Scale-Up Ethernet Framework

Scale-Up Ethernet Framework Specification

Table of Contents

Chapter 1: Introduction

As workloads continue to expand in complexity in areas such as machine learning and AI inference, the need for parallel processing power increases significantly. Scaling up a GPU (or more generically an XPU, to include various custom ML accelerators) cluster to a rack or multi-rack level is needed to enhance its ability to process larger datasets, train deeper neural networks, handle more simultaneous tasks, and reduce execution time and improve overall system efficiency.

Ethernet offers many advantages for designing such a fabric, from industry leading high-speed links and high-capacity switches to a well-developed ecosystem and well-understood operational methods. Multiple industry groups are currently involved in developing networking technologies for AI-based networks that extend Ethernet or use some of its components as a building block.

SUE provides a framework to provide low-latency, high-bandwidth connectivity for XPU scale up networks based on Ethernet. The intent of Scale-Up Ethernet (SUE) is to provide the transport and Ethernet datalink to move memory transactions between XPUs.

1.1 Deployment Model

SUE provides communication between XPUs in a multi-XPU system. Each SUE provides network connectivity that can be configured as one, two, or four ports. For example, an 800G SUE instance supports 1x 800G, 2x 400Gbps, or 4x 200Gbps. The primary network configuration is expected to be a single switch hop. The port configuration is chosen based on the required switch radix and SerDes count, as well as redundancy and failover considerations.

SUE is designed to enable multiple instances of SUE per XPU. For example, a single XPU can include 8 or 16 SUE instances. Figure 1 shows an example configuration of 64 XPUs with twelve 800G SUE instances per XPU. Twelve Ethernet switches are used with 64 ports each. In this example, with fully switched connectivity, any pair of XPUs has up to 9.6 Tb/s (12x 800G) bandwidth between them.

Alternatively, SUE can be used in a mesh with direct connection between a pair of XPUs, as illustrated in Figure 2.

Figure 1: Example Single Hop Switched Deployment

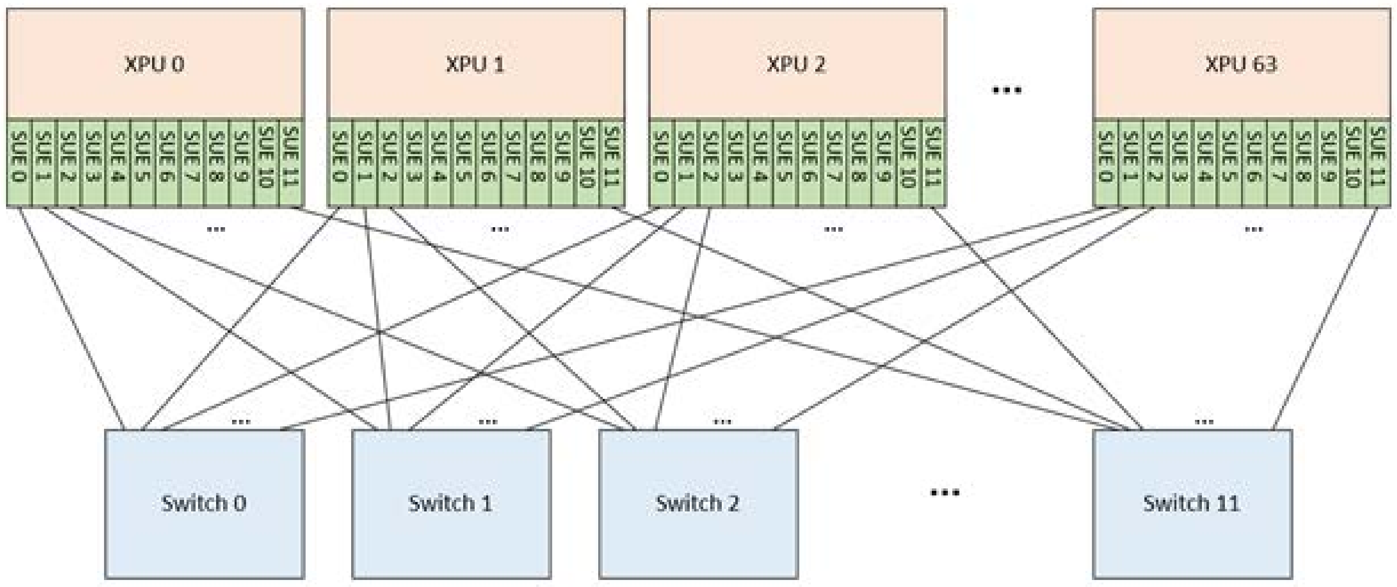
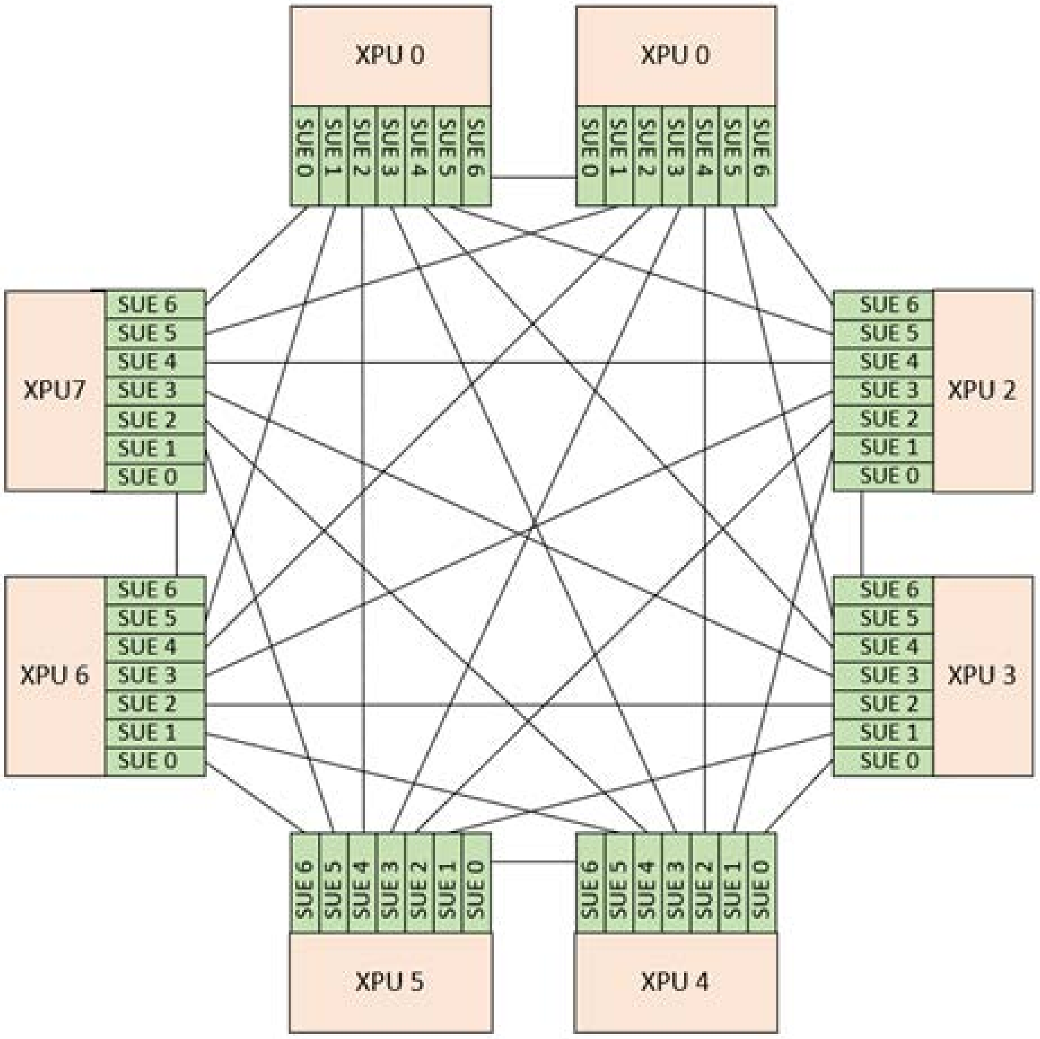


Figure 2: Example Mesh Deployment



1.2 SUE Overview

SUE provides a mechanism for generic command/response transactions. The set operations and structure of the commands are XPU-specific and transparent to SUE. This command channel can be used to implement a variety of services, such as put, get, and atomic operations. Additional services such as cache coherency can be defined. However, full implementation of these services is beyond the scope of SUE.

SUE views all transactions as commands. The XPU supports command/response by defining appropriate opcodes (for example, read and read response) and by mapping these to different traffic classes.

Figure 3 provides an overview of the SUE stack. The XPU NOC sends a command to SUE. SUE provides a signal duplex data interface that is similar to AXI or an AXI4 interface. This interface includes control and data. The control content is largely opaque. It includes a command, the remote XPU identifier, and fields used to determine if data is present as well as length information.

The mapping and packing layer organizes the command by destination and opportunistically packs multiple commands into a single SUE protocol data unit (PDU). Each SUE PDU is destined to a single XPU on a single traffic class. Virtual channel (VC) in the control is used to map the packet to the appropriate traffic class.

SUE provides two ordering modes: strict ordering and unordered. In strict mode, all transactions between a source and destination on a particular VC are passed from the SUE transport to the destination XPU in order. When configured for more than one port, SUE supports an unordered mode where the transactions are load balanced across the ports and order is not maintained.

Each SUE PDU is mapped to a single Ethernet packet. Reliability, data integrity, and encryption are provided at the packet level.

