCAN_NEXT_CNT	MAC table status when performing MAC table SCAN.	1
LRN::LATEST_POS_STATUS	Holds the latest CPU accessed MAC table location after a CPU_ACCESS_CMD has finished.	1

CPU access to the MAC table is performed by an indirect command-based interface where a number of fields are configured (in LRN::MAC_TABLE_ACCESS_CFG_*) followed by specifying the command (in LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD). The access is initiated by a shot bit that is cleared upon completion of the command (LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT). It is possible to trigger CPU interrupt upon completion (ANA_L2:COMMON:INTR.LRN_ACCESS_COMPLETE_INTR).

The following table lists the types of access possible.

Table 4-155. MAC Table Access Commands

Command	Purpose	Use
Learn	Insert/learn new entry in MAC table. Position given by HASH(MAC, FID).	Configure MAC and FID of the new entry in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Configure entry fields in LRN::MAC_ACCESS_CFG_2. Status of operation returned in: LRN::EVENT_STICKY.CPU_LRN_REFRESH_STICKY, LRN::EVENT_STICKY.CPU_LRN_INSERT_STICKY, LRN::EVENT_STICKY.CPU_LRN_REPLACE_STICKY, LRN::EVENT_STICKY.CPU_LRN_REPLACE_FAILED_STICKY, LRN::EVENT_STICKY.CPU_LRN_FAILED_STICKY, and LRN::EVENT_STICKY.LRN_MOVE_STICKY
Unlearn/Forget	Delete/unlearn entry in MAC table given by (MAC, FID). Position given by HASH(MAC, FID).	Configure MAC and FID of the entry to be unlearned in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Status of operation returned in: LRN::EVENT_STICKY.CPU_UNLEARN_STICKY and LRN::EVENT_STICKY.CPU_UNLEARN_FAILED_STICKY.
Lookup	Lookup entry in MAC table given by (MAC, FID). Position given by HASH(MAC, FID)	Configure MAC and FID of the entry to be looked up in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Status of lookup operation returned in: LRN::EVENT_STICKY.CPU_LOOKUP_STICKY and LRN::EVENT_STICKY.CPU_LOOKUP_FAILED_STICKY Entry fields are returned in LRN::MAC_ACCESS_CFG_2 if lookup succeeded.
Read direct	Read entry in MAC table indexed by (row, column)	Configure the index of the entry to read in LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW, LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_COL and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE. Status of read operation returned in LRN::EVENT_STICKY.CPU_READ_DIRECT_STICKY Entry fields are returned in LRN::MAC_ACCESS_CFG_0, LRN::MAC_ACCESS_CFG_1 and LRN::MAC_ACCESS_CFG_2 if read succeeded.

.....continued

Command	Purpose	Use
Write direct	Write entry in MAC table indexed by (row, column)	Configure the index of the entry to write in LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW, LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_COL and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE. Configure entry fields in LRN::MAC_ACCESS_CFG_0, LRN::MAC_ACCESS_CFG_1 and LRN::MAC_ACCESS_CFG_2 Status of write operation returned in LRN::EVENT_STICKY.CPU_WRITE_DIRECT_STICKY
Scan	Find next/Find all Searches through the MAC table and either stops at the first row with entries matching the specified filter conditions or perform the specified actions on all entries matching the specified filter conditions.	Configure the starting row for the scan in: LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW, LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_COL and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE. Configure an ending row for the scan (if searching through the complete table is not required): LRN::SCAN_LAST_ROW_CFG.SCAN_LAST_ROW For information about search filters and corresponding actions, see 4.21.4 SCAN Command.
Find smallest	Get the smallest entry in the MAC table numerically larger than the specified (FID, MAC). FID, and MAC are evaluated as a 61 bit number with the FID being most significant.	Configure MAC and FID of the starting point for the search in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Configure search filters to be used during find smallest: Use LRN::SCAN_NEXT_CFG.SCAN_NEXT_IGNORE_LOCKED_ENA to only search among entries with LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_LOCKED cleared. Use LRN::SCAN_NEXT_CFG.FID_FILTER_ENA to only search among entries with the VID/FID specified in LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID. Use LRN::SCAN_NEXT_CFG.ADDR_FILTER_ENA to only search among entries with the address specified in LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR and LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE
Clear all	Initialize the table	Clears the entire table.

4.21.3.1 Adding a Unicast Entry

Use the following steps to add a unicast entry to the MAC table using the CPU interface.

1. Configure the entry MAC address: LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_MAC_MSB <16 MS MAC bits > LRN::MAC_ACCESS_CFG_1.MAC_ENTRY_MAC_LSB < 32 LS MAC bit >
2. Configure the VLAN FID: LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID <fid value>

3. Configure the UPSID/UPSPN value:
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE 0 (=UPSID)
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR <UPSPN/port value >
4. Make the entry valid: LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_VLD 1
5. Perform access:
   LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD 0 (=LEARN)
   LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT 1
6. Verify that access succeeded: LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT == 0?
   LRN::EVENT_STICKY.CPU_LRN_INSERT_STICKY==1?

4.21.3.2 Adding Multicast Entry with Destination Set in PGID Table

Use the following steps to add Layer 2 multicast entries to the MAC table using the CPU interface.

1. Configure the entry MAC address: LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_MAC_MSB <16 MSB MAC bits > LRN::MAC_ACCESS_CFG_1.MAC_ENTRY_MAC_LSB < 32 LSB MAC bits >
2. Configure the VLAN ID:
   LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID <vid value>
3. Configure the PGID pointer:
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE 3 (=MC_IDX)
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR <pgid value >
4. Lock the entry (not subject to aging and automatic removal):
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_LOCKED 1
5. Make the entry valid:
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_VLD 1
6. Perform access:
   LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD 0 (=LEARN)
   LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT 1
7. Verify that access succeeded: LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT == 0?
   LRN::EVENT_STICKY.CPU_LRN_INSERT_STICKY==1?

4.21.4 SCAN Command

The SCAN command easily performs table management, such as port flushing, CPU-based aging, and port moves.

Scan can be configured with a variety of filters and modifiers that allows the CPU to easily identify and optionally modify entries matching the configured filter conditions.

The scan operates by running through all entries and whenever entries are found with matching conditions, the scan either stops with a status of the matching location (for example, a “find next” operation) or performs the configured modification actions for all entries (for example, “scan all” operation).

When the scan runs as “find next”, the scan commands are performed for finding the next matching entry. In this case it is possible to specify a starting row for a scan (LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE).

Whenever the scan stops with a matching entry, the CPU can process the entry and then sets the starting row to the row following the matching row and, issue another scan command. This can then be repeated until all entries have been scanned.

SCAN is started by setting LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD 5 (=SCAN) and setting the shot bit LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT

SCAN is completed when the shot bit is cleared in LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT.

The following table lists the supported SCAN filters.

Table 4-156. Scan Filters

Filter Type	Target::Register.Field	Description
FID filter	LRN::SCAN_NEXT_CFG.FID_FILTER_ENA LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID ANA_L2::SCAN_FID_CTRL ANA_L2::SCAN_FID_CFG	Finds entries with a specific VID/FID.
Port or Address filter	LRN::SCAN_NEXT_CFG.ADDR_FILTER_ENA, LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR, LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE LRN::SCAN_NEXT_CFG_1.SCAN_ENTRY_ADDR_MASK	Finds entries with a specific ADDR_TYPE and ADDR. Note that this can be combined with ADDR wildcards; for example, to find all entries related with a given ADDR_TYPE, and so on.
Automatic age scan port filter	LRN::SCAN_NEXT_CFG.SCAN_USE_PORT_FILTER_ENA ANA_L2::FILTER_OTHER_CTRL ANA_L2::FILTER_LOCAL_CTRL	Configures a port filter used for defining which ports are applicable for automatic aging.
Non-locked only filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_IGNORE_LOCKED_ENA	Finds non locked entries.
Aged only filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_AGED_ONLY_ENA	Finds only aged entries, that is, AGE_FLAG equal to maximum (configured in ANA_L2::LRN_CFG.AGE_SIZE).
AGE_INTERVAL filter	LRN::SCAN_NEXT_CFG.SCAN_AGE_INTERVAL_MASK	Finds entries with a specific AGE_INTERVAL.
AGE_FLAG filter	LRN::SCAN_NEXT_CFG.SCAN_AGE_FILTER_SEL MAC_ACCESS_CFG_2.MAC_ENTRY_AGE_FLAG	Finds only entries with a specific value.
NXT_LRN_ALL filter	LRN::SCAN_NEXT_CFG.NXT_LRN_ALL_FILTER_ENA MAC_ACCESS_CFG_2.MAC_ENTRY_NXT_LRN_ALL	Finds only entries with a specific value.
Ignore locked filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_IGNORE_LOCKED_ENA	Finds all or only non locked entries.
Aged only filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_AGED_ONLY_ENA	Finds all or only aged entries (AGE_FLAG != 0).

The following table lists the available SCAN related actions.

Table 4-157. Scan Modification Actions

Action Type	Target::Register.Field	Description
Scan next until found	LRN::SCAN_NEXT_CFG.SCAN_NEXT_UNTIL_FOUND_ENA	Controls if scan stops when finding an entry or scans through all entries.

.....continued		
Action Type	**Target::Register.Field**	**Description**
Update AGE_FLAG	LRN::SCAN_NEXT_CFG.SCAN_AGE_FLAG_UPDATE_SEL	Updates AGE_FLAG for found entries. See LRN::SCAN_NEXT_CFG.
Update NXT_LRN_ALL	LRN::SCAN_NEXT_CFG.SCAN_NXT_LRN_ALL_UPDATE_SEL	Update sNXT_LRN_ALL for found entries. See LRN::SCAN_NEXT_CFG.
Change address (MOVE)	LRN::SCAN_NEXT_CFG.SCAN_NEXT_MOVE_FOUND_ENA LRN::SCAN_NEXT_CFG_1.PORT_MOVE_NEW_ADDR LRN::SCAN_NEXT_CFG_1.SCAN_ENTRY_ADDR_MASK	Updates address for found entries.
Remove	LRN::SCAN_NEXT_CFG.SCAN_NEXT_REMOVE_FOUND_ENA	Removes found entries.
CPU age scan	LRN::SCAN_NEXT_CFG.SCAN_NEXT_INC_AGE_BITS_ENA	Performs CPU based age scan. Aged entries. For example, entries with AGE_FLAG equal to maximum (configured in ANA_L2::LRN_CFG.AGE_SIZE) are removed and Non aged entries, such as entries with AGE_FLAG less than maximum, gets the AGE_FLAG incremented. 4.21.6.1 Automatic Learning.

The following SCAN status is available.

- Match scan result for the last matching row is reported (LRN::LATEST_POS_STATUS.SCAN_NEXT_STATUS).
- Number of entries found during the scan found (LRN::SCAN_NEXT_CNT.SCAN_NEXT_CNT).
- Scan remove is reported via the sticky event (LRN::EVENT_STICKY.SCAN_REMOVED_STICKY)
- Scan match found is reported via the sticky event (LRN::EVENT_STICKY.ROW_WITH_SCAN_ENTRY_STICKY).

4.21.4.1 Initiating a Port Move for a Specific FID

The following example initiates a port move for specific FID.

- Move found entries:
  Set LRN::SCAN_NEXT_CFG.SCAN_NEXT_MOVE_FOUND_ENA = 1.
- Specify the FID filter:
  Set LRN::SCAN_NEXT_CFG.FID_FILTER_ENA = 1.

  Set LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID = <FID value>.
- Specify port filter:

  Set LRN::SCAN_NEXT_CFG.ADDR_FILTER_ENA = 1.

  Set LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE = 0 to specify address type UPSID_PN.

  Set LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR = <UPSID, UPSPN> to specify UPSID and UPSPN.
- Specify the new address:
  Set LRN::SCAN_NEXT_CFG_1.PORT_MOVE_NEW_ADDR = <new UPSID, new UPSPN>

  Set LRN::SCAN_NEXT_CFG_1.SCAN_ENTRY_ADDR_MASK = all ones.
- Scan through all rows by starting at row 0:
  Set LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW = 0.

  Set LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE = 0.
- Issue SCAN command:

Set LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD = 5.

Set LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT = 1.

- Wait until LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT is cleared.

4.21.5 Forwarding Lookups

This section describes forwarding handling. The following table lists the applicable registers.

Table 4-158. Forwarding Registers

Target::Register.Field	Description	Replication
ANA_L2:COMMON:FWD_CFG	Configures common forwarding options	1

4.21.5.1 DMAC-Based Forwarding

The analyzer performs destination lookup in the MAC table to determine if the destination MAC address is known or unknown.

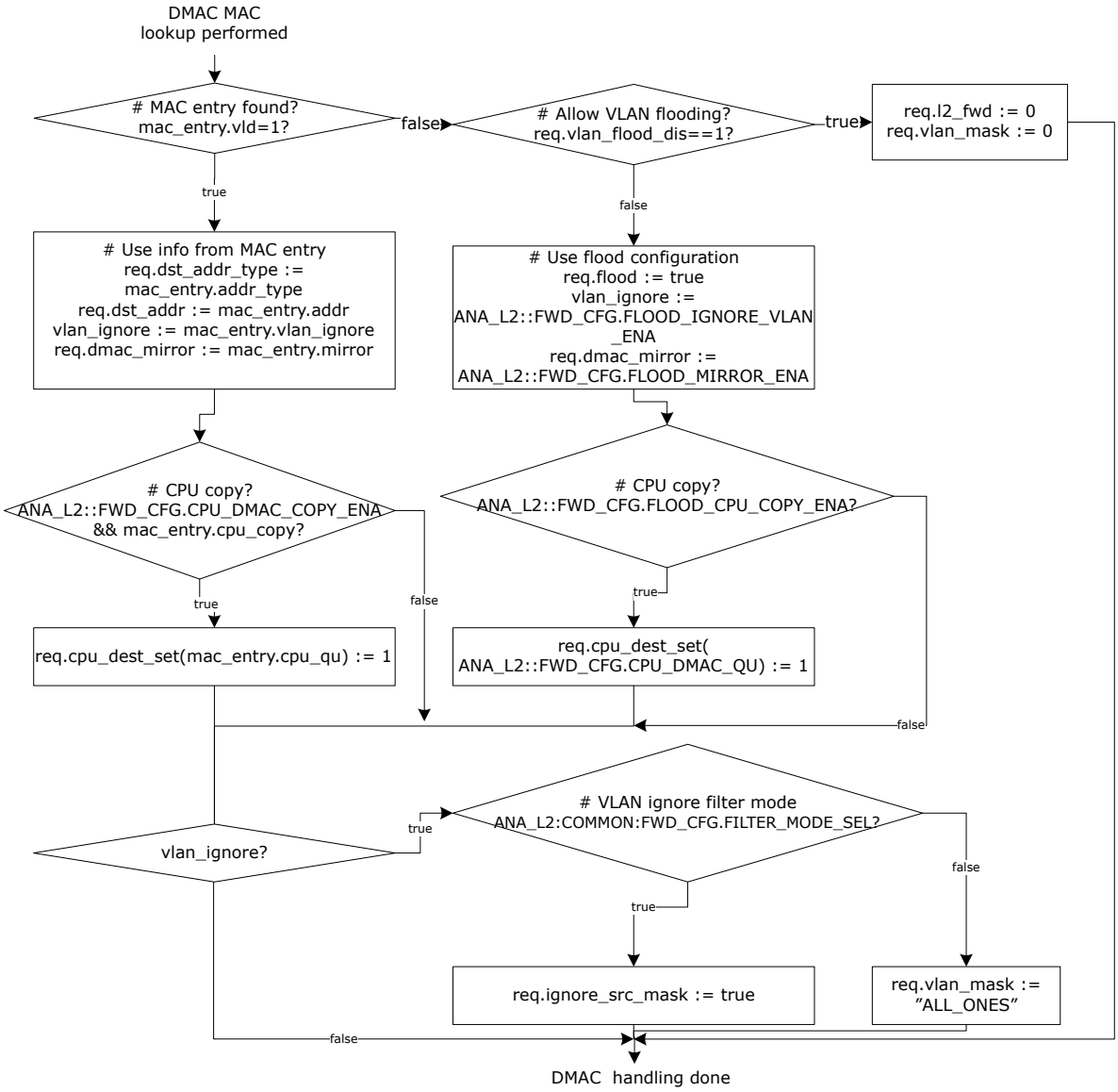
- If the destination is unknown, a flood forwarding decision is made if forwarding from unknown sources is allowed (ANA_L3:VLAN:VLAN_CFG.VLAN_SEC_FWD_ENA). The associated flood masks are located in the PGID table.
- If the destination address is known, the associated ADDR_TYPE and ADDR stored in the destination entry is used to find the destination port or ports in the PGID table.

Flooded traffic can be made available for mirroring (ANA_L2:COMMON:FWD_CFG.FLOOD_MIRROR_ENA) and CPU copying (ANA_L2:COMMON:FWD_CFG.FLOOD_CPU_COPY_ENA ANA_L2:COMMON:FWD_CFG.CPU_DMAC_QU).

Traffic with known DMAC and VLAN_IGNORE set LRN:COMMON:MAC_ACCESS_CFG_2.MAC_ENTRY_VLAN_IGNORE for CPU learned frames or LRN:COMMON:AUTO_LRN_CFG.AUTO_LRN_IGNORE_VLAN for auto learned entries) and optionally also flooded traffic (when ANA_L2:COMMON:FWD_CFG.FLOOD_IGNORE_VLAN_ENA set) can be configured to ignore VLAN port mask or source port mask (ANA_L2:COMMON:FWD_CFG.FILTER_MODE_SEL).

The DMAC lookup of the received {DMAC, EFID} in the MAC table is shown in the following figure.

Figure 4-45. DMAC Lookup



4.21.6 Source Check and Automated Learning

This section describes hardware (or automated) learning related to source checking. The following table lists the applicable registers.

Table 4-159. Hardware-Based Learning

Target::Register.Field	Description	Replication
ANA_L2::AUTO_LRN_CFG	Configures automatic learning per port	1
ANA_L2::LRN_SECUR_CFG	Configures secure forwarding per port	1

.....continued

Target::Register.Field	Description	Replication
ANA_L2::LRN_SECUR_LOCKED_CFG	Configures secure forwarding for static locked entries per port.	1
ANA_L2::LRN_COPY_CFG	Configures CPU copy of learn frames per port.	1
ANA_L2::MOVELOG_STICKY	Identifies ports with moved stations.	1
ANA_L2::LRN_CFG	Configures common learning properties.	1
ANA_L3:VLAN:VLAN_CFG.VLAN_FID	Configures filtering identifier (FID) to be used for learning.	Per VLAN
LRN::AUTO_LRN_CFG	Configures automatic learning options.	1
LRN::EVENT_STICKY	Signal various sticky learning events.	1
ANA_L2:PORT_LIMIT:PORT_LIMIT_ CTRL	Controls automatic learn limits per logical port and GLAG.	per port and GLAG (97 instances)
ANA_L2:PORT_LIMIT:PORT_LIMIT_ STATUS	Status for port limits.	per port and GLAG (97 instances)
ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL	Controls automatic learn limits per FID.	per FID (5,120 instances)
ANA_L2:LRN_LIMIT:FID_LIMIT_STATUS	Status for FID limits.	per FID (5,120 instances)
ANA_L2:ISDX_LIMIT:ISDX_LIMIT_CTRL	Controls automatic learn limits per ISDX limiter.	per ISDX limiter (1,536 instances)
ANA_L2:ISDX_LIMIT:ISDX_LIMIT_ STATUS	Status for ISDX limits.	per ISDX limiter (1,536 instances)

The device learns addresses from each received frame's SMAC field, so that subsequently forwarded frames whose destination addresses have been learned can be used to restrict transmission to the port or ports necessary to reach their destination.

The analyzer performs source lookup of the received {SMAC, IFID} in the MAC table to determine if the source station is known or unknown. For more information, see Figure 4-46. If the source station is known, a port move detection is performed by checking that the frame was received at the expected port as specified by the ADDR_TYPE and ADDR stored in the source entry. The expected port can be a logical port (represented as UPSID, UPSPN) or a global aggregated port (represented as GLAGID). Failing source checks can trigger a number of associated actions:

- Secure forwarding: Disable forwarding from unknown or moved stations. This action can be triggered by two configurations:
  VLAN-based Secure forwarding controlled per VLAN (ANA_L3:VLAN:VLAN_CFG.VLAN_SEC_FWD_ENA).

  Port-based secure forwarding controlled per port (ANA_L2::LRN_SECUR_CFG).
- Secure forwarding from locked entries: Disable forwarding from known but moved stations with locked MAC table entry (ANA_L2:COMMON:LRN_SECUR_LOCKED_CFG).
- Automatic learning enabled per port (ANA_L2::AUTO_LRN_CFG): Automatic learning of traffic received from an unknown source station will cause the MAC table to be updated with a new entry. Traffic received from a known source station with changed port will cause the known MAC table entry ADDR_TYPE and ADDR to be updated.
- CPU copy enabled per port (ANA_L2::LRN_COPY_CFG and ANA_L2::LRN_CFG.CPU_LRN_QU): A learn copy is sent to CPU.

These actions are controlled independently. In other words, it is possible to send a CPU copy as well as disable normal forwarding.

Figure 4-46. Source Check

The source lookup is also used to clear an Age flag (used as activity indication) in the MAC table to allow inactive MAC table entries to be pruned.

Port move detection is not performed for locked MAC table entries unless enabled (ANA_L2::LRN_CFG.LOCKED_PORTMOVE_DETECT_ENA).

Frames failing locked port move detection can be copied to CPU queue (ANA_L2::LRN_CFG.CPU_LRN_QU and ANA_L2::LRN_CFG.LOCKED_PORTMOVE_COPY_ENA).

4.21.6.1 Automatic Learning

The device can automatically add or update entries in the MAC table. New entries added by automatic learning are formatted as follows.

- VLD is set
- MAC is set to the frame's SMAC address
- FID set to the frame's REQ.FID
- ADDR_TYPE, ADDR is set to:
  UPSID_PN if received on a non aggregated port

  GLAG if received on a global aggregated port
- MAC_CPU_COPY (LRN::AUTO_LRN_CFG.AUTO_LRN_CPU_COPY)
- MAC_CPU_QU (LRN::AUTO_LRN_CFG.AUTO_LRN_CPU_QU)
- SRC_KILL (LRN::AUTO_LRN_CFG.AUTO_LRN_SRC_KILL_FWD)
- IGNORE_VLAN (LRN::AUTO_LRN_CFG.AUTO_LRN_IGNORE_VLAN)
- MIRROR (LRN::AUTO_LRN_CFG.AUTO_LRN_MIRROR)
- AGE_INTERVAL (LRN::AUTO_LRN_CFG.AUTO_AGE_INTERVAL)
- ISDX_LIMIT_IDX set to frame's REQ.ISDX_LIMIT_IDX retrieved from the ISDX table.
- All other fields are cleared

When a frame is received from a known station, that is, the MAC table already contains an entry for the received frame's (SMAC, VID/FID), the analyzer clears the AGED_FLAG for unlocked entries (LOCKED flag cleared) and optionally, locked entries (ANA_L2:COMMON:LRN_CFG.AGE_LOCKED_ENA). A cleared AGE_FLAG implies that the station is active. For more information, 4.21.7 Automated Aging (AUTOAGE).

For unlocked entries, ADDR_TYPE and ADDR fields are compared against the following:

- UPSID_PN if received on a non aggregated port
- GLAG if received on a global aggregated port
-

If there is a difference, the source entry ADDR_TYPE and ADDR is updated, which implies the station has moved to a new port.

Source check for entries stored with ADDR_TYPE=MC_IDX (=3) can be ignored (ANA_L2::LRN_CFG.IGNORE_MCIDX_PORTMOVE_ENA).

4.21.6.2 Port Move Log

A port move occurs when an existing MAC table entry for (MAC, FID) is updated with new port information (ADDR_TYPE and ADDR). Such an event is logged and reported (in ANA_L2::MOVELOG_STICKY) by setting the bit corresponding to the new port.

If port moves continuously occurs, it may indicate a loop in the network or a faulty link aggregation configuration.

Port moves for locked entries can be detected in ANA_L2::LRN_CFG.LOCKED_PORTMOVE_DETECT_ENA.

4.21.6.3 Port Learn Limits

It is possible to specify the maximum number of entries in the MAC table per port.

Port limits can be specified per logical port (local link aggregated port) and per global link aggregated port (GLAG) (ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LRN_CNT_LIMIT). These limits do not take multicast into account.

The current number of learned entries are always available in ANA_L2:PORT_LIMIT:PORT_LIMIT_STATUS.PORT_LRN_CNT and are updated whenever either an automatic learn operation or a CPU access is performed.

After a limit is reached, both automatic learning and CPU-based learning of unlocked entries are denied. A learn frame to be learned when the limit is reached can selectively be discarded, redirected to CPU, or copied to CPU (controlled in ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LIMIT_EXCEED_SEL). Trying to exceed the configured limit can also trigger interrupt (ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LIMIT_EXCEED_IRQ_ENA).

When learn limits are exceeded, a corresponding sticky bit is set (ANA_L2:PORT_LIMIT:PORT_LIMIT_STATUS.PORT_LRN_LIMIT_EXCEEDED_STICKY).

If a MAC table entry interrupt moves from one port to another port, it is also reflected in the current number of learned entries.

4.21.6.4 Filtering Identifier Learn Limits

It is possible to specify the maximum number of entries to be used per FID (ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LRN_CNT_LIMIT). These limits do not take multicast into account.

The current number of learned entries are always available in ANA_L2:LRN_LIMIT:FID_LIMIT_STATUS.FID_LRN_CNT.

After a limit is reached, both automatic learning and CPU-based learning of unlocked entries are denied. The learn frame that needs to be learned when the limit is reached can be discarded selectively, redirected to CPU, or copied to CPU (controlled by ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LIMIT_EXCEED_SEL). Attempts to exceed the configured limit can also trigger interrupt (ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LIMIT_EXCEED_IRQ_ENA).

When the learn limits are exceeded, a corresponding sticky bit is set (ANA_L2:LRN_LIMIT:FID_LIMIT_STATUS.FID_LRN_LIMIT_EXCEEDED_STICKY).

4.21.6.5 ISDX Learn Limits

It is possible to specify the maximum number of entries to be used per ISDX limiter (ANA_L2:ISDX_LIMIT:ISDX_LIMIT_CTRL.ISDX_LRN_CNT_LIMIT). These limits do not take multicast into account.

The ISDX limiter is assigned per ISDX in ANA_L2:ISDX:ISDX_LIMIT_CFG.ISDX_LIMIT_IDX. For non-service frames (ISDX = 0), a per-port ISDX limiter is configured in ANA_L2::PORT_ISDX_LIMIT_CFG.PORT_ISDX_LIMIT_IDX.

The current number of learned entries is always available in ANA_L2:ISDX_LIMIT:ISDX_LIMIT_STATUS.ISDX_LRN_CNT.

After a limit is reached, both automatic learning and CPU-based learning of unlocked entries are denied. A learn frame to be learned when the limit is reached can selectively be discarded, redirected to CPU, or copied to CPU (controlled by ANA_L2:ISDX_LIMIT:ISDX_LIMIT_CTRL.ISDX_LIMIT_EXCEED_SEL). Attempts to exceed the configured limit can also trigger interrupt (ANA_L2:ISDX_LIMIT:ISDX_LIMIT_CTRL.ISDX_LIMIT_EXCEED_IRQ_ENA).

When the learn limits are exceeded, a corresponding sticky bit is set (ANA_L2:ISDX_LIMIT:ISDX_LIMIT_STATUS.ISDX_LRN_LIMIT_EXCEEDED_STICKY).

4.21.6.6 Shared or Independent VLAN Learning in MAC Table

The device can be set up to do a mix of Independent VLAN Learning (IVL), where MAC addresses are learned separately on each VLAN and Shared VLAN Learning (SVL), where a MAC table entry is shared among a group of VLANs. For shared VLAN learning, a MAC address and a filtering identifier (FID) define each MAC table entry. A set of VIDs used for Shared VLAN learning maps to the same FID. Shared VLAN learning is only applicable for unicast traffic.

Addresses learned from frames with one VLAN Identifier (VID) may or may not be used to filter frames with the same SMAC but another VID, depending on management controls. If learned information is shared for two or more given VIDs, those VIDs map to a single Filtering Identifier (FID) and the term Shared VLAN Learning (SVL) is used to describe their relationship. If the information is not shared, the VIDs map to different FIDs and Independent VLAN Learning (IVL) is being used. FID is configured per VLAN (ANA_L3:VLAN:VLAN_CFG.VLAN_FID).

For example, if VID 16 and VID 17 use SVL and share FID 16, then a MAC entry (MAC #X, FID 16) is known for both VID 16 and VID 17.

If VID 16 and VID 17 use IVL and use FID 16 and FID 17, a MAC entry (MAC #X, FID 16) is only known for VID 16; that is, the MAC address is unknown for VID 17 (FID = 17).

In a VLAN-unaware operation, the device is set up to do SVL; that is, MAC addresses are learned for all VLANs. Protocol-based VLAN, where traffic is classified into multiple VLANs based on protocol, requires SVL.

4.21.7 Automated Aging (AUTOAGE)

This section describes how to configure automated age scanning. The following table lists the applicable registers.

Table 4-160. Automated Age Scan Registers Overview

Target::Register.Field	Description	Replication
LRN::AUTOAGE_CFG	Configures automated age scan of MAC table for each of the four AGE_INTERVALs.	4
LRN::AUTOAGE_CFG_1	Configures automated age scan of MAC table.	1
LRN::AUTOAGE_CFG_2	Configures automated age scan of MAC table.	1
LRN::EVENT_STICKY. AUTOAGE_SCAN_COMPLETED_STICKY	Signals completion of an automatic scan.	1
LRN::EVENT_STICKY. AUTOAGE_SCAN_STARTED_STICKY	Signals start of an automatic scan.	1
LRN::EVENT_STICKY. AUTOAGE_AGED_STICKY	Signals entries aged due to automatic aging.	1
LRN::EVENT_STICKY. AUTOAGE_REMOVE_STICKY	Signals entries removed due to automatic aging.	1
ANA_L2::FILTER_LOCAL_CTRL	Configures front port filters to be used by automatic aging and CPU scan.	1
ANA_L2::FILTER_OTHER_CTRL	Configures remote scan filter to be used by automatic aging and CPU scan.	1
ANA_L2::LRN_CFG.AGE_SIZE	Configures the AGE_FLAG size.	1

Entry AGE_FLAG is used to detect inactive sources. The AGE_FLAG field is cleared upon receiving traffic from a given source station and is periodically incremented by hardware to detect and delete inactive source entries.

The automated age scan scans the MAC table and checks age candidates (entries with LOCK flag set to 0) for activity.

If an entry's AGE_FLAG is set to maximum value (configured in ANA_L2::LRN_CFG.AGE_SIZE), the entry is deemed inactive and removed. If an entry's AGE_FLAG is less than the maximum value, the AGE_FLAG is incremented.

The flag is cleared when receiving frames from the station identified by the MAC table entry.

It is possible to configure and forget AUTOAGE, which means that once configured automated aging is performed without any further CPU assistance.

The interval between automatic aging can be controlled by specifying AGE_FLAG size (ANA_L2::LRN_CFG.AGE_SIZE), unit size (LRN::AUTOAGE_CFG.UNIT_SIZE), and the number of units between aging (LRN::AUTOAGE_CFG.PERIOD_VAL).

Setting LRN::AUTOAGE_CFG.UNIT_SIZE different from 0 to start the automatic aging.

To temporarily disable the automatic aging, set LRN::AUTOAGE_CFG. PAUSE_AUTO_AGE_ENA to 1. Additional control of automatic aging allows for instantaneously start and stop of current age scan (LRN::AUTOAGE_CFG_1.FORCE_HW_SCAN_SHOT and LRN::AUTOAGE_CFG_1.FORCE_HW_SCAN_STOP_SHOT).

It is also possible to do a controlled stopping of current age scan after it completes (LRN::AUTOAGE_CFG_1.FORCE_IDLE_ENA).

It is possible to selectively do age filtering of local ports specified in ANA_L2::FILTER_LOCAL_CTRL.FILTER_FRONTPORT_ENA or selectively do age filtering of entries learned on remote devices in a multichip configuration (ANA_L2::FILTER_OTHER_CTRL.FILTER_REMOTE_ENA). Both types must be enabled (LRN::AUTOAGE_CFG_1.USE_PORT_FILTER_ENA).

The state of automatic aging can be monitored (LRN::AUTOAGE_CFG_2.NEXT_ROW, LRN::AUTOAGE_CFG_2.SCAN_ONGOING_STATUS, LRN::EVENT_STICKY.AUTOAGE_SCAN_COMPLETED_STICKY and LRN::EVENT_STICKY.AUTOAGE_SCAN_STARTED_STICKY).

It is possible to observe whether entries are affected by a sticky event (LRN::EVENT_STICKY.AUTOAGE_AGED_STICKY and LRN::EVENT_STICKY.AUTOAGE_REMOVE_STICKY).

Example

For a 300-second age time, set four age periods of 75 seconds each:

1. Set ANA_L2::LRN_CFG.AGE_SIZE to 1.
2. Set LRN::AUTOAGE_CFG[0]. AUTOAGE_UNIT_SIZE to 3.
3. Set LRN::AUTOAGE_CFG[0]. AUTOAGE_PERIOD_VAL to 75.

4.21.8 Interrupt Handling

The following table lists the interrupt handling registers.

Table 4-161. Analyzer Interrupt Handling Registers

Target::Register.Field	Description	Replication
ANA_L2::INTR	Status of interrupt source	1
ANA_L2::INTR_ENA	Mask interrupt	1
ANA_L2::INTR_IDENT	Status of interrupt	1

A sticky status of all interrupt sources in the analyzer is available (ANA_L2::INTR), and all interrupt sources can individually be configured to trigger CPU interrupts (ANA_L2::INTR_ENA). The current CPU interrupt status is available (ANA_L2::INTR_IDENT).

The following analyzer interrupt sources exist.

- VCAP IS2 can be setup to trigger CPU interrupt (VCAP_S2_INTR) when an entry with enabled interrupt is hit (VCAP IS2 action INTR_ENA).
- It is possible to trigger interrupt (PORT_LRN_LIMIT_INTR) when learning on a port attempt to exceed a configured port learn limit (configured in ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LIMIT_EXCEED_IRQ_ENA).
- It is possible to trigger interrupt (FID_LIMIT_INTR) when learning on a FID attempt to exceed a configured limit (configured in ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LIMIT_EXCEED_IRQ_ENA).
- It is possible to trigger interrupt (ISDX_LIMIT_INTR) when learning on an ISDX limiter attempt to exceed a configured limit (configured in ANA_L2:ISDX_LIMIT:ISDX_LIMIT_CTRL.ISDX_LIMIT_EXCEED_IRQ_ENA).
- LRN access to the MAC table can be configured to trigger interrupt (LRN_ACCESS_COMPLETE_INTR) after completion. This is useful when multiple CPU scans must be performed as fast as possible, for example, when sorting the MAC addresses extracted from the entire MAC table using SCAN commands. For more information about using the SCAN command, see 4.21.4 SCAN Command.

4.22 Analyzer Access Control Forwarding, Policing, and Statistics

This section provides information about analyzer access control (ANA_AC) forwarding, policing, and statistics.

4.22.1 Mask Handling

This section describes how a number of port masks are applied to ensure correct forwarding.

- VLAN port mask from ANA_L3:VLAN: This mask is used to ensure frames are only forwarded within a dedicated VLAN. The VLAN mask handle protection by means of hardware assisted updates. For more information, see 4.15.2.1 VLAN Table Update Engine. The VLAN mask can be modified by VCAP CLM full action MASK_MODE and by VCAP IS2 action MASK_MODE.

- PGID port mask from ANA_AC:PGID: This mask is based on the DMAC lookup in the PGID table. One of the six flood PGID entries is used, if the DMAC is unknown. For DMAC known in the MAC table, the associated address is used to select the used PGID entry. The virtual forwarder selects between one of 32 sets of flooding and unicast port masks.
- Source port mask from ANA_AC:SRC: This mask is used to ensure frames are never forwarded to the port. It was received on and the used entry is found by looking up the source port.
- Global source port mask from ANA_AC:SRC: This mask is used to ensure frames are never forwarded to the global port. It was received on and the used entry is found by looking up the global stacking source port.
- Aggregation port mask from ANA_AC:AGGR: This mask is used to ensure frames are forwarded to one port within each aggregate destination port.
- Conversation-sensitive distribution (CSC) port mask from ANA_AC:CSD: This mask is used to ensure frames are forwarded to one port within each aggregate destination port.
- LAG reset port mask from ANA_AC:LAG_RST: This mask removes link aggregations globally defined by the aggregation or CSC port masks that do not apply to the virtual forwarder.
- REQ.port_mask and REQ.mask_mode: This mask is special in the sense that it is a general purpose mask that can be used for various security features and protection. The mask is controlled by VCAP CLM and VCAP IS2.

The following sections provide more information about the different port masks.

4.22.1.1 PGID Lookup

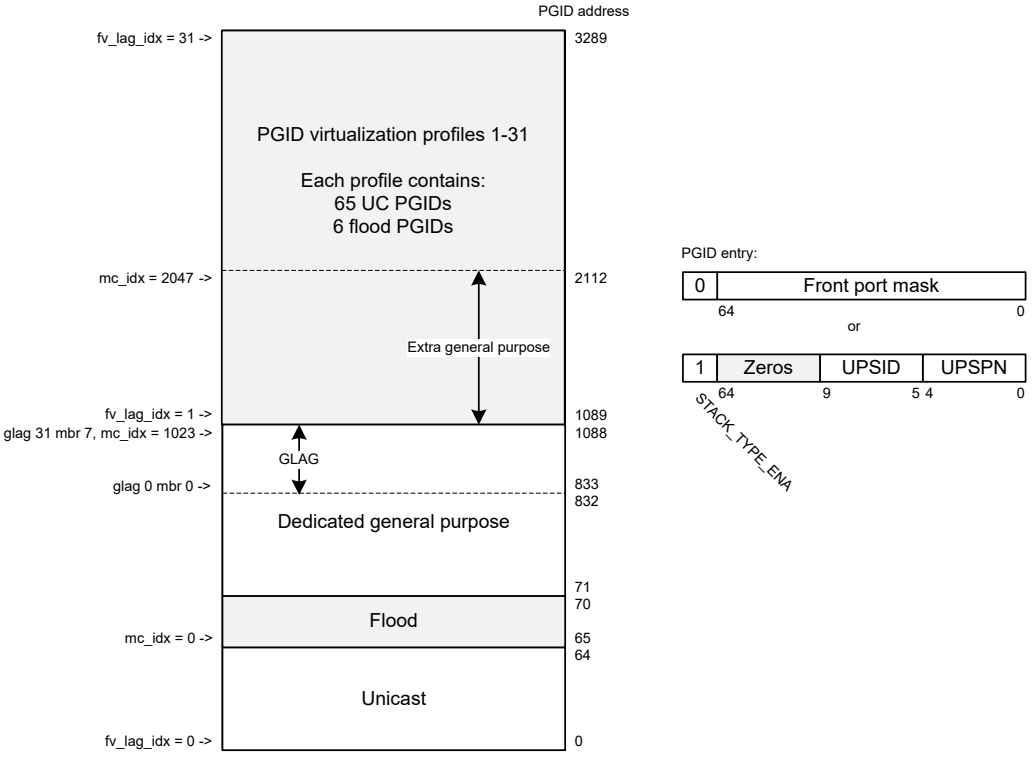
The following table lists the registers associated with PGID lookup.

Table 4-162. PGID Registers

Target::Register.Field	Description	Replication
ANA_AC:PGID:PGID_MISC_CFG	Configures how to interpret PGID_CFG and configures CPU copy with associated CPU queue.	per PGID
ANA_AC:PGID:PGID_CFG	Configures destination port mask or destination (UPSID,PN).	per PGID

The PGID table is organized as shown in the following figure.

Figure 4-47. PGID Layout



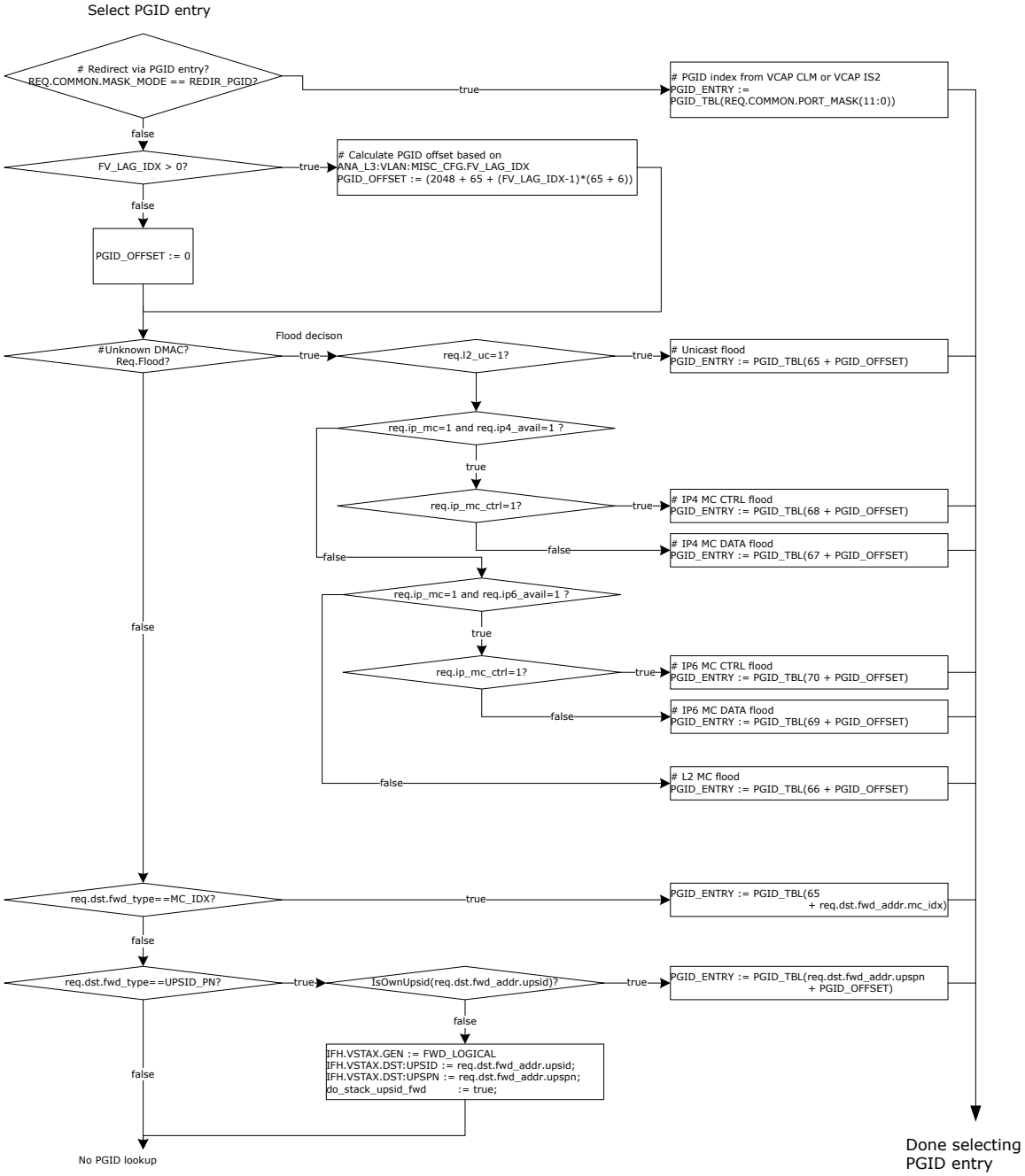
The forwarding process generates a forwarding decision that is used to perform a lookup in the PGID table. The following forwarding decisions are possible.

- Flood forwarding to one of the six flood masks located from address 65 to address 70 in the PGID table is used as destination mask.
- Unicast forwarding to {UPSID,UPSPN}. UPSID is checked against ANA_AC::COMMON_VSTAX_CFG.OWN_UPSID:
  If identical, destination mask is found based on lookup of UPSPN within the front port destinations of the PGID table (address 0 to 64).

  If not identical, destination mask is found based on lookup of UPSID in ANA_AC:UPSID:UPSID_CFG.
- The frame's forwarder virtual LAG profile (ANA_L3:VLAN:MISC.FV_LAG_IDX) offsets the PGID table for flood and unicast forwarding.
- Multicast forwarding: mc_idx is looked up in the PGID table with an offset of 65. Based on the returned entry's stack type bit:
  If cleared, the returned entry mask is interpreted as a destination mask.
  If set, the returned entry mask is {UPSID,UPSPN} and the destination mask is found based on lookup of UPSID in ANA_AC:UPSID:UPSID_CFG.
- REQ destination set (REQ.COMMON.PORT_MASK when REQ.COMMON.MASK_MODE=REPLACE_PGID). Destination set (REQ.COMMON.PORT_MASK from VCAP CLM or VCAP IS2 is directly used as destination mask without PGID lookup.
- REQ.L2_FWD cleared. Frame is discarded.
  It is possible to generate a CPU copy when using the PGID lookup. If ANA_AC:PGID:PGID_MISC_CFG.PGID_CPU_COPY_ENA is set in the used PGID entry, a copy is sent to CPU queue specified by ANA_AC:PGID:PGID_MISC_CFG.PGID_CPU_QU.

The following figure shows the PGID lookup decision.

Figure 4-48. PGID Lookup Decision Forwarding



The following debug events can be used to observe current forwarding:

ANA_AC:PS_STICKY:STICKY.PGID_CPU_MASK_STICKY,
ANA_AC:PS_STICKY:STICKY.NO_L2_L3_FWD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP6_MC_CTRL_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP6_MC_DATA_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP4_MC_CTRL_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP4_MC_DATA_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.L2_MC_FLOOD_STICKY, and ANA_AC:PS_STICKY:STICKY.UC_FLOOD_STICKY.

These events can be counted in the port statistics block by setting the corresponding counter mask: ANA_AC:PS_STICKY_MASK:STICKY_MASK.

4.22.1.2 Source Lookup

This section describes how to configure source port filtering. The following table lists the applicable registers.

Table 4-163. Source Table Registers

Target::Register.Field	Description	Replication
ANA_AC:SRC:SRC_CFG	Configures source port filtering	102

Source port filtering for each ingress port can be configured to ensure that frames are not bridged back to the port it was received on (ANA_AC:SRC[0-69]:SRC_CFG).

Ports part of a link aggregation group (LAG) are configured with identical source masks, with all member ports removed (bits corresponding to the member ports are cleared).

By default, the source port filtering table is indexed using the physical ingress port. Alternatively, the logical ingress port can be used (ANA_AC::PS_COMMON_CFG.SRC_LOOKUP_LPORT_ENA).

If a port is part of a global link aggregation group, a filter is applied to ensure that frames received at this port are not bridged back to any of the ports in the global link aggregation group of which the source port is part (ANA_AC:SRC[70-101]:SRC_CFG).

Local device ports part of a global link aggregation group must be cleared in the global aggregation source mask (via ANA_AC:SRC[70-101]:SRC_CFG).

Source port filtering can be disabled when VCAP CLM or VCAP IS2 action MASK_MODE is set to 4 (= REDIR_PGID).

Source port filtering is not applied when a frame is routed.

4.22.1.3 Aggregation Group Port Selection

This section describes how to configure the aggregation table. The following table lists the applicable registers.

Table 4-164. Aggregation Table Registers

Target::Register.Field	Description	Replication
ANA_AC:AGGR:AGGR_CFG	Configures aggregation port filtering.	16

The purpose of the aggregation table is to ensure that when a frame is destined for an aggregation group, it is forwarded to exactly one of the group's member ports.

The aggregation code REQ.AGGR_CODE generated in the classifier is used to look up an aggregation mask in the aggregation table. Aggregation mask is configured in ANA_AC:AGGR:AGGR_CFG.

For non-aggregated ports, there is a one-to-one correspondence between logical port (ANA_CL:PORT:PORT_ID_CFG.LPORT_NUM) and physical port, and the aggregation table does not change the forwarding decision.

For aggregated ports, all physical ports in the aggregation group map to the same logical port, and destination entries for a logical port in the PGID table must include all physical ports, which are members of the aggregation group. Therefore, all but one LAG member port must be removed from the destination port set.

The aggregation contribution can be disabled when VCAP CLM or VCAP IS2 action MASK_MODE is set to 4 (= REDIR_PGID).

For more information about link aggregation, see 4.14.1.9 Link Aggregation Code Generation

4.22.1.4 Global Link Aggregation Forwarding

This section describes how to configure forwarding to a GLAG. The following table lists the applicable registers.

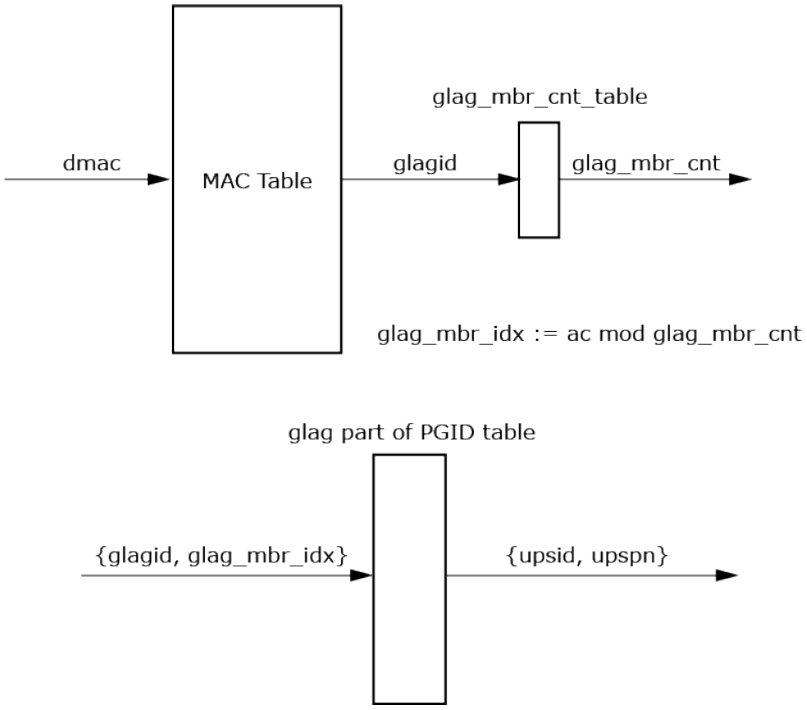
Table 4-165. GLAG Forward Registers

Target::Register.Field	Description	Replication
ANA_AC::COMMON_VSTAX_CFG	Configures VSTAX handles.	1
ANA_AC:GLAG:MBR_CNT_CFG	Configures number of members per GLAG.	32

The device performs a final port of exit (POE) calculation in ingress device when destination is a global aggregated port. GLAG POE calculation must be enabled (ANA_AC::COMMON_VSTAX_CFG.VSTAX2_GLAG_ENA). When enabled, GLAG forwarding uses a portion of the PGID table (addresses 833 to 1088). These addresses must use UPSID and UPSPN encoding by setting the stack interpret bit.

POE calculation, which selects one of the global aggregated as destination port, is shown in the following figure.

Figure 4-49. GLAG Port of Exit Calculation



The total number of ports part of a GLAG must be specified in ANA_AC:GLAG:MBR_CNT_CFG.

The corresponding destination must be configured in PGID table for each GLAG member.

4.22.1.5 Conversation-Sensitive Frame Distribution

This section describes how to configure the conversation-sensitive distribution (CSD) table. The following table lists the applicable registers.

Table 4-166. CSD Table Registers

Target::Register.Field	Description	Replication
ANA_AC::PS_COMMON_CFG.CSD_LOOKUP_SEL	Configures conversation identifier classification used for CSD port filtering.	Common
ANA_AC:CSD:CSD	Configures CSD port filtering.	4096

The purpose of the CSD table is to assign a conversation identifier and to use that to select a link aggregation member port when forwarded to an aggregation group. Conversation-sensitive distribution is an alternative to aggregation based on an aggregation code.

The conversation identifier is assigned to one of the following ways:

- Use frame's outer VID. The outer VID is derived after removing the number of VLAN tags specified in IFH.TAGGING.POP_CNT. Note that pushing of new VLAN tags are not taken into account.
- Use frame's classified VID (IFH.VSTAX.TAG.VID).
- Use frame's classified GVID (IFH.ENCAP.GVID).

The conversation identifier is used as index to look up a CSD mask in the CSD table.

For non-aggregated ports, there is a one-to-one correspondence between logical port (ANA_CL:PORT: PORT_ID_CFG.LPORT_NUM) and physical port, and the CSD table does not change the forwarding decision.

For aggregated ports, all physical ports in the aggregation group map to the same logical port, and destination entries for a logical port in the PGID table must include all physical ports, which are members of the aggregation group. Therefore, all but one LAG member port must be removed from the destination port set.

The CSD contribution can be disabled when VCAP CLM or VCAP IS2 action MASK_MODE is set to 4 (= REDIR_PGID).

4.22.1.6 Link Aggregation Reset Selection

This section describes how to configure the link aggregation reset table. The following table lists the applicable registers.

Table 4-167. Link Aggregation Reset Table Registers

Target::Register.Field	Description	Replication
ANA_AC:LAG_RST	Configures link aggregation reset table.	32

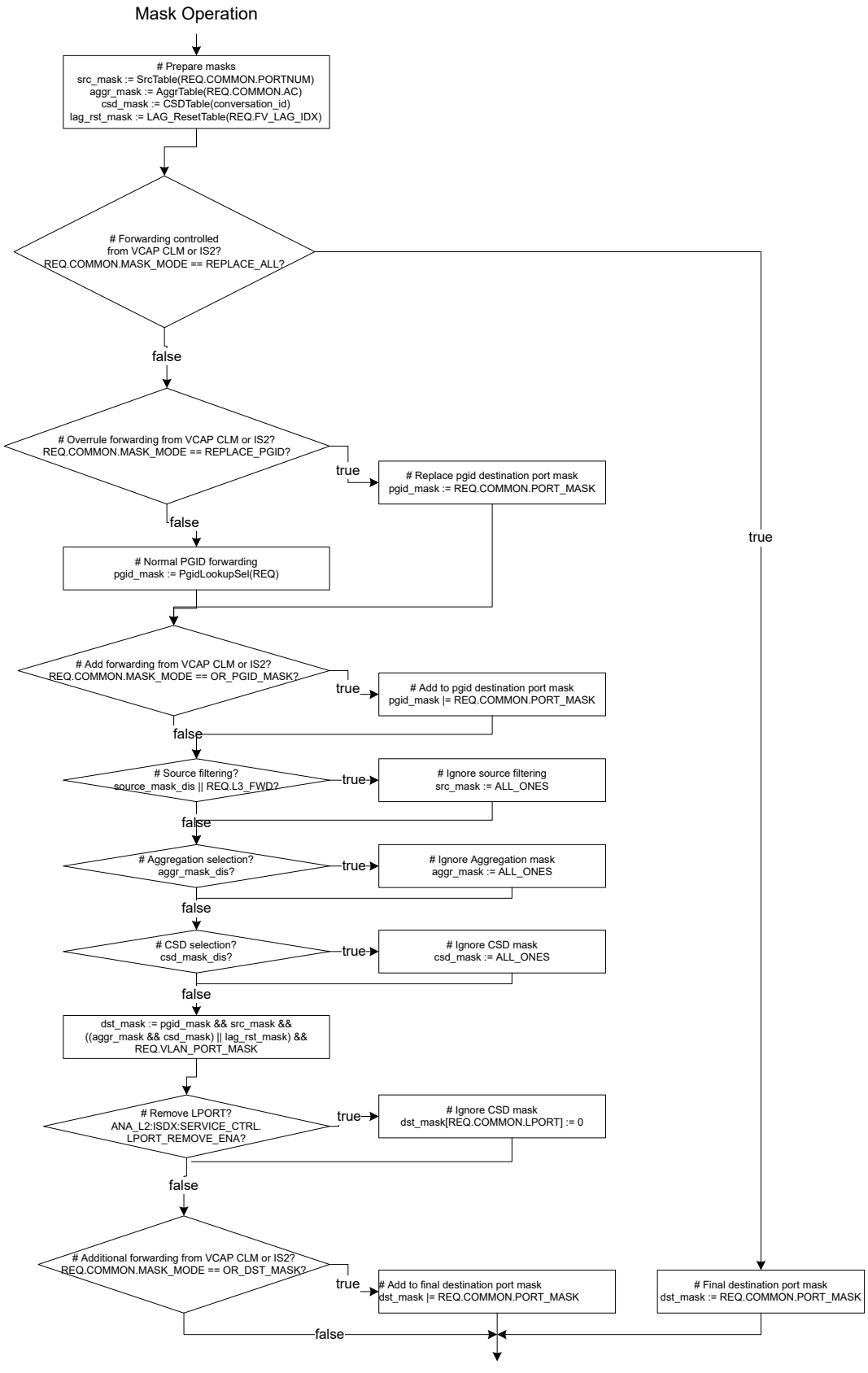
The purpose of the link aggregation reset table is to remove link aggregation groups not applying to the virtual forwarder. This is done by resetting specific bits in the aggregation and CSD port masks before applying the aggregation contribution to the forwarding decision.

The link aggregation reset table is indexed using the forwarder virtual LAG profile retrieved from the VLAN table (ANA_L3:VLAN:MISC.FV_LAG_IDX).

4.22.1.7 Port Mask Operation

The final destination port masks are deducted as shown in the following figure.

Figure 4-50. Port Mask Operation



4.22.2 Policing

This section describes the functions of the polices. The following table lists the applicable registers.

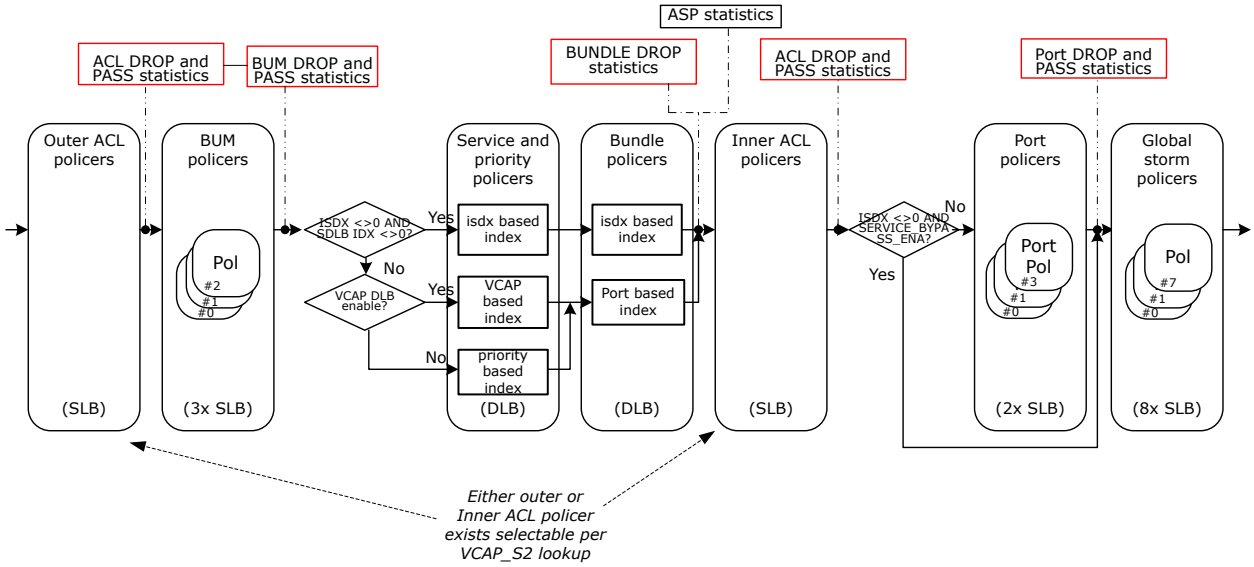
Table 4-168. Policer Control Registers

Target::Register.Field	Description	Replication
ANA_AC_POL::POL_ALL_CFG	Configures general control settings.	1
ANA_AC_POL::POL_ACL_CTRL	Configures VCAP policer's mode of operation.	64
ANA_AC_POL::POL_ACL_RATE_CFG	Configures VCAP policer's peak information rate	64
ANA_AC_POL::POL_ACL_THRES_CFG	Configures VCAP policer's burst capacity.	64
ANA_AC_POL::POL_STORM_CTRL	Configures storm policer's mode of operation.	8
ANA_AC_POL::POL_STORM_RATE_CFG	Configures storm policer's peak information rate	8
ANA_AC_POL::POL_STORM_THRES_CFG	Configures Storm policer's burst capacity.	8
ANA_AC_POL::POL_PORT_RATE_CFG	Configures port policer's peak information rate.	Per port per port policer
ANA_AC_POL::POL_PORT_THRES_CFG_0	Configures port policer's burst capacity.	Per port per port policer
ANA_AC_POL::POL_PORT_THRES_CFG_1	Configures port policer's hysteresis when used for flow control.	Per port per port policer
ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG	Configures port policer's mode of operation.	Per port per port policer
ANA_AC_POL:POL_PORT_CTRL:POL_PORT_GAP	Configures port policer's gap value used to correct size per frame.	Per port
ANA_AC_POL::POL_PORT_FC_CFG	Configures port policer's flow control settings	Per port
ANA_AC_POL::POL_STICKY	Captures various policer events for diagnostic purposes.	1

4.22.2.1 Policer Hierarchy

There are multiple levels of policing, which operates in configurable hierarchies, as shown in the following illustration.

Figure 4-51. Policer Hierarchy



At most, each frame can see one ACL policer. ACL policing can occur as either outer or inner ACL-based. VCAP IS2 action IS_INNER_ACL allows ACL policing to occur before or after service DLB policing and service statistics.

The ACL, port, broadcast/unicast/multicast (BUM), and storm policers are single leaky bucket (SLB) policers, whereas, the service and bundle policers are dual leaky bucket (DLB) policers. The service policers are MEF10.3 compliant policers.

The BUM policers group the leaky buckets into sets of three, covering broadcast, unicast, and multicast traffic.

The service policers can be used as priority policers for non-service traffic when ISDX is zero and ISDX_SDLB index is zero. The priority policer operation is controlled through ANA_L2:COMMON:FWD_CFG.QUEUE_DEFAULT_SDLB_ENA and ANA_L2:COMMON:PORT_DLB_CFG.QUEUE_DLB_IDX.

The bundle policers are only available for services. The index is controlled through ANA_L2:ISDX:MISC_CFG.BDLB_IDX. However, it is possible to use the bundle policers as port policers for non-service traffic when ISDX is zero. The port policer operation is controlled through ANA_L2:COMMON:FWD_CFG.PORT_DEFAULT_BDLB_ENA and ANA_L2:COMMON:PORT_DLB_CFG.PORT_DLB_IDX.

It is possible to specify a pipeline point for the service policers (ANA_L2:ISDX:MISC_CFG.PIPELINE_PT). This can be used to specify where in the processing flow BUM, SDLB, and BDLB policers are active (default set to NONE, which disables any pipeline effect).

The service statistics reflect the decisions made by the service policer. This implies that if a service policer marks a frame yellow, then the yellow service counter is incremented even though the frame might be discarded by one of the subsequent policers.

All policers can be configured to add between –64 and 63 bytes per frame to adjust the counted frame size for inter-frame gap and/or encapsulation.

The following table provides an overview of the main parameters associated with the supported policers. Note that the bursts and rate values shown for the service policers are examples only. The service policers are programmable and other combinations are possible. See 4.23 Leaky Bucket for SDLB.

Table 4-169. Policer Overview

Policer	Replication	Type of Policer	Burst, Granularity (Kilobytes)	Burst, Maximum (Kilobytes)	Rate, Granularity (Bits/sec)	Rate, Maximum (Megabits/sec [1])
BUM	3 x 1,024	Single leaky bucket	2	510	16,001	65
					128,008	523
					1,024,066	4,190
					8,192,524	25,000
Service	4,096	Dual leaky bucket	2488	2,046	4,882	10
				4,092	24,414	100
				8,184	305,250	2,500
				8,184	3,052,503	25,000
Priority	8 per port	See service	See service	See service	See service	See service
Bundle	1,024	Dual leaky bucket	2	510	16,001	65
					128,008	523
					1,024,066	4,190
					8,192,524	25,000
ACL	64	Single leaky bucket	8	504	25,040	25,000
Port	4 per port	Single leaky bucket	8	504	25,040	25,000
Storm	8	Single leaky bucket	8	504	25,040	25,000

Note:

1. One Kilobyte = 1,024 bytes, 1 Megabit/sec = 1,000,000 bits/sec.

4.22.2.2 BUM Policing

Each entry in the BUM single leaky bucket table contains the following fields.

Table 4-170. BUM Single Leaky Bucket Table Entries

Field	Bits	Description
ANA_AC_POL:BUM_SLB:SLB_CFG.TIMESCALE_VAL	2	Configures policer rate granularity.
ANA_AC_POL:BUM_SLB:SLB_CFG.CIR_INC_DP_VAL	2	Controls how much drop precedence (DP) is incremented for excess traffic.
ANA_AC_POL:BUM_SLB:SLB_CFG.GAP_VAL	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap and optional encapsulation through configuration.
ANA_AC_POL:BUM_SLB:LB_CFG[2:0].THRES_VAL	3 x 8	Configures burst size.
ANA_AC_POL:BUM_SLB:LB_CFG[2:0].RATE_VAL	3 x 12	Configures rate.

.....continued		
Field	**Bits**	**Description**
ANA_AC_POL:BUM_SLB:SLB_CFG.ENCAP_DATA_DIS	1	Configures if stripped encapsulation data (normalized data) is policed by the policer.
ANA_AC_POL:BUM_SLB:MISC_CFG.FRAME_RATE_ENA	1	Configures frame rate operation.

The BUM policers are indexed through the VLAN table (ANA_L3:VLAN:BUM_CFG.BUM_SLB_IDX). This indexing can be overruled for services through the ISDX table (ANA_L2:ISDX:MISC_CFG.BUM_SLB_IDX and ANA_L2:ISDX:MISC_CFG.BUM_SLB_ENA).

Each BUM policer contains three leaky buckets. Rates and thresholds are configured through ANA_AC_POL:BUM_SLB:LB_CFG).

Traffic for the three BUM leaky buckets is configurable in ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG. This provides a flexible allocation of traffic for the policers. For example, it is possible to have BUM configured as:

- Leaky bucket 0: Broadcast traffic (ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG[0].TRAFFIC_TYPE_MASK = 1)
- Leaky bucket 1: Unknown unicast traffic (ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG[1].TRAFFIC_TYPE_MASK = 4)
- Leaky bucket 2: Unknown multicast traffic (ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG[2].TRAFFIC_TYPE_MASK = 2)

BUM policers can be configured as frame-based policers (ANA_AC_POL:BUM_SLB:MISC_CFG.FRAME_RATE_ENA).

The BUM policers' unit size can be adjusted (ANA_AC_POL:COMMON_BUM_SLB:DLB_CTRL.BASE_TICK_CNT) to configure the smallest rate granularity. The granularity scaling can be configured using the TIMESCALE_VAL configuration parameter.

The BUM statistics count on the following events (configured through ANA_AC:STAT_GLOBAL_CFG_BUM:STAT_GLOBAL_EVENT_MASK):

- Bit 0: Count Broadcast traffic discarded by BUM policer
- Bit 1: Count Multicast traffic discarded by BUM policer
- Bit 2: Count Unicast traffic discarded by BUM policer
- Bit 3: Count Broadcast traffic applicable for BUM policer, but not discarded
- Bit 4: Count Multicast traffic applicable for BUM policer, but not discarded
- Bit 5: Count Unicast traffic applicable for BUM policer, but not discarded.

4.22.2.3 Priority Policing

The configuration parameters for priority policing are listed in the following table.

Table 4-171. Priority Policer Table Entry

Field	Description
ANA_L2::FWD.QUEUE_DEFAULT_SDLB_ENA	Enables priority policers for non-service frames.
ANA_L2:PORT:PORT_DLB_CFG.QUEUE_DLB_IDX	Specifies which DLB policers are used for priority policing.

Non-service frames (which ISDX = 0 and ANA_L2:ISDX:DLB_CFG.DLB_IDX = 0) can use the service policers as priority policers. This is enabled in ANA_L2::FWD_CFG.QUEUE_DEFAULT_SDLB_ENA. The frame's ingress port and classified QoS class select the priority policer. The configuration of the priority policer follows the configuration of the service policers. For more information, see 4.22.2.4 Bundle Dual Leaky Bucket (DLB) Policing.

Note that a DLB policer index enabled by VCAP IS2 action ACL_MAC[16] takes precedence over the priority policing. In that case, the frame is policed by the VCAP selected policer and not the priority policer.

4.22.2.4 Bundle Dual Leaky Bucket (DLB) Policing

Each entry in the bundle dual leaky bucket (BDLB) table contains the fields listed in the following table.

Table 4-172. Bundle Dual Leaky Bucket Table Entries

Field	Bits	Description
ANA_AC_POL:BDLB:DLB_CFG.TIMESCALE_VAL	2	Configures BDLB policer granularity.
ANA_AC_POL:BDLB:DLB_CFG.COLOR_AWARE_LVL	2	Corresponds to MEF color mode. Specifies the highest DP level that is treated as green (that is, service frames with DP level above COLOR_AWARE_LVL are considered yellow). COLOR_AWARE_LVL == 3 correspond to color-blind.
ANA_AC_POL:BDLB:DLB_CFG.COUPLING_MODE	1	MEF Coupling Flag (CF). Depending on the setting of COUPLING_MODE, LB_CFG[0] and LB_CFG[1] must be configured as follows: If COUPLING_MODE = 0, LB_CFG[0].RATE_VAL must be configured to MEF CIR. LB_CFG[0].THRES_VAL must be configured to MEF CBS. LB_CFG[1].RATE_VAL must be configured to MEF EIR. LB_CFG[1].THRES_VAL must be configured to MEF EBS. If COUPLING_MODE = 1, LB_CFG[0].RATE_VAL must be configured to MEF CIR. LB_CFG[0].THRES_VAL must be configured to MEF CBS. LB_CFG[1].RATE_VAL must be configured to MEF EIR + MEF CIR. LB_CFG[1].THRES_VAL must be configured to MEF EBS + MEF CBS.
ANA_AC_POL:BDLB:DLB_CFG.CIR_INC_DP_VAL	2	Controls how much drop precedence (DP) is incremented for excess traffic.
ANA_AC_POL:BDLB:DLB_CFG.GAP_VAL	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap, and optional encapsulation through configuration.
ANA_AC_POL:BDLB:LB_CFG[0].THRES_VAL	8	Configures committed burst size (CBS).
ANA_AC_POL:BDLB:LB_CFG[1].THRES_VAL	8	Configures BDLB's second threshold. See COUPLING_MODE in preceding rows.
ANA_AC_POL:BDLB:LB_CFG[0].RATE_VAL	12	Configures BDLB's second rate. See COUPLING_MODE in preceding rows.

Each MEF DLB policer supports the following configurations.

- Two rates. Specified in ANA_AC_POL:BDLB:LB_CFG[0-1].RATE_VAL.

- Two burst sizes. Specified in ANA_AC_POL:BDLB:LB_CFG[0-1].THRES_VAL.
- Color mode: Color-blind or color-aware. Specifies a DP level to separate traffic as green or yellow (ANA_AC_POL:BDLB:DLB_CFG.COLOR_AWARE_LVL). Frames classified with REQ.DP below or equal to COLOR_AWARE_LVL are treated as green. Frames with REQ.DP above COLOR_AWARE_LVL are treated as yellow. A policer is color-blind, if configured with a COLOR_AWARE_LVL of 3.
- Coupling flag. Coupled or uncoupled (ANA_AC_POL:BDLB:DLB_CFG.COUPLING_MODE).
- Change DP level. The increase to REQ.DP level when CIR rate is exceeded (ANA_AC_POL:BDLB:DLB_CFG.CIR_INC_DP_VAL).

In addition, the following parameters can also be configured per policer:

- Leaky bucket calculations can be configured to include or exclude preamble, inter-frame gap and an optional encapsulation (ANA_AC_POL:BDLB:DLB_CFG.GAP_VAL).
- Policers can be configured to police CPU traffic, front port forwarded traffic, or both (ANA_AC_POL:BDLB:DLB_CFG.TRAFFIC_TYPE_MASK).

The BDLB policers must also be enabled in ANA_AC_POL:COMMON_BDLB:DLB_CTRL.LEAK_ENA and ANA_AC_POL:COMMON_BDLB:DLB_CTRL.DLB_ADD_ENA.

The BDLB policer unit size can be adjusted (ANA_AC_POL:COMMON_BDLB: DLB_CTRL.BASE_TICK_CNT) to configure the smallest rate granularity. The granularity scaling can be configured using the TIMESCALE_VAL configuration parameter.

The following BDLB policer debug events are available:

- Dropping traffic due to BDLB policer can be identified (ANA_AC_POL:POL_ALL_CFG: POL_STICKY1.POL_BDLB_DROP_STICKY).
- Traffic received without triggering CIR and PIR policing (ANA_AC_POL:COMMON_BDLB:DLB_STICKY.CIR_PIR_OPEN_STICKY).
- Committed information rate exceeded (ANA_AC_POL:COMMON_BDLB:DLB_STICKY.CIR_EXCEEDED_STICKY).
- Peak information rate exceeded (ANA_AC_POL:COMMON_BDLB:DLB_STICKY.PIR_EXCEEDED_STICKY).

4.22.2.5 ACL Policing

Each ACL policer entry contains the fields listed in the following table.

Table 4-173. ACL Policer Table Entries

Field	Bits	Description
ACL_TRAFFIC_TYPE_MASK	2	Configures the traffic types to be taken into account by the policer.
FRAME_RATE_ENA	1	Configures frame rate mode for the ACL policer, where rates are measured in frames per second instead of bytes per second.
DP_BYPASS_LVL	2	Controls the lowest DP level that is taken into account. That is, traffic with DP_BYPASS_LVL below this value is ignored and not policed.
GAP_VALUE	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap, and optional encapsulation through configuration.
ACL_THRES	6	Configures ACL policer burst capacity.
ACL_RATE	17	Configures ACL policer rate.

At most, a frame can trigger one ACL policer. ACL policers are enabled by specifying a rate in ANA_AC_POL:POL_ALL_CFG:POL_ACL_RATE_CFG.ACL_RATE and also enabled by a VCAP IS2 hit. For more information, see 4.10.3.4 VCAP IS2.

The policer burst capacity can be specified with 4K granularity (ANA_AC_POL:POL_ALL_CFG:POL_ACL_THRES_CFG.ACL_THRES).

The following parameters can also be configured per policer.

- The leaky bucket calculation can be configured to include or exclude preamble, inter-frame gap, and an optional encapsulation (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.GAP_VALUE).
- Traffic with DP level below a certain level can be configured to bypass the policers (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.DP_BYPASS_LVL).
- Each policer can be configured to measure frame rates instead of bit rates (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.FRAME_RATE_ENA).
- Each ACL policer can be configured to operate on frames towards CPU and/or front ports (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.ACL_TRAFFIC_TYPE_MASK).

Traffic dropped by an ACL policer can be counted in the ACL statistics block and optionally also in the port statistic block. For more information, see 4.22.3 Analyzer Statistics

The following ACL policer debug events are available:

- Bypass of policer due to pipeline handling can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_PT_BYPASS_STICKY).
- Bypass of policer due to bypass level can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_BYPASS_STICKY).
- Dropping of traffic due to ACL policer can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_DROP_STICKY).
- Policers that are active but not closed can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_ACTIVE_STICKY).

For example, to use ACL policers to limit traffic to CPU only, the following configuration is required:

- Set up a VCAP S2 rule to enable ACL policer 5.
- Configure ACL policer 5 to only allow 100 frames per second towards CPU: # the rate is 10 times the configured value ANA_AC_POL:POL_ALL_CFG:POL_ACL_RATE_CFG[5].ACL_RATE = 10 ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL[5].FRAME_RATE_ENA = 1
- Do not accept burst: ANA_AC_POL:POL_ALL_CFG:POL_ACL_THRES_CFG[5].ACL_THRES = 0
- Only police traffic towards CPU: ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL[5].ACL_TRAFFIC_TYPE_MASK = 2

4.22.2.6 Port Policing

Each port policer entry contains the fields listed in the following table.

Table 4-174. Port Policer Table Entry

Field	Bits	Description
TRAFFIC_TYPE_MASK	8	Configures the traffic types to be taken into account by the policer.
FRAME_RATE_ENA	1	Configures frame rate mode for the port policer where rates are measured in frames per second instead of bytes per second.
LIMIT_NONCPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards front ports.
LIMIT_CPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards CPU.
CPU_QU_MASK	8	Configures policing of frames to the individual CPU queues for the port policer (see TRAFFIC_TYPE_MASK).
FC_ENA	1	Configures port policer to trigger flow control.

.....continued

Field	Bits	Description
FC_STATE	1	Current flow control state.
DP_BYPASS_LVL	2	Controls the lowest DP level that is taken into account. That is, traffic with DP_BYPASS_LVL below this value is ignored and not policed.
GAP_VALUE	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap, and optional encapsulation through configuration.
PORT_THRES0	6	Configures port policer burst capacity.
PORT_THRES1	6	Configures port policer hysteresis size when a port policer is in flow control mode (see FC_ENA).
PORT_RATE	20	Configures port policer rate.

Frames applicable for policing are for any frames received by the MAC and forwarded to the classifier. Short frames (less than 148 bytes) with errors, pause frames, or MAC control frames are not forwarded by the MAC, and therefore, not accounted for in the policers. That is, they are not policed and do not add to the rate measured by the policers.

Port policers are enabled by configuring the traffic type to be policed (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.TRAFFIC_TYPE_MASK) and specifying a corresponding rate (ANA_AC_POL:POL_PORT_CFG:POL_PORT_RATE_CFG.PORT_RATE).

The policer burst capacity can be specified in ANA_AC_POL:POL_PORT_CFG:POL_PORT_THRES_CFG_0.PORT_THRES0.

Policing of traffic destined for CPU port or ports can be controlled (in ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.TRAFFIC_TYPE_MASK(7) = 0 and CPU bypass mask in ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.CPU_QU_MASK).

Port policers can individually be configured to affect frames towards CPU ports (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.LIMIT_CPU_TRAFFIC_ENA) or front ports (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.LIMIT_NONCPU_TRAFFIC_ENA).

The port policers can be configured for either serial or parallel operation (ANA_AC_POL::POL_ALL_CFG.PORT_POL_IN_PARALLEL_ENA):

- Serial: The policers are checked one after another. If a policer is closed, the frame is discarded and the subsequent policer buckets are not updated with the frame.
- Parallel: The policers are working in parallel independently of each other. Each frame is added to a policer bucket, if the policer is open, otherwise, the frame is discarded. A frame may be added to one policer although another policer is closed.

The following parameters can also be configured per policer.

- The leaky bucket calculation can be configured to include or exclude preamble, inter-frame gap and an optional encapsulation (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_GAP.GAP_VALUE).
- Each policer can be configured to measure frame rates instead of bit rates (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.FRAME_RATE_ENA).
- Traffic with DP level below a certain level can be configured to bypass the policers (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.DP_BYPASS_LVL)

By default, a policer discards frames (to the affected port type: CPU and/or front port or ports) while the policer is closed. A discarded frame is not forwarded to any ports (including the CPU).

However, each port policer has the option to run in flow control where the policer instead of discarding frames instructs the MAC to issue flow control pause frames (ANA_AC_POL:POL_ALL_CFG:POL_PORT_FC_CFG.FC_ENA). When operating in Flow control mode, it is possible to specify a hysteresis, which controls when the policer can re-open after having closed

(ANA_AC_POL:POL_PORT_CFG:POL_PORT_THRES_CFG_1. PORT_THRES1). The current flow control state is accessible (ANA_AC_POL:POL_ALL_CFG:POL_PORT_FC_CFG.FC_STATE).

Note:
Flow control signaling out of ANA_AC must be enabled (ANA_AC_POL::POL_ALL_CFG.PORT_FC_ENA).

To improve fairness between small and large frames being policed by the same policer a hysteresis can be specified for drop mode (ANA_AC_POL:POL_PORT_CFG: POL_PORT_THRES_CFG_1.PORT_THRES1), which controls when the policer can re-open after having closed. By setting it to a value larger than the maximum transmission unit, it can be guaranteed that when the policer opens again, all frames have the same chance of being accepted.

Port policers can be configured to operate on logical ports instead of physical ports (ANA_AC_POL::POL_ALL_CFG.LPORT_POLICE_ENA), and thereby, allow policing of aggregated port bandwidth.

Traffic dropped by a policer can be counted by the port statistic block.

The following port policer debug events are available.

- Bypass of policer due to pipeline handling can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_PT_BYPASS_STICKY).
- Bypass of policer due to DP bypass level can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_BYPASS_STICKY).
- Dropping of traffic towards CPU due to policer can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_DROP_CPU_STICKY).
- Dropping of traffic towards front port due to policer can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_DROP_FWD_STICKY).
- Policers that are active, but not closed can be identified per policer (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_ACTIVE_STICKY).
- Policer triggering flow control can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_FC_STICKY).
- Policer leaving flow control state can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_FC_CLEAR_STICKY).

4.22.2.7 Storm Policing

Each storm policer entry contains the fields in the following table. The fields are all located within the ANA_AC_POL:POL_ALL_CFG register group.

Table 4-175. Storm Policer Table Entry

Field	Bits	Description
STORM_TRAFFIC_TYPE_MASK	8	Configures the traffic types to be taken into account by the policer.
STORM_FRAME_RATE_ENA	1	Configures frame rate mode for the ACL policer, where rates are measured in frames per second instead of bytes per second.
STORM_LIMIT_NONCPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards front ports.
STORM_LIMIT_CPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards CPU.
STORM_CPU_QU_MASK	8	Configures policing of frames to the individual CPU queues for the port policer (see TRAFFIC_TYPE_MASK).
STORM_GAP_VALUE	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap and optional encapsulation through configuration
STORM_THRES	6	Configures storm policer burst capacity.
STORM_RATE	20	Configures storm policer rate.

Frames applicable for policing are any frame received by the MAC and forwarded to the classifier. Short frames (less than 148 bytes) with errors, pause frames, or MAC control frames are not forwarded by the MAC and therefore not accounted for in the policers. That is, they are not policed and do not add to the rate measured by the policers.

Storm policers are enabled by configuring the traffic type to be policed (POL_STORM_CTRL.STORM_TRAFFIC_TYPE_MASK) and specifying a corresponding rate (POL_STORM_RATE_CFG.STORM_RATE).

The policer burst capacity can be specified in POL_STORM_THRES_CFG.

Policing of traffic destined for CPU port or ports can be controlled (POL_STORM_CTRL.STORM_TRAFFIC_TYPE_MASK(7) = 0 and CPU bypass mask in POL_STORM_CTRL.STORM_CPU_QU_MASK).

Storm policers can be configured individually to affect frames towards CPU ports (POL_STORM_CTRL.STORM_LIMIT_CPU_TRAFFIC_ENA) or front ports (POL_STORM_CTRL.STORM_LIMIT_NONCPU_TRAFFIC_ENA).

The following parameters can also be configured per policer.

- Leaky bucket calculation can be configured to include or exclude preamble, inter-frame gap, and an optional encapsulation (POL_ALL_CFG.STORM_GAP_VALUE).
- Each policer can be configured to measure frame rates instead of bit rates (POL_STORM_CTRL.STORM_FRAME_RATE_ENA).

Traffic dropped by a storm policer can be counted by the port statistic block.

The following storm policer debug events are available.

- Dropping of traffic towards CPU due to policer can be identified (POL_STICKY.POL_STORM_DROP_CPU_STICKY).
- Dropping of traffic towards front port due to policer can be identified (POL_STICKY.POL_STORM_DROP_FWD_STICKY).
- Policers that are active, but not closed, can be identified per policer (POL_STICKY.POL_STORM_ACTIVE_STICKY).

4.22.3 Analyzer Statistics

This section describes how to configure and collect statistics. There are y six statistics counter blocks in the analyzer.

- Port statistics: Four 40-bit counters are available for each port.
- Priority statistics: Two 40-bit counters are available for each priority.
- BUM policer statistics: Six 40-bit counters are available for each BUM policer.
- ACL policer statistics: Two 40-bit counters are available for each ACL policer.
- Ingress router leg statistics: Eight IPv4 40-bit counters and eight IPv6 40-bit counters are available for each ingress router leg.
- Egress router leg statistics: Eight IPv4 40-bit counters and eight IPv6 40-bit counters are available for each egress router leg.

Counters can be set up to count frames or bytes based on configurable criteria. This is described in the following sections.

4.22.3.1 Port Statistics

The following table lists the applicable port statistics registers.

Table 4-176. Analyzer Port Statistics Register

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_PORT.STAT_GLOBAL_EVENT_MASK	Configures global event mask per counter.	4
ANA_AC:STAT_GLOBAL_CFG_PORT:STAT_RESET.RESET	Initializes or resets all statistic counters and the sticky bits.	1

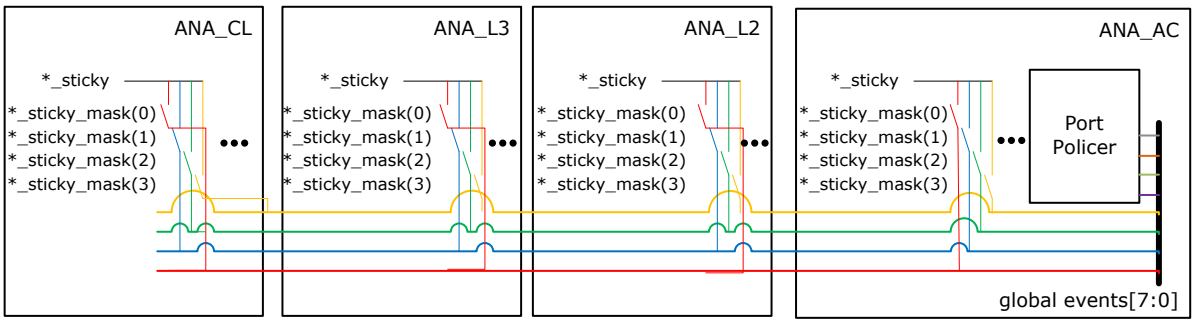
.....continued		
Target::Register.Field	**Description**	**Replication**
ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_PRIO_MASK	Configures counting of frames with certain priorities.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_CNT_FRM_TYPE	Configures the frame type to be counted.	per port× 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_CNT_BYTE	Configures counting of bytes or frames.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_LSB_CNT	Least significant 32 bits.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_MSB_CNT	Most significant 8 bits.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_EVENTS_STICKY	Sticky bits for counter events.	per port

The port statistics block allows counting of a wide variety of frame events. These are represented by a 12-bit event mask.

- Bits 0–3 represent any stick bit contribution from preceding blocks in ANA.
- Bits 4–11 represent port policer events.
- Bits 12–15 represent storm and ACL policer events and loopback frames.

The propagation of the event mask from the preceding blocks in the analyzer to ANA_AC is shown in the following figure, except for bits 8–11. See ANA_AC:STAT_GLOBAL_CFG_PORT:STAT_GLOBAL_EVENT_MASK.GLOBAL_EVENT_MASK.

Figure 4-52. Sticky Events Available as Global Events

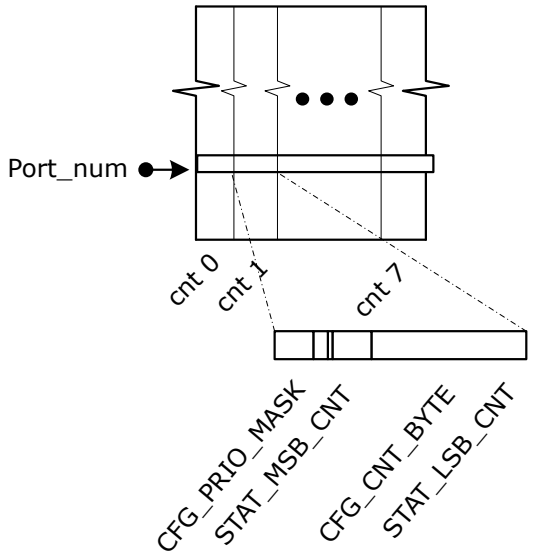


The global events [3:0] are typically allocated for detected control frames, frames dropped due to wrong configuration, frames dropped due to VLAN filtering, and so on.

As an example, a particular event that is of interest during debug, ANA_L2: STICKY.VLAN_IGNORE_STICKY can be configured to be the global event 0 in ANA_L2:STICKY_MASK.VLAN_IGNORE_STICKY_MASK(0).

The following figure shows the configuration and status available for each port statistics counter.

Figure 4-53. Port Statistics Counters



Before using the statistic counters, they must be initialized to clear all counter values and sticky bits (ANA_AC:STAT_GLOBAL_CFG_PORT:STAT_RESET.RESET).

For each counter, the type of frames that are counted must be configured (ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_CNT_FRM_TYPE).

Note: Frames without any destination are seen by the port statistics block as aborted.

Each counter must be configured, if bytes or frames are counted (ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_CNT_BYTE).

It is possible to limit counting to certain priorities (ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_PRIO_MASK).

Counter events that can trigger counting can be selected among the global events (ANA_AC:STAT_GLOBAL_CFG_PORT. STAT_GLOBAL_EVENT_MASK). There is one common event mask for each of the four counter sets.

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_PORT. STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_PORT.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the LSB part must be written first, followed by the MSB part.

The following pseudo code shows the port statistics functionality.

```
for (port_counter_idx = 0; port_counter_idx < 8; port_counter_idx++) {
  if
(STAT_CNT_CFG_PORT[frame.igr_port].STAT_CFG[port_counter_idx].CFG_PRIO_MASK[frame.pr
io]) {
    count = FALSE;
    global_event_mask = STAT_GLOBAL_EVENT_MASK[port_counter_idx].GLOBAL_EVENT_MASK;
    event_match =(STAT_GLOBAL_EVENT_MASK[port_counter_idx].GLOBAL_EVENT_MASK &
                  frame.event_mask);
    switch
(STAT_CNT_CFG_PORT[frame.igr_port].STAT_CFG[port_counter_idx].CFG_CNT_FRM_TYPE) {
    case 0x0:
      if (!frame.error && !event_match) count = 1;
      break;
```

```
    case 0x1:
      if (!frame.error && event_match) count = 1;
      break;
    case 0x2:
      if (frame.error && event_match) count = 1;
      break;
    case 0x3:
      if (event_match) count = 1;
      break;
    case 0x4:
      if (frame.error && !event_match) count = 1;
      break;
    case 0x5:
      if (frame.error) count = 1;
      break;
    }
    if (count) {
      if
(STAT_CNT_CFG_PORT[frame.igr_port].STAT_CFG[port_counter_idx].CFG_CNT_BYTE) {
        STAT_xSB_CNT[port_counter_idx] += frame.bytes;
      } else {
        STAT_xSB_CNT[port_counter_idx]++;
      }
    }
  }
}
```

Example: To set up port counter 3 to count bytes filtered by VLAN ingress filter, the following configuration is required.

- Enable VLAN drop events as global event 3:

  Set ANA_L3:L3_STICKY_MASK:VLAN_MSTP_STICKY_MASK[3].VLAN_IGR_FILTER_STICKY_MASK = 1.
- Initialize counters:

  Set ANA_AC:STAT_GLOBAL_CFG_PORT:STAT_RESET.RESET = 1.
- Set up port counters. For each active Ethernet port:

  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[3].CFG_CNT_FRM_TYPE = 3.

  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[3].CFG_CNT_BYTE = 1.

  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[3].CFG_PRIO_MASK = 0xFF.
- Enable global event 3 as trigger:

  Set ANA_AC:STAT_GLOBAL_CFG_PORT[3]. STAT_GLOBAL_EVENT_MASK = 8.

Example: To set up port counter 0 to count frames dropped by port policer, the following configuration is required.

- Set up port counters. For each active Ethernet port:

  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[0].CFG_CNT_FRM_TYPE = 3.

  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[0].CFG_CNT_BYTE = 0.

  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[0].CFG_PRIO_MASK = 0xFF.
- Enable global events 4 to 11 as trigger:

  Set ANA_AC:STAT_GLOBAL_CFG_PORT[0]. STAT_GLOBAL_EVENT_MASK = 0xFF0.

4.22.3.2 Priority Statistics

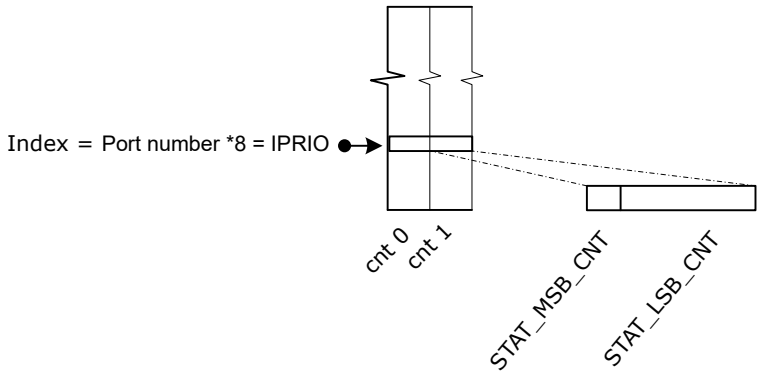
The following table lists the applicable priority statistics registers.

Table 4-177. Priority Statistics Registers

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_QUEUE: STAT_GLOBAL_CFG	Configures counting of bytes or frames per priority (= port × 8 + iprio).	2
ANA_AC:STAT_GLOBAL_CFG_QUEUE: STAT_GLOBAL_EVENT_MASK	Configures global event mask per priority (= port × 8 + iprio).	2
ANA_AC:STAT_CNT_CFG_QUEUE.STAT_LSB_CNT	Least significant 32 bits of counters per priority.	per priority × 2
ANA_AC:STAT_CNT_CFG_QUEUE.STAT_MSB_CNT	Most significant 8 bits of counters per priority.	per priority × 2

The following figure shows the configuration and status available for each priority statistics counter.

Figure 4-54. Priority Statistics



Each counter type must be configured to count either bytes or frames (ANA_AC:STAT_GLOBAL_CFG_QUEUE: STAT_GLOBAL_CFG.CFG_CNT_BYTE).

ACL events that can trigger counting can be selected among the following events:

- Count traffic applicable for priority policer, but not discarded
- Count traffic discarded by priority policer

There is one common event mask for each of the two priority counter types (ANA_AC:STAT_GLOBAL_CFG_QUEUE:STAT_GLOBAL_EVENT_MASK).

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_QUEUE.STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_QUEUE.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register, when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

4.22.3.3 Broadcast, Unicast, Multicast (BUM) Policer Statistics

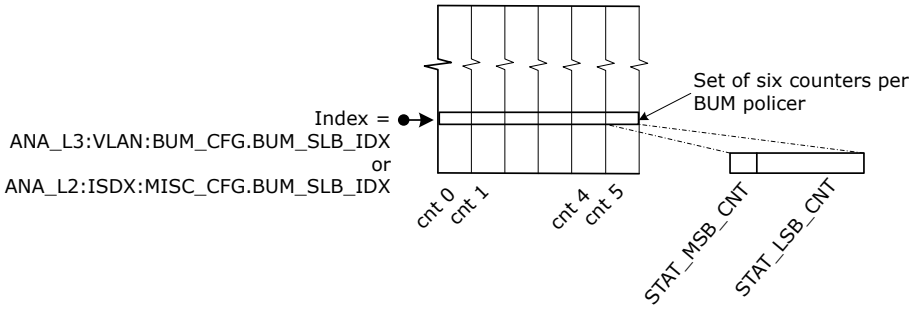
The following table lists the applicable BUM policer statistics registers.

Table 4-178. BUM Policer Statistics Registers

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_BUM: STAT_GLOBAL_CFG	Configures counting bytes or frames per BUM policer index.	2
ANA_AC:STAT_GLOBAL_CFG_BUM: STAT_GLOBAL_EVENT_MASK	Configures global event mask per BUM policer index.	2
ANA_AC:STAT_CNT_CFG_BUM.STAT_LSB_CNT	Least significant 32 bits of counters per BUM policer index.	per BUM policer index × 2
ANA_AC:STAT_CNT_CFG_BUM.STAT_MSB_CNT	Most significant 8 bits of counters per BUM policer index.	per BUM policer index × 2

The following figure shows the configuration and status available for each BUM policer.

Figure 4-55. BUM Policer Statistics



Each of the two counters in a counter set can be configured to which frames are counted and, whether bytes or frames are counted. This configuration is shared among all counter sets.

The BUM event mask supports counting the following frame types:

- Broadcast traffic discarded by BUM policer
- Multicast traffic discarded by BUM policer
- Unicast traffic discarded by BUM policer
- Broadcast traffic applicable for BUM policer, but not discarded
- Multicast traffic applicable for BUM policer, but not discarded
- Unicast traffic applicable for BUM policer, but not discarded
  The event mask is configured in ANA_AC:STAT_GLOBAL_CFG_BUM:STAT_GLOBAL_EVENT_MASK.

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_BUM.STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_ BUM.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register, when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

4.22.3.4 ACL Policer Statistics

The following table lists the applicable ACL policer statistics registers.

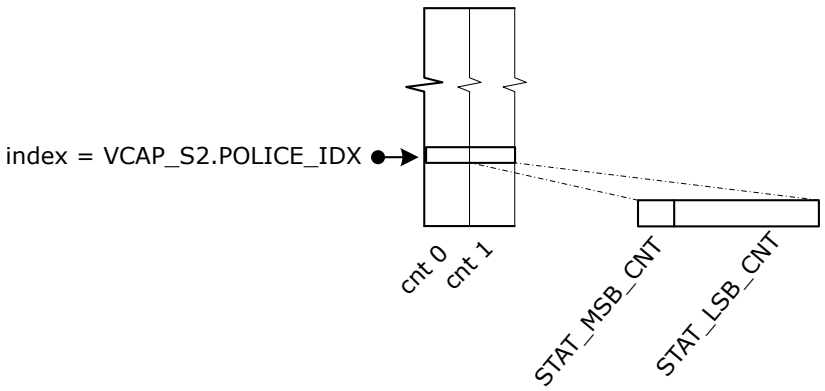
Table 4-179. ACL Policer Statistics Registers

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_ACL: STAT_GLOBAL_CFG	Configures counting of bytes or frames per ACL policer index.	2

.....continued		
Target::Register.Field	**Description**	**Replication**
ANA_AC:STAT_GLOBAL_CFG_ACL: STAT_GLOBAL_EVENT_MASK	Configures global event mask per ACL policer index.	2
ANA_AC:STAT_CNT_CFG_ACL.STAT_LSB_CNT	Least significant 32 bits of counters per ACL policer index.	per ACL policer index × 2
ANA_AC:STAT_CNT_CFG_ACL.STAT_MSB_CNT	Most significant 8 bits of counters per ACL policer index.	per ACL policer index × 2

The following figure shows the configuration and status available for each ACL policer statistics counter.

Figure 4-56. ACL Policer Statistics



Each counter type must be configured whether bytes or frames are counted (ANA_AC:STAT_GLOBAL_CFG_ACL: STAT_GLOBAL_CFG.CFG_CNT_BYTE).

ACL events that can trigger counting can be selected among the following events. ACL events that can trigger counting can be selected among the following events. If an ACL policer is triggered by an IS2 action, with IS_INNER_ACL = 1, it is termed as inner. Otherwise, it is termed as outer.

- Count CPU traffic applicable for outer ACL policer, but not discarded
- Count front port traffic applicable for outer ACL policer, but not discarded
- Count CPU traffic discarded by outer ACL policer
- Count front port traffic discarded by outer ACL policer
- Count CPU traffic applicable for inner ACL policer, but not discarded
- Count front port traffic applicable for inner ACL policer, but not discarded
- Count CPU traffic discarded by inner ACL policer
- Count front port traffic discarded by inner ACL policer

There is one common event mask for each of the two ACL policer counter types (ANA_AC:STAT_GLOBAL_CFG_ACL:STAT_GLOBAL_EVENT_MASK).

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_ACL.STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_ACL.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register, when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

4.22.3.5 Routing Statistics

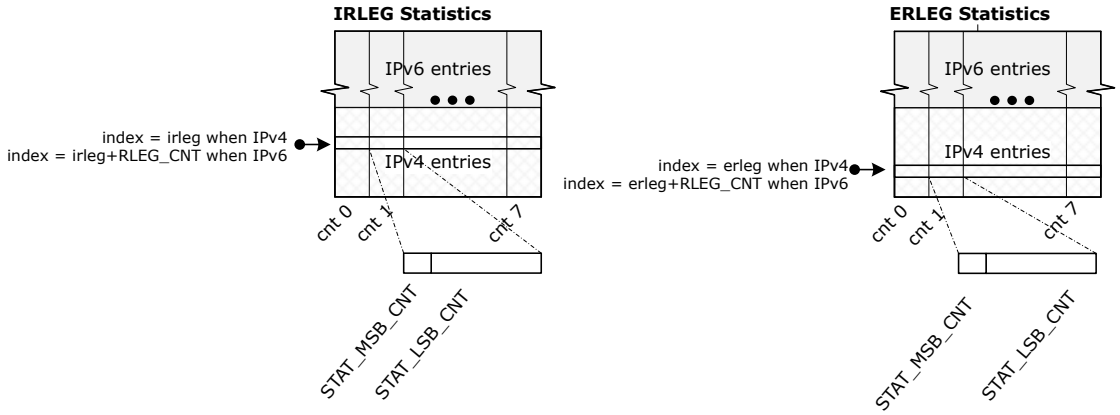
The following table lists the applicable routing statistics registers.

Table 4-180. Analyzer Routing Statistics Registers

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_IRLEG: STAT_GLOBAL_CFG	Configures counting of bytes or frames per ingress router leg counter.	8
ANA_AC:STAT_GLOBAL_CFG_IRLEG: STAT_GLOBAL_EVENT_MASK	Configures the global event mask per ingress router leg counter.	8
ANA_AC:STAT_CNT_CFG_IRLEG.STAT_LSB_CNT	Least significant 32 bits of ingress router leg counters.	Per RLEG per IP_version × 8
ANA_AC:STAT_CNT_CFG_IRLEG.STAT_MSB_CNT	Most significant 8 bits of ingress router leg counters.	Per RLEG per IP_version × 8
ANA_AC:STAT_GLOBAL_CFG_ERLEG: STAT_GLOBAL_CFG	Configures counting of bytes or frames per egress router leg counter.	8
ANA_AC:STAT_GLOBAL_CFG_ERLEG: STAT_GLOBAL_EVENT_MASK	Configures the global event mask per egress router leg counter.	8
ANA_AC:STAT_CNT_CFG_ERLEG.STAT_LSB_CNT	Least significant 32 bits of egress router leg counters.	Per RLEG per IP_version × 8
ANA_AC:STAT_CNT_CFG_ERLEG.STAT_MSB_CNT	Most significant 8 bits of egress router leg counters.	Per RLEG per IP_version × 8

The following figure shows the configuration and status available for ingress router leg and egress router leg statistics counter sets. For this device, the RLEG_CNT is 512.

Figure 4-57. Ingress and Egress Routing Statistics per Router Leg per IP Version



Each counter type must be configured to count either bytes or frames. (ANA_AC:STAT_GLOBAL_CFG_IRLEG: STAT_GLOBAL_CFG.CFG_CNT_BYTE and ANA_AC:STAT_GLOBAL_CFG_ERLEG: STAT_GLOBAL_CFG.CFG_CNT_BYTE)

Ingress router leg counter events that can trigger counting can be selected among the following events:

- acl_discarded - frame applicable for routing but discarded due to VCAP IS2 ingress discard action (VCAP IS2 lookup with action RT_DIS set).
- received IP unicast traffic - frame applicable for routing (hitting a router leg).
- received IP multicast traffic - frame applicable for routing (hitting a router leg).
- unicast_routed - frame is unicast routed.
- multicast_routed - frame is multicast routed

- ip_multicast_rpf_discarded - frame discarded due to failed Reverse Path Forwarding check (frame not received from configured interface)
- ip_ttl_discarded - frame discarded due to TTL < 2

There is one common event mask for each of the eight IRLEG counter types (ANA_AC:STAT_GLOBAL_CFG_IRLEG:STAT_GLOBAL_EVENT_MASK).

Egress router leg counter events that can trigger counting can be selected among the following events:

- acl_discarded - frame applicable for routing but discarded due to VCAP IS2 egress discard action (VCAP IS2 lookup with action RT_DIS set).
- unicast_routed traffic - frame is unicast routed.
- multicast_routed traffic - frame is multicast routed.
- ip_multicast_switched traffic - frame is bridged within ingress VLAN.
- ip_multicast_ttl_discarded - frame discarded due to TTL less than ERLEG configured TTL value.

There is a common event mask for each of the eight ERLEG counter types (ANA_AC:STAT_GLOBAL_CFG_ERLEG:STAT_GLOBAL_EVENT_MASK).

Each 40-bit counter consists of an MSB part (ANA_AC:STAT_CNT_CFG_IRLEG.STAT_MSB_CNT or ANA_AC:STAT_CNT_CFG_ERLEG.STAT_MSB_CNT) and an LSB part (ANA_AC:STAT_CNT_CFG_IRLEG.STAT_LSB_CNT or ANA_AC:STAT_CNT_CFG_ERLEG.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register, when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first.

4.22.4 Analyzer sFlow Sampling

This section describes sFlow sampling. The applicable registers are listed in the following table.

Table 4-181. Analyzer sFlow Registers

Target::Register.Field	Description	Replication
ANA_AC::PS_COMMON_CFG.SFLOW_ENA	Controls sFlow operation.	None
ANA_AC::SFLOW_CFG.SFLOW_CPU_QU	Configures CPU extraction queue for sFlow sampled frames.	None
ANA_AC::SFLOW_CTRL	Configures sFlow samplers (rate and type).	Per port
ANA_AC::SFLOW_CNT	Current sFlow sampler count values.	Per port
ANA_AC:PS_STICKY:STICKY	Various sticky debug events.	None
ANA_AC:PS_STICKY_MASK:STICKY_MASK	Mask to allow debug events to be counted in port stat block.	None

sFlow is a standard for monitoring high-speed switched networks through statistical sampling of incoming and outgoing frames. Each port in the device can be set up as an sFlow agent, monitoring the particular link and generating the sFlow data. If a frame is sFlow sampled, it is copied to the sFlow CPU extraction queue (ANA_AC::SFLOW_CFG.SFLOW_CPU_QU).

An sFlow agent is configured through ANA_AC::SFLOW_CFG with the following options:

- SFLOW_SAMPLE_RATE specifies the probability that the sampler copies a frame to the CPU. Each frame being candidate for the sampler has the same probability of being sampled. The probability is set in steps of 1/32767.
- SFLOW_DIR_SEL(0) enables incoming frames on the port as candidates for the sampler.
- SFLOW_DIR_SEL(1) enables outgoing frames on the port as candidates for the sampler.

Rx and Tx sampling can be enabled independently. If both are enabled, all incoming and outgoing traffic on the port is subject to the statistical sampling given by the rate in SFLOW_SAMPLE_RATE.

The CPU can extract information about a frame's sFlow handling in IFH field FWD.SFLOW_ID including sampling direction and sampler identify. Note that a given frame can only be sampled by one sFlow sampler. Although, it might be a candidate for sampling by multiple samplers.

A variety of sticky events allows debug of the sFlow setup. These events can furthermore be counted in the port statistics block by enabling the corresponding *_MASK. The following events are reported (in ANA_AC:PS_STICKY:STICKY):

- sFlow candidate is found (ANA_AC:PS_STICKY:STICKY.SFLOW_CAND_STICKY).
- sFlow source sample is sent to CPU (ANA_AC:PS_STICKY:STICKY.SFLOW_SRC_SAMPLE_STICKY).
- sFlow destination sample is sent to CPU (ANA_AC:PS_STICKY:STICKY.SFLOW_DST_SAMPLE_STICKY).

Current sample count and sample candidate count numbers are available (ANA_AC::SFLOW_CNT.SFLOW_SAMPLE_CNT and ANA_AC::SFLOW_CNT.SFLOW_FRAME_FWD_CNT).

These events can all be counted in the statistics block by enabling the corresponding *STICKY_MASK found in register group ANA_AC:PS_STICKY_MASK:STICKY_MASK[x]. (ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_CAND_STICKY_MASK, ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_DST_SAMPLE_STICKY_MASK, ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_SRC_SAMPLE_STICKY_MASK and ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_SAMPLE_STICKY_MASK).

4.22.5 Mirroring

This section describes how to configure mirroring. The following table lists the applicable registers.

Table 4-182. ANA_AC Mirror Registers

Target::Register.Field	Description	Replication
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_DIRECTION	Configures whether to probe ingress traffic, egress traffic, or both.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_MAC_MODE	Configures MAC address filtering, that is, only traffic to and/or from hosts known into MAC table with the mirror bit set are mirrored.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_VLAN_MODE	Configures VLAN filtering. That is, only traffic with a specific VID or with VLAN entry mirror bit set is mirrored.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_VID	Configures VID, when PROBE_VLAN_MODE is set to probe VID.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_CPU_SET	Configures mirroring of traffic to specific CPU ports.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_PHYS_RX_PORT	Configures, if Rx mirroring mirrors physical or masqueraded port.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_RX_CPU_AND_VD	Configures Rx mirror mask for CPU ports and VD ports.	3
ANA_AC:MIRROR_PROBE:PROBE _PORT_CFG	Configures mirroring of the ingress ports for PROBE_DIRECTION = RX_MIRROR and/or the set of egress ports for PROBE_DIRECTION = TX_MIRROR.	3
QFWD:SYSTEM:FRAME_COPY_CFG[9-11]	Configures forwarding options per probe.	3

To debug network problems, selected traffic can be mirrored using configurable probes, to a mirror port where a frame analyzer can be attached to analyze the frame flow.

The device has three independent mirroring probes. Each probe can be configured with a separate set of filtering conditions that must be fulfilled before traffic is mirrored. If multiple filtering conditions are enabled for a given probe they must all be fulfilled before mirroring occurs. For example, mirror probe 1 can be set up to only Rx mirror traffic from a given host in a specific VLAN, whereas probe 2 can be set up to mirror frames matching criteria in VCAP IS2. If a frame is mirrored by more than one mirror probe, the highest numbered probe is selected. This implies that only one mirror copy is made per frame, even if multiple probes are active.

The following traffic mirror conditions can be configured per probe:

- All frames received on a given port, also known as ingress mirroring (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(1) = 1 and ports to be RX mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_PORT_CFG)
- All frames transmitted on a given port, also known as egress mirroring (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(0) = 1 and ports to be TX mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_PORT_CFG)
- All frames sent to a specific CPU port (may be useful for software debugging) (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(0) = 1 and CPU ports to be mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_CPU_SET)
- All frames classified to a specific VID (enabled in ANA_AC:MIRROR_PROBE[x]. PROBE_CFG. PROBE_VLAN_MODE == 2 and VID set in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_VID)
- All frames classified to VLANs with VLAN -> VLAN_MIRROR_ENA set (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_VLAN_MODE == 1).
- All frames received from a known station with MAC -> MIRROR set (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_MAC_MODE(1) = 1).
- All frames send to a known station with MAC -> MIRROR set (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG. PROBE_MAC_MODE(0) = 1). This can also be flooded traffic (enabled via ANA_L2::FWD_CFG.FLOOD_MIRROR_ENA).
- Frames selected through configured VCAP entries (enabled by VCAP IS2 action MIRROR_PROBE). For such frames, the only other applicable mirror criteria is PROBE_DIRECTION.
- All frames received from CPU, also known as CPU ingress mirroring (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(1) = 1 and CPU ports to be RX mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_RX_CPU_AND_VD).

The mirror port configured per probe (QFWD:SYSTEM:FRAME_COPY_CFG[8-11]) can be any port on the device, including the CPU.

Rx-mirrored traffic sent to the mirror port is sent as received optionally with an additional Q-tag (enabled in REW::MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG).

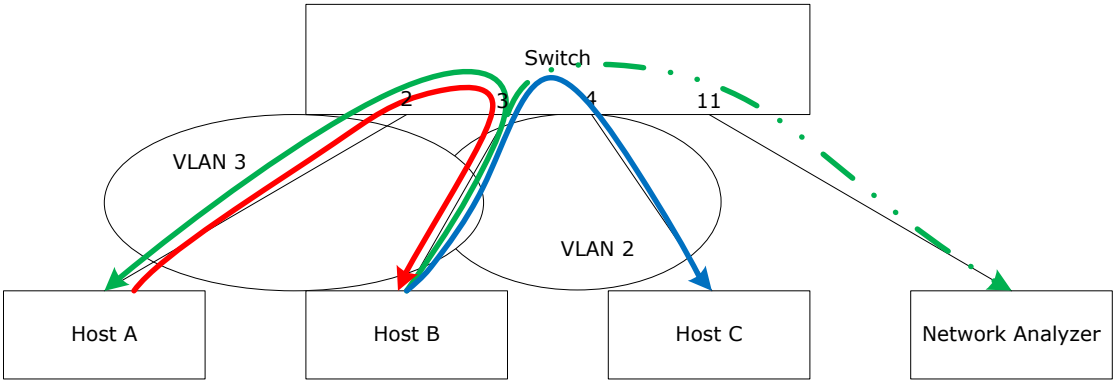
Tx-mirrored traffic sent to the mirror port is modified according to a configured Tx port (REW:COMMON:MIRROR_PROBE_CFG.MIRROR_TX_PORT) optionally with an additional Q-tag (enabled in REW::MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG).

For information about possible rewriter options for mirrored frames, see 4.28.16 Mirror Frames.

Example: Ingress port mirroring in specific VLAN

All traffic only in VLAN 3 from host B on port 3 (the probe port) is mirrored to port 11 (the mirror port) using probe 1, as shown in the following figure.

Figure 4-58. Ingress Mirroring in Specific VLAN



The following configuration is required:

- Enable RX mirroring on port 3:
  ANA_AC:MIRROR_PROBE[1].PROBE_CFG.PROBE_DIRECTION = 2 (= Rx only).

  ANA_AC:MIRROR_PROBE[1].PROBE_PORT_CFG = 0x8 (port 3).
- Enable VID filtering:
  ANA_AC:MIRROR_PROBE[1].PROBE_CFG.PROBE_VLAN_MODE = 2 and

  ANA_AC:MIRROR_PROBE[1].PROBE_CFG. PROBE_VID = 3.

4.23 Leaky Bucket for SDLB

The following section discusses leaky bucket for service dual leaky bucket (SDLB).

4.23.1 LB Sets and LB Groups

Leaky buckets are used to police or shape traffic. The buckets contain a number of tokens, each representing one byte of traffic. Tokens are added to the buckets with a configurable rate, and each bucket is allowed to contain a configurable maximum number of tokens, the bucket threshold. When frames are policed or shaped, then tokens are subtracted from the bucket. If the bucket has less than 0 tokens at the beginning of a frame, then the bucket is "closed" and the frame is dropped or held back.

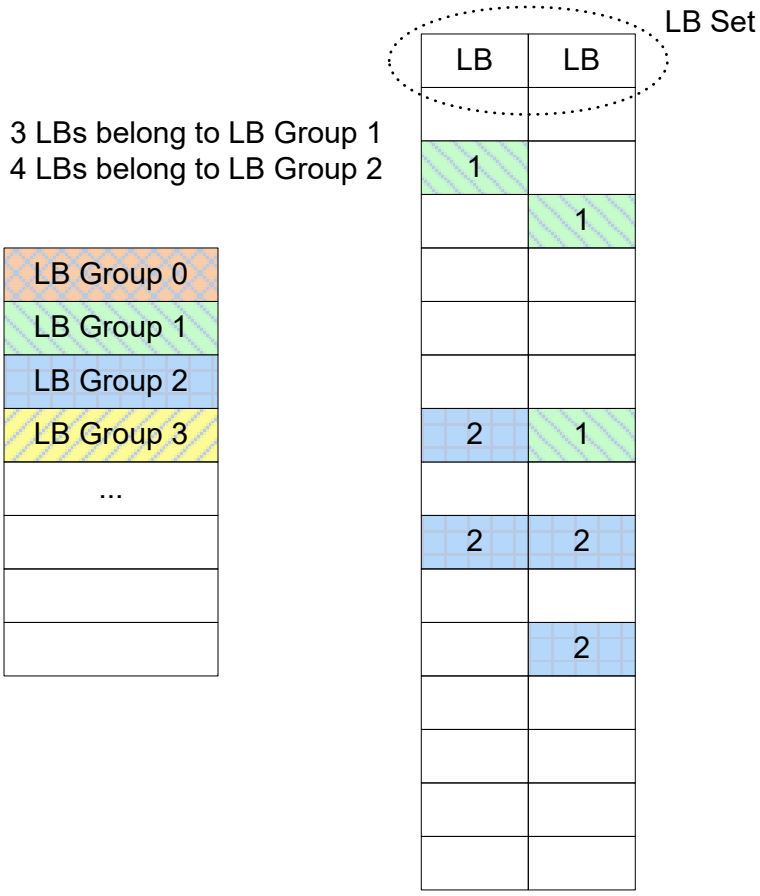
Within each LB block, LBs are grouped into "LB Sets" and "LB Groups".

