

Instructor - Emmanuel

1. Experience: Over 11 years in VLSI circuit design and verification, lead projects for Intel, Rambus, and Qualcomm.
2. Skills: Proficient in Verilog, SystemVerilog, VHDL, UVM, OVM/SAOLA, SVA, and EDA tools like Synopsys VCS, DC and Cadence Incisive and Mentor Questasim.
3. Leadership: Managed teams and projects from specification to sign-off, ensuring timely and efficient delivery.
4. Recognition: Received multiple awards for performance, delivery, quality, including Key Achiever and Customer Centric awards.
5. Project Contributions: Significant contributions to protocols such as USB, Ethernet, PCIE and much more.

Disclaimer

The information, content, and resources provided in this document are for informational purposes only and are subject to change without prior notice. ASICLab makes no guarantees, warranties, or representations, either express or implied, regarding the accuracy, completeness, or reliability of the information contained herein.

While we strive to ensure the content is free from errors, we cannot guarantee its absolute accuracy or suitability for any particular purpose. Users are advised to use the information at their own discretion and risk.

ASICLab reserves the right to make modifications, updates, or corrections to this document at any time without prior notification.

By accessing or using this document, you acknowledge and agree to this disclaimer in its entirety.

asiclab

PCI Express Protocol Introduction

asiclab

What is PCI?

- **Peripheral Component Interconnect**

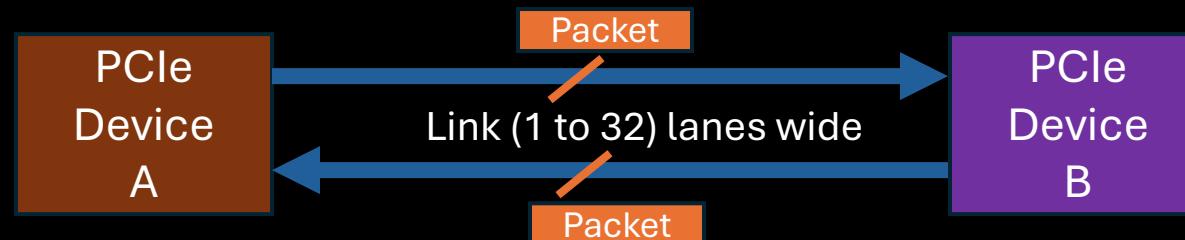
asiclab

- **Computer bus for attaching peripheral devices to a computer motherboard**

asiclab

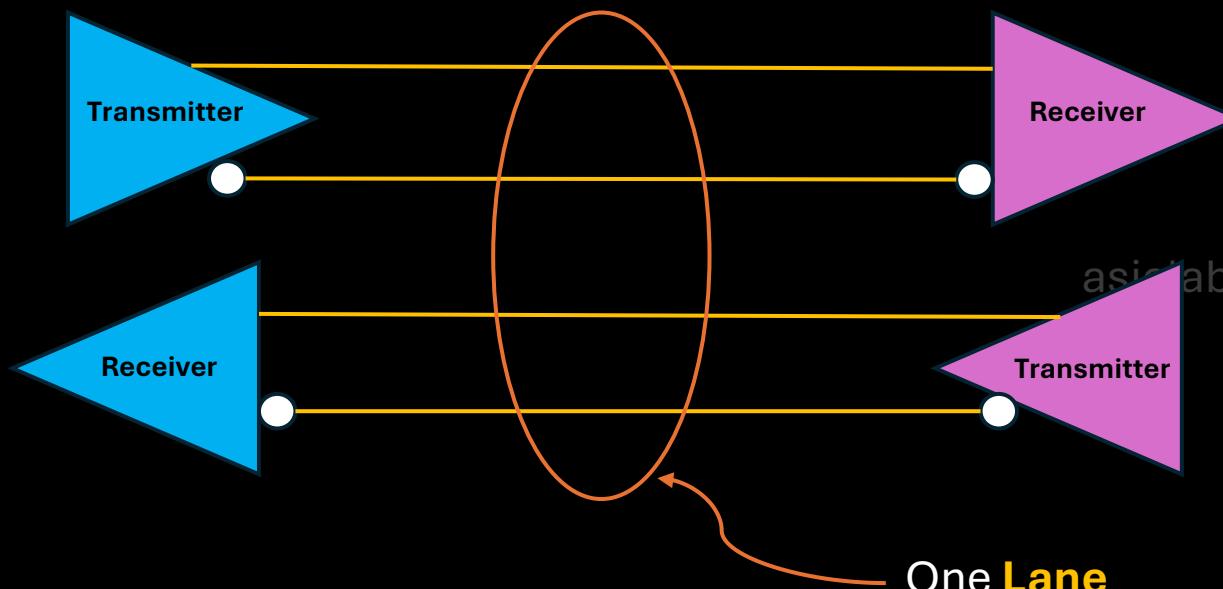
Introduction to PCI Express

- PCI Express (PCIe) is the new name for the technology formerly known as 3GIO
- PCIe's most drastic and obvious improvement over PCI is its point-point bus topology.
- Scalable: x1, x2, x4, x8, 12, x16, x32 links
- Symmetric: Same number of lanes in each direction
- Dual-Simplex connection
- 2.5, 5.0, 8.0, 16.0, 32.0 GT/s transfer rate in each direction
- Packet-based transaction protocol
- Software backward compatible with PCI & PCI-X



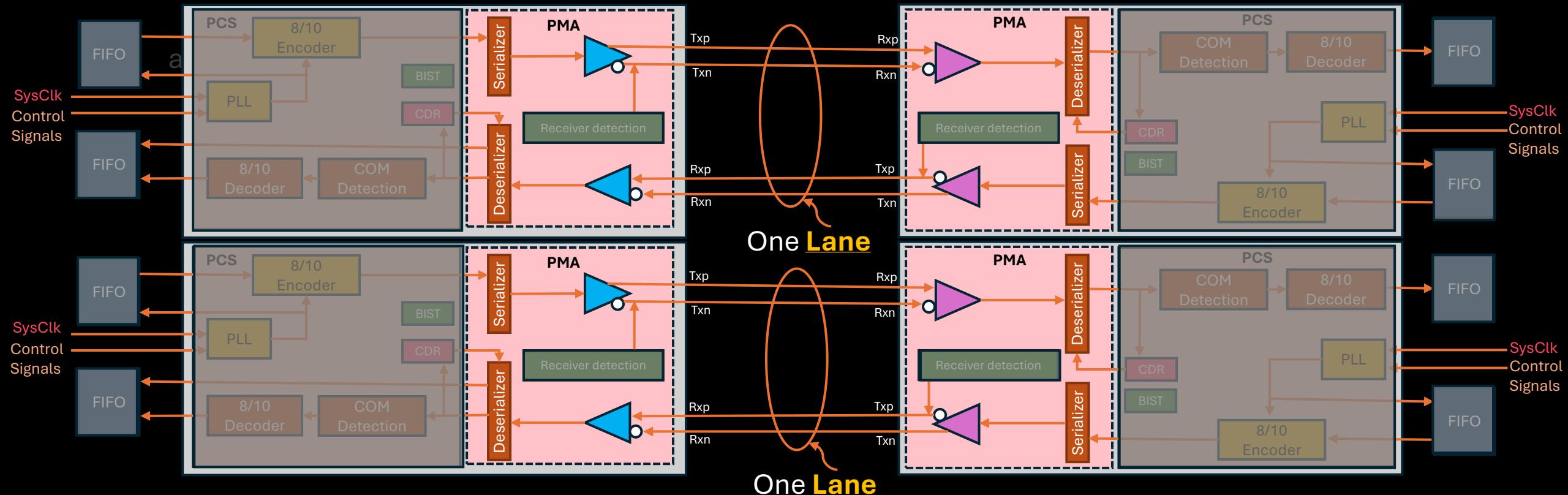
Link Width and Lanes

- Performance is scalable, based on the number of signal Lanes implemented
- Lane consists of a serial send/receive path made up of four wires (2 each for differential Tx and Rx)
- Link can have a minimum of 1 Lane or as many as 32 Lanes, and number of Lanes in a Link is called Link width. A Link connects two devices.



Lanes

▪ 2 Lane Example



PCIe Throughput

Version	Introduced	Line code	Transfer rate	Throughput						
			per lane	x1	x2	x4	x8	x16		
1	2003	NRZ	8b/10b	2.5 GT/s	0.250 GB/s	0.500 GB/s	1.000 GB/s	2.000 GB/s	4.000 GB/s	
2	2007			5.0 GT/s	0.500 GB/s	1.000 GB/s	2.000 GB/s	4.000 GB/s	8.000 GB/s	
3	2010			8.0 GT/s	0.985 GB/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	
4	2017		128b/130b	16.0 GT/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	
5	2019			32.0 GT/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s	
6	2022		PAM-4	1b/1b	64.0 GT/s	7.563 GB/s	15.125 GB/s	30.250 GB/s	60.500 GB/s	121.000 GB/s
			FEC	242B/256B FLIT	32.0 GBd					
7	2025 (planned)	PAM-4	1b/1b	128.0 GT/s	15.125 GB/s	30.250 GB/s	60.500 GB/s	121.000 GB/s	242.000 GB/s	
		FEC	242B/256B FLIT	64.0 GBd						

$$\text{Total Aggregate Bandwidth} = \frac{\text{Bandwidth} \times \text{Number of Lanes}}{2 * \text{Bits Per Byte}} * \frac{8}{10} * \frac{128}{130} * 2$$

Bidirectional

Example:

For Gen 5 with x8 lanes the transfer rate or Bandwidth is 32GT/s so,

$$\text{Total Aggregate Bandwidth} = \frac{32\text{GT/s} \times 8}{2 * 8} * \frac{128}{130} * 2 = 31.5076 \text{GB/s}$$

asiclab

GT = Giga-Transfers

Gb = Gigabits

GB = Gigabytes

PCIe Throughput

Version	Introduced	Line code	Transfer rate	Throughput						
			per lane	x1	x2	x4	x8	x16		
1	2003	NRZ	8b/10b	2.5 GT/s	0.250 GB/s	0.500 GB/s	1.000 GB/s	2.000 GB/s	4.000 GB/s	
2	2007			5.0 GT/s	0.500 GB/s	1.000 GB/s	2.000 GB/s	4.000 GB/s	8.000 GB/s	
3	2010			8.0 GT/s	0.985 GB/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	
4	2017		128b/130b	16.0 GT/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	
5	2019			32.0 GT/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s	
6	2022		PAM-4	1b/1b	64.0 GT/s	7.563 GB/s	15.125 GB/s	30.250 GB/s	60.500 GB/s	121.000 GB/s
			FEC	242B/256B FLIT	32.0 GBd					
7	2025 (planned)	PAM-4	1b/1b	128.0 GT/s	15.125 GB/s	30.250 GB/s	60.500 GB/s	121.000 GB/s	242.000 GB/s	
		FEC	242B/256B FLIT	64.0 GBd						

$$\text{Total Aggregate Bandwidth} = \frac{\text{Bandwidth} \times \text{Number of Lanes}}{2 * \text{Bits Per Byte}} * \frac{8}{10} * \frac{128}{130} * 2$$

Bidirectional

Example:

For Gen 6 with x1 lanes the transfer rate or Bandwidth is 64GT/s so,

$$\text{Total Aggregate Bandwidth} = \frac{64\text{GT/s} \times (256/257)}{2 * 8} * \frac{1}{1} * 2 \approx 7.968 \text{ GB/s}$$

asiclab

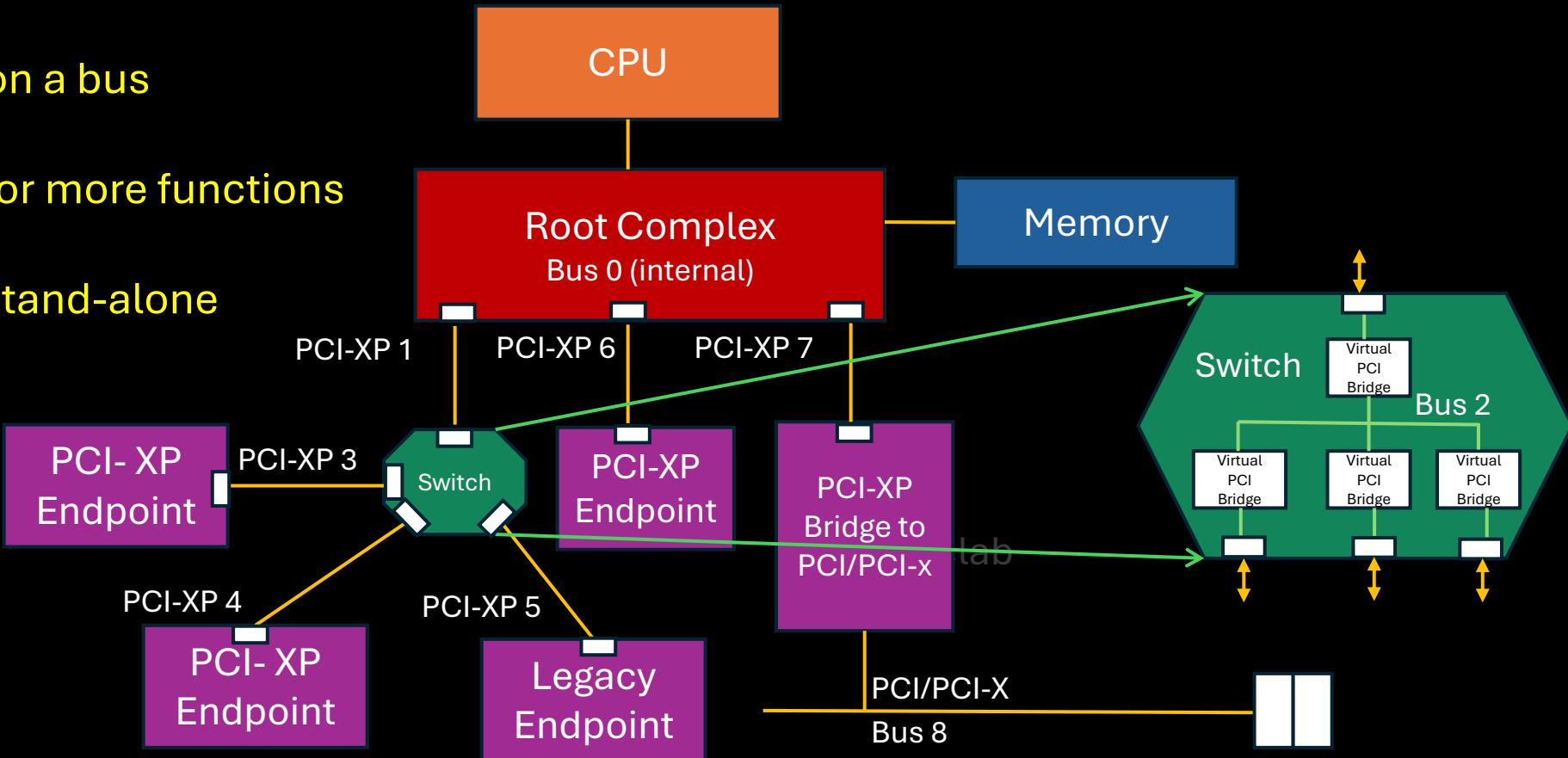
GT = Giga-Transfers

Gb = Gigabits

GB = Gigabytes

Topology

- Root complex
- A device resides on a bus
- A device has one or more functions
- Each function is stand-alone
- Switches



Root Complex

- **Root Complex** connects the CPU to the PCIe topology (e.g.: Chipset)
- Root Complex generate PCIe transaction requests on behalf of CPU, creating 4 different types of requests:
 - Configuration
 - Memory
 - I/O
 - Message

asiclab

Switches and Bridges

- **Switches** provide fan-out and aggregation
 - Connecting more than two ports requires a Switch
 - Switches act as packet routers
- asiclab Peer-to-peer support is mandatory
- **Bridge** connect different busses
 - Forward bridge example: PCIe to PCI, etc.
 - Reverse bridge example: PCI to PCIe

asiclab

Endpoints

- **Endpoints are Functions in PCIe topology that are not Switches or the Root Complex**
 - They only have an upstream port and always reside at the bottom of a PCIe topology “tree structure”
 - They can act as requester or completer for transactions

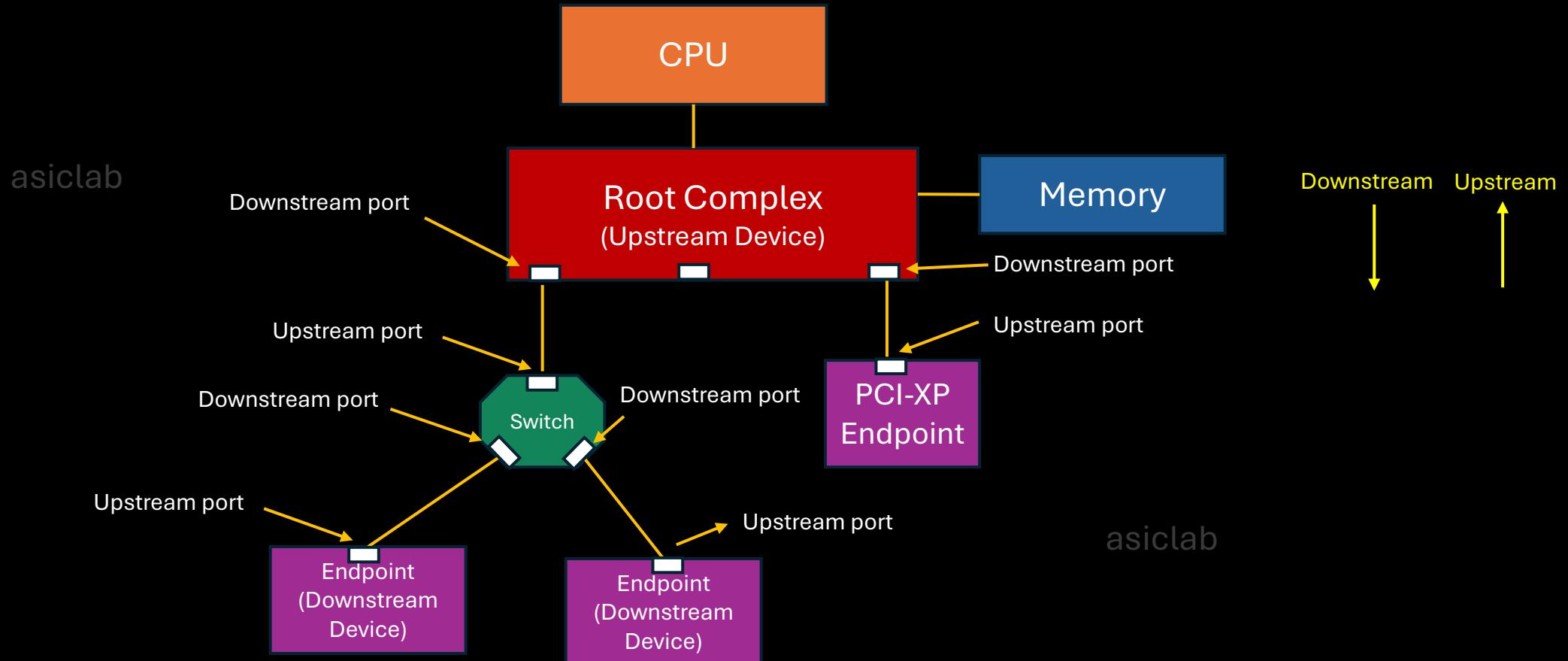
asiclab

Legacy / Native Endpoints

- **Legacy Endpoints** use older PCI bus operations to support backward compatibility
- **Legacy Endpoints are allowed to support things that Native PCIe Endpoints are not, such as:**
 - I/O transactions
 - Locked transactions
 - 32-bit-only memory addressing
- **Native PCIe Endpoints must support 64-bit addressing for prefetchable address ranges**

asiclab

Upstream Vs Downstream



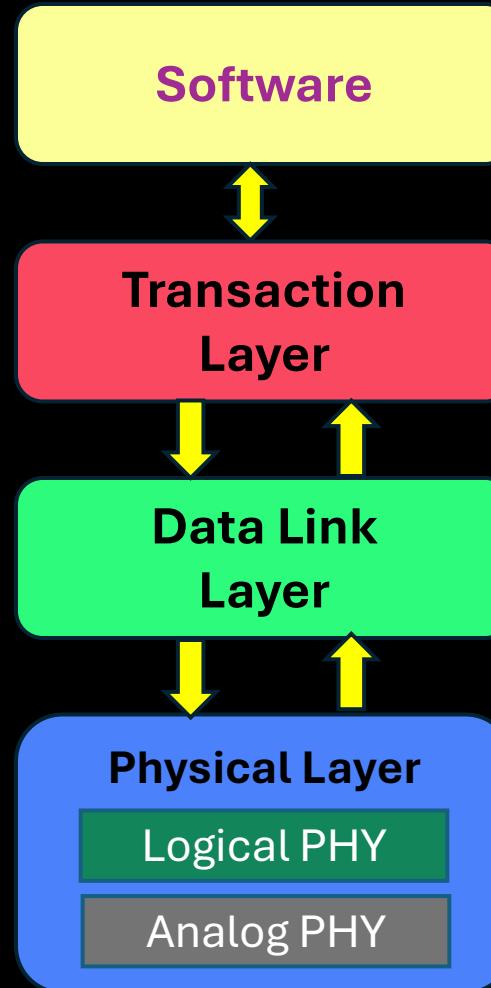
asiclab

PCIe Architecture Overview

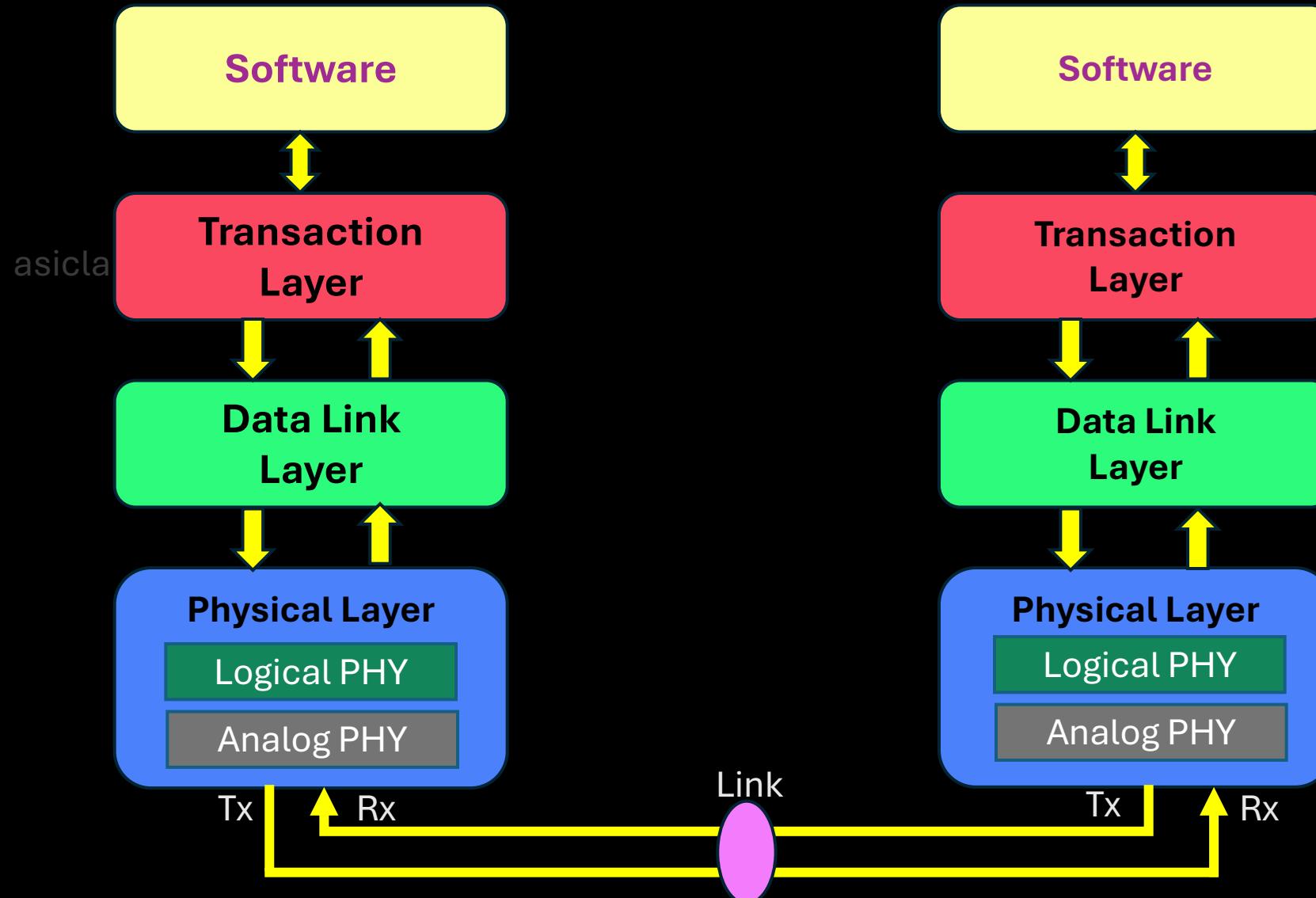
asiclab

Layered Architecture

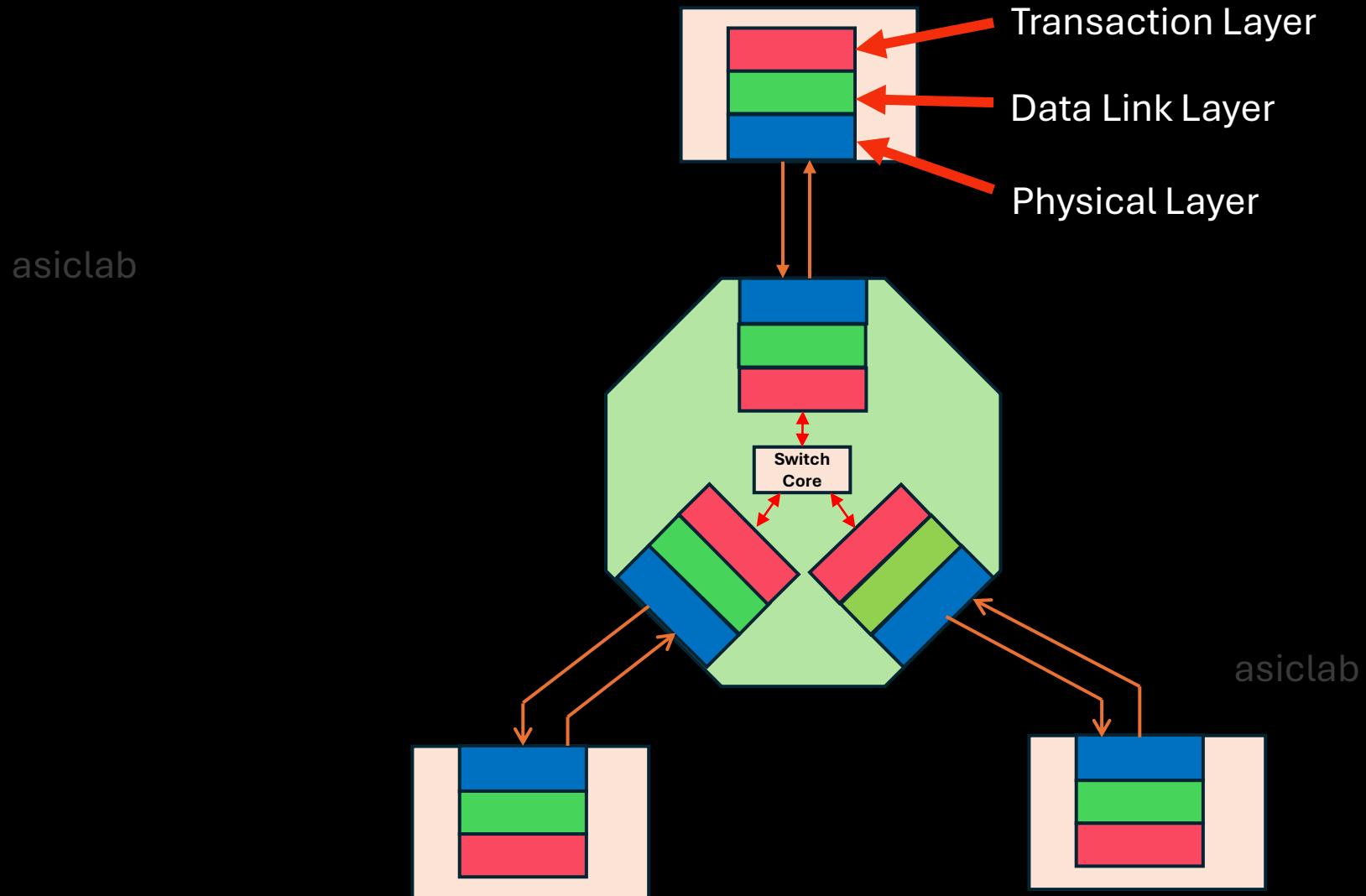
- Scalability
- Modularity
- Reuse
- Understandability



PCI Express



Layers in PCIe Devices



asiclab

Software view

asiclab

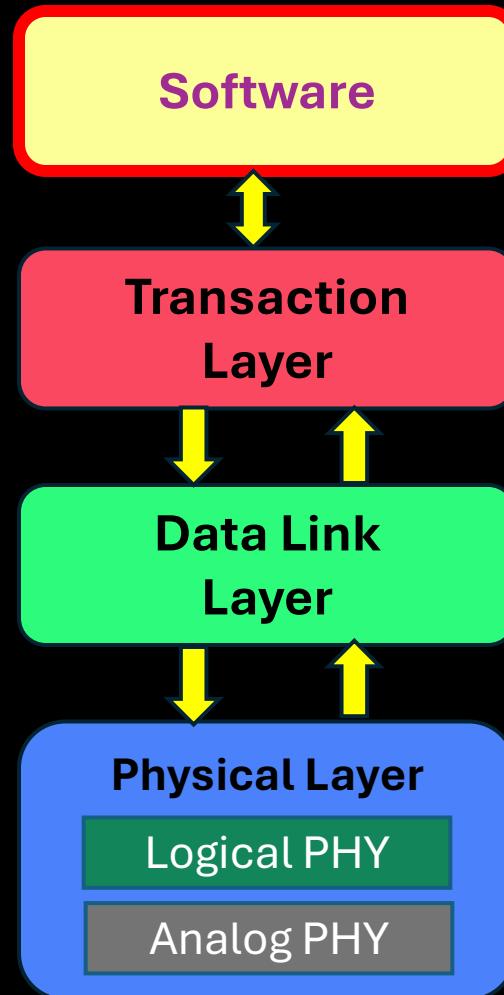
Do Not Distribute

© Copyright protected 2024



Software view

- PCIe tree topology discovery
- Device enumeration
asiclab
- Device programming model
- Device Capabilities discovery



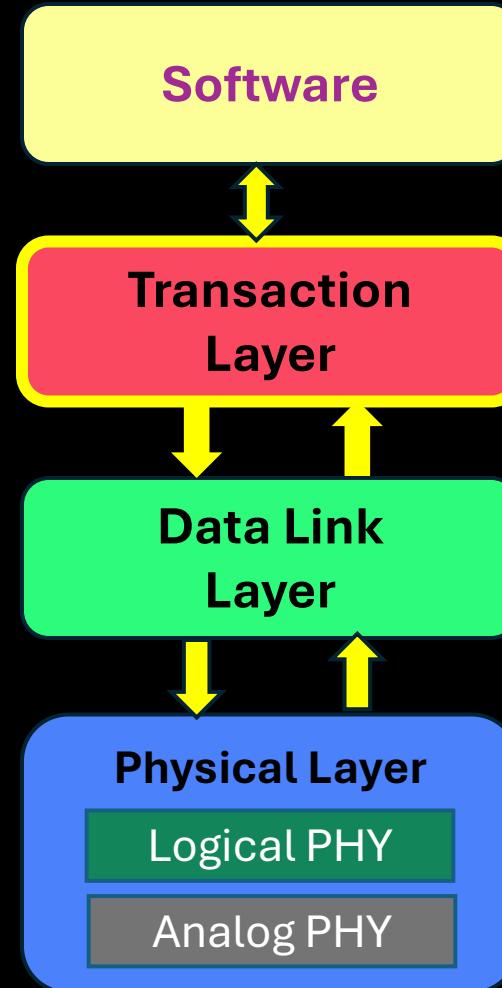
asiclab

Transaction Layer Overview

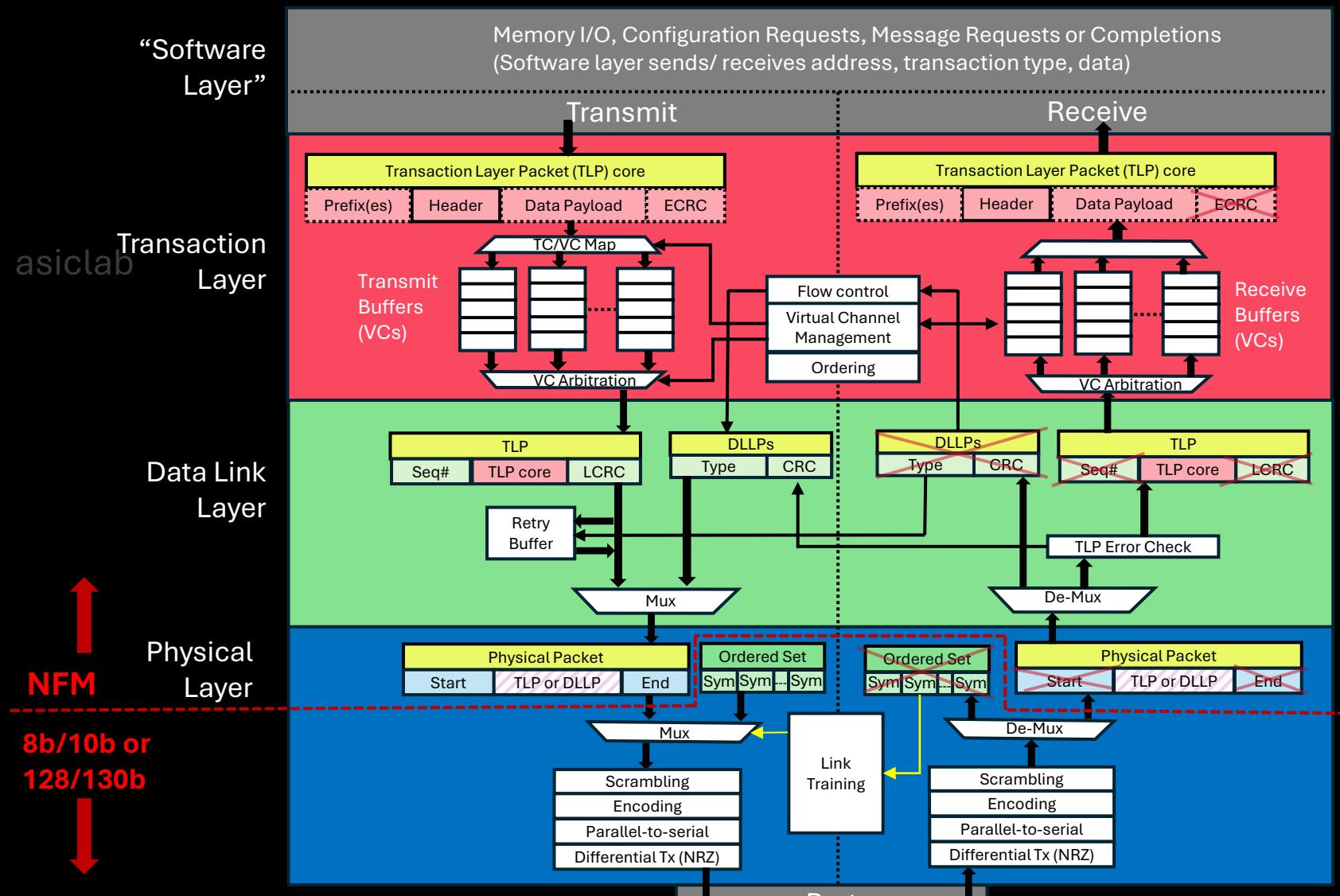
asiclab

Transaction Layer

- Basic Protocol
- Transaction types and services
 - asiclab
- Produce consumer model
- Ordering rules
- Quality of Service



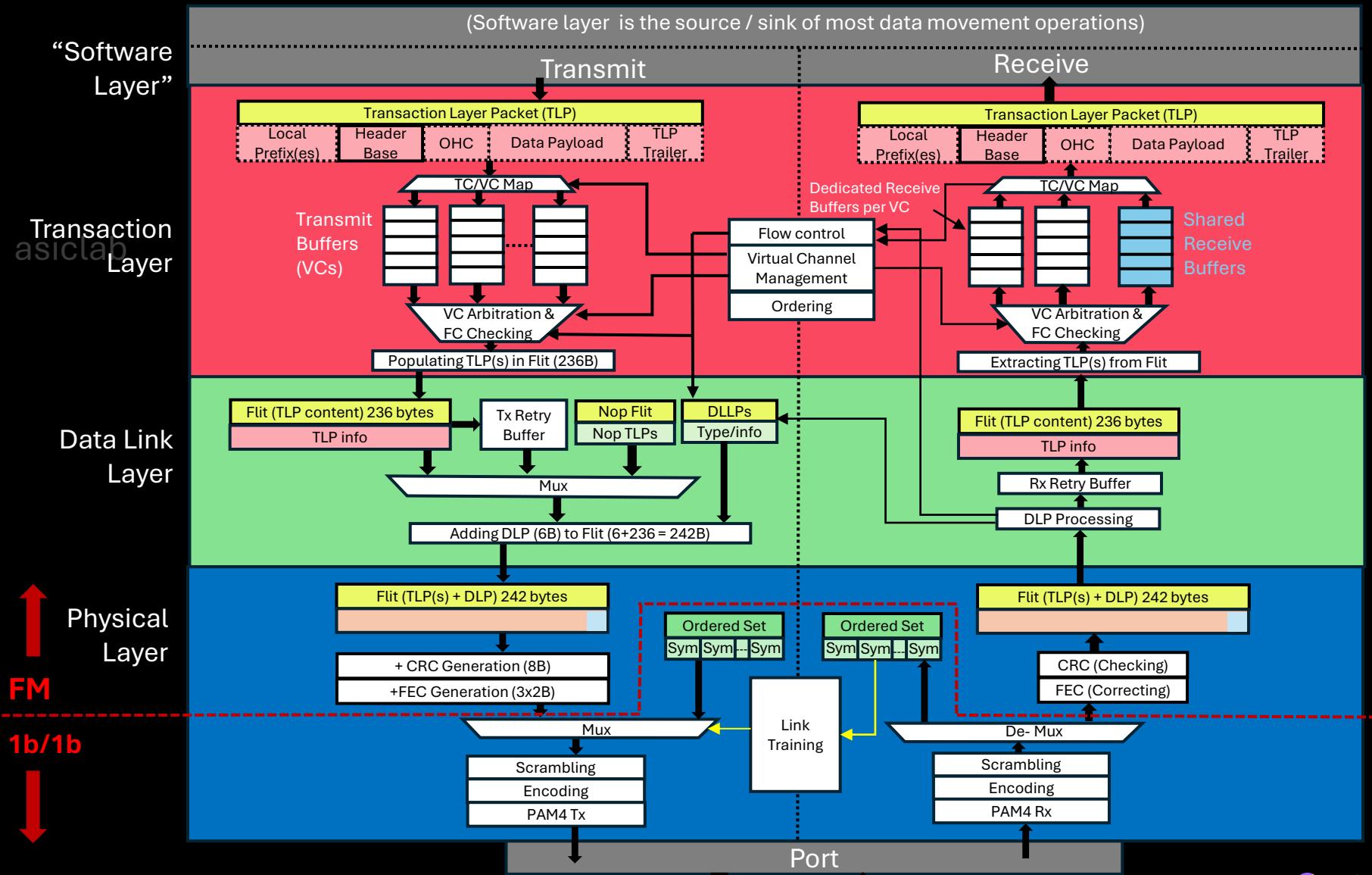
PCIe Device Layer Details 5.0 Non Flit Mode (NFM)



Do Not Distribute

© Copyright protected 2024
Link

PCIe Device Layer Details 6.0 (Flit Mode – FM) (1b/1b)

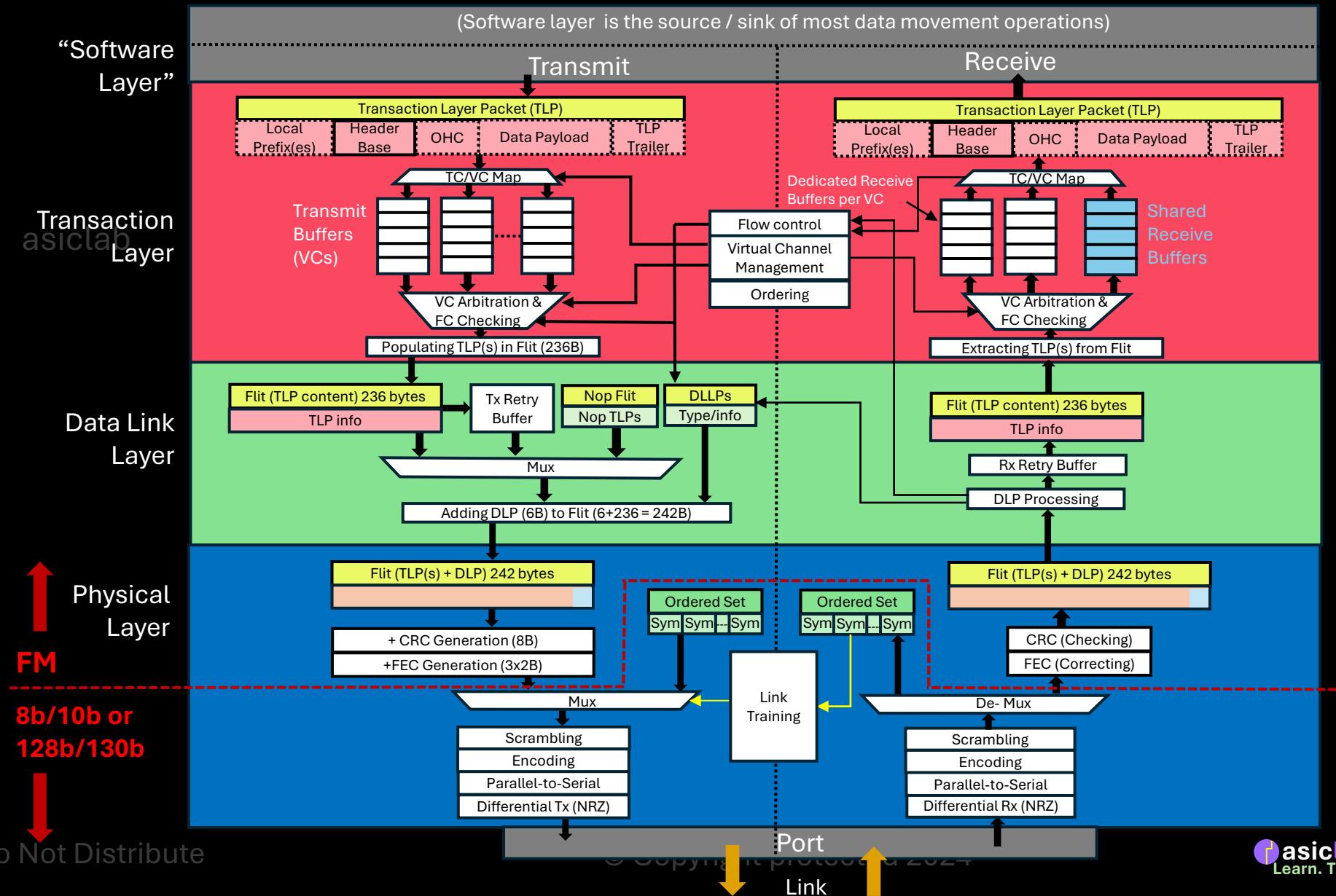


Do Not Distribute

© Copyright protected 2024

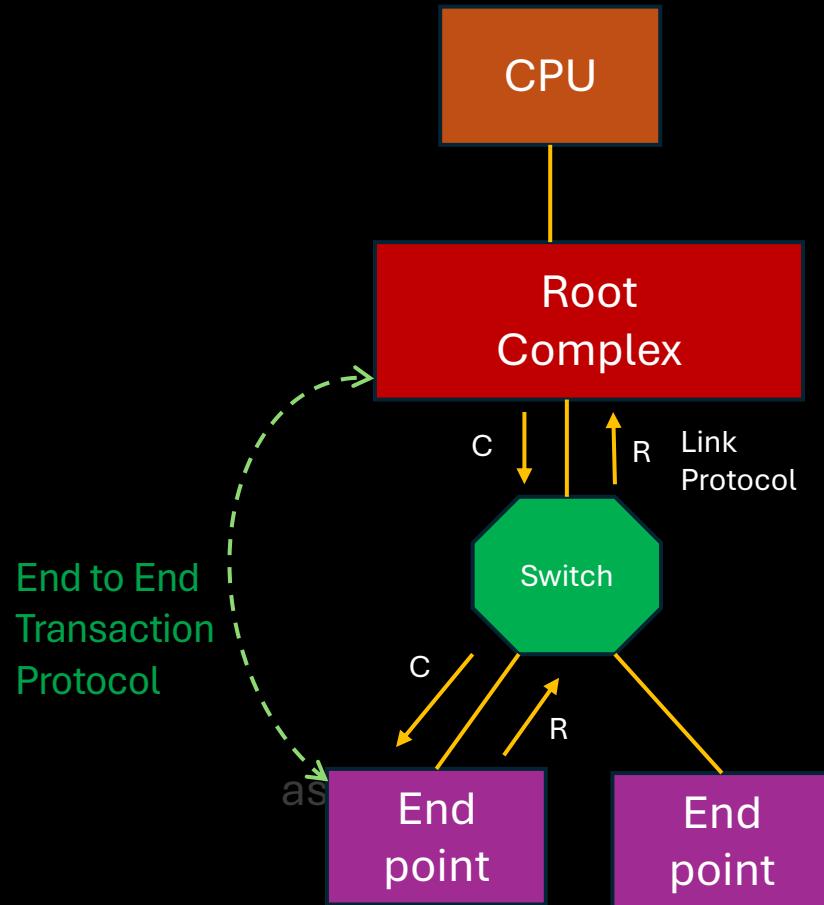
PCIe Device Layer Details 6.0

(8b/10b or 128b/130b)



Transaction Layer

- Full Split-transaction Packet Protocols
 - R – Request Packet
 - C – Completion Packet
 - These packets are referred to as “TLPs” – Transaction Layer Packets
- Transactions flow between two ends named Requester – completer
 - Switches are transient elements
 - Subject to ordering, flow control and Data Integrity Mechanisms



Transaction Layer – Transaction Types

- Backward compatibility with PCI is maintained by using the same memory, I/O and configuration address space
- A new transaction type is added for PCIe: Messages
- PCIe transaction types are shown in the following table:

asiclab

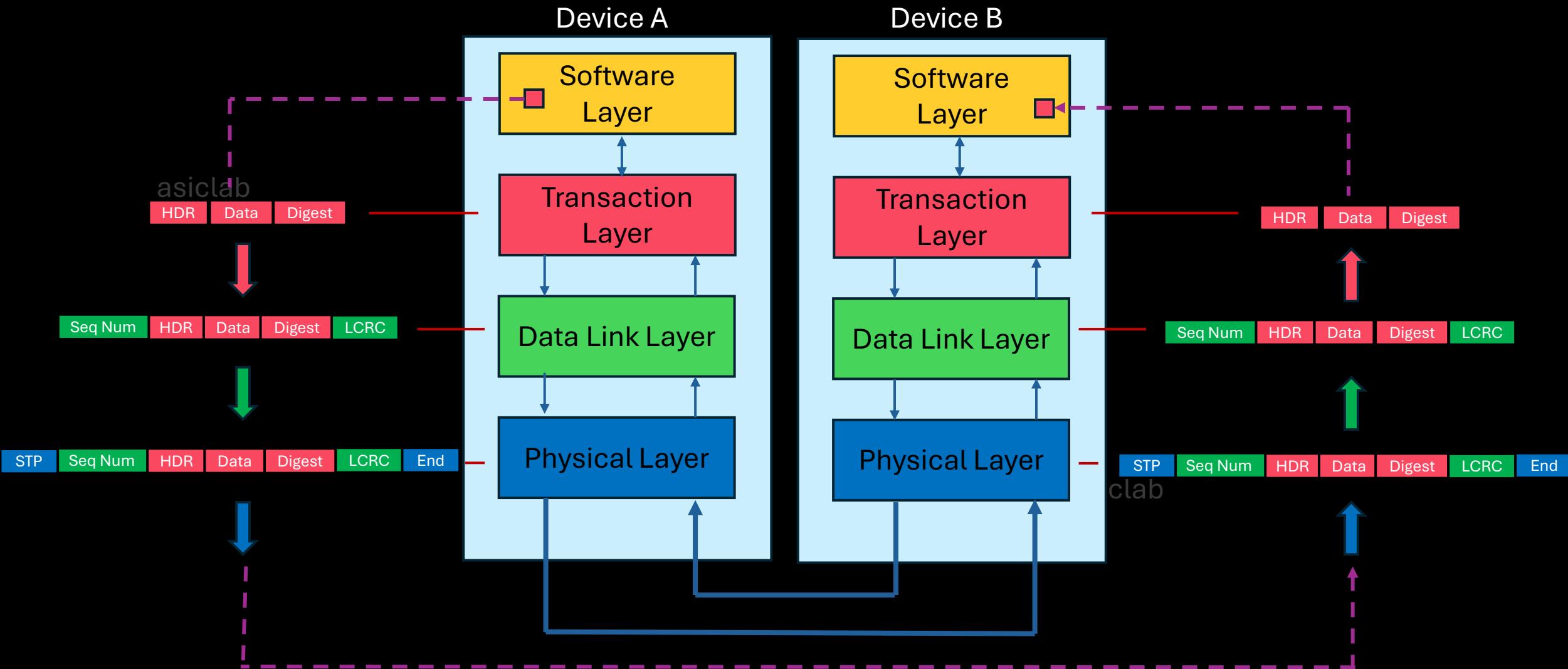
Transaction Type	Non-Posted or Posted
Memory Read	Non-Posted
Memory Write	Posted
Memory Read Lock	Non-Posted
IO Read	Non-Posted
IO Write	Non-Posted
Configuration Read (Type 0 and 1)	Non-Posted
Configuration Write (Type 0 and 1)	Non-Posted
Message	Posted
AtomicOp	Non-Posted

Transaction Layer – TLP Types

asiclab

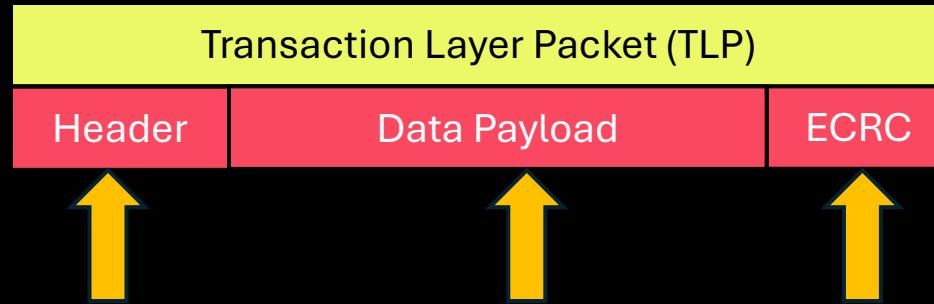
TLP Type	Abbreviated Name
Memory Read Request	MRd
Memory Write Request	MWr
Memory Read Request - Locked Access	MRdLk
IO Read Request	IORD
IO Write Request	IOWR
Configuration Read (Type 0 and 1)	CfgRd0, CfgRd1
Configuration Write (Type 0 and 1)	CfgWr0, CfgWr1
Message Request with Data	MsgD
Message Request without Data	Msg
Completion with Data	CplD
Completion without Data	Cpl
Completion Lock with Data	CplDLk
Completion Lock without Data	CplLk
AtomicOps	FetchAdd, Swap, CAS

TLP Assembly/Disassembly (8b/10b) (128b/130b)



TLP Core Structure

asiclab



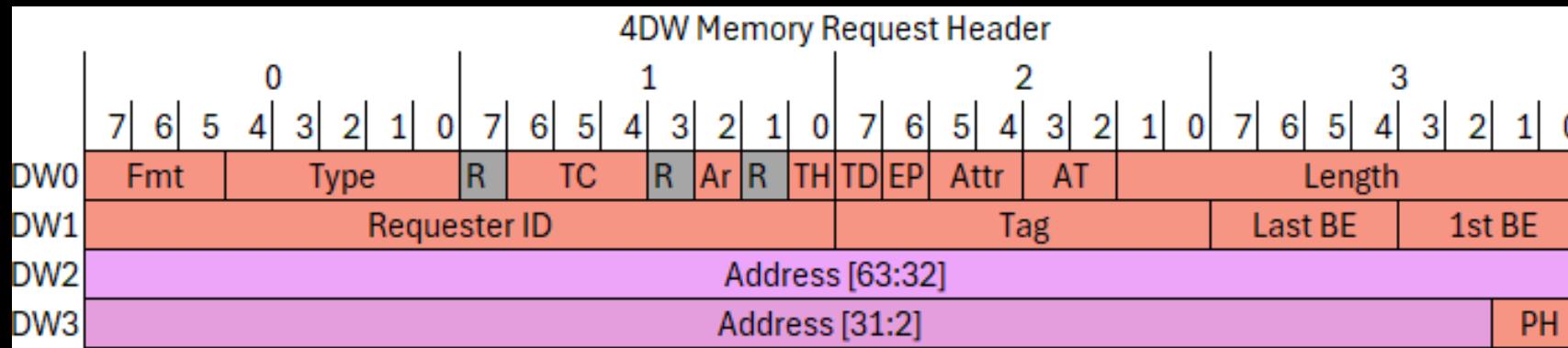
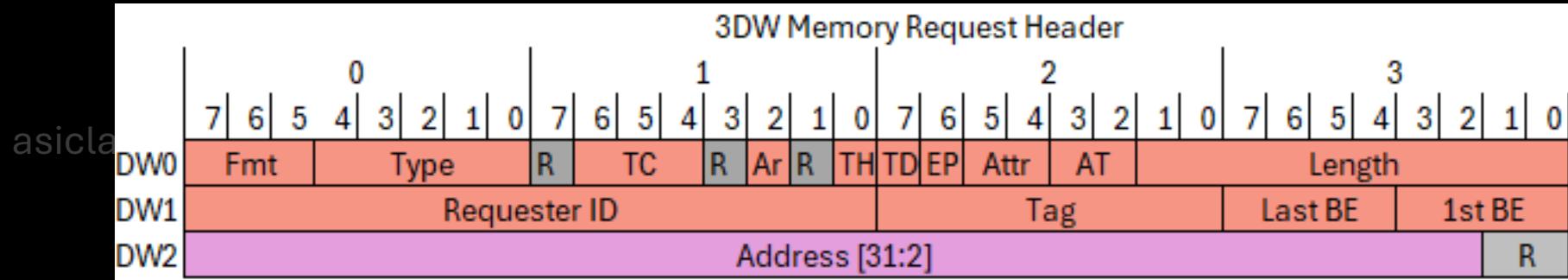
- **Header (3 or 4 DWs in size) may include:**
 - Address, requester ID, tag, transaction type, transfer size, requester ID/completion ID, byte enables, no snoop bit, relaxed ordering bit, traffic class bits...etc.
- **Data Payload (if present)**
 - Present for write request packets and completions with data asiclab
 - Contains between 1 and 1024 DWs of data
- **ECRC or Digest (Optional)**
 - If enabled, contains 32-bit ECRC used for end-to-end error checking (ECRC)

Transaction types

- Transactions Types for Different Address Spaces

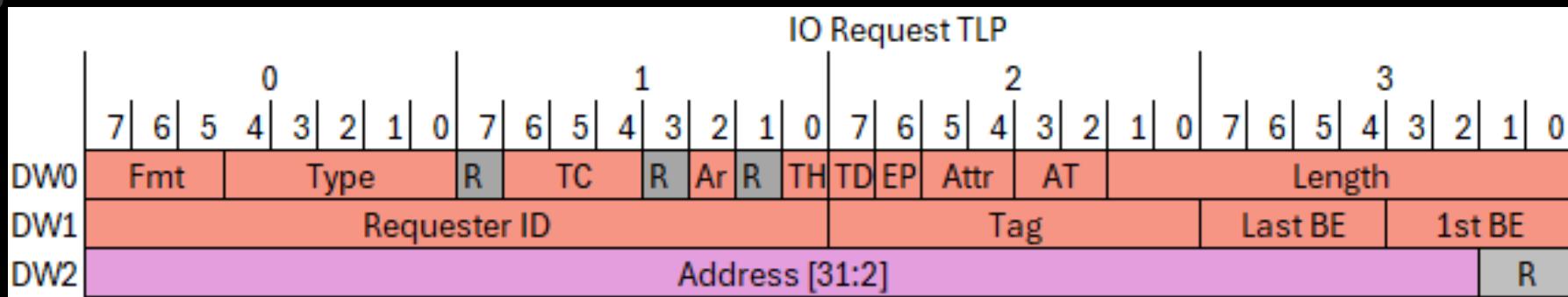
Address Space	Transaction Types	Basic Usage
Memory	Read	Transfer data to/from a memory-mapped location
	Write	
I/O	Read	Transfer data to/from an I/O-mapped function
	Write	
Configuration	Read	Device Function configuration/setup
	Write	
Message	Baseline (including Vendor Defined)	From event signaling mechanism to general purpose messaging

Memory Request



I/O Request

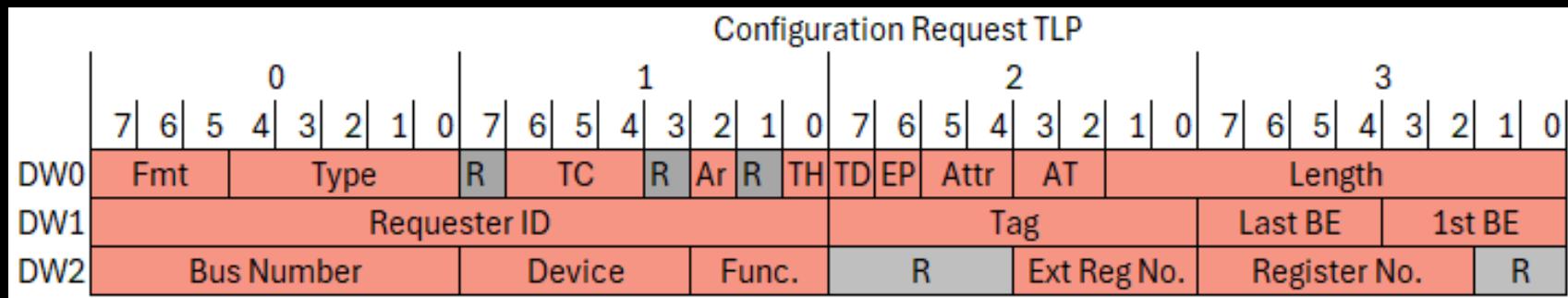
asiclab



asiclab

Configuration Request

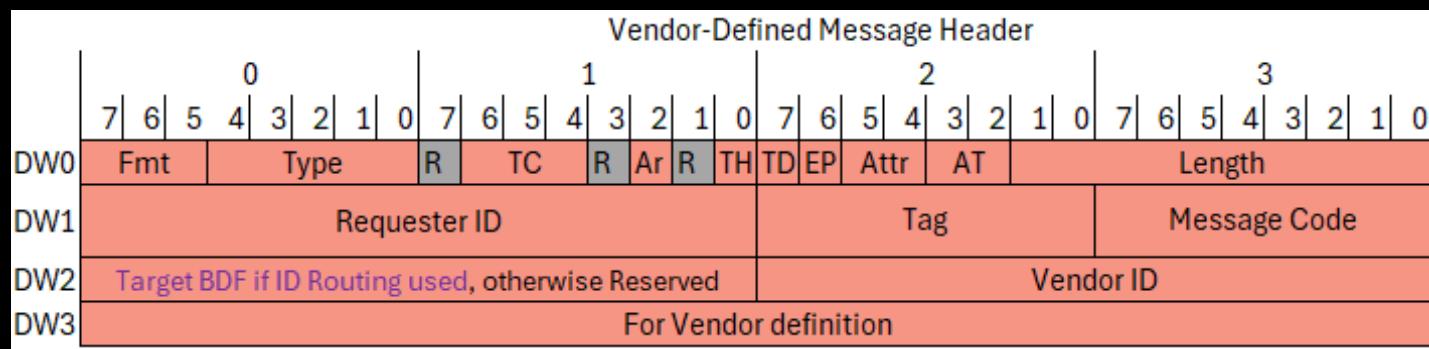
asiclab



asiclab

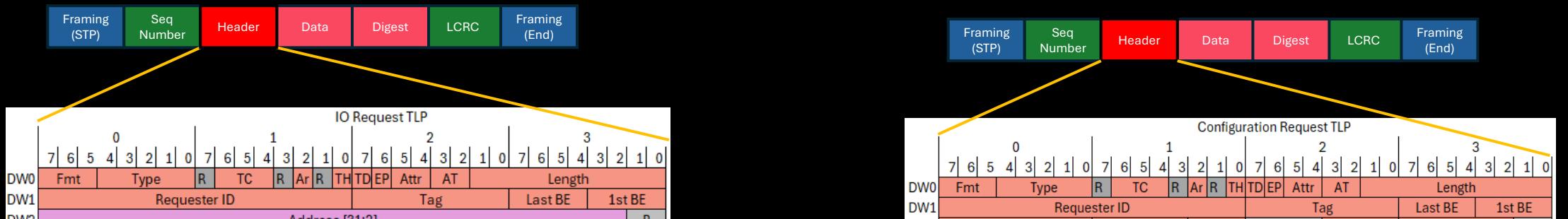
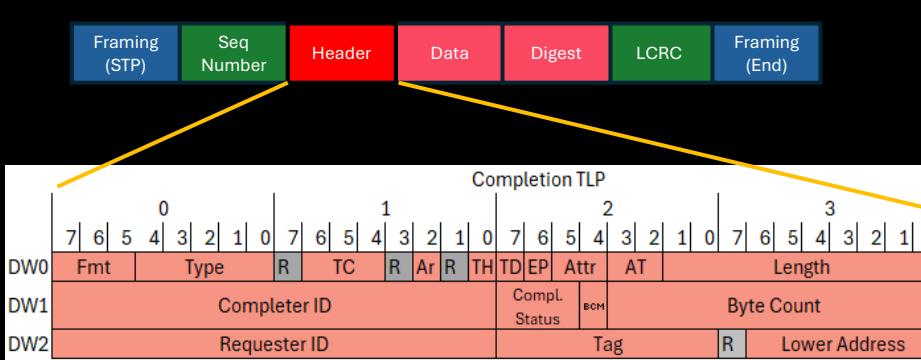
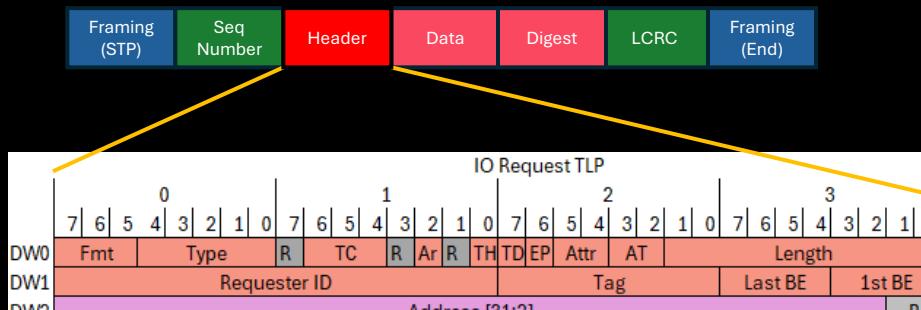
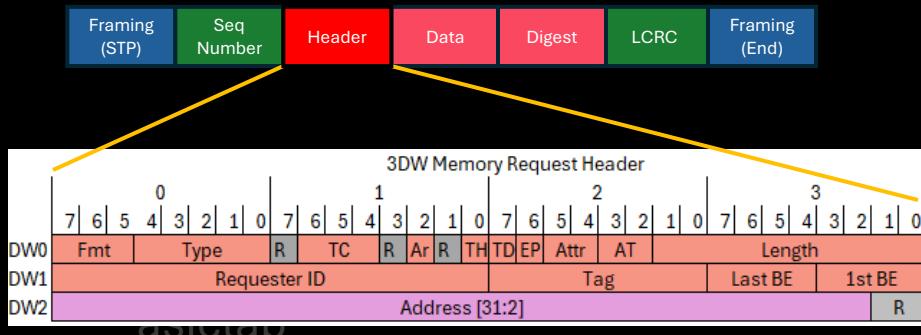
Message Request

asiclab

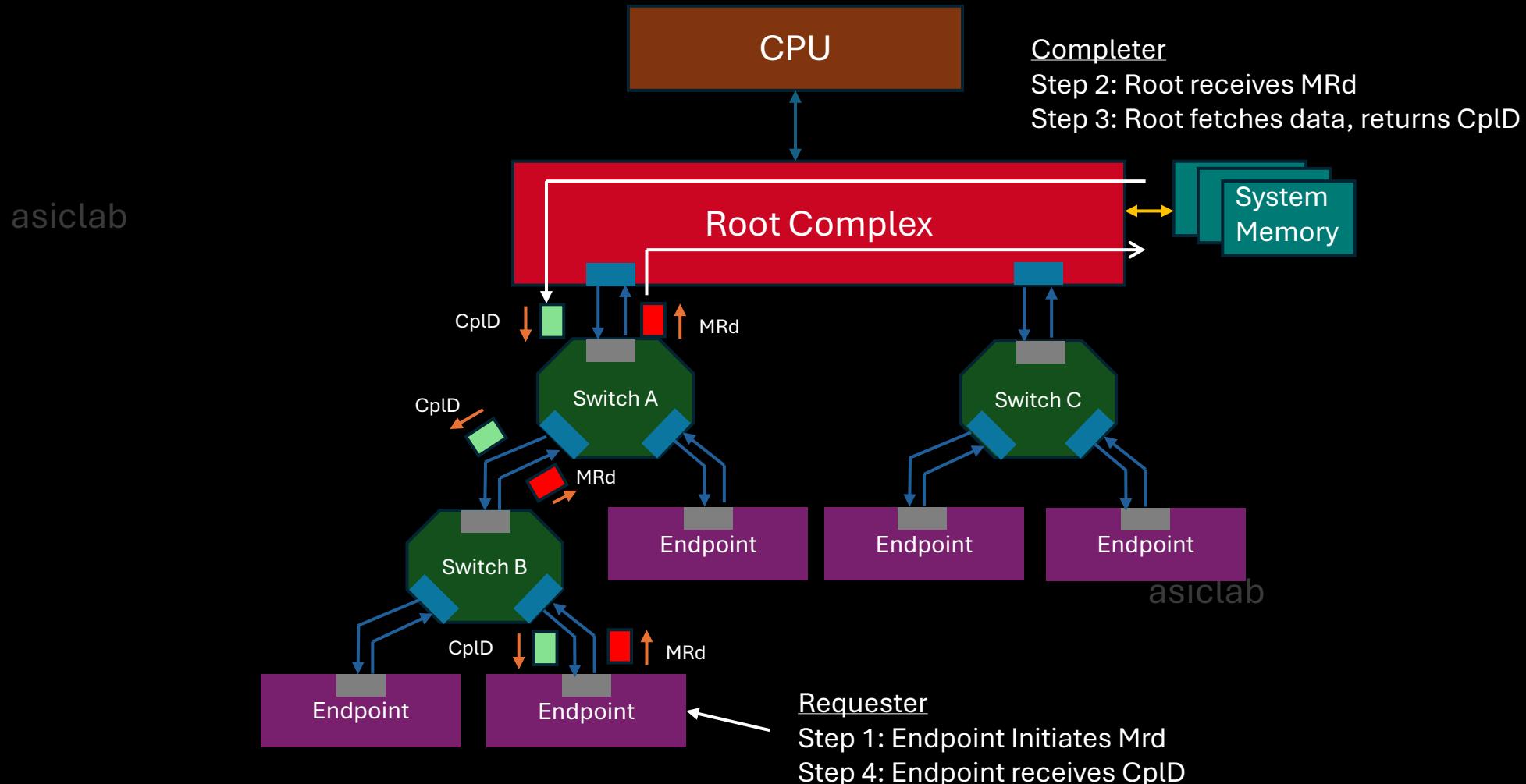


asiclab

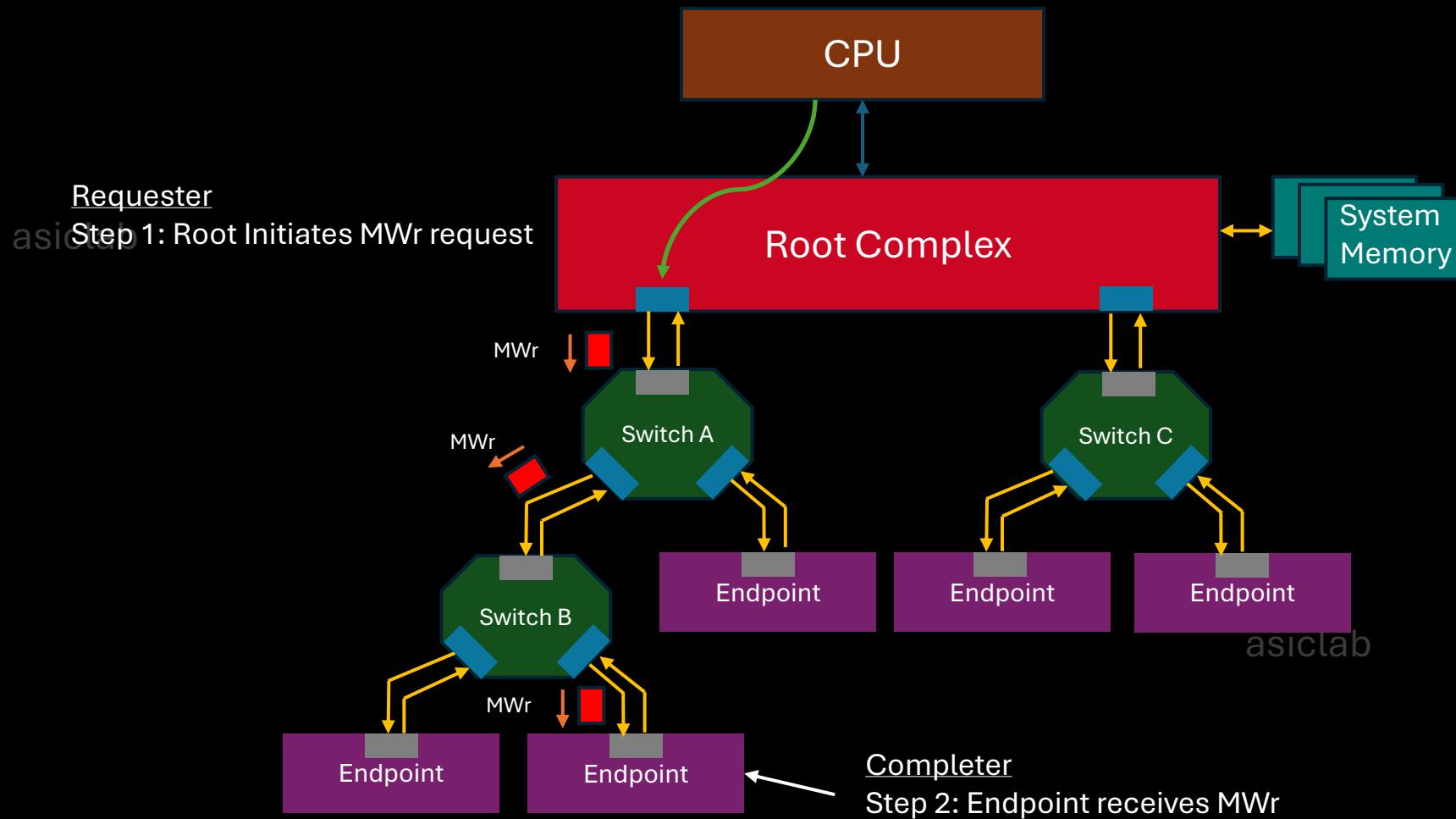
TLP Header format



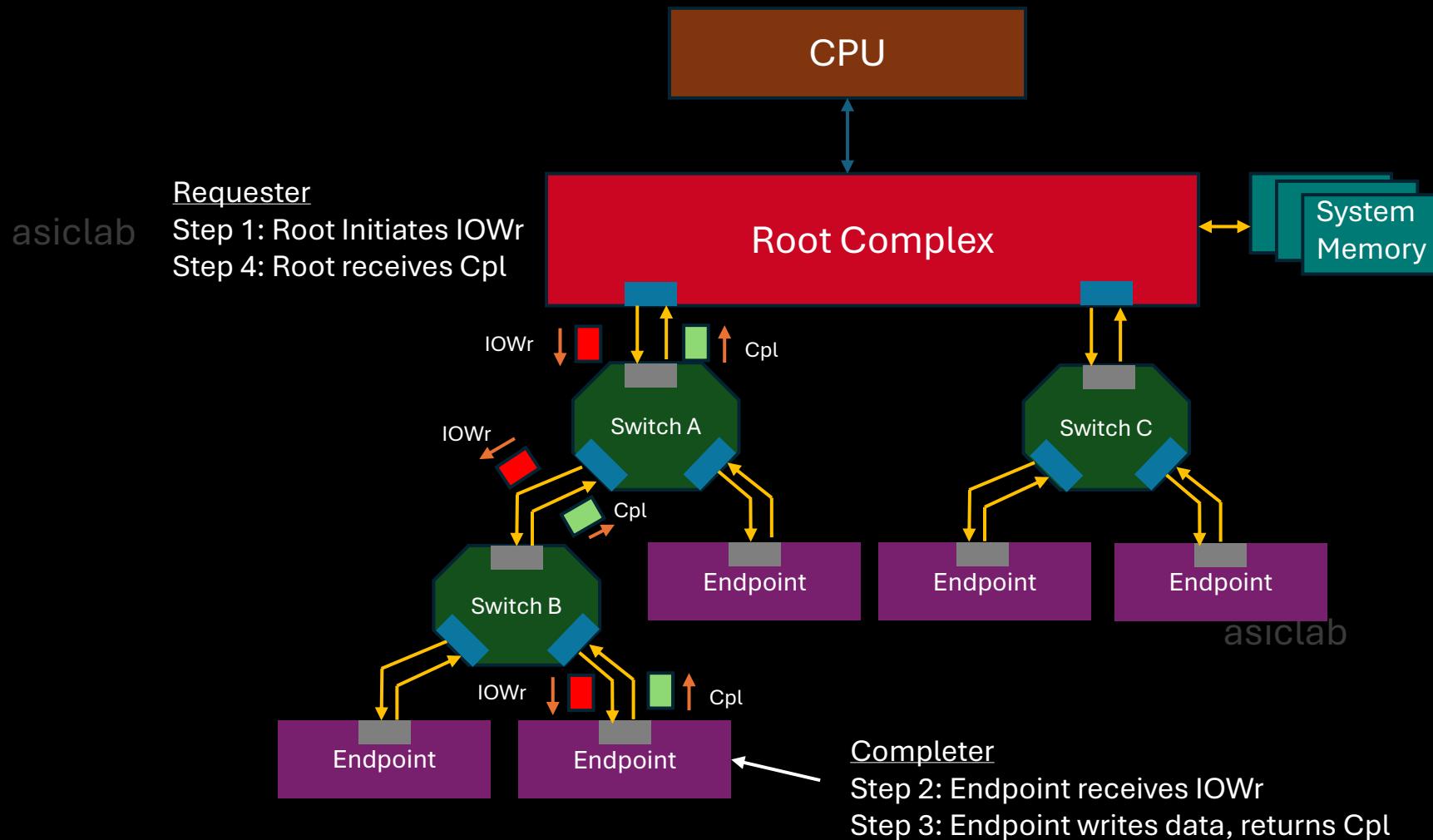
Example: Memory Read (Non-Posted)



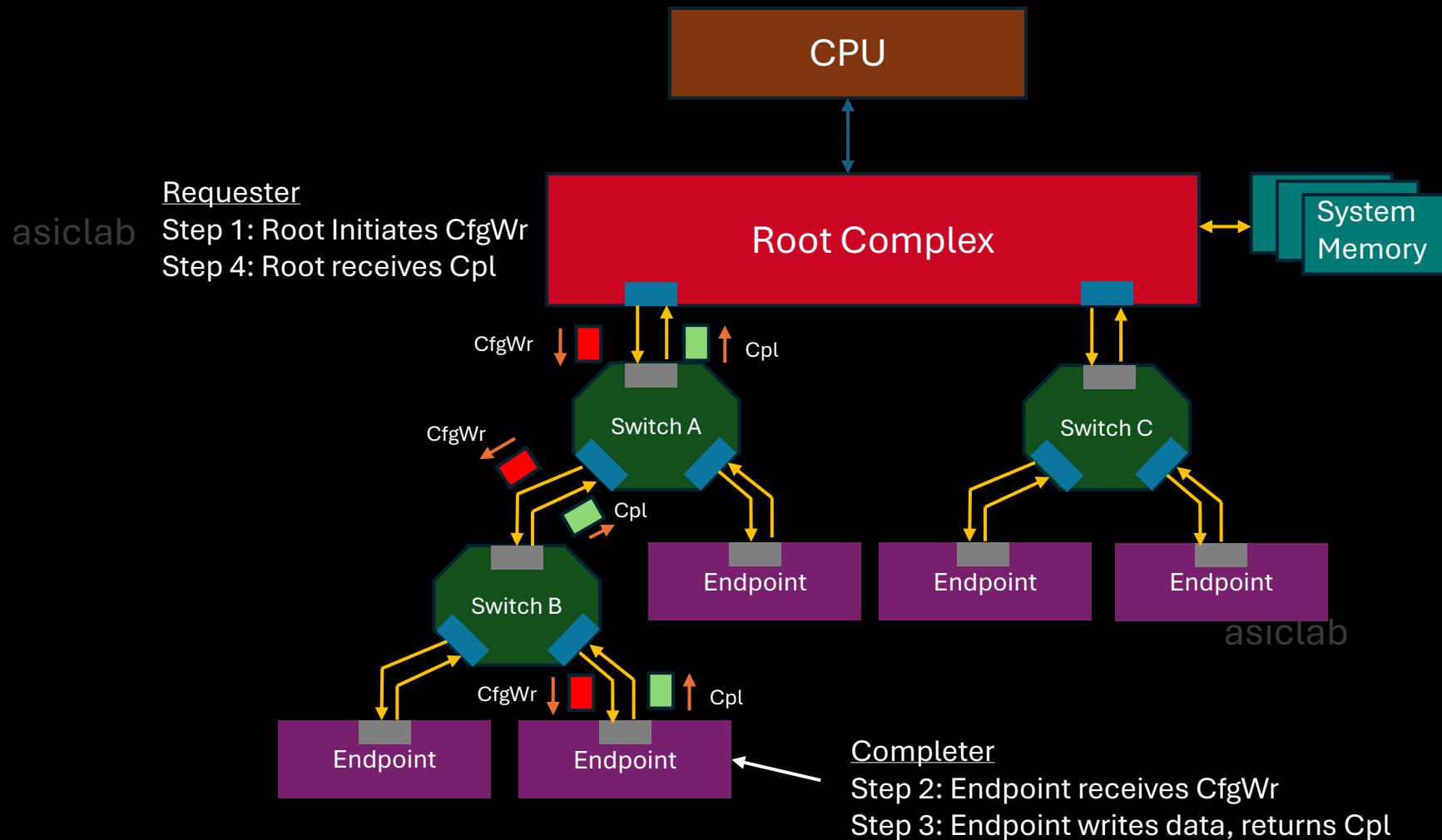
Example: Memory Write (Posted)