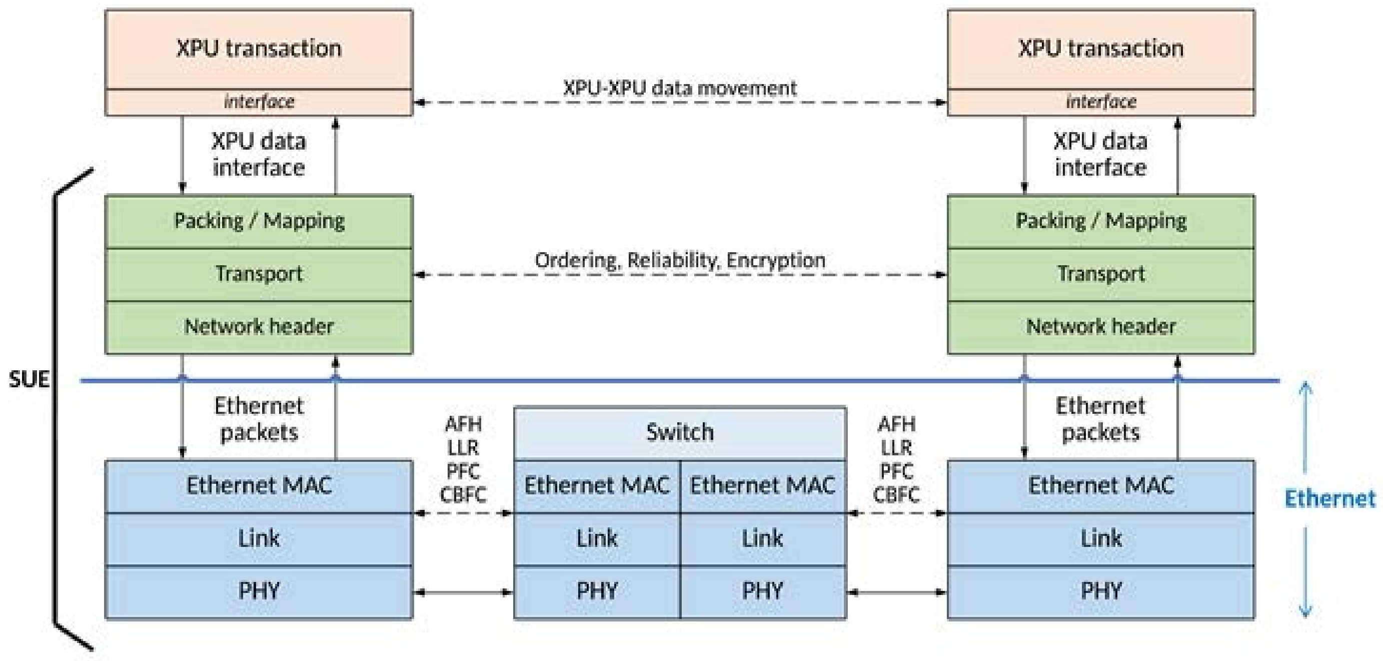
The network layer builds the header based on the destination XPU. This can be a standard Ethernet header or use an optimized format, as described in Section 3.5.1, Network Encapsulation.

Link Layer Retransmission (LLR) and priority flow control (PFC) or credit-based-flow control (CBFC) are used to provide lossless service. Above this, the SUE transport provides retransmission should an unrecoverable error occur.

At the destination, the SUE PDU is unpacked and each command is passed to the XPU.

Figure 3: SUE Stack

s



Chapter 2: SUE Requirements

The design philosophy of SUE is to provide low latency and high bandwidth in a manner that allows highly efficient implementations, enabling multiple instances to be instantiated with efficient area and power. Based on this philosophy, the following requirements were used to guide the development of SUE and are not strictly required. SUE can be adapted to specific use cases.

Table 1: SUE Requirements

	Requirements	**Notes**
Number of XPUs	Up to 1024	Single hop, low latency
Transaction types	One-sided operations[a] Memory load-store-atomics Smaller transfers	—
Memory architecture	Shared memory	For example, PGAS, registration outside scope of SUE
E2E latency	< 2us RTT	Round trip time
Cable length	Up to 10m from SUE to switch	For latency
Bandwidth per SUE	800Gbps	Scale to 1.6Tbs
SerDes	200Gbps	Supports 100G and 50G
Port configuration	1, 2 or 4	Enable higher radix switches, redundancy and failover
Virtual channels	Up to 4	Independent traffic classes selected using the qos field, avoid deadlock scenarios
ETH compliance	Support standard Ethernet	Provide option to compressed header format, LLR, CBFC

a. In this case, one-sided means that the operation does not involve awareness of the process(es) at the destination. The operation is acknowledged so that the source knows the operation successfully reached the destination.

SUE is not a general purpose network interface with a full suite of services available in modern NICs. It also does not define an XPU-level protocol to enable interoperation between different XPU devices.

2.1 Protocol Requirements: Background

There is a large amount of parallelism in XPUs and the scale-up transport protocol must enable XPU to XPU connectivity concurrently across many planes. To achieve power and area efficiency, and because the deployment topologies for XPU to XPU are constrained, a relatively simple transport is used. The transport protocol needs to provide for the reliable delivery of higher-layer data from the source to destination XPU across a single switch hop.

The scale-up topology can be both multi-plane and multi-rail, the same as the scale-out network. The key part of building an efficient transport protocol for such networks is effectively load balancing the data across the available planes and handling many-to-one traffic (incast) traffic patterns without causing head-of-line blocking in the network.

In the scale-out topology, this is achieved by having the transport layer explicitly include multipath operation, which is done today using QP scaling in collective libraries such as NCCL for RoCEv2, and natively by packet spraying transports such as Ultra Ethernet Transport. To deal with incast traffic patterns, PFC is used by RoCEv2 but deprecated in next generation transport to avoid the performance anomalies caused by head-of-line blocking; instead packet trimming is used where the networks avoid silent packet drops by forwarding the header of the dropped packet with priority to the destination.

Adopting the same solution for scale-up networks is not practical. Scale-up networks have different requirements (much higher bandwidth, lower RTT, smaller scale, and so on), which leads to different trade-offs.

Scale-out networks leveraging packet spraying are complex because they must support network reordering, which requires per PSN and per-path tracking at both the sender and the receiver. This complexity is useful for the scale-out network to avoid collisions but is simply not needed for the scale-up network: collisions within a plane cannot appear in a single-tier topology. The scale-out approach to use trimming and selective retransmissions is very useful for larger networks, but it is also complex as it requires packet tracking bitmaps and selective acknowledgments, each of which increase area and cost.

Scale-up transport speeds are expected to be an order of magnitude larger than scale-out speeds. The current bandwidth supported by scale-out NICs is 800Gbps, implying that many such NICs would be required for the scale-up network. This is challenging from an area and power perspective.

Finally, the endpoints that the scale-up transport must serve are all known in advance (all active XPUs). Connections can be set up statically at boot time and kept alive as long as both the plane and target XPU are alive. This further simplifies connection setup and scaling decisions in the transport protocol.

A much simpler solution is to use a single-path transport per plane. Relying on lossless operation results in a much simpler transport, where simple go-back-N is used to recover from infrequent losses, resulting in significantly smaller transport state and simpler send/receive processing. To further reduce packet loss caused by corruption, LLR (a switch feature that retransmits packets when the FCS checksum fails at the receiver) will be enabled when available.

To efficiently support a go-back-N transport, the network will be configured to enable lossless operation between XPUs either by using PFC or the newer CBFC. To avoid triggering flow control and the resulting head-of-line blocking, state-of-the-art congestion control should be used at the endpoints.

Using multiple network interfaces per XPU implies that traffic must be load balanced across those interfaces by an external module (for example, the XPU). When a single SUE instance provides multiple ports, that network instance can load balance only across its ports/planes when in unordered mode.

In other words, the *load balancing layer must spread traffic across all the planes of all its scale-up interfaces*. This is done above SUE.

Given the need for near-optimal performance, the per-plane transport must start at line rate. The only performance problem that can appear in a single-tier network is incast, when multiple senders start sending at the same time towards the same receiver.

2.2 SUE Lite Simplifications

The SUE Lite profile was created to reduce the size of SUE IP by up to 50 percent. In SUE Lite, the SUE reliable transport layer is removed and LLR is used to provide reliability between nodes. As a result of removing the reliable transport layer, congestion control is also removed. To further reduce size, the packing size is limited to 1K bytes and there is only a signal interface to the XPU. The Ethernet Port (MAC/Link/PHY) size remains unchanged. The SUE Lite stack is illustrated in Figure 4. The differences between the SUE and SUE Lite profiles are listed in Table 2.

Figure 4: SUE Lite Stack

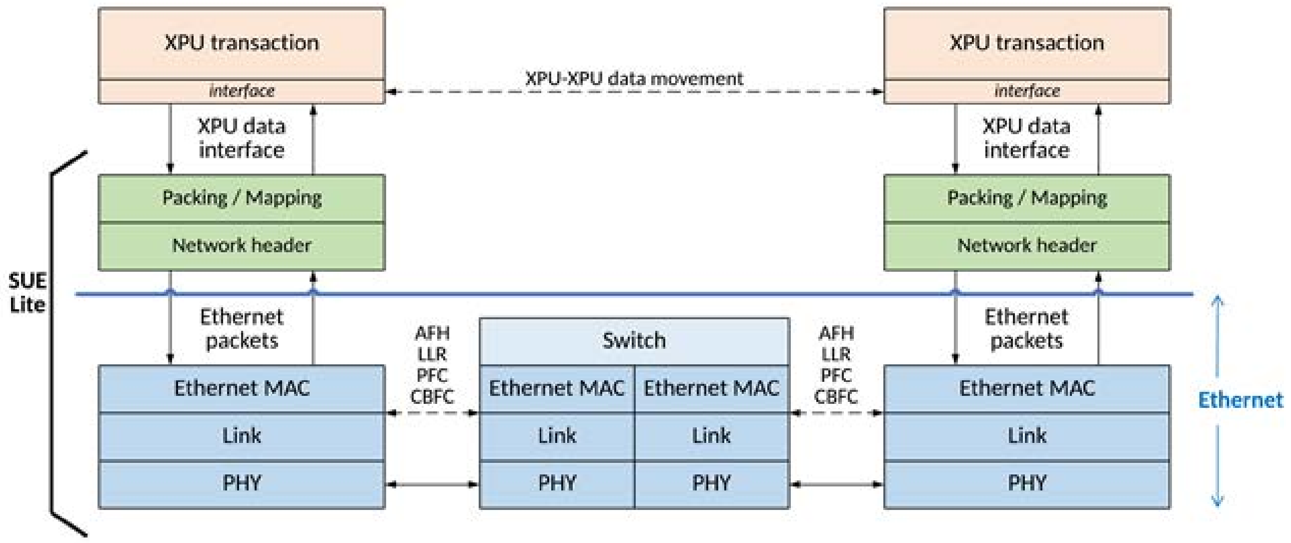


Table 2: SUE and SUE Lite Comparison

Attributes	SUE	SUE Lite
Transaction types	Write (Full/Partial, Posted/Non-Posted), Read, Message, Barrier	
Transaction size	256B	
DMA ready interface	Yes (Block write)	
Transaction packing	Yes	
LLR, CBFC, AFH Headers	Yes	
Core side interface	AXI4, Signal-based interface	Signal-based interface
End-to-end reliability	Yes (Reliable Transport Layer)	No (Hop-by-hop LLR only)
Congestion control	Yes (Fixed Window)	No
Partition Feature	Yes	No (Use address separation)