An LB Set typically consists of one or two LBs, for example, two leaky buckets for supporting MEF DLB policing/ shaping. LBs in an LB Set are located at the same address in the involved memories.

An LB Group consists of LBs with rates within some rate interval, see examples in section 4.23.2.2 LB Group Configuration. The LBs in an LB Group are grouped together using a linked list.

The following figure shows two configured LB Groups, such as, LB Group 1 and LB Group 2. LB Group 1 contains three LBs and LB Group 2 contain four LBs. Each LB Set contains two LBs. LBs within an LB Set may belong to different LB Groups.

Figure 4-59. LB Groups



4.23.2 Rate and Threshold Configuration

The LBs within an LB Group are periodically updated, each time adding tokens to the bucket. This controls the rate of the bucket. One LB is updated for every 2 clock cycles.

The following parameters control the rate at which tokens are added to each bucket.

Table 4-183. Rate Configuration Parameters

Parameter	Description	Replication
pup_tokens	Number of tokens to add to a bucket each time it is subject to a periodic update (PUP). Special encoding: PUP_TOKENS = max value => bucket permanently open.	Per LB
pup_interval	Time between periodic update of LBs in LB Group. Unit: 1 clock cycle	Per LB Group
FRM_RATE_ENA	Enable frame rate mode, that is, count frames not bytes.	Per LB Set
FRM_RATE_TOKENS	Number of tokens to subtract from a bucket on SoF events for LBs with FRM_RATE_ENA = 1.	Per LB Group

.....continued		
Parameter	**Description**	**Replication**
frm_adj	Number of tokens to subtract from bucket, when SoF is encountered.	Per LB Set or per LB Group
FRM_ADJ_ENA	If set, FRM_ADJ from LB Group subtracted from bucket, when SoF is encountered.	Per LB Set

The rate at which tokens are added to an LB is calculated as follows (tokens per second):

```
Rate = PUP_TOKENS x clk_frequency / PUP_INTERVAL
```

Each LB is allowed to contain a maximum number of tokens, the threshold. The threshold is configurable using the following parameters.

Table 4-184. Threshold Configuration Parameters

Parameter	Description	Replication
thres	Threshold value is used to specify the maximum number of tokens the bucket is allowed to contain. The unit for THRES is $2^{THRES\_OFFSET - THRES\_SHIFT}$ tokens. THRES_OFFSET is a fixed value for each LB block. THRES_SHIFT is configured per LB Group in LBGRP_TBL:LBGRP_MISC.THRES_SHIFT.	Per LB
thres_SHIFT	See description for THRES.	Per LB Group

For a given product, software—at startup—configures a set of LB Groups that supports the ranges of rates required for the product, and as LBs get configured they are placed in one of the configured LB Groups.

4.23.2.1 LB Group Configuration Rules

For leaky bucket blocks that contain DLBs, both LBs of the DLB are member of the same LB Group. For leaky bucket blocks that contain SLBs, the members of the LB Groups are individual LBs. The pointers related to an LB Group's linked list of LBs, can thus either be SLB pointers or DLB pointers, depending on the leaky bucket block. "XLB" is used as a generic term for SLB/DLB.

When configuring LB Groups, these rules must be observed:

1. The number of XLBs in a group must not exceed PUP_INTERVAL/4 - 1.
2. It must be possible to update all XLBs within the configured PUP_INTERVALs, that is,

$$\sum_{lbgrp\_idx=0}^{max\_lbgrp\_idx} (xlb\_cnt_{lbgrp\_idx} / (PUP\_INTERVAL_{lbgrp\_idx} / 2)) < 1$$

   Where, $xlb\_cnt_{lbgrp\_idx}$ is the number of XLBs assigned to the LB Group.

   If two SLBs within an LB Set belong to the same LB Group and are consecutive in the LB Group's linked list of SLBs (see, 4.23.4 Managing LB Group Membership and 4.23.4.6 Ordering SLBs within an LB Group), then these two SLBs need only to be counted once in $xlb\_cnt_{lbgrp\_idx}$ as they are updated concurrently.

   Note that if the preceding sum gets close to 1, then burst size may fluctuate slightly, since PUP may happen irregularly for higher values of lbgrp_idx. To reduce such fluctuation, rule III must be observed.
3. LB Groups must be ordered such that rising lbgrp_idx corresponds to rising PUP_INTERVAL. In case of matching intervals, the groups must be ordered such that rising lbgrp_idx corresponds to decreasing THRES_SHIFT. Groups with identical PUP_INTERVAL and THRES_SHIFT is a valid configuration.

4. Each XLB must be placed in the LB Group with the highest PUP_INTERVAL, that spans a rate interval covering the rates of the LBs within the XLB.

The preceding rules are referred to as "Rule I/II/III/IV".

To avoid violating Rule II, LB Groups with small PUP_INTERVAL and many XLBs must be avoided.

4.23.2.2 LB Group Configuration

Selecting an appropriate LB Group setup is a manual process where the following parameters must be considered:

- Number of XLBs
- Rates required
- Rate granularity and/or precision
- Min threshold required
- Bandwidth and bandwidth over-subscription ratio

The following examples assumes a clock cycle period of 4 ns (corresponding to a clock frequency of 250 MHz), that is, one LB is subject to PUP every 8 ns.

System bandwidth is assumed to be 100 Gbps.

4.23.2.2.1 Service Policing Example

The following example shows how LB Groups can be configured to fulfill the following requirements:

- 8 K XLBs
- 0-10 Mbps: 64 Kbps granularity, or better
- > 10 Mbps: Rate accuracy of 1%, or better
- Min threshold supported: 4096 for all rates

Given relatively a high number of XLBs, the main challenge for this example is to configure LB Groups with PUP_INTERVAL high enough to avoid violation of Rule II for any realistic configuration.

Table 4-185. LB Group Configuration - Service Policing

LB Group			LB Rates		LB Group Rate Interval	
lbgrp_idx	PUP_INTERVAL (Clock cycles)	Max XLBs in LB Group (PUP_INTERVAL/4) (LBs)	Min Rate & Granularity (PUP_TOKENS=1) (bps)	Max Rate (PUP_TOKENS=4094) (bps)	Min Rate (= Max Rate of prev. LB Group) (bps)	Max Rate (bps)
			1	**4094**		
4	250.000	62.500	8.000	32.752.000	8.000	32.752.000
3	80.000	20.000	25.000	102.350.000	32.752.000	102.350.000
2	8.000	2.000	250.000	1.023.500.000	102.350.000	1.023.500.000
1	800	200	2.500.000	10.235.000.000	1.023.500.000	10.235.000.000
0	400	100	5.000.000	20.470.000.000	10.235.000.000	20.470.000.000

In the preceding example, it is possible for an end-user to attempt to configure an unsupported LB setup. For example, if the following is configured,

- 200 XLBs of rate 1,030,000,000 bps (LB Group 1), threshold = 4096 bytes
- 2000 XLBs of rate 103,000,000 (LB Group 2) , threshold = 4096 bytes

then Rule II will be violated and SW reports an "out of resources" error.

However, such configuration represents an over-subscription of the system bandwidth of > 300% and may thus be considered unrealistic or unsupported. If such configuration *is* deemed realistic, then more groups can be added. The following table lists the LB Group configuration with more LB Groups.

Table 4-186. LB Group Configuration with More LB Groups

LB Group			LB Rates		LB Group Rate Interval	
lbgrp_idx	PUP_INTERVAL (Clock cycles)	Max XLBs in LB Group (PUP_INTERVAL/4) (LBs)	Min Rate & Granularity (PUP_TOKENS = 1) (bps)	Max Rate (PUP_TOKENS = 4094) (bps)	Min Rate (= Max Rate of prev. LB Group) (bps)	Max Rate (bps)
			1	**4094**		
6	250.000	62.500	8.000	32.752.000	8.000	32.752.000
5	80.000	20.000	25.000	102.350.000	32.752.000	102.350.000
4	40.000	10.000	50.000	204.700.000	102.350.000	204.700.000
3	8.000	2.000	250.000	1.023.500.000	204.700.000	1.023.500.000
2	4.000	1.000	500.000	2.047.000.000	1.023.500.000	2.047.000.000
1	800	200	2.500.000	10.235.000.000	2.047.000.000	10.235.000.000
0	400	100	5.000.000	20.470.000.000	10.235.000.000	20.470.000.000

Using fhe LB Group configuration in the preceding table, the 200 XLBs of rate 1,030,000,000 bps now belong to LB Group 2 and contribute with 200/2000 = 0.1 to the sum of Rule II and similarly the 2000 XLBs of rate 103,000,000 bps belong to LB Group 4 and also contribute with 0.1, thus leaving 80% of the PUP bandwidth for other buckets (further, oversubscribing the system bandwidth).

As mentioned in Rule IV, XLBs must be placed in the XLB Group with the highest PUP_INTERVAL, that spans a rate interval covering the XLB's rate. For example, if you create an XLB with a rate set to 1,030,000,000 bps, then this XLB must be placed in LB Group 2, not LB Group 1 or 0. If there is no room for more XLBs in LB Group 2 (due to Rule I), then an error is reported.

As a refinement to this algorithm, the LB's threshold may also be considered. Thus in above example if width of PUP_TOKENS is 13 and XLB's threshold is set to >= 8000, then the XLB may be placed in LB Group 3 and PUP_TOKENS for the XLB set to 8000. If you reconfigure the threshold to a value smaller than 8000 later, then the XLB must be moved to LB Group 2.

For more fore information about the recommended LB Group selection algorithm, see 4.23.2.3 LB Group Selection Algorithm.

4.23.2.3 LB Group Selection Algorithm

The following pseudo code shows the recommended algorithm for selecting LB Group for an SLB/DLB.

```
// Select LB Group for SLB
int SlbLbGroupSelect(
                     int lb_ir,  // Information rate, bps
                     int lb_tres // Threshold, bytes
                     )
{
  return GetBestLbGrpForLb(lb_ir, lb_thres);
} // SlbLbGroupSelect
```

```
// Select LB Group for LBs in DLB
//
// lb_ir_max=0 => IRmax disabled.
int DlbLbGroupSelect(
                     int  lb_ir0,        // CIR, bps
                     int  lb_ir_max0,    // CIRmax, bps
                     int  lb_tres0,      // CBS, bytes
                     int  lb_ir1,        // EIR, bps
                     int  lb_ir_max1,    // EIRmax, bps
                     int  lb_tres1,      // EBS, bytes
                     BOOL coupling_flag, // MEF 10.2 CF
                     )
{
  int lbgrp_idx0, lbgrp_idx1;

  if (INH_TOKENS_WID == 0) {
    lbgrp_idx0 = GetBestLbGrpForLb(lb_ir0, lb_thres0);
    if (coupling_flag) {
      // MEF 10.2, i.e. no CIRmax/EIRmax
      lbgrp_idx1 = GetBestLbGrpForLb(lb_ir0 + lb_ir1, lb_thres1);
    } else {
      lbgrp_idx1 = GetBestLbGrpForLb(lb_ir1, lb_thres1);
    }
  } else {
    // MEF 10.3
    int lb_rate0,   lb_rate1;

    lb_rate0 = Max(lb_ir0, lb_ir_max0);
    lb_rate1 = Max(lb_ir1, lb_ir_max1);

    lbgrp_idx0 = GetBestLbGrpForLb(lb_rate0, lb_thres0);
    lbgrp_idx1 = GetBestLbGrpForLb(lb_rate1, lb_thres1);
  }

  // Use the fastest of the two LB Groups
  return Min(lbgrp_idx0, lbgrp_idx1);
} // DlbLbGroupSelect

int GetBestLbGrpForLb(
                      int lb_rate, // Rate, bps
                      int lb_thres // Threshold, bytes
                      )
{
  int lbgrp_idx;

  // For large values of lb_thres, it is desirable not to
  // add the full lb_thres value for each PUP (though the resulting
  // threshold would be correct).
  int max_tokens_per_pup;
  if (lb_thres > 65536) {
    max_tokens_per_pup = lb_thres / 8;
  } else if (lb_thres > 32768) {
    max_tokens_per_pup = lb_thres / 4;
  } else {
    max_tokens_per_pup = lb_thres;
  }

  // Choose the LB Group with the longest PUP_INTERVAL, that can
  // support lb_rate while adding at most lb_thres tokens per PUP.
  for (lbgrp_idx = LBGRP_CNT-1; lbgrp_idx > 0; lbgrp_idx--) {
    int lbgrp_rate_largest =
      max_tokens_per_pup * 8 * clk_frequency /
        csr.lbgrp_tbl[lbgrp_idx].pup_interval;

    if (lb_rate < lbgrp_rate_largest) return lbgrp_idx;
  }
} // GetBestLbGrpForLb
```

4.23.3 Inheritance

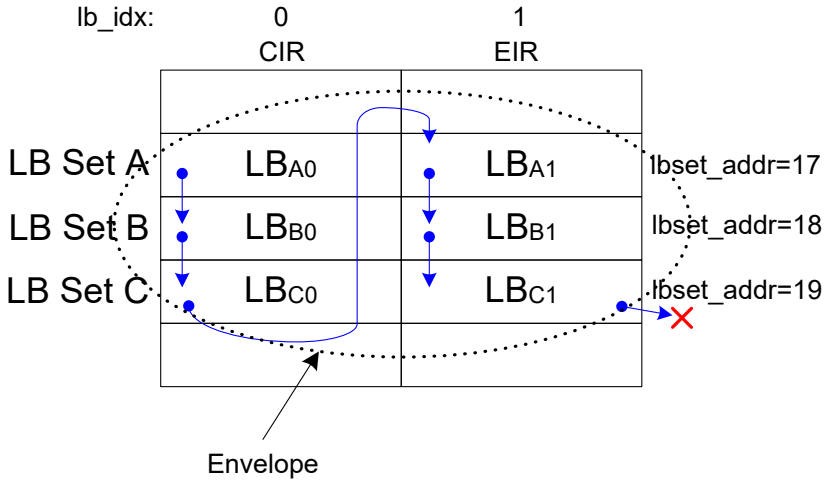
In order to support MEF 10.2 and MEF 10.3 bandwidth profiles (that are, CF, Envelope, and Rank), an inheritor LB can be configured for each LB. The inheritor LB receives any tokens overflowing from the ancestor LB.

Each LB (the "ancestor LB") has at most one inheritor LB. An inheritor LB may have multiple ancestor LBs, though, normally it has at most one ancestor LB.

LBs within a given LB Set can only have inheritor LBs in current LB Set or in *one* other LB Set.

The following figure shows an inheritance example. The arrows show the inheritance order. LB_{A0} has no ancestors and LB_{C1} has no inheritors. Any tokens overflowing in LB_{C1} are lost.

Figure 4-60. MEF 10.3 Inheritance Example



The following table lists the parameters used to configure inheritance.

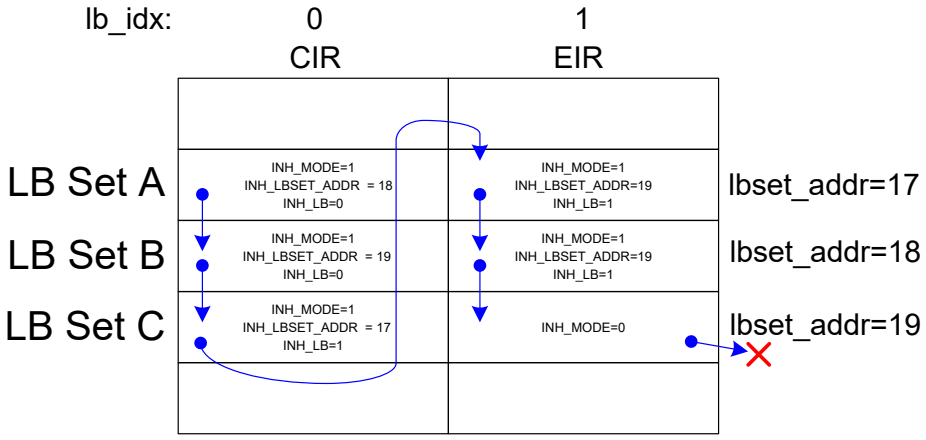
Table 4-187. Inheritance Configuration Parameters

Parameter	Width (bits)	Description	Replication
inh_mode	2	Inheritance mode 0: No inheritor, that is, any excess tokens are lost. 1: Spill over into an LB pointed to by INH_LBSET_ADDR and INH_LB. INH_LB is only applicable, if an LB Set has two or more LBs. 2: Spill over into inheritor next LB in the current LB Set. 3: Reserved	Per LB
inh_lb	1	Index of inheritor LB within inheritor LB Set. Only applicable, if each LB Set contains two or more LBs.	Per LB
inh_lbset_addr		Address of the inheritor LB Set.	Per LB Set
pup_tokens_max	13	Maximum number of tokens to add to bucket for each PUP, including inherited tokens. PUP_TOKENS_MAX must be >= PUP_TOKENS. This can be used to support MEF 10.3 $EIR_{max.}$	Per LB

As illustrated in the preceding figure, multiple levels of inheritance may be configured, that is, if the inheritor LB overflows, then this spillover may be inherited by another LB.

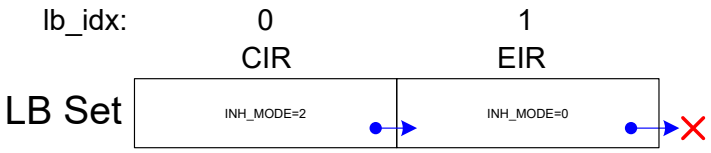
The following figure shows how to configure the inheritance of example as in the preceding figure.

Figure 4-61. MEF 10.3 Inheritance Example Configuration



If only MEF 10.2 bandwidth profiles are required, then the LB Set must be configured as shown in the following figure.

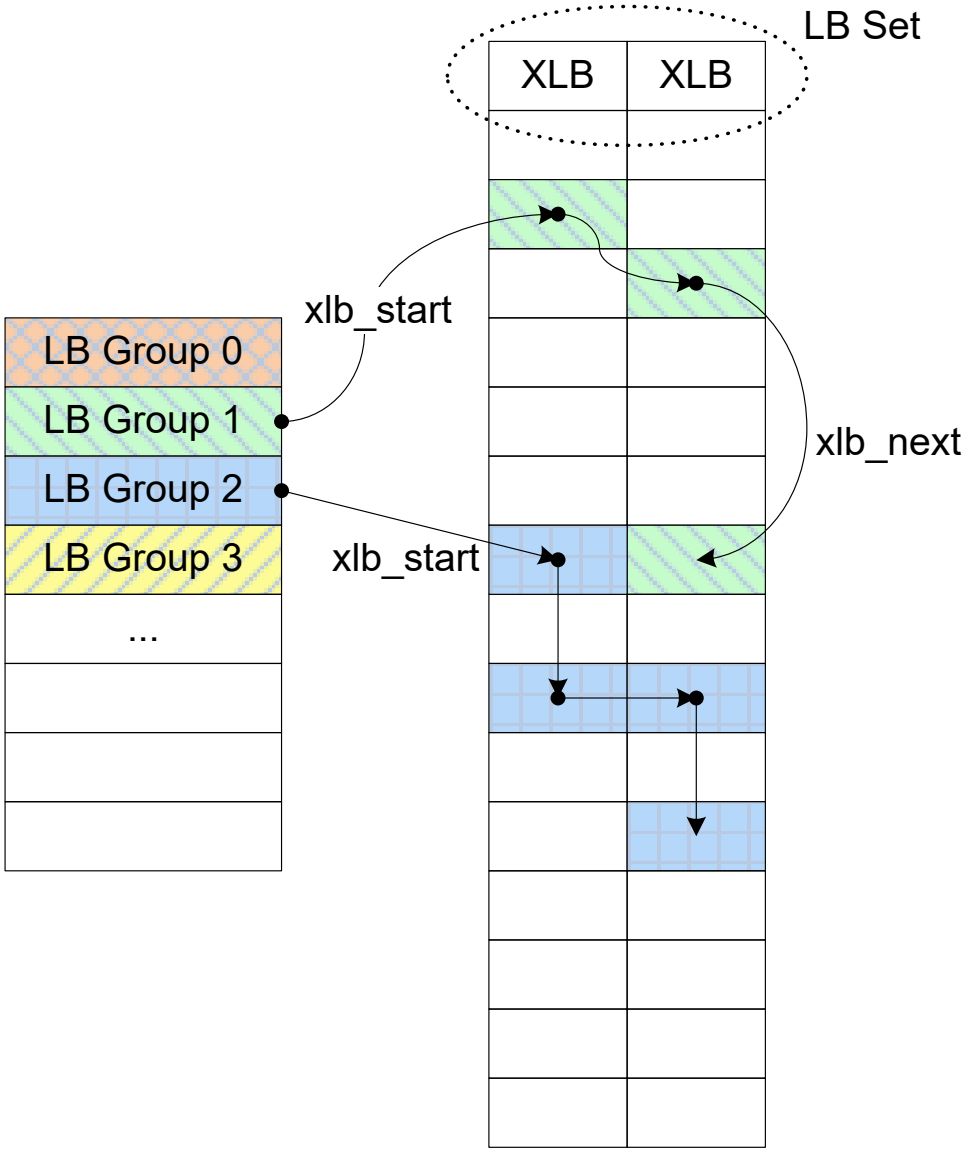
Figure 4-62. MEF 10.2 Inheritance Example Configuration



4.23.4 Managing LB Group Membership

The XLBs belonging to a given LB Group are linked together in a linked list, pointed to by the XLB_START pointer of the LB Group. The following figure shows the linked list of XLBs.

Figure 4-63. LB Groups' Linked list of XLBs



When adding or removing XLBs to and from an LB Group, the procedures described in the following sections must be used.

The related parameters are listed in the following table.

Table 4-188. LB List Related Parameters

Parameter	Width (bits)	Description	Replication
LBSET_START		Address of LB Set contaning first XLB of LB Group.	Per LB Group
Xlb_start		Index of first XLB of LB Group within an LB Set pointed to by LBSET_START. Only applicable, if each LB Set contains two or more XLBs.	Per LB Group

.....continued

Parameter	Width (bits)	Description	Replication
pup_lb_dt	12	Delta time between updating XLBs within an LB Set, that is, PUP_LB_DT specifies the number of clock cycles between each XLB update within the LB Set. It can be used to avoid many XLBs opening concurrently. It should be set to PUP_INTERVAL/<number of XLBs in LB Group>.	Per LB Group
PUP_ENA	1	Enable Periodic UPdate (PUP).	Per LB Group
lbset_next		LB Set containing next XLB in LB Group. To mark the end of LB Group's XLB list, set LBSET_NEXT and XLB_NEXT to point to current XLB.	Per XLB
Xlb_next		XLB - within the LB Set pointed by LBSET_NEXT, which navigates to the next XLB in the LB Group. Only applicable, if each LB Set contains two or more XLBs.	Per XLB
Xlb_next_tr_ena	1	Enable XLB pointer translation. Used when removing an XLB from an LB Group, see 4.23.4.4 Removing an XLB from an LB Group.	1
lbset_next_tr_org Xlb_next_tr_org lbset_next_tr_new Xlb_next_tr_new XLB_NEXT_TR_END		Parameters used for temporary translation of XLB pointer. Used when removing an XLB from an LB Group, see 4.23.4.4 Removing an XLB from an LB Group.	1

4.23.4.1 Adding XLB to Start of the LB Group

Use the following procedure to add an XLB - the "insertion XLB" - to an LB Group.

1. Set LBSET_NEXT and XLB_NEXT of the insertion XLB to point to the first XLB of the LB Group. If the LB Group is empty, then setup the insertion XLB to point to itself.
2. Change LBSET_START and XLB_START of the LB Group to point to the insertion XLB.
3. If the insertion XLB is the only XLB within the LB Group, set PUP_ENA = 1 for the LB Group.
4. If necessary, update the LB Group's PUP_LB_DT parameter.

4.23.4.2 Adding XLB to Middle of the LB Group

Use the following procedure to add an XLB - the "insertion XLB" - to an LB Group, between two XLBs in the LB Group's list of XLBs:

1. Set LBSET_NEXT and XLB_NEXT of the insertion XLB to point to the next XLB of the LB Group.
2. Set LBSET_NEXT and XLB_NEXT of the preceding XLB to point to the new XLB.

4.23.4.3 Adding an XLB to the End of the LB Group

Use the following procedure to add an XLB - the "insertion XLB" - to an LB Group, at the end of the LB Group's list of XLBs:

1. Set LBSET_NEXT and XLB_NEXT of the insertion XLB to point to itself.

2. Set LBSET_NEXT and XLB_NEXT of the preceding XLB to point to the new XLB.

4.23.4.4 Removing an XLB from an LB Group

Use the following procedure to remove an XLB - the "removal XLB" - from an LB Group:

If an LB Group contains only one XLB:

- Set PUP_ENA = 0 for the LB Group.

If an LB Group contains more than one XLBs:

1. Change the XLB_NEXT and LBSET_NEXT pointers of the preceding XLB to point to the XLB that follows the removal XLB. If the removal XLB is first in the LB Group's XLB list, then change the LB_START pointer of the LB Group.
2. Set LBSET_NEXT_TR_ORG and XLB_NEXT_TR_ORG to point to the removal XLB.
3. Set LBSET_NEXT_TR_NEW and XLB_NEXT_TR_NEW to point to the XLB that follows the removal XLB in the LB Group's XLB list. If the removal XLB is the last in the list, then set XLB_NEXT_TR_END to 1.
4. Set XLB_NEXT_TR_ENA = 1.
5. Set XLB_NEXT_TR_ENA = 0.
6. If necessary, update the LB Group's PUP_LB_DT parameter.

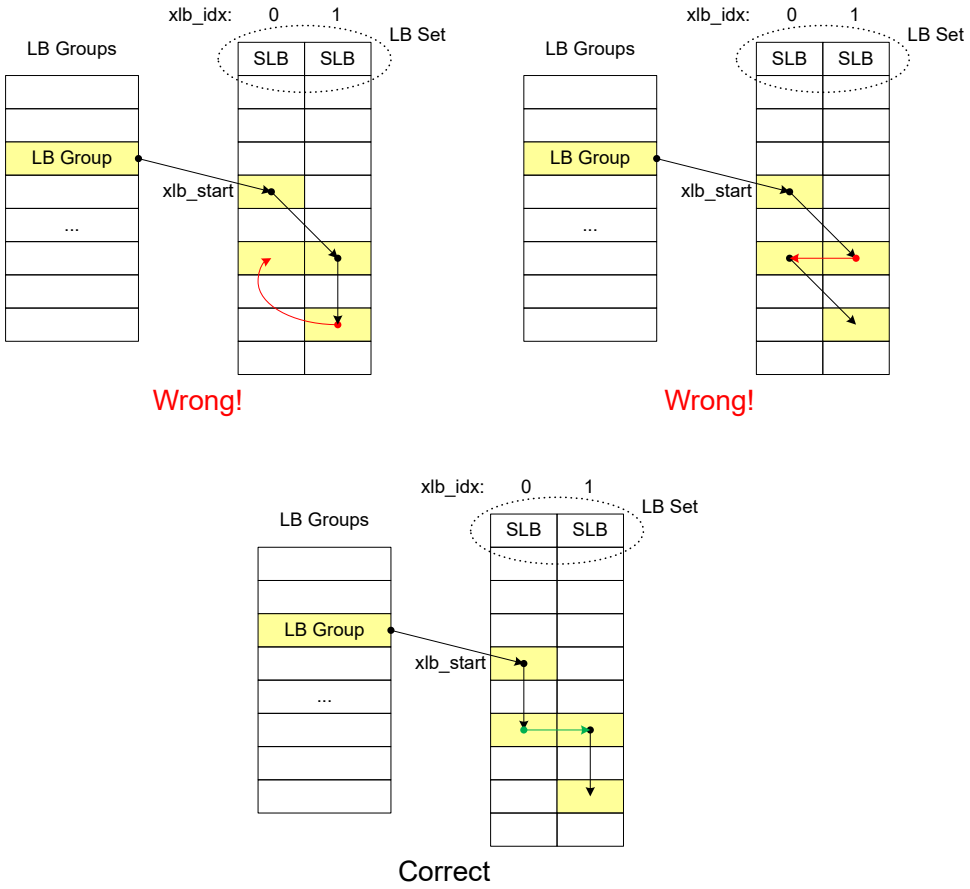
4.23.4.5 Moving an XLB between LB Groups

When the rate or threshold of the LBs of an XLB is changed, then it may be necessary to move the XLB from one LB Group to another LB Group. This is done by first following the procedure in 4.23.4.4 Removing an XLB from an LB Group followed by the procedure in 4.23.4.1 Adding XLB to Start of the LB Group.

4.23.4.6 Ordering SLBs within an LB Group

SLBs within an LB Group must be ordered consecutively, that is, by LB Set address and within each LB Set by SLB index.

Figure 4-64. Ordering SLBs within an LB Group



DLBs within an LB Group can be order arbitrarily.

4.24 VCAP ES2 Keys and Actions

VCAP ES2 is part of EACL and enables egress access control lists using VCAP functionality. This section provides detailed information about all available VCAP ES2 keys and actions.

For information about how to select which key to use and how the VCAP ES2 action is applied, see 4.25 Egress Access Control Lists.

4.24.1 VCAP ES2 Keys

VCAP ES2 supports a number of different keys that can be used for different purposes and frame types. The keys are of type X3, X6, or X12 depending on the number of words each key uses. The keys are grouped after size as shown in the following table.

Table 4-189. VCAP ES2 Keys and Sizes

Key Name	Number of Words	Key Type
IP4_VID	3 words	X3 type

.....continued		
Key Name	**Number of Words**	**Key Type**
MAC_ETYPE	6 words	X6 type
ARP		
IP4_TCP_UDP		
IP4_OTHER		
IP6_VID		
IP_7TUPLE	12 words	X12 type

The following table lists details for all VCAP ES2 keys and fields. When programming an entry in VCAP IS2, the associated key fields must be programmed in the listed order with the first field in the table starting at bit 0 in the entry. As an example, for a MAC_ETYPE entry, the first field X6_TYPE must be programmed at bits 3:0 of the entry, the second field FIRST must be programmed at bit 4, and so on.

Table 4-190. VCAP ES2 Key Details

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	IP6_VID	IP_7TUPLE
X6_TYPE	X6 type: 0: MAC_ETYPE. 1: ARP. 2: IP4_TCPUDP. 3: IP4_OTHER. 5: IP6_VID.	3		x	x	x	x	x	
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x	x	x	x	x
ACL_GRP_ID	ACL group identifier.	8	x	x	x	x	x	x	x
PROT_ACTIVE	Set if protection is active.	1	x	x	x	x	x	x	x
L2_MC	Set if frame's destination MAC address is a multicast address (bit 40 = 1).	1	x	x	x	x	x	x	x
L2_BC	Set if frame's destination MAC address is the broadcast address (FF-FF-FF-FF-FF-FF).	1	x	x	x	x	x	x	x
SERVICE_FRM	Set if classified ISDX > 0.	1	x	x	x	x	x	x	x
ISDX	Classified ISDX.	12	x	x	x	x	x	x	x
VLAN_TAGGED	Set if frame was received with a VLAN tag.	1	x	x	x	x	x	x	x
XVID	Classified VID.	13	x	x	x	x	x	x	x

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	IP6_VID	IP_7TUPLE
EGR_PORT_MASK_RNG	Selects content of EGR_PORT_MASK: 0: Physical/Logical egress port number 0-31. 1: Physical/Logical egress port number 32–63. 2: Physical/Logical egress port number 64. 3: Virtual Interface Number 0-31. 4: Virtual Interface Number 32-63. 5: Virtual Interface Number 64. 6: CPU queue Number 0-7. 7: Reserved.	3		x	x	x	x		x
EGR_PORT_MASK	See EGR_PORT_MASK_RNG.	32		x	x	x	x		x
IGR_PORT_SEL	Defines the content of IGR_PORT. 0: Physical/Logical port number. 1: ERLEG.	1		x	x	x	x		x
IGR_PORT	Ingress port number in case of L2 forwarded frame. ERLEG from IFH in case of L3 forwarded frame.	9		x	x	x	x		x
PCP	Classified PCP.	3	x	x	x	x	x		x
DEI	Classified DEI.	1	x	x	x	x	x		x
COSID	Classified COSID.	3	x	x	x	x	x		x
COLOR	Frame's color, which is the frame's drop precedence level mapped through DP_MAP configuration.	1	x	x	x	x	x		x
L3_RT	Set if frame is routed.	1	x	x		x	x	x	x
ES0_ISDX_KEY_ENA	Value of IFH.FWD.ES0_ISDX_KEY_ENA.	1	x	x	x	x	x		x
MIRROR_PROBE	Identifies frame copies generated as a result of mirroring. 0: Normal frame copy. 1: Mirror Probe 0. 2: Mirror Probe 1. 3: Mirror Probe 2.	2	x	x	x	x	x		x
L2_DMAC	Destination MAC address.	48		x					x
L2_SMAC	Source MAC address.	48		x	x				x

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	IP6_VID	IP_7TUPLE
ETYPE_LEN	Frame type flag indicating that the frame is EtherType encoded. Set if frame has EtherType >= 0x600.	1		x					
ETYPE	Frame's EtherType Overloading: OAM Y.1731 frames: ETYPE[6:0] contains OAM MEL flags, see the following. OAM_Y1731 is set to 1 to indicate the overloading of ETYPE. OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0b0000000 MEL=1: 0b0000001 MEL=2: 0b0000011 MEL=3: 0b0000111 MEL=4: 0b0001111 MEL=5: 0b0011111 MEL=6: 0b0111111 MEL=7: 0b1111111 Together with the mask, the following kinds of rules may be created: - Exact match. Fx. MEL = 2: 0b0000011 - Below. Fx. MEL< = 4: 0b000XXXX - Above. Fx. MEL> = 5: 0bXX11111 - Between. Fx. 3< = MEL< = 5: 0b00XX111 Where, 'X' means don't care.	16		x					
IP4	Frame type flag indicating the frame is an IPv4 frame. Set if frame is IPv4 frame (EtherType = 0x800 and IP version = 4)	1	x			x	x		x

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	IP6_VID	IP_7TUPLE
L2_PAYLOAD_ETYPE	Byte 0–7 of L2 payload after Type/Len field. Overloading: OAM Y.1731 frames (OAM_Y1731 = 1): Bytes 0–3: OAM PDU bytes 0-3 Bytes 4–5: OAM_MEPID Bytes 6–7: Unused.	64		x					
L3_FRAGMENT_TYPE	L3 Fragmentation type: 0: No Fragments: MF==0 && FO==0 1: Initial Fragments: MF==1 && FO == 0 2: Suspicious Fragment: FO!=0 && FO <= FRAGMENT_OFFSET_THRES 3: Valid Follow-up Fragment: FO > FRAGMENT_OFFSET_THRES where FRAGMENT_OFFSET_THRES is programmed in EACL::VCAP_ES2_FRAGMENT_CFG.FRAGMENT_OFFSET_THRES. MF: More Fragments flag in IPv4 header FO: Fragments Offset in IPv4 header	2				x	x		
ARP_ADDR_SPACE_OK	Set if hardware address is Ethernet.	1			x				
ARP_PROTO_SPACE_OK	Set if protocol address space is 0x0800.	1			x				
ARP_LEN_OK	Set if Hardware address length = 6 (Ethernet) and IP address length = 4 (IP).	1			x				
ARP_TARGET_MATCH	Target Hardware Address = SMAC (RARP).	1			x				
ARP_SENDER_MATCH	Sender Hardware Address = SMAC (ARP).	1			x				
ARP_OPCODE_UNKNOWN	Set if ARP opcode is none of the below mentioned.	1			x				

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	IP6_VID	IP_7TUPLE
ARP_OPCODE	ARP opcode. Only valid if ARP_OPCODE_UNKNOWN is cleared. 0: ARP request. 1: ARP reply. 2: RARP request. 3: RARP reply.	2			x				
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1				x	x		
L3_TTL_GT0	Set if IPv4 TTL / IPv6 hop limit is greater than 0.	1				x	x		x
L3_TOS	Frame's IPv4/IPv6 DSCP and ECN fields.	8				x	x		x
L3_IP4_DIP	IPv4 frames: Destination IPv4 address. IPv6 frames: Source IPv6 address, bits 63:32.	32	x		x	x	x		
L3_IP4_SIP	IPv4 frames: Source IPv4 address. IPv6 frames: Source IPv6 address, bits 31:0.	32	x		x	x	x		
L3_IP6_DIP	IPv6 frames: Destination IPv6 address. IPv4 frames: Bits 31:0: Destination IPv4 address.	128						x	x
L3_IP6_SIP	IPv6 frames: Source IPv6 address. IPv4 frames: Bits 31:0: Source IPv4 address.	128						x	x
DIP_EQ_SIP	Set if destination IP address is equal to source IP address.	1			x	x	x		x
L3_IP_PROTO	IPv4 frames (IP4 = 1): IP protocol. IPv6 frames (IP4 = 0): Next header.	8					x		
TCP_UDP	Frame type flag indicating the frame is a TCP or UDP frame. Set if frame is IPv4/IPv6 TCP or UDP frame (IP protocol/next header equals 6 or 17).	1							x

.....continued

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	IP6_VID	IP_7TUPLE
TCP	Frame type flag indicating the frame is a TCP frame. Set if frame is IPv4 TCP frame (IP protocol = 6) or IPv6 TCP frames (Next header = 6)	1				x			x
L4_DPORT	TCP/UDP destination port. Overloading for IP_7TUPLE: Non-TCP/UDP IP frames: L4_DPORT = L3_IP_PROTO.	16				x			x
L4_SPORT	TCP/UDP source port.	16				x			x
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, custom. Input into range checkers is taken from classified results (VID, DSCP) and frame (SPORT, DPORT, custom).	16	x			x			x
SPORT_EQ_DPORT	Set if TCP/UDP source port equals TCP/UDP destination port.	1				x			x
SEQUENCE_EQ0	Set if TCP sequence number is 0.	1				x			x
L4_FIN	TCP flag FIN.	1				x			x
L4_SYN	TCP flag SYN.	1				x			x
L4_RST	TCP flag RST.	1				x			x
L4_PSH	TCP flag PSH.	1				x			x
L4_ACK	TCP flag ACK.	1				x			x
L4_URG	TCP flag URG.	1				x			x
L3_PAYLOAD	Payload bytes after IP header. IPv4: IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header.	96					x		
L4_PAYLOAD	Payload bytes after TCP/UDP header. Overloading for IP_7TUPLE: Non TCP/UDP frames (TCP_UDP = 0): Payload bytes 0 – 7 after IP header. IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header for non TCP/UDP IPv4 frames.	64				x			x

.....continued									
Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	IP6_VID	IP_7TUPLE
OAM_CCM_CNTS_EQ0	Flag indicating if dual-ended loss measurement counters in CCM frames are used. This flag is set if the TxFCf, RxFCb, and TxFCb counters are set to all zeros.	1		x					
OAM_Y1731	Set if frame's EtherType = 0x8902. If set, ETYPE is overloaded with OAM MEL flags, See ETYPE.	1		x					

4.24.2 VCAP ES2 Actions

VCAP ES2 supports only one action, which applies to all VCAP ES2 keys. The action is listed in the following table. When programming an action in VCAP ES2, the associated action fields listed must be programmed in the listed order with the first field in the table starting at bit 0 in the action.

Table 4-191. VCAP ES2 Actions

Action Name	Description	Size
HIT_ME_ONCE	If set, the next frame hitting the rule is copied to CPU using CPU_QUEUE_NUM. CPU_COPY overrules HIT_ME_ONCE implying that if CPU_COPY is also set, all frames hitting the rule are copied to the CPU.	1
INTR_ENA	If set, an interrupt is triggered when this rule is hit.	1
FWD_MODE	Forward selector: 0: Forward. 1: Discard. 2: Redirect. 3: Copy.	2
COPY_QUEUE_NUM	QSYS queue number when FWD_MODE is redirect or copy. Same encoding as in AFI:DTI_TBL:DTI_PORT_QU	16
COPY_PORT_NUM	QSYS port number when FWD_MODE is redirect or copy. Same encoding as in AFI:DTI_TBL:DTI_PORT_QU	7
MIRROR_PROBE_ID	Signals a mirror probe to be placed in the IFH. Only possible when FWD_MODE is copy. 0: No mirroring. 1–3: Use mirror probe 0-2.	2
CPU_COPY	Generate a copy to the CPU.	1
CPU_QUEUE_NUM	CPU queue number. Used when CPU_COPY is set.	3
POLICE_ENA	If set, frame is policed by policer pointed to by POLICE_IDX.	1
POLICE_REMARK	If set, frames exceeding policer rates are marked as yellow but not discarded.	1

.....continued		
Action Name	**Description**	**Size**
POLICE_IDX	Policer index.	6
ES2_REW_CMD	Command forwarded to REW: 0: No action. 1: SWAP MAC addresses. 2: Do L2CP DMAC translation when entering or leaving a tunnel.	3
CNT_ID	Counter ID, used per lookup to index the VCAP counters (EACL:CNT_TBL). Multiple VCAP ES2 entries can use the same counter.	11
IGNORE_PIPELINE_CTRL	Ignore pipeline control. This enforces the use of the VCAP ES2 action even when the pipeline control has terminated the frame before VCAP ES2., for instance when the pipeline action is LBK_QS or the pipeline action is an inject action and the pipeline point is located in REW.	1

4.25 Egress Access Control Lists

The egress access control list (EACL) block performs the following tasks.

- Maps the VPORT number information received from QSYS into a physical/logical port number or interface number.
- Controls the VCAP ES2 lookups in terms of key selection, evaluation of range checkers, and control of specific VCAP ES2 key fields such as EGR_PORT_MASK and EGR_PORT_MASK_RNG.
- Generates VCAP ES2 keys and execute up to two lookups.
- Updates VCAP ES2 statistics based on match results.
- Assigns actions from VCAP ES2 matching or default actions when there is no match.
- Assign policers to specific frame flows based on match results and keeps associated policer statistics.

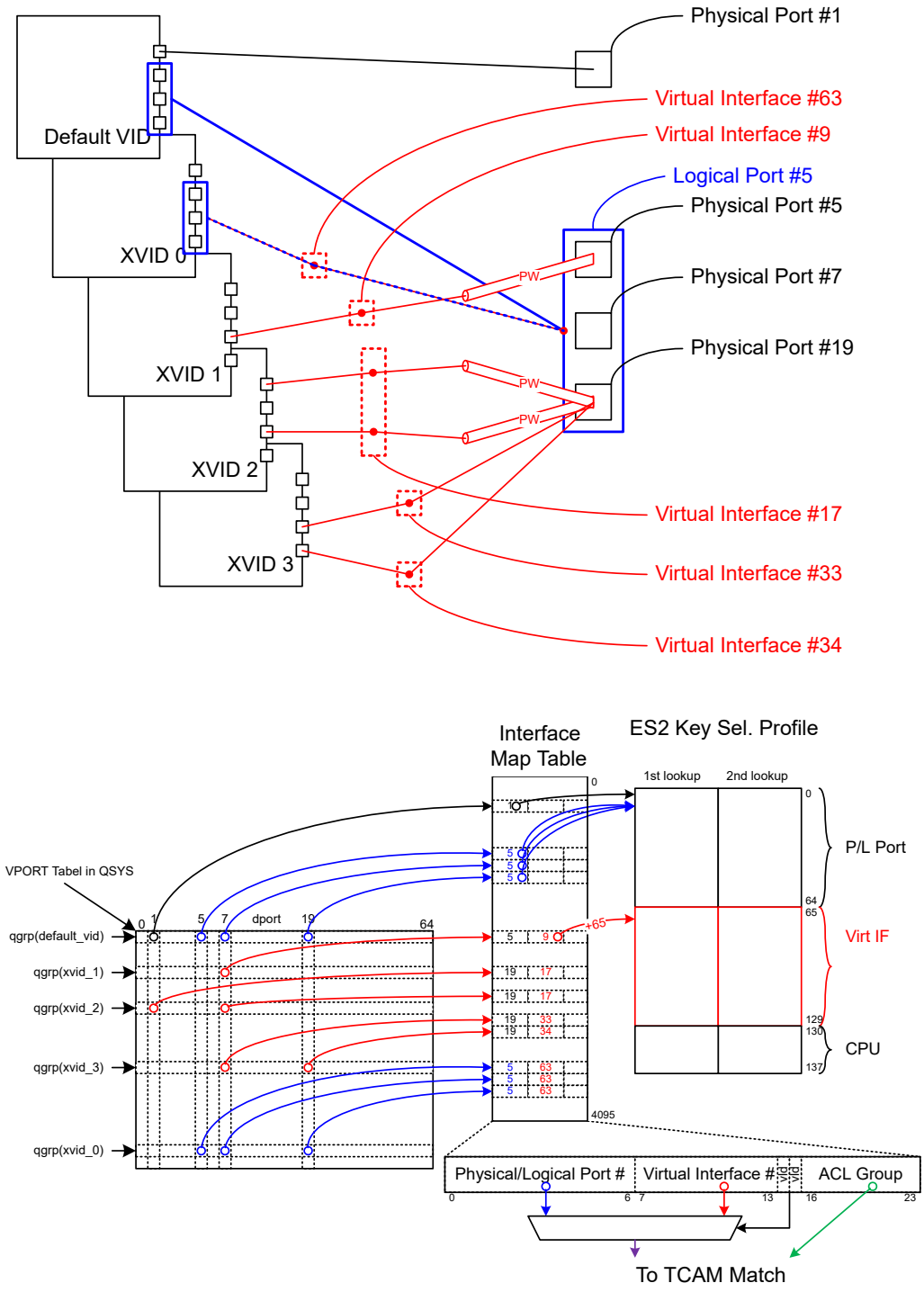
4.25.1 VCAP ES2 Operation

4.25.1.1 Interface Map Table

The interface map table maps a VPORT received from the queue system to a physical/logical egress port number or interface number. The VPORT table is located in XQS:QMAP_VPORT_TBL, and maps from a (QGRP, DST_PORT) to a VPORT. The queue system further maps the VPORT to a hierarchy of scheduling elements (SEs) which are connected to physical ports. To get to the same information in the EACL, the Interface map table must be setup accordingly. In addition to the port number, the interface map table also contains a virtual interface identifier and an ACL group identifier.

The following figure shows how a network example can be mapped to the concepts of the interface map table and influence the VCAP ES2 key selection and key generation.

Figure 4-65. Interface Map Table Example



The result of the interface map table lookup is used for two purposes:

- Provides a port number or interface number and an ACL group identifier to be used in the VCAP ES2 lookup
- Selects a VCAP ES2 key selection profile.

The interface map table instructs whether the physical/logical port number or the virtual interface number is to be used for the key selection profile and the VCAP ES2 key.

4.25.1.2 Profile Configuration and Key Selection

This section provides information about special profile configurations that control the key generation for VCAP ES2.

The following table lists the registers associated with profile configuration for VCAP ES2.

Table 4-192. Profile Configuration of VCAP ES2

Register	Description	Replication
EACL::VCAP_ES2_KEY_SEL	Configuration and enabling of the key selection for the VCAP ES2	Per profile per lookup

The profile configuration table is indexed the following way.

- Profiles associated with physical/logical port number are located from address 0 through 64.
- Profiles associated with virtual interfaces are located from address 65 through 129.
- Profiles associated with CPU destinations are located from address 130 through 137.

Each of the two lookups in VCAP ES2 must be enabled before use (VCAP_ES2_KEY_SEL.KEY_ENA) for a key to be generated and a match to be used. If a lookup is disabled for a profile, frames associated with the profile are not matched against rules in VCAP ES2 for that lookup.

Each frame is classified to one of seven overall VCAP ES2 frame types. The frame type determines which VCAP ES2 key types are applicable and controls which frame data to extract.

Table 4-193. VCAP ES2 Frame Types

Frame Type	Condition
IPv6 frame	The Type/Len field is equal to 0x86DD. The IP version is 6. Special IPv6 frames: • IPv6 TCP frame: Next header is TCP (0x6). • IPv6 UDP frame: Next header is UDP (0x11). • IPv6 Other frame: Next header is neither TCP nor UDP.
IPv4 frame	The Type/Len field is equal to 0x800. The IP version is 4. Special IPv4 frames: • IPv4 TCP frame: IP protocol is TCP (0x6). • IPv4 UDP frame: IP protocol is UDP (0x11). • IPv4 Other frame: IP protocol is neither TCP nor UDP.
(R)ARP frame	The Type/Len field is equal to 0x0806 (ARP) or 0x8035 (RARP).
OAM Y.1731 frame	The Type/Len field is equal to 0x8902 (ITU-T Y.1731).
SNAP frame	The Type/Len field is less than 0x600. The Destination Service Access Point field, DSAP is equal to 0xAA. The Source Service Access Point field, SSAP is equal to 0xAA. The Control field is equal to 0x3.
LLC frame	The Type/Len field is less than 0x600 The LLC header does not indicate a SNAP frame.

.....continued	
Frame Type	**Condition**
ETYPE frame	The Type/Len field is greater than or equal to 0x600, and the Type field does not indicate any of the previously mentioned frame types, that is, ARP, RARP, OAM, IPv4, or IPv6.

Each profile controls the key selection for each of the two lookups in VCAP ES2 through the VCAP_ES2_KEY_SEL register. For some frame types (OAM Y.1731, SNAP, LLC, EYTYPE) the key selection is given as only the MAC_ETYPE key can be used. For others frame types (ARP and IP) multiple keys can be selected from. The following table lists the applicable key types available for each frame type listed in Table 126, page 294.

Table 4-194. VCAP ES2 Key Selection

Frame Type	Applicable VCAP ES2 keys
IPv6 frames	IP6_VID IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
IPv4 frames	IP4_VID IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
(R)ARP frames	ARP MAC_ETYPE
OAM Y.1731 frames	MAC_ETYPE
SNAP frames	MAC_ETYPE
LLC frames	MAC_ETYPE
ETYPE frames	MAC_ETYPE

For information about VCAP ES2 keys and actions, see 4.24 VCAP ES2 Keys and Actions.

4.25.1.3 VCAP ES2 Range Checkers

The following table lists the registers associated with configuring VCAP ES2 range checkers.

Table 4-195. VCAP ES2 Range Checker Configuration

Register	Description	Replication
EACL::VCAP_ES2_RNG_CTRL	Configuration of the range checker types	Per range checker
EACL::VCAP_ES2_RNG_VALUE_CFG	Configuration of range start and end points	Per range checker

VCAP ES2 supports 16 global range checkers in the IP4_TCP_UDP, IP_7TUPLE, IP4_VID keys. All frames using these keys are compared against the range checkers and a 1-bit range “match/no match” flag is returned for each range checker. The combined 16 match/no match flags are used in the L4_RNG key field. So, it is possible to include for example ranges of DSCP values and/or ranges of TCP source port numbers in the ACLs.

Note, VCAP CLM and VCAP IS2 also supports range checkers but they are independent of the VCAP ES2 range checkers.

The range key is generated for each frame based on extracted frame data and the configuration in EACL::VCAP_ES2_RNG_CTRL. Each of the 16 range checkers can be configured to one of the following range types.

- TCP/UDP destination port range:

  Input to the range is the frame's TCP/UDP destination port number.

  Range is only applicable to TCP/UDP frames.
- TCP/UDP source port range:

  Input to the range is the frame's TCP/UDP source port number.

  Range is only applicable to TCP/UDP frames.
- TCP/UDP source and destination ports range: Range is matched if either source or destination port is within range.

  Input to the range are the frame's TCP/UDP source and destination port numbers.
  Range is only applicable to TCP/UDP frames.
- VID range:

  Input to the range is the classified VI.
  Range is applicable to all frames.
- DSCP range
  :

  Input to the range is the classified DSCP value.
  Range is applicable to IPv4 and IPv6 frames.

Range start points and range end points are configured in EACL::VCAP_ES2_RNG_VALUE_CFG with both the start point and the end point being included in the range. A range matches if the input value to the range (for instance the frame's TCP/UDP destination port) is within the range defined by the start point and the end point.

4.25.1.4 Miscellaneous VCAP ES2 Key Configurations

The following table lists the registers associated with configuration that control the specific fields in the VCAP ES2 keys.

Table 4-196. Other VCAP ES2 Key Configurations

Register	Description	Replication
EACL::TPID_CFG.TPID_VAL	Configures custom TPID value.	Per TPID
EACL::ETAG_CFG	Enables E-TAG awareness.	None
EACL::RTAG_CFG	Enables R-TAG awareness.	None
EACL::DP_MAP.DP	Maps drop precedence level to key field COLOR.	None
EACL::OPTIONS.ALLOW_FWD_AT_INVALID_SOF_DATA	Configures default forwarding behavior during resource exhaustion.	None

VCAP ES2 must be configured to identify VLAN tags and other supported tags. By default, customer tags (C-tags) and service tags (S-tags) are identified by TPIDs 0x8100 and 0x88A8. In addition, custom TPIDs can be identified by programming EACL_TPID_CFG. This should be done in accordance with ANA_CL::VLAN_STAG_CFG and REW::TPID_CFG.

Awareness of E-TAGs (IEEE 802.1BR Bridge Port Extension) and R-TAGs (IEEE 802.1CB Frame Replication and Elimination for Reliability) are enabled individually.

VCAP ES2 may experience an exhaustion of resources during which VCAP ES2 lookups no longer can be performed. It is configurable whether to discard or forward frames experiencing this.

4.25.1.5 Processing Actions

VCAP ES2 returns an action for each enabled VCAP ES2 lookup. If the lookup matches an entry in VCAP ES2 then the associated action is returned. A default action is returned when no entries are matched. The first VCAP ES2 lookup returns a common default action while the second VCAP ES2 lookup returns a default action per physical egress port. There are no difference between an action from a match and a default action.

VCAP ES2 contains 74 default actions that are allocated the following ways.

- 0–64: Per egress port default actions for second lookup.
- 65–72: Per CPU egress queue default actions for second lookup.
- 73: Common default action for first lookup.

The following table lists how the two VCAP ES2 action are combined when both VCAP ES2 actions are valid. For some cases, the processing is sequential, meaning that if both actions have settings for the same (for instance POLICE_IDX), the second processing takes precedence. Others have a resolution function (for instance FWD_MODE). Still others are sticky meaning they cannot be undone by the second processing if already set by the first.

Table 4-197. Combining VCAP ES2 Actions

Action Name	Combining Actions
HIT_STICKY	Both applied.
HIT_ME_ONCE	Sticky.
INTR_ENA	Sticky.
FWD_MODE	Resolution function. See Table 4-198.
COPY_QUEUE_NUM	Depends on the FWD_MODE resolution function.
COPY_PORT_NUM	Depends on the FWD_MODE resolution function.
MIRROR_PROBE_ID	Depends on the FWD_MODE resolution function.
CPU_COPY	Sticky.
CPU_QUEUE_NUM	Depends on EACL::ES2_CTRL.ACTION_CPU_COPY_SEL.
POLICE_ENA	Second lookup takes precedence.
POLICE_REMARK	Second lookup takes precedence if both actions have POLICE_ENA set to 1.
POLICE_IDX	Second lookup takes precedence if both actions have POLICE_ENA set to 1.
ES2_REW_CMD	Depends on EACL::ES2_CTRL.ACTION_REW_CMD_SEL.
CNT_ID	Both applied.
IGNORE_PIPELINE_CTRL	Processed individually for each action.

The FWD_MODE resolution function is described in FWD_MODE Resolution Function. The action resulting from a combination of REDIRECT and DISCARD is configured in EACL::ES2_CTRL.ACTION_REDIR_DISCARD_SEL. The destination resulting from a combination of two REDIRECTs or two DISCARDs is configured in EACL::ES2_CTRL.ACTION_REDIR_COPY_SEL.

Table 4-198. FWD_MODE Resolution Function

FWD_MODE from Second Lookup	FWD_MODE from First Lookup			
	FORWARD	DISCARD	REDIRECT	COPY
FORWARD	Forward	Discard	Redirect	Copy
DISCARD	Discard	Discard	Redirect/Discard	Discard
REDIRECT	Redirect	Redirect/Discard	Redirect 1st/2nd	Redirect
COPY	Copy	Discard	Redirect	Copy 1st/2nd

In case the resolved action from the FWD_MODE resolution function is redirect or copy and the policer result is to discard the respective frame copy, it is configurable through EACL::ES2_CTRL.EACL_ALLOW_FP_COPY whether the copied or redirected frame copy is discarded.

In case both actions result in a CPU copy request, EACL::ES2_CTRL.ACTION_CPU_COPY_SEL defines which of the CPU_QUEUE_NUM values is used as destination CPU queue.

In case a CPU copy action occurs and the original frame is to be discarded by the policer, it is configurable by EACL::ES2_CTRL.EACL_ALLOW_CPU_COPY whether a CPU copy is to be done although the original frame copy was discarded. Note that a CPU copy can either be triggered by a CPU copy action or by a HIT_ME_ONCE action.

If both actions are valid, ACTION_REW_CMD_SEL defines how the command to the rewriter is generated. It can be either the first lookup result, the second lookup result, a logical OR of both lookup results or a logical AND of both results.

4.25.1.6 VCAP ES2 Statistics and Diagnostics

The following table lists the registers associated with VCAP ES2 statistics and diagnostics.

Table 4-199. VCAP IS2 Statistics and Diagnostics

Register	Description	Replication
EACL::SEC_LOOKUP_STICKY	Sticky bits for each key type used in VCAP ES2.	Per lookup
EACL::ES2_CNT	VCAP ES2 match counters. Indexed with VCAP ES2 action CNT_ID.	2,048

Each key type use in a lookup in VCAP ES2 triggers a sticky bit being set in EACL::SEC_LOOKUP_STICKY. A sticky bit is cleared by writing 1 to it.

Each action returned by VCAP ES2 triggers one of 2,048 counters (EACL::ES2_CNT) to be incremented. This applies to both actions from matches and default actions. The index into the counters is provided by VCAP ES2 action CNT_ID. The CNT_ID is fully flexible meaning multiple actions can point to the same counter. The counters are writable so they can be preset to any value. A counter is cleared by writing 0.

4.26 Shared Queue System and Hierarchical Scheduler

The device includes a shared queue system with a per-port ingress queue and shared egress queues. The shared queue system has 32 megabits of buffer. It receives frames from 70 ports, stores them in a shared packet memory, and transmits them towards 70 destination ports. The first 65 of the ports are assigned directly to a specific front port, whereas the last five ports are internal ports.

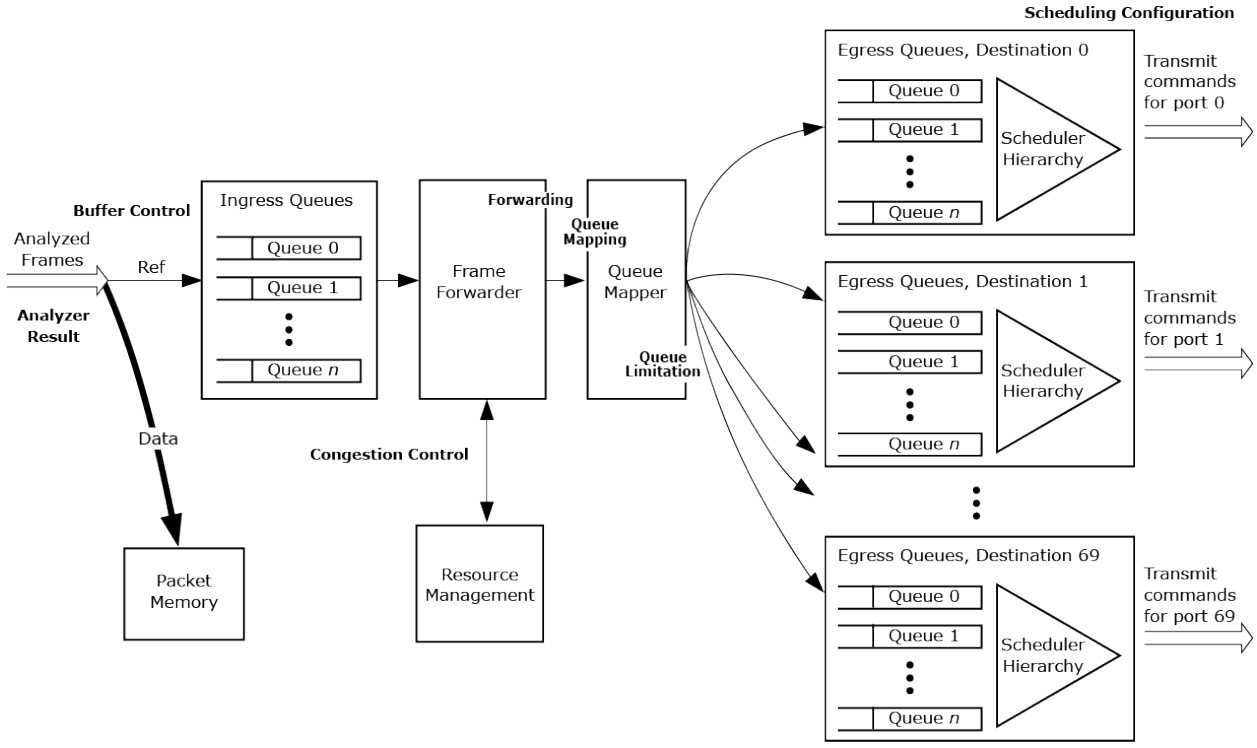
Table 4-200. Port Definitions in Shared Queue System

Ports	Connection
0–64	Front ports, 10 Mbps to 10 Gbps depending on port number

.....continued	
Ports	**Connection**
65–66	Frame injection and extraction CPU ports. The CPU connects these ports to DMA logic or a register interface.
67	IP multicast loopback port (VD0). When the queue system is requested to transmit to VD0, it generates multiple copies. All copies are, outside the queue system, looped back into the analyzer. The rewriter changes the ingress port number to VD0 before the loop. The analyzer generates a routed version of each of the frame copies coming from port VD0.
68	AFI loopback port (VD1). When the queue system is requested to send to VD1, it sends the request to the AFI. The AFI stores the request and can be configured to transmit the frame to any port in a programmed schedule. The AFI can transmit the frame to any of the egress ports. If VD1 is selected as destination by the AFI, the frame is looped back into the analyzer.
69	IP-in-IPv4 loopback port (VD2). When the queue system is requested to transmit to VD2, the frame is looped back into the analyzer.

The following figure provides an overview of shared queue system.

Figure 4-66. Shared Queue System Block Diagram



Frames are stored in the packet memory and linked into the ingress queue for the logical source port along with the analyzer result.

The analyzer result is then processed by a frame forwarder, frame by frame. Each of the frame transmission requests are added to an egress queue in the queue system attached to a specific egress port. A transmission request can be a normal switching request to another port, a mirror copy, for stacking updates, and for a CPU connected to one of the egress ports. For VD0, it can do multiple copies of the request. The frame forwarder can also request that a specific frame copy be discarded.

The frame forwarder is efficient because only frame references are switched. The frame data itself is not moved within the queue system.

The forwarding request is passed through a queue mapper, which then selects an egress queue based on the classified properties of the frame. For example, a simple switch may use the classified QoS class only and an advanced carrier switch may use MPLS-classified traffic class (TC).

A congestion control system can decide to discard some or all copies of the frame. This decision is based on consumption per QoS level. The queue mapper can be configured to change the drop decision using a congestion control mechanism based on queue sizes rather than only QoS class.

Each destination port has a part of the total shared queue pool attached. Transmission is scheduled towards the destination ports through a hierarchical scheduling system, where all queues belonging to each port are arbitrated. On the way out, frames pass through the rewriter where the frame data can be modified due to for instance routing or VLAN tagging. The queue system does not change the original frame data.

An ingress port operates in one of three basic modes: drop mode, port flow control, or priority-based flow control. The following table lists the modes and flow definitions.

Table 4-201. Ingress Port Modes

Mode	Flow
Drop mode (DROP)	The link partner is unaware of congestion and all frames received by the port are evaluated by the congestion control function where copies of the frame can be discarded.
Port flow control (FC)	The link partner is informed through pause frames (full-duplex) or collisions (half-duplex) that the port cannot accept more data. If the link partner obeys the pause requests, no incoming data is discarded by the port. If the pause frames are disobeyed by the link partner, the buffer control function in the queue system discards data before it reaches the ingress queues.
Priority-based flow control (PFC)	The link partner is informed about which QoS classes in the queue system are congested and the link partner should stop scheduling more frames belonging to these QoS classes. If the pause frames are disobeyed by the link partner, the buffer control function in the queue system discards data before it reaches the ingress queues.

4.26.1 Analyzer Result

The analyzer provides required information for switching frames in the shared queue system. The following table lists the information provided and a general description of the purpose.

Table 4-202. Analyzer Result

Group	Field	Function
Destination selection	Destination set	Set of ports to receive a normal switched copy.
Destination selection	CPU queue mask	Set of CPU queues to receive a copy.
Destination selection	Mirror	Requests mirror copy.
Destination selection	Learn all	Requests learn copy to stack ports.
Congestion control	Drop mode	This particular frame can be dropped even if ingress port is setup for flow control.
Congestion control	DP	Drop precedence level
Congestion control	Priorities	Ingress resource priority of the frame. Four classified classes are provided from the analyzer: QoS, PCP, COSID, and TC. All values between 0 and 7.
Queue mapping	SP	Super priority frame

.....continued		
Group	**Field**	**Function**
Queue mapping	QGRP	Queue group for service-based egress queuing.
Statistics	OAM type	Service counters can be configured to only count certain OAM frame types.

4.26.2 Buffer Control

Frames are stored in a shared packet memory bank, which contains 22,795 words of 184 bytes each. A memory manager controls which words the packet writer uses to store a frame.

With proper configuration, the packet memory bank always has a free word available for incoming frames. There are configurable thresholds for triggering pause frame requests on ports enabled for flow control and for discarding frames due to too large memory use when for instance pause frames are disobeyed by the link partner.

Frames discarded at this stage are discarded without considering the frame's destination ports. This kind of discarding is referred to as ingress tail dropping. It is an indication of head-of-line blocking and is not usually desired. For information about how it can be avoided by proper configuration of the congestion control system, see 4.26.4 Congestion Control.

The pause frame and discard mechanisms are activated when a threshold for the individual port's memory use is reached and when a threshold for the total use of memory is reached.

Pause frame generation can be based on port consumption (MAC flow control) or priority consumption per port (Priority-based flow control). It is enabled in QSYS::PAUSE_CFG and QSYS::PFC_CFG.

The following table lists the configuration registers involved in the buffer control.

Table 4-203. Buffer Control Registers Overview

Register	Description	Replication
QSYS::ATOP	Maximum allowed memory use for a port before tail dropping is activated.	Per port
QSYS::ATOP_TOT_CFG	Total memory use before tail dropping can be activated.	None
QSYS::PAUSE_CFG	Thresholds for when to start and stop port flow control based on memory use by the port.	Per port
QSYS::PFC_CFG	Enable priority-based flow control.	Per port
QSYS::PAUSE_TOT_CFG	Total memory use before flow control is activated.	None
QFWD::SWITCH_PORT_MODE	Enable drop mode for a port.	Per port

The following table lists available status information.

Table 4-204. Buffer Status Registers Overview

Register	Description	Replication
QSYS::MMGT_PORT_USE	Current consumption for a specific port	None
QSYS::MMGT_PORT_VIEW	Selection of port to show usage for	None
QSYS::MMGT_PRIO_USE	QSYS::MMGT_PRIO_USE Current consumption for a specific port and priority	None
QSYS::MMGT	Number of free words in packet memory	None

4.26.3 Forwarding

The frame forwarder processes frames at the head of the per-port ingress queues by adding references to the egress queues for each of the frames' destination ports. The reference points from the egress queues to the stored

frames. The forwarder can add from 312.5 million to 625 million references per second depending on the frame size distribution.

The ingress queues are selected by a weighted round-robin search, where the weights are configured in QFWD::SWITCH_PORT_MODE.FWD_URGENCY.

Each frame is processed as either a drop-mode frame or a flow control frame, which influences how the congestion control works. For a drop-mode frame, the congestion control can discard frame copies while for a flow control frame, it lets the frame stay at the head of the ingress queue blocking further processing of frames from the port until the congestion situation is over. A frame is processed as a drop-mode frame if one of the following conditions is met.

- The ingress port is configured to be an ingress drop port.
- The egress port is configured to be an egress drop port.
- The frame itself is analyzed to be a drop frame.

In terms of discarding frames, the forwarder can work in an ingress-focused statistics mode or an egressfocused statistics mode. The mode is configured per ingress port in XQS::STAT_CNT_CFG, which determines whether a frame received and discarded to a particular port is counted in the statistics for the egress port that lost a frame, or for the ingress port. If a frame is discarded towards several egress ports and the mode is set to ingress-focused, only one discard is counted. For the egress-focused mode, each dropped copy is counted.

4.26.3.1 Forward Pressure

The worst-case switching capacity in the frame forwarder is 312.5 million frames per second. This corresponds to 208 Gbps line rate with minimum-sized frames. However, if a large share of these frames have multiple destinations that are all close to the 64-byte minimum frame size, the processing speed is insufficient to process the destination copy one at a time. The consequence is that the ingress queues start to fill. The ingress queues contain both high and low priority traffic, unicast, and multicast frames. To avoid head-of-line blocking effects from the ingress queues filling up and eventually leading to tail dropping, a forward pressure system is in place. It can discard frame copies based on the size of the ingress queues. Forward pressure is configured in the QSYS::FWD_PRESSURE registers, where each port is allowed only a certain number of copies per frame if the number of pending transmit-requests in the port's ingress queue exceeds a threshold.

Another forwarder protection mechanism is referred to as aggressive taildrop mode. By this mechanism, the switching core is more protected against exhaustive ingress ports. It is chip-wide activated when any ingress queue exceeds the FWD_PRESSURE watermark for the port. If the switch is in this state, all ingress traffic belonging to a priority not under flow control (MAC/PFC) and with ingress memory consumption exceeding the PAUSE_START watermark for the port is subject to discarding towards destinations for which all reserved egress port memory used.

The following table lists the registers associated with configuring forward pressure.

Table 4-205. Forward Pressure Configuration Registers Overview

Register	Description	Replication
QSYS::FWD_PRESSURE.FWD_PRESSURE	Configures forwarding pressure limits per port.	Per port
QSYS::PAUSE_CFG.AGGRESSIVE_TAILDRO P_ENA	Enables aggressive tail discarding	Per port
QSYS::MMGT_IQ_STAT	Returns current ingress queue size.	Per port
QFWD::FWD_PRESSURE.FWD_PRESS_ DROP_CNT	Counts number of frame copies discarded due to this feature. Value is total discards for all ports in common.	None

4.26.3.2 Destination Selection

The forwarder adds references to the egress queues by evaluating the destination selection information shown in the following table from the analyzer.

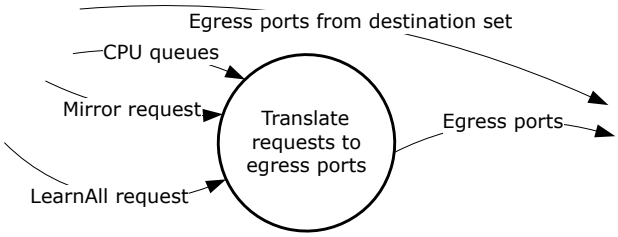
Table 4-206. Analyzer Destination Selection

Field	Function
Destination set	Set of ports to receive a normal switched copy. All ports except the internal extraction ports have a bit in this mask.
CPUQ	Set of CPU queues to receive a copy
Mirror	Request mirror copy
Learn-all	Request learn copy to stack ports

The destination set selects specific egress ports and the CPUQ, mirror, and learn-all fields are indirect frame destinations and must be translated to transmit requests through the frame copy configuration table. The table provides a specific egress port and QoS class to use in congestion control. The table is accessed through the QFWD::FRAME_COPY_CFG register.

The following illustration shows the translation of transmit requests.

Figure 4-67. Translation of Transmit Requests



The CPUQ field instructs which of the eight CPU extraction queues should get a copy of this frame. Each queue in the queue system is translated into either one of the two internal CPU ports or to a front port. The egress QoS class can be set to the same value as the ingress QoS class or to a specific value per CPU extraction queue. A super priority can also be used. For more information, see 4.26.9.2 Super Priorities.

The mirror field is the mirror probe number, which is set by the analyzer if the frame must be mirrored. There are three mirror probes available, and for each of the probes, associated egress port and egress QoS class can be set by the translation table.

Finally, if the learn-all field is set, the frame should also be forwarded to the stacking links. It is selectable to forward the frame to either of the two stacking links or to both, depending on the stacking topology.

The following table lists the configuration registers associated with the forwarding function.

Table 4-207. Frame Forwarder Registers Overview

Register	Description	Replication
QFWD::SWITCH_PORT_MODE	Enables forwarding to/from port. Forwarding speed configuration. Drop mode setting.	Per port
QFWD::FRAME_COPY_CFG	Translates CPUQ/mirror/learn all to enqueuing requests.	12
QFWD::FRAME_COPY_LRNA_CFG	Learn All frame copy extra port.	None
QFWD::CPUQ_DISCARD	Disables enqueuing to specific CPU queues.	None
QSYS::FWD_PRESSURE	Configures forwarding pressure limits per port.	Per port

.....continued		
Register	**Description**	**Replication**
QFWD::MIRROR_CFG	Configures whether discarded copies should still generate a mirror copy.	None

4.26.4 Congestion Control

The buffer control can only guarantee that no ingress port consumes more than a certain amount of memory. Discards made by the buffer control are tail drops. The congestion control is more advanced in that it facilitates intelligent discard where only frames belonging to a congested flow are discarded.

There are two independent congestion control systems in the device referred to as queue limitation and basic congestion avoidance. The queue limitation method typically shows the best performance as it keeps the queue drop conditions more independent of other queues as opposed to the basic congestion avoidance system where only classified priorities are used. The queue limitation method is described in more details in 4.26.6 Queue Congestion Control.

If the forwarder finds that some frame copies are made to a congested flow, it discards the copies (drop-mode frames) or lets the copies block further processing of the ingress queue (flow control ports).

The congestion control information from the analyzer result involved in the congestion control is listed in the following table.

Table 4-208. Analyzer Congestion Results

Field	Function
Drop mode	This frame can be dropped even if port is setup for flow control. This function is mainly used by the analyzer for stacking use where the ingress unit's source port determines how congestion should be handled.
DP	Drop precedence level. A value 0-3, telling how yellow the frame is.
QoS class	Frame's ingress QoS class

The current consumption of resources in the congestion controller is updated when a reference is added to one of the egress queues. The congestion controller tracks four different resources:

- Ingress packet memory consumption. Each frame uses a number of 184-byte words.
- Egress packet memory consumption. Each frame uses a number of 184-byte words.
- Ingress frame reference consumption. Each frame uses one frame reference per frame copy.
- Egress frame reference consumption. Each uses one frame reference per frame copy.

There are 22,795 memory words and 22,795 frame references in total.

The accounting is done based on QoS classes and port numbers. Each account has a threshold associated specifying how many resources the account is permitted to consume. The accounts are as follows:

- Consumption per port per QoS class. This is a reservation account.
- Consumption per port. A frame does not consume resources from this account before the resources belonging to the previous account are consumed. This is a reservation account.
- Consumption per QoS class. A frame does not consume resources from this account before the resources belonging to the previous two accounts are consumed. This is a sharing account.
- Consumption in addition to the above. A frame does not consume from this account before the resources belonging to the previous three accounts are consumed. This is a sharing account.

The congestion controller updates an accounting sheet with the listed accounts for each of the tracked resources.

The following illustration shows reservation accounts for the packet memory accounting sheets when a 700-byte frame from port 0 forwarded to ports 1 and 5 with QoS class 2 is added. The frame size requires the use of five words in the packet memory. The account for ingress port 0, QoS class 2 is reserved three words. Before the frame is added, it is checked if the account was fully utilized. If the account was not fully used, the frame is allowed into the queue system. After the frame is added, the account is updated with the five words and therefore uses two more than

was reserved. These are consumed from the port account. The port account is reserved 0 words and the added two words therefore also close the port account. This affects further frame reception.

In the egress side, there are no words reserved for the account per port per QoS class. The port account for port 1 is 8 words and 0 words for port 5.

Figure 4-68. Accounting Sheet Example

Packet memory, ingress user

Priority	0	1	2	3	4	5	6	7	P
Port									
0			5/3						2/0
1									
...									
...									
...									

Packet memory, egress user

Priority	0	1	2	3	4	5	6	7	P
Port									
0									
1			5/0						5/8
...									
5			5/0						5/0
...									

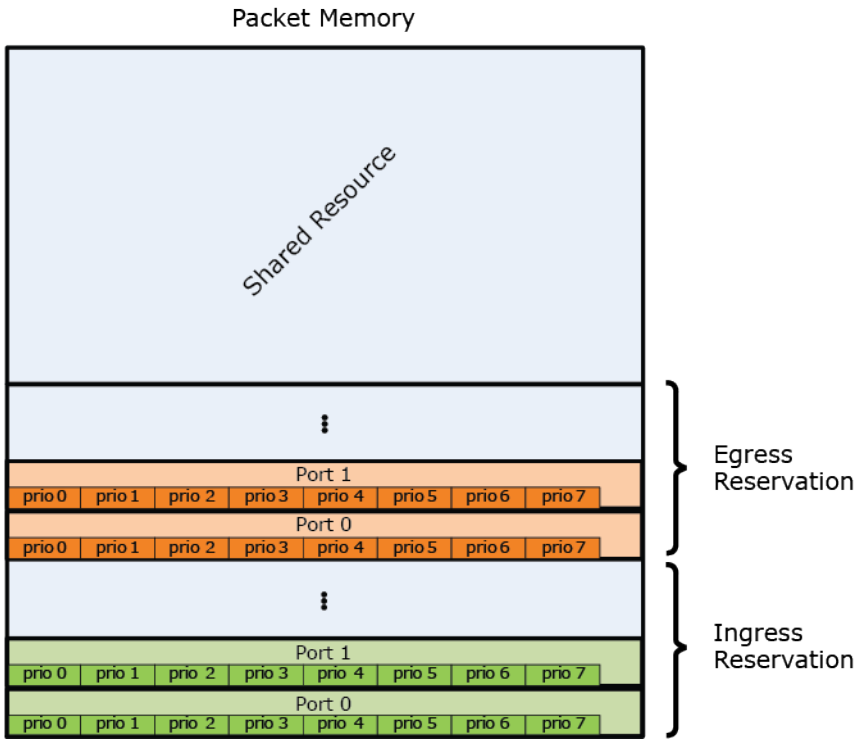
For PFC purposes, the congestion controller contains an additional accounting sheet of the ingress memory consumption. The sheet has its own set of thresholds. The result of the accounting is whether to send PFC pause frames towards the link partner.

The combined rule for allowing a frame copy is if both of the following are true.

- Either ingress or egress memory evaluation must allow the frame copy.
- Either the ingress or egress reference evaluation must allow the frame copy.

When the reserved accounts are closed, the shared accounts spanning multiple ports are checked. Various rules for regarding these accounts as open or closed exist. These rules are explained in the following through three major resource modes, which the congestion controller is intended to be used in.

Figure 4-69. Reserved and Shared Resource Overview

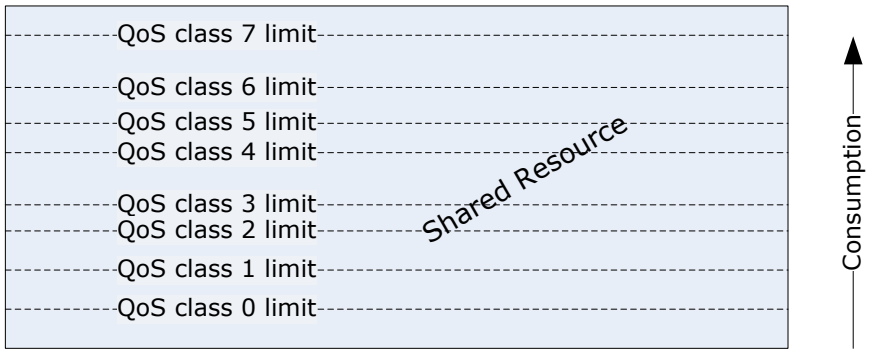


The resource modes are defined by the way the shared resources are distributed between classes of frames. Note, that there are many options for configuring different resource mode than the modes described in the following. This is outside the scope of this document.

4.26.4.1 Strict Priority Sharing

This resource mode considers higher QoS classes as more important than lower QoS classes. The shared area is one big pool shared between all QoS classes. Eight thresholds define the limit for each of the eight QoS classes. If the total consumption by all QoS classes exceeds one of the thresholds, the account associated with the particular QoS class is closed.

Figure 4-70. Strict Priority Sharing



One important property of this mode is that higher QoS classes can block lower QoS classes. A congested flow with a high QoS class uses all of the shared resources up to its threshold setting and frame with lower QoS classes are thus affected (lower burst capacity).

The following table shows the configuration settings for strict priority sharing mode.

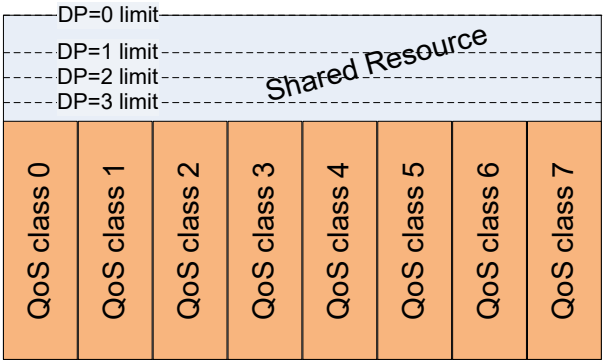
Table 4-209. Strict Priority Sharing Configuration Settings

Configuration	Setting
Reservation thresholds	These can be set to any desired value. The amount set aside for each account should be subtracted from the overall resource size, before the sharing thresholds are set up.
Ingress QoS class thresholds	Levels at which each QoS class should start to reject further enqueuing. Frames with multiple destinations are only accounted once because they are physically only stored once.
Ingress color thresholds	Set to their maximum value to disable. The color is not considered in this mode.
WRED group memberships	Disable all.
Egress sharing thresholds	Set all to 0 to only use ingress sharing control. Egress would account for multiple copies, which is not intended.
QoS reservation (QRES::RES_QOS_MODE)	Disable for all. No shared resources are set aside per QoS class.

4.26.4.2 Per Priority Sharing

This mode sets aside some memory for each QoS class and can in addition have an amount of memory reserved for green traffic only. This mode operates on ingress resources, in order to utilize that frames with multiple destinations are only stored once.

Figure 4-71. Per Priority Sharing



The following table shows the configuration settings for per priority sharing mode.

Table 4-210. Per Priority Sharing Configuration Settings

Configuration	Setting
Reservation thresholds	These can be set to any desired value, except for ingress per port reservation. The algorithm requires these to be zeroed.
Ingress QoS class thresholds	Set to amount of shared memory which should be set aside for each QoS class. The sum of all these should not exceed the amount of memory left after all the reservations has been subtracted.
Ingress color thresholds	Subtracting all reservations including QoS class shared areas may end up in some unused memory. Set the color thresholds to various levels to guarantee more space the greener.
WRED group memberships	Disable all.
Egress sharing thresholds	Set all to 0, as we only use ingress sharing control. Egress would account for multiple copies, which is not intended.
QoS reservation (QRES::RES_QOS_MODE)	Enable for all.

4.26.4.3 Weighted Random Early Discard (WRED) Sharing

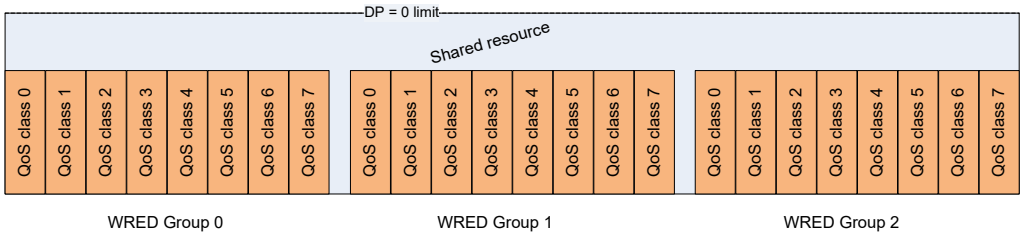
In this mode, the shared resource accounts close at random. A high consumption gives a high probability of discarding a frame. This sharing method only operates on the egress memory consumption.

There are three WRED groups, each having a number of ports assigned to it. Each WRED group contains 24 WRED profiles; one per QoS class per DP = 1, 2, 3. Each WRED profile defines a threshold for drop probability 0% and a threshold for drop probability 100%. Frames with DP = 0 value are considered green and have no WRED profile. Green frames should always be allowed.

Memory consumptions within a WRED group are tracked per QoS class.

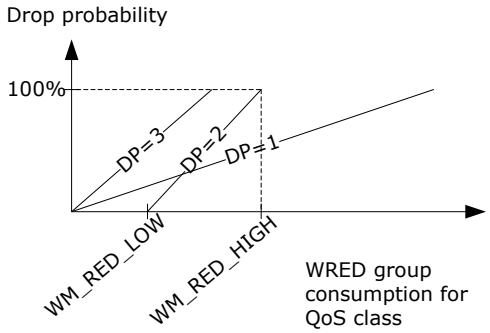
The WRED groups with each eight QoS classes are shown in the following figure. It is also possible to use fewer groups or different sizes for each QoS classes.

Figure 4-72. WRED Sharing



The following illustration shows an example of the WRED profiles for a QoS class within a WRED group. Thresholds are shown for DP = 2 only.

Figure 4-73. WRED Profiles



The WRED sharing is operating as an egress algorithm. The following table shows the configuration setting.

Table 4-211. WRED Sharing Configuration

Configuration	Setting
Reservation thresholds	These can be set to any desired value, except for ingress per port reservation. The algorithm requires these to be zeroed.
Ingress priority thresholds	Set to their maximum value, as the QoS classes should not have any sharing across WRED groups.
Ingress color thresholds	Set to their maximum.
WRED group memberships	Set to define the WRED groups. For each group, define as well the up to 24 WRED profiles (per QoS class and DP level)
Egress sharing thresholds	Set the QoS class thresholds to the highest possible consumption before any profile hits 100% probability. The color thresholds can then operate on the remainder of the resource in control and can all be set to the same value, allowing use of all the remaining. A discard due to the WRED profile takes precedence over other sharing states, so the area outside the group sections is only available for green traffic.
QoS reservation (QRES::RES_QOS_MODE)	Disable for all.

4.26.4.4 Threshold Configuration

There are a large number of thresholds for the reservations described. They are all found in the QRES::RES_CFG register, which is replicated 4,095 times:

Table 4-212. Threshold Configuration Overview

QRES::RES_CFG Replication	Description
0–559	Ingress memory reservation for port P priority Q accessed at index 8P+Q.
560-629	Ingress memory shared priority threshold for priority Q accessed at index 496+Q.
630–637	Ingress memory shared threshold for DP levels 1, 2, 3, and 0.
638–641	Ingress memory reservation for port P, shared between priorities, accessed at index 512+P.
1,024–2,047	Ingress reference thresholds. Same organization as for ingress memory but added offset 1,024.
2,048–3,071	Egress memory thresholds. Same organization as for ingress memory but added offset 2048.
3,072–4,095	Egress reference thresholds. Same organization as for ingress memory but added offset 3,072.

The following table lists other registers involved in congestion control.

Table 4-213. Congestion Control Registers Overview

Register	Description
QRES::RES_STAT (replicated same way as RES_CFG)	Returns maximum value the corresponding threshold has been compared with.
QRES::RES_STAT_CUR (replicated as RES_CFG)	Returns current value this threshold is being compared with.
QRES::WRED_PROFILE (72 profiles)	Profile description. 3 (grp) × 3 (DP) × 8 (prio) profiles exist.
QRES::WRED_GROUP	WRED group membership configuration per port.
QRES::RES_QOS_MODE	Enable QoS reservation for QoS classes.
QFWD::SWITCH_PORT_MODE	Controls whether yellow traffic can use reserved space (YEL_RSVD). Allows ports to use shared space (IGR_NO_SHARING, EGR_NO_SHARING).

4.26.4.5 Frame Forwarder Monitoring

The performance of the frame forwarder can be monitored through the registers listed in the following table.

Table 4-214. Frame Forwarder Monitoring Registers Overview

Register	Description
XQS::FWD_DROP_EVENTS	Sticky bits per port telling if drops from each individual ingress port occurred.
XQS::FWD_CPU_DROP_CNT	Counts number of frames not forwarded to the CPU due to congestion.
XQS::FWD_STAT_CNT	Counts number of frame copies added to the egress queues in total.
QFWD::FWD_PRESS_DROP_CNT	Counts number of frame copies discarded due to forward pressure. Value is the total discards across all ports.
QSYS::MMGT_IQ_STAT	Returns current ingress queue size.
QSYS::MMGT	Returns number of free references for the egress queues.

4.26.5 Queue Mapping

When a frame is forwarded to an egress port, a frame reference is inserted into one of the port's egress queues. The egress queues are read by a programmable scheduler consisting of a number of scheduler elements (SEs). Each SE serves 8 or 72 queues, configurable through the HSCH::HSCH_LARGE_ENA registers. Each SE selects the next input to transmit and forwards the decision to a next-layer SE, selecting from which of its inputs to transmit. There are a total of three layers, with layer 0 being the queue connected layer. Connections between the layers are fully configurable in HSCH::L0_CFG and HSCH::L1_CFG. For more information about SE functionality, see 4.26.7 Scheduling and 4.26.7.3 Bandwidth Control.

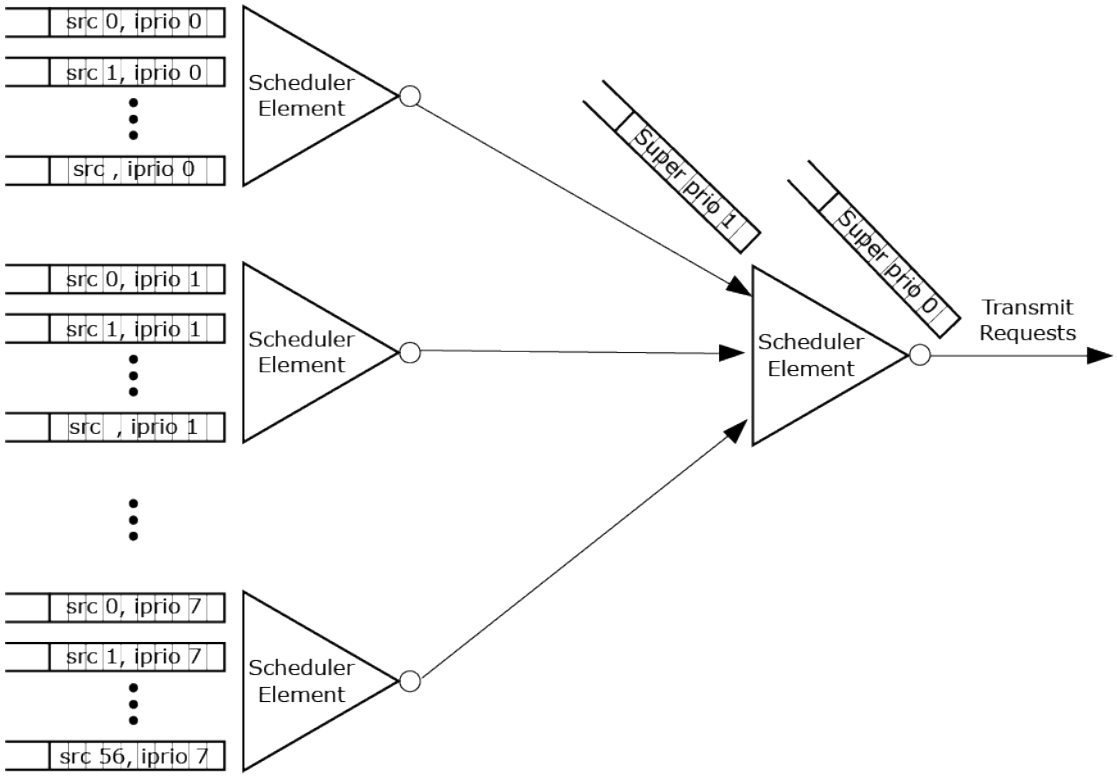
The scheduler includes a dual leaky bucket shaper to limit the traffic rate from inputs attached to it. To distribute the bandwidth between the inputs, the scheduler also contains a weighted round robin arbiter.

The number of queues available in total for all ports is 40,460. The queue mapper must be configured to how queues are assigned to each egress port. The overall traffic management for each port is the result of the queue mapping and scheduler hierarchy.

The following illustration shows an example of the default egress port hierarchy. At layer 0, eight SEs select between data in the same QoS level, with queues from each source port. This forms a basic traffic manager, where all source ports are treated fair between each other, and higher priorities are preferred. This is accomplished by having the layer 0 SEs select input in a round robin fashion, and the mid-layer select strictly higher inputs before lower inputs. In general, all SEs can be configured to various polices on how to select inputs.

The figure also shows the two super-priority queues available in all ports. The scheduler hierarchy configuration must be based on a desired scheme. The hardware default setup is providing a queue per (source, priority) per egress port. All queues with the same QoS level are arbitrated first, in layer 0.

Figure 4-74. Scheduler Hierarchy (Normal Scheduling Mode)

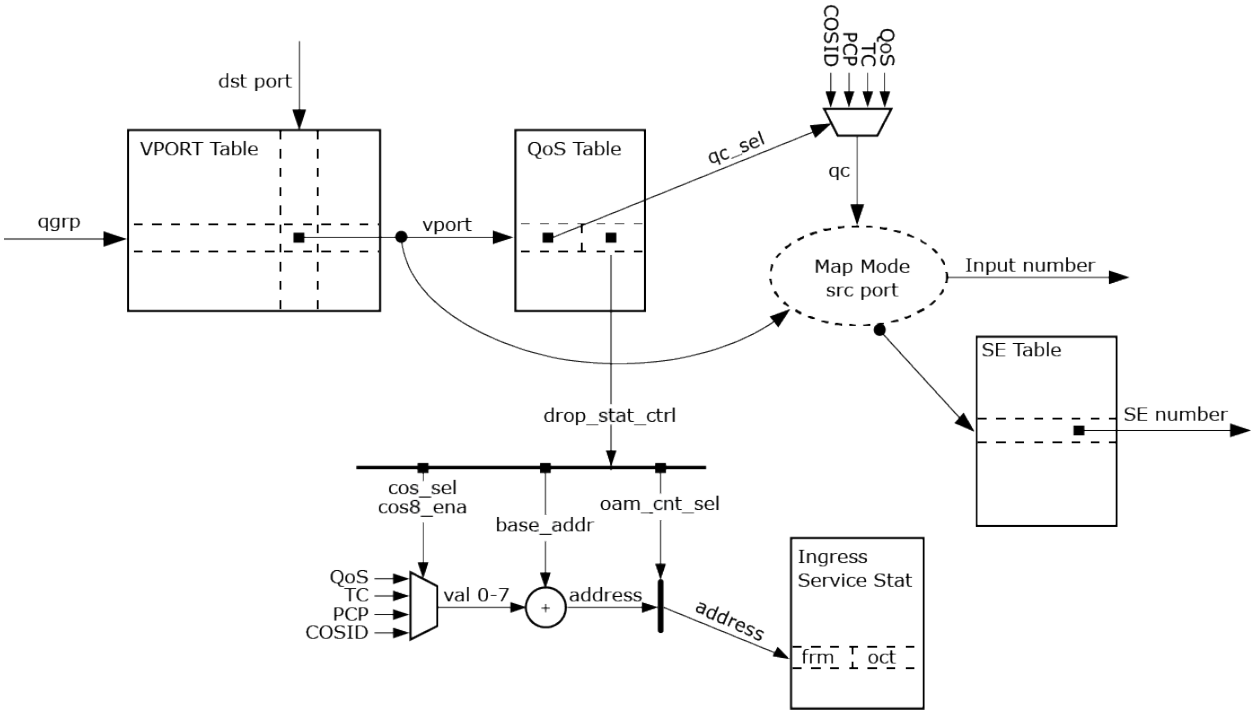


4.26.5.1 Mapping System

All queues in the system are referred to as a layer 0 SE number and an input number. For example, the default configuration puts all traffic for port 0 from source 7 in priority 0 into the queue (0,7); SE number 0, input 7. When the forwarder requests a frame to be added to a queue, the queue group—all classified priorities, the source port, and the destination port—are used to find the SE and input on it, through some configurable mapping tables. There are three basic hierarchy types, and the first part of the mapping procedure is to find

the hierarchy type in XQS::QMAP_PORT_MODE. The mode is found for frames with classified qgrp = 0 in the QMAP_MODE_NONSERVICE field and in QMAP_MODE_SERVICE for qgrp >0. Note that service and non-service mappings can be combined at the higher layers, in order to combine the two basic types of traffic in the application. An overview of the queue mapping tables is shown in the following figure. The drop statistics mode is also found by looking up information in these tables. For more information, see 4.26.9.4 Statistics.

Figure 4-75. Queue Mapping Tables



Mapping is accomplished by the following steps.

1. Finding virtual port (vport). Used only for looking up fields in other tables.

   vport = QMAP_VPORT_TBL (qgrp,dport)

2. Find queuing class

   (qc_cl)
   qc_sel = QMAP_QOS_TBL

   (vport)
   qc = (qos, cosid, pcp, tc) (qc_sel)

Find queue. This mapping depends on which of the following hierarchy modes are configured:

- Mode 0: Normal mode. This is the default mode on all ports. It is used for arbitrating all frames with the same queuing class, treating all source ports equally.

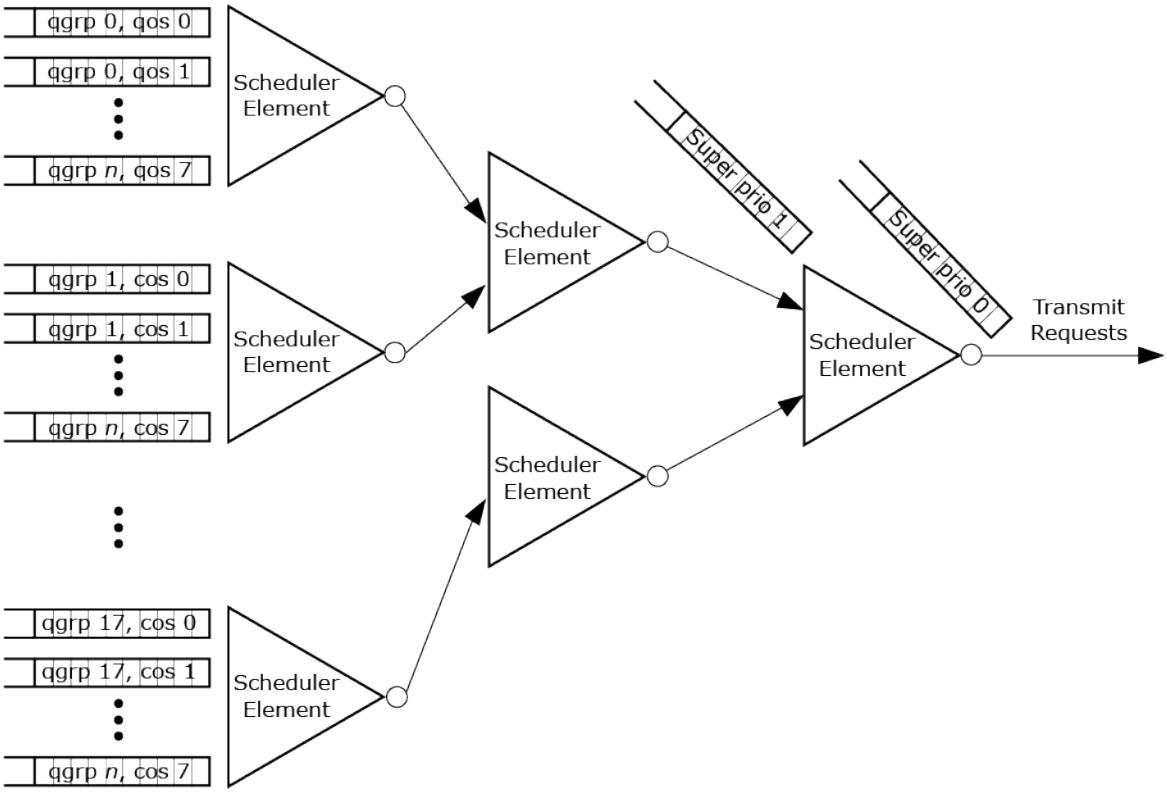
  SE = QMAP_SE_TBL (vport + qc), inp = source port.

- Mode 1: Group mode. This mode is used for having eight queues per queue group in layer 0. Used in Carrier applications where each service should have its own queues, with a traffic profile per queuing class.

  SE = QMAP_SE_TBL (vport), inp = qc.

  The following figure shows the group scheduling mode.

Figure 4-76. Group Scheduling Mode



- Mode 2: Microwave Backhaul (MBH) mode. This mode is used when many queue groups need to be arbitrated per queuing class, and they share a common outgoing quality of service policy. For example, it may be possible to have 768 queues per queuing class arbitrated on level 0 and 1, and class selected on level 2. In this case, the SE table is used by SE = QMAP_SE_TBL (vport + qc), Inp = qgrp MOD 64.

The following table lists the registers associated with assigning queues for each egress port.

Table 4-215. Queue Mapping Registers Overview

Configuration	Description
QMAP_PORT_MODE	Selects queue layer hierarchy type
QMAP_VPORT_TBL [1]	Map (qgrp,dport) into a virtual port
QMAP_QOS_TBL1	Selects per vport from which queueing QoS it should be derived
QMAP_SE_TBL1	Selects per vport a scheduling element

Note:

1. The QMAP_xx_TBLs are indirectly accessed by setting the target index in XQS:MAP_CFG_CFG.MAP_CFG_CFG following read or write to replication 0 of the register.

4.26.5.2 Service Statistics

As shown in the following illustration, drop statistics configuration is also being looked up in the mapping flow. The device has 8,192 service counter sets. On the ingress side, the queue system only counts discarded frames. On the egress side, it counts the amount of frames transmitted. In both cases, there is a counter for green frames and another counter for yellow frames.

The egress counter set to count is selected through the rewriter. The ingress counter is selected by looking up drop statistics information when the queuing class is looked up. That returns a base index, which is multiplied by four, and one of the four classified priority parameters is added. If the set is configured for using only four counters, the MSB of the priority is cleared. The resulting index will afterwards be counting drops, with 18 bits of octet counting, and 10 bits

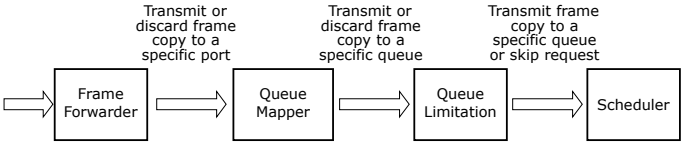
of frame counting. Optionally, all 28 bits can be used for frame counting. For more information about drop statistics, see 4.34.2.3 Statistics.

4.26.6 Queue Congestion Control

The congestion control system may choose to forward a frame to a specific queue or to discard it. In either case, the decision passes through a queue limitation function, which can alter the decision based on another view of the resources

The following figure shows the drop decision flow.

Figure 4-77. Drop Decision Flow



The queue limitation function checks the queue size against various configurable and dynamic values and may change a transmit request to a discard by canceling the request to the scheduler.

The queue limitation system uses queue sizes (in terms of packet memory usage) of all queues. Each queue is assigned to a share. The share is a configuration parameter for the port to which the queue belongs, as set in the QLIMIT_PORT_CFG register. A set of watermarks is afterwards used to control some queue limitation drop mechanisms, trying to discard data for congested queues without affecting well-balanced traffic.

The queue limitation must be enabled both for ingress and egress port in the potential forwarding. In addition, both ports must be enabled for using shared memory, otherwise only a fixed amount of memory is available for the queue. These configuration choices must be applied in QLIMIT_PORT_CFG.

A dynamic watermark in each memory share is constantly updated. It informs the current number of congested scheduling element the appropriate amount of memory requirement for each, in order for all the congested flows to have the same burst capacity through the switch. If the memory share is filled above a certain level, and the queues and SE elements use more than the fair part, frame copies will be discarded.

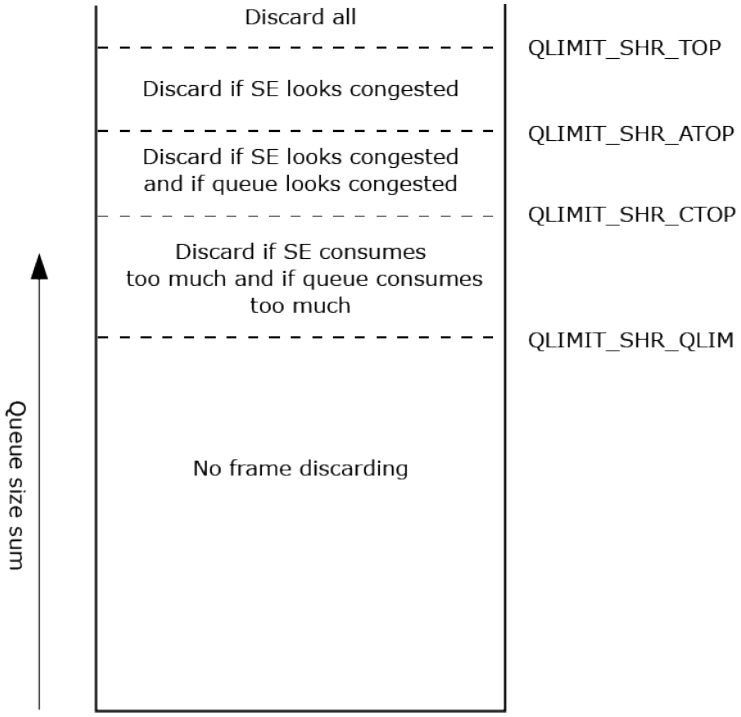
The drop precedence level is used to find a level below the fair share at which the drop condition kicks in. This is configured in QLIMIT_DP_CFG, where it also can be configured whether yellow traffic can be guaranteed a minimum consumption.

To support WRED behavior, the dynamic watermark can be added random noise with a configurable amplitude, which makes the drop probability for all flows vary randomly around the desired watermark. This is set up in QLIMIT_SHR_QDIVMAX_CFG.

To support DLB shaping, an SE can be set up to enable the EIR bucket when the filling exceeds a certain level.

The conditions under which a frame is discarded is shown in the following illustration The watermarks shown are defined per shared account and per color. If yellow traffic is present, it can be discarded fairly without affecting the green traffic.

Figure 4-78. Queue Limitation Share



The following table provides descriptions of the queue limitation conditions.

Table 4-216. Queue Limitation Conditions

State	Description	Traffic Condition
SE looks congested	The total size of queues into the layer-0 SE exceeds the QLIMIT_SE_CONG watermark.	The data on the specific SE is looking like growing, and the SE seems to receive more data than it has been granted egress bandwidth. When the filling gets very high, further data into that SE can be discarded.
Queue looks congested	The queue size exceeds the watermark in QLIMIT_QUE_CONG.	The queue seems to be growing. When the share filling is high, it can be chosen to discard further data into the queue.
SE consumes too much	The total SE use exceeds the dynamic fair share, which is the QLIMIT_SHR_QDIV watermark divided by the number of SEs looking congested. This division result is called DYN_WM.	There is a number of SEs in the scheduler building up data, and this particular SE is using more than the fair share of the buffer. Frames are discarded in the effort to grant equal burst capacities to all SEs seeking intermediate buffer.
Queue consumes too much	The queue size exceeds DYN_WM, which is the QLIMIT_SHR_CTOP watermark divided by the number of queues looking congested.	The queue seems to be one of the reasons for the growing memory consumption, and further additions to the queue can be avoided.

The following table lists the queue limitation registers.

Table 4-217. Queue Limitation Registers

Register Groups	Description
QLIMIT_QUEUE [1]	Reads current queue size.

.....continued	
Register Groups	**Description**
QLIMIT_SE[1]	Reads current SE queue sum and congestion count.
QLIMIT_CFG	Assigns ports to shares. Configures DP levels in common for all ports.
QLIMIT_SHR	Configures watermarks for share and monitor filling and dynamic watermarks.
QLIMIT_MON	Reads the status for each of the logical shares.

Note:

1. The QLIMIT_QUEUE registers are indirectly accessed by setting the target index in XQS:MAP_CFG_CFG.MAP_CFG_CFG following the read or write to replication 0 of the register

4.26.7 Scheduling

All egress queues are combined in a tree structure where the root is connected to an egress port. The scheduling capabilities consist of the following three aspects.

- Structure of the hierarchy
- Bandwidth limitation (shaping)
- Bandwidth distribution (arbitration between inputs)

4.26.7.1 Hierarchy Structure

The hierarchy consists of three layers: queue, middle, and outer.

- Layer 0 is the queue level. All inputs to scheduler elements are connected to a specific egress queue, as described in the previous section. This layer has 5,040 elements with eight inputs each. A number of those can be reconfigured to have 72 inputs, in which case they are called “large elements”. Only elements in multiples of eight can become large, so the seven following elements cannot be used as well as a.single element from index 4,480. A large element must have an index being 8 x N, and small elements (8 x N) to (8 x N)+7, and 4,480+N can not be used further.
- Layer 1 is the middle layer. There are 64 elements in this layer, all having 64 inputs. These inputs are connected to outputs of the layer 0 elements, through the HSCH::L0_CFG table.
- Layer 2 is the output layer. There is one scheduler element per egress port (70 elements). All elements have 16 inputs connected to outputs from layer 0 or 1 elements.

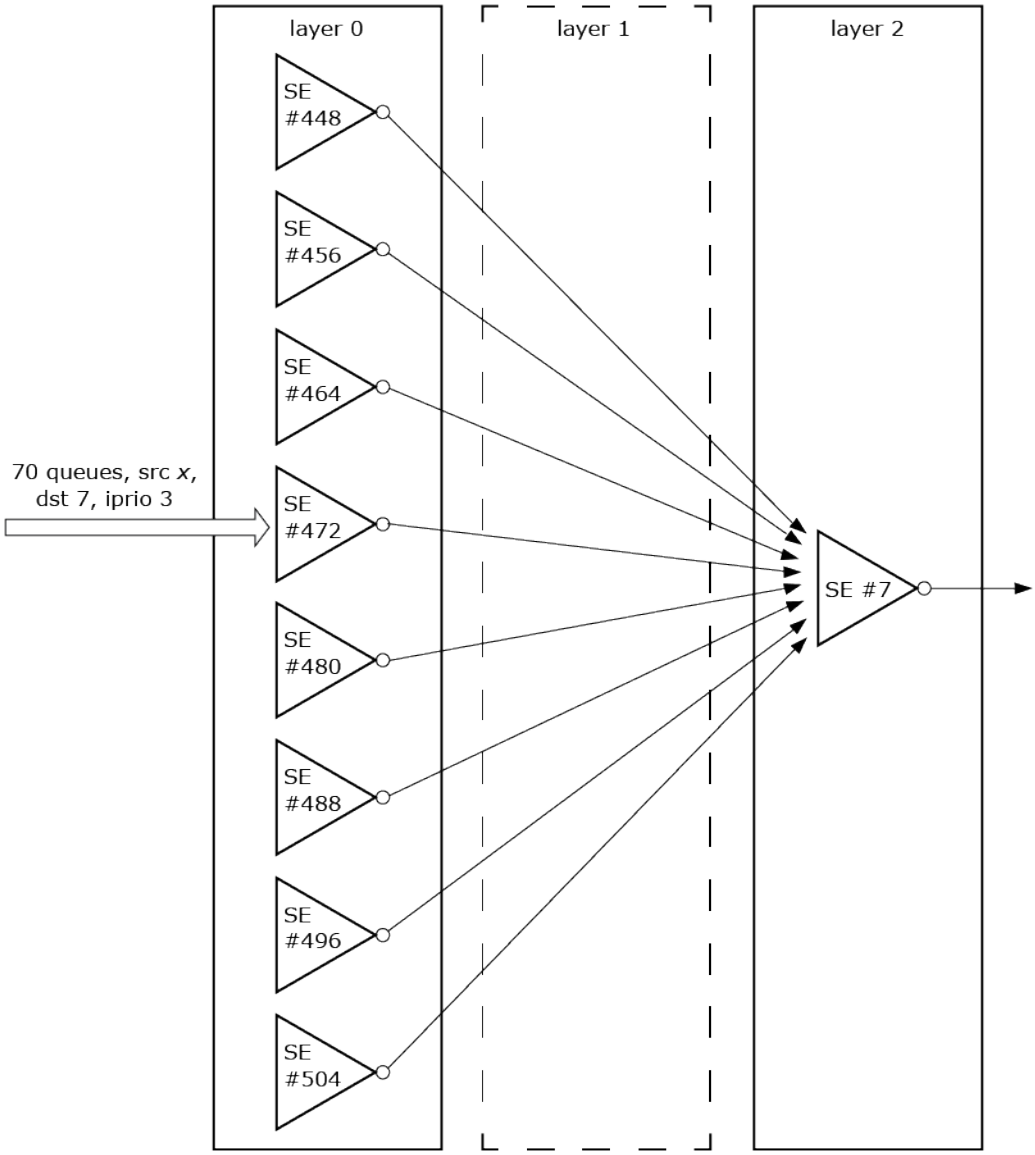
If the hierarchy needs are simple, layer 1 can be bypassed to save SE resources for other ports. It is configured in the L0_CFG mapping table.

4.26.7.2 Default Hierarchy

The initialization engine in the scheduler builds a default hierarchy for all ports, and no further configuration of it is needed. It sets all ports into normal mode, using eight large elements on layer 0, and one element on layer 2. The layer 0 elements are all configured for frame based round robin selection, and the layer 2 elements for strict selection of highest available input.

Port n uses elements 64n, 64n + 8, 64n + 16, ... for layer 0, and element n of layer 2.

Figure 4-79. Default Scheduling Hierarchy



4.26.7.3 Bandwidth Control

Each scheduler element has a dual leaky bucket shaper on its output, which may block further transmissions from that scheduler element. It is controlled through a leaky bucket system, filling the bucket when data is transmitted and leaking it at the configured rate. The bucket is closed when more data than the configured burst capacity has been filled into it.

The shaper has three rate modes that define what is added to a bucket:

- Bits transmitted excluding IFG
- Bits transmitted including IFG
- Frames transmitted

Each of the three layers have four linked lists configured. All scheduling elements with active shapers must be members of one of those. Each linked list is processed once every configurable interval. When a list is processed, all shapers in it will leak its configured rate setting.

Examples:

- Data rate shapers are connected in a linked list, which is traversed every 10 µs. The shaper rate granularity becomes 100 kbps.
- Frame rate shapers are connected in a linked list, which is traversed every 1 ms. The shaper rate granularity becomes 1 kilo frames per second.

Each scheduler element includes two leaky buckets to support DLB shaping. The committed information rate (CIR) bucket is open when the filling is below the configured limit. The excess information rate (EIR) shaper is always closed, unless the congestion control reports that a threshold is reached. Which congestion thresholds to react on are configurable for each shaper and should depend on the traffic groups the particular element is a part of. The effect of the DLB shaper is that the rate out of the shaper is CIR or CIR plus EIR, depending of the accumulation of data in the queue system.