Example: I/O Write (Non-Posted)



Example: Cfg Write (Non-Posted)



asiclab

Quality of Service (QoS) and Flow Control

asiclab

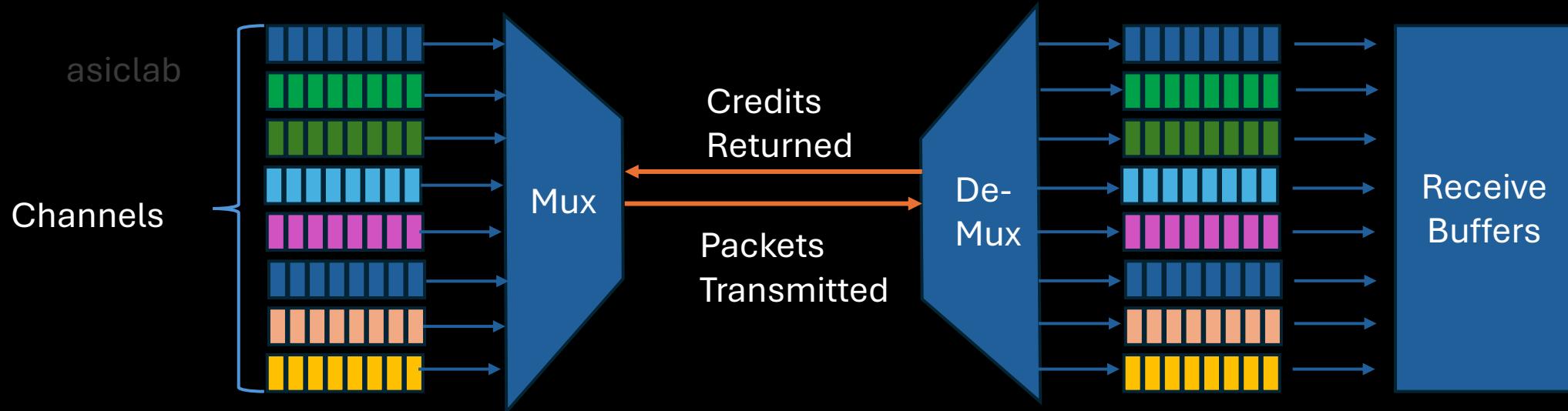
Quality of Service (QoS)

- **Quality of Service (QoS)** describes the ability of the network to manage transmission rate, effective bandwidth and latency
- Feature that make QoS possible:
 - Traffic Class – 8 TCs available
 - Virtual Channels – Up to 8 VCs available
 - Arbitration
- Two Classes of Transactions Supported:
 - Isochronous transactions
 - Asynchronous transactions

asiclab

asiclab

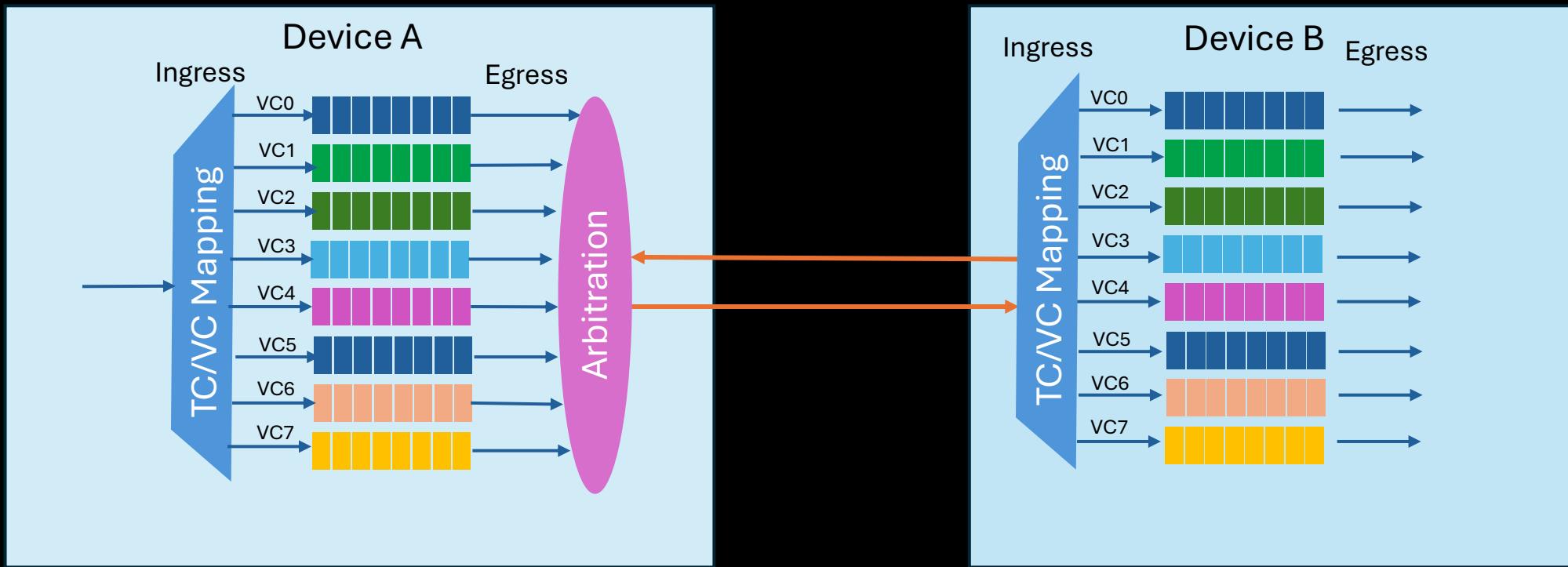
Quality of Service (QoS)



3DW Memory Request Header																
DW0	7	6	5	4	3	2	1	0	R	TC	R	Ar	R	TDEP	Attr	AT
DW1	Requester ID								Tag				Length		Last BE	
DW2	Address [31:2]															R

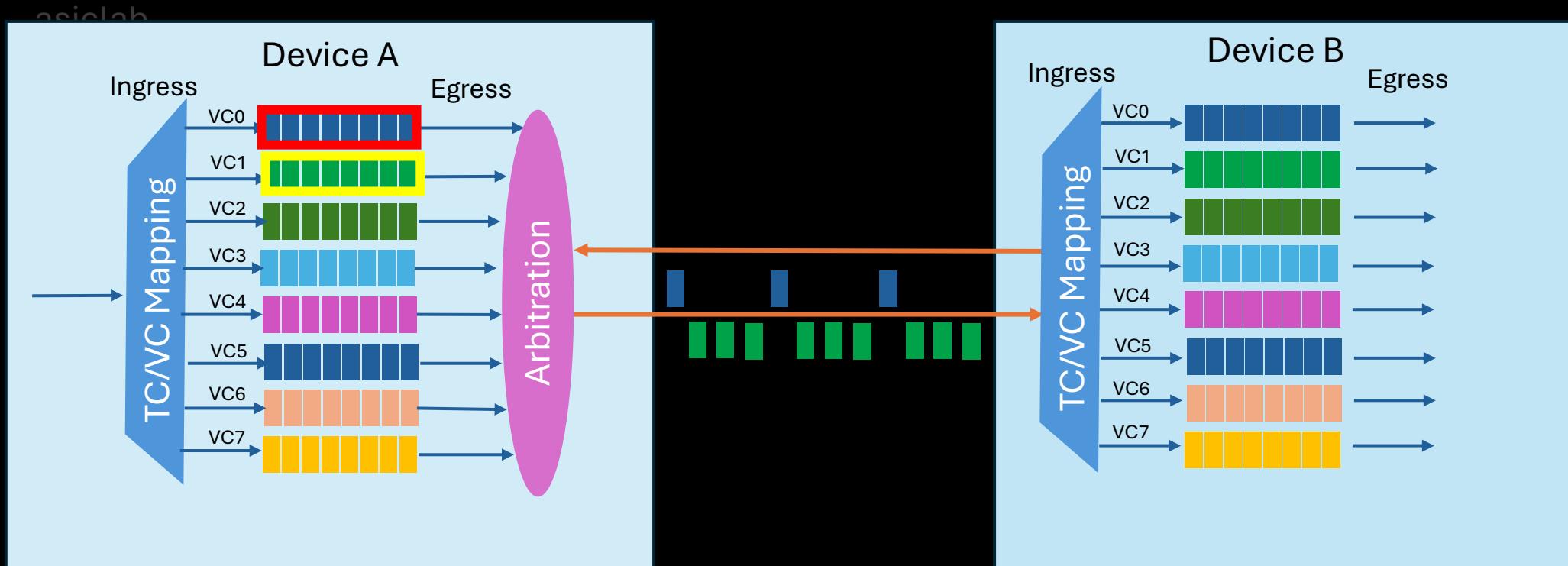
Quality of Service (QoS)

- Virtual Channels VC0 to VC7 mapping and Arbitration



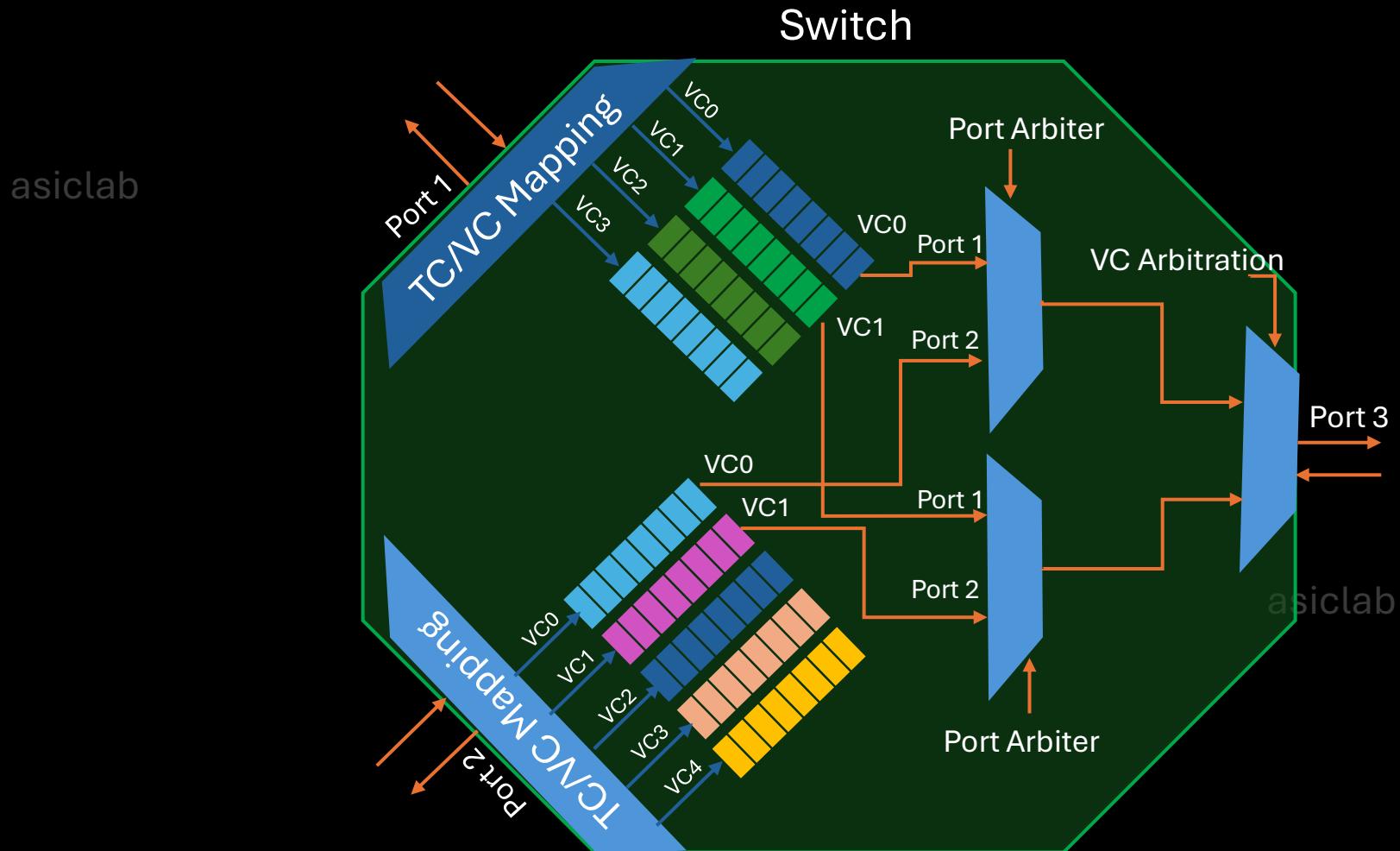
Quality of Service (QoS) Example

- Example of packets associated with VC1 are transmitted at 3 times the rate of packets with VC0 based on Traffic Class set

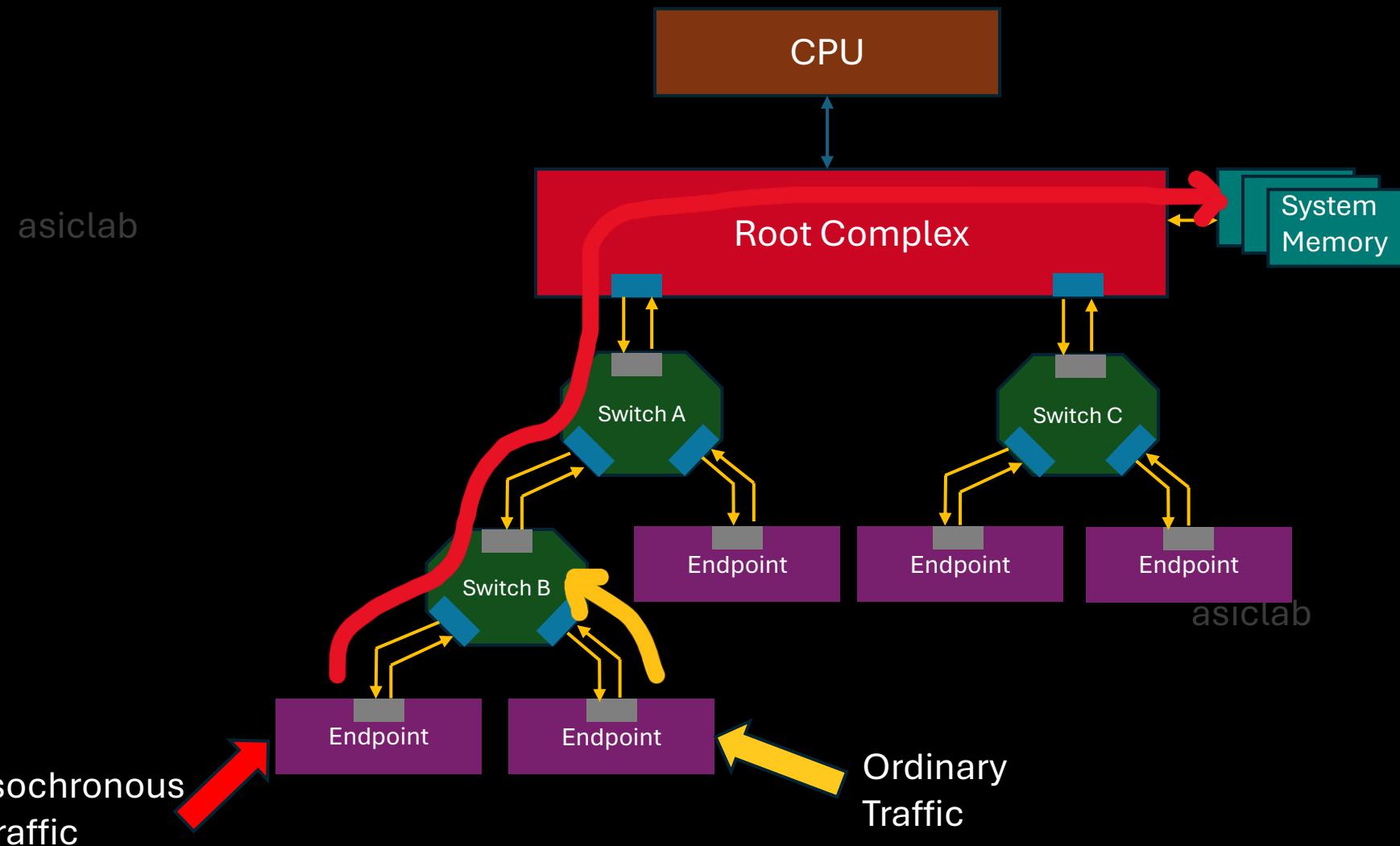


Quality of Service (QoS) Example

- VC Arbitration and Port Arbitration



Prioritized traffic example



QoS Definitions

- **Traffic Class**
 - A TLP header field that remains unchanged as a packet flows from its source to its ultimate destination
 - Examined at each “service point” (e.g.: Switch port)
 - TC value is assigned by software as an indicator of preferred priority
 - Every PCIe device supports TC0 at a minimum
- **Virtual Channel**
 - Implemented in hardware with separate buffers for each VC in each port
 - VCs enable multiple logical data flows over a single physical link
 - Every PCIe device supports VC0 at a minimum

Transaction ordering

- Packets of the same Traffic Class are ordered according to transaction ordering rules
 - Packet type dictate the ordering rules
 - asiclab • Strict Ordering
 - Relaxed Ordering
 - ID-Based Ordering
- Packets of different TCs have no ordering relationship

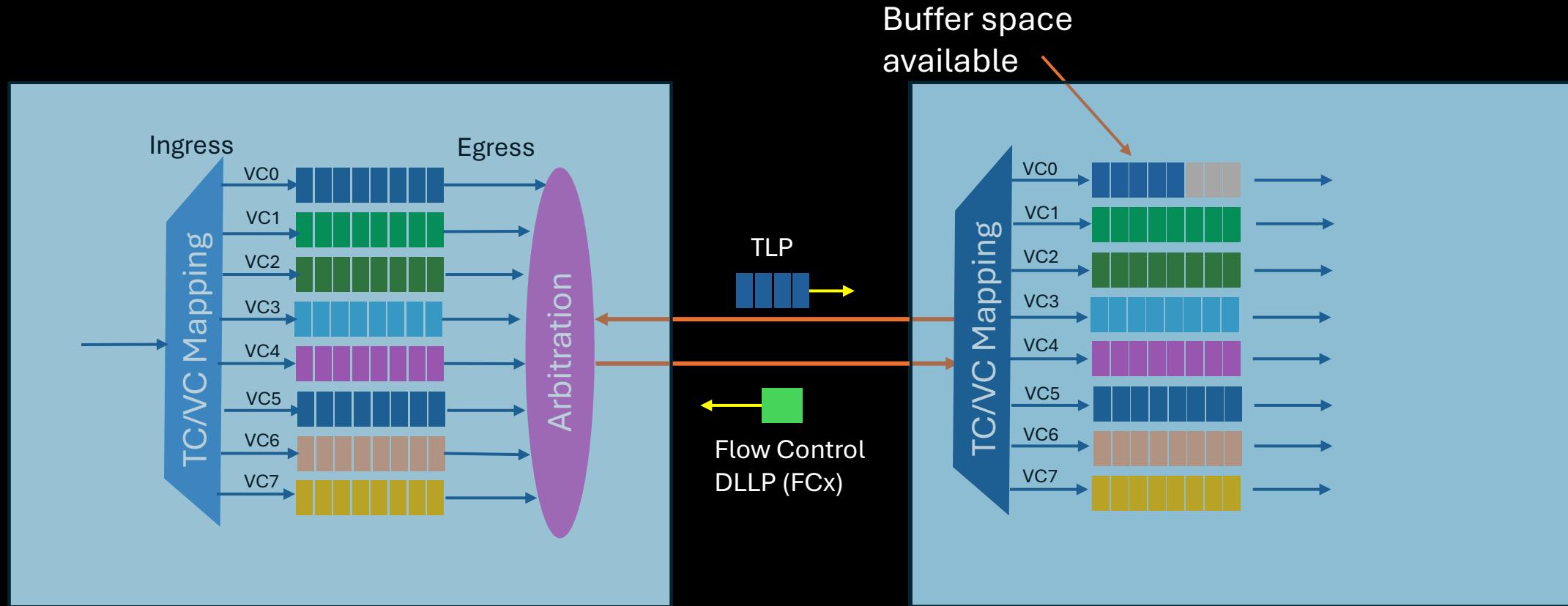
asiclab

Flow Control

- Eliminate inefficiencies of PCI (Retries and Disconnects)
- Receiver periodically updates transmitter on available buffer space
 - Flow control on each link (not end-to-end)
 - Separate mechanism for each Virtual Channel
- Transmitter won't send packet unless receiver has enough buffer space to take it
- Flow Control applies only to TLPs
- DLLPs and Ordered Sets have no Flow Control

asiclab

Flow Control - Example



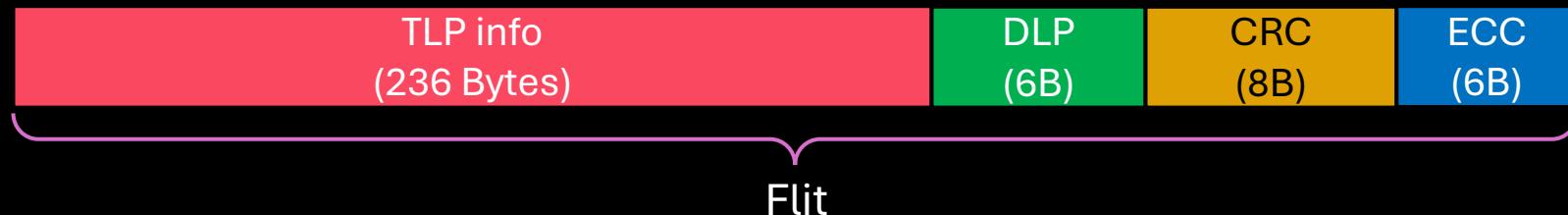
Receiver periodically sends Flow Control Update DLLPs

asiclab

PCIe 6.0 Overview

asiclab

What is a Flit: (New with PCIe 6.0)

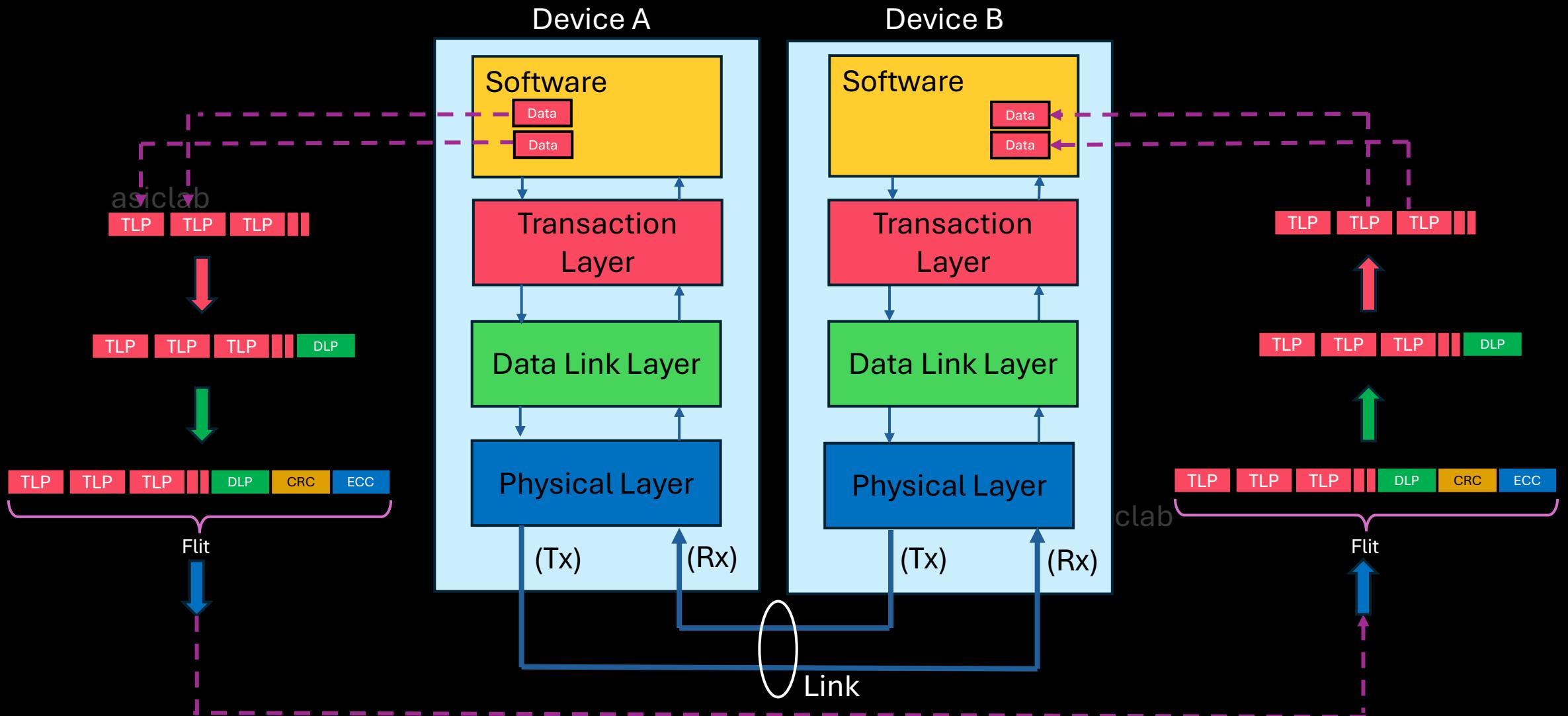


asiclab

- From Gen 6 Flit mode which is applicable for any speeds lower than Gen6 if the device supports it.
- While in Flit mode the basic unit of transfers are not TLP and DLLP packet anymore, instead all of them are now packed in Flits
- A Flit is 256 bytes in length:
 - 236 bytes of TLP traffic
 - 6 bytes of DLP traffic
 - 8 bytes of Flit CRC
 - 6 bytes of Flit ECCs (FEC info)
- One or more TLP can be packed within a Flit, some TLPs can span across multiple Flits based on the payload size
- ACK/NAK and Retry mechanism is not applicable at Flit level and not for individual TLPs

asiclab

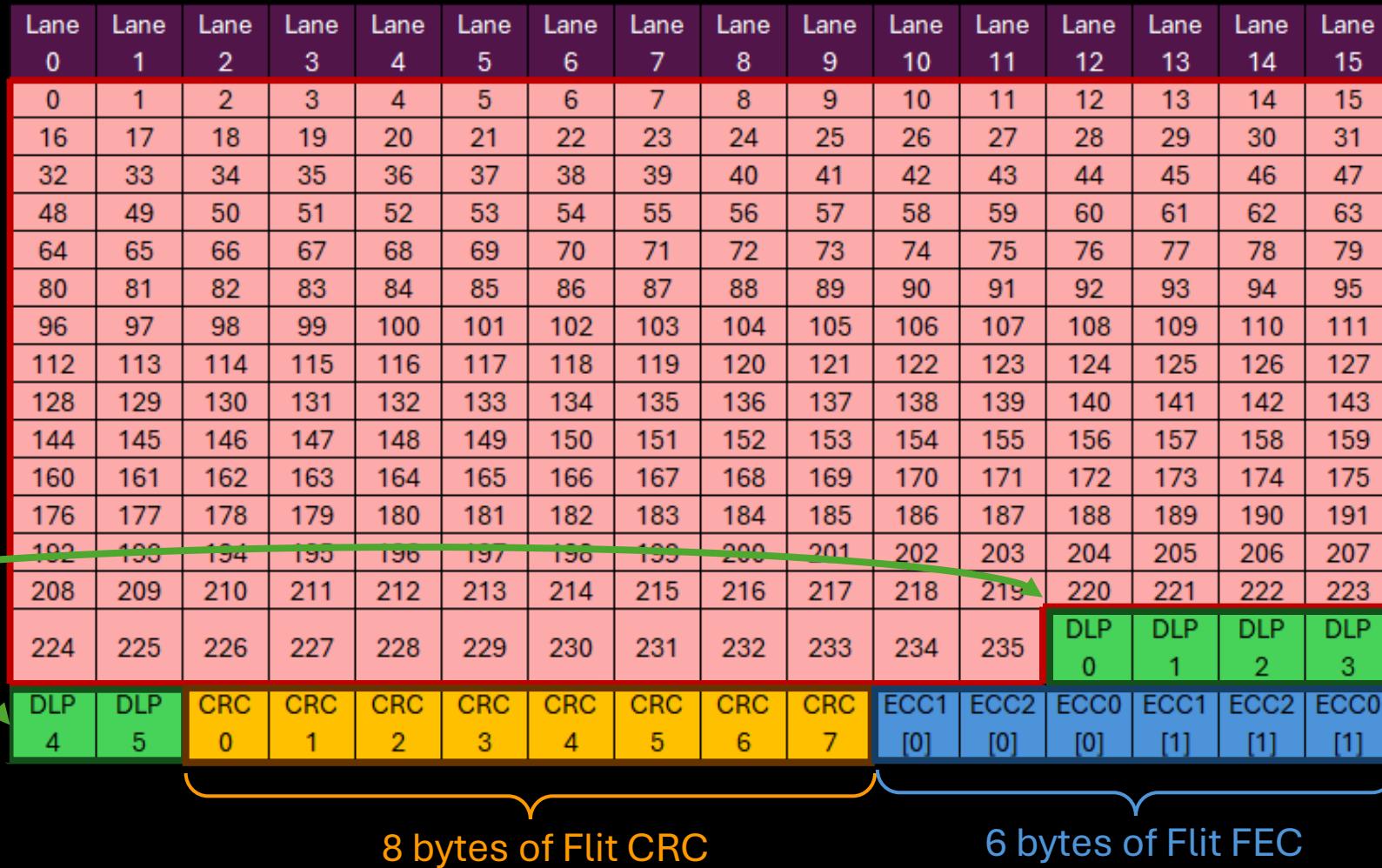
Flit Assembly/Disassembly (New with PCIe 6.0)



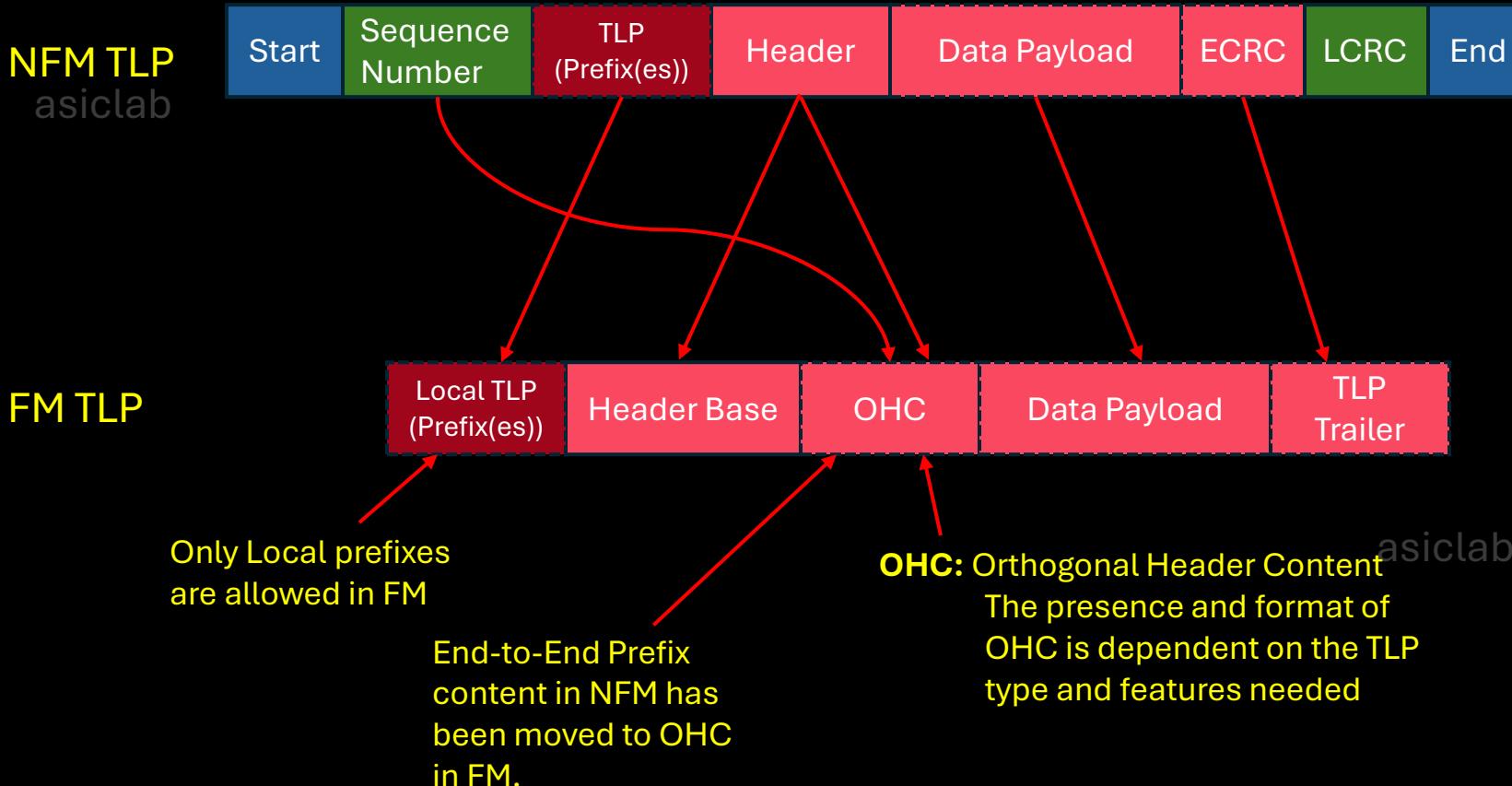
Flit represented across a x16 link: (New with PCIe 6.0)

asiclab
236 bytes of TLP traffic

6 bytes of DLP traffic



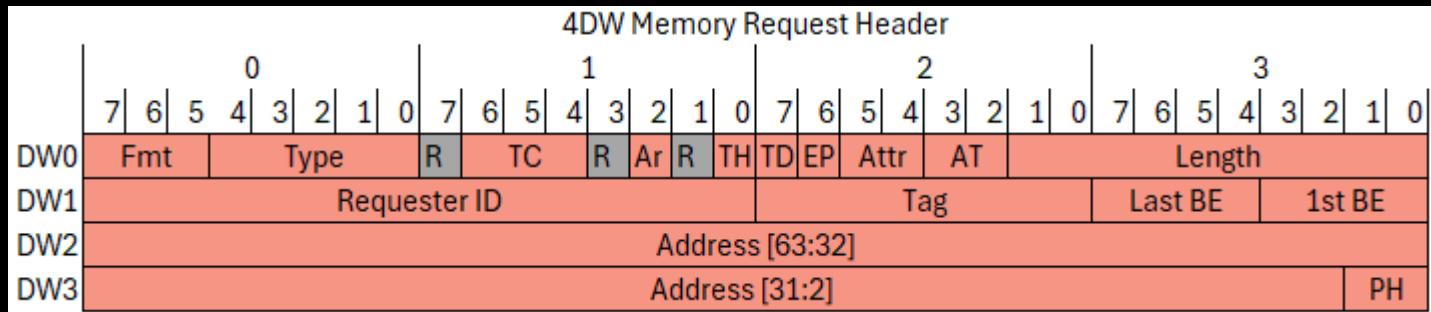
TLP Structure PCIe 6.0 (FM) (New with PCIe 6.0)



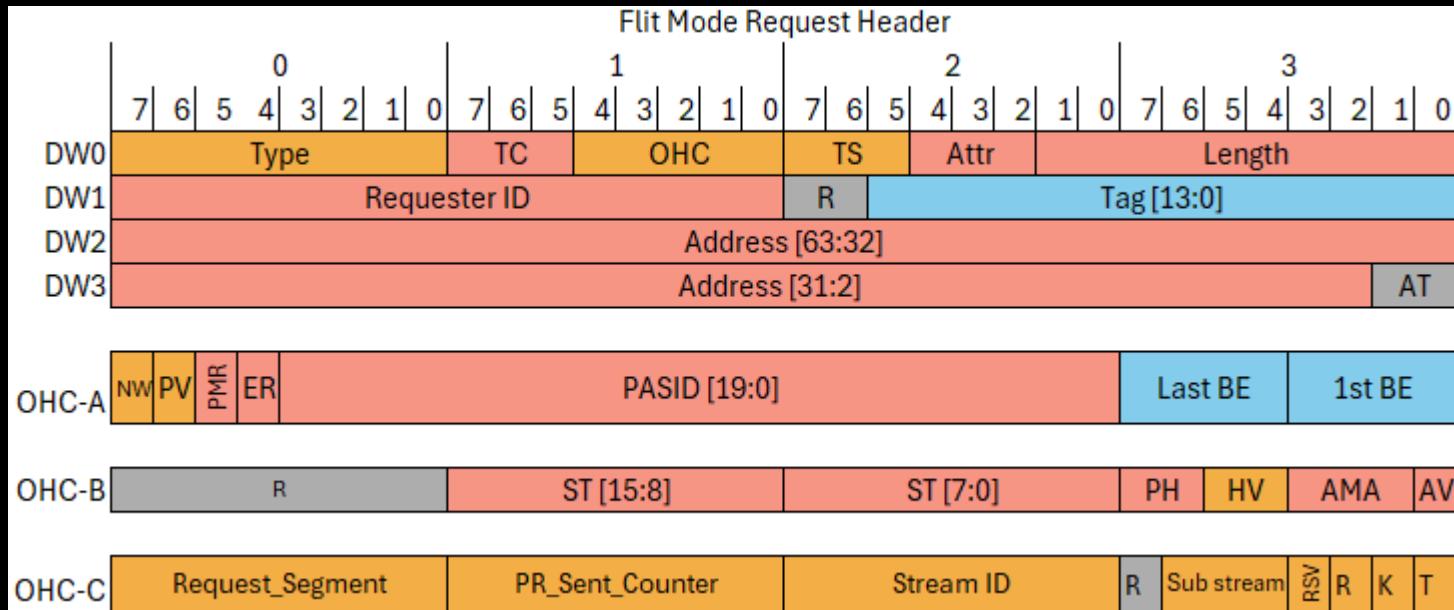
New Flit TLP Structure (example)

(New with PCIe 6.0)

NFM



FM



Key

	New with Flit Mode
	Content unchangeds vs non-Flit Mode
	Modified

asiclab

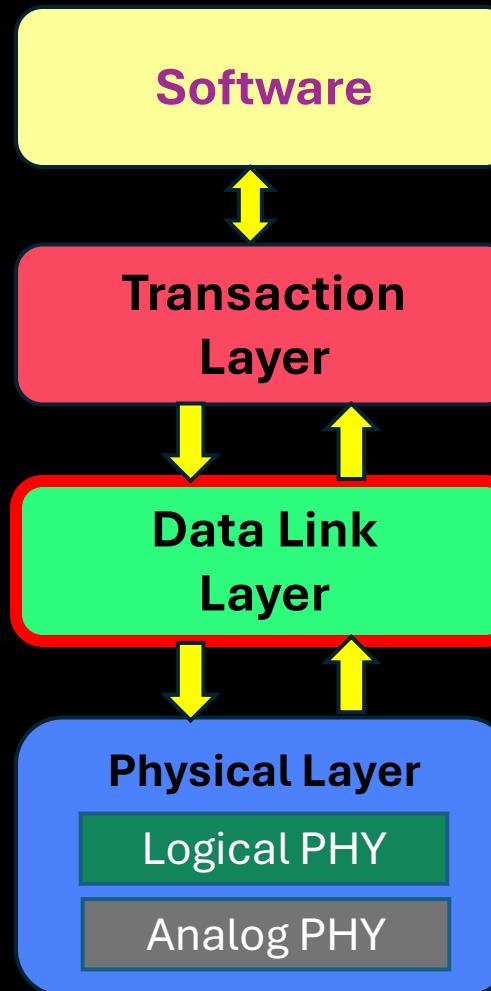
asiclab

Data Link Layer

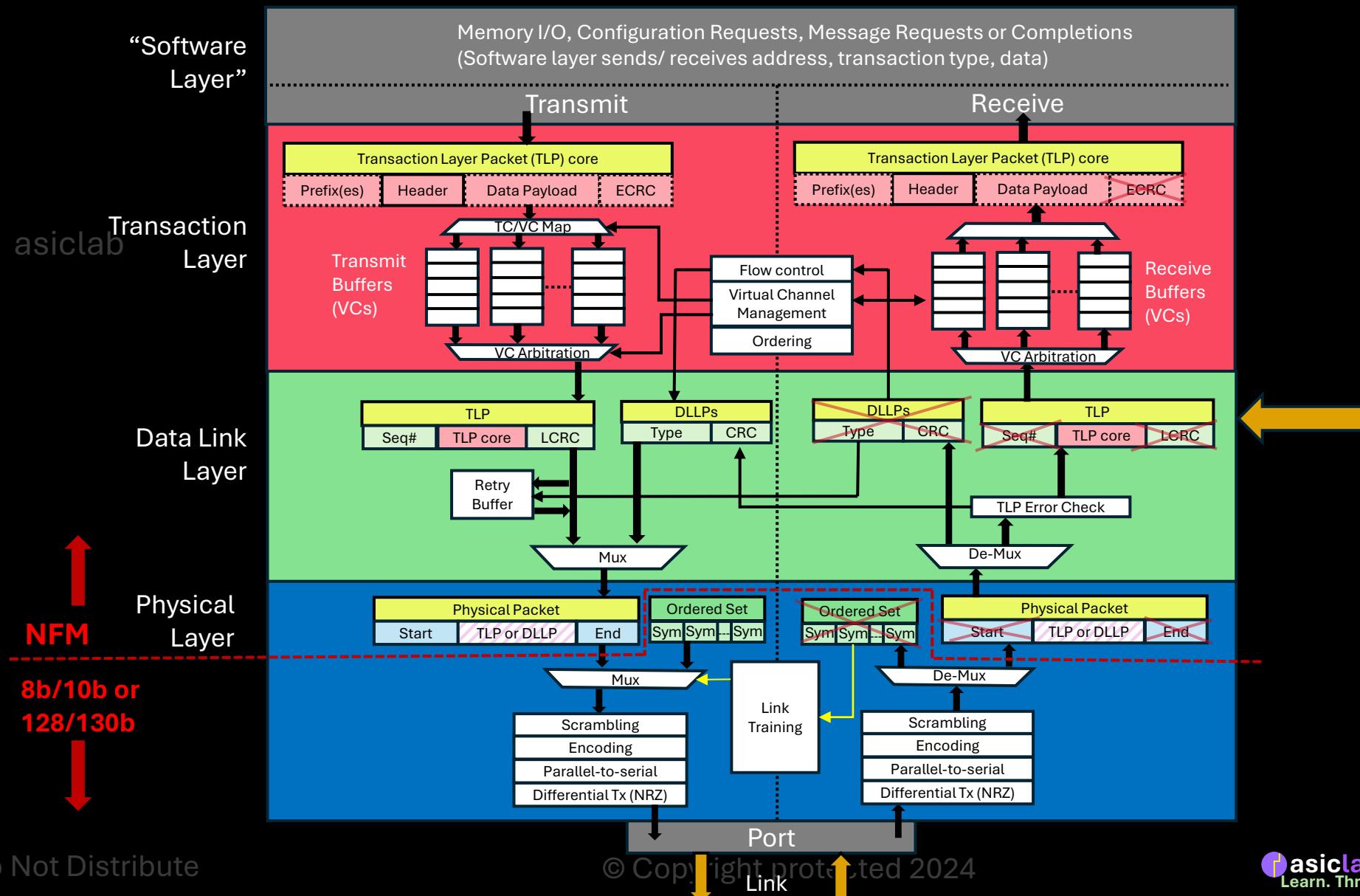
asiclab

Data Link Layer

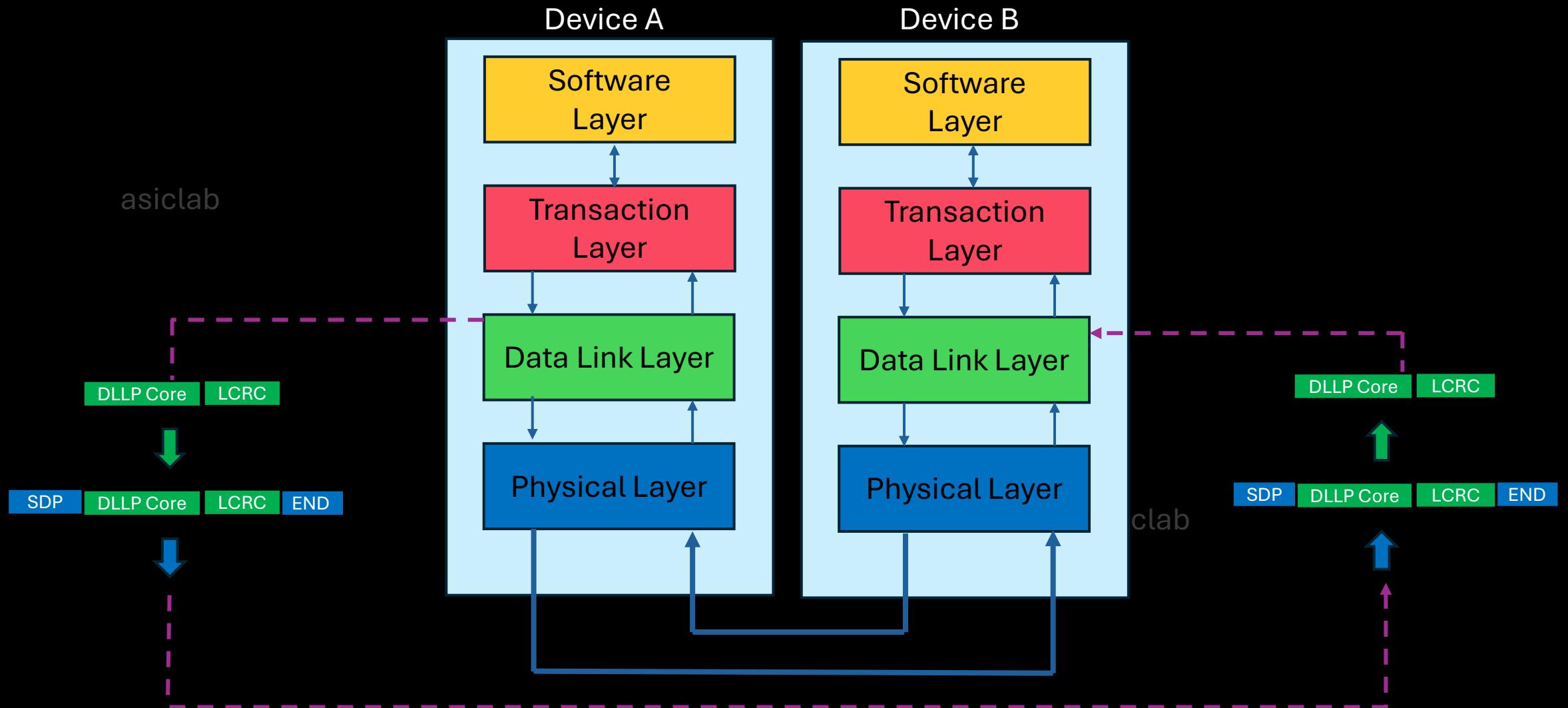
- Provides point to point reliable communication services
- Flow control
- Data integrity
- Link state protocol



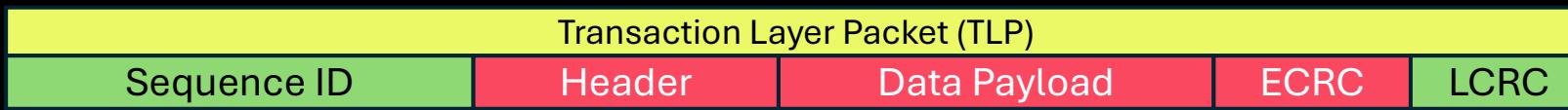
PCIe Device Layer Details 5.0 Non Flit Mode (NFM)



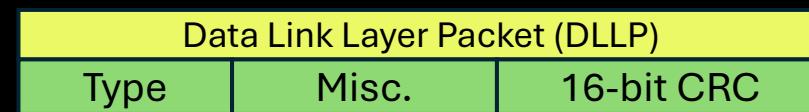
DLLP Origin and Destination (8b/10b)



TLP / DLLP Structure at Data Link Layer



asiclab

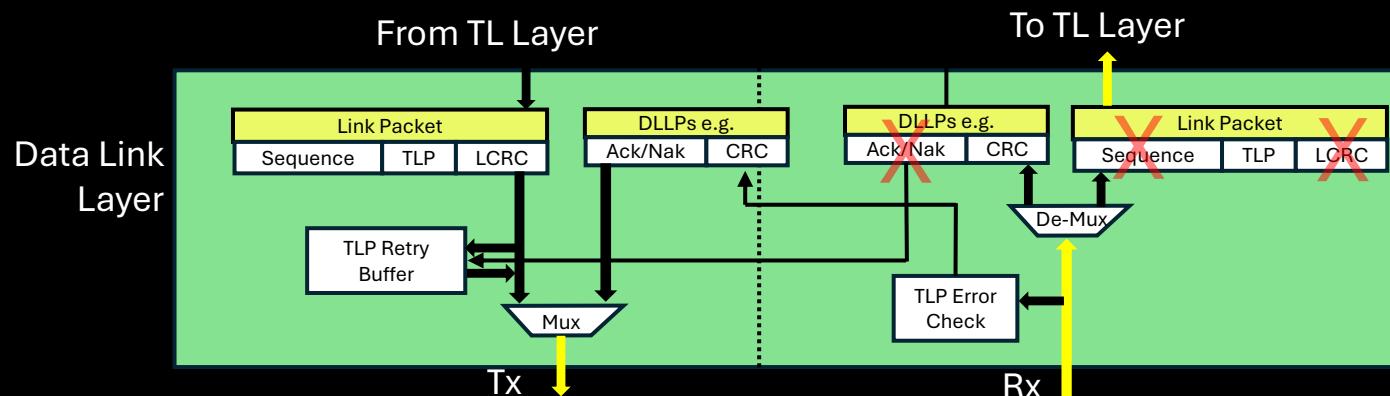


- TLP
 - Sequence ID field is 16 bits: a 12-bit value padded with 4 zeroes at the front end used to associate an Ack/Nak DLLP with a TLP in the Retry Buffer,
 - 32-bit LCRC used for error checking in receiver
- DLLPs are transaction overhead, so they need to be small
 - Type field is 1 byte
 - Miscellaneous field is 3 bytes
 - CRC is 2 bytes

asiclab

Data Link Layer Replay Mechanism

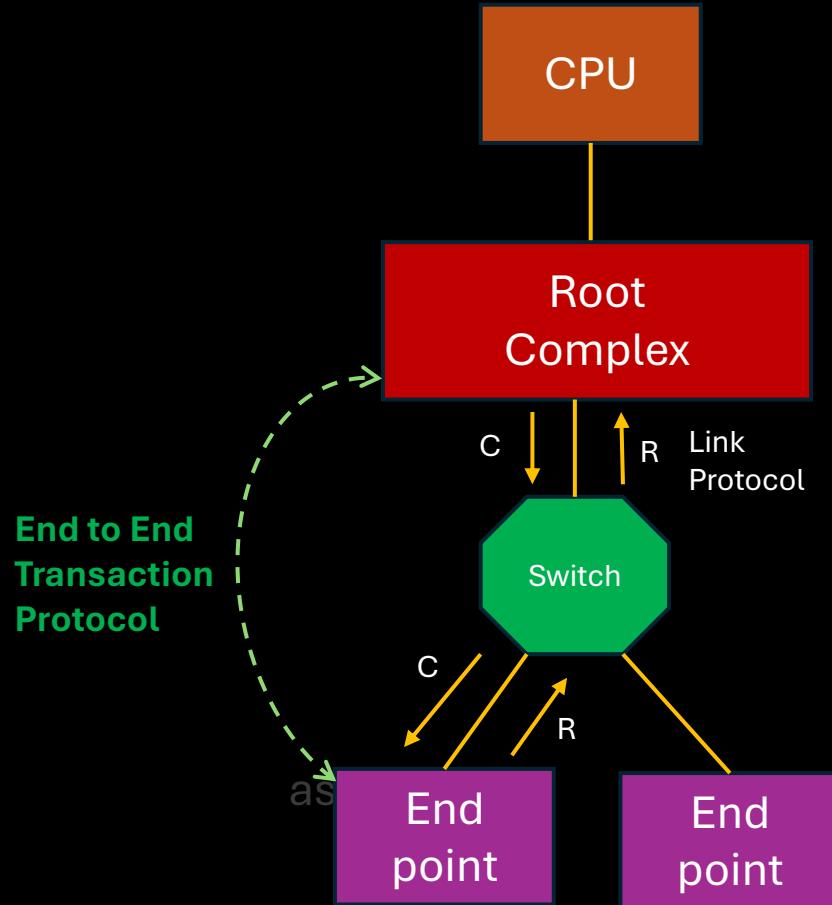
- A copy of each outgoing TLP is stored in the Retry Buffer until the neighbor acknowledges receipt
- Incoming TLPs are checked and an Ack or Nak is generated to acknowledge them



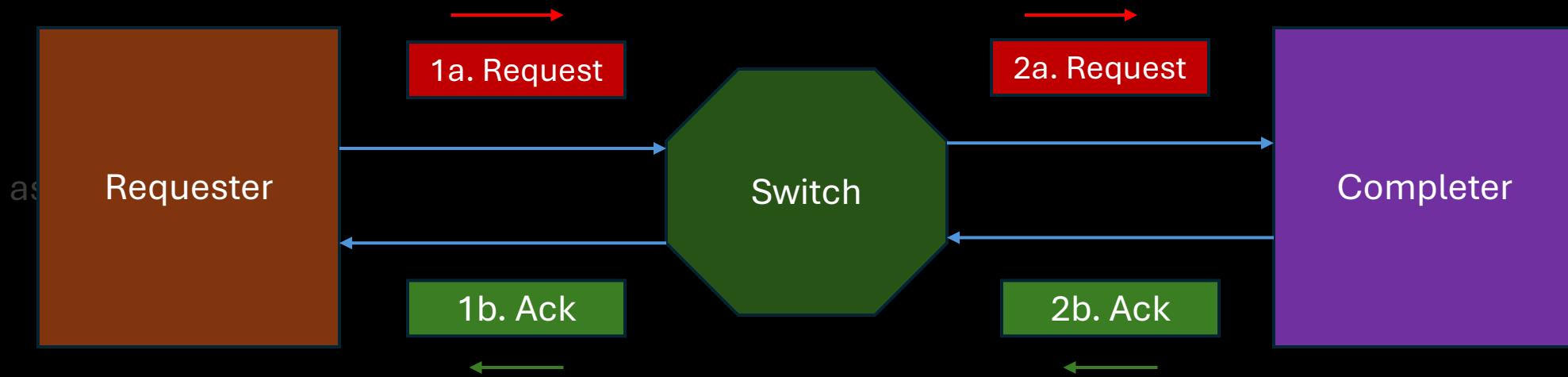
Data Link Layer or Link Layer

- Control Transactions flow between adjacent devices

asiclab

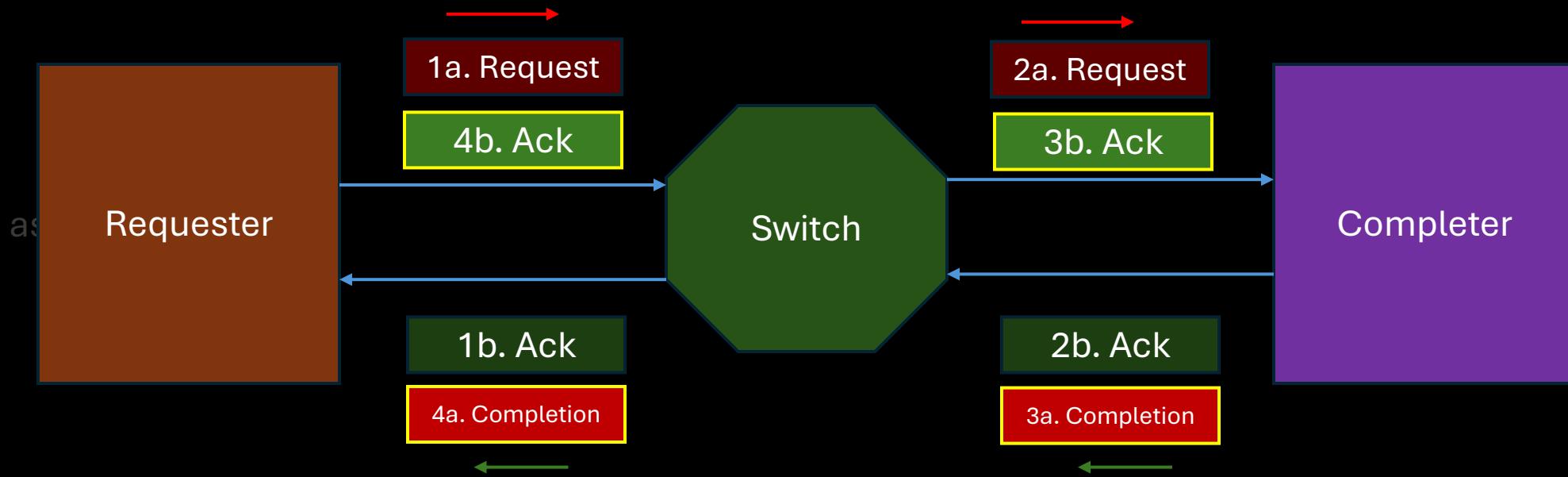


Ack/Nak Protocol, Non-Posted



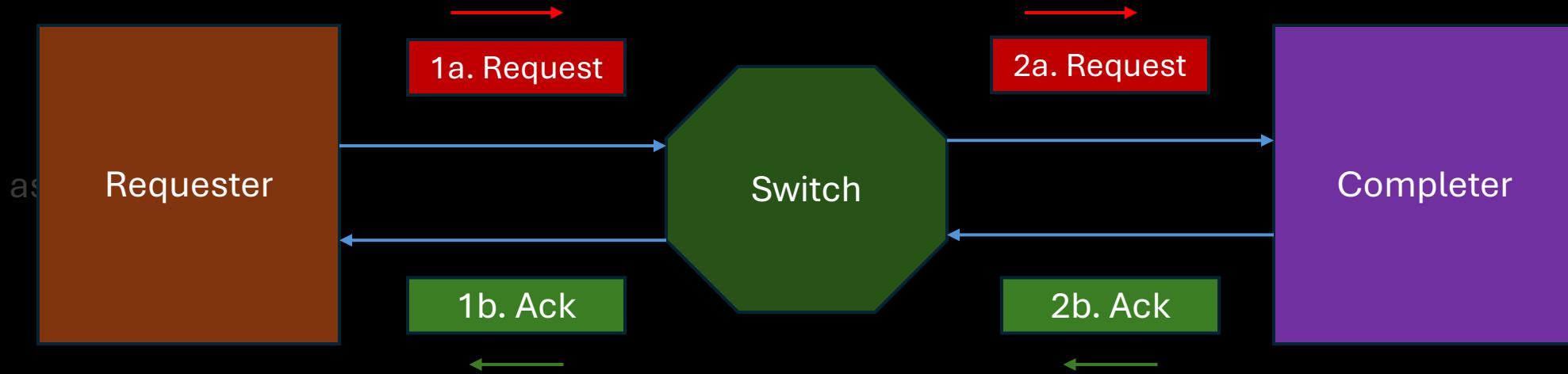
- Receiver performs Link-level data integrity check on every TLP transmission
- Returns Ack if not error, or Nak if error is detected

Ack/Nak Protocol, Non-Posted, Con't



asiclab

Ack/Nak Protocol, Posted



asiclab

Data Link Layer Packets (DLLPs)

- Ack DLLP: TLP Sequence number acknowledgement; used to indicate successful receipt of some number of TLPs
- Nak DLLP: TLP Sequence number negative acknowledgment; used to initiate a Data Link Layer Retry
- InitFC1, InitFC2, and UpdateFC DLLPs: For flow control
- DLLPs used for Link Power Management

asiclab

Other Data Link Layer Functions

- Flow Control Logic Initialization
 - Initialize flow control for default VC0 automatically
 - Initialize flow control for other channels as they are enabled

asiclab

- DLLPs used in initialization process
 - InitFC1-P, InitFC1-NP, InitFC1-Cpl,
InitFC2-P, InitFC2-NP, InitFC2-Cpl

asiclab

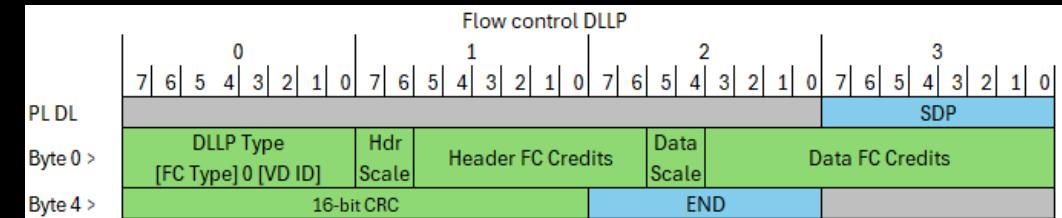
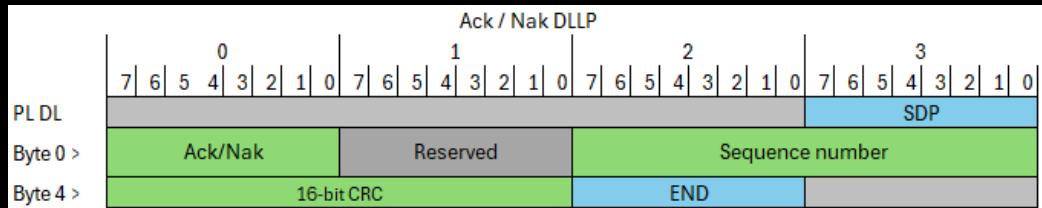
Link Power management

- Negotiate the level of link power down state
- Control Physical Layer PM state transitions

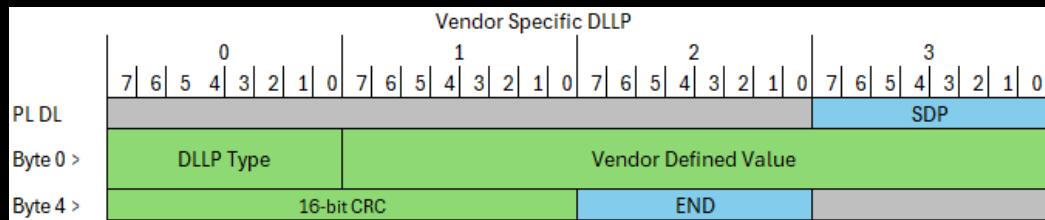
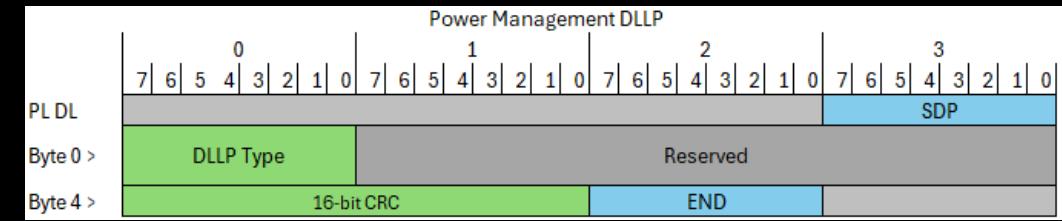
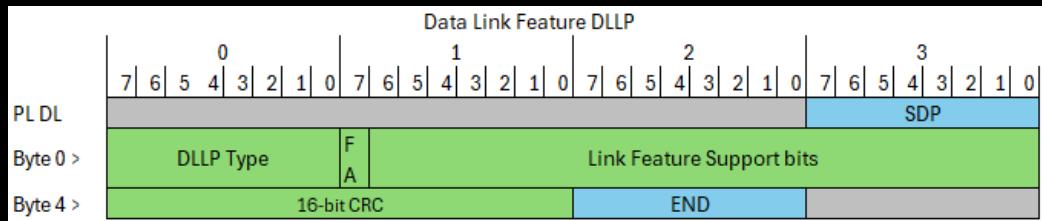
asiclab

asiclab

DLL Packet format



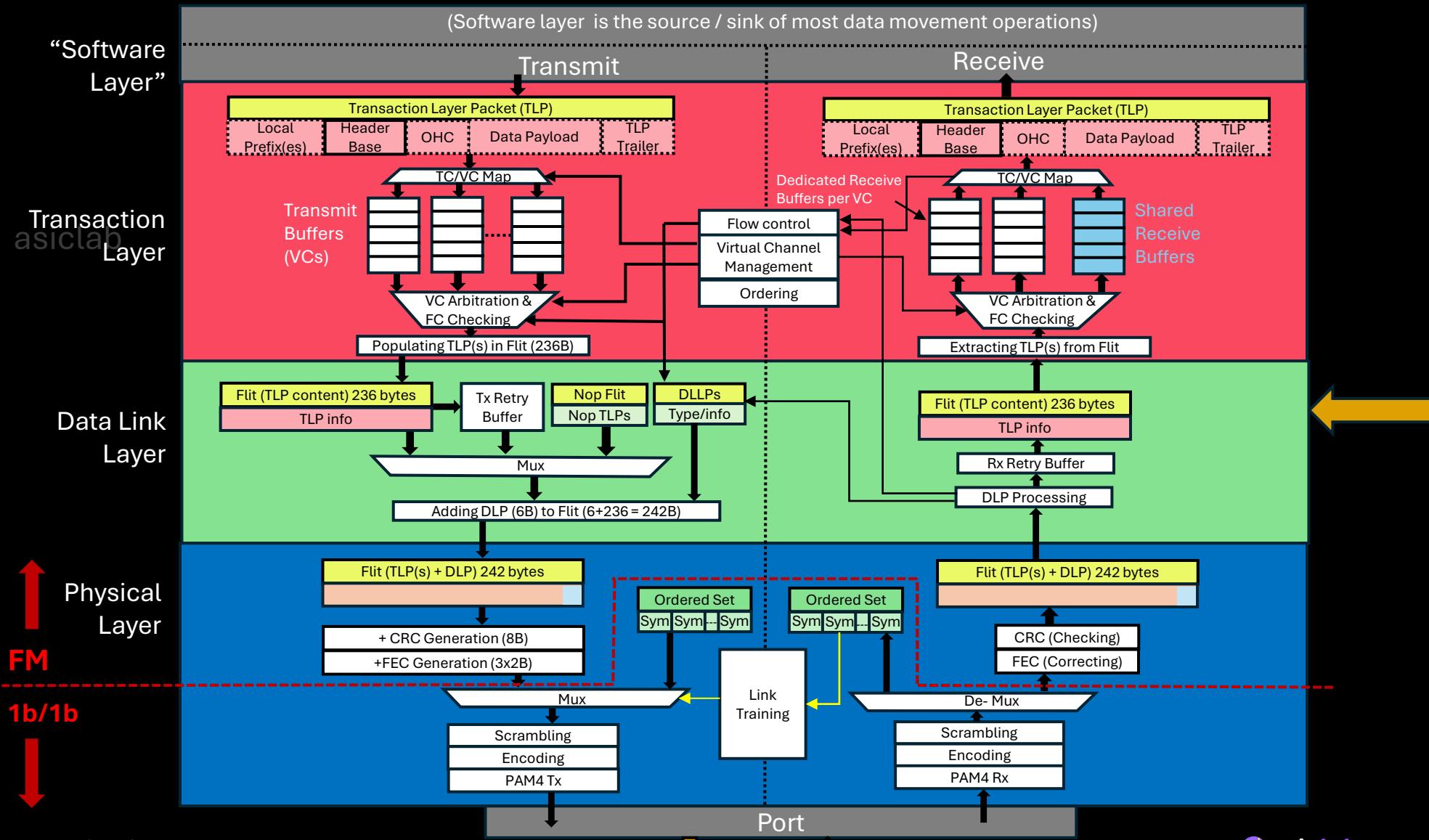
asiclab



asiclab

 Data Link Layer Header
 Physical Layer Header
DO NOT Distribute

PCIe Device Layer Details 6.0 (Flit Mode – FM) (1b/1b)



Do Not Distribute

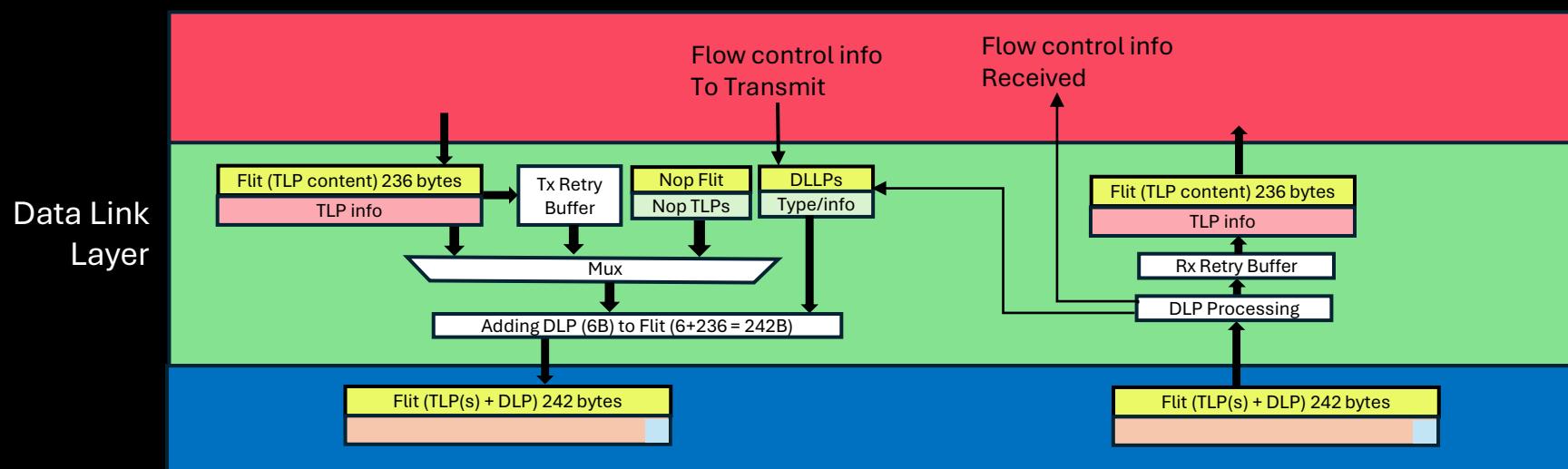
© Copyright protected 2024

Data Link Layer (Flit Mode – FM)

(New with PCIe 6.0)

- Data link layer insert the DLP portion of the flit
- TLP portion of the flit is inserted into the Tx Retry buffer before sending the Flit across the link
- Data link layer also inserts Nop Flits, which are necessary when the Transaction layer don't have anything to fill in the TLP portion
- DLP portion is not held in the Rx and the Tx buffers

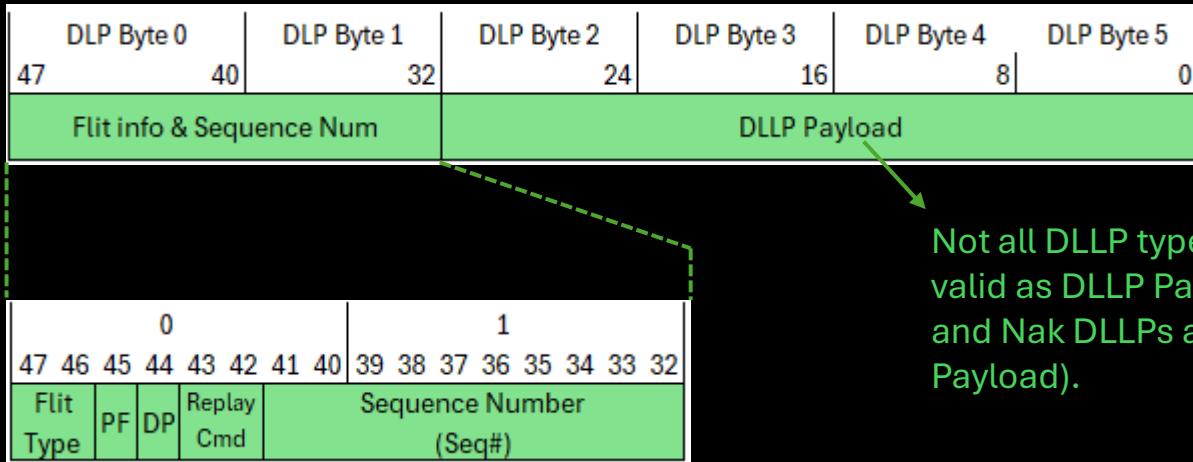
asiclab



DLL Packet format – FM (New with PCIe 6.0)

- DLLP Can be packed into the DLP portion based on one of the encoding field “DP”

asiclab



asiclab

Not all DLLP types from prior generations are valid as DLLP Payload when in FM (e.g. Ack and Nak DLLPs are not supported in the DLLP Payload).