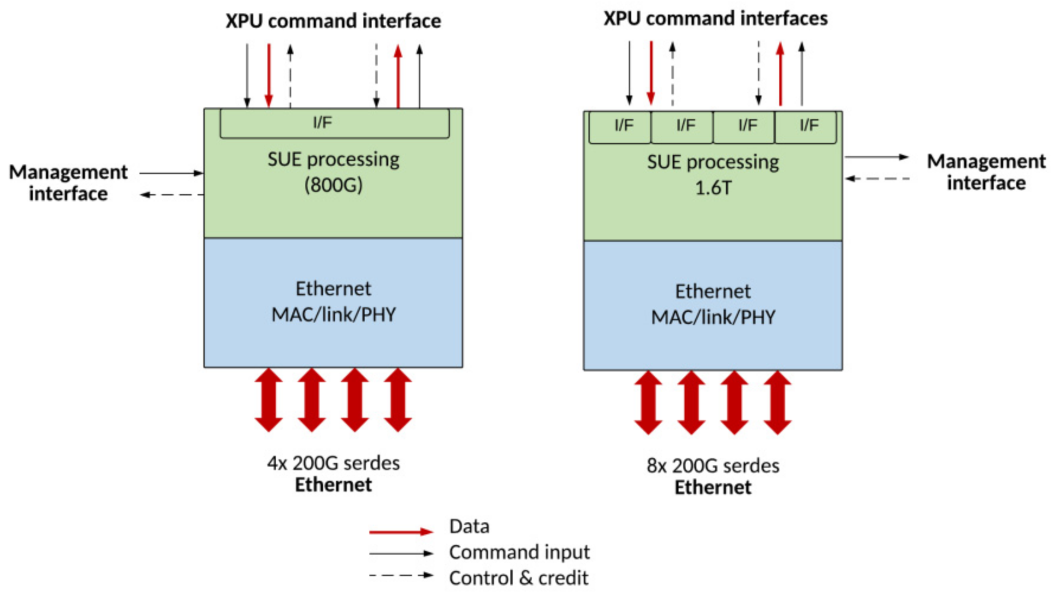
Chapter 3: SUE Interfaces

Each SUE instance has the following three interfaces, as illustrated in Figure 5:

- XPU command interface
- XPU management interface
- Ethernet interface

A description of each of these is provided in the following subsections.

Figure 5: SUE Interfaces



3.1 XPU Management Interface

SUE uses an AXI target interface for control. Register access is provided.

In addition to configuration and status, the control interface can be used to send and receive packets. The XPU builds the full packet with SUE adding the Ethernet FCS. This transport is unreliable. It supports rates up to 10K packets per second.

3.2 XPU Command Interface

The XPU to SUE interface can be one of the following types:

- Wire Interface, FIFO interface using credits
- AXI4 interface

The reason to choose one over the other is XPU-specific. A particular XPU can implement messages that do not map into an AXI4 interface, requiring a more open custom interface between the XPU's NOC and SUE. Figure 5 illustrates two different implementations of SUE IP. The first is an 800G implementation with a single 800G interface to the XPU NOC and a 4x200G Ethernet MAC/Link/PHY. The second implementation is a 1.6T implementation with 4x 400G interfaces to the XPU NOC and an 8x200G Ethernet MAC/Link/PHY. The exact configuration of the interface from the XPU NOC to SUE is up to the implementer.

3.2.1 XPU to SUE FIFO Interface

The XPU-SUE interface is essentially a FIFO interface using credits. The XPU writes a command to the SUE, as well as any associated data. SUE generates flow control on a per {destination, VC} basis and on an aggregate rate basis using a credit mechanism.
To initiate transfer of data, the XPU sends the information listed in Table 3 to SUE over the command interface. Input refers to signals from XPU to SUE.

Table 3: XPU to SUE Signal Interface

Field Name	I/O	Size (Bits)	Description
tx_cycle	Input	4	Data cycle input number, transfer up to 256B of data. The number of cycles is implementation specific depending on data bus width.
tx_valid	Input	2	Type of operation: none, command only, command with data.
tx_vc	Input	2	Virtual channel.
xpuid	Input	10	Remote XPU identifier.
tx_control	Input	144	Control fields, processed and packed into transaction to be sent to remote XPU. This carries command and length information. First bytes include command and length.
tx_data	Input	N	Data, serialized over multiple cycles depending on N. For example, if N = 256 bits (32B) then 8 cycles are used to transfer 256B.
tx_err	Input	1	Uncorrectable ECC error on interface, used to support ECC generation and checking where a Die-to-Die interface is present.
tx_credit	Output	6	Credit and valid for cmd and cmd+data, per VC.
tx_reset_in	Input	1	Reset.
tx_reset_out	Output	1	Indicates SUE is in reset.

As packets arrive, SUE verifies the network header and SUE PDU. It then unpacks and sends the transactions over the command interface. Table 4 lists the SUE to XPU signals on this interface.

Table 4: SUE to XPU Signal Interface

Field Name	I/O	Size (Bits)	Description
rx_cycle	Output	4	Data cycle input number, transfer up to 256B of data. The number of cycles is implementation specific depending on data bus width.
rx_valid	Output	2	Type of operation: none, command only, command with data.
rx_vc	Output	2	Virtual channel,
xpuid	Output	10	Remote XPU identifier.
rx_control	Output	144	Control fields, processed and packed into transaction to be sent to remote XPU. This carries command and length information. First bytes include command and length.
rx_data	Output	N	Data, serialized over multiple cycles depending on N. For example, if N = 256 bits (32B) then 8 cycles are used to transfer 256B.
rx_credit	Input	6	Credit & valid for cmd and cmd+data.
rx_reset_in	Input	1	Reset.
rx_reset_out	Output	1	Indicates SUE is in reset.

3.2.2 XPU to SUE AXI Interface

SUE can alternatively expose an AXI4 slave and AXI4 master interface per 400G Ethernet interface. Each AXI interface supports 5 independent channels as defined in the AXI4 specification (AW, W, B, AR, and R channels).

Each master and slave channel is associated with a separate credit interface. The credit interface returns credits from receiver to the sender. The credit loops are used to prevent overflow of downstream buffers.

Figure 6: SUE Datapath AXI Interface

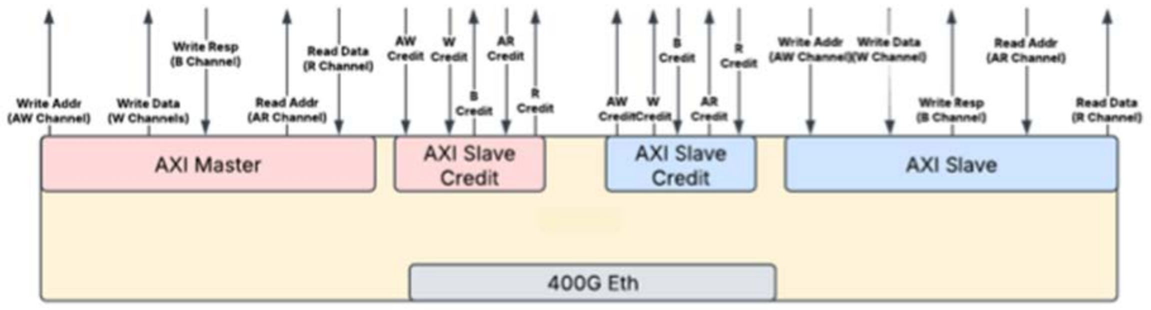
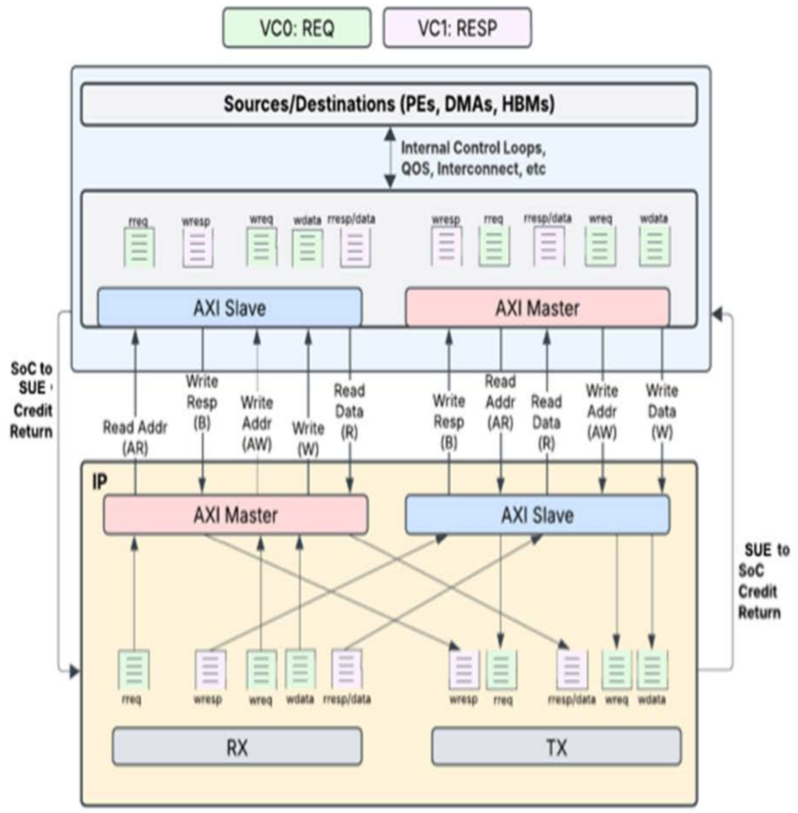


Figure 7 shows the AXI interface connectivity between the SoC logic and SUE in context with associated buffering and functionality and VC mapping.