The following table lists the registers associated with shaping.

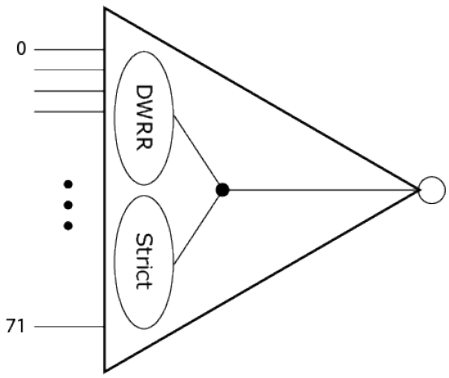
Table 4-218. Shaper Registers Overview

Register	Description
HSCH::SE_CFG	Selects filling mode
HSCH::CIR_CFG	Configures leak rate and burst capacity for CIR bucket
HSCH::EIR_CFG	Configures leak rate and burst capacity for EIR bucket
HSCH::SE_DLB_SENSE	Selects thresholds to control EIR with.
HSCH::HSCH_LEAK_LIST	Configures leak list periods and starting index
HSCH::SE_CONNECT	Links to next in leaking lists
HSCH::CIR_STATE	Returns current CIR bucket filling
HSCH::EIR_STATE	Returns current EIR bucket filling
HSCH::SE_STATE	Returns whether or not the shaper is open

4.26.7.4 Bandwidth Distribution

Each scheduler element selects one of its inputs as the next source of transmission if a scheduling process passes through it. A number of different options are available for how this arbitration takes place. A configured value for the scheduler element tells how many of the lower indexed inputs should be arbitrated with the DWRR method. Inputs with higher indexes are preferred and arbitrated strict.

Figure 4-80. Scheduler



The lower indexes are arbitrated in the effort to achieve a bandwidth ratio between them. In this regard, the bandwidth can be measured in frames, data bits, or line bits (including IFG). Each input is configured with a cost setting, and the algorithm strives to give n times more bandwidth to input A than B, if the cost configuration for A is n times lower than B.

Example: Input 4 has cost 7, and input 6 has cost 11. If the total bandwidth shaped out of the scheduler element is 180 Mbps, input 4 will get the 110 Mbps of it.

Example: All inputs have cost 1, and the element is configured for frames per second arbitration. Algorithm will by this strive to send an equal number of frames from each input.

Configuring costs is done indirectly by setting the HSCH_CFG_CFG.DWRR_IDX first (pointing to the scheduler element to configure), after which the costs are accessible through the HSCH_DWRR registers.

The following table lists the registers associated with the arbitration algorithm.

Table 4-219. Scheduler Arbitration Registers Overview

Register	Description
HSCH::SE_CFG	Selects arbitration mode (data/line/frame) and number of inputs running DWRR
HSCH::CFG_CFG	Selects for which element to configure/monitor DWRR algorithm
HSCH::DWRR_ENTRY	Configures costs for each input
HSCH::INP_STATE	Returns input states for a scheduler element

4.26.7.5 Queue Shapers

In addition to the shaping capabilities at the output of each scheduling element, it is also possible to shape individual queues. These shapers are single leaky bucket based. The device contains 6,800 such shapers that can be assigned to any of the queues. The assignment of queue shapers is done through the HSCH::QSHP_ALLOC_CFG register. This is replicated per scheduling elements in layer 0, and defines the range of inputs on the element that should have queue shapers assigned, and the index of the first of the 6800 shapers to allocate to it. The shapers must be linked together for the leaking process in HSCH::QSHP_CONNECT.

For example, to configure queue shapers on inputs 5 to 7 on elements 100 to 120, use the queue shapers from 400 to 462.

```
For I in 100 to 120 loop
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_ALLOC_CFG.QSHP_MIN := 5
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_ALLOC_CFG.QSHP_MAX := 7
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_ALLOC_CFG.QSHP_BASE := 3*(I-100)+400
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_CONNECT.SE_LEAK_LINK := I+1;
Endloop
(terminate chain)
HSCH:QSHP_ALLOC_CFG[120]:QSHP_CONNECT.SE_LEAK_LINK := 120
```

The leak process operates as for the SE shapers and must be configured in HSCH:HSCH_LEAK_LISTS[3].

The indexes of shapers allocated to a set of queues are only used in the QSHP_BASE settings. The shapers are accessed afterwards through the QSHP_CFG and QSHP_STATUS registers, accessed indirectly after setting HSCH_CFG_CFG.CFG_SE_IDX to the desired scheduling element.

For example, to configure the shaper on input 6 of element 118 to shape to 1.2 Mbps, set HSCH::HSCH_CFG_CFG.CFG_SE_IDX := 118 and HSCH:QSHP_CFG[6].QSHP_CIR_CFG.CIR_RATE := 12 (assuming the leak chain is configured for unit 100 kbps).

4.26.7.6 Priority-Based Flow Control

The scheduling hierarchy supports egress priority-based flow control (PFC) where the egress scheduling decisions can obey what the link partner has sent of PFC flow control frames. In HSCH_MISC:PFC_CFG it is configured where in the hierarchy PFC frames from each port should be applied. As an example with the hardware default hierarchy, the SE at which all priorities are arbitrated for port N, is the layer 2 element N. The PFC_CFG for port N should therefore be configured to 2:N. PFC must also be enabled in the ASM where the PFC frames received are processed (ASM::PFC_CFG).

4.26.7.7 Preemption Control

A frame scheduled for transmissions is associated with a flag telling whether the frame should be preempted. If the scheduler has initiated a transmission which can be preempted and another frame transfer not being preemptible (an express frame) becomes available, the express frame will preempt the current transmission and start as soon as the egress path has completed a valid fragment of the current frame. Whether a transmission is preemptible or not is configure per queue in HSCH_L0_CFG. If the SE the queue is attached to is large, all or none of the queues associated with it must be preemptible.

4.26.7.8 Miscellaneous Scheduler Features

The following table lists some features that might be useful.

Table 4-220. Miscellaneous Scheduler Registers Overview

Field	Description
LEAK_DIS	Disable leaking process.
FRM_ADJ	Sets the byte difference between line and data rate calculations.
DEQUEUE_DIS	Disables transmissions per port.

4.26.8 Queue System Initialization

The queue system automatically initializes all tables and data structures when setting RAM_CTRL.RAM_ENA to 1, followed by setting RAM_CTRL.RAM_INIT to 1. Software should afterwards wait for 150 µs or wait for the RAM_INIT bit to be cleared by hardware. After that the RESET_CFG.CORE_ENA must be raised, and the queue system is ready to operate. All thresholds and queue mappings are prepared with operational values. Configuration can afterwards be modified. The only per-port setting required to change in the queue system for initial bring-up of the queue system is the SWITCH_PORT_MODE.PORT_ENA, which must be set to one allowing switching of frames to and from the port.

4.26.9 Miscellaneous Features

This section provides information about other miscellaneous features of the queue system.

4.26.9.1 Frame Aging

The queue system supports discarding of frames pending in the queue system for too long time. This is known as frame aging. The frame aging period is configured in FRM_AGING.MAX_AGE, which controls the generation of a periodical aging tick.

The packet memory manager marks frames pending in the queue system when the aging tick occurs. Frames pending for more than two aging ticks are subject to frame aging and are discarded by the queue system when scheduled for transmission. As a consequence, frames can be discarded after pending between one or two aging periods in the queue system.

The following table lists the configuration registers associated with frame aging.

Table 4-221. Frame Aging Registers Overview

Field	Description	Replication
QSYS::FRM_AGING.MAX_AGE	Sets the aging period	None
HSCH::PORT_MODE.AGE_DIS	Disables aging checks per egress port	Per port
HSCH::PORT_MODE.FLUSH_ENA	Regards various frames as being too old	None

4.26.9.2 Super Priorities

There are two special queues available in each egress port: super priority queue 0 and 1. For more information, see Figure 4-74. These queues are selected before any other queues. They are intended for urgent protocol traffic injected by the CPU system. Frames in super priority queue 0 disregard the output shaper state and are not counted in the shaper's buckets. Frames in super priority queue 1 obey the output shaper state and are counted in the shaper's buckets.

The following frames can be added to the super priority queues.

- AFI injected frames: The AFI can be configured to inject into either of the super priority queues. For more information, see 4.27.6 Adding Injection Frame.
- CPU injected frames: Frames injected with an IFH with the SP flag set are inserted into the super priority queue as being set in the drop precedence (DP) IFH field.
- CPU extracted, mirror, learn all frames: Frames subject to the translation in QFWD::FRAME_COPY_CFG can configure use of either of the super priority queues.

4.26.9.3 Frame Modifications in the Queue System

The queue system typically passes data completely untouched from the ingress to the egress side, except for a few special cases.

Frame copies to the CPU can be sent into the rewriter with two different priority schemes, involving modification of the internal frame header. Either the frame takes the QoS class from the CPU extraction queue number or it takes the QoS class from the configuration in QFWD::FRAME_COPY_CFG. This is controlled in PORT_MODE.CPU_PRIO_MODE.

Frame copies due to learn-all stack synchronization can be truncated to 64 bytes to save bandwidth on the line. This is enabled in HSCH::PORT_MODE.TRUNC_ENA.

4.26.9.4 Statistics

The queue system counts frame discards and successful transmissions. Each counted event updates a counter instance, counting octets in a 40-bit counter and frames in a 32-bit counter. A frame is counted as discarded when none of the frame's destination ports are reached, excluding mirror and learn-all copies. If a frame is discarded, a counter instance, per ingress port, per QoS class per color, is updated. The color granularity here is only green or yellow, mapped from the analyzed DP value through the QSYS::DP_MAP settings. A frame transmission is a frame leaving the queue system towards the rewriter. In this direction, a counter instance per egress port, per QoS class per color, is updated. An abort counter instance per egress port counts frames discarded due to frame aging, queue system flushing, or abort requests from the rewriter.

There are statistics available per service counter index, which is provided by the analyzer for ingress service discard statistics and from the rewriter for egress service transmission statistics. The service counters are available per counter index, per color.

Access to the statistics is indirect. The XQS::STAT_CFG configuration register must be set to the port or service counter index to be accessed, and all counters related to this index can afterwards be read in the XQS::CNT registers. The following table lists the available port counters and how they are mapped. Unused addresses return undefined values. In the expressions, yel = 1 for reading yellow counters and yel = 0 for green. The mapping from classified DP to color is made in the QSYS::DP_MAP register.

The following table shows the addresses of the frame counters. The equivalent octet counters are found at the frame counter address plus 64 for the 32 least significant bits and at the frame counter address plus 128 for the eight most significant bits.

Table 4-222. Statistics Address Mapping

Statistics Index	Event
0–15	Frames received on the port viewed. The first eight counters are for green frames, QoS levels 0–7. The last eight counters are for yellow frames, defined by the QSYS::DP_MAP register.
16–31	Frames dropped on the port viewed. Drops are for the ingress port point of view or egress, depending on the DROP_COUNT_EGRESS option. There are eight green counters and eight yellow counters.
32	Frames dropped on the port viewed due to no destinations.
256–271	Transmitted frames on the port viewed. Same layout as for the receive set.
272	Transmit aborted frames for the port viewed. This can be due to aging, flushing, or abortion decision by the rewriter.
512 + 513	Green and yellow drops for service counter viewed.
768–769	Green and yellow transmissions for service counter viewed.

The service counters are reduced in size; 10 bits for the frame part and 18 bits for the octet part. However, STAT_CFG.STAT_SRV_PKT_ONLY can be set to use all 28 bits for frame counting.

All the counters wrap at their maximum capacity. The STAT_CFG.STAT_WRAP_DIS, however, can be set to change the behavior to clear on read and to saturate at the maximum value.

Tail drops done by the buffer control are accessible in the QSYS::MMGT_TAIL_DROP registers.

The following tables lists the registers associated with statistics access.

Table 4-223. Statistics Registers Overview

Register	Description	Replication
XQS::STAT_CFG	Selects which counter set index to access. Clear counters.	None
XQS::CNT	Replicated register with statistics access.	1,024
QSYS::DP_MAP	Maps DP levels to color.	None

Examples:

To read number of transmitted yellow frames for service counter 714:

1. Write 714 to STAT_VIEW in XQS::STAT_CFG.
2. Read XQS::CNT[385].

To read number of discarded green octets with QoS class 4 on port 9:

1. Write 9 to STAT_VIEW in XQS::STAT_CFG.
2. Read XQS::CNT[76] for the least significant bits.
3. Read XQS::CNT[140] for the most significant bits.

To read number of aborted frames on port 53:

1. Write 53 to STAT_VIEW in XQS::STAT_CFG.
2. Read XQS::CNT[272].

To read all counters for port 19:

1. Write 19 to STAT_VIEW in XQS::STAT_CFG.
2. Read XQS::CNT[0:15, 256:272].
3. Add 64 and 128 to above indices for octets.

4.26.9.5 Energy Efficient Ethernet (EEE) Control

The front ports are able to signal low power idle (LPI) towards the MACs, in case they request the PHYs to enter low power mode. A controller in each port module assures that frames are not transmitted when the PHYs are powered down. This controller powers the PHY down when there is nothing to transmit for a while, and it powers up the PHY when there is either data ready for a configurable time or when the queue system has urgent data ready requiring transmission as soon as possible.

The following table lists the registers with configuring EEE.

Table 4-224. EEE Registers Overview

Register	Description	Replication
QSYS::EEE_CFG	Configures the priorities per port which should bring the port alive as fast as possible.	Per port
QSYS::EEE_THRES	Configures the amount of cells or frames pending for a priority before the port should request immediate power up.	Per port

Example: Configure the device to power up the PHYs when 10 frames are pending.

Only ports 3 through 6 should regard their data as urgent and only for priorities 6 and 7. All other data should get transmitted when the EEE controller in the port has seen pending data for the controller configured time.

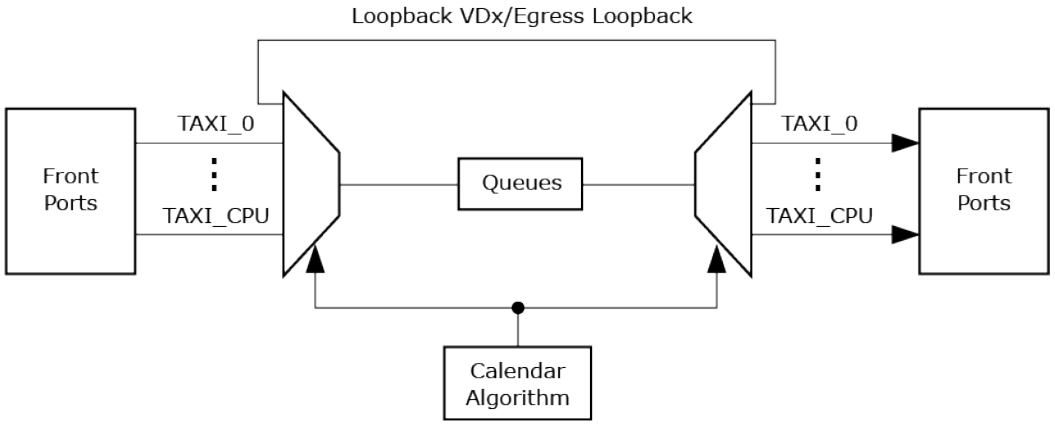
1. Write 10 to QSYS::EEE_THRES:EEE_HIGH_FRAMES.
2. Write 0xC0 to QSYS::EEE_CFG[3–6]::EEE_FAST_QUEUES.

4.26.9.6 Calendar Setup

There are a number of logical ports, all needing bandwidth into and out of the queue system. This is controlled through a calendar module that grants each port access to internal busses.

The following illustration shows an overview of the internal busses. All ports are connected to a port data bus (taxi bus) and granted access to the chip core through a mux obeying the calendar decision. Similarly, on the path towards the transmit interface, a mux controls which port transmits the next time.

Figure 4-81. Internal Bus



The front ports are connected to taxi busses and must be guaranteed bandwidth corresponding to their line speed.

In any port configuration, a subset of the ports is enabled. The calendar algorithm is the central place where bandwidth is distributed between the ports. The calendar algorithm can be controlled by two methods: automatic or sequenced.

In the automatic calendar configuration, a configuration per port defines how much egress bandwidth is granted to it, and that must not be lower than the physical interface speed. The CAL_AUTO registers set the granted speed for each port in the range 0-10 Gbps.

In the sequenced operation, a specific sequence of port numbers forms the calendar. The port sequence controls which port is granted the cycle, per bus cycle (two system clock periods).

The algorithm mode is configured in CAL_CTRL.CAL_MODE. It is possible through this register to program a manual sequence, to halt the calendar, and to select between sequenced and automatic calendars.

Both calendar modes may leave some slots unallocated and therefore idle on the busses. These idle cycles can be used for configuration access (which will wait for unused cycles) and other slow operations in the system (MAC table scan, for example). Slots allocated for a port and not used due to lack of data to transfer can be used by the internal ports through the HSCH::OUTB_* registers, where it is, per internal port, configured how often it will seek granting of an unused cycle. Setting internal CPU port 0 to seek access every 100 cycles means that it can get up to one frame transfer every 100 x 1.6 ns, or approximately 5 Gbps.

In the ingress side, the loopback paths use idle cycles to feed frame data into the ingress path.

The following table list the registers associated with configuring the calendar function.

Table 4-225. Calendar Registers Overview

Register	Description
CAL_CTRL	Calendar function mode. Configures auto-grant rate.
OUTB_SHARE_ENA	Enables granting unused bus cycles to the internal ports (CPU and VD) on the egress side.
OUTB_CPU_SHARE_ENA	Configures bandwidth sharing between the two internal CPU ports.
CAL_SEQ	Interface to the sequence memory. See register list.
CAL_AUTO	Per port setting with requested bus bandwidth.

4.26.10 Cut-Through Forwarding

By default, all frames are fully stored and verified for correct FCS before being forwarded to any egress queues. A frame transfer can be subject to cut-through forwarding where the transfer starts when some minimum amount of frame data is received.

The cut-through forwarding takes place if the following conditions are fulfilled.

- The egress port and priority must have cut-through forwarding enabled. This is set in XQS::FWD_CT_CFG.
- The ingress port and priority must have cut-through forwarding enabled, which is set in QSYS::IQUEUE_CFG
- The egress port must not operate at a higher bit rate than the ingress port. These speeds are enumerated in XQS:FWD_CT_CFG where each port are configured a speed setting. Cut-through forwarding cannot take place if the egress value is higher than the ingress value.
- The scheduling element to which the traffic is enqueued must not have a higher speed setting than the ingress port's speed setting (same enumeration as for the previous bullet). By default, this condition is disabled.

4.27 Automatic Frame Injector

The automatic frame injector (AFI) provides mechanisms for periodic injection of PDUs.

The following blocks are involved in frame injection.

- QSYS: The frames are sent into the queue system by the CPU and stored in the queue system until deleted by software.
- AFI: Using timers and other parameters, AFI controls the automatic frame injection.
- REW: For outbound data path injections, the REW performs frame modifications.
- Loopback path: The loopback path (VD1) between the disassembler and assembler is used to inject frames into the inbound data path.
- ANA: For inbound data path injections, the ANA performs modifications of frames.

AFI supports two types of automatic frame injection: timer triggered (TTI) and delay triggered (DTI).

- Timer Triggered Injection (TTI): Frame injection rate is controlled by timers with typical values of approximately 3 ms or higher. Only single-frame injections are supported; injection of sequences of frames is not supported.

  For each frame, jitter can be enabled for the associated injection timer.

  Timer triggered injection can be combined with delay triggered injection.
- Delay Triggered Injection (DTI): A sequence of frames, including configurable delay between frames, is injected.

  Delay triggered injection can be combined with TTI, such that DTI is used to inject a (high) background load (imitating normal user traffic), and TTI is used to inject OAM PDUs for measuring IR, FLR, FTD, and FDV.

  AFI supports up to 32 active DTI frame sequences at a time.

4.27.1 Injection Tables

The main tables used to configure frame injection are frame table, DTI table, and TTI table.

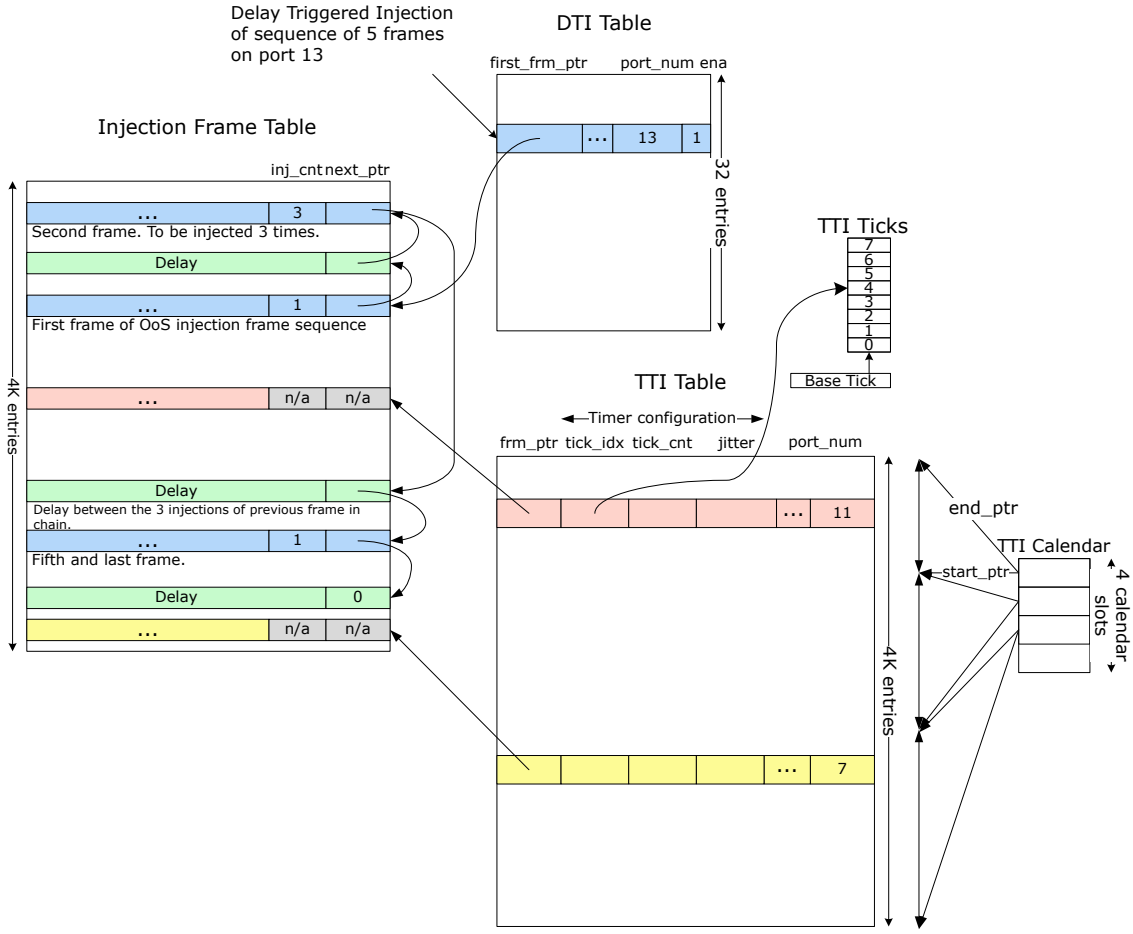
Injection frames for both TTI and DTI are configured in the frame table. The frame table itself does not hold the frame, but rather a pointer to where the frame is located in the buffer memory within QSYS.

For DTI, frames can be configured into a linked list of frames to be injected. Each frame in the linked list may be configured to be injected multiple times before proceeding with the next frame in the chain. Each frame in the sequence is followed by one or more delay entries, specifying the delay between injections.

Timers for each TTI frame are configured in the TTI table. TTIs are always single frame injections, so the inj_cnt and next_ptr fields in the frame table are unused for TTI. Delay entries are not used for TTI.

The TTI calendar controls the frequency with which different parts of the TTI table are serviced.

Figure 4-82. Injection Tables



4.27.2 Frame Table

The following table lists the registers associated with configuring parameters for each entry in the frame table.

Table 4-226. Frame Table Entry Register Overview (4,096 entries)

Register	Field	Description
FRM_NEXT_AND_TYPE	NEXT_PTR	Next entry in table. Does not apply for TTI.
FRM_NEXT_AND_TYPE	ENTRY_TYPE	Entry type: Frame or Delay entry. Does not apply for TTI.
ENTRY_TYPE=Frame FRM_ENTRY_PART0	INJ_CNT	Number of times to inject frame. Does not apply for TTI.
	FRM_RM	Frame removal. See 4.27.9 Removing Injection Frames.
	FRM_GONE	Frame removal.
	FRM_INFO	Frame information. See MISC:NEW_FRM_INFO.FRM_INFO.
ENTRY_TYPE=Delay FRM_ENTRY_PART0	DELAY	Delay between injection of start of frames. Unit is one system clock cycle. Does not apply for TTI.

4.27.3 Delay Triggered Injection

Delay triggered injection supports up to 32 flows at a time. For each DTI flow, the DTI table holds a pointer to a sequence of frames and delay entries, as well as the queue and port number, into which the frames are injected. The NEXT_PTR field of the frame table is used to link frames and delay entries together into a sequence. The frame sequence can have any length, only limited by the size of the frame table.

Each DTI sequence can be configured to be injected repeatedly until stopped, or injected a specified number of times. For example, if the frame sequence consists of five frames with INJ_CNT = 1, and the sequence is configured to be injected 1000 times, then a total of 5000 frames will be injected.

DTI can be used in conjunction with TTI, for example, using TTI to inject OAM PDUs for measuring performance metrics of the service.

The following table lists the registers associated with configuring the parameters for each of the 32 entries in the DTI table.

Table 4-227. DTI Table Entry Registers

Register	Field	Description	Replication
DTI_MODE	MODE	Mode of DTI instance.	Per DTI
DTI_MODE	TRAILING_DELAY_SEQ_CNT	Only applies “trailing delay” for every Nth sequence injection. See 4.27.3.3 Fine Tuning Bandwidth of Multiframe Sequence.	Per DTI
DTI_FRM	FIRST_FRM_PTR	Pointer to first frame in frame sequence.	Per DTI
DTI_FRM	NEXT_FRM_PTR	Pointer to next frame in frame sequence.	Per DTI
DTI_CNT	CNT	Counter (mode dependent).	Per DTI
DTI_PORT_QU	PORT_NUM	Port number on which injection queue transmits.	Per DTI
DTI_PORT_QU	QU_NUM	Injection queue. See 4.27.5 Injection Queues.	Per DTI
DTI_DURATION	DURATION	Duration of last DTI run	Per DTI

Table 4-228. Miscellaneous DTI Parameters

Register	Field	Description	Replication
DTI_CTRL	BW	Bandwidth of DTI flow 0: < 5 Gbps 1: >= 5 Gbps	Per DTI
DTI_CTRL	ENA	Enables DTI	Per DTI

Frame entries in a DTI sequence may point to the same frame in the queue system’s buffer memory. For example, shown in the following illustration, frame A and frame B shown may point to the same frame in the buffer memory.

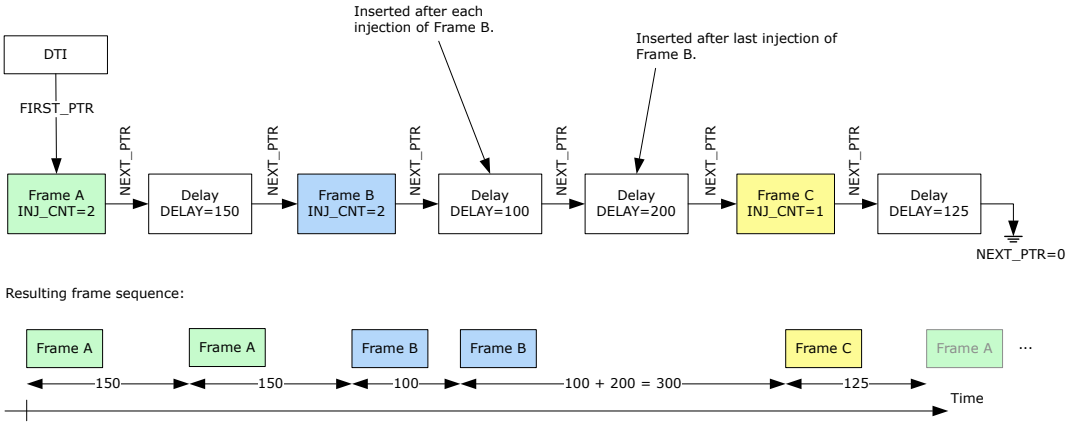
The delay entries in a DTI sequence must be configured such that they are longer than the transmission time (given the port speed) of the preceding frame in the sequence.

If a DTI sequence is configured with two consecutive frame entries, then these will be injected with a minimum delay of approximately 14 clock cycles.

4.27.3.1 Frame-Delay Sequence

A DTI flow typically consists of a sequence of alternating frame/delay entries in the frame table. If a given frame is to be injected multiple times (FRM_ENTRY_PART0.INJ_CNT>1), then the delay of the following entry in the sequence is applied after each injection. If additional delay is required after the last injection, then this can be achieved by an additional delay entry. The use of delay entries is shown in the following illustration.

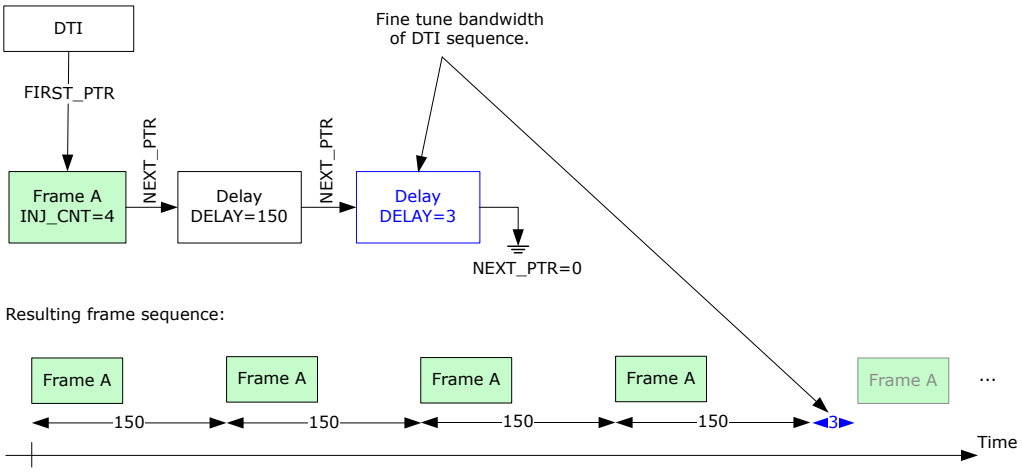
Figure 4-83. DTI Frame/Delay Sequence Example



4.27.3.2 Bandwidth Fine Tuning

The combination of INJ_CNT and two following delay entries can be used to fine tune the rate generated by a DTI flow. This is shown in the following figure, where a small delay after the four injections of frame A is used to fine tune the rate of the DTI flow.

Figure 4-84. Fine Tuning Bandwidth of DTI Sequence



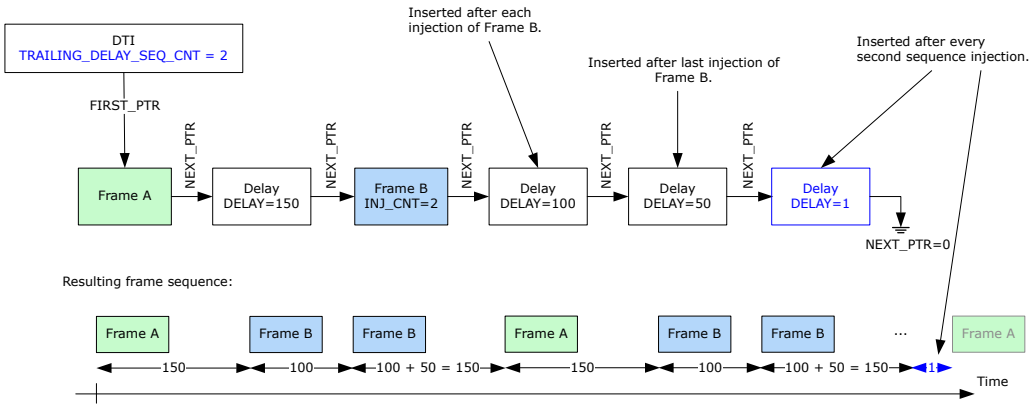
By combining the setting of INJ_CNT and the DELAY value of the last delay depicted, the bandwidth of the DTI sequence can be controlled with high granularity, because the additional (blue) delay will only be applied for every INJ_CNT injections of frame A.

The bandwidth adjustment mechanism shown may be insufficient for controlling the bandwidth of multi-frame sequences (sequences with different frames), because the last delay will be applied for every injection of the frame sequence.

4.27.3.3 Fine Tuning Bandwidth of Multiframe Sequence

For multiframe sequences, additional fine tuning of the bandwidth of the entire sequence can be achieved by configuring the last delay in the sequence, also known as the “trailing delay”, to only be applied for every Nth injection of the sequence. This is achieved using the TRAILING_DELAY_SEQ_CNT parameter and is shown in the following figure.

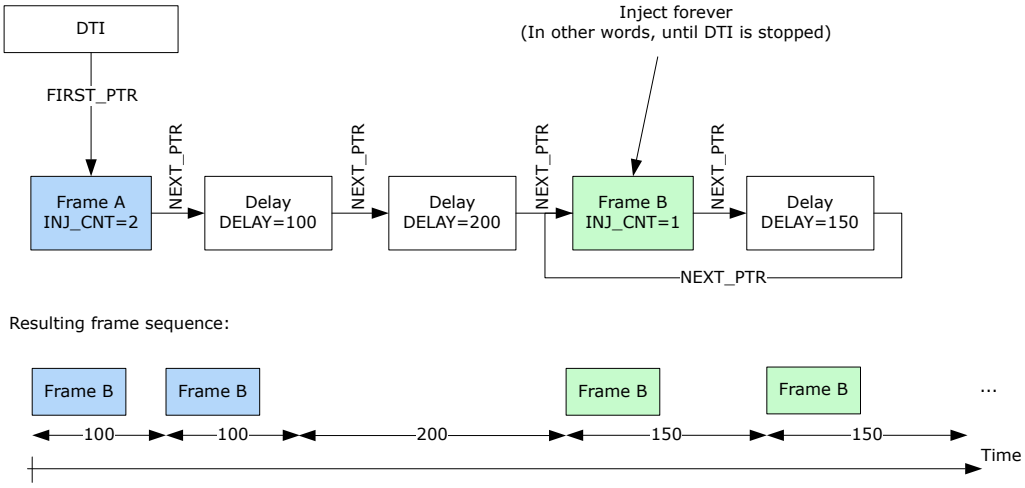
Figure 4-85. Fine Tuning Bandwidth of Multiframe DTI Sequence



4.27.3.4 Burst Size Test Using Single DTI

If a NEXT_PTR is set to point back to a previous entry in the sequence, then injection will continue forever, that is, until the DTI is stopped. This is depicted in the following illustration. The configuration shown can be used for leaky bucket burst size testing, where the first part of the sequence empties the bucket, followed by a steady frame flow with the rate of the leaky bucket.

Figure 4-86. DTI Frame/Delay Sequence Using Inject Forever

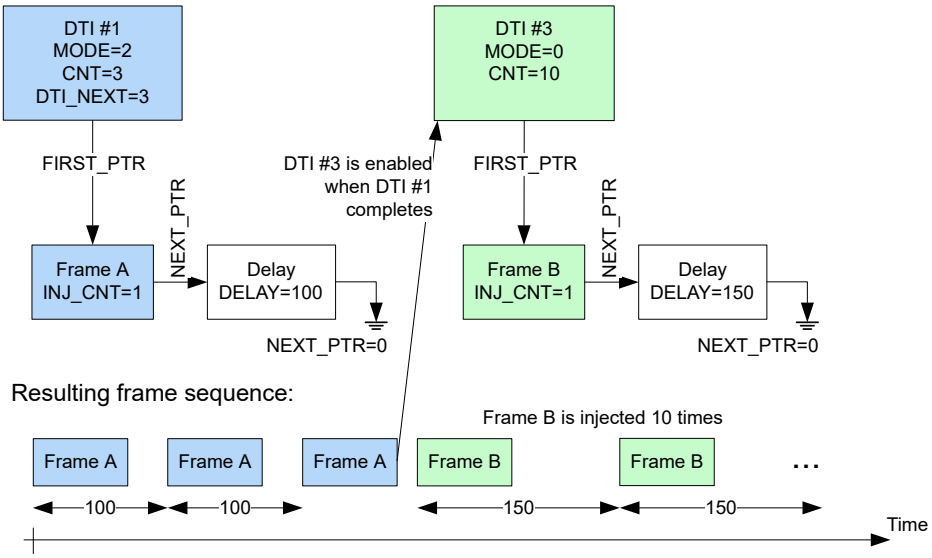


4.27.3.5 Burst Size Test Using DTI Concatenation

Two (or more) DTIs can be concatenated, such that when one DTI completes, another DTI is activated. This is depicted in the following illustration. It can be used for leaky bucket burst size testing, where the first part of the sequence empties the bucket, followed by a steady frame flow with the rate of the leaky bucket.

Note that when DTI concatenation is used, the last delay entry in the frame-delay sequence of the first DTI is not awaited before starting the next DTI. That is, the next DTI is started when the first DTI reads an entry with NEXT_PTR = 0. As shown in the illustration, the delay between the last Frame A and the first Frame B will not be 150 clock cycles, but instead approximately 12 clock cycles. If this is considered problematic, it can be addressed by using an additional DTI with a sequence consisting only of two delay entries. For example, DTI #2 could be inserted between DTI#1, and DTI#3 and DTI#2 could then be configured with a sequence consisting of two delay entries, with the first having delay = 150.

Figure 4-87. DTI Concatenation



4.27.3.6 DTI Bandwidth

The maximum bandwidth supported for DTI flows depend on

- Clock frequency
- Frame size
- Available bandwidth on the cell bus (that is, port configuration dependent)

For calculations based on clock frequency and frame size, see the description of AFI:DTI_TBL in the register list.

The bandwidth, which a DTI is injecting into a queue, must not exceed the bandwidth, which HSCH is transmitting from the queue, since then the port's FRM_OUT_MAX will then be reached, thus possibly affecting other DTIs injecting on the same port. For more information, see 4.11 Pipeline Points. The only exception to this is, if there is only one DTI injecting on the port and thus no other DTIs, which can be affected.

4.27.3.7 DTI Sequence Processing Time

For high bandwidth DTI flows, the time it takes to process the delays and frames in a DTI sequence must be taken into account:

- Processing a frame entry with no succeeding delay entry takes 12 clock cycles.
- Processing a frame entry with a succeeding delay entry takes 12 clock cycles.
- Processing a delay entry, which does not succeed a frame entry takes 12 clock cycles.

For example, a sequence consisting of a frame entry followed by two delay entries will take 24 clock cycles to process. If the frame is a 64-byte frame, and if the clock period is 4 ns, this will correspond to a maximum flow bandwidth of approximately 7 Gbps ((64 + 20) × 8 bits/(24 × 4 ns) = 7 Gbps), regardless of the delay values configured for the two delays.

Although it takes 12 clock cycles to process a delay entry, it is valid to configure a delay entry with a delay smaller than 12 clock cycles, because this will be compensated for when generating the delay between later frames in the sequence. For more information, see Figure 4-85. For example, the delay between the rightmost frame B and the succeeding frame A will be 150 + 12 clock cycles (instead of 150 + 1). To compensate for this, however, the delay between the succeeding two frames A will be 150 – 11 clock cycles.

4.27.4 Timer Triggered Injection

For TTIs, frame injection is controlled by timers. To control the time between injections, configure the following registers for each TTI:

- TTI_TIMER.TICK_IDX. For each TTI, one of eight configurable timer ticks shown in the following table must be chosen.
- TTI_TIMER.TIMER_LEN. The number of timer ticks that must elapse between each injection.

That is, the period between each injection becomes tick_period × TIMER_LEN.

The following table lists the registers associated with TTI timer tickers.

Table 4-229. TTI Timer Ticks Registers Overview

Register	Field	Description	Replication
TTI_TICK_LEN_0_3 TTI_TICK_LEN_4_7	LEN0 - LEN7	Length of timer ticks. The length of each of the eight timer ticks is specified as multiples of the length of the preceding timer tick (that is, <tick_idx>-1). Timer 0 is the shortest timer tick, and timer tick 7 is the longest. Tick 0 is specified in multiple of TTI_TICK_BASE (default 52 µs). Default configuration: tick_idx tick_len Tick_Period 0 1 52 µs (1 × 52 µs) 1 8 416 µs (8 × 52 µs) 2 8 3.3 ms (8 × 416 µs) 3 3 10 ms (3 × 3.3 ms) 4 10 100 ms (10 × 10 ms) 5 10 1 second (10 × 100 ms) 6 10 10 seconds (10 × 1 sec) 7 6 1 minute (6 × 10 sec)	8
TTI_TICK_BASE	BASE_LEN	Length of TTI base tick in system clock cycles.	1

The following table lists the registers associated with configuring parameters for each of the 4K entries in the TTI table.

Table 4-230. TTI Parameters (4096)

Register	Field	Description	Replication
TTI_PORT_QU	PORT_NUM	Port number on which injection queue transmits.	Per TTI
TTI_PORT_QU	QU_NUM	Injection queue. See 4.27.5 Injection Queues.	Per TTI
TTI_TIMER	TICK_IDX	Which of the eight timer ticks is used by the timer controlling the injection of this frame.	Per TTI
TTI_TIMER	JITTER_MODE	Enables jitter for the number of timer ticks between each injection of this frame. 0: No jitter. 1: Timer is set to a random value in the range [TIMER_LEN*0.75; TIMER_LEN]. 2: Timer is set to a random value in the range [TIMER_LEN*0.50; TIMER_LEN]. 3: Timer is set to a random value in the range [1;TIMER_LEN].	Per TTI

.....continued			
Register	**Field**	**Description**	**Replication**
TTI_TIMER	TIMER_LEN	Number of timer ticks between each injection of this frame. 0: Disables timer. 0xff: Injects frame next time the entry is serviced. After servicing, hardware sets TIMER_LEN to 0. This is intended to be used during removal of frame from buffer memory. See 4.27.9 Removing Injection Frames.	Per TTI
TTI_FRM	FRM_PTR	Points to the frame (in frame table), which the TTI injects.	Per TTI
TTI_TICKS	TICK_CNT	Number of ticks until next injection. Should be set to a random value in the range 1-TIMER_LEN before starting TTI.	Per TTI
TTI_MISC_CFG	INJ_CNT_ENA	Enables counting of injected frames in TTI_MISC:TTI_INJ_CNT.TTI_INJ_CNT.	Per TTI

An entry in the TTI table is checked approximately every two clock cycles. If the TTI's tick (configured through TICK_IDX) has changed, then the TICK_CNT is decremented. If it becomes zero, the frame is injected, and TICK_CNT is set to TIMER_LEN.

4.27.4.1 TTI Calendar

In default configuration, TTI table entries are serviced from 0-4095, then restarted again at entry 0.

When configuring the TTI table, the following must be considered.

- It takes up to 4 clock cycles, worst-case, to service an entry in the TTI table.
- A TTI must be serviced more frequently than the frequency of the TTI's tick (see TTI_TIMER.TICK_IDX).

This means that looping through the entire TTI table may take 4096 × 4 = 16384 clock cycles. With a clock period of for instance 4 ns, this corresponds to approximately 66 µs. In other words no TTI must use a tick faster than 66 us.

If TTIs faster than 66 µs are required, the TTI calendar can be used to have some entries in the TTI table serviced more often than others, meaning the TTI table can be split into sections with TTIs using ticks of different speeds.

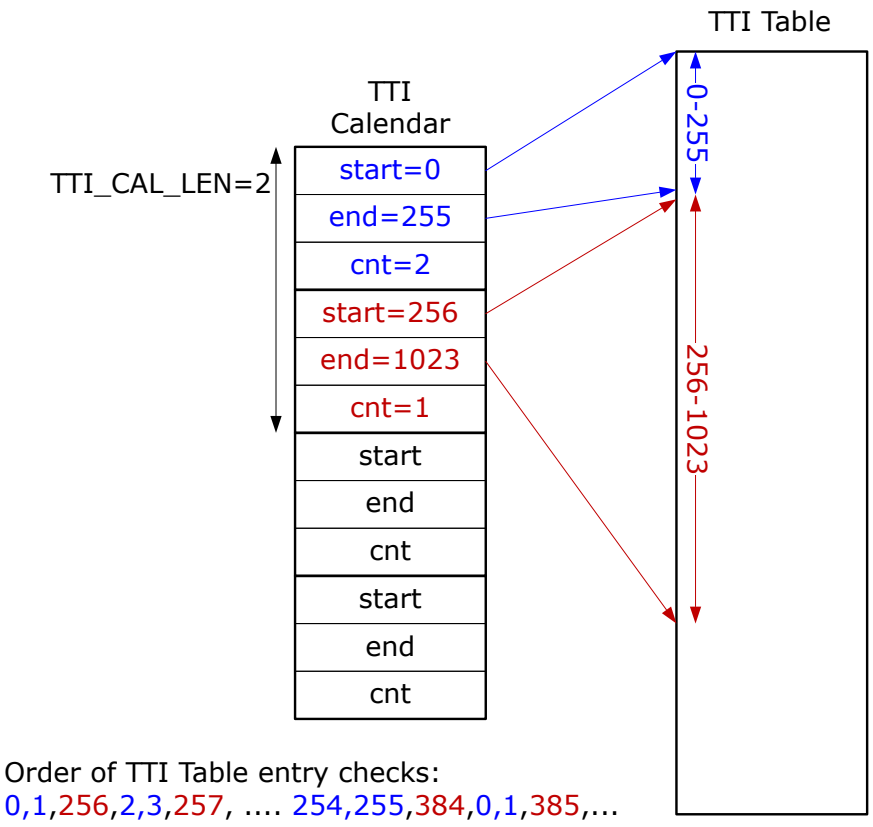
The TTI calendar consists of four entries, as described in Table 6

Table 4-231. TTI Calendar Registers Overview

Register	Field	Description	Replication
TTI_CAL_SLOT_PTRS	SLOT_START_PTR SLOT_END_PTR	Calendar slot's frame table start and end pointer.	4
TTI_CAL_SLOT_CNT	SLOT_CNT	Number of TTIs to service in slot before moving to next TTI calendar slot.	4
TTI_CTRL	TTI_CAL_LEN	Length of TTI calendar.	1
TTI_CTRL	TTI_INIT	When set, initializes calendar to start at calendar slot 0. Cleared by AFI when done.	1

The following figure shows a configuration for the TTI calendar. In this case, only TTI entry 0-1023 are being used.

Figure 4-88. TTI Calendar Example



In the example, it takes (256/2) × (1 + 2) × 4 = 1536 clock cycles to service all TTIs in the blue section. With a clock cycle for instance 4 ns, the TTIs in the blue section must not use ticks faster than 1536 × 4 = 6144 ns.

Similarly, the 768 TTIs in the red section must not use ticks faster than
(768/1) × (1 + 2) × 4 × 4 ns = ~36.9 µs

4.27.4.2 Other TTI Parameters

Other TTI-related parameters are listed in the following table.

Table 4-232. Miscellaneous TTI Registers Overview

Register	Field	Description	Replication
TTI_CTRL	TTI_ENA	Enables TTI. Before enabling TTI, TTI_INIT should be used to initialize calendar state.	1
TTI_INJ_CNT	TTI_INJ_CNT	Number of TTI injections. Enabled per TTI using TTI_TBL:TTI_MISC_CFG.INJ_CNT_ENA.	1

4.27.4.3 TTI Table Update Engine (AFI TUPE)

The AFI Table Update Engine (AFI TUPE) can be used to quickly update a large number of entries in the TTI Table such as disabling or enabling timers in fail-over scenarios. AFI TUPE related parameters are listed in the following table. The parameters prefixed with CRIT are used to specify the criteria, which TTIs fulfill in order for TUPE to apply.

Table 4-233. AFI TUPE Registers

AFI:TUPE Register	Field	Description	Replication
TUPE_MISC	TUPE_START	Start TUPE.	1
TUPE_MISC	CRIT_TUPE_CTRL_VAL_ENA	Enable use of CRIT_TUPE_CTRL_VAL and CRIT_TUPE_CTRL_MASK.	1
TUPE_MISC	CRIT_PORT_NUM_ENA	Enable use of CRIT_PORT_NUM_VAL.	1
TUPE_MISC	CRIT_QU_NUM_ENA	Enable use of CRIT_QU_NUM_VAL.	1
TUPE_MISC	CMD_TIMER_ENA_ENA	Enable use of CMD_TIMER_ENA_VAL.	1
TUPE_MISC	CMD_TIMER_ENA_VAL	TUPE command parameter: New TIMER_ENA value.	1
TUPE_MISC	CMD_PORT_NUM_ENA	Enable use of CMD_PORT_NUM_VAL.	1
TUPE_MISC	CMD_QU_NUM_ENA	Enable use of CMD_QU_NUM_VAL.	1
TUPE_ADDR	TUPE_START_ADDR	First address in TTI table for AFI TUPE to process.	1
TUPE_ADDR	TUPE_END_ADDR	Last address in TTI table for AFI TUPE to process.	1
TUPE_CRIT1	CRIT_PORT_NUM_VAL	TUPE criteria parameter controlling which TTI table entries to update.	1
TUPE_CRIT1	CRIT_QU_NUM_VAL	TUPE criteria parameter controlling which TTI table entries to update.	1
TUPE_CRIT2	CRIT_TUPE_CTRL_MASK	TUPE criteria parameter controlling which TTI table entries to update.	1
TUPE_CRIT3	CRIT_TUPE_CTRL_VAL	TUPE criteria parameter controlling which TTI table entries to update.	2
TUPE_CMD1	CMD_PORT_NUM_VAL	TUPE command parameter: New PORT_NUM value.	1
TUPE_CMD1	CMD_QU_NUM_VAL	TUPE command parameter: New QU_NUM value.	1

The following TTI parameters can be used as part of the TUPE criteria.

- PORT_NUM
- QU_NUM
- TUPE_CTRL

The parameters prefixed “CMD” specify the commands, which are applied to any TTIs, which fulfill the TUPE criteria. Using the TUPE command parameters, the following TTI parameters can be changed.

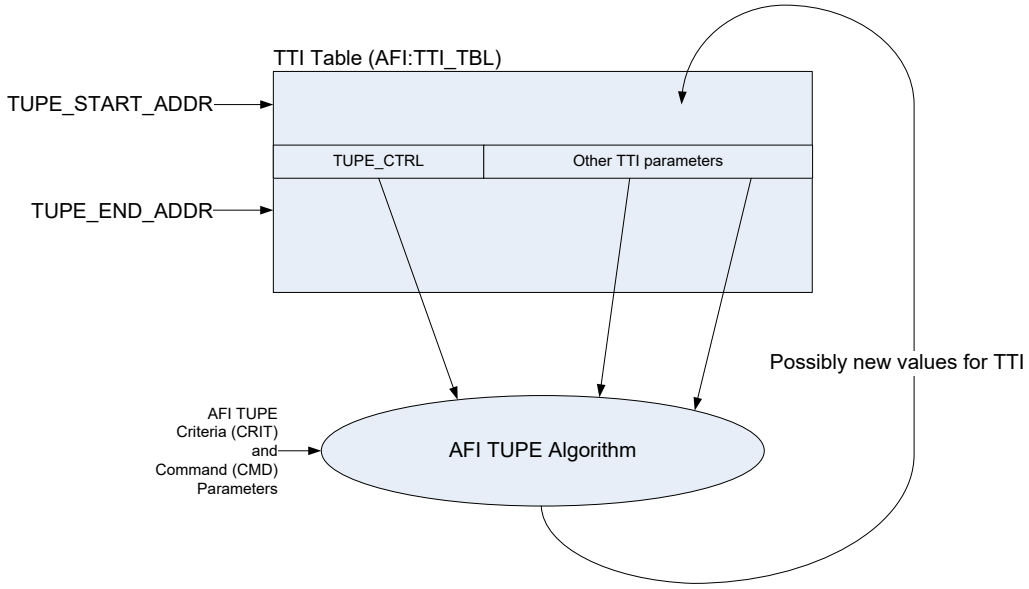
- TIMER_ENA
  Timers can be enabled/disabled.
- PORT_NUM
  Injections can be moved to a different port.
- QU_NUM
  Injections can be moved to a different queue.

While TUPE is running no injections will be performed for any TTIs which match the configured TUPE criteria and fall within the configured TUPE address range.

The TUPE_CTRL field in the TTI Table can be used to classify TTIs into different groups, such that a TUPE command can easily address all TTIs within such group.

The following figure shows the AFI TUPE functionality.

Figure 4-89. AFI TUPE



The full algorithm used by AFI TUPE is shown in the following pseudo code. Configuration parameters are prefixed “csr.”

```
// AFI Table UPdate Engine (AFI TUPE) Algorithm

bool tupe_ctrl_match;

for (addr = csr.tupe_start_addr; addr < csr.tupe_end_addr; addr++) {
  tti_tbl_entry = tti_tbl[addr];

  tupe_ctrl_match = FALSE;
  for (i = 0; i < 2; i++) {
    if ((tti_tbl_entry.tupe_ctrl & csr.crit_tupe_ctrl_mask) ==
        csr.crit_tupe_ctrl_val[i]) {
      tupe_ctrl_match = TRUE;
    }
  }

  if (
      (// Check matching TUPE_CTRL value
       tupe_ctrl_match)
      &&
      (// If enabled, check if TTI's PORT_NUM matches csr.crit_port_num_val
       !csr.crit_port_num_ena
       ||
       (tti_tbl_entry.port_num == csr.crit_port_num_val))
      &&
      (// If enabled, check if TTI's QU_NUM matches csr.crit_qu_num_val
       !csr.crit_qu_num_ena
       ||
```

```
        (tti_tbl_entry.qu_num == csr.crit_qu_num_val))
      ) {
    // TTI fulfills criterias => Apply TUPE command to TTI

    // timer_ena
    if (csr.cmd_timer_ena_ena) {
      tti_tbl_entry.timer_ena = csr.cmd_timer_ena_val;
    }

    // port_num
    if (csr.cmd_port_num_ena) {
      tti_tbl_entry.port_num = csr.cmd_port_num_val;
    }

    // qu_num
    if (csr.cmd_qu_num_ena) {
      tti_tbl_entry.qu_num = csr.cmd_qu_num_val;
    }

    // Write back to TTI table
    tti_tbl[addr] = tti_tbl_entry;
  }
}
```

4.27.5 Injection Queues

DTI and TTI may inject into any egress queue in the queue system. In injection context, the queues can be divided into the following categories.

- Normal queues: These queues are processed by each level of SEs in HSCH. Frames injected into such queues are thus subject to the M-DWRR and shaping algorithms of the SEs. Frames are injected into the tail of the selected queue.
- Port injection queue without shaping: These queues, one for each port, inject frames on the port with higher priority than normal queues. The state of the DLB shaper is disregarded and is not updated with the size of the transmitted frame.

  Injected frames are injected into the tail of the queue, but due to the high priority, the queue will normally be (almost) empty.
- Port injection queue with shaping: These queues, one for each port, inject frames on the port with higher priority than normal queues. The state of the port shaper is respected and is updated with the size of the transmitted frame.

  Injected frames are injected into the tail of the queue, but due to the high priority, the queue will normally be (almost) empty.

The queue into which a TTI or DTI injects frames is controlled by the QU_NUM parameter. The actual value to be used for this parameter depends on the HSCH configuration. For more information, see 4.26.5 Queue Mapping.

For injections into the ingress data path, frames must be looped through VD1 and QU_NUM must be configured to select a VD1 queue.

4.27.6 Adding Injection Frame

To add a frame for injection by AFI, the CPU must follow this procedure.

1. Set AFI::MISC_CTRL.AFI_ENA to 1 before using AFI.
2. Send the frame to AFI. AFI_INJ must be set in the IFH. For more information, see 4.2 Frame Headers.
3. Poll AFI::NEW_FRM_CTRL.VLD to await the frame being received by AFI.
4. When the frame has been received by AFI, then copy frame information from AFI::NEW_FRM_INFO to an unused entry in the Frame table.
5. Clear AFI::NEW_FRM_CTRL.VLD.
6. TTI: Set up the TTI table to have the frame injected with the desired interval and to the desired queue and port. For more information, see 4.27.4 Timer Triggered Injection.

7. DTI: For injection of a sequence of frames, repeat steps 1 through 4.

   Set up the DTI Table to inject the frames. For more information, see Table 4-227.

   Configure other DTI parameters. For more information, see Table 4-228.

4.27.7 Starting Injection

TTI is enabled by using TTI_INIT to initialize and then setting TTI_ENA = 1.

Each TTI is started by setting TIMER_LEN a non-zero value.

Each DTI is started by setting DTI_CTRL.ENA = 1.

4.27.8 Stopping Injection

Each TTI is stopped by setting TIMER_LEN to 0.

All TTIs can be stopped by setting TTI_ENA = 0.

Each DTI is stopped by setting DTI_CTRL.ENA = 0.

4.27.9 Removing Injection Frames

This section provides information about removing a single frame (TTI) or frame sequence (DTI) from the buffer memory.

4.27.9.1 Single Frame (TTI)

To remove a single frame from buffer memory, the frame must be injected one more time. This “removal injection” does not result in a frame transmission, but purely serves to remove the frame from the buffer memory.

Use the following procedure to remove a single frame from buffer memory.

1. Set the FRM_RM bit for frame in the frame table.
2. For TTI: Set TIMER_LEN to 0x1ff.
3. Poll the FRM_GONE bit until it gets set by AFI.

When performing removal injection, injection must be enabled for the corresponding port by setting AFI:PORT_TBL[<port>]:PORT_CFG.FRM_OUT_MAX != 0.

4.27.9.2 Frame Sequence (DTI)

Use the following procedure to remove a DTI frame sequence from buffer memory.

1. Disable the DTI by setting DTI_CTRL.ENA = 0.
2. Set the FRM_RM bit for each frame to be removed in the frame table.
3. Set DTI_FRM.NEXT_FRM_PTR to DTI_FRM.FIRST_FRM_PTR.
4. Set DTI_MODE.MODE = 0.
5. Set DTI_CNT.CNT = 1.
6. Set DTI_MODE.FRM_INJ_CNT to 0.
7. Optionally, set all delays in sequence to 0 to speed up the removal procedure.
8. Set DTI_CNT_DOWN.CNT_DOWN to 0.
9. Set DTI_CTRL.ENA to 1 to enable the DTI.
10. Poll the FRM_GONE for the last frame to be removed until the bit gets set by AFI.

This procedure causes the frames to be injected one last time and through this, be removed from buffer memory. The frames will not actually be transmitted on the destination port.

When performing removal injection, injection must be enabled for the corresponding port by setting AFI:PORT_TBL[<port>]:PORT_CFG.FRM_OUT_MAX != 0.

4.27.10 Port Parameters

To ensure that the queue system does not overflow with injected frames, for example, if a port is in flow control, an upper bound on the number of injected, but not yet transmitted, frames per port can be configured in FRM_OUT_MAX.

If FRM_OUT_MAX is reached, then any DTI injections are postponed until the number of outstanding frames falls below FRM_OUT_MAX. TTI injections uses a slightly higher limit of FRM_OUT_MAX+TTI_FRM_OUT_MAX. This ensures that TTI injection are still possible, even if a DTI flow faster than the port speed has been configured.

If PFC is used in conjunction with automatic frame injection, then either all frames must be injected into queues controlled by the same flow control priority, or all frames must be injected into queues that cannot be stopped by PFC.

If EEE is used in conjunction with automatic frame injection, then all frames for that port must be injected into queues with the same urgent configuration.

Setting FRM_OUT_MAX = 0 will stop all injections on port.

Table 4-234. Port Parameters

Register	Field	Description	Replication
PORT_CFG	FRM_OUT_MAX	Maximum number of injections outstanding for the port at-a-time.	Per port
TTI_PORT_FRM_OUT	TTI_FRM_OUT_MAX	Additional injections that can be outstanding before affecting TTI injection.	Per port

4.28 Rewriter

The switch core includes a rewriter (REW) that is common for all ports. The rewriter performs all frame editing prior to frame transmission. Each frame can be handled multiple times by the rewriter

- One time per destination port.
- One time towards the mirror target port.
- One time when copied or redirected to internal and/or external CPU.
- One time when looped back to the analyzer.

All frames that are input to the rewriter can be modified by previous blocks of the frame processing flow. The internal frame header informs the rewriter what ingress frame manipulations (popping) are performed on any given frame. The same ingress frame manipulation is performed for each copy of a frame, while the rewriter may perform per destination specific egress frame manipulations (pushing) to the frame.

The rewriter is the final step in the frame processing flow of the device.

4.28.1 Rewriter Operation

The rewriter supports the following frame operations.

- VLAN tag (802.1Q, 802.1ad) editing: tagging of frames and remapping of PCP and DEI.
- R-TAG (IEEE802.1CB) editing. Pop and push of one R-TAG.
- E-TAG (IEEE802.1BR) editing. Pop and push of one E-TAG.
- L3 routing. SMAC/DMAC, VID and TTL modifications for IPv4 and IPv6.
- IP-in-IPv4. Encapsulate and decapsulate IP-in-IPv4 frames.
- Interfacing to the loopback block. Loopback frames to ANA based on VCAP_ES0 actions. Rewrite MACs and IFH of looped frames.
- DSCP remarking: Rewriting the DSCP value in IPv4 and IPv6 frames based on classified DSCP value.
- Insert or update VSTAX headers.
- Handling of both Rx and Tx mirror frames.
- Padding of undersized frames to 64 bytes or 76 bytes.
- Frame FCS update control.
- Add preamble to all frames with an external port destination.
- Update Departure Service Point Statistics.

4.28.2 Supported Ports

The device operates with up to 65 physical ports, three loopback ports, and two DEVCPU extraction ports as possible frame destinations. These frame destinations are addressed by the rewriter using the egress port number.

4.28.3 Supported Frame Formats

The rewriter can identify and rewrite both headers and payload of the following frame types.

- Ethernet Frames
  - Ethernet link layer DMAC+SMAC
  - Maximum of three VLAN Q-tags (802.1Q, 802.1ad)
  - Payload (IPv4, IPv6, IP-in-IPv4, Y.1731 OAM, PTP, and so on)

The analyzer places information in the IFH to help the rewriter with frame identification. The IFH fields IFH.ENCAP.W16_POP_CNT and IFH.ENCAP.TYPE_AFTER_POP are used for this purpose.

4.28.3.1 Maximum Frame Expansion

The rewriter can push 60 bytes into a frame when the IFH is popped and when no other frame pop operations are done. The frame preamble will occupy 8 of the 60 bytes that can be pushed.

4.28.4 Rewriter Initialization

Rewriter initialization comprises the following step.

- Set REW::RAM_INIT.RAM_INIT to 1

Before the rewriter can be configured, the RAM-based configuration registers must be initialized by writing a 1 to the above RAM_INIT register. The RAM_INIT.RAM_INIT register is reset by hardware when the initialization is complete.

If VCAP_ES0 is enabled, it must be initialized. All the default port actions in VCAP_ES0 should also be initialized to ensure proper operation. For more information about initializing VCAP_ES0, see 4.10 Versatile Content-Aware Processor (VCAP™).

4.28.5 VCAP_ES0 Lookup

The rewriter performs stage 0 VCAP egress matching (VCAP_ES0 or just ES0) lookups for each frame when enabled. The following table lists the registers associated with VCAP_ES0 lookup.

Table 4-235. VCAP_ES0 Configuration Registers

Register	Description	Replication
ES0_CTRL. ES0_LU_ENA	Enables lookup in VCAP_ES0 to control advanced frame modifications.	None
ES0_CTRL. ES0_VD2_ENCAP_ID_ENA	When enabled and IFH.ENCAP.ENCAP_ID_RLEG_MODE = 1 and egress port = VD2, the ES0 XVID key field is set to IFH.ENCAP.ENCAP_ID_RLEG.	None
ES0_CTRL. ES0_FRM_LBK_CFG	Through ES0, it is possible to loop frames back to the analyzer using the LOOP_ENA and FWD_SEL actions. If this bit is set, a frame can only be looped once by ES0.	None
ES0_CTRL. ES0_DPORT_ENA	Setting this bit disables use of the PORT_CTRL.ES0_LPORT_NUM table for egress port to logical port mapping and forces use of the virtual DSTP received from QSYS as ES0 EGR_PORT key value. Correct virtual switch operation requires ES0_DPORT_ENA to be set.	None

.....continued		
Register	**Description**	**Replication**
ES0_CTRL. ES0_BY_RLEG	When enabled and IFH.FWD.DST_MODE = L3UC_ROUTING or IFH.FWD.DST_MODE = L3MC_ROUTING, the ES0 GVID key field is set to all ones, the XVID key field is set to IFH.ENCAP.ENCAP_ID_RLEG, and ES0 lookup is always done using VID key type. If IFH_FWD.DST_MODE does not indicate routing, the lookup is done normally regardless of the ES0_BY_RLEG configuration.	None
ES0_CTRL. ES0_BY_RT_FWD	When enabled and IFH.FWD.DST_MODE = ENCAP, IFH.ENCAP.RT_FWD = 1, and IFH.ENCAP.ENCAP_ID_RLEG_MODE = 3, the ES0 GVID key field is set to all ones, the XVID key field is set to IFH.ENCAP.ENCAP_ID_RLEG, and ES0 lookup is always done using VID key type. If IFH.ENCAP.RT_FWD does not indicate routing, the lookup is done normally regardless of the ES0_BY_RT_FWD configuration.	None
RTAG_ETAG_CTRL. ES0_ISDX_KEY_ENA	Force ES0 lookups to use a specific key, independently of the ISDX value and IFH.ES0_ISDX_KEY_ENA. Encoding: 0: Normal selection 1: Force ISDX lookups 2: Force VID lookups	Ports
PORT_CTRL. ES0_LPORT_NUM	Maps the configuration port to a logical port number to be used by ES0 keys. The port used in the lookup can be Tx-mirrored.	Ports

VCAP_ES0 lookup is enabled by setting REW::ES0_CTRL.ES0_LU_ENA = 1. The ES0 lookup is done for all frames when ES0 is enabled. Frames destined to CPU ports are not rewritten by ES0 actions unless this is configured in REW::IFH_CTRL_CPUVD.KEEP_IFH_SEL_CPUVD.

All ES0 actions are ignored when VCAP_ES0 lookups are disabled.

There are two ES0 key types: VID and ISDX. The key selection can be forced for each port by setting REW::RTAG_ETAG_CTRL.ES0_ISDX_KEY_ENA. The key selection is done as described below.

The VID key is used when RTAG_ETAG_CTRL.ES0_ISDX_KEY_ENA is different from 1 and at least one of the following four conditions are true.

1. IFH.FWD.ES0_ISDX_KEY_ENA = 0
2. REW::ES0_CTRL.ES0_BY_RLEG = 1 and (IFH.FWD.DST_MODE = L3UC_ROUTING or IFH.FWD.DST_MODE = L3MC_ROUTING)
3. REW::ES0_CTRL.ES0_BY_RT_FWD = 1 and IFH.ENCAP.RT_FWD = 1 and IFH.ENCAP.ENCAP_ID_RLEG_MODE = 3
4. RTAG_ETAG_CTRL.ES0_ISDX_KEY_ENA = 2

Because there are only two key types, the ISDX key is used when a VID key is not used.

Table 4-236. VCAP_ES0 Keys

Key Field	Description	Size	ISDX	VID
X1_TYPE	X1 type. 0: ISDX 1: VID	1	x	x

.....continued

Key Field	Description	Size	ISDX	VID
EGR_PORT	Logical egress port number. When ES0_CTRL.ES0_DPORT_ENA = 1: Global setting. The EGR_PORT is always controlled by the QSYS control field DSTP. No mapping is done for any egress ports. When ES0_CTRL.ES0_DPORT_ENA = 0: Configuration port mapped via REW::PORT_CTRL.ES0_LPORT_NUM. The default is no mapping (EGR_PORT = configuration port number). The configuration port is either the physical port or a Tx-mirror port number.	7	x	x

XVID	Classified VLAN
	Default value is:
	XVID = IFH.ENCAP.XVID_EXT & IFH.VSTAX.TAG.VID
	The following conditions will change the default.
	`# Use Encap ID`
	`if (REW::ES0_CTRL.ES0_VD2_ENCAP_ID_ENA = 1 and`
	`IFH.ENCAP.ENCAP_ID_RLEG_MODE = 1 and`
	`EGR_PORT = VD2)`
	`  XVID = IFH.ENCAP.ENCAP_ID_RLEG`
	`# Use legacy egress router leg`
	`elsif (REW::ES0_CTRL.ES0_BY_RLEG = 1 and (IFH.FWD.DST_MODE = L3UC_ROUTING or IFH.FWD.DST_MODE = L3MC_ROUTING))`
	`  XVID = IFH.DST.L3[UC|MC].ERLEG`
	`# Use egress router leg`
	`elsif (REW::ES0_CTRL.ES0_BY_RT_FWD = 1 and`
	`IFH.ENCAP.ENCAP_ID_RLEG_MODE = 3 and`
	`IFH.ENCAP.RT_FWD = 1)`
	`  XVID = IFH.ENCAP.ENCAP_ID_RLEG`
	`end if`
		13	x	x

COSID	Class of Service Identifier. Value is IFH.VSTAX.MISC.COSID, if REW::PORT_CTRL.VSTAX2_MISC_ISDX_ENA = 1 else 0.	3	x	X

.....continued

Key Field	Description	Size	ISDX	VID
TPID	IFH.ENCAP.TAG_TPID is used to generate the value of this key: 0: Untagged frame (IFH.VSTAX.TAG.WAS_TAGGED = 0) The following values are used when IFH.VSTAX.TAG.WAS_TAGGED = 1: 1: IFH.ENCAP.TAG_TYPE = 0 and IFH.VSTAX.TAG.TAG_TYPE = 0 (TPID=0x8100) 2,3: Never generated 4: IFH.ENCAP.TAG_TYPE = 0 and IFH.VSTAX.TAG.TAG_TYPE = 1 (TPID = 0x88A8) 5: IFH.ENCAP.TAG_TYPE = 1 (TPID = TPID_CFG[0].TPID_VAL) 6: IFH.ENCAP.TAG_TYPE = 2 (TPID = TPID_CFG[1].TPID_VAL) 7: IFH.ENCAP.TAG_TYPE = 3 (TPID = TPID_CFG[2].TPID_VAL)	3	x	x
COLOR	Frame color. Value is IFH.VSTAX.QOS.DP mapped through REW::DP_MAP.DP.	1	x	x
SERVICE_FRM	Service frame. Set to 1 if IFH.VSTAX.MISC.ISDX > 0 and REW::PORT_CTRL.VSTAX2_MISC_ISDX_ENA = 1 else 0.	1	x	x
PROT_ACTIVE	Protection is active. Value is IFH.ENCAP.PROT_ACTIVE.	1	x	x
ENCAP_ID_ENA	Set to 1 when ES0_CTRL.ES0_VD2_ENCAP_ID_ENA = 1 and IFH.ENCAP.ENCAP_ID_RLEG_MODE = 1 and EGR_PORT = VD2 else 0.	1	x	x
OAM_Y1731	Set 1 if IFH.ENCAP.PDU_TYPE = OAM_Y1731 else 0.	1	x	x

.....continued

Key Field	Description	Size	ISDX	VID
OAM_MEL_FLAGS	When IFH.ENCAP.PDU_TYPE = OAM_Y1731, the OAM_MEL value of Y1731 OAM frames is used to generate this key. If IFH.ENCAP.PDU_TYPE is not OAM_Y1731, the OAM_MEL_FLAGS key is always 0. Encoding of MD level / MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL = 0: 0b0000000 MEL = 1: 0b0000001 MEL = 2: 0b0000011 MEL = 3: 0b0000111 MEL = 4: 0b0001111 MEL = 5: 0b0011111 MEL = 6: 0b0111111 MEL = 7: 0b1111111 Together with the mask, the following kinds of rules may be created: • Exact match. For example, MEL = 2: 0b0000011 • Below. For example, MEL<= 4: 0b000XXXX • Above. For example, MEL >= 5: 0bXX11111 • Between. For example, 3 <= MEL <=5: 0b00XX111, where 'X' means do not care.	7	x	x
ISDX	Ingress Service Index. Value is IFH.VSTAX.MISC.ISDX if REW::PORT_CTRL.VSTAX2_MISC_ISDX_ENA = 1 else 0.	12	x	

GVID	Generic VLAN identifier.
	Default value is GVID = IFH.ENCAP.GVID
	The following conditions will change the default:
	`if (REW::ES0_CTRL.ES0_BY_RLEG = 1 and (IFH.FWD.DST_MODE = L3UC_ROUTING or IFH.FWD.DST_MODE = L3MC_ROUTING)) Force GVID = 0xFFF to identify routing.`
	`elsif (REW::ES0_CTRL.ES0_BY_RT_FWD = 1 and IFH.ENCAP.ENCAP_ID_RLEG_MODE = 3 and IFH.ENCAP.RT_FWD = 1) Force GVID = 0xFFF to identify routing. end if`
		12		x

The following table shows the actions available in VCAP_ES0. Default actions are selected by PORT_CTRL.ES0_LPORT_NUM when no ES0 entry is hit. When REW::ES0_CTRL.ES0_LU_ENA = 1, all of the default actions must be initialized to ensure correct frame handling.

Table 4-237. VCAP_ES0 Actions

Field Name	Description	Width
PUSH_OUTER_TAG	Controls tag A (outer tagging). 0: Port tagging. Push port tag as outer tag if enabled for port. No ES0 tag A. 1: ES0 tag A. Push ES0 tag A as outer tag. No port tag. 2: Forced port tagging. Push port tag as outer tag. No ES0 tag A. 3: Forced untagging. No outer tag pushed (no port tag, no ES0 tag A). The analyzer may push an outer tag (TAG_A) using IFH.TAGGING.PUSH_CNT when PUSH_OUTER_TAG = 3.	2
PUSH_INNER_TAG	Controls tag B (inner tagging). 0: Do not push ES0 tag B as inner tag. 1: Push ES0 tag B as inner tag. The analyzer may push an inner tag (TAG_B) using IFH.TAGGING.PUSH_CNT when PUSH_INNER_TAG = 0.	1
TAG_A_TPID_SEL	Selects TPID for ES0 tag A. 0: 0x8100. 1: 0x88A8. 2: Custom1. 3: Custom2. 4: Custom3. 5: Classified. Selected by ANA in IFH.VSTAX.TAG_TYPE and IFH.ENCAP.TAG_TPID: If IFH.ENCAP.TAG_TPID = STD_TPID: If IFH.VSTAX.TAG.TAG_TYPE = 0, then 0x8100 else 0x88A8 If IFH.ENCAP.TAG_TPID = CUSTOM[n]: REW::TPID_CFG[n]:TPID_VAL.	3
TAG_A_VID_SEL	Selects VID for ES0 tag A. 0: Classified VID + VID_A_VAL. 1: VID_A_VAL. 2: IFH.ENCAP.GVID + VID_A_VAL. 3: Reserved	2

.....continued

Field Name	Description	Width
TAG_A_PCP_SEL	Select PCP source for ES0 tag A. 0: Classified PCP. 1: PCP_A_VAL. 2: Reserved. 3: PCP of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and tot_pop_cnt > 0) else PCP_A_VAL. 4: Mapped using mapping table 0, otherwise use PCP_A_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use PCP_A_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_A_DEI_SEL	Select DEI source for ES0 tag A. 0: Classified DEI. 1: DEI_A_VAL. 2: REW::DP_MAP.DP [IFH.VSTAX.QOS.DP]. 3: DEI of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and tot_pop_cnt > 0) else DEI_A_VAL. 4: Mapped using mapping table 0, otherwise use DEI_A_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DEI_A_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_B_TPID_SEL	Selects TPID for ES0 tag B. 0: 0x8100. 1: 0x88A8. 2: Custom 1. 3: Custom 2. 4: Custom 3. 5: See TAG_A_TPID_SEL.	3
TAG_B_VID_SEL	Selects VID for ES0 tag B. 0: Classified VID + VID_B_VAL. 1: VID_B_VAL. 2: IFH.ENCAP.GVID + VID_B_VAL. 3: Reserved	2

.....continued

Field Name	Description	Width
TAG_B_PCP_SEL	Selects PCP source for ES0 tag B. 0: Classified PCP. 1: PCP_B_VAL. 2: Reserved. 3: PCP of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED=1 and tot_pop_cnt > 0) else PCP_B_VAL. 4: Mapped using mapping table 0, otherwise use PCP_B_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use PCP_B_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_B_DEI_SEL	Selects DEI source for ES0 tag B. 0: Classified DEI. 1: DEI_B_VAL. 2: REW::DP_MAP.DP [IFH.VSTAX.QOS.DP]. 3: DEI of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and tot_pop_cnt > 0) else DEI_B_VAL. 4: Mapped using mapping table 0, otherwise use DEI_B_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DEI_B_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_C_TPID_SEL	Selects TPID for ES0 tag C. 0: 0x8100. 1: 0x88A8. 2: Custom 1. 3: Custom 2. 4: Custom 3. 5: See TAG_A_TPID_SEL.	3

.....continued

Field Name	Description	Width
TAG_C_PCP_SEL	Selects PCP source for ES0 tag C. 0: Classified PCP. 1: PCP_C_VAL. 2: Reserved. 3: PCP of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED=1 and tot_pop_cnt>0) else PCP_C_VAL. 4: Mapped using mapping table 0, otherwise use PCP_C_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use PCP_C_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_C_DEI_SEL	Selects DEI source for ES0 tag C. 0: Classified DEI. 1: DEI_C_VAL. 2: REW::DP_MAP.DP [IFH.VSTAX.QOS.DP]. 3: DEI of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and tot_pop_cnt>0) else DEI_C_VAL. 4: Mapped using mapping table 0, otherwise use DEI_C_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DEI_C_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
VID_A_VAL	VID used in ES0 tag A. See TAG_A_VID_SEL.	12
PCP_A_VAL	PCP used in ES0 tag A. See TAG_A_PCP_SEL.	3
DEI_A_VAL	DEI used in ES0 tag A. See TAG_A_DEI_SEL.	1
VID_B_VAL	VID used in ES0 tag B. See TAG_B_VID_SEL.	12
PCP_B_VAL	PCP used in ES0 tag B. See TAG_B_PCP_SEL.	3
DEI_B_VAL	DEI used in ES0 tag B. See TAG_B_DEI_SEL.	1
VID_C_VAL	VID used in ES0 tag C. See TAG_C_VID_SEL.	12
PCP_C_VAL	PCP used in ES0 tag C. See TAG_C_PCP_SEL.	3
DEI_C_VAL	DEI used in ES0 tag C. See TAG_C_DEI_SEL.	1
POP_VAL	Controls popping of Q-tags. The final number of Q-tags popped is calculated as shown in section 4.28.7.2 VLAN Pop Decision.	2
UNTAG_VID_ENA	Controls insertion of tag C. Untag or insert mode can be selected. See PUSH_CUSTOMER_TAG.	1

.....continued

Field Name	Description	Width
PUSH_CUSTOMER_TAG	Selects tag C mode: 0: Do not push tag C. 1: Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 1. 2: Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 0. 3: Push tag C if UNTAG_VID_ENA = 0 or (C-TAG.VID ! = VID_C_VAL).	2
TAG_C_VID_SEL	Selects VID for ES0 tag C. The resulting VID is termed C-TAG.VID. 0: Classified VID. 1: VID_C_VAL. 2: IFH.ENCAP.GVID. 3: Reserved.	2
PUSH_ETAG	Enable pushing of IEEE802.1BR Bridge Port Extender tag (E-TAG). The E-TAG is always pushed as the outermost tag before any VLAN tags. 0: No E-TAG pushed 1: Push E-TAG if enabled by REW::COMMON_CTRL.ETAG_TPID_ENA	1
ECID_BASE_SEL	Selects E-CID_base source for E-TAG. 0: E-CID_base of popped E-TAG when IFH.TAGGING.RTAG_ETAG_AVAIL = 1, otherwise ECID_BASE_VAL. 1: ECID_BASE_VAL	1
ECID_EXT_SEL	Selects E-CID_ext source for E-TAG. 0: E-CID_ext of popped E-TAG when IFH.TAGGING.RTAG_ETAG_AVAIL = 1, otherwise ECID_EXT_VAL. 1: ECID_EXT_VAL	1
GRP_SEL	Selects GRP source for E-TAG. 0: GRP of popped E-TAG when IFH.TAGGING.RTAG_ETAG_AVAIL = 1, otherwise GRP_VAL. 1: GRP_VAL	1

.....continued

Field Name	Description	Width
IGR_ECID_SEL	Selects Ingr_E-CID (both Ingr_E-CID_base and Ingr_E-CID_ext) source for E-TAG. 0: Ingr_E-CID of popped E-TAG when IFH.TAGGING.RTAG_ETAG_AVAIL = 1, otherwise zero 1: E-CID of popped E-TAG (EXT, BASE) when IFH.TAGGING.RTAG_ETAG_AVAIL = 1 and “logical egress port = logical ingress port", otherwise zero 2: Zero 3: Reserved Logical ingress port: Mapping of IFH.SRC_PORT: L_IGR_PORT = REW::RTAG_ETAG_CTRL.IPE_TBL[IFH.SRC_PORT] Logical egress port: Mapping of cell bus port number: L_EGR_PORT := REW::RTAG_ETAG_CTRL.IPE_TBL[REW.CELLBUS.PORT_NUM]	2
ECID_BASE_VAL	ECID_BASE used in E-TAG. See ECID_BASE_SEL	12
ECID_EXT_VAL	ECID_EXT used in E-TAG. See ECID_EXT_SEL	8
GRP_VAL	GRP used in E-TAG. See GRP_SEL	2
EPCP_EDEI_SEL	Selects PCP and DEI source for E-TAG 0: Reserved 1: EPCP_VAL, EDEI_VAL 2: Reserved 3: E-PCP, E-DEI of popped when IFH.TAGGING.RTAG_ETAG_AVAIL = 1, otherwise EPCP_VAL, EDEI_VAL 4: Mapped using mapping table 0 if valid, otherwise use EPCP_VAL, EDEI_VAL 5: Mapped using mapping table 1 if valid, otherwise use mapping table 0 6: Mapped using mapping table 2 if valid, otherwise use EPCP_VAL, EDEI_VAL 7: Mapped using mapping table 3 if valid, otherwise use mapping table 2	3
EPCP_VAL	EPCP used in E-TAG. See EPCP_EDEI_SEL	3
EDEI_VAL	EDEI used in E-TAG. See EPCP_EDEI_SEL	1

.....continued

Field Name	Description	Width
RTAG_POP_ENA	Controls popping of R-TAG: 0: No action or transparent R-TAG mode 1: Pop R-TAG if IFH.TAGGING.RTAG_ETAG_AVAIL = 1 **Note:** The R-TAG cannot be popped unless all Q-tags in front of it are also popped. If the R-TAG is not popped it will remain unmodified in the egress frame regardless of the RTAG_PUSH_SEL configuration. The R-TAG is always popped if possible when RTAG_POP_ENA = 1 and IFH.TAGGING.RTAG_ETAG_AVAIL = 1	1
RTAG_PUSH_SEL	Controls pushing of R-TAG: 0: Transparent R-TAG mode or no push action if the R-TAG is popped (RTAG_POP_ENA = 1 and IFH.RTAG_ETAG_AVAIL = 1) 1: Push R-TAG as outer tag if IFH.TAGGING.RTAG_ETAG_AVAIL = 0 or the R-TAG is popped. Otherwise no action 2: Push R-TAG as middle tag if IFH.TAGGING.RTAG_ETAG_AVAIL = 0 or the R-TAG is popped. Otherwise no action 3: Push R-TAG as inner tag if IFH.TAGGING.RTAG_ETAG_AVAIL = 0 or the R-TAG is popped. Otherwise no action The sequence number in the pushed R-TAG is taken from IFH.TAGGING.SEQ_NO **Note:** RTAG_POP_ENA and RTAG_PUSH_SEL can be set at the same time, effectively replacing a sequence number with a new sequence number (sequence generator in ANA can run also for frames already containing an R-TAG) Transparent R-TAG mode (no change to incoming frame in terms of R-TAG) is achieved by not configuring anything. The transparent R-TAG is always pushed in the innermost tag position if Q-tags are also pushed or remain in position if all Q-tags in front of it where not popped. A maximum of 3 tags in total can be pushed by ES0. A pushed R-TAG will replace a pushed Q-tag in the position configured by RTAG_PUSH_SEL	2
DSCP_SEL	Selects source for DSCP. 0: Controlled by port configuration and IFH. 1: Classified DSCP via IFH. 2: DSCP_VAL. 3: Reserved. 4: Mapped using mapping table 0, otherwise use DSCP_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DSCP_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
DSCP_VAL	DSCP value.	6

.....continued

Field Name	Description	Width
PROTECTION_SEL	Change ENCAP_ID, ESDX_BASE and OAM_MEP_IDX based on IFH.ENCAP.PROT_ACTIVE. 0: No protection. Do not use protection index values. 1: Use <index>_P if ifh.encap.prot_active = 1. 2: Use <index>_P if ifh.encap.prot_active = 0. 3: Reserved.	2
RESERVED	Must be set to 0.	160
ENCAP_ID	Index into the MPLS encapsulation table (REW::ENCAP). Setting ENCAP_ID to zero disables pushing of MPLS encapsulation. 0: Disables the use of MPLS encapsulation. 1–1023: Selects ENCAP ID.	10
ENCAP_ID_P	Index to use when enabled by PROTECTION_SEL. The encoding is equal to ENCAP_ID.	10
POP_CNT	If POP_CNT > 0, then IFH.ENCAP.W16_POP_CNT is overruled with POP_CNT when popping MPLS labels. The encoding is: 0: Use IFH.ENCAP.W16_POP_CNT as word16 (2 bytes) pop count 1: Do not pop any bytes. 2: Pop 2 × 2 = 4 bytes. … 21: Pop 2 × 21 = 42 bytes. 22–31: Reserved. If the number of bytes popped by POP_CNT is larger than the value given by IFH.ENCAP.W16_POP_CNT, the IFH value is always used.	5
ESDX_BASE	Egress service index. Used to index egress service counter set (REW::CNT_CTRL.STAT_MODE).	13
ESDX_BASE _P	Egress service index to use when enabled by PROTECTION_SEL.	13
ESDX_COSID_ OFFSET	Egress counter set, offset per COSID. Stat index = ESDX_BASE + ESDX_COSID_OFFSET[(3 x COSID + 2) : 3 x COSID]	24
MAP_0_IDX	Index 0 into mapping resource A. Index bits 11:3. Index bits 2:0 are controller by MAP_0_KEY.	9
MAP_1_IDX	Index 1 into mapping resource A. Index bits 11:3. Index bits 2:0 are controller by MAP_1_KEY.	9
MAP_2_IDX	Index 0 into mapping resource B. Index bits 11:3. Index bits 2:0 are controller by MAP_2_KEY.	9
MAP_3_IDX	Index 1 into mapping resource B. Index bits 11:3. Index bits 2:0 are controller by MAP_2_KEY.	9

.....continued

Field Name	Description	Width
MAP_0_KEY	Key used when looking up mapping table 0. **QoS mappings:** 0: QoS (8 entries): IDX = IDX_A + QOS. 1: DP, QoS (32 entries): IDX = IDX_A + IFH..DP × 8 + QOS. **DSCP mappings:** 2: DSCP (64 entries): IDX = IDX_A + IFH..DSCP. 3: DP, DSCP (256 entries): IDX = IDX_A + 64 × IFH..DP + IFH..DSCP. 4: DP, TOS_PRE (32 entries): IDX = IDX_A + IFH..DP × 8 + TOS_PRE. **PCP mappings:** 5: PCP (8 entries): IDX = IDX_A + IFH..PCP. 6: DP, PCP (32 entries): IDX = IDX_A + IFH..DP × 8 + IFH..PCP. 7: DEI, PCP (16 entries): IDX = IDX_A + IFH..DEI × 8 + IFH..PCP. **TC mappings:** 8: TC: IDX = IDX_A + IFH..TC. 9: DP, TC: IDX = IDX_A + IFH..DP × 8 + IFH..TC. **Popped customer tag mappings:** 10: Popped PCP (8 entries): IDX = MAP_0_IDX + pop.PCP 11: DP, popped PCP (32 entries): IDX = MAP_0_IDX + IFH..DP × 8 + pop.PCP. 12: Popped DEI, popped PCP (16 entries): IDX = MAP_0_IDX + pop.DEI × 8 + pop.PCP. **COSID mappings:** 13: COSID (8 entries): IDX = IDX_A + IFH.VSTAX.MISC.COSID. 14: DP, COSID (32 entries): IDX = IDX_A + IFH..DP × 8 + IFH.VSTAX.MISC.COSID.	4
MAP_1_KEY	See MAP_0_KEY.	4
MAP_2_KEY	See MAP_0_KEY.	4
MAP_3_KEY	See MAP_0_KEY.	4
RESERVED	Must be set to 0.	34
FWD_SEL	ES0 Forward selector. 0: No action. 1: Copy to loopback interface. 2: Redirect to loopback interface. 3: Discard.	2
CPU_QU	CPU extraction queue. Used when FWD_SEL >0 and PIPELINE_ACT = XTR.	3

.....continued

Field Name	Description	Width
PIPELINE_PT	ES0 pipeline point. 0: REW_IN_SW. 1: REW_OU_SW. 2: REW_SAT. 3: Reserved.	2
PIPELINE_ACT	Pipeline action when FWD_SEL > 0. 0: XTR. CPU_QU selects CPU extraction queue 1: LBK_ASM.	1
MAC_IDX	Index to REW::MAC_TBL table of MAC addresses. Used to select a replacement SMAC or DMAC when this is enabled by VCAP_ES2 or to select a replacement DMAC when enabled by ES0.REPLACE_DMAC_ENA.	7
REPLACE_DMAC_ENA	When enabled, the frame's DMAC is replaced by the DMAC located at address ES0.MAC_IDX in the REW::MAC_TBL table. The DMAC is only replaced for frames where IFH.ENCAP.TYPE_AFTER = ETH. 0: No action 1: Replace DMAC with MAC obtained from REW::MAC_TBL[MAC_IDX].	1
SWAP_MAC_ENA	This setting is only active when FWD_SEL = 1 or FWD_SEL = 2 and PIPELINE_ACT = LBK_ASM. 0: No action. 1: Swap MACs and clear bit 40 in new SMAC.	1
LOOP_ENA	0: Forward based on PIPELINE_PT and FWD_SEL 1: Force FWD_SEL=REDIR, PIPELINE_ACT = LBK_ASM, PIPELINE_PT = REW_SAT, swap MACs, and clear bit 40 in new SMAC. Do not apply CPU_QU.	1
QS_INFO	This field is not used by the rewriter. It is passed directly to the QSYS and used by the HSCH in QSYS to do frame dependent adjustments of the scheduling algorithm. Two 6-bit frame adjustment values used in layer 0 and layer 1 scheduling elements. Interpreted as two's complement (signed) values: Bits 5–0: –32 to 31 Bits 11–6: –32 to 31	12

4.28.5.1 Rewriter Pipeline Handling

In order to achieve the correct statistics and processing of frames, it is important that frames can be extracted, injected, discarded, and looped at specific positions in the processing flow. Based on the position, processing—especially counters, are affected.

Note that a frame is always physically passed through the whole processing. Logically, certain elements of the processing can be disabled or enabled.

The following table lists the pipeline points available in the rewriter.

Table 4-238. Rewriter Pipeline Point Definitions

Pipeline Point	Source	Position in Flow
NONE	Default	0
REW_IN_MIP	Inner MIP configured in rewriter	1
REW_IN_SW	Inner software MEP controlled by VCAP_ES0	2
REW_IN_VOE	Service VOE in VOP	3
REW_OU_VOE	Service VOE in VOP	4
REW_OU_SW	Outer software MEP controlled by VCAP_ES0	5
REW_OU_MIP	Outer software MIP configured in rewriter	6
REW_SAT	SAT software MEP controlled by VCAP_ES0	7
REW_PORT_VOE	Port VOE in VOP	8
REW_VRAP	Set in VRAP reply frame	9

Each point in the flow assigns one of the possible pipeline actions listed in the following table.

Table 4-239. Pipeline Actions

Action Type	Name	Comment
Inject	INJ INJ_MASQ INJ_CENTRAL	Injection actions set by ANA are cleared when the frame enters the rewriter by setting the pipeline point and action to NONE.
Extract	XTR XTR_UPMEP	XTR is set by ES0 and MIP. XTR_UPMEP is also set by the VOP.
Loopback	LBK_ASM LBK_QS	LBK_QS is set in ANA. This action is treated as an injection action in the rewriter. LBK_ASM can be set by ES0 when looping frames back to ANA. This action is treated as an extraction in the rewriter.
None	NONE	Default. No pipeline point and action is assigned to frame.

Frames looped by the analyzer (PIPELINE_ACT = LBK_QS) have their ANA pipeline point changed to a REW pipeline point, which indicates the point in the rewriter flow in REW where they appear again after being looped. The following pipeline point translations are handled.

- PIPELINE_PT = ANA_PORT_VOE is changed to REW_PORT_VOE
- PIPELINE_PT = ANA_OU_VOE is changed to REW_OU_VOE
- PIPELINE_PT = ANA_IN_VOE is changed to REW_IN_VOE

Frame forward and counter updates are done based on the assigned pipeline points and actions. In the rewriter, pipeline points and actions can be assigned by ES0, the MIP, and the VOP.

Frames injected by a CPU can also have pipeline point and actions set in the injection IFH.

The rules listed in the following table apply when pipeline points are updated.

Table 4-240. Pipeline Point Updating Rules

Pipeline Action	Rule
None	The pipeline point and actions can be updated.

.....continued	
Pipeline Action	**Rule**
Inject	If a frame is injected in pipeline point N, the forwarding cannot be changed in pipeline point N-1 and it will not be counted in the N-1 point. Logically the frame does not exist in the N-1 point. Example: A frame is injected in the REW_VRAP pipeline point (Pipeline point = REW_VRAP, Action = INJ). The SW-MEP is configured to loop the same frame in the pipeline point REW_SAT. The frame will not be looped, because it was injected later in the pipeline (REW_VRAP > REW_SAT). The frame will not be counted by the egress service statics, which is located in the REW_PORT_VOE pipeline point for injected frames.
Extract	If a frame is extracted in pipeline point N the forwarding cannot be changed in pipeline point N+1 and it will not be counted in the N+1 point. Logically the frame does not exist in the N+1 point. Example: A frame is extracted in the REW_IN_MIP pipeline point (Pipeline point = REW_IN_MIP, Action = XTR). The SW-MEP is configured to loop the same frame in the pipeline point REW_SAT. The frame will not be looped because it has already been extracted earlier in the pipeline (REW_IN_MIP < REW_SAT). The frame will not be counted by the egress service statics, which is located in the REW_OU_SW pipeline point for extracted frames.

4.28.5.2 Frame Copy, Redirect, or Discard

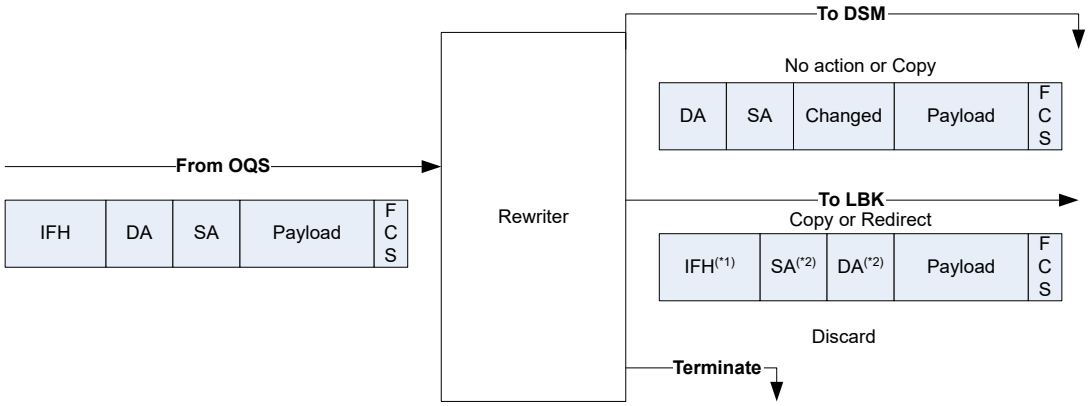
Frame forwarding may be changed by ES0 action FWD_SEL, the MIP or the VOP. Changing frame forwarding will update the frame pipeline point and action.

The following options are available to control frame forwarding.

- No action. Do not change frame forwarding.
- Copy frame to CPU via the loopback interface. The frame is looped back to the ANA where it is sent to the CPU. The IFH of the CPU copy is modified to select a CPU queue. The frame is also forwarded as normal to the destination port and can be modified by the normal rewriter operations.
- Redirect frame to ANA via loopback interface. When doing frame loopbacks, the destination port of the redirected frame will be left unchanged. The IFH of the frame is modified and the MAC addresses may be swapped. Redirected frames may also be sent to a CPU queue. This can be done as a standalone operation or together with the frame loopback. When redirecting, the frame is only sent to the loopback interface and not to the original destination port.
- Discard frame. The frame is terminated before transmission.

The following figure depicts options for frame forwarding.

Figure 4-90. Frame Forward Options



The following table lists the registers associated with the frame loopback options.

Table 4-241. Frame Loopback Options

Register	Description	Replication
ES0_CTRL.ES0_FRM_LBK_CFG	Using ES0 it is possible to loop frames back to ANA with the LOOP_ENA action. If this bit is set, a frame can only be looped once by ES0.	None

Frame forwarding is changed if allowed by the current frame pipeline point and action. For more information, see 4.28.5.1 Rewriter Pipeline Handling.

The following table shows the pipeline action when the frame forwarding is changed.

Table 4-242. Pipeline Action Updates

Forwarding Selection	Pipeline Action Updates
NONE	Unchanged
COPY	Unchanged
REDIR	CPU redirection: XTR / XTR_UPMEP Loopback frames: LBK_ASM
DISCARD	XTR / XTR_UPMEP

The pipeline updating rules ensures that a REDIR or DISCARD decision cannot be changed at a later pipeline point. A COPY may be changed at a later point to either a REDIR or DISCARD action.

- If a copy is changed to a discard, the CPU still receives the frame copy.
- If a copy is changed to a redirection, the CPU receives a copy and the frame is looped back or also sent to the redirection queue.

When frames are copied or redirected the IFH is updated as described in the following table.

Table 4-243. Loopback IFH Updates

IFH Field	Condition for Update
FWD.SRC_PORT	COPY or REDIR. The frame source port is changed to the current destination port.

.....continued	
IFH Field	**Condition for Update**
MISC.PIPELINE_PT	COPY or REDIR. The pipeline point in which the frame forwarding was updated.
MISC.PIPELINE_ACT	COPY or REDIR. The pipeline action assigned to the frame.
MISC.CPU_MASK	Any operation. A CPU copy may be generated by all forward selections.
FWD.DO_LBK	Rewriter loopback (ES0.LOOP_ENA). Set this bit to indicate that the frame has been looped. Can optionally be used to prevent multiple loops of the same frame.
FWD.UPDATE_FCS	VOP. Set by the VOP if a looped frame is edited.
VSTAX.MISC.ISDX	VOP. The frame ISDX can be changed by the VOP.
VSTAX.QOS.DP	VOP. The DP level can be set to 0 by the VOP.

4.28.5.3 Egress Statistics Update

The rewriter does not have Egress Service Index (ESDX) and port-based counters of its own. Instead, the egress statistic counters in the XQS:STAT are used for all frame and byte counting. For more information, see 4.26.9.4 Statistics.

The following counters are available to the rewriter as frame and octet counters.

- Per egress port: 2 × 8 + 1 = 17 counters (color × cell bus priority + abort)
- ESDX: 2 × 8192 = 16384 counters (color × ESDX)

Frame and octets are counted simultaneously on each index by two separate counters. The counter widths are 40 bits for octet counters and 32 bits for frame counters.

The following table lists the registers associated with ESDX counter configuration.

Table 4-244. Egress Statistics Control Registers

Register	Description	Replication
CNT_CTRL.STAT_MODE	Configures the egress service statistics mode. The following options for looking up the egress service counter set are available. In mode 0 and 1 yellow frames (DE = 1) are counted separately. In mode 2 and 3 multicast frames (DMAC bit 40 = 1) are counted separately. 0: Use ESDX from ES0 if hit, otherwise ISDX 1: Use ESDX from ES0 if hit, otherwise no counting 2: Use ISDX (ifh.vstax.misc.isdx as index) 3: Use classified VID (ifh.vstax.tag.vid)	Global
CNT_CTRL.VSTAX_STAT_ESDX_DIS	Configures ESDX counting for stacking ports. If this bit is set and PORT_CTRL.VSTAX_HDR_ENA = 1, all counting based on the ESDX value is disabled regardless of the CNT_CTRL.STAT_MODE configuration.	Global

.....continued

Register	Description	Replication
PORT_CTRL. PORT_STAT_INNER_ENA	Configures the port statistics pipeline point location. Two locations can be selected: Outer: The port statistics is located after REW_VRAP. All frames extracted in REW are not counted. All frames injected in REW are counted. Inner: The port statistics is located before REW_IN_MIP. All frames extracted in REW are counted. All frames injected in REW are not counted. This configuration only affects frames that are either injected or extracted in the rewriter. Other frames are always counted by the ports statistics regardless of the inner- or outer location. 0: Outer location 1: Inner location	
CNT_CTRL.STAT_CNT_FRM_ABORT_ENA	Enables counting of frames discarded by the rewriter. This bit only enables counting of frames discarded by ES0 or the SW_MIP. Frame discards done by the VOP are not included.	Global
PORT_CTRL.XTR_STAT_PIPELINE_PT	Configures the extraction statistics pipeline point. Frames extracted before the configured pipeline point are not counted by the ESDX counters. Configuring the REW_IN_MIP value causes all extracted frames to be counted. Default is REW_OU_SW.	Port
PORT_CTRL.INJ_STAT_PIPELINE_PT	Configures the injection statistics pipeline point. Frames injected after the configured pipeline point are not counted by the ESDX counters. Configuring the REW_VRAP value causes all injected frames to be counted. Default is REW_OU_VOE.	Port

4.28.5.4 Statistics Counter Index Calculation

The port counter index is always the physical port number and priority. The frame color is determined in the same manner as the ESDX counter.

The ESDX counter index calculation is controlled by the CNT_CTRL.STAT_MODE register.

Table 4-245. ESDX Index Selection

STAT_MODE	Description
0	Use ESDX if available: ESDX is available if ES0 lookup is enabled and if ES0.ESDX_BASE is different from 0. Index = ES0.ESDX_BASE + ES0.ESDX_COSID_OFFSET(IFH.VSTAX.MISC.COSID). If ESDX is not available: Index = IFH.VSTAX.MISC.ISDX (only 4096 available) Regardless of the index selection, the frame color is: Color = DP_MAP.DP(IFH.VSTAX.QOS.DP)
1	Similar to STAT_MODE = 0, except counting is disabled if the ESDX is not available.
2	Use ISDX: Index = IFH.VSTAX.MISC.ISDX. The frame color is used to count multicast frames: Color = FRAME.ETH_LINK_LAYER.DMAC (bit 40).
3	Use VID: Index = Classified VID (IFH.VSTAX.TAG.VID), if no tag is pushed Index = VID of outermost pushed tag, if a tag is pushed Index = REW:VMID:RLEG_CTRL.RLEG_EVID (IFH.DST..ERLEG) if L3. The frame color is used to count multicast frames: Color = FRAME.ETH_LINK_LAYER.DMAC (bit 40).

Both port counters and ESDX counters are updated based on pipeline points and actions. For more information, see 4.28.5.1 Rewriter Pipeline Handling. The ESDX statistics pipeline points are fixed.

The following table shows when the counters are updated.

Table 4-246. Egress Statistics Update

Counter	Condition for Update
Port	Pipeline action is not Extract (action different from XTR, XTR_UPMEP, and LBK_ASM). Frame is transmitted (abort = 0). Updated for all egress port numbers including virtual device (VD) and extraction CPU port numbers.

.....continued	
Counter	**Condition for Update**
ESDX	The egress port is a physical port. The frame matches the parameters configured in CNT_CTRL.STAT_MODE. For extracted frames (Action: XTR, XTR_UPMEP and LBK_ASM): The egress statics is located in the pipeline point, PORT_CTRL.XTR_STAT_PIPELINE_PT. Frames extracted in or after this point are counted. For injected frames (Action: INJ, INJ_x, LBK_QS): The egress statics is located the pipeline point, PORT_CTRL.INJ_STAT_PIPELINE_PT. Frames injected in or before this point are counted. Frame is transmitted (abort = 0). The pipeline action is different from XTR_LATE_REW.
Abort	A frame is aborted by REW or OQS (abort = 1). Counting of frames aborted by REW must be enabled by setting CNT_CTRL.STAT_CNT_FRM_ABORT_ENA.

4.28.5.5 Protection Active

Using ES0, the rewriter can change various indexes, based on the value of the IFH.ENCAP.PROT_ACTIVE field in the IFH.

- Use the ES0 key field PROT_ACTIVE to change all ES0 actions when protection is active.
- The numbers of ES0 keys are limited, so the same key can optionally generate two different indexes based on the value of IFH.ENCAP.PROT_ACTIVE. This is enabled using the ES0 action PROTECTION_SEL

4.28.6 Mapping Tables

The rewriter contains two mapping resources (A and B) that can be used to look up the following fields.

- DSCP value in IP frames
- DEI in 802.3 VLAN tags
- PCP in 802.3 VLAN tags

The following table lists the configuration registers associated with the mapping tables.

Table 4-247. Mapping Table Configuration Registers

Register	Description	Replication
MAP_RES_x (x= A or B)		
MAP_VAL_x:TC_VAL	Mapped TC value	4096 × 2
MAP_VAL_x:DSCP_VAL	Mapped DSCP value	4096 × 2
MAP_VAL_x:DEI_VAL	Mapped DEI value	4096 × 2
MAP_VAL_x:PCP_VAL	Mapped PCP value	4096 × 2

Each of the two resources provide all the fields listed. Two lookups are done in each resource for every frame. This generates four mapping results per frame.

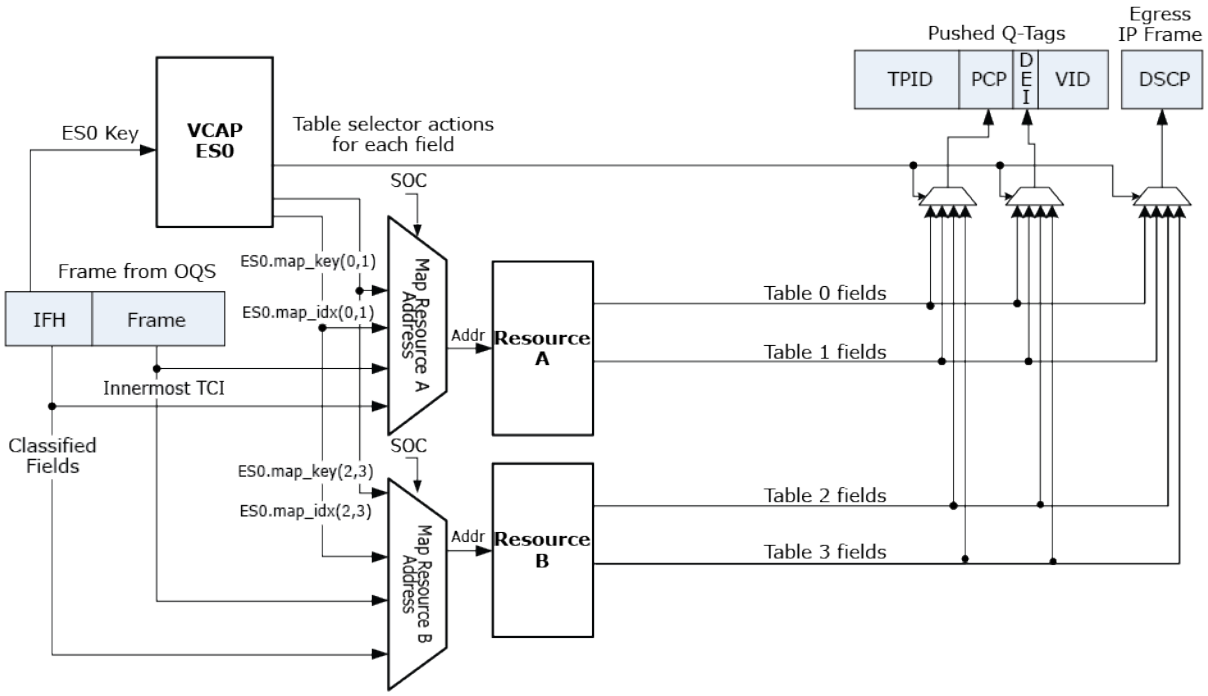
The four lookups implement mapping table 0 to 3. The address used for each lookup is controlled by the ES0 actions MAP_n_KEY and MAP_n_IDX (n = 0:3).

A mapping table address consists of two parts:

- A base index: ES0.MAP_n_IDX. This controls the MSB part of the table address.
- One of 15 different keys that use classified frame properties to generate the LSB of the table address. For more information about each key, see Table 4-235.

The following figure shows the mapping table functionality.

Figure 4-91. Mapping Tables



The mapping tables provide a flexible way to update frame fields. The following example shows how to configure and use mapping table 1.

Update PCP in VLAN tag A using mapping table 1. Use frame DSCP as index:

- Enable ES0
- Program an ES0 key
- Set ES0 action TAG_A_PCP_SEL to 5 for selected key
- Set ES0 action PUSH_OUTER_TAG = 1
- Update PCP by using Table 1:
  Select a base address in resource A. Base address 256 is used in this example
  Set ES0 action MAP_1_IDX = 256/8 = 32
- Use frame DSCP as key: Set ES0 action MAP_1_KEY = 2
- Program the PCP mapping in table 1: Addr = 256 to 256 + 63 (all DSCP values)
- Write REW:MAP_RES_A:$Addr/MAP_VAL_A PCP_VAL <New PCP>

Frame DSCP is used as key in this example. If the DSCP is not available (frame is non IP), the mapping is considered invalid and the PCP value is selected using this fallback method:

- Use mapping table 1 if key is valid, otherwise use mapping table 0.
- Use mapping table 0 if key is valid, otherwise use a fixed value.

In the example, a PCP value from table 0 is used instead for non IP frames, if the table 0 key is valid. Otherwise, the ES0 action PCP_A_VAL is used. Mapping tables 2 and 3 have the same functionality.

Mapping tables 0 and 1 share the 4096 addresses available in Resource A. Table 2 and 3 share 4096 addresses in Resource B.

4.28.7 VLAN Editing

The rewriter performs five steps related to VLAN editing for each frame and destination:

1. ES0 lookup: ES0 is looked up for each frame if enabled by the global configuration. The action from ES0 controls the pushing of VLAN tags.
2. VLAN popping: Zero to three VLAN tags are popped from the frame. Popping is controlled from the ANA by IFH.TAGGING.POP_CNT and by the action ES0.POP_VAL.
3. VLAN push decision: Decides the number of new tags to push and which tag source to use for each tag. Tag sources are port, ES0 (tags A to C), and routing. The analyzer can optionally enable pushing of ES0 VLAN tags A through C using the IFH.TAGGING.PUSH_CNT field.
4. Constructs the VLAN tags: The new VLAN tags are constructed based on the tag sources' configuration.
5. VLAN pushing: The new VLAN tags are pushed.

The outermost tag can optionally be controlled by port-based VLAN configuration.

Table 4-248. Port VLAN Tag Configuration Registers

Register	Description	Replication
TPID_CFG.TPID_VAL	Configures three custom TPID values. These must be configured identically in ANA_CL::VLAN_STAG_CFG.STAG_ETYPE_VAL.	3
PORT_VLAN_CFGPORT_*	Port VLAN TCI. Port _VID is also used for untagged set.	Ports
TAG_CTRL:TAG_CFG_OBEY_WAS_TAGGED	Controls how port tags are added. If this bit is set, the IFH field VSTAX.TAG.WAS_TAGGED must be 1 before a port tag is added to a frame.	Ports
TAG_CTRL:TAG_CFG	Controls port tagging. Four modes are supported.	Ports
TAG_CTRL:TAG_TPID_CFG	Selects Tag Protocol Identifier (TPID) for port tagging.	Ports
TAG_CTRL:TAG_VID_CFG	Selects VID in port tag.	Ports
TAG_CTRL:TAG_PCP_CFG	Selects PCP fields in port tag.	Ports
TAG_CTRL:TAG_DEI_CFG	Selects DEI fields in port tag.	Ports
DP_MAP.DP	Maps the four drop precedence levels (DP) to a drop eligible value (DE or color).	Global
PORT:PCP_MAP_DEx	Mapping table. Maps DP level and QoS class to new PCP values.	Per port per Q per DE
PORT:DEI_MAP_DEx	Mapping table. Maps DP level and QoS class to new DEI values.	Per port per Q per DE

4.28.7.1 ES0 Lookup

For each frame passing through the rewriter, an optional VCAP_ES0 lookup is done using the appropriate key. The action from ES0 is used in the following to determine the frame's VLAN editing.

4.28.7.2 VLAN Pop Decision

Popping of VLAN tags is controlled by both the rewriter and analyzer. The rewriter counts the number of Q-tags in the frame before any push- or pop operations are done, and this is the maximum number of tags that can be popped regardless of the IFH and ES0 pop control values.

The total number of popped tags (tot_pop_cnt) is calculated as described by the pseudo code below:

```
# frm_tags.num_qtags: Number of Q-TAGS in frame before any push or pop
operations are performed
# Delta pop count
if (ES0.ACTION.POP_VAL - IFH.TAGGING.PUSH_CNT) > 0 then
    d_pop_cnt = ES0.ACTION.POP_VAL - IFH.TAGGING.PUSH_CNT
else
    d_pop_cnt = 0
end if
# Total pop count
tot_pop_cnt = IFH.TAGGING.POP_CNT + d_pop_cnt
# Saturate pop count
if (IFH.TAGGING.POP_CNT + d_pop_cnt) > 3 then
    tot_pop_cnt = 3
end if
if tot_pop_cnt > frm_tags.num_qtags then
   tot_pop_cnt = frm_tags.num_qtags
end if
```

4.28.7.3 VLAN Push Decision

After popping the VLAN tags, the rewriter can push zero to three new VLAN tags into the outgoing frame. The number of tags to push is controlled by both the analyzer and the rewriter. The analyzer can set the IFH.TAGGING.PUSH_CNT to request pushing of ES0 TAG_A, TAG_B, or TAG_C. The rewriter pushes tags according to the port's tagging configuration in register TAG_CTRL.TAG_CFG and the action from a potential VCAP ES0 hit.

The total number of tags to push is calculated according to the pseudo code below:

```
# rew_push_cnt: Number of Q-tags pushed by ES0 actions, see Table 193, page 377 for
details

# Delta push count
if (IFH.TAGGING.PUSH_CNT - ES0.POP_VAL) > 0 then
  d_push_cnt = IFH.TAGGING.PUSH_CNT - ES0.POP_VAL
else
  d_push_cnt = 0
end if

# Total push count
tot_push_cnt = rew_push_cnt + d_push_cnt

# Saturate push count
if (rew_push_cnt + d_push_cnt) > 3 then
  tot_push_cnt = 3
end if
```

The analyzer can only push tags controlled by VACP_ES0 actions. The VCAP_ES0 action hit by a frame will determine how the pushed tags are formatted. If VCAP ES0 is disabled, the tag pushing cannot be controlled by the IFH.TAGGING.PUSH_CNT field.

The default value of IFH.TAGGING.PUSH_CNT is 0. The analyzer controls the value of this IFH field through VCAP CLM.