DP: DLLP Payload (IDLE,NOP, Payload Flit)

PF: Prior Flit

Flit Type: Flit Usage

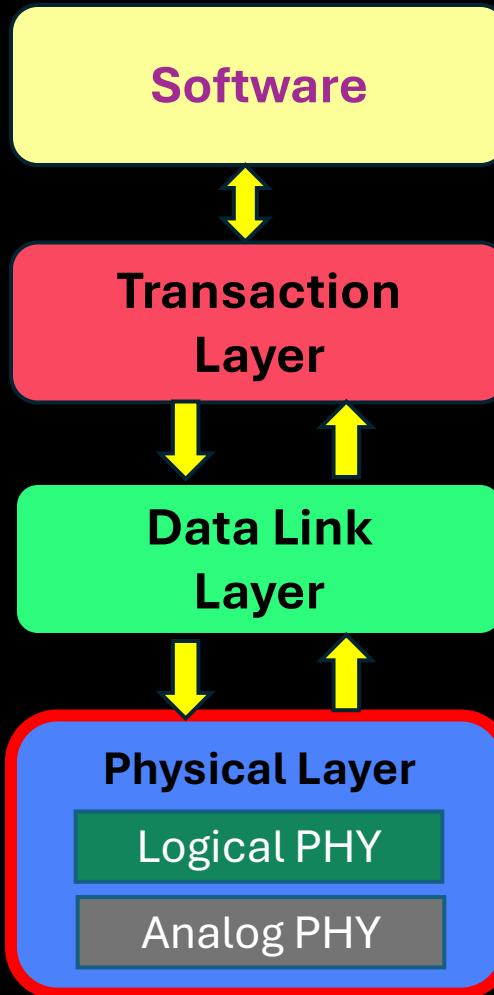
asiclab

Physical Layer

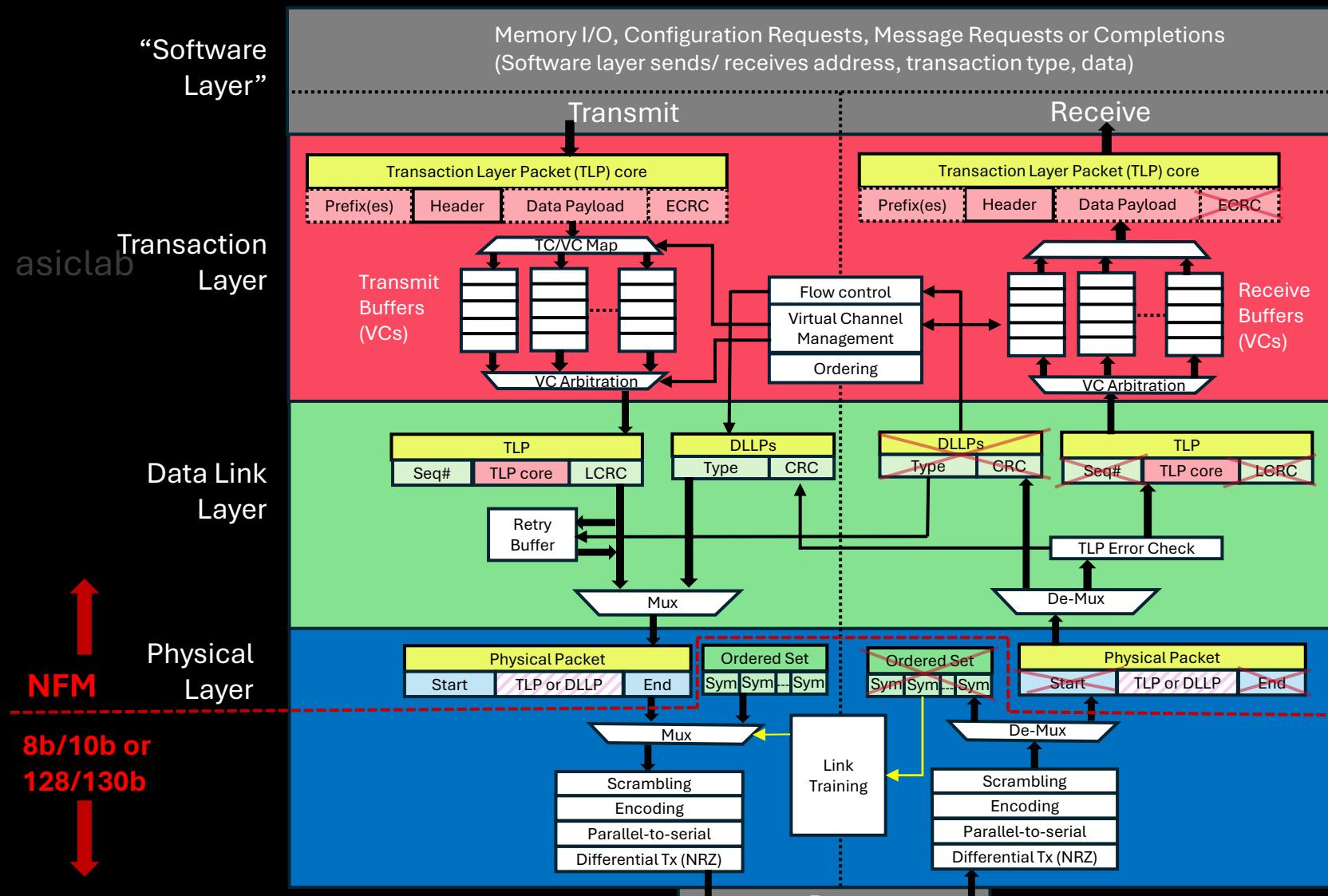
asiclab

Physical Layer

- Logical Phy or Digital Phy
 - Provides packet encapsulation
 - Maps and collects symbols over the lanes
 - 8/10b encode and decode
 - Scrambling
 - Negotiates link state and parameters with peer
 - Link width
 - Number of agents
 - Link data rate
 - Lane polarity
 - Lane reversal
 - Link Power states
 - Perform symbol lock
 - Performs lane to lane de-skew
 - Compensates for frequency difference
- Analog Phy or Analog Front End (Analog FE)
 - Sends and receives symbols over the media
 - Recovers Rx clock from symbol stream
 - Performs bit lock
 - Detects peer existence and activity on lance



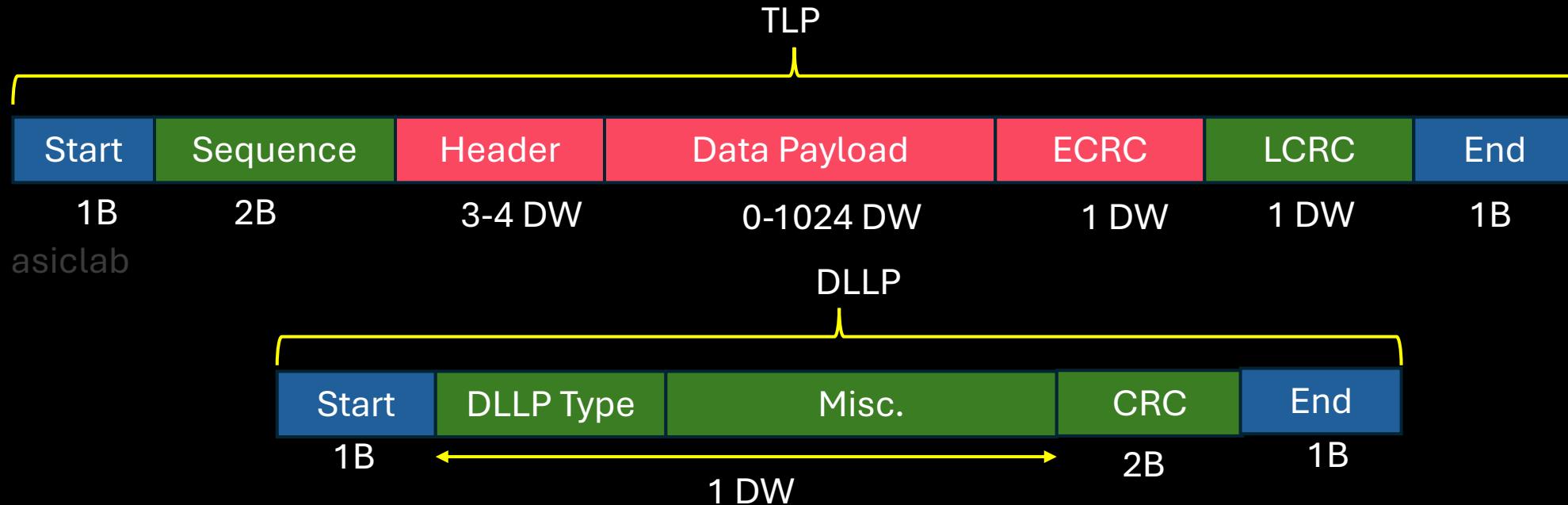
PCIe Device Layer Details 5.0 Non Flit Mode (NFM)



Do Not Distribute

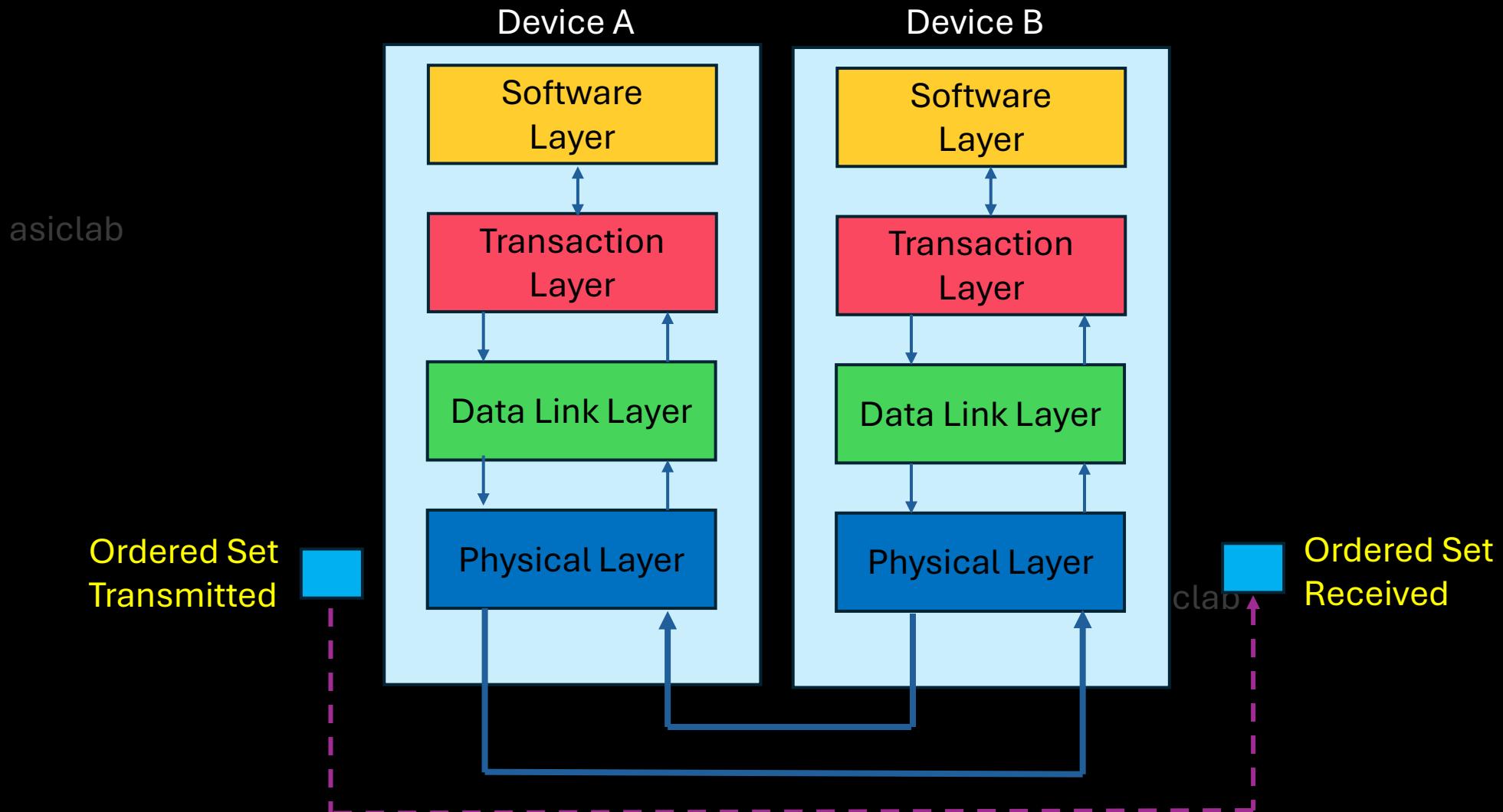
© Copyright protected 2024

TLP/DLLP at Physical Layer (8b/10b)



- Start Symbols
 - STP (Start TLP)
 - SDP (Start DLLP)
- End Symbols
 - END (end good for TLPs and end for DLLPs)
 - EDB (end bad for TLPs only)

Ordered Set Origin and Destination (8b/10b)

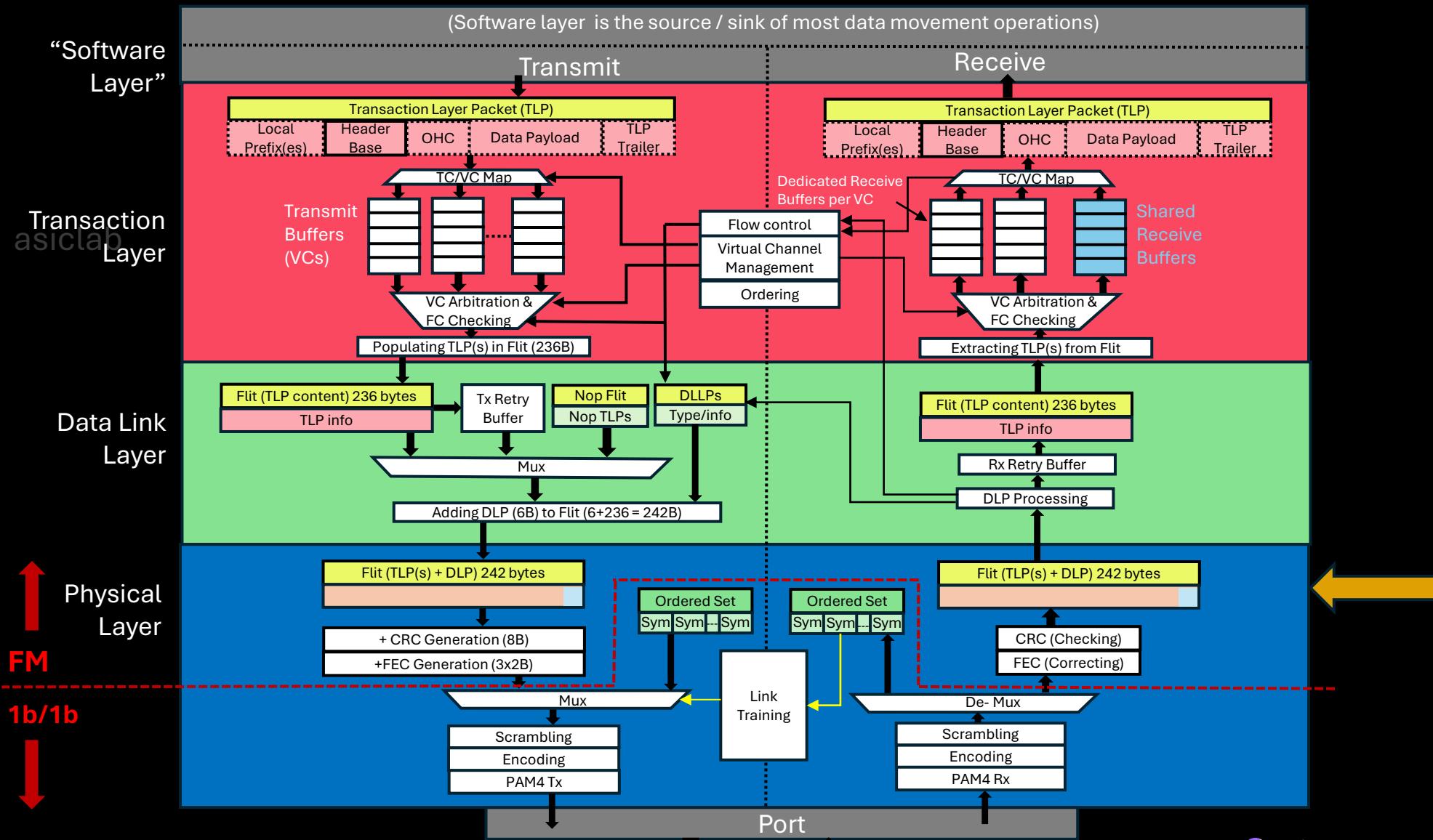


Ordered Set Structure

Symbol	TS1 Ordered set											
	7	6	5	4	3	2	1	0				
0	Start: COM (1Eh)											
1	Link Number											
2	Lane Number											
3	N_FTS											
4	Speed change	Auto Change deemphasis	32.0GT/s	16.0 GT/s	8.0 GT/s	5.0GT/s	2.5GT/s	Reserved				
5	Reserved		Compliance Receive	Disable Scrambling	Loopback	Disable Link	Hot Reset					
TS1 ID (D10.2)												
6	1 (EQ)	Transmitter Preset			Receiver Preset Hint							
	User preset	Transmitter Preset			Receiver Preset Hint							
TS1 ID (D10.2)												
7	Reserved	FS										
	Reserved	Pre-cursor coefficient										
TS1 ID (D10.2)												
8	Reserved	LF										
	Reserved	Cursor coefficient										
TS1 ID (D10.2)												
9	Parity	Reject Coefficients	Post-cursor coefficient									
10	TS1 ID (8'h4A)											
11	TS1 ID (8'h4A)											
12	TS1 ID (8'h4A)											
13	TS1 ID (8'h4A)											
14	TS1 ID (8'h4A) or DC balance symbol											
15	TS1 ID (8'h4A) or DC balance symbol											

Symbol	TS2 Ordered set																				
	7	6	5	4	3	2	1	0													
0	Start: COM K28.3 (2Dh)																				
1	Link Number																				
2	Lane Number																				
3	N_FTS																				
4	Speed change	Auto Change deemphasis	32.0GT/s	16.0 GT/s	8.0 GT/s	5.0GT/s	2.5GT/s	Reserved													
5	Reserved		Compliance Receive	Disable Scrambling	Loopback	Disable Link	Hot Reset														
TS2 ID (D5.2)																					
6	1 (EQ)	Transmitter Preset			Receiver Preset Hint																
	Request EQ	Quiesce Guarantee	Reserved																		
7	TS2 ID (8'h45)																				
8	TS2 ID (8'h45)																				
9	TS2 ID (8'h45)																				
10	TS2 ID (8'h45)																				
11	TS2 ID (8'h45)																				
12	TS2 ID (8'h45)																				
13	TS2 ID (8'h45)																				
14	TS2 ID (8'h45) or DC Balance symbol																				
15	TS2 ID (8'h45) or DC Balance symbol																				

PCIe Device Layer Details 6.0 (Flit Mode – FM) (1b/1b)

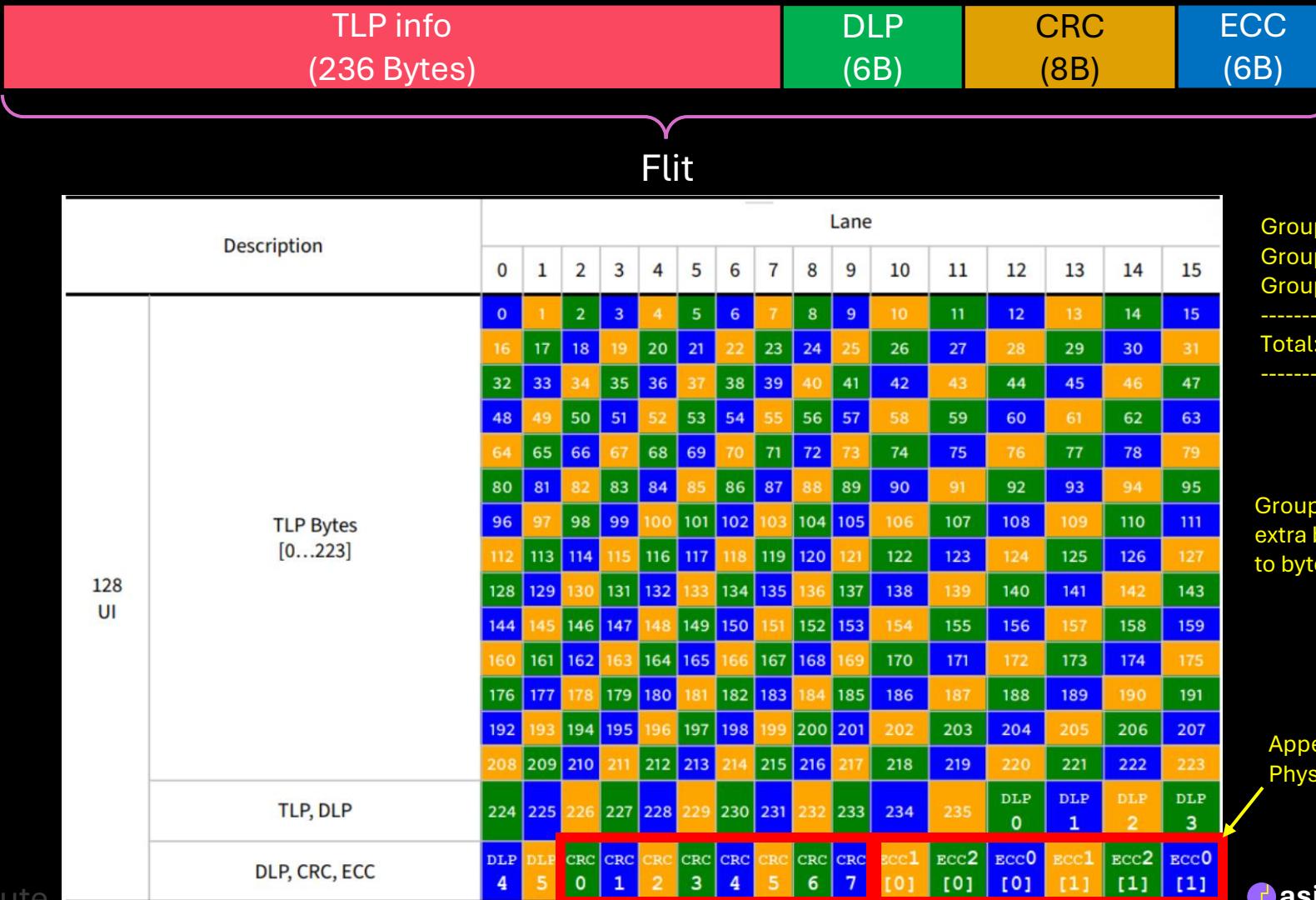


Do Not Distribute

© Copyright protected 2024

TLP/DLLP at Physical Layer (1b/1b)

asiclab



Do Not Distribute

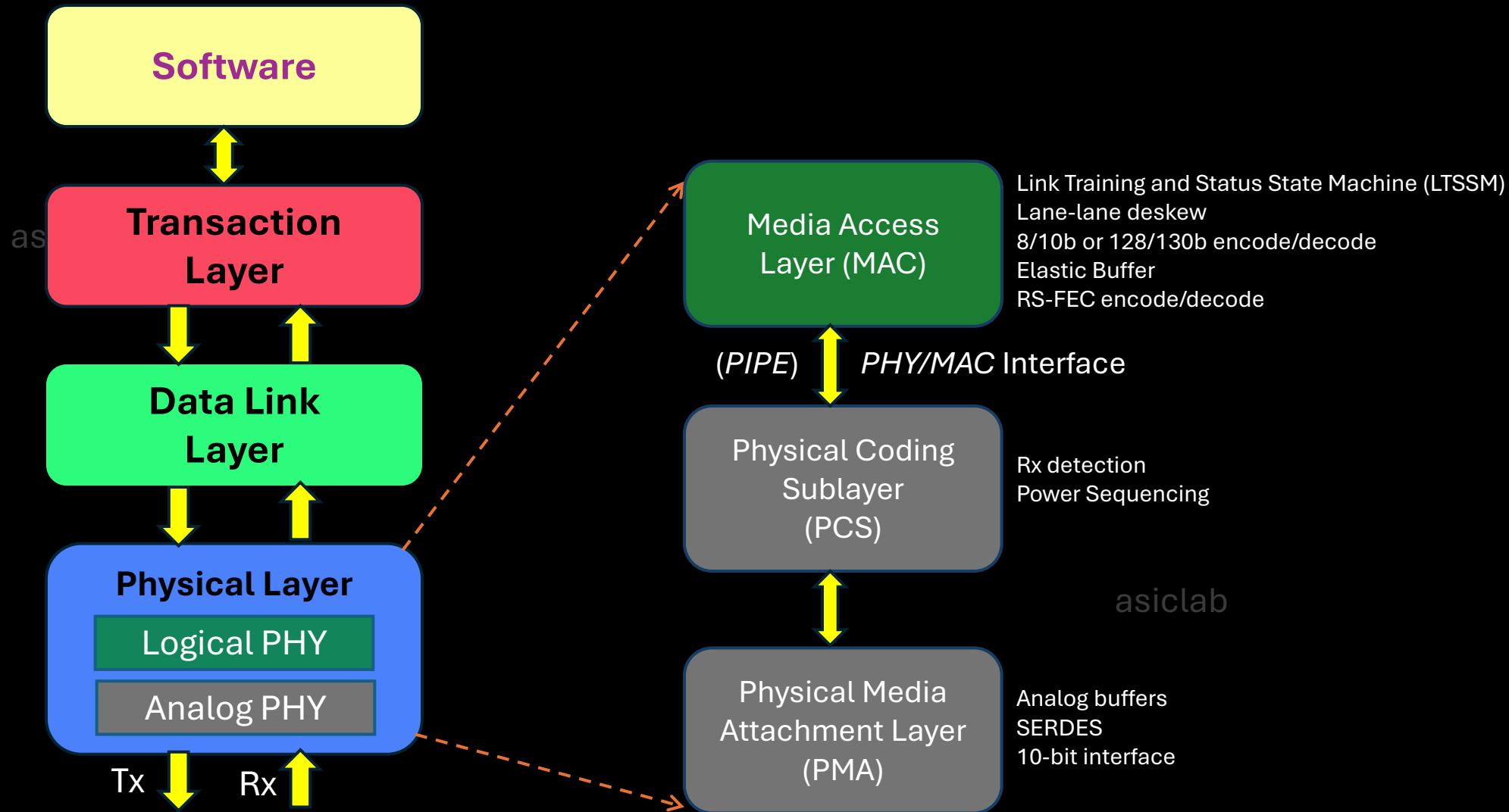
Group 1: 83B
Group 2: 83B
Group 3: 84B

Total: 250B

Groups 1 and 2 get an extra byte of 0 padded to byte 83

Appended by Physical layer

Physical Layer



Link Training and Initialization Tasks

- Detect receiver
- Bit lock per lane
- Symbol lock per lane
- Polarity Inversion
- Link numbering
- Link width and Lane numbering
- Lane reversal (optional)
- Lane-to-Lane de-skew on multi-Lane Links
- Link data rate determination and negotiation

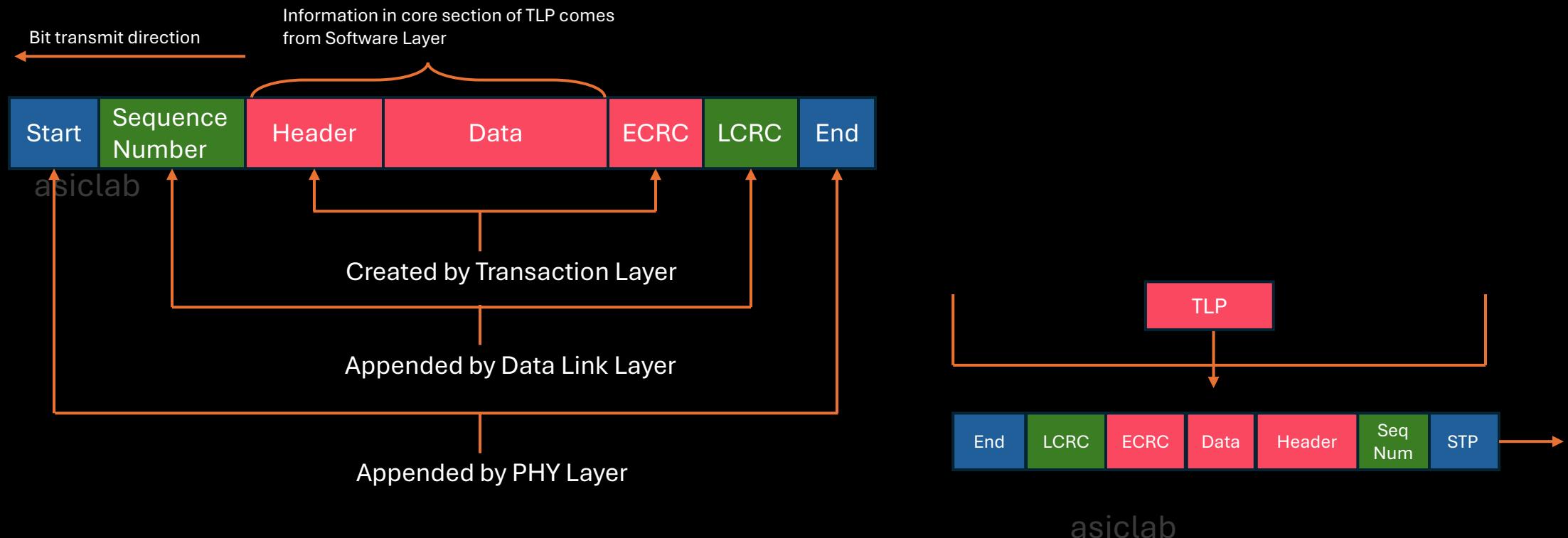
asiclab

Additional Physical Layer Features

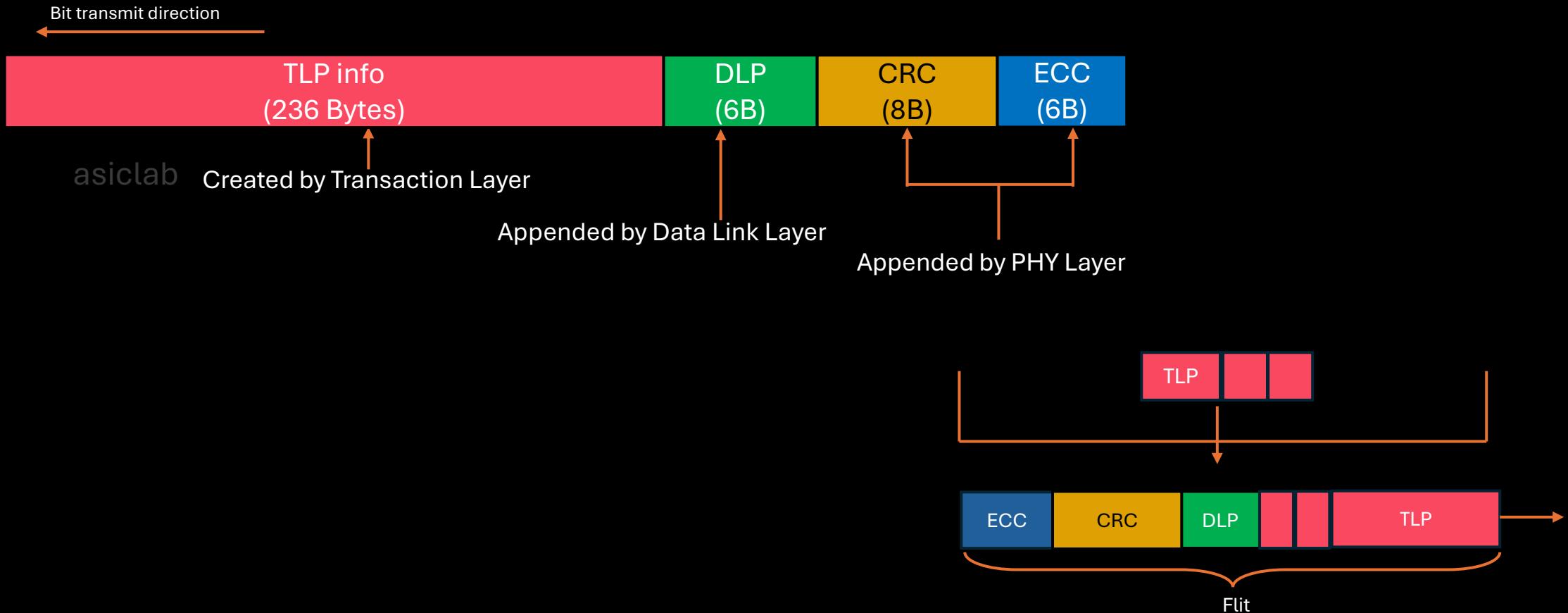
- Link Power management Logic
 - Power state: L0, L0s, L0p, L1, L2, L3
- ~~asynchronous~~ Reset Logic
 - Cold / warm reset
 - Hot reset
- Hot-plug control and status logic
 - Support not mandatory, but must comply with PCI hot-plug usage model if used

asiclab

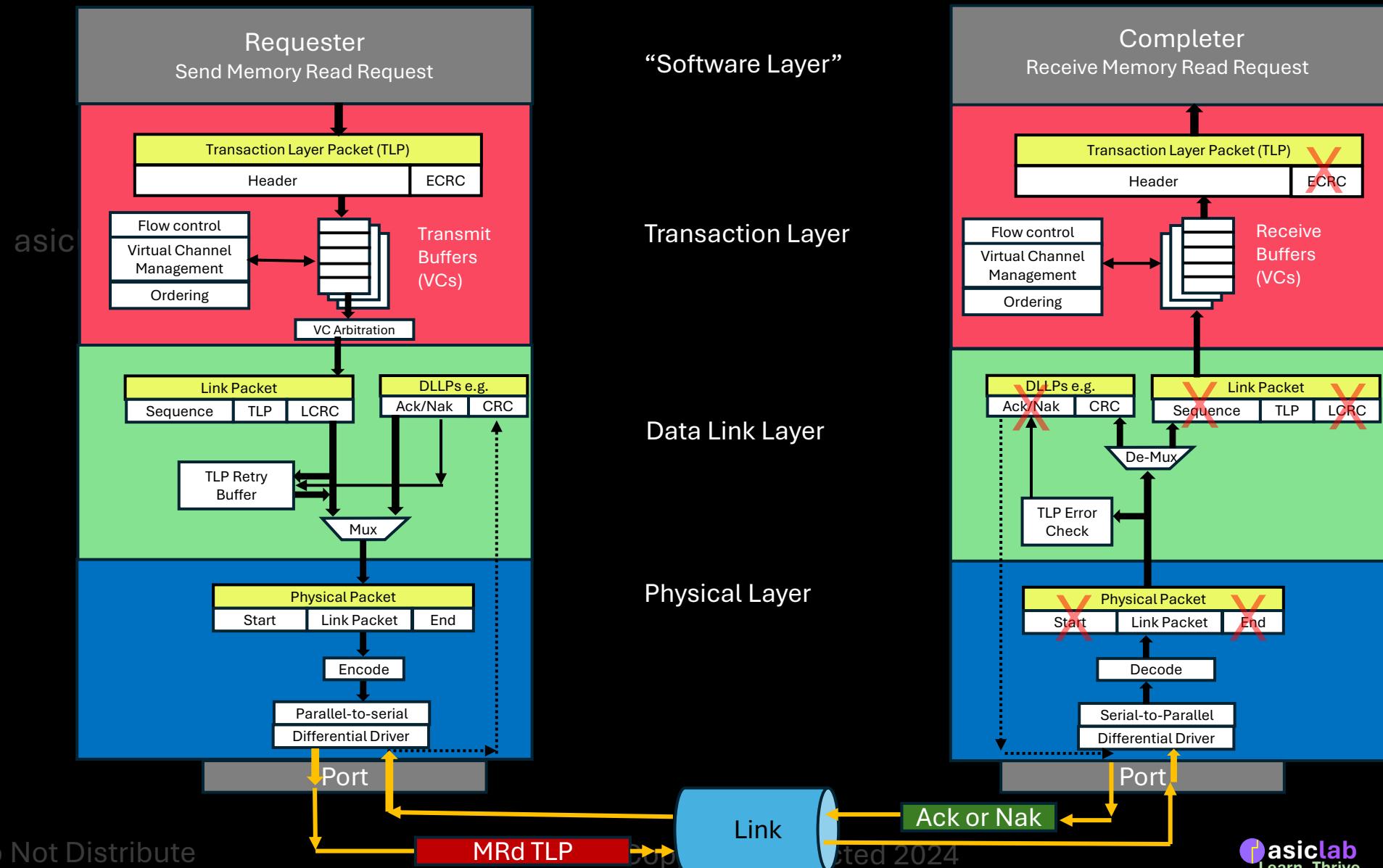
Packet format PCIe 5.0 (NFM)



Packet format PCIe 6.0 (FM)

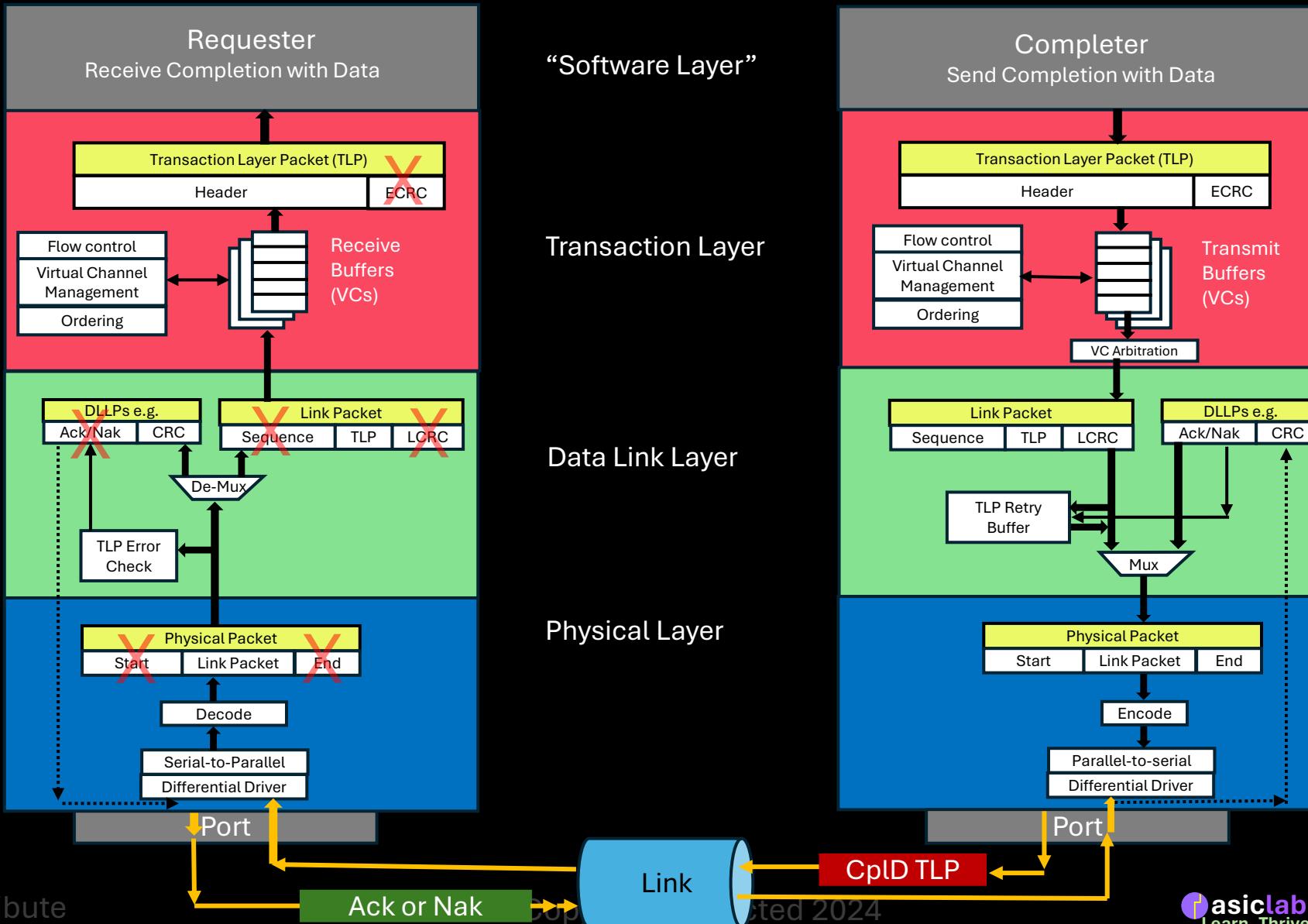


Example: Memory Read Request

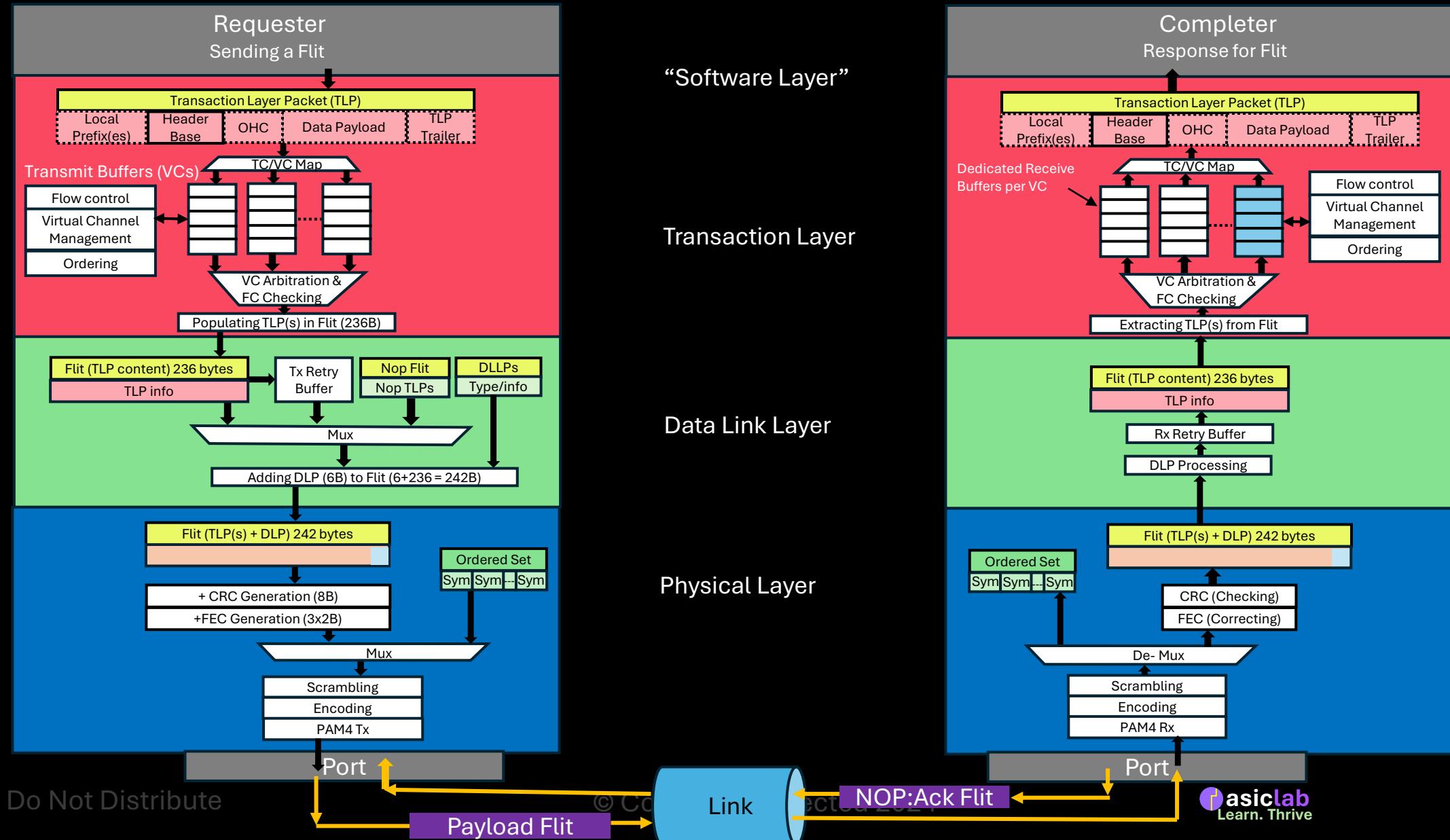


Associated Completion

asiclab



PCIe Device Layer Details 6.0 (Flit Mode – FM) (1b/1b)



PCIe Fabric Efficiency

- Factors that reduce efficiency:
 - 8b/10b overhead for Gen1 and Gen2 (20%)
 - 128b/130b overhead for Gen3 to Gen5 (1.56%)
 - Header (3-4DW) on all TLPs plus sequence number, ECRC, LCRC, Start and End Symbols
asiclab
 - Split transaction protocol overhead
 - DLLPs for Ack/Nak and Flow Control
- Factors that improve efficiency:
 - 1b/1b encoding
 - Using Flit transmission
 - Switch cut-through mode

asiclab

asiclab

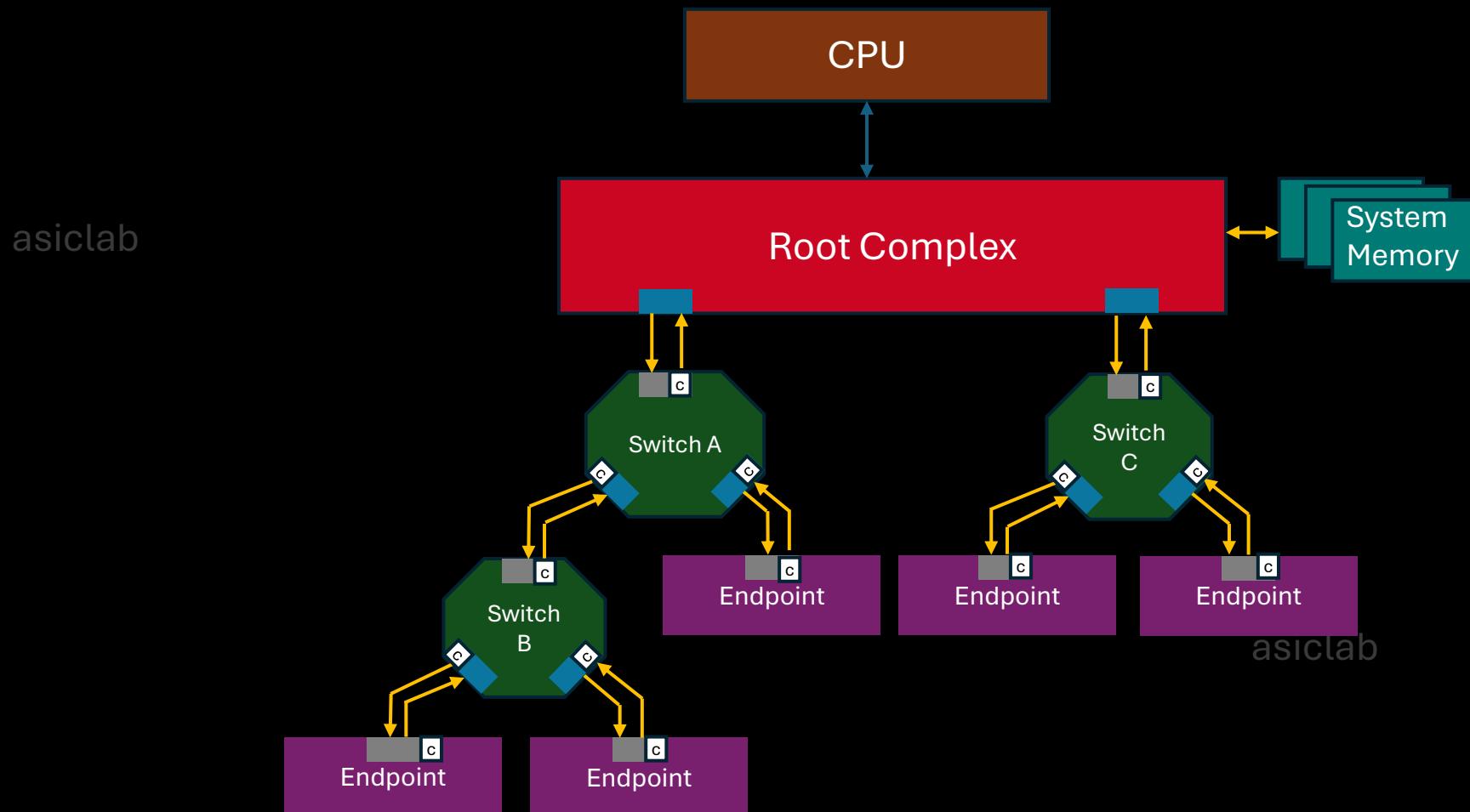
Configuration Overview

asiclab

What is PCIe Configuration Space?

- 1. Stores Device Info:** It's a special memory area where PCIe devices keep important information like their ID and settings.
- 2. Helps the System Identify Devices:** The operating system uses this space to find and set up devices connected to the computer.
- 3. Key Details Inside:**
 - Who made the device (Vendor ID)
 - What the device is (Device ID)
 - Where to access device memory (BARs)
- 4. Accessed by Address:** The system reads this space using a unique address (BDF) for each device.
- 5. Additional Features:** Some devices store extra capabilities like power saving or faster data handling here.
asiclab

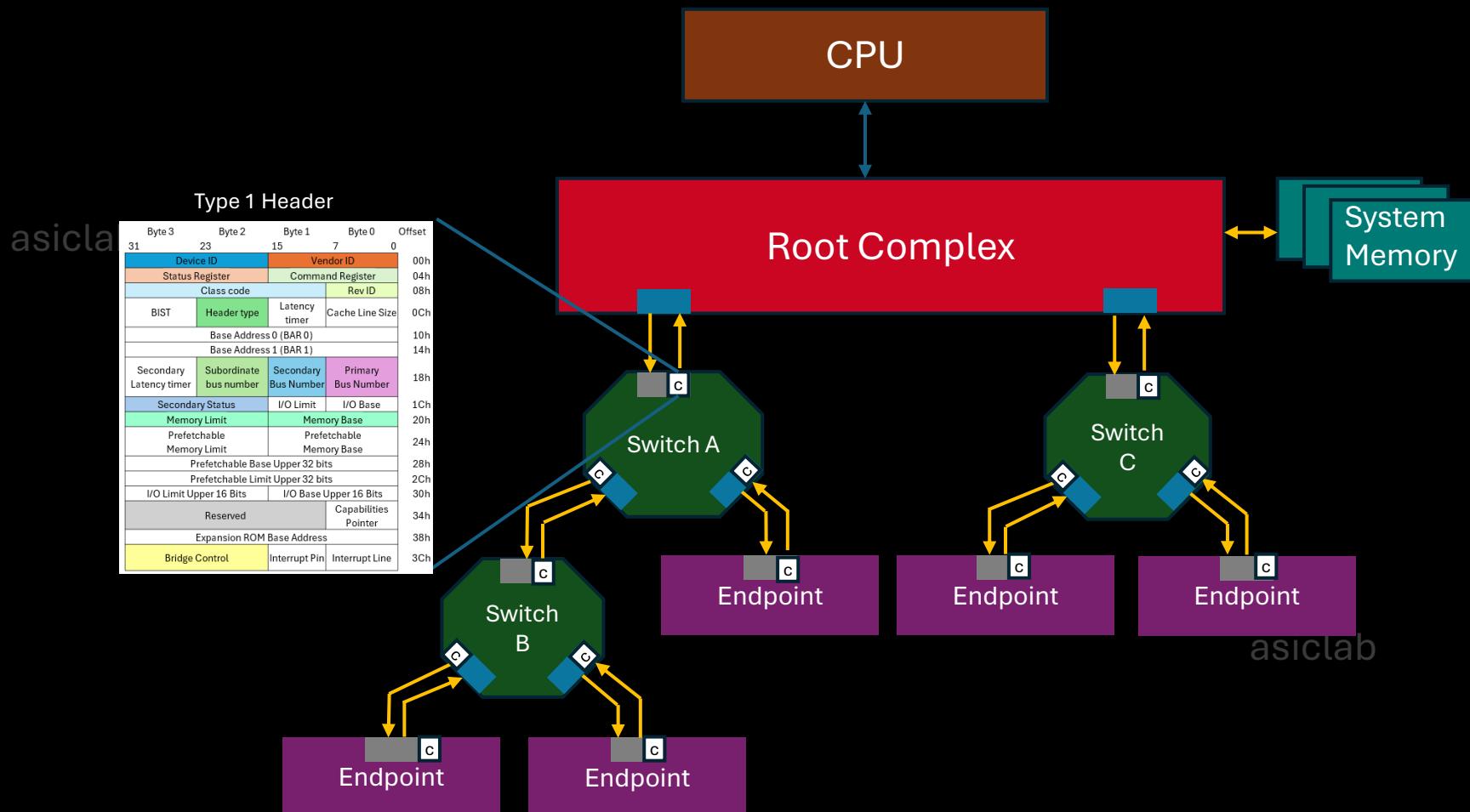
PCIe Configuration Space



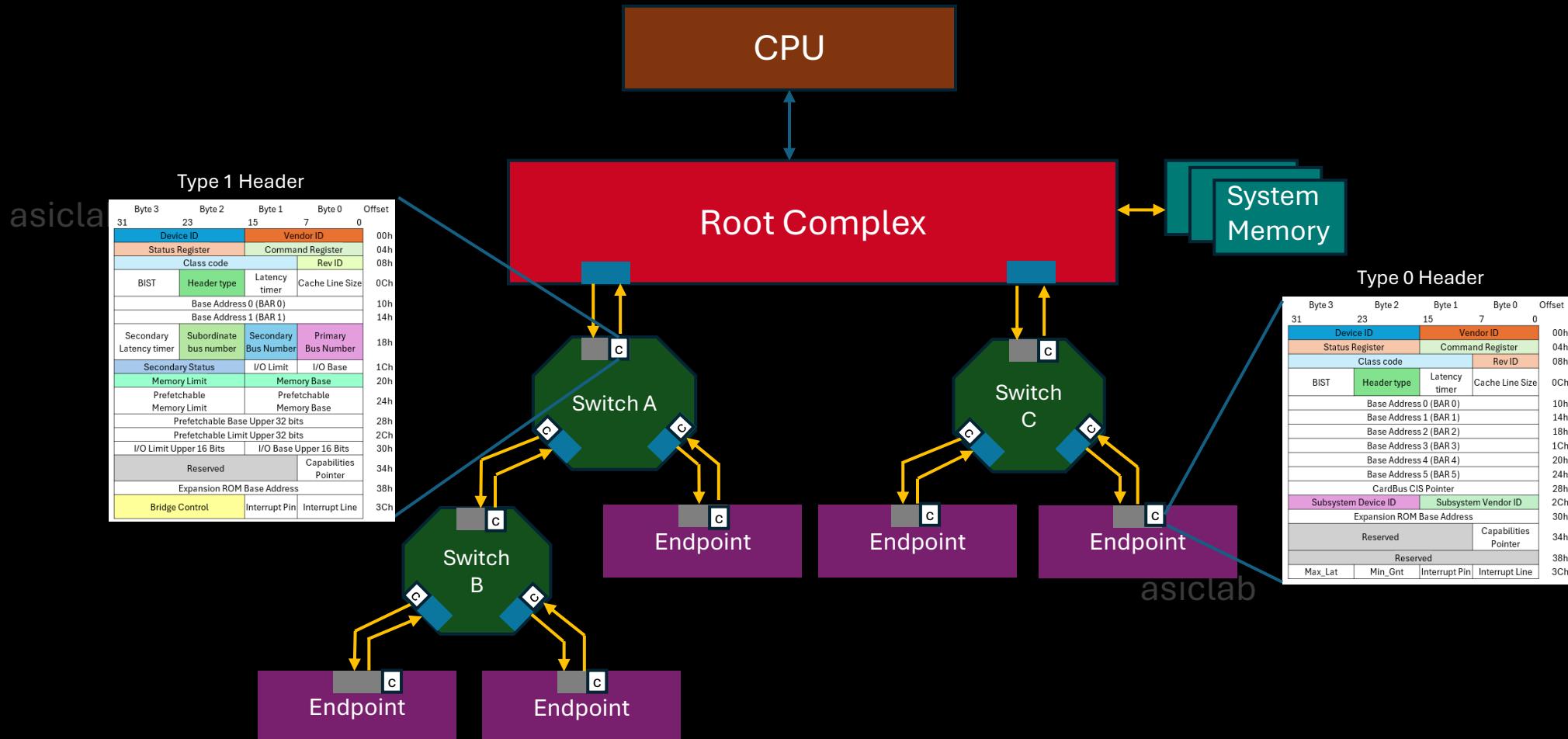
Prefetchable Memory: Can be read in advance for performance. Typically used for device memory that doesn't change often.

Do Not Distribute © Copyright protected 2024

PCIe Configuration Space



PCIe Configuration Space



Configuration Space – Type 0 Header

- Each function has one
- In this space the configuration registers are implemented
- On software execution, the RC initiates configurations transactions to a function's configuration registers
- 4KB space –
 - PCI compatible – 64 DWs
 - Header – First 16 DWs
 - PCI Express Extended configuration space 960 DWs

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
3	Device ID	2	1	Vendor ID 0
7	Status Register	6	C5	Command Register 4
B	Class Code	A	9	Re 8 ID
B	F	E	Header type	Latency timer
13	Base Address 0 (BAR 0)	12	11	Cache Line Size
	Base Address 1 (BAR 1)			10
	Base Address 2 (BAR 2)			14h
	Base Address 3 (BAR 3)			18h
	Base Address 4 (BAR 4)			1Ch
	Base Address 5 (BAR 5)			20h
	CardBus CIS Pointer			24h
	Subsystem Device ID		Subsystem Vendor ID	28h
	Expansion ROM Base Address			2Ch
	Reserved		Capabilities Pointer	30h
	Reserved			34h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	38h
				3Ch

<https://ctf.re/windows/kernel/pcie/tutorial/2023/02/14/pcie-part-1/>

Configuration Space – Type 0 Header

- Each function has one
- In this space the configuration registers are implemented
- On software execution, the RC initiates configurations transactions to a function's configuration registers
- 4KB space –
 - PCI compatible – 64 DWs
 - Header – First 16 DWs
 - PCI Express Extended configuration space 960 DWs

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

<https://ctf.re/windows/kernel/pcie/tutorial/2023/02/14/pcie-part-1/>

Configuration Space – Type 1 Header

- Layout of a PCI-to-PCI bridge's configuration header space

asiclab

Byte 3 31	Byte 2 23	Byte 1 15	Byte 0 7	Offset 0		
Device ID		Vendor ID		00h		
Status Register		Command Register		04h		
Class code		Rev ID		08h		
BIST	Header type	Latency timer	Cache Line Size	0Ch		
Base Address 0 (BAR 0)				10h		
Base Address 1 (BAR 1)				14h		
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	18h		
Secondary Status		I/O Limit	I/O Base	1Ch		
Memory Limit		Memory Base		20h		
Prefetchable Memory Limit		Prefetchable Memory Base		24h		
Prefetchable Base Upper 32 bits				28h		
Prefetchable Limit Upper 32 bits				2Ch		
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30h		
Reserved			Capabilities Pointer	34h		
Expansion ROM Base Address				38h		
Bridge Control		Interrupt Pin	Interrupt Line	3Ch		

Configuration Space – Type 0 Header

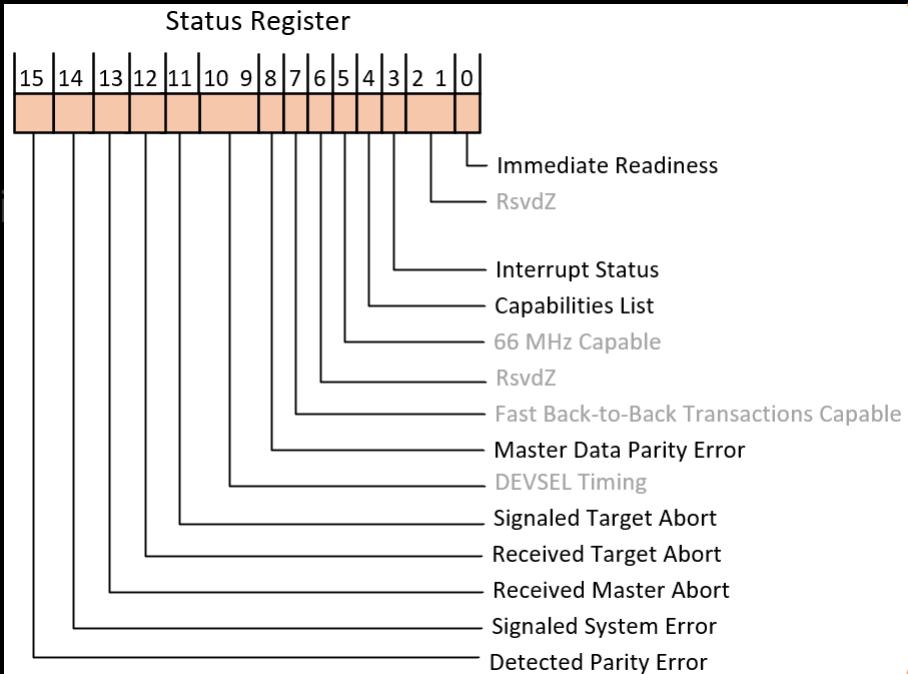
asiclab

Device ID	Vendor ID	Device Description
0x2922	0x8086	Intel SATA Controller (ICH9)
0x1DB6	0x10DE	NVIDIA GPU (GeForce RTX series)
0x43D5	0x1002	AMD Radeon RX 5700 XT
0x145F	0x1022	AMD Ryzen Processor Root Complex
0x0011	0x1AF4	Red Hat Virtio Network Device
0x005D	0x1D0F	Amazon Elastic Network Adapter (Nitro)
0xA808	0x144D	Samsung NVMe SSD Controller
0x8231	0x104C	Texas Instruments PCI-to-PCI Bridge
0x8168	0x10EC	Realtek PCIe Gigabit Ethernet Adapter
0x16F3	0x14E4	Broadcom NetXtreme Ethernet Controller

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
Class code			Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Base Address 2 (BAR 2)				18h
Base Address 3 (BAR 3)				1Ch
Base Address 4 (BAR 4)				20h
Base Address 5 (BAR 5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

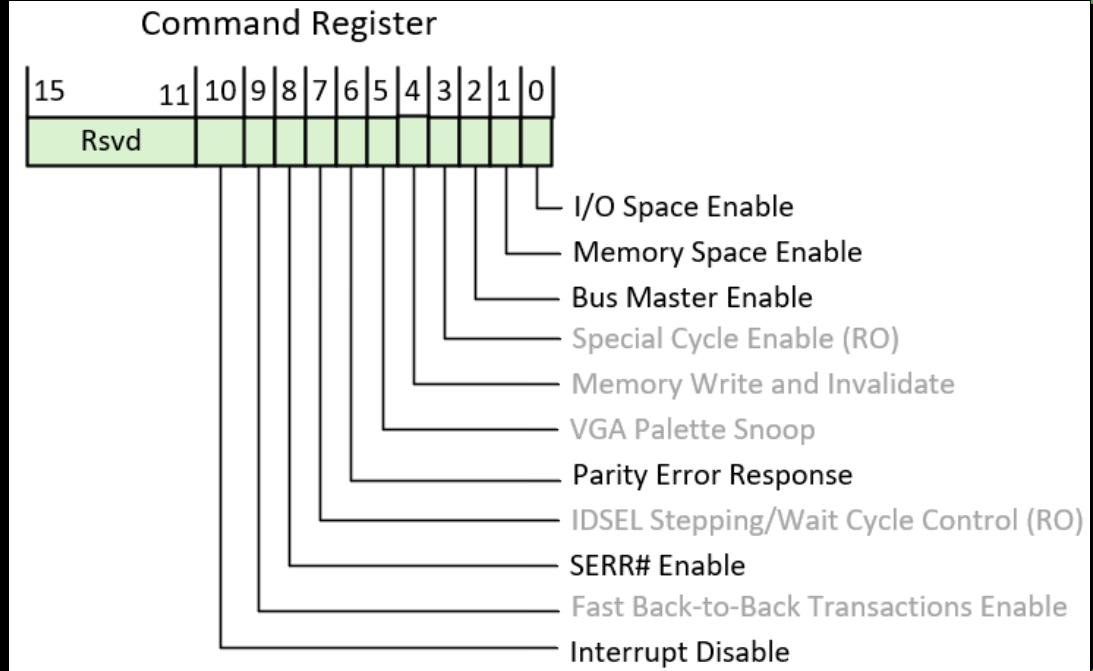
Configuration Space – Type 0 Header

as



Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
		Device ID	Vendor ID	00h
		Status Register	Command Register	04h
		Class code	Rev ID	08h
	BIST	Header type	Latency timer	0Ch
			Cache Line Size	
			Base Address 0 (BAR 0)	10h
			Base Address 1 (BAR 1)	14h
			Base Address 2 (BAR 2)	18h
			Base Address 3 (BAR 3)	1Ch
			Base Address 4 (BAR 4)	20h
			Base Address 5 (BAR 5)	24h
			CardBus CIS Pointer	28h
		Subsystem Device ID	Subsystem Vendor ID	2Ch
			Expansion ROM Base Address	30h
			Reserved	34h
			Capabilities Pointer	
			Reserved	38h
	Max_Lat	Min_Gnt	Interrupt Pin	3Ch
			Interrupt Line	

Configuration Space – Type 0 Header



Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
	Class code		Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
			Base Address 0 (BAR 0)	10h
			Base Address 1 (BAR 1)	14h
			Base Address 2 (BAR 2)	18h
			Base Address 3 (BAR 3)	1Ch
			Base Address 4 (BAR 4)	20h
			Base Address 5 (BAR 5)	24h
			CardBus CIS Pointer	28h
Subsystem Device ID		Subsystem Vendor ID		2Ch
		Expansion ROM Base Address		30h
		Reserved	Capabilities Pointer	34h
			Reserved	38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Configuration Space – Type 0 Header

asiclab

23 16

Base Class Code (RO)

- 00h = Device was built before Class Code definitions were finalized
- 01h = Mass Storage Controller
- 02h = Network Controller
- 03h = Display Controller
- 04h = Multimedia Device
- 05h = Memory Controller
- 06h = Bridge Device
- 07h = Simple Communication Controller
- 08h = Base System Peripheral
- 09h = Input Device
- 0Ah = Docking Station
- 0Bh = Processor
- 0Ch = Serial Bus Controller
- 0Dh = Wireless Controller
- 0Eh = Intelligent IO Controller
- 0Fh = Satellite Communication Controller
- 10h = Encryption/Decryption Controller
- 11h = Data Acquisition and Signal Processing Controller
- FFh = Device does not fit any defined class

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
Class code		Rev ID		08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Base Address 2 (BAR 2)				18h
Base Address 3 (BAR 3)				1Ch
Base Address 4 (BAR 4)				20h
Base Address 5 (BAR 5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved		Capabilities Pointer		34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Configuration Space – Type 0 Header

Base Class code = 0Dh (Wireless Controller)



Base Class code = 01h (Mass Storage Controller)



Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Base Address 2 (BAR 2)				18h
Base Address 3 (BAR 3)				1Ch
Base Address 4 (BAR 4)				20h
Base Address 5 (BAR 5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				2Ch
Reserved			Capabilities Pointer	30h
Reserved				34h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	38h
				3Ch

Configuration Space – Type 0 Header

Base Class code = 01h (Mass Storage Controller)



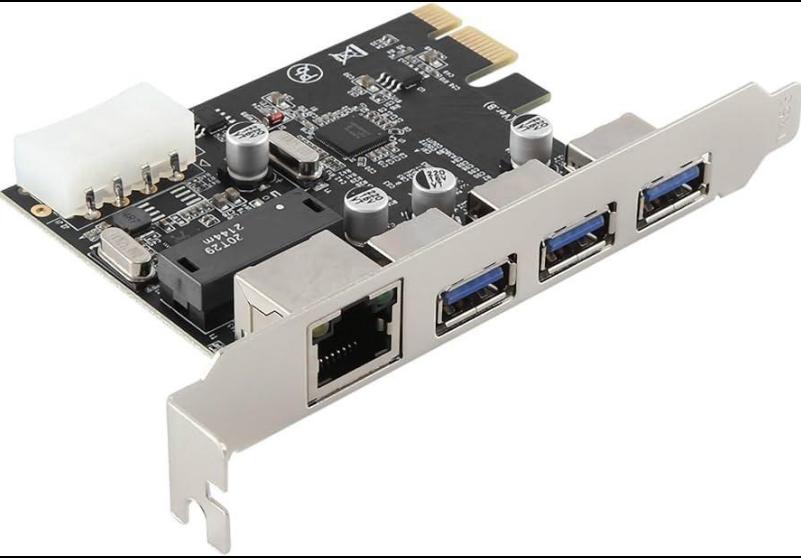
Base Class code = 01h (Mass Storage Controller)



Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code				Rev ID
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Base Address 2 (BAR 2)				18h
Base Address 3 (BAR 3)				1Ch
Base Address 4 (BAR 4)				20h
Base Address 5 (BAR 5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				34h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	38h
				3Ch

Configuration Space – Type 0 Header

Base Class code = 02h (Network Controller)



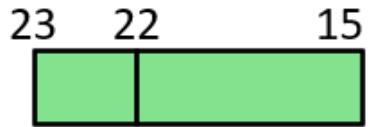
Base Class code = 03h (Display Controller)



Byte 3	Byte 2	Byte 1	Byte 0	Offset		
31	23	15	7	0		
Device ID			Vendor ID			
Status Register			Command Register			
Class code			Rev ID			
BIST	Header type	Latency timer	Cache Line Size			
Base Address 0 (BAR 0)						
Base Address 1 (BAR 1)						
Base Address 2 (BAR 2)						
Base Address 3 (BAR 3)						
Base Address 4 (BAR 4)						
Base Address 5 (BAR 5)						
CardBus CIS Pointer						
Subsystem Device ID			Subsystem Vendor ID			
Expansion ROM Base Address						
Reserved			Capabilities Pointer			
Reserved						
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch		

Configuration Space – Type 0 Header

asiclab



22:15 – Header Type (RO)

0000000b = Non-bridge function

0000001b = PCI-to-PCI Bridge

0000010b = CardBus Bridge

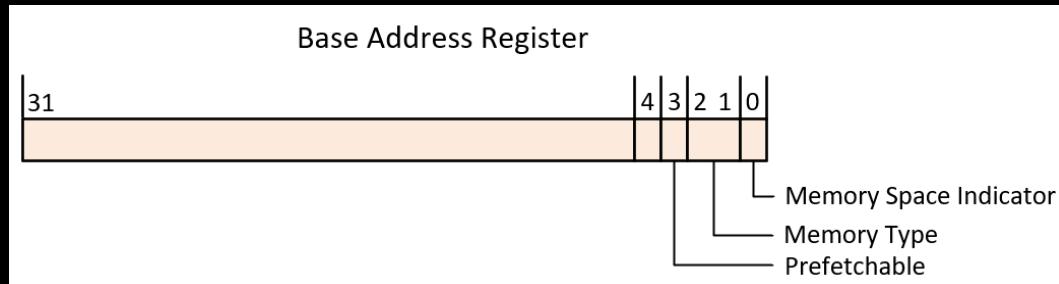
23 – Multi-Function Device (RO)

0b = NOT a multi-function Device

1b = Multi-function device

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
	Class code		Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
		Base Address 0 (BAR 0)		10h
		Base Address 1 (BAR 1)		14h
		Base Address 2 (BAR 2)		18h
		Base Address 3 (BAR 3)		1Ch
		Base Address 4 (BAR 4)		20h
		Base Address 5 (BAR 5)		24h
		CardBus CIS Pointer		28h
Subsystem Device ID		Subsystem Vendor ID		2Ch
		Expansion ROM Base Address		30h
	Reserved		Capabilities Pointer	34h
	Reserved			38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Configuration Space – Type 0 Header



Memory Space Indicator

0 – Memory Space

1 – I/O

Memory Type [2:1]

00 – Base register is 32 bits wide

01 – Reserved

10 – Base register is 64 bits wide

Prefetchable – 1

Non-prefetchable – 0

I/O – Reserved/0

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code		Rev ID		
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

Configuration Space – Type 0 Header

asiclab

1. Understand the Size Requirement
2. Allocate System Memory
3. Write the System memory start address allocated

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code		Rev ID		
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

Configuration Space – Device Specification

https://www.nvidia.com/content/dam/en-zz/Solutions/gtcs22/data-center/h100/PB-11133-001_v01.pdf

Table 1. Product Specifications

Specification	NVIDIA H100
Product SKU	P1010 SKU 200 NVPN: 699-21010-0200-xxx
Total board power	PCIe 16-pin 450 W or 600 W power mode: <ul style="list-style-type: none">• 350 W default• 350 W maximum• 200 W minimum PCIe 16-pin 300 W power mode: <ul style="list-style-type: none">• 310 W default• 310 W maximum• 200 W minimum
Thermal solution	Passive
Mechanical form factor	Full-height, full-length (FHFL) 10.5", dual-slot
GPU SKU	GH100-200
PCI Device IDs	Device ID: 0x2331 Vendor ID: 0x10DE Sub-Vendor ID: 0x10DE Sub-System ID: 0x1626
GPU clocks	Base: 1,125 MHz Boost: 1,755 MHz
Performance states	P0
VBIOS	EEPROM size: 8 Mbit UEFI: Supported

Specification	NVIDIA H100
PCI Express interface	PCI Express Gen5 x16; Gen5 x8; Gen4 x16 Lane and polarity reversal supported
Multi-Instance GPU (MIG)	Supported (seven instances)
Secure Boot (CEC)	Supported
Zero Power	Not supported
Power connectors and headers	One PCIe 16-pin auxiliary power connector
Weight	Board: 1200g grams (excluding bracket, extenders, and bridges) NVLink bridge: 20.5 grams per bridge (x 3 bridges) Bracket with screws: 20 grams Enhanced straight extender: 35 grams Long offset extender: 48 grams Straight extender: 32 grams

Table 3. Software Specifications

Specification	Description ¹
SR-IOV support	Supported -- 32 VF (virtual functions)
BAR address (physical function)	BAR0: 16 MiB ¹ BAR1: 128 GiB ¹ BAR3: 32 MiB ¹
BAR address (virtual function)	BAR0: 5 MiB, [256 KiB per VF] ¹ BAR1: 80 GiB, 64-bit [4 GiB per VF] ¹ BAR3: 640 MiB, 64-bit [32 MiB per VF] ¹
Message signaled interrupts	MSI-X: Supported MSI: Not supported
ARI Forwarding	Supported
Driver support	Linux: R520 or later Windows: R520 or later

1. Understand the Size Requirement
2. Allocate System Memory
3. Write the System memory start address allocated

GPU BAR Example:
BAR0: MMIO Registers
BAR1: VRAM aperture
BAR3: Indirect memory access

Memory Modules

https://www.nvidia.com/content/dam/en-zz/Solutions/gtcs22/data-center/h100/PB-11133-001_v01.pdf

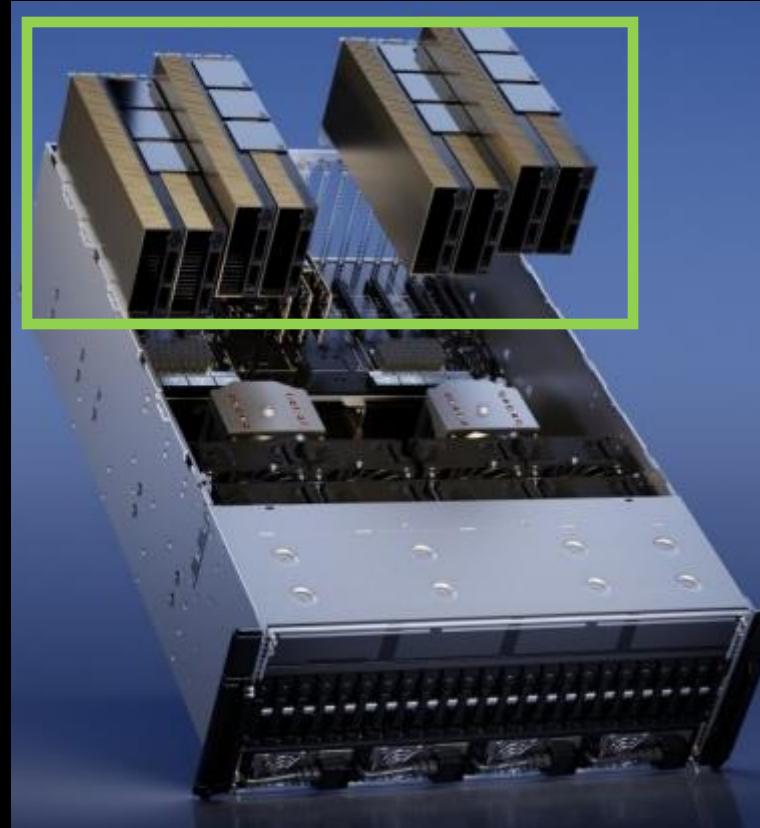
asiclab



Multiple Endpoints Example

https://www.nvidia.com/content/dam/en-zz/Solutions/gtcs22/data-center/h100/PB-11133-001_v01.pdf

asiclab



asiclab

Do Not Distribute

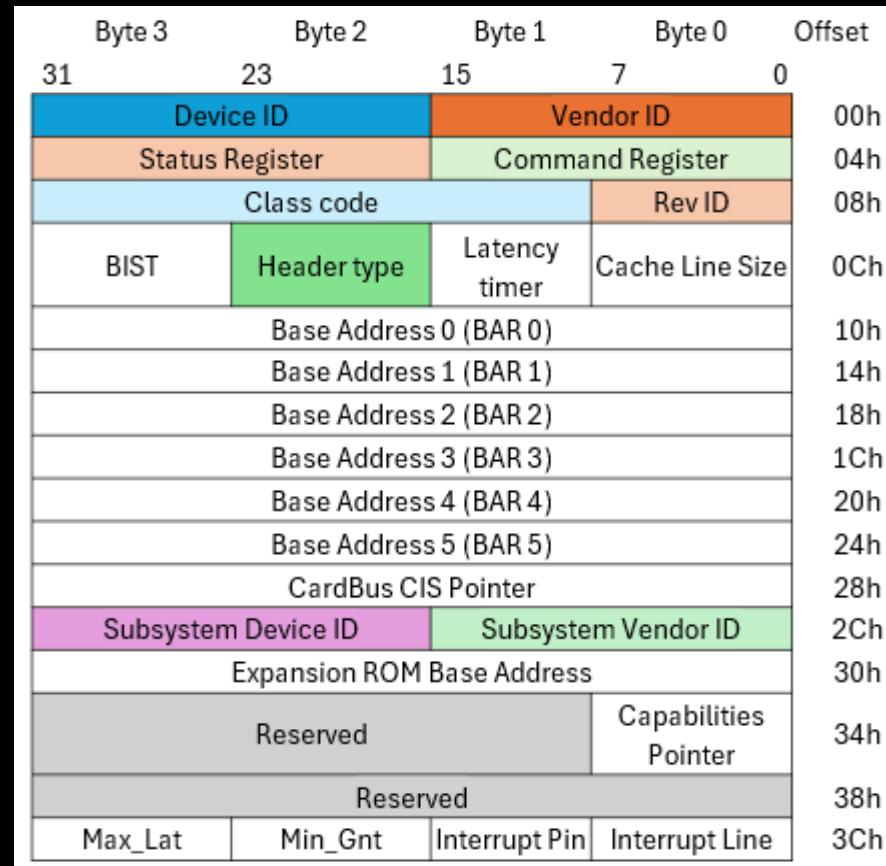
© Copyright protected 2024

asiclab
Learn. Thrive

Configuration Space – Type 0 Header

1. Understand the Size Requirement

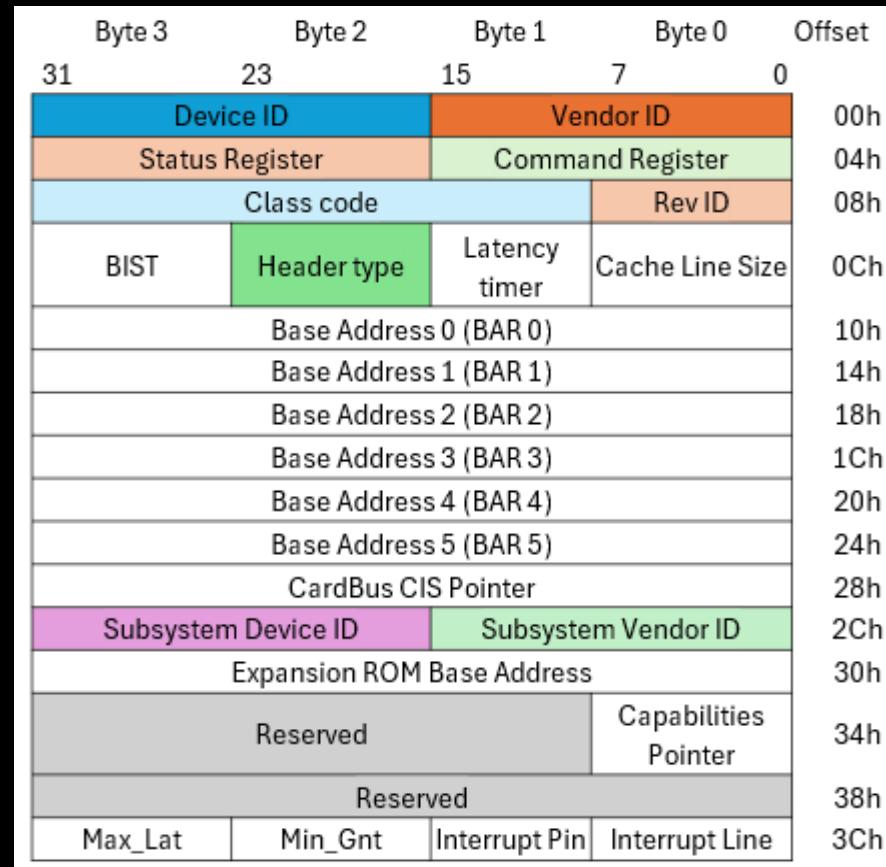
Bit Index	Hex (32 bit)	Binary (32 bit)	Decimal	Bytes
0	0x00000001	00000000000000000000000000000001	1	1 byte
1	0x00000002	00000000000000000000000000000010	2	2 bytes
2	0x00000004	000000000000000000000000000000100	4	4 bytes
3	0x00000008	0000000000000000000000000000001000	8	8 bytes
4	0x00000010	00000000000000000000000000000010000	16	16 bytes
5	0x00000020	000000000000000000000000000000100000	32	32 bytes
6	0x00000040	0000000000000000000000000000001000000	64	64 bytes
7	0x00000080	00000000000000000000000000000010000000	128	128 bytes
8	0x00000100	000000000000000000000000000000100000000	256	256 bytes
9	0x00000200	000000000000000000000000000000100000000	512	512 bytes
10	0x00000400	0000000000000000000000000000001000000000	1,024	1 KB (Kilobyte)
11	0x00000800	00000000000000000000000000000010000000000	2,048	2 KB
12	0x00001000	00000000000000000000000000000010000000000	4,096	4 KB
13	0x00002000	00000000000000000000000000000010000000000	8,192	8 KB
14	0x00004000	00000000000000000000000000000010000000000	16,384	16 KB
15	0x00008000	00000000000000000000000000000010000000000	32,768	32 KB
16	0x00010000	00000000000000000000000000000010000000000	65,536	64 KB
17	0x00020000	00000000000000000000000000000010000000000	1,31,072	128 KB
18	0x00040000	00000000000000000000000000000010000000000	2,62,144	256 KB
19	0x00080000	00000000000000000000000000000010000000000	5,24,288	512 KB
20	0x00100000	00000000000000000000000000000010000000000	10,48,576	1 MB (Megabyte)
21	0x00200000	00000000000000000000000000000010000000000	20,97,152	2 MB
22	0x00400000	00000000000000000000000000000010000000000	41,94,304	4 MB
23	0x00800000	00000000000000000000000000000010000000000	83,88,608	8 MB
24	0x01000000	00000000000000000000000000000010000000000	1,67,77,216	16 MB
25	0x02000000	00000000000000000000000000000010000000000	3,35,54,432	32 MB
26	0x04000000	00000000000000000000000000000010000000000	6,71,08,864	64 MB
27	0x08000000	000010000000000000000000000000000000000000	13,42,17,728	128 MB
28	0x10000000	000100000000000000000000000000000000000000	26,84,35,456	256 MB
29	0x20000000	001000000000000000000000000000000000000000	53,68,70,912	512 MB
30	0x40000000	0100	1,07,37,41,824	1 GB (Gigabyte)
31	0x80000000	1000	2,14,74,83,648	2 GB
32	0xFFFFFFF	11	4,29,49,67,295	4 GB



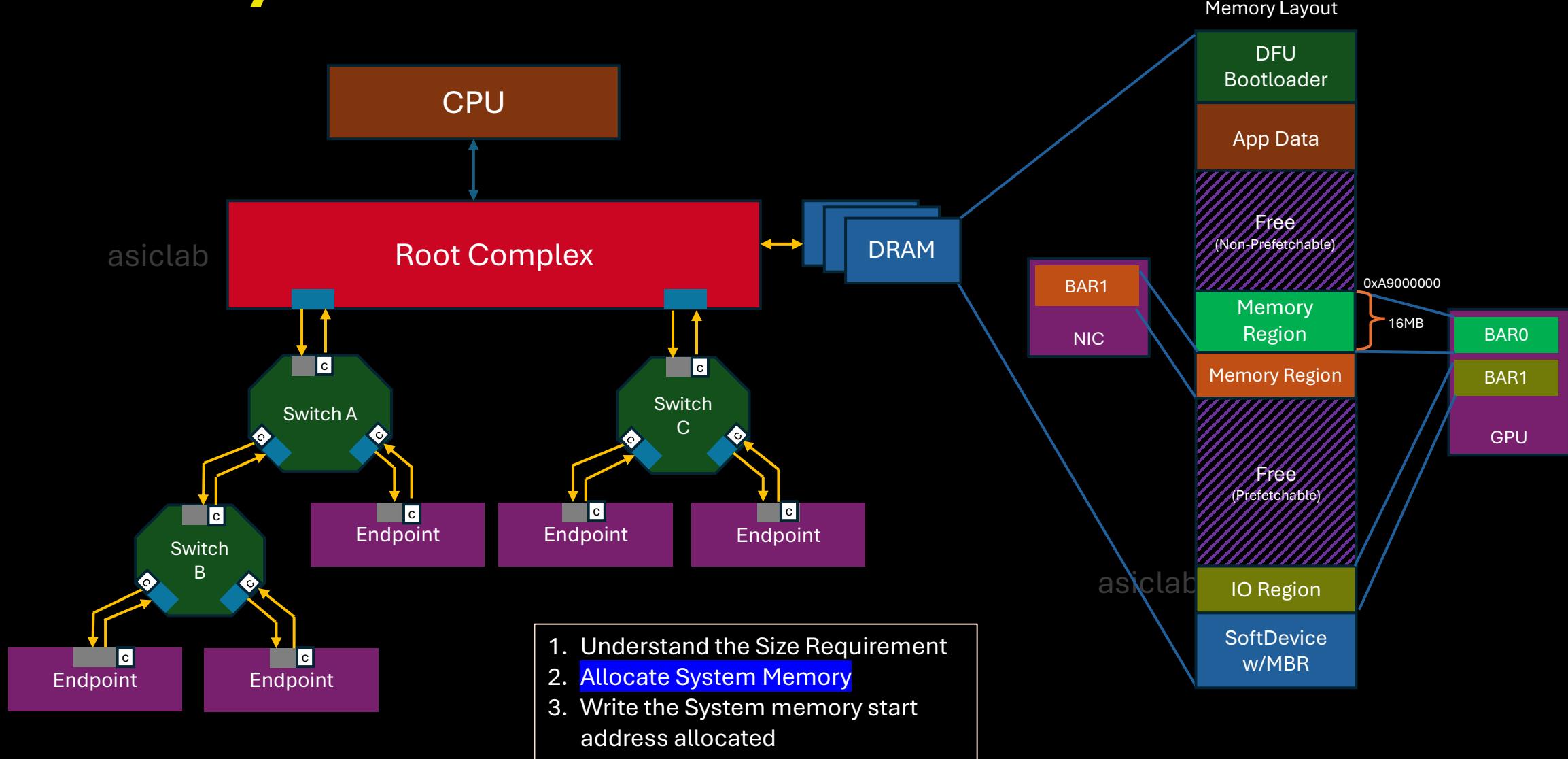
Configuration Space – Type 0 Header

1. Understand the Size Requirement

Bit Index	Hex (32 bit)	Binary (32 bit)	Decimal	Bytes
0	0x00000001	00000000000000000000000000000001	1	1 byte
1	0x00000002	00000000000000000000000000000010	2	2 bytes
2	0x00000004	000000000000000000000000000000100	4	4 bytes
3	0x00000008	0000000000000000000000000000001000	8	8 bytes
4	0x00000010	00000000000000000000000000000010000	16	16 bytes
5	0x00000020	000000000000000000000000000000100000	32	32 bytes
6	0x00000040	0000000000000000000000000000001000000	64	64 bytes
7	0x00000080	00000000000000000000000000000010000000	128	128 bytes
8	0x00000100	000000000000000000000000000000100000000	256	256 bytes
9	0x00000200	000000000000000000000000000000100000000	512	512 bytes
10	0x00000400	0000000000000000000000000000001000000000	1,024	1 KB (Kilobyte)
11	0x00000800	00000000000000000000000000000010000000000	2,048	2 KB
12	0x00001000	00000000000000000000000000000010000000000	4,096	4 KB
13	0x00002000	00000000000000000000000000000010000000000	8,192	8 KB
14	0x00004000	00000000000000000000000000000010000000000	16,384	16 KB
15	0x00008000	00000000000000000000000000000010000000000	32,768	32 KB
16	0x00010000	00000000000000000000000000000010000000000	65,536	64 KB
17	0x00020000	00000000000000000000000000000010000000000	1,31,072	128 KB
18	0x00040000	00000000000000000000000000000010000000000	2,62,144	256 KB
19	0x00080000	00000000000000000000000000000010000000000	5,24,288	512 KB
20	0x00100000	00000000000000000000000000000010000000000	10,48,576	1 MB (Megabyte)
21	0x00200000	00000000000000000000000000000010000000000	20,97,152	2 MB
22	0x00400000	00000000000000000000000000000010000000000	41,94,304	4 MB
23	0x00800000	00000000000000000000000000000010000000000	83,88,608	8 MB
24	0x01000000	00000000000000000000000000000010000000000	1,67,77,216	16 MB
25	0x02000000	00000000000000000000000000000010000000000	3,35,54,432	32 MB
26	0x04000000	00000000000000000000000000000010000000000	6,71,08,864	64 MB
27	0x08000000	000010000000000000000000000000000000000000	13,42,17,728	128 MB
28	0x10000000	000100000000000000000000000000000000000000	26,84,35,456	256 MB
29	0x20000000	001000000000000000000000000000000000000000	53,68,70,912	512 MB
30	0x40000000	0100	1,07,37,41,824	1 GB (Gigabyte)
31	0x80000000	1000	2,14,74,83,648	2 GB
32	0xFFFFFFF	11	4,29,49,67,295	4 GB



Memory Allocation



Prefetchable Memory: Can be read in advance for performance. Typically used for device memory that doesn't change often.