Figure 7: Datapath AXI Interface Connectivity

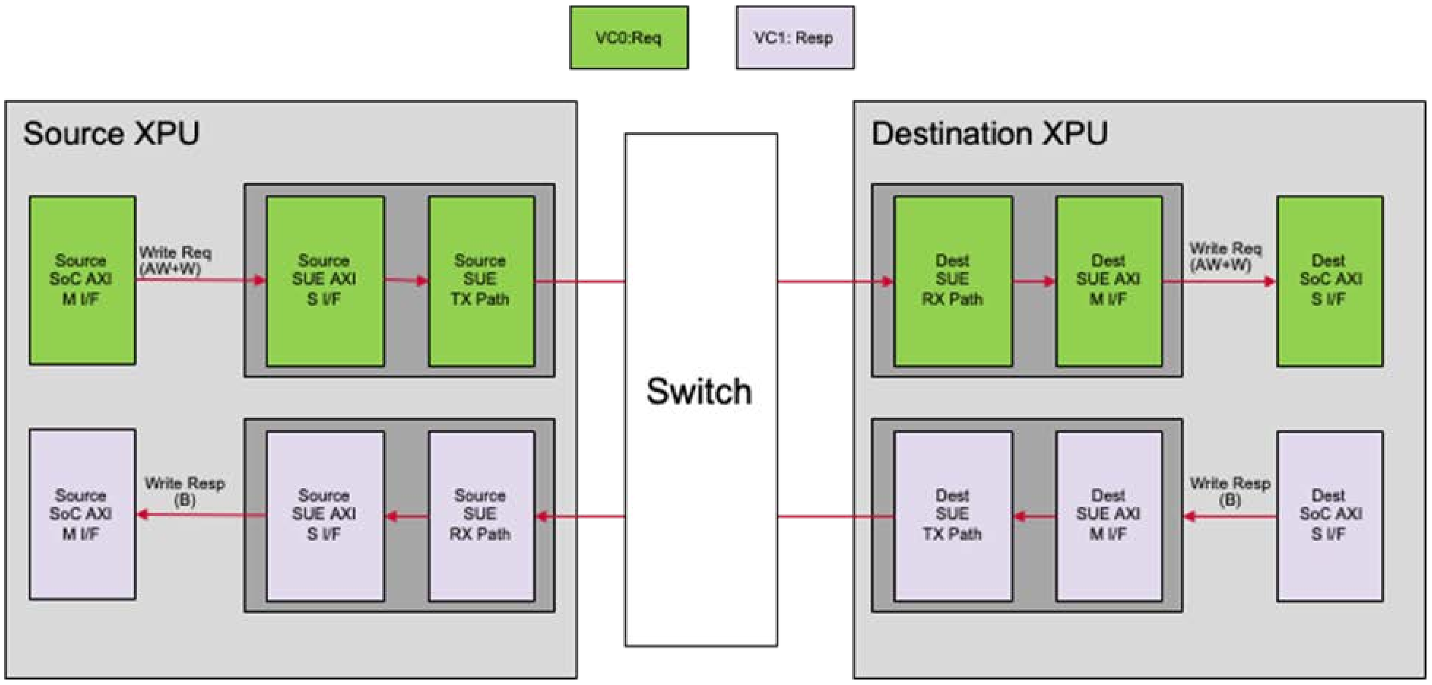


AXI transactions consist of requests (AR, AW, W transactions) and responses (B, R transactions). Requests are mapped to VC0 and responses are mapped to VC1. As shown in the previous figure, the IP instantiates buffers at the interface for each AXI channel interface (these are color coded as VC0 or VC1 in the figure). These buffers store "unpacked" AXI transactions. The interface buffers are relatively small and are separate from the larger "packed" data buffers that reside within TX and RX datapaths. The AXI USER bits (AWUSER, BUSER, ARUSER, RUSER) are used to carry xpu_id, type, and length information.

As shown in Figure 7, the SoC must instantiate per channel buffers near to SUE. This keeps the buffers shallow and independent of the architecture and implementation of the SoC. Buffers that receive transactions from SUE (Master.AR, Master.AW, Master.W, Slave.B, Slave.R), return credit to the IP as space is freed. Buffers that send transactions into the IP (Master.B, Master.R, Slave.AR, Slave.AW) track available credit—decrementing credit counters per each transaction sent and incrementing credit counters for each credit returned. The XPU-to-SUE channel interfaces are grouped within SUE into control (write resp + read req) and control/data (read resp/data, write req/write data). Write requests and write data use an independent channel, but transactions across the two are paired and arrive in the same order (per the AXI4 specification). Internally, SUE aligns these paired transactions at the head of the two interface buffers before moving them into the TX datapath as a single unit.

Figure 8 shows the AXI interfaces that are used to transfer the request and resulting response transactions between two XPU endpoints. The figure shows the write request/response flow. Read requests/responses follow the same flow, using the AR (read request) channel and R (read data) channel. In the case of reads, the read request channel is mapped to VC0 and the read data channel is mapped to VC1. The write response is either an ACK or NACK carried in the header of packets flowing in the reverse direction.

Figure 8: Datapath AXI Interface Connectivity (Write/Write Response)



The source XPU SoC logic issues a request from its master into the SUE slave. The request flows through the source SUE transmit datapath, the switch, the destination SUE's receive datapath and AXI master interface, and into the SoC's slave interface. The request uses VC0 end-to-end in this path.

The response flows in the reverse direction, starting from the destination SoC's AXI slave, it flows through the network and into the source XPU. The response is transferred back to the source SoC through the response channel on its master interface.

The bits on the AXI interfaces are grouped into AXI control bits and AXI data bits. The fields within these groupings are mostly opaque to the SUE. With a few exceptions, the bits are packed and transported to the destination without being interpreted or consumed.

3.3 SUE Encapsulation

SUE opportunistically packs the received transactions to the same {destination, VC}, creating up to 4096 bytes of packed SUE PDU. When scheduled for transmission, a reliability header (Figure 9) is added to the header of the packed PDU and a 32-bit CRC (R-CRC) is added to the tail for data integrity.

The RH carries the fields listed in Table 5 and illustrated in Figure 9.

Figure 9: SUE Reliability Header

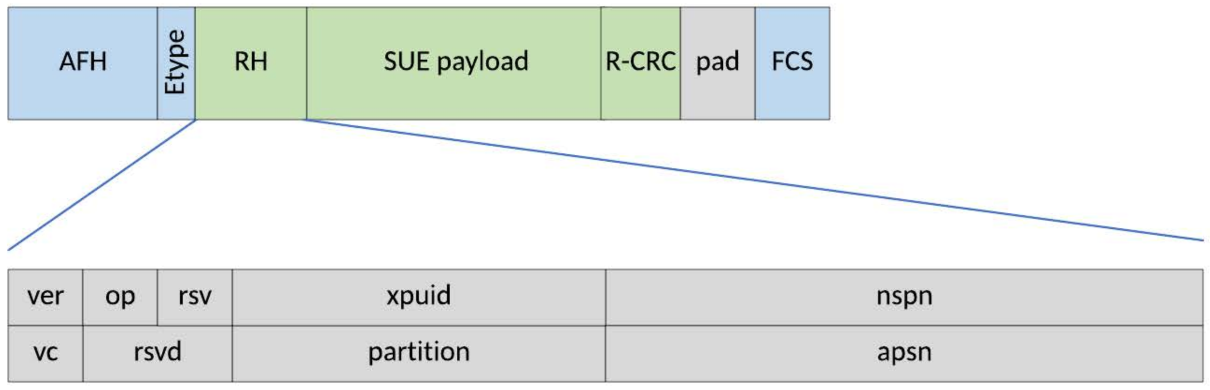


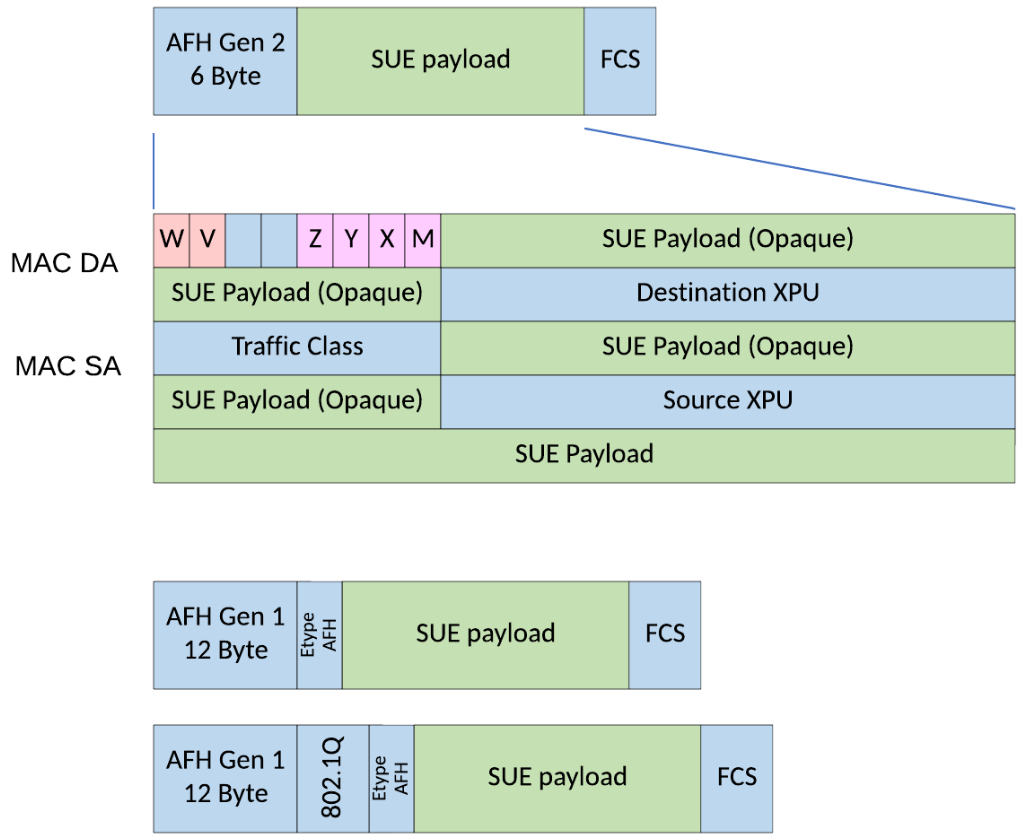
Table 5: SUE RH Fields

Field Name	Size (bits)	Description
ver	2	Version
op	2	Opcode indicating if apsn is an ACK, NACK or invalid
rsv	2	Reserved
xpuid	10	XPU identifier
npsn	16	Packet sequence number
vc	2	Virtual channel, maps to traffic class
rsvd	4	Reserved
partition	10	Used to provide multi-tenant isolation
apsn	16	ACK or NACK packet sequence number

3.4 SUE Lite Encapsulation

The SUE Lite profile removes the SUE transport layer. The only information SUE Lite uses are the Destination, Source XPU address, and VC—these all fit into the AFH Gen 2 6 byte header. The R-CRC to protect the payload is also removed to further optimize the overhead. The SUE Lite encapsulation is illustrated in Figure 10. SUE Lite opportunistically packs the received transactions to the same {destination, VC}, creating up to 1K bytes of packed SUE Lite PDU.