The number of tags pushed by the ES0 actions (rew_push_cnt) and the ES0 action POP_VAL (see pseudo code) determines how many additional tags that can be pushed by the analyzer. When the analyzer is used to modify the push count, the ES0 push actions should be configured as shown in the below table. Although it is possible to push tags combinations like A+C or B+C or C, that would be a misconfiguration in combination with IFH.TAGGING.PUSH_CNT.

Table 4-249. Recommended ES0 Actions when using IFH.TAGGING.PUSH_CNT

ES0 Tag Push Actions [1]	rew_push_cnt	d_push_cnt	tot_push_cnt	Tags Pushed
PUSH_OUTER_TAG = 3 PUSH_INNER_TAG = 0 PUSH_CUSTOMER_TAG = 0	0	0	0	No tags pushed
		1	1	TAG_A
		2	2	TAG_A, TAG_B
		3	3	TAG_A, TAG_B, TAG_C
PUSH_OUTER_TAG = 1 PUSH_INNER_TAG = 0 PUSH_CUSTOMER_TAG = 0 rew_push_cnt is 1 when TAG_A is pushed	1	0	1	TAG_A
		1	2	TAG_A, TAG_B
		2 or 3	3	TAG_A, TAG_B, TAG_C
PUSH_OUTER_TAG = 1 PUSH_INNER_TAG = 1 PUSH_CUSTOMER_TAG = 0 rew_push_cnt is 2 when TAG_B is pushed	2	0	2	TAG_A, TAG_B
		1, 2, or 3	3	TAG_A, TAG_B, TAG_C
PUSH_OUTER_TAG = 1 PUSH_INNER_TAG = 1 PUSH_CUSTOMER_TAG = 3 rew_push_cnt is 3 when TAG_C is pushed	3	0, 1, 2, or 3	3	TAG_A, TAG_B, TAG_C

Note:

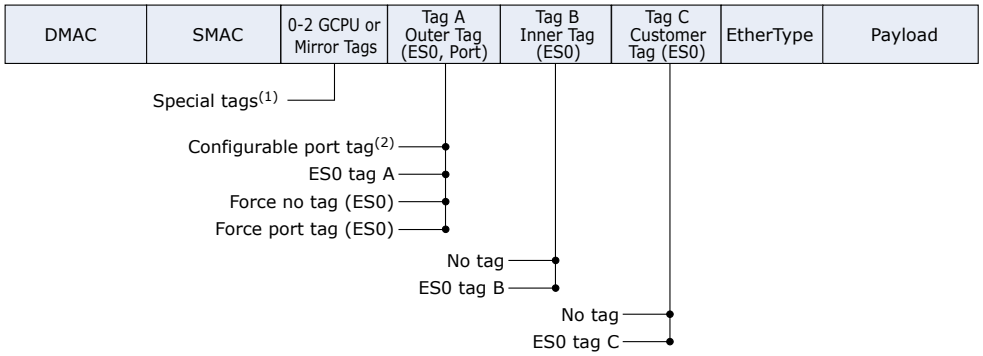
1. rew_push_cnt is the result of the settings in this column.

Q-tags are always pushed in the following order:

- Total push count = 1: Push TAG A (outer).
- Total push count = 2: Push TAG A (outer), TAG B (inner).
- Total push count = 3: Push TAG A (outer), TAG B (inner), TAG C (customer).

Two additional tags can be added for mirrored or GCPU frames. These special tags will always be the outermost ones.

Figure 4-92. VLAN Pushing



1. Mirror tag is controlled by REW::MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG.
   GCPU tag is controlled by REW::GCPU_CFG.GCPU_TAG_SEL.
2. Port tag is controlled by REW:PORT:TAG_CTRL.TAG_CFG.

The following table lists ES0 VLAN tag actions.

Table 4-250. ES0 VLAN Tag Actions

ES0_ACTION	TAG_CFG	Tagging Action
No ES0 hit	Controls port tag	The port tag is pushed according to TAG_CTRL.TAG_CFG as outermost tag. The following options are available: • Tag all frames with port tag. • Tag all frames with port tag, except if classified VID = 0 or classified VID = PORT_VLAN_CFG.PORT_VID. • Tag all frames with port tag, except if classified VID = 0. TAG_CTRL.TAG_VID_CFG selects between classified- or fixed-port VID. No inner tag.
PUSH_OUTER_TAG = 0 PUSH_INNER_TAG = 0	Controls port tag	Same as No ES0 hit action.
PUSH_OUTER_TAG = 1 PUSH_INNER_TAG = 0	Don not care	ES0 tag A is pushed as outer tag. No inner tag is pushed unless this is requested by the analyzer by setting IFH.TAGGING.PUSH_CNT
PUSH_OUTER_TAG = 2 PUSH_INNER_TAG = 0	Don not care	Port tag is pushed as outer tag. This overrules port settings in TAG_CFG. No inner tag.
PUSH_OUTER_TAG = 3 PUSH_INNER_TAG = 0	Don not care	No outer tag is pushed unless this is requested by the analyzer by setting IFH.TAGGING.PUSH_CNT.

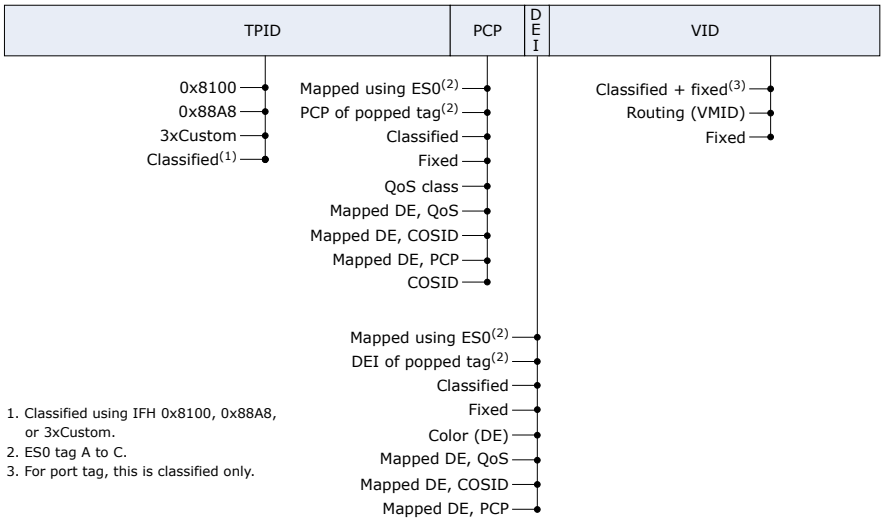
.....continued

ES0_ACTION	TAG_CFG	Tagging Action
PUSH_OUTER_TAG = 0 PUSH_INNER_TAG = 1	Controls port tag	Port tag is pushed according to TAG_CFG as outermost tag. The following options are available: • Tag all frames with port tag. • Tag all frames with port tag, except if classified VID = 0 or classified VID = PORT_TAG_DEFAULT.PORT_TCI_VID. • Tag all frames with port tag, except if classified VID = 0. • No port tag. TAG_CTRL.TAG_VID_CFG select between classified- or fixed-port VID. ES0 tag B is pushed as inner tag. ES0 tag B is effectively the outer tag if the port tag is not pushed.
PUSH_OUTER_TAG = 1 PUSH_INNER_TAG = 1	Don not care	ES0 tag A is pushed as outer tag. ES0 tag B is pushed as inner tag.
PUSH_OUTER_TAG = 2 PUSH_INNER_TAG = 1	Don not care	Port tag is pushed as outer tag. This overrules port settings in TAG_CFG. ES0 tag B is pushed as inner tag.
PUSH_OUTER_TAG = 3 PUSH_INNER_TAG = 1	Don not care	No outer tag is pushed. This overrules port settings in TAG_CFG. ES0 tag B is pushed as inner tag. ES0 tag B is effectively the outer tag, because no outer tag is pushed.
Tag C is controlled separately and will always be the inner most tag. All combinations of tags A to C are allowed.		
PUSH_CUSTOMER_TAG = 0	Controls port tag	Does not push customer tag (tag C) unless this is requested by the analyzer by setting IFH.TAGGING.PUSH_CNT.
PUSH_CUSTOMER_TAG = 1	Controls port tag	Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 1.
PUSH_CUSTOMER_TAG = 2	Controls port tag	Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 0.
PUSH_CUSTOMER_TAG = 3	Controls port tag	3: Push tag C if UNTAG_VID_ENA = 0 or (C-TAG.VID ! = VID_C_VAL) C-TAG.VID is controlled by TAG_C_VID_SEL.
UNTAG_VID_ENA	Controls port tag	See PUSH_CUSTOMER_TAG = 3.

4.28.7.4 Constructing VLAN Tags

The contents of the tag header are highly programmable when pushing a VLAN tag. The following figure shows the VLAN tag construction.

Figure 4-93. VLAN Tag Construction



The port tags (ES0.PUSH_OUTER_TAG = [0[2]), the ES0 tags A to C, and the special tags have individual configurations.

4.28.7.4.1 Port Tag: PCP

Use REW:PORT:TAG_CTRL.TAG_PCP_CFG to configure the following:

- Use the classified PCP.
- Use the egress port's port VLAN (PORT_VLAN_CFG.PORT_PCP).
- Use the QoS class directly.
- Map DE and QoS class to a new PCP value using the per-port table PCP_MAP_DEx.
- Map DE and COSID to a new PCP value using the per-port table PCP_MAP_DEx.
- Map DE and classified PCP to a new PCP value using the per-port table PCP_MAP_DEx.
- Use the COSID class directly.

4.28.7.4.2 Port Tag: DEI

Use REW:PORT_TAG_CTRL.TAG_DEI_CFG to configure the following:

- Use the classified DEI.
- Use the egress port's port VLAN (PORT_VLAN_CFG.PORT_DEI).
- Use mapped DP to DE level (color).
- Map DE and QoS class to a new DEI value using the per-port table DEI_MAP_DEx.
- Map DE and COSID to a new DEI value using the per-port table DEI_MAP_DEx.
- Map DE and classified PCP to a new DEI value using the per-port table DEI_MAP_DEx.

4.28.7.4.3 Port Tag: VID

Use REW:PORT:TAG_CTRL.TAG_VID_CFG to configure the following:

- Use the classified VID. This is either IFH.VSTAX.TAG.VID or IFH.ENCAP.GVID.
- Use the egress port's port VLAN (PORT_VLAN_CFG.PORT_VID).
- Use VMID from router leg lookup. When L3 routing is active, the classified VID is overwritten by the VMID.

4.28.7.4.4 Port Tag: TPID

Use REW:PORT:TAG_CTRL.TAG_TPID_CFG to configure the following:

- Use Ethernet type 0x8100 (C-tag).
- Use Ethernet type 0x88A8 (S-tag).

- Use one of three custom TPIDs programmed in REW::TPID_CFG.TPID_VAL.
- The TPID is classified by ANA and selected by IFH.
  Similar options for the ES0 tag A to tag C are available:

4.28.7.4.5 ES0 Tag: PCP

Use ES0 action TAG_x_PCP_SEL to configure the following:

- Use the classified PCP.
- Use ES0_ACTION.PCP_x_VAL.
- Use PCP of popped tag (IFH.VSTAX.TAG.WAS_TAGGED = 1 and tot_pop_cnt > 0) else PCP_x_VAL.
- Use the complex mapping functionality provided by ES0 to lookup a PCP value.

4.28.7.4.6 ES0 Tag: DEI

Use ES0 action TAG_x_DEI_SEL to configure the following:

- Use the classified DEI.
- Use ES0_ACTION.DEI_x_VAL.
- Use the complex mapping functionality provided by ES0 to lookup a DEI value.
- Use DEI of popped tag (IFH.VSTAX.TAG.WAS_TAGGED = 1 and tot_pop_cnt > 0) else DEI_x_VAL.
- Use the mapped DP to DE level (color) directly.

4.28.7.4.7 ES0 Tag: VID

Use ES0 action TAG_x_VID_SEL to configure the following:

- Tag A and B: Use the classified VID incremented with ES0.VID_x_VAL. This classified VID is either IFH.VSTAX.TAG.VID or IFH.ENCAP.GVID.
  Tag C: Use the classified VID. This classified VID is either IFH.VSTAX.TAG.VID or IFH.ENCAP.GVID.
- Use ES0_ACTION.VID_x_VAL.

4.28.7.4.8 ES0 Tag: TPID

Use ES0 action TAG_x_TPID_SEL to configure the following:

- Use Ethernet type 0x8100 (C-tag).
- Use Ethernet type 0x88A8 (S-tag).
- Use one of three custom TPIDs programmed in REW::TPID_CFG.TPID_VAL.
- TPID is classified by ANA and selected via IFH.

GCPU or mirror forwarding can optionally insert additional tags. These will always be added as the outermost tags. One or two special tags (A and B) can be added. The tag construction options as follows:

Special Tag A+B: PCP

- Use the classified PCP.
- Use fixed value.

Special Tag A+B: DEI

- Use fixed value.

Special Tag A+B: VID

- Use fixed value.

Special Tag A+B: TPID

- Use Ethernet type 0x8100 (C-tag).
- Use Ethernet type 0x88A8 (S-tag).
- Use one of three custom TPIDs programmed in REW::TPID_CFG.TPID_VAL.
- TPID is classified by ANA and selected by means of IFH.

4.28.8 DSCP Remarking

The rewriter supports two DSCP remarking schemes. The first remarking mode is controlled by port and IFH settings. The second remarking mode is controlled by ES0 actions and by the general mapping tables.

DSCP actions from ES0 override any remarking done by the port.

4.28.8.1 ES0-Based DSCP Remarking

ES0 controls DSCP remarking if ES0 lookups are enabled and the DSCP_SEL action is different from 0. For more information, see 4.28.6 Mapping Tables.

4.28.8.2 Legacy Port-Based Mode

Simple DSCP remapping can be done using a common table shared by all ports.

The following table shows the port-based DSCP editing registers associated with remarking.

Table 4-251. Port-Based DSCP Editing Registers

Register	Description	Replication
DP_MAP:DP	Maps the four drop precedence levels to a drop eligible value (DE). DE is also called color.	Global
DSCP_REMAP:DSCP_REMAP	Full one-to one DSCP remapping table common for all ports.	None
DSCP_MAP:DSCP_UPDATE_ENA	Selects DSCP from either frame or IFH.QOS.DSCP. If DSCP_MAP.DSCP_UPDATE_ENA = 1 and IFH.QOS.UPDATE_DSCP = 1: Selected DSCP = IFH.QOS.DSCP Else keep frame DSCP: Selected DSCP = Frame DSCP.	Ports
DSCP_MAP:DSCP_REMAP_ENA	Optionally remaps selected DSCP. If DSCP_MAP.DSCP_REMAP_ENA = 1 and IFH.QOS.TRANSPARENT_DSCP = 0: New DSCP = DSCP_REMAP [Selected DSCP]. Else do no remap: New DSCP = selected DSCP.	Ports

The following remarking options are available.

- No DSCP remarking. The DSCP value in the frame is untouched.
- Update the DSCP value in the frame with the value received from the analyzer in IFH.QOS.DSCP.
- Update the DSCP value in the frame with the frame DSCP mapped through DSCP_REMAP.DSCP_REMAP.
- Update the DSCP value in the frame with the value received from the analyzer (IFH.QOS.DSCP) mapped through DSCP_REMAP.DSCP_REMAP.

The ANA can set IFH.QOS.TRANSPARENT_DSCP to prevent DSCP mapping even if this is enabled for a port. The ANA can also force use of the frame DSCP for mapping by setting IFH.QOS.UPDATE_DSCP to 0.

The IP checksum is updated when changing the DSCP for IPv4 frames.

4.28.9 IEEE802.1BR Bridge Port Extension Tag

The rewriter can insert a Bridge Port Extension tag (E-TAG) into egress frames. The E-TAG is always pushed as the outermost tag of the Ethernet link layer. It can only be pushed when E-TAG awareness is enabled and when IFH.ENCAP.TYPE_AFTER_POP = ETH.

Table 4-252. E-Tag Configuration Registers

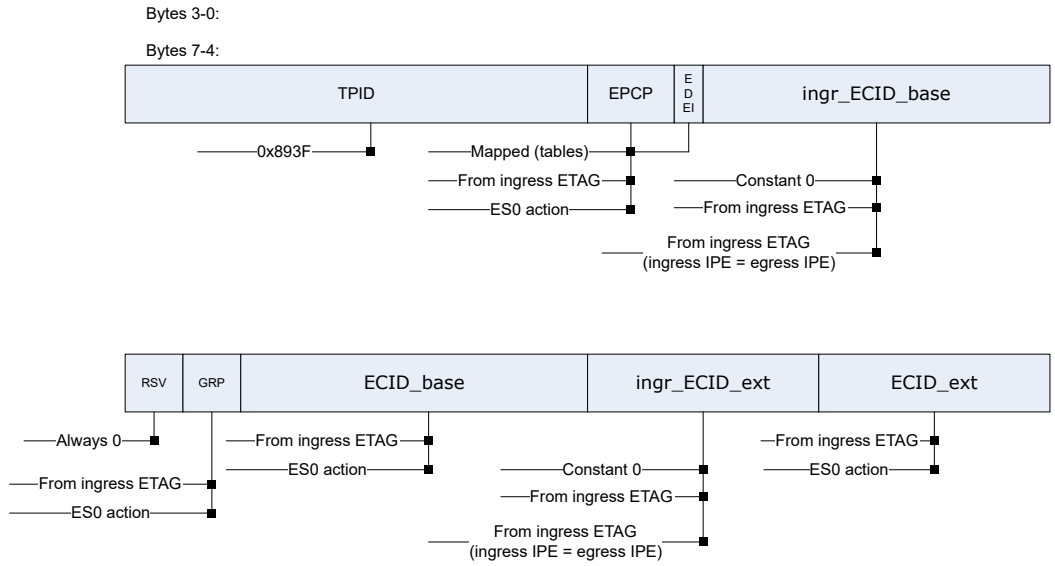
Register	Description	Replication
COMMON_CTRL.ETAG_TPID_ENA	Enable E-TAG awareness. If E-TAG aware, the frame's E-TAG is parsed and E-TAG information is available for E-TAG rewrites. If E-TAG unaware, the frame's E-TAG is not analyzed and the frame is considered a frame with ETYPE = 0x893F.	None

.....continued

Register	Description	Replication
RTAG_ETAG_CTRL. KEEP_ETAG	Controls IEEE802.1BR E-TAG handling. This bit must be set to 0 to allow full IEEE802.1BR E-TAG operation. When this bit is set to 1, the ingress E-TAG is kept unmodified in the frame when the IFH field IFH.TAGGING.RTAG_ETAG_AVAIL is set to 1. When IFH.TAGGING.RTAG_ETAG_AVAIL is 0, no E-TAG is pushed regardless of ES0 configuration	None
RTAG_ETAG_CTRL. IPE_TBL	Table to determine local ingress- and egress extension ports (L_IGR_PORT and L_EGR_PORT) when operating as an IEEE802.1BR controlling bridge. Use the unmapped port number, if the mapped value is outside port range (default). Logical ingress port. Mapping of IFH.SRC_PORT: L_IGR_PORT := IPE_TBL[IFH.SRC_PORT] Logical egress port. Mapping of cell bus port number: L_EGR_PORT := IPE_TBL[REW.CELLBUS.PORT_NUM]	None

The inserted E-TAG is constructed as shown in the below figure:

Figure 4-94. E-TAG Construction



4.28.10 FRER R-TAG

The rewriter supports insertion and removal of one IEEE802.1CB Frame Replication and Elimination for Reliability tag (R-TAG).

The rewriter is able to pop the ingress R-TAG and push a new egress R-TAG when R-TAG-awareness is enabled. The analyzer signals that an ingress frame contained an R-TAG by setting the IFH.TAGGING.RTAG_ETAG_AVAIL field.

Table 4-253. R-Tag Configuration Registers

Register	Description	Replication
COMMON_CTRL.RTAG_TPID_ENA	Enable R-TAG awareness. If R-TAG aware, any VLAN tags before or after the R-TAG are parsed (maximum two VLAN tags are supported together with an R-TAG). If R-TAG unaware, the frame's R-TAG is not analyzed and the frame is considered a frame with ETYPE=0xF1C1. Any VLAN tags before the R-TAG are still analyzed.	None

The R-TAG can be placed as outer, middle or inner tag. The rewriter will identify the R-TAG position and recognize up to three VLAN tags and one R-TAG in the frame. The analyzer has a similar functionality and will set the IFH.TAGGING.RTAG_ETAG_AVAIL field when a R-TAG is found in any of the positions described below.

The possible R-TAG locations in both ingress- and egress frames are:

- Triple tagged:

  DA-SA-QTAG-QTAG-QTAG-ETYPE. RTAG position = 0. No RTAG present (IFH.TAGGING.RTAG_ETAG_AVAIL = 0).

  DA-SA-RTAG-QTAG-QTAG-ETYPE. RTAG position = 1 (outer).

  DA-SA-QTAG-RTAG-QTAG-ETYPE. RTAG position = 2 (middle).

  DA-SA-QTAG-QTAG-RTAG-ETYPE. RTAG position = 3 (inner).
- Double tagged:

  DA-SA-QTAG-QTAG-ETYPE. RTAG position = 0. No RTAG present (IFH.TAGGING.RTAG_ETAG_AVAIL = 0).

  DA-SA-RTAG-QTAG-ETYPE. RTAG position = 1 (outer).

  DA-SA-QTAG-RTAG-ETYPE. RTAG position = 2 (middle).
- Single tagged:

  DA-SA-QTAG-ETYPE. RTAG position = 0. No RTAG present (IFH.TAGGING.RTAG_ETAG_AVAIL = 0).

  DA-SA-RTAG-ETYPE. RTAG position = 1 (outer).

The R-TAG operations done by the rewriter are controlled by two ES0 actions ES0.ACTION.RTAG_POP_ENA and ES0.ACTION.RTAG_PUSH_SEL. For more details, see, 4.28.5 VCAP_ES0 Lookup.

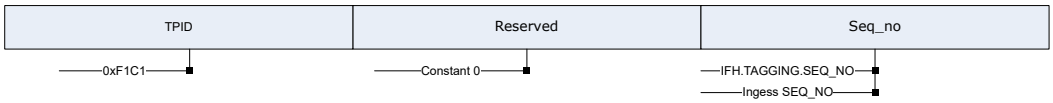
The R-TAG cannot be popped unless all VLAN tags in front of it are also popped. If the R-TAG is not popped it will remain unmodified in the egress frame regardless of the RTAG_PUSH_SEL configuration. The R-TAG is always popped if possible when RTAG_POP_ENA = 1 and IFH.RTAG_ETAG_AVAIL = 1.

Transparent R-TAG mode (no change to incoming frame in terms of R-TAG) is achieved by not configuring anything except COMMON_CTRL.RTAG_TPID_ENA. The transparent R-TAG is always pushed in the innermost tag position if VLAN tags are also pushed or remain in position if all VLAN tags in front of it is not popped.

The rewriter will push a new R-TAG into the egress frame when either the existing R-TAG is popped or when the ingress frame does not contain an R-TAG (IFH.RTAG_ETAG_AVAIL = 0). A maximum of 3 tags in total can be pushed by ES0. A pushed R-TAG will replace a pushed VLAN tag in the position configured by RTAG_PUSH_SEL. The possible R-TAG push positions are outer (1), middle (2), or inner (3).

RTAG_POP_ENA and RTAG_PUSH_SEL can be set at the same time, effectively replacing the SEQ_NO sequence number field with a new sequence number.

Figure 4-95. R-TAG Construction



4.28.11 VStaX Header Insertion

The rewriter controls insertion of the VStaX header. For more information about the header fields, see 4.2.3 VStaX Header.

The following registers are used for configuration of the rewriter specific stacking functionality.

Table 4-254. Rewriter VStaX Configuration Registers

Register	Description	Replication
COMMON_CTRL.OWN_UPSID	Configures own UPSID to be used for stacking.	None
CNT_CTRL.VSTAX_STAT_ESDX_DIS	Configures ESDX counting for stacking ports. If this bit is set and PORT_CTRL.VSTAX_HDR_ENA = 1, all counting based on the ESDX value is disabled regardless of the CNT_CTRL.STAT_MODE configuration.	None
PORT_CTRL.VSTAX_STACK_GRP_SEL	Selects logical stacking port (or stacking port group) membership.	Ports
PORT_CTRL.VSTAX_HDR_ENA	Enables insertion of stacking header in frame.	Ports
PORT_CTRL.VSTAX_PAD_ENA	Enables padding of frames to 76 bytes. Setting this bit will cause all frames on the port to be extended to 76 bytes instead of 64 bytes. This should only optionally be enabled for stacking ports (PORT_CTRL.VSTAX_HDR_ENA = 1). Setting this bit will prevent frames from becoming under sized in a receiving switch, when the VStaX header is removed. See ASM::ERR_STICKY.FRAGMENT_STICKY.	Ports
PORT_CTRL.VSTAX2_MIRROR_OBEY_WAS_TAGGED	Configures tagging of frames with VSTAX.GEN.FWD_MODE = VS2_FWD_GMIRROR. Only active on front ports for frames with this FWD_MODE. This is used to control the remote mirror tagging of frames that have been mirrored from one unit in the stack to another unit.	Ports
PORT_CTRL.VSTAX2_MISC_ISDX_ENA	Configures VSTAX MISC field decoding. Select between MISC- or AC mode.	Ports
VSTAX_PORT_GRP_CFG.VSTAX_TTL	Link TTL value.	2
VSTAX_PORT_GRP_CFG.VSTAX_LRN_ALL_HP_ENA	Changes priority of learn all frames to the highest priority (IFH.VSTAX.QOS = 7).	2

.....continued		
Register	**Description**	**Replication**
VSTAX_PORT_GRP_CFG.VSTAX_MODE	Controls whether forwarding modes specific to VStaX AF are translated to BF forwarding modes. If set to 0, the following translation is performed: fwd_logical -> fwd_lookup fwd_mc -> fwd_lookup If set to 1, no translation is performed. Translation is only required for advanced forwarding switches operating in a basic forwarding stack.	2

4.28.12 Forwarding to GCPU

The OQS can redirect CPU frames on selected CPU queues to the external ports instead of sending them to the internal CPU. This is called GCPU forwarding and is typically used if an external CPU is connected to the device through an Ethernet port.

GCPU forwarding is controlled per individual CPU queue. Stack ports have additional options for GCPU forwarded frames.

Table 4-255. GCPU Configuration Registers

Register	Description	Replication
GCPU_CFG.GCPU_KEEP_IFH	Used when GCPU frames are forwarded to a front port. Frames are sent with the IFH preserved. The IFH is encapsulated according to the configuration. Setting GCPU_KEEP_IFH to a value different from 0 overrides the GCPU_TAG_SEL and GCPU_DO_NOT_REW settings for front ports. No other rewrites are done to the frame. The GCPU_KEEP_IFH setting is not active if PORT_CTRL.KEEP_IFH_SEL is different from 0 or if PORT_CTRL.VSTAX_HDR_ENA = 1.	Per CPU Q
GCPU_CFG.GCPU_DO_NOT_REW	Used when GCPU frames are forwarded to a front port. Frames are not modified when forwarded to the GCPU except for the optional tags configured in GCPU_CFG.GCPU_TAG_SEL.	Per CPU Q
GCPU_CFG.GCPU_TAG_SEL	The destination port type determines the functionality. When a GCPU frame is forwarded to a stack port: Force a change of the VSTAX.TAG to the configured values in GCPU_TAG_CFG:0. When a GCPU frame is forwarded to a front port: Optionally add one or two Q-tags to the frame. The tags are configured using GCPU_TAG_CFG.	Per CPU Q
GCPU_CFG.GCPU_FWD_MODE	Used when GCPU frames are forwarded to a stack port. Configure forward mode of GCPU frames on stack ports by selecting value of VSTAX.GEN.FWD_MODE.	Per CPU Q

.....continued		
Register	**Description**	**Replication**
CPU_CFG.GCPU_UPSPN	Used when GCPU frames are forwarded to a stack port. CPU QUEUE to be used as destination queue for global CPU transmission. The value depends on configuration and the value of GCPU_CFG.GCPU_FWD_MODE. Controls the value of VSTAX.DST.DST_PN.	Per CPU Q
GCPU_CFG.GCPU_UPSID	Used when GCPU frames are forwarded to a stack port. UPSID to be used as destination in the VStaX header. Sets VSTAX.DST.DST_UPSID to configured value.	Per CPU Q
GCPU_TAG_CFG.TAG_TPID_SEL	Configuration of Q-Tags for GCPU frames. GCPU frames that are forwarded to a front port can optionally have one or two IEEE 802.3 VLAN tags added. The tags will be placed in the outer most position after the SMAC. GCPU VLAN tag Protocol Identifiers (TPID).	2
GCPU_TAG_CFG.TAG_VID_VAL	GCPU VLAN VID value.	2
GCPU_TAG_CFG.TAG_DEI_VAL	GCPU VLAN DEI values.	2
GCPU_TAG_CFG.TAG_PCP_SEL	Selection of GCPU VLAN PCP values.	2
GCPU_TAG_CFG.TAG_PCP_VAL	GCPU VLAN PCP values.	2

The IFH.MISC.CPU_MASK field is used to select the GCPU forwarding configuration. If multiple bits are set in this mask, the most significant bit is always used to control the GCPU forwarding. The ANA_AC::CPU_CFG.ONE_CPU_COPY_ONLY_MASK register is used to control how the CPU_MASK is set before the frame enters the rewriter. If the CPU_MASK field is 0, all GCPU forwarding is disabled.

GCPU frame forwarding must be enabled in the OQS per CPU queue using the QFWD::FRAME_COPY_CFG.FRMC_PORT_VAL registers.

4.28.13 IP-in-IPv4

The rewriter supports encapsulation and decapsulation of IP-in-IPv4 frames sent to the VD2 loopback port. The analyzer uses fields in the IFH to control which IP-in-IPv4 operations the rewriter must do to each frame on the VD2 interface. See 4.2.2 Internal Frame Header Layout for more details.

The frame is identified as an IP-in-IPv4 frame when the IFH field IFH.ENCAP.TYPE_AFTER_POP is CW and the first four bits of frame data located at position IFH.ENCAP.POP_W16_CNT*2 is either 4 (inner IPv4 frame) or 6 (inner IPv6 frame). The IFH.ENCAP.POP_W16_CNT field points to the inner IP header.

The IP-in-IPv4 flow requires that frames are looped back to the analyzer on the VD2 loopback port. The rewriter will only do the encapsulation and decapsulations rewrites when frames are sent to the VD2 interface.

IP-in-IPv4 decapsulation:

- The rewriter pops frame data located in front of the inner IP header and then adds a DMAC, SMAC and ETYPE to the frame. The version field of the inner IP header determines the value of ETYPE. This will decapsulate an IP-in-IPv4 frames.
- The analyzer replaces the inserted DMAC and SMAC when it receives the frame from the VD2 interface.

IP-in-IPv4 encapsulation:

- The rewriter pops frame data located in front of the inner IP header and then adds an outer IPv4 header to the frame when IFH.ENCAP.ENCAP_ID_RLEG_MODE = 1 and IFH.ENCAP.ENCAP_ID_RLEG > 0.
- The rewriter also adds DMAC, SMAC and ETYPE (0x0800) in front of the outer IPv4 header. This will create an IP-in-IPv4 frame.

The IFH.ENCAP.ENCAP_ID_RLEG field is used to lookup the outer IPv4 header in the ENCAP_IP4 table. The following table shows the IP-in-IPv4 encapsulation configuration registers.

Table 4-256. IP-in-IPv4 Encapsulation Configuration Registers

Register	Description	Replication
IPV4_ENCAP_CFG	The encapsulation table can be used to control IP-in-IPv4 encapsulations. Must be set to 1 when an entry is used for IP-in-IPv4 encapsulation	1,024
IP_HDR_CFG	Configures the outer IPv4 header of IP-in-IPv4 frames.	1,024
GRE_PROTOCOL_CFG	Enables and configures GRE header.	1,024
SIP_CFG	Source IPv4 address for outer IPv4 header in IP-in-IPv4 frames.	1,024
DIP_CFG	Destination IPv4 address for outer IPv4 header in IP-in-IPv4 frames.	1,024

4.28.14 Layer 3 Routing

The following registers are associated with routing (Layer 3, or L3). They are used when the analyzer signals routing by setting IFH.FWD.DST_MODE = L3UC_ROUTING or L3MC_ROUTING.

The analyzer can also signal routing using IFH.ENCAP.RT_FWD = 1. In this case, the MAC addresses and VMIDs are controlled directly by the analyzer, and the registers in the rewriter are not used.

Table 4-257. Routing SMAC Configuration Registers

Register	Description	Replication
RLEG_CFG_0.RLEG_MAC_LSB	Router Leg SMAC address used for IP routing (least significant bits). Must be set identical to ANA_L3::RLEG_0.RLEG_MAC_LSB.	None
RLEG_CFG_1. RLEG_MAC_TYPE_SEL	Configures how Router Leg MAC addresses are specified. Must be set identical to ANA_L3::RLEG_CFG_1.RLEG_MAC_TYPE_SEL. 0: RLEG_MAC:= RLEG_MAC_MSB(23:0) and (RLEG_MAC_LSB(23:0) + VMID(7:0) mod 2^{24}) 1: RLEG_MAC:= RLEG_MAC_MSB(23:0) and (RLEG_MAC_LSB(23:0) + VID(11:0) mod 2^{24} Others: RLEG_MAC:= RLEG_MAC_MSB(23:0) and RLEG_MAC_LSB(23:0) number of common address for all VLANs.	None
RLEG_CFG_1.RLEG_MAC_MSB	MSB part of SMAC used for IP routing. Must be set identical to ANA_L3::RLEG_MAC_MSB.	None

The IFH.DST.L3_[MC|UC]ROUTING.ERLEG fields are used to select the VID of routed frames using the Egress Mapped VLAN (VMID) configuration registers listed in the following table.

Table 4-258. L3 Routing Egress Mapping VLAN (EVMID)

Register	Description	Replication
RLEG_CTRL.RLEG_EVID	VID value assigned to this router leg.	511
RLEG_CTRL.DECAP_IRLEG	IP-in-IPv4 ingress router leg. Used when the outer IPv4 layer is removed.	511
RLEG_CTRL.RLEG_VSTAX2_WAS_TAGGED	Control the value of IFH.VSTAX.TAG.WAS_TAGGED field in the stack header for frames that are L3 forwarded to a stack port.	511

4.28.15 MAC Address Translation

The rewriter can be configured to translate the DMAC or SMAC address of a frame. The rewriter MAC table contains the MAC address translation configuration. It is indexed by the ES0 action ES0.MAC_IDX.

The MAC address translation is controlled by the REW_CMD received from the EACL block. For more details, see 4.24.2 VCAP ES2 Actions. The following translation commands are supported.

- Swap MAC addresses. Clear bit 40 in new SMAC.
- Translate the DMAC using VCAP_ES0 action.
- Translate the SMAC using VCAP_ES0 action.

The MAC translation table can also be used by the ES0 action REPLACE_DMAC_ENA. Using this action will cause a lookup in the translation table at the index ES0.MAC_IDX and an update of the frame DMAC.

The location of the MAC addresses in the frame is determined by the IFH.ENCAP.W16_POP_CNT field. The MAC addresses are only updated when IFH.ENCAP.TYPE_AFTER_POP = ETH.

The following table lists the registers for configuring the MAC address translation.

Table 4-259. MAC Address Translation Configuration Registers

Register	Description	Replication
MAC_TBL_CFG.MAC_REPL_OFFSET_VAL	Number of bits in frame's MAC counting from LSB which are not replaced if replacing the MAC address. For example, if set to 16, the 16 LSB bits in the frame's MAC address are not replaced while the 32 MSB bits are replaced with corresponding 32 bits from the associated MAC address.	128
EACL_MAC_HIGH.EACL_MAC_HIGH	MAC address bits 47:32 used for EACL MAC translation	128
EACL_MAC_LOW.EACL_MAC_LOW	MAC address bits 31:0 used for EACL MAC translation	128

4.28.16 Mirror Frames

The device offers three mirror probe sets that can be individually configured. The following table lists the rewriter configuration registers associated with mirroring.

Table 4-260. Mirror Probe Configuration Registers

Register	Description	Replication
MIRROR_PROBE_CFG.MIRROR_TX_PORT	The Tx port for each mirror probe (from where rewrite configuration was taken).	Mirror probes
MIRROR_PROBE_CFG.REMOTE_ENCAP_ID	Selects encapsulation ID to use for remote mirror frames.	Mirror probes

.....continued		
Register	**Description**	**Replication**
MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG	Enables encapsulation of mirrored frames. One or two Q-tags (Q-in-Q) or the encapsulation table can be used. In tag mode, the VLAN tags are added to the outer most position after the SMAC. In encapsulation mode, an entry in the ENCAP table is used for encapsulation. This overrides any encapsulation selected by ES0 for the frame.	
MIRROR_TAG_x_CFG:TAG_x_PCP_VAL	Mirror Q-tag PCP value.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_PCP_SEL	Selection of mirror Q-tag PCP values.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_DEI_VAL	Mirror Q-tag DEI values.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_VID_VAL	Mirror Q-tag VID values.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_TPID_SEL	The tag protocol identifier (TPID) for the remote mirror VLAN tag.	2× mirror probes

The device supports mirroring frames from either Rx ports (ingress mirror) or Tx ports (egress mirror). Frames either received on a port (Rx mirror) or frames sent to a port (Tx mirror) are copied to a mirror destination port.

For Tx mirroring, the rewriter rewrites the frame sent to the mirror destination port, exactly as the original frame copy is rewritten when forwarded to the mirror probe port. The mirror probe port number must be configured in the rewriter using register MIRROR_TX_PORT.

For Rx mirroring, the rewriter makes no modifications to the frame, so an exact copy of the original frame as received on the mirror probe port (ingress port) is sent out on the mirror destination port.

The analyzer controls both Rx and Tx mirroring. For more information, see 4.22.5 Mirroring. The rewriter obtains mirror information for the frame from the internal frame header. IFH.FWD.MIRROR_PROBE indicates if the frame is a mirror copy and if so, to which of three mirror probes the frame belongs. Encoding is as follows.

- MIRROR_PROBE = 0: Frame is not a mirror copy
- MIRROR_PROBE = 1: Frame is mirrored using mirror probe set 1
- MIRROR_PROBE = 2: Frame is mirrored using mirror probe set 2
- MIRROR_PROBE = 3: Frame is mirrored using mirror probe set 3

In addition, the IFH.FWD.RX_MIRROR bit indicates if the frame is Rx mirrored (1) or Tx mirrored (0).

When mirroring traffic to a device that is not directly attached to the device, it may be required to push one or two VLAN tags to mirrored frame copies as to not violate general VLAN membership for the mirror destination port. These VLAN tags are called “remote mirror VLAN tags” and can be configured individually for each mirror probe set. When enabled, the remote mirror VLAN tags are pushed onto all mirrored traffic by the rewriter whether it is an Rx mirrored frame or a Tx mirrored frame.

For Tx mirroring, care must be taken if maximum frame expansion is already required by the rewriter for the TX mirror probe port. In this case enabling remote mirror VLAN tags or encapsulation would add an extra frame expansion, which can cause frame disruption on the mirror destination port. It is not possible to add remote mirror encapsulation to a frame that already has encapsulation added by ES0. In this case the remote mirror encapsulation ID will replace the ID selected by the ES0.ENCAP_ID action.

4.28.17 Internal Frame Header Insertion

The rewriter can be configured to insert the internal frame header (IFH) into the frame upon transmission. For more information about the IFH, 4.2 Frame Headers.

Insertion of the IFH can be configured on any port, but it is intended for frames forwarded to the internal CPU and the NPI port (external CPU). When IFH insertion is enabled on a given port, all frames leaving that port contain the IFH. It is possible to change the IFH configuration for Virtual devices VD0 to VD2 and the CPU ports, but this should not be done for normal operation.

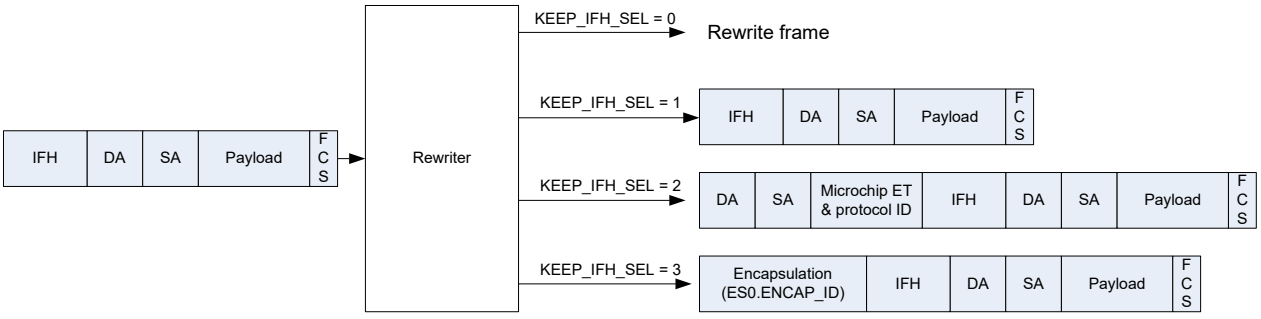
The following table lists the registers for configuring the insertion of extraction IFH into the frame.

Table 4-261. IFH Configuration Registers

Register	Description	Replication
PORT_CTRL.KEEP_IFH_SEL	Configures the rewriter IFH mode for the port. See also ASM::PORT_CFG.INJ_FORMAT_CFG. 0: Normal mode. 1: Keep IFH without modifications. Frames are not updated. IFH is kept. 2: Encapsulate IFH. The frame's DMAC, SMAC and a fixed TAG with ETYPE = 8880 (Microchip) and EPID=0x000B are inserted in front of the IFH: [FRM_DMAC][FRM_SMAC][0x8880][0x000B][IFH][FRAME]. 3: Encapsulate IFH using the ENCAP table. Use ES0 to generate an ENCAP_ID and insert the encapsulation in front of the IFH:[ENCAP][IFH][FRAME]	Ports
PORT_CTRL.KEEP_IFH_SEL_CPUVD	CPU and VD port IFH configuration. For normal operation the default value should not be changed. Encoding is 2 bits per port: Bits 1–0: Port 65 (CPU0) Bits 3–2: Port 66 (CPU1) Bits 5–4: Port 67 (VD0) Bits 7–6: Port 68 (VD1) Bits 9–8: Port 69 (VD2) Same encoding as PORT_CTRL.KEEP_IFH_SEL	—

The following figure shows the supported formats using KEEP_IFH_SEL.

Figure 4-96. Supported KEEP_IFH_SEL Formats



Note that for KEEP_IFH_SEL = 1, the frames transmitted on external ports are not valid Ethernet frames, because the extraction IFH precedes the DMAC. This format cannot be used for forwarding to a standard Ethernet device. The CPU ports, VD0 and VD1 keep the IFH and use KEEP_IFH_SEL = 1.

The frame formats using KEEP_IFH_SEL = 2 or KEEP_IFH_SEL = 3 are primarily used on a port that is directly attached to an external CPU. When the IFH is included using these frame formats, the IFH is included in the frame's FCS calculation and a valid Ethernet frame is transmitted.

4.28.18 Frame Injection from Internal CPU

To instruct the switch core how to process the frame, the internal CPU injects frames to switch core using the internal frame header. The format of injected frames from the internal CPU is similar to the KEEP_IFH_SEL = 1 format.

On arrival to the rewriter, such injected frames are no different from any other frames. However, the IFH contains information related to rewriter functionality specifically targeting efficient frame injection:

- Do not rewrite (IFH.FWD.DO_NOT_REW):
  If this injection IFH bit is set, the rewriter forward frames unchanged to the egress port no matter any other rewriter configurations, except IFH.FWD.update_fcs.
- Update frame check sequence (IFH.FWD.UPDATE_FCS):
  If this injection IFH bit is set, the rewriter updates the FCS before forwarding a frame to the egress port. By setting this bit in the Injection IFH, the internal CPU can leave it to switch core hardware to make sure the FCS value included with the frame is correct even though no other frame modifications are performed by the switch core.
- Swap MACs (IFH.ENCAP.SWAP_MAC): Swap MACs in the Ethernet link layer and clear bit 40 of the new SMAC before sending the frame to the egress port. Swapping the MACs will also force the frame FCS to be updated. Cannot be used if IFH.FWD.DO_NOT_REW = 1.

4.29 Disassembler

This section provides information about the disassembler (DSM) block. The disassembler is mainly responsible for disassembling cells into Taxi words and forwarding them to the port modules. In addition, it has several other responsibilities, such as:

- MAC Control sublayer PAUSE function with generation of PAUSE frames and stop forwarding traffic on reception of PAUSE frames
- Aging of frames that are stopped by flow control
- Controlling transmit data rate according to IEEE802.3ar
- Handle preemptable traffic

The following table lists replication terminology.

Table 4-262. Replication Terminology

Replication Terminology	Description	Replication Value
Per port	Per physical ports.	70
Per taxi	Per taxi bus	9

4.29.1 Setting Up Ports

The following table lists the registers associated with setting up basic ports.

Table 4-263. Basic Port Setup Configuration Registers

Target::Register.Field	Description	Replication
DSM::BUF_CFG.CSC_STAT_DIS	Disables collection of statistics in the disassembler.	Per port
DSM::DEV_TX_STOP_WM_CFG.DEV_TX_STOP_WM	Watermark for maximum fill level of port module TX buffer.	Per port
DSM::DEV_TX_STOP_WM_CFG.DEV10G_SHADOW_ENA	Enable low speeds for 10 G capable ports.	Per port

4.29.2 Maintaining the Cell Buffer

The cell buffer is responsible for aligning and buffering the cell data received from the rewriter before the data are forwarded on the Taxi bus.

The cell buffer can be flushed for each port separately.

The following table lists registers associated with configuring buffer maintenance.

Table 4-264. Buffer Maintenance Configuration Register

Target::Register.Field	Description	Replication
DSM::CLR_BUF.CLR_BUF	Flush the cell buffer.	Per port

Before a buffer is flushed, ensure that no data is forwarded to this port by stopping the QSYS forwarding and disabling FC/PFC so that frames waiting for transmission will be sent. FC can be disabled in DSM::RX_PAUSE_CFG.RX_PAUSE_EN.

4.29.3 Assigning Taxi Bus Bandwidth to Ports

Each Taxi bus is equipped with a calendar, which is used for bandwidth allocation towards the ports attached to the Taxi bus. A calendar configuration consists of up-to 64 slots, which may be allocated to ports or left unused. Each slot accounts for 1 clock cycle. The following table lists the registers associated with configuring the disassembler Taxi bus calendars.

Table 4-265. Registers for Configuring Disassembler Taxi Bus Calendar

Target::Register.Field	Description	Replication
DSM::TAXI_CAL_CFG.CAL_IDX	Calendar index	Per taxi
DSM::TAXI_CAL_CFG.CAL_CUR_LEN	Index of last entry	Per taxi
DSM::TAXI_CAL_CFG.CAL_CUR_VAL	Value of calendar at index CAL_IDX	Per taxi
DSM::TAXI_CAL_CFG.CAL_PGM_VAL	Value to write into calendar at CAL_IDX	Per taxi
DSM::TAXI_CAL_CFG.CAL_PGM_ENA	Enable programming of calendar	Per taxi

Programming sequence:

1. Write CAL_PGM_ENA to 1.
2. For each slot i in 0 to L-1 (where L is the length of the desired calendar) write first CAL_IDX to i and then PGM_VAL to the Taxi index that the slot is allocated to. In order to leave a slot empty, write PGM_VAL to a Taxi index that is not assigned to a port.
3. Write CAL_PGM_ENA to 0.

4.29.4 Setting Up MAC Control Sublayer PAUSE Function

This section provides information about setting up the MAC control sublayer PAUSE function.

4.29.4.1 PAUSE Frame Generation

The main sources for triggering generation of PAUSE frames are the port level and buffer level watermarks in the queue system. It is also possible to trigger PAUSE generation based on the analyzer policing state.

For watermark-based PAUSE generation, a hysteresis is implemented. FC is indicated when the level of used resources exceeds the high watermark and it is released when the level falls below the low watermark. The following table lists the registers associated with configuration PAUSE frame generation.

Table 4-266. PAUSE Frame Generation Configuration Registers

Target::Register.Field	Description	Replication
DSM::ETH_FC_CFG.FC_ANA_ENA	Controls generation of FC based on FC request from the analyzer.	Per port
DSM::ETH_FC_CFG.FC_QS_ENA	Controls the generation of FC based on FC request from the queue system.	Per port
DSM::MAC_CFG.TX_PAUSE_VAL	The pause_time as defined by IEEE802.3 Annex 31B2	Per port
DSM::MAC_CFG.TX_PAUSE_XON_XOFF	TX PAUSE zero on deassert.	Per port
DSM::MAC_ADDR_BASE_HIGH_CFG.MAC_ADDR_HIGH	Bits 47-24 of SMAC address used in MAC_Control frames.	Per port
DSM::MAC_ADDR_BASE_LOW_CFG.MAC_ADDR_LOW	Bits 23-0 of SMAC address used in MAC_Control frames.	Per port

DSM::MAC_CFG.TX_PAUSE_VAL defines the timer value for generated PAUSE frames. If the state does not change by half of the time specified in this bit group, a new PAUSE frame is generated.

To generate a zero value pause frame when the reason for pausing has disappeared, set DSM::MAC_CFG.TX_PAUSE_XON_XOFF to 1.

The DMAC of a generated pause frame is always the globally assigned 48-bit multicast address 01 80 C2 00 00 01.

The SMAC of a generated pause frame is defined by DSM::MAC_ADDR_BASE_HIGH_CFG.MAC_ADDR_HIGH and DSM::MAC_ADDR_BASE_LOW_CFG.MAC_ADDR_LOW.

4.29.4.2 Reaction on Received PAUSE Frame

The assembler detects received PAUSE frames and forwards the pause_time to the disassembler, which stops data transmission (if enabled) for the period defined by pause_time. If PFC is enabled for a port, then RX_PAUSE_EN must be disabled in the disassembler.

It is possible to configure the disassembler to not stop traffic locally but to instead forward the stop information to the queue system. This is configured by setting DSM::RX_PAUSE_CFG.FC_OBEY_LOCAL to 1.

Note that this behavior does not conform to IEEE 802.3, and the time until data transmission is stopped is increased.

The following table lists the registers associated with configuring PAUSE frame reception.

Table 4-267. PAUSE Frame Reception Configuration Registers

Target::Register.Field	Description	Replication
DSM::RX_PAUSE_CFG.RX_PAUSE_EN	Enables flow control in Rx direction.	Per port
DSM::RX_PAUSE_CFG.FC_OBEY_LOCAL	Configures whether flow control is obeyed locally in the disassembler or flow control information is sent to the queue system.	Per port

4.29.4.3 PFC Pause Frame Generation

When PFC is enabled for a port, PFC PAUSE frames are generated when requested by the queue system.

PFC is enabled on a per port basis in the disassembler. The queue system must also be setup to generate PFC requests towards the disassembler. Regular flow control and PFC cannot operate simultaneously on the same port. The available PFC configurations in the disassembler are listed in the following table.

Table 4-268. PFC PAUSE Frame Generation Configuration Registers

Target::Register.Field	Description	Replication
DSM::ETH_PFC_CFG.PFC_MIN_UPDATE_TIME	Minimum time between two PFC PDUs when PFC state changes after transmission of PFC PDU.	Per port
DSM::ETH_PFC_CFG.PFC_XOFF_MIN_UPDATE_ENA	After sending PFC PDU with flow control deasserted for all priorities, enforces a PFC_MIN_UPDATE_TIME delay before allowing transmission of next PFC PDU.	Per port
DSM::ETH_PFC_CFG.PFC_ENA	Enables PFC operation for the port.	Per port

4.29.5 Setting up Flow Control in Half-Duplex Mode

The following table lists the configuration registers for half-duplex mode flow control. For more information about setting up flow control in half-duplex mode, see 4.29.4.1 PAUSE Frame Generation.

Table 4-269. Half-Duplex Mode Flow Control Configuration Register

Target::Register.Field	Description	Replication
DSM::MAC_CFG.HDX_BACKPRESSURE	Enables HDX backpressure instead of FDX FC when FC is generated.	Per port

To enable half-duplex flow control, DSM::MAC_CFG.HDX_BACKPRESSURE must be set to 1. In this mode, the disassembler forwards the FC status to the port modules, which then collides incoming frames.

4.29.6 Setting Up Frame Aging

The following tables provide the aging configuration and status register settings. If frame aging is enabled, the disassembler discards frames that are stuck, for example, due to flow control or frames that are received from the queue system with an era value eligible for aging. Frames discarded by aging are counted in DSM::AGED_FRMS.AGED_FRMS_CNT.

Table 4-270. Aging Configuration Register

Target::Register.Field	Description	Replication
DSM::BUF_CFG.AGING_ENA	Enable aging of frames stuck in the disassembler buffer system for long periods.	Per port

Table 4-271. Aging Status Register

Target::Register.Field	Description	Replication
DSM::AGED_FRMS.AGED_FRMS_CNT	Count number of aged frames.	Per port

4.29.7 Setting Up Transmit Data Rate Limiting

The disassembler can limit the transmit data rate as specified in IEEE802.3ar, which defines the following three rate limiting methods. The disassembler allows enabling of any combination of the following three methods.

- Frame overhead
- Payload data rate
- Frame rate

The following table lists the registers associated with rating limiting.

Table 4-272. Rate Limiting Common Configuration Registers

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_ACCUM_MODE_ENA	Enables for accumulated rate limit mode.	Per port
DSM::TX_RATE_LIMIT_MODE.PAYLOAD_PREAM_CFG	Defines whether the preamble is counted as payload in txRateLimitPayloadRate and txRateLimitFrameRate mode.	Per port
DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG	Defines if what is configured as header size in TX_RATE_LIMIT_HDR_SIZE::TX_RATE_LIMIT_HDR_CFG is subtracted form the payload.	Per port
DSM::TX_RATE_LIMIT_MODE.IPG_SCALE_VAL	Scales the IPG calculated by txRateLimitFrameOverhead and/or txRateLimitPayloadRate by a power of 2.	Per port
DSM::TX_RATE_LIMIT_HDR_CFG.TX_RATE_LIMIT_HDR_SIZE	Defines how much of the frame is seen as header and not counted as payload.	per port

If more than one of the rate limiting modes previously mentioned are enabled, the additional inter-packet gap is the maximum of each of the values generated by one of the enabled modes. However, if only the frame overhead and the data payload data rate modes are enabled, the additional inter-packet gap can be calculated as the sum of the additional gaps retrieved from each of the two modes. To enable the function, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_ACCUM_MODE_ENA to 1.

If the preamble of a frame is not to be counted as payload, set DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG to 0.

In addition, if parts of the frame are to be seen as header and not counted as payload, set DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG to 1. DSM::TX_RATE_LIMIT_HDR_CFG.TX_RATE_LIMIT_HDR_SIZE defines how much of the frame that should be considered as header. Note that this register is global for all ports enabled by DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG. The payload configuration applies to Payload Data Rate and Frame Rate mode.

To enable bandwidth limiting down to very low data rates, the calculated IPG can be scaled. In other words, multiplied by a power of 2 in the range 2 to 1,024. By this it is possible to limit the data rate down to 2.3% (for 1536 byte frames).

4.29.7.1 Setting Up Frame Overhead

The following table lists the frame overhead configuration registers.

Table 4-273. Rate Limiting Frame Overhead Configuration Registers Overview

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_FRAME_OVERHEAD_ENA	Enable txRateLimitFrameOverhead mode.	Per port
DSM::RATE_CTRL.FRM_GAP_COMP	Inter-packet gap to be used.	Per port

To increase the minimum inter-packet gap for all packets, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_FRAME_OVERHEAD_ENA to 1, and set DSM::RATE_CTRL.FRM_GAP_COMP to a value between 13 to 255.

For more information about the scaling feature if the GAP must be above 255, see 4.29.7 Setting Up Transmit Data Rate Limiting.

4.29.7.2 Setting Up Payload Data Rate

The following table lists the payload data rate configuration registers.

Table 4-274. Rate Limiting Payload Data Rate Configuration Registers

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE. TX_RATE_LIMIT_PAYLOAD_RATE_ENA	Enable txRateLimitPayloadRate mode.	Per port
DSM::TX_IPG_STRETCH_RATIO_CFG. TX_FINE_IPG_STRETCH_RATIO	Stretch ratio.	Per port

To limit the data rate relative to the transmitted payload, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_PAYLOAD_RATE_ENA to 1.

The inter-packet gap increase can be expressed as:

$\Delta IPG = 2^S \times (256/R) \times L$

Where:

S: Scaling factor DSM::TX_RATE_LIMIT_MODE.IPG_SCALE_VAL

R: Stretch ratio DSM::TX_IPG_STRETCH_RATIO_CFG.TX_FINE_IPG_STRETCH_RATIO

L: Payload length, possibly modified according to DSM::TX_RATE_LIMIT_MODE.PAYLOAD_PREAM_CFG and DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG

Note: Total IPG is 12 byte standard IPG plus IPG.

Values for R must not be below 1152 or above 518143.

Values for S must not be above 10.

Fractional parts of the resulting IPG value are carried forward to the next IPG.

Higher values for S allow for finer target accuracy/granularity on the cost of an increased IPG jitter.

The utilization can be expressed as:

U = L / (L + IPG) or

U = R / (R + 2S × 256)

4.29.7.3 Setting Up Frame Rate

The following table lists the frame rate configuration registers.

Table 4-275. Rate Limiting Frame Rate Registers

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE. TX_RATE_LIMIT_FRAME_RATE_ENA	Enables txRateLimitFrameRate mode.	Per port
DSM::TX_FRAME_RATE_START_CFG. TX_FRAME_RATE_START	Frame rate start.	Per port

To define a minimum frame spacing, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_FRAME_RATE_ENA to 1.

At start of payload, a counter is loaded with the value stored in DSM::TX_FRAME_RATE_START_CFG.TX_FRAME_RATE_START.

What is counted as payload is defined by DSM::TX_RATE_LIMIT_MODE.PAYLOAD_PREAM_CFG and DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG.

With every payload byte transmitted, the counter is decremented by 1.

At end of frame, the GAP will be extended by the counter value, if above 12.

4.29.8 Error Detection

The following table lists the error detection status registers.

Table 4-276. Error Detection Status Registers

Target::Register.Field	Description	Replication
DSM::BUF_OFLW_STICKY.BUF_OFLW_STICKY	Cell buffer had an overflow.	Per port
DSM::BUF_UFLW_STICKY.BUF_UFLW_STICKY	Cell buffer had an underrun.	Per port
DSM::TX_RATE_LIMIT_STICKY.TX_RATE_LIMIT_STICKY	IPG was increased by one of the three rate limiting modes.	Per port.

If one or both of BUF_OFLW_STICKY and BUF_UFLW_STICKY sticky bits are set, the port module was set up erroneously.

TX_RATE_LIMIT_STICKY is set when an IPG of a frame is increased due to one of the three rate limiting modes.

4.30 Layer 1 Timing

There are four recovered clock outputs that provide timing sources for external timing circuitry in redundant timing implementations. The following tables list the registers and pins associated with Layer 1 timing.

Table 4-277. Layer 1 Timing Configuration Registers

Register	Description
HSIOWRAP::SYNC_ETH_CFG	Configures recovered clock output. Replicated per recovered clock output.
CLKGEN:LCPLL1:LCPLL1_SYNCE_CFG	Additional LCPLL recovered clock configuration.
SD_LANE[0-32]:SYNC_ETH_CFG	Additional recovered clock configurations. Replicated per SERDES.

The recovered clock outputs are mapped on GPIO overlaid functions. For more information, see 5.10.8.1 GPIO Overlaid Functions.

It is possible to recover receive timing from any of the serDes ports. For aggregated port interfaces line QSGMII and USXGMII links, timing must be recovered in the PHY, because the QSGMII/USXGMII link retimes during aggregation of data streams.

The HSIOWRAP:SYNC_ETH_CFG register provides per recovered clock divider configuration, allowing division before sending out clock frequency.

The recovered clock outputs also have individual divider configuration (through SD_LANE[0-32]:SYNC_ETH_CFG) to allow division of SerDes receive frequency. Note that division with 32/66 will result in a gapped clock that normally is not recommended for SyncE operation.

The recovered clocks are single-ended outputs. The suggested divider settings in the following tables are selected to make sure to not output too high frequency via the input and outputs. A maximum frequency of 161.33 MHz is allowed.

Table 4-278. Recovered Clock Settings for 1 GE or Lower

Lane	Output Frequency	Register Settings for Output *n*, Lane *m*
0–32	25 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=4, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=0

.....continued		
Lane	**Output Frequency**	**Register Settings for Output *n*, Lane *m***
0–32	31.25 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=1, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=0

Table 4-279. Recovered Clock Settings for 2.5 GE

Lane	Output Frequency	Register Settings for Output *n*, Lane *m*
0–32	31.25 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=4, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=1
0–32	62.50 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=4, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=0

Table 4-280. Recovered Clock Settings for 5 GE

Lane	Output Frequency	Register Settings for Output *n*, Lane *m*
0–12	20.1416015625 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=2, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=1
13–32	20.1416015625 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=0 and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=1
0–12	80.56640625 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=6, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=0
13–32	80.56640625 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=6, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=0

Table 4-281. Recovered Clock Settings for 10 GE

Lane	Output Frequency	Register Settings for Output *n*, Lane *m*
13–32	20.1416015625 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=1, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=1
13–32	80.56640625 MHz	SYNC_ETH_CFG[n].RECO_CLK_ENA=1, SYNC_ETH_CFG[n].SEL_RECO_CLK_DIV=6, and SYNC_ETH_CFG[n].SEL_RECO_CLK_SRC=(DEV+1) SD_LANE[m]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG. SD_RECO_CLK_DIV=1

It is possible to automatically squelch the clock output when the device detects a loss of signal or PCS loss of lock on an incoming data stream. This can be used for failover in external timing recovery solutions. Auto squelching is enabled by setting SD_LANE[n]:SYNC_ETH_CFG:SYNC_ETH_SD_CFG.SD_AUTO_SQUELCH_ENA.

The signal detect pin can also be used to squelch the recovered clock. This is enabled by setting SD_LANE[n]:SD_CFG:SD_CFG.SD_ENA and should be used when the ports are connected to optical modules.

When squelching the clock, it will stop when detecting loss of signal (or PLL lock). The clock will stop on either on high or low level.

4.31 VRAP Engine

The Versatile Register Access Protocol (VRAP) engine allows external equipment to access registers in the device through any of the Ethernet ports on the device. The VRAP engine interprets incoming VRAP requests, and executes read, write, and read-modify-write commands contained in the frames. The results of the register accesses are reported back to the transmitter through automatically generated VRAP response frames.

The device supports version 1 of VRAP. All VRAP frames, both requests and responses, are standard Ethernet frames. All VRAP protocol fields are big-endian formatted.

The registers listed in the following table control the VRAP engine.

Table 4-282. VRAP Registers

Register	Description	Replication
ANA_CL:PORT:CAPTURE_CFG. CPU_VRAP_REDIR_ENA	Enable redirection of VRAP frames to CPU.	Per port
ANA_CL::CPU_PROTO_QU_CFG. CPU_VRAP_QU	Configure the CPU extraction queue for VRAP frames.	None
QFWD:SYSTEM:FRAME_COPY_CFG. FRMC_PORT_VAL	Map VRAP CPU extraction queue to a CPU port. One CPU port must be reserved for VRAP frames.	Per CPU extraction queue
DEVCPU_QS:XTR:XTR_GRP_CFG.MODE	Enable VRAP mode for reserved CPU port.	Per CPU port
DEVCPU_QS:INJ:INJ_GRP_CFG.MODE	Enable VRAP mode injection for reserved CPU port.	Per CPU port
ASM:CFG:PORT_CFG.INJ_FORMAT_CFG	The IFH without prefix must be present for VRAP response frames.	Per port
ASM:CFG:PORT_CFG.NO_PREAMBLE_ENA	Expect no preamble in front of IFH and frame.	Per port

.....continued		
Register	**Description**	**Replication**
DEVCPU_GCB::VRAP_ACCESS_STAT	VRAP access status.	None

The VRAP engine processes incoming VRAP frames that are redirected to the VRAP CPU extraction queue by the basic classifier. For more information about the VRAP filter in the classifier, see 4.14.1.11 CPU Forwarding Determination.

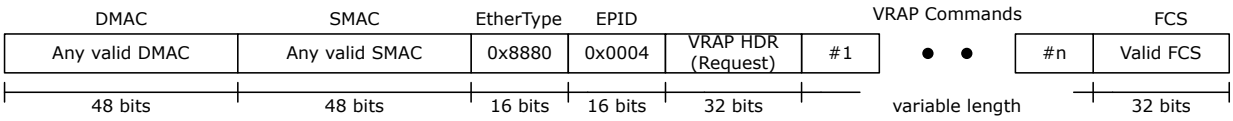
The VRAP engine is enabled by allocating one of the two CPU ports as a dedicated VRAP CPU port (DEVCPU_QS:XTR:XTR_GRP_CFG.MODE and DEVCPU_QS:INJ:INJ_GRP_CFG.MODE). The VRAP CPU extraction queue (ANA_CL::CPU_PROTO_QU_CFG.CPU_VRAP_QU) must be mapped as the only CPU extraction queue to the VRAP CPU port (QFWD:SYSTEM:FRAME_COPY_CFG.FRMC_PORT_VAL). In addition, the VRAP CPU port must enable the use of IFH without prefix and without preamble (ASM::PORT_CFG)

The complete VRAP functionality can be enabled automatically at chip startup by using special chip boot modes. For more information, see 5.1 VCore Configurations.

4.31.1 VRAP Request Frame Format

The following figure shows the format of a VRAP request frame.

Figure 4-97. VRAP Request Frame Format



VRAP request frames can optionally be VLAN tagged with one VLAN tag.

The EtherType = 0x8880 and the Ethernet Protocol Identifier (EPID) = 0x0004 identify the VRAP frames. The subsequent VRAP header is used in both request and response frames.

The VRAP commands included in the request frame are the actual register access commands. The VRAP engine supports the following five VRAP commands:

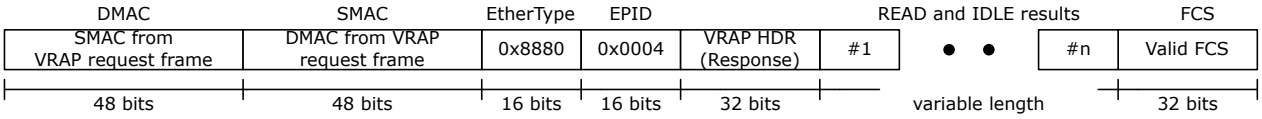
- READ: Returns the 32-bit contents of any register in the device.
- WRITE: Writes a 32-bit value to any register in the device.
- READ-MODIFY-WRITE: Does read/modify/write of any 32-bit register in the device.
- IDLE: Does not access registers; however, it is useful for padding and identification purposes.
- PAUSE: Does not access registers, but causes the VRAP engine to pause between register access.

Each of the VRAP commands are described in the following sections. Each VRAP request frame can contain multiple VRAP commands. Commands are processed sequentially starting with VRAP command 1, 2, and so on. There are no restrictions on the order or number of commands in the frame.

4.31.2 VRAP Response Frame Format

The VRAP response frame follows the VRAP request frame in terms of VLAN tagging: If the VRAP request was tagged, the VRAP response is also tagged using the same VLAN.

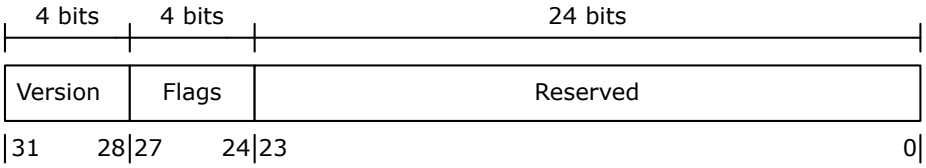
Only the READ and IDLE commands require data to be returned to the transmitter of the VRAP request frame. However, even if the VRAP request frame does not contain READ and IDLE commands, a VRAP response frame is generated with enough padding data (zeros) to fulfill the minimum transfer unit size.



4.31.3 VRAP Header Format

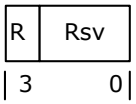
Both VRAP request and VRAP response frames contain a 32-bit VRAP header. The VRAP header is used to identify the protocol version and to differentiate between VRAP requests and VRAP response frames. The device supports VRAP version 1. A valid VRAP request frame must use Version = 1 and Flags.R = 1. Frames with an invalid VRAP header are discarded and not processed by the VRAP engine. The following figure shows the VRAP header.

Figure 4-98. VRAP Header Format



Version:
Set to 0x1 in VRAP response frames.
VRAP request frames are ignored if Version <> 1

Flags:
4 bits are used for Flags. Only one flag is defined:



R: 0=Request, 1=Response
Rsv: Set to 0 in VRAP response frames and ignored in VRAP request frames.

Reserved:
Set to 0 in VRAP response frames and ignored in VRAP request frames.

4.31.4 VRAP READ Command

The VRAP READ command returns the contents of any 32-bit register inside the device. The 32-bit read-data result is put into the VRAP response frame.

The READ command is 4 bytes wide and consists of one 32-bit address field, which is 32-bit aligned. The 2 least significant bits of the address set to 01. The following figures show the request command and the associated response result.

Figure 4-99. READ Command

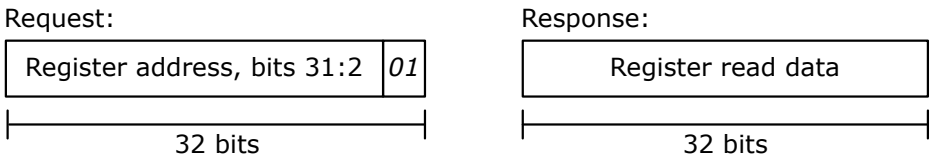
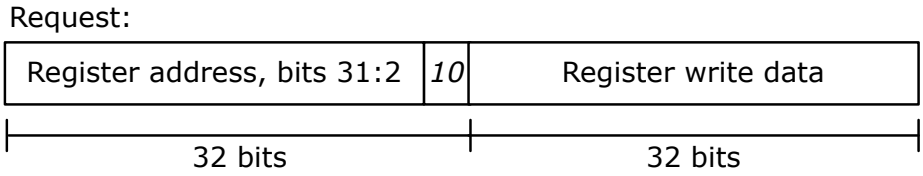


Figure 4-100. WRITE Command

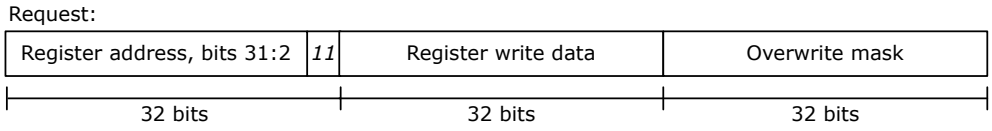


4.31.5 VRAP READ-MODIFY-WRITE Command

The READ-MODIFY-WRITE command does read/modify/write-back on any 32-bit register inside the device.

The READ-MODIFY-WRITE command is 12 bytes wide and consists of one 32-bit address field, which is 32-bit aligned. The two least significant bits of the address set to 11 followed by one 32-bit write-data field followed by one 32-bit overwrite-mask field. For bits set in the overwrite mask, the corresponding bits in the write data field are written to the register while bits cleared in the overwrite mask are untouched when writing to the register. The following figure shows the command.

Figure 4-101. READ-MODIFY-WRITE Command

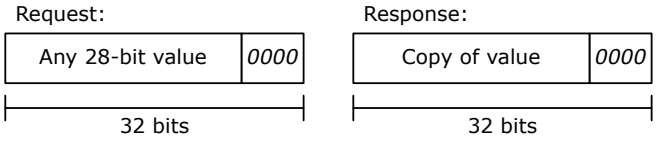


4.31.6 VRAP IDLE Command

The IDLE command does not access registers in the device. Instead it just copies itself (the entire command) into the VRAP response. This can be used for padding to fulfill the minimum transmission unit size, or an external CPU can use it to insert a unique code into each VRAP request frame so that it can separate different replies from each other.

The IDLE command is 4 bytes wide and consists of one 32-bit code word with the four least significant bits of the code word set to 0000. The following figure depicts the request command and the associated response.

Figure 4-102. IDLE Command

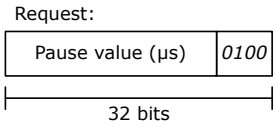


4.31.7 VRAP PAUSE Command

The PAUSE command does not access registers in the device. Instead it causes the VRAP engine to enter a wait state and remain there until the pause time has expired. This can be used to ensure sufficient delay between VRAP commands when this is needed.

The PAUSE command is 4 bytes wide and consists of one 32-bit code word with the four least significant bits of the code word set to 0100. The wait time is controlled by the 28 most significant bits of the wait command. The time unit is 1 us. The following figure depicts the PAUSE command.

Figure 4-103. PAUSE Command

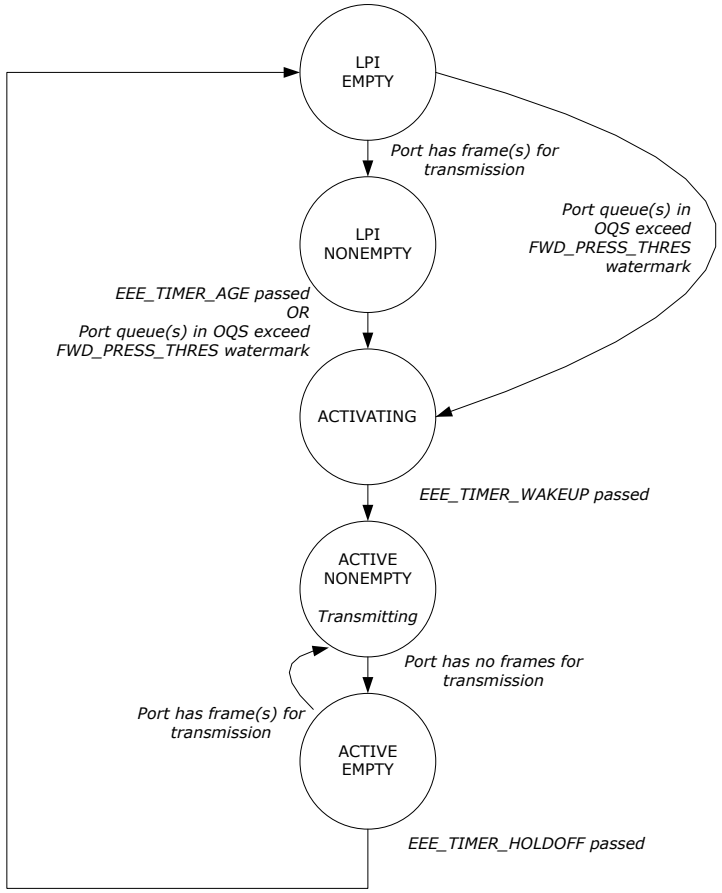


4.32 Energy Efficient Ethernet

The Ethernet ports support Energy Efficient Ethernet (EEE) to reduce power consumption when there is little or no traffic on a port.

The state diagram in the following figure shows the EEE LPI algorithm implemented in the device.

Figure 4-104. EEE State Diagram



- LPI_EMPTY: Low power mode. Port has no frames for transmission.
- LPI_NONEMPTY: Low power mode. Port has frames for transmission, but the number of frames is very limited (below EEE_URGENT_THRES in OQS) or EEE_TIMER_AGE has not passed yet.
- ACTIVATING: Port is being activated prior to restarting frame transmission. The timer EEE_TIMER_WAKEUP controls the length of the activation time.
- ACTIVE_NONEMPTY: Port is active and transmitting. This is the normal state for ports without EEE enabled.
- ACTIVE_EMPTY: Port is active, but currently has no frames for transmission. If frames for transmission do not become available within EEE_TIMER_HOLDOFF, then the port enters low power mode (in state LPI EMPTY).

The state transition timing values are configured in the EEE_CFG register per port module, and the levels at which the queue system asserts forward pressure are configured in the QSYS::EEE_CFG and QSYS::EEE_THRES registers.

As illustrated, the “port has frames for transmission” condition is true if there are frames for transmission and the scheduling algorithm allows transmission of any of the available frames. So it is possible that a port enters LPI_EMPTY or stays in LPI_EMPTY, even though there are frames in one or more queues.

It is also possible, though not likely, that forward pressure is set for one or more queues when going into LPI_EMPTY. This occurs when the scheduling algorithm has decided to transmit from other queue or queues than those signalling forward pressure and the transmission of these frames causes the port leaky bucket to close.

The “forward pressure signalled for queues” condition does not look at whether the queues with forward pressure are actually allowed to transmit.

For ports that are shaped to a low bandwidth, the forward pressure watermark in OQS should be set a low value, such that it is mainly the port shaping that controls when to exit LPI.

4.33 CPU Injection and Extraction

This section provides an overview of how the CPU injects and extracts frames to and from the switch core.

The switch core forwards CPU extracted frames to eight CPU extraction queues that per queue are mapped to one of the two CPU ports (by means of QFWD:SYSTEM:FRAME_COPY_CFG[0:7]) that the CPU can access. For each CPU extraction queue, it is possible to specify the egress priority (QFWD:SYSTEM:FRAME_COPY_CFG.FRMC_QOS_ENA). For each CPU port, there is strict priority selection between the egress queues.

For injection, there are two CPU injection groups that can simultaneously forward frames into the analyzer. The CPU can access the injection and extraction groups through either a Frame DMA or direct register access.

For more information about injecting and extracting CPU traffic through front ports used as NPI port, see 4.2.1 Internal Frame Header Placement; 4.9.2 Setting Up a Port for Frame Injection; and 4.28.12 Forwarding to GCPU.

4.33.1 Frame Injection

The traffic at any injection pipeline point from CPU, NPI, and AFI using the IFH (IFH.MISC.PIPELINE_ACT:= INJ_MASQ and IFH.MISC.PIPELINE_PT: = <value>).

When a frame is injected at a given pipeline point, it is logically not processed in the flow before reaching the injection point. For example, a frame injected at the ANA_CLM pipeline point bypasses VRAP detection and basic classification, which requires the frame to be injected with an IFH configured with all the fields normally set by basic classification.

Injecting a frame in a rewriter pipeline point causes the frame to bypass the complete analyzer and destination port must be specified in IFH (IFH.MISC.DPORT = <egress port>).

Frames can only be injected as unicast, which allows injection into a single egress port (configured in IFH.MISC.DPORT) with all IFH.ENCAP operators available in the rewriter such as OAM and PTP detection and hardware handling (if IFH.FWD.DST_MODE is set to ENCAP). Frames cannot be injected as multicast with multiple egress ports. This must be solved using multiple unicast injections.

4.33.1.1 Masqueraded Injection

The device supports masqueraded injection of frames from CPU where processing of the frame is done as if the frame was received on the masqueraded port (IFH.MISC.PIPELINE_ACT=INJ_MASQ and IFH.FWD.SRC_PORT=<masqueraded port>). A masquerade-injected frame sees the same source filter, classification, and learning as the physical port being masqueraded.

4.33.1.2 Injection Pipeline Points

The device supports specifying where in the processing flow a frame is injected by specifying a pipeline point for the frame (IFH.MISC.PIPELINE_PT). The frame will then only be processed after the specified injection point.

4.33.2 Frame Extraction

The CPU receives frames through eight CPU extraction queues. These extraction queues can be mapped to one of the two CPU ports or any of the front ports (QFWD:SYSTEM:FRAME_COPY_CFG[0-7]). The CPU extraction queues use memory resources from the shared queue system and are subject to the thresholds and congestion rules programmed for the CPU extraction queues and the shared queue system in general.

A frame can be extracted for multiple reasons. For example, a BPDU with a new source MAC address can be extracted as both a BPDU and a learn frame with two different CPU extraction queues selected. By default, the highest CPU extraction queue number receives a copy of the frame, but it is also possible to generate multiple frame copies to all the selected CPU extraction queues (ANA_AC::CPU_CFG.ONE_CPU_COPY_ONLY_MASK).

The CPU can read frames from the CPU port in two ways:

- Reading registers in the CPU system..
- Using the Frame DMA in the CPU system. For more information, see 5.7 Frame DMA.

CPU extracted frames include an IFH (36-bytes) placed in front of the DMAC. The IFH contains relevant side band information about the frame, such as the frame's classification result (VLAN tag information, DSCP, QoS class) and the reason for sending the frame to the CPU. For more information about the contents of the IFH, 4.2 Frame Headers.

4.33.3 Forwarding to CPU

Several mechanisms can be used to trigger redirection or copying of frames to the CPU. The following blocks can generate traffic to a CPU queue.

- Analyzer classifier ANA_CL
- Analyzer Layer 3 ANA_L3
- Analyzer Access Control Lists ANA_ACL
- Analyzer Layer 2 ANA_L2
- Analyzer Access Control ANA_AC
- Rewriter

Frames copied or redirected to the CPU from both the rewriter and the Up-MEPs are looped and passed another time through the analyzer.

4.33.4 Automatic Frame Injection (AFI)

The AFI block supports repeated (automated) injection, which requires the injection frame to be sent to the AFI block with a single IFH bit set (IFH.FWD.AFI_INJ) and with an IFH header configured as for normal CPU injection.

After an injection frame is sent to the AFI block, the frame can be added for AFI injection. For more information, see 4.27.6 Adding Injection Frame.

Frames injected from the AFI block are treated as any CPU injection, where IFH fields controls how injection occurs.

An Up-MEP injected frame must be masqueraded to appear as being received at the right ingress port (by setting IFH.MISC.PIPELINE_ACT = INJ_MASQ and IFH.FWD.SRC_PORT = <masqueraded port>).

A Down-MEP injected frame must be directly injected into the right egress port as shown in the following figure.

Frames from the AFI can either be injected directly into an egress port or they can appear as if they were received on an ingress port (masquerading). Masqueraded frames are subsequently processed by the analyzer like any other frame received on the port.

4.34 Priority-Based Flow Control (PFC)

Priority-based flow control (PFC, IEEE 802.1Qbb) is supported on all ports for all QoS classes. The device provides independent support for transmission of pause frames and reaction to incoming pause frames, which allows asymmetric flow control configurations.

4.34.1 PFC Pause Frame Generation

The queue system monitors the congestion per ingress and egress queue. If a flow control condition is asserted, the queue system forwards the congestion status to the disassembler. The disassembler stores the congestion status per (port, priority). Based on the congestion status, the disassembler will transmit PFC frames for each port.

4.34.1.1 Queue System

The queue system has a set of dedicated PFC watermarks. If the memory consumption of a queue exceeds a PFC watermark, a flow control condition is asserted, and the updated congestion status is forwarded the disassembler.

4.34.1.2 Disassembler

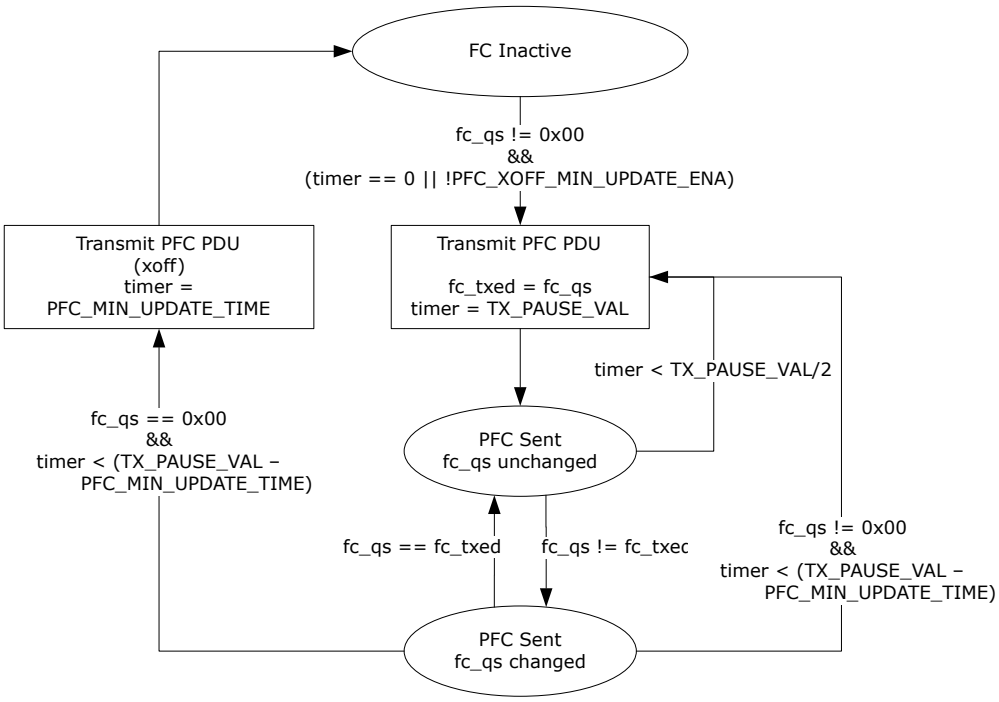
The disassembler (DSM) stores the congestion status per queue (per port, per priority) and generates PFC PDU based on the information. For more information about the available PFC configuration registers, see 4.29.4.3 PFC Pause Frame Generation.

The following figure shows the algorithm controlling the generation of PFC frames for each port. The 8-bit congestion state from the queue system is fc_qs. TX_PAUSE_VAL and PFC_MIN_UPDATE_TIME are configurable per port. The TX_PAUSE_VAL configuration is shared between LLFC and PFC.

When a priority gets congested, a PFC PDU is generated to flow control this priority. When half of the signaled timer value has expired, and the queue is still congested, a new PFC PDU is generated. If other priorities get congested while the timer is running, then PFC_MIN_UPDATE_TIME is used to speed up the generation of the next PFC PDU. The timer is shared between all priorities per port.

Every PFC PDU signals the flow control state for all priorities of the port. In other words, the priority_enable_vector of the PFC PDU is always set to 0x00ff, independent of whether or not a priority is enabled for flow control. For disabled priorities, the pause time value is always set to 0.

Figure 4-105. PFC Generation per Port



fc_qs: Congestion state per priority from QS

4.34.1.3 Statistics

The number of transmitted pause frames is counted in the TX_PAUSE_CNT counter and shared with the counter for transmitted link-layer flow control pause frames. Statistics are handled locally in the DEV10G port modules. For other port module types, the assembler handles the statistics.

4.34.2 PFC Frame Reception

Received PFC frames are detected by the assembler. The status is forwarded to the queue system, which will pause the egress traffic according to the timer value specified in the pause frames.

4.34.2.1 Assembler

The assembler identifies PFC PDUs and stores the received pause time values in a set of counters. The counters are decremented according to the link speed of the port. The assembler signals a stop indication to the queue system for all (port, priority) with pause times different from 0.

For information about the PFC configuration of the assembler, see 4.9.4 Setting Up PFC.

4.34.2.2 Queue System

The queue system stores the PFC stop indications received from the assembler per (port, priority). It pauses the transmission for all egress queues with an active PFC stop indication. The queue system does not require any PFC configuration.

4.34.2.3 Statistics

PFC pause frames are recognized as control frames and counted in the RX_UNSUP_OPCODE_CNT counter. Statistics are handled locally in the DEV10G port modules. For other port module types, the assembler handles the statistics.

4.35 Protection Switching

The following types of hardware-assisted protection switching are supported:

- Ethernet ring protection switching
- Port protection switching
-

Ring protection is also supported over a link aggregation group (LAG).

4.35.1 Ethernet Ring Protection Switching

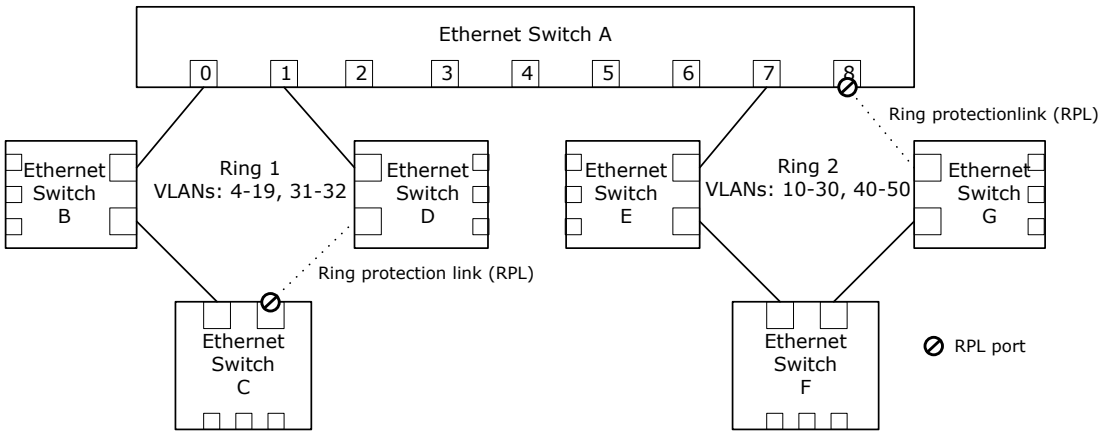
When protection switching occurs in an Ethernet ring (ITU-T G.8032), the following actions must be taken to restore connectivity.

- MAC addresses learned on the ring ports for the involved FIDs must be flushed from the MAC table. This is done using the scan functionality in the LRN block. Using ANA_L2:COMMON:SCAN_FID_CFG.SCAN_FID_VAL MAC, addresses for up to 16 FIDs can be flushed at a time.
- The RPL port must be changed to forwarding state and ports interfacing to the failed link must be changed to blocking state. For rings supporting many VLANs, this can be done efficiently using VLAN TUPE.

The following illustration shows an Ethernet ring protection example. Switch A is part of two rings, ring 1 and ring 2. Ring 1 uses 18 VIDs, VID 4-19 and 31-32, for forwarding traffic. Ring 2 uses 32 VIDs, VID 10-30 and 40-50.

To allow a VLAN TUPE command to identify the VIDs used by each of the two rings, a bit in the TUPE_CTRL field in the VLAN table is allocated for each of the two rings. In this example, ring 1 uses bit 0 in TUPE_CTRL, and ring 2 uses bit 1.

Figure 4-106. Ethernet Ring Protection Example



The following table shows the configuration of the VLAN table of switch A. The VLAN_PORT_MASK bit shows only the value for the ring ports. For the other ports (ports 2-6), the VLAN_PORT_MASK bits are assumed to be 0.

Table 4-283. VLAN Table Configuration Before APS

Address (VID)	TUPE_CTRL	VLAN_PORT_MASK	VLAN_FID
4–9	0x01	0x0003	4–9

.....continued			
Address (VID)	**TUPE_CTRL**	**VLAN_PORT_MASK**	**VLAN_FID**
10–19	0x03	0x0083	10–19
31–32	0x01	0x0003	31–32
20–30, 40–50	0x02	0x0080	20–30, 40–50

VID 4–9 and 31–32 are only used by ring 1, so only bit 0 is set in TUPE_CTRL. VLAN_PORT_MASK includes the two ring ports of ring 1, port 0, and port 1.

VID 10–19 are used by both ring 1 and ring 2, so bit 0 and bit 1 are both set in TUPE_CTRL. VLAN_PORT_MASK includes the ring ports of ring 1 as well as port 7, used by ring 2. Port 8 is not included, because it is in blocking state.

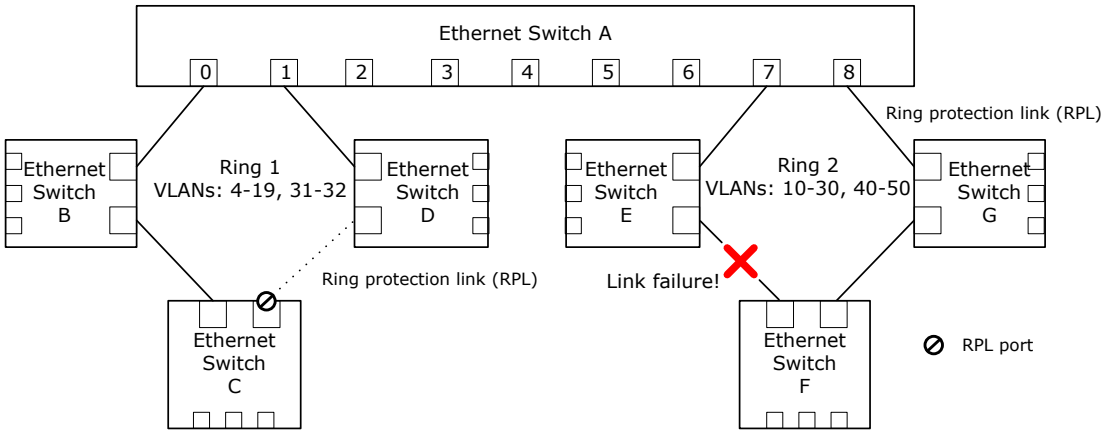
VID 20–30 and 40–50 are only used by ring 2, so only bit 1 is set in TUPE_CTRL. VLAN_PORT_MASK includes port 7.

In this example, independently learning is used for all VLANs; that is, a separate FID is used for each VID.

To also block reception of frames on any blocking ports, ANA_L3:COMMON:VLAN_FILTER_CTRL must be configured to include all ring ports. In this example ports 0,1,7 and 8 must be set in ANA_L3:COMMON:VLAN_FILTER_CTRL.

The following illustration shows the same set up as the previous illustration, but now a link in ring 2 has failed. As a result, the RPL of ring 2 must be activated, meaning port 8 of switch A must be changed to forwarding state.

Figure 4-107. Ethernet Ring with Failure



To avoid having to perform write access to the VLAN_PORT_MASKs of each of VID 10–30 and 40–50, VLAN TUPE can be used instead. The following table lists the VLAN TUPE parameters to reconfigure the VLAN table.

Table 4-284. VLAN TUPE Command for Ring Protection Switching

Field	Value	Comment
TUPE_CTRL_BIT_MASK	0x0002	Bits 0 and 1 of TUPE_CTRL are used as bit mask. The TUPE command is applied only to VIDs used by ring 2, so only bit 1 is set.
TUPE_CTRL_BIT_ENA	1	Enables use of TUPE_CTRL_BIT_MASK.
TUPE_START_ADDR	10	To make VLAN TUPE complete faster, only cover VID 10–50.
TUPE_END_ADDR	50	
TUPE_CMD_PORT_MASK_SET	0x0100	Enables forwarding on port 8.

.....continued		
Field	**Value**	**Comment**
TUPE_START	1	Starts VLAN TUPE. The device clears TUPE_START when TUPE is completed.

After the VLAN TUPE command is completed, the VLAN table is updated as shown in the following table.

Table 4-285. VLAN Table Configuration After APS

Addr (VID)	TUPE_CTRL	VLAN_PORT_MASK	VLAN_FID
4–9	0x01	0x0003	4–9
10–19	0x03	0x0183	10–19
31–32	0x01	0x0003	31–32
20–30, 40–50	0x02	0x0180	20–30, 40–50

The TUPE_CTRL field is 16 bits wide, so using the described approach allows up 16 Ethernet rings with VLAN TUPE assisted protection switching. However, any unused bits in the 65-bit wide VLAN_PORT_MASK can be used in the same manner as TUPE_CTRL bits. For example, a 24-port switch only using bits 0–23 in the VLAN_PORT_MASK have an additional 41 bits, which can be used as an extension of the TUPE_CTRL field.

For Ethernet rings using only a few VIDs, there is little to gain in using VLAN TUPE and new values could instead be written directly to the VLAN_PORT_MASK of the relevant VIDs.

In addition to running the VLAN TUPE commands listed in the previous table, any MAC addresses learned on the ring ports of ring 2 must be flushed from the MAC table. In the above example, ring 2 uses 32 VIDs for forwarding traffic. Each of these use a specific FID. In this example, FID is set equal to VID. MAC addresses for up to 16 FIDs can be flushed at a time, so to flush MAC addresses for 32 FIDs, two flush commands must be executed. The first flush command is shown in the following table. The second flush command is identical, except SCAN_FID_VAL is set to 26–30, 40–50.

Table 4-286. MAC Table Flush, First Command

Field	Value	Comment
ANA_L2::SCAN_FID_CTRL.SCAN_FID_ENA	1	Enables use of SCAN_FID_VAL.
ANA_L2::SCAN_FID_CFG.SCAN_FID_VAL	10–25	Flushes FIDs 10–25. Flushing can be done for up to 16 discrete FID values at a time. If flushing needs to be done for less than 16 FIDs. the unused FID values must be set to 0x1FFFF.
LRN::SCAN_NEXT_CFG.FID_FILTER_ENA	1	Enables FID specific flushing.
LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID	0x1FFFF	0x1FFFF => not used.
LRN::AUTOAGE_CFG_1.USE_PORT_FILTER_ENA	1	Enables flushing for specific ports.
ANA_L2::FILTER_LOCAL_CTRL	0x0180	Flushes port 7 and 8.
LRN::SCAN_NEXT_CFG.SCAN_NEXT_REMOVE_FOUND_ENA	1	Removes matches. In other words flushing.

.....continued		
Field	**Value**	**Comment**
LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT	1	Start flushing. The device clears MAC_TABLE_ACCESS_SHOT when flushing has completed. If desirable the device can be configured to generate an interrupt when flushing has completed. This can be used to trigger any subsequent flush commands.

Flushing MAC addresses from the MAC table usually takes longer than running VLAN TUPE. As a result, to have the protection switching complete as fast as possible, start MAC table flushing first, followed by VLAN TUPE. The two will then run in parallel, and when both have completed, the protection switching is done.

4.35.2 Link Aggregation

For ring protection, the working/protection entity can be forwarded on a LAG, providing an additional layer of hardware assisted protection switching.

When a link within the LAG fails, the 16 port masks in ANA_AC:AGGR must be updated to avoid forwarding on the failed link.

To avoid consuming multiple entries in VCAP ES0 for services being forwarded on a LAG, REW:COMMON:PORT_CTRL.ES0_LPORT_NUM must be setup such that a common port number is used in VCAP ES0 keys, regardless of the actual LAG member chosen.

4.35.3 Port Protection Switching

The device supports several methods through which the 1:1 port protection switching can be supported:

- Link Aggregation Groups. A LAG with two ports can be configured and ANA_AC:AGGR can be set to only use one of the links in the LAG.
- Disabling the standby port in ANA_L3:MSTP.
- Disabling the standby port using ANA_L3:COMMON:PORT_FWD_CTRL and ANA_L3:COMMON:PORT_LRN_CTRL. This is the recommended approach, because it interferes minimally with other features of the device.

Regardless of the method, a logical port number must be configured for the involved ports such that MAC addresses are learned on the logical port number to avoid the need for flushing MAC table entries when protection switching occurs. The logical port number is configured in ANA_CL:PORT:PORT_ID_CFG.LPORT_NUM.

REW:COMMON:PORT_CTRL.ES0_LPORT_NUM must be used to avoid consuming multiple VCAP ES0 entries. For more information, see 4.35.2 Link Aggregation.

5. VCore System and CPU Interfaces

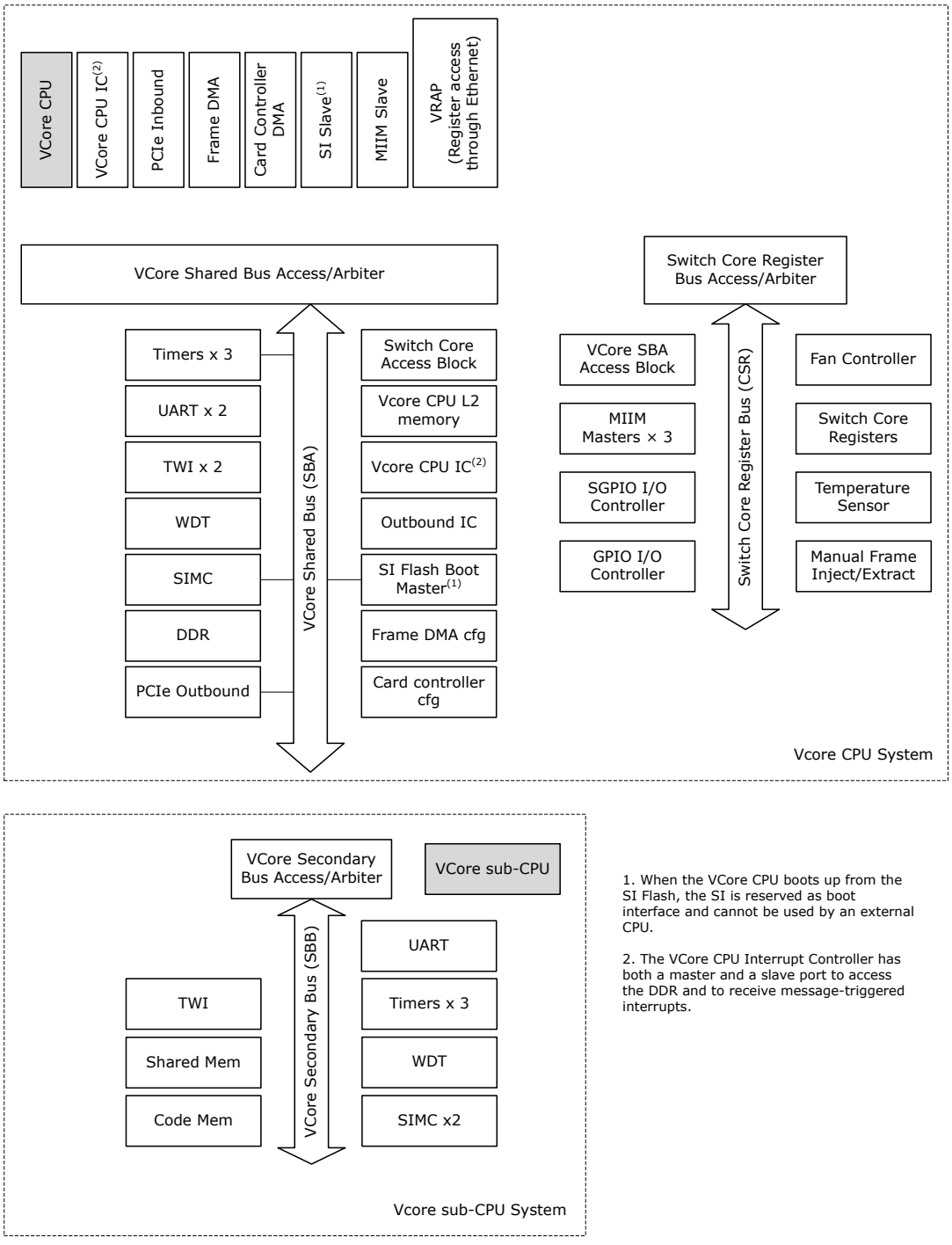
This section provides information about the functional aspects of blocks and interfaces related to the VCore on-chip microprocessor system and external CPU systems.

The device contains a powerful VCore CPU system that is based on an embedded ARM Cortex A53 microprocessor and a high bandwidth Ethernet Frame DMA engine. The VCore system can control the device independently or it can support an external CPU, relieving the external CPU of the otherwise time-consuming tasks of transferring frames, maintaining the switch core, and handling networking protocols. The VCore CPU system incorporates also a secondary sub-CPU subsystem, which is based on an embedded lightweight ARM Cortex M3 microprocessor with its own peripherals.

When the VCore CPU/sub-CPU is enabled, it either automatically boots up from serial Flash or an external CPU can manually load a code-image to the device and then start the VCore CPU/sub-CPU.

An external CPU can be connected to the device through the PCIe interface, the serial interface (SI), dedicated MIIM slave interface, or through Versatile Register Access Protocol (VRAP) formatted Ethernet frames via the NPI Ethernet port. Through these interfaces, the external CPU has access to the complete set of registers in the device and can control the device without help from the VCore CPU if needed.

Figure 5-1. VCore System Block Diagram



1. When the VCore CPU boots up from the SI Flash, the SI is reserved as boot interface and cannot be used by an external CPU.

2. The VCore CPU Interrupt Controller has both a master and a slave port to access the DDR and to receive message-triggered interrupts.

5.1 VCore Configurations

The behavior of the VCore system after reset is determined by four VCore strapping pins that are overlaid on GPIO pins. For more information, see 5.10.8.1 GPIO Overlaid Functions. The value of these GPIOs is sampled shortly after releasing reset to the device. By using resistors to pull the GPIOs either low or high, software can use these GPIO pins for other functions after reset has been released.

The PCIe controller is enabled at endpoint mode regardless of the strapping. If another configuration other than the default needs to be applied, then software executing on the VCore CPU or sub-CPU must perform the appropriate configurations prior to the PCIe enumeration. For more information regarding configuring the PCIe controller refer to 5.6 PCIe Dual Mode Controller.

Table 5-1. VCore Configurations

VCore_CFG [3:0]	Behavior
0000	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
0001	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
0010	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
0011	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
0100	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
0101	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
0110	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
0111	The VCore sub-CPU is enabled and boots from SI (the SI slave is disabled).
1000	The VCore CPU is enabled and boots from SI (the SI slave is disabled).
1001	The VCore CPU is enabled and boots from SI (the SI slave is disabled).
1010	The VCore CPU is enabled and boots from SI (the SI slave is disabled).
1011	The VCore CPU is enabled and boots from SI (the SI slave is disabled).
1100	The VCore CPU is enabled and boots from SI (the SI slave is disabled).
1101	Reserved
1110	MIIM slave is enabled (MIIM slave pins are overlaid on GPIOs.) Automatic boot is disabled, and SI slave is enabled.
1111	Automatic boot is disabled, and SI slave is enabled.

The strapping value determines the reset value for the CPU::GENERAL_CTRL and CPU::SUBCPU_SYS_CFG registers. After startup, the behavior of the VCore system can be modified by changing some of the fields in this register. Common scenarios are:

- After starting the device with automatic boot disabled, an external CPU can manually boot the VCore CPU from SI Flash by writing CPU::GENERAL_CTRL.IF_SI_OWNER = 1, CPU::GENERAL_CTRL.BOOT_MODE_ENA = 1, and CPU::GENERAL_CTRL.VCORE_CPU_DIS = 0. Setting GENERAL_CTRL.IF_SI_OWNER=1 will disable the SI slave, so the external CPU must be using another interface than SI.
- After starting the device with automatic boot disabled, an external CPU can manually boot the VCore sub-CPU from SI Flash by writing CPU::GENERAL_CTRL.IF_SI_OWNER = 1, CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE = 1, CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_DIS

= 0, CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE = 1. Setting GENERAL_CTRL.IF_SI_OWNER=1 will disable the SI slave, so the external CPU must be using another interface than SI.

- After starting the device with the VCore CPU disabled, initializing the DDR and loading software into DDR memory, an external CPU can boot the VCore CPU from DDR memory by writing CPU::GENERAL_CTRL.BOOT_MODE_ENA = 0 and CPU::GENERAL_CTRL.VCORE_CPU_DIS = 0.
- After starting the device with the VCore sub-CPU disabled, an external CPU can boot the VCore sub-CPU from the internal memory by writing CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE = 1, CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_DIS = 0, CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE_PROT_AMBA = 1, load code to internal memory (mapped to the local address space of VCore system), CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE = 0.
- After automatically booting from SI Flash, the VCore CPU/sub-CPU can release the SI interface and enable the SI slave by writing CPU::GENERAL_CTRL.IF_SI_OWNER = 0. This enables SI access from an external CPU to the device. A special PCB design is required to make SI work for both Flash and external CPU access.
- The MIIM slave can be manually enabled by writing CPU::GENERAL_CTRL.IF_MIIM_SLV_ENA = 1. The MIIM slave automatically takes control of the appropriate GPIO pins.

The strapping value is available at CPU::GENERAL_STAT.VCORE_CFG. Software may read the strapping value in order to differentiate the applications.

The device incorporates multiple JTAG controllers sharing a single JTAG interface. The ownership of the JTAG interface is controlled through strapping. For more information see 5.12 JTAG Interface.

5.2 Clocking and Reset

The following table lists the registers associated with reset and the watchdog timer.

Table 5-2. Clocking and Reset Configuration Registers

Register	Description
CPU::RESET	Soft resets for various components of VCore system
CPU::RESET_PROT_STAT	VCore reset protection and status scheme
DEVCPU_GCB::SOFT_RST	Initiating chip-level soft reset.

The VCore CPU runs 1 GHz, the DDR controller runs 625 MHz, and the rest of the VCore system runs at 500 MHz and 250 MHz.

The VCore can be soft-reset by setting RESET.VCORE_RST. By default, this resets both the VCore CPU and the VCore system. Refer to CPU::RESET and CPU::RESET_PROT_STAT for a fine-grain soft reset scheme for the various VCore system components.

The frame DMA must be disabled prior to a soft reset of the VCore system. When a soft reset of the VCore CPU is issued, the PCIe and DDR controllers are not affected and will continue to operate throughout the soft reset of the VCore CPU.

The VCore system comprises all the blocks attached to the VCore Shared Bus (SBA), including the PCIe, DDR, frame DMA, SI slave, and MIIM slave blocks. The device includes all the blocks attached to the Switch Core Register Bus (CSR) including the VRAP slave. For more information about the VCore System blocks, see Figure 5-1.

The device can be soft-reset by writing SOFT_RST.SOFT_CHIP_RST. The VCore system and CPU can be protected from a device soft-reset by writing RESET_PROT_STAT.SYS_RST_PROT_VCORE = 1 before initiating soft-reset. In this case, a chip-level soft reset is applied to all other blocks except the VCore system and the VCore CPU. When protecting the VCore system and CPU from a soft reset, the frame DMA must be disabled prior to a chip-level soft reset. The SERDES and PLL blocks can be protected from reset by writing to SOFT_RST.SOFT_SWC_RST instead of SOFT_CHIP_RST.

The VCore general purpose registers (CPU::GPR) and GPIO alternate modes (DEVCPU_GCB::GPIO_ALT) are not affected by a soft-reset. These registers are only reset when an external reset is asserted.

5.2.1 Watchdog Timer

The VCore system has a built-in watchdog timer (WDT) with a configurable timeout cycle. The watchdog timer is configured, enabled, disabled, or reset through the registers in the following table. The watchdog timer is disabled by default.

Table 5-3. WDT Registers

Register	Description
WDT::WDT_CR	Control
WDT::WDT_TORR	Timeout range
WDT::WDT_CCVR	Current counter value
WDT::WDT_CRR	Counter restart
WDT::WDT_STAT	Interrupt status
WDT::WDT_EOI	Interrupt clear
WDT::WDT_COMP_PARAM_5	Read-only component parameters
WDT::WDT_COMP_PARAM_4	Read-only component parameters
WDT::WDT_COMP_PARAM_3	Read-only component parameters
WDT::WDT_COMP_PARAM_2	Read-only component parameters
WDT::WDT_COMP_PARAM_1	Read-only component parameters
WDT::WDT_COMP_VERSION	Component version
WDT::WDT_COMP_TYPE	Component type
CPU::RESET	Soft resets for various components of VCore system
CPU::RESET_PROT_STAT	VCore reset protection and status scheme

The watchdog timer can be configured to generate an interrupt on the first timeout and a reset on the second, or trigger a reset on the first timeout. After the watchdog timer is enabled, it must be regularly restarted by software by writing WDT::WDT_CRR = 0x76. Restarting the watchdog timer automatically clears the interrupt. The reset pulse width is configurable. The watchdog timer resets the entire VCore system, including itself. The watchdog timer can be protected from its own reset or from a RESET.VCORE_RST by setting RESET_PROT_STAT.VCORE_WDT_RST_PROT_WDT and RESET_PROT_STAT.VCORE_RST_PROT_WDT respectively. The components of the VCore system may be protected from a watchdog timer reset through RESET_PROT_STAT.VCORE_RST_PROT_AMBA and RESET_PROT_STAT.VCORE_RST_PROT_PDBG. The VCore CPU cannot be protected and is always affected by a watchdog timer timeout.

5.3 Shared Bus

The VCore shared bus is AMBA-based incorporating an AXI fabric (36-bit address and 128-bit data bus at 500 MHz), an AHB bus (32-bit address and 32-bit data bus at 250 MHz) and an APB bus (32-bit address and 32-bit data bus at 250 MHz). It has dedicated master and slave interfaces that interconnect all the blocks in the VCore system. All the masters are connected to the VCore Shared Bus Access/Arbiter block; only they can start accesses on the bus.

The shared bus uses byte addresses, and transfers of 8, 16, 32, 64, or 128 bits can be made. To increase performance, bursting of multiple words on the shared bus can be performed. Additionally, multiple outstanding transactions may be invoked to hide latency and maximize performance further. Such an example would be the Frame DMA using PCIe Outbound to access the memory of an external CPU.

All slaves are mapped into the VCore address space and can be accessed directly by masters on the shared bus. Two possible mappings of VCore shared bus slaves are boot mode and normal mode.

- Boot mode: Boot mode is active after power-up and reset of the VCore system. In this mode, the SI Flash Boot Master is mirrored into the lowest address region.
- Normal mode: In normal mode, the DDR controller is mirrored into the lowest address region.

Changing between boot mode and normal mode is done by first writing and then reading CPU::GENERAL_CTRL.BOOT_MODE_ENA. A change takes effect during the read.

The device supports up to 8 gigabyte (GB) of DDR memory.

Table 5-4. Shared Bus Memory Map

	Boot Mode			Normal Mode		
0x000000000	4 GB	Mirror SPI boot Flash	0x000000000	8 GB	Mirror DDR3/4	
0x100000000						
0x200000000	4 GB	Mirror NAND Flash	0x200000000			
0x300000000	4 GB	reserved	0x300000000	4 GB	reserved	
0x400000000	4 GB	SPI boot flash	0x400000000	4 GB	SPI boot flash	
0x500000000	4 GB	NAND flash	0x500000000	4 GB	NAND flash	
0x600000000	4 GB	reserved	0x600000000	4 GB	reserved	
0x610000000	256 MB	Chip Registers - VCore	0x610000000	256 MB	Chip Registers – VCore	
0x620000000	256 MB	Chip registers - CSR	0x620000000	256 MB	Chip registers - CSR	
0x630000000	256 MB	PCIe Outbound	0x630000000	256 MB	PCIe Outbound	
0x640000000	256 MB	Sub-CPU system	0x640000000	256 MB	Sub-CPU system	
0x700000000		reserved	0x700000000		reserved	
0x900000000	8 GB	DDR	0x900000000	8 GB	DDR	
0xFFFFFFFFF	28 GB	PCIe Outbound	0xFFFFFFFFF	28 GB	PCIe Outbound	

Note:
When the VCore system is protected from a soft reset using CPU::RESET_PROT_STAT.VCORE_RST_PROT_AMBA, a soft reset will not change shared bus memory mapping. For more information about protecting the VCore system when using a soft reset, see 5.5.1 Register Access and Multimaster Systems.

If the boot process copies the SI Flash image to DDR, and if the contents of the SI memory and the DDR memory are the same, software can execute from the mirrored region when swapping from boot mode to normal mode.

Otherwise, software must be executing from the fixed SI Flash region when changing from boot mode to normal mode.

5.3.1 VCore Shared Bus Access/Arbitration

The VCore shared bus supports multiple outstanding transactions, static priority between masters and limited visibility of slaves by the masters. The following table shows the regions visibility from various masters.

Table 5-5. Master/Slave Visibility On Vcore Shared Bus

	Chip Registers (VCore & CSR)	DDR	DDR through VCore CPU L2 Memory	PCIe Outbound	VCore sub-CPU System
VCore CPU IC	No	Yes	No	No	No
VCore CPU	Yes	Yes	No	Yes	Yes
PCIe Inbound	Yes	No	Yes	No	Yes
MIIM slave/SI slave/VRAP	Yes (CSR)	No	No	No	No
VCore SBA Access Block	Yes	No	Yes	Yes	Yes
Frame DMA	No	No	Yes	Yes	No
Card Controller DMA	No	No	Yes	No	Yes

The DDR region is available only to VCore CPU and VCore CPU IC. Dedicated hardware resources ensure that access to DDR is arbitrated only between these two masters with full bandwidth, without being affected by any other ongoing transactions. Other masters that need to access DDR have to go through the L2 memory of the VCore CPU.

The VCore shared bus arbitrates between masters that want to access the bus. A strict prioritized arbitration scheme is used to regulate the accesses. The following table shows the priority order of the masters on the VCore shared bus.