Non-Prefetchable Memory: Must be read on demand, often used for I/O where data may change frequently. Risk of data loss.

Configuration Space – Type 0 Header

1. Understand the Size Requirement
2. Allocate System Memory
3. Write the System memory start address allocated

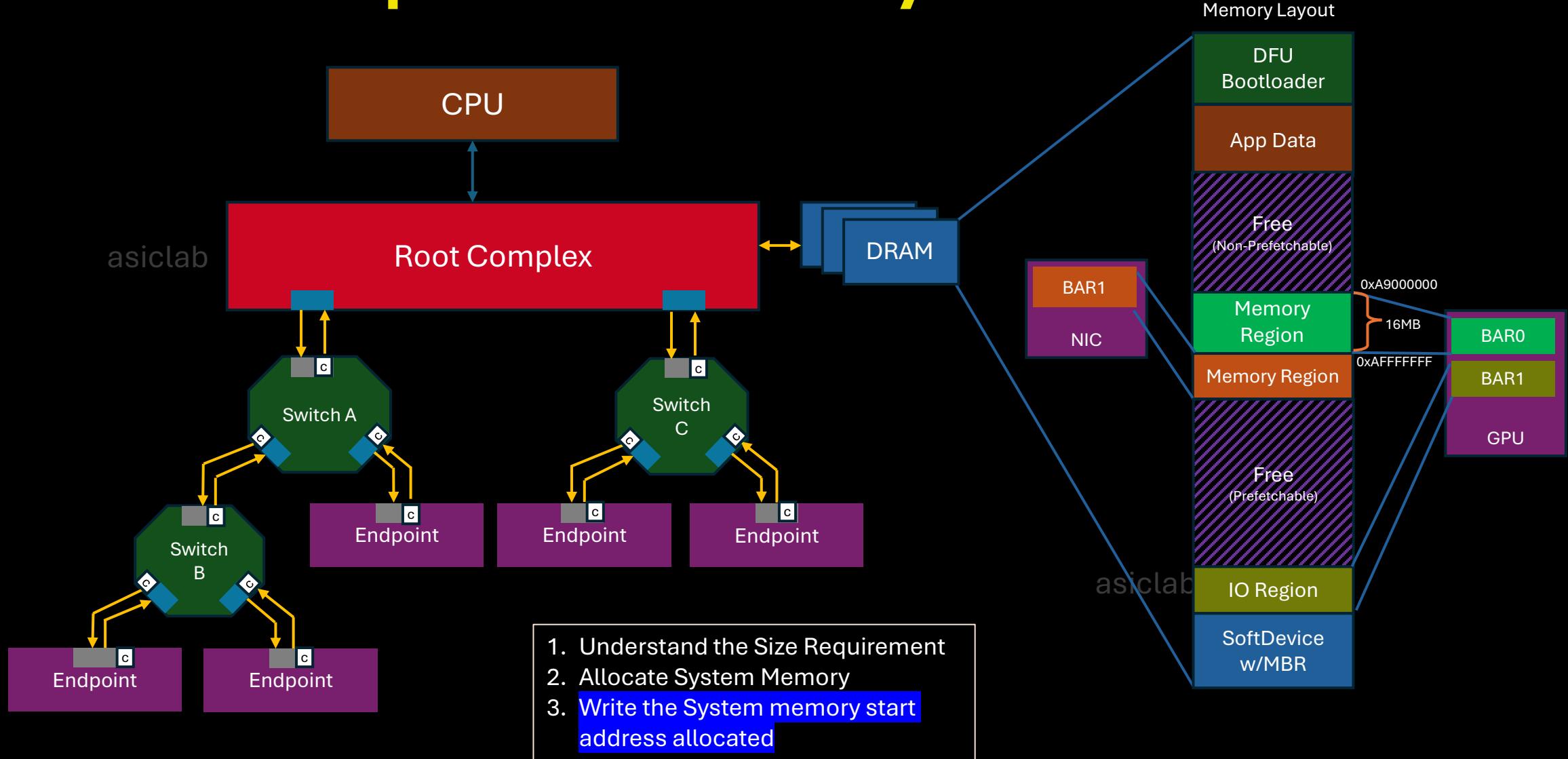
asiclab

Bit Index	Hex (32 bit)	Binary (32 bit)	Decimal	Bytes
24	0x01000000	00000001000000000000000000000000	1,67,77,216	16 MB

A9 – Start address (BAR0 becomes 0xA9000000)
Zeros appended for the higher side.

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

Address Map of PCI-Based System



Address Calculation

1. Understand the Size Requirement
2. Set the size in the System Memory
3. Write the System memory start address allocated

Address Calculation:

Start Address = 0xA9 00 00 00

Size of memory required = 16 MB = $16 * 1024 * 1024 = 16,777,216$ Bytes

Last Address = Start Address + Size – 1

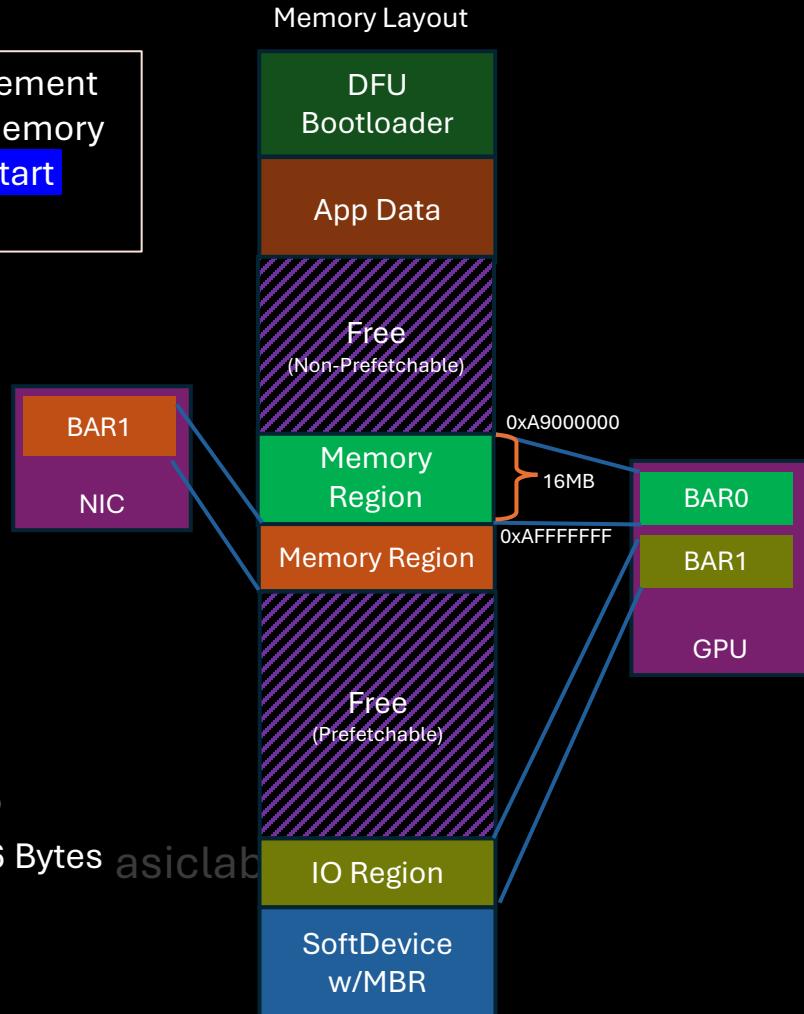
16 MB = 0x01 00 00 00 or 0000 0001 0000 0000 0000 0000 0000 0000b

$$\begin{aligned}\text{Last Address} &= 0xA9 00 00 00 + 0x01 00 00 00 - 1 \\ &= 0xAA 00 00 00 - 1 \\ &= 0xAF FF FF FF\end{aligned}$$

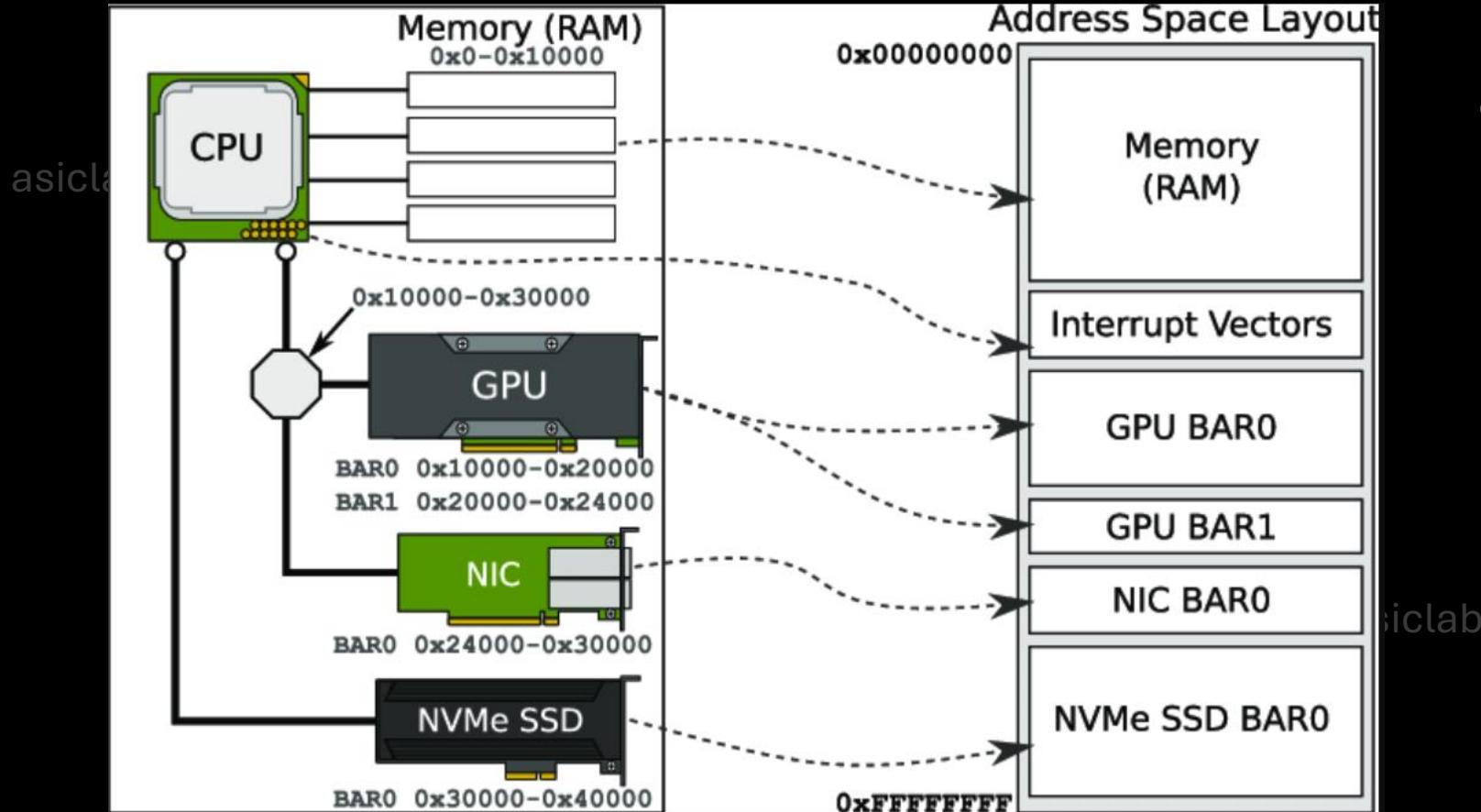
Do Not Distribute

© Copyright protected 2024

asiclab
Learn. Thrive



Memory Mapping



GPU
BAR0: MMIO Registers
BAR1: VRAM aperture
BAR2/BAR3: Indirect memory access

NIC
BAR0: Tx/Rx Buffers

NVMe SSD
BAR0: Control Registers

Configuration Space – Type 0 Header

1. Understand the Size Requirement

XXXX XXXX|0000 0000 0000 0000 0000 0000
Uninitialized BAR

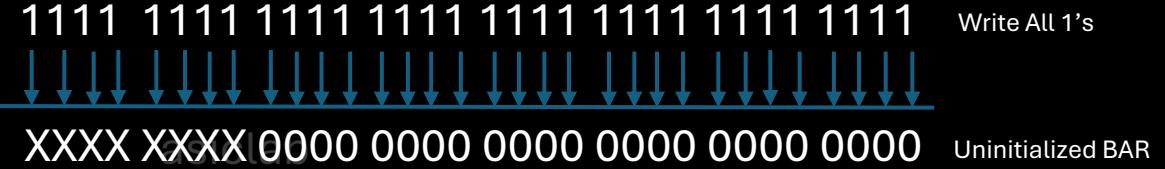
The BAR calculation involves:

$$\begin{aligned}24^{\text{th}} \text{ bit} &= 2^{24} \\&= 2^4 * 2^{10} * 2^{10} \\&= 16,777,216 \text{ Bytes} \\&= 16 \text{ MB}\end{aligned}$$

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

Configuration Space – Type 0 Header

1. Understand the Size Requirement



$$\begin{aligned}24^{\text{th}} \text{ bit} &= 2^{24} \\&= 2^4 * 2^{10} * 2^{10} \\&= 16,777,216 \text{ Bytes} \\&= 16 \text{ MB}\end{aligned}$$

The BAR calculation involves:

Step 1: Writing all 1s (0xFFFFFFFF) to the BAR register.

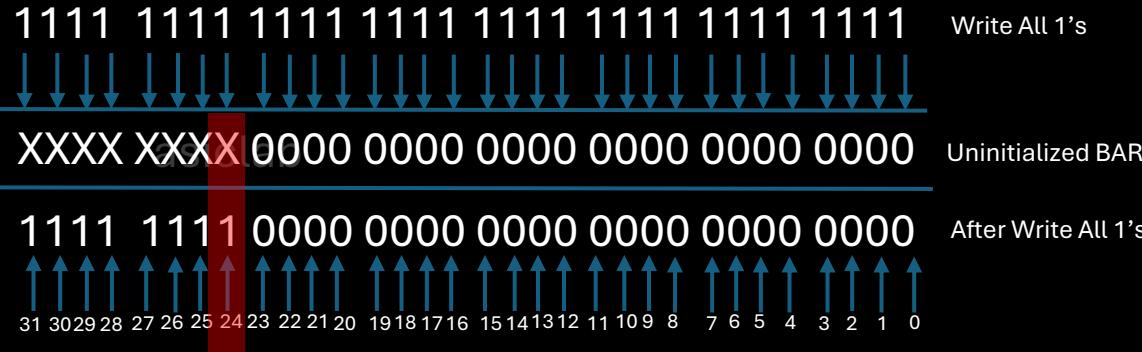
Step 2: Reading back the value and masking the non-address bits to get the size of the memory region. (Eg: `bar_value &= ~0xF;`)

Step 3: Subtracting the result from 0xFFFFFFFF and adding 1 to determine the actual size. (Eg: `unsigned int bar_size = (~bar_value + 1);`)

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

Configuration Space – Type 0 Header

1. Understand the Size Requirement



$$\begin{aligned}24^{\text{th}} \text{ bit} &= 2^{24} \\&= 2^4 * 2^{10} * 2^{10} \\&= 16,777,216 \text{ Bytes} \\&= 16 \text{ MB}\end{aligned}$$

The BAR calculation involves:

Step 1: Writing all 1s (0xFFFFFFFF) to the BAR register.

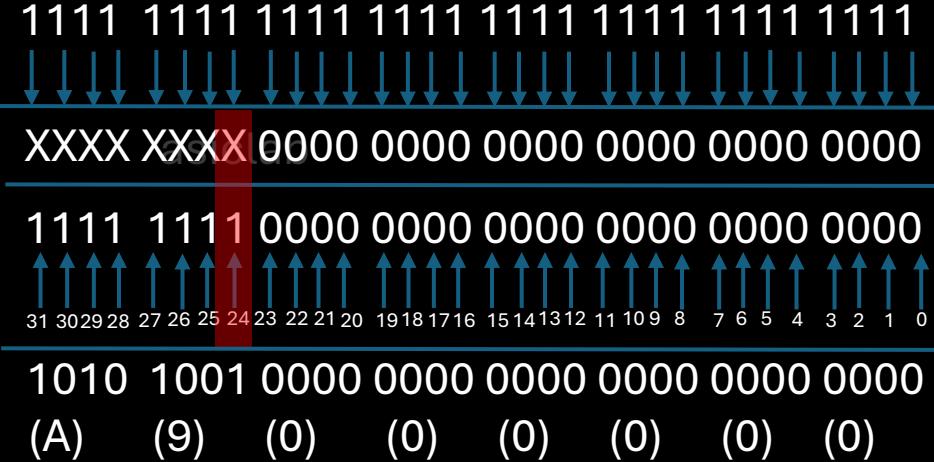
Step 2: Reading back the value and masking the non-address bits to get the size of the memory region. (Eg: `bar_value &= ~0xF;`)

Step 3: Subtracting the result from 0xFFFFFFFF and adding 1 to determine the actual size. (Eg: `unsigned int bar_size = (~bar_value + 1);`)

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

Configuration Space – Type 0 Header

1. Understand the Size Requirement



Write All 1's

Uninitialized BAR

After Write All 1's

BAR Written with Base Address

$$\begin{aligned}24^{\text{th}} \text{ bit} &= 2^{24} \\&= 2^4 * 2^{10} * 2^{10} \\&= 16,777,216 \text{ Bytes} \\&= 16 \text{ MB}\end{aligned}$$

The BAR calculation involves:

Step 1: Writing all 1s (0xFFFFFFFF) to the BAR register.

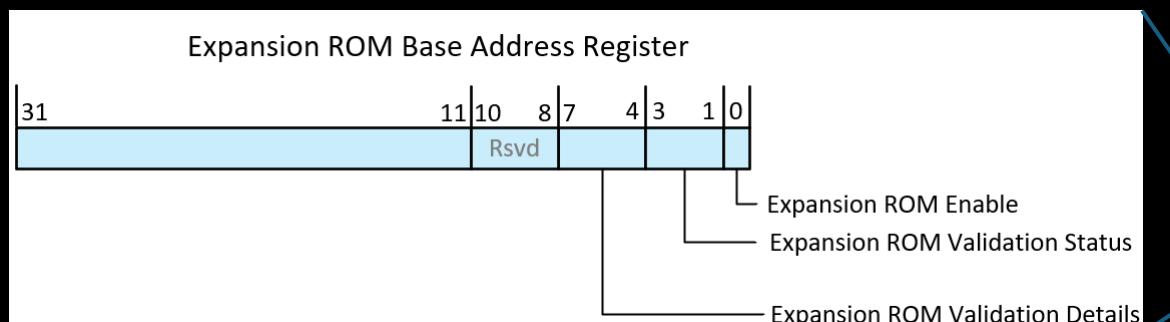
Step 2: Reading back the value and masking the non-address bits to get the size of the memory region. (Eg: `bar_value &= ~0xF;`)

Step 3: Subtracting the result from 0xFFFFFFFF and adding 1 to determine the actual size. (Eg: `unsigned int bar_size = (~bar_value + 1);`)

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

Configuration Space – Type 0 Header

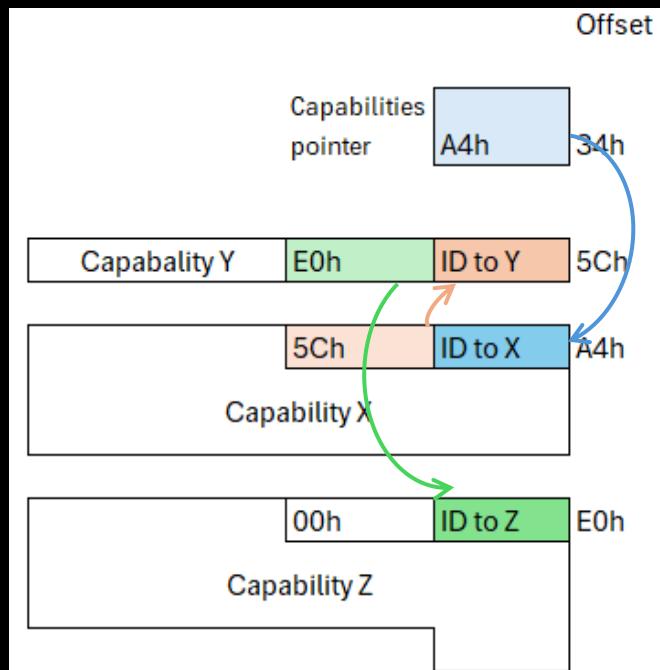
asiclab



Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code			Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Base Address 2 (BAR 2)				18h
Base Address 3 (BAR 3)				1Ch
Base Address 4 (BAR 4)				20h
Base Address 5 (BAR 5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved			Capabilities Pointer	2Ch
Reserved				30h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	34h
				38h
				3Ch

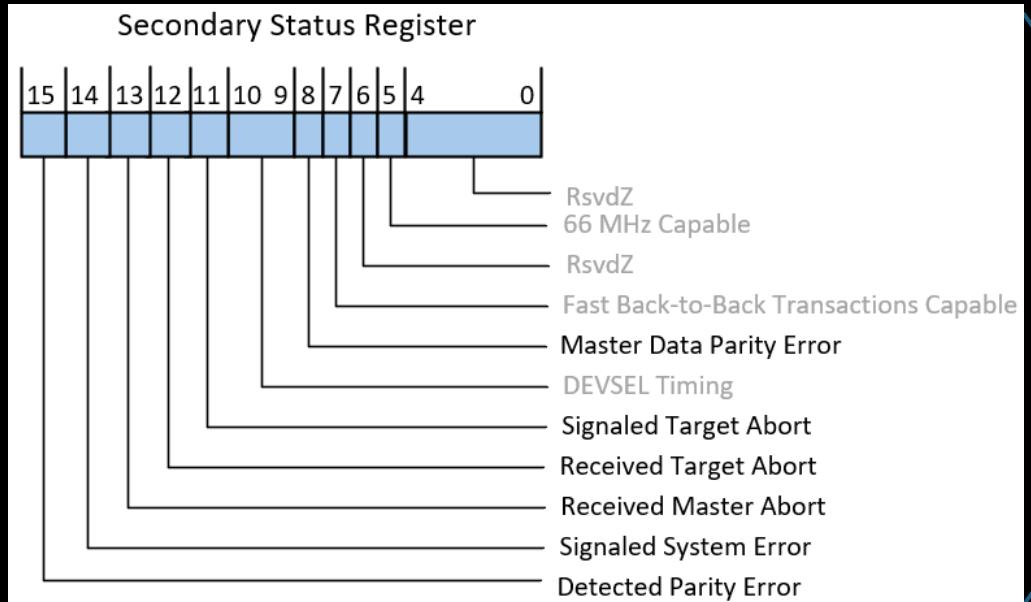
Configuration Space – Type 0 Header

- Capabilities Pointer –
 - Capabilities List Bit in the Status Register is enabled
 - Point To the first DW in the Capabilities linked list



Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID			Vendor ID	
Status Register			Command Register	
Class code			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Base Address 2 (BAR 2)				
Base Address 3 (BAR 3)				
Base Address 4 (BAR 4)				
Base Address 5 (BAR 5)				
CardBus CIS Pointer				
Subsystem Device ID		Subsystem Vendor ID		
Expansion ROM Base Address				
Reserved				Capabilities Pointer
Reserved				
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	

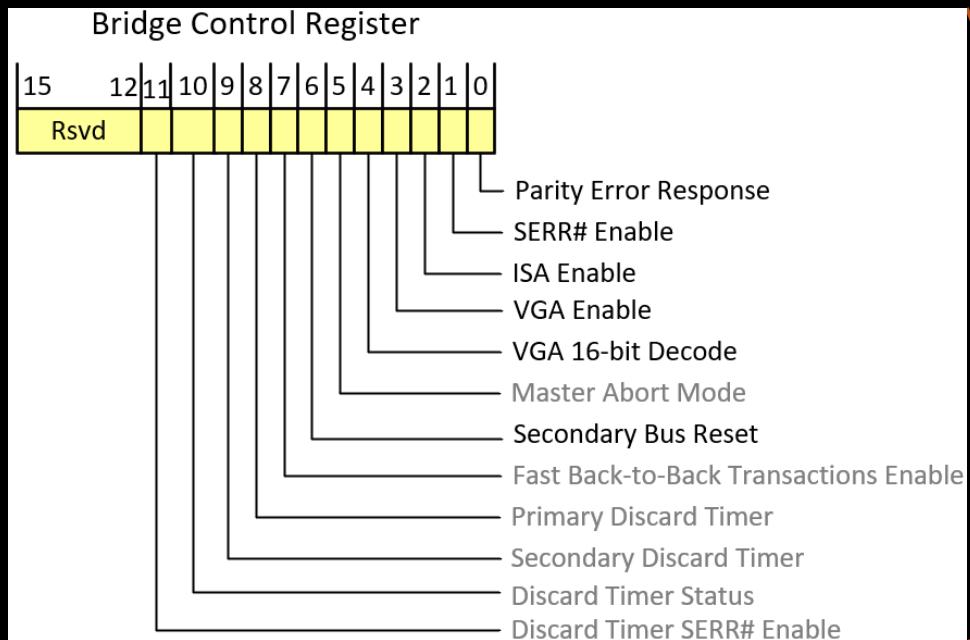
Configuration Space – Type 1 Header



Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
	Class code		Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
		Base Address 0 (BAR 0)		10h
		Base Address 1 (BAR 1)		14h
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	18h
Secondary Status		I/O Limit	I/O Base	1Ch
Memory Limit		Memory Base		20h
Prefetchable Memory Limit		Prefetchable Memory Base		24h
	Prefetchable Base Upper 32 bits			28h
	Prefetchable Limit Upper 32 bits			2Ch
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30h
	Reserved		Capabilities Pointer	34h
	Expansion ROM Base Address			38h
Bridge Control	Interrupt Pin	Interrupt Line		3Ch

Configuration Space – Type 1 Header

asiclab



Byte 3	Byte 2	Byte 1	Byte 0	Offset		
31	23	15	7	0		
Device ID		Vendor ID				
Status Register		Command Register				
Class code			Rev ID			
BIST	Header type	Latency timer	Cache Line Size			
Base Address 0 (BAR 0)						
Base Address 1 (BAR 1)						
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number			
Secondary Status		I/O Limit	I/O Base			
Memory Limit		Memory Base				
Prefetchable Memory Limit		Prefetchable Memory Base				
Prefetchable Base Upper 32 bits						
Prefetchable Limit Upper 32 bits						
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits				
Reserved			Capabilities Pointer			
Expansion ROM Base Address						
Bridge Control		Interrupt Pin	Interrupt Line			

Configuration Space

- PCI Capabilities Space

asiclab

PCI-compatible register area can be accessed in PCI manner or Via PCI Express Enhanced Configuration Mechanism

PCI-Express Capability Structure
Must be implemented within this area

256 Byte Configuration Register space (per Function)

PCI Header 16DWs

PCI device-specific and New capability register sets 48 DWs

Offset 000h

Offset 040h

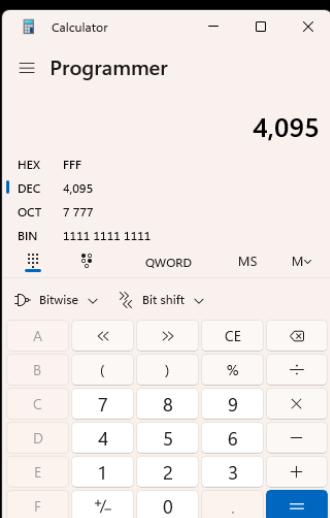
Offset 100h

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID			Vendor ID	00h
Status Register			Command Register	04h
		Class code		08h
			Rev ID	
BIST	Header type	Latency timer	Cache Line Size	0Ch
		Base Address 0 (BAR 0)		10h
		Base Address 1 (BAR 1)		14h
		Base Address 2 (BAR 2)		18h
		Base Address 3 (BAR 3)		1Ch
		Base Address 4 (BAR 4)		20h
		Base Address 5 (BAR 5)		24h
		CardBus CIS Pointer		28h
Subsystem Device ID	Subsystem Vendor ID			2Ch
	Expansion ROM Base Address			30h
		Reserved	Capabilities Pointer	34h
			Reserved	38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Express Extended Configuration Space 960DWs

Optional PCI Express Extended Capability register sets are implemented within this area:

- Advanced error reporting
- Virtual Channel capability
- Device Serial Number capability
- Power budgeting capability



Do Not Distribute

© Copyright

Offset FFFh

asiclab
Learn. Thrive

Calculation:
16DW = 64 Bytes
48DW = 192 Bytes

256 Bytes

960DW = 3840 Bytes
16+48+960 = 1024 DWs

1024 * 4 = 4096 = 4Kb

PCIe Configuration

CPUs can directly access Memory and IO space, but configuration must be indirectly addressed, requiring logic to interpret CPU commands into Configuration commands.

1. ~~asIC~~: Configuration Access Mechanism (Legacy) Indirect through IO addresses
2. ECAM: Enhanced Configuration Access Mechanism, Indirect through Memory addresses

asiclab

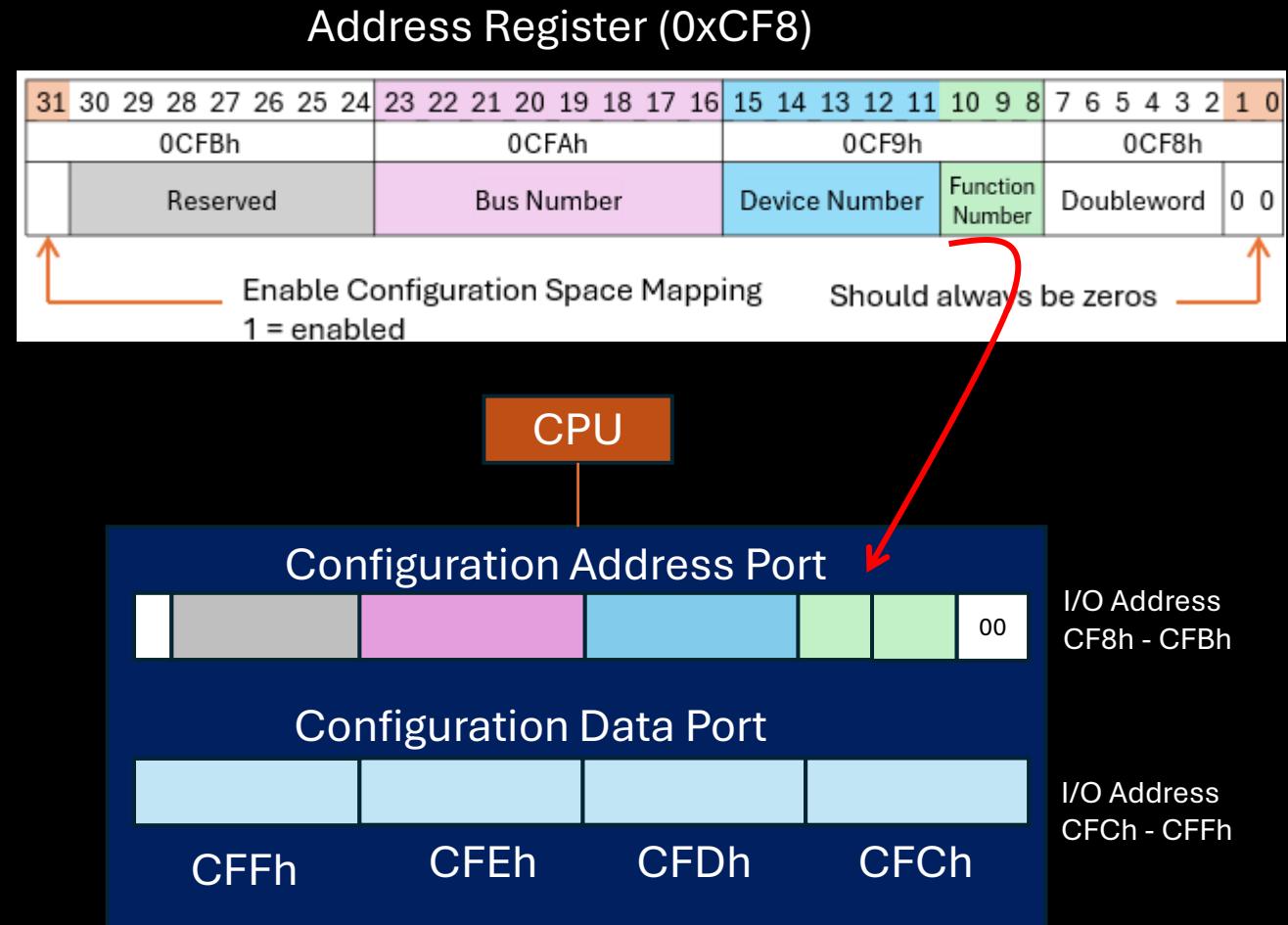
Configuration Access Mechanisms

- PCI Express configuration access mechanism
 - Each function's 4KB configuration space starts at a 4KB-aligned address within the 256MB memory space set aside as configuration space
 - Address bits 63:28 indicates the 256MB-aligned base address of the overall asiclab Enhanced configuration address range
 - Address bits 27:20 select the target bus (1-of-256).
 - Address bits 19:15 select the target device (1-of-32) on the bus.
 - Address bits 14:12 select the target function (1-of-8) within the device.
 - Address bits 11:2 selects the target dword (1-of-1024) within the selected function's configuration space.
 - Address bits 1:0 define the start byte location within the selected dword.
- To read from Bus 4, Device 0, Function 0's, Vendor ID configuration register
 - `mov ax,[00000000]50400000]`
 - This is two-byte access; “5” indicates the base address of the configuration space for this platform

Configuration Access Mechanism (Legacy)

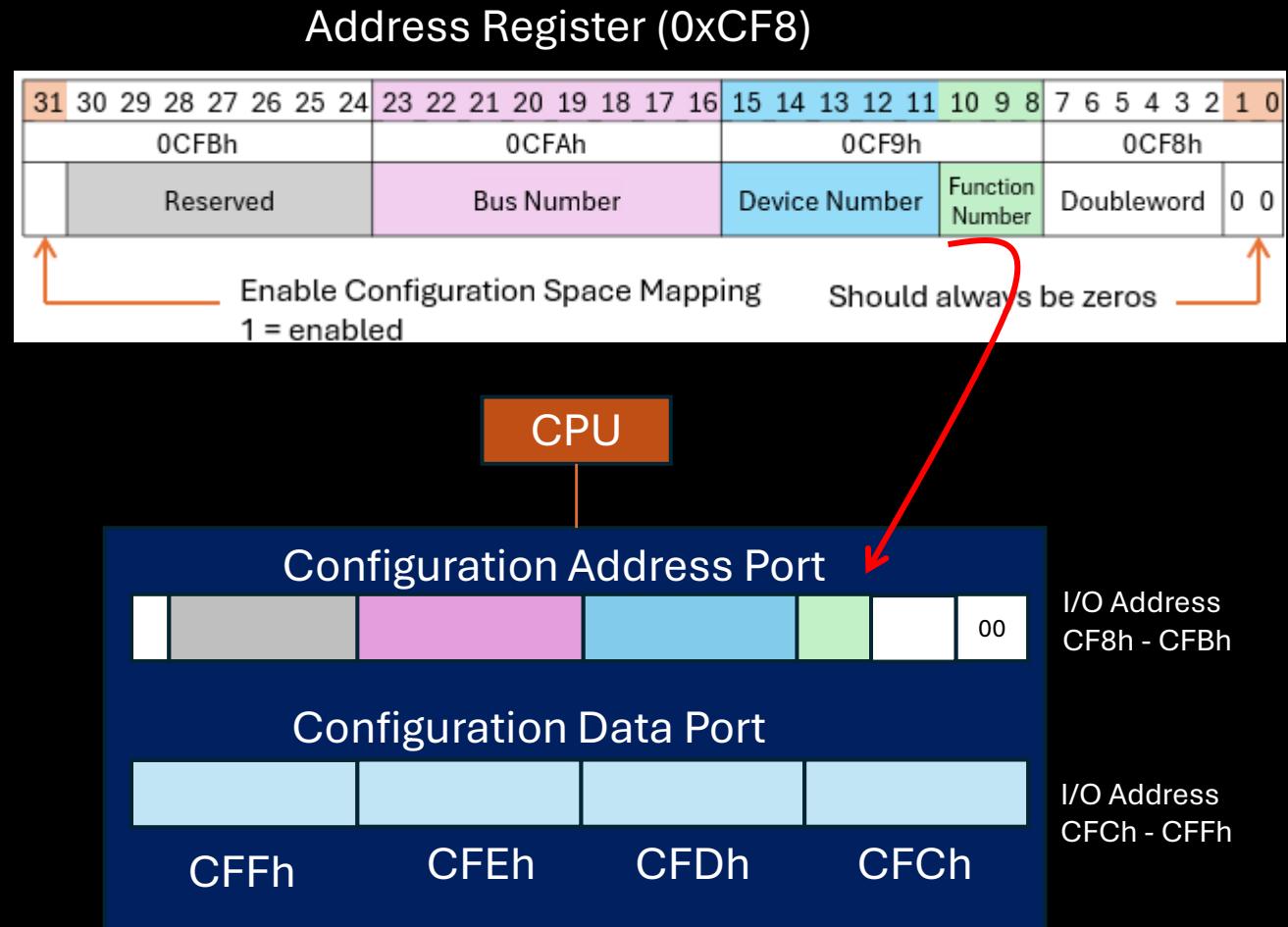
- Processor reads and writes to I/O address 0CF8h and 0CFCh are converted to configuration reads and writes by the Host Bridge

- Advantage: Uses very little address space
- Disadvantage: Require 2 steps (write to CF8h, then read or write to CFCh); allows multiple threads to interfere with each other

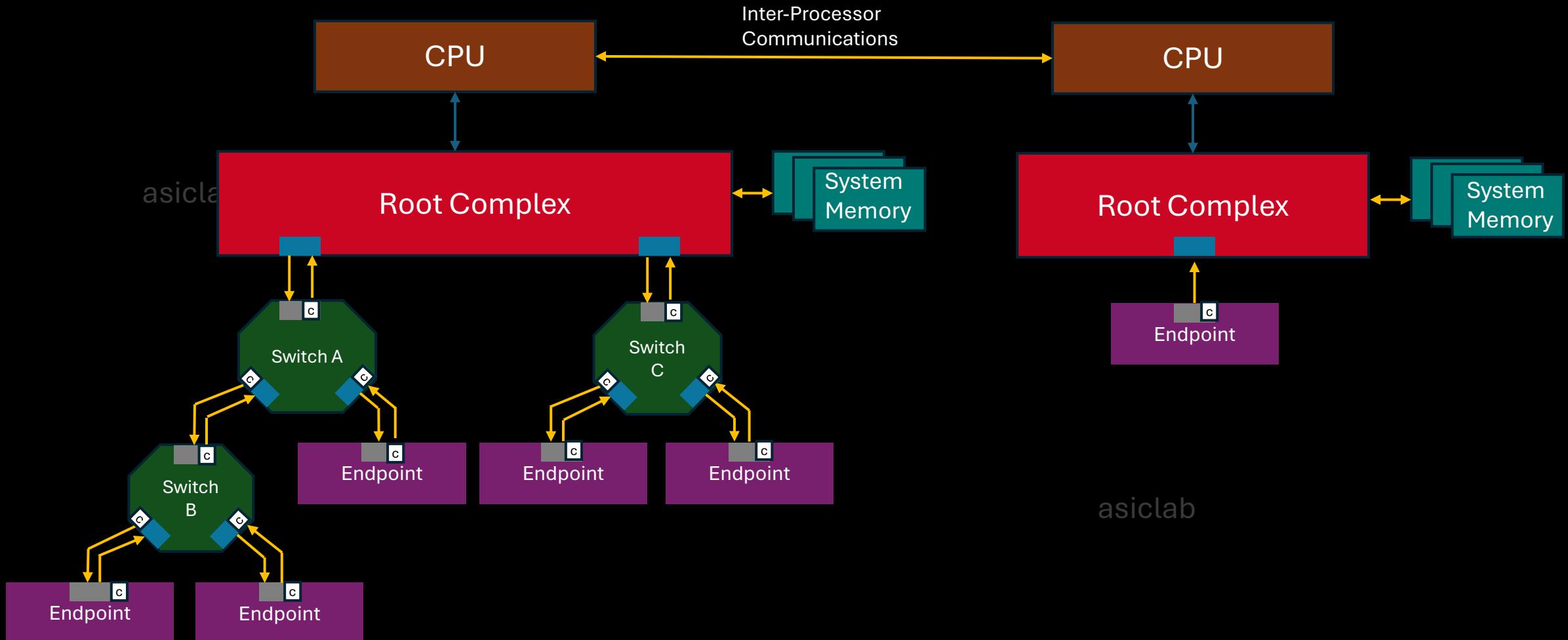


Configuration Access Mechanism (Legacy)

7.5.1.1	Type 0/1 Common Configuration Space.....
7.5.1.1.1	Vendor ID Register (Offset 00h)
7.5.1.1.2	Device ID Register (Offset 02h)
7.5.1.1.3	Command Register (Offset 04h)
7.5.1.1.4	Status Register (Offset 06h)
7.5.1.1.5	Revision ID Register (Offset 08h)
7.5.1.1.6	Class Code Register (Offset 09h)
7.5.1.1.7	Cache Line Size Register (Offset 0Ch) ...
7.5.1.1.8	Latency Timer Register (Offset 0Dh) ...
7.5.1.1.9	Header Type Register (Offset 0Eh)
7.5.1.1.10	BIST Register (Offset 0Fh)
7.5.1.1.11	Capabilities Pointer (Offset 34h)
7.5.1.1.12	Interrupt Line Register (Offset 3Ch)
7.5.1.1.13	Interrupt Pin Register (Offset 3Dh).....
7.5.1.1.14	Error Registers

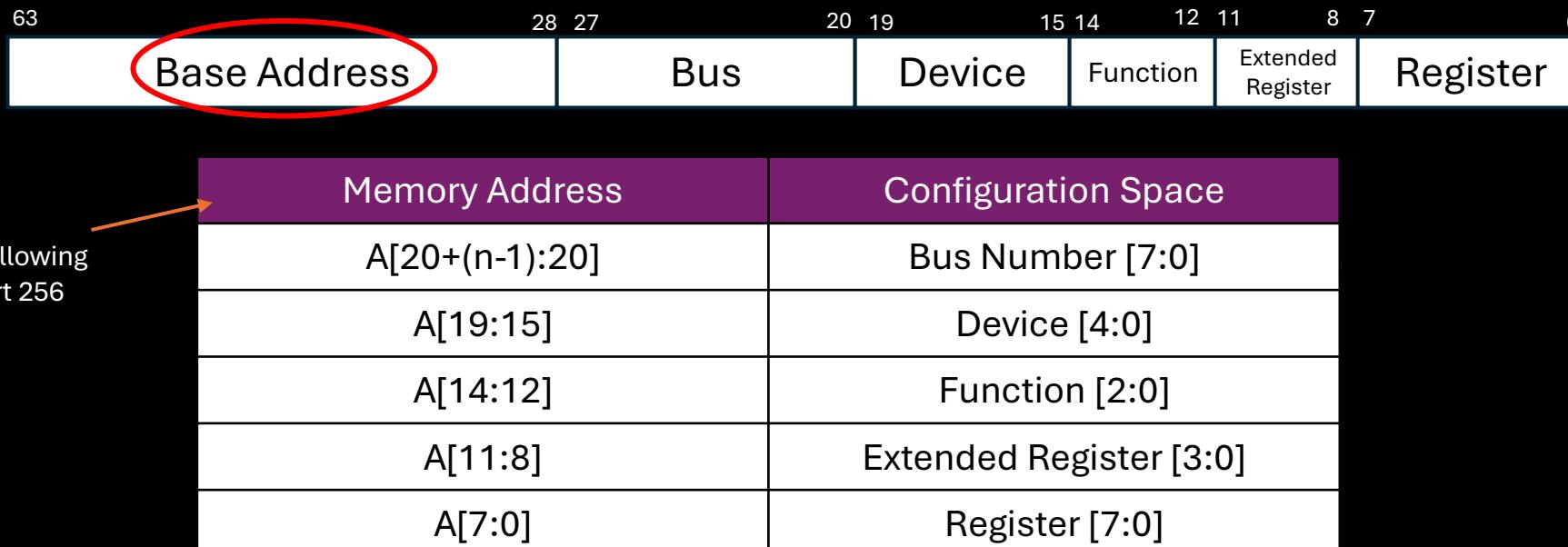


Configuration Access Mechanism (Legacy)

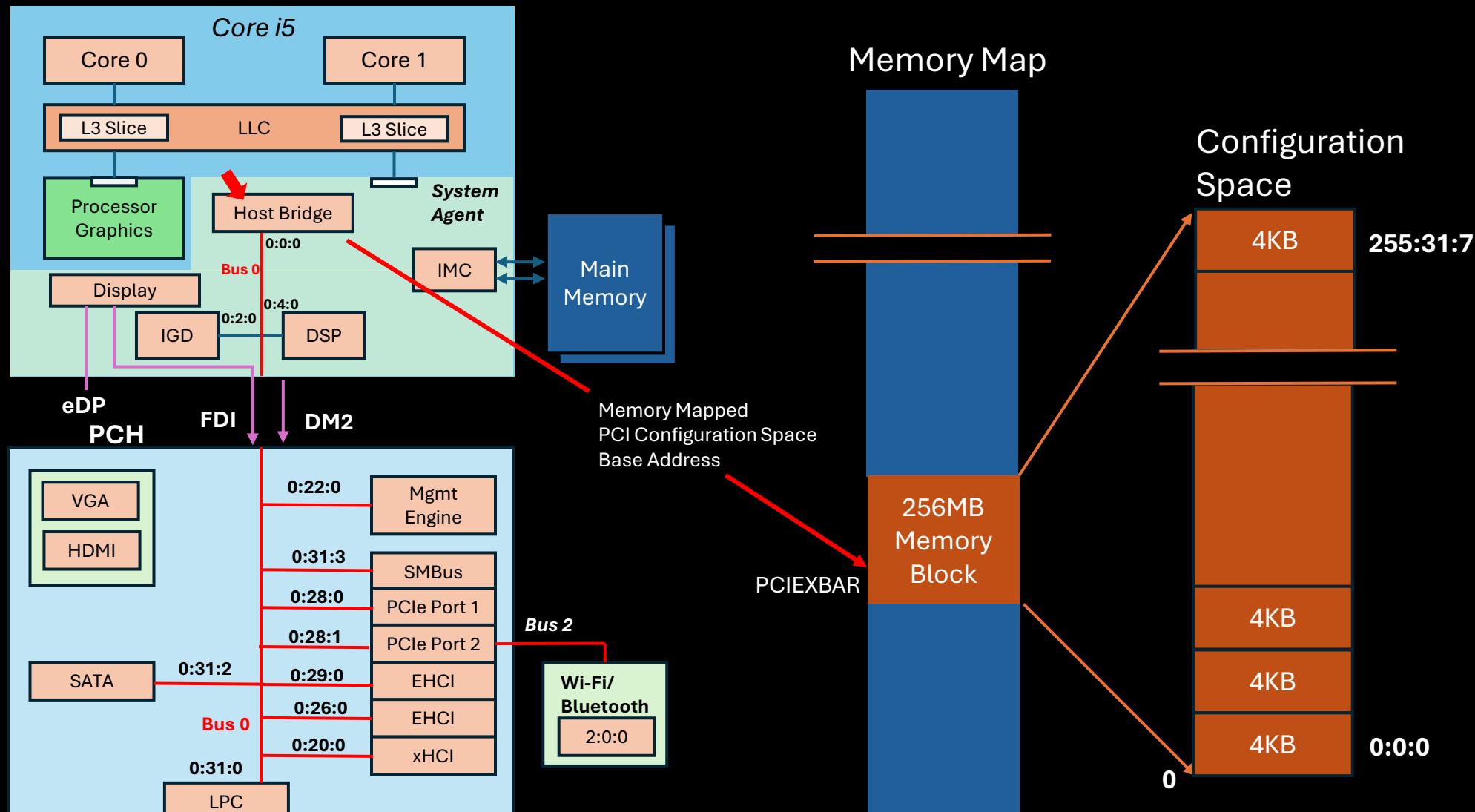


Enhanced Configuration Access Mechanism

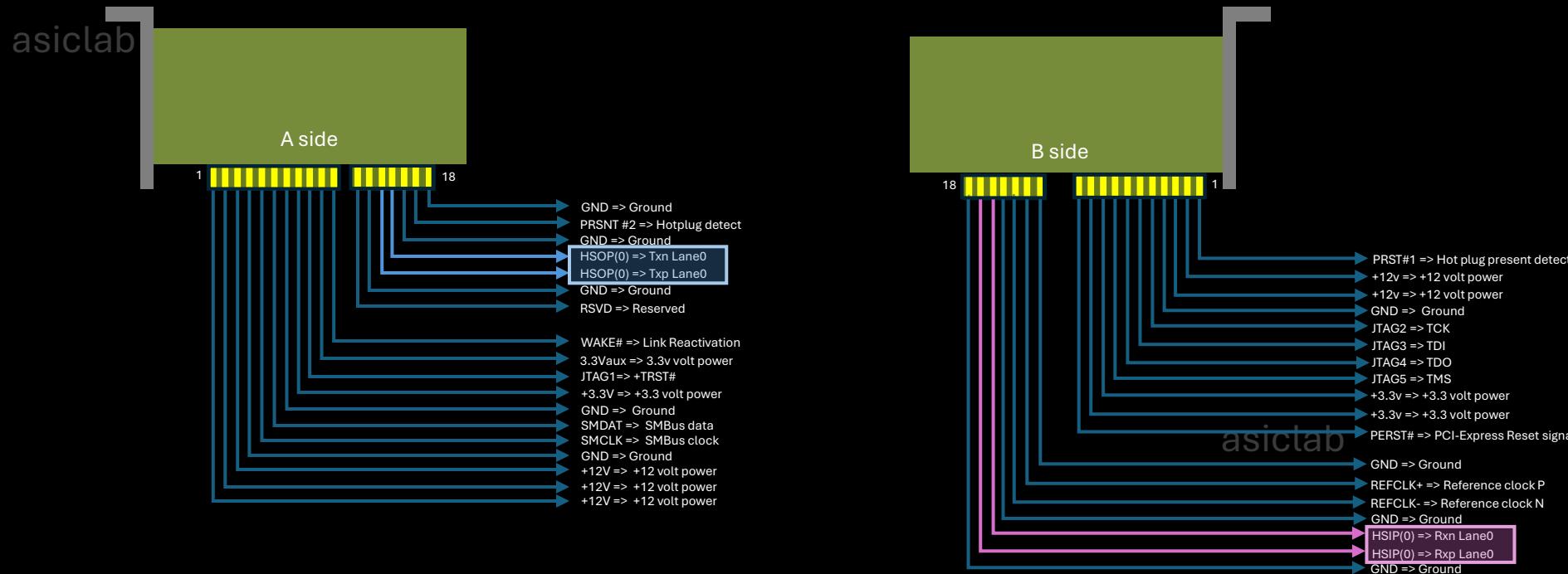
- Memory access within a programmed range is translated into configuration cycle by the Host Bridge
 - Advantage: one-step access, no possible interference between tasks
 - Disadvantage: large memory range dedicated for this purpose
- asiclab • 28-bit address mapped into system memory address space
- Bits A[63:28] are defined by a Base Address register – Memory cycles whose upper bits match this base will generate configuration cycles



Enhanced Configuration Access Mechanism

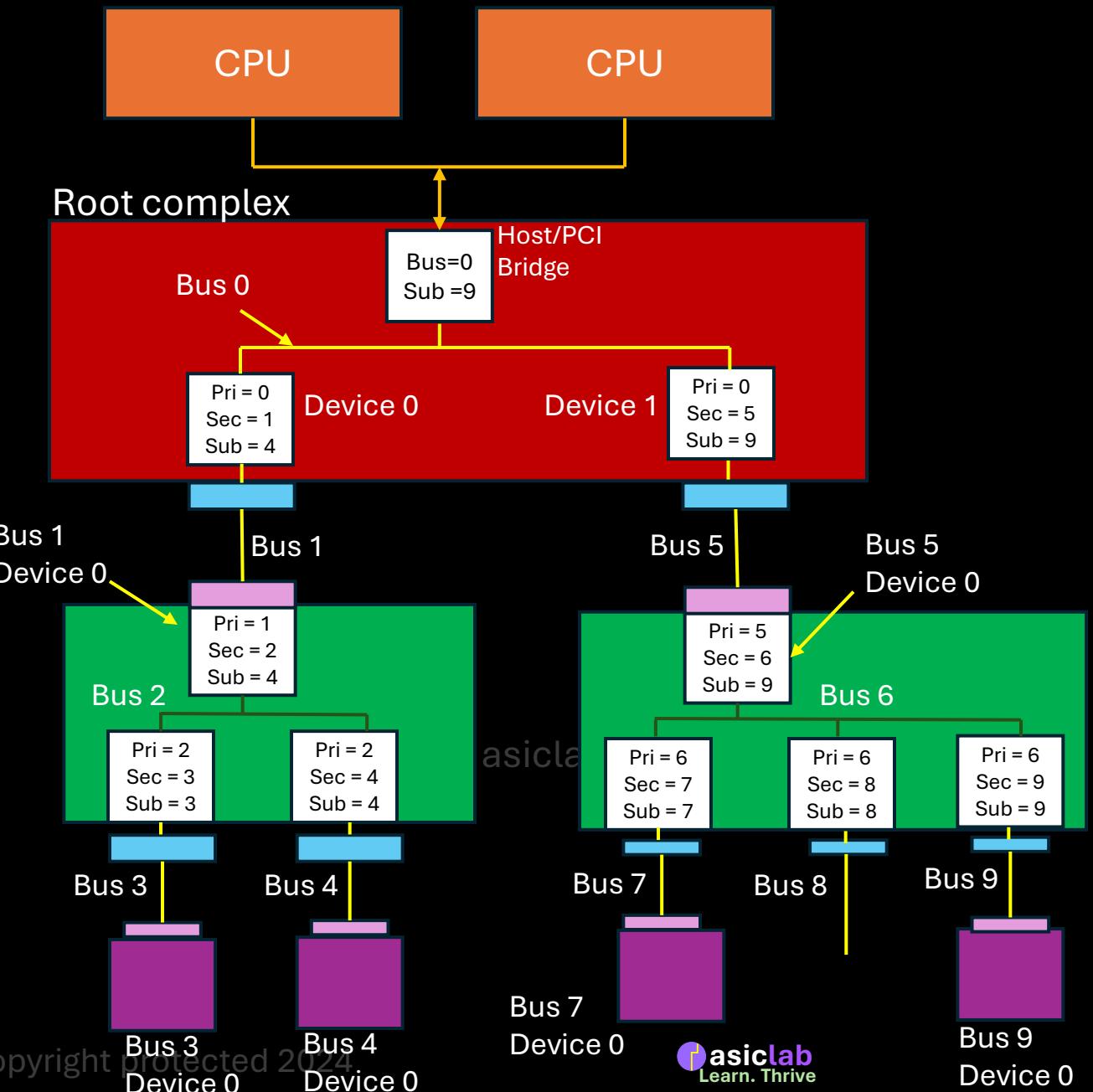


PCIe x1 Pinout



Topology

- Primary, Secondary and Subordinate Bus
 - Bus on upstream side – Primary
 - Bus on downstream side - asic
 - Secondary
 - Highest numbered Bus under the present Bus – Subordinate
- On power-up
 - Configuration software hasn't yet scanned the fabric
 - Enumeration process is done, and fabric topology is scanned



asiclab

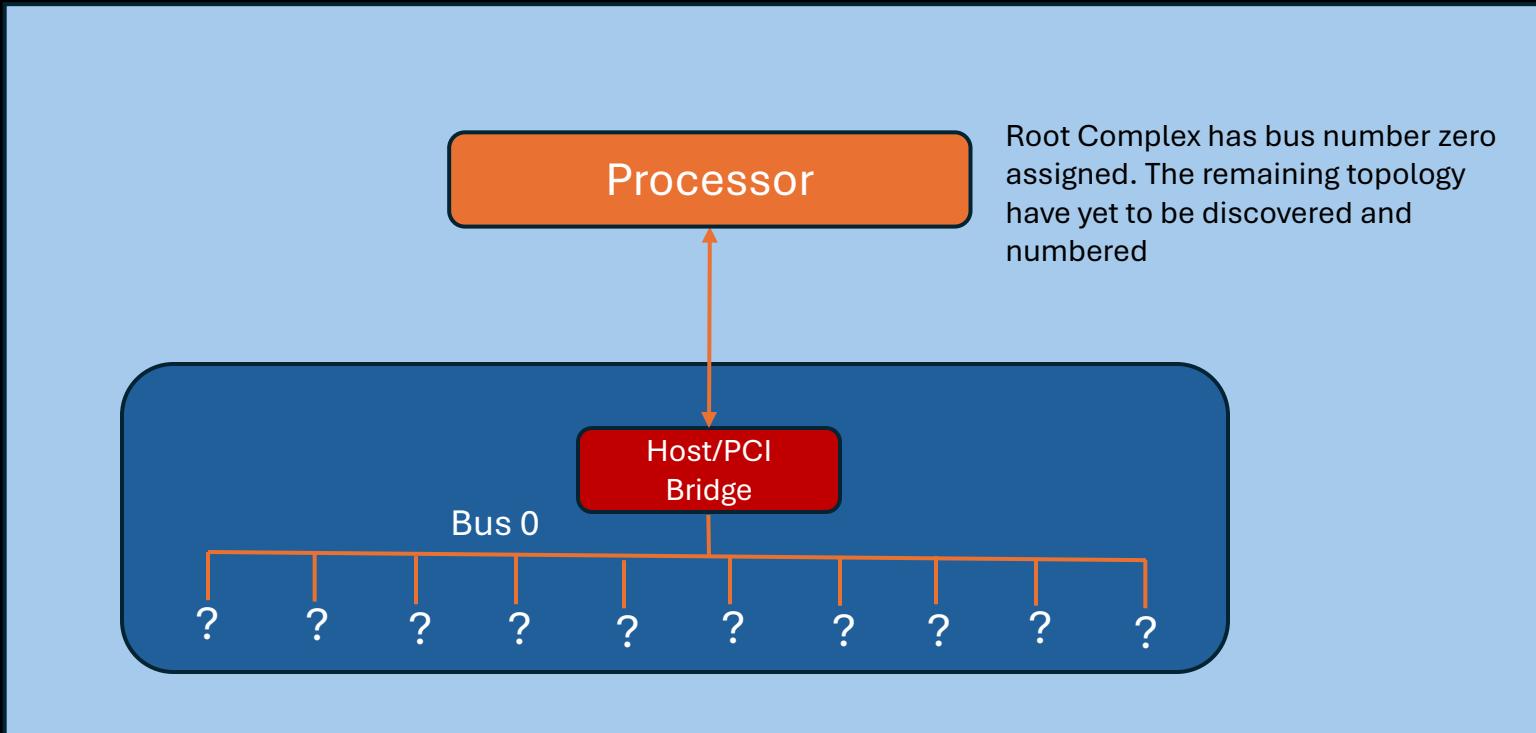
Enumeration

asiclab

What is enumeration?

- Process of scanning the PCIe fabric to discover its topology
- Software knows that there will be a host/PCI bridge and bus number 0 will be on the secondary side of that bridge

asiclab

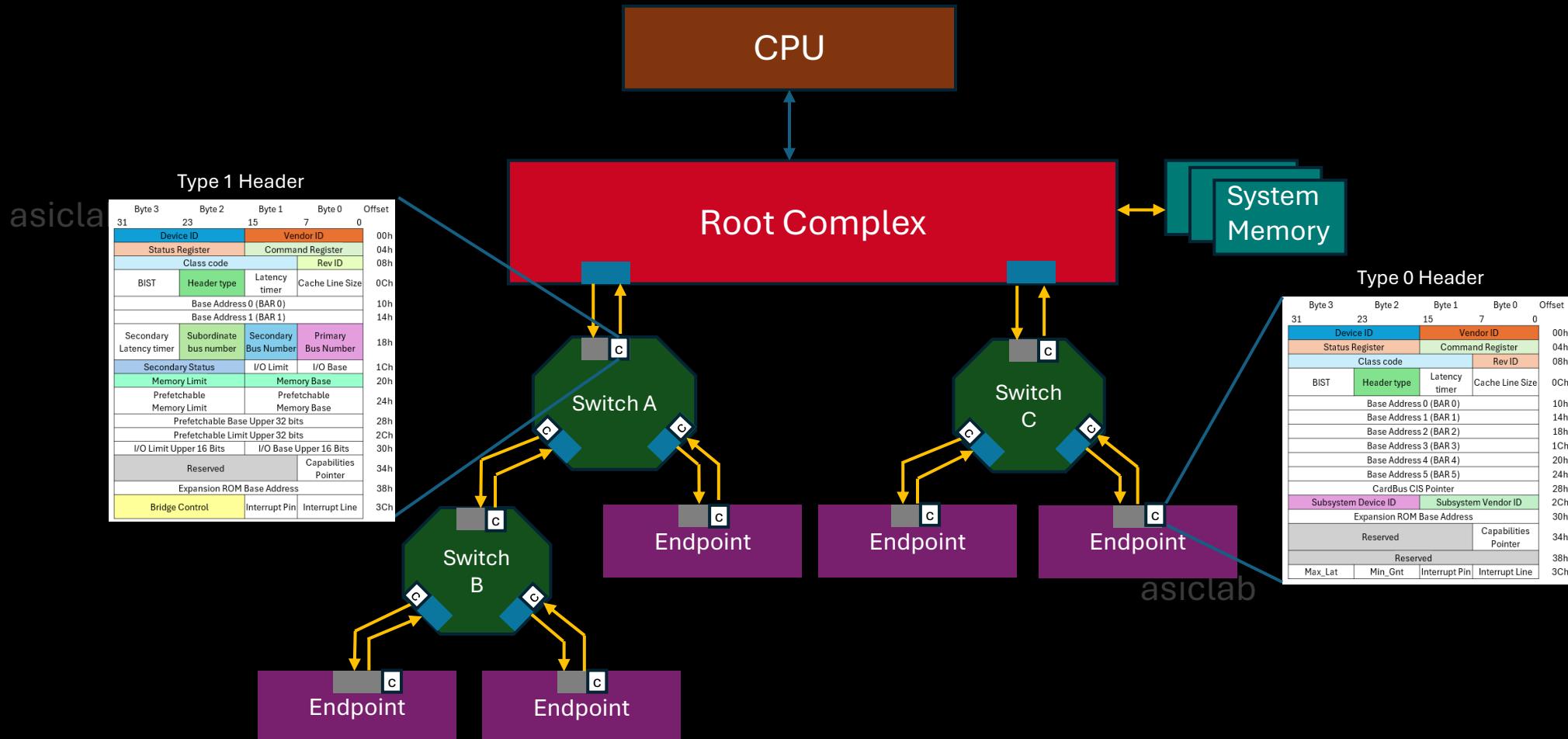


What is enumeration?

- Process of detecting devices connected to CPU
- Initiated by CPU when power on
- Assigning address to device
 - ID => Bus, Device, Function
 - Memory base address/range
- Feature of the device

asiclab

PCIe Configuration Space



Enumeration

- Discovering a Function
 - Device not present
 - Device not ready

asiclab

asiclab

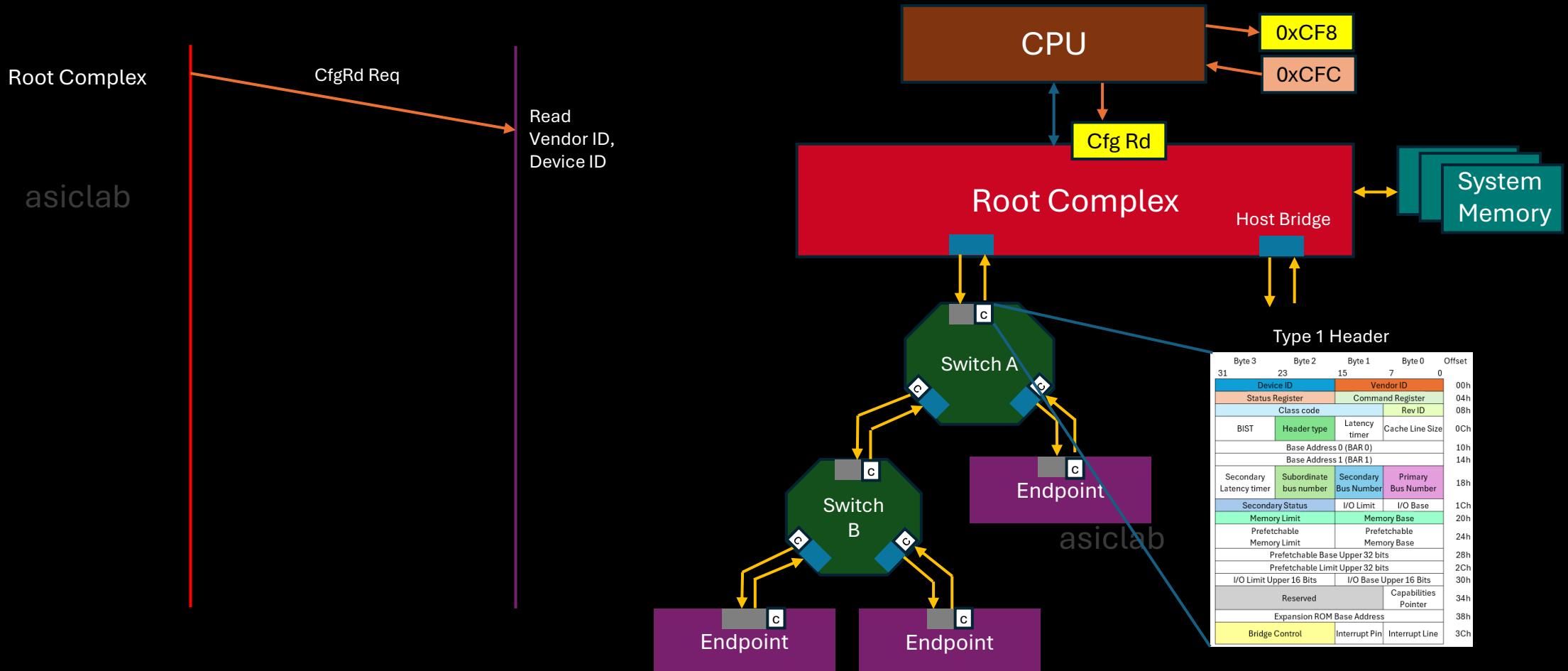
Enumeration

- Discovering a Function
 - Device not present
 - Device not ready

asiclab

asiclab

Discovering a Function



Configuration Access Mechanism (Legacy)

asiclab

$2^8 = 256$ Buses
 $2^5 = 32$ Devices
 $2^3 = 8$ Functions

Address Register (0xCF8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

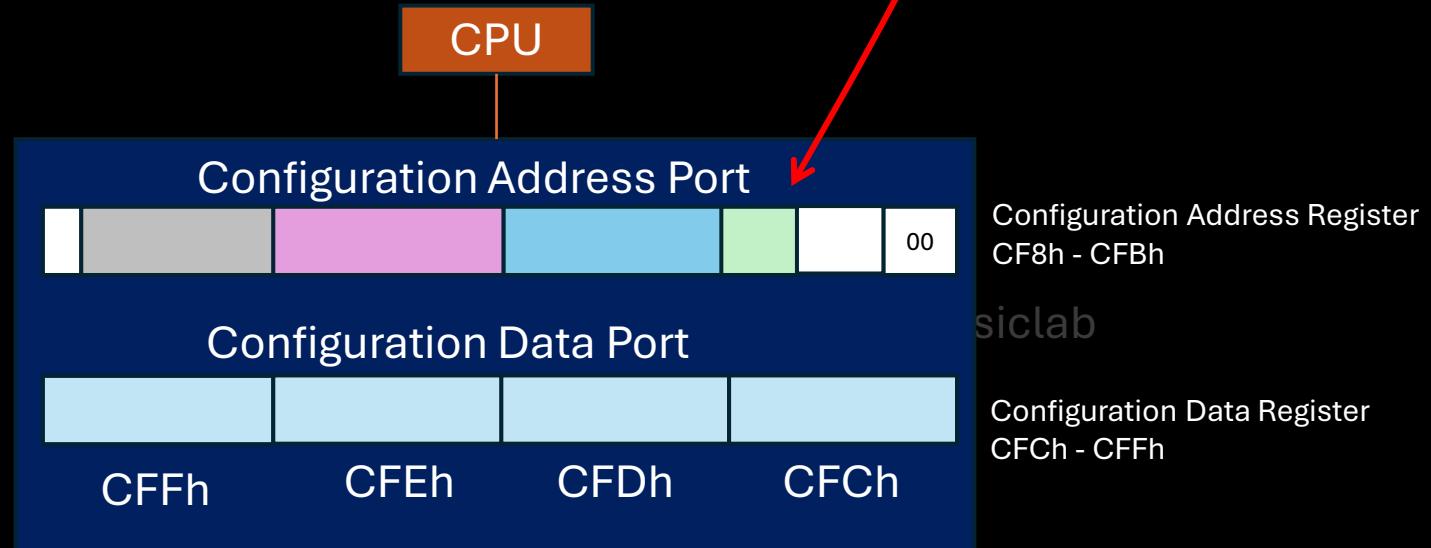
0CFBh 0CFAh 0CF9h 0CF8h

Reserved Bus Number Device Number Function Number Doubleword 0 0

Enable Configuration Space Mapping
1 = enabled

Should always be zeros

CPU



Discovering a Function

- The configuration software discovers the existence of a Function by reading from its VendorID register.
- VendorID – a unique 16-bit value assigned to each vendor by PCI-SIG.
- VendorID value is hardwired into the vendorID register of each Function designed by the vendor.
- By reading this register in all the possible combinations of BDF numbers in the system, enumeration software can search through the entire topology to learn which devices are present.
- There are two problems that can arise during this process:
 - Targeted device is not present
 - Device is present but unprepared to respond

asiclab

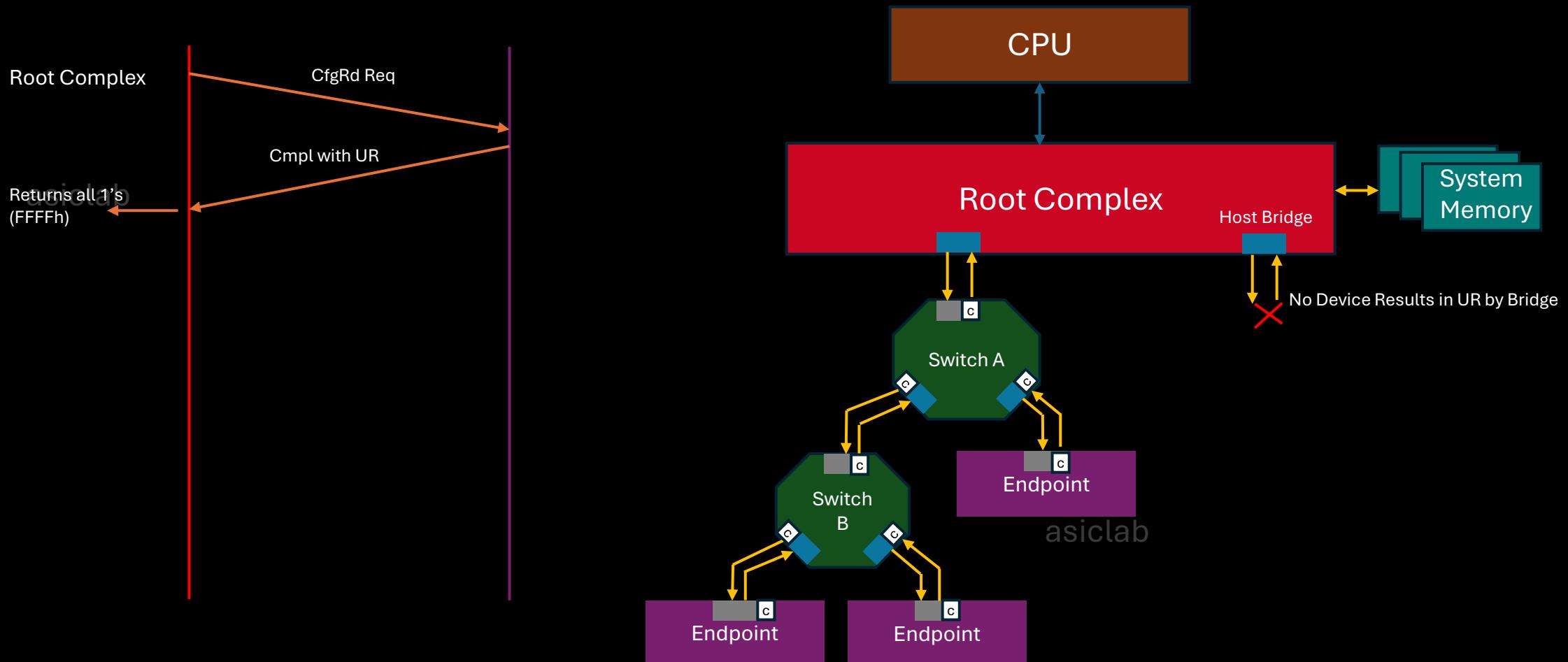
Enumeration

- Discovering a Function
 - Device not present
 - Device not ready

asiclab

asiclab

Device not present



Device not present

- A Configuration Read Request to a non-existent device will result in the bridge above the target device returning a Completion without data that has a status of UR (Unsupported Request)
- Root Complex returns all ones (FFFFh) to the processor for the data when the Completion is seen during enumeration.
- If enumeration software gets all ones (FFFFh) for a read, it understood that the device wasn't present.
- UR result would be seen as an error during runtime, it's an expected result that isn't considered an error during enumeration.
- PCIe makes a note of this event by setting the 'Unsupported Request Status' bit is given in the PCIe Capability register block.
- UR condition will be noted without marking it as an error, and that's important because a detected error might stop the enumeration process to call the system error handler.