Figure 10: SUE Lite Encapsulation



3.5 Ethernet Interface

SUE provides standard Ethernet interface(s) using 200G or 100G SerDes rates. Table 6 describes the supported configurations for an 800G SUE instance with an 800G XPU-SUE interface. Table 7 describes the supported configurations for a 1.6T SUE instance with a 4x400G XPU-SUE interface

Table 6: Ethernet Ports per 800G SUE Instance Example

Ethernet Port Speed	# Ports per SUE Instance	SerDes Lane Speed	# SerDes Lane per Ethernet Port
800G	1	200G	4
400G	2	200G	2
200G	4	200G	1
400G	1	100G	4
200G	2	100G	2
100G	4	100G	1

Table 7: Ethernet Ports per 1.6T SUE with 4x400G XPU to SUE Instance Example

Ethernet Port Speed	# Ports per SUE Instance	SerDes Lane Speed	# SerDes Lane per Ethernet Port
400G	4	200G	2
200G	8	200G	1
400G	2	100G	4
200G	4	100G	2
100G	8	100G	1

3.5.1 Network Encapsulation

SUE uses the destination XPU identifier, xpuid, and VC to build the network header. SUE can forward memory transactions using one of the following methods:

- Standard Ethernet formats - Ethernet Header, IPv4/IPv6, UDP.
- AI Forwarding Header Gen 1 (AFH Gen 1) - A Layer 2 format that maintains the existing Ethernet MAC destination address and source address formatting. The header can use less bits (16 bits to 32 bits) in the MAC address for the forwarding lookup. The Ethertype field identifies the packet as AFH Gen 1. A second optional Ethertype field can be added to the packet if you need to disambiguate multiple transport headers.
- AI Forwarding Header Gen 2 (AFH Gen 2) - A more optimized Layer 2 format, where the forwarding information is reduced to 6 or 12 bytes. The remaining bytes in the Ethernet destination address and source address can be used for user-defined functionality. The Ethertype field is used to disambiguate multiple transport headers.

When SUE is using standard Ethernet and IP headers, the SUE packet is identified by UDP port number (Figure 11). When SUE utilizes AFH Gen 1, the SUE packets are identified by the Ethertype field (Figure 12). When SUE uses AFH Gen 2, it is based on the Structured Local Address Plan in IEEE 802.c-2017, which defines two options: 12B and 6B headers. The AFH formats reduce the overhead on the wire.

Figure 11: Standard Ethernet Format

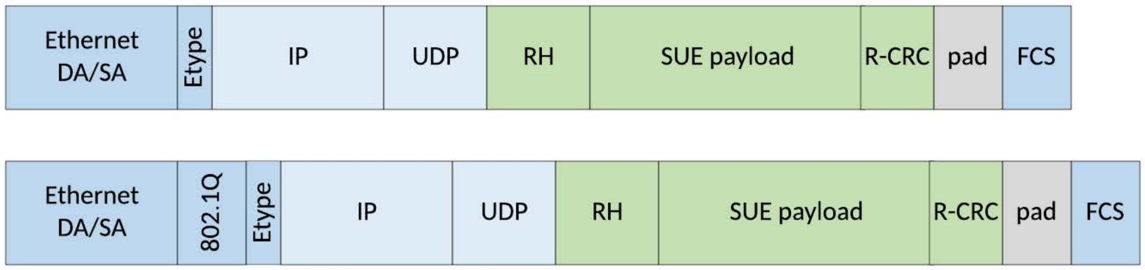
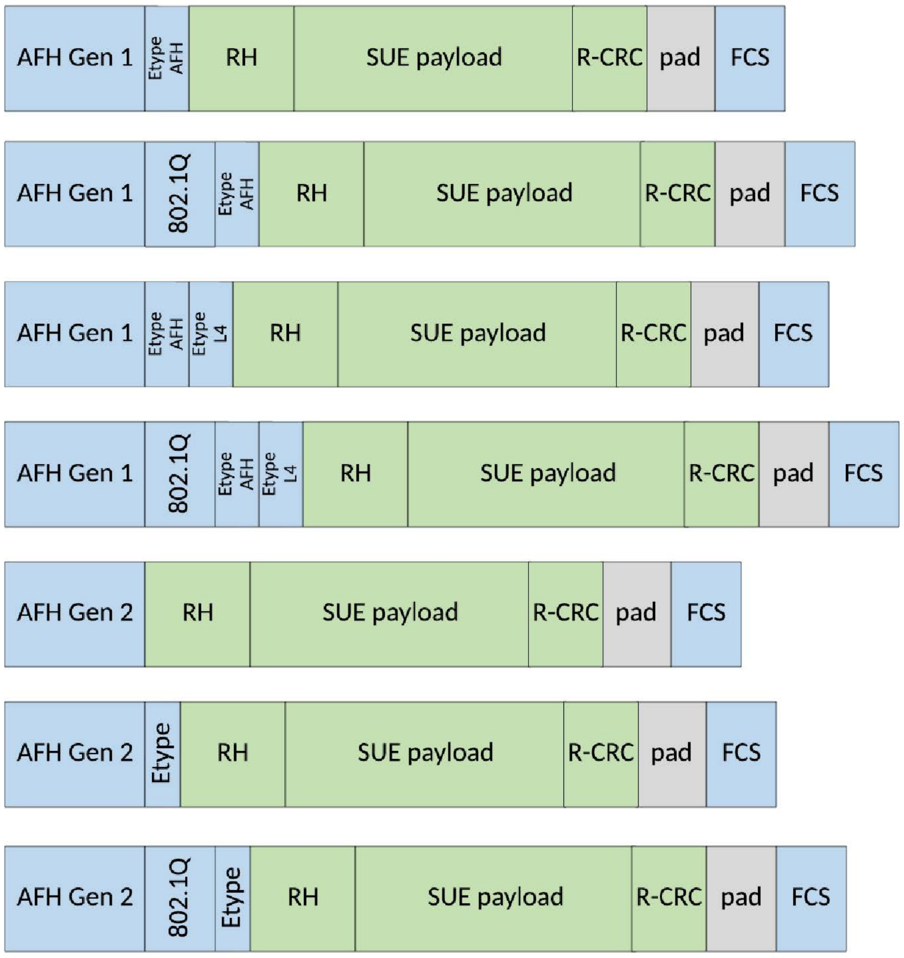


Figure 12: AFH Gen 1 and Gen 2



3.5.2 AI Fabric Header Gen 2

AFH Gen 2 is an optimized header that is aligned with existing Ethernet standards to minimize change while providing a much smaller network header. Based on the Structured Local Address Plan in IEEE 802.c-2017, two options are defined: 12B and 6B headers.

Using Administratively Assigned Identifier with IEEE-compliant encoding, as illustrated below, the XPU identifiers are mapped to 32b or 16b values and are populated in the Destination Address and Source Address areas:

- M = 0/1 (multicast)
- V = 0 (current version)
  - W = 0 (Normal Format with Hop Count and Entropy)
  - W = 1 (Compressed format, no hop count or entropy)
- V = 1 (future)
- X = 1 (locally assigned)

Y = Z = 0 (AAI encoding per SLAP)

Figure 13: AFH Gen 2 Fabric Header Compression

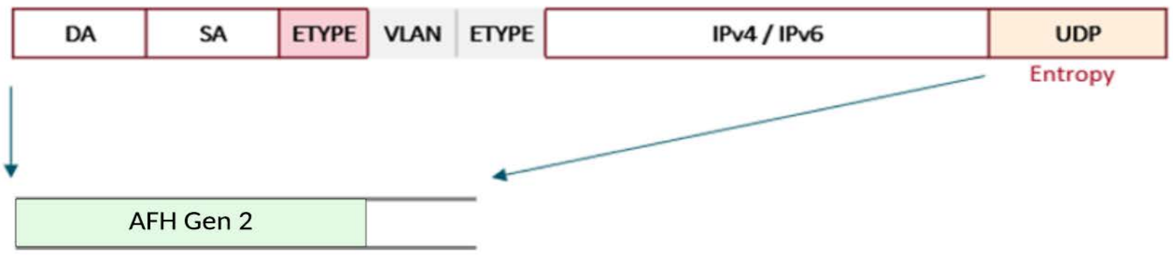
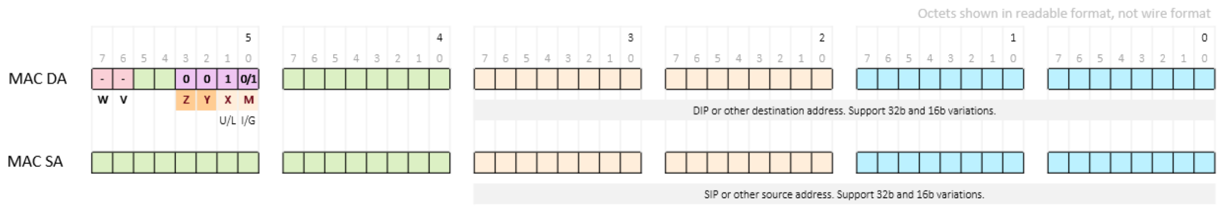
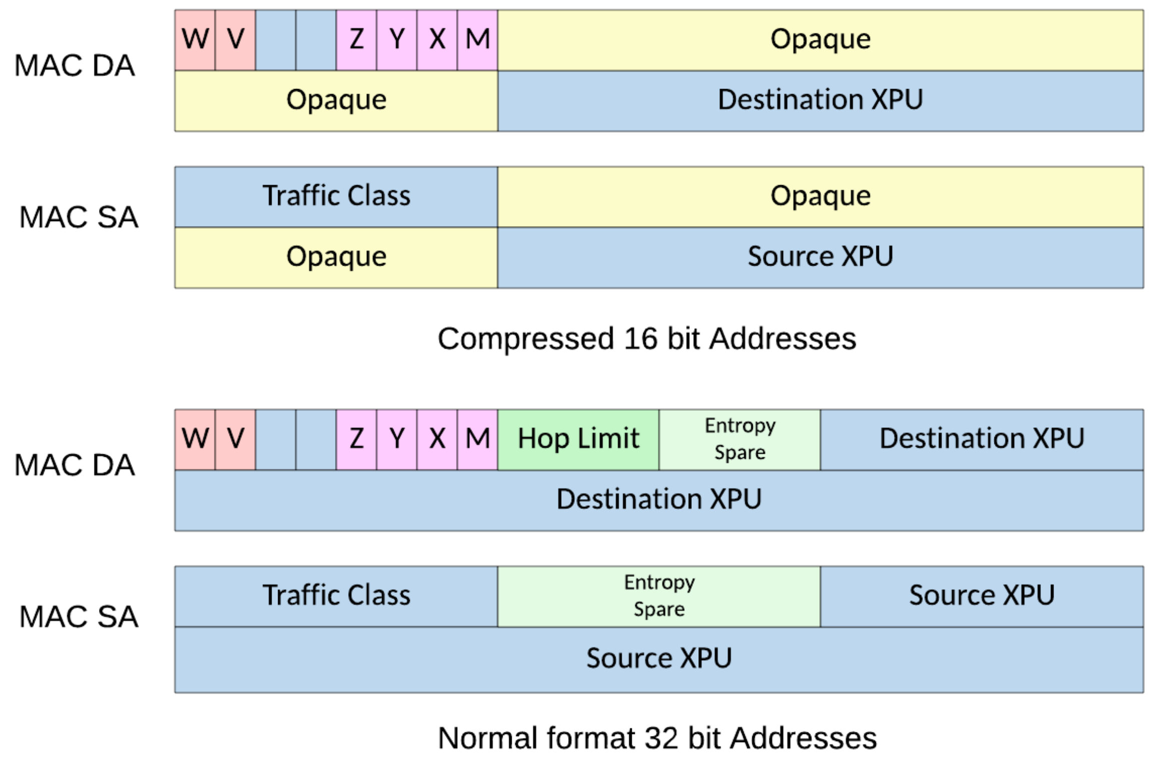


Figure 14: AFH Gen 2 Field Mapping


Octets shown in readable format, not wire format
5
4
3
2
1
0
7 6 5 4 3 2 1 0
MAC DA
- - 0 0 1 0/1
W V
Z Y X M
U/L I/G
DIP or other destination address. Support 32b and 16b variations.
MAC SA
SIP or other source address. Support 32b and 16b variations.