Table 5-6. SBA Master Interfaces Ordered High To Low Priority

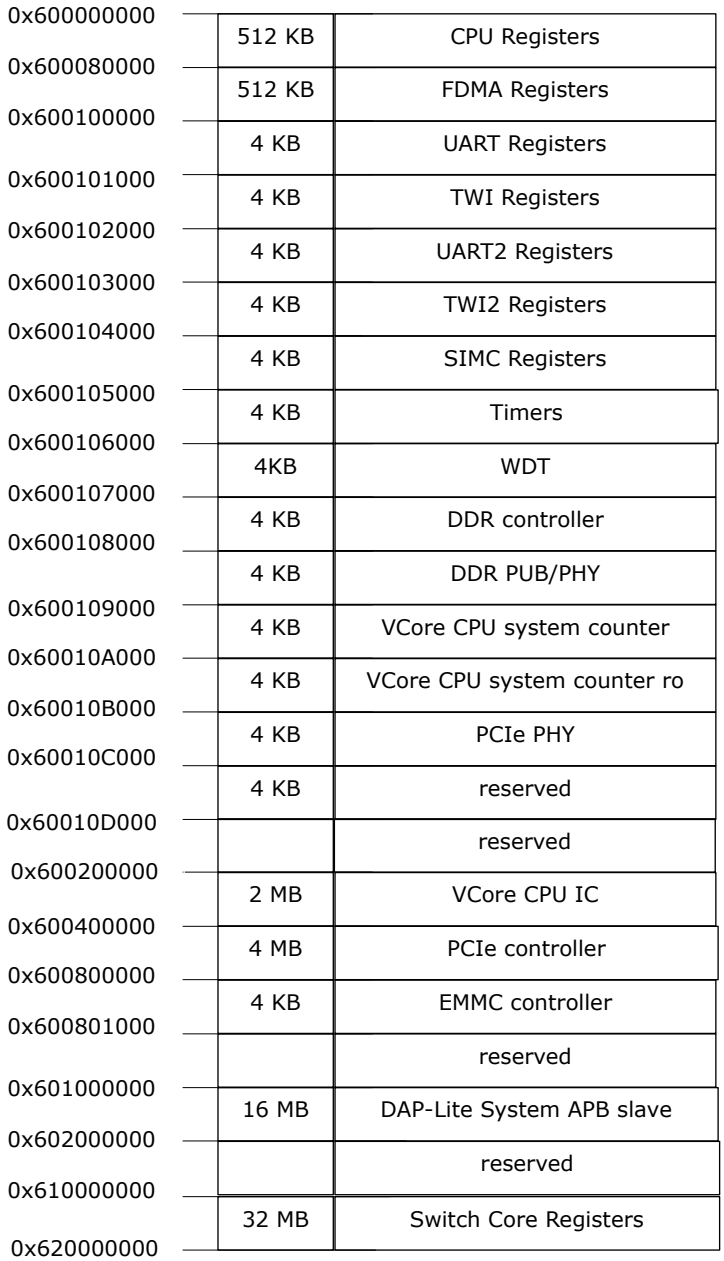
Master interface
VCore CPU IC
VCore CPU
PCIe inbound
MIIM slave
SI slave
VRAP
VCore SBA access block
Frame DMA
Card controller DMA

The VCore shared bus is also accessible from the VCore sub-CPU system. For more information, see 5.11 VCore Sub-CPU System.

5.3.2 Chip Register Region

Registers in the VCore domain and inside the Switch Core are memory mapped into the Chip registers region of the shared bus memory map. All registers are 32-bit wide and must only be accessed using 32-bit reads and writes. Bursts are supported.

Figure 5-2. Chip Registers Detailed Memory Map



The registers in the 0x600000000 through 0x60FFFFFFF region are physically located inside the VCore System, so read and write access to these registers is fast (done in a few clock cycles). All registers in this region are considered fast registers.

Registers in the 0x610000000 through 0x61FFFFFFF region are located inside the switch core; access to registers in this range takes approximately 500 ns (system clock at 500 MHz). The DEVCPU_ORG and DEVCPU_QS targets are special; registers inside these two targets are faster; access to these two targets takes approximately 100 ns.

When more than one CPU is accessing registers, the access time may be increased. For more information, see 5.5.1 Register Access and Multimaster Systems.

5.3.3 SI Flash Region

Read access from the SI Flash region initiates Flash formatted read access on the SI pins of the device by means of the SI boot controller. For more information about the SI boot controller and protocol, see 5.10.1 SI Boot Controller.

The SI Flash region cannot be written, writing to the SI interface must be implemented by using the SI Master Controller. For more information, see 5.10.2 SI Master Controller. For legacy reasons it is also possible to write to the SI interface by using the SI Boot Master's software "bit-banging" register interface. For more information, see 5.10.1 SI Boot Controller.

5.3.4 DDR Region

Read and write access from or to the DDR region maps to access on the DDR interface of the device by means of the DDR controller. For more information about the DDR controller and initialization of DDR memory, see 5.8 DDR (DDR3/DDR4) Memory Controller.

5.3.5 PCIe Region

Read and write access from or to the PCIe Outbound region maps to outbound read and write access on the PCIe interface of the device by means of the PCIe controller. For more information about the PCIe controller and how to reach addresses in PCIe environments, see 5.6 PCIe Dual Mode Controller.

5.4 VCore CPU

The VCore CPU system is based on a powerful ARM Cortex-A53 processor. Below are the high-level features:

- Dual-core.
- High speed CPU system running at 1GHz clock frequency.
- 32KB of L1 Instruction cache.
- 32KB of L1 Data cache.
- 256KB of L2 cache, shared by both cores.
- High speed ACE interface to interact with external DDR memory (up to 8GB)
- High speed Accelerator Coherency Port (ACP).
- Neon and Floating-Point Unit (FPU) support.
- Cryptographic extensions in the Neon and FPU of each CPU.
- GIC-500 as centralized interrupt controlling resource for supporting and managing interrupts.
- Conventional JTAG based external debug.

When the automatic boot is enabled using the VCore strapping pins (see 5.1 VCore Configurations), the VCore CPU automatically starts to execute code from the SI Flash at byte address 0.

The following is a typical automatic boot sequence.

1. Speed-up boot interface by increasing the SI clock frequency (5.10.1 SI Boot Controller)
2. Initialize DDR. For more information, see 5.8 DDR (DDR3/DDR4) Memory Controller).
3. Copy boot-loader from Flash to DDR memory.
4. Change memory map from boot mode to normal mode. For more information, see 5.3 Shared Bus.
5. Enable the QUAD mode on the SI Flash. To do so, use the manual control mode of the 5.10.1 SI Boot Controller.
6. Enable the QUAD mode on the 5.10.1 SI Boot Controller.
7. Copy the rest of the code-image from Flash to DDR (use the SI Flash mapping in Normal mode, 5.3 Shared Bus).

When automatic booting of the VCore CPU is disabled, an external CPU or the VCore sub-CPU is still able to start the VCore CPU through registers. For more information, see 5.1 VCore Configurations.

5.4.1 Cache Coherency

The processor is equipped with two L1 caches per core and a shared L2 cache common to both the cores. L1 and L2 caches cannot be disabled individually. Both these caches can be invalidated on processor reset based on configuration control.

ACP interface of the processor is used by FDMA or PCI-E end-points for frame injection/extraction. ACP interface serves to reduce software cache maintenance operations while making coherent requests to shared memory across multiple masters enabling them to allocate data into L2 cache coherently.

The L2 memory system integrates the snoop control unit (SCU) which maintains coherency between the caches of both the cores. SCU uses ACE bus interface to access the main memory system (DDR3/4).

5.4.2 VCore CPU Interrupt Controller

The VCore CPU IC is the ARM CoreLink GIC-500 Generic Interrupt Controller. The IC supports the following interrupt types.

- Software generated interrupts (SGI)
- Private peripheral interrupt (PPI)
- Shared peripheral interrupt (SPI)
- Message based interrupt (LPI)

Software Generated Interrupts are triggered by software running on either core.

PPIs are dedicated pins to GIC, which are connected to the sources from the VCore CPU. The following table lists the PPI interrupt sources.

Table 5-7. List Of PPI Interrupt Sources

PPI Source	Index
Non-secure physical timer	14
Secure physical timer	13
Virtual timer	11
Hypervisor timer	10
Virtual CPU maintenance timer	9
PMU overflow interrupt	7

SPIs are dedicated pins to GIC, which are connected to the various interrupt sources across the chip. The following table lists the SPI interrupts.

Table 5-8. List Of Spi Interrupts

Source Name	Index	Description
EXT_SRC0	0	External interrupt source 0. See 5.10.13.5 GPIO Mapped Interrupts.
EXT_SRC1	1	External interrupt source 1. See 5.10.13.5 GPIO Mapped Interrupts.
Reserved	2	Reserved.
MSHC_WAKE	3	eMMC/SD card controller wake interrupt. See 5.10.14 Mobile Storage Host Controller.
MSHC	4	eMMC/SD card controller interrupt. See 5.10.14 Mobile Storage Host Controller.
WDT	5	Watchdog timer interrupt. See 5.2.1 Watchdog Timer.
R0	6	Timer 0 interrupt. See 5.10.4 Timers.
TIMER1	7	Timer 1 interrupt. See 5.10.4 Timers.
TIMER2	8	Timer 2 interrupt. See 5.10.4 Timers.

.....continued

Source Name	Index	Description
UART	9	UART interrupt. See 5.10.5.1 UART Interrupt.
UART2	10	UART2 interrupt. See 5.10.5.1 UART Interrupt.
TWI	11	TWI interrupt. See 5.10.6.3 Two-Wire Serial Interface Interrupt.
TWI2	12	TWI2 interrupt. See 5.10.6.3 Two-Wire Serial Interface Interrupt.
SIMC	13	Serial master Controller interrupt. See 5.10.2.3 SIMC Interrupts.
SUBCPU_LOCK	14	VCore sub-CPU lock interrupt.
SW0	15	Software interrupt 0. See 5.5.5 Mailbox and Semaphores.
SW1	16	Software interrupt 1. See 5.5.5 Mailbox and Semaphores.
SGPIO0	17	Serial GPIO interrupt 0. See 5.10.9.2 SIO Interrupt.
SGPIO1	18	Serial GPIO interrupt 1. See 5.10.9.2 SIO Interrupt.
SGPIO2	19	Serial GPIO interrupt 2. See 5.10.9.2 SIO Interrupt.
GPIO	20	Parallel GPIO interrupt. See 5.10.8.2 GPIO Interrupt.
MIIM0	21	MIIM controller 0 interrupt. See 5.10.7.4 MII Management Interrupt.
MIIM1	22	MIIM controller 1 interrupt. See 5.10.7.4 MII Management Interrupt.
MIIM2	23	MIIM controller 2 interrupt. See 5.10.7.4 MII Management Interrupt.
MIIM3	24	MIIM controller 3 interrupt. See 5.10.7.4 MII Management Interrupt.
FDMA	25	Frame DMA interrupt. See 5.7.4 FDMA Events and Interrupts.
ANA	26	Analyzer interrupt.
PTP_RDY	27	Timestamp ready interrupt.
PTP_SYNC	28	PTP synchronization interrupt.
ITGR	29	Memory integrity interrupt. See 5.10.12 Memory Integrity Monitor.
XTR_RDY	30	Extraction data ready interrupt.
INJ_RDY	31	Injection ready interrupt.
PCIE	32	PCIe interrupt. See 5.6.7 Power Management.
EACL	34	EACL interrupt.
REW	35	Rewriter interrupt.
DDR	36	DDR controller interrupt.
CPU[9:0]	46:37	Software interrupts CPU::SHARED_INTR.
KR_SD10G	66:47	KR10G serDes modules interrupts.
KR_SD6G	79:67	KR6G serDes modules interrupts.
DEV	144:80	Port modules interrupts.

.....continued

Source Name	Index	Description
PROC_ERR	145	ECC fatal error from L2.
PROC_EXT_ERR	146	External error triggered from AXI bus towards processor.
PCIE_LEGACY_INTRD	147	Legacy interrupt D from PCIe.
PCIE_LEGACY_INTRC	148	Legacy interrupt C from PCIe.
PCIE_LEGACY_INTRB	149	Legacy interrupt B from PCIe.
PCIE_LEGACY_INTRA	150	Legacy interrupt A from PCIe.
PCIE_RC_SYS_ERR_INT	151	PCIe RC system error.
PCIE_RC_PME_INT	152	PCIe RC PME interrupt.
PCIE_RC_AER_ERR_INT	153	PCIe RC AER error interrupt.
PCIE_RC_SII_INT	154	PCIe RC SII interrupt.
PCIE_RC_WAKE_INT	155	PCIe RC wake interrupt.

LPIs are special message based interrupts, which are triggered by writes to a memory-mapped register (GITS_TRANSLATER) in the generic interrupt controller (GIC-500). PCIe end-points target a write to this register to trigger LPIs to the processor.

5.4.3 Processor Debug

The device implements conventional JTAG based invasive debug (with core halted using breakpoints and watchpoints, and a debug connection to examine and modify registers and memory). In addition, each core of the processor can initiate debug (self-hosted debug) which is muxed through the APB port along with JTAG access. For self-hosted debug, the debug target runs additional monitor software that runs on the processor itself rather than expensive interface hardware to connect a second host computer. For more information on accessing the debug target from the pins of the device, see 5.12 JTAG Interface.

5.5 External CPU Support

An external CPU attaches to the device through PCIe, SI, MIIM, or VRAP. Through these interfaces, an external CPU can access (and control) the device. For more information about interfaces and connections to device registers, see Figure 5-5.

Inbound PCIe access is performed on the VCore Shared Bus in the same way as the ordinary VCore CPU access. Through of the Switch Core Access block, it is possible to access the switch core register (CSR) bus. For more information about supported PCIe BAR-regions, see 5.6.3.1 Base Address Registers.

The SI, MIIM, and VRAP interfaces attach to the VCore Shared Bus, but they can only access the switch core register bus (CSR) through the switch core access block. Through the VCore SBA access block, it is possible to access the entire address space of the VCore shared bus. For more information, see 5.5.4 Access to the VCore Shared Bus.

The external CPU can coexist with the internal VCore CPU and hardware-semaphores and interrupts are implemented for inter-CPU communication. For more information, see 5.5.5 Mailbox and Semaphores.

5.5.1 Register Access and Multimaster Systems

There are three different groups of registers in the device.

- Switch Core
- Fast Switch Core
- VCore

The Switch Core registers and Fast Switch Core registers are separated into individual register-targets and attached to the Switch Core Register bus (CSR). The Fast Switch Core registers are placed in the DEVCPU_QS and DEVCPU_ORG register-targets. Access to Fast Switch Core registers is less than 125 ns (system clock at 500 MHz); other Switch Core registers take no more than 500 ns to access.

The VCore registers are attached directly to the VCore Shared Bus. The access time to VCore registers is negligible (a few clock cycles).

Although multiple masters can access VCore registers and Switch Core registers in parallel without noticeable penalty to the access time, the following exception applies.

- When accessing the Switch Core Register Bus (CSR) all masters go through the Switch Core Access block. The Switch Core Access block can serve only one transaction at a time. VCore shared bus arbitration applies in multi-masters attempting to access CSR through the Switch Core Access block. Ownership of the CSR should be resolved by use of software (for example, by using the build-in semaphores).

5.5.2 Serial Interface in Slave Mode

This section provides information about the function of the serial interface in slave mode.

The following table lists the registers associated with SI slave mode.

Table 5-9. SI Slave Mode Register

Register	Description
DEVCPU_GCB::IF_CTRL	Configuration of endianness and bit order.
DEVCPU_GCB::IF_CFGSTAT	Configuration of padding.
CPU::GENERAL_CTRL	SI interface ownership.
DEVCPU_GCB::SI_SLV_SS	Select which GPIO-mapped SPI_nCS[n] to use when owning the SPI2 interface

The serial interface implements a SPI-compatible protocol that allows an external CPU to perform read and write access to register targets within the device. Endianness and bit order is configurable, and several options for high frequencies are supported.

The serial interface is available to an external CPU when the VCore CPU does not use the SI for Flash or external SI access.

The following table lists the serial interface pins when the SI slave is configured as owner of SI interface through GENERAL_CTRL.IF_SI_OWNER.

Table 5-10. SI Slave Mode Pins

Pin Name	I/O	Description
SI_nCS0	I	Active low chip select
SI_Clk	I	Clock input
SI_DI	I	Data input (MOSI)
SI_DO	O	Data output (MISO)

Alternatively, the SI slave may be configured to own the SPI2 interface (which is mapped on GPIOs) through GENERAL_CTRL.IF_SI2_OWNER. In this case, DEVCPU_GCB::SI_SLV_SS selects which SPI_nCS[n]/GPIO to use for the SI slave. For more information, see 5.10.8.1 GPIO Overlaid Functions.

SI_DI is sampled on rising edge of SI_Clk. SI_DO is driven on falling edge of SI_Clk. There are no requirements on the logical values of the SI_Clk and SI_DI inputs when SI_nEn is asserted or deasserted, they can be either 0 or 1. SI_DO is only driven during read-access when read-data is shifted out of the device.

The external CPU initiates access by asserting chip select and then transmitting one bit read/write indication, 23 address bits, and 32 bits of write-data (or don't care bits when reading).

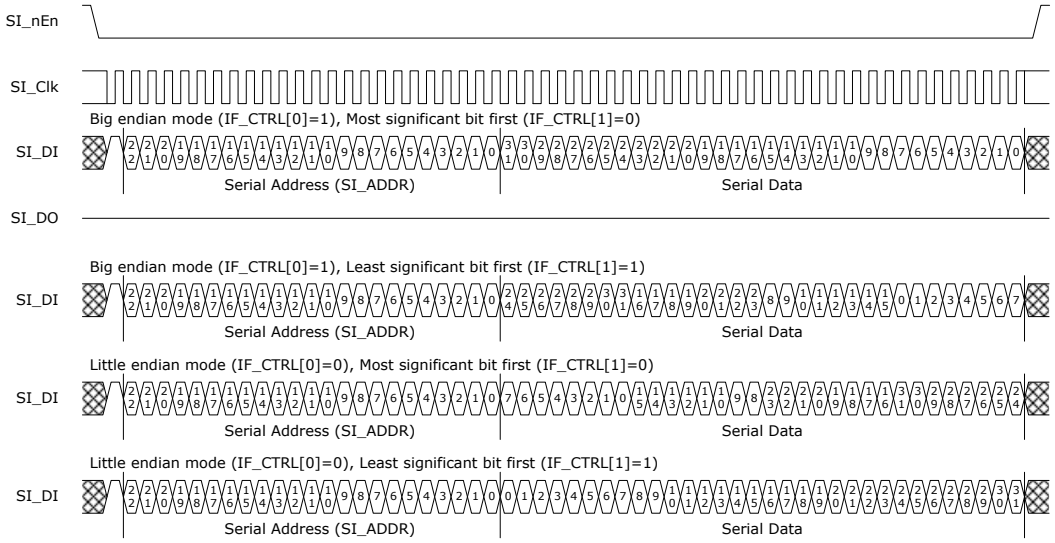
With the register address of a specific register (REG_ADDR), the SI address (SI_ADDR) is calculated as:

SI_ADDR = (REG_ADDR & 0x01FFFFFF)>>2

Data word endianness is configured through IF_CTRL[0]. The order of the data bits is configured using IF_CTRL[1].

The following figure shows various configurations for write access. The order of the data bits during writing, as depicted, is also used when the device is transmitting data during read operations.

Figure 5-3. Write Sequence for SI



When reading registers using the SI interface, the device needs to prepare read data after receiving the last address bit. The access time of the register that is read must be satisfied before shifting out the first bit of read data. For information about access time, see 5.5.1 Register Access and Multimaster Systems. The external CPU must apply one of the following solutions to satisfy read access time.

- Use SI_Clk with a period of minimum twice the access time for the register target. For example, for normal switch core targets (single Master): 1/(2 × 400 ns) = 1.25 MHz (maximum).
- Pause the SI_Clk between shifting of serial address bit 0 and the first data bit with enough time to satisfy the access time for the register target.
- Configure the device to send out padding bytes before transmitting the read data to satisfy the access time for the register target. For example, 1 dummy byte allows enough read-time for the SI clock to run up to 16 MHz in a single master system (see below for calculation).

The device is configured for inserting padding bytes by writing to IF_CFGSTAT.IF_CFG, these bytes are transmitted before the read data. The maximum frequency of the SI clock is calculated as:

(IF_CFGSTAT.IF_CFG × 8 – 1.5)/access-time. For example, for normal switch core targets (single Master), 1 byte padding give (1 × 8 – 1.5) / 400 ns = 16 MHz (maximum).

The maximum applicable SI clock frequency depends on the implementation of the external SI master, as well as the timing characteristics of the SI pins. The SI_DO output is kept tristated until the actual read data is transmitted.

The following figures show the options for serial read access. The following figures show only one mapping of read data, little endian with most significant bit first. Any of the mappings can be configured and applied to the read data in the same way as for the write data.

Figure 5-4. Read Sequence for SI_Clk Slow

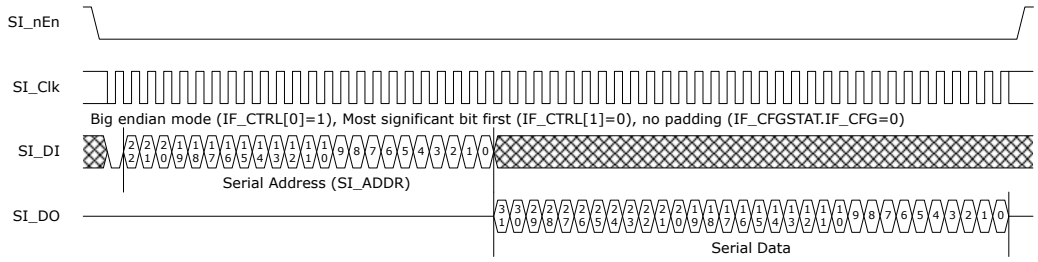


Figure 5-5. Read Sequence for SI_Clk Pause

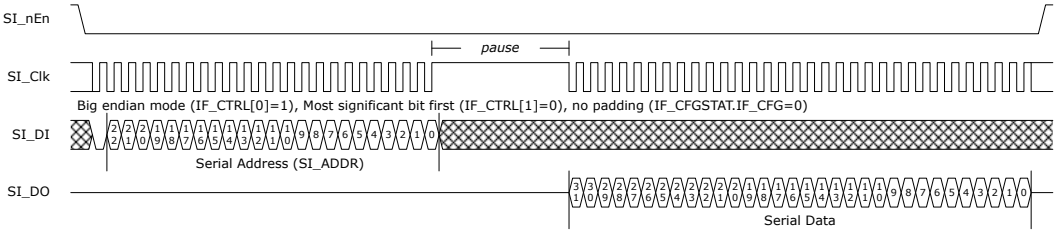
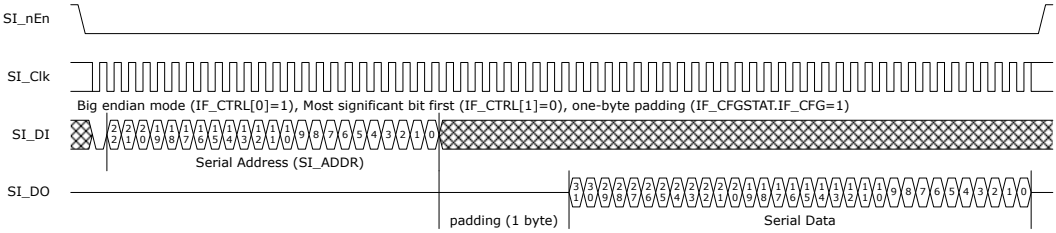


Figure 5-6. Read Sequence for One-Byte Padding



When dummy bytes are enabled (IF_CFGSTAT.IF_CFG), the SI slave logic enables an error check that sends out 0x88888888 and sets IF_CFGSTAT.IF_STAT if the SI master does not provide enough time for register read.

When using SI, the external CPU must configure the IF_CTRL register after power-up, reset, or chip-level soft reset. The IF_CTRL register is constructed so that it can be written what ever may be the state of the interface. For more information about constructing write-data for this register, see the instructions in IF_CTRL.IF_CTRL.

5.5.3 MIIM Interface in Slave Mode

This section provides the functional aspects of the MIIM slave interface.

The MIIM slave interface allows an external CPU to perform read and write access to the Switch Core register targets in device. Register access is done indirectly, because the address and data fields of the MIIM protocol are smaller than those used by the register targets. Transfers on the MIIM interface are using the Management Frame Format protocol specified in IEEE 802.3, Clause 22.

The MIIM slave pins on the device are overlaid functions on the GPIO interface. MIIM slave mode is enabled by configuring the appropriate VCore_CFG strapping pins. For more information, see 5.1 VCore Configurations. When MIIM slave mode is enabled, the appropriate GPIO pins are automatically overtaken. For more information, see 5.10.8.1 GPIO Overlaid Functions.

The following table lists the pins of the MIIM slave interface.

Table 5-11. MIIM Slave Pins

Pin Name	I/O	Description
MIIM_SLV_MDC/GPIO	I	MIIM slave clock input
MIIM_SLV_MDIO/GPIO	I/O	MIIM slave data input/output
MIIM_SLV_ADDR/GPIO	I	MIIM slave address select

MIIM_SLV_MDIO is sampled or changed on the rising edge of MIIM_SLV_MDC by the MIIM slave interface.

The MIIM slave can be configured to answer on one of the two different PHY addresses using the MIIM_SLV_ADDR pin. Setting the MIIM_SLV_ADDR pin to 0 configures the MIIM slave to use PHY address 0, and setting it to 1 configures the MIIM slave to use PHY address 31.

The MIIM slave has seven 16-bit MIIM registers defined as listed in the following table.

Table 5-12. MIIM Slave Pins

Register Address	Register Name	Description
0	ADDR_REG0	Bits 15:0 of the address to read or write. The address field must be formatted as word address.
1	ADDR_REG1	Bits 31:16 of the address to read or write.
2	DATA_REG0	Bits 15:0 of the data to read or write. Returns 0x8888 if a register read error occurred.
3	DATA_REG1	Bits 31:16 of the data to read or write. The read or write operation is initiated after this register is read or written. Returns 0x8888 if read while busy or a register read error occurred.
4	DATA_REG1_INCR	Bits 31:16 of data to read or write. The read or write operation is initiated after this register is read or written. When the operation is complete, the address register is incremented by one. Returns 0x8888 if read while busy or a register read error occurred.
5	DATA_REG1_INERT	Bits 31:16 of data to read or write. Reading or writing to this register does not cause a register access to be initiated. Returns 0x8888 if a register read error occurred.
6	STAT_REG	The status register gives the status of any ongoing operations. Bit 0: Busy. Set while a register read/write operation is in progress. Bit 1: Busy_rd. Busy status during the last read or write operation. Bit 2: Err. Set if a register access error occurred. Others: Reserved.

A 32-bit switch core register read or write transaction over the MIIM interface is done indirectly because of the limited data width of the MIIM frame. First, the address of the register inside the device must be set in the two 16-bit address registers of the MIIM slave using two MIIM write transactions. Afterwards the two 16-bit data registers can be read or written to access the data value of the register inside the device. Thus, it requires up to four MIIM transactions to perform a single read or write operation on a register target.

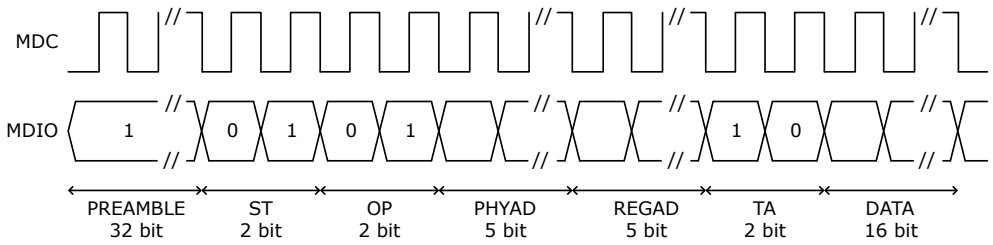
The address of the register to read/write is set in registers ADDR_REG0 and ADDR_REG1. The data to write to the register pointed to by the address in ADDR_REG0 and ADDR_REG1 is first written to DATA_REG0 and then to DATA_REG1. When the write transaction to DATA_REG1 is completed, the MIIM slave initiates the switch core register write.

With the register address of a specific register (REG_ADDR), the MIIM address (MIIM_ADDR) is calculated as:

MIIM_ADDR = (REG_ADDR & 0x01FFFFFF)>>2

The following figure shows a single MIIM write transaction on the MIIM interface.

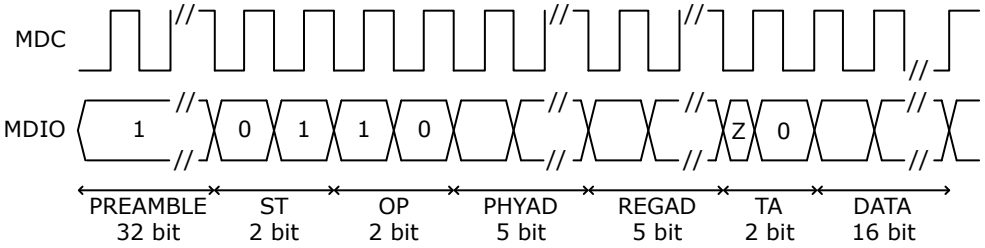
Figure 5-7. MIIM Slave Write Sequence



A read transaction is done in a similar way. First, read the DATA_REG0 and then read the DATA_REG1. As with a write operation, the switch core register read is not initiated before the DATA_REG1 register is read. In other words, the returned read value is from the previous read transaction.

The following figure shows a single MIIM read transaction on the MIIM interface.

Figure 5-8. MIIM Slave Read Sequence



5.5.4 Access to the VCore Shared Bus

This section provides information about how to access the VCore shared bus from an external CPU attached by means of the VRAP, SI, or MIIM. The following table lists the registers associated with the VCore shared bus access.

Table 5-13. VCore Shared Bus Access Registers

Register	Description
DEVCPU_GCB::VA_CTRL	Status for ongoing accesses
DEVCPU_GCB::VA_ADDR_LSB	Configuration of shared bus address
DEVCPU_GCB::VA_ADDR_MSB	Configuration of shared bus address
DEVCPU_GCB::VA_DATA	Data register
DEVCPU_GCB::VA_DATA_INCR	Data register, access increments VA_ADDR
DEVCPU_GCB::VA_DATA_INERT	Data register, access does not start new access

An external CPU performs 32-bit reads and writes to the VCore Shared Bus (SBA) through the VCore Access (VA) registers. In the VCore system, a dedicated master on the shared bus handles VA access. For information about arbitration between masters on the shared bus, see 5.3.1 VCore Shared Bus Access/Arbitration.

The SBA address is configured in VA_ADDR_LSB and VA_ADDR_MSB. Writing to VA_DATA starts an SBA write with the 32-bit value that was written to VA_DATA. Reading from VA_DATA returns the current value of the register and starts an SBA read access. When the read-access completes, the result is automatically stored in the VA_DATA register.

The VA_DATA_INCR register behaves similar to VA_DATA, except that after starting an access, the VA_ADDR register is incremented by four (so that it points to the next word address in the SBA domain). Reading from the VA_DATA_INCR register returns the value of VA_DATA, writing to VA_DATA_INCR overwrites the value of VA_DATA.

Note:
By using VA_DATA_INCR, sequential word addresses can be accessed without having to manually increment the VA_ADDR_LSB and VA_ADDR_MSB registers between each access.

The VA_DATA_INERT register provides direct access to the VA_DATA value without starting access on the SBA. Reading from the VA_DATA_INERT register returns the value of VA_DATA, writing to VA_DATA_INERT overwrites the value of VA_DATA.

The VCore shared bus is capable of returning error-indication when illegal register regions are accessed. If a VA access results in an error-indication from the SBA, the VA_CTRL.VA_ERR field is set, and the VA_DATA is set to 0x88888888.

Note:
SBA error indications only occur when non-existing memory regions or illegal registers are accessed. This will not happen during normal operation, so the VA_CTRL.VA_ERR indication is useful during debugging only.

Example: Reading from CPU::GRP[1] through the VA registers. The CPU::GPR[1] register is the second register in the SBA VCore Registers region. Set VA_ADDR_MSB to 0x to 0x6 and VA_ADDR_LSB to 0x00000004, read once from VA_DATA (and discard the read-value). Wait until VA_CTRL.VA_BUSY is cleared, then VA_DATA contains the value of the CPU::GRP[1] register. Using VA_DATA_INERT (instead of VA_DATA) to read the data is appropriate because this does not start a new SBA access.

5.5.4.1 Optimized Reading

SBA access is typically much faster than the CPU interface, which is used to access the VA registers. The VA_DATA register (VA_DATA_INCR and VA_DATA_INERT) return 0x88888888 while VA_CTRL.VA_BUSY is set. This means that it is possible to skip checking for busy between read access to SBA.

For example, after initiating a read access from SBA, software can proceed directly to reading from VA_DATA, VA_DATA_INCR, or VA_DATA_INERT

- If the second read is different from 0x88888888, then the second read returned valid read data (the SBA access was done before the second read was performed).
- If the second read is equal to 0x88888888, VA_CTRL must be read. If VA_CTRL.VA_BUSY_RD is cleared (and VA_CTRL.VA_ERR_RD is also cleared), then 0x88888888 is the actual read data. If VA_CTRL.VA_BUSY_RD is set, the SBA access was not yet done at the time of the second read. Start over again by repeating the read from VA_DATA.

Optimized reading can be used for single-read and sequential-read access. For sequential reads the VA_ADDR_LSB and VA_ADDR_MSB are only incremented on successful (non-busy) reads.

5.5.5 Mailbox and Semaphores

This section provides information about the semaphores and mailbox features for CPU-to-CPU communication. The following table lists the registers associated with mailbox and semaphores.

Table 5-14. Mailbox and Semaphore Registers

Register	Description
DEVCPU_ORG::MAILBOX_SET	Atomic set of bits in the mailbox register.
DEVCPU_ORG::MAILBOX_CLR	Atomic clear of bits in the mailbox register.
DEVCPU_ORG::MAILBOX	Current mailbox state.
DEVCPU_ORG::SEMA_CFG	Configuration of semaphore interrupts.

.....continued	
Register	**Description**
DEVCPU_ORG::SEMA0	Taking of semaphore 0.
DEVCPU_ORG::SEMA1	Taking of semaphore 1.
DEVCPU_ORG::SEMA0_OWNER	Current owner of semaphore 0.
DEVCPU_ORG::SEMA1_OWNER	Current owner of semaphore 1.

The mailbox is a 32-bit register that can be set and cleared atomically using any CPU interface (including the VCore CPU). The MAILBOX register allows reading (and writing) of the current mailbox value. Atomic clear of individual bits in the mailbox register is done by writing a mask to MAILBOX_CLR. Atomic setting of individual bits in the mailbox register is done by writing a mask to MAILBOX_SET.

The device implements two independent semaphores per DEVCPU_ORG target. The semaphores are part of the Switch Core Register Bus (CSR) block and are accessible by means of the fast switch core registers.

Any CPU can attempt to take a semaphore n by reading SEMA0.SEMA0 or SEMA1.SEMA1. If the result is 1, the corresponding semaphore was successfully taken and is now owned by that interface. If the result is 0, the semaphore was not free. After successfully taking a semaphore, all additional reads from the corresponding register will return 0. To release a semaphore write 1 to SEMA0.SEMA0 or SEMA1.SEMA1.

Note:
Any CPU can release semaphores; it does not have to be the one that has taken the semaphore, this allows implementation of handshaking protocols.

The current status for a semaphore is available in SEMA0_OWNER.SEMA0_OWNER and SEMA1_OWNER.SEMA1_OWNER. See register description for encoding of owners.

Software interrupt is generated when semaphores are freed or taken. Interrupt polarity is configured through SEMA_CFG.SEMA_INTR_POL. Semaphore 0 is hooked up to SW0 interrupt and semaphore 1 is hooked up to SW1 interrupt. For configuration of software-interrupt, see 5.10.13 Outbound Interrupt Controller.

In addition to interrupting on semaphore state, software interrupt can be manually triggered by writing directly to the CPU::INTR_FORCE register.

Software interrupts (SW0 and SW1) can be individually mapped to either the VCore CPU or to external interrupt outputs (to an external CPU).

5.6 PCIe Dual Mode Controller

This device implements a single-lane PCIe Gen3 dual mode controller which can operate in either endpoint or root complex mode and can be hooked up to any PCIe capable system.

Using the PCIe interface in endpoint mode, an external CPU can access (and control) the device. Ethernet frames can be injected and extracted using registers or the device can be configured to DMA Ethernet frames autonomously to or from the PCIe memory. The device's entire address space is available through the PCIe interface, using either direct or indirect accesses. Both the FDMA and the VCore-IV CPUs can generate PCIe read/write requests to any memory region in PCIe memory space. However, it is up to the software running on an external CPU to set up or communicate the appropriate PCIe memory mapping information to the device. The PCIe controller implements two PCIe physical functions, which facilitates the development of two independent software drivers for an external CPU. Function 0 is purposed for switch core configuration, while function 1 can be used for communicating with a PoE controller executing in the sub-CPU of VCore-IV.

As a root complex, the device can configure and control other PCIe endpoints.

The local address space of the device can be mapped to 32-bit or 64-bit memory regions in PCIe memory space through the ATU of the PCIe controller. The ATU supports 16 inbound and 16 outbound translations. The device implements 28GB of local address space which can be used for outbound requests. This space can be used for mapping endpoint requirements during enumeration at root complex mode, or it can be used to access the remote root complex memory space in endpoint mode.

The defaults for the PCIe controller's capabilities region and the extended capabilities region are listed in the registers list's description of the PCIe registers. The most important parameters are:

- Vendor (and subsystem vendor) ID: 0x101B (Vitesse Semiconductor)
- Device ID: 0xB006, device family ID
- Revision ID: 0x00, device family revision ID
- Class Code: 0x028000, Ethernet network controller
- Two physical functions in EP mode, non-Bridge, INTA, Message Signaled Interrupt (MSI), and MSI(-X) capable device

By default, both physical functions have the same vendor ID, device ID, revision ID, class, subsystem device ID and subsystem vendor ID. An external CPU may require that the two PCIe functions have different subsystem device IDs to load the appropriate driver. These parameters can be customized before enabling the PCIe device. However, this requires a manual bring-up procedure performed by software running locally on a VCore-IV CPU. The following table shows the registers for customizing these parameters per function. For more information, see 5.6.2.2 End Point Mode Initialization.

Table 5-15. Customization of device and vendor ID parameters for PCIe functions

Register	Description
PCIE_DM_EP::PF0_TYPE0_HDR_DEVICE_ID_VENDOR_ID_REG	Device and Vendor ID for PF0
PCIE_DM_EP::PF0_TYPE0_HDR_CLASS_CODE_REVISION_ID	Revision ID for PF0
PCIE_DM_EP::PF0_TYPE0_HDR_SUBSYSTEM_ID_SUBSYSTEM_VENDOR_ID_REG	Subsystem device and vendor ID for PF0
PCIE_DM_EP::PF1_TYPE0_HDR_DEVICE_ID_VENDOR_ID_REG	Device and Vendor ID for PF1
PCIE_DM_EP::PF1_TYPE0_HDR_CLASS_CODE_REVISION_ID	Revision ID for PF0
PCIE_DM_EP::PF1_TYPE0_HDR_SUBSYSTEM_ID_SUBSYSTEM_VENDOR_ID_REG	Subsystem device and vendor ID for PF1

The device family 0xB006 covers several register compatible devices. The software drivers must determine the actual device ID and revision by reading DEVCPU_GCB::CHIP_ID from the device's memory mapped registers region.

For information about base address registers, see 5.6.3.1 Base Address Registers.

The device is power management capable and implements PCI Bus Power Management Interface Specification Revision 1.2. For more information, see 5.6.7 Power Management.

The device is MSI, MSIX capable. Up to four 64-bit outbound messages are supported in endpoint mode. Messages can be generated on rising and falling edges of each of the two external VCore-IV interrupt destinations. Inbound MSI/MSI-X messages can trigger interrupts towards the VCore CPU. For more information, see 5.4.2 VCore CPU Interrupt Controller.

5.6.1 Accessing Controller Registers

All PCIe registers (including the configuration space registers) are mapped to the local address space of the device and can be accessed by the VCore-IV CPUs. During manual bring-up of the PCIe endpoint a VCore-IV CPU customizes most of the endpoint's read-only configuration registers, such as Vendor ID and BAR mask registers. The PCIe endpoint's read-only configuration register values must not be changed after the endpoint is enabled. PCIe registers can also be accessed by a remote root complex using PCIe MRd/MWr requests through BAR mappings. For more information about BAR registers, see 5.6.3.1 Base Address Registers.

Table 5-16. PCIe Controller Register Groups

Register Group	Description
PCIE_DM_EP	PCIe endpoint memory map
PCIE_DM_RC	PCIe root complex memory map
CPU:PCIE	PCIe DM controller configuration

The remote root complex accesses the PCIe configuration space registers of the device through PCIe CfgWr/CfgRd requests.

The remote root complex can access the MSI-X table of each function through PCIe MRd/MWr requests to BAR4 respectively. The ATU is accessible through BAR4 of Function 0. PCIe memory mappings of the device when acting as an endpoint are described in 5.6.3.1 Base Address Registers.

5.6.2 Device Initialization

PCIe device initialization is done by a VCore-IV CPU. The mode of operation (endpoint or root complex) is selected as part of the boot up process.

CPU:PCIE:PCIERST_CFG.POWERONRST_VAL = 0

CPU:PCIE:PCIERST_CFG.POWERONRST_FORCE = 1

CPU:CPU_REGS:PCIE_SYS_CFG.PCIE_RC_EP_MODE = 0 for endpoint, 1 for root complex

CPU:PCIE:PCIERST_CFG.POWERONRST_FORCE = 0

Once reset is deasserted, PCIe PHY takes approximately 3 ms to generate the stable pipe clk that drives core_clk towards PCIe Controller bringing it out of reset.

5.6.2.1 Root Complex Mode Initialization

To initialize the root complex mode, perform the following steps.

1. Disable automatic link initialization and training

   CPU:PCIE_CFG.LTSSM_DIS = 1
2. Optional Gen3 PHY equalization related configuration

   PCIE_DM_RC::GEN3_EQ_CONTROL_OFF

   PCIE_DM_RC::GEN3_EQ_FB_MODE_DIR_CHANGE_OFF
3. Enable automatic link initialization and training

   CPU:PCIE_CFG.LTSSM_DIS = 0

5.6.2.2 End Point Mode Initialization

To initialize the end point mode, perform the following steps.

1. Disable automatic link initialization and training

   CPU:PCIE_CFG.LTSSM_DIS = 1
2. Optional Gen3 PHY equalization related configuration

   PCIE_DM_EP::GEN3_EQ_CONTROL_OFF

   PCIE_DM_EP::GEN3_EQ_FB_MODE_DIR_CHANGE_OFF
3. Optionally change parameters listed in Table 5-15.
4. Optionally configure BAR advertised space and parameters by updating the registers listed in as shown in 5.6.3.1.2 Configuring the BAR registers. To disable a BAR write the corresponding mask to 0.
5. Optionally change the default mapping of BAR registers to the local address space as explained in 5.6.3.3 Mapping BARs to Local Address Space.
6. Enable automatic link initialization and training

   CPU:PCIE_CFG.LTSSM_DIS = 0

The last step in initialization process enables the controller to start link training, and once the link is operational, the root complex will then start the enumeration process and configure the PCIe endpoints for communication.

5.6.3 Inbound Requests

In root complex mode, all the inbound accesses falling outside the Base and Limit register offsets will be accepted. The controller can receive PCIE memory and message requests (MRd, MWr, Msg, and MsgD). For example, an endpoint accessing the device's DDR space or an endpoint raising an interrupt towards the root complex. Inbound accesses need to go through the IATU (Internal Address Translation Unit) to translate addresses from PCIe space to local address space.

In endpoint mode, all in bound configuration and message requests (CfgRd0, CfgRd1, CfgWr0, CfgWr1, Msg, and MsgD) will be accepted. An inbound memory request (MRd, MWr) will be accepted only if the request matches any of the supported base address registers (BARs).

5.6.3.1 Base Address Registers

As an endpoint, the device implements a Type 0 configuration header. Configuration space exists per function. Most parameters are configurable through the configuration registers in the following table. BAR4 of both functions is reserved to access internal registers of the controller (IATU and MSI-X configurations). All BARs have programmable masks except for BAR2 of PF1.

Table 5-17. Bar Configuration Registers

Register	Description
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR0_REG	PF0 BAR0 register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR0_MASK_REG	PF0 BAR0 mask register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR1_REG	PF0 BAR1 register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR1_MASK_REG	PF0 BAR1 mask register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR2_REG	PF0 BAR2 register
PCIE_DM_EP:PF0_TYPE0_HDR:BAR2_MASK_REG	PF0 BAR2 mask register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR3_REG	PF0 BAR3 register
PCIE_DM_EP:PF0_TYPE0_HDR:BAR3_MASK_REG	PF0 BAR3 mask register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR4_REG	PF0 BAR4 register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR4_MASK_REG	PF0 BAR4 mask register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR5_REG	PF0 BAR5 register
PCIE_DM_EP:PF0_TYPE0_HDR:PF0_TYPE0_HDR_BAR5_MASK_REG	PF0 BAR5 mask register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR0_REG	PF1 BAR0 register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR0_MASK_REG	PF1 BAR0 mask register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR1_REG	PF1 BAR1 register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR1_MASK_REG	PF1 BAR1 mask register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR2_REG	PF1 BAR2 register
Non-programmable mask for BAR2 PF1	
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR3_REG	PF1 BAR3 register
Non-programmable mask for BAR3 PF1	

.....continued	
Register	**Description**
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR4_REG	PF1 BAR4 register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR4_MASK_REG	PF1 BAR4 mask register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR5_REG	PF1 BAR5 register
PCIE_DM_EP:PF1_TYPE0_HDR:PF1_TYPE0_HDR_BAR5_MASK_REG	PF1 BAR5 mask register

5.6.3.1.1 Default BAR configuration

The following table lists the default memory space advertised by the PCIe physical functions.

Table 5-18. Default BAR Parameters

Physical Function	BAR	Region Type	Locatable	Prefetchable	Advertised Space	Programmable Mask
0	BAR0	Memory	64 bit	No	32 MB	Yes
0	BAR2	Memory	64 bit	No	1 MB	Yes
0	BAR4	Memory	64 bit	No	2 MB	No
1	BAR0	Memory	64 bit	No	32 MB	Yes
1	BAR2	Memory	64 bit	No	256 KB	No
1	BAR4	Memory	64 bit	No	2 MB	No

5.6.3.1.2 Configuring the BAR registers

The PCIe controller allows the user to configure the BAR registers by writing to the registers of Table 5-19. In order to allow gain access to these registers to modify their value the field CPU:PCIE:PCIE_CFG.PCIE_DBI_ACCESS_ENA must assume value 1. For example, to set the advertised size of BAR2 of PF0 to X MB, where X is a power of 2:

CPU:PCIE:PCIE_CFG.PCIE_DBI_ACCESS_ENA = 1

PCIE_DM_EP:PF0_TYPE0_HDR:BAR2_MASK_REG = (X-1)

CPU:PCIE:PCIE_CFG.PCIE_DBI_ACCESS_ENA = 0

5.6.3.2 Configuring Inbound Address Translation

The device implements a local inbound address translation in addition to the IATU implemented in the PCIe controller. This local translation is enabled by default, it is partially configurable and it can be bypassed through the registers listed in the following table.

Table 5-19. Local Inbound Address Translation Configuration Registers

Register	Description
CPU:PCIE:PCIEMST_PF1_BAR0_OFFSET_LOW	Configurable offset for PF1, BAR0
CPU:PCIE:PCIEMST_PF1_BAR0_OFFSET_HIGH	Configurable offset for PF1, BAR0
CPU:PCIE:PCIEMST_PF1_BAR0_MASK_LOW	Translation mask for PF1, BAR0
CPU:PCIE:PCIEMST_PF1_BAR0_MASK_HIGH	Translation mask for PF1, BAR0
CPU:PCIE:PCIEMST_PF0_BAR2_OFFSET_LOW	Configurable offset for PF0, BAR2
CPU:PCIE:PCIEMST_PF0_BAR2_OFFSET_HIGH	Configurable offset for PF0, BAR2

.....continued	
Register	**Description**
CPU:PCIE:PCIEMST_PF0_BAR2_MASK_LOW	Translation mask for PF0, BAR2
CPU:PCIE:PCIEMST_PF0_BAR2_MASK_HIGH	Translation mask for PF0, BAR2
CPU:PCIE:PCIESLV_IATU_BYPASS	Bypass local inbound translation

The IATU of the PCIE controller supports 16 inbound translation regions with configurable modes, offsets and sizes per translation. The VCore-IV CPU can access these registers when CPU:PCIE:PCIE_CFG.PCIE_DBI_ACCESS_ENA = 3.

Table 5-20. IATU Configuration Registers

Register	Description
PCIE_DM_EP:PF0_ATU_CAP	Configuration of PCIe controller IATU in endpoint mode
PCIE_DM_RC:PF0_ATU_CAP	Configuration of PCIe controller IATU in root-complex mode

5.6.3.2.1 Translation Example for Memory Access in RC Mode

Following code block is a translation example for memory access in the RC mode.

```
CPU:PCIE:PCIE_CFG.PCIE_DBI_ACCESS_ENA = 3
Register group PCIE_DM_RC:PF0_ATU_CAP
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>.IATU_REGION_CTRL_2_OFF_INBOUND_<idx>_REGION_EN
= 0x1
IATU_LWR_BASE_ADDR_OFF_INBOUND_<idx> = <Lower 32 bits of Base start address of
region>
IATU_UPPER_BASE_ADDR_OFF_INBOUND_<idx> = <Upper 32 bits of Base start address of
region>

If region size > 4GB,

IATU_REGION_CTRL_1_OFF_INBOUND_<idx>.IATU_REGION_CTRL_1_OFF_INBOUND_<idx>_INCREASE_R
EGION_SIZE = 0x1
IATU_LIMIT_ADDR_OFF_INBOUND_<idx> = <Lower 32 bits of end address of region>
IATU_UPPR_LIMIT_ADDR_OFF_INBOUND_<idx> = <Upper 32 bits of end address of region>

If region size < 4GB,

IATU_LIMIT_ADDR_OFF_INBOUND_<idx> = <Lower 32 bits of end address of region>

IATU_LWR_TARGET_ADDR_OFF_INBOUND_<idx> = <Lower 32 bits of Target address to
translate>
IATU_UPPER_TARGET_ADDR_OFF_INBOUND_<idx> = <Upper 32 bits of Target address to
translate>
CPU:PCIE:PCIE_CFG.PCIE_DBI_ACCESS_ENA = 0
```

5.6.3.2.2 Translation Example for Memory Access in EP Mode

As mentioned in 5.6.3 Inbound Requests, in endpoint mode memory accesses are accepted only if they are matched against a BAR region. The IATU of the PCIE controller allows to configure a translation on a BAR match. To use the IATU for inbound memory accesses in endpoint mode, the local inbound address translation must be bypassed. Note that the configuration of the IATU inbound translations for memory accesses in endpoint mode can be done by the external CPU after the enumeration. This is because the IATU configuration registers are statically mapped at offset 0x100000 of PF0 BAR4.

```
Register group PCIE_DM_EP:PF0_ATU_CAP
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>.
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>_REGION_EN = 1
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>.
```

```
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>_MATCH_MODE = 1
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>. IATU_REGION_CTRL_2_OFF_INBOUND_<idx>_BAR_NUM
= <BAR>
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>.
IATU_REGION_CTRL_2_OFF_INBOUND_<idx>_FUNC_NUM_MATCH_EN = 1
IATU_REGION_CTRL_1_OFF_INBOUND_<idx>.
IATU_REGION_CTRL_1_OFF_INBOUND_<idx>_CTRL_1_FUNC_NUM = <PF>
IATU_LWR_TARGET_ADDR_OFF_INBOUND_<idx> = <LSB of targeted offset>
IATU_UPPER_TARGET_ADDR_OFF_INBOUND_<idx> = <MSB of targeted offset>
```

5.6.3.3 Mapping BARs to Local Address Space

Inbound memory requests at endpoint mode are matched against the BAR registers. The address of a request is relative to the BAR offset. Requests to BAR0 and BAR2 of PF0 and PF1 are forwarded to the device. The address needs to be translated to access locations mapped in the local address space of the device. Requests to BAR4 of PF0 and PF1 are statically mapped to MSI-X configuration and are served by the PCIe controller itself as shown in the following table.

Table 5-21. Reserved Bars By Pcie Controller

Physical Function	BAR	Mapping
0	BAR4	0x0: MSI-X table
1	BAR4	0x0: MSI-X table

5.6.3.3.1 Default BAR Mapping

By default, the local inbound address translation described in 5.6.3.2 Configuring Inbound Address Translation is active. The default BAR mapping is shown in the following table.

Table 5-22. Default Pf0 Bar Mapping Using Local Inbound Address Translation

Physical Function	BAR	Mapping	Reconfigurable BAR
0	BAR0	0x610000000 32MB CSR Space	No
0	BAR2	0x600000000 2 MB Vcore CFG Space	Yes
1	BAR0	0x610000000 32MB CSR Space	Yes
1	BAR2	0x630000000 256KB of PoE Space	No

5.6.3.3.2 BAR Mapping using IATU

Alternatively, the IATU may be used to map BAR0 and BAR2 requests of both PF0 and PF1 to the local address space. In such case the local inbound translation needs to be bypassed by setting CPU:PCIE:PCIE_CFG.BYPASS_INBOUND_AT = 1.

The steps to set-up a BAR translation are listed in 5.6.3.2.2 Translation Example for Memory Access in EP Mode.

5.6.4 Outbound Accesses

In endpoint mode the device can initiate memory accesses towards the root complex. The device will trigger these accesses by writing/reading to/from the 28 GB PCIe target space. Before these accesses can be initiated, the remote root complex needs to configure the endpoint device to perform these accesses and program the outbound address translation accordingly. For example, the remote root complex needs to configure the FDMA in the device with the proper address offsets and setup an outbound translation so that the FDMA can access the remote memory in the root complex. Similarly, in case of MSI interrupt generation, the remote root complex has to configure the device with addresses as to where each of the interrupt vectors needs to be targeted.

In root complex mode the device can initiate configuration and memory accesses to remote endpoints. The device uses the 256 MB of PCIe EP cfg region to map and initiate config requests towards the endpoints. Also, it uses the 28 GB PCIe target region to map and initiate memory requests towards the endpoints. All of the outbound accesses of the device go through the iATU to map the local address to PCIe address space. The iATU can also be configured to generate configuration TLP's (CfgRd0, CfgRd1, CfgWr0, CfgWr1) while address translation is done for requests matching the region that handles the 256MB of PCIe endpoint configuration space.

5.6.4.1 Configuring Outbound Address Translation

Outbound address translation is carried out by configuring the iATU of the controller as shown below. These steps are the same both for RC and EP mode of operation. The device supports 16 independent outbound translation regions of configurable size 64 KB-16 GB.

5.6.4.1.1 Outbound Translation to Memory Space

The following example shows outbound translation to memory space

```
Register group PCIE_DM_EP/RC:PF0_ATU_CAP
IATU_REGION_CTRL_2_OFF_OUTBOUND_<idx>.IATU_REGION_CTRL_2_OFF_OUTBOUND_<idx>_REGION_E
N = 0x1
IATU_LWR_BASE_ADDR_OFF_OUTBOUND_<idx> = <Lower 32 bits of Base start address of
region>
IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_<idx> = <Upper 32 bits of Base start address of
region>

If region size > 4GB,
IATU_REGION_CTRL_1_OFF_OUTBOUND_<idx>.IATU_REGION_CTRL_1_OFF_OUTBOUND_<idx>_INCREASE
_REGION_SIZE = 0x1
IATU_LIMIT_ADDR_OFF_OUTBOUND_<idx> = <Lower 32 bits of end address of region>
IATU_UPPR_LIMIT_ADDR_OFF_OUTBOUND_<idx> = <Upper 32 bits of end address of region>
If region size < 4GB,
IATU_LIMIT_ADDR_OFF_OUTBOUND_<idx> = <Lower 32 bits of end address of region>

IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_<idx> = <Lower 32 bits of Target address to
translate>
IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_<idx> = <Upper 32 bits of Target address to
translate>
```

5.6.4.1.2 Outbound Translation to Configuration Space

Configuring an address translation to use Cfg TLPs to access the configuration space of remote endpoints is the same as in 5.6.4.1.1 Outbound Translation to Memory Space, except that the target address would be a 32 bit PCIe address with lower 16 bits being the register address while the upper 16 bits consists of BDF (Bus,Device, and Function) number. An additional configuration is required to convert the accesses to CFG TLP's.

```
PCIE_DM_RC:IATU_REGION_CTRL_1_OFF_OUTBOUND_<idx>
IATU_REGION_CTRL_1_OFF_OUTBOUND_0_TYPE = 0x4
```

5.6.5 Interrupt Generation

PCIe 3.0 DM controller in EP mode supports three modes of handling interrupts:

- Legacy mode
- MSI mode
- MSI-X mode

All the interrupt sources from the device are mapped to two destination interrupts, as described in 5.10.13.2 Interrupt Destination Configuration. While MSI or MSI-X mode of interrupt generation is enabled, these interrupts can be configured to generate an interrupt request to PCIE DM Controller either on rising edge or falling edge or both. Configuration pertaining to this is to be done in:

- CPU:PCIE_INTR_CFG[0-1].INTR_FALLING_ENA
- CPU:PCIE_INTR_CFG[0-1].INTR_RISING_ENA

PCIe interrupt controller serves the rising edge, falling edge interrupts in following order: R0->R1->F0->F1.

Since the device supports two functions as an endpoint, interrupts can be enabled per function. PCIe interrupts can be enabled per function via CPU:PCIE_INTR_COMMON_CFG[0-1].PCIE_INTR_ENA. When MSI or MSI-X interrupts

aren't enabled in PCIe Configuration space, legacy interrupts will be generated based on PCIE_INTR_ENA. Which of the two available interrupts should be used to generate a legacy interrupt for the function can be configured in CPU:PCIE_INTR_COMMON_CFG[0-1].LEGACY_MODE_INTR_SEL.

If MSI mode of interrupts is enabled in PCIe configuration space, based on INTR_RISING_ENA or INTR_FALLING_ENA enabled, an MSI interrupt request will be raised to PCIE DM Controller with function number, traffic class, vector number(in this case LSB 5 bits) information configured in CPU:PCIE_INTR_CFG[0-1].

MSIX mode of interrupts can be enabled per function by enabling them in configuration space as well as CPU:PCIE_INTR_COMMON_CFG[0-1].MSIX_INTR_ENA. Both the configurations should be in sync for MSIX to work. MSIX interrupt generation needs an address to be configured in CPU:PCIE:PCIE_MSIX_DBELL_LOW, CPU:PCIE:PCIE_MSIX_DBELL_HIGH. This is a special address which shouldn't be used for anything else in address map.(Hint: Use a reserved address from the local address space). Same selected doorbell address should be configured in MSIX_ADDRESS_MATCH_LOW_OFF, MSIX_ADDRESS_MATCH_HIGH_OFF controller register. Also, required information such as function number, traffic class, vector number need to be configured in CPU:PCIE_INTR_CFG[0-1].

5.6.6 Interrupt Reception

Acting as root complex, the device receives PCIe interrupts as legacy or MSI messages. For legacy interrupts, the PCIe controller asserts an interrupt towards the interrupt controller of the VCore-IV CPU. For MSI/MSI-X interrupts, the PCIe controller performs a write to a pre-configured address which was specified as part of the enumeration while configuring the endpoints. The interrupt controller of the VCore-IV CPU must be configured accordingly.

5.6.7 Power Management

The PCIe device supports D0, D1, and D3 device power-management states and associated link power-management states. The switch core does not automatically react to changes in the PCIe endpoint's power management states. It is, however, possible to enable a VCore-IV interrupt on device power state changes and then have the VCore-IV CPU software make application-specific changes to the device operation depending on the power management state. Because the device does not implement a dedicated auxiliary power for the PCIe device, the device is operated from the VDD core power supply. Before the power management driver initializes the device, software can force auxiliary power detection by writing PCIE_AUX_CFG = 3, which causes the device in endpoint mode to report that it is capable of emitting Power Management Events (PME) messages in the D3c state. The current device power management state is available at PCIE_STAT.PM_STATE. A change in this field's value sets the CIE_INTR.INTR_PM_STATE sticky bit.

Table 5-23. Power Management Registers

Register	Description
CPU::PCIE_CFG	Configuration of WAKE output
CPU::PCIE_STAT	Current power management state
CPU::PCIE_AUX_CFG	Configuration of auxiliary power detection
CPU::PCIE_INTR	PCIe interrupt sticky events
CPU::PCIE_INTR_ENA	Enable of PCIe interrupts
CPU::PCIE_INTR_IDENT	Currently interrupting PCIe sources

To enable this interrupt, set PCIE_INTR_ENA.INTR_PM_STATE_ENA. The current state of the PCIe device interrupt towards the VCore-IV interrupt controller is shown in PCIE_INTR_IDENT register (if different from zero, then interrupt is active). The endpoint can emit PMEs if the PME_En bit is set in the PM Capability Register Set and if the endpoint is in power-down mode. Outbound request from either SBA or FDMA trigger PME. A change in status for an enabled outbound interrupt (either legacy or MSI or MSIX) triggers PME. This feature can be disabled by setting CPU::PCIE_INTR_COMMON_CFG.WAKEUP_ON_INTR_DIS.

In the D3 state, the endpoint transmits a beacon. The beacon function can be disabled and instead drive the WAKE output using the overlaid GPIO function. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Disable WAKE by setting PCIE_CFG.WAKE_DIS. The polarity of the WAKE output is configured in PCIE_CFG.WAKE_POL. The drive scheme is configured in PCIE_CFG.WAKE_OE.

5.7 Frame DMA

This section describes the Frame DMA engine (FDMA). When FDMA is enabled, Ethernet frames can be extracted or injected autonomously to or from the device's DDR3/DDR3L memory and/or PCIe memory space. Linked list data structures in memory are used for injecting or extracting Ethernet frames. The FDMA generates interrupts when frame extraction or injection is done and when the linked lists need updating.

Table 5-24. Fdma Register List

Register	Description
CPU::FDMA_CH_ACTIVATE	Enables channels.
CPU::FDMA_CH_RELOAD	Adds DCBs to channels.
CPU::FDMA_CH_DISABLE	Disables channels.
CPU::FDMA_CH_FORCEDIS	Disables channels with force.
CPU::FDMA_CH_CNT	Per channel event counters.
CPU::FDMA_DCB_LLP	Linked-list pointer per channel.
CPU::FDMA_DCB_LLP_PREV	Previous linked-list pointer per channel.
CPU::FDMA_CH_ACTIVE	Indicates channels statuses.
CPU::FDMA_CH_PENDING	Channels indication on linked-list status.
CPU::FDMA_CH_CFG	Configuration per channel.
CPU::FDMA_CH_TRANSLATE	Per channel translation offset added to DCB, status and DB pointers.
CPU::FDMA_CH_INJ_TOKEN_CNT	Token counter per injection channel.
CPU::FDMA_CH_INJ_TOKEN_TICK_RLD	Periodic addition of tokens per injection channel.
CPU::FDMA_CH_INJ_TOKEN_TICK_CNT	Down-counter per injection channel used for the periodic addition of tokens.
CPU::FDMA_PORT_CFG	Configuration per port.
CPU::FDMA_PORT_CTRL	Per port control handles.
CPU::FDMA_INTR_DCB	DCB done event for channels.
CPU::FDMA_INTR_DCB_ENA	Enables interrupt on DCB done event.
CPU::FDMA_INTR_DB	DB done event for channels.
CPU::FDMA_INTR_DB_ENA	Enables interrupt on DB done event.
CPU::FDMA_INTR_ERR	Error event.
CPU::FDMA_INTR_ENA	Enables interrupts.
CPU::FDMA_INTR_IDENT	Identify interrupt source.
CPU::FDMA_ERRORS	Error event description.

.....continued	
Register	**Description**
CPU::FDMA_IDLECNT	Idle time counter.
CPU::FDMA_CTRL	FDMA reset.
CPU::FDMA_PORT_STAT	Per port counter of injected frames.

The FDMA implements two extraction channels, one per switch core port towards the VCore CPU system and a total of six injection channels. Extraction channels are mapped one-to-one to the CPU ports, while injection channels can be individually assigned to any CPU port.

FDMA channel 0 through 5 corresponds to CPU port 0 injection direction when FDMA_CH_CFG[channel].CH_INJ_PORT is set to 0.

FDMA channel 0 through 5 corresponds to CPU port 1 injection direction when FDMA_CH_CFG[channel].CH_INJ_PORT is set to 1.

FDMA channel 6 corresponds to CPU port 0 extraction direction.

FDMA channel 7 corresponds to CPU port 1 extraction direction.

The FDMA implements a strict priority scheme among channels. Extraction channels are prioritized over injection channels and secondarily channels with higher channel number are prioritized over channels with lower number. On the other hand, ports are being served on an equal-bandwidth principle both on injection and extraction directions. The equal-bandwidth principle will not force an equal bandwidth. Instead, it ensures that the ports perform at their best considering the operating conditions.

When more than one injection channels are enabled for injection on the same CPU port, then priority determines which channel can inject data. Ownership is re-arbitrated on frame boundaries.

5.7.1 DMA Control Block Structures

The FDMA processes linked lists of DMA Control Block Structures (DCBs). The DCBs have the same basic structure for both injection and extraction. A DCB must be placed on a 64-bit word-aligned address in memory. Each DCB has a per-channel configurable amount of associated data blocks in memory, where the frame data is stored. The data blocks that are used by extraction channels must be placed on 64-bit word aligned addresses in memory, and their length must be a multiple of 128 bytes. A DCB carries the pointer to the next DCB of the linked list, the INFO word which holds information for the DCB, and a pair of status word and memory pointer for every data block that it is associated with.

Figure 5-9. FDMA DCB Layout

NEXT PTR	
Reserved	INFO
DATA0 PTR	
Reserved	STATUS0
DATA1 PTR	
Reserved	STATUS1
DATA2 PTR	
Reserved	STATUS2
DATA3 PTR	
Reserved	STATUS3
DATA4 PTR	
Reserved	STATUS4
DATA5 PTR	
Reserved	STATUS5
DATA6 PTR	
Reserved	STATUS6
DATA7 PTR	
Reserved	STATUS7
DATA8 PTR	
Reserved	STATUS8
DATA9 PTR	
Reserved	STATUS9
DATA10 PTR	
Reserved	STATUS10
DATA11 PTR	
Reserved	STATUS11
DATA12 PTR	
Reserved	STATUS12
DATA13 PTR	
Reserved	STATUS13
DATA14 PTR	
Reserved	STATUS14

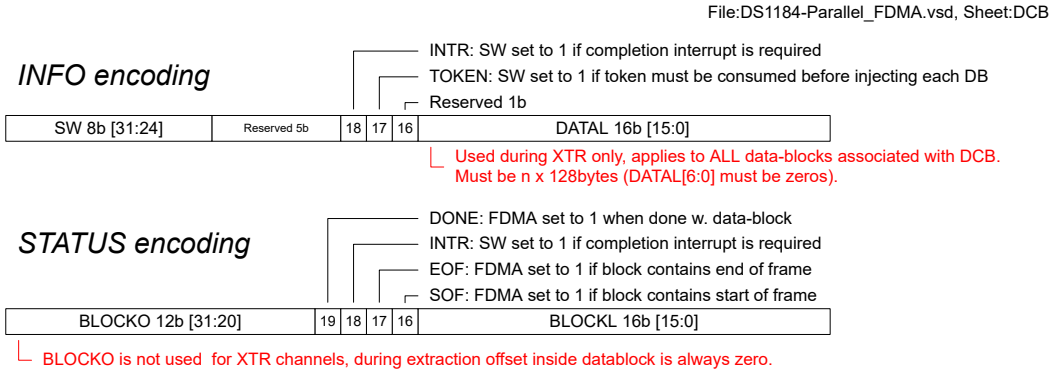
An Ethernet frame can be contained inside one data block (if the data block is big enough) or the frame can be spread across multiple data blocks. A data block never contains more than one Ethernet frame. Data blocks that contain start-of-frame and/or end-of-frame have special bits set in their status word in the DCB respectively.

The FDMA makes use of parallel accesses when fetching the data blocks, which may be served out of order. The FDMA reorders the fetched data before processing and injecting the frames.

Regarding injection channels, frame data inside the data blocks can be placed at any byte offset and have any byte length, provided that byte offset plus length does not exceed the size of the data block. Byte offset and length is configured using special fields inside the status word of every data block. Software can specify both when setting up injection DCBs.

For extraction channels, software should not specify the length of frame data in the data blocks; the FDMA automatically calculates and updates length for extraction. The block offset is not used in extraction. The FDMA will write the extracted data starting at the first address of a data block. The available space per data block is the same for all the data blocks associated to a DCB and it is set in the information word of the DCB itself.

Figure 5-10. Encoding of INFO and STATUS fields of DCB.



As mentioned before, the amount of data blocks that can be associated to a DCB is configurable. Moreover, a DCB does not have to use all of them. The FDMA processes the data-block related fields of a DCB in-order, and terminates the processing of a DCB when an invalid data-block field is encountered or when the end of the DCB is reached. For extraction DCBs, a data-block field is invalid if the data-block pointer is not 64-bit aligned. For injection DCBs, a data-block field is invalid if the block length is set to zero.

- Example for injection:

  A DCB for injection is set-up. It is configured to carry 12 pointers to data blocks but only 3 of them are used. The first, DATA0_PTR, points to a data block that contains a full frame. The frame data starts 5 bytes after the location indicated by DATA0_PTR. STATUS0 has SOF and EOF bits set. BLOCKL carries the length of frame data and BLOCKO indicates the offset between DATA0_PTR and frame data start location, which was said to be 5. The second and the third data blocks carry a frame that spreads across both of them. STATUS1 has SOF bit set and STATUS2 has EOF bit set. Each of the two statuses (STATUS1 and STATUS2) have BLOCKL set to the length of the partial frame that the respective data blocks hold. The frame data start 102 bytes after the location pointed by DATA1_PTR, therefore BLOCKO of STATUS1 is set to 102. Likewise, BLOCKO of STATUS3 is set to 56. The DCB is terminated by writing 0 to BLOCKL of STATUS3.
- Example for extraction

  A DCB for extraction is set-up. It is configured to carry 12 pointers to data blocks but only 3 of them are used. Three data blocks of equal size have been allocated in memory. The DATAL field of the DCB INFO is set to this size. DATA0_PTR, DATA1_PTR and DATA2_PTR are set to point to the 3 data blocks which were allocated in memory. The DCB is terminated by invalidating DATA3_PTR with a non-64-bit aligned address.

  DCBs are linked together by the DCB's NEXT_PTR field. The last DCB in a chain must have an invalid NEXT_PTR value. NEXT_PTR is invalid when it is not 64-bit aligned. Chains consisting of a single DCB are allowed.

5.7.2 Enabling, Disabling and adding DCBs to FDMA Channels

To enable a channel (ch), write a valid DCB pointer to FDMA_DCB_LLP[ch] and enable the channel by setting FDMA_CH_ACTIVATE.CH_ACTIVATE[ch]. This makes the FDMA load the DCB from memory and start either injection or extraction. The FDMA continually fetches DCBs for active channels until it reaches end of chain.

To schedule a channel for disabling, set FDMA_CH_DISABLE.CH_DISABLE[ch]. An active channel may not disable immediately. If the channel is currently fetching a DCB, it will finish fetching and then it will disable. If FDMA_CH_FORCEDIS.CH_FORCEDIS[ch] is used, the channel will be disabled immediately.
Note:
The data blocks of the fetched DCBs of a channel will be processed even though the channel may have been disabled.

Adding a DCB to an active channel is a 3-step process:

- CPU shall write the 64-bit NEXT_PTR field of the last DCB of the linked list with the LSB set to '1' and make sure that the write has been flushed to physical memory.
- CPU shall overwrite the LSB of NEXT_PTR field of the last DCB of the linked list to '0'.
- CPU shall set FDMA_CH_RELOAD.CH_RELOAD[ch] for the channel that is being modified.

This procedure requires that the last DCB in a chain must never be deallocated while an FDMA channel is active.
Note:
The FDMA does not have to reach to the end of a chain before the chain is extended.

Channels can be in one of three states: NOT ACTIVE, ACTIVE and PENDING, or ACTIVE and NOT PENDING. Channels are NOT ACTIVE by default. When the channel has not reached the end of its DCB chain or it is set to reload, and it is not scheduled to disable, then it is PENDING. Otherwise, it is NOT PENDING.

An interrupt can be set when a DCB has been fetched. In order for a DCB to trigger this interrupt, the field INTR in the INFO word of the DCB needs to be set. If this field is set for the last DCB of a chain, then an interrupt will be asserted when the DCB chain reaches to the last DCB.

Note:
The address of the current DCB is available in FDMA_DCB_LLP_PREV[ch].

5.7.3 Data Block Completion

Fetching DCBs and processing the data blocks that the DCBs are associated to are two decoupled procedures in the FDMA. When fetching DCBs, the FDMA buffers per channel the statuses and the locations of the data blocks. When the FDMA has finished with processing a data block, the FDMA will overwrite the status field of the block in the DCB, setting the DONE field. In case of extraction channels, additionally to the DONE field, the BLOCKO, BLOCKL, SOF, and EOF are also updated by the FDMA. BLOCKO is always set to 0 during extraction.

The FDMA can be set to emit an interrupt at data block completion. This is done by setting the INTR field of the status word of the data block in the DCB. Moreover, the FDMA can be configured through FDMA_CH_CFG[ch].CH_INTR_DB_EOF_ONLY to emit this interrupt only if the data block contains the end of a frame.

5.7.4 FDMA Events and Interrupts

Each FDMA channel can generate three events: DCB-event, DB-event and ERR-event. These events cause a bit to be set in FDMA_INTR_DCB, FDMA_INTR_DB and FDMA_INTR_ERR respectively.

DCB-event occurs when a DCB is fetched and the INFO word of the DCB has the INTR field set.

DB-event occurs when FDMA finishes processing a data block and the data block has the INTR field in its status word in the associated DCB set. A channel may be configured to trigger this interrupt only when the data block holds the end of a frame. This is done by setting FDMA_CH_CFG[ch].CH_INTR_DB_EOF_ONLY.

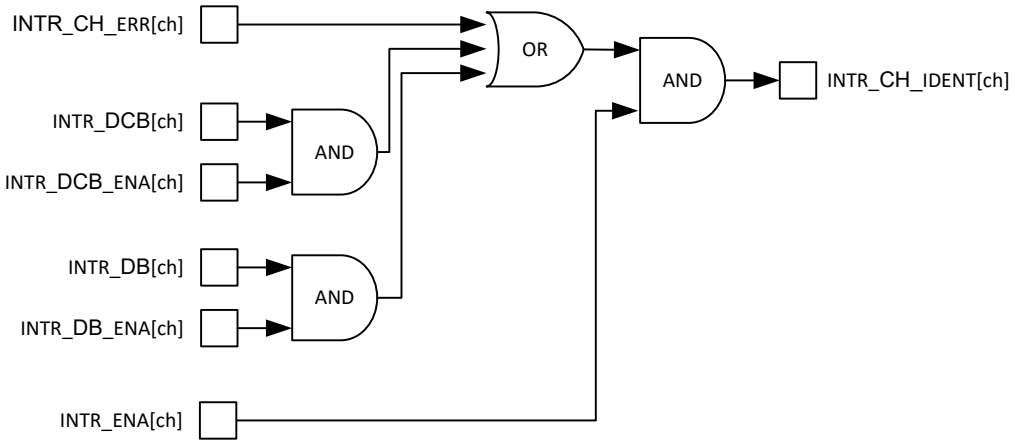
ERR-event is an error indication that is set for a channel or a port if it encounters a problem during normal operation. This indication is implemented so that software can take proper action. For more information about the errors that are detected and the FDMA response to these errors see FDMA_ERRORS in the register descriptions.

Both of the DCB and DB events can be enabled for channel interrupt through FDMA_INTR_DCB_ENA.INTR_DCB_ENA[ch] and FDMA_INTR_DB_ENA.INTR_DB_ENA[ch] respectively. The ERR event is always enabled.

The FDMA_INTR_ENA.INTR_CH_ENA[ch] field enables interrupt from individual channels and FDMA_INTR_IDENT.INTR_CH_IDENT[ch] field shows which channels are currently interrupting. While INTR_CH_IDENT is non-zero, the FDMA is indicating interrupt towards the VCore interrupt controller.

The following figure shows the FDMA channel interrupt hierarchy.

Figure 5-11. FDMA Channel Interrupt Hierarchy



5.7.5 FDMA Extraction

During extraction, the FDMA extracts Ethernet frame data from the Queuing System and saves it into the data block that is currently loaded by the FDMA extraction channel. The FDMA continually processes data blocks until it runs out of data blocks. A channel runs out of data blocks if it reaches the end of a DCB chain or it is disabled, and all of the previously accepted data blocks are used to extract frame data from the Queueing System.

When an extraction channel writes the status word of a data block, it updates the SOF, EOF, BLOCKL, BLOCKO, and DONE indications. BLOCKO is set to 0, as the FDMA writes in the data blocks without offset, always starting at the first location.

5.7.6 FDMA Injection

During injection, the FDMA reads out-of-order multiple Ethernet frame data blocks. The frame data is reordered before it is injected into the queuing system. The FDMA continually processes data blocks.

When an injection channel writes the status word of a data block, it sets the DONE indication. All other status word fields remain unchanged (write value is stored the first time the injection channel loads the DCB).

The rate by which the FDMA injects frames can be shaped by using tokens. Each injection channel has an associated token counter (FDMA_CH_INJ_TOKEN_CNT[ich]). A DCB that has the INFO.TOKEN field set causes the injection channel to deduct one from the token counter before every data block of the DCB can be injected. If the token counter is at 0, the injection channel postpones injection until the channel's token counter is set to a value different from 0.

Tokens can be added to the token counter by writing the FDMA_CH_INJ_TOKEN_CNT[ich] register. Tokens can also be added automatically (with a fixed interval) by using the dedicated token tick counter. Setting FDMA_CH_INJ_TOKEN_TICK_RLD[ich] to a value n (different from 0) will cause one token to be added to that injection channel every n x 200 ns.

The injection ports can be configured for 0-padding of injection frames to a configurable minimum frame length. Additionally, FCS 0-padding may be used, reserving space for possible FCS addition.

5.7.7 Counters

The FDMA implements the counters which are summarized in the following table.

Table 5-25. FDMA Counters

Register	Field	Description
FDMA::FDMA_CH_CNT[ch]	CH_CNT_FRM	Per-channel counter of DBs with EOF set in STATUS ()
	CH_CNT_DB_INTR	Per-channel counter of DBs with INTR set in STATUS ()
	CH_CNT_DCB	Per-channel counter of fetched DCBs.
FDMA_PORT_STAT[CPU port]	PORT_INJ_FRM_CNT	Per-port counter of frames that are injected to the switch core.

5.7.8 Switch Configuration

As the FDMA interacts with the switching fabric, some additional configuration is required to setup FDMA injection and extraction.

Table 5-26. Mapping Of CPU Ports To Switch Ports

CPU Port Index	Switch Port Index
0	65
1	66

Table 5-27. Register List For Switch Configuration

Register	Description
DEVCPU_QS::XTR_GRP_CFG	CPU port ownership, extraction direction.
DEVCPU_QS::INJ_GRP_CFG	CPU port ownership, injection direction.
QFWD::FRAME_COPY_CFG	Mapping of switch extraction queues to CPU ports.
DSM::DEV_TX_STOP_WM	Flow control configuration through watermark.
ASM::PORT_CFG	Enables IFH and disables FCS recalculation (for injected frames).

To perform extraction, perform the following steps.

1. Set ownership of CPU port extraction direction to FDMA by configuring DEVCPU_QS::XTR_GRP_CFG[CPU port].MODE = 2.
2. Map switch extraction queues to CPU ports by configuring QFWD::FRAME_COPY_CFG[0-7].FRM_PORT_VAL to the corresponding switch port indices.
3. Configure the flow control watermark in DSM by setting DSM::DEV_TX_STOP_WM_CFG[Switch port].DEV_TX_STOP_WM = 100.

To perform injection, perform the following steps.

1. Set ownership of CPU port injection direction to FDMA by configuring DEVCPU_QS::INJ_GRP_CFG[CPU port].MODE = 2.
2. Enable IFH, disable VSTAX awareness, and set no preamble prepended by setting:
   - ASM::PORT_CFG[Switch port].INJ_FORMAT_CFG = 0x1.
   - ASM::PORT_CFG[Switch port].NO_PREAMBLE_ENA = 1.
   - ASM::PORT_CFG[Switch port].VSTAX2_AWR_ENA =0

5.8 DDR (DDR3/DDR4) Memory Controller

This section describes the feature set supported by the DDR memory controller, basic overview about the memory controller, and how to configure and initialize the memory controller.

5.8.1 Features

The DDR memory controller supports the following features.

- Supports 32-bit SDRAM data bit interface with an optional 8-bit sideband ECC support
- Supports 1 or 2 memory ranks
- Programmable support for Half Data-bus Width (16-bit) memory configuration
- Programmable SDRAM parameters
- DDR PHY Interface (DFI) support for easy integration with industry standard DFI 3.1 compliant PHYs
- Flexible address mapper logic to allow application specific mapping of row, column, bank, and rank bits
- Option ECC support
  - Single error correction/double error detection (SEC/DED)
  - Automatic data scrubbing of correctable errors
  - Support for highly efficient read-modify-write when byte enables are used with ECC enabled
  - Automatic logging of both correctable and uncorrectable errors
  - Ability to “poison” the write data by adding correctable/uncorrectable errors, for use in testing ECC error handling
- DDR4 protocol up to 1250 MT/s are supported by the DDR memory controller. The following DDR4 specific features are supported:
  - Data Bus Inversion (DBI)
  - Maximum power saving mode
  - Multi-purpose register (MPR) reads and writes
  - Per DRAM addressability
  - Fine granularity refresh
  - Cyclic redundancy check (CRC) on write data
  - Command/Address Parity
  - Programmable Preamble
- APB interface for the memory controller software accessible registers
- Host port with AMBA AXI interface

5.8.2 Overview

The DDR memory controller has single AMBA 4 AXI port on the host side to accept memory access requests. The configuration registers are programmed through the AMBA 3.0 APB software interface.

The memory controller works with JEDEC® compliant DDR3/DDR4 memory modules. The controller operates at Max 500 MHz to support the speed of 2000 MT/s on DDR3 interface and it operates at Max 312.5 MHz to support the speed of 1250 MT/s on DDR4 interface. It supports five-byte lanes (32-bit data bus along with 8-bit sideband ECC) in either one or two ranks memory configuration. The controller supports SDRAM with memory data width = x8 or x16. It also supports half data-bus width mode operation where it uses only lower 16-bit of total 32-bit data bus. The controller supports up to 18-bit address bus. The maximum amount of physical memory that can be attached to the controller (per byte lane) is 2 gigabytes.

The memory controller supports optional Single Bit Single Error Correction / Double Error Detection Error Correction code (SECDED ECC) by using one of the byte lane (via DDR_UMCTL2::ECCCFG0.ECC_MODE = 0x4). In full bus width mode, it uses fifth data lane for sending the ECC information, and in case of half bus width mode, it uses third byte lane for sending the ECC information. The 8-bit ECC code will be stored in a separate SDRAM memory (n x 8) where n is the size of the memory.

The memory controller receives read transaction, write transaction, and data through the AMBA 4 AXI port. These transactions are queued internally and scheduled for access in order to the SDRAM while satisfying SDRAM protocol timing requirements, transactions priorities, and dependencies between the transactions. The memory controller in

turn issues commands on the DDR PHY Interface (DFI) to the PHY module, which launches and captures data to and from the SDRAM. Figure is the system-level block diagram of the DDR memory controller.

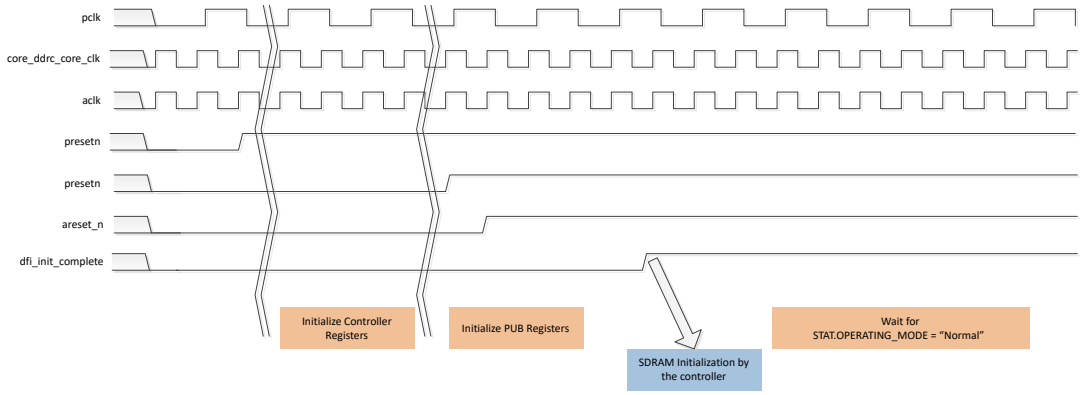
5.8.3 Initialization

This section and the subsequent sections describe the programming sequence that must be executed at power-up to bring the DDR memory controller, the PHY, and the memories into a state where DDR reads and writes can be performed.

5.8.3.1 Controller Initialization

This section provides information about how to configure and initialize the DDR memory controller. The following figure shows the relationship between clocks and resets for initialization of registers.

Figure 5-12. Simplified Relationship Between Clocks and Resets for Register Initialization



After power-up, the following steps are required to bring up the memory controller:

1. Assert the DDR controller reset (core_ddrc_rstn), AMBA AXI reset (aresetn_0), APB reset for DDR controller and DDR PHY (presetn), and DDR PHY reset (ctl_rst_n)
2. Start the DDR clock (core_ddrc_core_clk), AMBA AXI clock (aclk_0), APB clock for DDR controller and DDR PHY (pclk), and DDR PHY clock (phy_ctl_clk)
3. Deassert APB reset for DDR controller (presetn) once the clocks are active and stable
4. Allow 10 APB clock cycles for synchronization of APB reset (presetn) to DDR clock (core_ddrc_core_clk) and AMBA AXI clock (aclk) domains and to permit initialization of end logic
5. Initialize the memory controller registers
6. Deassert the DDR controller reset (core_ddrc_rstn) and AMBA AXI reset (aresetn_0)
7. Deassert the DDR PHY reset (ctl_rst_n) and APB reset for DDR PHY (presetn)

The following table lists the registers that are used to control the clocks and resets of the DDR memory controller and DDR PHY.

Table 5-28. Clocks and Resets registers for DDR controller and DDR PHY

Registers	Description
CPU::DDRCTRL_RST	DDR controller/Phy reset configuration
CPU::DDRCTRL_CLK	DDR Controller/Phy clock configuration
CPU::RESET.MEM_RST	Reset Settings

The following table defines the value for DDR4 SDRAM timing parameters, which are required while configuring the DDR memory controller and DDR PHY.

Table 5-29. DDR4 Timing and Mode Parameters

Parameter	Value
tCK	1600 ps

.....continued

Parameter	Value
tRFC(min)	2 Gb = 160 ns 4 Gb = 260 ns 8 Gb = 350 ns 16 Gb = 550 ns
tXPR	tRFC(min) + 10 ns
DFI_CLK_PRD	3.2 ns
tZQINITc	1024 nCK
MR	0xA40
EMR	0x501
EMR2	0x20
EMR3	0x0
MR4	0x0
MR5	0x400
MR6	0x899
CLc	16 nCK
CWLc	14 nCK
RD_PREAMBLE	1
WR_PREAMBLE	1
tCAL	0
ALc	0 (AL disabled) CLc – 1 nCK CLc – 2 nCK
BL	8
tWRc	15
tWRc_CRC _DM	(tWRc + 5) nCK
tFAWc_512	16 nCK
tFAWc_1K	21 nCK
tFAWc_2K	30 nCK
tFAWc	x4 device = tFAWc_512 x8 device = tFAWc_1K x16 device = tFAWc_2K
tRASc_MIN	33 nCK
tRASc_MAX	70200 nCK
tRCc	48 nCK
tXPc	6 nCK

.....continued

Parameter	Value
tRTPc	8 nCK
tWTRc	8 nCK
tWTRc_L_CRC _DM	(tWTRc + 5) nCK
tPLc	4 tCK
WLc	(ALc + CWLc) nCK if CA parity is disabled (ALc + CWLc + tPLc) nCK if CA Parity is enabled
RLc	(ALc + CLc) nCK if CA parity is disabled (ALc + CLc + tPLc) nck if CA Parity is enabled
tMRDc	8
tMODc	24 nCK
tMODc_PAR	(tMODc + tPLc) nCK
tMRDc_PAR	(tMODc + tPLc) nCK
tMOD	0 if (tMODc == 24) 1 if (tMODc == 25) 2 if (tMODc == 26) 3 if (tMODc == 27) 4 if (tMODc == 28) 5 if (tMODc == 29) 6 if (tMODc == 30)
tRCDc	15 nCK
tRPc	15 nCK
tCCDc_L	6 nCK
tRRDc_L_512	6 nCK
tRRDc_L_1K	6 nCK
tRRDC_L_2K	7 nCK
tRRDc	x4 device = tRRDc_L_512 x8 device = tRRDc_L_1K x16 device = tRRDc_L_2K
tCKSRXc	10 nCK
tCKSREc	10 nCK
tCKEc	5 nCK
tXS_tRFCc	Roundup(tXPR / tCK) nCK
tDLLKc	768 nCK

.....continued	
Parameter	**Value**
tXS_tRFC4c	2Gb = 100 nCK 4Gb = 120 nCK 8Gb = 170 nCK 16Gb = 270 nCK
tCCDc_S	4
tRRDc_S_512	4 nCK
tRRDc_S_1K	4 nCK
tRRDc_S_2K	6 nCK
tRRDc_S	x4 device = tRRDc_S_512 x8 device = tRRDc_S_1K x16 device = tRRDc_S_2K
tWTRc_S	3 nCK
tWTRc_S_CRC _DM	(tWTRc_S + 5) nCK
tREFIc	Roundup(7800000 / tCK) nCK
tRFCc_2G	160 nCK
tRFCc_4G	260 nCK
tRFCc_8G	350 nCK
tRFCc_16G	550 nCK
tRFCc	2Gb = tRFCc_2G 4Gb = tRFCc_4G 8Gb = tRFCc_8G 16Gb = tRFCc_16G
tPLc	4 tCK

The following table defines the value for DDR3 SDRAM timing parameters, which are required while configuring the DDR memory controller and DDR PHY.

Table 5-30. DDR3 Timing and Mode Parameters

Parameter	Value
tCK	2000 MT/s = 1000 ps
tRFC(min)	512 Mb = 90 ns 1 Gb = 110 ns 2 Gb = 160 ns 4 Gb = 260 ns 8 Gb = 350 ns
tXPR	tRFC(min) + 10 ns
DFI_CLK_PRD	2000 MT/s = 2.0 ns

.....continued

Parameter	Value
tZQINITc	640 nCK
MR	0x1124
EMR	0x0
EMR2	0x28
EMR3	0x0
CLc	2000 MT/s = 14 nCK
CWLc	2000 MT/s = 10 nCK
ALc	0 (AL disabled) CLc – 1 nCK CLc – 2 nCK
BL	8
tWRc	2000 MT/s = 15
tFAWc_1K	2000 MT/s = 25 nCK
tFAWc_2K	2000 MT/s = 35 nCK
tFAWc	x4 device = tFAWc_2K if (DENSITY == 8 Gb) else tFAWc_1K x8 device = tFAWc_2K if (DENSITY == 8 Gb) else tFAWc_1K x16 device = tFAWc_2K
tRASc_MIN	2000 MT/s = 33 nCK
tRASc_MAX	2000 MT/s = 70200 nCK
tRCc	2000 MT/s = 47 nCK
tXPc	2000 MT/s = 6 nCK
tXPDLLc	2000 MT/s = 24 nCK
tRTPc	2000 MT/s = 8 nCK
tWTRc	2000 MT/s = 8 nCK
WLc	(ALc + CWLc) nCK
RLc	(ALc + CLc) nCK
tMRDc	4
tMODc	2000 MT/s = 15 nCK
tMOD	0 if (tMODc == 12) 1 if (tMODc == 13) 2 if (tMODc == 14) 3 if (tMODc == 15) 4 if (tMODc == 16) 5 if (tMODc == 17)
tRCDc	2000 MT/s = 14 nCK

.....continued	
Parameter	**Value**
tRPc	2000 MT/s = 14 nCK
tCCDc	2000 MT/s = 4 nCK
tRRDc_1K	2000 MT/s = 5 nCK
tRRDC_2K	2000 MT/s = 6 nCK
tRRDc	x4 device = tRRDc_2K if (DENSITY == 8 Gb) else tRRDc_1K x8 device = tRRDc_2K if (DENSITY == 8 Gb) else tRRDc_1K x16 device = tRRDc_2K
tCKSRXc	2000 MT/s = 10 nCK
tCKSREc	2000 MT/s = 10 nCK
tCKEc	2000 MT/s = 5 nCK
tXS_tRFCc	Roundup(tXPR / tCK) nCK
tDLLKc	2000 MT/s = 512 nCK
tREFIc	Roundup(7800000 / tCK) nCK
tRFCc_512M	2000 MT/s = 90 nCK
tRFCc_1G	2000 MT/s = 110 nCK
tRFCc_2G	2000 MT/s = 160 nCK
tRFCc_4G	2000 MT/s = 260 nCK
tRFCc_8G	2000 MT/s = 350 nCK
tRFCc	512 Mb = tRFCc_512M 1 Gb = tRFCc_1G 2Gb = tRFCc_2G 4Gb = tRFCc_4G 8Gb = tRFCc_8G 16Gb = tRFCc_16G

The following table describes the memory controller registers configuration for accessing DDR4 SDRAM. The values for these registers are sampled directly in the destination domains (core_ddrc_core_clk/aclk_0 domains) without synchronizing circuits. The deassertion of the signal presetn is synchronized to the destination domain clocks and used to deassert the reset on the static registers in their destination domains. Therefore, these registers must be programmed when core_ddrc_core_clk/aclk_0 are stable, present is deasserted (at least 4 cycles of clock in the destination domain) and core_ddrc_rstn/aresetn_0 are asserted.

Table 5-31. Memory Controller Configuration for DDR4 SDRAM

Register Field	Description
DDR_UMCTL2::MSTR.DDR3	Set to 0
DDR_UMCTL2::MSTR.DDR4	Set to 1 to select DDR4 SDRAM
DDR_UMCTL2::MSTR.DATA_BUS_WIDTH	Set to 0x0 for full DQ Bus width mode Set to 0x1 for half DQ Bus width mode

.....continued

Register Field	Description
DDR_UMCTL2::MSTR.ACTIVE_RANKS	Set to 0x0 for one rank device Set to 0x1 for two ranks device
DDR_UMCTL2::MSTR.DEVICE_CONFIG	Set to 0x1 to indicate x8 device used in the system Set to 0x2 to indicate x16 device used in the system
DDR_UMCTL2::INIT0.PRE_CKE_x1024	Set to Roundup(500000 ns / (DFI_CLK_PRD x 1024 x 2))
DDR_UMCTL2::INIT0.POST_CKE_x1024	Set to Roundup(tXPR / (DFI_CLK_PRD x 1024 x 2))
DDR_UMCTL2::INIT1.DRAM_RSTN_x1024	Set to Roundup(200000 ns / (DFI_CLK_PRD x 1024 x 2))
DDR_UMCTL2::INIT3.MR	Set to MR
DDR_UMCTL2::INIT3.EMR	Set to EMR
DDR_UMCTL2::INIT4.EMR3	Set to EMR3
DDR_UMCTL2::INIT4.EMR2	Set to EMR2
DDR_UMCTL2::INIT5.DEV_ZQINIT_x32	Roundup(tZQINITc / (2 x 32)) + 1
DDR_UMCTL2::INIT6.MR5	Set to MR5
DDR_UMCTL2::INIT6.MR6	Set to MR6
DDR_UMCTL2::INIT6.MR4	Set to MR4
DDR_UMCTL2::DBICTL	Set to 0x1
DDR_UMCTL2::ODTCFG.WR_ODT_HOLD	Set to 0x6
DDR_UMCTL2::ODTCFG.WR_ODT_DELAY	Set to tCAL
DDR_UMCTL2::ODTCFG.RD_ODT_HOLD	Set to 0x6
DDR_UMCTL2::ODTCFG.RD_ODT_DELAY	Set to (CLc – CWLc – RD_PREAMBLE + WR_PREAMBLE + tCAL)
DDR_UMCTL2::DRAMTMG0.WR2PRE	Set to ((WLc + (BL/2) + tWRc) / 2)
DDR_UMCTL2::DRAMTMG0.T_FAW	Set to Roundup(tFAWc / 2)
DDR_UMCTL2::DRAMTMG0.T_RAS_MIN	Set to Rounddown(tRASc_MIN / 2)
DDR_UMCTL2::DRAMTMG0.T_RAS_MAX	Set to Rounddown((tRASc_MAX – 1) / (2 x 1024))
DDR_UMCTL2::DRAMTMG1.T_RC	Set to Roundup(tRCc / 2)
DDR_UMCTL2::DRMATMG1.T_XP	Set to Roundup(tXPc / 2)
DDR_UMCTL2::DRAMTMG1.RD2PRE	Set to ((ALc + tRTPc) / 2)
DDR_UMCTL2::DRAMTMG2.WR2RD	Set to Roundup((CWLc + (BL / 2) + tWTRc) / 2)
DDR_UMCTL2::DRAMTMG2.WRITE_LATENCY	Set to Roundup(WLc / 2)
DDR_UMCTL2::DRAMTMG2.READ_LATENCY	Set to Roundup(RLc / 2)
DDR_UMCTL2::DRAMTMG2.RD2WR	Set to Roundup((RLc + (BL / 2) + 1 + WR_PREAMBLE - WLc))
DDR_UMCTL2::DRAMTMG3.T_MRD	Set to Roundup((tMRDc + tCAL) / 2)

.....continued

Register Field	Description
DDR_UMCTL2::DRAMTMG3.T_MOD	Set to Roundup((tMODc + tCAL) / 2)
DDR_UMCTL2::DRAMTMG4.T_RCD	Set to Roundup((tRCDc – ALc) / 2)
DDR_UMCTL2::DRAMTMG4.T_RP	Set to Rounddown(tRPc / 2) + 1
DDR_UMCTL2::DRAMTMG4.T_CCD	Set to Roundup(tCCDc_L / 2)
DDR_UMCTL2::DRAMTMG4.T_RRD	Set to Roundup(tRRDc / 2)
DDR_UMCTL2::DRAMTMG5.T_CKSRX	Set to Roundup(tCKSRXc / 2)
DDR_UMCTL2::DRAMTMG5.T_CKSRE	Set to Roundup(tCKSREc / 2)
DDR_UMCTL2::DRAMTMG5.T_CKESR	Set to Roundup((tCKEc + 1) / 2)
DDR_UMCTL2::DRAMTMG5.T_CKE	Set to Roundup(tCKEc / 2)
DDR_UMCTL2::DRAMTMG8.T_XS_FAST_x32	Set to Roundup(tXS_tRFC4c / (2 x 32)) + 1
DDR_UMCTL2::DRAMTMG8.T_XS_ABORT_x32	Set to Roundup(tXS_tRFC4c / (2 x 32))
DDR_UMCTL2::DRAMTMG8.T_XS_DLL_x32	Set to Roundup(tDLLKc / (2 x 32))
DDR_UMCTL2::DRAMTMG8.T_XS_x32	Set to Roundup(tXS_tRFCc / (2 x 32))
DDR_UMCTL2::DRAMTMG9.DDR4_WR_PREAMBLE	Set to 0
DDR_UMCTL2::DRAMTMG9.T_CCD_S	Set to Roundup((tCCDc_S + 1) / 2)
DDR_UMCTL2::DRAMTMG9.T_RRD_S	Set to Roundup(tRRDc_S / 2)
DDR_UMCTL2::DRAMTMG9.WR2RD_S	Set to Roundup((CWLc + (BL / 2) + tWTRc_S) / 2)
DDR_UMCTL2::DFITMG0.DFI_TPHY_WRLAT	Set to ((WLc + tCAL - 2) / 2)
DDR_UMCTL2::DFITMG0.DFI_T_RDDATA_EN	Set to ((RLc + tCAL – 4) / 2)
DDR_UMCTL2::DFITMG0.DFI_TPHY_WRDATA	Set to 1
DDR_UMCTL2::DFITMG0.DFI_T_CTRL_DELAY	Set to 2
DDR_UMCTL2::DFITMG1.DFI_T_DRAM_CLK_ENABLE	Set to 1
DDR_UMCTL2::DFITMG1.DFI_T_DRAM_CLK_DISABLE	Set to 2
DDR_UMCTL2::DFITMG1.DFI_T_CMD_LAT	Set to tCAL
DDR_UMCTL2::DFITMG1.DFI_T_WRDATA_DELAY	Set to 3
DDR_UMCTL2::DFIUPD0.DIS_AUTO_CTRLUPD_SRX	Set to 1
DDR_UMCTL2::DFIUPD1	Set to 0x4000FF
DDR_UMCTL2::RFSHCTL0.REFRESH_BURST	Set to 1
DDR_UMCTL2::RFSHTMG.T_RFC_NOM_x32	Set to (tREFIc / (2 x 32))
DDR_UMCTL2::RFSHTMG.T_RFC_MIN	Set to (tRFCc / 2)
DDR_UMCTL2::RFSHCTL1	Set to 0x400020
DDR_UMCTL2::RANKCTL.DIFF_RANK_RD_GAP	Set to 2
DDR_UMCTL2::RANKCTL.DIFF_RANK_WR_GAP	Set to 2
DDR_UMCTL2::ADDRMAP0.ADDRMAP_CS_BIT0	x16 device = 23 x8 device = 24

.....continued	
Register Field	**Description**
DDR_UMCTL2::ADDRMAP1.ADDRMAP_BANK_B0	Set to 24
DDR_UMCTL2::ADDRMAP1.ADDRMAP_BANK_B1	Set to 24
DDR_UMCTL2::ADDRMAP1.ADDRMAP_BANK_B2	Set to 63
DDR_UMCTL2::ADDRMAP2	Set to 0x0
DDR_UMCTL2::ADDRMAP3	Set to 0x0
DDR_UMCTL2::ADDRMAP4.ADDMAP_COL_B10	Set to 31
DDR_UMCTL2::ADDRMAP4.ADDMAP_COL_B11	Set to 31
DDR_UMCTL2::ADDRMAP5	Set to 0x04040404
DDR_UMCTL2::ADDRMAP6	Set to 0x04040404
DDR_UMCTL2::ADDRMAP7	Set to 0x00000F0F
DDR_UMCTL2::ADDRMAP8.ADDRMAP_BG_B0	Set to 26
DDR_UMCTL2::ADDRMAP8.ADDRMAP_BG_B1	x16 device = 63 x8 device = 26
DDR_UMCTL2::ECCCFG0.ECC_MODE	Set to 0x4 to enable ECC mode (SECDED)

The following table describes the memory controller register configuration for accessing DDR3 SDRAM.

Table 5-32. Memory Controller Configuration for DDR3 SDRAM

Register Field	Description
DDR_UMCTL2::MSTR.DDR3	Set to 1 to select DDR3 SDRAM
DDR_UMCTL2::MSTR.DATA_BUS_WIDTH	Set to 0x0 for full DQ Bus width mode Set to 0x1 for half DQ Bus width mode
DDR_UMCTL2::MSTR.ACTIVE_RANKS	Set to 0x0 for one rank device Set to 0x1 for two ranks device
DDR_UMCTL2::MSTR.DEVICE_CONFIG	Set to 0x1 to indicate x8 device used in the system Set to 0x2 to indicate x16 device used in the system
DDR_UMCTL2::INIT0.PRE_CKE_x1024	Set to Roundup(500000 ns / (DFI_CLK_PRD x 1024 x 2))
DDR_UMCTL2::INIT0.POST_CKE_x1024	Set to Roundup(tXPR / (DFI_CLK_PRD x 1024 x 2))
DDR_UMCTL2::INIT1.DRAM_RSTN_x1024	Set to Roundup(200000 ns / (DFI_CLK_PRD x 1024 x 2))
DDR_UMCTL2::INIT3.MR	Set to MR
DDR_UMCTL2::INIT3.EMR	Set to EMR
DDR_UMCTL2::INIT4.EMR3	Set to EMR3
DDR_UMCTL2::INIT4.EMR2	Set to EMR2
DDR_UMCTL2::INIT5.DEV_ZQINIT_x32	Roundup(tZQINITc / (2 x 32)) + 1
DDR_UMCTL2::ODTCFG.WR_ODT_HOLD	Set to 0x6

.....continued

Register Field	Description
DDR_UMCTL2::ODTCFG.WR_ODT_DELAY	Set to 0x0
DDR_UMCTL2::ODTCFG.RD_ODT_HOLD	Set to 0x6
DDR_UMCTL2::ODTCFG.RD_ODT_DELAY	Set to (CLc – CWLc)
DDR_UMCTL2::DRAMTMG0.WR2PRE	Set to ((WLc + (BL/2) + tWRc) / 2)
DDR_UMCTL2::DRAMTMG0.T_FAW	Set to Roundup(tFAWc / 2)
DDR_UMCTL2::DRAMTMG0.T_RAS_MIN	Set to Rounddown(tRASc_MIN / 2)
DDR_UMCTL2::DRAMTMG0.T_RAS_MAX	Set to Rounddown((tRASc_MAX – 1) / (2 x 1024))
DDR_UMCTL2::DRAMTMG1.T_RC	Set to Roundup(tRCc / 2)
DDR_UMCTL2::DRMATMG1.T_XP	Set to Roundup(tXPc / 2)
DDR_UMCTL2::DRAMTMG1.RD2PRE	Set to ((ALc + tRTPc) / 2)
DDR_UMCTL2::DRAMTMG2.WR2RD	Set to Roundup((CWLc + (BL / 2) + tWTRc) / 2)
DDR_UMCTL2::DRAMTMG2.RD2WR	Set to Roundup((RLc + (BL / 2) + 2 - WLc))
DDR_UMCTL2::DRAMTMG3.T_MRD	Set to Roundup(tMRDc / 2)
DDR_UMCTL2::DRAMTMG3.T_MOD	Set to Roundup(tMODc / 2)
DDR_UMCTL2::DRAMTMG4.T_RCD	Set to Roundup((tRCDc – ALc) / 2)
DDR_UMCTL2::DRAMTMG4.T_RP	Set to Rounddown(tRPc / 2) + 1
DDR_UMCTL2::DRAMTMG4.T_CCD	Set to Roundup(tCCDc / 2)
DDR_UMCTL2::DRAMTMG4.T_RRD	Set to Roundup(tRRDc / 2)
DDR_UMCTL2::DRAMTMG5.T_CKSRX	Set to Roundup(tCKSRXc / 2)
DDR_UMCTL2::DRAMTMG5.T_CKSRE	Set to Roundup(tCKSREc / 2)
DDR_UMCTL2::DRAMTMG5.T_CKESR	Set to Roundup((tCKEc + 1) / 2)
DDR_UMCTL2::DRAMTMG5.T_CKE	Set to Roundup(tCKEc / 2)
DDR_UMCTL2::DRAMTMG8.T_XS_DLL_x32	Set to Roundup(tDLLKc / (2 x 32))
DDR_UMCTL2::DRAMTMG8.T_XS_x32	Set to Roundup(tXS_tRFCc / (2 x 32))
DDR_UMCTL2::DFITMG0.DFI_TPHY_WRLAT	Set to ((WLc - 2) / 2)
DDR_UMCTL2::DFITMG0.DFI_T_RDDATA_EN	Set to ((RLc – 4) / 2)
DDR_UMCTL2::DFITMG0.DFI_TPHY_WRDATA	Set to 1
DDR_UMCTL2::DFITMG0.DFI_T_CTRL_DELAY	Set to 2
DDR_UMCTL2::DFITMG1.DFI_T_DRAM_CLK_ENABLE	Set to 1
DDR_UMCTL2::DFITMG1.DFI_T_DRAM_CLK_DISABLE	Set to 2
DDR_UMCTL2::DFITMG1.DFI_T_WRDATA_DELAY	Set to 3
DDR_UMCTL2::DFIUPD0.DIS_AUTO_CTRLUPD_SRX	Set to 1
DDR_UMCTL2::DFIUPD1	Set to 0x4000FF
DDR_UMCTL2::RFSHCTL0.REFRESH_BURST	Set to 1
DDR_UMCTL2::RFSHTMG.T_RFC_NOM_x32	Set to (tREFIc / (2 x 32))

.....continued	
Register Field	**Description**
DDR_UMCTL2::RFSHTMG.T_RFC_MIN	Set to (tRFCc / 2)
DDR_UMCTL2::RFSHCTL1	Set to 0x400020
DDR_UMCTL2::RANKCTL.DIFF_RANK_RD_GAP	Set to 2
DDR_UMCTL2::RANKCTL.DIFF_RANK_WR_GAP	Set to 2
Example Register configuration for ADDRMAP0-8 using DDR3_x16_4Gb device	
DDR_UMCTL2::ADDRMAP0.ADDRMAP_CS_BIT0	Set to 22
DDR_UMCTL2::ADDRMAP1.ADDRMAP_BANK_B0	Set to 23
DDR_UMCTL2::ADDRMAP1.ADDRMAP_BANK_B1	Set to 23
DDR_UMCTL2::ADDRMAP1.ADDRMAP_BANK_B2	Set to 23
DDR_UMCTL2::ADDRMAP2	Set to 0x0
DDR_UMCTL2::ADDRMAP3	Set to 0x0
DDR_UMCTL2::ADDRMAP4.ADDMAP_COL_B10	Set to 31
DDR_UMCTL2::ADDRMAP4.ADDMAP_COL_B11	Set to 31
DDR_UMCTL2::ADDRMAP5	Set to 0x04040404
DDR_UMCTL2::ADDRMAP6	Set to 0x0F040404
DDR_UMCTL2::ADDRMAP7	Set to 0x00000F0F
DDR_UMCTL2::ADDRMAP8.ADDRMAP_BG_B0	Set to 63
DDR_UMCTL2::ADDRMAP8.ADDRMAP_BG_B1	Set to 63
DDR_UMCTL2::ECCCFG0.ECC_MODE	Set to 0x4 to enable ECC mode (SECDED)

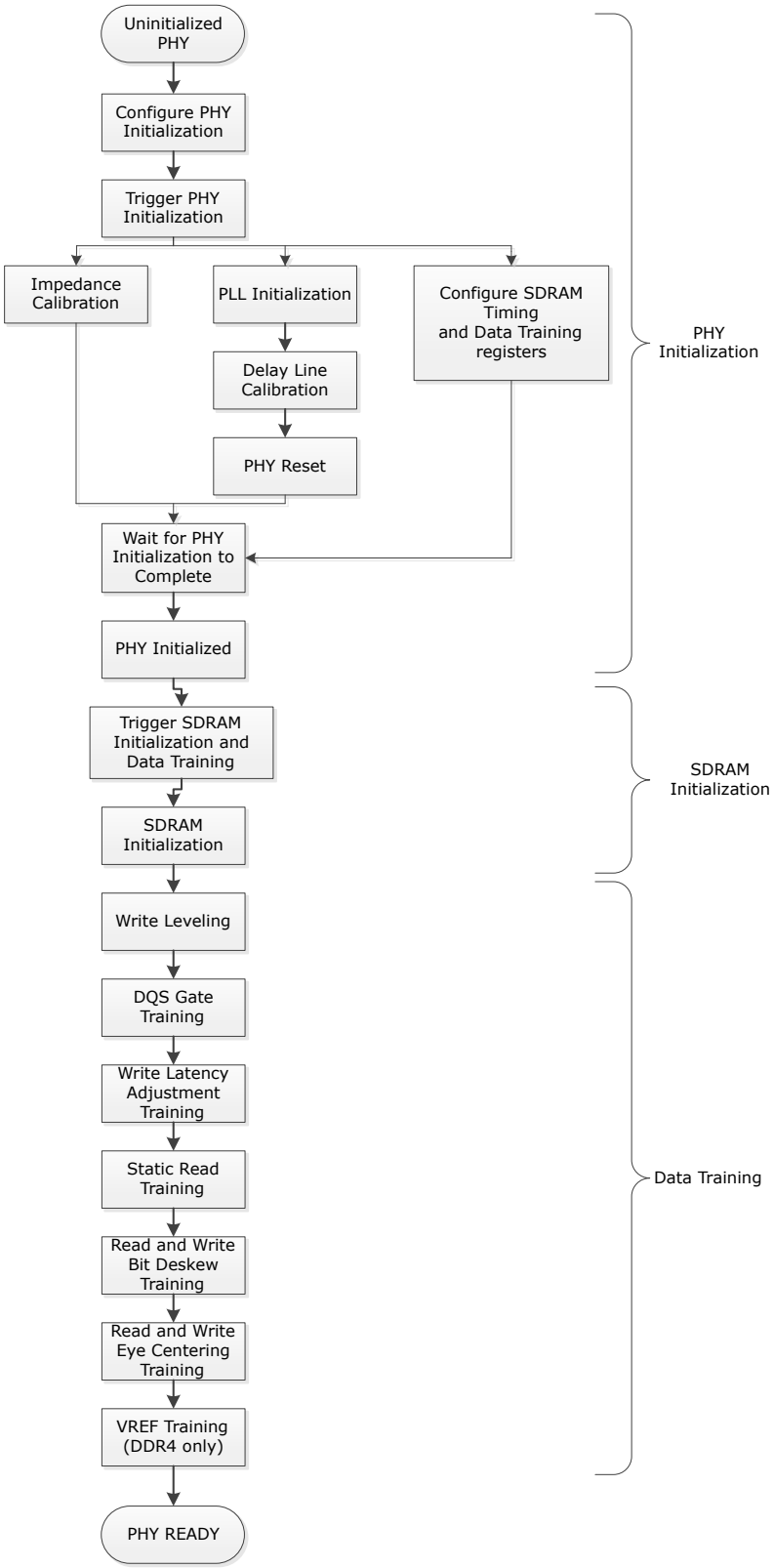
5.8.4 PHY Initialization

Before the controller can access the memory, the PHY must be initialized. The initialization sequence has the following phases.

- PHY initialization
- DRAM initialization
- Data training

The following figure shows a high-level illustration of the initialization sequence of the PHY.

Figure 5-13. PHY Initialization Flow



The following table summarizes the PHY initialization procedure.

Table 5-33. PHY Initialization Procedure

Steps	Description	Notes
1	After configuring all the controller registers, deassert core_ddrc_rstn and aresetn_0.	—
2	Start PHY initialization by accessing relevant PUB registers (for example, DXnGCR, DCR, PTR*, MR*, DTPR*)	Please refer Table 5-34
3	PHY initialization is comprised of initializing the PHY PLL(s), running the initial impedance calibration, running delay line calibration, and assertion of PHY reset. These functions can all be triggered at the same time by writing DDR_PHY::PIR = x0000_0073. The initial impedance calibration can be run in parallel with the PLL initialization and subsequent delay line calibration.	—
4	Monitor PHY initialization status by polling the PUB register DDR_PHY::PGSR0.IDONE.	—
5	Indicate to the PUB that the controller performs SDRAM initialization by setting DDR_PHY::PIR.INIT and DDR_PHY::PIR.CTLDINIT, and poll DDR_PHY::PGSR0.IDONE.	—
6	Set DDR_UMCTL2::SWCTL.SW_DONE to 0	—
7	Set DDR_UMCTL2::DFIMISC.DFI_INIT_COMPLETE_EN to 1	Controller waits for phy_dfi_init_complete to be asserted
8	Set DDR_UMCTL2::SWCTL.SW_DONE to 1	—
9	Wait for DDR_UMCTL2::SWSTAT.SW_DONE_ACK to become 1	—
10	Wait for controller to move to “normal” operating mode by monitoring STAT.OPERATING_MODE signal	Controller performs SDRAM initialization automatically

The following table lists the PUB registers to be configured for the PLL initialization and delay line calibration.