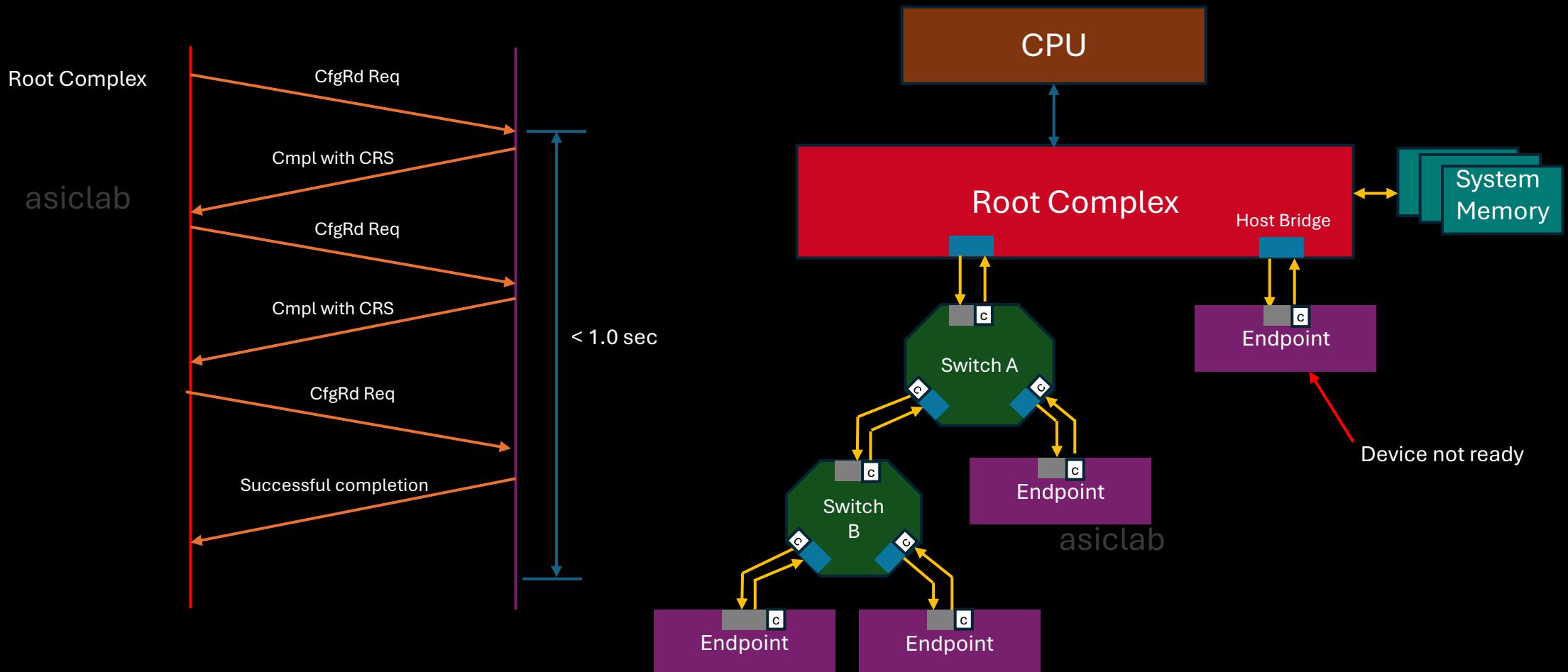
Enumeration

- Discovering a Function
 - Device not present
 - Device not ready

asiclab

asiclab

Device not ready

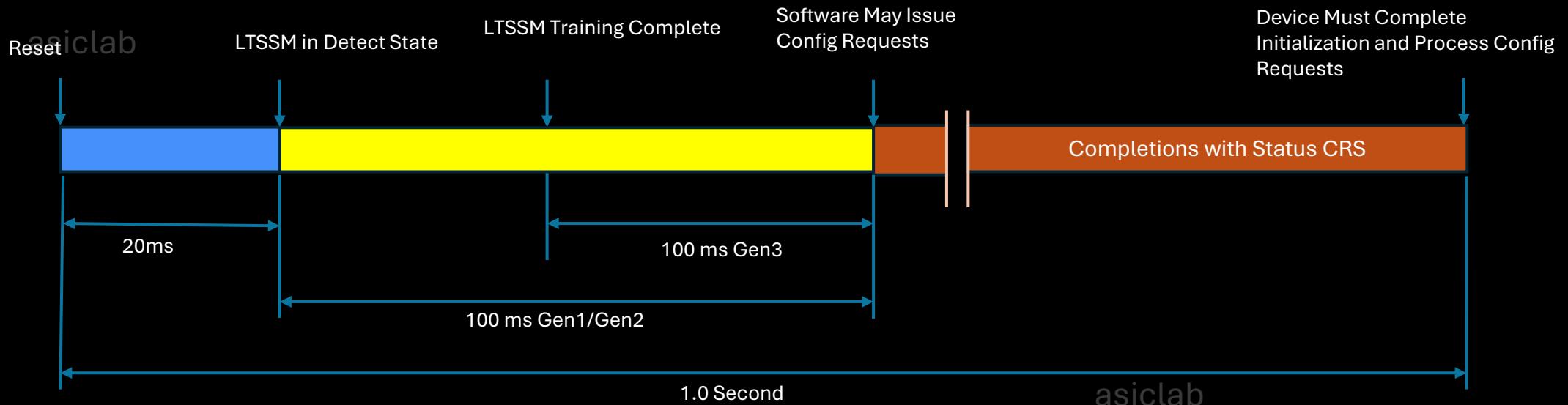


Device not ready

- There is a timing consideration for configuration:
 - If the data rate is 5.0GT/s or less(Gen1,2 speed), software must wait 100ms after reset before initiating a Configuration Request
 - If the data rate is higher than 5.0GT/s (Gen3 speed) software must wait until 100ms after Link training completes before attempting this.
- PCIe Functions must always give a Completion with specific status when they are temporarily unable to respond to a configuration access
 - Configuration retry status (CRS)
 - CRS is only legal in response to a configuration request.
 - CRS is also only valid for the one second after reset.
- The way the Root Complex handles a CRS Completion in response to a Configuration Read Request is implementation specific, except for the period following a system reset.
 - There are two options for what the Root will do next, based on the setting of the CRS software visibility bit in its Root Control Registers

asiclab

Configuration Retry Status



Configuration Retry Status

PCIe Spec Requirements,

1. Host must wait minimum of 100ms before deasserting PERST on power-on (even in case of hot plug).
2. Device must enter LTSSM.DETECT within 20ms of PERST deassertion (this does not mean to enable link training but requires Clk Config Rdy and Dualport Config done)
asiclab
3. Host can issue PCIe config accesses after 100ms of LinkUp.
4. Device responds with UR before this time.
5. Device can respond with CRS after this time upto 1sec from PERST deassertion (provided the Host completed link training within the 1sec window)
6. Device must set PCIe Cfg Rdy within 1sec of PERST deassertion (provided the Host completed link training within the 1sec window)
7. On PERST, device must go EIDLE within 3us (arbitrary number but means immediately) asiclab
8. On FLR, device can return UR before 100ms but must return CRS after 100ms (& FW must not be required to set FLRDONE). Device must set PCIe Cfg Rdy within 1sec of FLR to indicate FLR completed & PCIe link remains up all the time

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages
 - Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request

asiclab

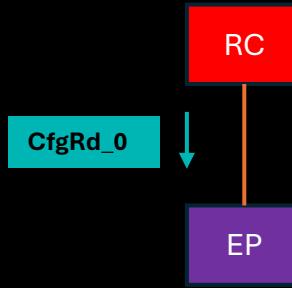
Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data		Successful Completion (SC)

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab

- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request



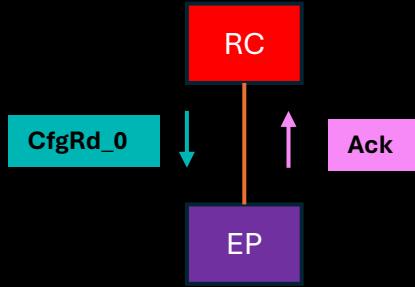
Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab

- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request



Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab



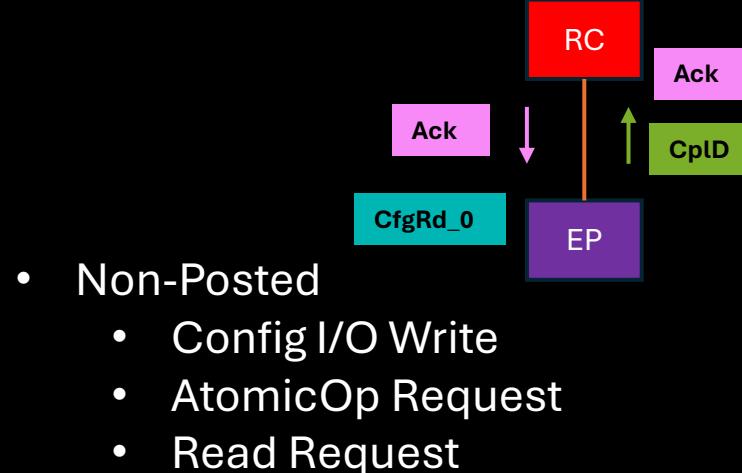
- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request

Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab



- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request

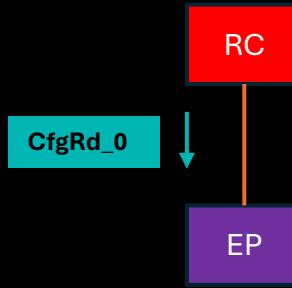
Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
1	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab

- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request



Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
1	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab



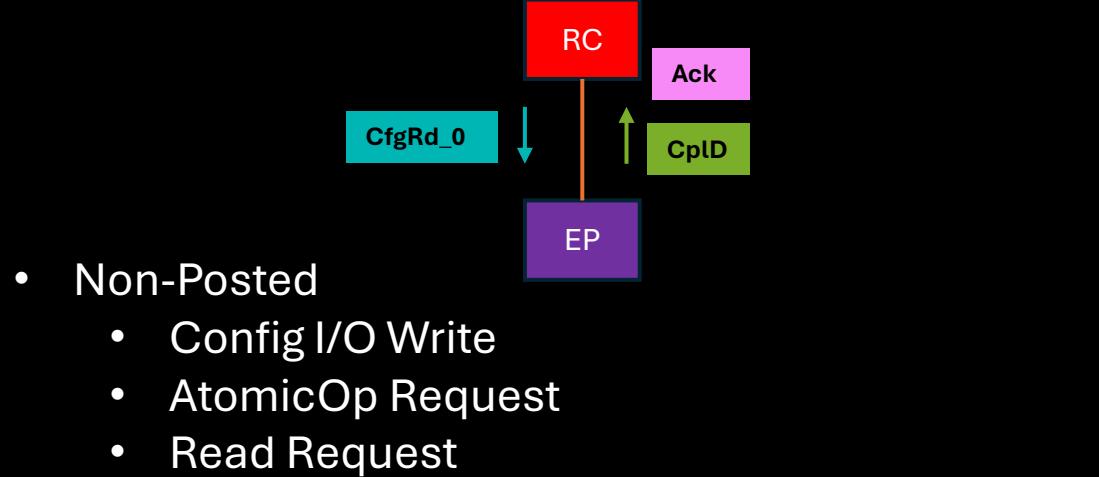
- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request

Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab

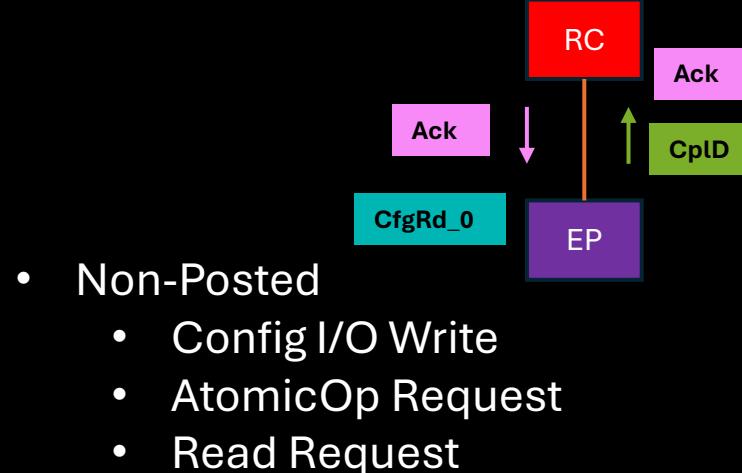


Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab



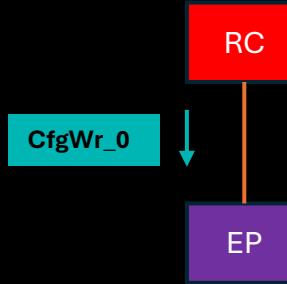
Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
1	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab

- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request



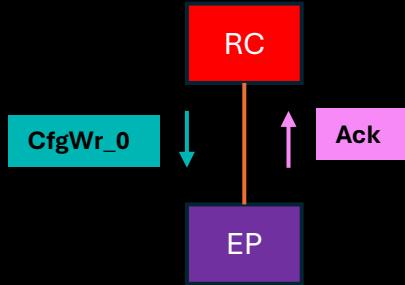
Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab

- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request



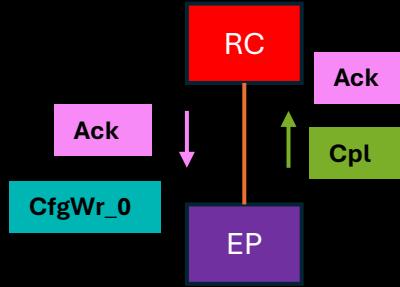
Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
0	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Transaction Examples

- Request Type categorized as
 - Posted
 - Memory Write
 - Messages

asiclab

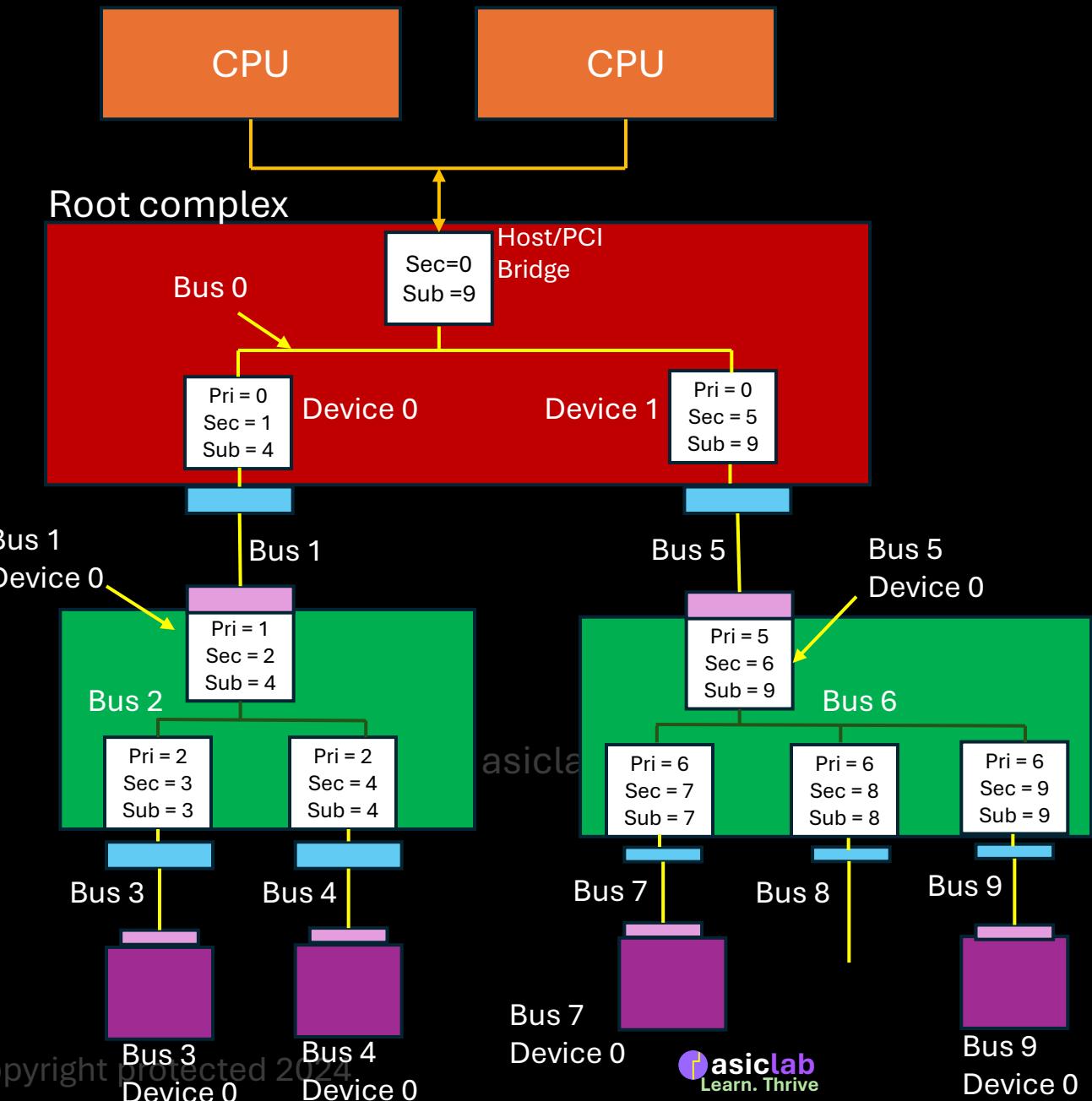
- Non-Posted
 - Config I/O Write
 - AtomicOp Request
 - Read Request



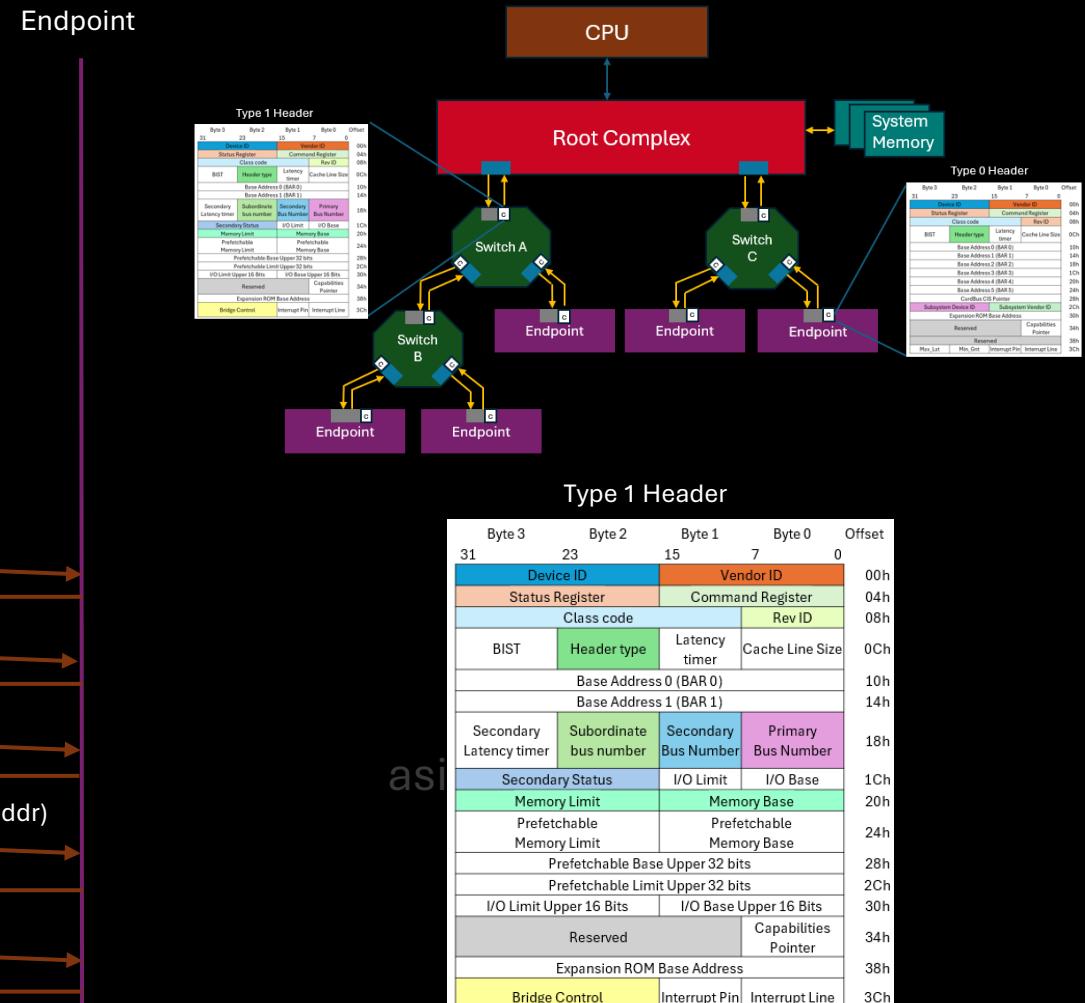
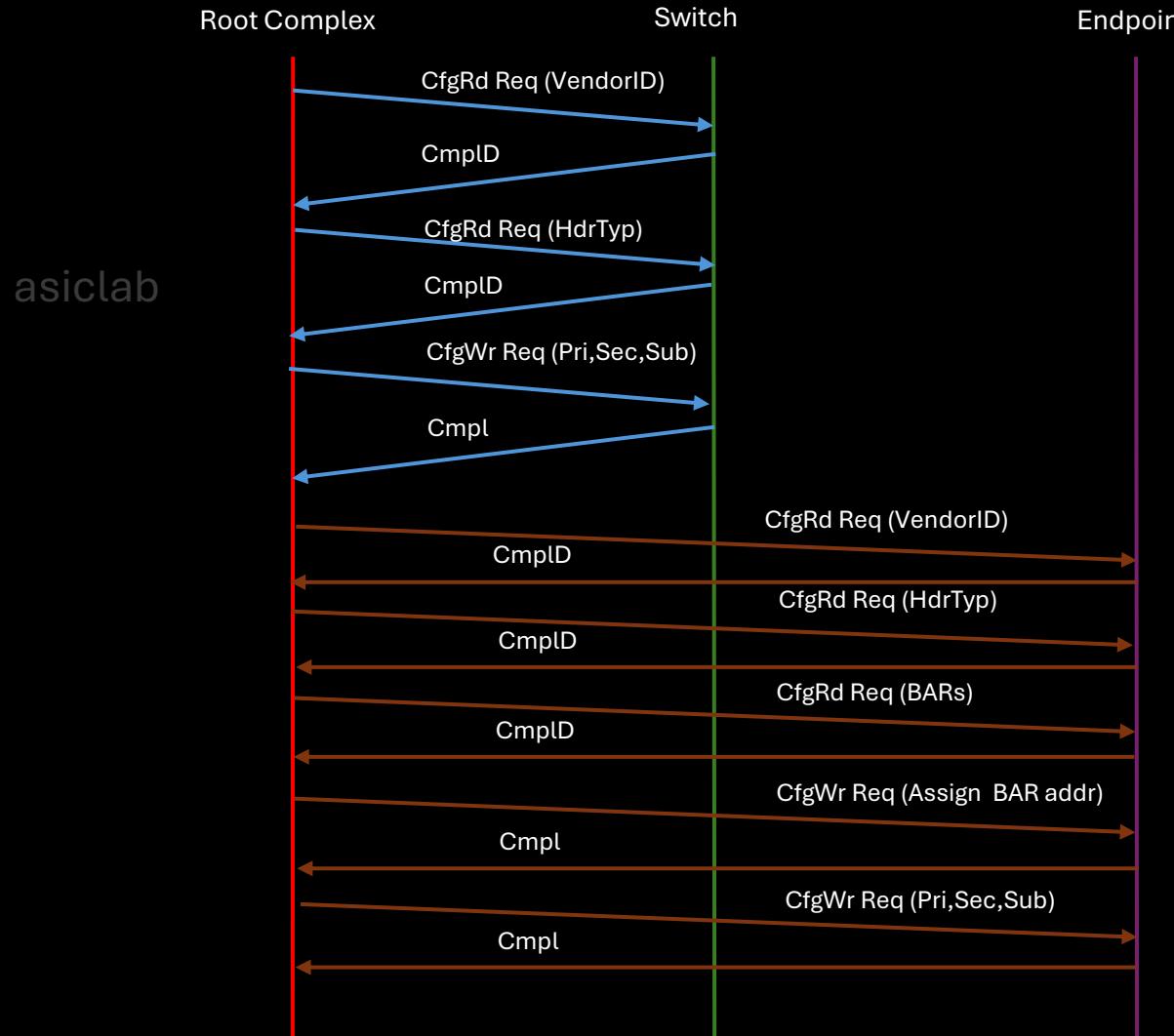
Sample Number	PCI-Express Packet	Direction	Sequence Number	Tag	Length	Register Number	Data	Payload	Completion Status
0	CfgRd_0	PCIe-101:Down	7A7	00	001	15	No data		
1	Ack	PCIe-102:Up	7A7						
1	CplD	PCIe-102:Up	809	00	001		With data 00000008	Successful Completion (SC)	
1	Ack	PCIe-101:Down	809						
2	CfgRd_0	PCIe-101:Down	7A8	00	001	01	No data		
2	Ack	PCIe-102:Up	7A8						
3	CplD	PCIe-102:Up	80A	00	001		With data 00000000	Successful Completion (SC)	
3	Ack	PCIe-101:Down	80A						
4	CfgWr_0	PCIe-101:Down	7A9	00	001	01	With data 79000003		
4	Ack	PCIe-102:Up	7A9						
5	Cpl	PCIe-102:Up	80B	00	000		No data	Successful Completion (SC)	

Enumeration Process

- Begin at Bus 0
- Read Vendor ID in each Function 0 of each possible Device on the Bus
 - If valid (not FFFFh) continue
 - If not – the Device doesn't exist
 - Continue to check next Device on bus
- Check Header Type
 - If Type 01h (Bridge) – continue downstream
- Update Device's Pri / Sec / Sub Registers and upstream buses sub
- Continue in a DFS manner



Enumeration Process

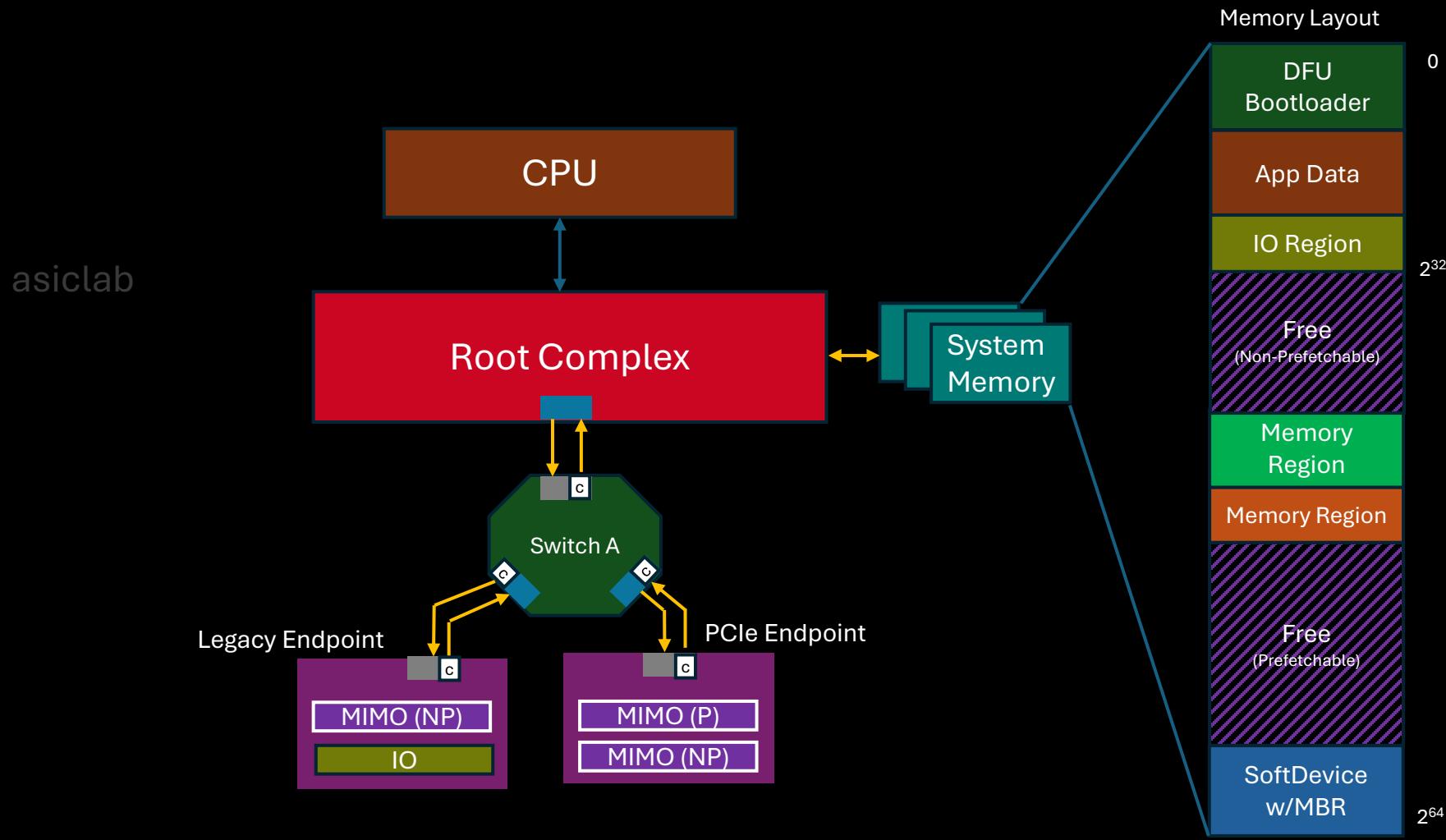


asiclab

Address Space and Transaction Routing

asiclab

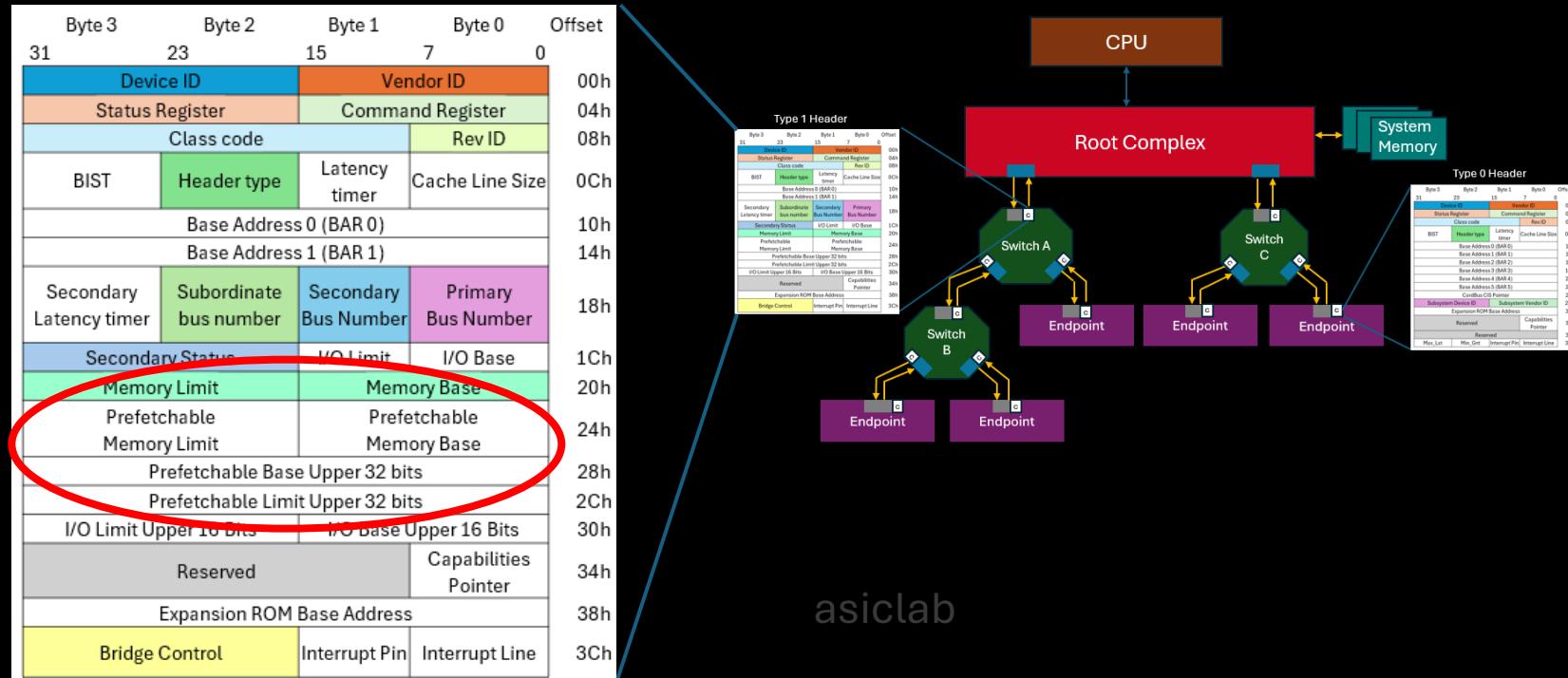
Memory and IO Space Address Maps



Memory and IO Space Address Maps

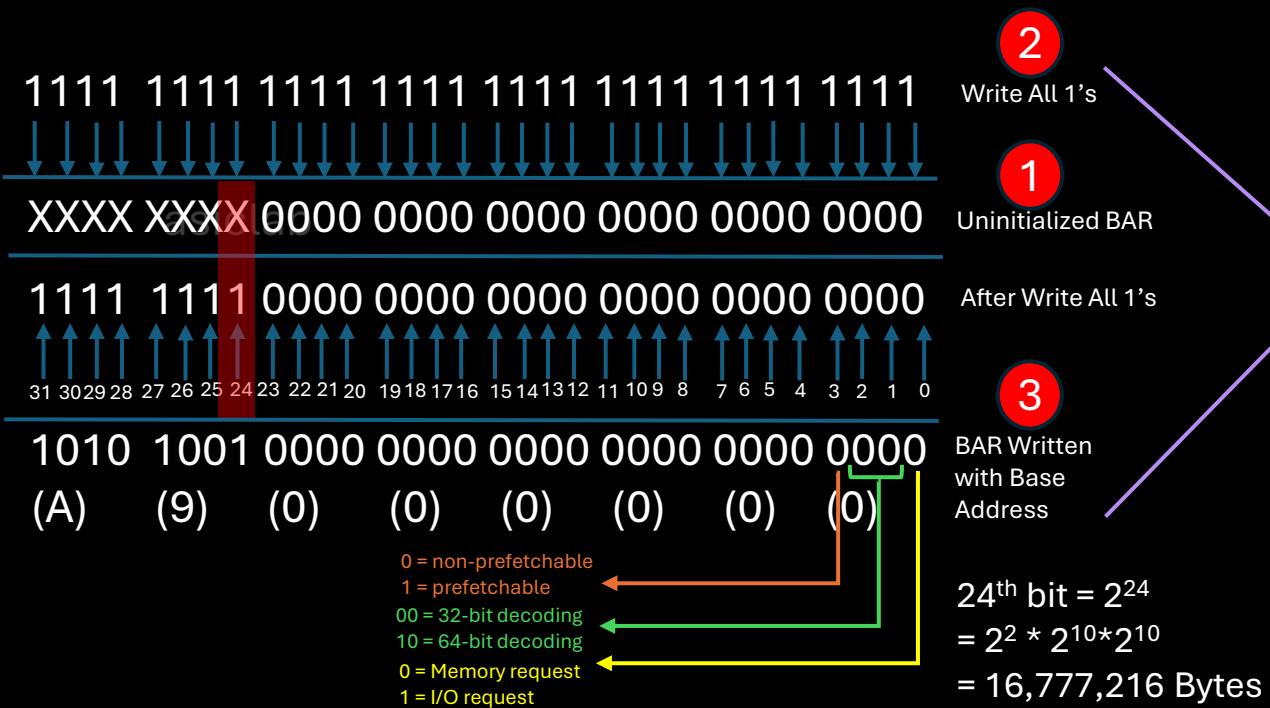
- **Prefetchable** space is safe to read ahead: a memory location can be read, the contents discarded, and the same content read again without problems
- **Non-Prefetchable** memory has side effects associated with reading from a location. Once read, the data is lost and cannot be read again.

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		
Status Register		Command Register		
Class code		Rev ID		
BIST	Header type	Latency timer	Cache Line Size	
Base Address 0 (BAR 0)				
Base Address 1 (BAR 1)				
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	
Secondary Status		I/O Limit	I/O Base	
Memory Limit		Memory Base		
Prefetchable Memory Limit		Prefetchable Memory Base		
Prefetchable Base Upper 32 bits				
Prefetchable Limit Upper 32 bits				
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		
Reserved		Capabilities Pointer		
Expansion ROM Base Address				
Bridge Control		Interrupt Pin	Interrupt Line	



PCI transactions don't define a transfer size. Bridges must guess the size on reads, and benefit from knowing whether the address range is prefetchable.

32-Bit Non-Prefetch Mem BAR Setup



The BAR calculation involves:

Step 1: Writing all 1s (0xFFFFFFFF) to the BAR register.

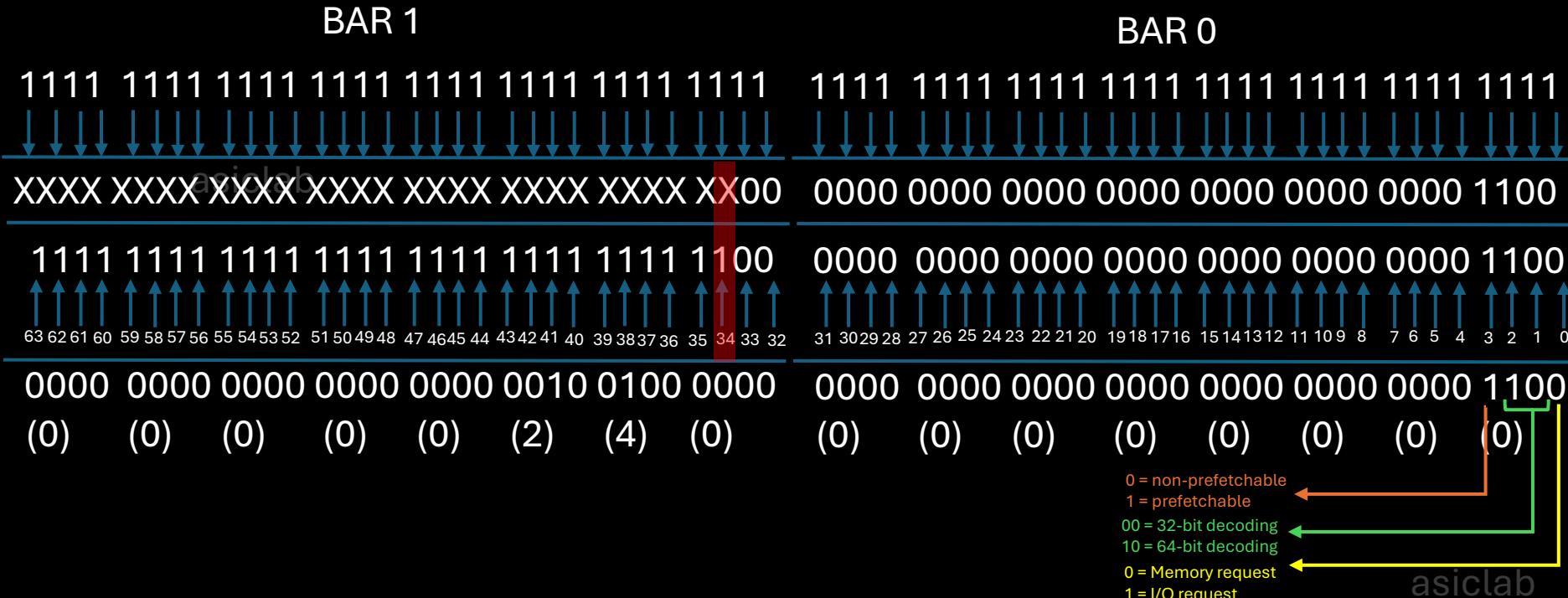
Step 2: Reading back the value and masking the non-address bits to get the size of the memory region. (Eg: `bar_value &= ~0xF;`)

Step 3: Subtracting the result from 0xFFFFFFFF and adding 1 to determine the actual size. (Eg: `unsigned int bar_size = (~bar_value + 1);`)

Type 0 Header

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
	Class code		Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
				10h
			Base Address 0 (BAR 0)	14h
			Base Address 1 (BAR 1)	18h
			Base Address 2 (BAR 2)	1Ch
			Base Address 3 (BAR 3)	20h
			Base Address 4 (BAR 4)	24h
			Base Address 5 (BAR 5)	28h
			CardBus CIS Pointer	2Ch
	Subsystem Device ID		Subsystem Vendor ID	30h
			Expansion ROM Base Address	34h
			Reserved	38h
			Capabilities Pointer	3Ch
	Max_Lat	Min_Gnt	Interrupt Pin	

64-Bit Prefetch Mem BAR Setup



The BAR calculation involves:

Step 1: Writing all 1s (0xFFFFFFFF) to the BAR register.

Step 2: Reading back the value and masking the non-address bits to get the size of the memory region. (Eg: bar_value &= ~0xF;)

Step 3: Subtracting the result from 0xFFFFFFFF and adding 1 to determine the actual size. (Eg: unsigned int bar_size = (~bar_value + 1);

64-Bit Prefetch Mem BAR Setup

Bit Index	Hex (64 bit)	Binary (64 bit)	Decimal Value	Memory Size
32	0x0000000010000000	00	4,29,49,67,296	4 GB
33	0x0000000020000000	00	8,58,99,34,592	8 GB
34	0x0000000040000000	00	17,17,98,69,184	16 GB
35	0x0000000080000000	00	34,35,97,38,368	32 GB
36	0x0000001000000000	00	68,71,94,76,736	64 GB
37	0x0000002000000000	00	1,37,43,89,53,472	128 GB
38	0x0000004000000000	00	2,74,87,79,06,944	256 GB
39	0x0000008000000000	00	5,49,75,58,13,888	512 GB
40	0x0000010000000000	00	10,99,51,16,27,776	1 TB
41	0x0000020000000000	00	21,99,02,32,55,552	2 TB
42	0x0000040000000000	00	43,98,04,65,11,104	4 TB
43	0x0000080000000000	00	87,96,09,30,22,208	8 TB
44	0x0000100000000000	00	1,75,92,18,60,44,416	16 TB
45	0x0000200000000000	00	3,51,84,37,20,88,832	32 TB
46	0x0000400000000000	00	7,03,68,74,41,77,664	64 TB
47	0x0000800000000000	00	14,07,37,48,83,55,328	128 TB
48	0x0001000000000000	00	28,14,74,97,67,10,656	256 TB
49	0x0002000000000000	00	56,29,49,95,34,21,312	512 TB
50	0x0004000000000000	00	1,12,58,99,90,68,42,620	1 PB
51	0x0008000000000000	00	2,25,17,99,81,36,85,240	2 PB
52	0x0010000000000000	00	4,50,35,99,62,73,70,490	4 PB
53	0x0020000000000000	00	9,00,71,99,25,47,40,990	8 PB
54	0x0040000000000000	00	18,01,43,98,50,94,81,900	16 PB
55	0x0080000000000000	00	36,02,87,96,01,89,63,900	32 PB
56	0x0100000000000000	00	72,05,75,94,03,79,27,900	64 PB
57	0x0200000000000000	00	1,44,11,51,88,07,58,55,000	128 PB
58	0x0400000000000000	00	2,88,23,03,76,15,17,11,000	256 PB
59	0x0800000000000000	00	5,76,46,07,52,30,34,23,000	512 PB
60	0x1000000000000000	00	11,52,92,15,04,60,68,40,000	1 EB
61	0x2000000000000000	001000	23,05,84,30,09,21,36,90,000	2 EB
62	0x4000000000000000	0100	46,11,68,60,18,42,73,80,000	4 EB
63	0x8000000000000000	1000	92,23,37,20,36,85,47,70,000	8 EB

Type 0 Header

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
Class code		Rev ID		08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Base Address 2 (BAR 2)				18h
Base Address 3 (BAR 3)				1Ch
Base Address 4 (BAR 4)				20h
Base Address 5 (BAR 5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID	Subsystem Vendor ID			2Ch
Expansion ROM Base Address				30h
Reserved		Capabilities Pointer		34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

32-Bit Prefetch Mem BAR Setup

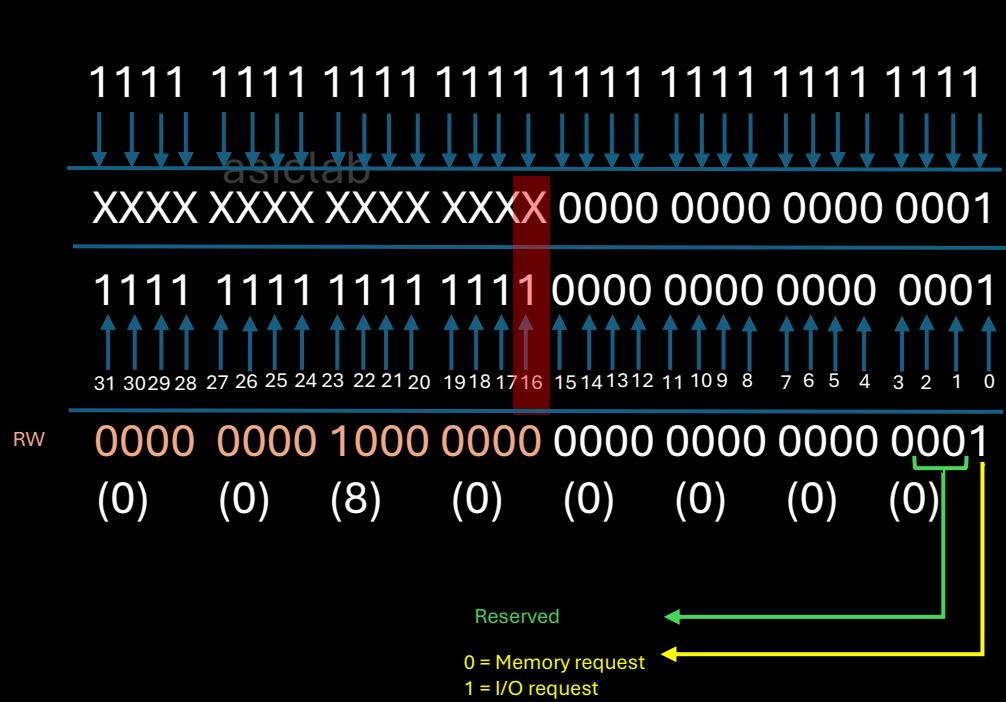
asic

Bit Index	Hex (32 bit)	Binary (32 bit)	Decimal	Bytes
0	0x00000001	00000000000000000000000000000001	1	1 byte
1	0x00000002	00000000000000000000000000000010	2	2 bytes
2	0x00000004	000000000000000000000000000000100	4	4 bytes
3	0x00000008	0000000000000000000000000000001000	8	8 bytes
4	0x00000010	00000000000000000000000000000010000	16	16 bytes
5	0x00000020	000000000000000000000000000000100000	32	32 bytes
6	0x00000040	0000000000000000000000000000001000000	64	64 bytes
7	0x00000080	00000000000000000000000000000010000000	128	128 bytes
8	0x00000100	00000000000000000000000000000010000000	256	256 bytes
9	0x00000200	000000000000000000000000000000100000000	512	512 bytes
10	0x00000400	0000000000000000000000000000001000000000	1,024	1 KB (Kilobyte)
11	0x00000800	00000000000000000000000000000010000000000	2,048	2 KB
12	0x00001000	00000000000000000000000000000010000000000	4,096	4 KB
13	0x00002000	00000000000000000000000000000010000000000	8,192	8 KB
14	0x00004000	00000000000000000000000000000010000000000	16,384	16 KB
15	0x00008000	00000000000000000000000000000010000000000	32,768	32 KB
16	0x00010000	00000000000000000000000000000010000000000	65,536	64 KB
17	0x00020000	00000000000000000000000000000010000000000	1,31,072	128 KB
18	0x00040000	00000000000000000000000000000010000000000	2,62,144	256 KB
19	0x00080000	00000000000000000000000000000010000000000	5,24,288	512 KB
20	0x00100000	00000000000000000000000000000010000000000	10,48,576	1 MB (Megabyte)
21	0x00200000	00000000000000000000000000000010000000000	20,97,152	2 MB
22	0x00400000	00000000000000000000000000000010000000000	41,94,304	4 MB
23	0x00800000	00000000000000000000000000000010000000000	83,88,608	8 MB
24	0x01000000	00000000000000000000000000000010000000000	1,67,77,216	16 MB
25	0x02000000	00000000000000000000000000000010000000000	3,35,54,432	32 MB
26	0x04000000	00000000000000000000000000000010000000000	6,71,08,864	64 MB
27	0x08000000	00000000000000000000000000000010000000000	13,42,17,728	128 MB
28	0x10000000	00000000000000000000000000000010000000000	26,84,35,456	256 MB
29	0x20000000	00100000000000000000000000000000	53,68,70,912	512 MB
30	0x40000000	01000000000000000000000000000000	1,07,37,41,824	1 GB (Gigabyte)
31	0x80000000	10000000000000000000000000000000	2,14,74,83,648	2 GB
32	0xFFFFFFF	11111111111111111111111111111111	4,29,49,67,295	4 GB

Type 1 Header

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
Class code		Rev ID		08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Base Address 2 (BAR 2)				18h
Base Address 3 (BAR 3)				1Ch
Base Address 4 (BAR 4)				20h
Base Address 5 (BAR 5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID	Subsystem Vendor ID			2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

IO BAR Setup



1
① Uninitialized BAR
2
② Write All 1's
3
③ BAR Written with Base Address

After Write All 1's
16th bit = 2^{16}
= $2^6 * 2^{10}$
= 65,536 Bytes
= 64 KB

Type 1 Header				
Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
		Class code	Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
				10h
		Base Address 0 (BAR 0)		14h
		Base Address 1 (BAR 1)		18h
		Base Address 2 (BAR 2)		1Ch
		Base Address 3 (BAR 3)		20h
		Base Address 4 (BAR 4)		24h
		Base Address 5 (BAR 5)		28h
		CardBus CIS Pointer		2Ch
	Subsystem Device ID	Subsystem Vendor ID		30h
		Expansion ROM Base Address		34h
		Reserved	Capabilities Pointer	38h
		Reserved		3Ch
	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line

The BAR calculation involves:

Step 1: Writing all 1s (0xFFFFFFFF) to the BAR register.

Step 2: Reading back the value and masking the non-address bits to get the size of the memory region. (Eg: `bar_value &= ~0xF;`)

Step 3: Subtracting the result from 0xFFFFFFFF and adding 1 to determine the actual size. (Eg: `unsigned int bar_size = (~bar_value + 1);`)

Resizable BARs - Motivation

- A **problem** arises when system DRAM and Function memory resources require more addressable space than the platform can give. This may result in:
 - Reduced space for system memory
 - Function memory not being allocated, or allocated with a sub-optimal size
- **Solution:** new registers report several possible memory sizes that will work
 - Only devices requesting large memory resources are likely to use this

Resizable BARs - Motivation

- Software learns which BAR sizes are available by reading that info from the new Resizable BAR Extended Capability Structure
- Software chooses optimal memory size for current platform conditions and programs the BAR size
- Hardware will then report the programmed BAR size when enumeration software queries the configuration header

asiclab

Configuration Space – Device Specification

https://www.nvidia.com/content/dam/en-zz/Solutions/gtcs22/data-center/h100/PB-11133-001_v01.pdf

Table 1. Product Specifications

Specification	NVIDIA H100
Product SKU	P1010 SKU 200 NVPN: 699-21010-0200-xxx
Total board power	PCIe 16-pin 450 W or 600 W power mode: <ul style="list-style-type: none">• 350 W default• 350 W maximum• 200 W minimum PCIe 16-pin 300 W power mode: <ul style="list-style-type: none">• 310 W default• 310 W maximum• 200 W minimum
Thermal solution	Passive
Mechanical form factor	Full-height, full-length (FHFL) 10.5", dual-slot
GPU SKU	GH100-200
PCI Device IDs	Device ID: 0x2331 Vendor ID: 0x10DE Sub-Vendor ID: 0x10DE Sub-System ID: 0x1626
GPU clocks	Base: 1,125 MHz Boost: 1,755 MHz
Performance states	P0
VBIOS	EEPROM size: 8 Mbit UEFI: Supported

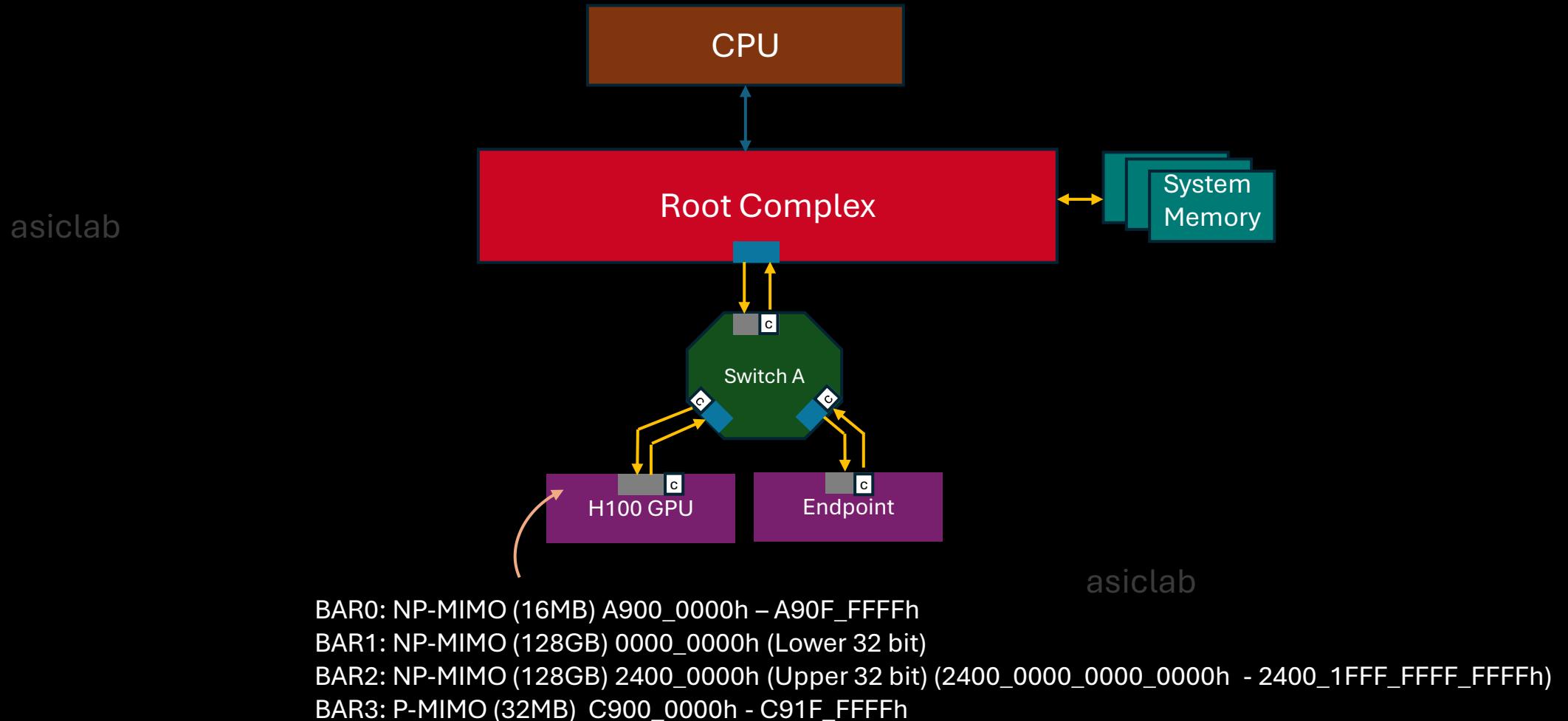
Specification	NVIDIA H100
PCI Express interface	PCI Express Gen5 x16; Gen5 x8; Gen4 x16 Lane and polarity reversal supported
Multi-Instance GPU (MIG)	Supported (seven instances)
Secure Boot (CEC)	Supported
Zero Power	Not supported
Power connectors and headers	One PCIe 16-pin auxiliary power connector
Weight	Board: 1200g grams (excluding bracket, extenders, and bridges) NVLink bridge: 20.5 grams per bridge (x 3 bridges) Bracket with screws: 20 grams Enhanced straight extender: 35 grams Long offset extender: 48 grams Straight extender: 32 grams

Table 3. Software Specifications

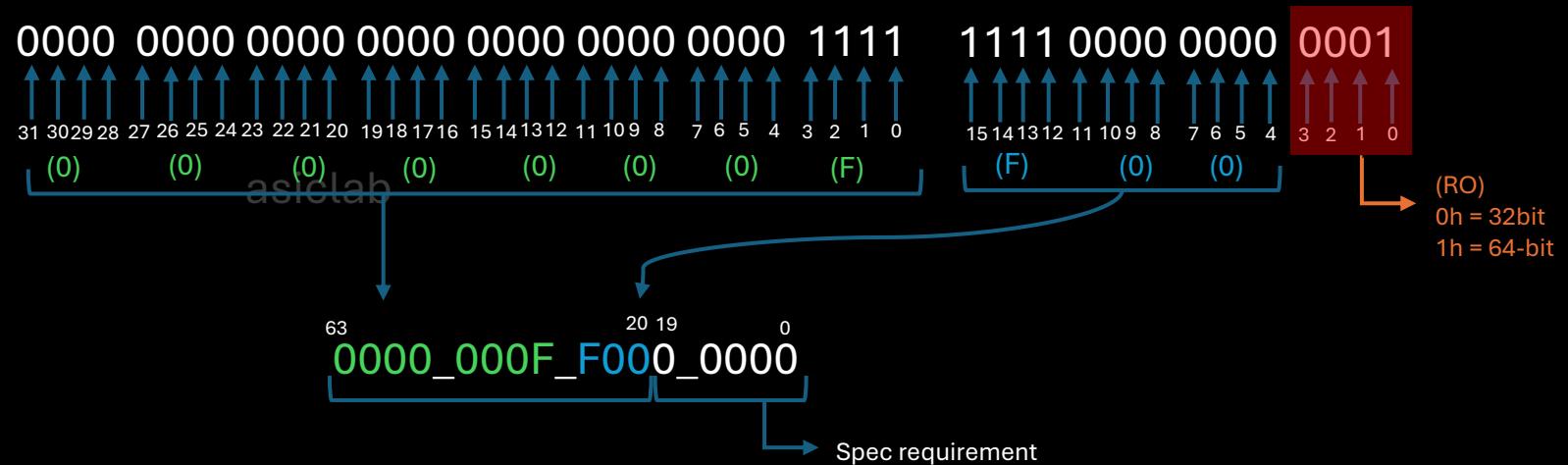
Specification	Description ¹
SR-IOV support	Supported -- 32 VF (virtual functions)
BAR address (physical function)	BAR0: 16 MiB ¹ BAR1: 128 GiB ¹ BAR3: 32 MiB ¹
BAR address (virtual function)	BAR0: 5 MiB, [256 KiB per VF] ¹ BAR1: 80 GiB, 64-bit [4 GiB per VF] ¹ BAR3: 640 MiB, 64-bit [32 MiB per VF] ¹
Message signaled interrupts	MSI-X: Supported MSI: Not supported
ARI Forwarding	Supported
Driver support	Linux: R520 or later Windows: R520 or later

GPU BAR Example:
BAR0: MMIO Registers
BAR1: VRAM aperture
BAR3: Indirect memory access

Example Topology For Base/Limit Setup



Example Prefetch Mem Base Setup

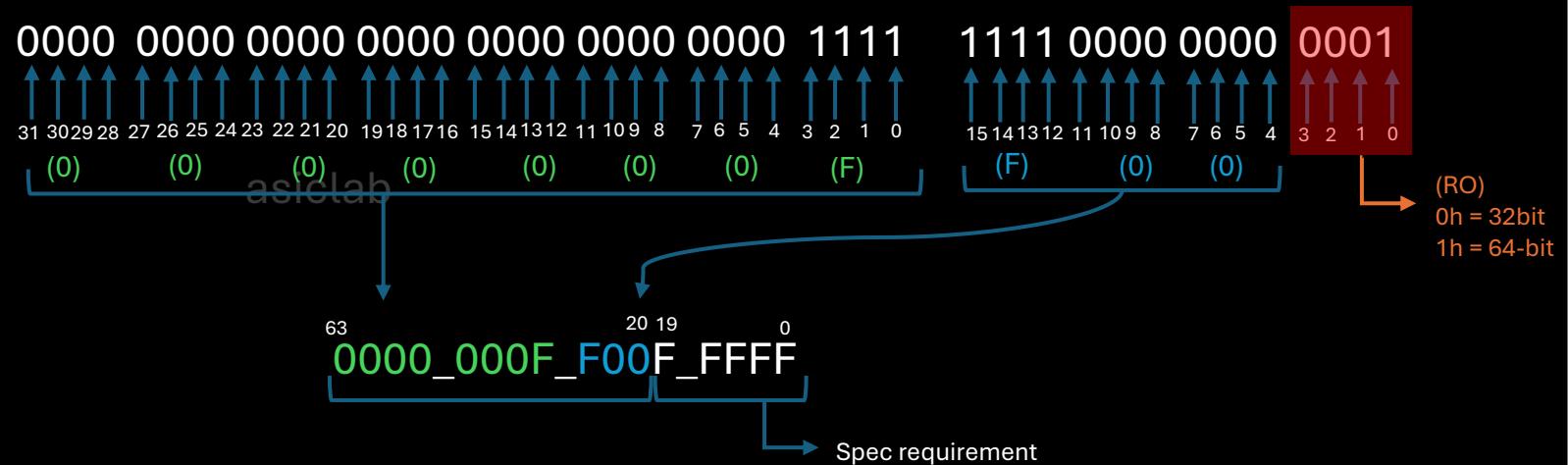


PCI Express Bridge Specification requirement: 5.1.2.8

- For the purpose of address decoding, the bridge assumes that the lower 20 address bits of the memory base address (not implemented in the Memory Base register) are zero.
- Similarly, the bridge assumes that the lower 20 address bits of the memory limit address (not implemented in the Memory Limit register) are F FFFFh.
- Thus, the bottom of the defined memory address range will be aligned to a 1-MB boundary and the top of the defined memory address range will be one less than a 1-MB boundary.

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID			Vendor ID	00h
Status Register		Command Register		04h
		Class code	Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
				10h
		Base Address 0 (BAR 0)		14h
		Base Address 1 (BAR 1)		18h
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	1Ch
				20h
Secondary Status		I/O Limit	I/O Base	24h
(Non-Prefetchable) Memory Limit		(Non-Prefetchable) Memory Base		28h
Prefetchable Memory Limit		Prefetchable Memory Base		2Ch
Prefetchable Base Upper 32 bits				30h
Prefetchable Limit Upper 32 bits				34h
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		38h
Reserved		Capabilities Pointer		3Ch
		Expansion ROM Base Address		
Bridge Control	Interrupt Pin	Interrupt Line		

Example Prefetch Mem Limit Setup

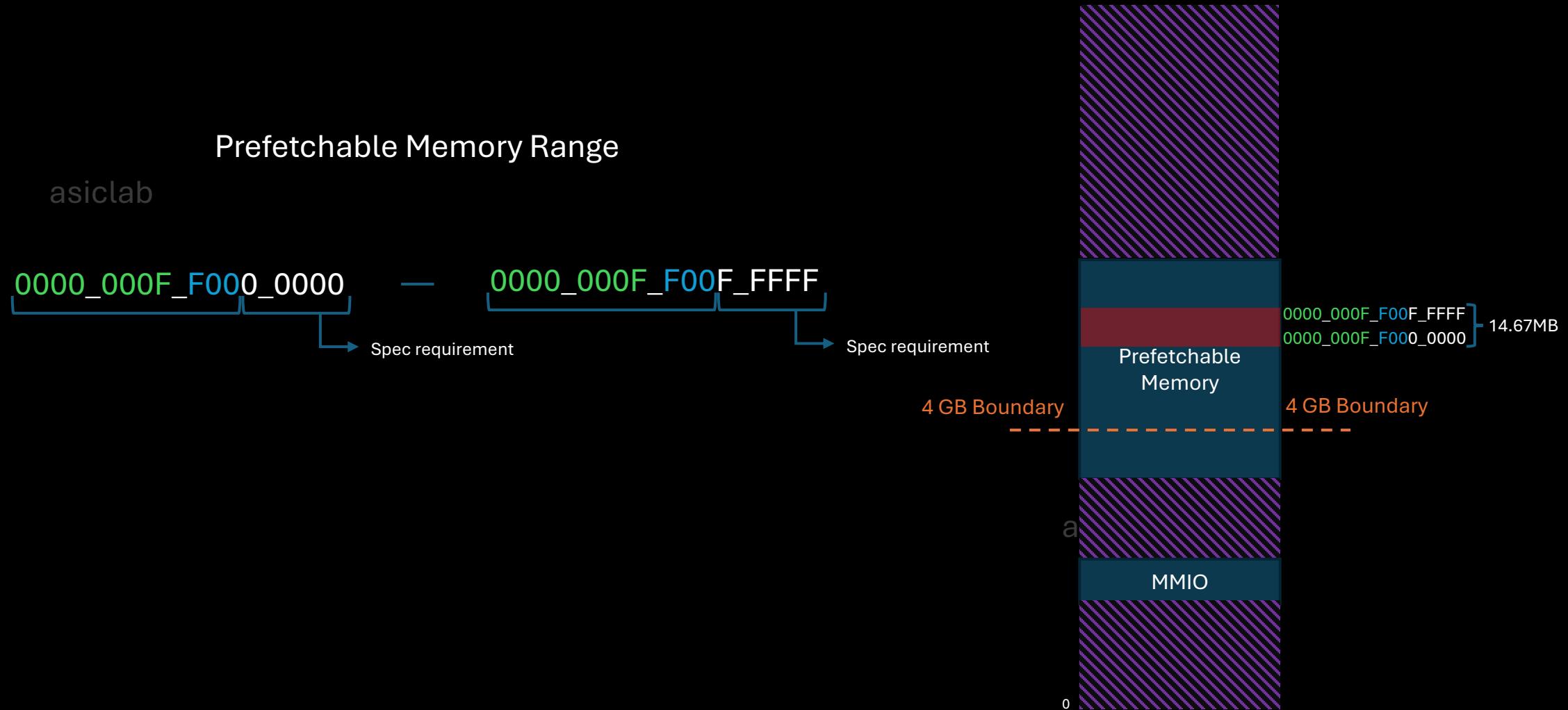


PCI Express Bridge Specification requirement: 5.1.2.8

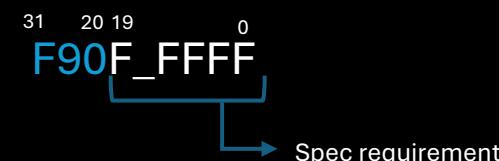
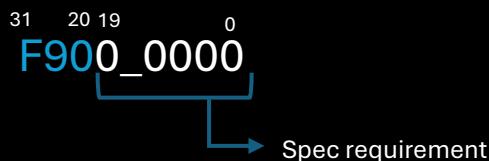
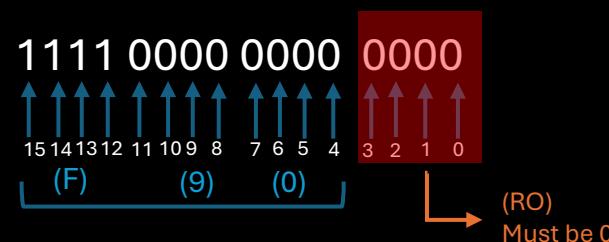
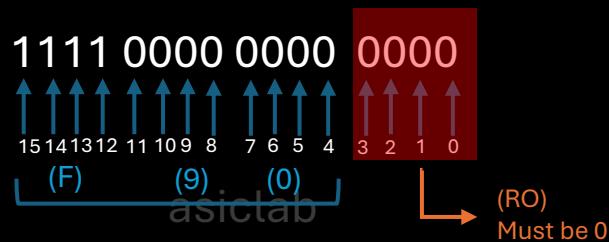
- For the purpose of address decoding, the bridge assumes that the lower 20 address bits of the memory base address (not implemented in the Memory Base register) are zero.
- Similarly, the bridge assumes that the lower 20 address bits of the memory limit address (not implemented in the Memory Limit register) are F FFFFh.
- Thus, the bottom of the defined memory address range will be aligned to a 1-MB boundary and the top of the defined memory address range will be one less than a 1-MB boundary.

	Byte 3	Byte 2	Byte 1	Byte 0	Offset		
31	Device ID	Vendor ID			00h		
23	Status Register	Command Register			04h		
15	Class code			Rev ID	08h		
7	BIST	Header type	Latency timer	Cache Line Size	0Ch		
0	Base Address 0 (BAR 0)				10h		
	Base Address 1 (BAR 1)				14h		
	Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	18h		
	Secondary Status		I/O Limit	I/O Base	1Ch		
	(Non-Prefetchable) Memory Limit		(Non-Prefetchable) Memory Base		20h		
	Prefetchable Memory Limit		Prefetchable Memory Base		24h		
	Prefetchable Base Upper 32 bits				28h		
	Prefetchable Limit Upper 32 bits				2Ch		
	I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30h		
	Reserved			Capabilities Pointer	34h		
	Expansion ROM Base Address				38h		
	Bridge Control		Interrupt Pin	Interrupt Line	3Ch		

Example Prefetch Mem Base/Limit Setup



Example Non-Prefetch Mem Base/Limit Setup



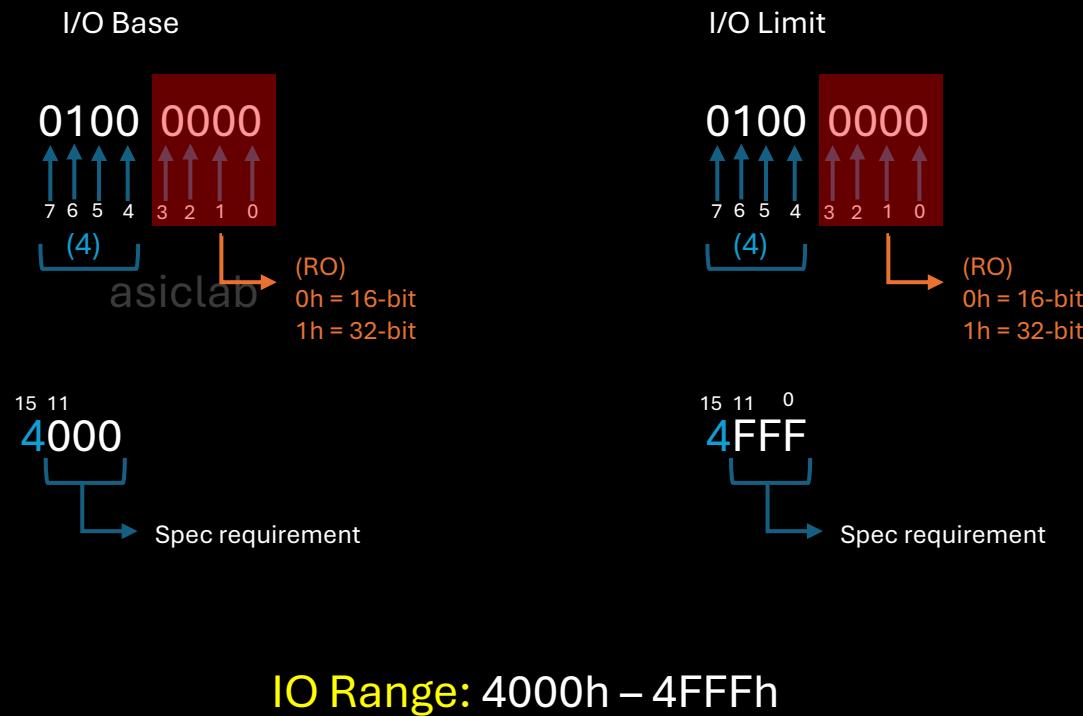
Non-prefetchable Memory Range: F900_0000h – F90F_FFFFh

PCI Express Bridge Specification requirement: 5.1.2.8

- For the purpose of address decoding, the bridge assumes that the lower 20 address bits of the memory base address (not implemented in the Memory Base register) are zero.
- Similarly, the bridge assumes that the lower 20 address bits of the memory limit address (not implemented in the Memory Limit register) are F FFFFh.
- Thus, the bottom of the defined memory address range will be aligned to a 1-MB boundary and the top of the defined memory address range will be one less than a 1-MB boundary.