The resulting format for the AI Fabric Header is illustrated in Figure 15.

Figure 15: AFH Gen 2 Normal and Compressed Formats



3.5.3 AFH Gen 1

The AFH Gen 1 uses standard Ethernet MAC Destination and Source addresses, but the switch hardware can just look at 16 bits to 32 bits of the address to perform the forwarding decision. The XPU-id is mapped into a 16-bit destination address for the forwarding lookup. To have priority mapping of traffic with the non-shim format, an IEEE 802.1Q VLAN header needs to be added to the frame. The Gen 1 format without Shim header is illustrated in Figure 16.

The AFH Gen 1 format also supports a Shim header that contains many fields that are similar to an IP header. The Shim header is illustrated in Figure 17.

Figure 16: AFH Gen 1 No Shim Header

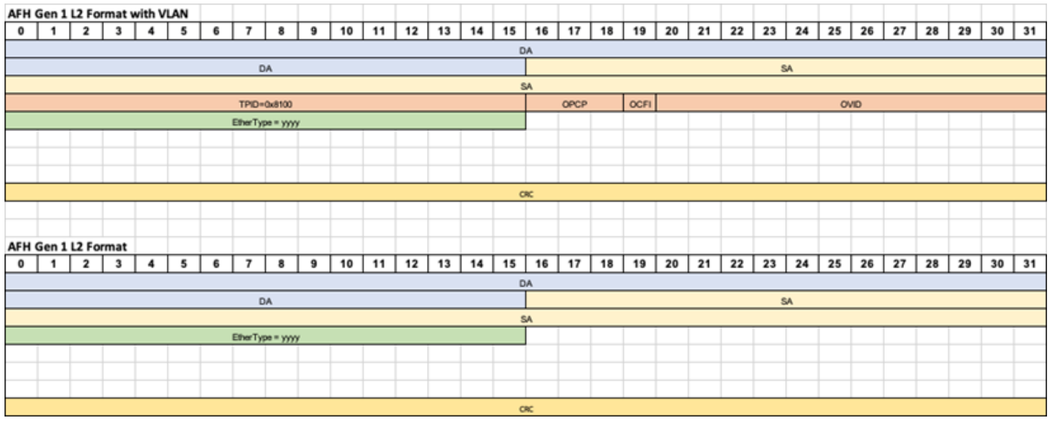
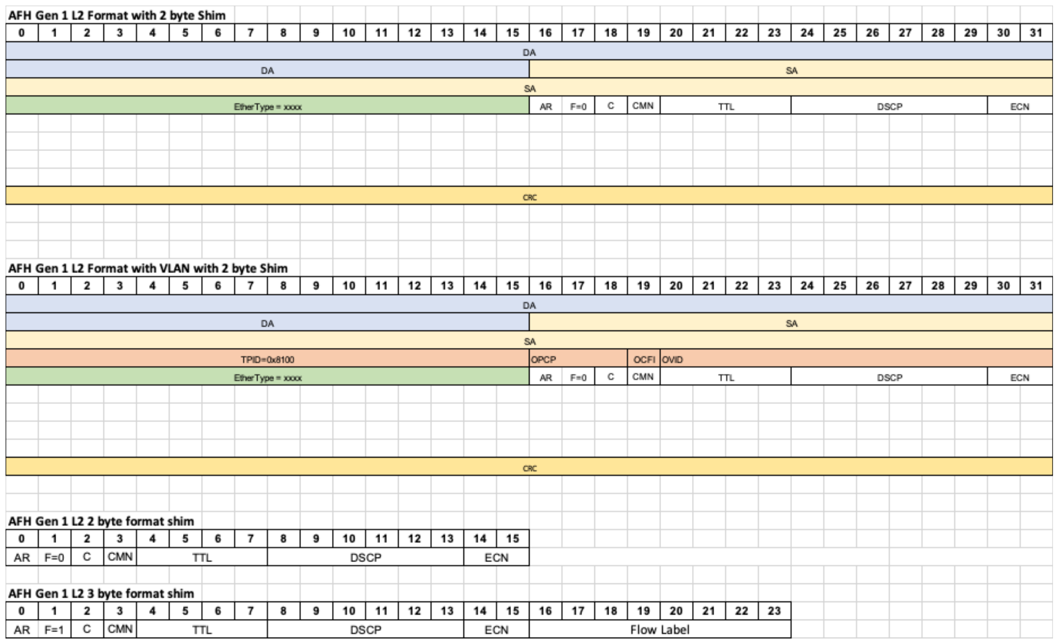


Figure 17: AFH Gen 1 with Shim Header



3.5.4 FEC

FEC options with a smaller FEC block size enable lower latency. Options such as RS-272 can be selected over the standard IEEE RS-544. RS-272 corrects less errors with the tradeoff of improved latency.

At 400G and 800G speeds, the IEEE requires the interleaving of FEC blocks. The latency can be reduced by supporting non-interleaved or less interleaved modes of FEC blocks across lanes.

3.5.5 Link Layer Retransmission (LLR)

LLR is used to improve the reliability beyond FEC, such that if an individual packet is corrupted, it can be retransmitted between peer devices, rather than waiting for the end points of the transaction to determine if the packet has been lost or corrupted. LLR, regardless of FEC, will allow for a packet to be retransmitted if it was dropped because of link layer errors between two switches if FEC could not correct the packet.

3.5.6 Priority Flow Control

PFC is used in Ethernet networks to provide lossless links because of congestion. PFC is defined in IEEE 802.1Qbb. PFC utilizes the IEEE 802.1P bits (Priority) in the 802.1Q header to flow control a specific priority, rather than the entire link as done with 802.3x Pause. The SUE must use at least two lossless PFC classes—one for requests, and one for acknowledgments—so that requests do not block acknowledgments and cause a deadlock situation. Many implementations of PFC can use the DSCP value in the IP header to map to an IEEE 802.1P priority value, if the IEEE 802.1Q header is absent from the packet.

3.5.7 Credit Based Flow Control (CBFC)

CBFC is a feature standardized in the Ultra Ethernet Consortium (UEC) to provide lossless links because of congestion. The following list describes the advantages CBFC has over PFC:

- For the same amount of buffer, CBFC can support more lossless classes than PFC. PFC provides for 8 classes, and CBFC supports 32 classes.
- The sender knows the credit usage of each VC and SUE can use this information for scheduling and load balancing of traffic.
- CBFC allows for a smaller amount of buffering than PFC for the same link length/delay.

Different CBFC classes are used to avoid deadlock between the XPUs and the switch. The VC is used to map to the CBFC class.

3.5.8 Link Failures

The SUE provides link status for all the Ethernet ports attached to it.

When operating in a two or four port configuration with unordered mode, SUE determines when a remote XPU is not reachable over a specific port/plane. When detected, SUE moves the associated work to another port.

Chapter 4: SUE Operation

4.1 Memory Model and Semantics

SUE uses a shared memory model. Address translation, when required, is handled by the XPU outside of the SUE instance.

SUE is defined to support services for load/store (put, get, atomic). The source XPU issues a command and SUE delivers it, along with associated data, to the destination XPU and provides a completion indication to the source XPU.

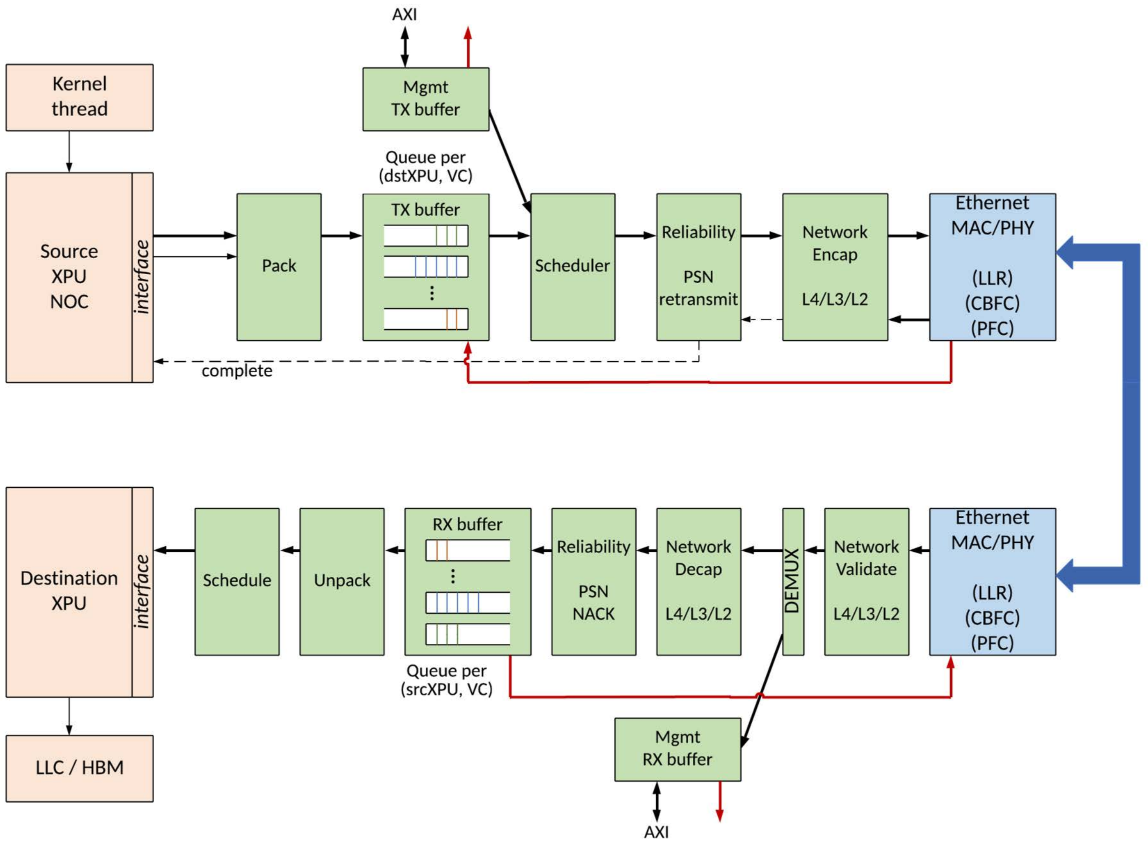
SUE uses one-sided memory semantics. Each command is done unilaterally from the source, with no indication provided to the destination. Memory registration and management, such as protection domains, are handled outside of SUE. SUE provides a partition field that can be used for access control. Examples of similar technologies include PCIe and UALink. A one-sided memory transaction is acknowledged by the receiver as having arrived there—not that the data is placed into memory.