Table 5-34. Configuration of PHY Initialization registers

Register Field	Description
DDR_PHY::PGCR1.IODDRM	Set to 0x1
DDR_PHY::PGCR2.TREFPRD	Set to (tRASc_max – 400)
DDR_PHY::PTR0.TPLLPD	Set to 500
DDR_PHY::PTR0.TPLLGS	Set to 2000
DDR_PHY::PTR1.TPLLLOCK	Set to 12500
DDR_PHY::PTR1.TPLLRST	Set to 4500
DDR_PHY::ZQCR.PGWAIT	Set to 7

5.8.5 SDRAM Initialization

During the previous phase (PHY initialization), it should be noted that when the PUB is performing PHY initialization, the PUB registers associated with SDRAM timing and Data training can be configured. The following table lists the

PUB registers associated with the (DDR4) SDRAM timing and Data training sequence. Refer Table 5-29 for the parameter values.

Table 5-35. PHY Utility Block (PUB) Configuration for DDR4 SDRAM

Register Field	Description
DDR_PHY::DCR.DDRMD	Set to 0x4 for DDR4
DDR_PHY::SCHCR1.ALLRANK	Set to 1 for multi-rank configuration
DDR_PHY::MR0_DDR4	Set to MR
DDR_PHY::MR1_DDR4	Set to EMR
DDR_PHY::MR2_DDR4	Set to EMR2
DDR_PHY::MR3_DDR4	Set to EMR3
DDR_PHY::MR4_DDR4	Set to MR4
DDR_PHY::MR5_DDR4	Set to MR5
DDR_PHY::MR6_DDR4	Set to MR6
DDR_PHY::DTPR0.TRTP	Set to tRTPc
DDR_PHY::DTPR0.TRP	Set to tRPc
DDR_PHY::DTPR0.TRAS	Set to tRASc_MIN
DDR_PHY::DTPR0.TRRD	Set to tRRDc
DDR_PHY::DTPR1.TMRD	Set to tMRDc
DDR_PHY::DTPR1.TMOD	Set to tMOD
DDR_PHY::DTPR1.TFAW	Set to tFAWc
DDR_PHY::DTPR2.TXS	Set to tXS_tRFCc
DDR_PHY::DTPR2.TCKE	Set to tCKEc
DDR_PHY::DTPR3.TDLLK	Set to tDLLKc
DDR_PHY::DTPR4.TXP	Set to tXPc
DDR_PHY::DTPR4.TRFC	Set to tRFCc
DDR_PHY::DTPR5.TWTR	Set to tWTRc
DDR_PHY::DTPR5.TRCD	Set to tRCDc
DDR_PHY::DTPR5.TRC	Set to tRCc
DDR_PHY::PTR3.TDINIT0	Set to 0x7A120
DDR_PHY::PTR3.TDINIT1	Set to tXS_tRFCc
DDR_PHY::PTR4.TDINIT2	Set to 0x30D40
DDR_PHY::PTR4.TDINIT3	Set to tZQINITc
DDR_PHY::DXCCR.DQSRES	Set to 0x4 for selecting the on-die pull-down resistor = 500 ohm for DQS pins
DDR_PHY::DXCCR.DQSNRES	Set to 0xC for selecting the on-die pull-up resistor = 500 ohm for DQSN pins
DDR_PHY::DSGCR.CUAEN	Set to 1
DDR_PHY::DSGCR.SDRMODE	Set to 0x0

.....continued	
Register Field	**Description**
DDR_PHY::DTCR0.DTRPTN	Set to 0xF
DDR_PHY::DTCR0.DTMPR	Set to 1
DDR_PHY::DTCR0.RFSHDT	Set to 2
DDR_PHY::DTCR1.RANKEN	Set to number of ranks that are enabled for data-training and write-levelling. For one-rank device, set to 0x1 For two-rank device, set to 0x3
DDR_PHY::PGCR7.WRPSTEX	Set to 0x1
DDR_PHY::DX4GCR0.DX4GCR0_DXEN	Set to 0x0 if ECC mode is disabled
DDR_PHY::DX0GTR0.DX0GTR0_DGSL	Set to 0x2
DDR_PHY::DX1GTR0.DX1GTR0_DGSL	Set to 0x2
DDR_PHY::DX2GTR0.DX2GTR0_DGSL	Set to 0x2
DDR_PHY::DX3GTR0.DX3GTR0_DGSL	Set to 0x2
DDR_PHY::DX4GTR0.DX4GTR0_DGSL	Set to 0x2
DDR_PHY::RANKIDR.RANKWID	Set to 0x1 to configure values for Rank1
DDR_PHY::DX0GTR0.DX0GTR0_DGSL	Set to 0x2
DDR_PHY::DX1GTR0.DX1GTR0_DGSL	Set to 0x2
DDR_PHY::DX2GTR0.DX2GTR0_DGSL	Set to 0x2
DDR_PHY::DX3GTR0.DX3GTR0_DGSL	Set to 0x2
DDR_PHY::DX4GTR0.DX4GTR0_DGSL	Set to 0x2

The following table lists the PUB registers associated with the (DDR3) SDRAM timing and Data training sequence. Refer Table 5-30 for the parameter values.

Table 5-36. PHY Utility Block (PUB) Configuration for DDR3 SDRAM

Register Field	Description
DDR_PHY::DCR.DDRMD	Set to 0x3 for DDR3
DDR_PHY::SCHCR1.ALLRANK	Set to 1 for multi-rank configuration
DDR_PHY::MR0	Set to MR
DDR_PHY::MR1	Set to EMR
DDR_PHY::MR2	Set to EMR2
DDR_PHY::MR3	Set to EMR3
DDR_PHY::DTPR0.TRTP	Set to tRTPc
DDR_PHY::DTPR0.TRP	Set to tRPc
DDR_PHY::DTPR0.TRAS	Set to tRASc_MIN
DDR_PHY::DTPR0.TRRD	Set to tRRDc
DDR_PHY::DTPR1.TMRD	Set to tMRDc

.....continued

Register Field	Description
DDR_PHY::DTPR1.TMOD	Set to tMOD
DDR_PHY::DTPR1.TFAW	Set to tFAWc
DDR_PHY::DTPR2.TXS	Set to tXS_tRFCc
DDR_PHY::DTPR2.TCKE	Set to (tCKEc + 1)
DDR_PHY::DTPR3.TDLLK	Set to tDLLKc
DDR_PHY::DTPR4.TXP	Set to tXPDLLc
DDR_PHY::DTPR4.TRFC	Set to tRFCc
DDR_PHY::DTPR5.TWTR	Set to tWTRc
DDR_PHY::DTPR5.TRCD	Set to tRCDc
DDR_PHY::DTPR5.TRC	Set to tRCc
DDR_PHY::PTR3.TDINIT0	Set to 0x7A120
DDR_PHY::PTR3.TDINIT1	Set to tXS_tRFCc
DDR_PHY::PTR4.TDINIT2	Set to 0x30D40
DDR_PHY::PTR4.TDINIT3	Set to tZQINITc
DDR_PHY::DXCCR.DQSRES	Set to 0x4 for selecting the on-die pull-down resistor = 500 ohm for DQS pins
DDR_PHY::DXCCR.DQSNRES	Set to 0xC for selecting the on-die pull-up resistor = 500 ohm for DQSN pins
DDR_PHY::DSGCR.CUAEN	Set to 1
DDR_PHY::DSGCR.SDRMODE	Set to 0x0
DDR_PHY::DTCR0.DTRPTN	Set to 0xF
DDR_PHY::DTCR0.DTMPR	Set to 1
DDR_PHY::DTCR0.RFSHDT	Set to 2
DDR_PHY::DTCR1.RANKEN	Set to number of ranks that are enabled for data-training and write-levelling. For one-rank device, set to 0x1 For two-rank device, set to 0x3
DDR_PHY::PGCR7.WRPSTEX	Set to 0x1
DDR_PHY::DX4GCR0.DX4GCR0_DXEN	Set to 0x0 if ECC mode is disabled
DDR_PHY::DX0GTR0.DX0GTR0_DGSL	Set to 0x2
DDR_PHY::DX1GTR0.DX1GTR0_DGSL	Set to 0x2
DDR_PHY::DX2GTR0.DX2GTR0_DGSL	Set to 0x2
DDR_PHY::DX3GTR0.DX3GTR0_DGSL	Set to 0x2
DDR_PHY::DX4GTR0.DX4GTR0_DGSL	Set to 0x2
DDR_PHY::RANKIDR.RANKWID	Set to 0x1 to configure values for Rank1
DDR_PHY::DX0GTR0.DX0GTR0_DGSL	Set to 0x2

.....continued	
Register Field	**Description**
DDR_PHY::DX1GTR0.DX1GTR0_DGSL	Set to 0x2
DDR_PHY::DX2GTR0.DX2GTR0_DGSL	Set to 0x2
DDR_PHY::DX3GTR0.DX3GTR0_DGSL	Set to 0x2
DDR_PHY::DX4GTR0.DX4GTR0_DGSL	Set to 0x2

After the PHY PLLs have been initialized and the delay lines and PHY SSTL I/Os have been calibrated, the interface is ready for initializing the SDRAMs. The SDRAM must be correctly initialized before further training of the PHY can be executed. In order to perform SDRAM initialization by memory controller, the DDR_PHY::PIR register of the PUB block must be programmed with 0x00040001 to transfer control of the DFI interface from the PUB to the PHY. This is only required for the first SDRAM initialization triggered after reset. The status of the SDRAM initialization should be monitored by polling the PUB register DDR_PHY::PGSR0.IDONE. The following table summarizes the DDR4 SDRAM initialization sequence executed by the memory controller.

Table 5-37. DDR4 SDRAM Initialization Sequence

Steps	Description
1	Maintain dfi_reset_n low for duration specified by INIT1.DRAM_RSTN_X1024
2	Issue NOP/deselect for duration specified by INIT0.PRE_CKE_X1024
3	Assert CKE and issue NOP/deselect for INIT0.POST_CKE_X1024
4	Issue MRS (mode register set) command to load MR3 with INIT4.EMR3 value followed by NOP/deselect for duration of DRAMTMG3.T_MRD
5	Issue MRS (mode register set) command to load MR6 with INIT7.MR6 value followed by NOP/deselect for duration of DRAMTMG3.T_MRD
6	Issue MRS (mode register set) command to load MR5 with INIT6.MR5 value followed by NOP/deselect for duration of DRAMTMG3.T_MRD.
7	Issue MRS (mode register set) command to load MR4 with INIT7.MR4 value followed by NOP/deselect for duration of DRAMTMG3.T_MRD.
8	Issue MRS command to load MR2 with INIT4.EMR2 followed by NOP/deselect for duration of DRAMTMG3.T_MRD.
9	Issue MRS command to load MR1 with INIT4.EMR followed by NOP/deselect for duration of DRAMTMG3.T_MRD.
10	Issue MRS command to load MR0 with INIT3.MR followed by NOP/deselect for duration of DRAMTMG3.T_MOD.
11	Issue ZQCL command to start ZQ calibration and wait for INIT5.DEV_ZQINIT_X32.
12	Wait for INIT5.DEV_ZQINIT_X32 counting to finish.
13	The controller is now ready for normal operation.

The following table summarizes the DDR3 SDRAM initialization sequence executed by the memory controller.

Table 5-38. DDR3 SDRAM Initialization Sequence

Steps	Description
1	Maintain dfi_reset_n low for duration specified by INIT1.DRAM_RSTN_X1024
2	Issue NOP/deselect for duration specified by INIT0.PRE_CKE_X1024
3	Assert CKE and issue NOP/deselect for INIT0.POST_CKE_X1024

.....continued	
Steps	**Description**
4	Issue MRS (mode register set) command to load MR2 with INIT4.EMR2 value followed by NOP/deselect for duration of DRAMTMG3.T_MRD
5	Issue MRS (mode register set) command to load MR3 with INIT4.EMR3 value followed by NOP/deselect for duration of DRAMTMG3.T_MRD
6	Issue MRS (mode register set) command to load MR1 with INIT3.EMR value followed by NOP/deselect for duration of DRAMTMG3.T_MRD.
7	Issue MRS (mode register set) command to load MR0 with INIT3.MR value followed by NOP/deselect for duration of DRAMTMG3.T_MRD.
11	Issue ZQCL command to start ZQ calibration and wait for INIT5.DEV_ZQINIT_X32.
12	Wait for INIT5.DEV_ZQINIT_X32 counting to finish.
13	The controller is now ready for normal operation.

5.8.6 Data Training

After the PHY and SDRAM have been successfully initialized, the PHY may be trained for optimum operating timing margins. This includes write leveling, the training of the DQS gating during reads, write latency adjustment, bit deskew, and the training of the read and write data eyes. The following table summarizes the data training procedure for DDR4 SDRAM.

Table 5-39. Data Training Procedure for DDR4 SDRAM

Steps	Description	Notes
1	Disable auto-refreshes and power-down by setting DDR_UMCTL2::RFSHCTL3.DIS_AUTO_REFRESH = 1, DDR_UMCTL2::PWRCTL.POWERDOWN_EN = 0, and set DDR_UMCTL2::DFIMISC.DFI_INIT_COMPLETE_EN to 0.	Mask transitions in phy_dfi_init_complete during PUB training
2	Assert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=0) to remove the initial data corruption seen in the read data.	
3	Deassert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=1).	
4	Configure DDR_PHY::PIR register with value 0x00000E01 to trigger write levelling, DQS gate training, and write latency adjust training.	
5	Monitor the status of above training operation by polling the PUB register DDR_PHY::PGSR0.IDONE. If above training operation is successful, trigger static read training, read and write bit deskew training, read and write eye centering training, and VREF training (DDR4 only). Perform below steps to trigger rest of the data training operations in static read mode.	
6	Configure DDR_PHY::PGCR3.RDMODE to 0x1, to enable static read mode.	
7	Configure DDR_PHY::PIR register with value 0x0003F001 to execute static read training, read and write bit deskew training, read and write eye centering training, and VREF training (DDR4 only).	
8	Monitor the status of above training operation by polling the PUB register DDR_PHY::PGSR0.IDONE. After detecting DDR_PHY::PGSR0.IDONE = 1, read the content of DDR_PHY::PGSR0 register to check the status of all the steps performed during data training operations.	
9	Compare the read data (DDR_PHY::PGSR0) with value 0x80006FFF. If the read data matches with the value, data training operation is completed successfully.	

.....continued		
Steps	**Description**	**Notes**
10	Assert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=0).	
11	Deassert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=1).	
12	Enable auto-refreshes by setting DDR_UMCTL2::RFSHCTL3.DIS_AUTO_REFRESH = 0, and set DDR_UMCTL2::DFIMISC.DFI_INIT_COMPLETE_EN to 1.	After successful completion of the data training operation, keep the power-down mode in disable state as SDRAM is now available for normal write/read access.

The following table summarizes the data training procedure for DDR3 SDRAM.

Table 5-40. Data Training Procedure for DDR3 SDRAM

Steps	Description	Notes
1	Disable auto-refreshes and power-down by setting DDR_UMCTL2::RFSHCTL3.DIS_AUTO_REFRESH = 1, DDR_UMCTL2::PWRCTL.POWERDOWN_EN = 0, and set DDR_UMCTL2::DFIMISC.DFI_INIT_COMPLETE_EN to 0.	Mask transitions in phy_dfi_init_complete during PUB training
2	Assert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=0) to remove the initial data corruption seen in the read data.	
3	Deassert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=1).	
4	Configure DDR_PHY::PIR register with value 0x00000E01 to trigger write levelling, DQS gate training, and write latency adjust training.	
5	Monitor the status of above training operation by polling the PUB register DDR_PHY::PGSR0.IDONE. If above training operation is successful, trigger static read training, read and write bit deskew training, read and write eye centering training, and VREF training (DDR4 only). Perform below steps to trigger rest of the data training operations in static read mode.	
6	Configure DDR_PHY::PGCR3.RDMODE to 0x1, to enable static read mode.	
7	Configure DDR_PHY::PIR register with value 0x0001F001 to execute static read training, read and write bit deskew training, and read and write eye centering training.	
8	Monitor the status of above training operation by polling the PUB register DDR_PHY::PGSR0.IDONE. After detecting DDR_PHY::PGSR0.IDONE = 1, read the content of DDR_PHY::PGSR0 register to check the status of all the steps performed during data training operations.	
9	Compare the read data (DDR_PHY::PGSR0) with value 0x80002FFF. If the read data matches with the value, data training operation is completed successfully.	
10	Assert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=0).	
11	Deassert the PHY FIFO reset (DDR_PHY::PGCR0.PHYFRST=1).	

.....continued		
Steps	**Description**	**Notes**
12	Enable auto-refreshes by setting DDR_UMCTL2::RFSHCTL3.DIS_AUTO_REFRESH = 0, and set DDR_UMCTL2::DFIMISC.DFI_INIT_COMPLETE_EN to 1.	After successful completion of the data training operation, keep the power-down mode in disable state as SDRAM is now available for normal write/read access.

5.8.7 DDR4 Specific Features

This section describes the unique features of DDR4 configurations, which are supported by memory controller and DDR PHY.

5.8.7.1 Data Bus Inversion (DBI)

Data Bus inversion is a power saving DDR4 specific feature which aims at minimizing:

- DDR4: Logic 0s (consume more power than logic 1s)
- Simultaneous switching outputs

This is done by inverting each byte of read or write data, if the number of 0s (DDR4) in that byte is greater than 4. A DBI signal (1 bit per byte of data) is used to indicate whether this has been done or not. For writes, the DBI signal is shared with dfi_wrdata_mask on the DFI interface. For reads, there is a dedicated dfi_rddata_dbi signal. DBI is supported only for x8 and x16 devices. It is not supported for x4 devices. For DDR4, both write DBI and Data Mask cannot be enabled simultaneously. As the write DBI signal uses the dfi_wrdata_mask, masked writes are not allowed in DBI mode for DDR4.

The DFI Specification indicates that DBI can be implemented either in the controller or in PHY. In the controller, DBI is controlled by DDR_UMCTL2::DFIMISC.PHY_DBI_MODE. If this is set to ‘1’, DBI generation and data inversion are performed in the PHY. The read and write data on the DFI interface are same as that of when DBI is disabled (though masked writes are not allowed), and the signal dfi_rddata_dbi is ignored. When DDR_UMCTL2::DFIMISC.PHY_DBI_MODE is set to 0, DBI generation and data inversion is done by the controller. During write operation, the DBI value is put on the signal dfi_wrdata_mask. During read operation, dfi_rddata_dbi is evaluated and used to determine if the polarity of the read data needs to be inverted.

The register DBICTL contains three fields namely RD_DBI_EN, WR_DBI_EN, and DM_EN, which correspond to the three fields in the DRAM mode register, to enable, read DBI, write DBI, and data masking respectively. Software must ensure that these values correspond to the values written to the mode register.

If read DBI is enabled, the JEDEC Specification indicates that the CAS latency (CL) for DDR4 should be increased. Therefore, the following memory controller registers must be adjusted accordingly as they depend on CL or RL:

- DDR_UMCTL2::DFITMG0.DFI_T_RDDATA_EN
- DDR_UMCTL2::DRAMTMG2.READ_LATENCY
- DDR_UMCTL2::DRAMTMG2.RD2WR

The required sequence for enabling or disabling DBI is as follows:

1. Set DDR_UMCTL2::DBG1.DIS_DQ = 1.
2. Poll DDR_UMCTL2::DBGCAM.WR_DATA_PIPELINE_EMPTY and DDR_UMCTL2::DBGCAM.RD_DATA_PIPELINE_EMPTY, to ensure all outstanding commands have been sent on the DFI.
3. Set DDR_UMCTL2::DBICTL.RD_DBI_EN, DDR_UMCTL2::DBICTL.WR_DBI_EN, DDR_UMCTL2::DBICTL.DM_EN as required.
4. Set the DRAM mode register to match the above settings.
5. Adjust CL in MR0 for DDR4.
6. Adjust the registers depending on CL (or RL) and AL (listed above) as appropriate.
7. Set DDR_UMCTL2::DBG1.DIS_DQ = 0.

5.8.7.2 Per DRAM Addressability (PDA)

DDR4 allows programmability of a given device on a rank. As an example, this feature can be used to program different ODT or Vref values on DRAM devices on a given rank. The controller supports this feature. Accessing devices using PDA mode requires the controller to perform a specific sequence of MRS commands to the memory.

To change a memory device setting using PDA mode, complete the following steps.

1. Disable write DBI (if it has been already enabled) using the sequence mentioned in section 5.8.7.1 Data Bus Inversion (DBI). The controller does not support write DBI and PDA mode. Thus, the user must ensure that write DBI mode has been disabled before initiating the PDA sequence.
2. Select the rank of the memory device to be accessed using the register DDR_UMCTL2::MRCTRL0.MR_RANK. It is only possible to access one rank at a time in PDA mode.
3. Select the devices to be accessed within that rank using the register DDR_UMCTL2::MRCTRL2.MR_DEVICE_SEL.
4. Write the DDR_UMCTL2::MRCTRL0.MR_TYPE, DDR_UMCTL2::MRCTRL0.MR_ADDR, and DDR_UMCTL2::MRCTRL1.MR_DATA to define the MRS transaction to be executed while in PDA mode.
5. Write the register DDR_UMCTL2::MRCTRL0.PDA_EN to 1. This bit enables the PDA sequence in the controller for the subsequent MRS transaction.
6. Write the DDR_UMCTL2::MRCTRL0.MR_WR to 1. The PDA sequence is now triggered. The controller performs the MRS to enter PDA mode, then performs the PDA access and finally performs the MRS to exit PDA mode.
7. Poll the register DDR_UMCTL2::MRSTAT.PDA_DONE and wait for DDR_UMCTL2::MRSTAT.PDA_DONE has been set to 1. The controller asserts DDR_UMCTL2::MRSTAT.PDA_DONE once the PDA mode exit is complete.
8. Once the sequence has been completed, you must set DDR_UMCTL2::MRCTRL0.PDA_EN to 0. This action sets DDR_UMCTL2::MRSTAT.PDA_DONE to 0.
9. Re-enable write DBI, if it had been enabled beforehand.

5.8.7.3 Write CRC

CRC feature provides real time error detection on DDR4 write data bus. CRC is supported for writes only and not reads. It is supported for all device configurations (x4, x8, and x16). CRC polynomial used by DDR4 DRAM and the controller is ATM-8 HEC, X8+X2+X+1. All the write bursts with CRC enabled are extended by two data beats on the DRAM bus. Write bursts on the DFI are extended by an equivalent amount. The additional data beats are used to transfer the CRC information. DRAM compares against the CRC checksum coming from the controller. If two check sums do not match, DRAM flags an error. The error comes in as a short pulse (6 to 10 clocks) on the ALERT_n signal. Back-to-back CRC errors can cause the error pulse to be long. In this case, MPR read is required to identify the cause of the error. In addition to making the ALERT_n go low, SDRAM also does the following:

- Sets CRC Error flag to 1 (MR5[3])
- Sets CRC Error status to 1 (Page 1, MPR3[7])

MR5[3] should be reset to 0 through MR write by software to clear the CRC error flag and status bits. There are two types of CRC errors:

- Type 1 with DM (Data Mask) disabled: In this case, the data captured by DRAM is written to DRAM even when error is flagged. The CRC computation does not delay the memory write and hence DRAM write integrity is not maintained.
- Type 2 with DM (Data Mask) enabled: In this case, the data captures by DRAM is NOT written to DRAM when error is flagged. CRC computation causes a delay in memory Write and hence integrity of the DRAM memory is maintained.

There is some ambiguity in the JEDEC Specification regarding the use of DM in CRC calculation. The register DDR_UMCTL2::CRCPARCTL1.CRC_INC_DM controls whether to include or exclude DM bits in the CRC calculation. This control is provided to give maximum flexibility to the user in case different DRAM vendors decide to implement their CRC differently.

If both CRC and DM are enabled (via DDR_UMCTL2::CRCPARCTL1.CRC_ENABLE and DDR_UMCTL2::DBICTL.DM_EN), certain timing and control registers need to be adjusted/configured accordingly:

- DDR_UMCTL2::DRAMTMG0.WR2PRE - use tWRc_CRC_DM instead of tWRc in the equation for this register

- DDR_UMCTL2::DRAMTMG2.WR2RD - use tWTRc_L_CRC_DM instead of tWTRc in the equation for this register
- DDR_UMCTL2::DRAMTMG9.WR2RD_S - use tWTRc_S_CRC _DM instead of tWTRc_S in the equation for this register
- Configure DDR_UMCTL2::ODTCFG.WR_ODT_HOLD register field with (5 + WR_PREAMBLE + 1)
- Enable write CRC function in controller by setting DDR_UMCTL2::CRCPARCTL1.CRC_ENABLE = 1
- Enable SDRAM write CRC feature by configuring DDR_UMCTL2::INIT4.EMR2[27] = 1 and DDR_PHY::MR2_DDR4.WRCRC = 1 while performing controller/PHY initialization.
- Similarly, update WR (Write Recovery) value for SDRAM by configuring DDR_UMCTL2::INIT3.MR[26:24] and DDR_PHY:MR0_DDR4.WR_11_9 register fields with tWRc_CRC_DM parameter value while performing controller/PHY initialization.

Note:
See Table 5-29 for parameter values.

5.8.7.4 Command/Address (CA) Parity

Command/address (CA) parity takes the CA parity signal (PAR) input carrying the parity bit for the generated address and commands signals, and matches it to the internally generated parity from the captured address and commands signals. CA parity is supported in the DLL enabled state only; if the DLL is disabled, CA parity is not supported.

CA parity is disabled or enabled via an MRS command. If CA parity is enabled by programming a non-zero value to CA parity latency in the MR, the DRAM will ensure that there is no parity error before executing commands. There is an additional delay required for executing the commands versus when parity is disabled. The delay is programmed in the MR when CA parity is enabled (parity latency) and applied to all commands which are registered by CS_n (rising edge of CK_t and falling CS_n). The command is held for the time of the parity latency (PL) before it is executed inside the device. ALERT_n will go active when the DRAM detects a CA parity error.

CA parity covers ACT_n, RAS_n/A16, CAS_n/A15, WE_n/A14, the address bus including bank address and bank group bits, and C[2:0] on 3DS devices; the control signals CKE, ODT, and CS_n are not covered. The DRAM treats any unused address pins internally as zeros; for example, if a common die has stacked pins but the device is used in a monolithic application, then the address pins used for stacking and not connected are treated internally as zeros.

The convention for parity is even parity; for example, valid parity is defined as an even number of ones across the inputs used for parity computation combined with the parity signal. In other words, the parity bit is chosen so that the total number of ones in the transmitted signal, including the parity bit, is even.

If CA parity feature is enabled (via DDR_UMCTL2::CRCPARCTL1.PARITY_ENABLE), certain timing and control registers need to be adjusted/configured accordingly:

- Configure DDR_UMCTL2::DRAMTMG1.T_XP register field with “Roundup((tXPc + tPLc) / 2)”
- Add “tPLc” parameter value to the equation used for calculating the configure value for DDR_UMCLT2::DRAMTMG9.WR2RD_S and DDR_UMCLT2::DRAMTMG2.WR2RD register fields
- DDR_UMCTL2::DRAMTMG3.T_MOD – use tMODc_PAR instead of tMODc in the equation for this register
- DDR_UMCTL2::DRAMTMG3.T_MRD – use tMRDc_PAR instead of tMRDc in the equation for this register
- DDR_PHY::DTPR1.TMRD - use tMRDc_PAR instead of tMRDc in the equation for this register
- DDR_PHY.DTPR1.TMOD – use tMODc_PAR instead of tMODc in the equation used for calculating the tMOD parameter value which is used to configure this register
- Enable CA Parity Latency mode by configuring DDR_UMCTL2::INIT6.MR5[2:0] and DDR_PHY::MR5_DDR4.CAPM register fields with tPLc parameter value while performing controller/PHY initialization.

The dfi_alert_n input indicates that a parity/CRC error is detected by the SDRAM/RDIMM/LRDIMM.
For each falling edge of dfi_alert_n, a counter DDR_UMCTL2::CRCPARSTAT.DFI_ALERT_ERR_CNT is incremented. This counter can be cleared by writing a '1' to the self-clearing register DDR_UMCTL2::CRCPARCTL0.DFI_ALERT_ERR_CNT_CLR.

An interrupt status register (DDR_UMCTL2::CRCPARSTAT.DFI_ALERT_ERR_INT) and an interrupt signal "dfi_alert_err_intr" are also provided. This interrupt status register is set to 1 whenever dfi_alert_n goes low.
If enabled (by using DDR_UMCTL2::CRCPARCTL0.DFI_ALERT_ERR_INT_EN = 1) this interrupt signal is also asserted when dfi_alert_n goes Low. These can be cleared by writing a '1' to the self-clearing register DDR_UMCTL2::CRCPARCTL0.DFI_ALERT_ERR_INT_CLR.

5.8.8 ECC support

The term “ECC lane” is defined as a word of data, of width equal to the SDRAM width, on which the ECC calculations are performed. The controller supports Single Bit Single Error Correction / Double Error Detection Error Correction code (SEC/DED ECC) for configurations where the SDRAM data width is configured to be 16 or 32 bits.

If DDR_UMCTL2::ECCCFG0.ecc_mode = “100”, single-bit SEC/DED ECC is enabled. In this mode, the controller performs the following functions.

- On writes, the ECC is calculated across each ECC lane, and the resulting ECC code is written as an additional byte along with the data in the ECC lane. This additional ECC byte is always written to the uppermost byte of SDRAM (byte 2 for 16-bit SDRAM, byte 4 for 32-bit SDRAM, or byte 8 for 64-bit SDRAM).
- On reads, the ECC lane including the ECC byte is read from SDRAM. This is then “decoded”. A check is performed to verify that the ECC byte is as expected, based on the data in the ECC lane. If it is correct, the data is sent to the SoC core as normal.
- On read-modify-write (RMW) operations, a read is first performed, as described above. The read data is then combined with the write data received from the HIF, making use of the write mask received from the HIF to over-write certain bytes of the data that is read. The ECC is then calculated on the resulting word (per ECC lane), and the write is performed as described above.

5.8.8.1 Controller Behavior during ECC errors

When the controller detects a correctable ECC error, it performs the following:

1. Sends the corrected data to the SoC core as part of the read data
2. Sends the ECC error information to the APB register module
3. Performs a RMW operation to correct the data present in SDRAM (only if ECC scrubbing is enabled – DDR_UMCTL2::ECCCFG0.DIS_SCRUB=0). This RMW operation is invisible to the core.

When the controller detects an uncorrectable error, it does the following:

1. Sends the data with error to the SoC core as part of the read data
2. Sends the ECC error information to the APB register module
3. Generates an error response SLVERR on the AXI interface

In addition to the above, when the controller detects an uncorrectable error during the read part of a normal RMW command (not an ECC scrub), it does the following:

1. During the subsequent write part of the normal RMW command, it performs a 2-bit corruption to the recalculated ECC word
2. This corruption ensures that there is no accidental ECC auto-correction, of any data with uncorrectable errors

5.8.8.2 ECC Error Reporting Registers

A wide range of information about the detected errors can be obtained by reading the ECC error reporting registers.

- CPU::DDRC_INTR_RAW_STAT.ECC_CORRECTED_ERR_INTR_RAW_STAT
- CPU::DDRC_INTR_RAW_STAT.ECC_UNCORRECTED_ERR_INTR_RAW_STAT
- CPU::DDRC_INTR_MASKED_STAT. ECC_CORRECTED_ERR_INTR_STAT
- CPU::DDRC_INTR_MASKED_STAT. ECC_UNCORRECTED_ERR_INTR_STAT

There are also two interrupts, ecc_corrected_err_intr and ecc_uncorrected_err_intr, which are asserted when corrected or uncorrected errors are detected.

5.9 DDR4 multiPHY PHY Utility Block (PUB)

This section describes the feature set supported by the DDR4 multiPHY PUB and basic overview about the PUB.

5.9.1 Features

DWC DDR4 multiPHY PUB supports the following features.

- DDR4, DDR3 operation
  - Compatible with JEDEC standard DDR4 up to 1250 Mbps

  - Compatible with JEDEC standard DDR3 up to 2000 Mbps
- Maximum controller clock frequency of 500 MHz resulting in maximum SDRAM data rate of 2000 Mbps
- Support for DDR4/3 Shared-AC mode (allows two independent data channels to share a single address/ command lane)
- Complete PHY initialization, training and control
- Automatic DQS gate training
- Automatic background Delay line calibration and VT compensation
- Automatic write levelling
- Automatic read and write data bit deskew
- Write data mask, Write DBI and Read DBI
- Automatic DQ/DQS eye training
- At-speed built-in-self-test (BIST) loopback testing on both the address and data channels.
- PHY control and configuration registers
  - APB interface to configuration registers
  - Additional JTAG interface to configuration registers
- Compatible with DFI 3.1 interface
  - Includes support for DFI low-power interface
  - DFI interface may be clocked in 1:1 or 1:2 frequency ratio modes

5.9.2 Overview

The PHY Utility Block (PUB) provides control features to ease the customer implementation of digitally controlled DDR PHY features such as initialization, read DQS training, data eye training, delay line calibration and VT compensation, write leveling, output impedance calibration, and programmable configuration controls. The PUB has built-in self-test features to provide support for production testing of the PHY. It also provides a DFI interface to the PHY. The PUB performs, in sequence, various tasks required by the PHY before it can commence normal DDR operations.

Access to the PUB internal control features and registers is through a dedicated APB interface. A JTAG interface is also provided as an additional second configuration port to co-exist with the APB main configuration port.

The PUB is driven off two clocks, the controller clock (ctl_clk) and the configuration clock (pclk). The controller clock is the same clock driving the memory controller and is half the frequency of the SDRAM clock (ck). The configuration clock can run at a frequency equal to or less than the controller clock. The configuration clock is used only for external configuration register write/read protocol. All internal PUB registers, including configuration registers, are on the high-speed clock.

5.9.3 DDR4 multiPHY PUB JTAG Interface

In addition to the APB interface, all DDR4 PHY registers are also accessed using a JTAG interface. This interface is useful to access or trigger PHY test and/or debug modes without requiring full functional vectors. A JTAG interface co-exists with the APB interface. For more information on the device's JTAG interface see 5.12 JTAG Interface.

The PUB JTAG interface implements a limited set of the IEEE 1149.1 protocol using a TAP controller that resides in the user chip logic. Chip logic connects to the PUB JTAG interface using trst_n, tclk, tdi, tdo, tap_shift_dr, tap_capture_dr, and tap_update_dr pins, where tap_* signals are derived from the TAP controller states.

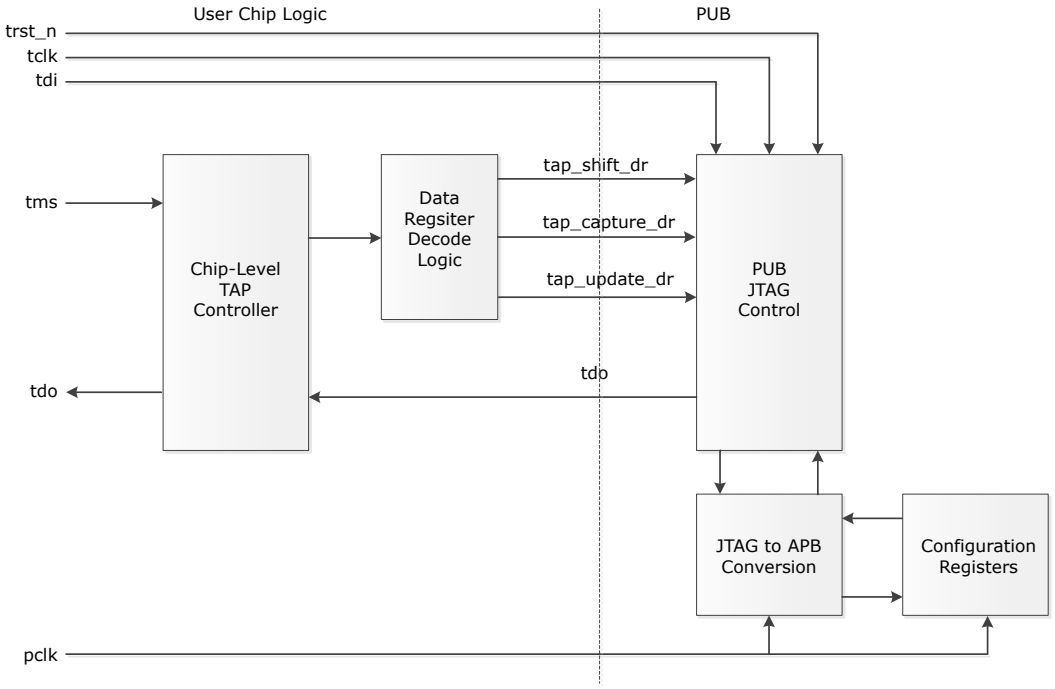
The following figure shows how the PUB JTAG interface is connected to the TAP controller. The PUB JTAG interface is simply viewed as a Design-Specific Test Data Register to the TAP controller. The PUB JTAG interface does not provide boundary scan control to the PHY SSTL I/Os. It is used only to access PUB configuration registers.

Glue logic is implemented to interface the TAP controller to the PUB JTAG interface. This is shown as the data register decode logic in . The glue logic generates the signals tap_shift_dr, tap_cature_dr, and tap_update_dr. These signals are derived from the correspondingly names states of the TAP controller.

The JTAG interface is designed to work with an existing primary configuration interface, i.e APB interface. Therefore, valid signals and a running clock must exist on the primary interface for the JTAG interface to work. This is illustrated in and is necessitated by the fact that all the configuration registers physically exist in the clock domain of the primary interface. To run the JTAG interface without an active primary interface, such as on an ATE, connect the clock and

reset of the primary interface to the JTAG clock and JTAG reset, respectively. In this case, all the other inputs of the primary interface must be driven to their inactive states.

Figure 5-14. JTAG Connectivity



A (2 + g + 32)-bit data register is used to create read/write transactions on the configuration port; g is the width of the configuration port address. The JTAG instruction register is not supported. This also means that no JTAG instructions are supported.

Table 5-41. TAP Data Register

Bits	Name	Description
[1:0]	INSTR	Instruction: Select the instruction to be performed on the configuration port. Valid Values are: 00 = Reserved. No action on the configuration port 01 = Load configuration register address and execute a register read operation 10 = Load configuration register address and data, and execute a register write operation 11 = Reserved. No action on the configuration port
[g+1 : 2] [1]	ADDR	Address: Address for the register to be read or written
[33+g : g+2]	DATA	Data: Data to be written to a register or read from a register

Note:

1. g = Width of the configuration port address. Valid value is 10.

To execute a register write, the register address and data for the write operation is shifted into the TAP data register together with the write instruction (INSTR = 2'b10). Once this information has been shifted in, the actual write to the register will complete a maximum of five JTAG test clock (tclk) cycles later.

To execute a read, the register address and read instruction is first shifted into the TAP data register. The actual register read completes a maximum of eight JTAG test clock cycles later after which the read data is ready to be shifted out of the TAP data register.

5.10 VCore System Peripherals

This section describes the subblocks of the VCore system. They are primarily intended to be used by the VCore CPU. However, an external CPU can also access and control these through the shared bus.

5.10.1 SI Boot Controller

This section provides information about the SPI boot master that allows booting from a Flash hooked up to the serial interface. For information about how to write to the serial interface, see 5.10.2 SI Master Controller. For information about using an external CPU to access device registers using the serial interface, see 5.5.2 Serial Interface in Slave Mode.

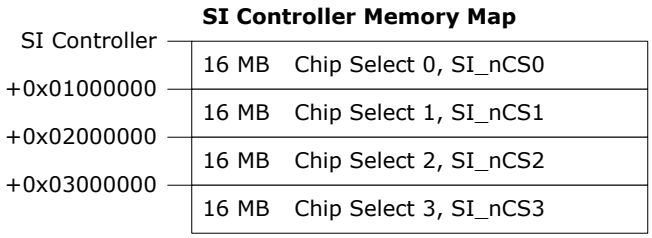
The following table lists the registers associated with the SI boot controller.

Table 5-42. SI Boot Controller Configuration Registers

Register	Description
ICPU_CFG::SPI_MST_CFG	Serial interface speed and address width
ICPU_CFG::SW_MODE	Legacy manual control of the serial interface pins
ICPU_CFG::GENERAL_CTRL	SI interface ownership

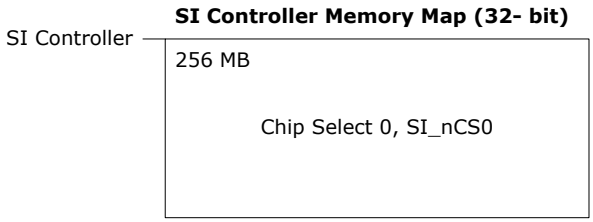
By default the SI Boot Controller is operating in 24bit address mode. In this mode there are four programmable chip selects when the VCore system controls the SI. Through individually mapped memory regions, each chip select can address up to 16 megabytes (MB) of memory.

Figure 5-15. SI Boot Controller Memory Map in 24 bit Mode



The SI Boot Controller can be reconfigured for 32bit address mode via SPI_MST_CFG.A32B_ENA. In 32bit mode the entire SI region of 256 megabytes (MB) is addressed via chip select 0.

Figure 5-16. SI Boot Controller Memory Map in 32 bit Mode



Reading from the memory region for a specific SI chip select generates SI read on that chip select. The VCore CPU can execute code directly from Flash by executing from the SI boot controller's memory region. For 32 bit capable SI FLASH devices the VCore must start up in 24 bit mode, during the boot-process it must then manually re-configure the FLASH device (and SI Boot Controller) into 32 bit mode then continue booting.

The SI boot controller accepts 8-bit, 16-bit, and 32-bit read access with or without bursting. Writing to the SI requires manual control of the SI pins using software. Setting SW_MODE.SW_PIN_CTRL_MODE places all SI pins under software control. Output enable and the value of SI_Clk, SI_DO, SI_nEn[3:0] are controlled through the SW_MODE register. The value of the SI_DI pin is available in SW_MODE.SW_SPI_SDI. The software control mode is provided for legacy reasons, new implementations should use the dedicated master controller writing to the SI interface, for more information, see 5.10.2 SI Master Controller.

Note:
The VCore CPU cannot execute code directly from the SI boot controller's memory region at the same time as manually writing to the serial interface.

The following table lists the serial interface pins when the SI Boot Controller is configured as owner of SI interface via GENERAL_CTRL.IF_SI_OWNER.

Table 5-43. Serial Interface Pins

Pin Name	I/O	Description
SI_nCS0 SI_nCS[3:1]/GPIO	O	Active low chip selects. Only one chip select can be active at any time. Chip selects 1 through 3 are overlaid functions on the GPIOs. For more information, see 5.10.8.1 GPIO Overlaid Functions.
SI_Clk	O	Clock output.
SI_DO	O	Data output (MOSI).
SI_DI	I	Data input (MISO).

The SI boot controller does speculative prefetching of data. After reading address n, the SI boot controller automatically continues reading address n + 1, so that the next value is ready if requested by the VCore CPU. This greatly optimizes reading from sequential addresses in the Flash, such as when copying data from Flash into program memory.

Figure 5-17. SI Read Timing in Normal Mode

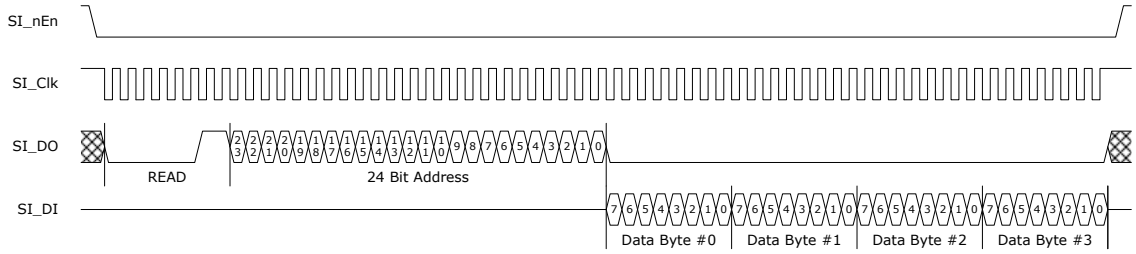
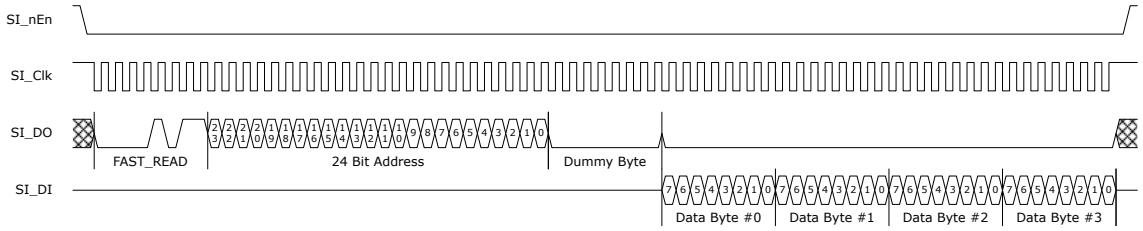


Figure 5-18. SI Read Timing in Fast Mode



The figures depicts 24 bit address mode, when controller is set to 32 bit mode (via SPI_MST_CFG.A32B_ENA) 32 address bits are transferred instead of 24.

The default timing of the SI boot controller operates with most serial interface Flash devices. Use the following process to calculate the optimized SI parameters for a specific SI device.

1. Calculate an appropriate frequency divider value as described in SPI_MST_CFG.CLK_DIV. The SI operates at no more than 25 MHz, and the maximum frequency of the SPI device must not be exceeded. The VCore system frequency in the device is 250 MHz.
2. The SPI device may require a FAST_READ command rather than normal READ when the SI frequency is increased. Setting SPI_MST_CFG.FAST_READ_ENA makes the SI boot controller use FAST_READ commands.
3. Calculate SPI_MST_CFG.CS_DESELECT_TIME so that it matches how long the SPI device requires chip-select to be deasserted between accesses. This value depends on the SI clock period that results from the SPI_MST_CFG.CLK_DIV setting.

These parameters must be written to SPI_MST_CFG. The CLK_DIV field must either be written last or at the same time as the other parameters. The SPI_MST_CFG register can be configured while also booting up from the SI.

When the VCore CPU boots from the SI interface, the default values of the SPI_MST_CFG register are used until the SI_MST_CFG is reconfigured with optimized parameters. This implies that SI_Clk is operating at approximately 8.1 MHz, with normal read instructions, and maximum gap between chip select operations to the Flash.

Note The SPI boot master does optimized reading. SI_DI (from the Flash) is sampled just before driving falling edge on SI_CLK (to the Flash). This greatly relaxes the round trip delay requirement for SI_CLK to SI_DI, allowing high Flash clock frequencies.

5.10.2 SI Master Controller

This section describes the SPI Master Controller (SIMC) and how to use it for accessing external SPI slave devices, such as programming of a serially attached FLASH device on the boot interface. VCore booting from serial FLASH is handled by the SI Boot Master.

The following table lists the registers associated with the SI Master Controller.

Table 5-44. SI Master Controller Configuration Registers

Register	Description
SIMC::CTRLR0	Transaction Configuration
SIMC::CTRLR1	Configurations for Receive-only Mode
SIMC::SIMCEN	SI Master Controller Enable
SIMC::SER	Slave Select Configuration
SIMC::BAUDR	Baud Rate Configuration
SIMC::TXFTLR	TX FIFO Threshold Level
SIMC::RXFTLR	RX FIFO Threshold Level
SIMC::TXFLR	TX FIFO Fill Level
SIMC::RXFLR	RX FIFO Fill Level
SIMC::SR	Various Status Indications
SIMC::IMR	Interrupt Enable
SIMC::ISR	Interrupt Sources
SIMC::RISR	Unmasked Interrupt Sources
SIMC::TXOICR	Clear of Transmit FIFO Overflow Interrupt
SIMC::RXOICR	Clear of Receive FIFO Overflow Interrupt
SIMC::RXUICR	Clear of Receive FIFO Underflow Interrupt
SIMC::DR	TX/RX FIFO Access
CPU::GENERAL_CTRL	Interface configurations.

The SI Master Controller supports Motorola SPI and Texas Instruments SSP protocols. The default protocol is SPI, enable SSP by setting CTRLR0.FRF = 1 and GENERAL_CTRL.SSP_MODE_ENA = 1. The protocol baud rate is programmable via BAUDR.SCKDV; the maximum baud rate is 25 MHz.

Before the SI Master Controller can be used, it must be set as owner of the SI interface. This is done by writing GENERAL_CTRL.IF_SI_OWNER = 2. Alternatively, the SI Slave may be configured to own the SPI2 interface (which is mapped on GPIOs) via GENERAL_CTRL.IF_SI2_OWNER. For more information, see 5.10.8.1 GPIO Overlaid Functions.

The SI Master Controller has a programmable frame size; the frame size is the smallest unit when transferring data over the SI interface. Via CTRLR0.DFS the frame size is configured in the range 4 to 16 bits. When reading or writing words from the transmit/receive FIFO, the number of bits that is stored per FIFO-word is always equal to frame size (as programmed in CTRLR0.DFS).

The controller operates in one of three major modes (the mode is configured in CTRLR0.TMOD):

- Transmit and receive:

  Software paces SI transactions. For every data frame that software writes to the transmission FIFO another data frame is made available in the receive FIFO (receive data from the serial interface). Transaction will go on for as long as there is data available in the transmission FIFO.
- Transmit only:

Software paces SI transactions. The controller only transmit data frames, receive data is discarded. Transaction will go on for as long as there is data available in the transmission FIFO.

- Receive only:

  The controller paces SI transactions. The controller only receive data, software requests a specific number of data frames via CTRLR1.NDF, the transaction will go on until all data frames has been read from the SI interface. Transaction is initiated by software writing one data frame to transmission FIFO, this frame is not used for anything else than starting the transaction. The SI_DO output is undefined during “Receive Only” transfers. Receive data frames is put into the receive FIFO as they are read from SI.

For SPI mode, chip-select will be asserted throughout transfers. Transmit transfers end when the transmit FIFO runs out of data frames. This implies that software must make sure it is not interrupted while writing data for a multi-frame transfer. When software wants multiple distinct transfers, then wait for SR.TFE=1 and SR.BUSY=0 before initiating new transfer. SR.BUSY is an indication of ongoing transfers, however it is not asserted immediately when writing frames to the FIFO, therefore software must also check the SR.TFE (transmit FIFO empty) indication.

The SI Master Controller support up to 16 chip-selects. Chip-select 0 is mapped to the SI_nEN pin of the SI interface, the rest are available via overlaid GPIO functions. Software controls which chip-select(s) to activate for a transfer via the SER register. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Table 5-45. SI Master Controller Pins

Pin Name	I/O	Description
SI_nCS0 SPI_nCS[15:0]/GPIO	O	Active low chip selects. Chip selects 1 through 15 are overlaid functions on the GPIOs. Chip select 0 is both on a dedicated pin and a GPIO. For more information, see 5.10.8.1 GPIO Overlaid Functions.
SI_Clk	O	Clock output.
SI_DO	O	Data output (MOSI).
SI_DI	I	Data input (MISO).

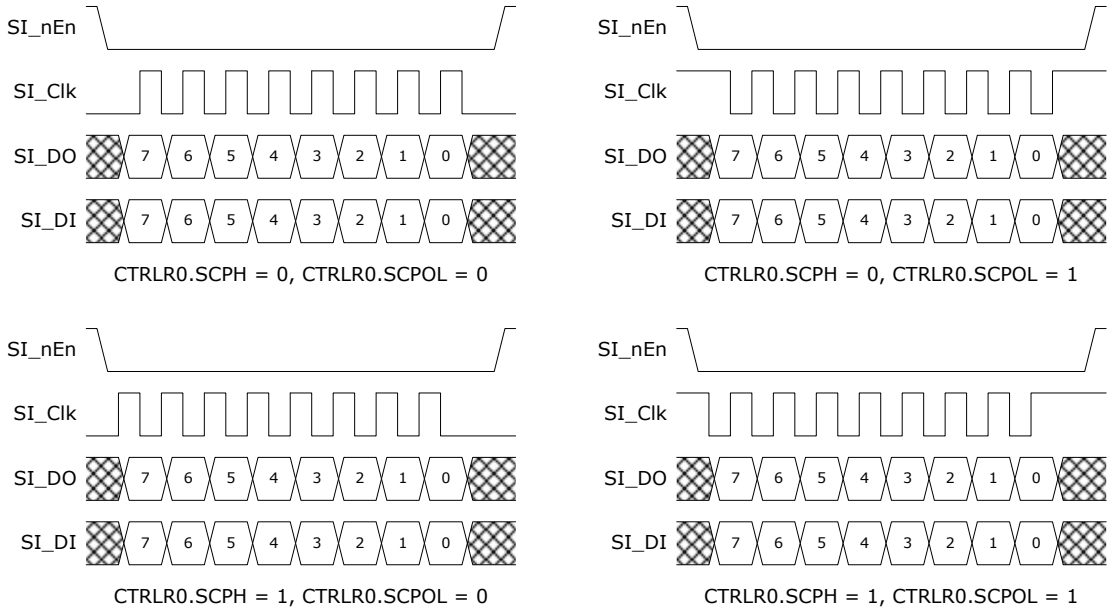
Writing to any DR replication index put data into the transmission FIFO, reading from any DR replication index take out data from the receive FIFO. FIFO status indications are available via SR register’s TFE, TFNF, RFF, and RFNE fields. It is possible to interrupt on FIFO status. For more information, see 5.10.2.3 SIMC Interrupts.

After doing all initial configurations, the SI Master Controller is enabled by writing SIMCEN.SIMCEN = 1. Some registers can only be written when controller is in disabled state; this is noted in the register-descriptions for the affected registers.

5.10.2.1 SPI Protocol

When the SI Master Controller is configured for SPI mode then clock polarity and position is configurable via CTRLR0.SCPH and CTRLR0.SCPOL, the following figure shows the 4 possible combinations for 8 bit transfers.

Figure 5-19. SIMC SPI Clock Configurations

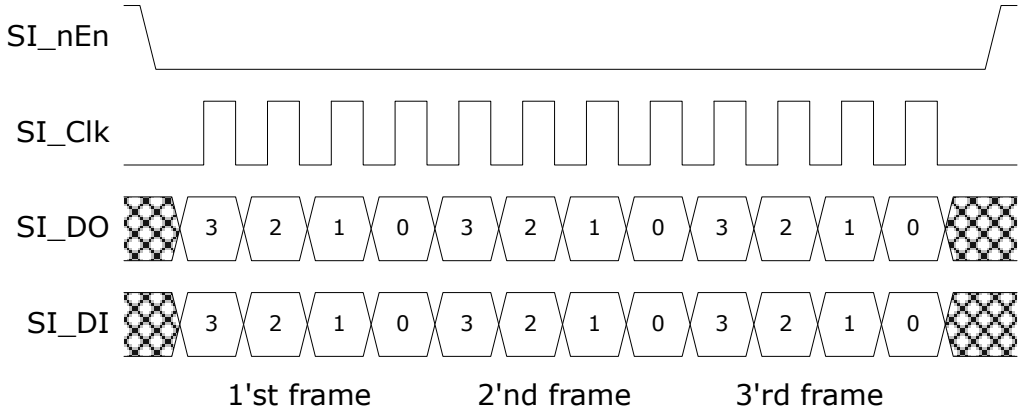


Data is sampled in the middle of the data-eye.

When using SPI protocol and transferring more than one data frame at the time, the controller performs one consecutive transfer. The following figure shows a transfer of three data frames of 4 bits each.

Note:
Transmitting transfers end when the transmit FIFO runs out of data frames. Receive only transfers end when the pre-defined number of data-frames is read from the SI interface.

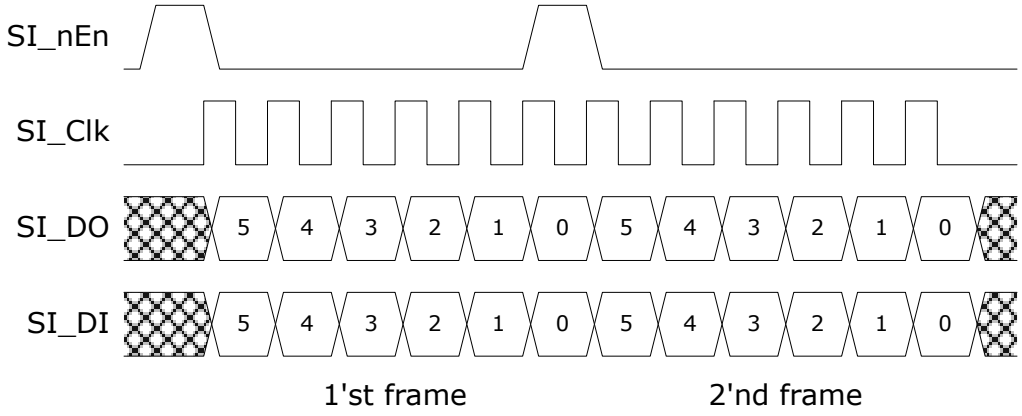
Figure 5-20. SIMC SPI 3x Transfers



5.10.2.2 SSP Protocol

The SSP protocol for transferring two 6-bit data frames is shown in the following illustration. When using SSP mode, CTRLR0.SCPH and CTRLR0.SCPOL must remain zero.

Figure 5-21. SIMC SSP 2x Transfers



5.10.2.3 SIMC Interrupts

The SI master controller has 5 interrupt sources.

- RXF interrupt is asserted when the fill level of the receive FIFO (available via RXFLR) exceeds the programmable level set in RXFTLR. This interrupt is non-sticky, it is deasserted when fill level is decreased below the programmable level.
- RXO interrupt is asserted on receive FIFO overflow. This interrupt is cleared by reading the RXOICR register.
- RXU interrupt is asserted when reading from empty receive FIFO. This interrupt is cleared by reading the RXUICR register.
- TXO interrupt is asserted when writing to a full transmit FIFO. This interrupt is cleared by reading the TXOICR register.
- TXE interrupt is asserted when the fill level if the transmit FIFO (available via TXFLR) is below the programmable level set in TXFTLR. This interrupt is non-sticky, it is deasserted when fill level is increased above the programmable level.

The raw (unmasked) status of these interrupts is available via the RISR register. Each of the interrupts can be individually masked via IMR register; by default all interrupts are enabled. The currently active interrupts is shown in the ISR register.

If the RXO, RXU, and TXO interrupts occur during normal use of the SI master controller, it is an indication of software problems.

Interrupt is asserted towards the VCore Interrupt Controller when any enabled interrupt is asserted and the SI master controller is enabled.

5.10.2.4 SIMC Programming Example

This example shows how to use the SI Master Controller to interface with the SI slave of another device. The slave device will be initialized for big endian / most significant bit first mode and then the DEVCPU_GCB::GPR register is written with the value 0x01234567. It is assumed that the other device's SI slave is connected on SI_nCS1 (and that appropriate GPIO alternate function has been enabled).

The SI slave is using a 24-bit address field and a 32-bit data field. This example uses 4 x 16 bit data frames to transfer the write-access. This means that 8 bit too much will be transferred, this is not a problem for the SI slave interface; it ignores data above and beyond the 56 bit required for a write-access.

For information about initialization and how to calculate the 24 bit SI address field, see 5.5.2 Serial Interface in Slave Mode. The address-field when writing DEVCPU_GCB::GPR is 0xC04001 (including write-indication).

The following steps are required to bring up the SI Master Controller and write twice to the SI slave device.

1. Write GENERAL_CTRL.IF_SI_OWNER = 2 to make SI Master Controller the owner of the SI interface.
2. Write BAUDR = 250 to get 1 MHz baud rate.
3. Write SIR = 2 to use SI_nEn1.
4. Write CTRLR0 = 0x10F for 16 bit data frame and "Transmit Only".
5. Write SIMCEN = 1 to enable the SI Master Controller.

6. Write DR[0] = 0xC040, 0x6A81, 0x8181, 0x8100 to configure SI Slave for big endian / most significant bit first mode.
7. Wait for SR.TFE = 1 and SR.BUSY = 0 because chip select must be deasserted between accesses to the SI slave.
8. Write DR[0] = 0xC040, 0x0101, 0x2345, 0x6700 to write DEVCPU_GCB::GPR in slave device.
9. Wait for SR.TFE = 1 and SR.BUSY = 0 then disable the SI Master Controller by writing SIMCEN = 0.

Note:
The above procedure disconnects the SI Boot Master from the SI interface, booting must be done before attempting to overtake the boot-interface!.

When reading from the SI slave CTRLR0.TMOD must be configured for transmit and receive. One byte padding is appropriate for a 1 MHz baud rate.

5.10.3 SI Slave-Select as a Shared Resource

The device implements 16 chip select IOs which are mapped on GPIOs. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Table 5-46. Chip Select Pins

Pin Name	I/O	Description
SI_nCS0 SPI_nCS[15:0]/GPIO	I/O	Active low chip selects. Chip selects 1 through 15 are overlaid functions on the GPIOs. Chip select 0 is both on a dedicated pin and a GPIO. For more information, see 5.10.8.1 GPIO Overlaid Functions.

The chip select pins of the device are shared among the 5.5.2 Serial Interface in Slave Mode, the 5.10.1 SI Boot Controller, and the 5.10.2 SI Master Controller. It is the programmer's responsibility to map the chip select to the various drivers. The chip select driving value may be overridden by software through CPU::SS_FORCE and CPU::SS_FORCE_ENA. For more information on chip select usage, read the detailed descriptions of the registers listed in the following table.

Table 5-47. Slave-Select Registers

Register	Description
DEVCPU_GCB::SI_SLV_SS	Slave-select mask for 5.5.2 Serial Interface in Slave Mode
DEVCPU_GCB::SPI_MASTER_SS0_MASK	Slave-select mask for 5.10.1 SI Boot Controller CS0
DEVCPU_GCB::SPI_MASTER_SS1_MASK	Slave-select mask for 5.10.1 SI Boot Controller CS1
DEVCPU_GCB::SPI_MASTER_SS2_MASK	Slave-select mask for 5.10.1 SI Boot Controller CS2
DEVCPU_GCB::SPI_MASTER_SS3_MASK	Slave-select mask for 5.10.1 SI Boot Controller CS3
SIMC::SER	Slave-select mask for 5.10.2 SI Master Controller
CPU::SS_FORCE	Soft-force slave select
CPU::SS_FORCE_ENA	Enable soft-force slave select

Note:
When the VCore sub-CPU system is enabled, the GPIO mapping of SPI_nCS[8:5] may be overtaken by the VCore sub-CPU system.

5.10.4 Timers

This section provides information about the timers. The following table lists the registers associated with timers.

Table 5-48. Timer Registers

Register	Description
TIMERS::TIMER1LOADCOUNT	Value to be loaded to timer 1
TIMERS::TIMER1CURRENTVAL	Current value of timer 1 – read only
TIMERS::TIMER1CONTROLREG	Control register of timer 1
TIMERS::TIMER1EOI	Clear interrupt for timer 1
TIMERS::TIMER1INTSTAT	Interrupt status for timer 1
TIMERS::TIMER2LOADCOUNT	Value to be loaded to timer 2
TIMERS::TIMER2CURRENTVAL	Current value of timer 2 – read only
TIMERS::TIMER2CONTROLREG	Control register of timer 2
TIMERS::TIMER2EOI	Clear interrupt for timer 2
TIMERS::TIMER2INTSTAT	Interrupt status for timer 2
TIMERS::TIMER3LOADCOUNT	Value to be loaded to timer 3
TIMERS::TIMER3CURRENTVAL	Current value of timer 3 – read only
TIMERS::TIMER3CONTROLREG	Control register of timer 3
TIMERS::TIMER3EOI	Clear interrupt for timer 3
TIMERS::TIMER3INTSTAT	Interrupt status for timer 3
TIMERS::TIMERSINTSTATUS	Interrupt status of all timers
TIMERS::TIMERSEOI	Clear all timer interrupts
TIMERS::TIMERSRAWINTSTATUS	Unmasked interrupt status of all timers
TIMERS::TIMERS_COMP_VERSION	Read-only component version

There are three decrementing 32-bit timers in the VCore system that run on the 250MHz clock. Software can access each timer value through the TIMER1CURRENTVAL, TIMER2CURRENTVAL and TIMER3CURRENTVAL registers. These registers are read-only.

When timer 1 is enabled through field TIMER1CONTROLREG_TIMER_ENABLE, it decrements from TIMER1LOADCOUNT until it reaches zero. Depending on the mode configured in TIMER1CONTROLREG_TIMER_MODE, the counter wraps to TIMER1LOADCOUNT in user-defined mode or to its max value in free-running mode. If the timer 1 interrupt is unmasked through TIMER1CONTROLREG_TIMER_INTERRUPT_MASK, then the timer generates an interrupt every time it crosses through value 0. The interrupt status of this timer can be read at TIMER1INTSTAT. The timer interrupt is cleared by reading TIMER1EOI. Similar configurations and functionality applies for timers 2 and 3.

The unmasked interrupt status of all timers can be read at TIMERSRAWINTSTATUS, while the masked interrupt status for all timers is available at TIMERSINTSTATUS. Reading TIMERSEOI clears all active timer interrupts.

5.10.5 UARTs

This section provides information about the UART (Universal Asynchronous Receiver/Transmitter) controllers. There are two independent UART controller instances in the VCore System: UART and UART2. These instances are identical copies and anywhere in this description the word UART can be replaced by UART2.

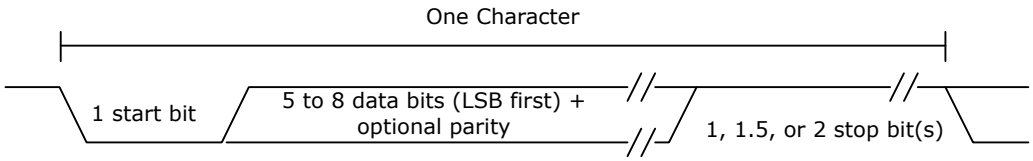
The following table lists the registers associated with the UART.

Table 5-49. UART Registers

Register	Description
UART::RBR_THR	Receive buffer/transmit buffer/Divisor (low).
UART::IER	Interrupt enable/Divisor (high).
UART::IIR_FCR	Interrupt identification/FIFO control.
UART::LCR	Line control.
UART::MCR	Modem control.
UART::LSR	Line status.
UART::MSR	Modem status.
UART::SCR	Scratchpad.
UART::USR	UART status.

The VCore system UART is functionally based on the industry-standard 16550 UART (RS232 protocol). This implementation features a 16-byte receive and a 16-byte transmit FIFO.

Figure 5-22. UART Timing



The number of data-bits, parity, parity-polarity, and stop-bit length are all programmable using LCR.

The UART pins on the device are overlaid functions on the GPIO interface. Before enabling the UART, the VCore CPU must enable overlaid modes for the appropriate GPIO pins. For more information, see 5.10.8.1 GPIO Overlaid Functions.

The following table lists the pins of the UART interfaces.

Table 5-50. UART Interface Pins

Pin Name	I/O	Description
UART_RX/GPIO	I	UART receive data
UART_TX/GPIO	O	UART transmit data
UART2_RX/GPIO	I	UART2 receive data
UART2_TX/GPIO	O	UART2 transmit data

The baud rate of the UART is derived from the VCore system frequency. The divider value is indirectly set through the RBR_THR and IER registers. The baud rate is equal to the VCore system clock frequency, divided by 16, and multiplied by the value of the baud rate divisor. A divider of zero disables the baud rate generator and no serial communications occur. The default value for the divisor register is zero.

Example: Configure a baud rate of 9600 in a 250 MHz VCore System. To generate a baud rate of 9600, the divisor register must be set to 0x65C (250 MHz/(16 × 9600 Hz)). Set LCR.DLAB and write 0x5C to RBR_THR and 0x06 to IER (this assumes that the UART is not in use). Finally, clear LCR.DLAB to change the RBR_THR and IER registers back to the normal mode.

By default, the FIFO mode of the UART is disabled. Enabling the 16-byte receive and 16-byte transmit FIFOs (through IIR_FCR) is recommended.

Note:
Although the UART itself supports RTS and CTS, these signals are not available on the pins of the device.

5.10.5.1 UART Interrupt

The UART can generate interrupt whenever any of the following prioritized events are enabled (through IER).

- Receiver error
- Receiver data available
- Character timeout (in FIFO mode only)
- Transmit FIFO empty or at or below threshold (in programmable THRE interrupt mode)

When an interrupt occurs, the IIR_FCR register can be accessed to determine the source of the interrupt. Note that the IIR_FCR register has different purposes when reading or writing. When reading, the interrupt status is available in bits 0 through 3. For more information about interrupts and how to handle them, see the IIR_FCR register description.

Example: Enable Interrupt When Transmit FIFO is Below One-Quarter Full. To get this type of interrupt, the THRE interrupt must be used. First, configure TX FIFO interrupt level to one-quarter full by setting IIR_FCR.TET to 10; at the same time, ensure that the IIR_FCR.FIFOE field is also set. Set IER.PTIME to enable the THRE interrupt in the UART. In addition, the VCore interrupt controller must be configured for the CPU to be interrupted. For more information, see 5.10.13 Outbound Interrupt Controller.

5.10.6 Two-Wire Serial Interface

This section provides information about the TWI (two-wire serial interface) controllers. There are two independent TWI controller instances in the VCore System: TWI and TWI2. These instances are identical copies and anywhere in this description the word TWI can be replaced by TWI2.

The following table lists the registers associated with the two-wire serial interface.

Table 5-51. Two-Wire Serial Interface Registers

Register	Description
TWI::IC_CON	General configuration.
TWI::IC_TAR	Target address.
TWI::IC_SAR	Slave address.
TWI::IC_DATA_CMD	Receive/transmit buffer and command.
TWI::IC_SS_SCL_HCNT	Standard speed high time clock divider.
TWI::IC_SS_SCL_LCNT	Standard speed low time clock divider.
TWI::IC_FS_SCL_HCNT	Fast speed high time clock divider.
TWI::IC_FS_SCL_LCNT	Fast speed low time clock divider.
TWI::IC_INTR_STAT	Masked interrupt status.
TWI::IC_INTR_MASK	Interrupt mask register.
TWI::IC_RAW_INTR_STAT	Unmasked interrupt status.
TWI::IC_RX_TL	Receive FIFO threshold for RX_FULL interrupt.

.....continued	
Register	**Description**
TWI::IC_TX_TL	Transmit FIFO threshold for TX_EMPTY interrupt.
TWI::IC_CLR_*	Individual CLR_* registers are used for clearing specific interrupts. See register descriptions for corresponding interrupts.
TWI::IC_ENABLE	Control register.
TWI::IC_STATUS	Status register.
TWI::IC_TXFLR	Current transmit FIFO level.
TWI::IC_RXFLR	Current receive FIFO level.
TWI::IC_SDA_HOLD	SDA hold time length register.
TWI::IC_TX_ABRT_SOURCE	Arbitration sources.
TWI::IC_SDA_SETUP	Data delay clock divider.
TWI::IC_ACK_GEN_CALL	Acknowledge of general call.
TWI::IC_ENABLE_STATUS	General two-wire serial controller status.
TWI::IC_FS_SPKLEN	Spike suppression configuration.
CPU::TWI_CONFIG	Extra configuration of SDA hold-delay.
CPU::TWI_SPIKE_FILTER_CFG	Extra configuration of SDA spike filter.

The two-wire serial interface controller is compatible with the industry standard two-wire serial interface protocol. The controller supports standard speed up to 100 kbps and fast speed up to 400 kbps. Multiple bus masters, as well as both 7-bit and 10-bit addressing, are also supported.

By default, the two-wire serial interface controller operates as master only.

The two-wire serial interface pins on the device are overlaid functions on the GPIO interface. Before enabling the two-wire serial interface, the VCore CPU must enable overlaid functions for the appropriate GPIO pins. For more information, see 5.10.8.1 GPIO Overlaid Functions.

The following table lists the pins of the two-wire serial interface.

Table 5-52. Wire Serial Interface Pins

Pin Name	I/O	Description
TWI_SCL/GPIO	I/O	TWI clock, open-collector output.
TWI_SDA/GPIO	I/O	TWI data, open-collector output.
TWI_SCL_GATEn/GPIO	I/O	TWI multiplexed clocks (30 instances in total), open-collector outputs.
TWI_SCL_GATEn_AD/GPIO	O	TWI multiplexed clock outputs (2 instances in total)
TWI2_SCL/GPIO	I/O	TWI2 clock, open-collector output.
TWI2_SDA/GPIO	I/O	TWI2 data, open-collector output.

Setting IC_ENABLE.ENABLE enables the controller. The controller can be disabled by clearing the IC_ENABLE.ENABLE field, there is a chance that disabling is not allowed (at the time when it is attempted); the IC_ENABLE_STATUS register shows if the controller was successfully disabled.

The two-wire serial interface controller has an 8-byte combined receive and transmit FIFO.

5.10.6.1 Two-Wire Serial Interface Frequency Configuration

Before enabling the controller, the user must decide on either standard or fast mode (IC_CON.SPEED) and configure clock dividers for generating the correct timing (IC_SS_SCL_HCNT, IC_SS_SCL_LCNT, IC_FS_SCL_HCNT, IC_FS_SCL_LCNT). The configuration of the divider registers depends on the VCore system clock frequency along with some constraints imposed by timing requirements. The TWI controller is clocked with the 250/500 MHz clock of the VCore system.

The minimum timing requirements for TWI () specify that the low time of the SCL clock must be at least 1300 ns, while the high time of the SCL clock must be at least 600 ns. The total SCL period (low plus high time) must be 2500 ns in fast mode and 10000 ns in standard mode.

Table 5-53. TWI Minimum Timing Requirements

Parameter	Value
SS_SCL_LOW	>= 1300 (ns)
SS_SCL_HIGH	>= 600 (ns)
SS_SCL	10000 (ns)
SS_SCL	SS_SCL_LOW + SS_SCL_HIGH
FS_SCL_LOW	>= 1300 (ns)
FS_SCL_HIGH	>= 600 (ns)
FS_SCL	2500 (ns)
FS_SCL	FS_SCL_LOW + FS_SCL_HIGH

defines a range of applicable durations for FS_SCL_LOW, FS_SCL_HIGH, SS_SCL_LOW and SS_SCL_HIGH. The values for the configuration registers are computed as:

- IC_SS_SCL_LCNT = ceil(SS_SCL_LOW / VCore clock period) – 1
- IC_SS_SCL_HCNT = ceil(SS_SCL_HIGH / VCore clock period) – 8
- IC_FS_SCL_LCNT = ceil(FS_SCL_LOW / VCore clock period) – 1
- IC_FS_SCL_HCNT = ceil(FS_SCL_HIGH / VCore clock period) – 8

When IC_FS_SPKLEN is used for spike suppression, care must be taken to ensure that the following constraints are respected.

- IC_SS_SCL_LCNT > IC_FS_SPKLEN+7
- IC_FS_SCL_LCNT > IC_FS_SPKLEN+7
- IC_SS_SCL_HCNT > IC_FS_SPKLEN+5
- IC_FS_SCL_HCNT > IC_FS_SPKLEN+5

The following table shows how the configuration registers affect the timing parameters of the two-wire serial interface. Depending on the requirements, the configuration registers should be trimmed accordingly to meet timing constraints.

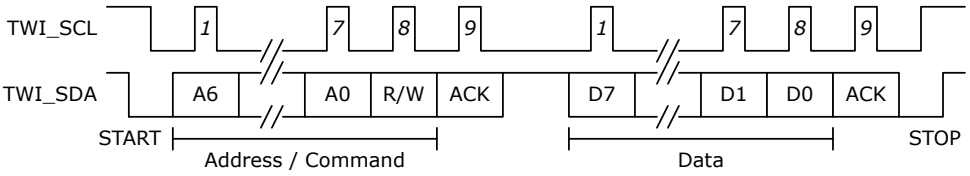
Table 5-54. Two-Wire Serial Interface Timing Parameters

Timing Parameter	Symbol	Standard Speed	Fast Speed
Setup time for a repeated START condition	tSU;STA	IC_SS_SCL_LCNT	IC_FS_SCL_HCNT
Hold time (repeated) START condition	tHD;STA	IC_SS_SCL_HCNT	IC_FS_SCL_HCNT

.....continued			
Timing Parameter	**Symbol**	**Standard Speed**	**Fast Speed**
Setup time for STOP condition	tSU;STO	IC_SS_SCL_HCNT	IC_FS_SCL_HCNT
Bus free time between a STOP and a START condition	tBUF	IC_SS_SCL_LCNT	IC_FS_SCL_LCNT
Spike length	tSP	IC_FS_SPKLEN	IC_FS_SLKLEN
Data hold time	tHD;DAT	IC_SDA_HOLD	IC_SDA_HOLD
Data setup time	tSU;DAT	IC_SDA_SETUP	IC_SDA_SETUP

Some two-wire serial devices require a hold time on SDA after SCK when transmitting from the two-wire serial interface controller. The device supports a configurable hold delay through the TWI::IC_SDA_HOLD and CPU::TWI_CONFIG registers.

Figure 5-23. Two-Wire Serial Interface Timing for 7-bit Address Access



During normal operation of the two-wire serial interface controller, the IC_STATUS register shows the activity and FIFO states.

5.10.6.2 Two-Wire Serial Interface Addressing

To configure either 7-bit or 10-bit addressing, use CFG.MASTER_10BITADDR.

The two-wire serial interface controller can generate both General Call and START Byte. Initiate this through IC_TAR.SPECIAL or TAR.GC_OR_START. The target/slave address is configured using the TAR register.

5.10.6.3 Two-Wire Serial Interface Interrupt

The two-wire serial interface controller can generate a multitude of interrupts. All of these are described in the RAW_INTR_STAT register. The IC_RAW_INTR_STAT register contains interrupt fields that are always set when their trigger conditions occur. The IC_INTR_MASK register is used for masking interrupts and allowing interrupts to propagate to the IC_INTR_STAT register. When set in the IC_INTR_STAT register, the two-wire serial interface controller asserts an interrupt towards the VCore system.

The IC_RAW_INTR_STAT register also specifies what is required to clear the specific interrupts. When the source of the interrupt is removed, reading the appropriate IC_CLR_* register (for example, IC_CLR_RX_OVER) clears the interrupt.

5.10.6.4 Built-in Two-Wire Serial Multiplexer

The device has built-in support for connecting to multiple two-wire serial devices that use the same two-wire serial address. This is done by using the multiplexed clock outputs (TWI_SCL_GATEn and TWI_SCL_GATEn_AD) for the two-wire serial devices rather than the TWI_SCL. Depending on which device or devices it needs to talk to, software can then enable/disable the various clocks. This multiplexing feature is available for the 1’st TWI controller instance.

From the two-wire serial controller’s point of view, it does not know if it is using TWI_SCL or TWI_SCL_GATEn/ TWI_SCL_GATEn_AD clocks. When using multiplexed mode, software needs to enable/disable individual clock connections before initiating the access operation using the two-wire serial controller. Feedback on the clock (from slow two-wire serial devices) is automatically done for the TWI_SCL_GATEn lines that are enabled. Clock stretching is not supported on TWI_SCL_GATEn_AD lines.

To enable multiplexed clocks, first configure the TWI_SCL_GATEn/TWI_SCL_GATEn_AD overlaid mode in the GPIO controller. Individual clocks are then enabled by writing 1 to the corresponding GPIO output bit (in

DEVCPU_GCB::GPIO_OUT). To disable the clock, write 0 to the GPIO output bit. Disabled clocks are not driven during access and the feedback is disabled.

Note:
In multiprocessor systems, the DEVCPU_GCB::GPIO_OUT_SET and DEVCPU_GCB::GPIO_OUT_CLR registers can be used to avoid race conditions.

5.10.7 MII Management Controller

This section provides information about the MII Management controllers. The following table lists the registers associated with the MII Management controllers.

Table 5-55. MIIM Registers

Register	Description
DEVCPU_GCB::MII_STATUS	Controller status
DEVCPU_GCB::MII_CMD	Command and write data
DEVCPU_GCB::MII_DATA	Read data
DEVCPU_GCB::MII_CFG	Clock frequency configuration
DEVCPU_GCB::MII_SCAN_0	Auto-scan address range
DEVCPU_GCB::MII_SCAN_1	Auto-scan mask and expects
DEVCPU_GCB::MII_SCAN_LAST_RSLTS	Auto-scan result
DEVCPU_GCB::MII_SCAN_LAST_RSLTS_VLD	Auto-scan result valid
DEVCPU_GCB::MII_SCAN_RSLTS_STICKY	Differences in expected versus read auto-scan

The device contains four MIIM controllers with the same functionality. Data is transferred on the MIIM interface using the Management Frame Format protocol specified in IEEE 802.3, Clause 22 or the MDIO Manageable Device protocol defined in IEEE 802.3, Clause 45. The Clause 45 protocol differs from the Clause 22 protocol by using indirect register access to increase the address range. The controller supports both Clause 22 and Clause 45.

The MIIM interface pins for the second and third controllers are overlaid functions on the GPIO interface. Before using these MIIM controllers, the overlaid functions for the appropriate GPIO pins must first be enabled. For more information, see 5.10.8.1 GPIO Overlaid Functions.

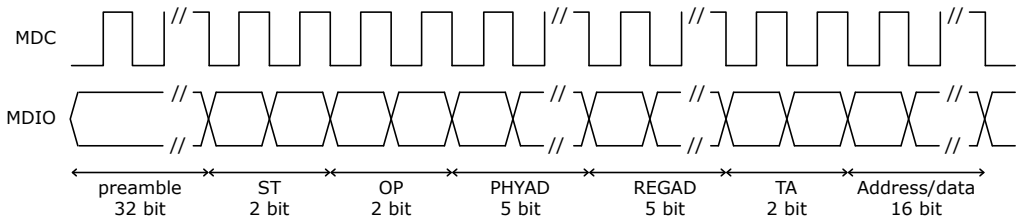
Table 5-56. MIIM Management Controller Pins

Pin Name	I/O	Description
MIIM0_MDC	O	MIIM clock for controller 0
MIIM0_MDIO	I/O	MIIM data input/output fro controller 0
MIIM1_MDC/GPIO	O	MIIM clock for controller 1
MIIM1_MDIO/GPIO	I/O	MIIM data input/output for controller 1
MIIM2_MDC/GPIO	O	MIIM clock for controller 2
MIIM2_MDIO/GPIO	I/O	MIIM data input/output for controller 2

.....continued		
Pin Name	**I/O**	**Description**
MIIM3_MDC/GPIO	O	MIIM clock for controller 3
MIIM3_MDIO/ GPIO	I/O	MIIM data input/output for controller 3

The MDIO signal is changed or sampled on the falling edge of the MDC clock by the controller. The MDIO pin is tri-stated in-between access and when expecting read data.

Figure 5-24. MII Management Timing



5.10.7.1 Clock Configuration

The frequency of the management interface clock generated by the MIIM controller is derived from the VCore system frequency. The MIIM clock frequency is configurable and is selected with MII_CFG.MIIM_CFG_PRESCALE. The calculation of the resulting frequency is explained in the register description for MII_CFG.MIIM_CFG_PRESCALE. The maximum frequency of the MIIM clock is 25 MHz.

5.10.7.2 MII Management PHY Access

Reads and writes across the MII management interface are performed through the MII_CMD register. Details of the operation, such as the PHY address, the register address of the PHY to be accessed, the operation to perform on the register (for example, read or write), and write data (for write operations), are set in the MII_CMD register. When the appropriate fields of MII_CMD are set, the operation is initiated by writing 0x1 to MII_CMD.MIIM_CMD_VLD. The register is automatically cleared when the MIIM command is initiated. When initiating single MIIM commands, MII_CMD.MIIM_CMD_SCAN must be set to 0x0.

When an operation is initiated, the current status of the operation can be read in MII_STATUS. The fields MII_STATUS.MIIM_STAT_PENDING_RD and MII_STATUS.MIIM_STAT_PENDING_WR can be used to poll for completion of the operation. For a read operation, the read data is available in MII_DATA.MIIM_DATA_RDDATA after completion of the operation. The value of MII_DATA.MIIM_DATA_RDDATA is only valid if MII_DATA.MIIM_DATA_SUCCESS indicates no read errors.

The MIIM controller contains a small command FIFO. Additional MIIM commands can be queued as long as MII_STATUS.MIIM_STAT_OPR_PEND is cleared. Care must be taken with read operations, because multiple queued read operations will overwrite MII_DATA.MIIM_DATA_RDDATA.

Note:
A typical software implementation will never queue read operations, because the software needs read data before progressing the state of the software. In this case, MIIM_STATUS.MIIM_STAT_OPR_PEND is checked before issuing MIIM read or write commands, for read-operations MII_STATUS.MIIM_STAT_BUSY is checked before returning read result.

By default, the MIIM controller operates in Clause 22 mode. To access Clause 45 compatible PHYs, MII_CFG.MIIM_ST_CFG_FIELD and MII_CMD.MIIM_CMD_OPR_FIELD must be set according to Clause 45 mode of operation.

5.10.7.3 PHY Scanning

The MIIM controller can be configured to continuously read certain PHY registers and detect if the read value is different from an expected value. If a difference is detected, a special sticky bit register is set or a CPU interrupt is

generated, or both. For example, the controller can be programmed to read the status registers of one or more PHYs and detect if the Link Status changed since the sticky register was last read.

The PHYs are read sequentially with the low and high PHY numbers specified in MII_SCAN_0 as range bounds. The accessed address within each of the PHYs is specified in MII_CMD.MIIM_CMD_REGAD. The scanning begins when a 0x1 is written to MII_CMD.MIIM_CMD_SCAN and a read operation is specified in MII_CMD.MIIM_CMD_OPR_FIELD. Setting MII_CMD.MIIM_CMD_SINGLE_SCAN stops the scanning after all PHYs have been scanned once. The remaining fields of MII_CMD register are not used when scanning is enabled.

The expected value for the PHY register is set in MII_SCAN_1.MIIM_SCAN_EXPECT. The expected value is compared to the read value after applying the mask set in MII_SCAN_1.MIIM_SCAN_MASK. To “don’t care” a bit-position, write a 0 to the mask. If the expected value for a bit position differs from the read value during scanning, and the mask register has a 1 for the corresponding bit, a mismatch for the PHY is registered.

The scan results from the most recent scan can be read in MII_SCAN_LAST_RSLTS. The register contains one bit for each of the possible 32 PHYs. A mismatch during scanning is indicated by a 0. MII_SCAN_LAST_RSLTS_VLD will indicate for each PHY if the read operation performed during the scan was successful. The sticky-bit register MII_SCAN_RSLTS_STICKY has the mismatch bit set for all PHYs that had a mismatch during scanning since the last read of the sticky-bit register. When the register is read, its value is reset to all-ones (no mismatches).

5.10.7.4 MII Management Interrupt

The MII management controllers can generate interrupts during PHY scanning. Each MII management controller has a separate interrupt signal to the interrupt controller. Interrupt is asserted when one or more PHYs have a mismatch during scan. The interrupt is cleared by reading the MII_SCAN_RSLTS_STICKY register, which resets all MII_SCAN_RSLTS_STICKY indications.

5.10.8 GPIO Controller

This section provides information about the use of GPIO pins. Many of the GPIO pins have overlaid functions. For more information, see 5.10.8.1 GPIO Overlaid Functions.

The following table lists the registers associated with GPIO.

Table 5-57. GPIO Registers

Register	Description
DEVCPU_GCB::GPIO_OUT	Value to drive on GPIO outputs
DEVCPU_GCB::GPIO_OUT_SET	Atomic set of bits in GPIO_OUT
DEVCPU_GCB::GPIO_OUT_CLR	Atomic clear of bits in GPIO_OUT
DEVCPU_GCB::GPIO_IN	Current value on the GPIO pins
DEVCPU_GCB::GPIO_OE	Enable of GPIO output mode (drive GPIOs)
DEVCPU_GCB::GPIO_ALT	Enable of overlaid GPIO functions
DEVCPU_GCB::GPIO_INTR	Interrupt on changed GPIO value
DEVCPU_GCB::GPIO_INTR_ENA	Enable interrupt on changed GPIO value
DEVCPU_GCB::GPIO_INTR_IDENT	Currently interrupting sources
DEVCPU_GCB::GPIO_SD_MAP	Mapping of parallel signal detects

The GPIO pins are individually programmable. GPIOs are inputs by default and can be individually changed to outputs through GPIO_OE. The value of the GPIO pins is reflected in the GPIO_IN register. GPIOs that are in output mode are driven to the value specified in GPIO_OUT.

In a system where multiple different CPU threads (or different CPUs) may work on the GPIOs at the same time, the GPIO_OUT_SET and GPIO_OUT_CLR registers provide a way for each thread to safely control the output value of individual GPIOs, without having to implement locked regions and semaphores.

The GPIO_ALT registers are only reset by external reset to the device. This means that software reset of the DEVCPU_GCB is possible without affecting the mapping of overlaid functions on the GPIOs.

5.10.8.1 GPIO Overlaid Functions

Most of the GPIO pins have overlaid (alternative) functions that can be enabled through replicated GPIO_ALT registers.

To enable a particular GPIO pin with the alternative function, set the G_ALT[n] register field in the replicated registers as follows:

- Overlaid mode 1, set GPIO_ALT[1][n], GPIO_ALT[0][n] = 1
- Overlaid mode 2, set GPIO_ALT[1][n], GPIO_ALT[0][n] = 2
- Overlaid mode 3, set GPIO_ALT[1][n], GPIO_ALT[0][n] = 3
- Normal GPIO mode, set GPIO_ALT[1][n], GPIO_ALT[0][n] = 0

When the MIIM slave mode is enabled through the VCore_CFG strapping pins, specific GPIO pins are overtaken and used for the MIIM slave interface. Similarly, other interfaces may also take over GPIO pins.

During reset, the VCore_CFG interface is sampled and used for VCore configuration. After reset, the device is released, and the GPIOs can be used for outputs or inputs. For more information, see 5.1 VCore Configurations.

The following table maps the GPIO pins and available overlaid functions.

Table 5-58. GPIO Overlaid Functions

Name	Overlaid Function 1	Overlaid Function2	Overlaid 14Function3	Interface	Strapping
GPIO_0	SG0_Clk	PLL_STAT0	—	DFT/CLKM_CFG0 / RO_NW	—
GPIO_1	SG0_DO	—	—	DFT/CLKM_CFG1 / RO_NE	—
GPIO_2	SG0_DI	—	—	DFT/CLKM_OUT / RO_SE	—
GPIO_3	SG0_LD	—	—	DFT/GPR_NE / RO_SW	—
GPIO_4	SG1_Clk	—	—	RO_C	—
GPIO_5	SG1_DO	—	—	—	—
GPIO_6	IRQ0_IN	IRQ0_OUT	SFP15_SD	—	—
GPIO_7	IRQ1_IN	IRQ1_OUT	SFP16_SD	—	—
GPIO_8	PTP_0	—	SFP17_SD	—	—
GPIO_9	PTP_1	SFP6_SD	TWI_SCL_GATE15 *1	—	—
GPIO_10	UART_RxD	—	—	—	—
GPIO_11	UART_TxD	—	—	—	—
GPIO_12	SG1_LD	—	—	—	LCPLL_RSVD
GPIO_13	SG1_DI	—	—	—	—

.....continued

Name	Overlaid Function 1	Overlaid Function2	Overlaid 14Function3	Interface	Strapping
GPIO_14	TWI_SCL	TWI_SCL_GATE0 *1	—	—	—
GPIO_15	TWI_SDA		—	—	—
GPIO_16	SPI_nCS1	TWI_SCL_GATE1 *1	SFP18_SD	—	—
GPIO_17	SPI_nCS2	TWI_SCL_GATE2 *1	SFP19_SD	—	—
GPIO_18	SPI_nCS3	TWI_SCL_GATE3 *1	SFP20_SD	—	—
GPIO_19	PCI_Wake	TWI_SCL_GATE4 *1	SFP21_SD	SPI2_SCK	—
GPIO_20	IRQ0_OUT	TWI_SCL_GATE5 *1	SFP22_SD	SPI2_D0	—
GPIO_21	IRQ1_OUT	TACHO	SFP23_SD	SPI2_D1	—
GPIO_22	TACHO	IRQ0_OUT	TWI_SCL_GATE16 *1	SPI3_SCK	—
GPIO_23	PWM	UART3_TxD	TWI_SCL_GATE17 *1	SPI3_DO	—
GPIO_24	PTP_2	UART3_RxD	TWI_SCL_GATE18 *1	SPI3_DI	—
GPIO_25	PTP_3	SPI_nCS4	TWI_SCL_GATE19 *1	—	—
GPIO_26	UART2_RxD	SPI_nCS5	TWI_SCL_GATE20 *1	SPI_nCS5	—
GPIO_27	UART2_TxD	SPI_nCS6	TWI_SCL_GATE21 *1	SPI_nCS6	—
GPIO_28	TWI2_SCL	SPI_nCS7	SFP24_SD	SPI_nCS7	—
GPIO_29	TWI2_SDA	SPI_nCS8	SFP25_SD	SPI_nCS8	—
GPIO_30	SG2_Clk	SPI_nCS9	PWM	—	LCPLL0_CONF0
GPIO_31	SG2_LD	SPI_nCS10	TWI_SCL_GATE22_AD*1	—	LCPLL0_CONF1
GPIO_32	SG2_DO	SPI_nCS11	TWI_SCL_GATE23_AD*1	—	LCPLL_CONF2
GPIO_33	SG2_DI	SPI_nCS12	SFP26_SD	PI_SLV_PG_ADDR3	—
GPIO_34		TWI_SCL_GATE6 *1	EMMC_nRST	PI_NCS	—

.....continued

Name	Overlaid Function 1	Overlaid Function2	Overlaid 14Function3	Interface	Strapping
GPIO_35	SFP27_SD	TWI_SCL_GATE7 *1	CARD_nDETECT	PI_NWR	—
GPIO_36	SFP28_SD	TWI_SCL_GATE8 *1	CARD_WP	PI_NOE	—
GPIO_37	SFP29_SD		CARD_LED	PI_NDONE	PI_NDONE
GPIO_38		TWI_SCL_GATE9 *1	EMMC_CMD	PI_ADDR2	—
GPIO_39	SPI2_SCK	TWI_SCL_GATE10 *1	EMMC_CK	PI_ADDR3	—
GPIO_40	SPI2_D0	TWI_SCL_GATE11 *1	EMMC_D0	PI_DATA0	—
GPIO_41	SPI2_D1	TWI_SCL_GATE12 *1	EMMC_D1	PI_DATA1	—
GPIO_42	SPI2_D2	TWI_SCL_GATE13 *1	EMMC_D2	PI_DATA2	—
GPIO_43	SPI2_D3	TWI_SCL_GATE14 *1	EMMC_D3	PI_DATA3	—
GPIO_44	SPI_nCS14	SFP7_SD	EMMC_D4	PI_DATA4	—
GPIO_45	SPI_nCS15	SFP8_SD	EMMC_D5	PI_DATA5	—
GPIO_46		SFP9_SD	EMMC_D6	PI_DATA6	—
GPIO_47		SFP10_SD	EMMC_D7	PI_DATA7	—
GPIO_48	TWI3_SCL	SPI_nCS13	SFP30_SD	PI_SLV_PG_ADDR0/ UART3_TxD	—
GPIO_49	TWI3_SDA	—	SFP31_SD	PI_SLV_PG_ADDR1/ UART3_RxD	—
GPIO_50	SFP0_SD	—	TWI_SCL_GATE24*1	PI_SLV_PG_ADDR2/ TWI3_SCL	—
GPIO_51	SFP1_SD	SPI_nCS0	TWI_SCL_GATE25*1	PI_SLV_PG_ADDR3/ TWI3_SDA	—
GPIO_52	SFP2_SD	MIIM3_MDC	TWI_SCL_GATE26*1	MIIM_SLV_MDC	—
GPIO_53	SFP3_SD	MIIM3_MDIO	TWI_SCL_GATE27*1	MIIM_SLV_MDIO	—
GPIO_54	SFP4_SD	PTP_2	TWI_SCL_GATE28*1	MIIM_SLV_ADDR	—
GPIO_55	SFP5_SD	PTP_3	PCI_Wake	SPI4_SCK	—
GPIO_56	MIIM1_MDC	SFP11_SD	TWI_SCL_GATE29*1	SPI4_DO	—
GPIO_57	MIIM1_MDIO	SFP12_SD	TWI_SCL_GATE30*1	SPI4_DI	—

.....continued					
Name	Overlaid Function 1	Overlaid Function2	Overlaid 14Function3	Interface	Strapping
GPIO_58	MIIM2_MDC	SFP13_SD	TWI_SCL_GATE31*1	SPI_D2	—
GPIO_59	MIIM2_MDIO	SFP14_SD	—	SPI_D3	—
GPIO_60	RECO_CLK0	—	—	VCORE_CFG0	VCORE_CFG0
GPIO_61	RECO_CLK1	—	—	VCORE_CFG1	VCORE_CFG1
GPIO_62	RECO_CLK2	PLL_STAT0	—	VCORE_CFG2	VCORE_CFG2
GPIO_63	RECO_CLK3	—	—	VCORE_CFG3	VCORE_CFG3

5.10.8.2 GPIO Interrupt

The GPIO controller continually monitors all inputs and set bits in the GPIO_INTR register whenever a GPIO changes its input value. By enabling specific GPIO pins in the GPIO_INTR_ENA register, a change indication from GPIO_INTR is allowed to propagate (as GPIO interrupt) from the GPIO controller to the VCore Interrupt Controller.

The currently interrupting sources can be read from GPIO_INTR_IDENT. This register is the result of a binary AND between the GPIO_INTR and GPIO_INTR_ENA registers.

5.10.8.3 Parallel Signal Detect

The GPIO controller has 32 programmable parallel signal detects named SFP0_SD through SFP31_SD. When parallel signal detect is enabled for a front-port index it overrides the signal-detect/loss-of-signal value provided by the serial GPIO controller.

To enable parallel signal detect *n*, first configure which port-index from the serial GPIO controller must be overwritten by setting GPIO_SD_MAP[*n*].G_SD_MAP. Then enable the SFP*n*_SD function on the GPIOs.

The following table lists parallel signal detect pins, which are overlaid on GPIOs. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Table 5-59. Parallel Signal Detect Pins

Register	I/O	Description
SFPn_SD/ GPIO	I	Parallel signal detect n, where n is 0–31

5.10.9 Serial GPIO Controller

The device features three Serial GPIO (SIO) controllers. By using a serial interface, the SIO controllers significantly extend the number of available GPIOs with a minimum number of additional pins on the device. The primary purpose of the SIO controllers is to connect control signals from SFP modules and to act as an LED controller.

Each SIO controller supports up to 128 serial GPIOs (SGPIOs) organized into 32 ports, with four SGPIOs per port.

The following table lists the registers associated with the SIO controller.

Table 5-60. SIO Registers

Register	Description
DEVCPU_GCB::SIO_INPUT_DATA	Input data

.....continued	
Register	**Description**
DEVCPU_GCB::SIO_CFG	General configuration
DEVCPU_GCB::SIO_CLOCK	Clock configuration
DEVCPU_GCB::SIO_PORT_CFG	Output port configuration
DEVCPU_GCB::SIO_PORT_ENA	Port enable
DEVCPU_GCB::SIO_PWM_CFG	PWM configuration
DEVCPU_GCB::SIO_INTR_POL	Interrupt polarity
DEVCPU_GCB::SIO_INTR_RAW	Raw interrupt status
DEVCPU_GCB::SIO_INTR_TRIGGER	Interrupt trigger mode configuration
DEVCPU_GCB::SIO_INTR	Currently interrupting SGPIOs
DEVCPU_GCB::SIO_INTR_ENA	Interrupt enable
DEVCPU_GCB::SIO_INTR_IDENT	Currently active interrupts
DEVCPU_GCB::HW_SGPIO_SD_CFG	Enable alternative mapping of input SPGIOs to device and SerDes signal detect
DEVCPU_GCB::HW_SGPIO_TO_SD_MAP_CFG	Configure alternative mapping of input SPGIOs to device signal detect
DEVCPU_GCB::HW_SPGIO_TO_SERDES_SD_MAP_CFG	Enable alternative mapping of input SPGIOs to SerDes signal detect

The following table lists the pins of the SIO controllers; these are overlaid on GPIO pins. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Table 5-61. SIO Controller Pins

Pin Name	I/O	Description
SG0_CLK/GPIO SG1_CLK/GPIO SG2_CLK/GPIO	O	SIO clock output, frequency is configurable using SIO_CLOCK.SIO_CLK_FREQ.
SG0_DO/GPIO SG1_DO/GPIO SG2_DO/GPIO	O	SIO data output.
SG0_DI/GPIO SG1_DI/GPIO SG2_DI/GPIO	I	SIO data input.
SG0_LD/GPIO SG1_LD/GPIO SG2_LD/GPIO	O	SIO load data, polarity is configurable using SIO_CFG.SIO_LD_POLARITY.

The SIO controller works by shifting SGPIO values out on SGn_DO through a chain of shift registers on the PCB. After shifting a configurable number of SGPIO bits, the SIO controller asserts SGn_LD, which causes the shift registers to apply the values of the shifted bits to outputs. The SIO controller can also read inputs while shifting out SGPIO values on SGn_DO by sampling the SGn_DI input. The values sampled on SGn_DI are made available to software.

If the SIO controller is only used for outputs, the use of the load signal is optional. If the load signal is omitted, simpler shift registers (without load) can be used, however, the outputs of these registers will toggle during shifting.

When driving LED outputs, it is acceptable that the outputs will toggle when SGPIO values are updated (shifted through the chain). When the shift frequency is fast, the human eye is not able to see the shifting through the LEDs.

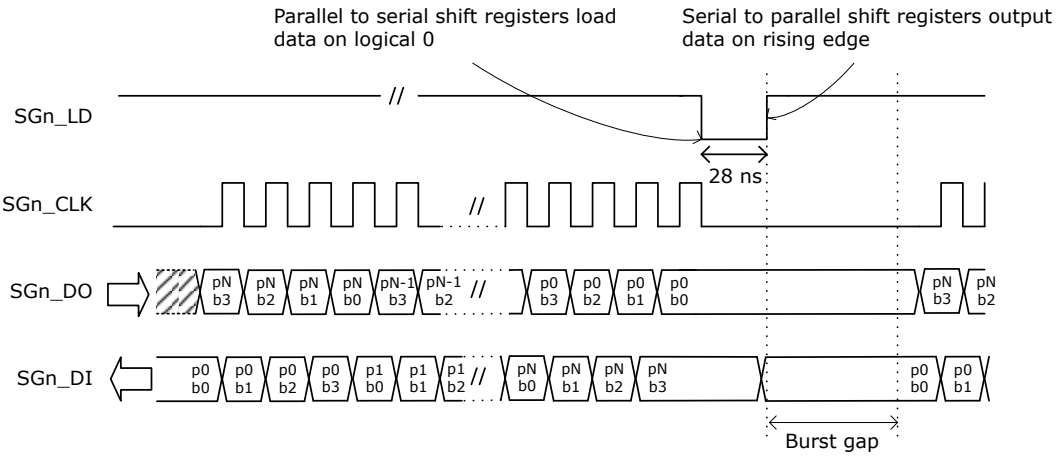
The number of shift registers in the chain is configurable. The SIO controller allows enabling of individual ports through SIO_PORT_ENA; only enabled ports are shifted out on SGn_DO. Ports that are not enabled are skipped during shifting of GPIO values.

Note:
SIO_PORT_ENA allows skipping of ports in the SGPIO output stream that are not in use. The number of GPIOs per (enabled) port is configurable as well, through SIO_CFG.SIO_PORT_WIDTH, this can be set to 1, 2, 3, or 4 bits. The number of bits per port is common for all enabled ports, so the number of shift registers on the PCB must be equal to the number of enabled ports times the number of SGPIOs per port.

Enabling of ports and configuration of SGPIOs per port applies to both output mode and input mode. Unlike a regular GPIO port, a single SGPIO position can be used both as output and input. That is, software can control the output of the shift register AND read the input value at the same time. Using SGPIOs as inputs requires load-capable shift registers.

Regular shift registers and load-capable shift-registers can be mixed, which is useful when driving LED indications for integrated PHYs while supporting reading of link status from SFP modules, for example.

Figure 5-25. SIO Timing



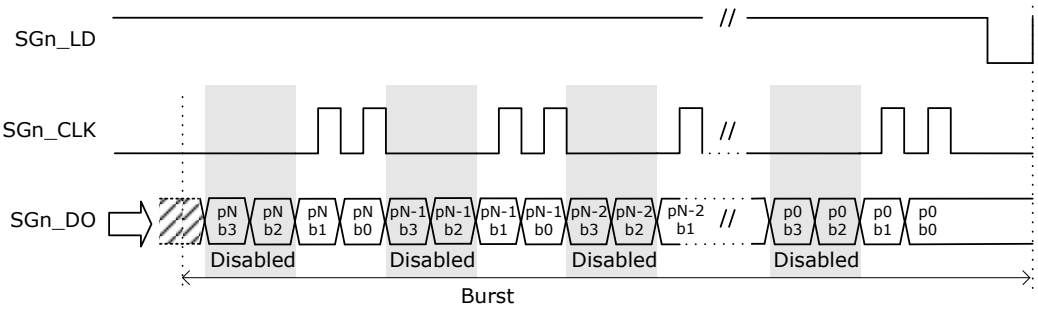
The SGPIO values are output in bursts followed by assertion of the SGn_LD signal. Values can be output as a single burst or as continuous bursts separated by a configurable burst gap. The maximum length of a burst is 32 × 4 data cycles. The burst gap is configurable in steps through SIO_CFG.SIO_BURST_GAP_DIS, SIO_CFG.SIO_BURST_GAP, and SIO_CFG.SIO_REDUCED_GAP.

A single burst is issued by setting SIO_CFG.SIO_SINGLE_SHOT. The field is automatically cleared by hardware when the burst is finished. To issue continuous bursts, set SIO_CFG.SIO_AUTO_REPEAT. The SIO controller continues to issue bursts until SIO_CFG.SIO_AUTO_REPEAT is cleared.

SGPIO output values are configured in SIO_PORT_CFG.BIT_SOURCE. The input value is available in SIO_INPUT_DATA.

The following figure shows what happens when the number of SGPIOs per port is configured to two (through SIO_CFG.SIO_PORT_WIDTH). Disabling of ports (through SIO_PORT_ENA) is handled in the same way as disabling the SGPIO ports.

Figure 5-26. SIO Timing with SGPIOs Disabled



The frequency of the SGn_CLK clock output is configured through SIO_CLOCK.SIO_CLK_FREQ. The SGn_LD output is asserted after each burst; this output is asserted for a period of 25 ns to 30 ns. The polarity of SGn_LD is configurable through SIO_CFG.SIO_LD_POLARITY.

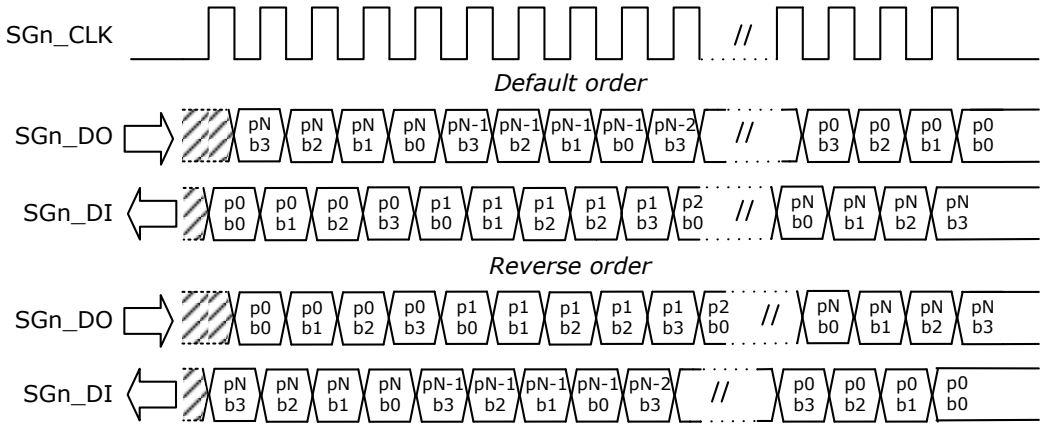
The SGn_LD output can be used to ensure that outputs are stable when serial data is being shifted through the registers. This can be done by using the SGn_LD output to shift the output values into serial-to-parallel registers after the burst is completed. If serial-to-parallel registers are not used, the outputs will toggle while the burst is being shifted through the chain of shift registers. A universal serial-to-parallel shift register outputs the data on a positive-edge load signal, and a universal parallel-to-serial shift register shifts data when the load pin is high, so one common load signal can be used for both input and output serial « parallel conversion.

The assertion of SGn_LD happens after the burst to ensure that after power up, the single burst will result in well-defined output registers. Consequently, to sample input values one time, two consecutive bursts must be issued. The first burst results in the input values being sampled by the serial-to-parallel registers, and the second burst shifts the input values into the SIO controller.

The port order required in the serial bitstream depends on the physical layout of the shift register chain. Often the input and output port orders must be opposite in the serial streams. The port order of the input and output bitstream is independently configurable in SIO_CFG.SIO_REVERSE_INPUT and SIO_CFG.SIO_REVERSE_OUTPUT.

The following figure shows the port order.

Figure 5-27. SGPIO Output Order



5.10.9.1 Output Modes

The output mode of each SGPIO can be individually configured in SIO_PORT_CFG.BIT_SOURCE. The SIO controller features three output modes:

- Static
- Blink
- Link activity

The output mode can additionally be modified with PWM (SIO_PORT_CFG.PWM_SOURCE) and configurable polarity (SIO_PORT_CFG.BIT_POLARITY).

- Static Mode:

  The static mode is used to assign a fixed value to the SGPIO, for example, fixed 0 or fixed 1.
- Blink Mode:

  The blink mode makes the SGPIO blink at a fixed rate. The SIO controller features two blink modes that can be set independently. A SGPIO can then be configured to use either blink mode 0 or blink mode 1. The blink outputs are configured in SIO_CFG.SIO_BMODE_0 and SIO_CFG.SIO_BMODE_1. To synchronize the blink modes between different devices, reset the blink counter using SIO_CFG.SIO_BLINK_RESET. All the SIO controllers on a device must be reset at same time to maintain the synchronization. The burst toggle mode of blink mode 1 toggles the output with every burst.

Table 5-62. Blink Modes

Register	Description
Blink Mode 0	0: 20 Hz blink frequency 1: 10 Hz blink frequency 2: 5 Hz blink frequency 3: 2.5 Hz blink frequency
Blink Mode 1	0: 20 Hz blink frequency 1: 10 Hz blink frequency 2: 5 Hz blink frequency 3: Burst toggle

- Link Activity Mode:

  The link activity mode makes the output blink when there is activity on the port module (Rx or Tx). The mapping between SIO port number and device is listed in the table below.

Table 5-63. SIO Controller Port Mapping

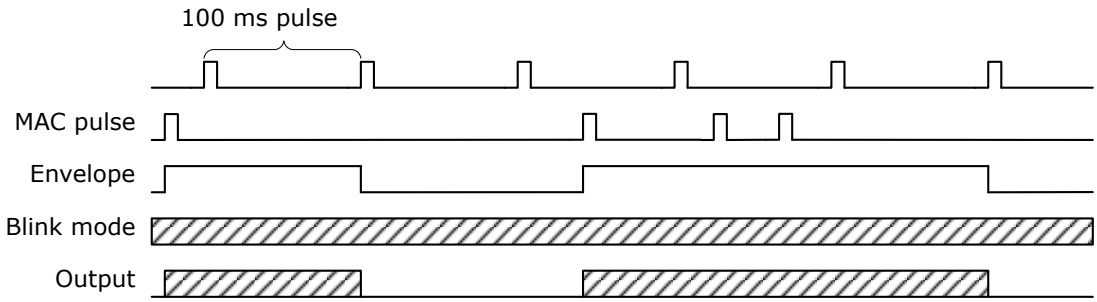
SIO Port	SG0 Mapping	SG1 Mapping	SG2 Mapping
0	Device 0	Device 32	Device 64
1	Device 1	Device 33	Device 1
2	Device 2	Device 34	Device 2
3	Device 3	Device 35	Device 3
4	Device 4	Device 36	Device 4
5	Device 5	Device 37	Device 5
6	Device 6	Device 38	Device 6
7	Device 7	Device 39	Device 7
8	Device 8	Device 40	Device 8
9	Device 9	Device 41	Device 9
10	Device 10	Device 42	Device 10
11	Device 11	Device 43	Device 11
12	Device 12	Device 44	Device 12
13	Device 13	Device 45	Device 13

.....continued			
SIO Port	**SG0 Mapping**	**SG1 Mapping**	**SG2 Mapping**
14	Device 14	Device 46	Device 14
15	Device 15	Device 47	Device 15
16	Device 16	Device 48	Device 48
17	Device 17	Device 49	Device 49
18	Device 18	Device 50	Device 50
19	Device 19	Device 51	Device 51
20	Device 20	Device 52	Device 52
21	Device 21	Device 53	Device 53
22	Device 22	Device 54	Device 54
23	Device 23	Device 55	Device 55
24	Device 24	Device 56	Device 56
25	Device 25	Device 57	Device 57
26	Device 26	Device 58	Device 58
27	Device 27	Device 59	Device 59
28	Device 28	Device 60	Device 60
29	Device 29	Device 61	Device 61
30	Device 30	Device 62	Device 62
31	Device 31	Device 63	Device 63

The link activity mode uses an envelope signal to gate the selected blinking pattern (blink mode 0 or blink mode 1). When the envelope signal is asserted, the output blinks, and when the envelope pattern is de-asserted, the output is turned off. To ensure that even a single packet makes a visual blink, an activity pulse from the port module is extended to minimum 100 ms. If another packet is sent while the envelope signal is asserted, the activity pulse is extended by another 100 ms. The polarity of the link activity modes can be set in SIO_PORT_CFG.BIT_SOURCE.

The following figure shows the link activity timing.

Figure 5-28. Link Activity Timing



5.10.9.2 SIO Interrupt

The SIO controller can generate interrupts based on the value of the input value of the SGPIOs. All interrupts are level sensitive.

The SIO controller has interrupt registers that each has one bit for each of the 128 GPIOs:

1. Set the respective bit of SIO_INTR_ENA high to enable a GPIO interrupt.
2. SIO_INTR_RAW is high if the corresponding input bit is high (corrected for polarity as configured in SIO_INTR_POL). This register changes when the input changes.
3. SIO_INTR is high if the condition of the configured trigger mode for the bit is met. The trigger mode can be configured in SIO_INTR_TRIGGER0 and SIO_INTR_TRIGGER1 between level-triggered, edge-triggered, falling-edge-triggered, and rising-edge-triggered interrupt. This register is a sticky bit vector and can only be cleared by software. A bit is cleared by writing a 1 to the bit position.
4. SIO_INTR_IDENT is the result of SIO_INTR with the disabled interrupts (from SIO_INTR_ENA) removed. This register changes when SIO_INTR or the enable registers change.

The SIO controller has one interrupt output connected to the main interrupt controller, which is asserted when one or more interrupts in SIO_INTR_IDENT are active. To determine which SGPIO is causing the interrupt, the CPU must read this register. The interrupt output remains high until all interrupts in SIO_INTR_IDENT are cleared (either by clearing SIO_INTR or disabling the interrupts in SIO_INTR_ENA).

5.10.9.3 Loss of Signal Detection

The SIO controller can propagate loss of signal detection inputs directly to the signal detection input of the port modules. This is useful when, for example, SFP modules are connected to the device. The mapping between SIO ports and port modules is the same as for the link activity outputs; port 0 is connected to port module 0, port 1 is connected to port module 1, and so on. Some devices support double mappings. To select between the low of the high mapping DEVCPU_GCB::HW_SGPIO_SD_CFG.SD_HIGH_ENA is used.

The value of SGPIO bit 0 of each SIO port is forwarded directly to the loss of signal input on the corresponding port module. The port module must enable the loss of signal input locally.