Type 1 Header				
Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
	Class code		Rev ID	08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
				10h
		Base Address 0 (BAR 0)		14h
		Base Address 1 (BAR 1)		
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	18h
				1Ch
Secondary Status		I/O Limit	I/O Base	20h
(Non-Prefetchable) Memory Limit		(Non-Prefetchable) Memory Base		
Prefetchable Memory Limit		Prefetchable Memory Base		24h
		Prefetchable Base Upper 32 bits		28h
		Prefetchable Limit Upper 32 bits		2Ch
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30h
		Reserved	Capabilities Pointer	34h
				38h
		Expansion ROM Base Address		3Ch
		Bridge Control	Interrupt Pin	Interrupt Line

Example IO Base/Limit Setup

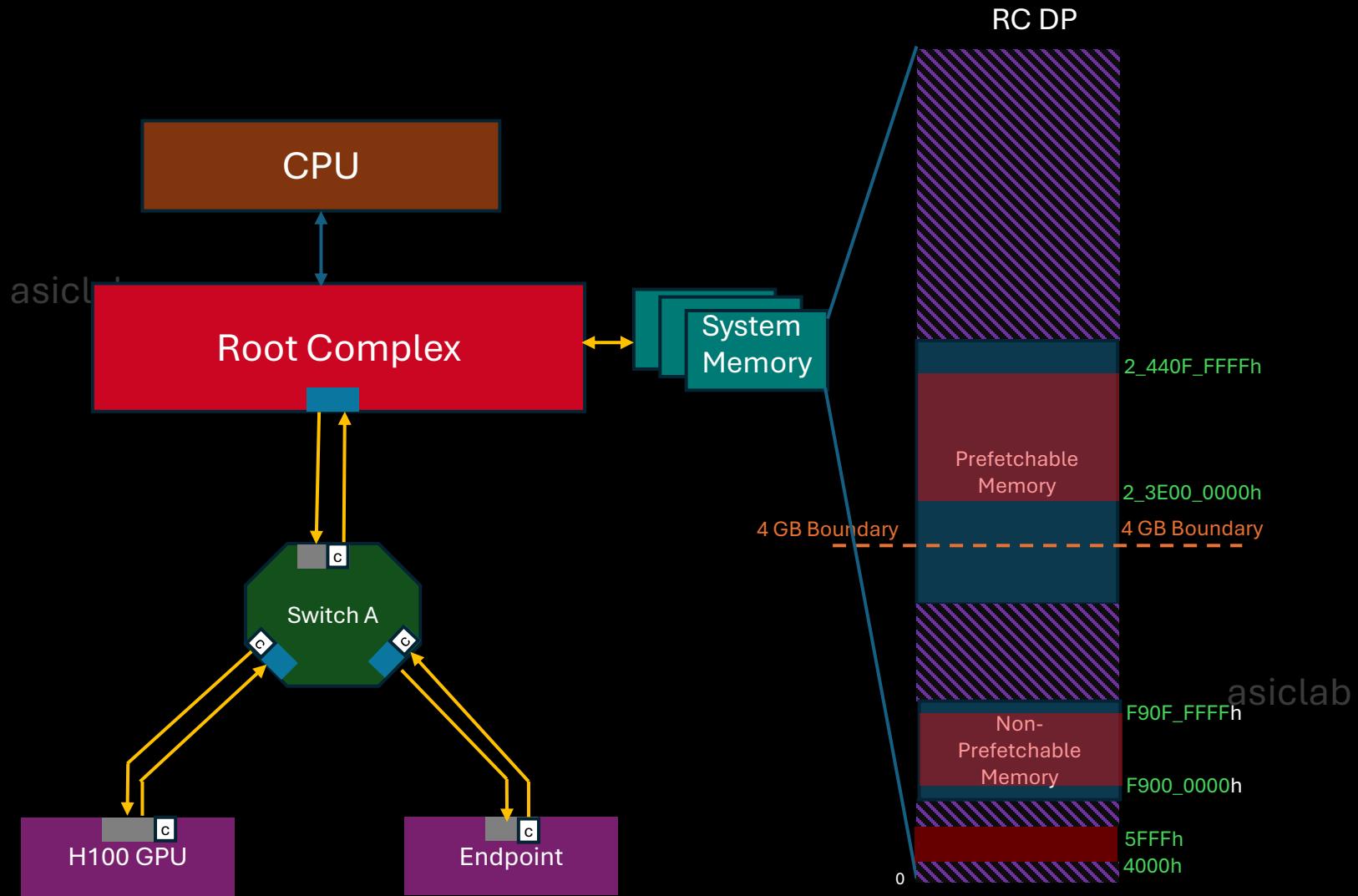


PCI Express Bridge Specification requirement: 5.1.2.8

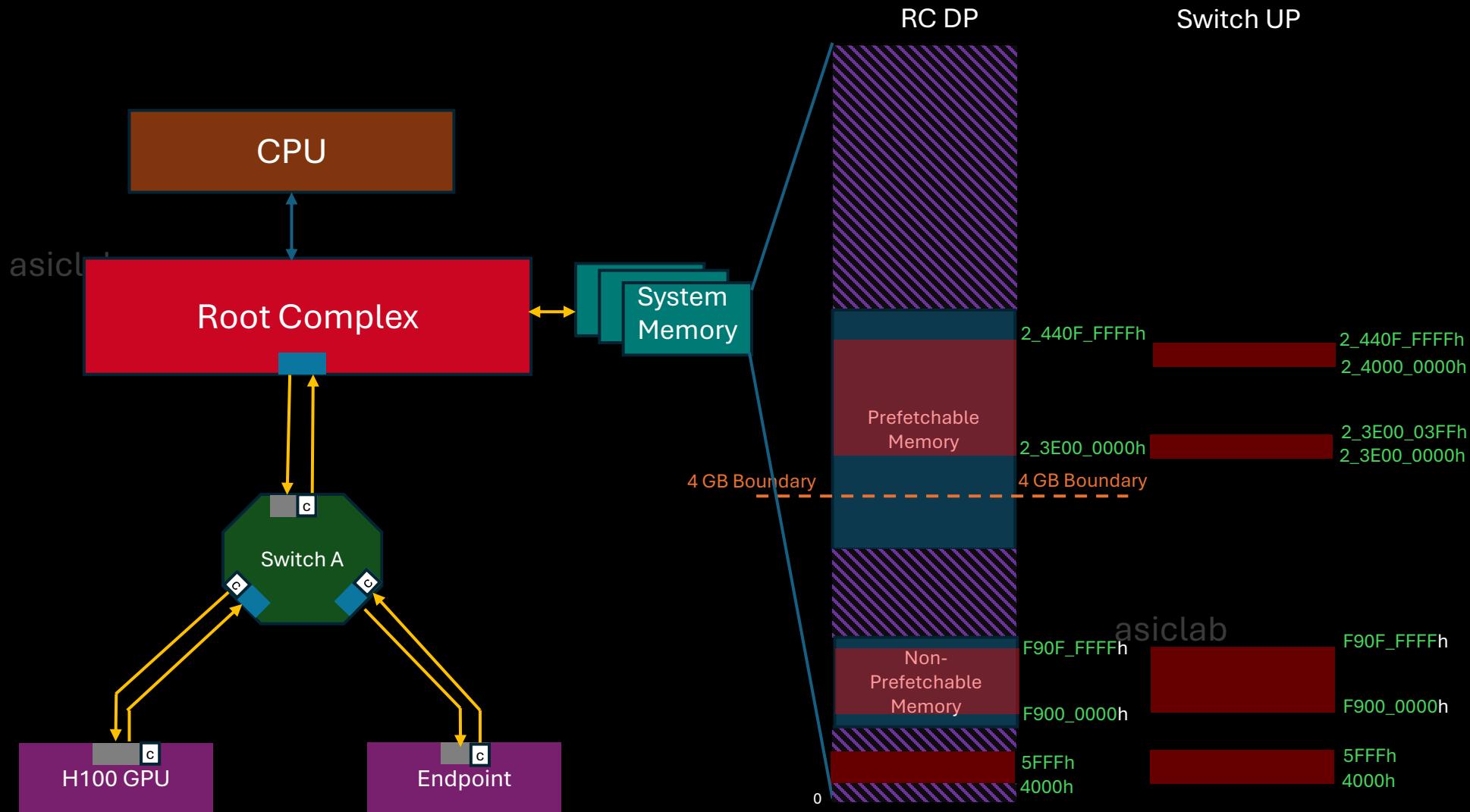
- For the purpose of address decoding, the bridge assumes that the lower 20 address bits of the memory base address (not implemented in the Memory Base register) are zero.
- Similarly, the bridge assumes that the lower 20 address bits of the memory limit address (not implemented in the Memory Limit register) are F FFFFh.
- Thus, the bottom of the defined memory address range will be aligned to a 1-MB boundary and the top of the defined memory address range will be one less than a 1-MB boundary.

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
Device ID		Vendor ID		00h
Status Register		Command Register		04h
Class code				08h
BIST	Header type	Latency timer	Cache Line Size	0Ch
Base Address 0 (BAR 0)				10h
Base Address 1 (BAR 1)				14h
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	18h
Secondary Status		I/O Limit	I/O Base	1Ch
(Non-Prefetchable) Memory Limit		(Non-Prefetchable) Memory Base		20h
Prefetchable Memory Limit		Prefetchable Memory Base		24h
Prefetchable Base Upper 32 bits				28h
Prefetchable Limit Upper 32 bits				2Ch
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30h
Reserved			Capabilities Pointer	34h
Expansion ROM Base Address				38h
Bridge Control		Interrupt Pin	Interrupt Line	3Ch

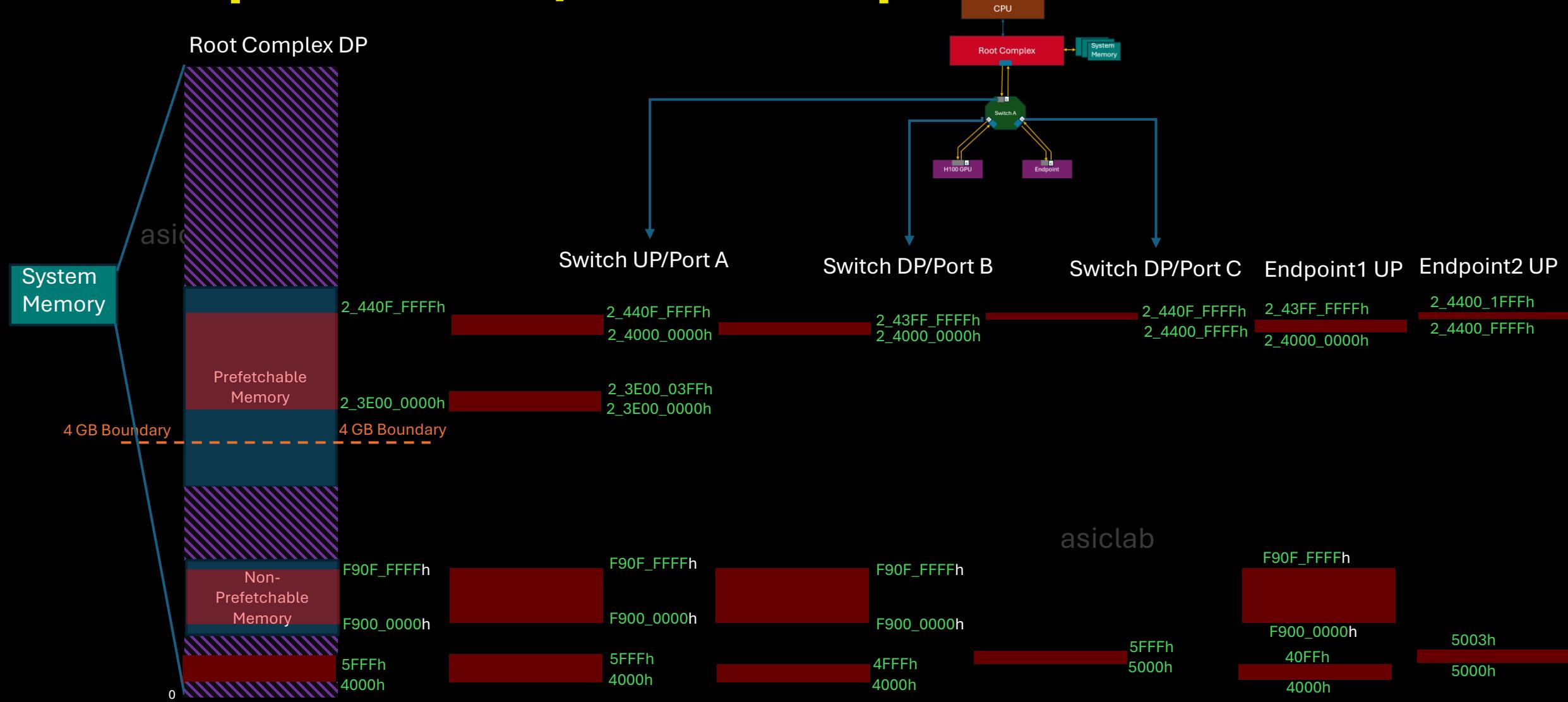
Example Register Setup



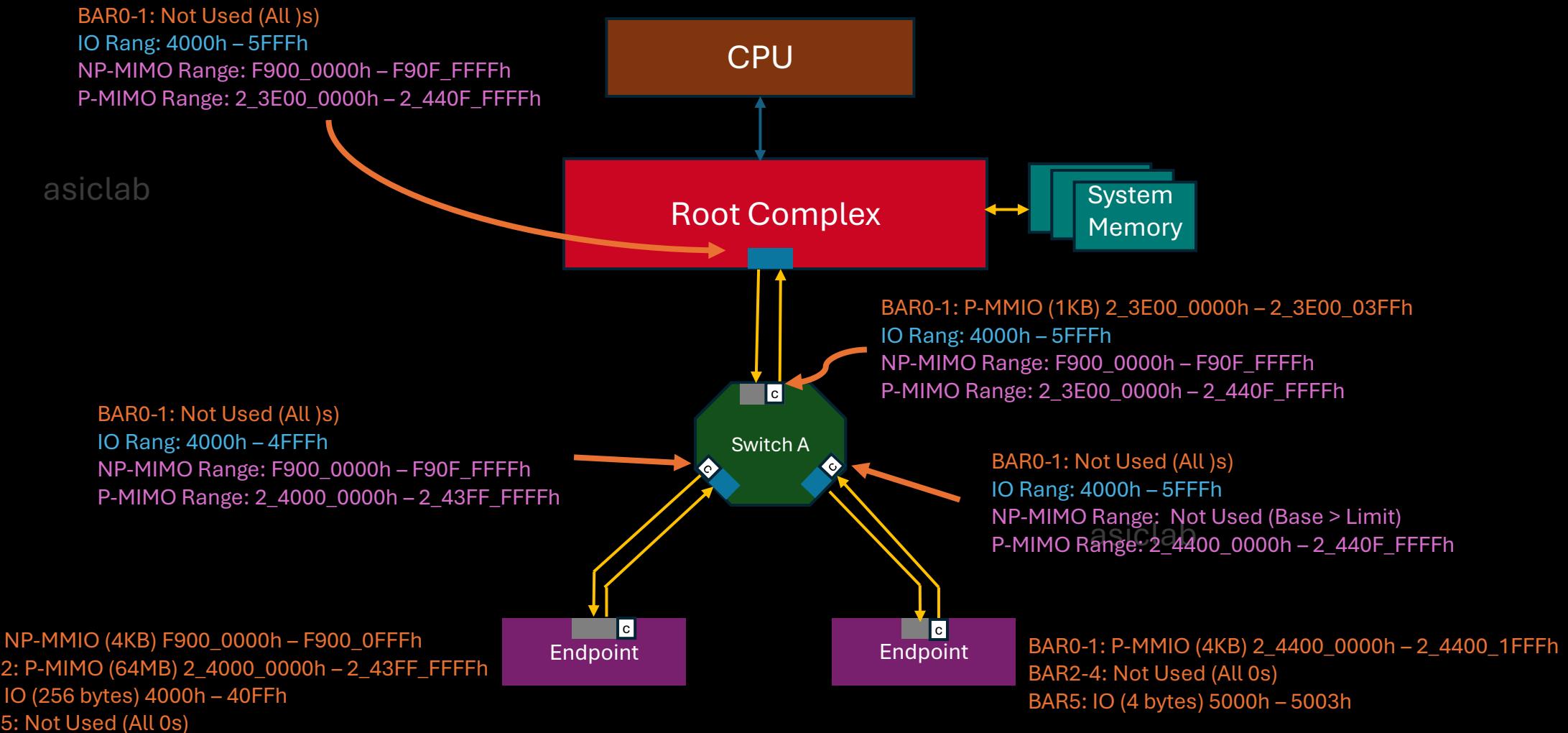
Example Register Setup



Example IO Base/Limit Setup

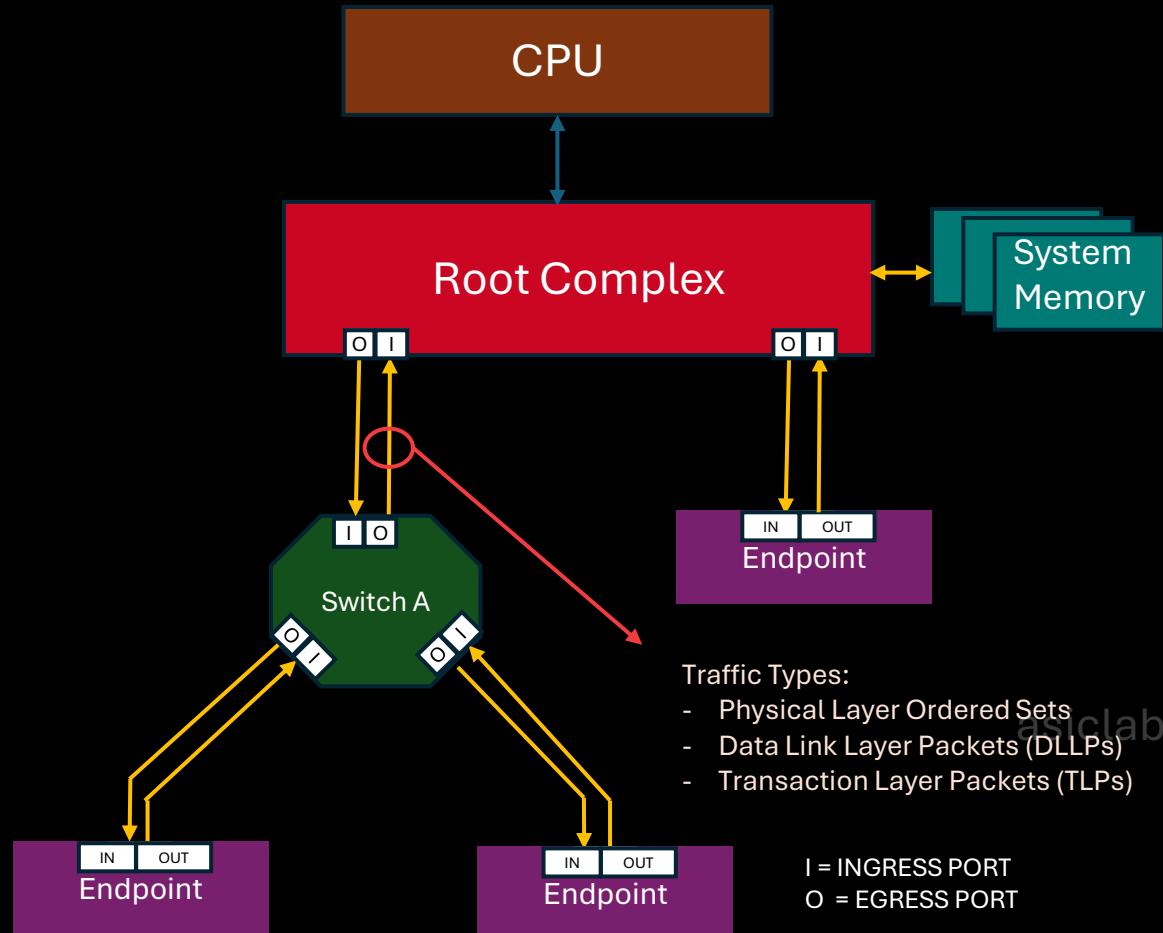


Example Register Setup



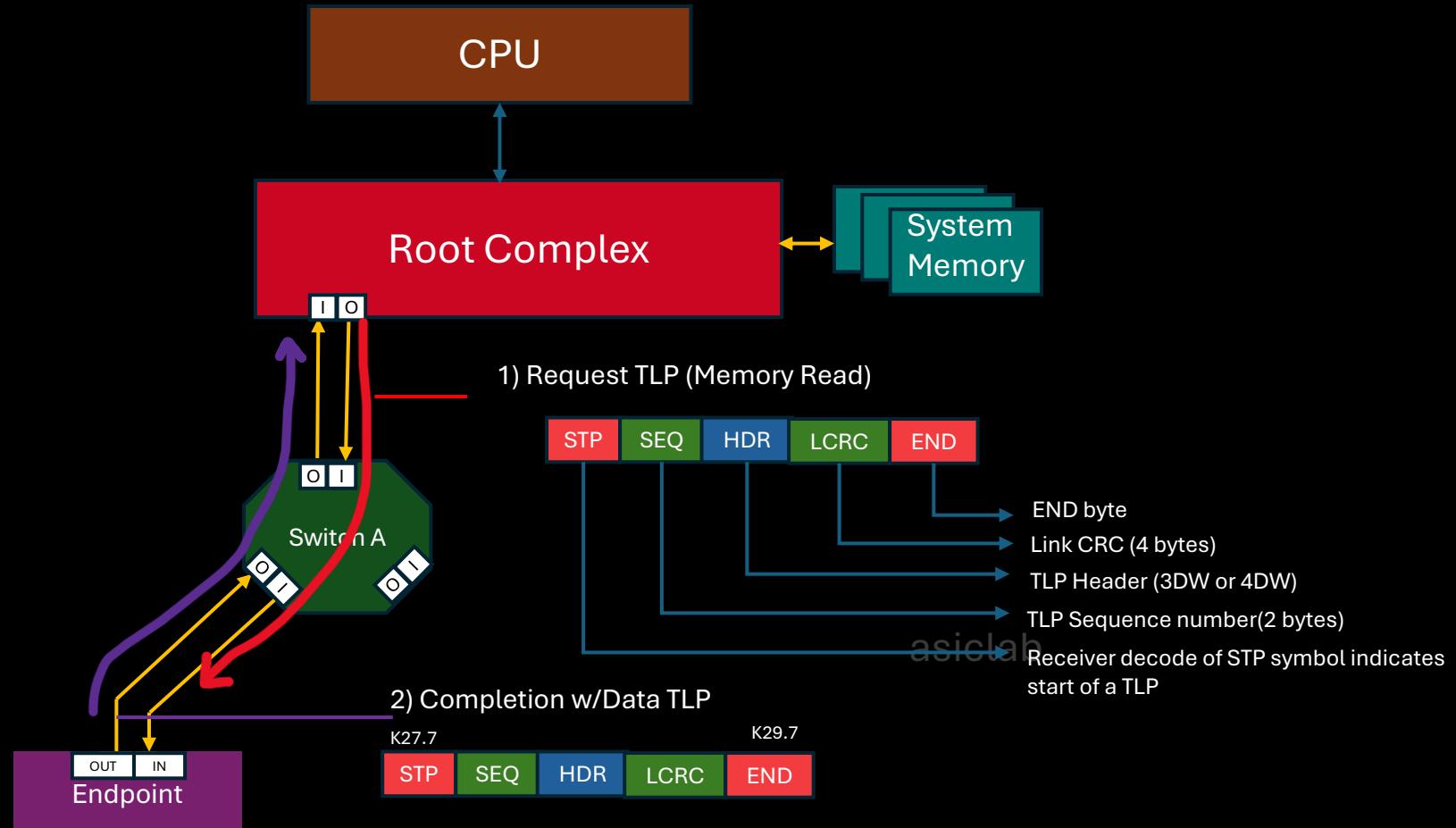
Switch Routing

asiclab



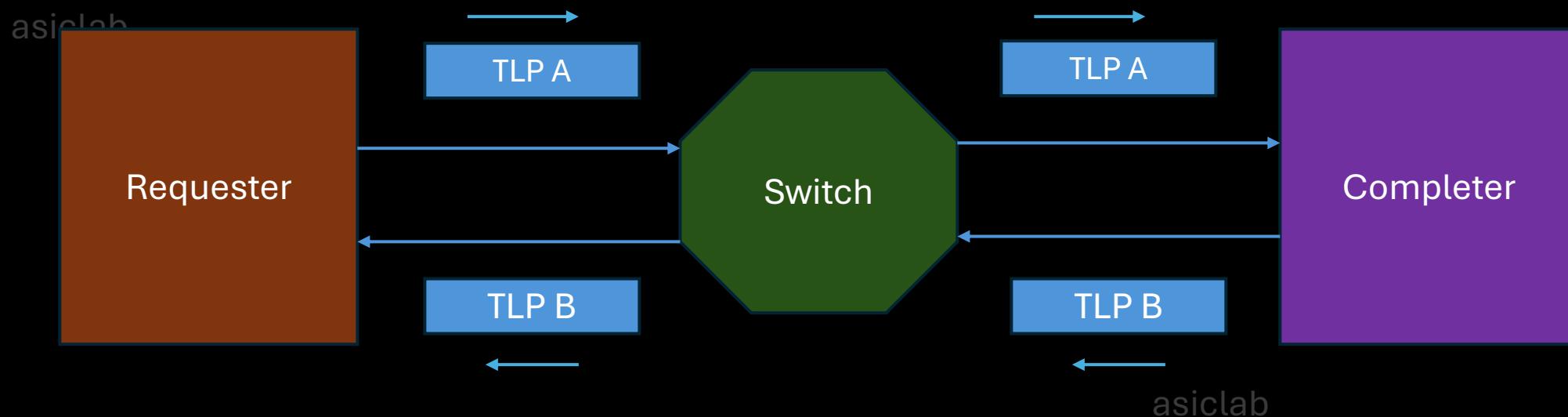
Non-Posted (Split) Transaction Routing

asiclab



Link Traffic Routed Through Fabric

- Traffic types on each Link:
 - Transaction Layer Packets (TLPs)
 - Data Link Layer Packets (DLLPs)
 - Ordered Sets



Note: Only TLPs are routed. DLLPs and Ordered Sets are never routed to another Link because they're only used to manage the local link.

Four Transaction Types

- Four basic transaction types:
 - Memory
 - I/O
 - Configuration
 - Message
- Switches are not permitted to split a large packet into multiple smaller ones, but a Root Complex that supports peer-to-peer traffic is allowed to split them according to packet format rules

asiclab

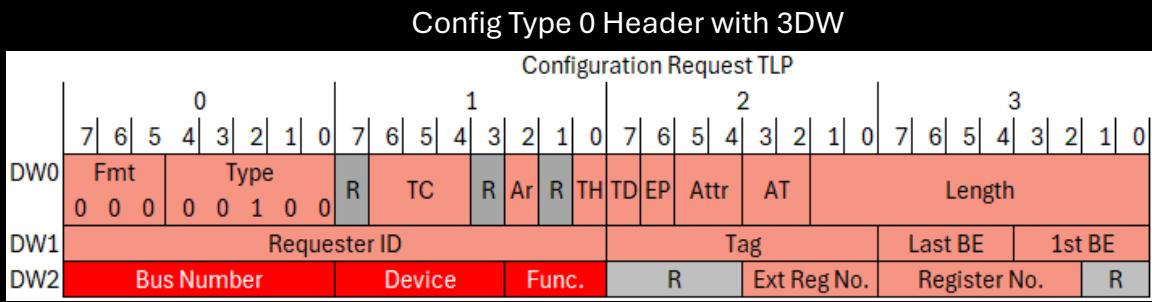
Three Methods of Packet Routing

- Every TLP is routed based on one of these:
 - Address Routing
 - ID Routing
 - Implicit Routing

Packet Type	Routing Method
MRd, MRdLk, MWr	Address Routing
IORd, IOWr	Address Routing
CfgRd0, CfgWr0, CfgRd1, CfgWr1	ID Routing
Cpl, CplD, CplLk	ID Routing
Msg, MsgD	Implicit, Address or ID
FetchAdd, Swap, CAS	Address Routing

Method 1: ID Routing (3DW)

asiclab



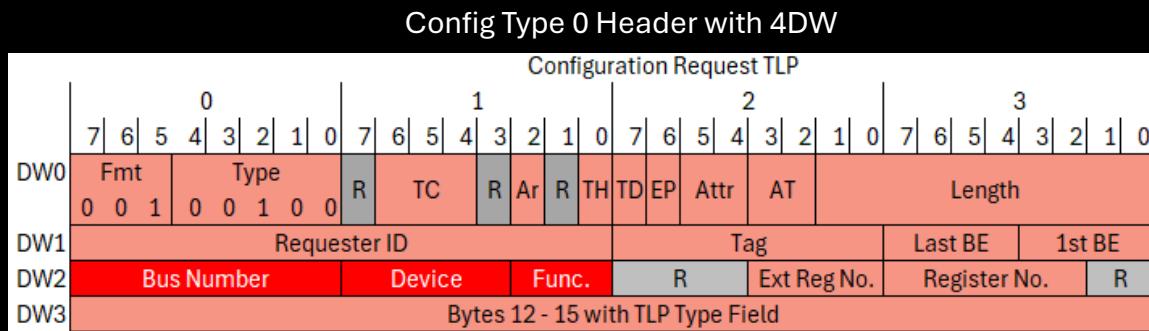
Encoding	Description
000b	TLP, 3DW Header, no data
001b	TLP, 4DW Header, no data
010b	TLP, 3DW Header, with data
011b	TLP, 4DW Header, with data
100b	Prefix DW

Encoding	Description
00000b	Memory Request
00001b	Memory Read Lock Request
11011b	Deferrable Memory Write Request
00010b	IO Request
00100b	Configuration Type 0 Request
00101b	Configuration Type 1 Request
10 r r r b	Message Request
01010b	Completion
01011b	Completion Lock
01100b	AtomicOP: Fetch and Add Request
01101b	AtomicOP: Unconditional Swap Request
01110b	AtomicOP: Compare and Swap Request
0L3L2L1L0	Locl TLP Prefix (Fmt[2:0] = 100b)
1E3E2E1E0	End-to-End TLP Prefix (Fmt[2:0]= 100b)

asiclab

Method 1: ID Routing (4DW)

asiclab

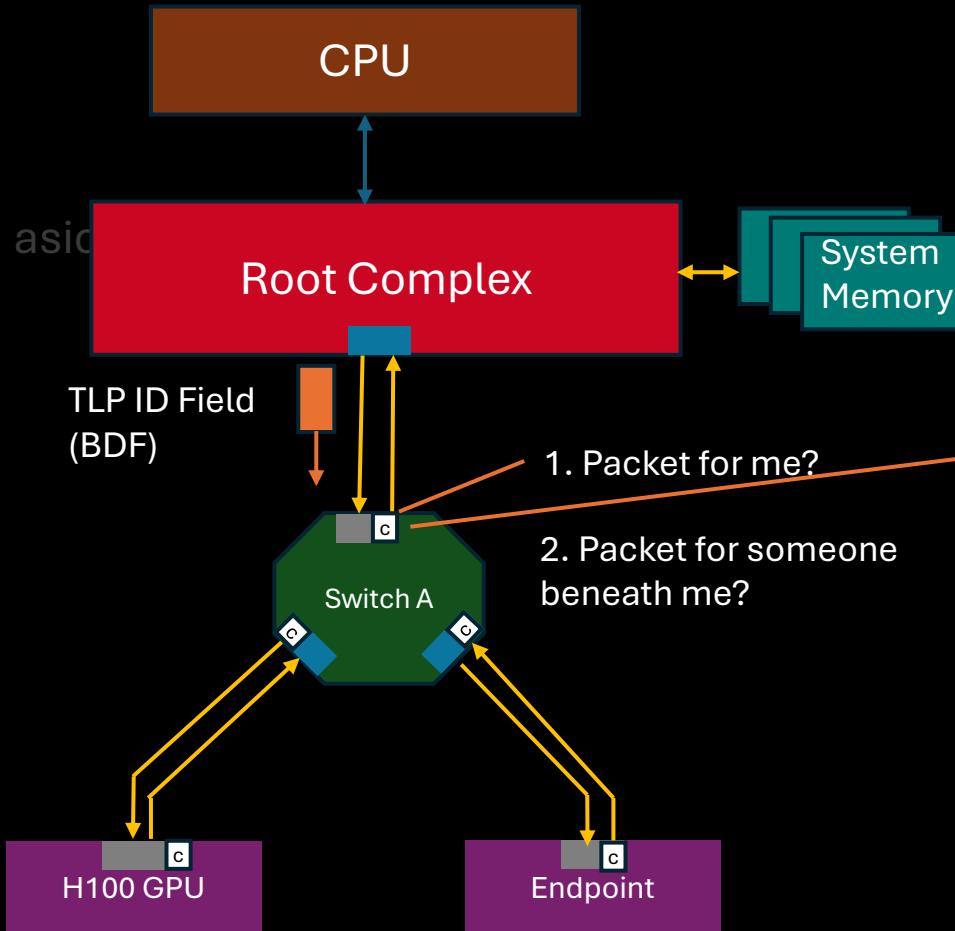


Encoding	Description
000b	TLP, 3DW Header, no data
001b	TLP, 4DW Header, no data
010b	TLP, 3DW Header, with data
011b	TLP, 4DW Header, with data
100b	Prefix DW

Encoding	Description
00000b	Memory Request
00001b	Memory Read Lock Request
11011b	Deferrable Memory Write Request
00010b	IO Request
00100b	Configuration Type 0 Request
00101b	Configuration Type 1 Request
10rrrb	Message Request
01010b	Completion
01011b	Completion Lock
01100b	AtomicOP: Fetch and Add Request
01101b	AtomicOP: Unconditional Swap Request
01110b	AtomicOP: Compare and Swap Request
0L3L2L1L0	Locl TLP Prefix (Fmt[2:0] = 100b)
1E3E2E1E0	End-to-End TLP Prefix (Fmt[2:0]= 100b)

asiclab

Method 1: ID Routing; Switch Check

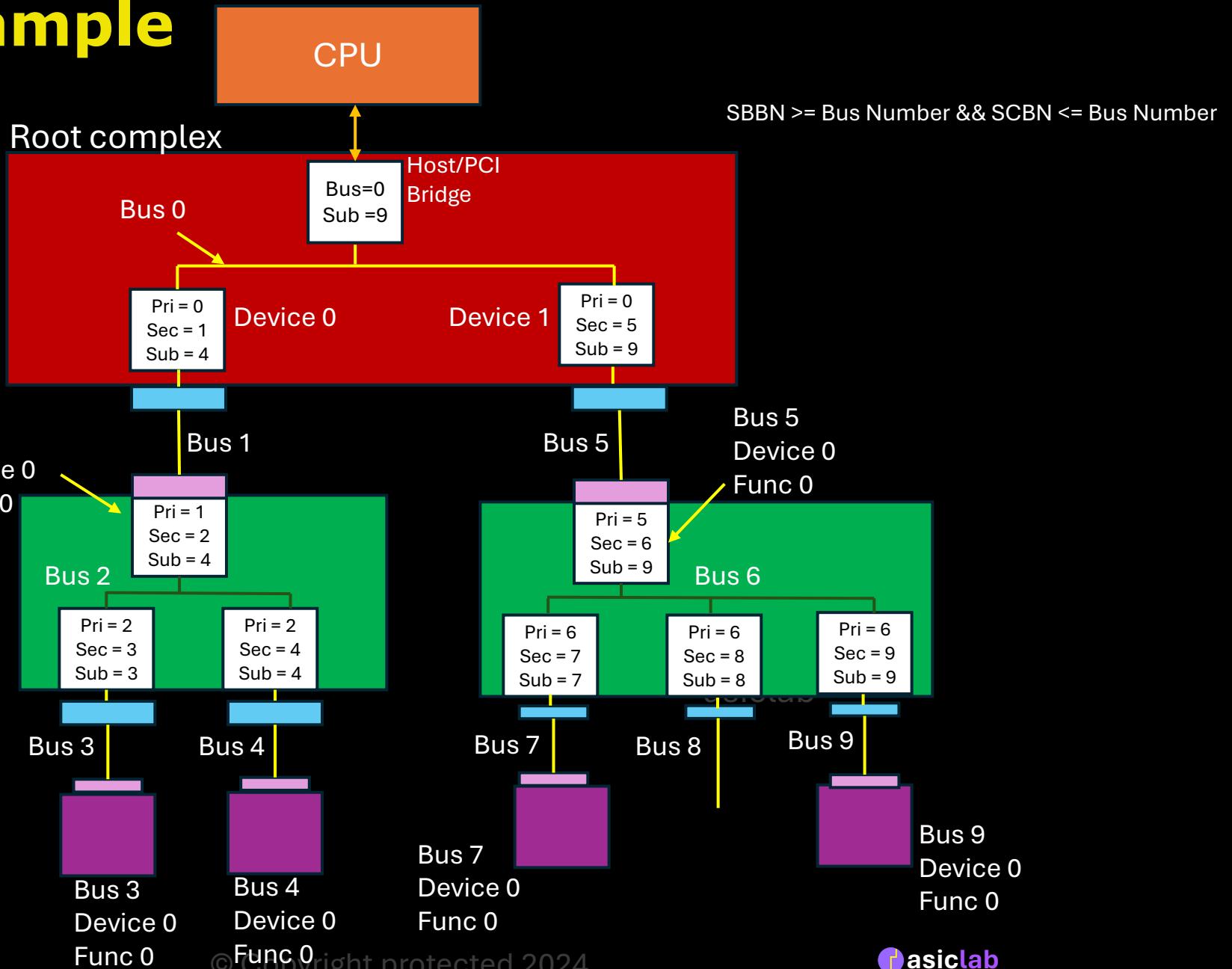


Type 1 Header

Byte 3	Byte 2	Byte 1	Byte 0	Offset
31	23	15	7	0
				00h
		Device ID	Vendor ID	04h
	Status Register	Command Register		08h
	Class code		Rev ID	0Ch
BIST	Header type	Latency timer	Cache Line Size	10h
		Base Address 0 (BAR 0)		14h
		Base Address 1 (BAR 1)		
Secondary Latency timer	Subordinate bus number	Secondary Bus Number	Primary Bus Number	18h
				1Ch
Secondary Status	I/O Limit	I/O Base		20h
(Non-Prefetchable) Memory Limit	(Non-Prefetchable) Memory Base			24h
Prefetchable Memory Limit	Prefetchable Memory Base			28h
	Prefetchable Base Upper 32 bits			2Ch
	Prefetchable Limit Upper 32 bits			30h
I/O Limit Upper 16 Bits	I/O Base Upper 16 Bits			34h
	Reserved	Capabilities Pointer		38h
	Expansion ROM Base Address			3Ch
	Bridge Control	Interrupt Pin	Interrupt Line	

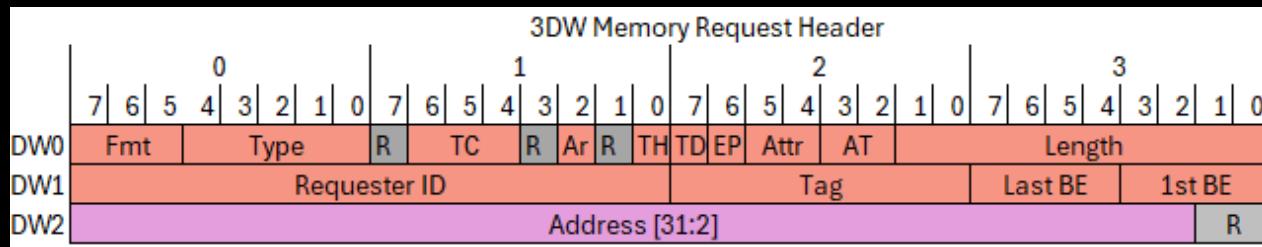
Method 1: Example

asiclab



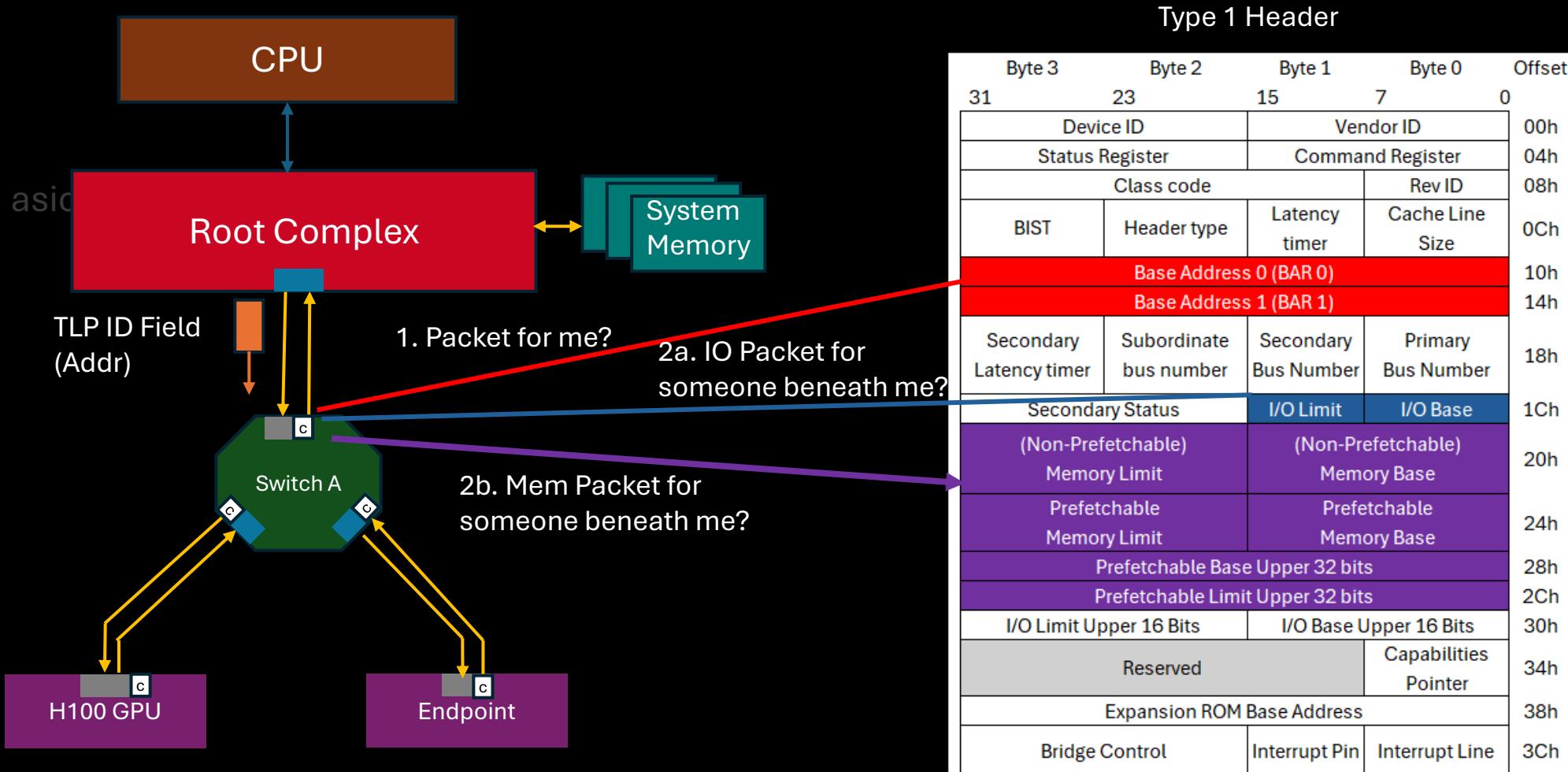
Method 2: Address Routing

asiclab

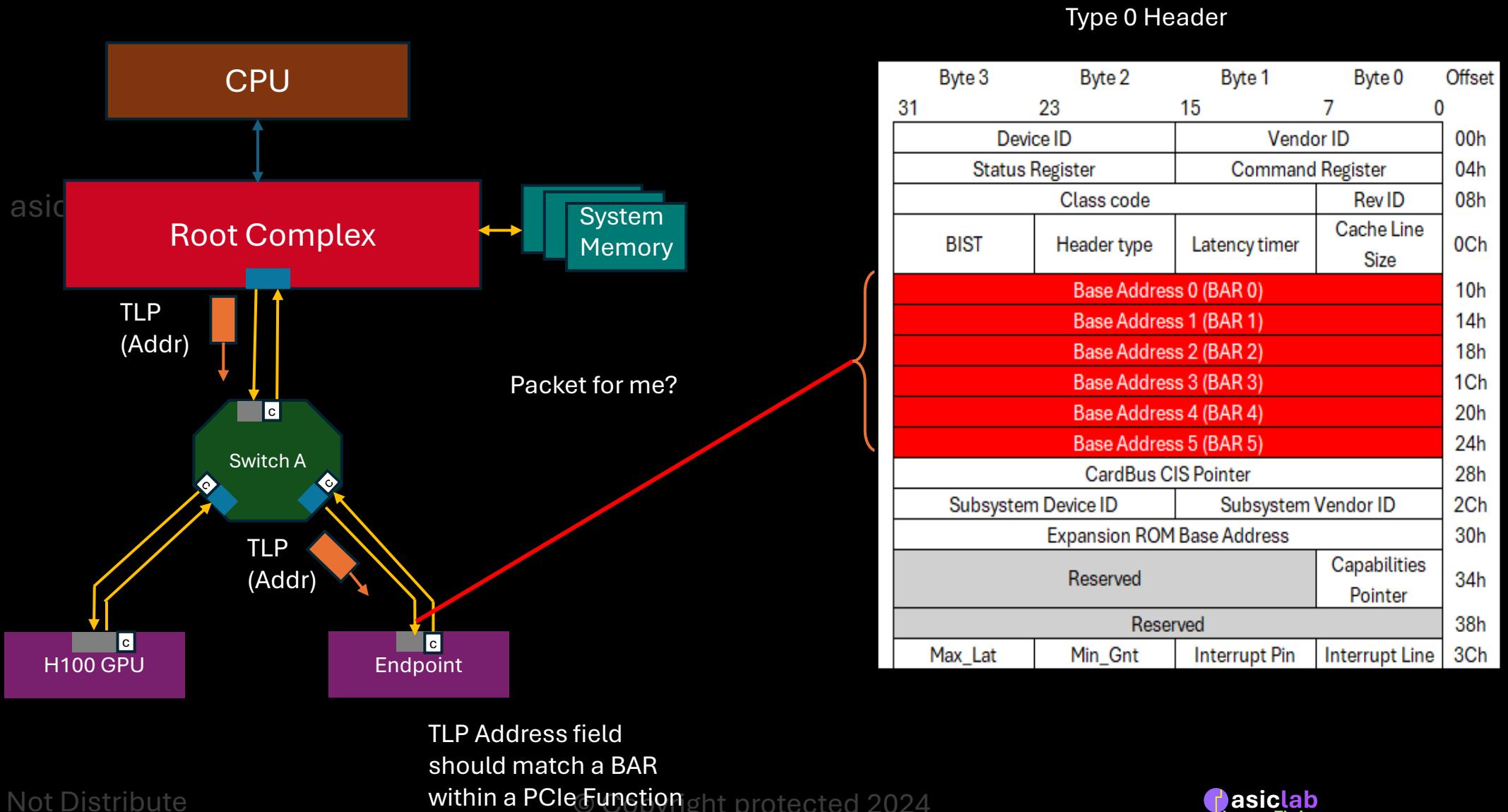


asiclab

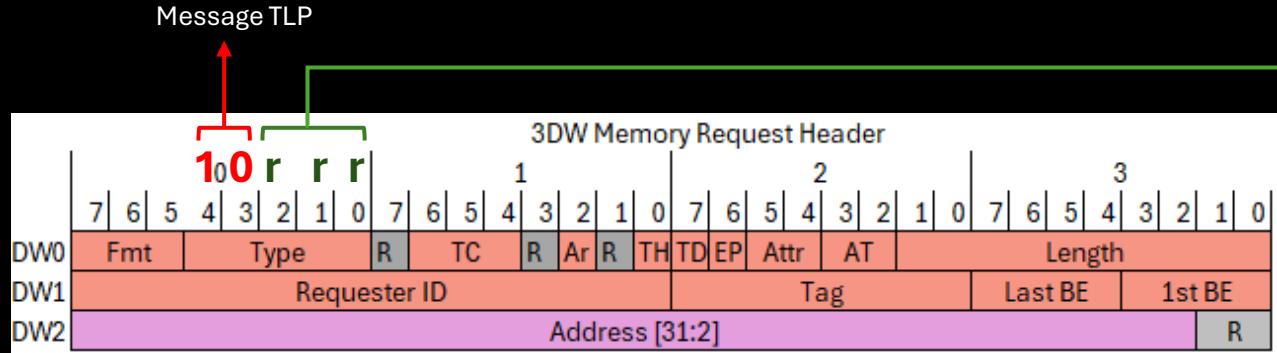
Method 2: Address Routing; Switch Check



Method 2: Address Routing; Switch Check



Method 3: Implicit Routing



Routing information

000b = **Implicit**: Route to Root Complex
001b = Route by Address (Uses Address fields)
010b = Route by ID (Uses Requester ID field)
011b = **Implicit**: Broadcast by Root Complex
100b = **Implicit**: Local – Terminate at Receiver
101b = **Implicit**: Gather and route to Root Complex
All other = Reserved

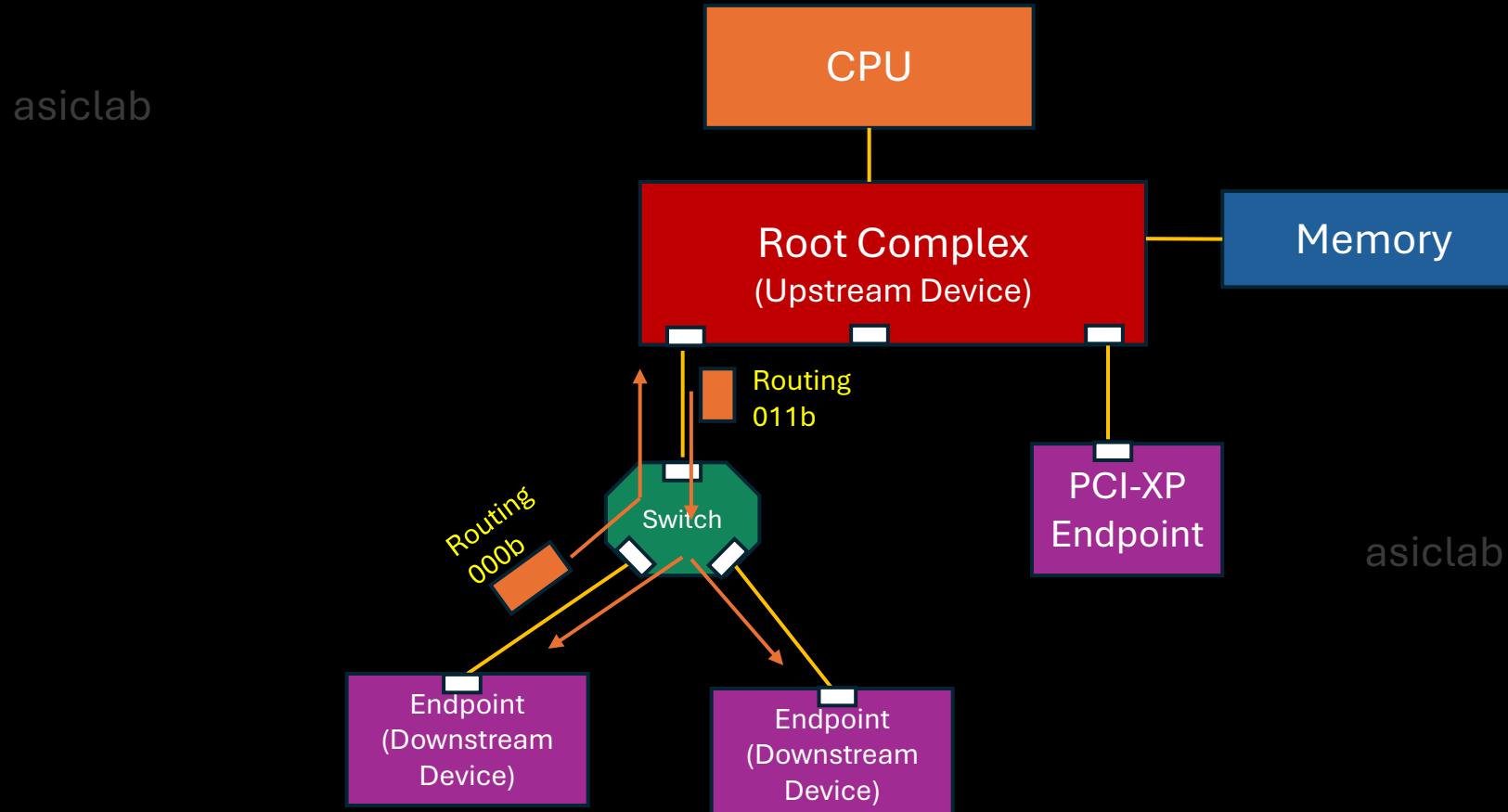
Encoding	Description
000b	TLP, 3DW Header, no data
001b	TLP, 4DW Header, no data
010b	TLP, 3DW Header, with data
011b	TLP, 4DW Header, with data
100b	Prefix DW

Encoding	Description
00000b	Memory Request
00001b	Memory Read Lock Request
11011b	Deferrable Memory Write Request
00010b	IO Request
00100b	Configuration Type 0 Request
00101b	Configuration Type 1 Request
10 r r r b	Message Request
01010b	Completion
01011b	Completion Lock
01100b	AtomicOP: Fetch and Add Request
01101b	AtomicOP: Unconditional Swap Request
01110b	AtomicOP: Compare and Swap Request
0L ₁ L ₂ L ₁ L ₀	Loc TLP Prefix (Fmt[2:0] = 100b)
1E ₃ E ₂ E ₁ E ₀	End-to-End TLP Prefix (Fmt[2:0]= 100b)

Method 3: Implicit Routing

In Implicit routing, a device verifies that it is intended recipient based on the routing type

Examples: Root Complex sees message with routing code 000b, or Endpoint sees message routing code 011b

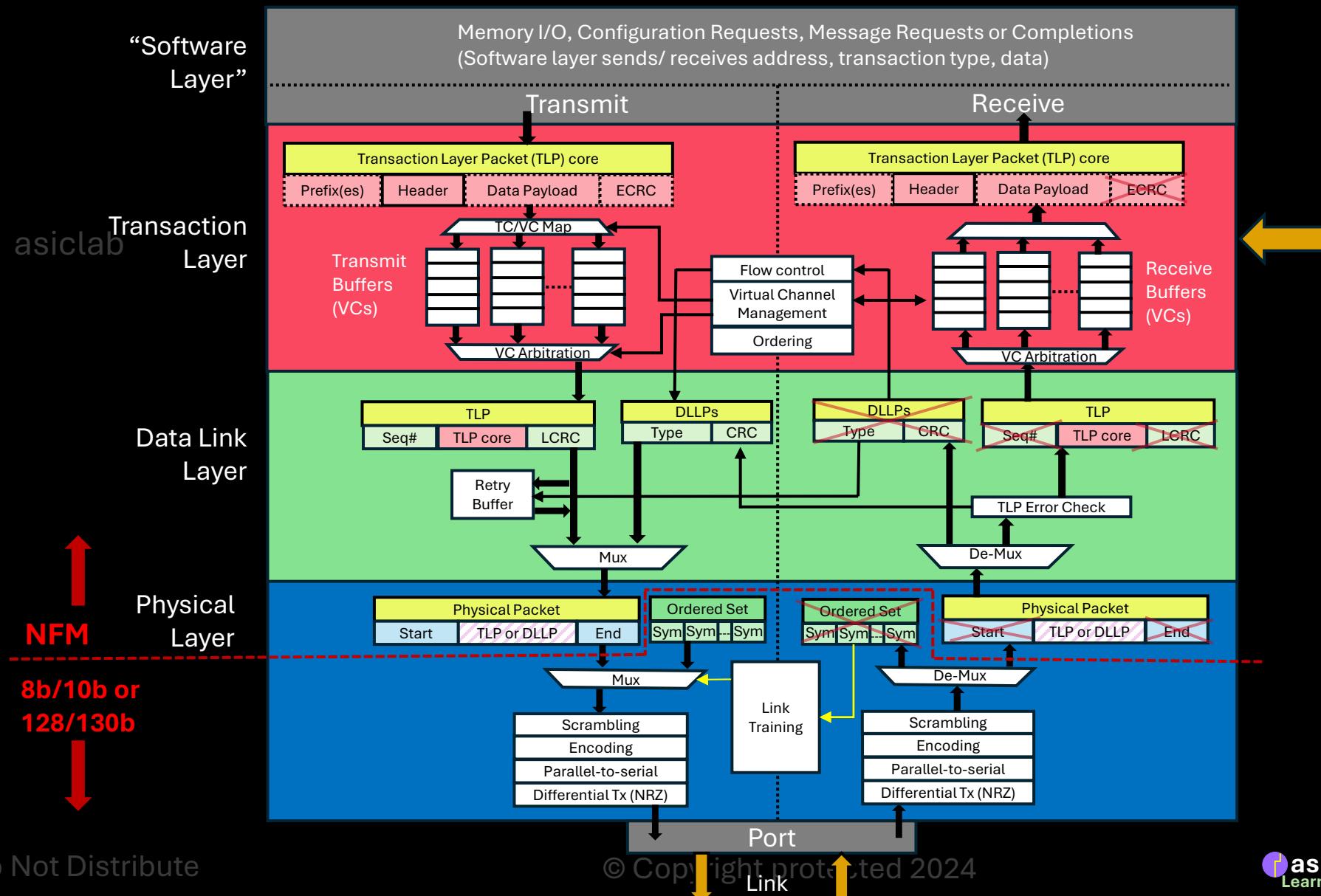


asiclab

Transaction Layer

asiclab

PCIe Device Layer Details 5.0 Non Flit Mode (NFM)

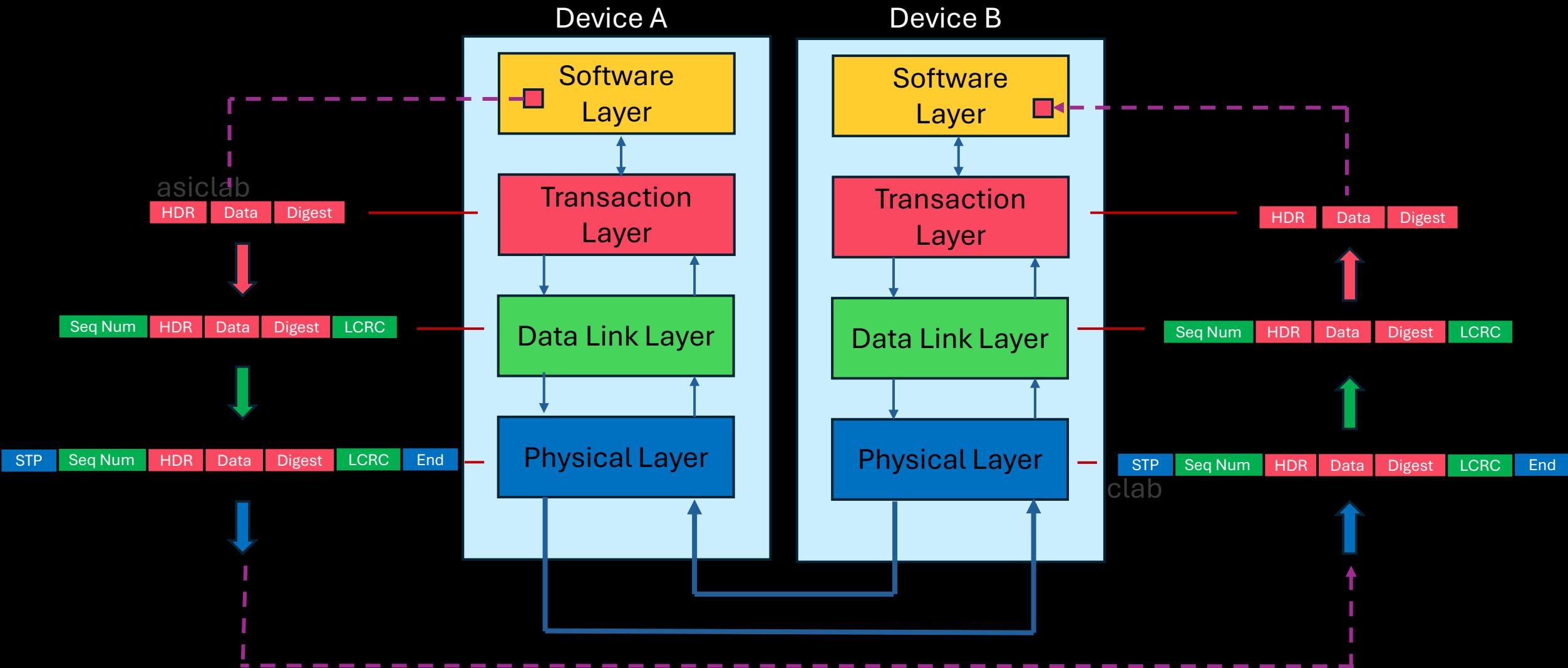


Do Not Distribute

© Copyright protected 2024
Link

asiclab
Learn. Thrive

TLP Assembly/Disassembly (8b/10b) (128b/130b)



Posted and Non-Posted Transactions

- Posted transactions consist of a single TLP sent to the completer; no response is expected
- Non-Posted transactions are split: a Request TLP will be answered later by one or more Completion TLPs.

asiclab

Transaction Type	Non-Posted or Posted
Memory Read	Non-Posted
Memory Write	Posted
Memory Read Lock	Non-Posted
IO Read	Non-Posted
IO Write	Non-Posted
Configuration Read (Type 0 and 1)	Non-Posted
Configuration Write (Type 0 and 1)	Non-Posted
Message	Posted
AtomicOp	Non-Posted

Transaction Layer Packets

- Memory Request TLP
- Completion TLP
- Configuration TLP
- IO Request TLP
- ~~asiclab~~ Message of Vendor Defined Message TLP

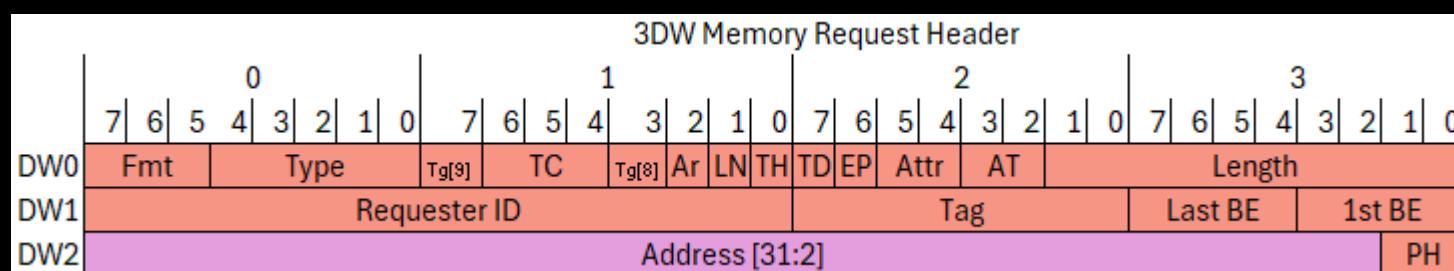
asiclab

asiclab

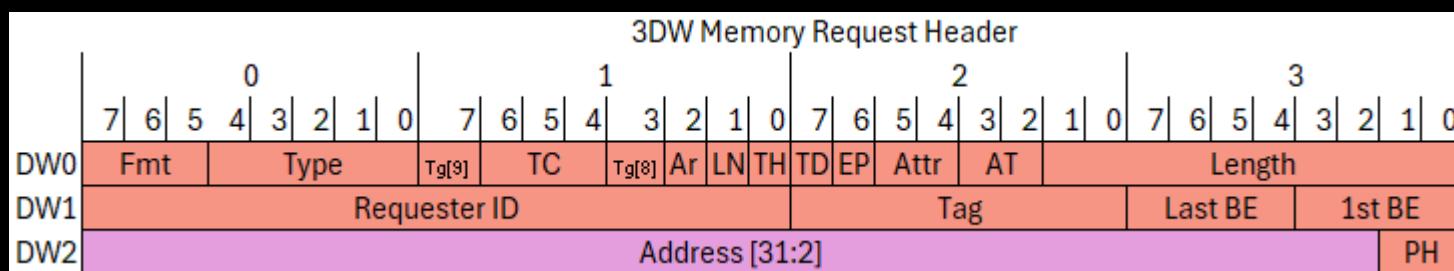
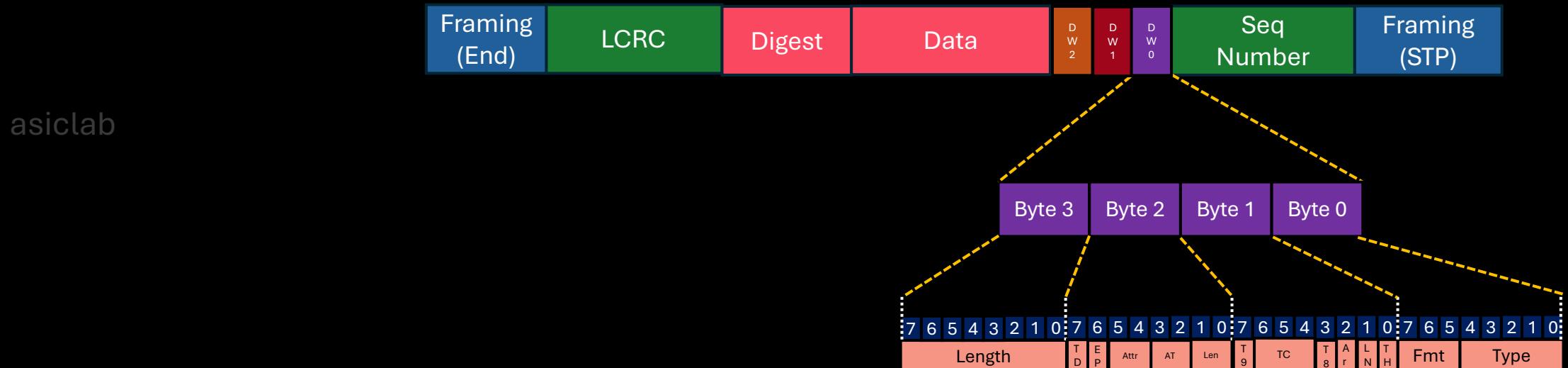
Memory Request TLP

asiclab

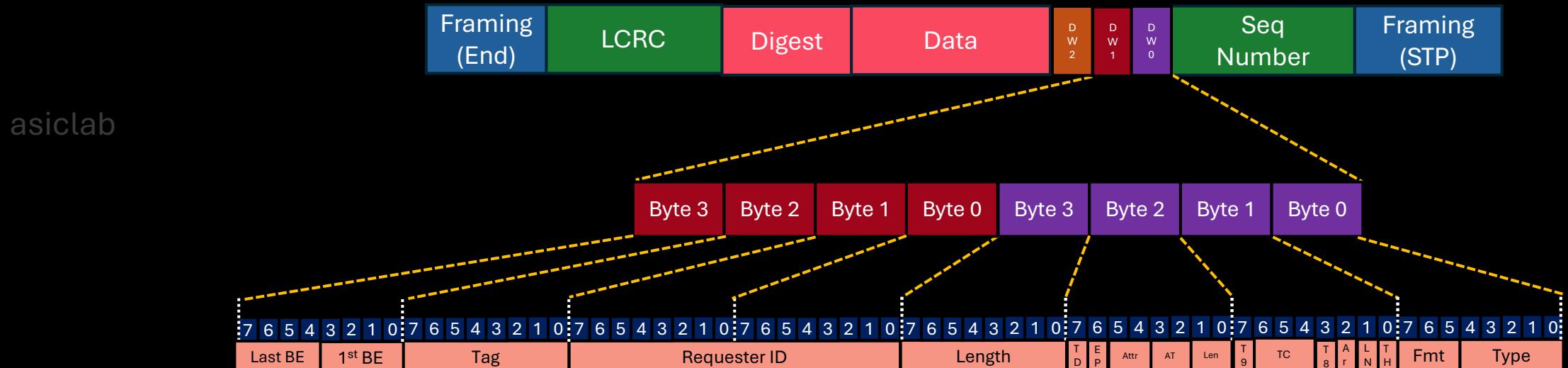
TLP Header format



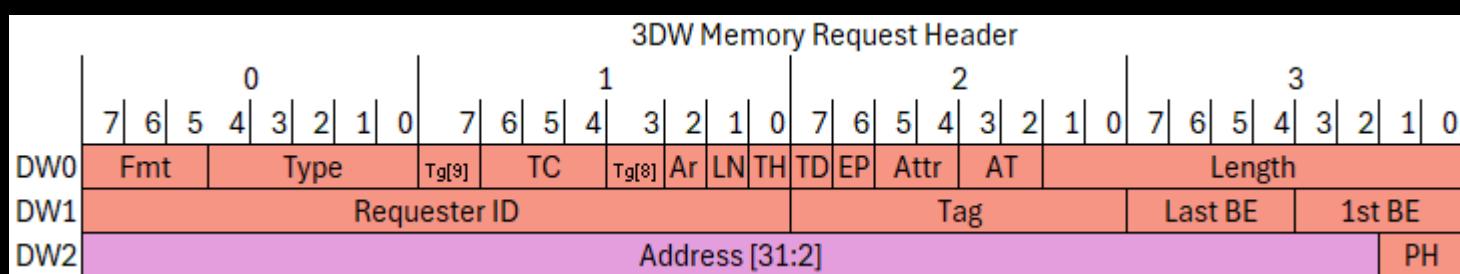
TLP Header format



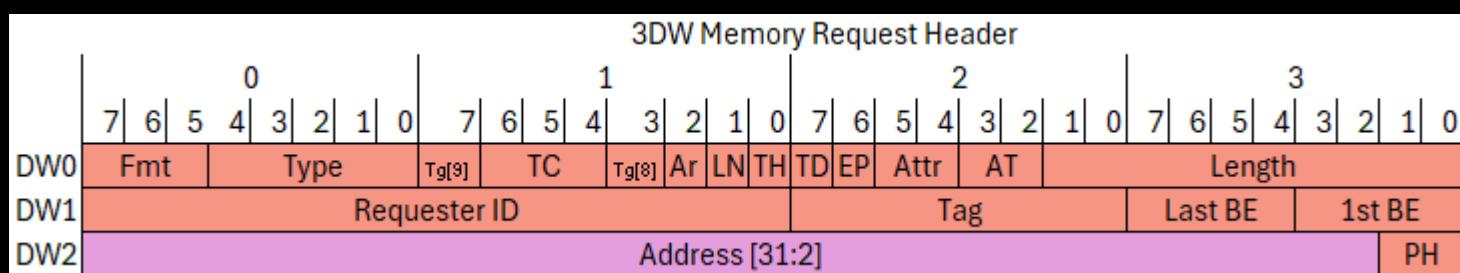
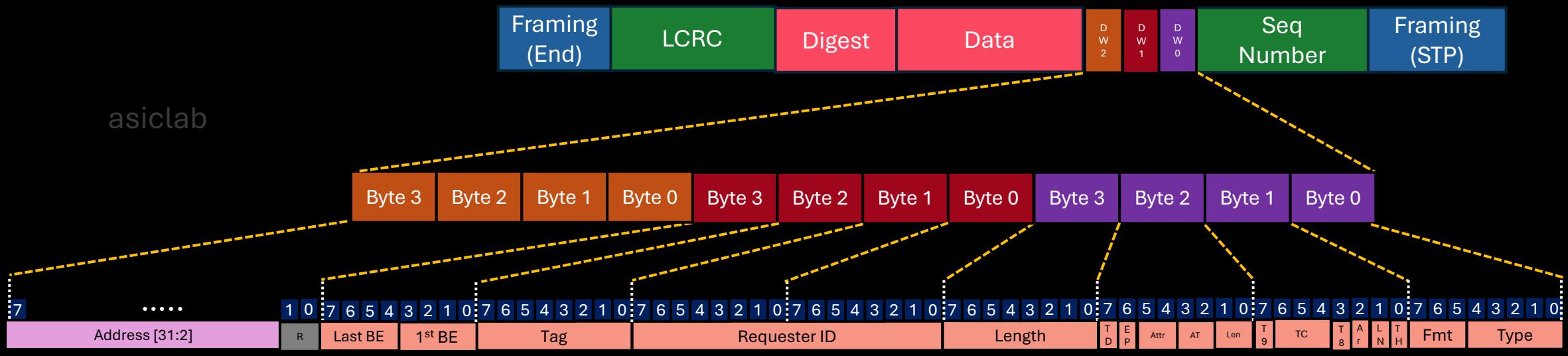
TLP Header format



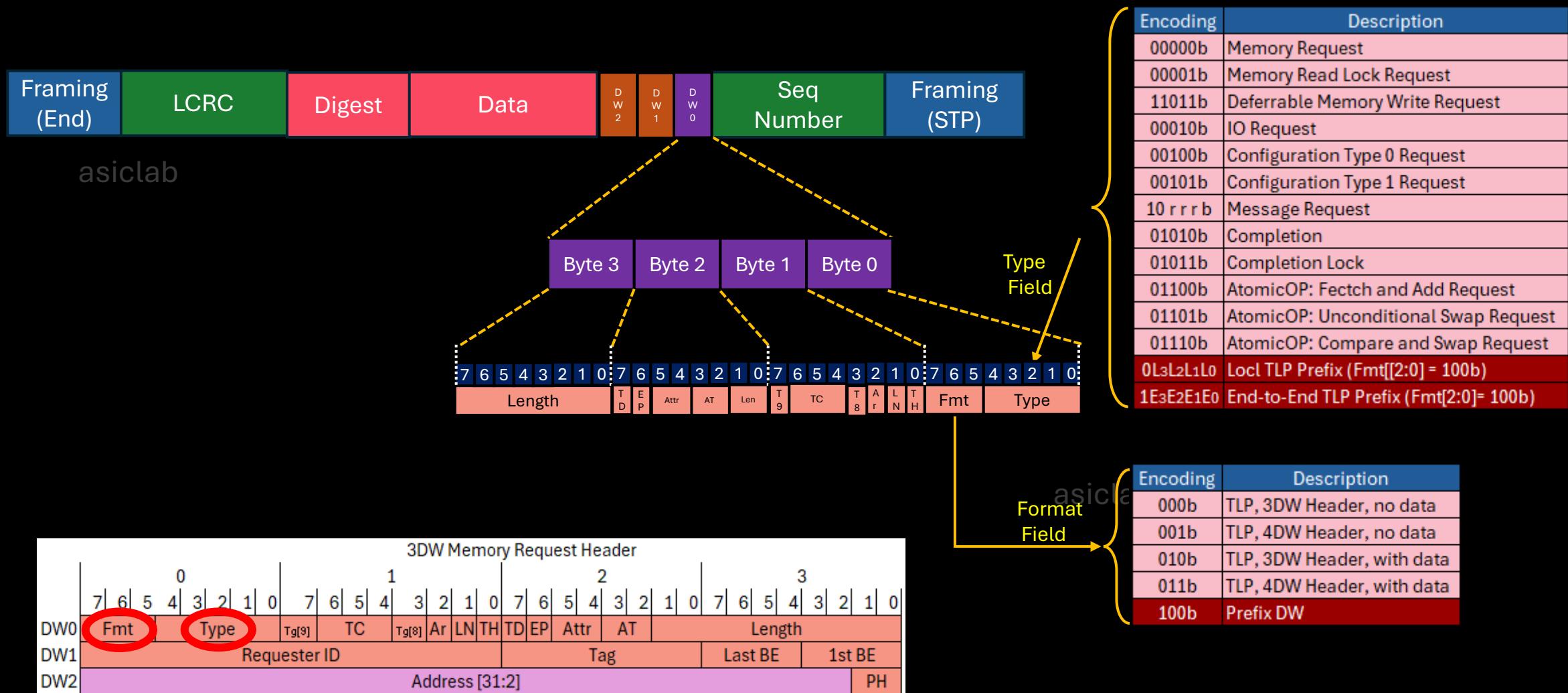
asiclab



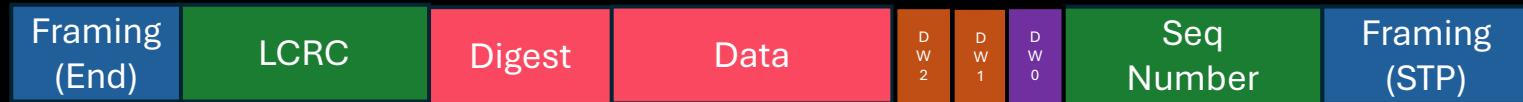
TLP Header format



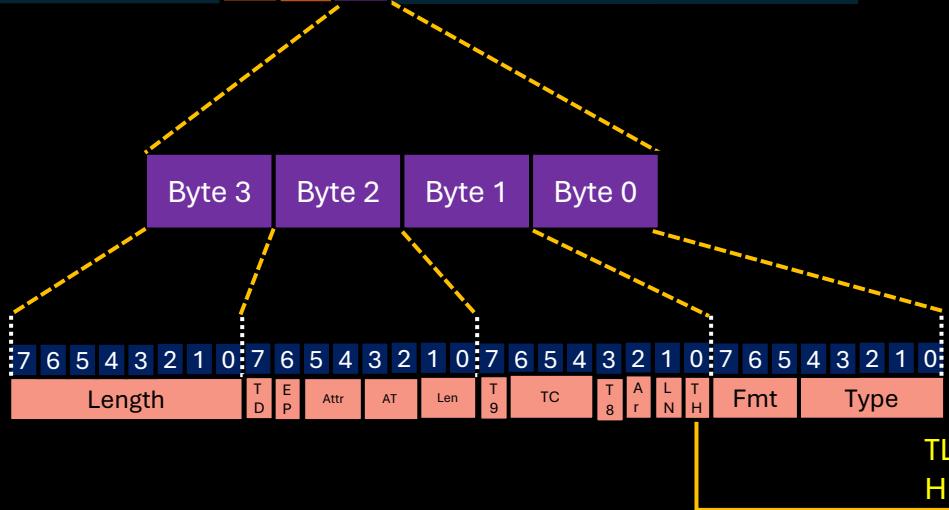
TLP Header format



TLP Header format



asiclab



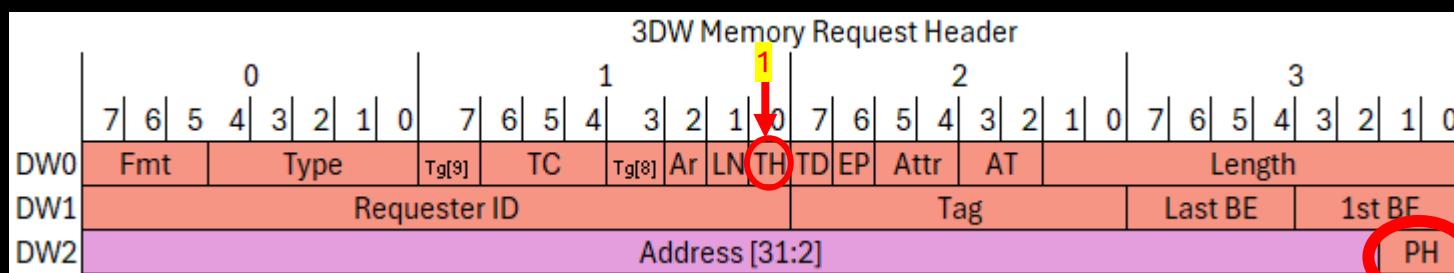
Processing Hints (PH)

00b: Bi-directional data; frequent reads/writes by both device and host

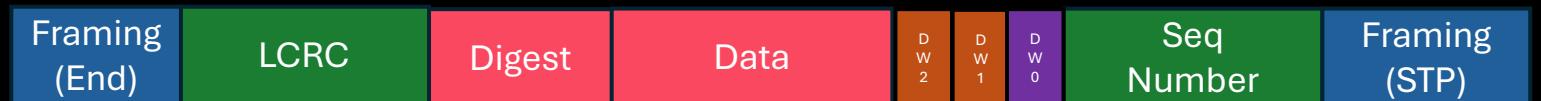
01b: Requester; frequent reads/writes by device

10b: Target; frequent reads/writes by host

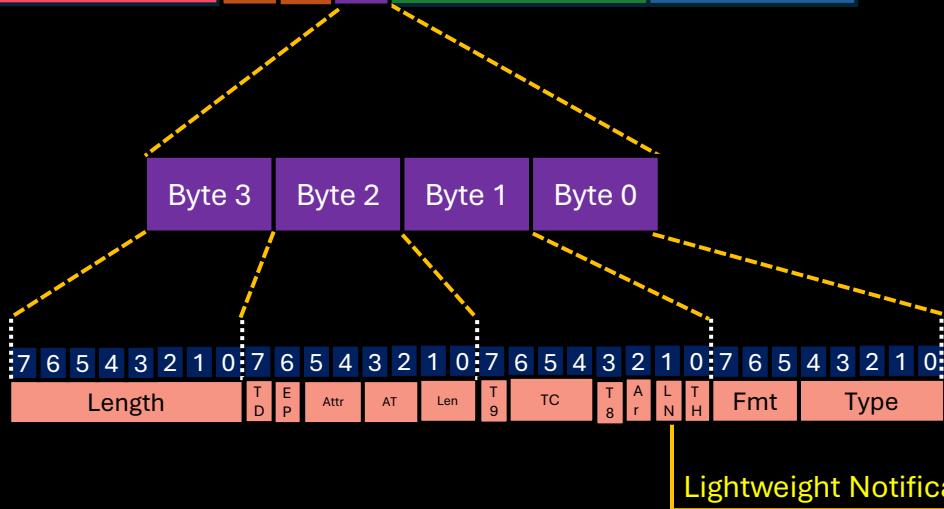
11b: Target with priority; frequent reads/writes by host indicates high temporal locality



TLP Header format



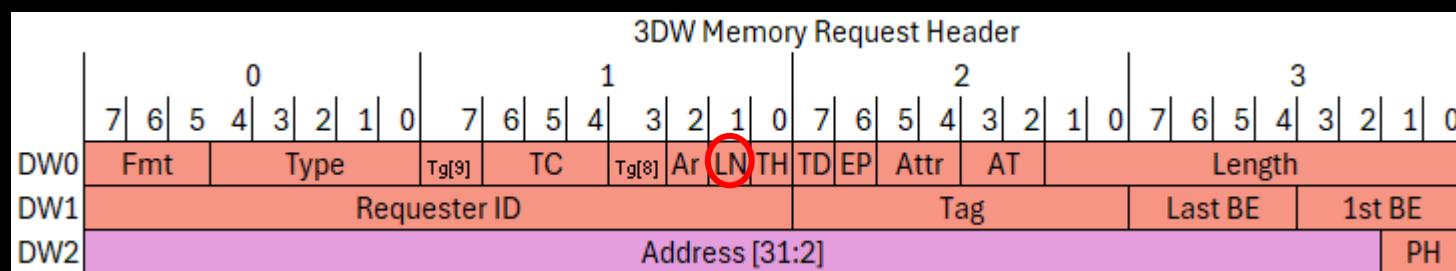
asiclab



Lightweight Notification Field
Lightweight notification is an optional feature where an Endpoint can request push notifications if a registered cacheline changes (lightweight cache coherency)