This is in contrast to network semantics used by RDMA over Converged Ethernet (RoCE), InfiniBand, and TCP/IP. These require establishment of a connection to the remote XPU.

4.2 SUE Processing Overview

Figure 18 illustrates the processing flow for SUE.

Figure 18: SUE Processing Flow



The SUE processing flow follows the following steps:

1. The XPU kernel issues a command to SUE over the XPU command interface.
2. SUE accepts the operation and packs the control and data, if present, into the per-destination transmit buffer. The buffer function determines when a queue should be serviced, based on packing and flow control state (for example, CBFC input), and indicates to the scheduler when a queue is eligible.
   a. The first bytes of the control are the command type and length. The command type defines the length of the control information to be carried to the destination (in 2B units) and whether data is present. If data is present, the data length is defined.
   b. Based on these two lengths, the command (control and data) is packed.
3. The scheduler provides weighted round robin across VCs and arrival order-based scheduling within a VC.
   a. When a packet is transmitted with the control interface, it is merged into the data path.
4. Each packed group of commands is encapsulated with a reliability header that assigns a PSN based on {output port, destination XPU}. PSNs are assigned as monotonically increasing numbers which increase by 1 with each packet. If there is an ACK or NACK to be sent, this is added to the RH. The R-CRC is added over the RH and packed operations to create the SUE PDU.
   a. There is a connection between XPUs on a per physical port basis; the connection state is compact, consisting of the next expected PSN and a few control fields to manage ACKs and NACKs.

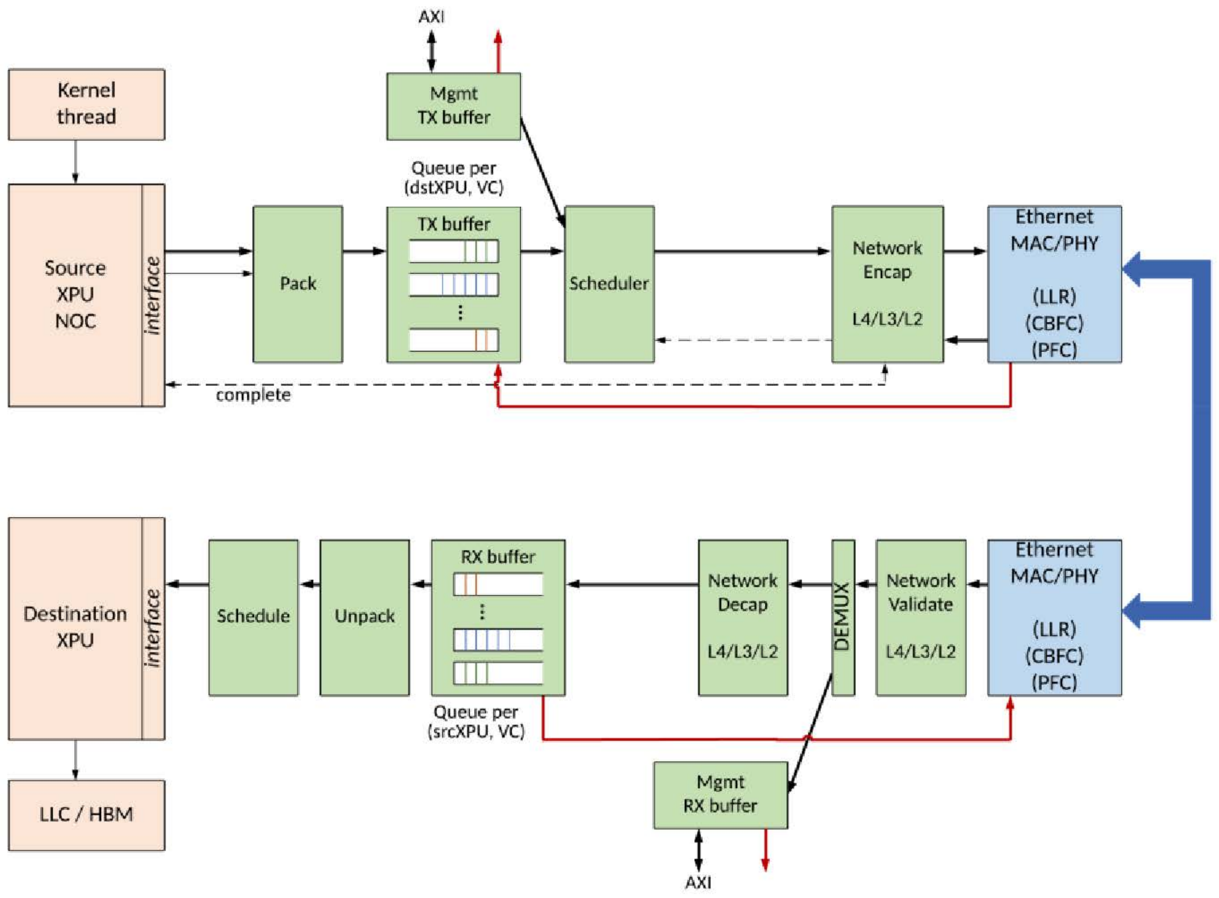
   b. SUE provides in-order delivery on each plane. Use of lossless traffic classes and LLR reduce the likelihood of packet loss dramatically, however it is still possible. If SUE determines a packet loss event occurred, it uses GoBackN to recover.

5. A network header is added to the SUE PDU. The destination XPU and VC are used to look up the required address fields.
6. The Ethernet packet is transmitted and arrives at the destination.
7. Ethernet validity checks are performed and any packets destined to the control interface are demultiplexed.
8. The Ethernet header is verified and then the RH header checked. If the expected PSN for that {Source XPU, VC} is received, the PDU is put in the RX buffer and scheduled to be sent to the XPU NOC.
   a. If an unexpected PSN is received, a NACK is sent to the source and any arriving packets from the source are dropped until the expected PSN arrives.
   b. When the expected PSN is received, the packet is acknowledged.
9. The RX buffer receives commands, which are then scheduled, unpacked and sent to the XPU NOC.

4.3 SUE Lite Processing Overview

Figure 19 illustrates the processing flow for SUE Lite.

Figure 19: SUE Lite Processing Flow



The SUE Lite processing flow follows the following steps.

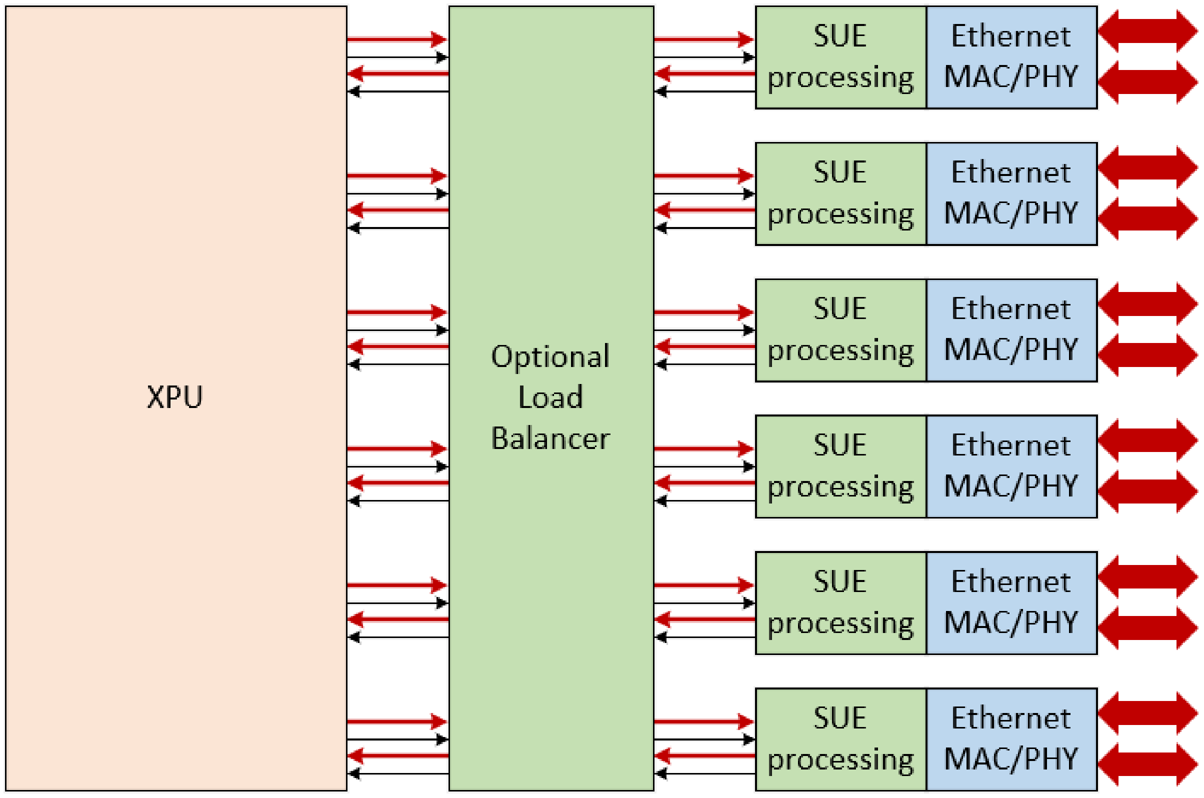
1. The XPU kernel issues a command to SUE over the XPU command interface.
2. SUE Lite accepts the operation and packs the control and data, if present, into the per-destination transmit buffer. The XPU can indicate that the operation should be transmitted without waiting for additional operations. The buffer function determines when a queue should be serviced, based on the packing and flow control state (for example, CBFC input), and informs the scheduler when a queue is eligible.
   a. The first bytes of the control are the command type and length. The command type defines the length of the control information to be carried to the destination (in 2B units) and whether data is present. If data is present, the data length is defined.
   b. Based on these two lengths, the command (control and data) is packed.
3. The Scheduler provides weighted round robin across VCs and arrival order-based scheduling within a VC.
   a. When a packet is transmitted with the control interface, it is merged into the data path.
4. Each packed group of commands is encapsulated.
   a. SUE Lite expects the switch element (if present) to provide in-order delivery on each plane. Use of lossless traffic classes and LLR dramatically reduce the likelihood of packet loss.
5. A network header is added to the SUE Lite PDU. The destination XPU and VC are used to look up the required address fields.