Loss of signal can also be taken directly from overlaid functions on the regular GPIOs. In that case, the input from the SIO controller is ignored. For more information, see 5.10.8.3 Parallel Signal Detect.

Alternatively, any SGPIO input can be mapped to any device or SerDes loss of signal input through HW_SGPIO_TO_SD_MAP_CFG and HW_SGPIO_TO_SERDES_SD_MAP_CFG. This configurable mapping is enabled by HW_SGPIO_SD_CFG.SD_MAP_SEL.

The polarity of the loss of signal input is configured using SIO_INT_POL, meaning the same polarity must be used for loss of signal detect and interrupt.

5.10.10 FAN Controller

The device includes a fan controller that can be used to control and monitor a system fan. A pulse-width-modulation (PWM) output regulates the fan speed. The fan speed is monitored using a TACHO input. The fan controller is especially powerful when combined with the internal temperature sensor. For more information, see 5.10.11 Temperature Sensor.

The following table lists the registers associated with the fan controller.

Table 5-64. Fan Controller Registers

Register	Description
DEVCPU_GCB::FAN_CFG	General configuration
DEVCPU_GCB::PWM_FREQ	PWM frequency configuration
DEVCPU_GCB::FAN_CNT	Fan revolutions counter

The following table lists fan controller pins, which are overlaid on GPIOs. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Table 5-65. Fan Controller Pins

Register	I/O	Description
TACHO/GPIO	I	TACHO input for counting revolutions
PWM/GPIO	O	PWM fan output

The PWM output can be configured to a variety of frequencies. For more information refer to the DEVCPU_GCB::PWM_FREQ register description.

The low frequencies can be used for driving three-wire fans using a FET/transistor. Frequencies around 25 kHz can be used for four-wire fans that use the PWM input internally to control the fan. The duty cycle of the PWM output is programmable from 0% to 100%, with 8-bit accuracy. The polarity of the output can be controlled by FAN_CFG.INV_POL, so a duty-cycle of 100%, for example, can be either always low or always high.

The PWM output pin can be configured to act as a normal output or as an open-collector output, where the output value of the pin is kept low, but the output enable is toggled. The open-collector output mode is enabled by setting FAN_CFG.PWM_OPEN_COL_ENA.

Note:
By using open-collector mode, it is possible to externally pull-up to a higher voltage than the maximum GPIO I/O supply.

The speed of the fan is measured using a 16-bit counter that counts the rising edges on the TACHO input. A fan usually gives one to four pulses per revolution, depending on the fan type. The counter value is available in the FAN_CNT register. Depending on the value of FAN_CFG.FAN_STAT_CFG, the FAN_CNT register is updated in two different ways:

- If FAN_CFG.FAN_STAT_CFG is set, the FAN_CNT register behaves as a 16-bit wrapping counter that shows the total number of ticks on the TACHO input.
- If FAN_CFG.FAN_STAT_CFG is cleared, the FAN_CNT register is updated one time per second with the number of TACHO ticks received during the last second.

Optionally, the TACHO input is gated by the polarity-corrected PWM output by setting FAN_CFG.GATE_ENA, so that only TACHO pulses received while the polarity corrected PWM output is high are counted. Glitches on the TACHO input can occur right after the PWM output goes high. As a result, the gate signal is delayed by 10 µs when PWM goes high. There is no delay when PWM goes low, and the length of the delay is not configurable. Software must read the counter value in FAN_CNT and calculate the RPM of the fan.

An example of how to calculate the RPM of the fan is if the fan controller is configured to 100 Hz and a 20% duty cycle, each PWM pulse is high in 2 ms and low in 8 ms. If gating is enabled, the gating of the TACHO input is open

in 1.99 ms and closed in 8.01 ms. If the fan is turning with 100 RPMs and gives two TACHO pulses per revolution, it will ideally give 200 pulses per minute. TACHO pulses are only counted in 19.99% of the time, so it will give 200 × 0.1999 = 39.98 pulses per minute. If the additional 10 µs gating time is ignored, the counter value is multiplied by 2.5 to get the RPM value, because there is a 20% duty cycle with two TACHO pulses per revolution. By multiplying with 2.5, the RPM value is calculated to 99.95, which is 0.05% off the correct value (due to the 10 µs gating time).

5.10.11 Temperature Sensor

This section provides information about the on-die temperature sensor. When enabled the temperature sensor logic will continually monitor the temperature of the die and make this available for software.

The following table lists the registers associated with the temperature monitor.

Table 5-66. Temperature Sensor Registers

Register	Description
HSIOWRAP::TEMP_SENSOR_CFG	Enabling of sensor.
HSIOWRAP::TEMP_SENSOR_STAT	Temperature value

The temperature sensor is enabled by setting TEMP_SENSOR_CFG.SAMPLE_ENA. After this the temperature sensor will sample temperature every 16 ms and show current temperature via TEMP_SENSOR_STAT.TEMP.

The TEMP_SENSOR_CFG.CLK_CYCLES_1US register must be configured to match the system clock frequency.

The formula for converting TEMP field value to centigrade temperature is:

Temp(C) = TEMP_SENSOR_STAT.TEMP / 4096 * 352.3 – 109.4

It will take approximately 16 ms after setting SAMPLE_ENA until the first temperature sample is ready. The TEMP_SENSOR_STAT.TEMP_VALID field will be set when the temperature value is available.

5.10.12 Memory Integrity Monitor

Soft errors happen in all integrated circuits, these are a result of natural alpha decay, cosmic radiation, or electrical disturbances in the environment in which the device operates. The chance of soft-errors happening in a memory (RAM) is higher than for flip flop based logic, because the memory structures are physically small and changes require less outside force than in flip flops. The device has built-in protection from soft errors by using error correcting code (ECC) on critical memories. In addition, the device allows monitoring and reporting of soft-error events.

The following table lists the registers associated with the memory integrity monitor.

Table 5-67. Integrity Monitor Registers

Register	Description
DEVCPU_GCB::MEMITGR_CTRL	Trigger monitor state changes.
DEVCPU_GCB::MEMITGR_STAT	Current state of the monitor and memory status.
DEVCPU_GCB::MEMITGR_INFO	Shows indication when in DETECT state.
DEVCPU_GCB::MEMITGR_IDX	Shows memory index when in DETECT state.
DEVCPU_GCB::MEMITGR_DIV	Monitor speed.

The memory integrity monitor looks for memory soft-error indications. Correctable (single bit) and non-correctable (multibit or parity) indications are detected during memory read and can be reported to software via “ITGR” software interrupt, see 5.4.2 VCore CPU Interrupt Controller for how to enable this interrupt.

The memory integrity monitor operates in three different states: IDLE, LISTEN, and DETECT. After a reset, the monitor starts in the IDLE state.

- IDLE:

The monitor is deactivated and in quiet mode. In this state, the memories still correct, detect, and store indications locally, but they are not able to report indications to the monitor.

- LISTEN:

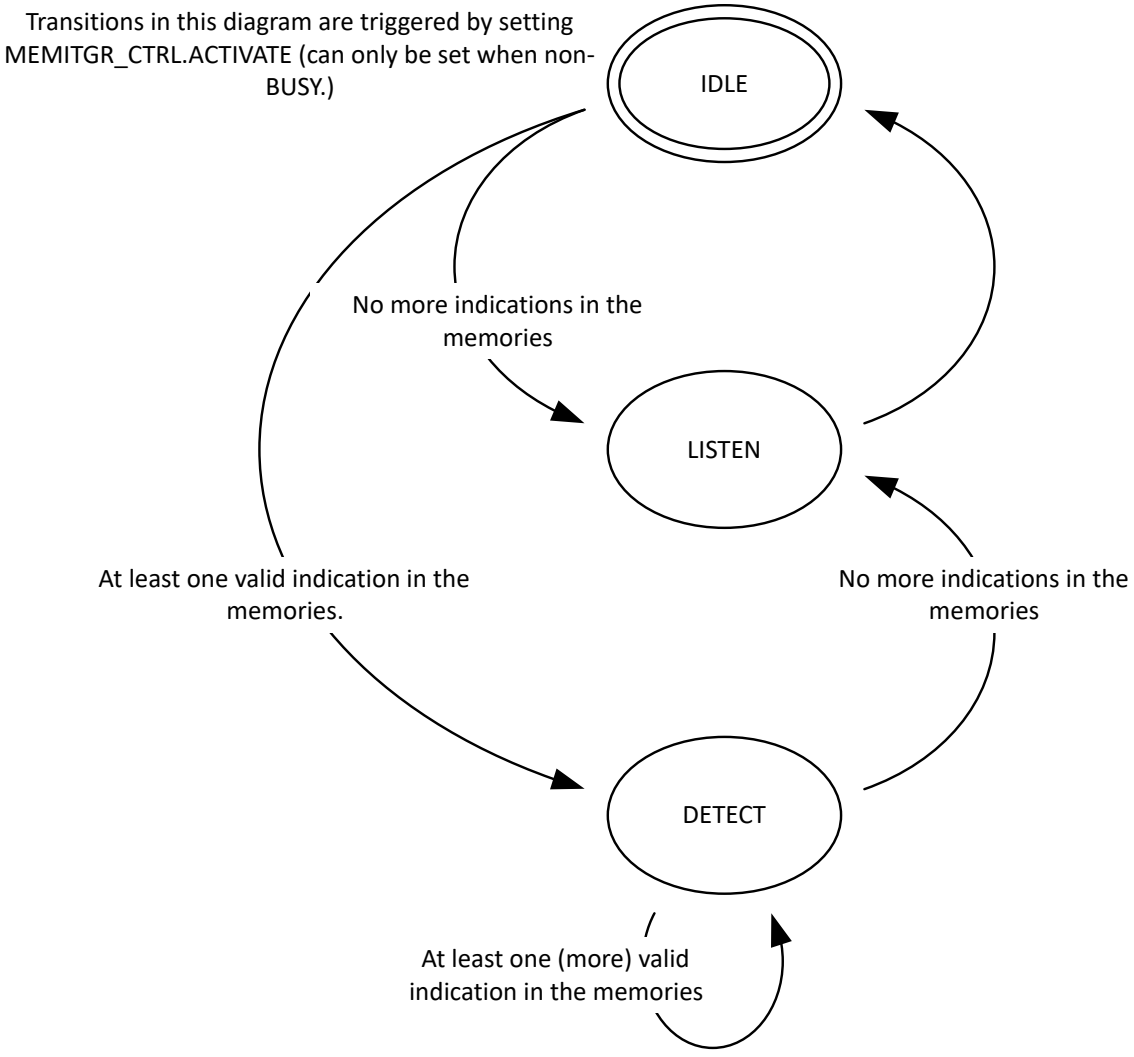
  In is state, the monitor looks for indications in the memories. MEMITGR_STAT.INDICATION is set (and interrupt is asserted) when indications are detected.

- DETECT:

  This state is used when indications are read from the memories. It means that a valid indication is available in MEMITGR_INFO and the corresponding memory index in MEMITGR_IDX.

The current state of the monitor is reported in MEMITGR_STAT.MODE_IDLE, MEMITGR_STAT.MODE_DETECT, and MEMITGR_STAT.MODE_LISTEN. Software initiates transitions between states by setting the one-shot MEMITGR_CTRL.ACTIVATE field. It may take some time to transition from one state to the next. The MEMITGR_CTR.ACTIVATE field is not cleared before the next state is reached (also the MEMITGR_STAT.MODE_BUSY field is set while transitioning between states).

Figure 5-29. Monitor State Diagram



The first time after reset that MEMITGR_CTRL.ACTIVATE is set, the monitor resets the detection logic in all the memories and transitions directly from IDLE to LISTEN state.

Before setting MEMITGR_CTRI.ACTIVATE for the first time, speed up the monitor by setting MEMITGR_DIV.MEM_DIV to the value specified in the registers list. The VCore CPU memories implement another error detection and correction mechanism and are not monitored.

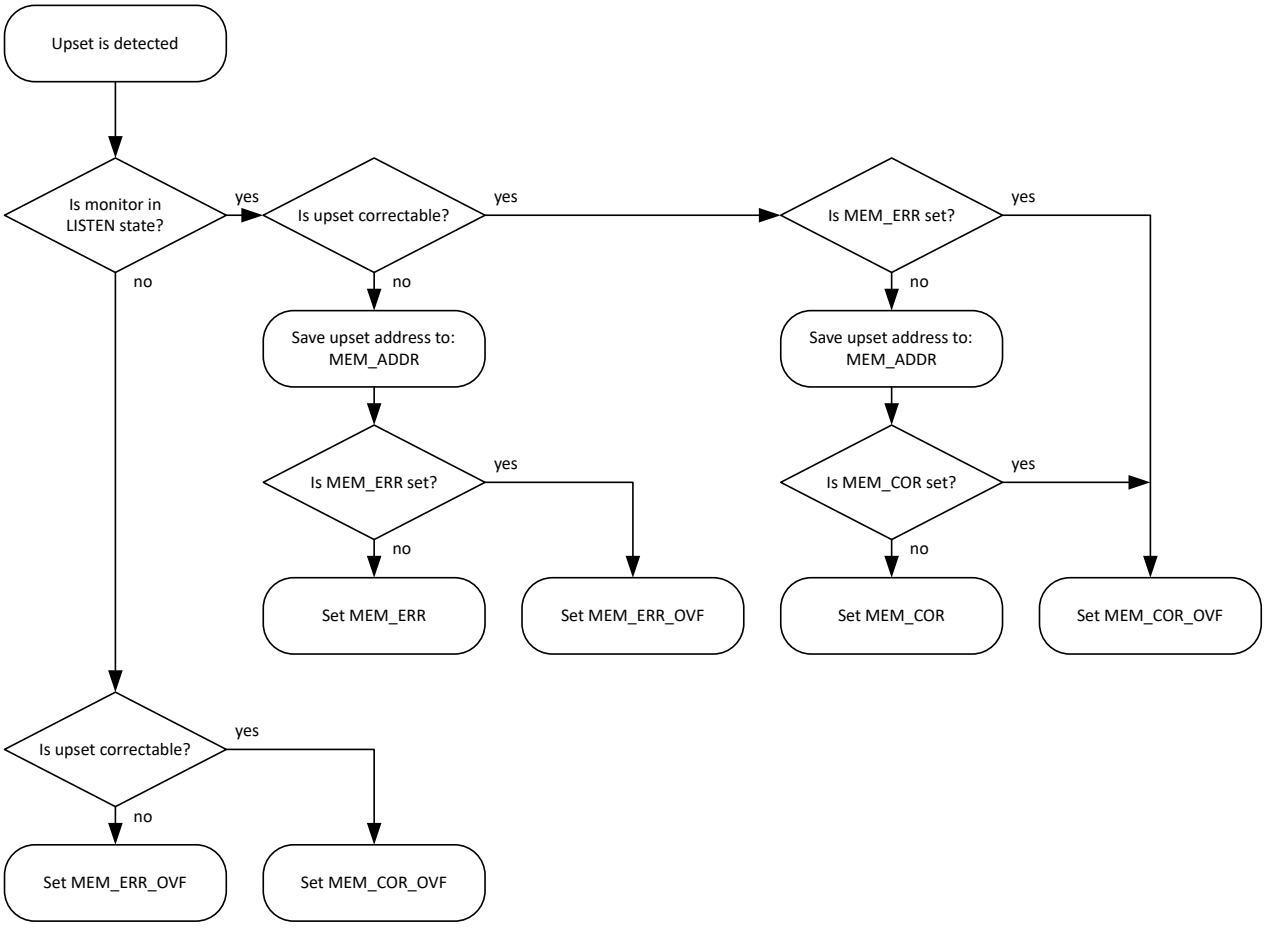
The memories of the PCIe controller are clock-gated and bypassed when PCIe interface is not enabled; enable monitoring of PCIe controller memories by setting CPU::PCIE_CFG.MEM_RING_CORE_ENA. This field must not be set when PCIe interface is not enabled. The VCore sub-CPU system memories are monitored when the VCore sub-CPU system is enabled and not forced into reset through CPU::SUBCPU_SYS_CFG. The VCore CPU, VCore CPU Interrupt Controller, and DDR controller memories are not monitored by the memory integrity controller.

To read out indications, first transition from LISTEN to IDLE, then continue transitioning until the LISTEN state is reached. Every time the monitor ends up in the DETECT state, an indication is available in the MEMITGR_INFO and MEMITGR_IDX registers. Each memory stores one indication. Indications are cleared when they are read by way of the monitor. Each indication contains four flags and one memory address.

- The MEM_ERR flag is set when a non-correctable upset is detected and the corresponding address is available in MEM_ADDR.
- The MEM_ERR_OVF flag is set when a non-correctable upset is detected for which address could not be stored.
- The MEM_COR flag is set when a correctable upset is detected and the corresponding address is available in MEM_ADDR.
- The MEM_COR_OVF flag is set when a correctable upset is detected for which address could not be stored.

Information about non-correctable upsets is prioritized over correctable upsets. Address can only be saved when the monitor is in LISTEN mode. The following flowchart shows how the detection logic sets flags and address.

Figure 5-30. Memory Detection Logic



If the MEM_ERR_OVF or MEM_COR_OVF flag is set, at least one event has occurred for which the address could not be stored.

The following table shows ECC-enabled memories in the device, their index, and the recommended approach for handling indications. If the controller reports an index that is not mentioned in the list, the recommended approach is to reboot the device.

Table 5-68. Memories with Integrity Support

Index	Description
Others	For unlisted indexes, the recommended approach is to reboot the device.

Reading from uninitialized memory locations has a high plausibility of triggering non-correctable or correctable indications. This is useful when developing integrity monitor software driver. For example, powering up a system without initializing the VCAPs and reading actions and sticky bits will trigger monitor indications. Note that the contents of memories are not changed by device reset, so power cycle is needed to reset the memories.

5.10.13 Outbound Interrupt Controller

This section provides information about the outbound VCore interrupt controller.

The following table lists the registers associated with this interrupt controller.

Table 5-69. Interrupt Controller Registers

Register	Description
CPU::INTR_RAW	Current value of interrupt inputs
CPU::INTR_BYPASS	Force non-sticky function
CPU::INTR_TRIGGER	Configure edge or level sensitive events
CPU::INTR_FORCE	Force events (for software debug)
CPU::INTR_STICKY	Currently logged events
CPU::INTR_ENA	Enable of interrupt sources
CPU::INTR_ENA_CLR	Atomic clear of bits in INTR_ENA
CPU::INTR_ENA_SET	Atomic set of bits in INTR_ENA
CPU::INTR_IDENT	Currently enabled and interrupting sources
CPU::DST_INTR_MAP	Mapping of interrupt sources to destinations
CPU::DST_INTR_IDENT	Currently enabled, mapped, and interrupting sources per destination
CPU::DEV_INTR_POL	Polarity of module interrupt inputs
CPU::DEV_INTR_RAW	Current value of module interrupts
CPU::DEV_INTR_BYPASS	Force non-sticky function for module interrupts
CPU::DEV_INTR_TRIGGER	Configure edge or level sensitive events for module interrupts
CPU::DEV_INTR_STICKY	Currently logged module interrupt events
CPU::DEV_INTR_ENA	Enable of module interrupts
CPU::DEV_INTR_IDENT	Currently interrupting and enabled module interrupts

.....continued	
Register	**Description**
CPU::KR6G_INTR_POL	Polarity of KR6G interrupt inputs
CPU::KR6G_INTR_RAW	Current value of KR6G interrupts
CPU::KR6G_INTR_BYPASS	Force non-sticky function for KR6G interrupts
CPU::KR6G_INTR_TRIGGER	Configure edge or level sensitive events for KR6G interrupts
CPU::KR6G_INTR_STICKY	Currently logged KR6G interrupt events
CPU::KR6G_INTR_ENA	Enable of KR6G interrupts
CPU::KR6G_INTR_IDENT	Currently interrupting and enabled KR6G interrupts
CPU::KR10G_INTR_POL	Polarity of KR10G interrupt inputs
CPU::KR10G_INTR_RAW	Current value of KR10G interrupts
CPU::KR10G_INTR_BYPASS	Force non-sticky function for KR10G interrupts
CPU::KR10G_INTR_TRIGGER	Configure edge or level sensitive events for KR10G interrupts
CPU::KR10G_INTR_STICKY	Currently logged KR10G interrupt events
CPU::KR10G_INTR_ENA	Enable of KR10G interrupts
CPU::KR10G_INTR_IDENT	Currently interrupting and enabled KR10G interrupts
CPU::EXT_SRC_INTR_POL	Polarity of external interrupt inputs.
CPU::EXT_DST_INTR_POL	Polarity of external interrupt outputs.
CPU::EXT_INTR_DRV	Drive mode for external interrupt outputs

The interrupt controller maps interrupt sources from VCore and switch-core blocks, port modules, and external interrupt inputs to two interrupt destinations, which can be transmitted from the device using the overlaid functions on GPIOs or using PCIe inband interrupt signaling.

The following table lists the available interrupt sources in the device. The index can be used to access the appropriate bits in CPU::INTR_* and CPU::DST_INTR_* registers.

Table 5-70. Interrupt Sources

Source Name	Index	Description
DEV	0	Aggregated port module interrupt. This interrupt is asserted if there is an active and enabled interrupt from any of the port modules. See 5.10.13.3 Port Module Interrupts.

.....continued

Source Name	Index	Description
KR6G	1	Aggregated KR6G serdes module interrupt. This interrupt is asserted if there is an active and enabled interrupt from any of the KR6G serdes modules. See 5.10.13.4 KR6G and KR10G Module Interrupts.
KR10G	2	Aggregated KR10G serdes module interrupt. This interrupt is asserted if there is an active and enabled interrupt from any of the KR10G serdes modules. See 5.10.13.4 KR6G and KR10G Module Interrupts.
EXT_SRC0	3	External interrupt source 0. See 5.10.13.5 GPIO Mapped Interrupts.
EXT_SRC1	4	External interrupt source 1. See 5.10.13.5 GPIO Mapped Interrupts.
Reserved	5	Reserved.
MSHC_WAKE	6	eMMC/SD card controller wake interrupt. See 5.10.14 Mobile Storage Host Controller.
MSHC	7	eMMC/SD card controller interrupt. See 5.10.14 Mobile Storage Host Controller.
WDT	8	Watchdog timer interrupt. See 5.2.1 Watchdog Timer.
TIMER0	9	Timer 0 interrupt. See 5.10.4 Timers.
TIMER1	10	Timer 1 interrupt. See 5.10.4 Timers.
TIMER2	11	Timer 2 interrupt. See 5.10.4 Timers.
UART	12	UART interrupt. See 5.10.5.1 UART Interrupt.
UART2	13	UART2 interrupt. See 5.10.5.1 UART Interrupt.
TWI	14	TWI interrupt. See 5.10.6.3 Two-Wire Serial Interface Interrupt.
TWI2	15	TWI2 interrupt. See 5.10.6.3 Two-Wire Serial Interface Interrupt.
SIMC	16	Serial Master Controller interrupt. See 5.10.2.3 SIMC Interrupts.
SUBCPU_LOCK	17	VCore Sub-CPU lock interrupt.
SW0	18	Software interrupt 0. See 5.5.5 Mailbox and Semaphores.
SW1	19	Software interrupt 1. See 5.5.5 Mailbox and Semaphores.
SGPIO0	20	Serial GPIO interrupt 0. See 5.10.9.2 SIO Interrupt.
SGPIO1	21	Serial GPIO interrupt 1. See 5.10.9.2 SIO Interrupt.
SGPIO2	22	Serial GPIO interrupt 2. See 5.10.9.2 SIO Interrupt.
GPIO	23	Parallel GPIO interrupt. See 5.10.8.2 GPIO Interrupt.
MIIM0	24	MIIM Controller 0 interrupt. See 5.10.7.4 MII Management Interrupt.
MIIM1	25	MIIM Controller 1 interrupt. See 5.10.7.4 MII Management Interrupt.
MIIM2	26	MIIM Controller 2 interrupt. See 5.10.7.4 MII Management Interrupt.

.....continued		
Source Name	**Index**	**Description**
MIIM3	27	MIIM Controller 3 interrupt. See 5.10.7.4 MII Management Interrupt.
FDMA	28	Frame DMA interrupt. See 5.7.4 FDMA Events and Interrupts.
ANA	29	Analyzer interrupt.
PTP_RDY	30	Timestamp ready interrupt.
PTP_SYNC	31	PTP synchronization interrupt.
ITGR	32	Memory integrity interrupt. See 5.10.12 Memory Integrity Monitor.
XTR_RDY	33	Extraction data ready interrupt.
INJ_RDY	34	Injection ready interrupt.
PCIE	35	PCIe interrupt. See 5.6.7 Power Management.
EACL	37	EACL interrupt.
REW	38	Rewriter interrupt.
DDR	39	DDR controller interrupt.
CPU[9:0]	49:40	Software interrupts CPU::SHARED_INTR.

The following table lists the available interrupt destinations in the device.

Table 5-71. Interrupt Destinations

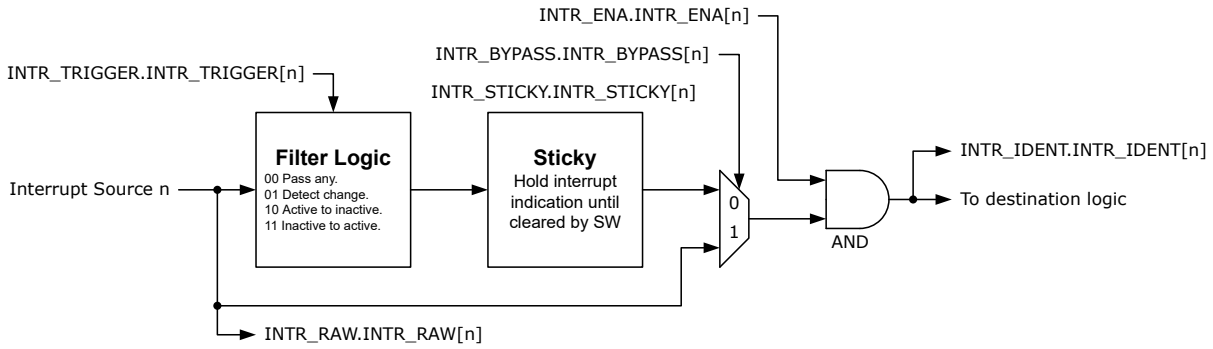
Destination Name	Description
EXT_DST0	External interrupt destination 0, see 5.10.13.5 GPIO Mapped Interrupts.
EXT_DST1	External interrupt destination 1, see 5.10.13.5 GPIO Mapped Interrupts.

All interrupts, events, and indications inside in the interrupt controller are active high. If an interrupt source supports polarity correction, it is applied before going into the interrupt controller. If an interrupt destination supports polarity correction, it is applied after leaving the interrupt controller.

5.10.13.1 Interrupt Source Configuration

Interrupt sources are handled identically inside the interrupt controller. This section describes interrupt source n, which refers to the bit index of that interrupt source in the INTR_* and DST_INTR_* registers. The following illustration shows the logic associated with a single interrupt source.

Figure 5-31. Interrupt Source Logic



The current value of an interrupt source is available in INTR_RAW.INTR_RAW[n]. INTR_STICKY.INTR_STICKY[n] is set when the interrupt controller detects an interrupt. There are following detection methods.

- When INTR_TRIGGER.INTR_TRIGGER[n] is set to level-activated, the interrupt controller continually sets INTR_STICKY.INTR_STICKY[n] for as long as the interrupt source is active.
- When INTR_TRIGGER.INTR_TRIGGER[n] is set to edge-triggered, the interrupt controller only sets INTR_STICKY.INTR_STICKY[n] when the interrupt source changes value.
- When INTR_TRIGGER.INTR_TRIGGER[*n*] is set to falling-edge-triggered, the interrupt controller only sets INTR_STICKY.INTR_STICKY[*n*] when the interrupt source changes from active to inactive value.
- When INTR_TRIGGER.INTR_TRIGGER[*n*] is set to rising-edge-triggered, the interrupt controller only sets INTR_STICKY.INTR_STICKY[*n*] when the interrupt source changes from inactive to active value.

Software can clear INTR_STICKY.INTR_STICKY[n] by writing 1 to bit n. However, the interrupt controller will immediately set this bit again if the source input is still active (when INTR_TRIGGER is 0) or if it sees a triggering event on the source input (when INTR_TRIGGER different from 0).

The interrupt source is enabled in INTR_ENA.INTR_ENA[n]. When INTR_STICKY.INTR_STICKY[n] is set and the interrupt is enabled, the interrupt is indicated towards the interrupt destinations. For more information, see 5.10.13.2 Interrupt Destination Configuration. An active and enabled interrupt source sets INTR_IDENT.INTR_IDENT[n].

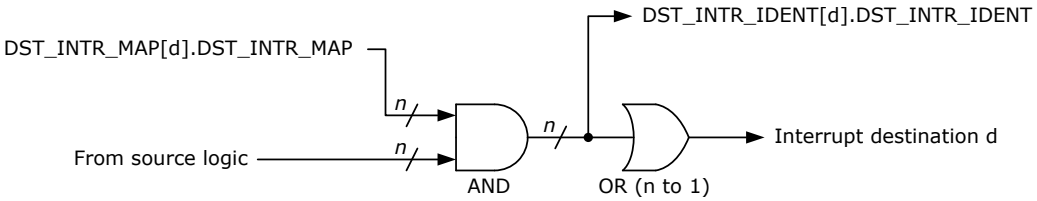
On rare occasions it is desirable to bypass the stickiness of interrupt sources and use INTR_RAW.INTR_RAW[n] directly instead of INTR_STICKY.INTR_STICKY[n]. Set INTR_BYPASS.INTR_BYPASS[n] to enable bypass and ignore INTR_STICKY and INTR_TRIGGER configurations.

Note:
The bypass function may be useful for some software interrupt handler architectures. It should only be used for interrupt sources that are guaranteed to be sticky in the source block. For example, the GPIO interrupts that are generated from sticky bits in DEVCPU_GCB::GPIO_INTR may be applicable for the bypass mode.

5.10.13.2 Interrupt Destination Configuration

The two interrupt destinations are handled identically in the interrupt controller. This section describes destination d, which refers to the replication index of that interrupt in the DST_INTR_* registers. The following figure shows the logic associated with a single interrupt destination.

Figure 5-32. Interrupt Destination Logic



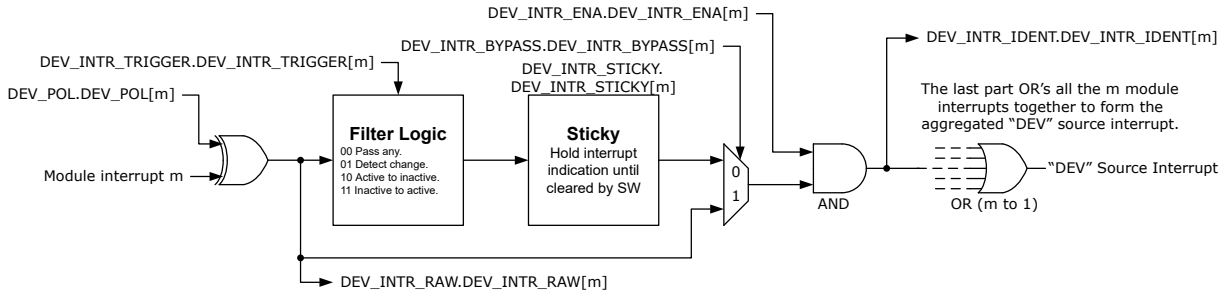
The interrupt destination can enable individual sources for interrupt by writing a mask to DST_INTR_MAP[d].DST_INTR_MAP. When a source is enabled in DST_INTR_MAP then an interrupt from this source will be propagated to the interrupt destination.

The currently active and enabled source interrupts for a destination can be seen by reading DST_INTR_IDENT[d].DST_INTR_IDENT.

5.10.13.3 Port Module Interrupts

Each port module can generate an interrupt. Because there are too many modules to handle the interrupts in parallel with the other source interrupts in the INTR_* registers, the port module interrupts are aggregated in a separate source interrupt hierarchy before being presented to the interrupt controller source logic as the DEV source interrupt.

Figure 5-33. Port Module Interrupt Logic



The module interrupt polarity is configurable in DEV_INTR_POL.DEV_INTR_POL[m].

DEV_INTR_RAW, DEV_INTR_TRIGGER, DEV_INTR_STICKY, DEV_INTR_BYPASS, DEV_INTR_ENA, and DEV_INTR_IDENT works in the same way as the INTR_* counterparts, see 5.10.13.1 Interrupt Source Configuration for more information.

The final step when handling module interrupts is an aggregation of all individual module interrupts to the DEV source interrupt.

5.10.13.4 KR6G and KR10G Module Interrupts

Each SerDes module generates a KR interrupt. Similarly to 5.10.13.3 Port Module Interrupts, the KR interrupts are aggregated in two separate source interrupt hierarchies, which are aggregated to source interrupts KR6G and KR10G respectively. The KR module interrupt logic is identical to the depicted in where the DEV* registers are replaced by KR6G* and KR10G* for the two hierarchies.

5.10.13.5 GPIO Mapped Interrupts

The interrupt controller supports two external GPIO mapped source interrupts (inputs to the device) and two external GPIO destination interrupts (outputs from the device). The external interrupts are mapped to GPIOs using overlaid functions. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Source and destination interrupts work independently from each other and they can be used at the same time. The polarity (active high or low) of source and destination interrupts is configured in EXT_SRC_INTR_POL and EXT_DST_INTR_POL respectively.

Table 5-72. External Interrupt Pins

Register	I/O	Description
IRQ0_IN/GPIO	I	External Source Interrupt 0; polarity is configured in EXT_SRC_INTR_POL.EXT_INTR_POL[0].
IRQ1_IN/GPIO	I	External Source Interrupt 1; polarity is configured in EXT_SRC_INTR_POL.EXT_INTR_POL[1].
IRQ0_OUT/ GPIO	O	External Destination Interrupt 0; polarity is configured in EXT_DST_INTR_POL.EXT_INTR_POL[0].
IRQ1_OUT/ GPIO	O	External Destination Interrupt 1; polarity is configured in EXT_DST_INTR_POL.EXT_INTR_POL[1].

For destination interrupts, it is possible to drive the output pin permanently or emulate open-collector output.

- To drive permanently configure EXT_INTR_DRV[e] = 0.

- To emulate open collector output configure EXT_INTR_DRV[e] = 1 and EXT_INTR_POL[e] = 0. To safely enable open-collector output, the EXT_INTR_DRV and EXT_INTR_POL registers must be configured before enabling the overlaid function in the GPIO controller.

Note:
Open collector output mode is needed when multiple interrupt sources are hooked up to the same interrupt wire on the PCB and the wire is pulled high with a resistor. Each interrupt source can then drive the wire low via open-collector output when they want to signal interrupt.

5.10.14 Mobile Storage Host Controller

The device is equipped with an SD/eMMC host controller. The supported card types and modes of operation are listed in the following table. The controller is enhanced with a DMA to facilitate transfers to/from the main memory.

Table 5-73. SD/eMMC Host Controller Support of Card Types and Modes

Card Type	Mode	Card Clock Frequency (MHz)	Bus Width
eMMC	Legacy	25	1/4/8
eMMC	High speed SDR	50	1/4/8
eMMC	High speed DDR	50	1/4/8
SD	SDR12	25	1/4
SD	SDR25	50	1/4
SD	SDR50	100	1/4
SD	DDR50	50	1/4

The following table lists the registers of the mobile storage host controller. The controller interface is mapped on GPIOs. For more information, see 5.10.8.1 GPIO Overlaid Functions.

Table 5-74. SD/eMMC Host Controller Registers

Register	Description
MSHC::SDMASA_R	DMA configuration.
MSHC::BLOCKSIZE_R	DMA configuration.
MSHC::BLOCKCOUNT_R	DMA configuration.
MSHC::ARGUMENT_R	SD/eMMC command argument.
MSHC::XFER_MODE_R	Controls the operation of data transfers.
MSHC::CMD_R	Used to provide information related to a command and a response packet.
MSHC::RESP01_R	Stores the 39-28 bits of the response field.
MSHC::RESP23_R	Stores the 71-40 bits of the response field.
MSHC::RESP45_R	Stores the 103-72 bits of the response field.
MSHC::RESP67_R	Stores the 135-104 bits of the response field.
MSHC::BUF_DATA_R	Used to access the buffer data.
MSHC::PSTATE_REG	Present status register.

.....continued

Register	Description
MSHC::HOST_CTRL1_R	Controls the operation of the host controller.
MSHC::PWR_CTRL_R	Controls the bus power for the card.
MSHC::BGAP_CTRL_R	Controls any operation related to block gap.
MSHC::WUP_CTRL_R	Wake-up control.
MSHC::CLK_CTRL_R	Controls the card clock.
MSHC::TOUT_CTRL_R	Data timeout configuration.
MSHC::SW_RST_R	Software reset.
MSHC::NORMAL_INT_STAT_R	Interrupt status.
MSHC::ERROR_INT_STAT_R	Error interrupt status.
MSHC::NORMAL_INT_STAT_EN_R	Raw interrupt enable.
MSHC::ERROR_INT_STAT_EN_R	Raw interrupt enable.
MSHC::NORMAL_INT_SIGNAL_EN_R	Interrupt mask.
MSHC::ERROR_INT_SIGNAL_EN_R	Interrupt mask.
MSHC::AUTO_CMD_STAT_R	Respond status for CMD12 and CMD23.
MSHC::HOST_CTRL2_R	Controls host controller operation.
MSHC::CAPABILITIES1_R	Read-only capabilities register.
MSHC::CAPABILITIES2_R	Read-only capabilities register.
MSHC::CURR_CAPABILITIES1_R	Read-only capabilities register.
MSHC::CURR_CAPABILITIES2_R	Read-only capabilities register.
MSHC::FORCE_AUTO_CMD_STAT_R	Force AUTO_CMD_STAT_R.
MSHC::FORCE_ERROR_INT_STAT_R	Force ERROR_INT_STAT_R.
MSHC::ADMA_ERR_STAT_R	Stores the ADMA state during an ADMA error.
MSHC::ADMA_SA_LOW_R	Lower 32-bit system address for DMA transfer.
MSHC::ADMA_SA_HIGH_R	Upper 32-bit system address for DMA transfer.
MSHC::ADMA_ID_LOW_R	Lower 32-bit integrated descriptor address.
MSHC::ADMA_ID_HIGH_R	Upper 32-bit integrated descriptor address.
MSHC::MSHC_CTRL_R	Controls the operation of the controller.
MSHC::MBIU_CTRL_R	Configures AHB interface.
MSHC::EMMC_CTRL_R	Controls eMMC operation.
MSHC::BOOT_CTRL_R	Controls eMMC boot operation.
MSHC::AT_CTRL_R	Controls some aspects of auto-tuning features.

.....continued	
Register	**Description**
MSHC::AT_STAT_R	Used for software managed tuning.
MSHC::CQCAP	Read-only capabilities of the command queuing engine.

5.10.14.1 Card Clock Frequency Configuration

Following is an example of card clock frequency configuration.

```
CLK_CTRL_R.CLK_GEN_SELECT = 0
CLK_CTRL_R.UPPER_FREQ_SEL = 0
CLK_CTRL_R.FREQ_SEL = X,
```

Where X is:

128 for 390 kHz

2 for 25 MHz

1 for 50 MHz

0 for 100 MHz

CLK_CTRL_R.INTERNAL_CLK_EN = 1

CLK_CTRL_R.PLL_ENABLE = 1

Wait until CLK_CTRL_R.INTERNAL_CLK_STABLE becomes 1.

5.10.14.2 Host Setup for eMMC

Following is an example of host setup for eMMC.

```
PWR_CTRL_R.SD_BUS_VOL_VDD1 = 7
TOUT_CTRL_R.TOUT_CNT = 3
HOST_CTRL2_R.UHS2_IF_ENABLE = 0
EMMC_CTRL_R.CARD_IS_EMMC = 1
Set card clock frequency to 390kHz through 5.10.14.1  Card Clock Frequency
Configuration
HOST_CTRL2_R.HOST_VER4_ENABLE = 1
HOST_CTRL2_R.ADDRESSING = 1
HOST_CTRL2_R.UHS2_IF_ENABLE = 0
PWR_CTRL_R.SD_BUS_PWR_VDD1 = 1
HOST_CTRL2_r.UHS_MODE_SEL = 0
CLK_CTRL_R.SD_CLK_EN 1
```

Wait for 180 us.

5.10.14.3 Host Setup for SD

Following is an example of the host setup for SD.

```
PWR_CTRL_R.SD_BUS_VOL_VDD1 = 7
TOUT_CTRL_R.TOUT_CNT = 3
HOST_CTRL2_R.ADDRESSING = 1
HOST_CTRL2_R.ASYNC_INT_ENABLE = 1
PWR_CTRL_R.SD_BUS_PWR_VDD1 = 1
HOST_CTRL2_R.UHS_MODE_SEL = 0
CLK_CTRL_R.SD_CLK_EN 1
```

Wait for 180 us.

5.10.14.4 Send Command to Card

To send command to card, perform the following steps.

1. Wait until PSTATE_REG.CMD_INHIBIT is 0.

2. Wait until PSTATE_REG.CMD_INHIBIT_DAT is 0 if required.
3. Program ARGUMENT_R
4. Registers CMD_R and XFER_MODE_R are mapped to the same 32 bit-aligned address. Refer to register descriptions for details regarding their fields. Send a command to the card with an atomic write access covering the combined fields of CMD_R and XFER_MODE_R.
5. Wait for NORMAL_INT_STAT_R.CMD_COMPLETE.
6. Clear NORMAL_INT_STAT_R.CMD_COMPLETE.
7. Response status is available at RESP01_R, RESP23_R, RESP45_R and RESP67_R.

5.10.14.5 Initialization of eMMC Card

To initialise eMMC card, perform the following steps.

1. SEND_OP_COND, CMD1: Ask card to send its Operating Condition Register contents in the response on the CMD line and read the response to deduct card size.
2. ALL_SEND_CID, CMD2: Ask card to send its CID number on the CMD line.
3. SET_RELATIVE_ADDR, CMD3: Assign relative address to the card.
4. SELECT/DESELECT_CARD, CMD7: Select card using the address that was configured in the previous step.

5.10.14.6 Initialization of SD Card

To initialise SD card, perform the following steps.

1. GO_IDLE_STATE, CMD0: Reset the SD memory card.
2. SEND_IF_COND, CMD8: Voltage check.
3. APP_CMD, CMD55: Next command is application command.
4. SD_SEND_OP_COND, ACMD41: Ask the card to send its size and initialize card.
5. Voltage switch routine
    5.1. CMD11
    5.2. CLK_CTRL_R.SD_CLK_EN = 0
    5.3. Check PSTATE_REG.DAT_3_0 is 0.
    5.4. HOST_CTRL2_R.SIGNALING_EN = 1
    5.5. Wait 5 ms
    5.6. Check that HOST_CTRL2_R.SIGNALING_EN is still 1
    5.7. CLK_CTRL_R.SD_CLK_EN = 1
    5.8. Wait 1 ms
    5.9. Check PSTATE_REG.DAT_3_0 is 15.
6. ALL_SEND_CID, CMD2: Ask card to send its CID number on the CMD line.
7. SEND_RELATIVE_ADDR: Ask card to publish a new relative address.
8. SELECT/DESELECT_CARD, CMD7: Select card.
9. Verify that the transfer was completed by checking and clearing NORMAL_INT_STAT_R.XFER_COMPLETE.

5.10.14.7 Configuration of eMMC Speed and Bus Mode

After configuring the host (5.10.14.2 Host Setup for eMMC) and initializing the eMMC card (5.10.14.5 Initialization of eMMC Card), and before accessing the card, the speed and the bus mode need to be configured. The supported modes are listed in Table 5-73.

In order to set the speed, send the command SWITCH (CMD6) to the card to modify the speed mode. Verify that the transfer was completed by checking and clearing NORMAL_INT_STAT_R.XFER_COMPLETE. The host must be configured accordingly. If a high speed mode is chosen, then HOST_CTRL1_R.HIGH_SPEED_EN must be set. The procedure in 5.10.14.1 Card Clock Frequency Configuration describes how to change the card clock frequency.

Send command SWITCH (CMD6) to set the bus width. Verify that the transfer was completed by checking and clearing NORMAL_INT_STAT_R.XFER_COMPLETE. Then configure HOST_CTRL1_R.EXT_DAT_XFER and HOST_CTRL1_R.DAT_XFER_WIDTH accordingly.

Enable the desired mode in the host by configuring HOST_CTRL2_R.UHS_MODE_SEL.

5.10.14.8 Configuration of SD Speed and Bus Mode

After configuring the host (5.10.14.3 Host Setup for SD) and initializing the SD card (5.10.14.6 Initialization of SD Card), and before accessing the card, the speed and the bus mode need to be configured. The supported modes are listed in Table 5-73.

To set the bus width, send to the SD card the command APP_CMD (CMD55), followed by SET_BUS_WIDTH (ACMD6). Then configure HOST_CTRL1_R.DAT_XFER_WIDTH accordingly.

To set the speed, send the command SWITCH (CMD6) to the card. If mode SDR12 is chosen, then HOST_CTRL1_R.HIGH_SPEED_EN must be cleared. Otherwise, it must be set. Configure HOST_CTRL2_R.UHS_MODE_SEL with the desired mode and use the steps described in 5.10.14.1 Card Clock Frequency Configuration to set the card clock frequency.

5.10.14.9 Writing to the Card without DMA

To write to a card without DMA, perform the following steps.

1. To enable multiple block access in a single data transfer, send the command SET_BLOCK_COUNT (CMD23) specifying the block count.
2. Configure the block size and block count through BLOCKSIZE_R.XFER_BLOCK_SIZE and BLOCKCOUNT_R.BLOCK_CNT.
3. Send command WRITE_BLOCK (CMD24) specifying the sector address to write to for single block or WRITE_MULTIPLE_BLOCK (CMD25) for multiple blocks, using the procedure described in 5.10.14.4 Send Command to Card. When configuring the XFER_MODE_R, set XFER_MODE_R.BLOCK_COUNT_ENABLE to 1, XFER_MODE_R.DATA_XFER_DIR to 0 and XFER_MODE_R.DMA_ENABLE to 0. For multiple blocks also set XFER_MODE_R.MULTI_BLK_SEL to 1.
4. For each block repeat:
    4.1. Wait for NORMAL_INT_STAT_R.BUF_WR_READY.
    4.2. Clear NORMAL_INT_STAT_R.BUF_WR_READY by writing 1.
    4.3. Write word-by-word the block of data to BUF_DATA_R.BUF_DATA.
5. Verify that the transfer was completed by checking and clearing NORMAL_INT_STAT_R.XFER_COMPLETE.

5.10.14.10 Reading from the Card without DMA

To read from a card without DMA, perform the following steps.

1. To enable multiple block access in a single data transfer, send the command SET_BLOCK_COUNT (CMD23) specifying the block count.
2. Configure the block size and block count through BLOCKSIZE_R.XFER_BLOCK_SIZE and BLOCKCOUNT_R.BLOCK_CNT.
3. Send command READ_SINGLE_BLOCK (CMD17) specifying the sector address to read from (or READ_MULTIPLE_BLOCK CMD18 for multiple blocks), using the procedure described in 5.10.14.4 Send Command to Card. When configuring the XFER_MODE_R, set XFER_MODE_R.BLOCK_COUNT_ENABLE to 1, XFER_MODE_R.DATA_XFER_DIR to 1 and XFER_MODE_R.DMA_ENABLE to 0. For multiple blocks also set XFER_MODE_R.MULTI_BLK_SEL to 1.
4. For each block repeat:
    4.1. Wait for NORMAL_INT_STAT_R.BUF_RD_READY.
    4.2. Clear NORMAL_INT_STAT_R.BUF_RD_READY by writing 1.
    4.3. Read word-by-word one block of data from BUF_DATA_R.BUF_DATA.
5. Verify that the transfer was completed by checking and clearing NORMAL_INT_STAT_R.XFER_COMPLETE.

5.10.14.11 Memory Transfer Using SDMA

To transfer memory using SDMA, perform the following steps.

1. To enable multiple block access in a single data transfer, send the command SET_BLOCK_COUNT (CMD23) specifying the block count.
2. Set HOST_CTRL1_R.DMA_SEL to 0 to select SDMA mode.
3. Write to ADMA_SA_LOW_R.ADMA_SA_LOW and ADMA_SA_HIGH_R.ADMA_SA_HIGH the targeted address in memory.

4. Configure the block size and block count through BLOCKSIZE_R.XFER_BLOCK_SIZE and BLOCKCOUNT_R.BLOCK_CNT.
5. If reading from the card, send READ_SINGLE_BLOCK (CMD17) or READ_MULTIPLE_BLOCK(CMD18), using the procedure described in 5.10.14.4 Send Command to Card. When configuring the XFER_MODE_R, set XFER_MODE_R.BLOCK_COUNT_ENABLE to 1, XFER_MODE_R.DATA_XFER_DIR to 1 and XFER_MODE_R.DMA_ENABLE to 1. For multiple blocks, also set XFER_MODE_R.MULTI_BLK_SEL to 1.
6. If writing to the card, send WRITE_BLOCK (CMD24) or WRITE_MULTIPLE_BLOCK(CMD25), using the procedure described in 5.10.14.4 Send Command to Card. When configuring the XFER_MODE_R, set XFER_MODE_R.BLOCK_COUNT_ENABLE to 1, XFER_MODE_R.DATA_XFER_DIR to 0 and XFER_MODE_R.DMA_ENABLE to 1. For multiple blocks, also set XFER_MODE_R.MULTI_BLK_SEL to 1.
7. Repeat:
    7.1. Wait for NORMAL_INT_STAT_R.XFER_COMPLETE or NORMAL_INT_STAT_R.DMA_INTERRUPT
    7.2. If DMA_INTERRUPT is seen, update ADMA_SA_LOW_R.ADMA_SA_LOW for the next block and clear DMA_INTERRUPT.
    7.3. If XFER_COMPLETE is seen, clear XFER_COMPLETE and stop.

5.10.14.12 Memory Transfer Using ADMA2

To transfer memory using ADMA2, perform the following steps.

1. To enable multiple block access in a single data transfer, send the command SET_BLOCK_COUNT (CMD23) specifying the block count.
2. Set HOST_CTRL1_R.DMA_SEL to 2 to select ADMA2 mode.
3. Create Descriptor table for ADMA in system memory.
4. Write the address in system memory of the ADMA descriptor table to ADMA_SA_LOW_R.ADMA_SA_LOW and ADMA_SA_HIGH_R.ADMA_SA_HIGH.
5. Configure the block size and block count through BLOCKSIZE_R.XFER_BLOCK_SIZE and BLOCKCOUNT_R.BLOCK_CNT.
6. If reading from the card, send READ_SINGLE_BLOCK (CMD17) or READ_MULTIPLE_BLOCK(CMD18), using the procedure described in 5.10.14.4 Send Command to Card. When configuring the XFER_MODE_R, set XFER_MODE_R.BLOCK_COUNT_ENABLE to 0, XFER_MODE_R.DATA_XFER_DIR to 1 and XFER_MODE_R.DMA_ENABLE to 1. For multiple blocks also set XFER_MODE_R.MULTI_BLK_SEL to 1.
7. If writing to the card, send WRITE_BLOCK (CMD24) or WRITE_MULTIPLE_BLOCK(CMD25), using the procedure described in 5.10.14.4 Send Command to Card. When configuring the XFER_MODE_R, set XFER_MODE_R.BLOCK_COUNT_ENABLE to 0, XFER_MODE_R.DATA_XFER_DIR to 0 and XFER_MODE_R.DMA_ENABLE to 1. For multiple blocks also set XFER_MODE_R.MULTI_BLK_SEL to 1.
8. Verify that the transfer was completed by checking and clearing NORMAL_INT_STAT_R.XFER_COMPLETE.

5.10.14.13 Memory Transfer Using ADMA3

To transfer memory using ADMA3, perform the following steps.

1. Set HOST_CTRL1_R.DMA_SEL to 3.
2. Create Command Descriptors and ADMA2 Descriptors in system memory.
3. Create Integrated Descriptors in system memory.
4. Write Integrated DMA Descriptor address to ADMA_ID_LOW_R.ADMA_ID_LOW and ADMA_ID_HIGH_R.ADMA_ID_HIGH.
5. Verify that the transfer was completed by checking and clearing NORMAL_INT_STAT_R.XFER_COMPLETE.

5.11 VCore Sub-CPU System

The purpose of the VCore sub-CPU is to run the POE-manager software. Additionally, it may be used to perform chip initialization in SKUs where the master CPU system is not employed.

It consists of a small low-power processor with integrated interrupt controller and debug logic, along with a set of standard CPU peripherals and a dedicated standard SPI master module that is expected to be used for communicating with the POE controllers. For more information, see 5. VCore System and CPU Interfaces.

The sub-CPU system uses its own reset scheme and can run independently of the main VCore CPU system. A shared memory and an external interrupt from the main VCore CPU system towards the sub-CPU system are deployed, among others, to achieve communication between the two systems.

5.11.1 Features

The Vcore sub-CPU system operates at 100 MHz and features the following:

- ARM Cortex-M3 processor
  - Little endian
  - Memory protection unit
  - SWJ-DP (Serial wire and JTAG, full debug with data matching)
  - Integrated NVIC
- External interrupt input
- Peripherals
  - UART
  - I2C
  - 2 x SPI master controllers with single chip select
  - 3 x Timers
  - WDT
- 64KB Code memory serving both as instruction and data memory
- 16KB Shared memory
- Reset controller
- Booting sources
  - Code memory
  - SPI flash
- Accessibility
  - Main CPU system to sub-CPU system
  - Sub-CPU system to main CPU system

5.11.2 Interfaces

The VCore sub-CPU system bears the following interfaces.

- Master interface towards the VCore Shared Bus Access/Arbiter (). It provides access to Mirror SPI boot Flash and SPI boot Flash regions (), as well as to CPU Registers, FDMA Registers, and Switch Core Registers ().
- Slave interface to the VCore Shared Bus Access/Arbiter (). It provides access to the VCore sub-CPU system from the VCore CPU or from an external CPU.
- Debug interface. For more information on accessing the debug interface, see 5.12 JTAG Interface.

5.11.3 Clocking and Reset

The entire VCore sub-CPU system is clocked at 100 MHz and it incorporates its own reset controller. The controller receives hard and soft resets, both internal and external to the sub-CPU system, and asserts resets according to the configuration.

Some regions of the VCore sub-CPU system may be protected from certain reset requests. There are three reset-protect domains:

- subcpu: The VCore sub-CPU.
- wdt: The watchdog timer may be protected from a reset independently.
- fabric: The remaining VCore sub-CPU system. It includes the VCore Secondary Bus, along with all of the peripherals except for the WDT.

The following table lists the available resets, the configurable reset protection options, and the areas that are affected by each reset.

Table 5-75. VCore sub-CPU System Reset Scheme

Reset	Issue Domain	Type	Affects	Configurable Protection	Description
master_rst	global	pin	all	-	Chip reset, asserted at power-up.
system_rst	global	soft	all	all	Soft chip reset. The entire sub-CPU system may be protected.
poe_sys_rst_force	VCore CPU sys	soft	all	fabric	Soft force-reset that keeps the sub-CPU system under reset. The fabric domain may be protected. Typically used for loading the program into the code memory before releasing the sub-CPU.
poe_rst_req	sub-CPU sys	soft	all	fabric, wdt	Soft reset for the sub-CPU system, triggered by the sub-CPU software.
lock_rst_req	sub-CPU sys	hard	subcpu	subcpu	Hard reset due to sub-CPU entering locked-up state.
wdt_rst_req	sub-CPU sys	hard	all	fabric, wdt	Hard reset due to WDT timeout.
wdt_rst_force	sub-CPU sys	soft	wdt	-	Force reset on WDT. Used to disable the WDT.

The VCore sub-CPU reset scheme control and status registers are listed in the following table. For more information, refer to the register descriptions.

Table 5-76. Reset Control and Status Registers

Register	Description
DEVCPU_GCB::SOFT_RST	System-wide soft reset
CPU::SUBCPU_SYS_CFG	Clock gating and soft reset control from VCore CPU system towards VCore sub-CPU system.
SUBCPU_SYS_CFG::GENERAL_CTRL	VCore sub-CPU system soft resets.
SUBCPU_SYS_CFG::RESET_PROTECT	Configures reset protection scheme.
SUB_CPU_SYS_CFG::RESET_STAT	Reset-cause register.

5.11.3.1 Watchdog Timer

The VCore sub-CPU system SUBCPU_WDT is identical to 5.2.1 Watchdog Timer with the following differences:

- The clock frequency is 100 MHz
- It resets the VCore sub-CPU system by issuing `wdt_rst_req. See Table 5-75`.
- Reset force and reset protection are configurable through SUBCPU_SYS_CFG::GENERAL_CTRL and SUBCPU_SYS_CFG::RESET_PROTECT.

5.11.4 Enabling the VCore Sub-CPU System

The device may enable the VCore sub-CPU system upon power-up depending on the VCore strapping that is applied (Table 5-1). In all modes that the VCore sub-CPU is not set to boot, the VCore sub-CPU system is disabled.

To enable the VCore sub-CPU system do the following:

1. Force reset to sub-CPU system by setting CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE = 1 CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE_PROT_AMBA = 0.
2. Enable the clock by setting CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_DIS = 0.
3. If it is required to initialize the contents of the code memory, then:
    3.1. Release the VCore sub-CPU system from reset keeping only the VCore sub-CPU under reset by setting CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE_PROT_AMBA = 1.
    3.2. Initialize code memory. The mapping of the code memory to the VCore Shared Bus is shown in .
    3.3. Configure the VCore sub-CPU to boot from the code memory instead of the SPI Flash by setting SUBCPU_SYS_CFG::GENERAL_CTRL.BOOT_MODE_ENA = 0.
    3.4. Read back SUBCPU_SYS_CFG::GENERAL_CTRL.BOOT_MODE_ENA so that the configured value is applied.
4. Release reset by writing CPU::SUBCPU_SYS_CFG.SUBCPU_SYS_RST_FORCE = 0.

5.11.5 Shared Bus

The shared bus of the sub-CPU system (namely VCore Secondary Bus in Figure 5-1) is AMBA-based, incorporating an AHB and APB fabric (32-bit address and 32-bit data bus at 100 MHz). It has dedicated master and slave interfaces that interconnect all the blocks in the VCore sub-CPU system, as well as the main VCore CPU system. All the masters are connected to the VCore Secondary Bus Access/Arbiter block; only they can start accesses on the bus.

The shared bus uses byte addresses, and transfers of 8, 16, or 32 bits can be made. To increase performance, bursting of multiple words on the shared bus can be performed.

All slaves are mapped into the VCore sub-CPU address space and can be accessed directly by the VCore sub-CPU. Two possible mappings of the shared bus slaves are boot mode and normal mode.

- Boot mode: Boot mode is active after power-up and reset of the VCore sub-CPU system. In this mode, the SI Flash Boot Master is mirrored into the lowest address region.
- Normal mode: In normal mode, the code memory is mirrored into the lowest address region.

Changing between boot mode and normal mode is done by first writing and then reading SUBCPU_SYS_CFG::GENERAL_CTRL.BOOT_MODE_ENA. A change takes effect during the read.

Figure 5-34. Shared Bus Memory Map (VCore Secondary Bus)

		Normal Mode	Boot Mode
0x00000000	64 KB	Code memory	Mirror SPI Boot Flash(1)
0x00010000		reserved	reserved
0x20000000	64 KB	Code memory	Code memory
0x20010000		reserved	reserved
0x40000000	1 KB	SUBCPU_UART	SUBCPU_UART
0x40000400	1 KB	SUBCPU_TWI	SUBCPU_TWI
0x40000800	1 KB	SUBCPU_SIMC	SUBCPU_SIMC
0x40000C00	1 KB	SUBCPU_TIMERS	SUBCPU_TIMERS
0x40001000	1 KB	SUBCPU_WDT	SUBCPU_WDT
0x40001400	1 KB	SUBCPU_SIMC2	SUBCPU_SIMC2
0x40001800		reserved	reserved
0xA0000000	16 KB	Shared memory	Shared memory
0xA0008000		reserved	reserved
0xA1000000	64 KB	SUBCPU_SYS_CFG	SUBCPU_SYS_CFG
0xA1010000		reserved	reserved
0xB0000000		CPU Registers(1)	CPU Registers(1)
0xB0080000		FDMA Registers(1)	FDMA Registers(1)
0xC0000000		Switch Core Registers(1)	Switch Core Registers(1)
0xD0000000	256 MB	SPI Boot Flash(1)	SPI Boot Flash(1)
0xE0000000	1 MB	Cortex-M3 regs	Cortex-M3 regs
0xE0100000		reserved	reserved
0xFFFFFFFF			

1. Access towards the main VCore CPU system.

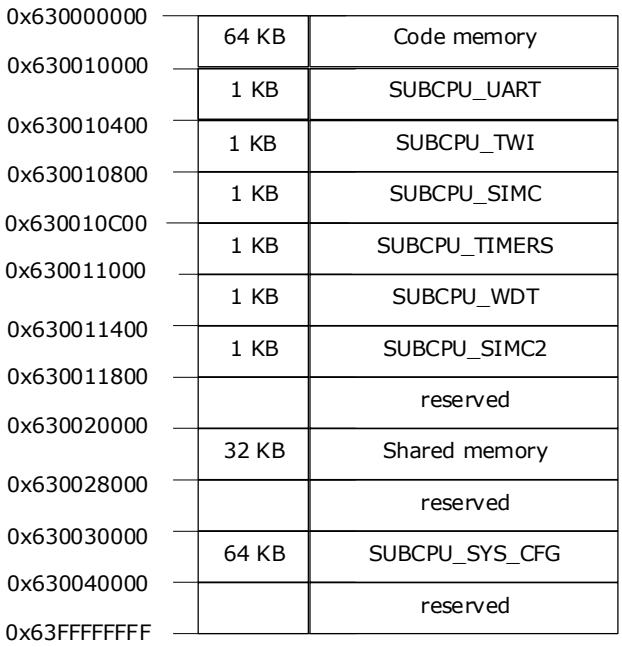
Note:
When the fabric reset protection domain is active, a soft reset will not change shared bus memory mapping. For more information about resets and protection, see 5.11.3 Clocking and Reset.

If the boot process copies the SI Flash image to Code memory, and if the contents of the SI memory and the Code memory are the same, software can execute from the mirrored region when swapping from boot mode to normal mode. Otherwise, software must be executing from the fixed SI Flash region when changing from boot mode to normal mode.

5.11.6 VCore Sub-CPU system Mapping to VCore CPU System Memory Map

Certain regions of the sub-CPU system shared bus are mapped also to the main VCore Shared Bus (Sub-CPU System in 5.3 Shared Bus). This mapping is shown in the following figure.

Figure 5-35. Mapping of VCore sub-CPU system to VCore Shared Bus



5.11.7 Interrupts

The VCore sub-CPU has an integrated interrupt controller. The user is referred to the technical reference manual of the VCore sub-CPU for information regarding the usage of the integrated interrupt controller. The interrupt sources are listed in the following table.

Table 5-77. Interrupt Sources

Source Name	Index	Description
SUBCPU_UART	0	UART interrupt. See 5.10.5.1 UART Interrupt.
SUBCPU_TWI	1	TWI interrupt. See 5.10.6.3 Two-Wire Serial Interface Interrupt.
SUBCPU_SIMC	2	Serial Master Controller interrupt. See 5.10.2.3 SIMC Interrupts.
SUBCPU_SIMC2	3	Serial Master Controller interrupt. See 5.10.2.3 SIMC Interrupts.
SUBCPU_TIMER0	4	Timer 0 interrupt. See 5.10.4 Timers.
SUBCPU_TIMER1	5	Timer 1 interrupt. See 5.10.4 Timers.
SUBCPU_TIMER2	6	Timer 2 interrupt. See 5.10.4 Timers.
SUBCPU_WDT	7	Watchdog timer interrupt. See 5.2.1 Watchdog Timer.
CPU[9:0]	17:8	Software interrupts CPU::SHARED_INTR.
KR	50:18	KR interrupts from SerDes modules.
PCIE	51	PCIe interrupt. See 5.6.7 Power Management.

Under certain error conditions the VCore sub-CPU may enter a lock-state. The lock state is communicated to the main VCore CPU system through an outbound interrupt.

5.11.8 VCore sub-CPU system Peripherals

The peripherals of the sub-CPU system are mostly identical to the respective peripherals of the main VCore CPU system.

5.11.8.1 Timers

The VCore sub-CPU system SUBCPU_TIMERS are identical to the VCore CPU system TIMERS. The clock frequency is 100MHz. For more information, see 5.10.4 Timers.

5.11.8.2 UART

The VCore sub-CPU system SUBCPU_UART is identical to the VCore CPU system. See 5.10.5 UARTs. The clock frequency is 100 MHz. The interface is mapped on GPIOs as UART3. To enable the GPIO Interface mode for UART3 write SUBCPU_SYS_CFG::GENERAL_CTRL.UART_GPIOS_ENA to 1. For more information, see 5.10.8.1 GPIO Overlaid Functions.

5.11.8.3 Two-Wire Serial Interface

The VCore sub-CPU system SUBCPU_TWI is similar to the VCore CPU system 5.10.6 Two-Wire Serial Interface, with the following differences.

- The clock frequency is 100 MHz.
- There is no built-in serial multiplexer.
- The interface is mapped as TWI3 on GPIOs. To enable the TWI3 on GPIOs Interface mode write SUBCPU_SYS_CFG::GENERAL_CTRL.I2C_GPIOS_ENA to 1.
- The extra configuration of SDA hold delay and spike filter is done through SUBCPU_SYS_CFG::TWI_CONFIG and SUBCPU_SYS_CFG::TWI_SPIKE_FILTER_CFG.

5.11.8.4 SI Master Controllers

The VCore sub-CPU system SI Master Controllers (SUBCPU_SIMC, SUBCPU_SIMC2) are similar to the VCore CPU system. See 5.10.2 SI Master Controller, with the following differences.

- The clock frequency is 100 MHz.
- There is only one chip select.
- The interfaces of SUBCPU_SIMC and SUBCPU_SIMC2 are mapped to GPIO interfaces SPI3 and SPI4 and take ownership automatically. For more information, see 5.10.8.1 GPIO Overlaid Functions.
- SPI_nCS[7:5] on GPIOs can be used by the SIMC controllers. The mapping of the controller's single chip select to any of the SPI_nCS[7:5] is done by writing to SUBCPU_SYS_CFG::SS_MASK. For more information, see 5.10.3 SI Slave-Select as a Shared Resource.
- Software can override the chip select value through SUBCPU_SUS_CFG::SS_FORCE and SUBCPU_SYS_CFG::SS_FORCE_ENA.

5.12 JTAG Interface

This section describes the multiplexing implemented for routing the chip-level JTAG interface to access either the CPU JTAG Debug Port (JTAG-DP), sub-CPU JTAG Debug Port (JTAG-DP), DDR4 multiPHY PUB configuration registers, or scan chain tap controller.

JTAG_SEL0 and JTAG_SEL1 chip-level input pins control the mapping of the chip-level JTAG interface as shown in the following table.

Table 5-78. Mapping of the Chip-Level JTAG Interface

JTAG_SEL0	JTAG_SEL1	JTAG Target
0	0	Accessing DDR4 multiPHY PUB configuration registers
0	1	Accessing sub-CPU JTAG-DP

.....continued		
JTAG_SEL0	**JTAG_SEL1**	**JTAG Target**
1	0	Accessing CPU JTAG-DP
1	1	Scan chain

The following table lists the JTAG interface pins.

Table 5-79. JTAG Interface Psins

Pin Name	I/O	Description
JTAG_TCK	inout	Test Clock Input: The test clock input (TCK) provides the clock for the test logic
JTAG_TDI	inout	Test Data Input: Serial test instructions and data are received by the test logic at TDI.
JTAG_TDO	inout	Test Data Output: TDO is the serial output for test instructions and data from the test logic
JTAG_TMS	inout	Test Mode Select: The signal received at TMS is decoded by the TAP controller to control test operations.
JTAG_NTRST	inout	Test Reset: The optional TRST* input provides for asynchronous initialization of the TAP controller
JTAG_SEL0	inout	Select Input for top-level JTAG MUX
JTAG_SEL1	inout	Select Input for top-level JTAG MUX

6. Registers

Information about the registers for this product is available at github.com/microchip-ung/sparx-5_reginfo. To view or print the information, navigate to the website.

The registers are common to the VSC7546 and VSC7549 devices. Registers not applicable to a specific device are marked accordingly.

7. Electrical Specifications

This section provides the DC characteristics, AC characteristics, recommended operating conditions, and stress ratings for the VSC7546 and VSC7549 SparX-5 devices.

7.1 DC Characteristics

This section contains the DC specifications for the devices.

7.1.1 Internal Pull-Up or Pull-Down Resistors

Internal pull-up or pull-down resistors are specified in the following table. For more information about signals with internal pull-up or pull-down resistors, see 8.2 Pins by Function for the specific device.

All internal pull-up resistors are connected to their respective I/O supply.

Table 7-1. Internal Pull-Up or Pull-Down Resistors

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Internal pull-up resistor	R_{PU}	26	46	71	kΩ
Internal pull-down resistor	R_{PD}	27	48	102	kΩ

7.1.2 SerDes Reference Clock Input

The following table lists the DC specifications for the SerDes reference clock input signals, such as REFCLK1, REFCLK2, and REFCLK3.

Table 7-2. SerDes Reference Clock Inputs

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Clock source output DC impedance	Z_{C-DC}	40	50	60	Ω
Input DC differential termination[1]	$\|R_I\|$	80		100	Ω
Differential peak-to-peak input swing	$\|V_{ID}\|$	600	—	1600	mVppd
Input common-mode voltage	V_{CM}	—	900	—	mV
Single-ended input voltage	V_{SE}	–0.35	—	1.1	V

Clock signal must be AC coupled.

Note:

1. The REFCLK2 input will have a lower input impedance until the ports have been configured (approximately 11 ohm).

7.1.3 Core Reference Clock Input

The following table lists the DC specifications for the core reference clock input, REFCLK0.

Table 7-3. Core Reference Clock Input

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Input voltage range	V_{IP}, V_{IN}	–25	—	1800	mV	—
Single-ended input swing	$\|V_{SE}\|$	100	—	1800[1]	mV	—
Differential peak-to-peak input swing	$\|V_{ID}\|$	200	—	1600	mVppd	—
Input common-mode voltage	V_{CM}	—	0.39 × V_{DD18}	—	mV	—

Note:

1. Input common-mode voltage and amplitude must not exceed V_{DD18}.

7.1.4 DDR SDRAM Interface

This section provides the DC specifications for the DDR4, DDR3, and DDR3L interfaces.

The DDR3 SDRAM interface supports the requirements of SDRAM devices as described in the JEDEC DDR3 specifications. The SDRAM interface signals are compatible with JESD79-3F (DDR3 SDRAM SPECIFICATION, July 2012) and JESD79-3-1A, January 2013. The SSTL I/O buffers have programmable on-die termination (ODT).

7.1.4.1 DDR4 SDRAM Interface

The following table lists the DC specifications for the SDRAM interface signals in DDR4 operation.

Table 7-4. DDR4 SDRAM Signals

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Input reference voltage[1]	DDR_V_{REF}	49% V_{DD_IODDR}	51% V_{DD_IODDR}	V	V_{DD_IODDR} = 1.2 V.
Input voltage high	$V_{IH(DC)}$	0.7 × V_{DD_IODDR}	—	V	—
Input voltage low	$V_{IL(DC)}$	—	0.3 × V_{DD_IODDR}	V	—
Output voltage high	$V_{OH(DC)}$	0.9 × V_{DD_IODDR}	—	V	Not terminated, 1 pF load.
Output voltage low	$V_{OL(DC)}$	—	0.1 × V_{DD_IODDR}	V	Not terminated, 1 pF load.
Input leakage current	$\|I_L\|$	—	2.1	µA	0 V ≤ VI ≤ V_{DD_IODDR}.
Output sink DC current	I_{OL}	17	—	mA	External 34 Ω termination to V_{DD_IODDR}.

Note:

1. DDR_V_{REF} is expected to track variations in V_{DD_IODDR}. Peak-to-peak AC noise on DDR_V_{REF} must not exceed ±1 % of DDR_V_{REF}.

7.1.4.2 DDR3 SDRAM Interface

The following table lists the DC specifications for the SDRAM interface signals in DDR3 operation.

Table 7-5. DDR3 SDRAM Signals

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Input reference voltage [1]	DDR_VREF	49% V_{DD_IODDR}	51% V_{DD_IODDR}	V	V_{DD_IODDR} = 1.5 V.
Input voltage high	$V_{IH(DC)}$	DDR_VREF + 0.10	V_{DD_IODDR}	V	—
Input voltage low	$V_{IL(DC)}$	–0.3	DDR_VREF – 0.10	V	—
Output voltage high	$V_{OH(DC)}$	0.8 × V_{DD_IODDR}	—	V	Not terminated, 1 pF load.
Output voltage low	$V_{OL(DC)}$	–0.3	0.2 × V_{DD_IODDR}	V	Not terminated, 1 pF load.
Input leakage current	$\|I_L\|$	—	2.1	µA	0 V ≤ VI ≤ V_{DD_IODDR}.
Output source DC current [2]	I_{OH}	–6	—	mA	External 40 Ω termination to VDD_IODDR/2.
Output sink DC current[2]	I_{OL}	6	—	mA	External 40 Ω termination to VDD_IODDR/2.

Notes:

1. DDR_V_{REF} is expected to track variations in V_{DD_IODDR}. Peak-to-peak AC noise on DDR_V_{REF} must not exceed ±2% of DDR_V_{REF}.
2. With 34 Ω output driver impedance.

7.1.4.3 DDR3L SDRAM Interface

The following table lists the DC specifications for the SDRAM interface signals in DDR3L operation.

Table 7-6. DDR3L SDRAM Signals

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Input reference voltage [1]	DDR_V_{REF}	49% V_{DD_IODDR}	51% V_{DD_IODDR}	V	V_{DD_IODDR} = 1.35 V.
Input voltage high	$V_{IH(DC)}$	DDR_VREF + 0.09	VDD_IODDR	V	—
Input voltage low	$V_{IL(DC)}$	–0.3	DDR_VREF –0.09	V	—
Output voltage high	$V_{OH(DC)}$	0.8 × V_{DD_IODDR}	—	V	Not terminated, 1 pF load
Output voltage low	$V_{OL(DC)}$	—	0.2 × VDD_IODDR	V	Not terminated, 1 pF load.
Input leakage current	$\|I_L\|$	—	2.1	µA	0 V ≤ VI ≤ V_{DD_IODDR}.
Output source DC current [2]	I_{OH}	–6	—	mA	External 40 Ω termination to V_{DD_IODDR}/2.
Output sink DC current [2]	I_{OL}	6	—	mA	External 40 Ω termination to V_{DD_IODDR}/2.

Notes:

1. DDR_V_{REF} is expected to track variations in V_{DD_IODDR}. Peak-to-peak AC noise on DDR_V_{REF} must not exceed ±2% of DDR_V_{REF}.
2. With 34 Ω output driver impedance.

7.1.5 SERDES6G/10G

This section provides the DC specifications for the 10G transceiver. The transceiver supports the following modes on the S13 to S24 port.

- 100BASE-FX
- SGMII
- SFP
- 1000BASE-KX
- 2.5GBASE-KX
- QSGMII
- 5GBASE-KR
- 10GBASE-KR
- SFP+ (SFI)
- USXGMII

The transceiver supports the following modes on S0 to S12:

- 100BASE-FX
- SGMII
- SFP
- 1000BASE-KX
- 2.5GBASE-KX
- 5GBASE-KR

The following table lists the 10G transmitter specifications.

Table 7-7. 10G Transmitter

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	—
Differential peak-to-peak output voltage [1]	VO_{DIFF}	300	—	800	mVppd	100BASE-FX, SFP, SGMII, QSGMII
Differential peak-to-peak output voltage[1]	VO_{DIFF}	800 [2]	—	1200	mVppd	10GBASE-KR, 1000BASE-KX, 2.5GBASE-KX, 5GBASE-KR, USXGMII
Differential peak-to-peak output voltage with Tx disabled	V_{OD_IDLE}	—	—	30	mVppd	—

Notes:

1. Differential output swing is register configurable.
2. The minimum drive level is the lowest guaranteed drive level achievable with the maximum amplitude configuration applied.

The following table lists the 10G receiver specifications.

Table 7-8. 10G Receiver

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	—
Differential peak-to-peak input voltage range	V_{I_DIFF}	100	—	1200	mVppd	Clean eye sensitivity
AC-coupling[1]	—	—	—	100	nF	—
Single-ended input voltage	V_{SE}	–0.35	—	1.1	V	—

Note:

1. AC-coupling should be done at receiver.

7.1.6 SERDES25G

This section provides the DC specifications for the 25G transceiver. The transceiver supports the following modes:

- SGMII
- SFP
- 1000BASE-KX
- 2.5GBASE-KX
- 5GBASE-KR
- 10GBASE-KR
- SFP+ (SFI)

The following table lists the 25G transmitter specifications.

Table 7-9. 25G Transmitter

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	—
Differential peak-to-peak output voltage [1]	V_{O_DIFF}	300	—	800	mVppd	SFP, SGMII, QSGMII

.....continued						
Parameter	**Symbol**	**Minimum**	**Typical**	**Maximum**	**Unit**	**Condition**
Differential peak-to-peak output voltage [1]	V_{O_DIFF}	800[2]	—	1200	mVppd	10GBASE-KR, 1000BASE-KX, 2.5GBASE-KX, 5GBASE-KR, USXGMII
Differential peak-to-peak output voltage with Tx disabled	V_{OD_IDLE}	—	—	30	mVppd	—

Notes:

1. Differential output swing is register configurable.
2. The minimum drive level is the lowest guaranteed drive level achievable with the maximum amplitude configuration applied.

The following table lists the 25G receiver specifications.

Table 7-10. 25G Receiver

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	—
Differential peak-to-peak input voltage range	V_{I_DIFF}	100	—	1200	mVppd	Clean eye sensitivity
Electrical idle detection threshold	V_{E_IDLE}	65	—	175	mVppd	—
AC-coupling [1]			—	100	nF	—
Single-ended input voltage	V_{SE}	–0.35	—	1.1	V	—

Note:

1. AC-coupling should be done at receiver.

7.1.7 PCIe

This section provides the DC specifications for the PCIe transceiver. The following table lists the PCIe transmitter specifications.

Table 7-11. PCIe Transmitter

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	—
Differential peak-to-peak output voltage[1]	V_{O_DIFF}	800 [2]	—	1300	mVppd	No EQ.
Differential output voltage during EIEOS in full swing mode	$V_{TX\text{-}EIEOS\text{-}FS}$	250	—	—	mVppd	8G mode.
Differential output voltage during EIEOS in reduced swing mode	$V_{TX\text{-}EIEOS\text{-}RS}$	232	—	—	mVppd	8G mode.
DC Common mode voltage	$V_{TX\_DC\_CM}$	0	—	1.0	V	—
Differential DC output voltage with Tx disabled	$V_{TX\text{-}IDLE\text{-}DIFF\text{-}DC}$	—	—	5	mV	5G and 8G mode.
TX Receiver Detect voltage	$V_{TX\_RCV\_DETECT}$	—	—	600	mV	—

.....continued						
Parameter	**Symbol**	**Minimum**	**Typical**	**Maximum**	**Unit**	**Condition**
Change in Common mode Voltage	$V_{TX\_RCV\_DETECT}$	—	—	100	mV	—
Common mode Voltage P/N difference	$V_{TX\_CM\_DC\_LINE\_DELTA}$	—	—	25	mV	P and N output.

Notes:

1. Differential output swing is register configurable.
2. The minimum drive level is the lowest guaranteed drive level achievable with the maximum amplitude configuration applied.

The following table lists the PCIe receiver specifications.

Table 7-12. PCIe Receiver

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	—
DC single-ended resistance	Z_{RX-DC}	40	50	60	Ω	—
DC Common mode impedance during reset and power down - positive	$Z_{RX\_HIGH\_IMP\_DC\_POS}$	10k	—	—	Ω	Positive voltage applied.
DC Common mode impedance during reset and power down - negative	$Z_{RX\_HIGH\_IMP\_DC\_NEG}$	1k	—	—	Ω	Negative voltage applied.
Differential peak-to-peak input voltage range	V_{I_DIFF}	175	—	1200	mVppd	2.5G mode.
Differential peak-to-peak input voltage range	V_{I_DIFF}	120	—	1200	mVppd	5G mode.
Differential peak-to-peak input voltage range	V_{I_DIFF}	—	—	1200	mVppd	8G mode.
Single-ended input voltage	V_{SE}	–0.35	—	1.1	V	—
AC-coupling[1]	—	75	—	265	nF	—

Note:

1. AC-coupling should be done at receiver.

7.1.8 GPIO, MIM, SI, JTAG, and Miscellaneous Signals

This section provides the DC specifications for the GPIO, MIM, SI, JTAG, and miscellaneous signals.

The following I/O signals comply with the specifications provided in this section:

Table 7-13. I/O Signals

GPIO[63:0]	JTAG_nTRST	JTAG_ICE_nEN
SI_CLK	JTAG_TMS	nRESET
SI_DI	JTAG_TDO	—
SI_DO	JTAG_TCK	—

SI_nCS0	JTAG_TDI	MDIO0 MDC0 JTAG_SEL0 JTAG_SEL1 JTAG_CPU_nRST

The outputs and inputs meet or exceed the requirements of the LVTTL and LVCMOS standard, JEDEC JESD8-B (September 1999) standard, unless otherwise stated. The inputs are Schmitt-trigger for noise immunity.

Table 7-14. GPIO, MIM, SI, JTAG, and Miscellaneous Signals DC Specifications

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Output high voltage[1]	V_{OH}	2.4	—	V	I_{OH}[3] = –10 mA
Output low voltage[1]	V_{OL}		0.4	V	I_{OL}[4] = 10 mA
Output DC current	I_O	–12	12	mA	—
Input high voltage	V_{IH}	2.0	3.47	V	—
Input low voltage	V_{IL}	–0.3	0.8	V	—
Input high current[2]	I_{IH}	–10	10	µA	$V_I = V_{DD\_IO33}$
Input low current[2]	I_{IL}	–10	10	µA	V_I = 0 V

Notes:

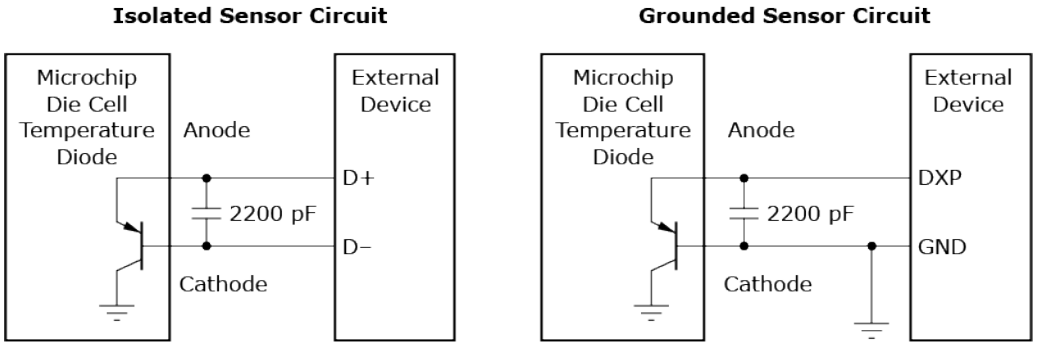
1. GPIOs have programmable drive strength. Values apply when using drive strength two or higher.
2. Input high current and input low current equals the maximum leakage current, excluding the current in the built-in pull resistors.
3. For MDIO0, SI_D0, SI_D1, SI_nCS0, and JTAG_TDO the I_{OH} is -6 mA.
4. For MDIO0, SI_D0, SI_D1, SI_nCS0, and JTAG_TDO the I_{OL} is 6 mA.

7.1.9 Thermal Diode

The devices include an on-die diode and internal circuitry for monitoring die temperature (junction temperature). The operation and accuracy of the diode is not guaranteed and should only be used as a reference.

The on-die thermal diode requires an external thermal sensor, located on the board or in a stand-alone measurement kit. Temperature measurement using a thermal diode is very sensitive to noise. The following illustration shows a generic application design.

Figure 7-1. Thermal Diode



Note:
Microchip does not support or recommend operation of the thermal diode under reverse bias.

The following table provides the diode parameter and interface specifications with the pins connected internally to VSS in the device.

Table 7-15. Thermal Diode Parameters

Parameter	Symbol	Typical	Maximum	Unit
Forward bias current	I_{FW}	See note [1]	1	mA
Diode ideality factor	n	1.013	—	

Note:

1. Typical value is device dependent.

The ideality factor, n, represents the deviation from ideal diode behavior as exemplified by the following diode equation:

$I_{FW} = I_S (e^{(qV)/(nkT)}-1)$

where, I_S = saturation current, q = electron charge, V_D = voltage across the diode, k = Boltzmann constant, and T = absolute temperature (Kelvin).

7.2 AC Characteristics

This section provides the AC specifications for the VSC7546 and VSC7549 SparX-5 devices. All SerDes inputs and outputs should be AC-coupled and work in differential mode.

7.2.1 Reference Clock

The signal applied to the REFCLK differential inputs must comply with the requirements listed in the following tables at the pin of the devices.

The following table lists the AC specifications for the REFCLK0 reference clock.

Table 7-16. REFCLK0 Reference Clock

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
REFCLK frequency REFCLK_SEL = 000	*f*	–100 ppm	125	2000 ppm	MHz	—
REFCLK frequency REFCLK_SEL = 001	*f*	–100 ppm	156.25	2000 ppm	MHz	—
REFCLK frequency REFCLK_SEL = 010	*f*	–100 ppm	250	2000 ppm	MHz	—
REFCLK frequency REFCLK_SEL = 100	*f*	–100 ppm	25	2000 ppm	MHz	—
Clock duty cycle		40	—	60	%	Measured at 50% threshold.
Rise time and fall time	t_R, t_F		—	0.5	ns	Within ±200 mV relative to common mode.
REFCLK input peak-to-peak jitter, bandwidth from 2.5 kHz to 10 MHz [1]	—	—	—	20	ps	—

Note:

1. Peak-to-peak values are typically higher than the RMS value by a factor of 10 to 14.

The following table lists the AC specifications for the REFCLK1 and REFCLK2 SerDes reference clocks.

Table 7-17. REFCLK1/REFCLK2 SerDes Reference Clocks

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
REFCLK frequency	*f*	–100 ppm	156.25	100 ppm	MHz	—
Clock duty cycle	—	40	—	60	%	Measured at 50% threshold.
Rise time and fall time	t_R, t_F	1	—	4	V/ns	Differential ±250 mV threshold.
Rise/fall matching	t_{Diff}	—	—	10	%	Single-ended
Random Jitter	R_J	—	—	1.0	ps, rms%	From 50 kHz to 10 MHz bandwidth.
Jitter transfer from REFCLK to SerDes outputs, bandwidth from 10 kHz to 1 MHz	—	—	—	0.3	dB	—
Jitter transfer from REFCLK to SerDes outputs, bandwidth from 1 MHz to 10 MHz	—	—	—	4	dB	—
Jitter transfer from REFCLK to SerDes outputs, bandwidth above 10 MHz	—	—	—	2 – 20 × log (*f*/ 10 MHz)	dB	—

.....continued

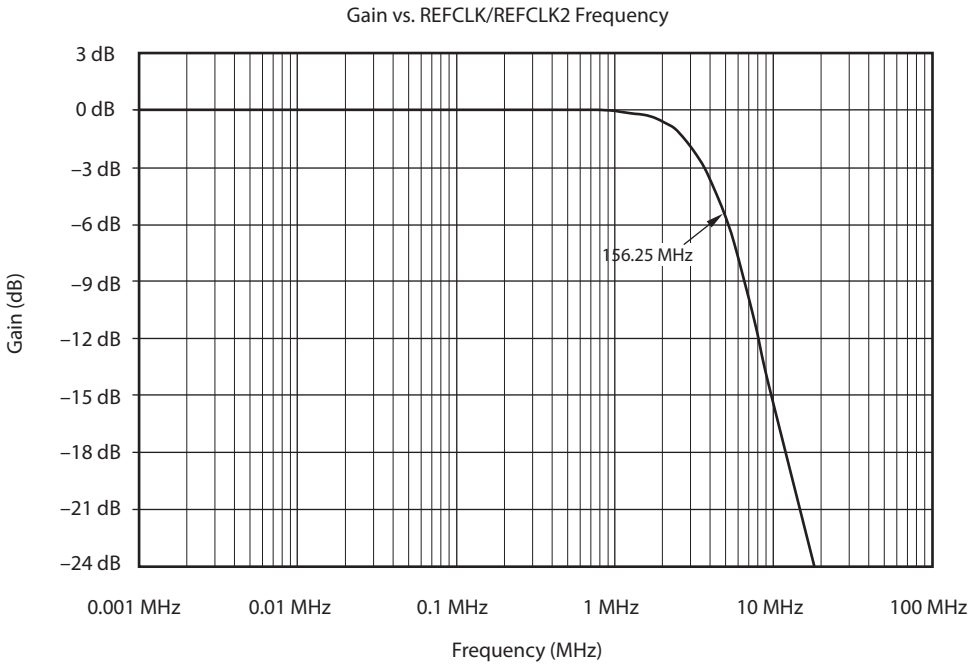
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
REFCLK input peak-to-peak jitter, bandwidth from 2.5 kHz and 10 MHz[1]	—	—	—	20	ps	To meet G.8262 1G SyncE jitter generation specification.
REFCLK input peak-to-peak jitter, bandwidth from 20 kHz and 20 MHz[1]	—	—	—	4	ps	To meet G.8262 10G SyncE jitter generation specification.
REFCLK input peak-to-peak jitter, bandwidth from 20 kHz and 20 MHz[1]	—	—	—	4	ps	To meet G.8262 25G SyncE jitter generation specification.

Note:

1. Peak-to-peak values are typically higher than the RMS value by a factor of 10 to 14.

The following illustration shows jitter transfer curves from REFCLK1 and REFCLK2 to all high-speed outputs.

Figure 7-2. REFCLK1/REFCLK2 Jitter Transfer Curves



The following table lists the AC specifications for the REFCLK3 PCIe reference clocks.

Table 7-18. REFCLK3 PCIe Reference Clocks

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
REFCLK3 frequency [1]	f	–300 ppm	100	300 ppm	MHz	—
Clock duty cycle	—	40		60	%	Measured at 50% threshold.
Differential clock rise and fall transitions	t_R, t_F	1		4	V/ns	Differential ±250 mV threshold.
Rise/fall matching	t_{Diff}	—	—	10	%	Single-ended
Random Jitter	R_J	—	—	1.0	ps, rms%	From 50 kHz to 10 MHz bandwidth.
Jitter transfer from REFCLK to SerDes outputs, bandwidth from 10 kHz to 1 MHz	—	—	—	0.3	dB	—
Jitter transfer from REFCLK to SerDes outputs, bandwidth from 1 MHz to 16 MHz	—	—	—	1	dB	5G PCIe mode.
Jitter transfer from REFCLK to SerDes outputs, bandwidth above 16 MHz	—	—	—	1 – 20 × log (f/16 MHz)	dB	5G PCIe mode.

.....continued						
Parameter	**Symbol**	**Minimum**	**Typical**	**Maximum**	**Unit**	**Condition**
Jitter transfer from REFCLK to SerDes output, bandwidth from 4 MHz to 5 MHz.	—	—	—	1	ps	8G PCIe mode.
Jitter transfer from REFCLK to SerDes output, bandwidth above 5 MHz	—	—	—	1 – 20 × log (*ƒ*/ 5 MHz)	ps	8G PCIe mode.
SSC frequency range	FSSC	30	—	33	kHz	—
SSC frequency deviation	FSSC-FREQ-DEVIATION	–5000	—	0	ppm	—

Note:

1. Excluding the spread spectrum clocking.

7.2.2 SerDes

This section describes the AC specifications for the SerDes transceiver. The transceiver supports 100BASE-FX, SFP, 1000BASE-KX, and SGMII modes.

The following table lists the AC characteristics for the SerDes transmitter.

Table 7-19. 100BASE-FX, SGMII, SFP, 1000BASE-KX, 2.5G, 2.5GBASE-KX, 5GBASE-KR Transmitter

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate	—	125 – 100 ppm	125 + 100 ppm	Mbps	100BASE-FX (IEEE 802.3, clause 26).
Data rate	—	1.25 – 100 ppm	1.25 + 100 ppm	Gbps	SGMII (Cisco), SFP (SFP-MSA), 1000BASE-KX (IEEE 802.3, clause 70).
Data rate	—	3.125 – 100 ppm	3.125 + 100 ppm	Gbps	2.5G, 2.5GBASE-KX (IEEE 802.3 clause 128).
Data rate	—	5.15625 – 100 ppm	5.15625 + 100 ppm	Gbps	5GBASE-KR (IEEE 802.3, clause 130), 5G-USXGMII
Differential output return loss	RLO_{SDD22}	—	–10 –10 + 10 x log (*ƒ*/625 MHz)	dB	1000BASE-KX 50 MHz to 625 MHz 625 MHz to 1250 MHz.
Differential output return loss	RLO_{SDD22}	—	–10 –10 + 10 x log (*ƒ*/625 MHz)	dB	2.5GBASE-KX: 100 MHz to 625 MHz 625 MHz to 2000 MHz.
Differential output return loss	RLO_{SDD22}	—	–10 –10 + 13.275 x log (*ƒ*/1289 MHz)	dB	5GBASE-KR 100 MHz ≤ f < 1289 MHz 1289 MHz ≤ f < 3750 MHz.

.....continued

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Differential output return loss	RLO_{SDD22}	—	–9 –9 + 12 x log (ƒ/2500 MHz)	dB	5G-USXGMII (derived from clause 72, 10GBase-KR) 50 MHz ≤ f < 2500 MHz 2500 MHz ≤ f ≤ 3750 MHz.
Common-mode output return loss	RLO_{SCC22}	—	–7 + 13.275 x log (ƒ/625 MHz)	dB	2.5GBASE-KX: 100 MHz to 625 MHz 625 MHz to 2000 MHz
Common-mode output return loss	RLO_{SCC22}	—	–7 + 13.275 x log (ƒ/1250 MHz)	dB	5GBASE-KR: 100 MHz to 1250 MHz 1250 MHz to 3750 MHz
Rise time and fall time[1]	t_R, t_F	60	320	ps	20% to 80%, 1000BASE-KX.
Rise time and fall time	t_R, t_F	30	100	ps	20% to 80%, 2.5GBASE-KX, QSGMII.
Rise time and fall time	t_R, t_F	20	60	ps	20% to 80%, 5GBASE-KR
Rise time and fall time	t_R, t_F	24	47	ps	20% to 80%, 5G-USXGMII.
Interpair skew	t_{SKEW}	—	20	ps	
Random jitter	R_J	—	0.15	UIPP	At BER 10–12, 1000BASE-KX, 5GBASE-KR, 5G-USXGMII.
Random jitter	R_J	—	0.2	UI_{PP}	At BER 10–12, 102.5GBASE-KX .
Deterministic jitter	D_J	—	0.10	UI_{PP}	At BER 10–12, 1000BASE-KX.
Deterministic jitter	D_J	—	0.12	UI_{PP}	Duty Cyle distortion of 0.035UI is considered part of DJ, 5G-USXGMII.
Deterministic jitter	D_J	—	0.15	UI_{PP}	Duty Cyle distortion of 0.035UI is considered part of DJ, 5G-USXGMII.
Total jitter	T_J	—	0.25	UI_{PP}	At BER 10–12, 1000BASE-KX.
Total jitter	T_J	—	0.28	UI_{PP}	At BER 10–12, 5G-USXGMII.
Total jitter	T_J	—	0.32	UI_{PP}	At BER 10–12, 2.5GBASE-KX .
Total jitter	T_J	—	0.3	UI_{PP}	At BER 10–12, 5GBASE-KR .
Wideband SyncE jitter	WJT	—	0.5	UI_{PP}	Measured according to ITU-T G.8262, section 8.3.
Eye mask	X1	—	0.125	UI	
Eye mask	X2	—	0.325	UI	
Eye mask	Y1	350		mV	
Eye mask	Y2	—	800	mV	

Note:

1. Slew rate is programmable.

The following table lists AC characteristics for the SerDes receiver.

Table 7-20. 100BASE-FX, SGMII, SFP, 1000BASE-KX 2.5G, 2.5GBASE-KX, 5GBASE-KR Receiver

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate	—	125 – 100 ppm	125 + 100 ppm	Mbps	100BASE-FX.
Data rate	—	1.25 – 100 ppm	1.25 + 100 ppm	Gbps	SGMII, SFP, 1000BASE-KX.
Data rate	—	3.125 – 100 ppm	3.125 + 100 ppm	Gbps	2.5GBASE-KX.
Data rate	—	5.15625 – 100 ppm	5.15625 + 100 ppm	Gbps	5GBASE-KR.
Differential input return loss	RLI_{SDD11}	—	–10	dB	50 MHz to 1289 MHz.
Differential input return loss	RLI_{SDD11}	—	–10 + 13.275 x log (f/ 1289 MHz)	dB	1289 MHz to 3750 MHz.
Common-mode input return loss	RLO_{SCC11}	—	–7 + 13.275 x log (f/ 1250 MHz)	dB	100 MHz to 1250 MHz 1250 MHz to 3750 MHz
Jitter tolerance, total [1]	TOL_{TJ}	600	—	ps	1G mode measured according to IEEE 802.3 Clause 38.6.8.
Jitter tolerance, deterministic[1]	TOL_{DJ}	370	—	ps	1G mode measured according to IEEE 802.3 Clause 38.6.8.
Jitter tolerance, duty cycle distortion	TOL_{DCD}	1.4	—	ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Jitter tolerance, data dependent	TOL_{DDJ}	2.2	—	ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Jitter tolerance, random	TOL_{RJ}	2.27	—	ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Wideband SyncE jitter tolerance	WJT	312.5	—	UI_{P-P}	1G mode. 10 Hz to 12.1 Hz. Measured according to ITU-T G.8262, section 9.2.
Wideband SyncE jitter tolerance	WJT	3750/f	—	UI_{P-P}	1G mode. 12.1 Hz to 2.5 kHz (f) Measured according to ITU-T G.8262, section 9.2.
Wideband SyncE jitter tolerance	WJT	1.5	—	UI_{P-P}	1G mode. 2.5 kHz to 50 kHz. Measured according to ITU-T G.8262, section 9.2.

Note:

1. Jitter requirements represent high-frequency jitter (above 637 kHz) and not low-frequency jitter or wander.

The following table lists the AC characteristics for the QSGMII transmitter.

Table 7-21. QSGMII Transmitter

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate		5.0 – 100 ppm	5.0 + 100 ppm	Gbps	—
Differential output return loss	RLO_{SDD22}	—	–8	dB	100 MHz to 2.5 GHz.
Differential output return loss	RLO_{SDD22}	—	–8 + 16.6 x log (f/2.5 GHz)	dB	2.5 GHz to 5.0 GHz.
Common-mode output return loss	RLO_{SCC22}	—	–6		100 MHz to 2.5 GHz.
Rise time and fall time [2]	t_R, t_F	30	130	ps	20% to 80%.
Random jitter[1]	R_J	—	0.15	UI_{P-P}	—
Deterministic jitter	D_J	—	0.15	UI_{P-P}	—
Duty cycle distortion (part of DJ)	DCD	—	0.05	UIP-P	—
Total jitter	T_J	—	0.30	UI_{P-P}	—
Eye mask	X1	—	0.15	UI_{P-P}	Near-end.
Eye mask	X2	—	0.40	UI_{P-P}	Near-end.
Eye mask	Y1	200	—	mV	Near-end.
Eye mask	Y2	—	450	mV	Near-end.

Notes:

1. Jitter requirements represent high-frequency jitter (above 637 kHz) and not low-frequency jitter or wander.
2. Slew rate is programmable.

The following table lists the AC characteristics for the QSGMII receiver.

Table 7-22. QSGMII Receiver

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate	—	5.0 – 100 ppm	5.0 + 100 ppm	Gbps	—
Differential input return loss	RLI_{SDD11}	—	–8	dB	100 MHz to 2.5 GHz.
Differential input return loss	RLI_{SDD11}	—	–8 + 16.6 x log (f/2.5 GHz)	dB	2.5 GHz to 5.0 GHz.
Common-mode input return loss	RLI_{SCC11}	—	–6	dB	100 MHz to 2.5 GHz .
Sinusoidal jitter maximum	SJ_{MAX}	—	5	UI_{P-P}	For low sinusoidal jitter frequencies below (baud/1667).
Sinusoidal jitter, high frequency	SJ_{HF}	—	0.05	UI_{P-P}	—
Deterministic jitter (uncorrelated bounded high-probability jitter)	UBHPJ	—	0.15	UI_{P-P}	—

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Data-dependent jitter (correlated bounded high-probability jitter)	CBHPJ	—	0.30	UI_{P-P}	—
Total jitter	T_J	—	0.60	UI_{P-P}	Sinusoidal jitter excluded.
Eye mask	R_X1	—	0.30	UI_{P-P}	—
Eye mask	R_Y1	50	—	mV	—
Eye mask	R_Y2	—	450	mV	—

The following table lists the AC characteristics for the 10 transmitter output.

Table 7-23. 10G Transmitter Output (SFI Point B, Host)

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Termination mismatch	ΔZ_M	—	5	%	—
AC common-mode voltage	V_{OCM_AC}	—	15	mV_{RMS}	—
Total jitter[1]	T_J	—	0.28	UI	Measured at point B, as specified in SFF-8431 revision 4.1.
Data-dependent jitter	DDJ	—	0.1	UI	Measured at point B, as specified in SFF-8431 revision 4.1.
Pulse shrinkage jitter	DDPWS	—	0.055	UI_{RMS}	Measured at point B, as specified in SFF-8431 revision 4.1. 6 dB channel loss. (For 9 dB channel loss, worst case is 0.08 UI.)
Uncorrelated jitter	UJ	—	0.023	UI_{RMS}	Measured at point B, as specified in SFF-8431 revision 4.1.
Wideband SyncE jitter	TWJ	—	0.5	UI_{P-P}	Measured according to ITU-T G.8262 section 8.3.
Eye mask	X1	—	0.12	UI	Measured at 5e-5 mask hit ratio.
Eye mask	X2	—	0.33	UI	Measured at 5e-5 mask hit ratio.
Eye mask	Y1	95	—	mV	Measured at 5e-5 mask hit ratio. Maximum SFI channel loss of 3 dB.
Eye mask	Y2	—	350	mV	—

Note:

1. With a jitter-free reference clock. Any RefClk jitter with a frequency content below 7 MHz will add to the jitter generated at the 10G output.

The following table lists the AC characteristics for the 10 transmitter output for direct attach copper.

Table 7-24. 10G Transmitter Output (SFI Point B, Host) for Direct Attach Copper

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Voltage modulation amplitude, peak-to-peak	VMA	300	—	UI	SFF-8431 section D.7
Transmitter QSQ	Q_{SQ}	63.1	—	—	QSQ - 1/RN if the 1 and 0 noise levels are identical. See SFF-8431, section D.8.
Output AC common-mode voltage	—	—	12	mV_{RMS}	See SFF-8431 section D.15.
Output TWDPc	TWDPc	—	10.7	dBe	Host electrical output measured using SFF-8431 Appendix G, including copper direct attach stressor.

The following table lists the AC characteristics for the 10G receiver input.

Table 7-25. 10G Receiver Input (SFI Point C and C“, Host)

Parameter	Symbol	Minimum	Maximum	Unit	Condition
99% jitter	$99\%_{JIT\_P\text{-}P}$	—	0.42	UI	Calibrated and measured at point C”, as specified in SFF-8431 revision 4.1
Pulse width shrinkage jitter	$DDPWS_{JIT\_P\text{-}P}$	—	0.3	UI	Calibrated and measured at point C”, as specified in SFF-8431 revision 4.1.
Total jitter tolerance	TOLJIT_P-P RLI_{SDD11}	—	0.70	UI	Calibrated and measured at point C”, as specified in SFF-8431 revision 4.1
Eye mask X1	X1	—	0.35	UI	Calibrated and measured at point C”, as specified in SFF-8431 revision 4.1
Eye mask Y1	Y1	150	—	mV	Calibrated and measured at point C”, as specified in SFF-8431 revision 4.1
Eye mask Y2	Y2	—	425	mV	Calibrated and measured at point C”, as specified in SFF-8431 revision 4.1
Wideband SyncE jitter tolerance	WJT	2488	—	$UI_{P\text{-}P}$	10 Hz to 12.1 Hz. Measured according to ITU-T G.8262 section 9.2.

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Wideband SyncE jitter tolerance	WJT	30000/f	—	UI_{P-P}	12.1 Hz to 20 kHz (f). Measured according to ITU-T G.8262 section 9.2.
Wideband SyncE jitter tolerance	WJT	1.5	—	UI_{P-P}	20 kHz to 40 kHz. Measured according to ITU-T G.8262 section 9.2.

The following table lists the AC characteristics for the 10G transmitter output for direct attach copper.

Table 7-26. 10G Receiver Input (SFI Point C, Host) for Direct Attach Copper

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Waveform distortion penalty of ISI generator	WDP_C	—	9.3	—	dBe	Copper stressor according to SFF-8431 section E.3.1.
Transmitter QSQ	Q_{SQ}	—	63.1	—		QSQ - 1/RN if the 1 and 0 noise levels are identical. See SFF-8431, section D.8.
Post-channel fixed noise source	N_O	—	2.14	—	mV_{RMS}	RMS voltage measured over one symbol period at the output of module compliance board (MCB) in a 12 GHz bandwidth. Source for QSW should be disabled during this calibration. See SFF-8431 section E.3.1.
Differential voltage modulation	VMA	—	180	—	mV	Square pattern with eight 1s and eight 0s.

The following table lists the AC characteristics for the 10G transmitter output.

Table 7-27. 10G Transmitter Output (SFI Point A, ASIC)

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate	—	10.3125 – 100 ppm	10.3125 + 100 ppm	Gbps	10 Gbps LAN 10Gbps SFP+ Cu
Termination mismatch at 1 MHz	ΔZ_M	—	5	%	50 MHz to 1.25 GHz.
Rise time and fall time	t_R, t_F	—	24	ps	20% to 80%.
AC common-mode voltage	V_{CM}	—	12	mV_{RMS}	—

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Differential output return loss	SDD22	—	–12	UI	—
Differential output return loss	SDD22	—	See note [1]	UI	—
Common-mode return loss [2]	SCC22	85	–9	UI	—
Common-mode return loss[2]	SCC22		See note [3]	UI	—

Notes:

1. Return loss value is given by the equation SDD22(dB) = –8.15 + 13.33 Log10(f/5.5 GHz).
2. The test set common-mode reference impedance is 25 Ω
3. Return loss value is given by the equation SCC22(dB) = –8.15 + 13.33 Log10(f/5.5), with f in GHz.

The following table lists the AC characteristics for the SFI input receiver.

Table 7-28. SFI Input Receiver (SFI Point D, ASIC)

Parameter	Symbol	Minimum	Maximum	Unit	Condition
RXIN input data rate, 10 Gbps LAN	—	10.3125 – 100 ppm	10.3125 + 100 ppm	Gbps	10 Gbps LAN mode 10 Gbps SFP+Cu
RXIN linear mode differential input data swing	$\Delta VRXIN_{LI\ NEAR}$	180	600	mV	Voltage modulation amplitude (VMA)
RXIN limiting mode differential input data swing	$\Delta VRXIN_{LI\ MITING}$	300	850	mV	Measured peak-to-peak
RXIN AC common-mode voltage	V_{CM}	—	15	mV_{RMS}	—
Differential input return loss	SDD11	—	–12	dB	0.01 GHz to 2.8 GHz
Differential input return loss	SDD11	—	See note [1]	dB	2.8 GHz to 11.1 GHz
Differential to common-mode conversion [2]	SCD11	—	–15	dB	0.01 GHz to 11.1 GHz

Notes:

1. Return loss value is given by the equation SDD11(dB) = –8.15 + 13.33 Log10(f/5.5), with f in GHz.
2. The test set common-mode reference impedance is 25 Ω.

The following table lists the AC characteristics for the 10GBASE-KR transmitter.

Table 7-29. 10GBASE-KR Transmitter, 10G USXGMII (10.3125Gbps, 64b66b), 10G USGMII

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate	—	10.3125 – 100 ppm	10.3125 + 100 ppm	Gbps	KR, USXGMII
Data rate	—	10.0 – 100 ppm	10.0 + 100 ppm	Gbps	O-USGMII
Differential output return loss[1]	RLO_{SDD22}	—	–9	dB	50 MHz to 2.5 GHz
Differential output return loss	RLO_{SDD22}	—	–9 + 12 x log (*f*/ 2.5 GHz)	dB	2.5 GHz to 7.5 GHz
Common-mode output return loss	RLO_{SCC22}	—	–6	dB	50 MHz to 2.5 GHz

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Common-mode output return loss	RLO_{SCC22}	—	–6 + 12 x log (*f*/ 2.5 GHz)	dB	2.5 GHz to 7.5 GHz
Rise time and fall time	t_R, t_F	—	47	UI_{P-P}	—
Random jitter	R_J	—	0.15	UI_{P-P}	—
Deterministic jitter	D_J	—	0.15	UI_{P-P}	—
Duty cycle distortion (part of DJ)	DCD	—	0.035	UI_{P-P}	—
Total jitter[2]	TJ	—	0.28	UI_{P-P}	—

Notes:

1. Informative, system related: maximum insertion loss is 15 dB @ 5 GHz (10 Gbps) (vs ~25 dB based on the KR spec). This is to allow low cost interface implementation, while meeting system requirements.
2. With a jitter-free reference clock. Any RefClk jitter with a frequency content below 7 MHz will add to the jitter generated at the 10 Gbps output.

The following table lists the AC characteristics for the 10GBASE-KR receiver.

Table 7-30. 10GBASE-KR Receiver, 10G USXGMII (10.3125Gbps, 64b66b), 10G USGMII (Octal-USGMII, 8x 1.25Gbps, 8b10b)

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate	—	10.3125 – 100 ppm	10.3125 + 100 ppm	Gbps	KR, USXGMII
Data rate	—	10.0– 100 ppm	10.0 + 100 ppm	Gbps	USGMII
Differential input return loss [1]	RLI_{SDD11}	—	–9	dB	50 MHz to 2.5 GHz
Differential input return loss	RLI_{SDD11}	—	–9 + 12 x log (*f*/ 2.5 GHz)	dB	2.5 GHz to 7.5 GHz

Note:

1. Maximum insertion loss is 15 dB @ 5 GHz (10 Gbps) (vs ~25 dB based on the KR spec). This is to allow low cost interface implementation, while meeting system requirements.

Note:

The following table lists the AC characteristics for the PCIe transmitter.

Table 7-31. PCIe Transmitter AC

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate[1]	—	2.5 – 300 ppm	2.5 + 300 ppm	Gbps	2.5GT/s mode
Data rate[1]	—	5 – 300 ppm	5 + 300 ppm	Gbps	5GT/s mode
Data rate[1]	—	8 – 300 ppm	8 + 300 ppm	Gbps	8GT/s mode
De-emphasis level ratio 3.5dB	$T_{TX-DE-RATIO-3.5dB}$	3	4	dB	2.5G and 5G mode
De-emphasis level ratio 5dB	$T_{TX-DE-RATIO-6dB}$	5.5	6.5	dB	5G mode

.....continued

Parameter	Symbol	Minimum	Maximum	Unit	Condition
TX boost ratio for full swing	$V_{TX-BOOST-FS}$	8	—	dB	8G mode
TX boost ratio for reduced swing	$V_{TX-BOOST-RS}$	2.5	—	dB	8G mode
Pseudo package loss	$PS21_{TX}$	–3.0	—	dB	8G mode
Lone pulse width	$T_{MIN-PULSE}$	0.9	—	UI	5G mode
TX Eye opening	T_{TX-EYE}	0.75	—	UI	2.5G and 5G modes
TX Eye median to max jitter	$T_{TX-EYE-MEDIAN-to-MAX-JITTER}$	—	0.125	UI	2.5G mode
Rise/fall time mismatch	$T_{RF-MISMATCH}$	—	1	UI	5G mode, 20%/80%
Differential output return loss	RLO_{SDD22}	—	–10	dB	0.05 to 1.25GHz
Differential output return loss	RLO_{SDD22}	—	–8	dB	1.25GHz to 2.5GHz
Differential output return loss	RLO_{SDD22}	—	–4	dB	2.5GHz to 4GHz
Common-mode output return loss	RLO_{SCC22}	—	–6	dB	50 MHz to 2.5 GHz.
Common-mode output return loss	RLO_{SCC22}	—	–4	dB	2.5 GHz to 4 GHz.
Peak AC common-mode voltage	$V_{TX-CM-AC-P}$	—	20	mV_P	2.5G mode
Peak-peak AC common-mode voltage	$V_{TX-CM-AC-PP}$	—	150	mV_{PP}	5G and 8G mode
Peak-peak AC common-mode voltage below 500MHz	$V_{TX-CM-AC\_LF-PP}$	—	100	mV_{PP}	5G mode
Peak-peak AC common-mode voltage below 500MHz	$V_{TX-CM-AC\_LF-PP}$	—	50	mV_{PP}	8G mode
Differential peak output voltage with Tx disabled	$V_{TX-IDLE-DIFF-AC}$	—	20	mV_{peak}	—
Deterministic Jitter above 1.5MHz	$T_{TX-HF-DJ-DD}$	—	0.15	UI	5G mode, jitter above 1.5MHz
Jitter below 1.5MHz	$T_{TX-LF-RMS}$	—	3	ps_{RMS}	5G mode, jitter above 1.5MHz
Total uncorrelated jitter	T_{TX-UTJ}	—	31.25	ps_{pp}	8G mode @10-12

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Total uncorrelated deterministic jitter	$T_{TX-UDJDD}$	—	12	ps_{pp}	8G mode
Total uncorrelated PWJ	$T_{TX-UPW-TJ}$	—	24	ps_{pp}	8G mode @10-12
Deterministic DjDD uncorrelated PWJ	$T_{TX-UPW-DJDD}$	—	10	ps_{pp}	8G mode
Data Dependent Jitter	T_{TX-DDJ}	—	18	ps_{pp}	8G mode

Note:

1. Excluding the spread spectrum clocking.

The following table lists the AC characteristics for the PCIe receiver.