This is valid only for Memory Reads/Writes and Completions

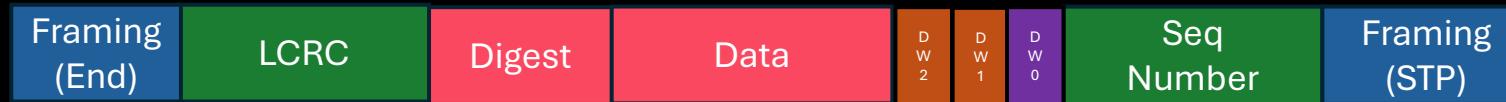
asiclab



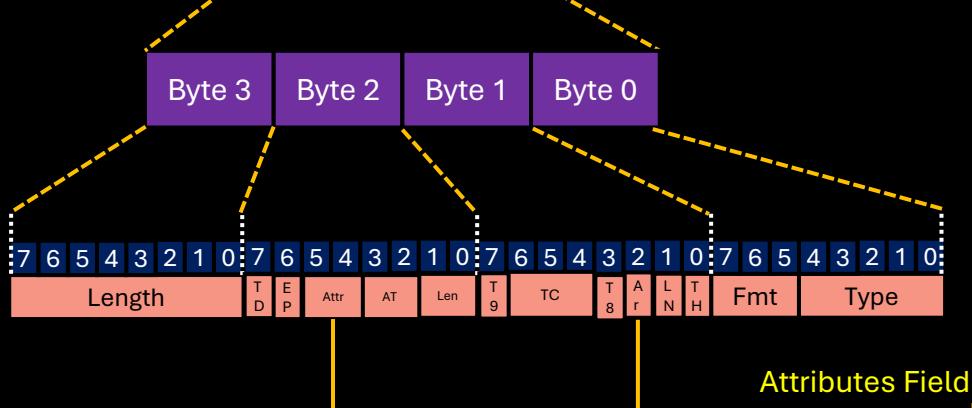
0 not an LN Read, LN Write or LN Completion

1 this is an LN Read, LN Write or LN Completion

TLP Header format

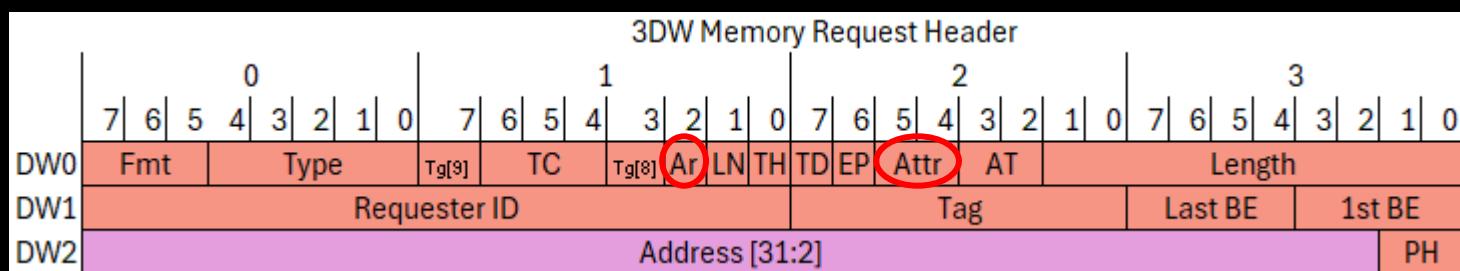


asiclab

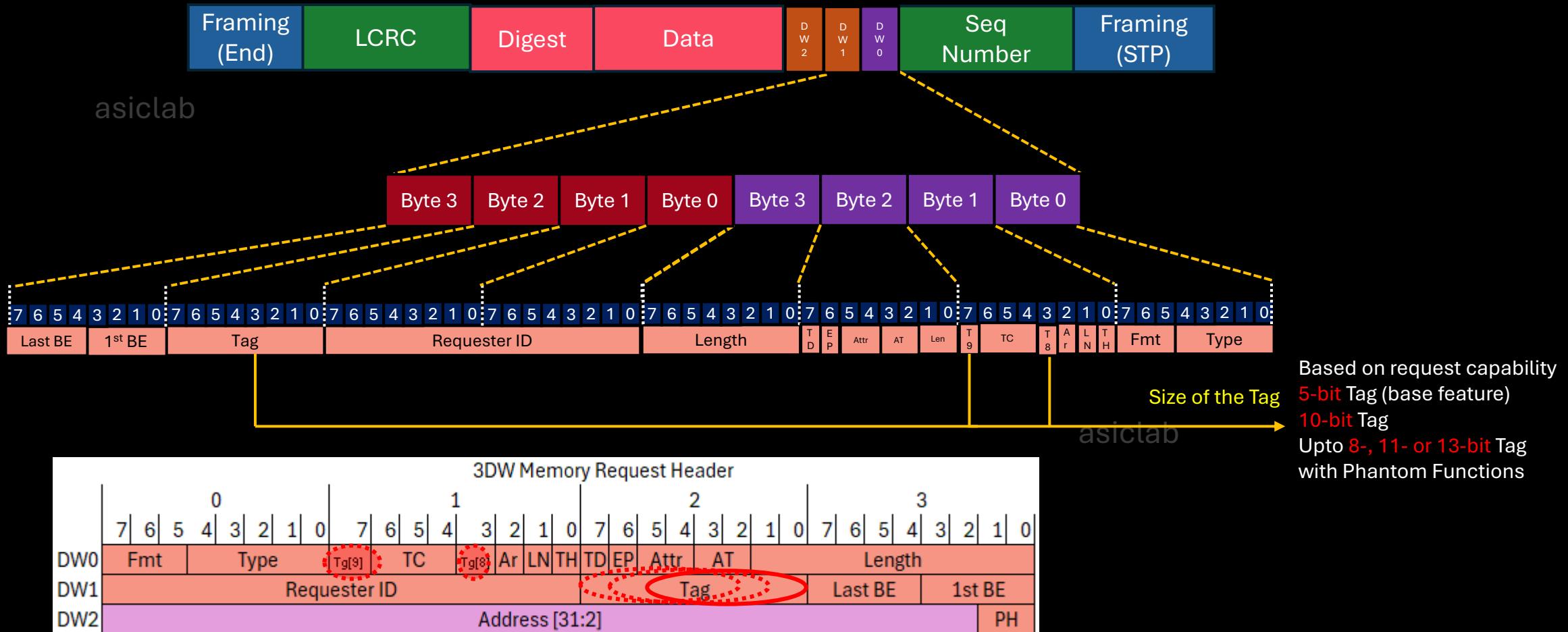


asiclab

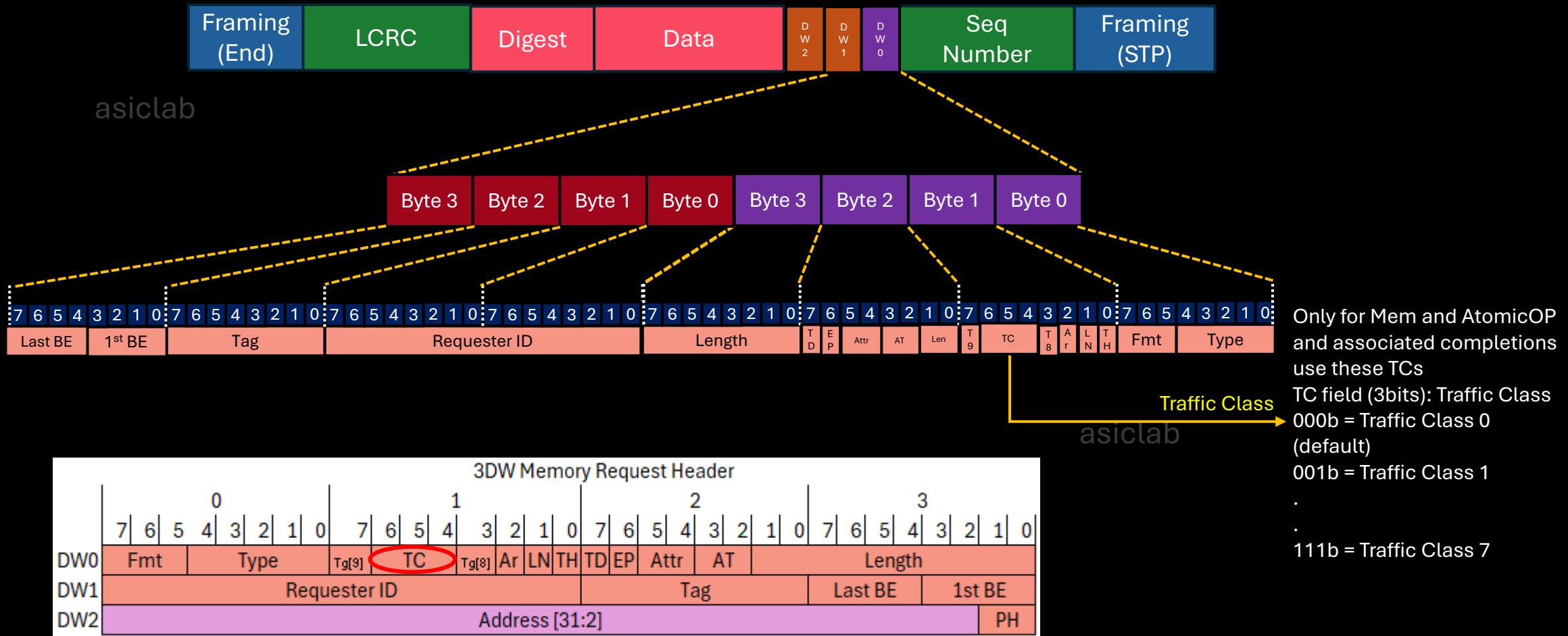
- Attr field (3 bits): Attributes
- Byte 1, Bit 2 (ID-based Ordering)
 - If =0, No ID-based Ordering to be used when routing this TLP
 - If =1, ID-based Ordering is to be used when routing this TLP
- Byte 2, Bit 5 (Relaxed Ordering)
 - If =0, PCI Strong Ordering Model
 - If =1, PCI-X Relaxed Ordering Model
- Byte 2, Bit 4 (No Snoop)
 - If =0, Snoop required
 - If =1, No Snoop required



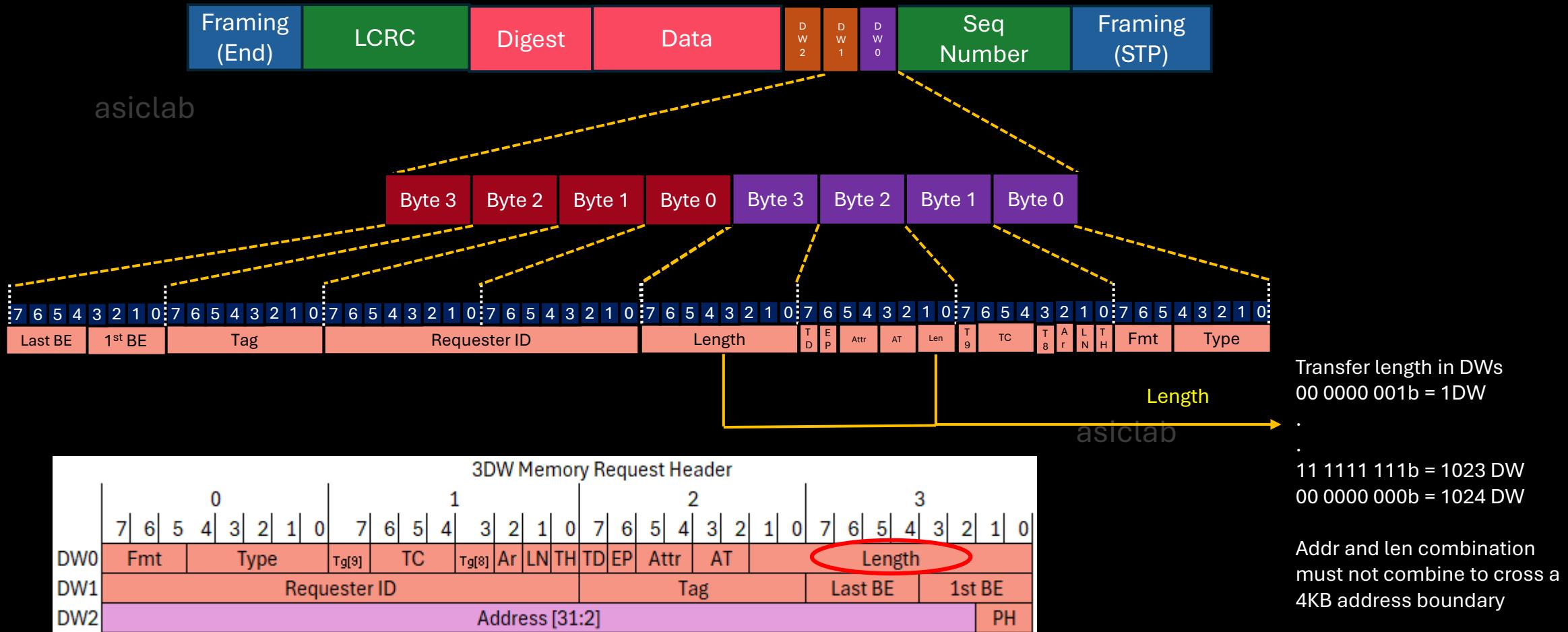
TLP Header format



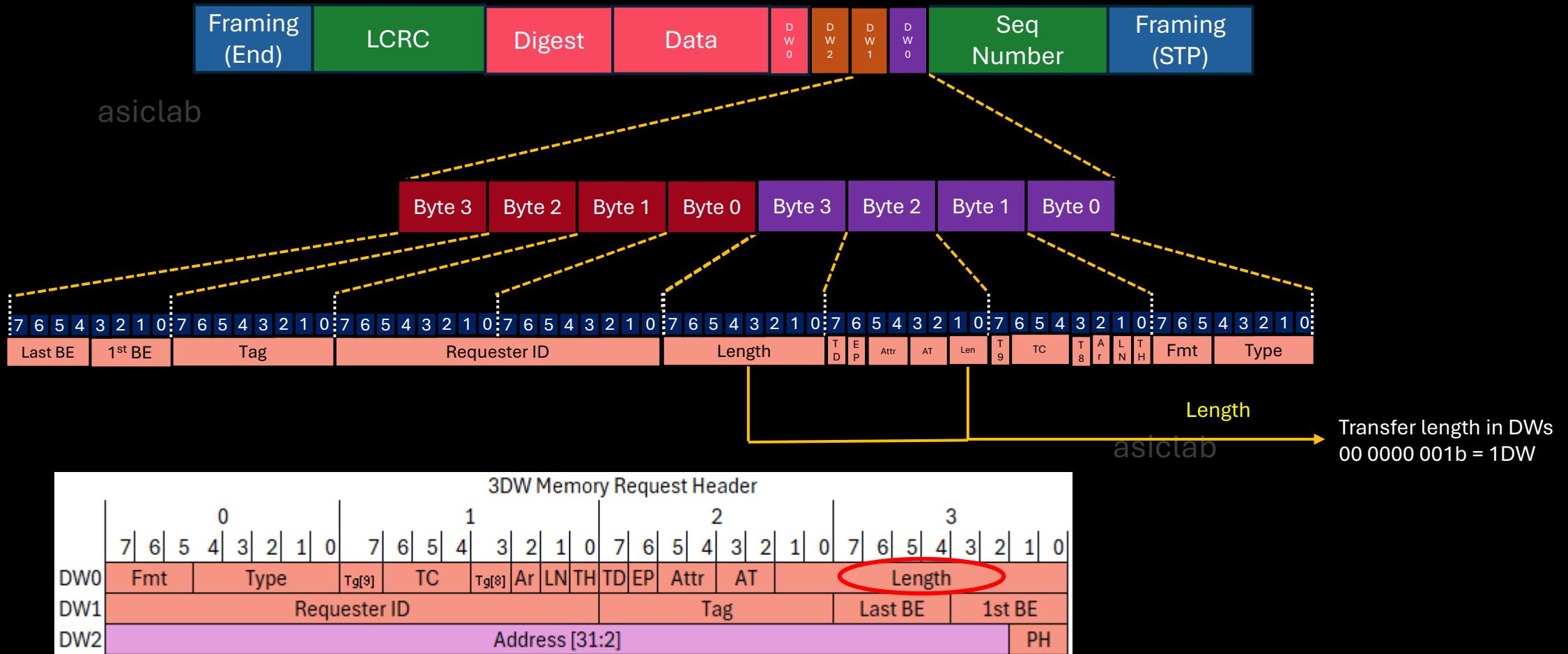
TLP Header format



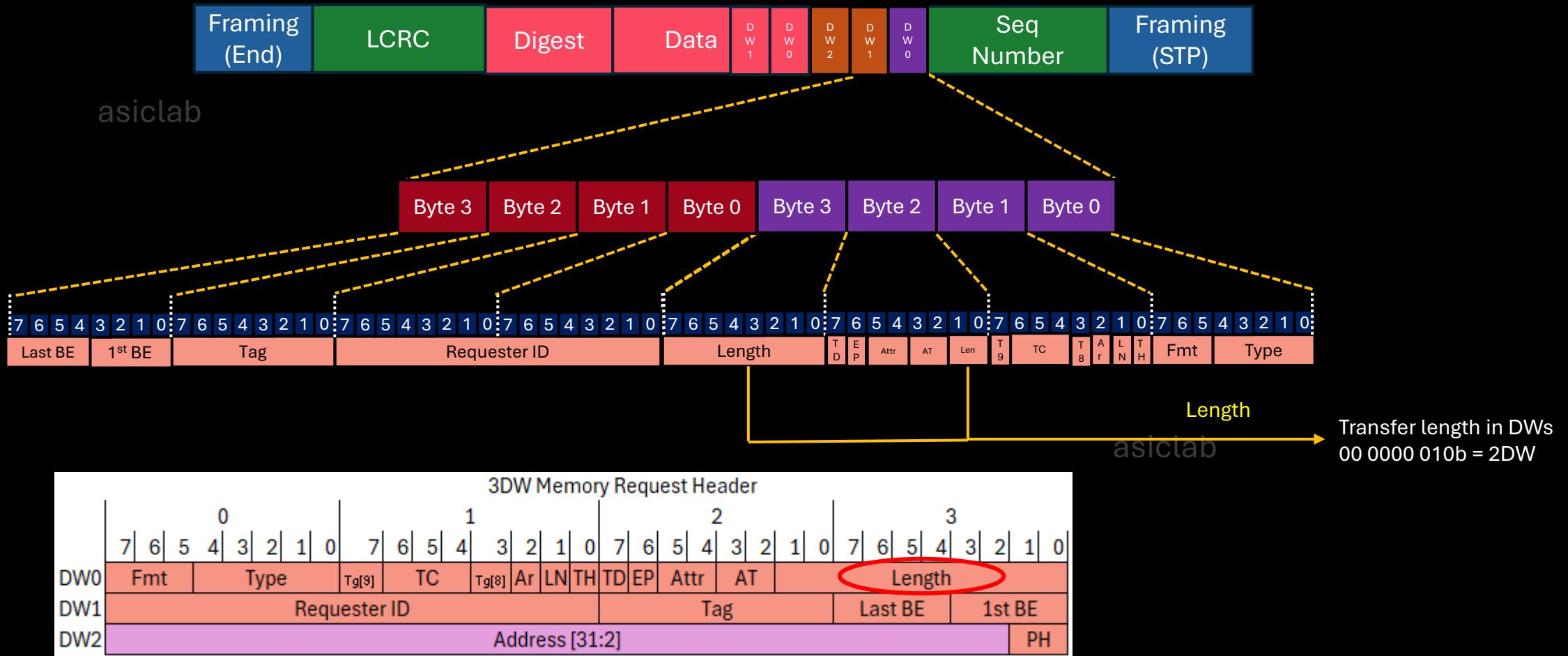
TLP Header format



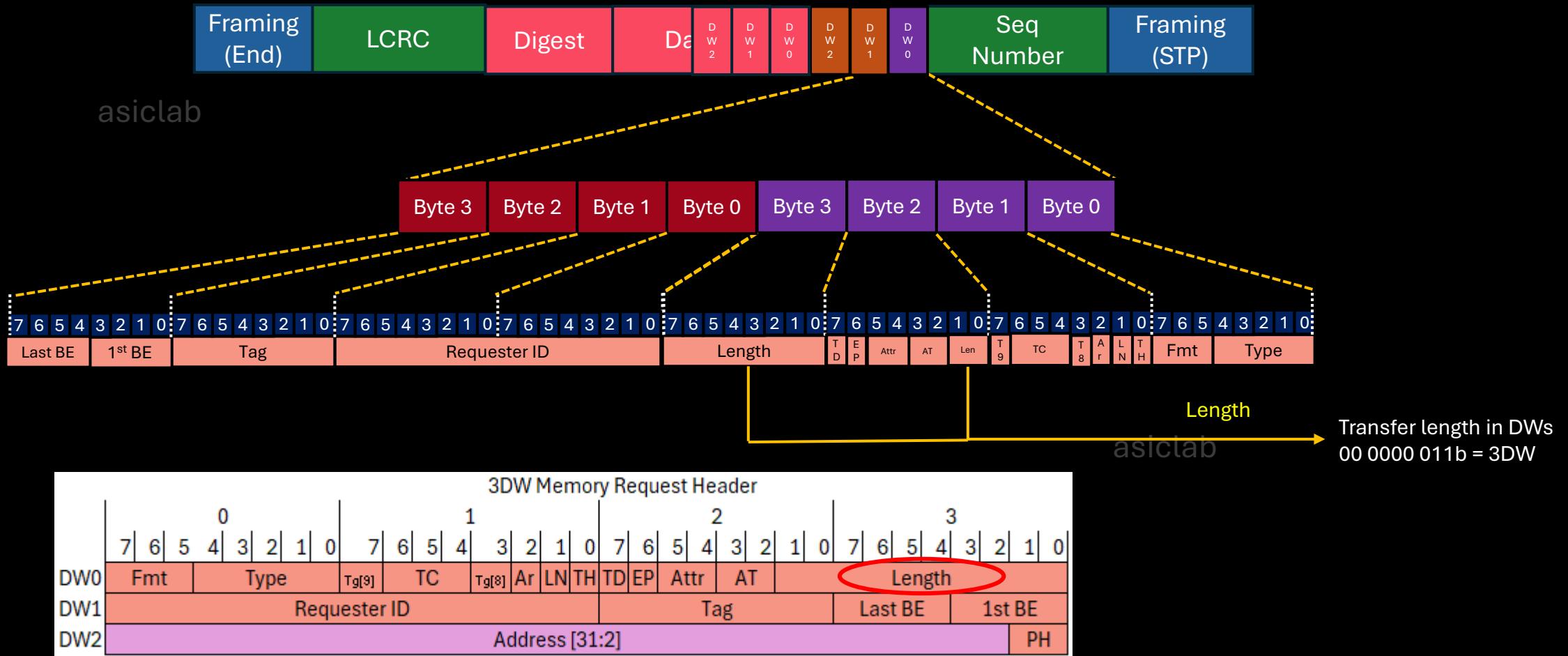
TLP Header format



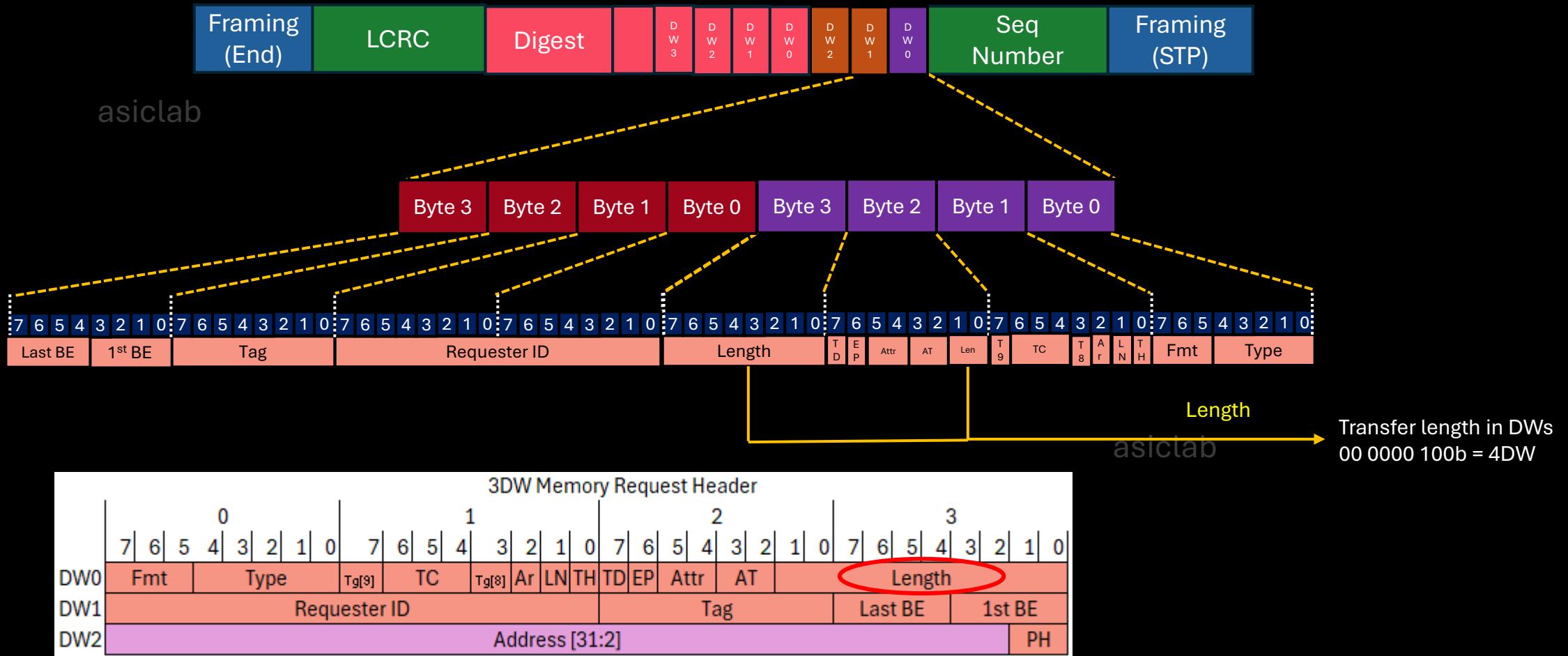
TLP Header format



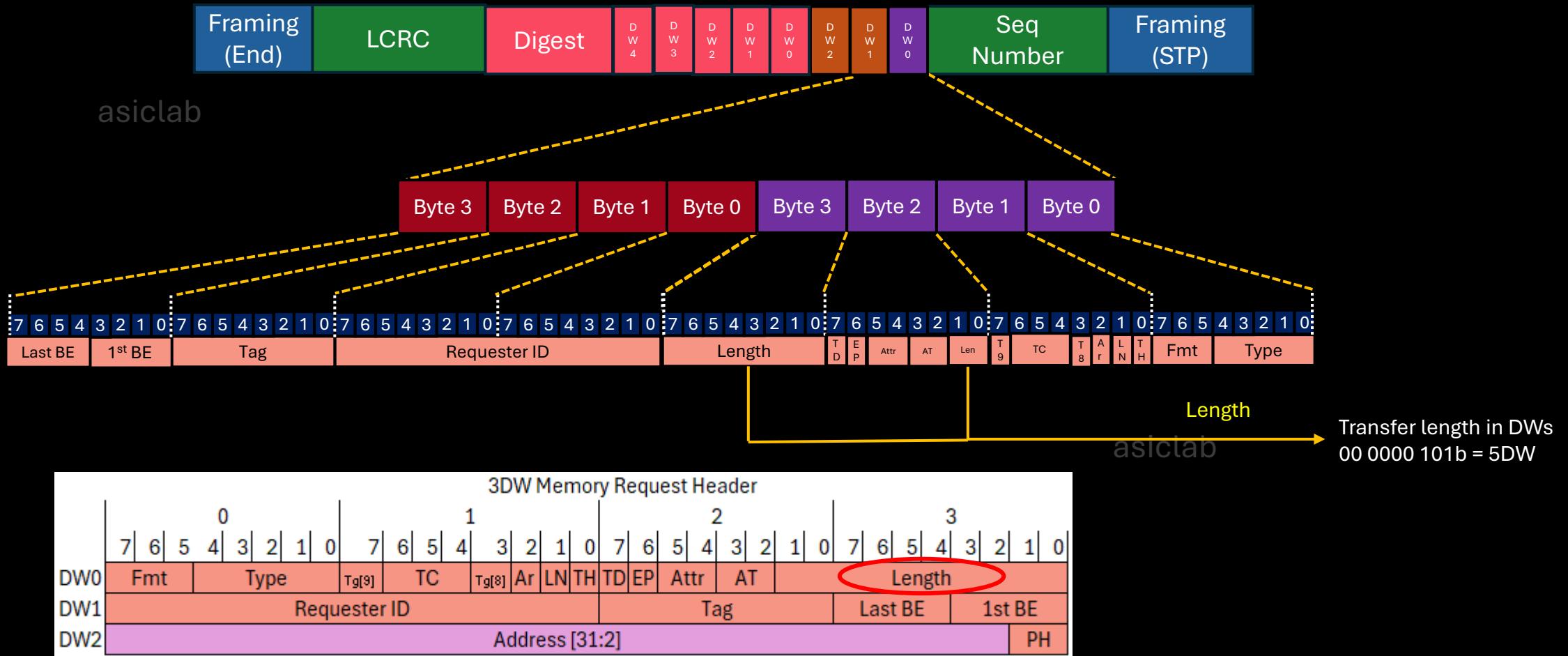
TLP Header format



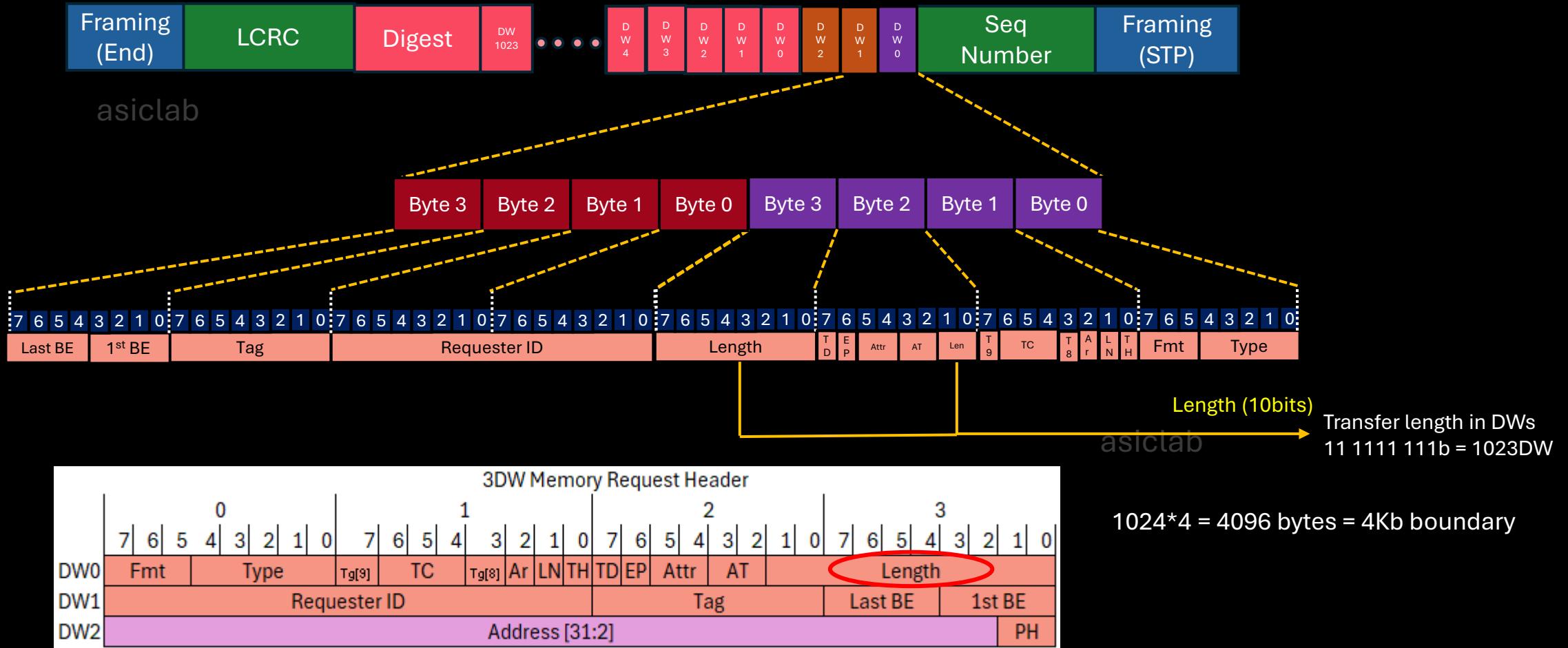
TLP Header format



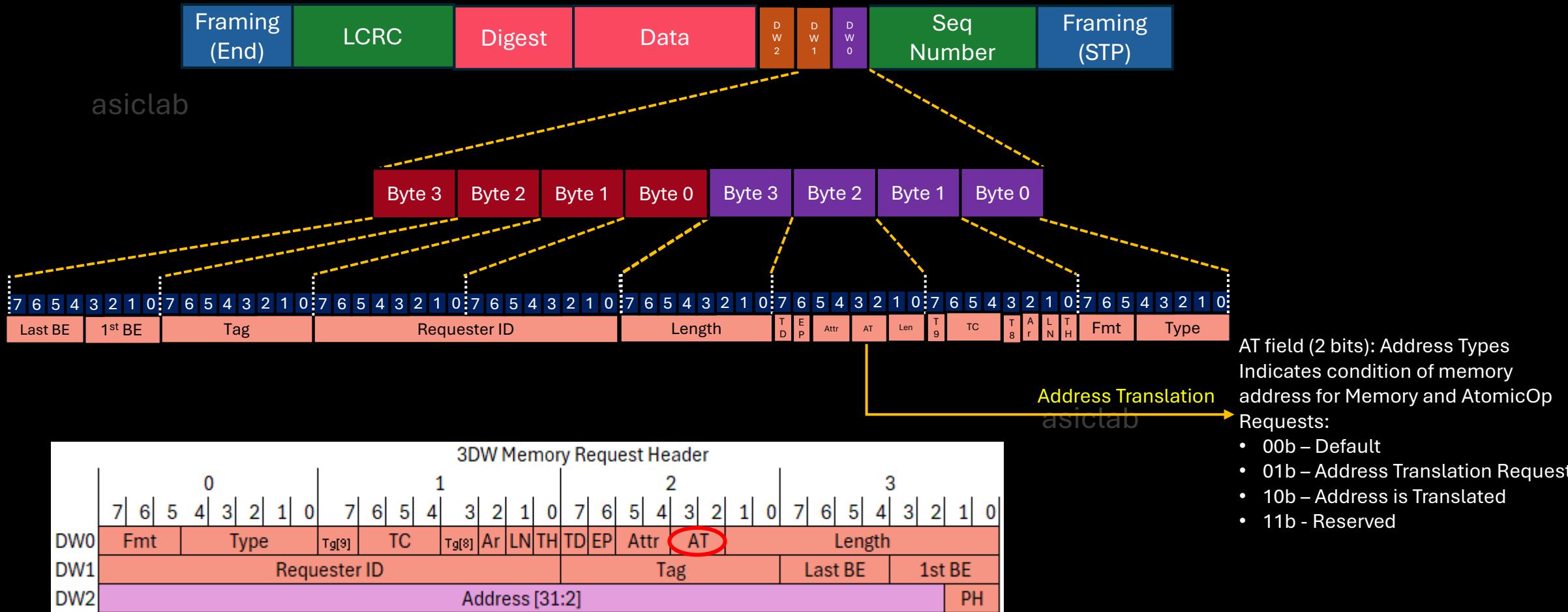
TLP Header format



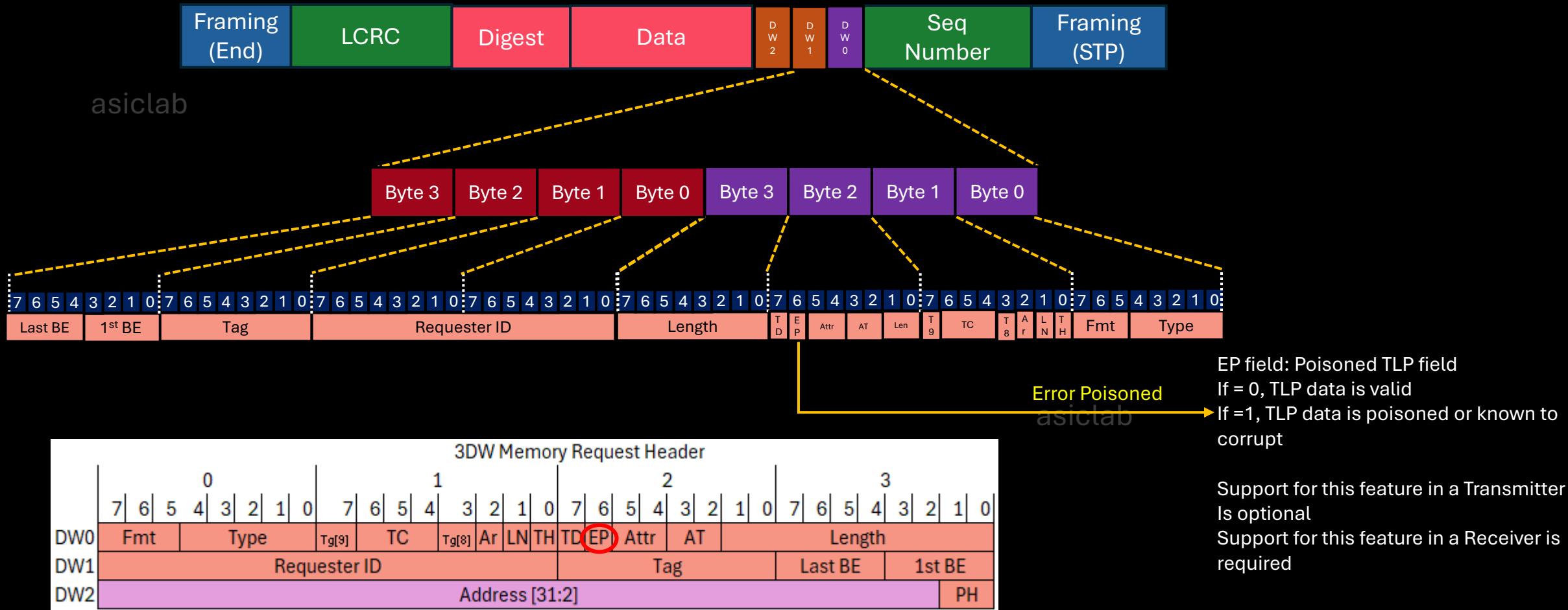
TLP Header format



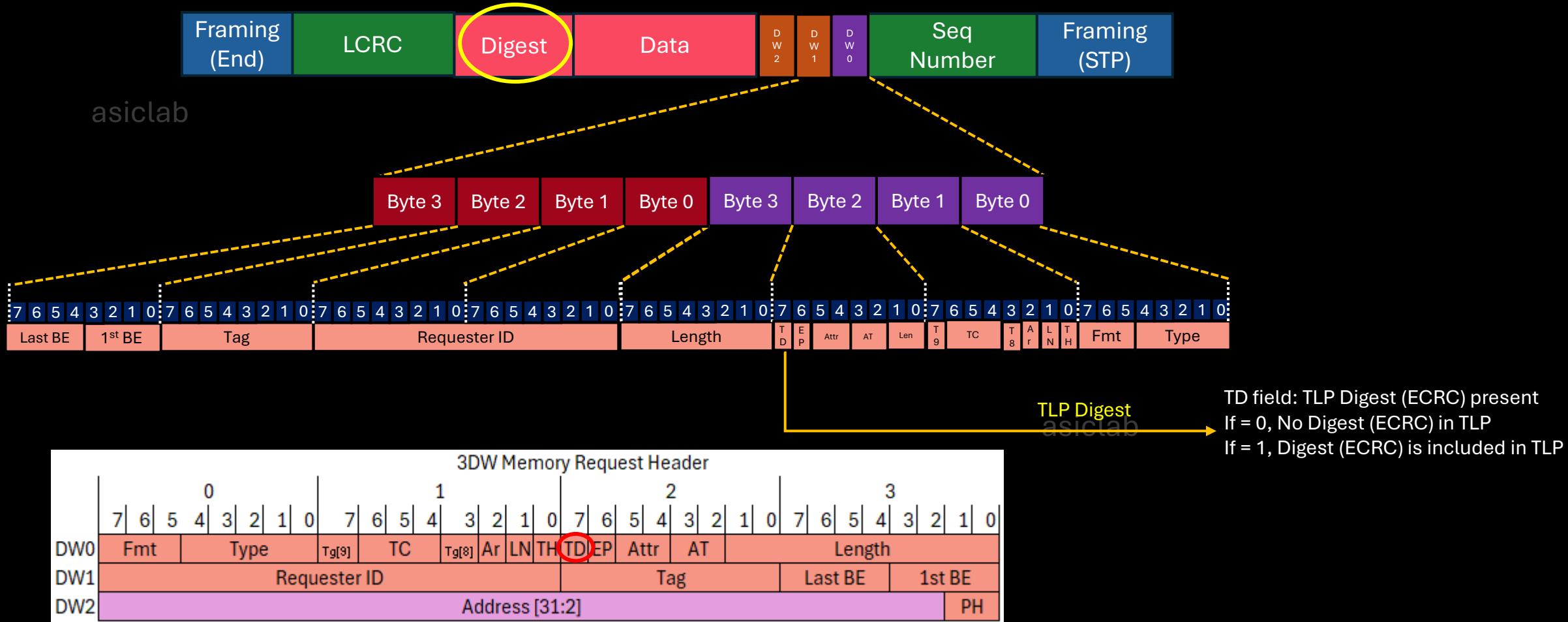
TLP Header format



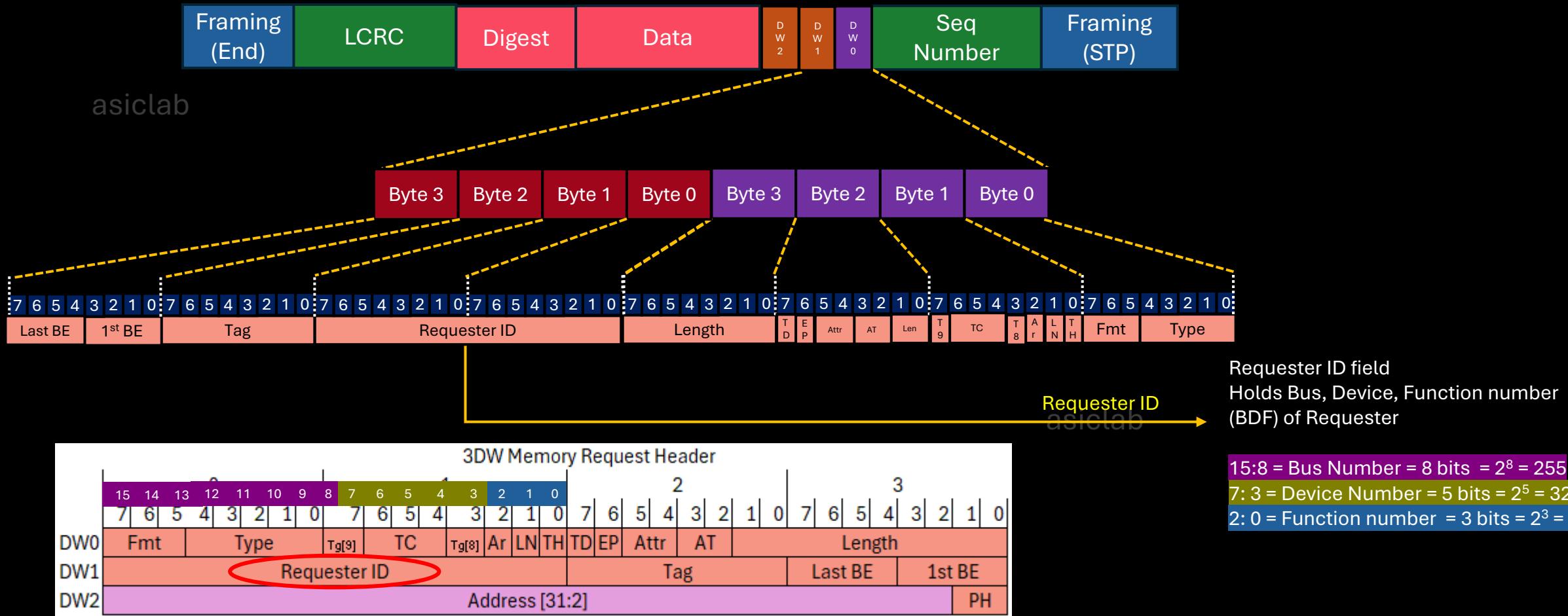
TLP Header format



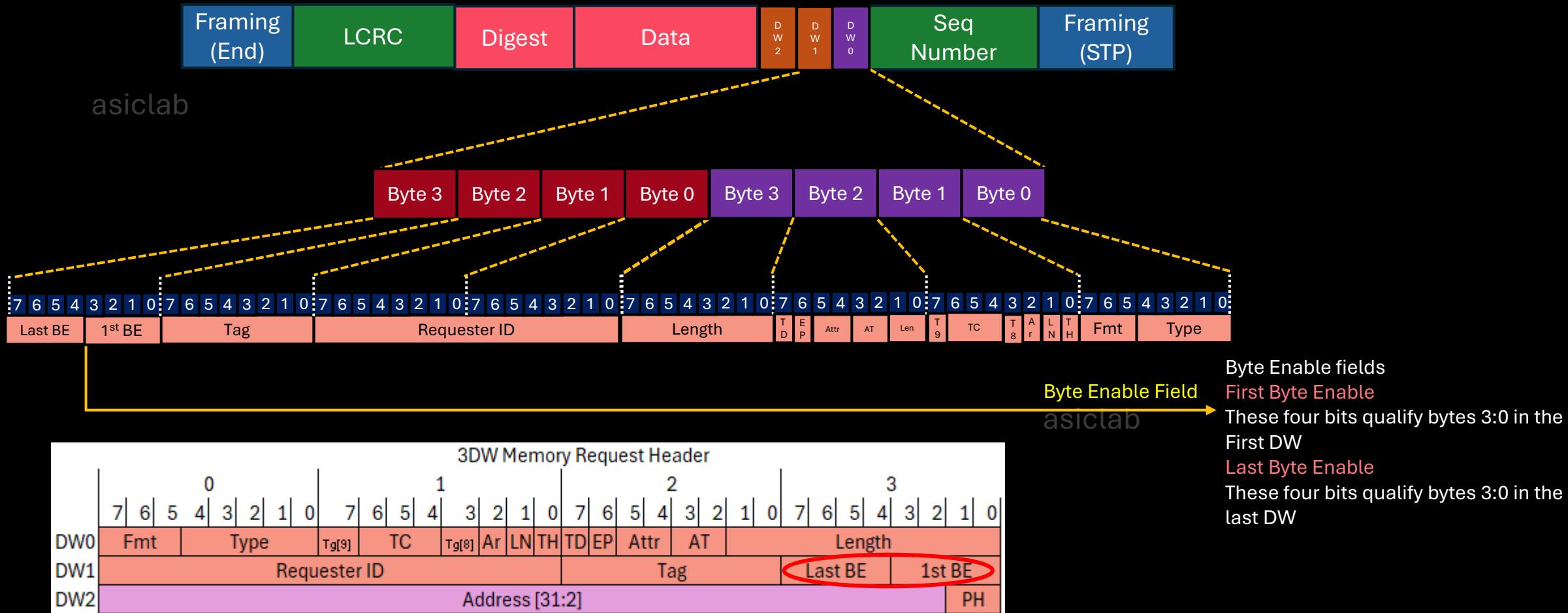
TLP Header format



TLP Header format



TLP Header format



TLP Header format

asiclab

Question:

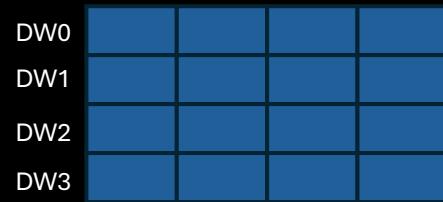
Starting from address location 0x3. I want to transfer 10 bytes of data what is the 1st BE and Last BE

asiclab

TLP Header format

Starting from address location 0x3. I want to transfer 10 bytes of data what is the 1st BE and Last BE

asiclab



asiclab

TLP Header format

Starting from address location 0x3. I want to transfer 10 bytes of data what is the 1st BE and Last BE

asiclab

DW0	0x3	0x2	0x1	0x0
DW1	0x7	0x6	0x5	0x4
DW2	0xB	0xA	0x9	0x8
DW3	0xF	0xE	0xD	0xC

asiclab

TLP Header format

Starting from address location 0x3. I want to transfer 10 bytes of data what is the 1st BE and Last BE

asiclab

1 Byte or 8 bits			
DW0	0x3	0x2	0x1
DW1	0x7	0x6	0x5
DW2	0xB	0xA	0x9
DW3	0xF	0xE	0xD

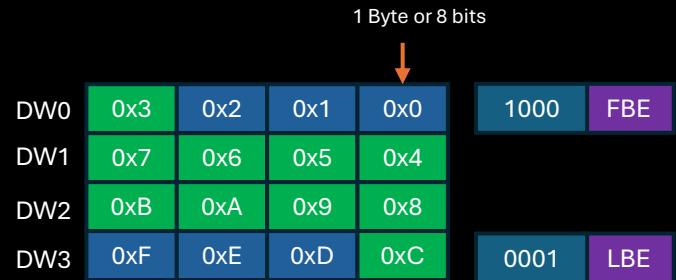


asiclab

TLP Header format

Starting from address location 0x3. I want to transfer 10 bytes of data what is the 1st BE and Last BE

asiclab



Answer: In order to transfer 10 bytes of data from address location 0x3 the 1st BE is 1000 and Last Byte Enable is 0001

asiclab

TLP Header format

Starting from address location 0x3. I want to transfer 10 bytes of data what is the 1st BE and Last BE

asiclab

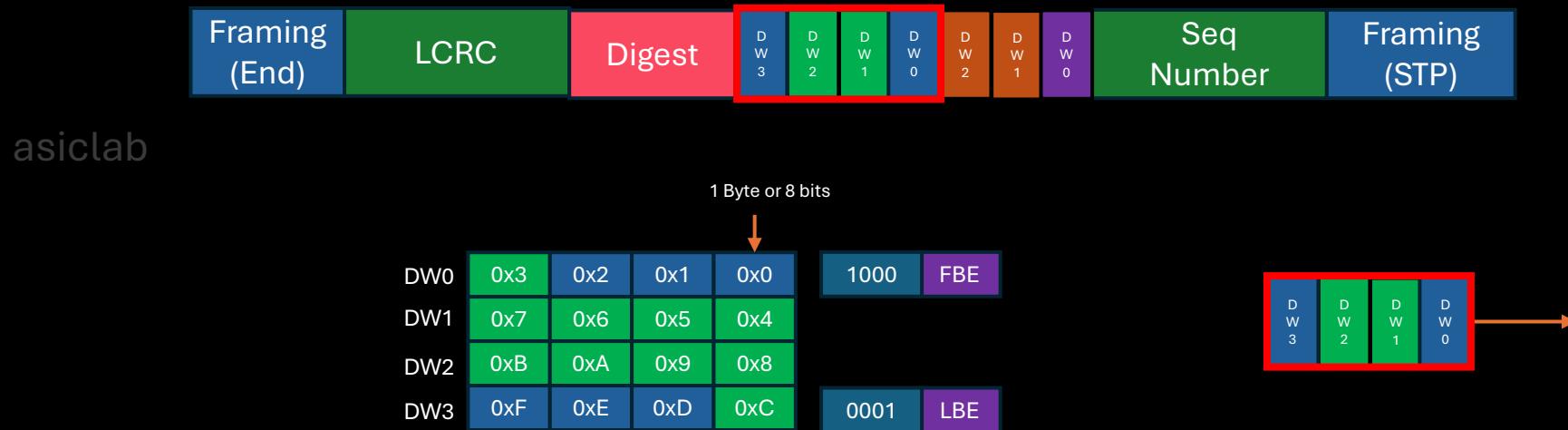


Answer: In order to transfer 10 bytes of data from address location 0x3 the 1st BE is 1000 and Last Byte Enable is 0001

asiclab

TLP Header format

Starting from address location 0x3. I want to transfer 10 bytes of data what is the 1st BE and Last BE



Answer: In order to transfer 10 bytes of data from address location 0x3 the 1st BE is 1000 and Last Byte Enable is 0001

TLP Header format

asiclab

Question:

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

asiclab

TLP Header format

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

asiclab

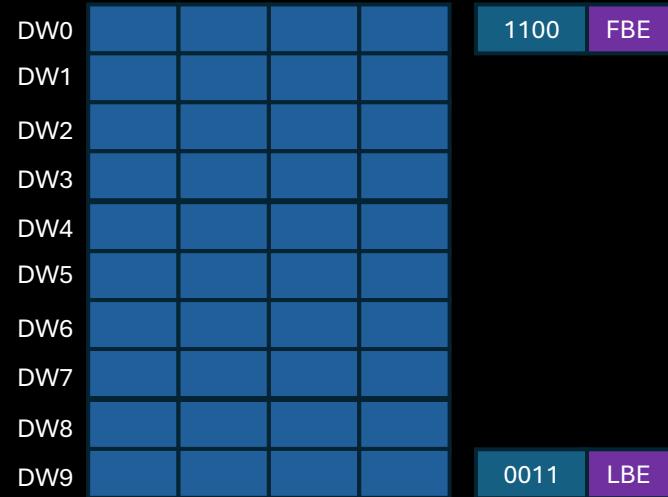
DW0				
DW1				
DW2				
DW3				
DW4				
DW5				
DW6				
DW7				
DW8				
DW9				

asiclab

TLP Header format

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

asiclab

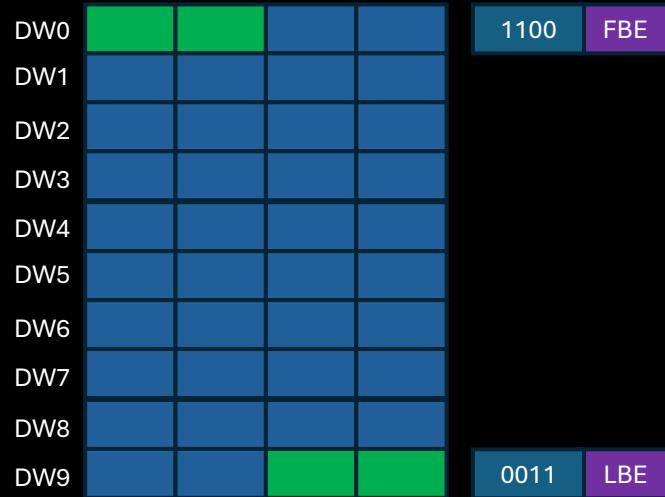


asiclab

TLP Header format

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

asiclab

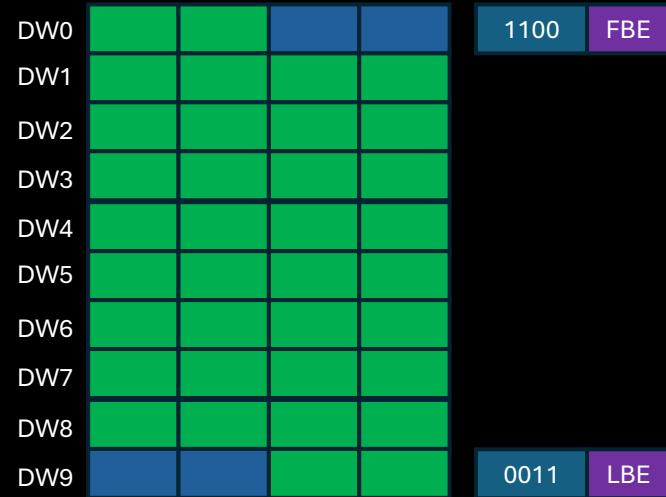


asiclab

TLP Header format

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

asiclab



asiclab

TLP Header format

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

asiclab

DW0	0x3	0x2	0x1	0x0	1100	FBE
DW1	0x7	0x6	0x5	0x4		
DW2	0xB	0xA	0x9	0x8		
DW3	0xF	0xE	0xD	0xC		
DW4	0x13	0x12	0x11	0x10		
DW5	0x17	0x16	0x15	0x14		
DW6	0x1B	0x1A	0x19	0x18		
DW7	0x1F	0x1E	0x1D	0x1C		
DW8	0x23	0x22	0x21	0x20		
DW9	0x27	0x26	0x25	0x24	0011	LBE

asiclab

Answer: Total transferred bytes were: $10\text{DW} - 4\text{bytes} = (10 * 4\text{bytes}) - 4\text{bytes} = 40\text{bytes} - 4\text{bytes} = 36\text{ bytes}$

TLP Header format

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

asiclab

DW0	0x3	0x2	0x1	0x0	1100	FBE
DW1	0x7	0x6	0x5	0x4		
DW2	0xB	0xA	0x9	0x8		
DW3	0xF	0xE	0xD	0xC		
DW4	0x13	0x12	0x11	0x10		
DW5	0x17	0x16	0x15	0x14		
DW6	0x1B	0x1A	0x19	0x18		
DW7	0x1F	0x1E	0x1D	0x1C		
DW8	0x23	0x22	0x21	0x20		
DW9	0x27	0x26	0x25	0x24	0011	LBE

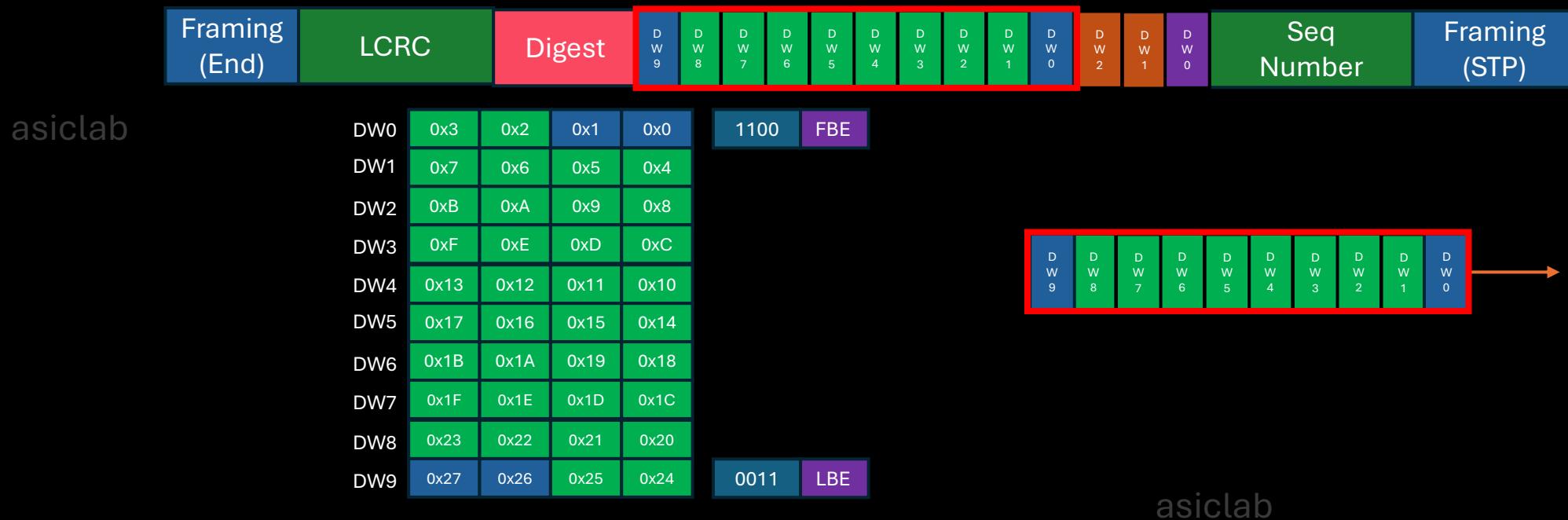


asiclab

Answer: Total transferred bytes were: $10\text{DW}-4\text{bytes}=(10*4\text{bytes})-4\text{bytes}=40\text{bytes}-4\text{bytes}=36\text{ bytes}$

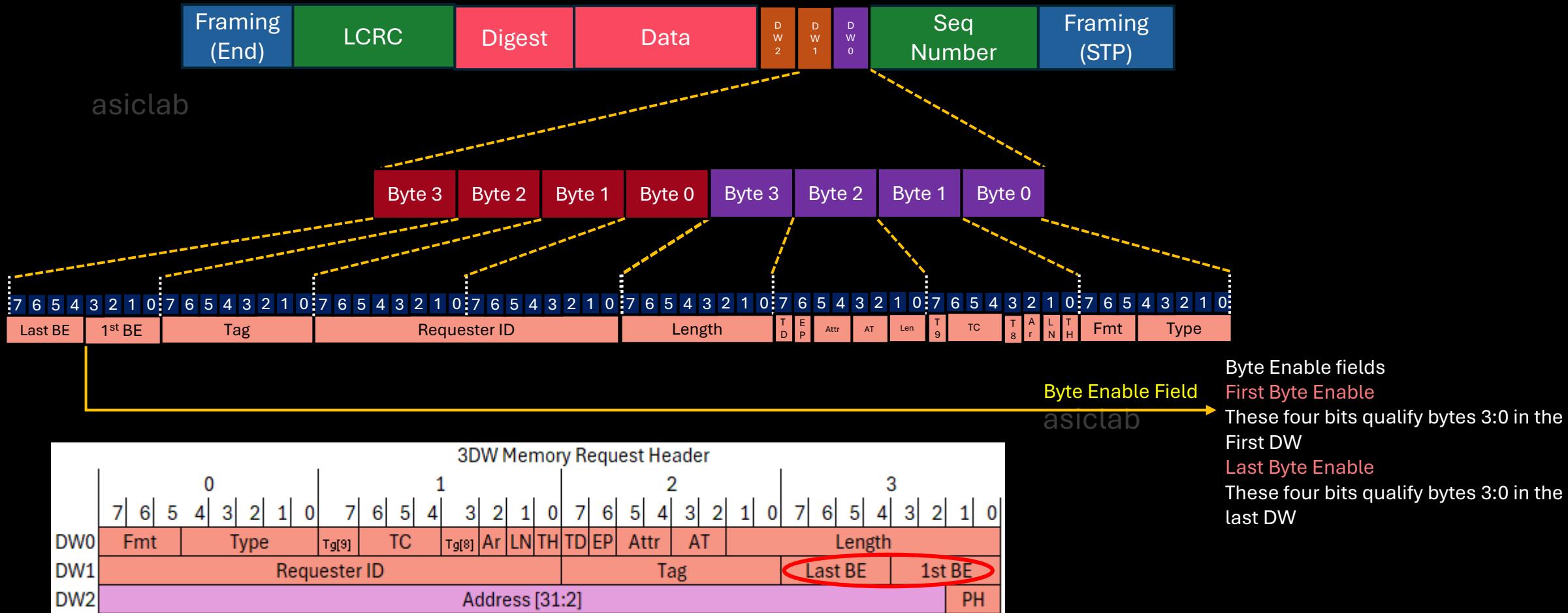
TLP Header format

If the length is 10 or 10 DW and the FBE and Last byte enable are 1100 and 0011 how many bytes are transferred

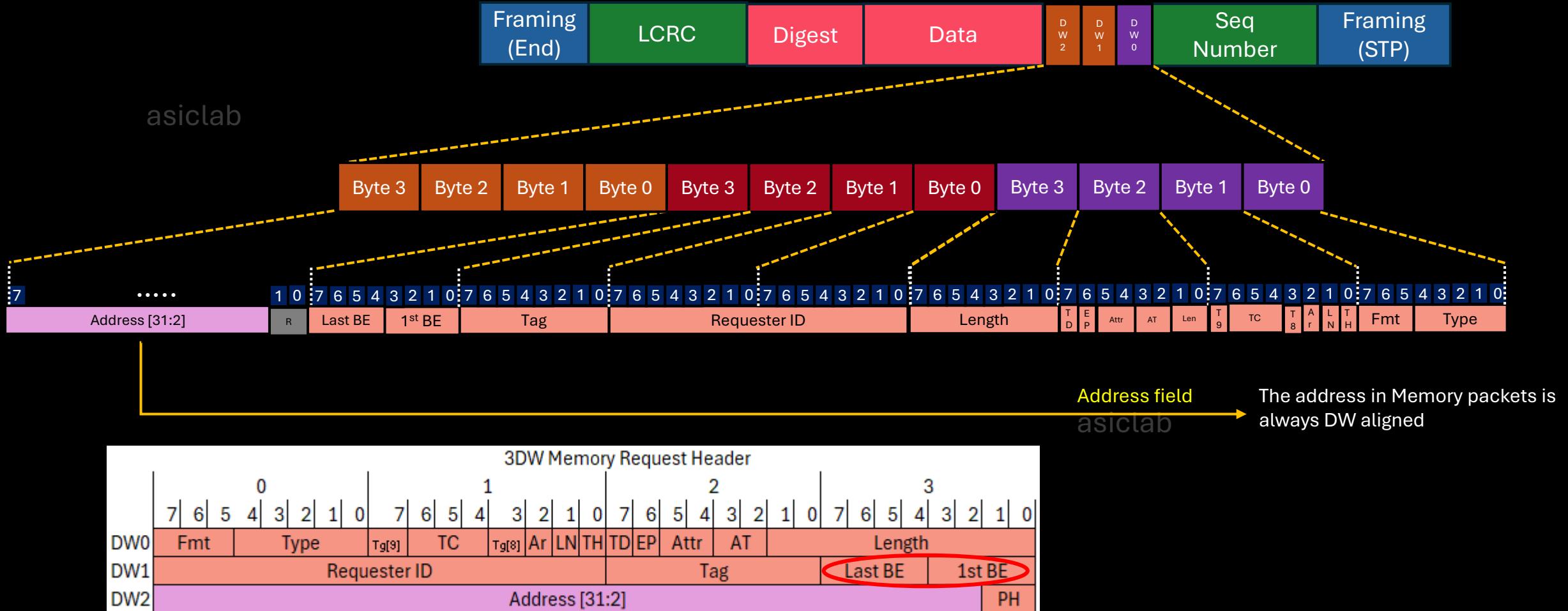


Answer: Total transferred bytes were: $10\text{DW} - 4\text{bytes} = (10 * 4\text{bytes}) - 4\text{bytes} = 40\text{bytes} - 4\text{bytes} = 36\text{ bytes}$

TLP Header format

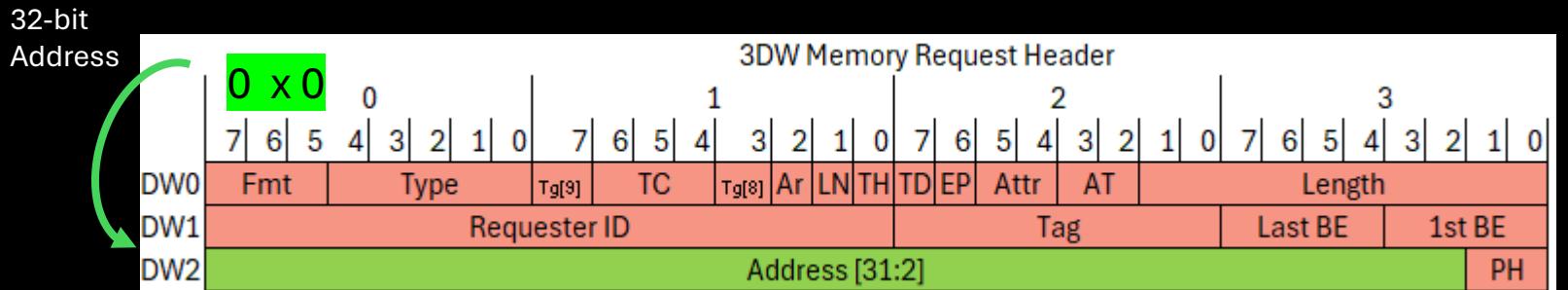
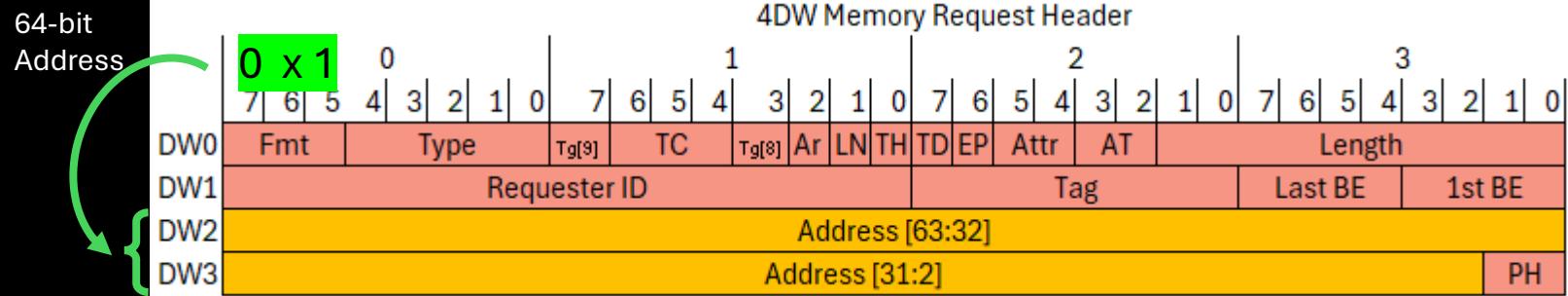


TLP Header format



TLP Header format

System Memory Map



2^{64}

2^{32}

0

- The address in Memory packets is always DW aligned.
- Memory TLPs targeting an address below 4GB (2^{32}) **MUST** use the 3DW header

asiclab

Completion TLP

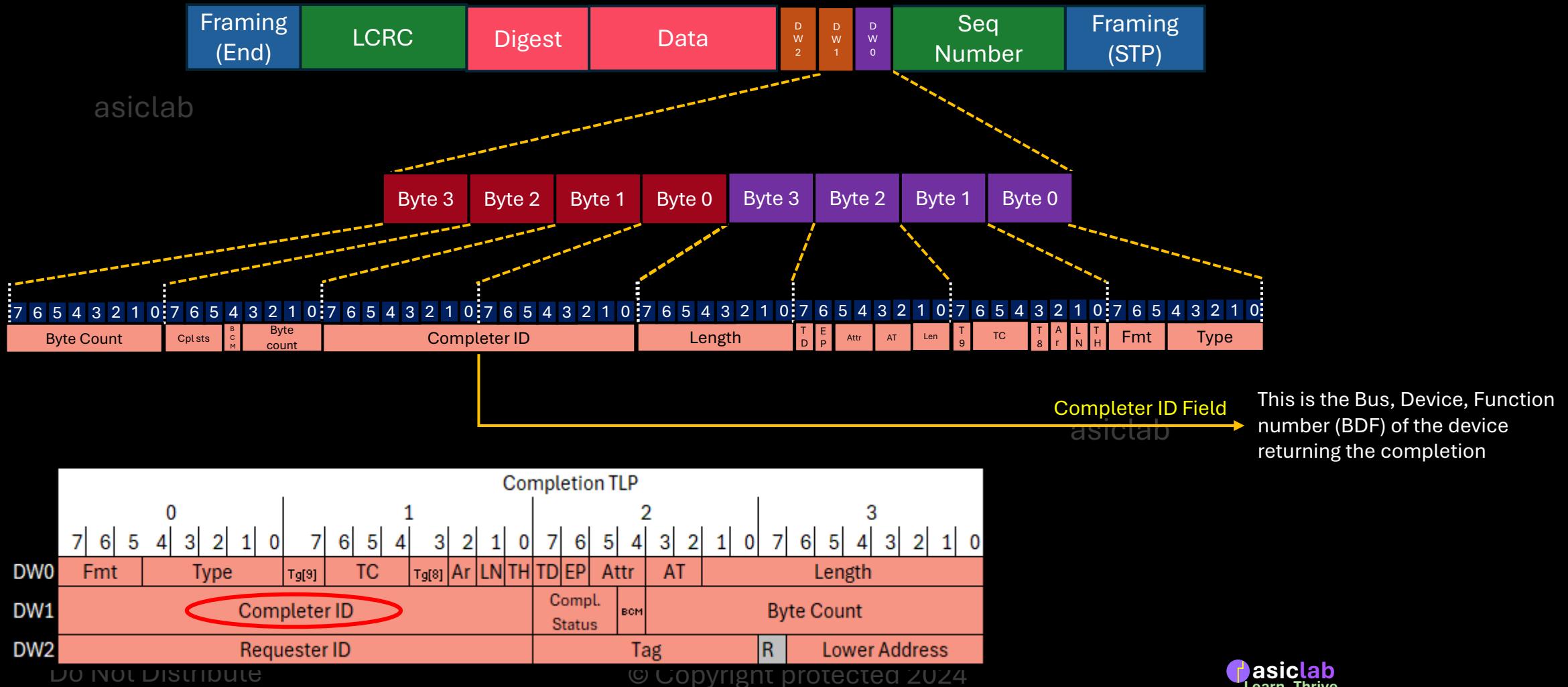
asiclab

Do Not Distribute

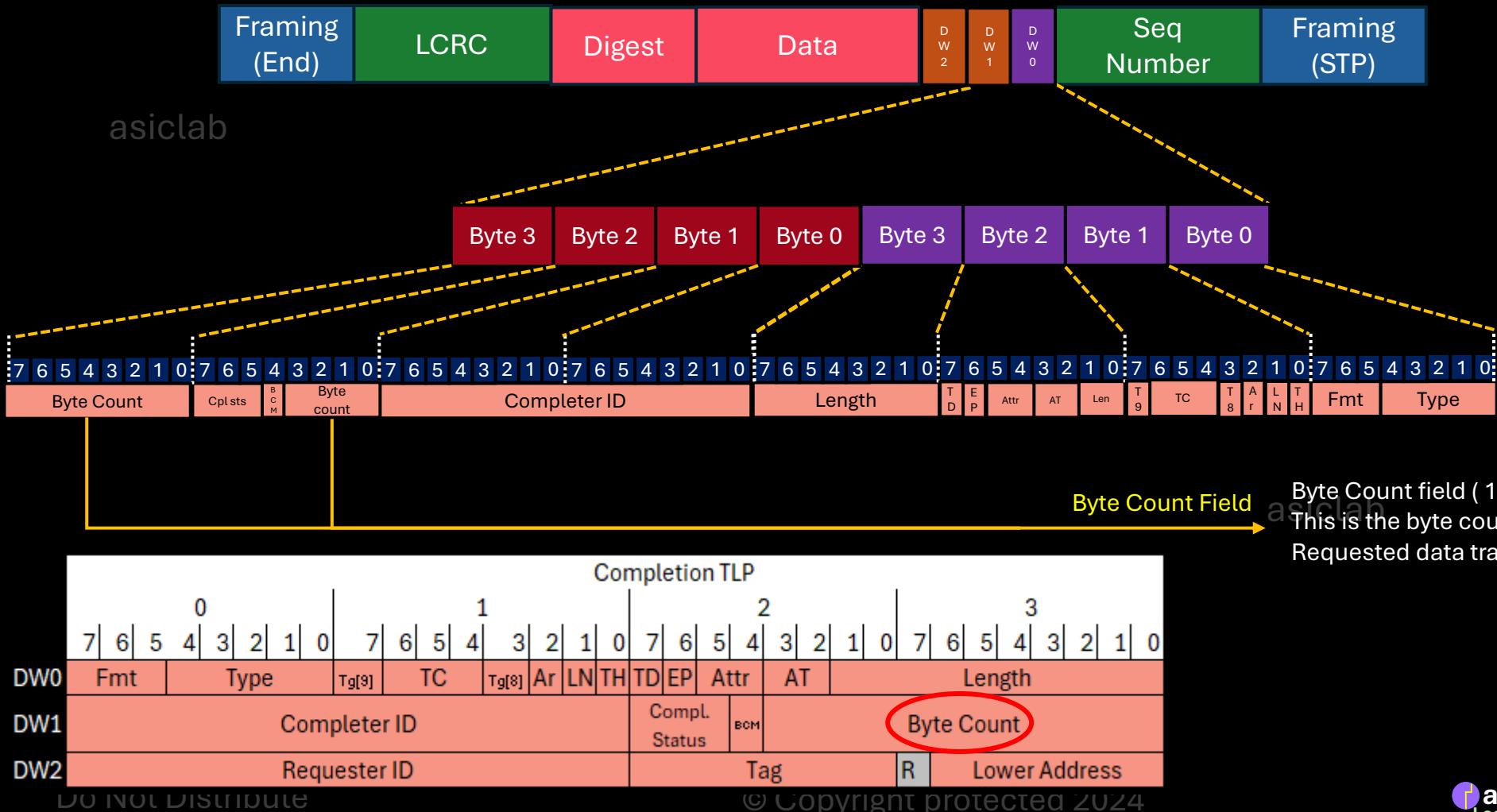
© Copyright protected 2024



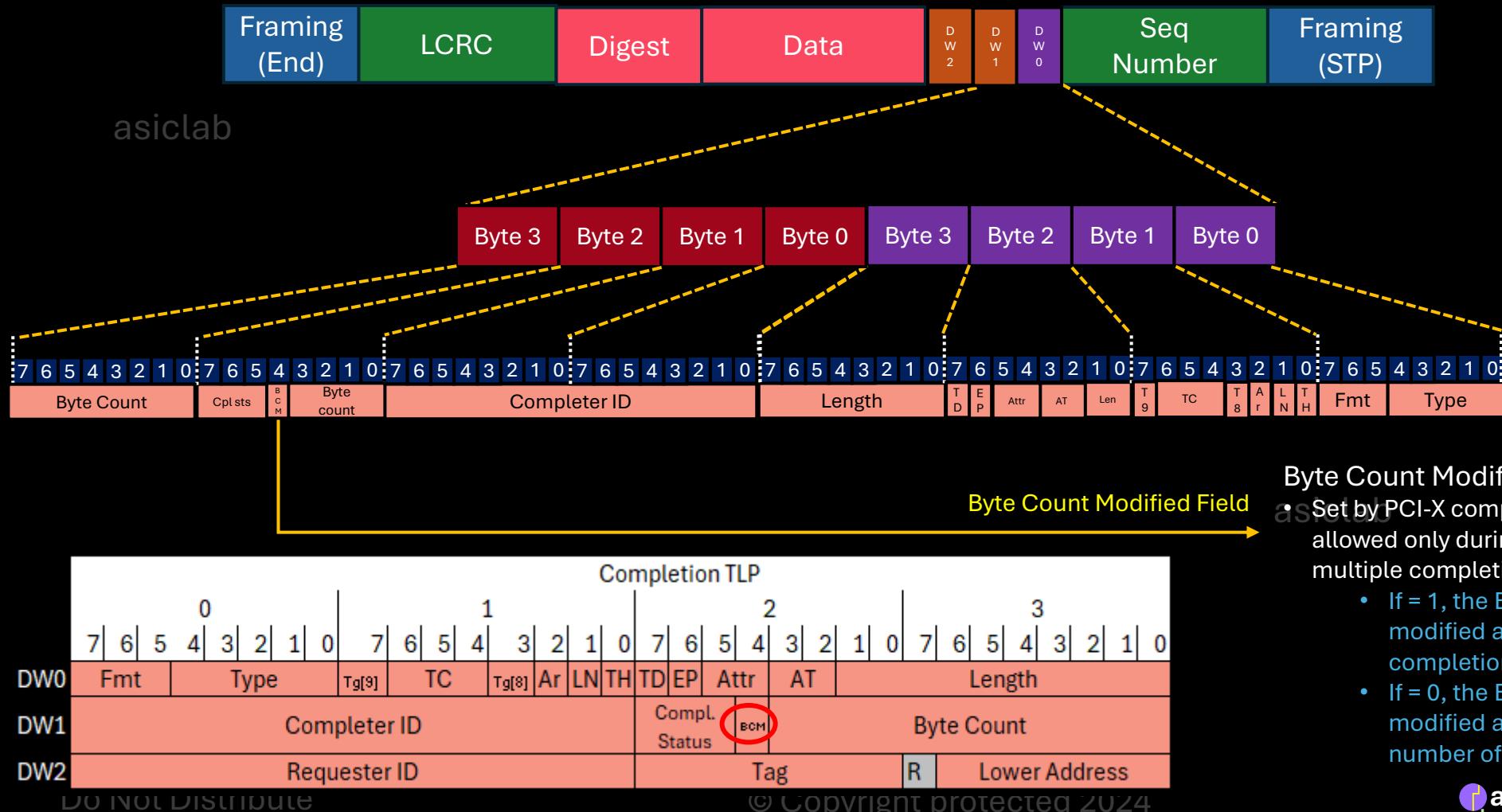
TLP Header format



TLP Header format



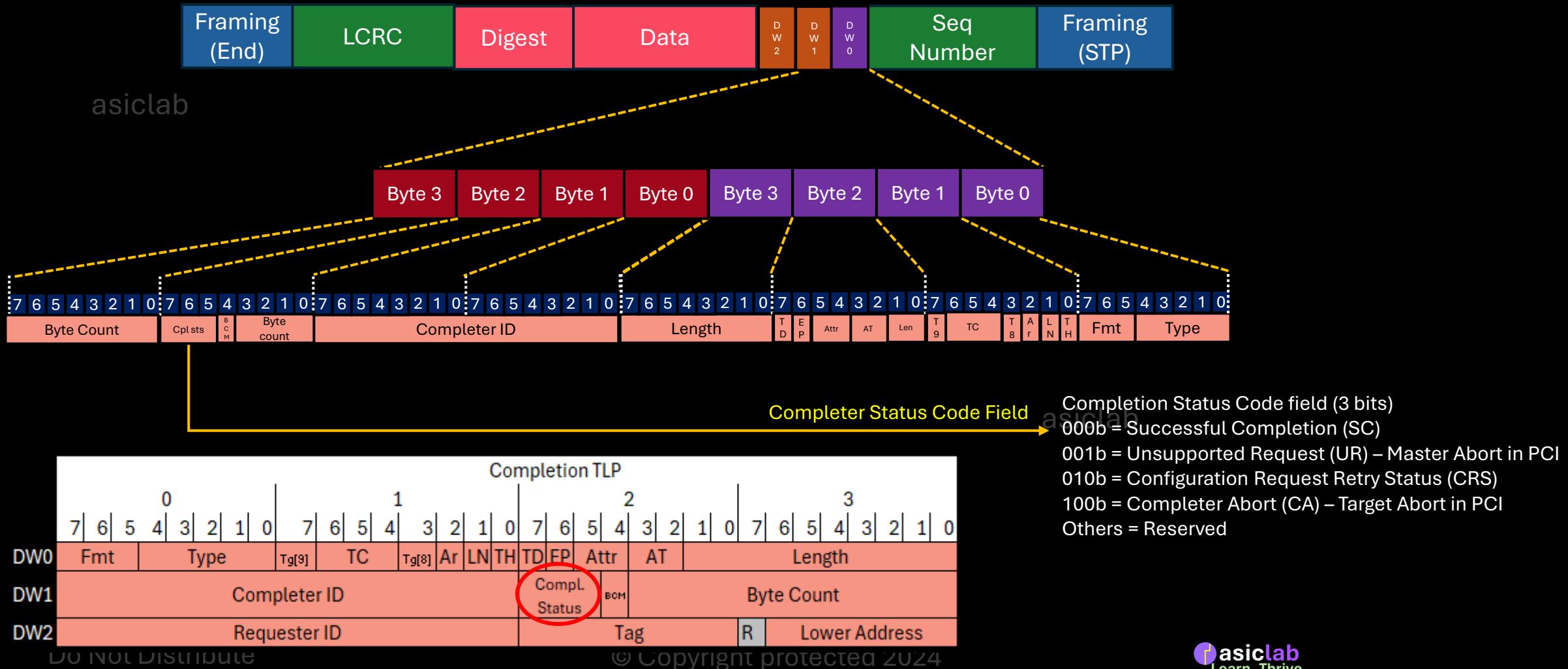
TLP Header format



Byte Count Modified field (1 bits)