6. The Ethernet packet is transmitted and arrives at the destination.
7. Ethernet validity checks are performed, and any packets destined to the control interface are demultiplexed.
8. The Ethernet header is verified.
9. The RX buffer receives commands, which are then scheduled, unpacked, and sent to the XPU NOC.

4.4 Load Balancing

It is presumed that there will be multiple instances of SUE connected to an XPU. Load balancing across the SUE instances can be handled in software or by including a hardware block in the XPU that distributes the operations across the SUE modules. Each SUE instance provides congestion state to enable this option. Refer to Figure 20.
Within the SUE, load balancing can be enabled when using two or more ports. In this case, SUE will dynamically assign each packed group of operations to a port based on available bandwidth.

Figure 20: Example XPU Load Balancer

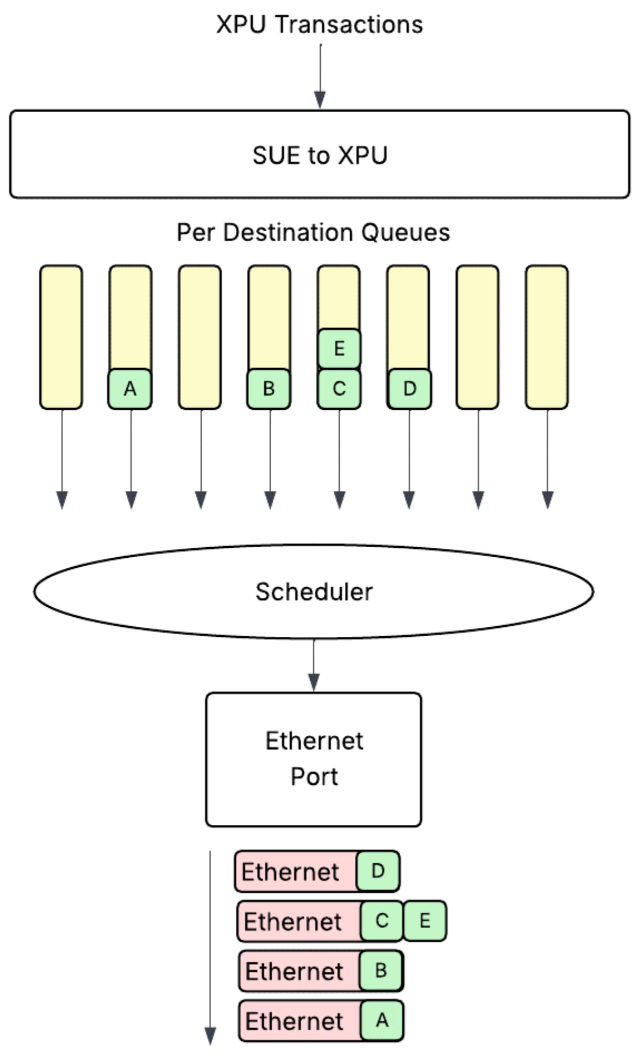


4.5 Packing

The XPU to SUE interface is at higher speed than the SUE to Ethernet interface. There is flow control implemented across multiple VCs to prevent over running the buffers in SUE. This speed mismatch will cause transactions to accumulate in the per destination queues in SUE. The scheduler in SUE outbound to the Ethernet interface will go through the queues looking for the next transactions to send. If there are multiple transactions in a queue, SUE will pack them together into outbound Ethernet packets. If a destination queue has a single transaction, the transaction is sent. SUE does not delay transactions for reasons of packing. SUE will pack up to a preconfigured limit, for example 2K, and send the transactions before moving onto the next eligible queue.

In Figure 21, the packing process is illustrated. The XPU sends SUE transactions A to E. The transactions are queued within SUE on a per-destination basis. The scheduler is work conserving; the scheduler will transmit the transactions in the next available queue. In this case, the next available queue is the one with A in it. After transmitting A, the scheduler selects the queue with B in it and transmits it. The scheduler then selects the queue with C in it. Transaction E is queued up to the same destination so they are packed together into the same frame and sent. The scheduler then selects the queue with D and transmits it. SUE does not wait for a queue to accumulate transactions to a particular size, but opportunistically packs transactions together to optimize packet-on-wire efficiency and minimize latency of transactions.

Figure 21: SUE Packing Example

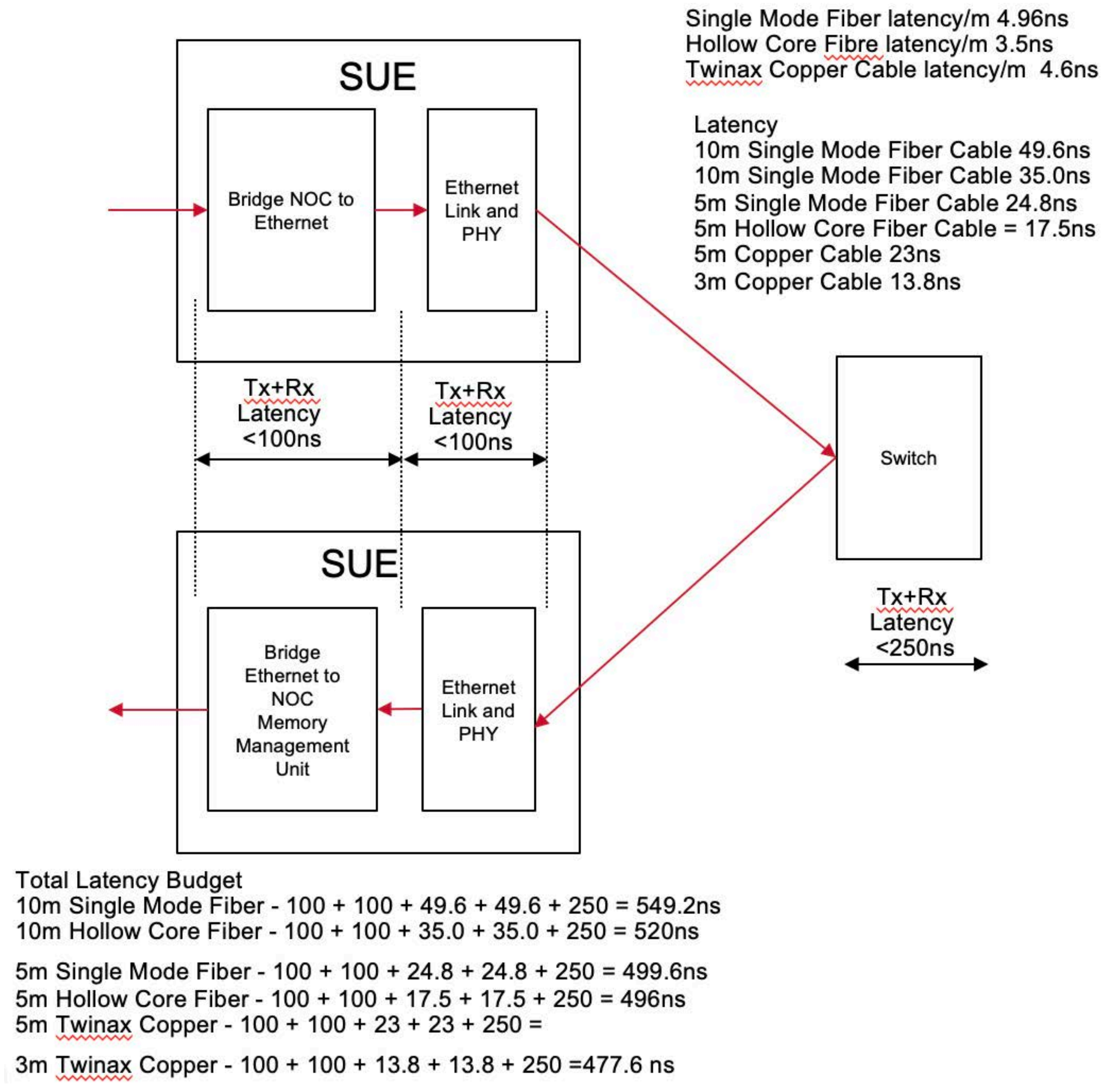


Appendix A: Latency Budget

A.1 SUE E2E One Way Latency Budget

The throughput of a multi-XPU shared memory system is determined by the latency of the XPU communication. Therefore, minimizing the latency of the interconnect is of great importance. The length and type of cabling, switch latency are all critical components. Figure 22 illustrates the latency of the various components in the SUE system.

Figure 22: SUE End-to-End One-Way Latency Budget



Single Mode Fiber latency/m 4.96ns
Hollow Core Fibre latency/m 3.5ns
Twinax Copper Cable latency/m 4.6ns

Latency
10m Single Mode Fiber Cable 49.6ns
10m Single Mode Fiber Cable 35.0ns
5m Single Mode Fiber Cable 24.8ns
5m Hollow Core Fiber Cable = 17.5ns
5m Copper Cable 23ns
3m Copper Cable 13.8ns

Total Latency Budget
10m Single Mode Fiber - 100 + 100 + 49.6 + 49.6 + 250 = 549.2ns
10m Hollow Core Fiber - 100 + 100 + 35.0 + 35.0 + 250 = 520ns

5m Single Mode Fiber - 100 + 100 + 24.8 + 24.8 + 250 = 499.6ns
5m Hollow Core Fiber - 100 + 100 + 17.5 + 17.5 + 250 = 496ns
5m Twinax Copper - 100 + 100 + 23 + 23 + 250 =

3m Twinax Copper - 100 + 100 + 13.8 + 13.8 + 250 =477.6 ns

Appendix B: Glossary

Acronym/Term	Definition
SUE	Scale Up Ethernet
CBFC	Credit based flow control
LLR	Link Layer Retransmission
NIC	Network Interface Card
NOC	Network on chip, referring to interconnect within XPU
QoS	Quality of service
XPU	Generic term for ML/AI/HPC accelerator including GPUs, CPUs, etc.

Revision History

Scale-Ethernet-RM103; July 10, 2025

- Added SUE Lite information.
- Updated the SUE Stack, SUE interfaces, SUE Reliability, Standard Ethernet Format, AFH Gen 1 and Gen 2, and AFH Gen 2 Normal and Compressed Formats images.
- Updated the XPU Command Interface section.
- Updated the XPU to SUE AXI Interface section.

Scale-Ethernet-RM102; June 13, 2025

- Added AXI interface from XPU to SUE.
- Updated AFH Packet Headers.
- Added details on packing.

Scale-Ethernet-RM101; May 1, 2025

Deleted duplicate image on page 20.

Scale-Ethernet-RM100; April 28, 2025

Initial document version.

BROADCOM®