Table 7-32. PCIe Receiver AC

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate [1]	—	2.5 – 300 ppm	2.5 + 300 ppm	Gbps	2.5GT/s mode
Data rate [1]	—	5 – 300 ppm	5 + 300 ppm	Gbps	5GT/s mode
Data rate [1]	—	8 – 300 ppm	8 + 300 ppm	Gbps	8GT/s mode
Receiver time eye opening	—	0.40	—	UI	2.5G mode, BER < 10-12
Receiver time eye opening post RX EQ	$T_{RX-ST-8G}$	0.30	—	UI	8G mode, BER < 10-12
Receiver height eye opening	V_{RX-EYE}	120	—	mV_{PP}	5G mode
Receiver height eye opening post RX EQ	$V_{RX-ST-8G}$	25	—	mV_{PP}	8G mode
Electrical idle detect threshold	$V_{RX-IDLE-DET-DIFFp-p}$	65	175	mV_{PP}	—
Differential input return loss	RLI_{SDD11}	—	–10	dB	50 MHz to 1.25 GHz.
Differential input return loss	RLI_{SDD11}	—	–8	dB	1.25 GHz to 2.5GHz.
Differential input return loss	RLI_{SDD11}	—	–5	dB	2.5 GHz to 4 GHz.
Common-mode return loss	RLI_{SCC11}	—	–6	dB	50 MHz to 2.5 GHz.
Common-mode return loss	RLI_{SCC11}	—	–5	dB	2.5 GHz to 4 GHz.
Random jitter tolerance	$T_{RX-ST-RJ-8G}$	—	3	ps_{RMS}	8G mode
Sinusoidal jitter tolerance	$T_{RX-ST-SJ-8G}$	—	1	UI_{PP}	8G mode, from 0.03 MHz to 1 MHz
Sinusoidal jitter tolerance	$T_{RX-ST-SJ-8G}$	—	1 – 0.9 × log(f/1 MHz)	UI_{PP}	8G mode, 1 MHz < f <10 MHz

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Sinusoidal jitter tolerance	$T_{RX-ST-SJ-8G}$	—	0.1	UI_{PP}	8G mode, 10 MHz to 100 MHz

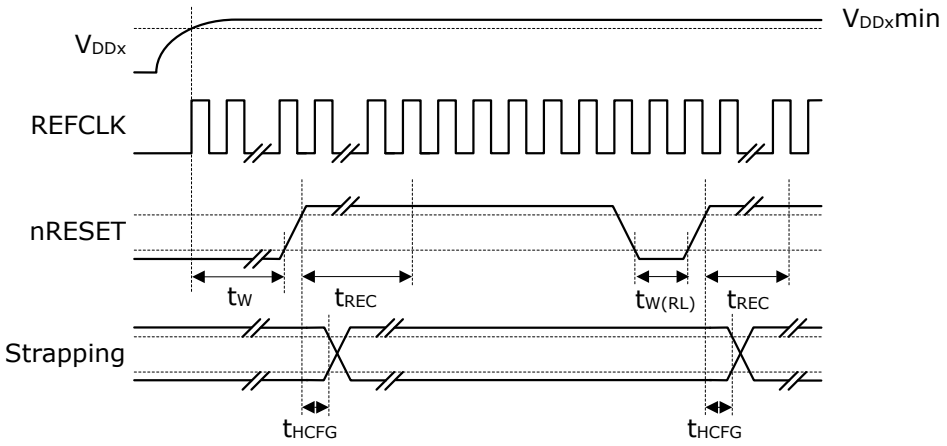
Note:

1. Excluding the spread spectrum clocking.

7.2.3 Reset Timing

The nRESET signal waveform and the required measurement points for the timing specification are shown in the following figure.

Figure 7-3. nRESET Signal Timing Specifications



The signal applied to the nRESET input must comply with the specifications listed in the following table at the reset pin of the devices.

Table 7-33. nRESET Timing Specifications

Parameter	Symbol	Minimum	Maximum	Unit
nRESET assertion time after power supplies and clock stabilizes	t_W	2	—	ms
Recovery time from reset inactive to device fully active	t_{REC}	—	10	ms
nRESET pulse width	$t_{W(RL)}$	100	—	ns
Hold time for GPIO-mapped strapping pins relative to nRESET	t_{HCFG}	50	—	ns

7.2.4 MII Management

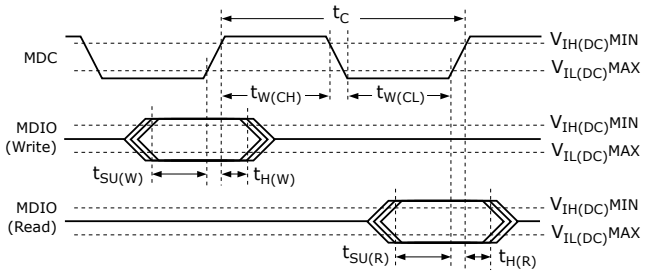
All AC specifications for the MII Management (MIIM) interface meet or exceed the requirements of IEEE 802.3-2002 (clause 22.2-4).

All MIIM AC timing requirements are specified relative to the input low and input high threshold levels. The levels are dependent on the MIIM0 interface supply mode.

The MIIM0 interface has a separate supply pin, allowing operation in either 2.5 V or 3.3 V. mode. MIIM1 and MIIM2 are GPIO alternate functions and operate only in 3.3 V mode.

The following illustration shows the MIIM waveforms and required measurement points for the signals.

Figure 7-4. MIIM Timing Diagram



The set up time of MDIO relative to the rising edge of MDC is defined as the length of time between when the MDIO exits and remains out of the switching region and when MDC enters the switching region. The hold time of MDIO relative to the rising edge of MDC is defined as the length of time between when MDC exits the switching region and when MDIO enters the switching region.

All MIIM signals comply with the specifications in the following table. The MDIO signal requirements are requested at the pin of the devices.

Table 7-34. MIIM Timing Specifications

Parameter	Symbol	Minimum	Maximum	Unit	Condition
MDC frequency[1]	f	0.488[2]	20.83	MHz	—
MDC cycle time [3]	t_C	48	2048 [4]	ns	—
MDC time high	$t_{W(CH)}$	20	—	ns	C_L = 50 pF
MDC time low	$t_{W(CL)}$	20	—	ns	C_L = 50 pF
MDIO setup time to MDC on write	$t_{SU(W)}$	15	—	ns	C_L = 50 pF
MDIO hold time from MDC on write	$t_{H(W)}$	15	—	ns	C_L= 50 pF
MDIO setup time to MDC on read	$t_{SU(R)}$	46	—	ns	C_L = 50 pF on MDC
MDIO hold time from MDC on read	$t_{H(R)}$	-11	—	ns	C_L = 50 pF

Notes:

1. For the maximum value, the device supports an MDC clock speed of up to 20 MHz for faster communication with the PHYs. If the standard frequency of 2.5 MHz is used, the MIIM interface is designed to meet or exceed the IEEE 802.3 requirements of the minimum MDC high and low times of 160 ns and an MDC cycle time of minimum 400 ns, which is not possible at faster speeds.
2. Value is for 250 MHz core clock. With 500 MHz core clock the value is 0.977 MHz.
3. Calculated as $t_C = 1/f$.
4. Value is for 250 MHz core clock. With 500 MHz core clock the value is 1024 ns.

7.2.5 Serial Interface (SI) Boot Master Mode

The following table lists the timing specifications for SI boot master mode.

Table 7-35. SI Boot Timing Specifications for Master Mode

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Clock frequency	f	—	25 [1]	MHz	—
Clock cycle time	t_C	40	—	ns	—

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Clock time high	$t_{W(CH)}$	16	—	ns	—
Clock time low	$t_{W(CL)}$	16	—	ns	—
Clock rise time and fall time	t_R, t_F	—	10	ns	Between $V_{IL(MAX)}$ and $V_{IH(MIN)}$. C_L= 30 pF.
SI_DO setup time to clock	$t_{SU(DO)}$	10	—	ns	—
SI_DO hold time from clock	$t_{H(DO)}$	10	—	ns	—
Enable active before first clock	t_{LEAD}	10	—	ns	—
Enable inactive after clock	t_{LAG}	5	—	ns	—
SI_DI setup time to clock	$t_{SU(DI)}$	42	—	ns	—
SI_DI hold time from clock	$t_{H(DI)}$	–10	—	ns	—

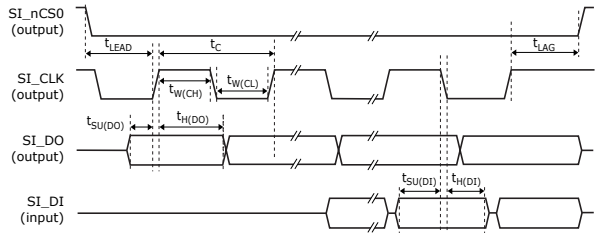
Note:

1. Frequency is programmable. The startup frequency is 8.1 MHz.

7.2.6 Serial Interface (SI) Master Mode

All serial interface (SI) timing requirements for master mode are specified relative to the input low and input high threshold levels. The following figure shows the timing parameters and measurement points.

Figure 7-5. SI Timing Diagram for Master Mode



All SI signals comply with the specifications shown in the following table. The SI input timing requirements are requested at the pins of the devices.

Table 7-36. SI Timing Specifications for Master Mode

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Clock frequency	f	—	20[1]	MHz	—
Clock cycle time	t_C	50	—	ns	—
Clock time high	$t_{W(CH)}$	20	—	ns	—
Clock time low	$t_{W(CL)}$	20	—	ns	—
Clock rise time and fall time	t_R, t_F	—	10	ns	Between $V_{IL(MAX)}$ and $V_{IH(MIN)}$. C_L= 30 pF.
SI_DO setup time to clock	$t_{SU(DO)}$	10	—	ns	—
SI_DO hold time from clock	$t_{H(DO)}$	10	—	ns	—

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Enable active before first clock	t_{LEAD}	10	—	ns	—
Enable inactive after clock	tLAG	15	—	ns	—
SI_DI setup time to clock	$t_{SU(DI)}$	48	—	ns	—
SI_DI hold time from clock	$t_{H(DI)}$	–5	—	ns	—

Note:

1. Frequency is programmable. The startup frequency is 4 MHz.

7.2.7 Serial Interface (SI) for Slave Mode

All serial interface (SI) slave mode timing requirements are specified relative to the input low and input high threshold levels. The following figures show the timing parameters and measurement points for SI input and output data.

Figure 7-6. SI Input Data Timing Diagram for Slave Mode

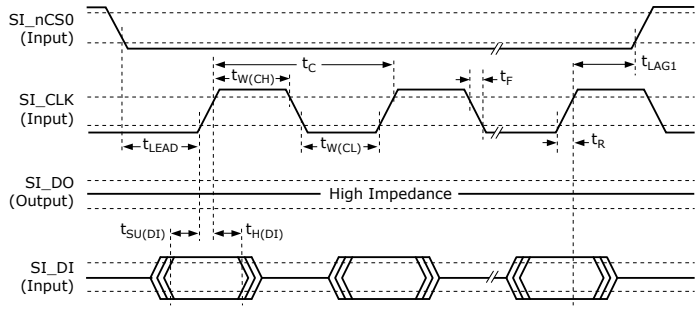
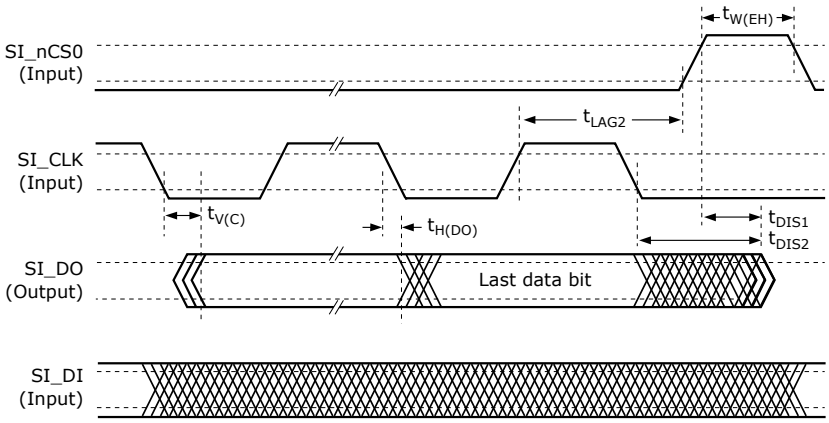


Figure 7-7. SI Output Data Timing Diagram for Slave Mode



All SI signals comply with the specifications shown in the following table. The SI input timing requirements are requested at the pins of the devices.

Table 7-37. SI Timing Specifications for Slave Mode

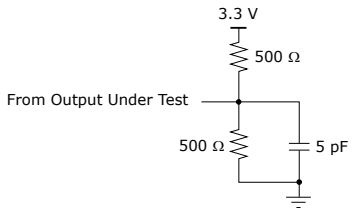
Parameter	Symbol	Minimum	Maximum	Unit	Condition
Clock frequency	*f*	—	25	MHz	—

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Clock cycle time	t_C	40	—	ns	—
Clock time high	$t_{W(CH)}$	16	—	ns	—
Clock time low	$t_{W(CL)}$	16	—	ns	—
SI_DI setup time to clock	$t_{SU(DI)}$	7	—	ns	—
SI_DI hold time from clock	$t_{H(DI)}$	1	—	ns	—
Enable active before first clock	t_{LEAD}	10	—	ns	—
Enable inactive after clock (input cycle) [1]	t_{LAG1}	25	—	ns	—
Enable inactive after clock (output cycle)	t_{LAG2}	See note[2]	—	ns	—
Enable inactive width	$t_{W(EH)}$	20	—	ns	—
SI_DO valid after clock	$t_{V(C)}$		52	ns	C_L = 30 pF.
SI_DO hold time from clock	$t_{H(DO)}$	11	—	ns	C_L = 0 pF.
SI_DO disable time [3]	t_{DIS1}	—	20	ns	See Figure 7-7.
SI_DO disable time [3]	t_{DIS2}	—	20	ns	See Figure 7-7.

Notes:

1. t_{LAG1} is defined only for write operations to the devices, not for read operations.
2. The last rising edge on the clock is necessary for the external master to read in the data. The lag time depends on the necessary hold time on the external master data input.
3. Pin begins to float when a 300 mV change from the loaded V_{OH} or V_{OL} level occurs.

Figure 7-8. SI_DO Disable Test Circuit



7.2.8 DDR SDRAM Interface

This section provides the AC characteristics for the DDR4, DDR3, and DDR3L SDRAM interface.

The following table lists the AC specifications for the DDR4 SDRAM input signals.

Table 7-38. DDR4 SDRAM Input Signal AC Characteristics

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Reference Voltage for CA inputs	$V_{REFCA(DC)}$	0.49 × V_{DD_IODDR}	—	0.51 × V_{DD_IODDR}	V	—
DDR4 CA AC input logic high	$V_{IH(AC)}$	V_{REFCA}+0.100	—	V_{DD_IODDR}	V	—
DDR4 CA AC input logic low	VIL(AC)	—	—	V_{REFCA}+0.100	V	—

.....continued						
Parameter	**Symbol**	**Minimum**	**Typical**	**Maximum**	**Unit**	**Condition**
DDR4 average clock period	$t_{CK(avg)}$	1600	—	—	ps	—
DDR4 Absolute clock HIGH/LOW pulse width	$t_{CHL(abs)}$	0.45	—	—	$t_{CK(avg)}$	—
DDR4 DQ input setup time relative to DQS1	t_{DS}	0.19	—	—	UI	—
DDR4 DQ input hold time relative to DQS [1]	t_{DH}	0.19	—	—	UI	—
Quarter period offset from clock edge	t_{90}	—	0.25 × t_{CK}	—	ns	

Note:

1. These requirements are dependent on the operation frequency of the DDR SDRAM interface. Stated limits are for DDR4-1250.

The following table lists the AC specifications for the DDR4 SDRAM input signals.

Table 7-39. DDR4 SDRAM Output Signal AC Characteristics

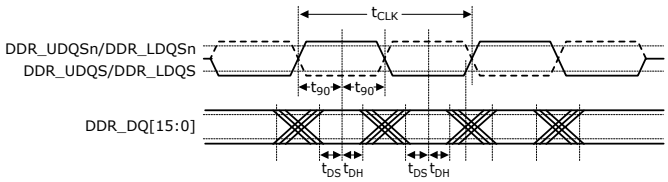
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
DDR CMD termination voltage	V_{TT}	—	0.5 × V_{DDQ}	—	V	—
DDR4 CMD AC output logic high	$V_{OH(AC)}$	V_{TT} + (0.1 × V_{DDQ})	—	—	V	—
DDR4 CMD AC output logic low	$V_{OL(AC)}$	—	—	V_{TT} – (0.1 × V_{DDQ})	V	—
DDR4 DQ AC output logic high	$V_{OH(AC)}$	0.85 × V_{DDQ}	—	—	V	—
DDR4 DQ AC output logic low	$V_{OL(AC)}$	—	—	0.55 × V_{DDQ}	V	—
DDR4 average clock period	$t_{CK(avg)}$	1600	—	—	ps	—
DDR4 Absolute clock HIGH/LOW pulse width	$t_{CHL(abs)}$	0.45	—	—	$t_{CK(avg)}$	—
DDR4 DQS to DQ skew, per lane	t_{DQSQ}	—	—	0.18	UI	
DDR4 DQ output hold time relative DQS[1]	t_{QH}	0.74	—	—	UI	—
Quarter period offset from clock edge	t_{90}	—	0.25 × t_{CK}	—	ns	—

Note:

1. These requirements are dependent on the operation frequency of the DDR SDRAM interface. Stated limits are for DDR4-1250.

The following figure shows the DDR3/DDR3L SDRAM input timing diagram.

Figure 7-9. DDR SDRAM Input Timing Diagram



The following table lists the AC specifications for the DDR3 and DDR3L SDRAM input signals.

Table 7-40. DDR3/DDR3L SDRAM Input Signal AC Characteristics

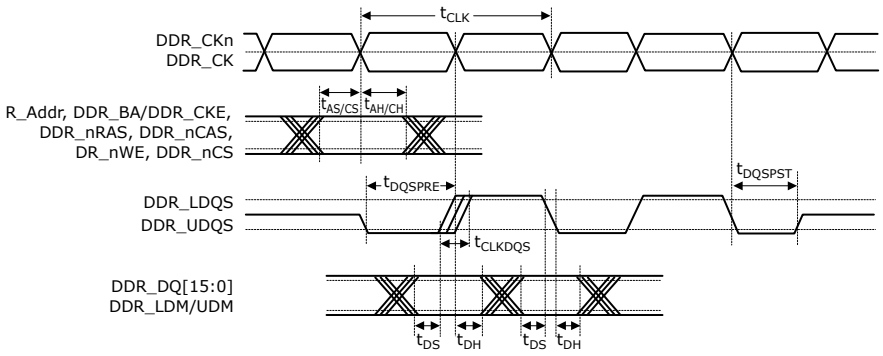
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
DDR3 input voltage high	$V_{IH(AC)}$	DDR_V_{REF} + 0.175	—	V_{DD_IODDR} + 0.3	V	—
DDR3 input voltage low	$V_{IL(AC)}$	–0.3	—	DDR_V_{REF} – 0.175	V	—
DDR3 differential input voltage	$V_{ID(AC)}$	0.4	—	V_{DD_IODDR}	V	—
DDR3 differential crosspoint voltage	$V_{IX(AC)}$	0.5 × V_{DD_IODDR} – 0.150	—	0.5 × VDD_IODDR + 0.150	V	—
DDR3L input voltage high	$V_{IH(AC)}$	DDR_V_{REF} + 0.160	—	V_{DD_IODDR} + 0.3	V	—
DDR3L input voltage low	$V_{IL(AC)}$	–0.3	—	DDR_V_{REF} – 0.160	V	—
DDR3L differential input voltage	$V_{ID(AC)}$	0.4	—	V_{DD_IODDR}	V	—
DDR3L differential crosspoint voltage	$V_{IX(AC)}$	0.5 × V_{DD_IODDR} – 0.150	—	0.5 × V_{DD_IODDR} + 0.150	V	—
CMD/ADDR input setup time relative to CK/CKn [1]	t_{IS}	140	—	—	ps	—
DDR_DQ[15:8] input setup time relative to DDR_UDQS [1]	t_{DS}	100	—	—	ps	—
DDR_DQ[7:0] input hold time relative to DDR_LDQS [1]	t_{DH}	60	—	—	ps	—
DDR_DQ[15:8] input hold time relative to DDR_UDQS [1]	t_{DH}	45	—	—	ps	—
Quarter period offset from clock edge	t_{90}	—	0.25 × t_{CLK}	—	ns	t_{CLK} = 1.0 ns

Note:

1. These requirements are dependent on the operation frequency of the DDR SDRAM interface. Stated limits are for DDR3-2000 (DDR3-2133).

The following figure shows the timing diagram for the DDR3 and DDR3L SDRAM outputs.

Figure 7-10. DDR SDRAM Output Timing Diagram



The following table lists the AC characteristics for the DDR3 and DDR3L SDRAM output signals.

Table 7-41. DDR3/DDR3L SDRAM Output Signal AC Characteristics

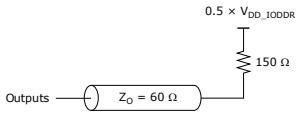
Parameter	Symbol	Minimum	Typical	Maximum	Unit
DDR_CK cycle time 1000 MHz (DDR3-2133) [1]	t_{CLK}	—	1.0	—	ns
DDR_CK/CKn duty cycle	—	48	—	52	%
DDR3 CK to DQS skew	t_{DQSS}	–0.27	—	0.27	t_{CK}
DDR3 DQS to DQ skew, per group, per access	t_{DQSQ}	—	—	80	ps
DDR3 DQ/DM output hold time relative to DQS	t_{QH}	0.38	—	—	t_{CK}

Note:

1. Timing reference is DDR_CK/DDR_CKn crossing DDR_VREF. Actual max data rate is DDR3-2000.

The following figure shows the test load circuit for the DDR3 outputs.

Figure 7-11. Test Load Circuit for DDR3 Outputs

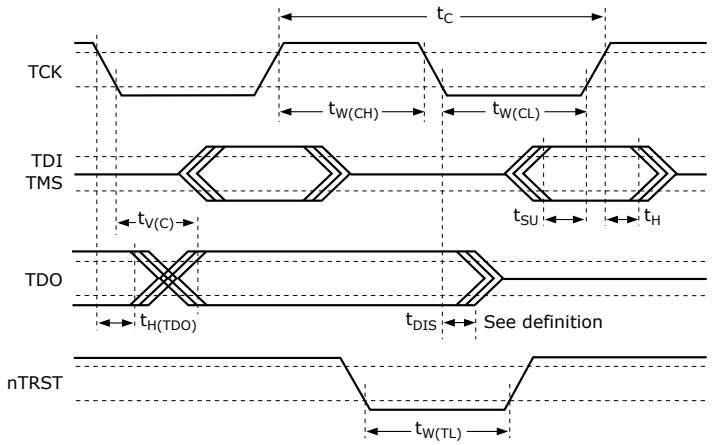


7.2.9 JTAG Interface

All AC specifications for the JTAG interface meet or exceed the requirements of IEEE 1149.1-2001.

The following illustration shows the JTAG transmit and receive waveforms and required measurement points for the different signals.

Figure 7-12. JTAG Interface Timing Diagram



All JTAG signals comply with the specifications in the following table. The JTAG receive signal requirements are requested at the pin of the devices.

The JTAG_nTRST signal is asynchronous to the clock and does not have a setup or hold time requirement.

Table 7-42. JTAG Interface AC Specifications

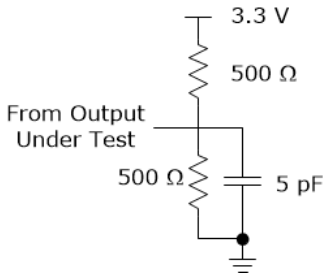
Parameter	Symbol	Minimum	Maximum	Unit	Condition
TCK frequency	f	—	10	MHz	—
TCK cycle time	t_C	100	—	ns	—
TCK high time	$t_{W(CH)}$	40	—	ns	—
TCK low time	$t_{W(CL)}$	40	—	ns	—
Setup time to TCK rising	t_{SU}	10	—	ns	—
Hold time from TCK rising	t_H	10	—	ns	—
TDO valid after TCK falling	$t_{V(C)}$		28	ns	C_L = 10 pF
TDO hold time from TCK falling	$t_{H(TDO)}$	0	—	ns	C_L = 0 pF
TDO disable time [1]	t_{DIS}	—	30	ns	See Figure 7-12.
nTRST time low	$t_{W(TL)}$	30	—	ns	—

Note:

1. The pin begins to float when a 300 mV change from the actual VOH/VOL level occurs.

The following illustration shows the test circuit for the TDO disable time.

Figure 7-13. Test Circuit for TDO Disable Time

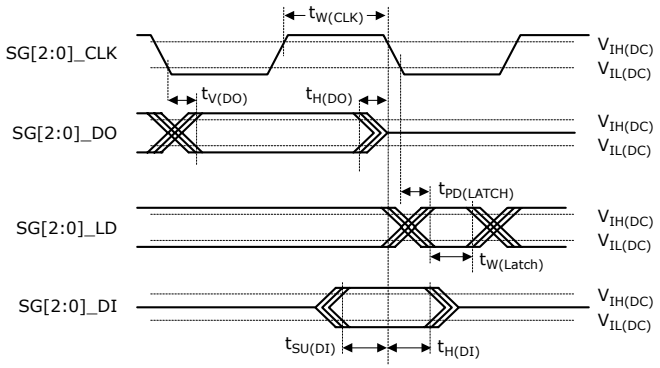


7.2.10 Serial Inputs/Outputs

This section provides the AC characteristics for the serial I/O signals: SG[2:0]_CLK, SG[2:0]_DO, SG[2:0]_DI, and SG[2:0]_LD. These signals are GPIO alternate functions.

The serial I/O timing diagram is shown in the following figure.

Figure 7-14. Serial I/O Timing Diagram



The following table lists the serial I/O timing specifications.

Table 7-43. Serial I/O Timing Specifications

Parameter	Symbol	Minimum	Maximum	Unit
Clock frequency [1]	f	—	25	MHz
SG[2:0]_CLK clock pulse width	$t_{W(CLK)}$	40	60	%
SG[2:0]_DO valid after clock falling	$t_{V(DO)}$	—	1	ns
SG[2:0]_DO hold time from clock falling	$t_{H(DO)}$	—	–2.1	ns
SG[2:0]_LD propagation delay from clock falling	$t_{PD(LATCH)}$	40	—	ns
SG[2:0]_LD width	$t_{W(LATCH)}$	10	—	ns
SG[2:0]_DI setup time to clock	$t_{SU(DI)}$	40	—	ns
SG[2:0]_DI hold time from clock	$t_{H(DI)}$	–11	—	ns

Note:

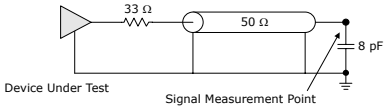
1. The SIO clock frequency is programmable.

7.2.11 Recovered Clock Outputs

This section provides the AC characteristics for the recovered clock output signals: RCVRD_CLK[3:0].

The following illustration shows the test circuit for the recovered clock output signals.

Figure 7-15. Test Circuit for Recovered Clock Output Signals



The following table lists the AC specifications for the recovered clock outputs.

Table 7-44. Recovered Clock Output AC Specifications

Parameter	Symbol	Minimum	Maximum	Unit	Condition
RCVRD_CLK[3:0] clock frequency	*f*	—	161.33	MHz	
Clock duty cycle[1]	t_C	40	60	%	Measured at 50% threshold.
RCVRD_CLK[3:0] rise time and fall time	t_R, t_F	—	1	ns	—
Squelching delay from SGMII signal to RCVRD_CLK[3:0]	—	—	200	ns	Squelch enabled.

Note:

1. Not valid when the SD_RECO_CLK_DIV register is configured with the divider 66/32 as this produces a gabbed clock.

7.2.12 eMMC Timing

The following table lists the eMMC timing specifications. The following figure displays eMMC timing diagram.

Figure 7-16. eMMC Timing Diagram

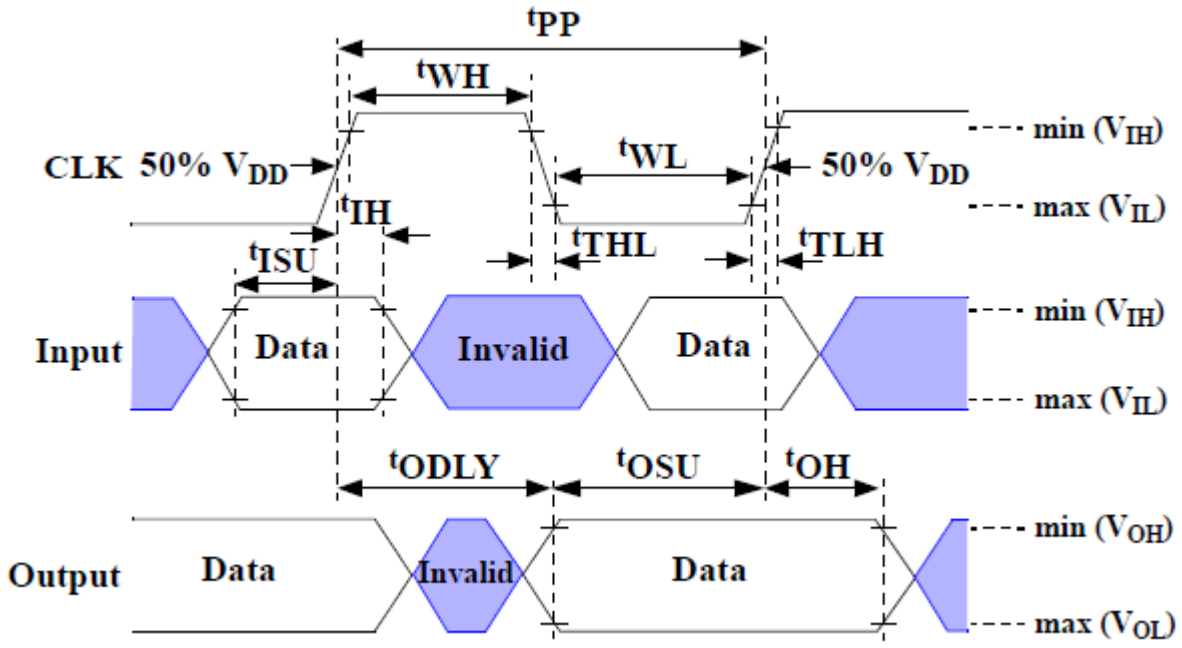


Table 7-45. Serial I/O Timing Specifications

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Clock frequency data transfer mode [1]	*f*PP	—	52	MHz	—
Clock frequency Identification mode	f_{OD}	—	400	kHz	—
Clock high time	t_{WH}	6.5	—	ns	50% of VDDIO33
Clock low time	t_{WL}	6.5	—	ns	50% of VDDIO33

.....continued					
Parameter	**Symbol**	**Minimum**	**Maximum**	**Unit**	**Condition**
Clock and signal rise/fall time	$t_{THL/TLH}$	—	3	ns	Between minVIH and max VIL. Maximum 30pF load
Output hold time	t_{OH}	0	—	ns	50% of VDDIO33. Maximum 30 pF load Measured from rising clock edge
Output delay time	t_{ODLY}	—	1.5	ns	50% of VDDIO33. Maximum 30 pF load Measured from rising clock edge
Input setup time	t_{ISU}	17.4	—	ns	50% of VDDIO33. Maximum 30 pF load
Input hold time	t_{IH}	–4.9	—	ns	50% of VDDIO33. Maximum 30 pF load

Note:

1. The clock frequency is programmable.

7.2.13 Two-Wire Serial Interface

This section provides the AC specifications for the two-wire serial interface signals TWI_SCL and TWI_SDA. The two-wire serial interface signals are GPIO alternate functions.

The two-wire serial interface signals are compatible with the Philips I2C-BUS specifications, except for the minimum rise time and fall time requirements for fast mode.

Figure 7-17. Two-Wire Serial Read Timing Diagram

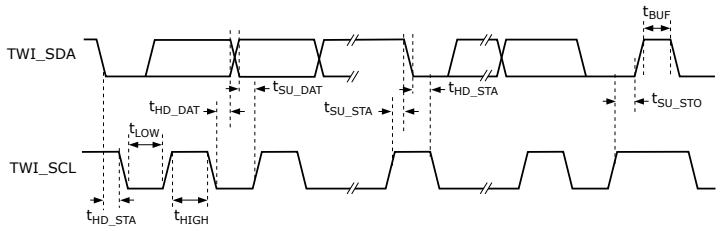
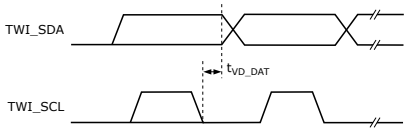


Figure 7-18. Two-Wire Serial Write Timing Diagram



The following table lists the AC specifications for the serial interface. Standard mode is defined as 100 kHz and fast mode is 400 kHz. The data in this table assumes that the software-configurable two-wire interface timing parameters, SS_SCL_HCNT, SS_SCL_LCNT, FS_SCL_HCNT, and FS_SCL_LCNT, are set to valid values for the selected speed.

Table 7-46. Two-Wire Serial Interface AC Specifications

		Standard Mode		Fast Mode			
Parameter	**Symbol**	**Minimum**	**Maximum**	**Minimum**	**Maximum**	**Unit**	**Condition**
TWI_SCL clock frequency	f	—	100		400	kHz	—
TWI_SCL low period	t_{LOW}	4.7	—	1.3	—	µs	—
TWI_SCL high period	t_{HIGH}	4.0	—	0.6	—	µs	—
TWI_SCL and TWI_SDA rise time	—	—	1000		300	ns	—
TWI_SCL and TWI_SDA fall time	—	—	300		300	ns	—
TWI_SDA setup time to TWI_SCL fall	t_{SU_DAT}	250	—	100	300	ns	—
TWI_SDA hold time to TWI_SCL fall [1]	t_{HD_DAT}	300	3450	300	900	ns	300 ns delay enabled in ICPU_CFG::TWI_CONFIG register.
Setup time for repeated START condition	t_{SU_STA}	4.7	—	0.6	—	µs	—
Hold time after repeated START condition	t_{HD_STA}	4.0	—	0.6	—	µs	—
Bus free time between STOP and START conditions	t_{BUF}	4.7	—	1.3	—	µs	—
Clock to valid data out [2]	t_{VD_DAT}	300	—	300	—	ns	—
Pulse width of spike suppressed by input filter on TWI_SCL or TWI_SDA	—	0	5	0	5	ns	—

Notes:

1. An external device must provide a hold time of at least 300 ns for the TWI_SDA signal to bridge the undefined region of the falling edge of the TWI_SCL signal.
2. Some external devices may require more data in hold time (target device's t_{HD_DAT}) than what is provided by t_{VD_DAT}, for example, 300 ns to 900 ns. The minimum value of t_{VD_DAT} is adjustable; the value given represents the recommended minimum value, which is enabled in ICPU_CFG::TWI_CONFIG.TWI_DELAY_ENABLE.

7.3 Current and Power Consumption

This section provides the current and power consumption requirements for the VSC7546 and VSC7549 SparX-5 devices.

7.3.1 Current Consumption

The following tables show the operating current for the devices. Typical current consumption values are over a full traffic load, nominal process and supply voltages, and at 25 °C case temperature. Maximum current consumption values are over full traffic load and worst-case process, temperature, and supply settings.

The following table shows current consumption values for VSC7546 and VSC7549 SparX-5.

Table 7-47. Operating Current

Parameter	Symbol	Maximum (T_{J_MAX} = 105 °C)	Unit
V_{DD} operating current, 0.9 V	I_{DD}	12	A
V_{DD_IO18} operating current, 1.8 V	I_{DD_IO18}	0.1	A
V_{DD_H18} operating current, 1.8 V	I_{DD_H18}	1	A
V_{DD_HV2} operating current, 1.0 V	I_{DD_HV2}	0.1	A
V_{DD_HV} operating current, 0.9 V	I_{DD_HV}	0.2	A
V_{DD_HS2} operating current, 1.0 V	I_{DD_HS2}	3.5	A
V_{DD_HS} operating current, 0.9 V	I_{DD_HS}	3.8	A
V_{DD_IODDR} operating current [1] , 1.2 V, 1.35 V or 1.5 V	I_{DD_IODDR}	0.4	A
V_{DD_PLLDDR} operating current, 1.8 V	I_{DD_PLLDDR}	0.1	A
V_{DD_IO33} operating current [2] , 3.3 V	I_{DD_IO33}	0.1	A

Notes:

1. DDR3/DDR4 on-die termination is disabled.
2. Unloaded pins.

Note:
The values above are intended for power supply design only.

7.3.2 Power Consumption

The following table shows the typical and maximum power consumption of the devices, based on current consumption and with DDR3 on-die termination disabled. Typical power consumption values are over nominal process and supply settings and at 25 °C case temperature. Maximum power consumption values are over maximum temperature and at maximum supply voltages.

The following table shows power consumption values for the VSC7546 and VSC7549 SparX-5 device.

Table 7-48. Power Consumption

Part Number	Typical	Maximum	Unit
VSC7546	13.9	16.7	W
VSC7549	15.1	17.9	W

In many applications the total power consumption can be minimized if not all SerDes and Clock Management Units are used. The following table shows the typical power consumption for some configurations excluding PCIe (~0.3W).

Table 7-49. Example configurations (Excluding PCIe)

Configuration	Typical	Unit	Part Number
24x1G (QSGMII) + 4x10G	7.2	W	VSC7546
8x10G	7.7	W	VSC7549

7.3.3 Power Supply Sequencing

During power on and off, V_{DD_HS}, V_{DD_HS2}, V_{DD_HV}, and V_{DD_HV2} must never be more than 300 mV above VDD. The maximum rising slope of the V_{DD_HS}, V_{DD_HS2}, V_{DD_HV}, V_{DD_HV2}, and V_{DD_H18} supplies during power turn-on must be below 5 V/ms to limit the inrush current.

V_{DD_HS}, V_{DD_HS2}, V_{DD_HV}, and V_{DD_HV2} must be powered, even if the associated interface is not used. These power supplies must not remain at ground or left floating.

V_{DD}, V_{DD_IODDR}, and $V_{DDPLLDDR}$ must power on simultaneously, or in the following sequence: V_{DD} - V_{DD_IODDR} - $V_{DDPLLDDR}$. It is recommended to power V_{DD} before V_{DD_IODDR} and V_{DD_IODDR} before $V_{DDPLLDDR}$.

During power on and off, V_{DD_IO33} must never be more than 1.8 V above V_{DD_IO18}.

In summary, the following power up sequence should be followed: 0.9V before 1.0V before 1.2V before 1.35V/1.5V before 1.8V before 3.3V.

The nRESET and JTAG_nTRST inputs must be held low until all power supply voltages have reached their recommended operating condition values.

7.4 Operating Conditions

The following table shows the recommended operating conditions.

Table 7-50. Recommended Operating Conditions

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Power supply voltage for core supply	V_{DD}	0.85	0.90	0.95	V
Power supply voltage for I/O circuits	V_{DD_IO18}	1.70	1.80	1.90	V
Power supply voltage for SerDes analog circuits	V_{DD_H18}	1.70	1.80	1.90	V
Analog supply for SerDes 25G PLLs1[1]	V_{DD_HV2}	0.85	1.00	1.05	V
Analog supply for SerDes <25G PLLs	V_{DD_HV}	0.85	0.9	0.95	V
Power supply voltage for SERDES 25G[1]	V_{DD_HS2}	0.85	1.00	1.05	V
Power supply voltage for SERDES <25G	V_{DD_HS}	0.85	0.9	0.95	V
Power supply voltage for DDR3 interface	V_{DD_IODDR}	1.425	1.500	1.575	V
Power supply voltage for DDR3L interface	V_{DD_IODDR}	1.28	1.35	1.45	V
Power supply voltage for DDR4 interface	V_{DD_IODDR}	1.14	1.20	1.26	V
Power supply voltage for DDR3/DDR3L/DDR4 PLL	$V_{DDPLLDDR}$	1.70	1.80	1.90	V
Power supply voltage for GPIO and miscellaneous I/O	V_{DD_IO33}	3.13	3.30	3.47	V
Maximum rising slope of the V_{DD_HS}, V_{DD_HS2}, V_{DD_HV}, V_{DD_HV2}, and V_{DD_H18} supplies during power turn-on				5	V/ms
Operating temperature[2]	T	0		105	°C

Notes:

1. When operating at line speeds above 12Gbps, the minimum voltage is 0.95 V.
2. Minimum specification is ambient temperature, and the maximum is junction temperature.

7.5 Stress Ratings

⚠WARNING Stresses listed in the following table may be applied to devices one at a time without causing permanent damage. Functionality at or exceeding the values listed is not implied. Exposure to these values for extended periods may affect device reliability.

Table 7-51. Stress Ratings

Parameter	Symbol	Minimum	Maximum	Unit
Power supply voltage for core supply	V_{DD}	–0.3	1.10	V
Power supply voltage for analog circuits	V_{DD_IO18}, V_{DD_H18}	–0.3	1.96	V
Reference supply for SerDes and PLL	V_{DD_HV2}, V_{DD_HV}	–0.3	1.10	V
Power supply voltage for SerDes	V_{DD_HS2}, V_{DD_HS}	–0.3	1.10	V
Power supply voltage for DDR I/O buffers	V_{DD_IODDR}	–0.3	1.8	V
Power supply voltage for GPIO and miscellaneous I/O	V_{DD_IO33}	–0.3	3.63	V
Power supply voltage for DDR PLL	$V_{DDPLLDDR}$	–0.3	1.96	V
Storage temperature	T_S	–55	125	°C

⚠WARNING This device can be damaged by electrostatic discharge (ESD) voltage. Microchip recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures may adversely affect reliability of the device.

8. Pin Descriptions

The SparX-5 device has 672 pins, which are described in this section.

The pin information is also provided as an attached Microsoft Excel file, so that you can copy it electronically. In Adobe Reader, double-click the attachment icon.

8.1 Pin Diagram

The following illustration is a representation of the SparX-5 device, as seen from the top view looking through the device. For clarity, the device is shown in two halves, the top left and the top right.

Figure 8-1. Pin Diagram, Top Left

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	S32_TXN	VDDHS2_1	S31_TXN	VDDHS2_2	S30_TXN	VDDHS2_3	S29_TXN	VDDHS2_4	S28_TXN	VDDHS2_5	S27_TXN	VDDHS2_6	S26_TXN	VDDHS2_7	S25_TXN
B	S32_TXP	VDDHS2_9	S31_TXP	VDDHS2_10	S30_TXP	VDDHS2_11	S29_TXP	VDDHS2_12	S28_TXP	VDDHS2_13	S27_TXP	VDDHS2_14	S26_TXP	VDDHS2_15	S25_TXP
C	VSS_1	VSS_2	VSS_3	VSS_4	VSS_5	VSS_6	VSS_7	VSS_8	VSS_9	VSS_10	VSS_11	VSS_12	VSS_13	VSS_14	VSS_15
D	S32_RXN	VSS_27	S31_RXN	VSS_28	S30_RXN	VSS_29	S29_RXN	VSS_30	S28_RXN	VSS_31	S27_RXN	VSS_32	S26_RXN	VSS_33	S25_RXN
E	S32_RXP	VSS_40	S31_RXP	VSS_41	S30_RXP	VSS_42	S29_RXP	VSS_43	S28_RXP	VSS_44	S27_RXP	VSS_45	S26_RXP	VSS_46	S25_RXP
F	VSS_49	VSS_50	VSS_51	VSS_52	VSS_53	VDDHV2_1	VSS_54	VDDHV2_2	VSS_55	VDDH18_1	VSS_56	VDDH18_2	VSS_57	VDDH18_3	VSS_58
G	REFCLK2_N	VSS_59	VSS_60	RESERVED_8	VSS_61	VDDHV2_3	VSS_62	VDDHV2_4	VSS_63	VDDH18_5	VSS_64	VDDH18_6	VSS_65	VDDH18_7	VSS_66
H	REFCLK2_P	VSS_71	VSS_72	VSS_73	VSS_74	VSS_75	VSS_76	VSS_77	VSS_78	VSS_79	VSS_80	VSS_81	VSS_82	VSS_83	VSS_84
J	VSS_95	RESERVED_6	THERMDA	THERMDC	VDDIO18_1	VSS_96	VSS_97	VDD_1	VDD_2	VSS_98	VSS_99	VDD_3	VDD_4	VSS_100	VSS_101
K	nRESET	MDC0	MDIO0	RESERVED_0	VDDIO18_2	VSS_108	VSS_109	VDD_9	VDD_10	VSS_110	VSS_111	VDD_11	VDD_12	VSS_112	VSS_113
L	GPIO_60	GPIO_61	GPIO_62	GPIO_63	VDDIO33_1	VSS_120	VSS_121	VDD_17	VDD_18	VSS_122	VSS_123	VDD_19	VDD_20	VSS_124	VSS_125
M	GPIO_56	GPIO_57	GPIO_58	GPIO_59	VDDIO33_2	VSS_132	VSS_133	VDD_25	VDD_26	VSS_134	VSS_135	VDD_27	VDD_28	VSS_136	VSS_137
N	GPIO_52	GPIO_53	GPIO_54	GPIO_55	VDDIO33_3	VSS_144	VSS_145	VDD_33	VDD_34	VSS_146	VSS_147	VDD_35	VDD_36	VSS_148	VSS_149
P	GPIO_48	GPIO_49	GPIO_50	GPIO_51	VDDIO33_4	VSS_156	VSS_157	VDD_41	VDD_42	VSS_158	VSS_159	VDD_43	VDD_44	VSS_160	VSS_161
R	GPIO_34	GPIO_35	GPIO_36	GPIO_37	VDDIO33_5	VSS_168	VSS_169	VDD_49	VDD_50	VSS_170	VSS_171	VDD_51	VDD_52	VSS_172	VSS_173
T	GPIO_30	GPIO_31	GPIO_32	GPIO_33	VDDIO33_6	VSS_180	VSS_181	VDD_57	VDD_58	VSS_182	VSS_183	VDD_59	VDD_60	VSS_184	VSS_185
U	GPIO_26	GPIO_27	GPIO_28	GPIO_29	VDDIO33_7	VSS_192	VSS_193	VDD_65	VDD_66	VSS_194	VSS_195	VDD_67	VDD_68	VSS_196	VSS_197
V	GPIO_22	GPIO_23	GPIO_24	GPIO_25	VDDIO33_8	VSS_204	VSS_205	VDD_73	VDD_74	VSS_206	VSS_207	VDD_75	VDD_76	VSS_208	VSS_209
W	GPIO_18	GPIO_19	GPIO_20	GPIO_21	VDDIO33_9	VSS_216	VSS_217	VDD_81	VDD_82	VSS_218	VSS_219	VDD_83	VDD_84	VSS_220	VSS_221
Y	GPIO_14	GPIO_15	GPIO_16	GPIO_17	VDDIO33_10	VSS_228	VSS_229	VDD_89	VDD_90	VSS_230	VSS_231	VDD_91	VDD_92	VSS_232	VSS_233
AA	GPIO_10	GPIO_11	GPIO_12	GPIO_13	VDDIO33_11	VSS_240	VSS_241	VDD_97	VDD_98	VSS_242	VSS_243	VDD_99	VDD_100	VSS_244	VSS_245
AB	GPIO_4	GPIO_5	GPIO_6	GPIO_7	VDDIO33_12	VSS_252	VSS_253	VDD_105	VDD_106	VSS_254	VSS_255	VDD_107	VDD_108	VSS_256	VSS_257
AC	GPIO_0	GPIO_1	GPIO_2	GPIO_3	VDDIO33_13	VSS_264	VSS_265	VSS_266	VSS_267	VSS_268	VSS_269	VSS_270	VSS_271	VSS_272	VSS_273
AD	VDDIO18_3	VDDIO18_4	VSS_284	VSS_285	VSS_286	VDDHS_15	VSS_287	VDDHS_16	VSS_288	VDDHS_17	VSS_289	VDDHS_18	VSS_290	VDDHS_19	VSS_291
AE	VSS_297	VSS_298	VSS_299	RESERVED_9	VDDH18_9	VDDHS_25	S1_RXP	VDDHS_26	S3_RXP	VDDHS_27	S5_RXP	VDDHS_28	S7_RXP	VDDHS_29	S9_RXP
AF	VDDIO33_14	VDDIO33_15	VDDIO33_16	VSS_300	VDDH18_11	S0_RXP	S1_RXN	S2_RXP	S3_RXN	S4_RXP	S5_RXN	S6_RXP	S7_RXN	S8_RXP	S9_RXN
AG	SI_CLK	JTAG_SEL0	JTAG_CPU_nRST	VSS_301	REFCLK1_P	S0_RXN	VSS_302	S2_RXN	VSS_303	S4_RXN	VSS_304	S6_RXN	VSS_305	S8_RXN	VSS_306
AH	SI_D0	JTAG_TDO	JTAG_nTRST	VSS_313	REFCLK1_N	VSS_314	S1_TXP	VSS_315	S3_TXP	VSS_316	S5_TXP	VSS_317	S7_TXP	VSS_318	S9_TXP
AJ	SI_D1	JTAG_TMS	JTAG_TCK	VSS_327	VDDHV_5	S0_TXP	S1_TXN	S2_TXP	S3_TXN	S4_TXP	S5_TXN	S6_TXP	S7_TXN	S8_TXP	S9_TXN
AK	SI_nCS0	JTAG_TDI	JTAG_SEL1	VSS_328	VDDHV_8	S0_TXN	NoBall_5	S2_TXN	NoBall_6	S4_TXN	NoBall_7	S6_TXN	NoBall_8	S8_TXN	NoBall_9

Figure 8-2. Pin Diagram, Top Right

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
VDDHS2_8	S24_TXN	NoBall_1	S22_TXN	NoBall_2	S20_TXN	NoBall_3	S18_TXN	NoBall_4	VDDHS_1	PCIE_TXN	VDDHV_1	PCIE_RXN	VDDHV_2	REFCLK3_N	A
VDDHS2_16	S24_TXP	S23_TXN	S22_TXP	S21_TXN	S20_TXP	S19_TXN	S18_TXP	S17_TXN	VDDHS_2	PCIE_TXP	VDDHV_3	PCIE_RXP	VDDHV_4	REFCLK3_P	B
VSS_16	VSS_17	S23_TXP	VSS_18	S21_TXP	VSS_19	S19_TXP	VSS_20	S17_TXP	VSS_21	VSS_22	VSS_23	VSS_24	VSS_25	VSS_26	C
VSS_34	S24_RXN	VSS_35	S22_RXN	VSS_36	S20_RXN	VSS_37	S18_RXN	VSS_38	VDDHS_3	VSS_39	DDR_DM4	DDR_DQ33	RESERVED_7	DDR_DM3	D
VSS_47	S24_RXP	S23_RXN	S22_RXP	S21_RXN	S20_RXP	S19_RXN	S18_RXP	S17_RXN	VDDHS_4	VSS_48	DDR_DQ35	DDR_DQ37	DDR_DQ25	DDR_DQ27	E
VDDH18_4	VDDHS_5	S23_RXP	VDDHS_6	S21_RXP	VDDHS_7	S19_RXP	VDDHS_8	S17_RXP	VDDHS_9	RESERVED_10	DDR_DQ39	DDR_DQ38	DDR_DQ29	DDR_DQ31	F
VDDH18_8	VDDHS_10	VSS_67	VDDHS_11	VSS_68	VDDHS_12	VSS_69	VDDHS_13	VSS_70	VDDHS_14	RESERVED_2	DDR_DQ36	DDR_DQ34	DDR_DQ30	DDR_DQ28	G
VSS_85	VSS_86	VSS_87	VSS_88	VSS_89	VSS_90	VSS_91	VSS_92	VSS_93	VSS_94	RESERVED_3	DDR_DQ32	DDR_DQS_t4	DDR_DQ26	DDR_DQ24	H
VSS_102	VSS_103	VDD_5	VDD_6	VSS_104	VSS_105	VDD_7	VDD_8	VSS_106	VSS_107	RESERVED_4	DDR_DQS_c4	DDR_ZCTRL	DDR_DQS_t3	DDR_DQS_c3	J
VSS_114	VSS_115	VDD_13	VDD_14	VSS_116	VSS_117	VDD_15	VDD_16	VSS_118	VSS_119	VDDPLLDDR_1	DDR_A16	DDR_RZQ	DDR_DM2	DDR_DQ17	K
VSS_126	VSS_127	VDD_21	VDD_22	VSS_128	VSS_129	VDD_23	VDD_24	VSS_130	VSS_131	VDDPLLDDR_2	DDR_BG1	DDR_BA1	DDR_DQ19	DDR_DQ21	L
VSS_138	VSS_139	VDD_29	VDD_30	VSS_140	VSS_141	VDD_31	VDD_32	VSS_142	VSS_143	VDDIODDR_1	DDR_A5	DDR_ALERT_n	DDR_DQ23	DDR_DQ22	M
VSS_150	VSS_151	VDD_37	VDD_38	VSS_152	VSS_153	VDD_39	VDD_40	VSS_154	VSS_155	VDDIODDR_2	DDR_A7	DDR_A13	DDR_DQ20	DDR_DQ18	N
VSS_162	VSS_163	VDD_45	VDD_46	VSS_164	VSS_165	VDD_47	VDD_48	VSS_166	VSS_167	VDDIODDR_3	DDR_CK_c	DDR_CK_t	DDR_DQ16	DDR_DQS_t2	P
VSS_174	VSS_175	VDD_53	VDD_54	VSS_176	VSS_177	VDD_55	VDD_56	VSS_178	VSS_179	VDDIODDR_4	DDR_CS_n1	DDR_CS_n0	DDR_DQS_c2	DDR_VREFODQ	R
VSS_186	VSS_187	VDD_61	VDD_62	VSS_188	VSS_189	VDD_63	VDD_64	VSS_190	VSS_191	VDDIODDR_5	DDR_A15	DDR_A12	DDR_DM1	DDR_DQ9	T
VSS_198	VSS_199	VDD_69	VDD_70	VSS_200	VSS_201	VDD_71	VDD_72	VSS_202	VSS_203	VDDIODDR_6	DDR_A3	DDR_A1	DDR_DQ11	DDR_DQ13	U
VSS_210	VSS_211	VDD_77	VDD_78	VSS_212	VSS_213	VDD_79	VDD_80	VSS_214	VSS_215	VDDIODDR_7	DDR_VREFOCA	DDR_A17	DDR_DQ15	DDR_DQ14	V
VSS_222	VSS_223	VDD_85	VDD_86	VSS_224	VSS_225	VDD_87	VDD_88	VSS_226	VSS_227	VDDIODDR_8	DDR_A9	DDR_ODT1	DDR_DQ12	DDR_DQ10	W
VSS_234	VSS_235	VDD_93	VDD_94	VSS_236	VSS_237	VDD_95	VDD_96	VSS_238	VSS_239	VDDIODDR_9	DDR_ODT0	DDR_CKE1	DDR_DQ8	DDR_DQS_t1	Y
VSS_246	VSS_247	VDD_101	VDD_102	VSS_248	VSS_249	VDD_103	VDD_104	VSS_250	VSS_251	VDDIODDR_10	DDR_CKE0	DDR_ACT_n	DDR_DQS_c1	DDR_DM0	AA
VSS_258	VSS_259	VDD_109	VDD_110	VSS_260	VSS_261	VDD_111	VDD_112	VSS_262	VSS_263	VDDIODDR_11	DDR_A10	DDR_A4	DDR_DQ5	DDR_DQ1	AB
VSS_274	VSS_275	VSS_276	VSS_277	VSS_278	VSS_279	VSS_280	VSS_281	VSS_282	VSS_283	VDDIODDR_12	DDR_A0	DDR_A6	DDR_DQ7	DDR_DQ3	AC
VDDHS_20	VSS_292	VDDHS_21	VSS_293	VDDHS_22	VSS_294	VDDHS_23	VSS_295	VDDHS_24	VSS_296	VDDPLLDDR_3	DDR_A2	DDR_PAR	DDR_DQ6	DDR_DQ4	AD
VDDHS_30	S11_RXP	VDDHS_31	VDDH18_10	VDDHS_32	S14_RXP	VDDHS_33	S16_RXP	VDDHS_34	VDDIO18_5	VDDPLLDDR_4	DDR_A8	DDR_A11	DDR_DQ2	DDR_DQ0	AE
S10_RXP	S11_RXN	S12_RXP	VDDH18_12	S13_RXP	S14_RXN	S15_RXP	S16_RXN	VDDH18_13	VDDIO18_6	RESERVED_5	DDR_A14	DDR_BG0	DDR_DQS_t0	DDR_DQS_c0	AF
S10_RXN	VSS_307	S12_RXN	VSS_308	S13_RXN	VSS_309	S15_RXN	VSS_310	VDDH18_14	GPIO_8	VSS_311	DDR_BA0	DDR_RESETn	RESERVED_1	VSS_312	AG
VSS_319	S11_TXP	VSS_320	VSS_321	VSS_322	S14_TXP	VSS_323	S16_TXP	VSS_324	GPIO_9	VDDIO33_17	VSS_325	VDDIO33_18	VSS_326	VDDIO33_19	AH
S10_TXP	S11_TXN	S12_TXP	VDDHV_6	S13_TXP	S14_TXN	S15_TXP	S16_TXN	VDDHV_7	GPIO_38	GPIO_40	GPIO_42	GPIO_44	GPIO_46	REFCLK0_P	AJ
S10_TXN	NoBall_10	S12_TXN	VDDHV_9	S13_TXN	NoBall_11	S15_TXN	NoBall_12	VDDHV_10	GPIO_39	GPIO_41	GPIO_43	GPIO_45	GPIO_47	REFCLK0_N	AK

8.2 Pins by Function

This section contains the functional pin descriptions for the SparX-5 device.

The following table lists the definitions for the pin type symbols.

Table 8-1. Pin Type Symbol Definitions

Symbol	Pin Type	Description
A	Analog input	Analog input for sensing variable voltage levels.
ABIAS	Analog bias	Analog bias pin.
DIFF	Differential	Differential signal pair.
I	Input	Input signal.
O	Output	Output signal.

.....continued		
Symbol	**Pin Type**	**Description**
I/O	Bidirectional	Bidirectional input or output signal.
O	Output	Output signal.
OZ	3-state output	Output.
LVDS	Input or output	Low voltage differential signal.
LVCMOS	Input or output	Low voltage CMOS signal.
SSTL	Input or output	Stub-series terminated logic signal.
CML	Input or output	Current mode logic signal.
PD	Pull-down	On-chip pull-down resistor to VSS.
PU	Pull-up	On-chip pull-up resistor to VDD_IO.
3V		3.3 V-tolerant.
ST	Schmitt-trigger	Input has Schmitt-trigger circuitry.
TD	Termination differential	Internal differential termination.
P	Power	Power supply.

The following table lists the functional pin descriptions for the (VSC7546 and VSC7549) SparX-5 device.

Table 8-2. Functional Pin Description

Name	Number	Signal Type	Level	Description
DDR_VREFODQ	R30	A	Analog	DDR reference voltage output for DQ. Maximum output 6 uA. Do not use.
DDR_VREFOCA	V27	A	Analog	DDR reference voltage output for control signals. Maximum output 6 uA. Do not use.
DDR_ZCTRL	J28	A	Analog	DDR internal reference voltage. Do not connect.
DDR_RZQ	K28	A	Analog	Reference pin for ZQ calibration. Connect 240 Ω 1% resistor to ground.
VDDHV_1	A27	P	—	Analog supply for SerDes.
VDDHV_2	A29	P	—	Analog supply for SerDes.
VDDHV_3	B27	P	—	Analog supply for SerDes.
VDDHV_4	B29	P	—	Analog supply for SerDes.
VDDHV_5	AJ5	P	—	Analog supply for SerDes.
VDDHV_6	AJ19	P	—	Analog supply for SerDes.
VDDHV_7	AJ24	P	—	Analog supply for SerDes.
VDDHV_8	AK5	P	—	Analog supply for SerDes.
VDDHV_9	AK19	P	—	Analog supply for SerDes.
VDDHV_10	AK24	P	—	Analog supply for SerDes.

.....continued

Name	Number	Signal Type	Level	Description
VDDHV2_1	F6	P	—	Analog supply for SerDes.
VDDHV2_2	F8	P	—	Analog supply for SerDes.
VDDHV2_3	G6	P	—	Analog supply for SerDes.
VDDHV2_4	G8	P	—	Analog supply for SerDes.
REFCLK0_N	AK30	I	CML	System reference clock input.
REFCLK0_P	AJ30	I	CML	System reference clock input
REFCLK3_N	A30	I	CML	PCIe reference clock input.
REFCLK3_P	B30	I	CML	PCIe reference clock input.
REFCLK2_N	G1	I	CML	Serdes reference clock input.
REFCLK2_P	H1	I	CML	Serdes reference clock input.
REFCLK1_N	AH5	I	CML	Serdes reference clock input.
REFCLK1_P	AG5	I	CML	Serdes reference clock input.
DDR_RESETn	AG28	O	SSTL	DDR reset output.
DDR_BA0	AG27	I/O	SSTL	DDR Bank Address.
DDR_BG0	AF28	B	SSTL	DDR4 Bank Group output. DDR3 bank address 2 (BA2) output.
DDR_A14	AF27	B	SSTL	DDR address.
DDR_A11	AE28	B	SSTL	DDR address.
DDR_A8	AE27	B	SSTL	DDR address.
DDR_PAR	AD28	B	SSTL	DDR command and address parity output.
DDR_A2	AD27	B	SSTL	DDR address.
DDR_A6	AC28	B	SSTL	DDR address.
DDR_A0	AC27	B	SSTL	DDR address.
DDR_A4	AB28	B	SSTL	DDR address.
DDR_A10	AB27	B	SSTL	DDR address.
DDR_ACT_n	AA28	B	SSTL	DDR4 Activation command (ACT) output. DDR3 Row Address Strobe (RAS#).
DDR_CKE0	AA27	O	SSTL	DDR Clock Enable.
DDR_CKE1	Y28	O	SSTL	DDR Clock Enable.
DDR_ODT0	Y27	B	SSTL	DDR On Die Termination enable.
DDR_ODT1	W28	B	SSTL	DDR On Die Termination enable.
DDR_A9	W27	B	SSTL	DDR address.
DDR_A17	V28	B	SSTL	DDR4 address (A17). DDR3 Column Address Strobe (CAS#).
DDR_CK_t	P28	B	SSTL	DDR Clock output.

.....continued

Name	Number	Signal Type	Level	Description
DDR_CK_c	P27	B	SSTL	DDR Clock output.
DDR_A1	U28	B	SSTL	DDR address.
DDR_A3	U27	B	SSTL	DDR address.
DDR_A12	T28	B	SSTL	DDR address.
DDR_A15	T27	B	SSTL	DDR address.
DDR_CS_n0	R28	B	SSTL	DDR Chip Select.
DDR_CS_n1	R27	B	SSTL	DDR Chip Select.
DDR_A13	N28	B	SSTL	DDR address.
DDR_A7	N27	B	SSTL	DDR address.
DDR_ALERT_n	M28	B	SSTL	DDR alert.
DDR_A5	M27	B	SSTL	DDR address.
DDR_BA1	L28	B	SSTL	DDR Bank address.
DDR_BG1	L27	B	SSTL	DDR4 bank group output.
DDR_A16	K27	B	SSTL	DDR4 address (A16). DDR3 Write Enable (WE#).
DDR_DQ0	AE30	B	SSTL	DDR data.
DDR_DQ2	AE29	B	SSTL	DDR data.
DDR_DQ4	AD30	B	SSTL	DDR data.
DDR_DQ6	AD29	B	SSTL	DDR data.
DDR_DQ7	AC29	B	SSTL	DDR data.
DDR_DQS_c0	AF30	B	SSTL	DDR data strobe.
DDR_DQS_t0	AF29	B	SSTL	DDR data strobe.
DDR_DQ5	AB29	B	SSTL	DDR data.
DDR_DQ3	AC30	B	SSTL	DDR data.
DDR_DQ1	AB30	B	SSTL	DDR data.
DDR_DM0	AA30	B	SSTL	DDR data mask and bus inversion (DDR4).
DDR_DQ16	P29	B	SSTL	DDR data.
DDR_DQ18	N30	B	SSTL	DDR data.
DDR_DQ20	N29	B	SSTL	DDR data.
DDR_DQ22	M30	B	SSTL	DDR data.
DDR_DQ23	M29	B	SSTL	DDR data.
DDR_DQS_c2	R29	B	SSTL	DDR data strobe.
DDR_DQS_t2	P30	B	SSTL	DDR data strobe.
DDR_DQ21	L30	B	SSTL	DDR data.
DDR_DQ19	L29	B	SSTL	DDR data.

.....continued

Name	Number	Signal Type	Level	Description
DDR_DQ17	K30	B	SSTL	DDR data.
DDR_DM2	K29	B	SSTL	DDR data mask and bus inversion (DDR4).
DDR_DQ32	H27	B	SSTL	DDR data.
DDR_DQ34	G28	B	SSTL	DDR data.
DDR_DQ36	G27	B	SSTL	DDR data.
DDR_DQ38	F28	B	SSTL	DDR data.
DDR_DQ39	F27	B	SSTL	DDR data.
DDR_DQS_c4	J27	B	SSTL	DDR data strobe.
DDR_DQS_t4	H28	B	SSTL	DDR data strobe.
DDR_DQ37	E28	B	SSTL	DDR data.
DDR_DQ35	E27	B	SSTL	DDR data.
DDR_DQ33	D28	B	SSTL	DDR data.
DDR_DM4	D27	B	SSTL	DDR data mask and bus inversion (DDR4).
DDR_DQ8	Y29	B	SSTL	DDR data.
DDR_DQ10	W30	B	SSTL	DDR data.
DDR_DQ12	W29	B	SSTL	DDR data.
DDR_DQ14	V30	B	SSTL	DDR data.
DDR_DQ15	V29	B	SSTL	DDR data.
DDR_DQS_c1	AA29	B	SSTL	DDR data strobe.
DDR_DQS_t1	Y30	B	SSTL	DDR data strobe.
DDR_DQ13	U30	B	SSTL	DDR data.
DDR_DQ11	U29	B	SSTL	DDR data.
DDR_DQ9	T30	B	SSTL	DDR data.
DDR_DM1	T29	B	SSTL	DDR data mask and bus inversion (DDR4).
DDR_DQ24	H30	B	SSTL	DDR data.
DDR_DQ26	H29	B	SSTL	DDR data.
DDR_DQ28	G30	B	SSTL	DDR data.
DDR_DQ30	G29	B	SSTL	DDR data.
DDR_DQ31	F30	B	SSTL	DDR data.
DDR_DQS_c3	J30	B	SSTL	DDR data strobe.
DDR_DQS_t3	J29	B	SSTL	DDR data strobe.
DDR_DQ29	F29	B	SSTL	DDR data.
DDR_DQ27	E30	B	SSTL	DDR data.

.....continued

Name	Number	Signal Type	Level	Description
DDR_DQ25	E29	B	SSTL	DDR data.
DDR_DM3	D30	B	SSTL	DDR data mask and bus inversion (DDR4).
GPIO_63	L4	B, PU	LVCMOS	General purpose input/output.
GPIO_62	L3	B, PU	LVCMOS	General purpose input/output.
GPIO_61	L2	B, PU	LVCMOS	General purpose input/output.
GPIO_60	L1	B, PU	LVCMOS	General purpose input/output.
GPIO_59	M4	B, PU	LVCMOS	General purpose input/output.
GPIO_58	M3	B, PU	LVCMOS	General purpose input/output.
GPIO_57	M2	B, PU	LVCMOS	General purpose input/output.
GPIO_56	M1	B, PU	LVCMOS	General purpose input/output.
GPIO_55	N4	B, PU	LVCMOS	General purpose input/output.
GPIO_54	N3	B, PU	LVCMOS	General purpose input/output.
GPIO_53	N2	B, PU	LVCMOS	General purpose input/output.
GPIO_52	N1	B, PU	LVCMOS	General purpose input/output.
GPIO_51	P4	B, PU	LVCMOS	General purpose input/output.
GPIO_50	P3	B, PU	LVCMOS	General purpose input/output.
GPIO_49	P2	B, PU	LVCMOS	General purpose input/output.
GPIO_48	P1	B, PU	LVCMOS	General purpose input/output.
GPIO_37	R4	B, PU	LVCMOS	General purpose input/output.
GPIO_36	R3	B, PU	LVCMOS	General purpose input/output.
GPIO_35	R2	B, PU	LVCMOS	General purpose input/output.
GPIO_34	R1	B, PU	LVCMOS	General purpose input/output.
GPIO_33	T4	B, PU	LVCMOS	General purpose input/output.
GPIO_32	T3	B, PU	LVCMOS	General purpose input/output.
GPIO_31	T2	B, PU	LVCMOS	General purpose input/output.
GPIO_30	T1	B, PU	LVCMOS	General purpose input/output.
GPIO_29	U4	B, PU	LVCMOS	General purpose input/output.
GPIO_28	U3	B, PU	LVCMOS	General purpose input/output.
GPIO_27	U2	B, PU	LVCMOS	General purpose input/output.
GPIO_26	U1	B, PU	LVCMOS	General purpose input/output.
GPIO_25	V4	B, PU	LVCMOS	General purpose input/output.
GPIO_24	V3	B, PU	LVCMOS	General purpose input/output.
GPIO_23	V2	B, PU	LVCMOS	General purpose input/output.
GPIO_22	V1	B, PU	LVCMOS	General purpose input/output.

.....continued

Name	Number	Signal Type	Level	Description
GPIO_21	W4	B, PU	LVCMOS	General purpose input/output.
GPIO_20	W3	B, PU	LVCMOS	General purpose input/output.
GPIO_19	W2	B, PU	LVCMOS	General purpose input/output.
GPIO_18	W1	B, PU	LVCMOS	General purpose input/output.
GPIO_17	Y4	B, PU	LVCMOS	General purpose input/output.
GPIO_16	Y3	B, PU	LVCMOS	General purpose input/output.
GPIO_15	Y2	B, PU	LVCMOS	General purpose input/output.
GPIO_14	Y1	B, PU	LVCMOS	General purpose input/output.
GPIO_13	AA4	B, PU	LVCMOS	General purpose input/output.
GPIO_12	AA3	B, PU	LVCMOS	General purpose input/output.
GPIO_11	AA2	B, PU	LVCMOS	General purpose input/output.
GPIO_10	AA1	B, PU	LVCMOS	General purpose input/output.
GPIO_7	AB4	B, PU	LVCMOS	General purpose input/output.
GPIO_6	AB3	B, PU	LVCMOS	General purpose input/output.
GPIO_5	AB2	B, PU	LVCMOS	General purpose input/output.
GPIO_4	AB1	B, PU	LVCMOS	General purpose input/output.
GPIO_3	AC4	B, PU	LVCMOS	General purpose input/output.
GPIO_2	AC3	B, PU	LVCMOS	General purpose input/output.
GPIO_1	AC2	B, PU	LVCMOS	General purpose input/output.
GPIO_0	AC1	B, PU	LVCMOS	General purpose input/output.
GPIO_8	AG25	B, PU	LVCMOS	General purpose input/output.
GPIO_9	AH25	B, PU	LVCMOS	General purpose input/output.
GPIO_38	AJ25	B, PU	LVCMOS	General purpose input/output.
GPIO_39	AK25	B, PU	LVCMOS	General purpose input/output.
GPIO_40	AJ26	B, PU	LVCMOS	General purpose input/output.
GPIO_41	AK26	B, PU	LVCMOS	General purpose input/output.
GPIO_42	AJ27	B, PU	LVCMOS	General purpose input/output.
GPIO_43	AK27	B, PU	LVCMOS	General purpose input/output.
GPIO_44	AJ28	B, PU	LVCMOS	General purpose input/output.
GPIO_45	AK28	B, PU	LVCMOS	General purpose input/output.
GPIO_46	AJ29	B, PU	LVCMOS	General purpose input/output.
GPIO_47	AK29	B, PU	LVCMOS	General purpose input/output.
JTAG_SEL0	AG2	I, PD	LVCMOS	JTAG select.
JTAG_TMS	AJ2	B, PU	LVCMOS	JTAG test mode select.
JTAG_TDI	AK2	I, PU	LVCMOS	JTAG test data input.

.....continued

Name	Number	Signal Type	Level	Description
JTAG_TCK	AJ3	I, PU	LVCMOS	JTAG clock.
JTAG_SEL1	AK3	I, PU	LVCMOS	JTAG select.
JTAG_TDO	AH2	B, PU	LVCMOS	JTAG test data out.
JTAG_CPU_nRST	AG3	I, PU	LVCMOS	JTAG CPU reset.
JTAG_nTRST	AH3	I, PU	LVCMOS	JTAG Test Controller Reset
MDIO0	K3	B, PU	LVCMOS	MIIM I/O
MDC0	K2	B, PU	LVCMOS	MIIM clock.
THERMDC	J4	A	Analog	On-die thermal diode cathode.
THERMDA	J3	A	Analog	On-die thermal diode anode.
nRESET	K1	I, PU	LVCMOS	Reset input. Active low
PCIE_TXP	B26	O	DIFF	PCIe serdes output.
PCIE_TXN	A26	O	DIFF	PCIe serdes output
PCIE_RXP	B28	I	DIFF	PCIe serdes input.
PCIE_RXN	A28	I	DIFF	PCIe serdes input.
RESERVED_5	AF26	A	Analog	Leave floating.
RESERVED_4	J26	B	SSTL	Leave floating.
RESERVED_3	H26	B	SSTL	Leave floating.
RESERVED_2	G26	A	Analog	Leave floating.
RESERVED_10	F26	A	Analog	Leave floating.
RESERVED_7	D29	A	—	Leave floating.
RESERVED_8	G4	A	—	Leave floating.
RESERVED_0	K4	I, PD	LVCMOS	Pull high.
RESERVED_6	J2	A	Analog	Leave floating.
RESERVED_9	AE4	A		Leave floating.
RESERVED_1	AG29	P		Connect to VSS.
S17_TXP	C24	O	DIFF	Ethernet serdes output.
S17_TXN	B24	O	DIFF	Ethernet serdes output.
S17_RXN	E24	I	DIFF	Ethernet serdes input.
S17_RXP	F24	I	DIFF	Ethernet serdes input.
S18_TXP	B23	O	DIFF	Ethernet serdes output.
S18_TXN	A23	O	DIFF	Ethernet serdes output.
S18_RXN	D23	I	DIFF	Ethernet serdes input.
S18_RXP	E23	I	DIFF	Ethernet serdes input.
S19_TXP	C22	O	DIFF	Ethernet serdes output.
S19_TXN	B22	O	DIFF	Ethernet serdes output.

.....continued

Name	Number	Signal Type	Level	Description
S19_RXN	E22	I	DIFF	Ethernet serdes input.
S19_RXP	F22	I	DIFF	Ethernet serdes input.
S20_TXP	B21	O	DIFF	Ethernet serdes output.
S20_TXN	A21	O	DIFF	Ethernet serdes output.
S20_RXN	D21	I	DIFF	Ethernet serdes input.
S20_RXP	E21	I	DIFF	Ethernet serdes input.
S21_TXP	C20	O	DIFF	Ethernet serdes output.
S21_TXN	B20	O	DIFF	Ethernet serdes output.
S21_RXN	E20	I	DIFF	Ethernet serdes input.
S21_RXP	F20	I	DIFF	Ethernet serdes input.
S22_TXP	B19	O	DIFF	Ethernet serdes output.
S22_TXN	A19	O	DIFF	Ethernet serdes output.
S22_RXN	D19	I	DIFF	Ethernet serdes input.
S22_RXP	E19	I	DIFF	Ethernet serdes input.
S23_TXP	C18	O	DIFF	Ethernet serdes output
S23_TXN	B18	O	DIFF	Ethernet serdes output.
S23_RXN	E18	I	DIFF	Ethernet serdes input.
S23_RXP	F18	I	DIFF	Ethernet serdes input.
S24_TXP	B17	O	DIFF	Ethernet serdes output.
S24_TXN	A17	O	DIFF	Ethernet serdes output.
S24_RXN	D17	I	DIFF	Ethernet serdes input.
S24_RXP	E17	I	DIFF	Ethernet serdes input.
S13_TXP	AJ20	O	DIFF	Ethernet serdes output.
S13_TXN	AK20	O	DIFF	Ethernet serdes output.
S13_RXN	AG20	I	DIFF	Ethernet serdes input.
S13_RXP	AF20	I	DIFF	Ethernet serdes input.
S14_TXP	AH21	O	DIFF	Ethernet serdes output.
S14_TXN	AJ21	O	DIFF	Ethernet serdes output.
S14_RXN	AF21	I	DIFF	Ethernet serdes input.
S14_RXP	AE21	I	DIFF	Ethernet serdes input.
S15_TXP	AJ22	O	DIFF	Ethernet serdes output.
S15_TXN	AK22	O	DIFF	Ethernet serdes output.
S15_RXN	AG22	I	DIFF	Ethernet serdes input.
S15_RXP	AF22	I	DIFF	Ethernet serdes input.
S16_TXP	AH23	O	DIFF	Ethernet serdes output.

.....continued

Name	Number	Signal Type	Level	Description
S16_TXN	AJ23	O	DIFF	Ethernet serdes output.
S16_RXN	AF23	I	DIFF	Ethernet serdes input.
S16_RXP	AE23	I	DIFF	Ethernet serdes input.
S25_TXP	B15	O	DIFF	Ethernet serdes output.
S25_TXN	A15	O	DIFF	Ethernet serdes output.
S25_RXN	D15	I	DIFF	Ethernet serdes input.
S25_RXP	E15	I	DIFF	Ethernet serdes input.
S26_TXP	B13	O	DIFF	Ethernet serdes output.
S26_TXN	A13	O	DIFF	Ethernet serdes output.
S26_RXN	D13	I	DIFF	Ethernet serdes input.
S26_RXP	E13	I	DIFF	Ethernet serdes input.
S27_TXP	B11	O	DIFF	Ethernet serdes output.
S27_TXN	A11	O	DIFF	Ethernet serdes output.
S27_RXN	D11	I	DIFF	Ethernet serdes input.
S27_RXP	E11	I	DIFF	Ethernet serdes input.
S28_TXP	B9	O	DIFF	Ethernet serdes output.
S28_TXN	A9	O	DIFF	Ethernet serdes output.
S28_RXN	D9	I	DIFF	Ethernet serdes input.
S28_RXP	E9	I	DIFF	Ethernet serdes input.
S29_TXP	B7	O	DIFF	Ethernet serdes output.
S29_TXN	A7	O	DIFF	Ethernet serdes output.
S29_RXN	D7	I	DIFF	Ethernet serdes input.
S29_RXP	E7	I	DIFF	Ethernet serdes input.
S30_TXP	B5	O	DIFF	Ethernet serdes output.
S30_TXN	A5	O	DIFF	Ethernet serdes output.
S30_RXN	D5	I	DIFF	Ethernet serdes input.
S30_RXP	E5	I	DIFF	Ethernet serdes input.
S31_TXP	B3	O	DIFF	Ethernet serdes output.
S31_TXN	A3	O	DIFF	Ethernet serdes output.
S31_RXN	D3	I	DIFF	Ethernet serdes input.
S31_RXP	E3	I	DIFF	Ethernet serdes input.
S32_TXP	B1	O	DIFF	Ethernet serdes output.
S32_TXN	A1	O	DIFF	Ethernet serdes output.
S32_RXN	D1	I	DIFF	Ethernet serdes input.
S32_RXP	E1	I	DIFF	Ethernet serdes input.

.....continued

Name	Number	Signal Type	Level	Description
S0_TXP	AJ6	O	DIFF	Ethernet serdes output.
S0_TXN	AK6	O	DIFF	Ethernet serdes output.
S0_RXN	AG6	I	DIFF	Ethernet serdes input.
S0_RXP	AF6	I	DIFF	Ethernet serdes input.
S1_TXP	AH7	O	DIFF	Ethernet serdes output.
S1_TXN	AJ7	O	DIFF	Ethernet serdes output.
S1_RXN	AF7	I	DIFF	Ethernet serdes input.
S1_RXP	AE7	I	DIFF	Ethernet serdes input.
S2_TXP	AJ8	O	DIFF	Ethernet serdes output.
S2_TXN	AK8	O	DIFF	Ethernet serdes output.
S2_RXN	AG8	I	DIFF	Ethernet serdes input.
S2_RXP	AF8	I	DIFF	Ethernet serdes input.
S3_TXP	AH9	O	DIFF	Ethernet serdes output.
S3_TXN	AJ9	O	DIFF	Ethernet serdes output.
S3_RXN	AF9	I	DIFF	Ethernet serdes input.
S3_RXP	AE9	I	DIFF	Ethernet serdes input.
S4_TXP	AJ10	O	DIFF	Ethernet serdes output.
S4_TXN	AK10	O	DIFF	Ethernet serdes output.
S4_RXN	AG10	I	DIFF	Ethernet serdes input.
S4_RXP	AF10	I	DIFF	Ethernet serdes input.
S5_TXP	AH11	O	DIFF	Ethernet serdes output.
S5_TXN	AJ11	O	DIFF	Ethernet serdes output.
S5_RXN	AF11	I	DIFF	Ethernet serdes input.
S5_RXP	AE11	I	DIFF	Ethernet serdes input.
S6_TXP	AJ12	O	DIFF	Ethernet serdes output.
S6_TXN	AK12	O	DIFF	Ethernet serdes output.
S6_RXN	AG12	I	DIFF	Ethernet serdes input.
S6_RXP	AF12	I	DIFF	Ethernet serdes input.
S7_TXP	AH13	O	DIFF	Ethernet serdes output.
S7_TXN	AJ13	O	DIFF	Ethernet serdes output.
S7_RXN	AF13	I	DIFF	Ethernet serdes input.
S7_RXP	AE13	I	DIFF	Ethernet serdes input.
S8_TXP	AJ14	O	DIFF	Ethernet serdes output.
S8_TXN	AK14	O	DIFF	Ethernet serdes output.
S8_RXN	AG14	I	DIFF	Ethernet serdes input.

.....continued

Name	Number	Signal Type	Level	Description
S8_RXP	AF14	I	DIFF	Ethernet serdes input.
S9_TXP	AH15	O	DIFF	Ethernet serdes output.
S9_TXN	AJ15	O	DIFF	Ethernet serdes output.
S9_RXN	AF15	I	DIFF	Ethernet serdes input.
S9_RXP	AE15	I	DIFF	Ethernet serdes input.
S10_TXP	AJ16	O	DIFF	Ethernet serdes output.
S10_TXN	AK16	O	DIFF	Ethernet serdes output.
S10_RXN	AG16	I	DIFF	Ethernet serdes input.
S10_RXP	AF16	I	DIFF	Ethernet serdes input.
S11_TXP	AH17	O	DIFF	Ethernet serdes output.
S11_TXN	AJ17	O	DIFF	Ethernet serdes output.
S11_RXN	AF17	I	DIFF	Ethernet serdes input.
S11_RXP	AE17	I	DIFF	Ethernet serdes input.
S12_TXP	AJ18	O	DIFF	Ethernet serdes output.
S12_TXN	AK18	O	DIFF	Ethernet serdes output.
S12_RXN	AG18	I	DIFF	Ethernet serdes input.
S12_RXP	AF18	I	DIFF	Ethernet serdes input.
SI_CLK	AG1	B, PU	LVCMOS	Serial interface clock.
SI_D1	AJ1	B, PU	LVCMOS	Serial interface data in.
SI_D0	AH1	B, PU	LVCMOS	Serial Interface active low chip select.
SI_nCS0	AK1	B, PU	LVCMOS	Serial interface clock.
VDD_1	J8	P	—	Core supply.
VDD_2	J9	P	—	Core supply.
VDD_3	J12	P	—	Core supply.
VDD_4	J13	P	—	Core supply.
VDD_5	J18	P	—	Core supply.
VDD_6	J19	P	—	Core supply.
VDD_7	J22	P	—	Core supply.
VDD_8	J23	P	—	Core supply.
VDD_9	K8	P	—	Core supply.
VDD_10	K9	P	—	Core supply.
VDD_11	K12	P	—	Core supply.
VDD_12	K13	P	—	Core supply.
VDD_13	K18	P	—	Core supply.

.....continued

Name	Number	Signal Type	Level	Description
VDD_14	K19	P	—	Core supply.
VDD_15	K22	P	—	Core supply.
VDD_16	K23	P	—	Core supply.
VDD_17	L8	P	—	Core supply.
VDD_18	L9	P	—	Core supply.
VDD_19	L12	P	—	Core supply.
VDD_20	L13	P	—	Core supply.
VDD_21	L18	P	—	Core supply.
VDD_22	L19	P	—	Core supply.
VDD_23	L22	P	—	Core supply.
VDD_24	L23	P	—	Core supply.
VDD_25	M8	P	—	Core supply.
VDD_26	M9	P	—	Core supply.
VDD_27	M12	P	—	Core supply.
VDD_28	M13	P	—	Core supply.
VDD_29	M18	P	—	Core supply.
VDD_30	M19	P	—	Core supply.
VDD_31	M22	P	—	Core supply.
VDD_32	M23	P	—	Core supply.
VDD_33	N8	P	—	Core supply.
VDD_34	N9	P	—	Core supply.
VDD_35	N12	P	—	Core supply.
VDD_36	N13	P	—	Core supply.
VDD_37	N18	P	—	Core supply.
VDD_38	N19	P	—	Core supply.
VDD_39	N22	P	—	Core supply.
VDD_40	N23	P	—	Core supply.
VDD_41	P8	P	—	Core supply.
VDD_42	P9	P	—	Core supply.
VDD_43	P12	P	—	Core supply.
VDD_44	P13	P	—	Core supply.
VDD_45	P18	P	—	Core supply.
VDD_46	P19	P	—	Core supply.
VDD_47	P22	P	—	Core supply.
VDD_48	P23	P	—	Core supply.

.....continued

Name	Number	Signal Type	Level	Description
VDD_49	R8	P	—	Core supply.
VDD_50	R9	P	—	Core supply.
VDD_51	R12	P	—	Core supply.
VDD_52	R13	P	—	Core supply.
VDD_53	R18	P	—	Core supply.
VDD_54	R19	P	—	Core supply.
VDD_55	R22	P	—	Core supply.
VDD_56	R23	P	—	Core supply.
VDD_57	T8	P	—	Core supply.
VDD_58	T9	P	—	Core supply.
VDD_59	T12	P	—	Core supply.
VDD_60	T13	P	—	Core supply.
VDD_61	T18	P	—	Core supply.
VDD_62	T19	P	—	Core supply.
VDD_63	T22	P	—	Core supply.
VDD_64	T23	P	—	Core supply.
VDD_65	U8	P	—	Core supply.
VDD_66	U9	P	—	Core supply.
VDD_67	U12	P	—	Core supply.
VDD_68	U13	P	—	Core supply.
VDD_69	U18	P	—	Core supply.
VDD_70	U19	P	—	Core supply.
VDD_71	U22	P	—	Core supply.
VDD_72	U23	P	—	Core supply.
VDD_73	V8	P	—	Core supply.
VDD_74	V9	P	—	Core supply.
VDD_75	V12	P	—	Core supply.
VDD_76	V13	P	—	Core supply.
VDD_77	V18	P	—	Core supply.
VDD_78	V19	P	—	Core supply.
VDD_79	V22	P	—	Core supply.
VDD_80	V23	P	—	Core supply.
VDD_81	W8	P	—	Core supply.
VDD_82	W9	P	—	Core supply.
VDD_83	W12	P	—	Core supply.

.....continued

Name	Number	Signal Type	Level	Description
VDD_84	W13	P	—	Core supply.
VDD_85	W18	P	—	Core supply.
VDD_86	W19	P	—	Core supply.
VDD_87	W22	P	—	Core supply.
VDD_88	W23	P	—	Core supply.
VDD_89	Y8	P	—	Core supply.
VDD_90	Y9	P	—	Core supply.
VDD_91	Y12	P	—	Core supply.
VDD_92	Y13	P	—	Core supply.
VDD_93	Y18	P	—	Core supply.
VDD_94	Y19	P	—	Core supply.
VDD_95	Y22	P	—	Core supply.
VDD_96	Y23	P	—	Core supply.
VDD_97	AA8	P	—	Core supply.
VDD_98	AA9	P	—	Core supply.
VDD_99	AA12	P	—	Core supply.
VDD_100	AA13	P	—	Core supply.
VDD_101	AA18	P	—	Core supply.
VDD_102	AA19	P	—	Core supply.
VDD_103	AA22	P	—	Core supply.
VDD_104	AA23	P	—	Core supply.
VDD_105	AB8	P	—	Core supply.
VDD_106	AB9	P	—	Core supply.
VDD_107	AB12	P	—	Core supply.
VDD_108	AB13	P	—	Core supply.
VDD_109	AB18	P	—	Core supply.
VDD_110	AB19	P	—	Core supply.
VDD_111	AB22	P	—	Core supply.
VDD_112	AB23	P	—	Core supply.
VDDHS_1	A25	P	—	Serdes supply for S0-S24.
VDDHS_2	B25	P	—	Serdes supply for S0-S24.
VDDHS_3	D25	P	—	Serdes supply for S0-S24.
VDDHS_4	E25	P	—	Serdes supply for S0-S24.
VDDHS_5	F17	P	—	Serdes supply for S0-S24.
VDDHS_6	F19	P	—	Serdes supply for S0-S24.

.....continued

Name	Number	Signal Type	Level	Description
VDDHS_7	F21	P	—	Serdes supply for S0-S24.
VDDHS_8	F23	P	—	Serdes supply for S0-S24.
VDDHS_9	F25	P	—	Serdes supply for S0-S24.
VDDHS_10	G17	P	—	Serdes supply for S0-S24.
VDDHS_11	G19	P	—	Serdes supply for S0-S24.
VDDHS_12	G21	P	—	Serdes supply for S0-S24.
VDDHS_13	G23	P	—	Serdes supply for S0-S24.
VDDHS_14	G25	P	—	Serdes supply for S0-S24.
VDDHS_15	AD6	P	—	Serdes supply for S0-S24.
VDDHS_16	AD8	P	—	Serdes supply for S0-S24.
VDDHS_17	AD10	P	—	Serdes supply for S0-S24.
VDDHS_18	AD12	P	—	Serdes supply for S0-S24.
VDDHS_19	AD14	P	—	Serdes supply for S0-S24.
VDDHS_20	AD16	P	—	Serdes supply for S0-S24.
VDDHS_21	AD18	P	—	Serdes supply for S0-S24.
VDDHS_22	AD20	P	—	Serdes supply for S0-S24.
VDDHS_23	AD22	P	—	Serdes supply for S0-S24.
VDDHS_24	AD24	P	—	Serdes supply for S0-S24.
VDDHS_25	AE6	P	—	Serdes supply for S0-S24.
VDDHS_26	AE8	P	—	Serdes supply for S0-S24.
VDDHS_27	AE10	P	—	Serdes supply for S0-S24.
VDDHS_28	AE12	P	—	Serdes supply for S0-S24.
VDDHS_29	AE14	P	—	Serdes supply for S0-S24.
VDDHS_30	AE16	P	—	Serdes supply for S0-S24.
VDDHS_31	AE18	P	—	Serdes supply for S0-S24.
VDDHS_32	AE20	P	—	Serdes supply for S0-S24.
VDDHS_33	AE22	P	—	Serdes supply for S0-S24.
VDDHS_34	AE24	P	—	Serdes supply for S0-S24.
VDDHS2_1	A2	P	—	Serdes supply for S25-S32.
VDDHS2_2	A4	P	—	Serdes supply for S25-S32.
VDDHS2_3	A6	P	—	Serdes supply for S25-S32.
VDDHS2_4	A8	P	—	Serdes supply for S25-S32.
VDDHS2_5	A10	P	—	Serdes supply for S25-S32.
VDDHS2_6	A12	P	—	Serdes supply for S25-S32.
VDDHS2_7	A14	P	—	Serdes supply for S25-S32.

.....continued

Name	Number	Signal Type	Level	Description
VDDHS2_8	A16	P	—	Serdes supply for S25-S32.
VDDHS2_9	B2	P	—	Serdes supply for S25-S32.
VDDHS2_10	B4	P	—	Serdes supply for S25-S32.
VDDHS2_11	B6	P	—	Serdes supply for S25-S32.
VDDHS2_12	B8	P	—	Serdes supply for S25-S32.
VDDHS2_13	B10	P	—	Serdes supply for S25-S32.
VDDHS2_14	B12	P	—	Serdes supply for S25-S32.
VDDHS2_15	B14	P	—	Serdes supply for S25-S32.
VDDHS2_16	B16	P	—	Serdes supply for S25-S32.
VDDH18_1	F10	P	—	Analog Serdes supply.
VDDH18_2	F12	P	—	Analog Serdes supply.
VDDH18_3	F14	P	—	Analog Serdes supply.
VDDH18_4	F16	P	—	Analog Serdes supply.
VDDH18_5	G10	P	—	Analog Serdes supply.
VDDH18_6	G12	P	—	Analog Serdes supply.
VDDH18_7	G14	P	—	Analog Serdes supply.
VDDH18_8	G16	P	—	Analog Serdes supply.
VDDH18_9	AE5	P	—	Analog Serdes supply.
VDDH18_10	AE19	P	—	Analog Serdes supply.
VDDH18_11	AF5	P	—	Analog Serdes supply.
VDDH18_12	AF19	P	—	Analog Serdes supply.
VDDH18_13	AF24	P	—	Analog Serdes supply.
VDDH18_14	AG24	P	—	Analog Serdes supply.
VDDIO18_1	J5	P	—	Analog Serdes supply.
VDDIO18_2	K5	P	—	Analog Serdes supply.
VDDIO18_3	AD1	P	—	Analog Serdes supply.
VDDIO18_4	AD2	P	—	Analog Serdes supply.
VDDIO18_5	AE25	P	—	Analog Serdes supply.
VDDIO18_6	AF25	P	—	Analog Serdes supply.
VDDPLLDDR_1	K26	P	—	DDR PLL supply.
VDDPLLDDR_2	L26	P	—	DDR PLL supply.
VDDPLLDDR_3	AD26	P	—	DDR PLL supply.
VDDPLLDDR_4	AE26	P	—	DDR PLL supply.
VDDIO33_1	L5	P	—	3.3V I/O power supply.
VDDIO33_2	M5	P	—	3.3V I/O power supply.

.....continued

Name	Number	Signal Type	Level	Description
VDDIO33_3	N5	P	—	3.3V I/O power supply.
VDDIO33_4	P5	P	—	3.3V I/O power supply.
VDDIO33_5	R5	P	—	3.3V I/O power supply.
VDDIO33_6	T5	P	—	3.3V I/O power supply.
VDDIO33_7	U5	P	—	3.3V I/O power supply.
VDDIO33_8	V5	P	—	3.3V I/O power supply.
VDDIO33_9	W5	P	—	3.3V I/O power supply.
VDDIO33_10	Y5	P	—	3.3V I/O power supply.
VDDIO33_11	AA5	P	—	3.3V I/O power supply.
VDDIO33_12	AB5	P	—	3.3V I/O power supply.
VDDIO33_13	AC5	P	—	3.3V I/O power supply.
VDDIO33_14	AF1	P	—	3.3V I/O power supply.
VDDIO33_15	AF2	P	—	3.3V I/O power supply.
VDDIO33_16	AF3	P	—	3.3V I/O power supply.
VDDIO33_17	AH26	P	—	3.3V I/O power supply.
VDDIO33_18	AH28	P	—	3.3V I/O power supply.
VDDIO33_19	AH30	P	—	3.3V I/O power supply.
VDDIODDR_1	M26	P	—	3.3V I/O power supply.
VDDIODDR_2	N26	P	—	3.3V I/O power supply.
VDDIODDR_3	P26	P	—	3.3V I/O power supply.
VDDIODDR_4	R26	P	—	3.3V I/O power supply.
VDDIODDR_5	T26	P	—	3.3V I/O power supply.
VDDIODDR_6	U26	P	—	3.3V I/O power supply.
VDDIODDR_7	V26	P	—	3.3V I/O power supply.
VDDIODDR_8	W26	P	—	3.3V I/O power supply.
VDDIODDR_9	Y26	P	—	3.3V I/O power supply.
VDDIODDR_10	AA26	P	—	3.3V I/O power supply.
VDDIODDR_11	AB26	P	—	3.3V I/O power supply.
VDDIODDR_12	AC26	P	—	3.3V I/O power supply.
VSS_1	C1	P	—	Ground.
VSS_2	C2	P	—	Ground.
VSS_3	C3	P	—	Ground.
VSS_4	C4	P	—	Ground.
VSS_5	C5	P	—	Ground.
VSS_6	C6	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_7	C7	P	—	Ground.
VSS_8	C8	P	—	Ground.
VSS_9	C9	P	—	Ground.
VSS_10	C10	P	—	Ground.
VSS_11	C11	P	—	Ground.
VSS_12	C12	P	—	Ground.
VSS_13	C13	P	—	Ground.
VSS_14	C14	P	—	Ground.
VSS_15	C15	P	—	Ground.
VSS_16	C16	P	—	Ground.
VSS_17	C17	P	—	Ground.
VSS_18	C19	P	—	Ground.
VSS_19	C21	P	—	Ground.
VSS_20	C23	P	—	Ground.
VSS_21	C25	P	—	Ground.
VSS_22	C26	P	—	Ground.
VSS_23	C27	P	—	Ground.
VSS_24	C28	P	—	Ground.
VSS_25	C29	P	—	Ground.
VSS_26	C30	P	—	Ground.
VSS_27	D2	P	—	Ground.
VSS_28	D4	P	—	Ground.
VSS_29	D6	P	—	Ground.
VSS_30	D8	P	—	Ground.
VSS_31	D10	P	—	Ground.
VSS_32	D12	P	—	Ground.
VSS_33	D14	P	—	Ground.
VSS_34	D16	P	—	Ground.
VSS_35	D18	P	—	Ground.
VSS_36	D20	P	—	Ground.
VSS_37	D22	P	—	Ground.
VSS_38	D24	P	—	Ground.
VSS_39	D26	P	—	Ground.
VSS_40	E2	P	—	Ground.
VSS_41	E4	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_42	E6	P	—	Ground.
VSS_43	E8	P	—	Ground.
VSS_44	E10	P	—	Ground.
VSS_45	E12	P	—	Ground.
VSS_46	E14	P	—	Ground.
VSS_47	E16	P	—	Ground.
VSS_48	E26	P	—	Ground.
VSS_49	F1	P	—	Ground.
VSS_50	F2	P	—	Ground.
VSS_51	F3	P	—	Ground.
VSS_52	F4	P	—	Ground.
VSS_53	F5	P	—	Ground.
VSS_54	F7	P	—	Ground.
VSS_55	F9	P	—	Ground.
VSS_56	F11	P	—	Ground.
VSS_57	F13	P	—	Ground.
VSS_58	F15	P	—	Ground.
VSS_59	G2	P	—	Ground.
VSS_60	G3	P	—	Ground.
VSS_61	G5	P	—	Ground.
VSS_62	G7	P	—	Ground.
VSS_63	G9	P	—	Ground.
VSS_64	G11	P	—	Ground.
VSS_65	G13	P	—	Ground.
VSS_66	G15	P	—	Ground.
VSS_67	G18	P	—	Ground.
VSS_68	G20	P	—	Ground.
VSS_69	G22	P	—	Ground.
VSS_70	G24	P	—	Ground.
VSS_71	H2	P	—	Ground.
VSS_72	H3	P	—	Ground.
VSS_73	H4	P	—	Ground.
VSS_74	H5	P	—	Ground.
VSS_75	H6	P	—	Ground.
VSS_76	H7	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_77	H8	P	—	Ground.
VSS_78	H9	P	—	Ground.
VSS_79	H10	P	—	Ground.
VSS_80	H11	P	—	Ground.
VSS_81	H12	P	—	Ground.
VSS_82	H13	P	—	Ground.
VSS_83	H14	P	—	Ground.
VSS_84	H15	P	—	Ground.
VSS_85	H16	P	—	Ground.
VSS_86	H17	P	—	Ground.
VSS_87	H18	P	—	Ground.
VSS_88	H19	P	—	Ground.
VSS_89	H20	P	—	Ground.
VSS_90	H21	P	—	Ground.
VSS_91	H22	P	—	Ground.
VSS_92	H23	P	—	Ground.
VSS_93	H24	P	—	Ground.
VSS_94	H25	P	—	Ground.
VSS_95	J1	P	—	Ground.
VSS_96	J6	P	—	Ground.
VSS_97	J7	P	—	Ground.
VSS_98	J10	P	—	Ground.
VSS_99	J11	P	—	Ground.
VSS_100	J14	P	—	Ground.
VSS_101	J15	P	—	Ground.
VSS_102	J16	P	—	Ground.
VSS_103	J17	P	—	Ground.
VSS_104	J20	P	—	Ground.
VSS_105	J21	P	—	Ground.
VSS_106	J24	P	—	Ground.
VSS_107	J25	P	—	Ground.
VSS_108	K6	P	—	Ground.
VSS_109	K7	P	—	Ground.
VSS_110	K10	P	—	Ground.
VSS_111	K11	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_112	K14	P	—	Ground.
VSS_113	K15	P	—	Ground.
VSS_114	K16	P	—	Ground.
VSS_115	K17	P	—	Ground.
VSS_116	K20	P	—	Ground.
VSS_117	K21	P	—	Ground.
VSS_118	K24	P	—	Ground.
VSS_119	K25	P	—	Ground.
VSS_120	L6	P	—	Ground.
VSS_121	L7	P	—	Ground.
VSS_122	L10	P	—	Ground.
VSS_123	L11	P	—	Ground.
VSS_124	L14	P	—	Ground.
VSS_125	L15	P	—	Ground.
VSS_126	L16	P	—	Ground.
VSS_127	L17	P	—	Ground.
VSS_128	L20	P	—	Ground.
VSS_129	L21	P	—	Ground.
VSS_130	L24	P	—	Ground.
VSS_131	L25	P	—	Ground.
VSS_132	M6	P	—	Ground.
VSS_133	M7	P	—	Ground.
VSS_134	M10	P	—	Ground.
VSS_135	M11	P	—	Ground.
VSS_136	M14	P	—	Ground.
VSS_137	M15	P	—	Ground.
VSS_138	M16	P	—	Ground.
VSS_139	M17	P	—	Ground.
VSS_140	M20	P	—	Ground.
VSS_141	M21	P	—	Ground.
VSS_142	M24	P	—	Ground.
VSS_143	M25	P	—	Ground.
VSS_144	N6	P	—	Ground.
VSS_145	N7	P	—	Ground.
VSS_146	N10	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_147	N11	P	—	Ground.
VSS_148	N14	P	—	Ground.
VSS_149	N15	P	—	Ground.
VSS_150	N16	P	—	Ground.
VSS_151	N17	P	—	Ground.
VSS_152	N20	P	—	Ground.
VSS_153	N21	P	—	Ground.
VSS_154	N24	P	—	Ground.
VSS_155	N25	P	—	Ground.
VSS_156	P6	P	—	Ground.
VSS_157	P7	P	—	Ground.
VSS_158	P10	P	—	Ground.
VSS_159	P11	P	—	Ground.
VSS_160	P14	P	—	Ground.
VSS_161	P15	P	—	Ground.
VSS_162	P16	P	—	Ground.
VSS_163	P17	P	—	Ground.
VSS_164	P20	P	—	Ground.
VSS_165	P21	P	—	Ground.
VSS_166	P24	P	—	Ground.
VSS_167	P25	P	—	Ground.
VSS_168	R6	P	—	Ground.
VSS_169	R7	P	—	Ground.
VSS_170	R10	P	—	Ground.
VSS_171	R11	P	—	Ground.
VSS_172	R14	P	—	Ground.
VSS_173	R15	P	—	Ground.
VSS_174	R16	P	—	Ground.
VSS_175	R17	P	—	Ground.
VSS_176	R20	P	—	Ground.
VSS_177	R21	P	—	Ground.
VSS_178	R24	P	—	Ground.
VSS_179	R25	P	—	Ground.
VSS_180	T6	P	—	Ground.
VSS_181	T7	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_182	T10	P	—	Ground.
VSS_183	T11	P	—	Ground.
VSS_184	T14	P	—	Ground.
VSS_185	T15	P	—	Ground.
VSS_186	T16	P	—	Ground.
VSS_187	T17	P	—	Ground.
VSS_188	T20	P	—	Ground.
VSS_189	T21	P	—	Ground.
VSS_190	T24	P	—	Ground.
VSS_191	T25	P	—	Ground.
VSS_192	U6	P	—	Ground.
VSS_193	U7	P	—	Ground.
VSS_194	U10	P	—	Ground.
VSS_195	U11	P	—	Ground.
VSS_196	U14	P	—	Ground.
VSS_197	U15	P	—	Ground.
VSS_198	U16	P	—	Ground.
VSS_199	U17	P	—	Ground.
VSS_200	U20	P	—	Ground.
VSS_201	U21	P	—	Ground.
VSS_202	U24	P	—	Ground.
VSS_203	U25	P	—	Ground.
VSS_204	V6	P	—	Ground.
VSS_205	V7	P	—	Ground.
VSS_206	V10	P	—	Ground.
VSS_207	V11	P	—	Ground.
VSS_208	V14	P	—	Ground.
VSS_209	V15	P	—	Ground.
VSS_210	V16	P	—	Ground.
VSS_211	V17	P	—	Ground.
VSS_212	V20	P	—	Ground.
VSS_213	V21	P	—	Ground.
VSS_214	V24	P	—	Ground.
VSS_215	V25	P	—	Ground.
VSS_216	W6	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_217	W7	P	—	Ground.
VSS_218	W10	P	—	Ground.
VSS_219	W11	P	—	Ground.
VSS_220	W14	P	—	Ground.
VSS_221	W15	P	—	Ground.
VSS_222	W16	P	—	Ground.
VSS_223	W17	P	—	Ground.
VSS_224	W20	P	—	Ground.
VSS_225	W21	P	—	Ground.
VSS_226	W24	P	—	Ground.
VSS_227	W25	P	—	Ground.
VSS_228	Y6	P	—	Ground.
VSS_229	Y7	P	—	Ground.
VSS_230	Y10	P	—	Ground.
VSS_231	Y11	P	—	Ground.
VSS_232	Y14	P	—	Ground.
VSS_233	Y15	P	—	Ground.
VSS_234	Y16	P	—	Ground.
VSS_235	Y17	P	—	Ground.
VSS_236	Y20	P	—	Ground.
VSS_237	Y21	P	—	Ground.
VSS_238	Y24	P	—	Ground.
VSS_239	Y25	P	—	Ground.
VSS_240	AA6	P	—	Ground.
VSS_241	AA7	P	—	Ground.
VSS_242	AA10	P	—	Ground.
VSS_243	AA11	P	—	Ground.
VSS_244	AA14	P	—	Ground.
VSS_245	AA15	P	—	Ground.
VSS_246	AA16	P	—	Ground.
VSS_247	AA17	P	—	Ground.
VSS_248	AA20	P	—	Ground.
VSS_249	AA21	P	—	Ground.
VSS_250	AA24	P	—	Ground.
VSS_251	AA25	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_252	AB6	P	—	Ground.
VSS_253	AB7	P	—	Ground.
VSS_254	AB10	P	—	Ground.
VSS_255	AB11	P	—	Ground.
VSS_256	AB14	P	—	Ground.
VSS_257	AB15	P	—	Ground.
VSS_258	AB16	P	—	Ground.
VSS_259	AB17	P	—	Ground.
VSS_260	AB20	P	—	Ground.
VSS_261	AB21	P	—	Ground.
VSS_262	AB24	P	—	Ground.
VSS_263	AB25	P	—	Ground.
VSS_264	AC6	P	—	Ground.
VSS_265	AC7	P	—	Ground.
VSS_266	AC8	P	—	Ground.
VSS_267	AC9	P	—	Ground.
VSS_268	AC10	P	—	Ground.
VSS_269	AC11	P	—	Ground.
VSS_270	AC12	P	—	Ground.
VSS_271	AC13	P	—	Ground.
VSS_272	AC14	P	—	Ground.
VSS_273	AC15	P	—	Ground.
VSS_274	AC16	P	—	Ground.
VSS_275	AC17	P	—	Ground.
VSS_276	AC18	P	—	Ground.
VSS_277	AC19	P	—	Ground.
VSS_278	AC20	P	—	Ground.
VSS_279	AC21	P	—	Ground.
VSS_280	AC22	P	—	Ground.
VSS_281	AC23	P	—	Ground.
VSS_282	AC24	P	—	Ground.
VSS_283	AC25	P	—	Ground.
VSS_284	AD3	P	—	Ground.
VSS_285	AD4	P	—	Ground.
VSS_286	AD5	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_287	AD7	P	—	Ground.
VSS_288	AD9	P	—	Ground.
VSS_289	AD11	P	—	Ground.
VSS_290	AD13	P	—	Ground.
VSS_291	AD15	P	—	Ground.
VSS_292	AD17	P	—	Ground.
VSS_293	AD19	P	—	Ground.
VSS_294	AD21	P	—	Ground.
VSS_295	AD23	P	—	Ground.
VSS_296	AD25	P	—	Ground.
VSS_297	AE1	P	—	Ground.
VSS_298	AE2	P	—	Ground.
VSS_299	AE3	P	—	Ground.
VSS_300	AF4	P	—	Ground.
VSS_301	AG4	P	—	Ground.
VSS_302	AG7	P	—	Ground.
VSS_303	AG9	P	—	Ground.
VSS_304	AG11	P	—	Ground.
VSS_305	AG13	P	—	Ground.
VSS_306	AG15	P	—	Ground.
VSS_307	AG17	P	—	Ground.
VSS_308	AG19	P	—	Ground.
VSS_309	AG21	P	—	Ground.
VSS_310	AG23	P	—	Ground.
VSS_311	AG26	P	—	Ground.
VSS_312	AG30	P	—	Ground.
VSS_313	AH4	P	—	Ground.
VSS_314	AH6	P	—	Ground.
VSS_315	AH8	P	—	Ground.
VSS_316	AH10	P	—	Ground.
VSS_317	AH12	P	—	Ground.
VSS_318	AH14	P	—	Ground.
VSS_319	AH16	P	—	Ground.
VSS_320	AH18	P	—	Ground.
VSS_321	AH19	P	—	Ground.

.....continued

Name	Number	Signal Type	Level	Description
VSS_322	AH20	P	—	Ground.
VSS_323	AH22	P	—	Ground.
VSS_324	AH24	P	—	Ground.
VSS_325	AH27	P	—	Ground.
VSS_326	AH29	P	—	Ground.
VSS_327	AJ4	P	—	Ground.
VSS_328	AK4	P	—	Ground.

9. Package Information

The SparX-5 (VSC7546YKY and VSC7549YKY) package is lead-free (Pb-free), 888-pin, flip chip ball grid array (FCBGA) with a 25 mm × 25 mm body size, 0.8 mm pin pitch, and 2.56 mm maximum height.

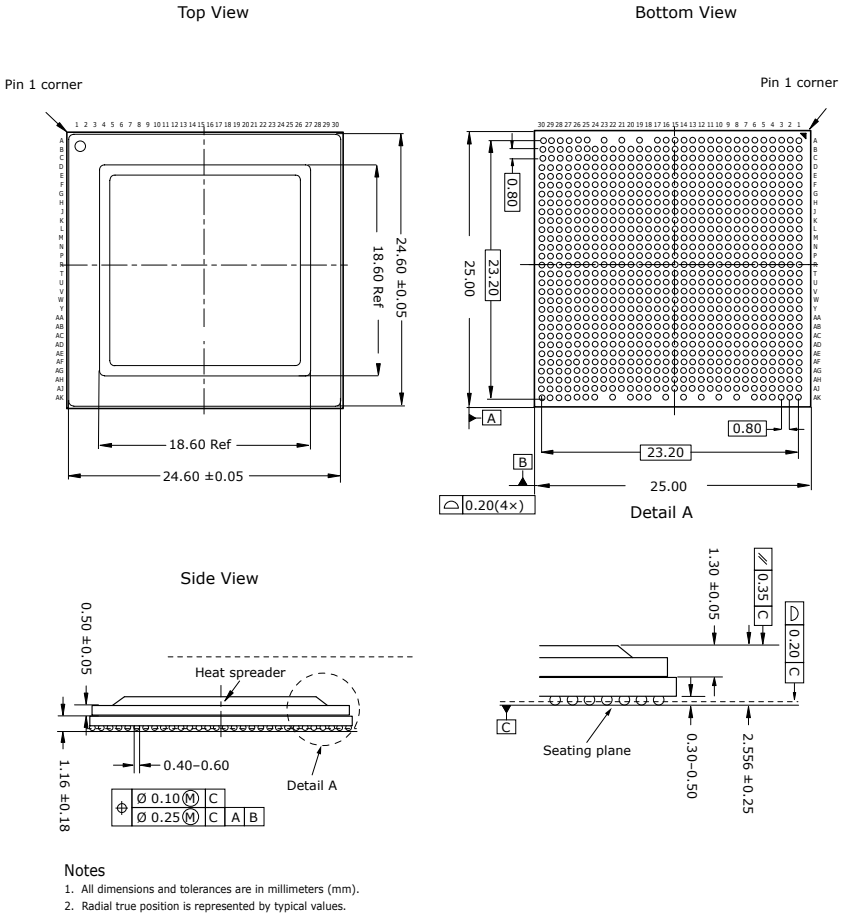
Lead-free products from Microchip comply with the temperatures and profiles defined in the joint IPC and JEDEC standard IPC/JEDEC J-STD-020. For more information, see the IPC and JEDEC standard.

This section provides the package drawing, thermal specifications, and moisture sensitivity rating for the device.

9.1 Package Drawing

The following illustration shows the package drawing for the devices. The drawing contains the top view, bottom view, side view, detail views, dimensions, tolerances, and notes.

Figure 9-1. Package Drawing



9.2 Thermal Specifications

Thermal specifications for these devices are based on the JEDEC JESD51 family of documents. These documents are available on the JEDEC Web site at www.jedec.org. The thermal specifications are modeled using a four-layer test board with two signal layers, a power plane, and a ground plane (2s2p PCB). For more information about the thermal measurement method used for this device, see the JESD51-1 standard.

Table 9-1. Thermal Resistances

Symbol	°C/W	Parameter
θ_{JCtop}	0.91	Die junction to package case top

.....continued		
Symbol	**°C/W**	**Parameter**
θ_{JB}	8.1	Die junction to printed circuit board
θ_{JA}	19.8	Die junction to ambient
θ_{JMA} at 1 m/s	14.5	Die junction to moving air measured at an air speed of 1 m/s
θ_{JMA} at 2 m/s	12.5	Die junction to moving air measured at an air speed of 2 m/s

To achieve results similar to the modeled thermal measurements, the guidelines for board design described in the JESD51 family of publications must be applied. For information about applications using FCBGA packages, see the following:

- JESD51-2A, Integrated Circuits Thermal Test Method Environmental Conditions, Natural Convection (Still Air)
- JESD51-6, Integrated Circuit Thermal Test Method Environmental Conditions, Forced Convection (Moving Air)
- JESD51-8, Integrated Circuit Thermal Test Method Environmental Conditions, Junction-to-Board
- JESD51-9, Test Boards for Area Array Surface Mount Package Thermal Measurements

9.3 Moisture Sensitivity

This device is rated moisture sensitivity level 4 as specified in the joint IPC and JEDEC standard IPC/JEDEC J-STD-020. For more information, see the IPC and JEDEC standard.

10. Ordering Information

The SparX-5 (VSC7546-V/5CC and VSC7549-V/5CC) package is lead-free (Pb-free), 888-pin, flip chip ball grid array (FCBGA) with a 25 mm × 25 mm body size, 0.8 mm pin pitch, and 2.56 mm maximum height.

Lead-free products from Microchip comply with the temperatures and profiles defined in the joint IPC and JEDEC standard IPC/JEDEC J-STD-020. For more information, see the IPC and JEDEC standard.

The following table lists the ordering information for the devices.

Table 10-1. Ordering Information

Part Order Number	Description
VSC7546-V/5CC	64 Gbps Enterprise Ethernet switch Lead-free, 888-pin FCBGA with a 25 mm × 25 mm body size, 0.8 mm pin pitch, and 2.56 mm maximum height
VSC7549-V/5CC	90 Gbps Enterprise Ethernet switch Lead-free, 888-pin FCBGA with a 25 mm × 25 mm body size, 0.8 mm pin pitch, and 2.56 mm maximum height

11. Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Table 11-1. Revision History

Revision	Date	Description
C	April 2021	The following is a summary of changes in revision C of this document. • Removed single port USXGMII support from the 10G USXGMII Line Ports section. For more information, see 3.1.8 10G USXGMII Line Ports. • Updated the SerDes section. For more information, see 4.4.1.3 SERDES25G. • Updated the SerDes Reference Clock Inputs table. For more information, see Table 7-2. • Updated the SERDES25G section. For more information, see 7.1.6 SERDES25G. • Updated the GPIO, MIM, SI, JTAG, and Miscellaneous Signals DC Specifications table. For more information, see Table 7-14. • Updated the 25G Transmitter table. For more information, see Table 7-9.
B	February 2021	The following is a summary of changes in revision B of this document. • Corrected the Mapping of VCore sub-CPU system to VCore Shared Bus table. For more information, see Figure 5-35. • Corrected the DDR (DDR3/DDR4) Memory Controller section. There were conditional issues, which created faulty output. For more information, see 5.8 DDR (DDR3/DDR4) Memory Controller. • Updated the SerDes section. For more information, see 7.2.2 SerDes. • Updated the DDR SDRAM Interface section. For more information, see 7.2.8 DDR SDRAM Interface.
A	January 2021	Revision A was published in January 2021. It was the first publication of this document.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-8087-7

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820