- If = 1, the Byte Count field has been modified and contains the count for this completion only, not the total remaining
 - If = 0, the Byte Count field has not been modified and reflects the total remaining number of bytes to transfer

TLP Header format

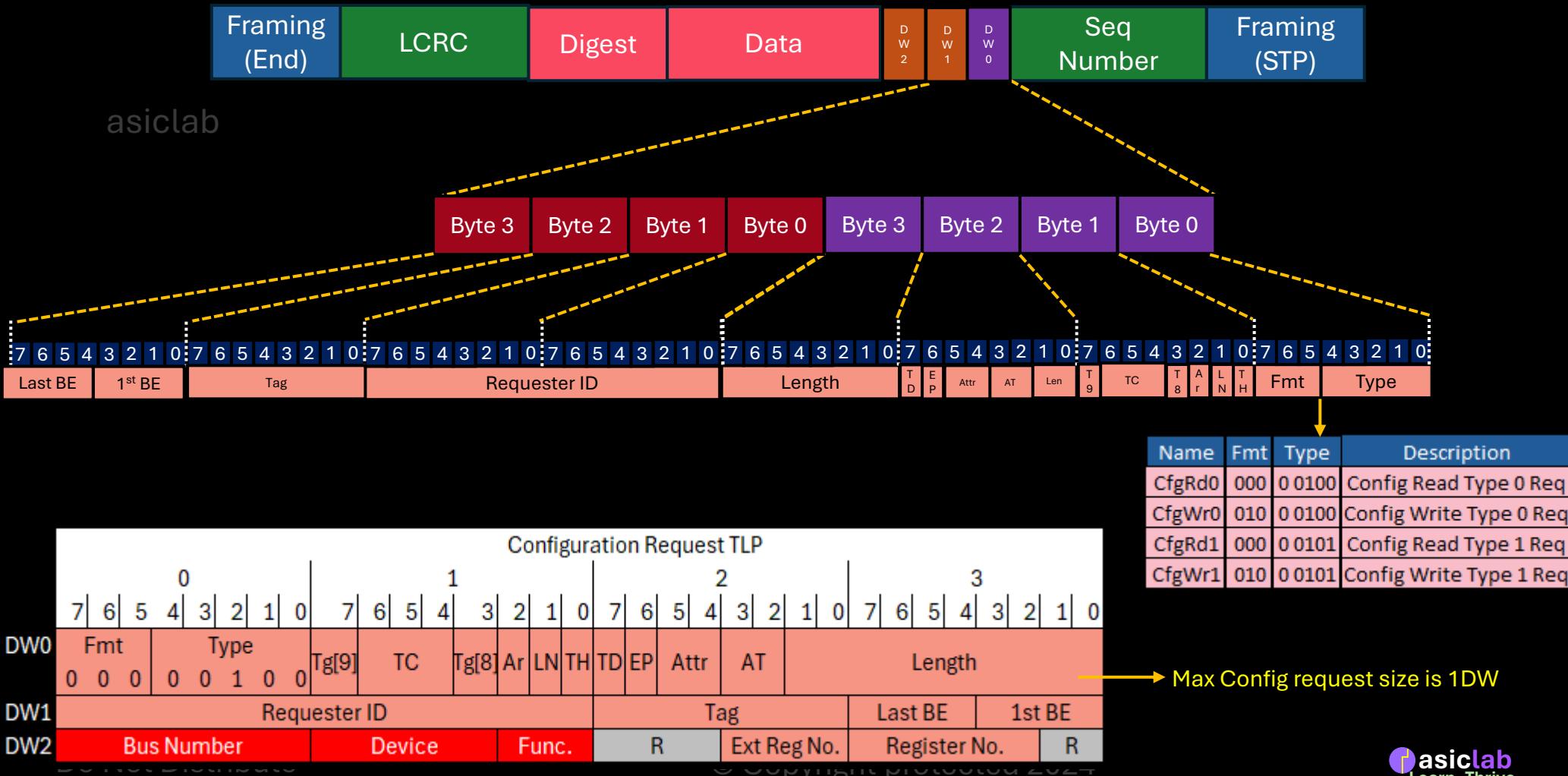


asiclab

Configuration Request TLP

asiclab

TLP Header format



asiclab

IO Request TLP

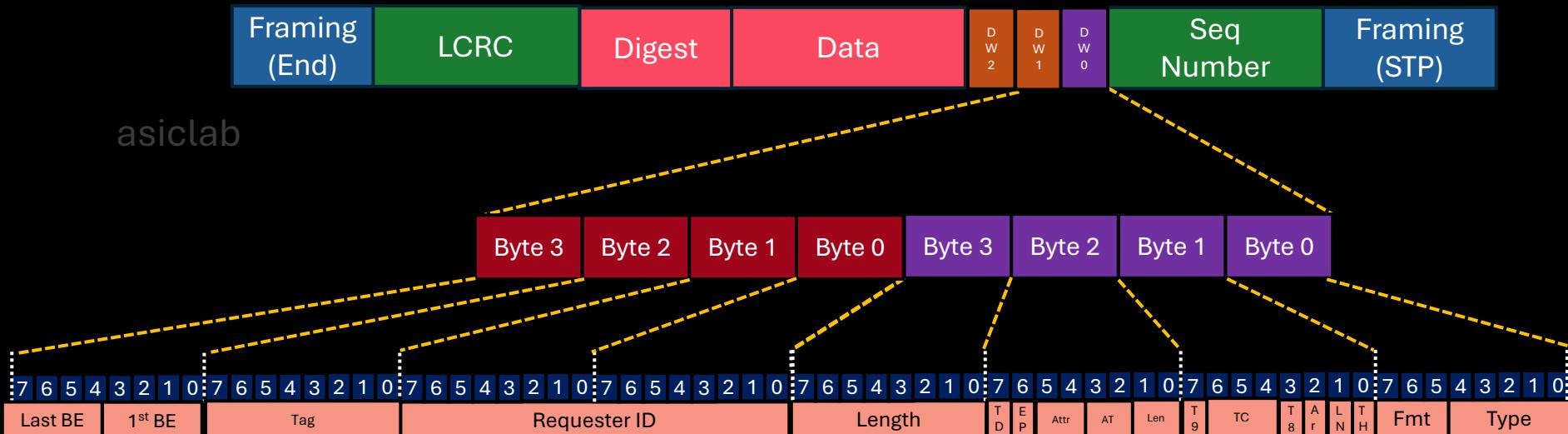
asiclab

Do Not Distribute

© Copyright protected 2024

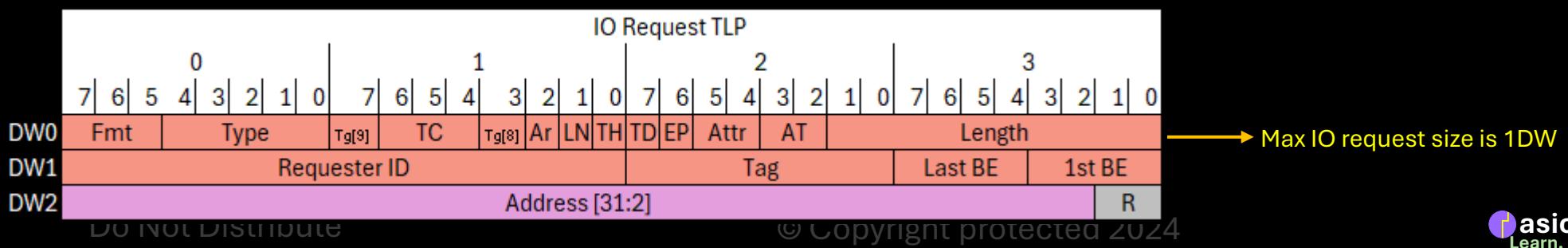


TLP Header format



- IO requests are routed based on address
- PCIe supports a max IO address space of 32 bits

asiclab

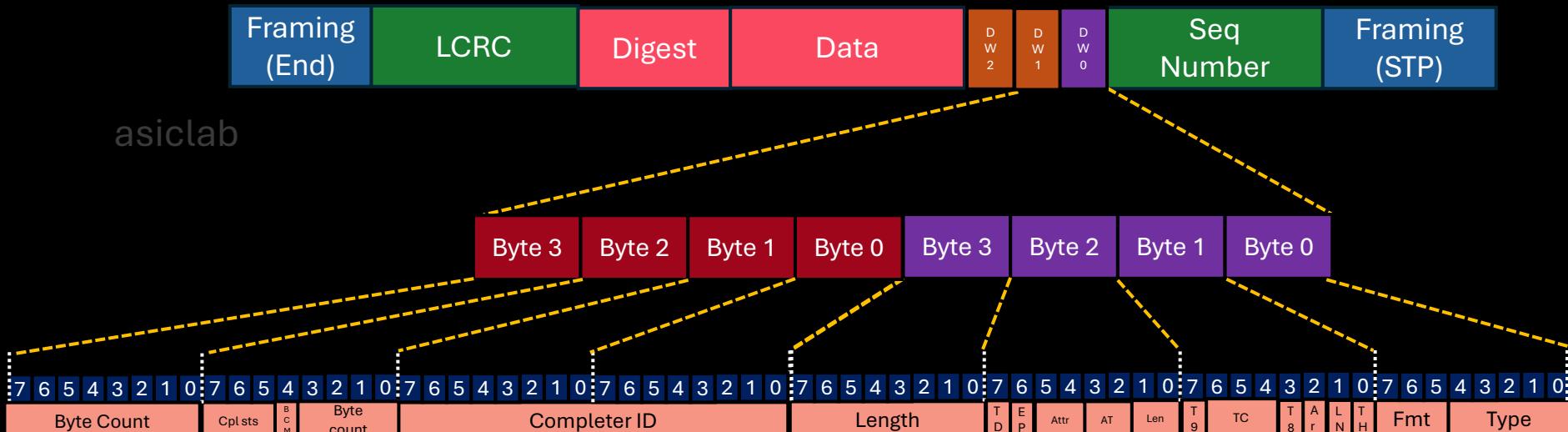


asiclab

Message or Vendor Defined Message TLP

asiclab

TLP Header format



- Message transactions are Posted and consist of a Request only
- Message transactions may or may not have a data payload

asiclab

Vendor-Defined Message Header																						
DW0	Fmt	Type	T _{g[9]}	TC	T _{g[8]}	Ar	LN	TH	TD	EP	Attr	AT	Length									
DW1	Requester ID								Tag			Message Code										
DW2	Target BDF if ID Routing used, otherwise Reserved								Vendor ID													
DW3	For Vendor definition																					

TLP Header format

Type field definition

- Byte 0, bits 4:3: 10b (Message TLP)
- Byte 0, bits 2:0:r r r (Message Routing Sub-Field)
 - 000b = Route to Root Complex (Implicit)
 - 001b = Route by Address (Uses Address fields)
 - 010b = Route by ID (Uses Requester ID field)
 - 011b = Broadcast by Root Complex (Implicit)
 - 100b = Local – Terminate at Receiver (Implicit)
 - 101b = Gather and route to Root Complex (Implicit)
 - Others = Reserved



Vendor-Defined Message Header																					
DW0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0					
	Fmt	Type	Tg[9]	TC	Tg[8]	Ar	LN	TH	TD	EP	Attr	AT			Length						
DW1	Requester ID						Tag				Message Code										
DW2	Target BDF if ID Routing used, otherwise Reserved								Vendor ID												
DW3	For Vendor definition																				

asiclab

TLP Header format

Message Code field (8 bits)

0000 0000b = Unlock Message

0001 0000b = Latency Tolerance Reporting Message (LTR)

0001 0010b = Optimized Buffer Flush/Fill Message (OBFF)

0001 xxxx b = Power Management Messages

0010 xxxx b = Legacy Interrupt Messages (INTx)

0011 00xx b = Error Messages

0100 xxxx b = Ignored Messages

0101 0000b = Set Slot Power Limit Message

0101 0010b – 0101 0011b = Precision Time Measurement (PTM)

0111 111xb = Vendor-Defined Messages

Vendor-Defined Message Header																				
DW0	Fmt	Type	0	1	2	3	4	5	6	7	Tg[9]	TC	1	3	2	1				
DW1	Requester ID				Tag				Message Code											
DW2	Target BDF if ID Routing used, otherwise Reserved								Vendor ID											
DW3	For Vendor definition																			

TLP Header format

Message Code field (8 bits)

0111 1110b = Type 0 Vendor-Defined Message

Completer treats as Unsupported Request (UR) if not supported

0111 1111b = Type 1 Vendor-Defined Message

Completer silently drops (no error) if not supported

Allow routing types:

Route to Root (000b), Route by ID(010b), Broadcast by Root (011b), Local (100b)

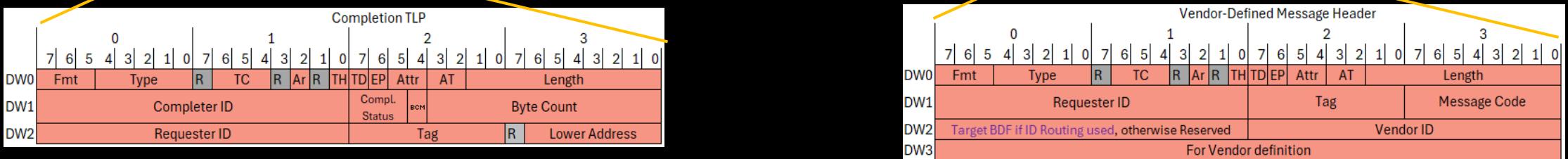
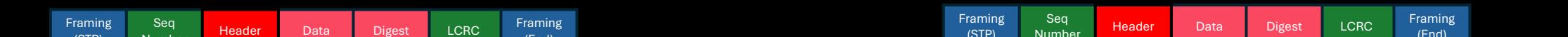
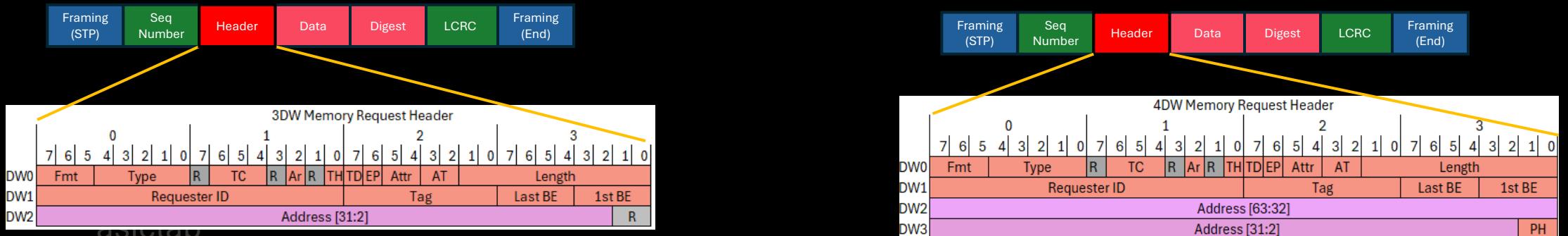
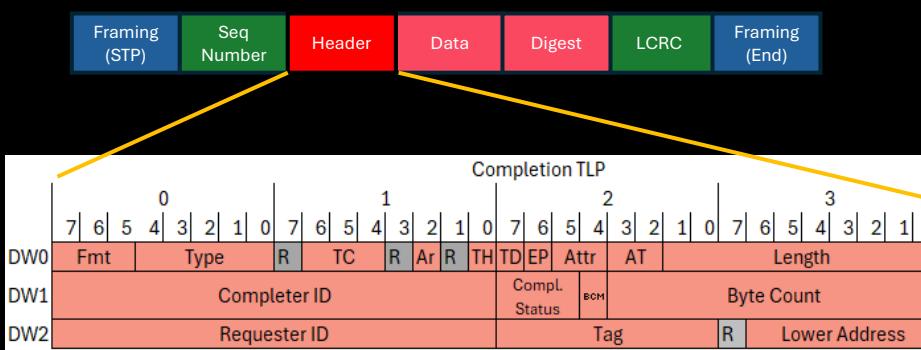
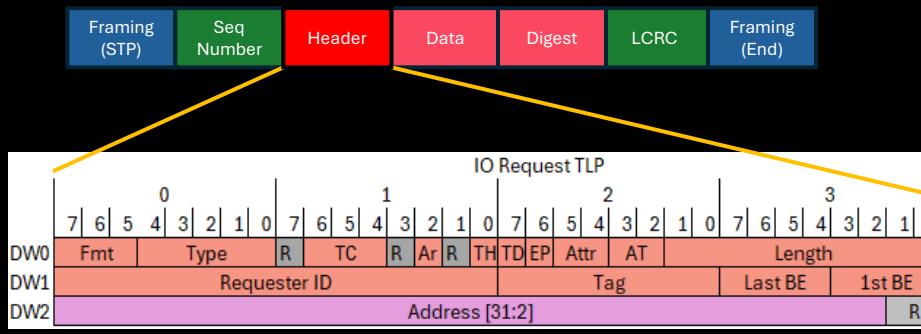
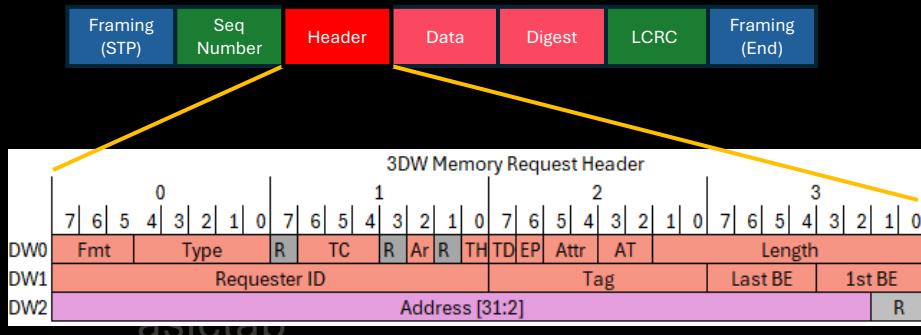
Vendor-Defined Message Header																				
DW0	Fmt	0	4	3	2	1	0	7	6	5	4	1	3	2	1	0				
DW1	Requester ID				Tag				Message Code											
DW2	Target BDF if ID Routing used, otherwise Reserved								Vendor ID											
DW3	For Vendor definition																			

TLP Header format

- The PCI-SIG has defined their own “Vendor-Defined Messages”
 - Uses Type 1 Vendor-Defined Message with a Vendor ID of 0001h (PCI-SIG)
- Subtype field defines the specific message type:
 - 0000 0000b = Lightweight Notification (LN)
 - 0000 1000b = Device Readiness Status (DRS)
 - 0000 1001b = Function Readiness Status (FRS)

Vendor-Defined Message Header															
DW0	Fmt	0	4	3	2	1	0	7	6	5	4	3	2	1	0
	Type	Tg[9]	TC	Tg[8]	Ar	LN	TH	TD	EP	Attr	AT	Length			
DW1	Requester ID								Tag		Message Code				asiclab
DW2	Target BDF if ID Routing used, otherwise Reserved								Vendor ID						
DW3	Subtype		For Vendor definition												

TLP Header format



asiclab

PCIe Gen 6.0

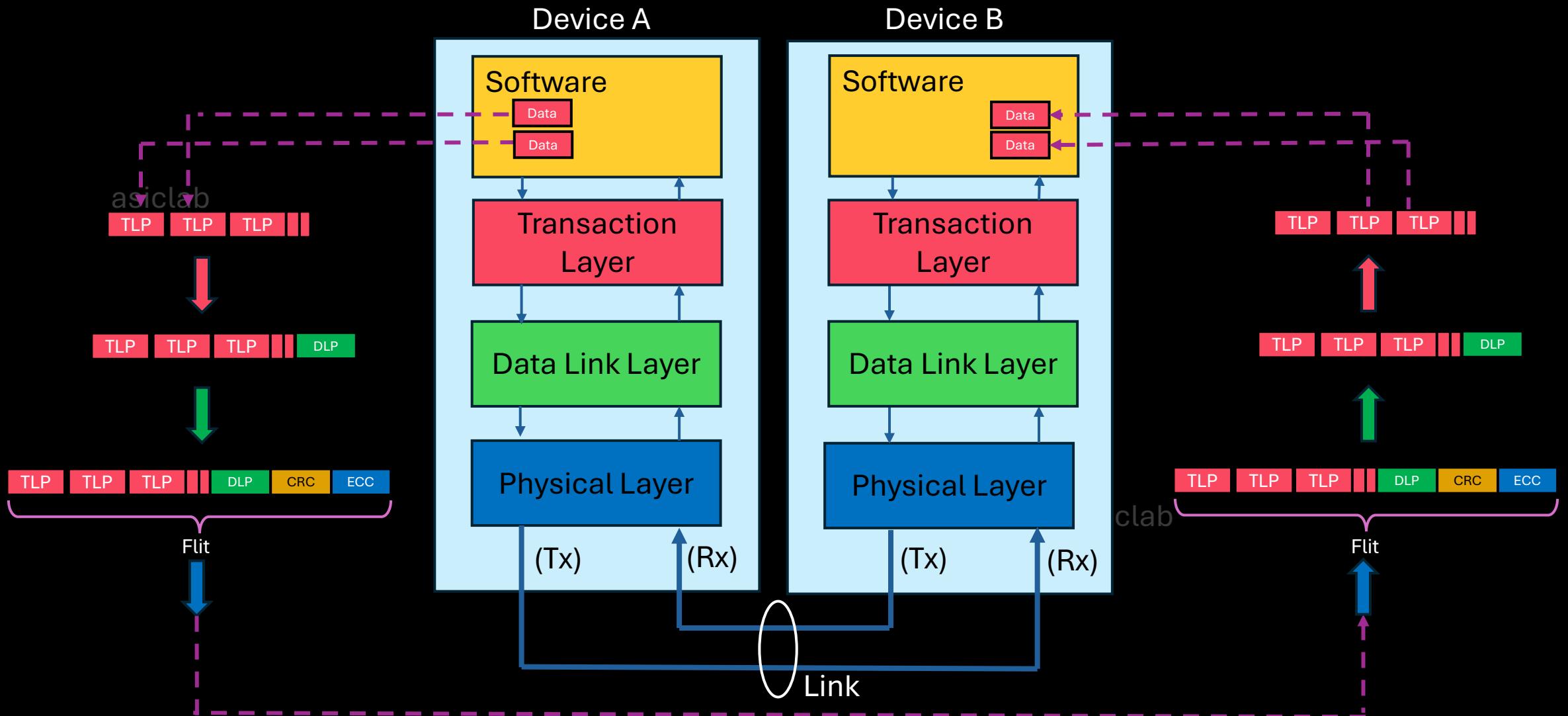
asiclab

Do Not Distribute

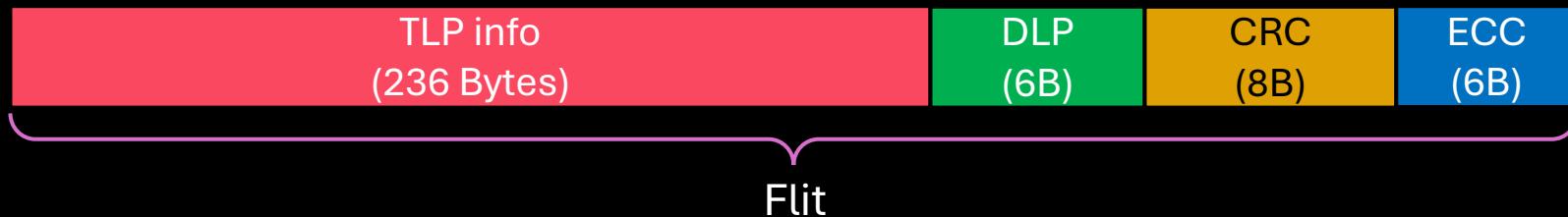
© Copyright protected 2024



Flit Assembly/Disassembly (New with PCIe 6.0)



What is a Flit: (New with PCIe 6.0)

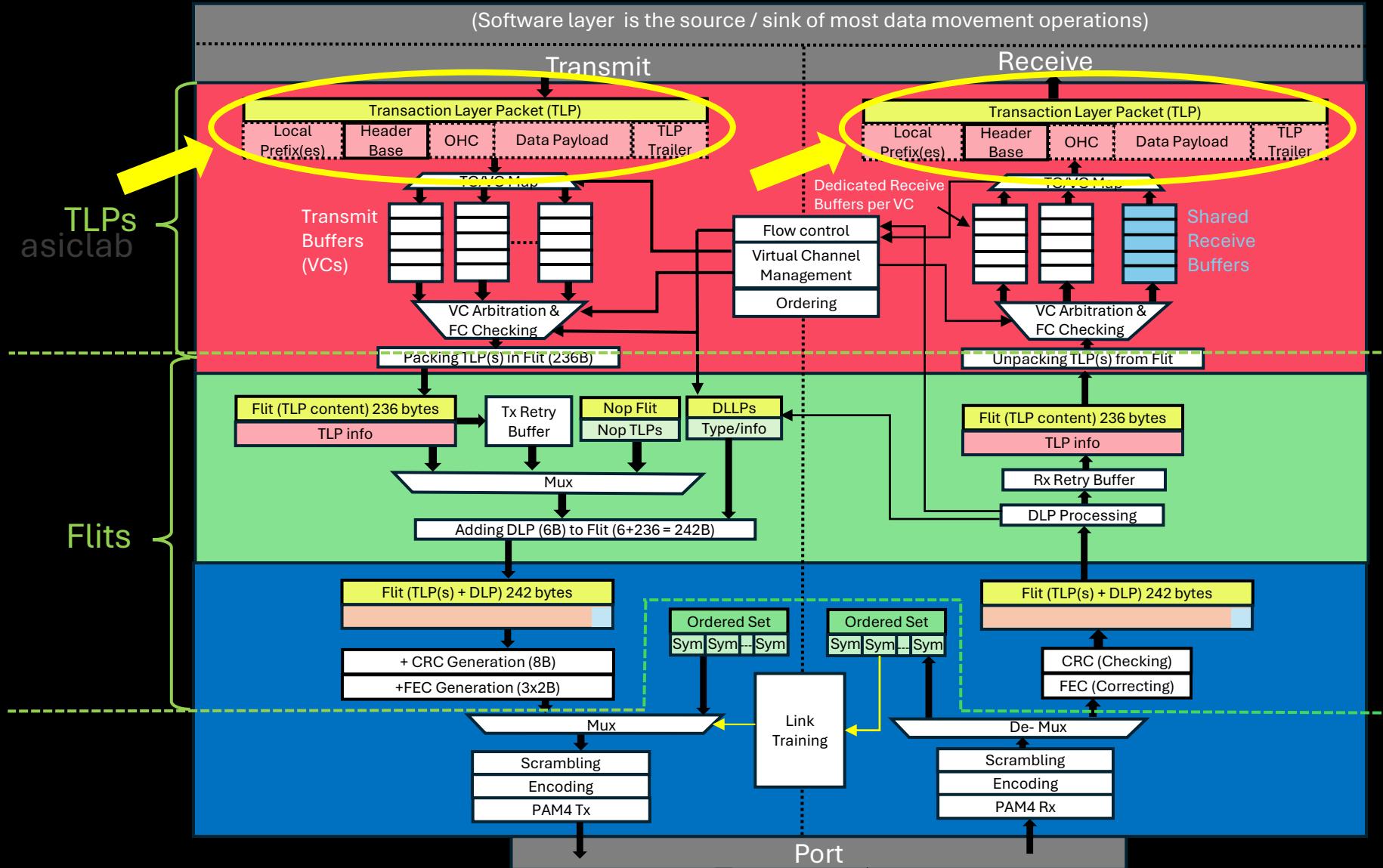


asiclab

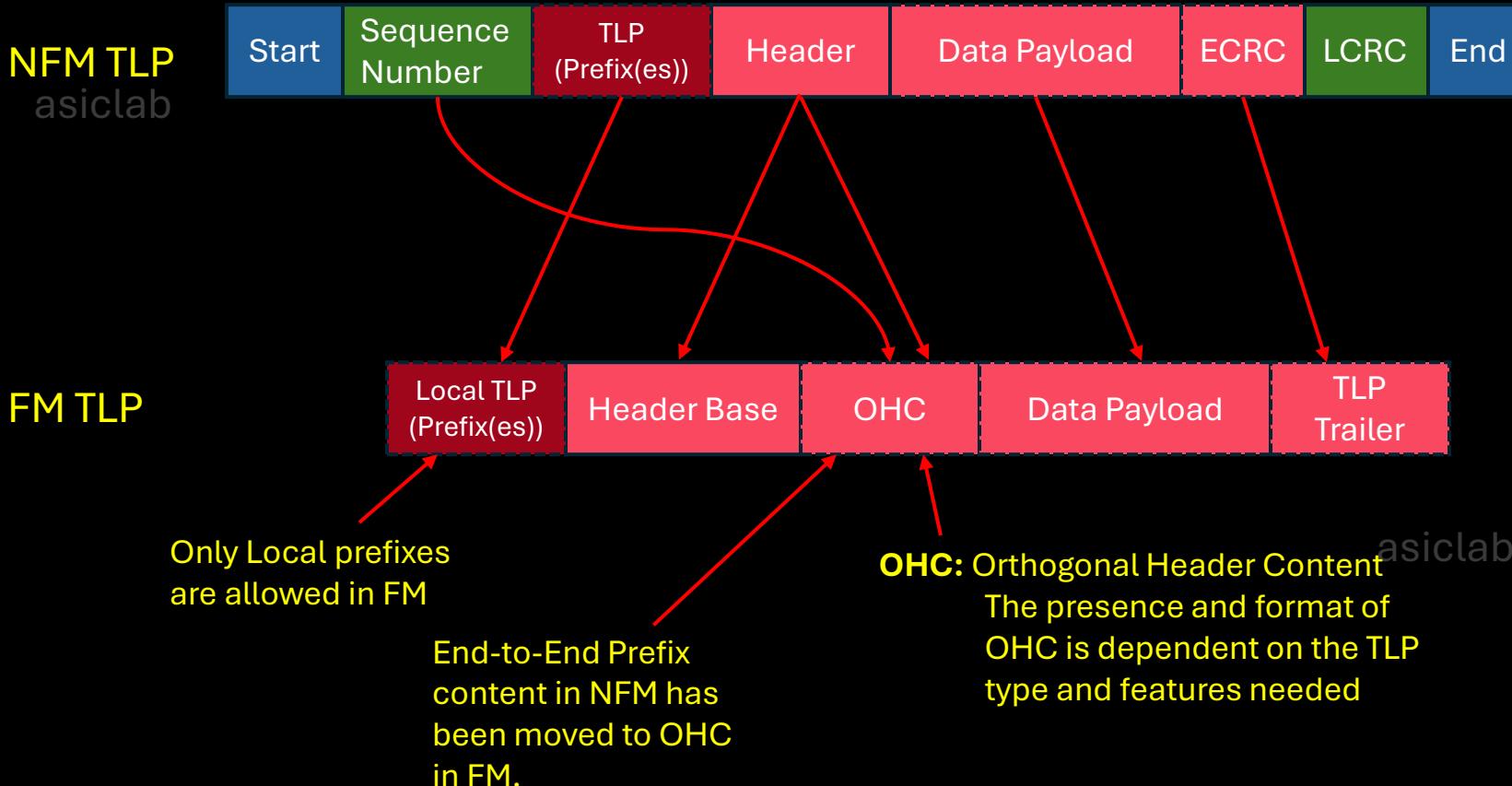
- From Gen 6 Flit mode which is applicable for any speeds lower than Gen6 if the device supports it.
- While in Flit mode the basic unit of transfers are not TLP and DLLP packet anymore, instead all of them are now packed in Flits
- A Flit is 256 bytes in length:
 - 236 bytes of TLP traffic
 - 6 bytes of DLP traffic
 - 8 bytes of Flit CRC
 - 6 bytes of Flit ECCs (FEC info)
- One or more TLP can be packed within a Flit, some TLPs can span across multiple Flits based on the payload size
- ACK/NAK and Retry mechanism is not applicable at Flit level and not for individual TLPs

asiclab

TLP Packing within a Flit (Flit Mode – FM) (1b/1b)

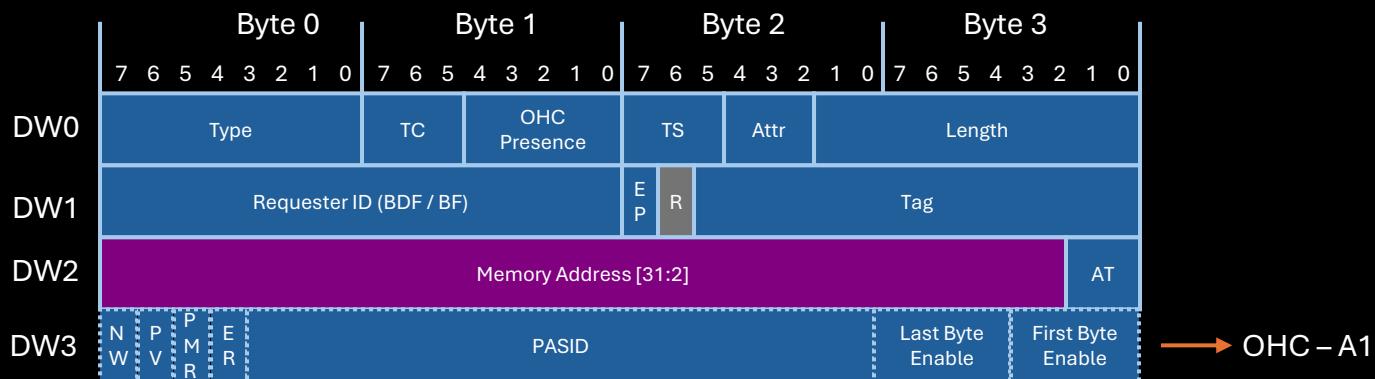


TLP Structure PCIe 6.0 (FM) (New with PCIe 6.0)



Flit format

asiclab



asiclab

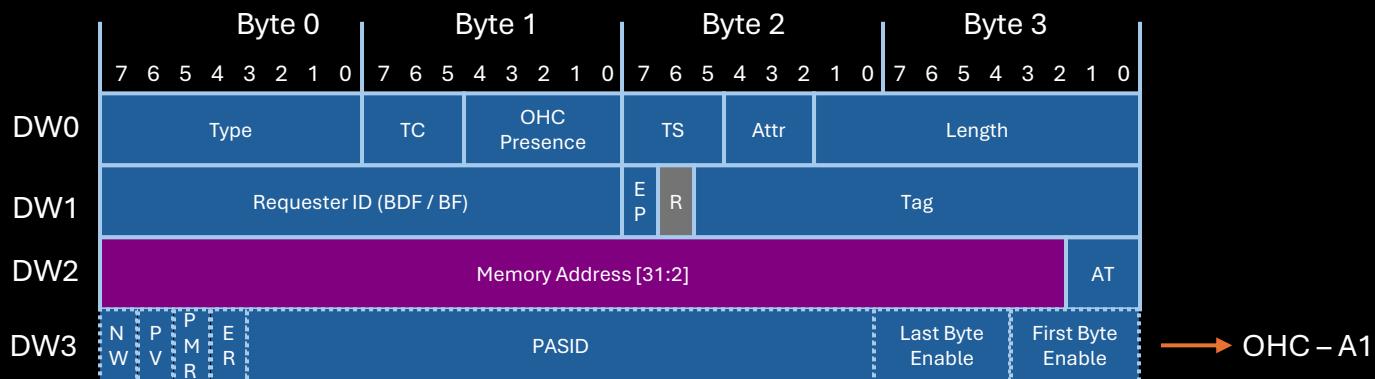
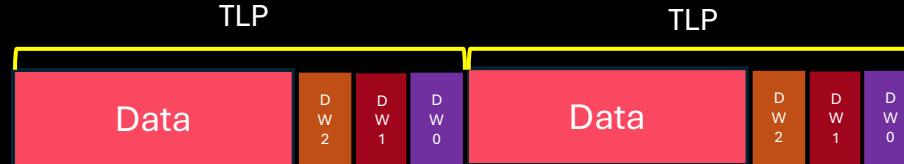
→ OHC - A1

Do Not Distribute

© Copyright protected 2024

Flit format

asiclab



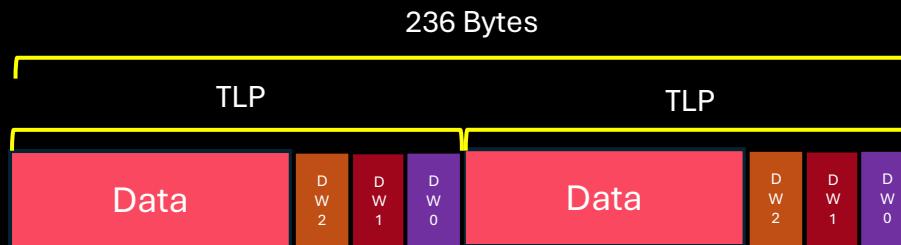
asiclab

→ OHC – A1

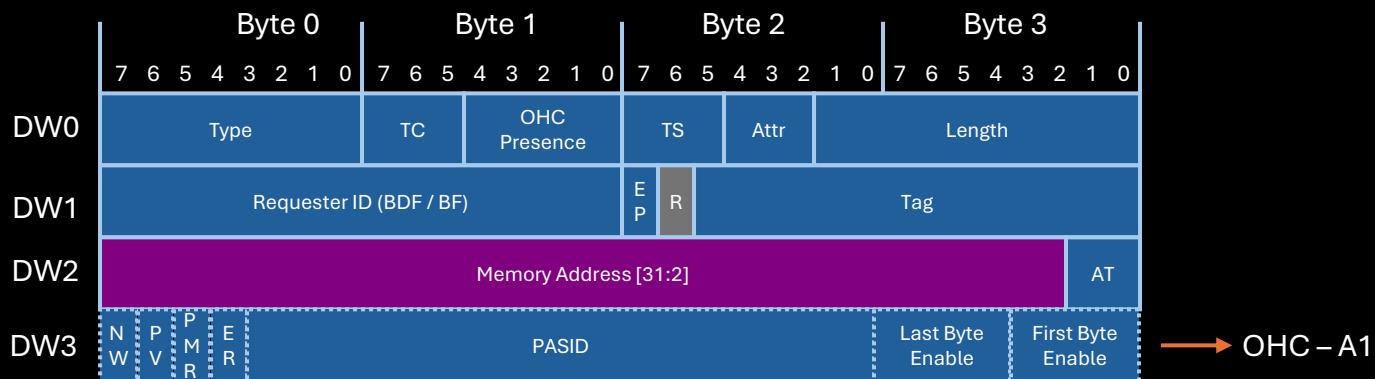
Do Not Distribute

© Copyright protected 2024

Flit format



asiclab

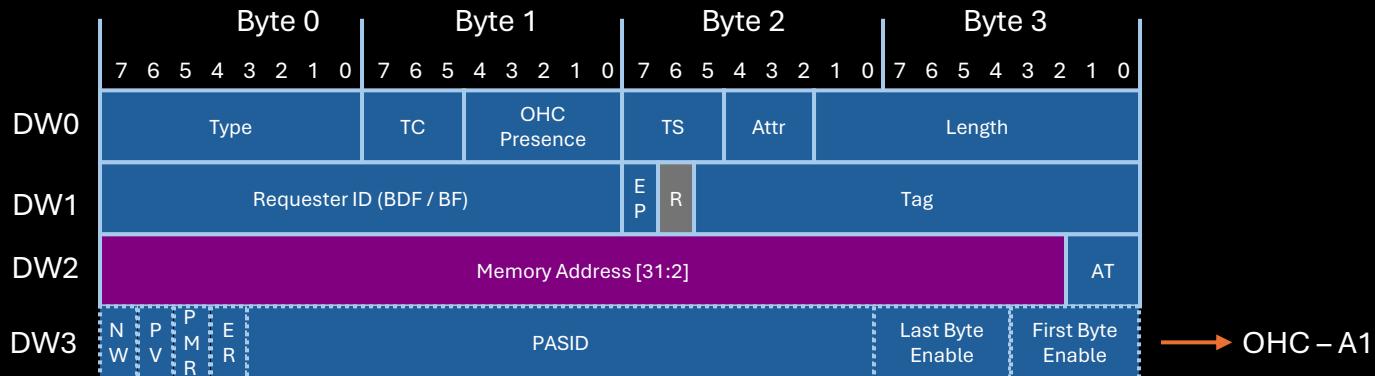
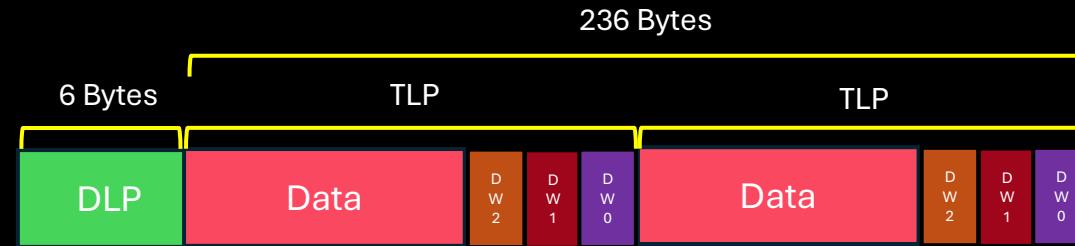


Do Not Distribute

© Copyright protected 2024

Flit format

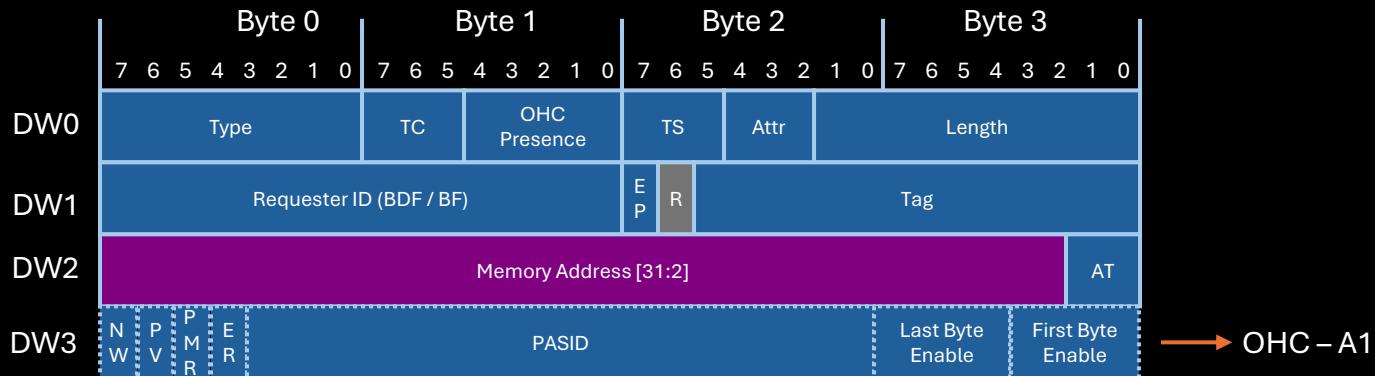
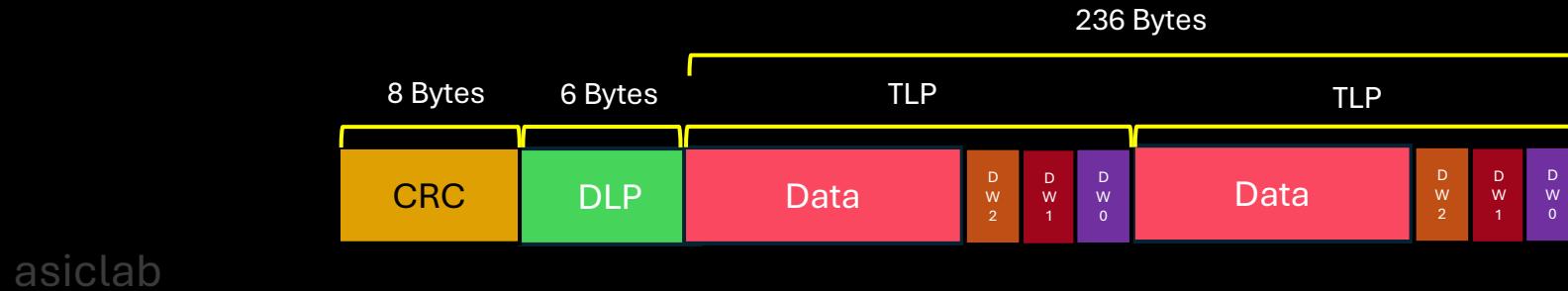
asiclab



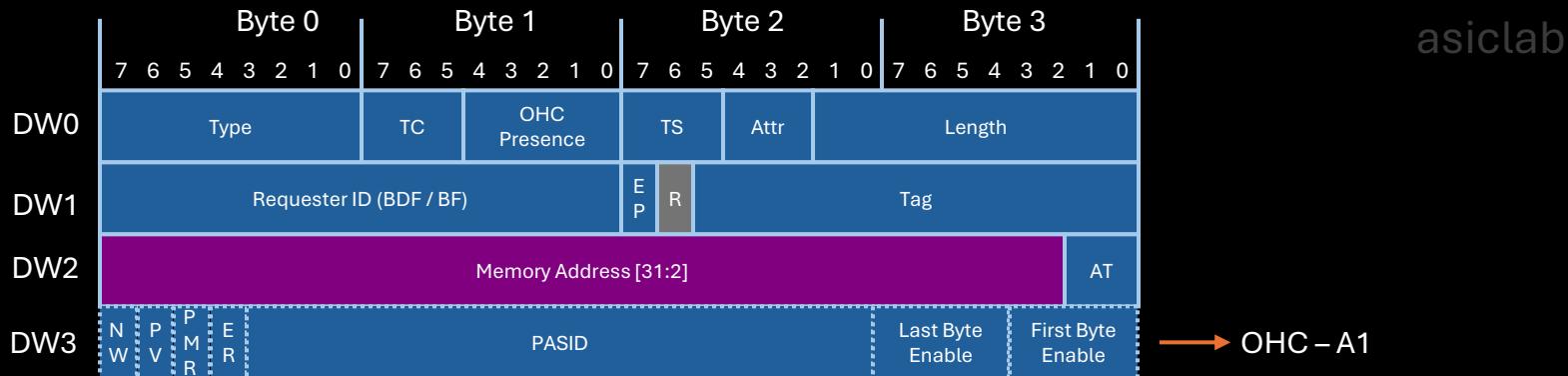
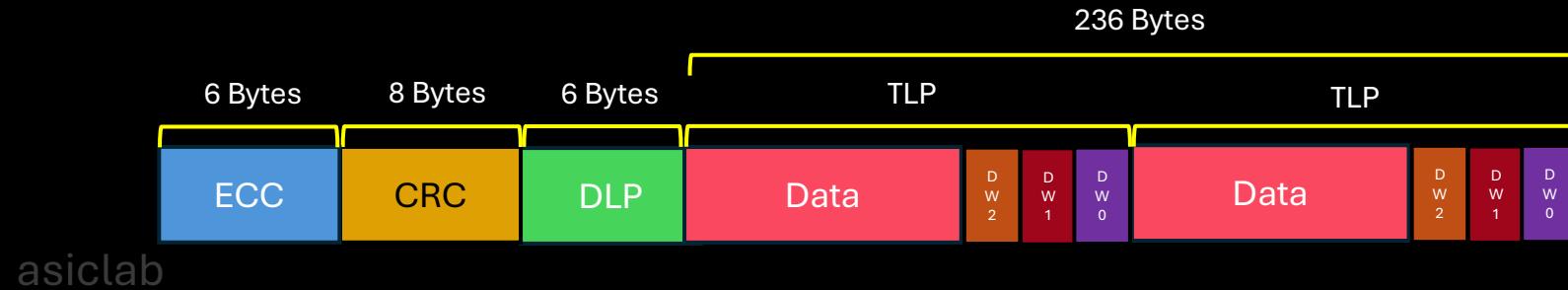
Do Not Distribute

© Copyright protected 2024

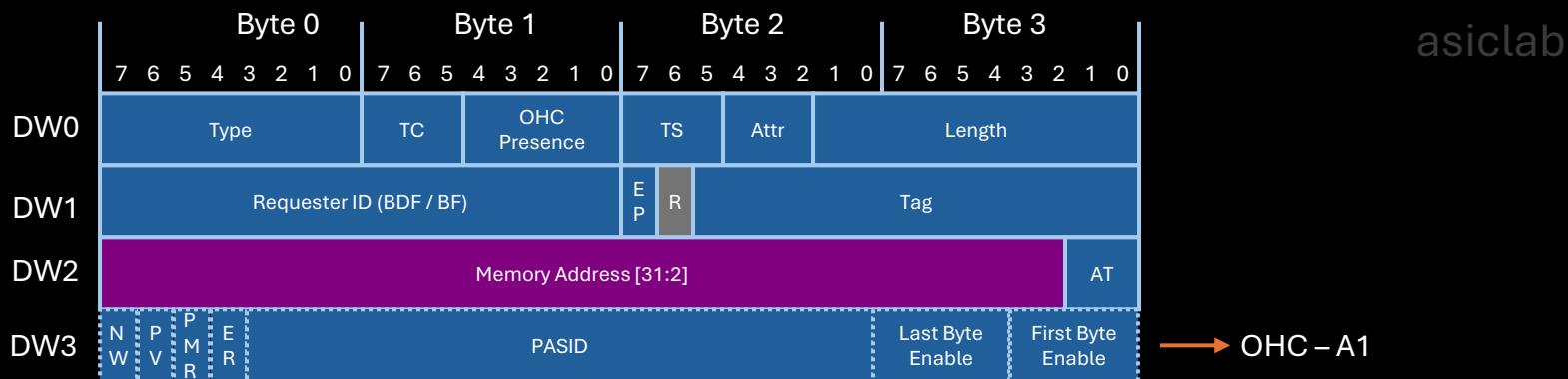
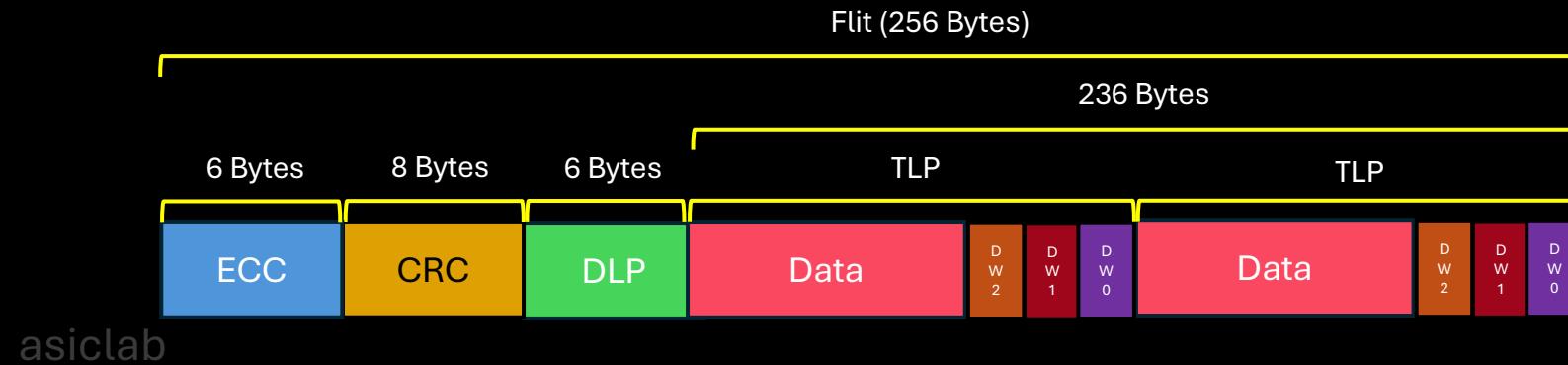
Flit format



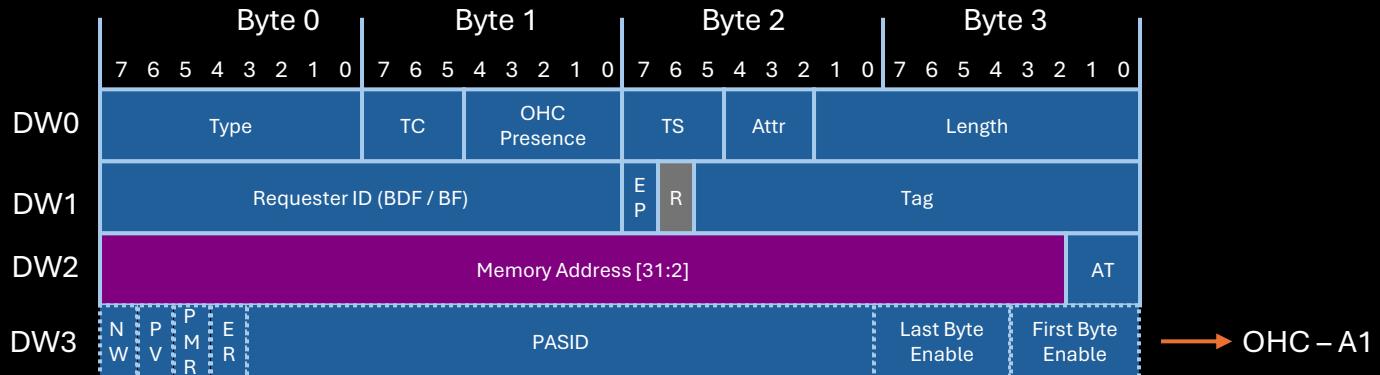
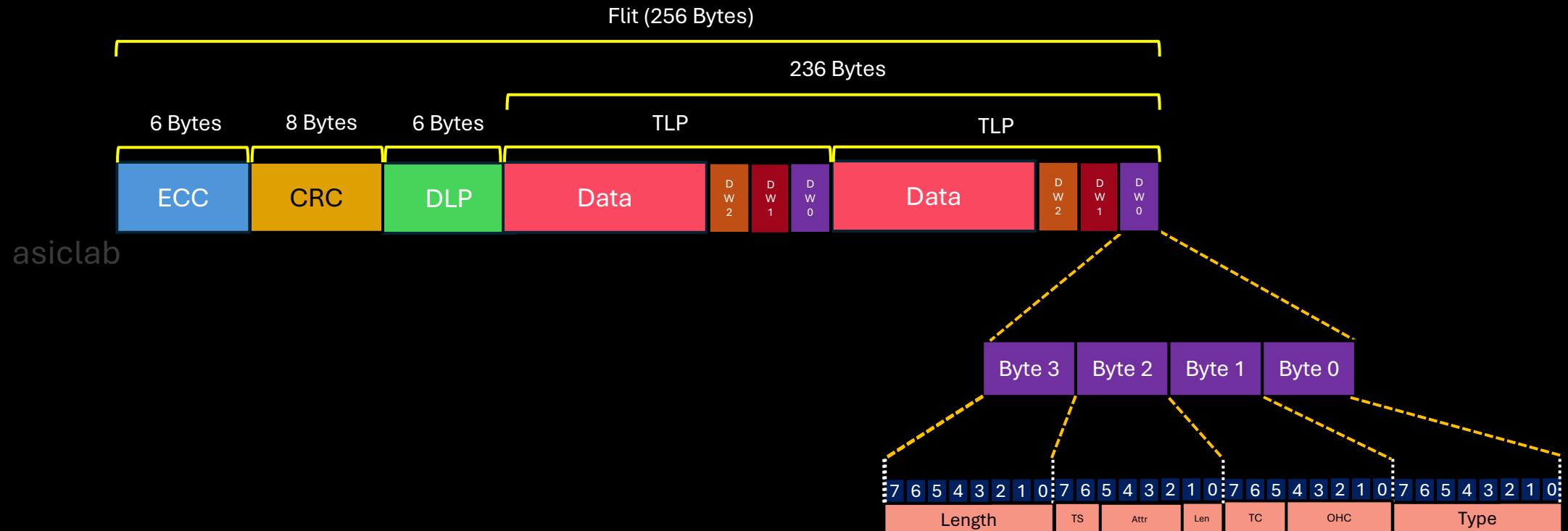
Flit format



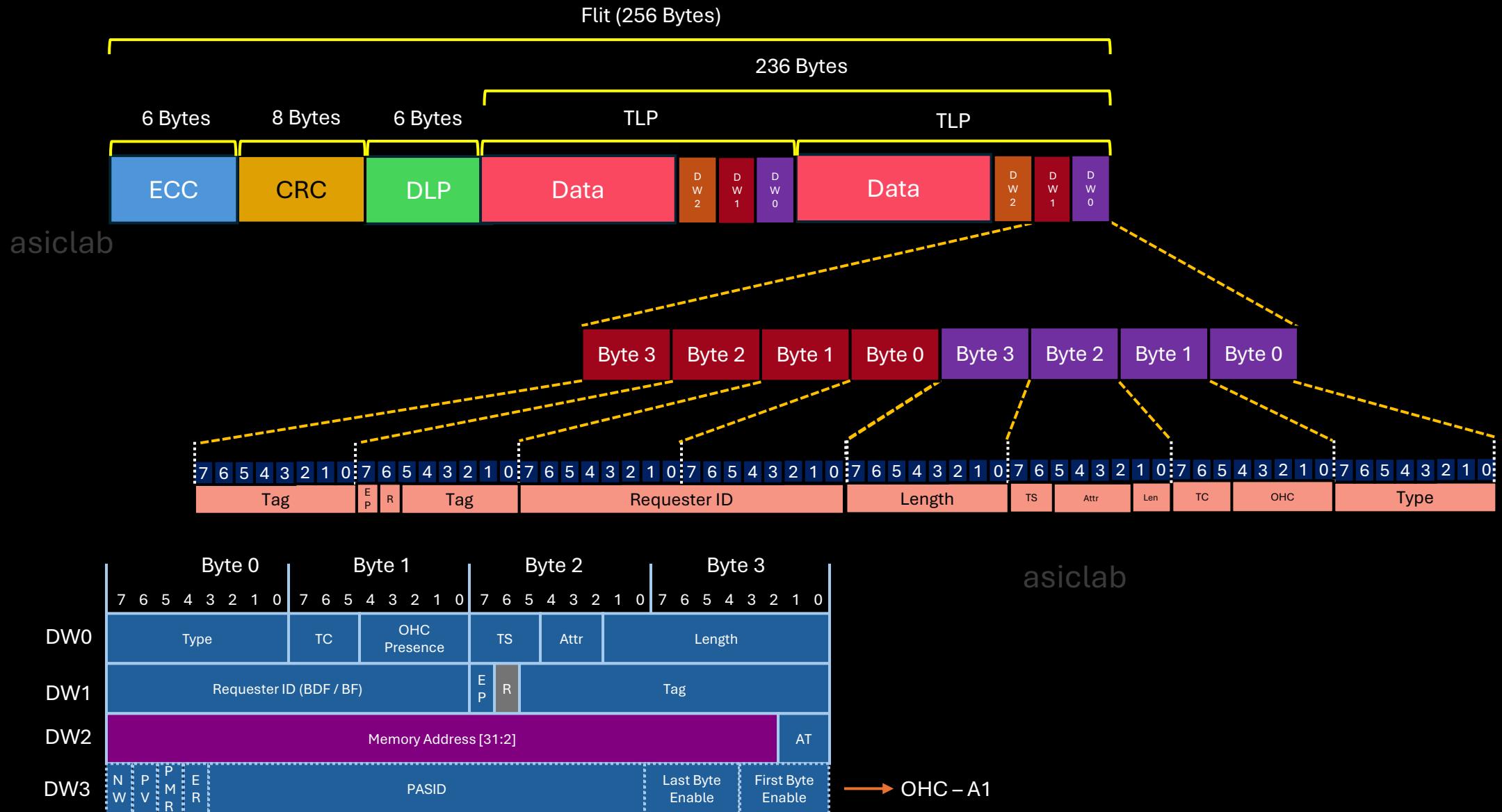
Flit format



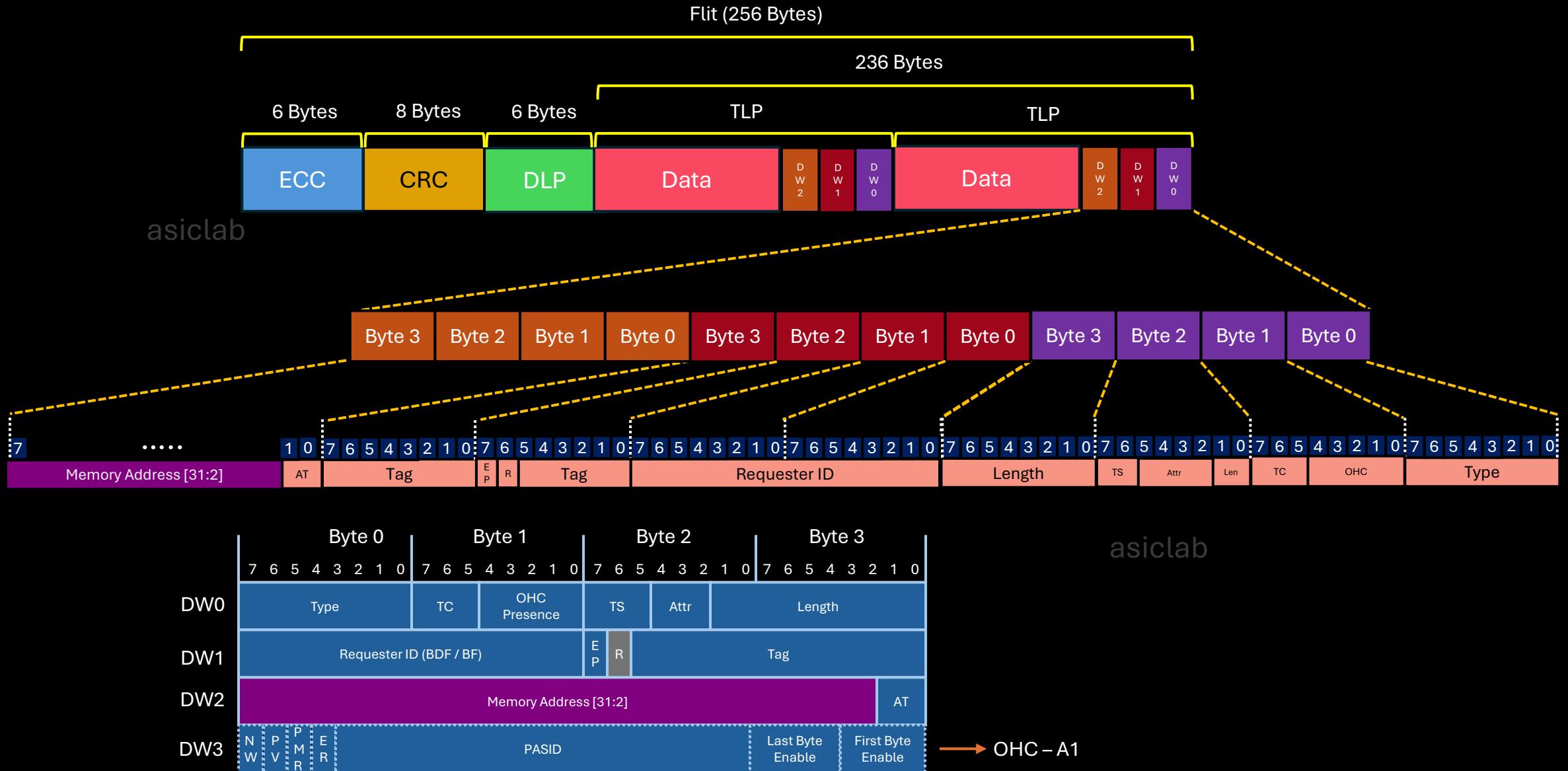
Flit format



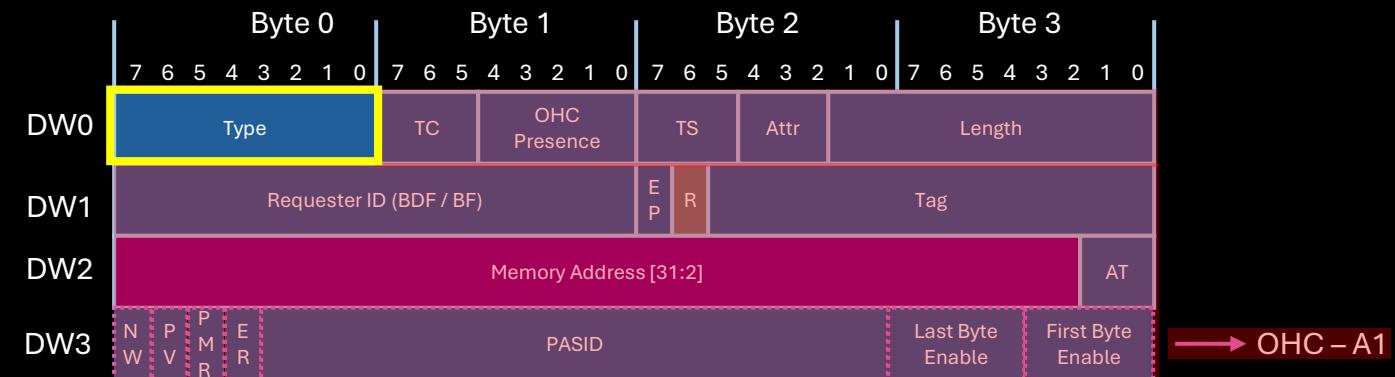
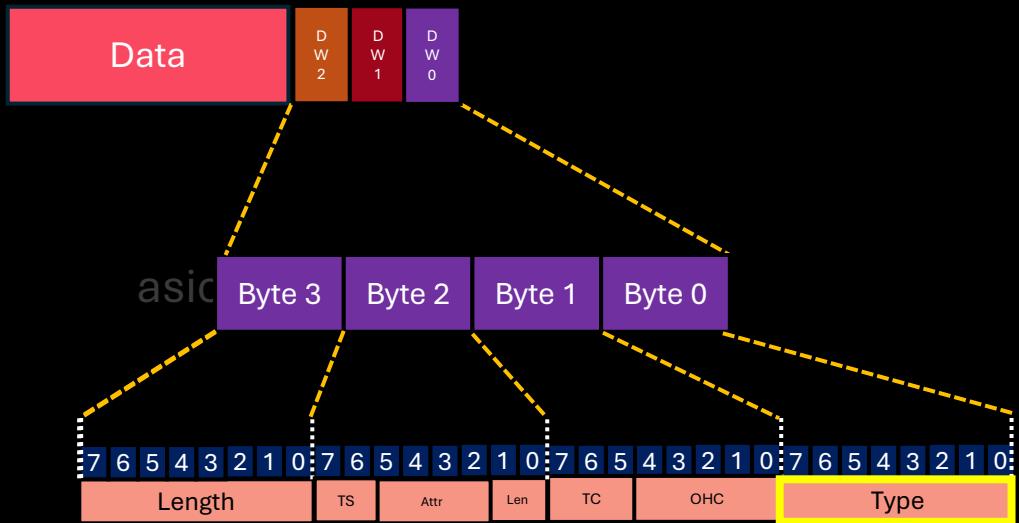
Flit format



Flit format

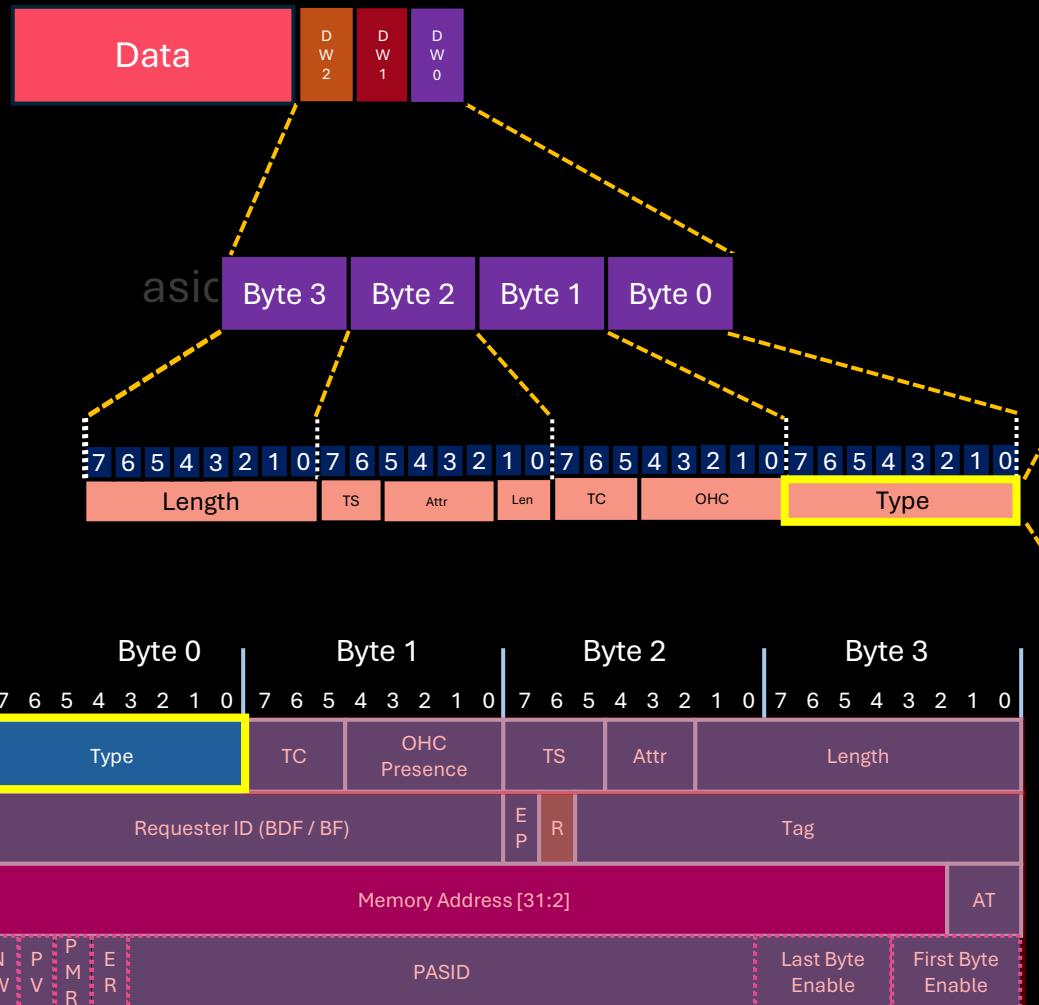


Flit format



asiclab

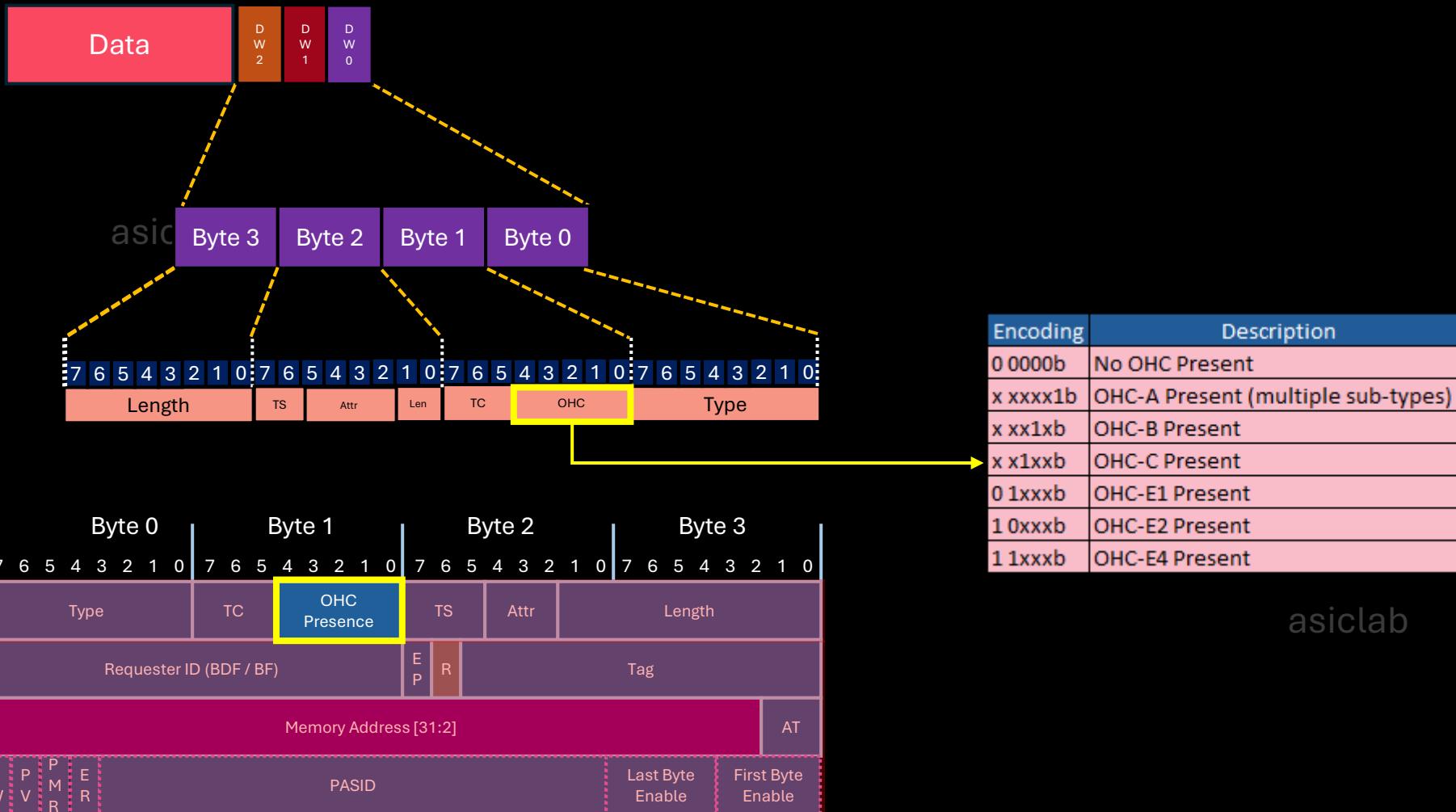
Flit format



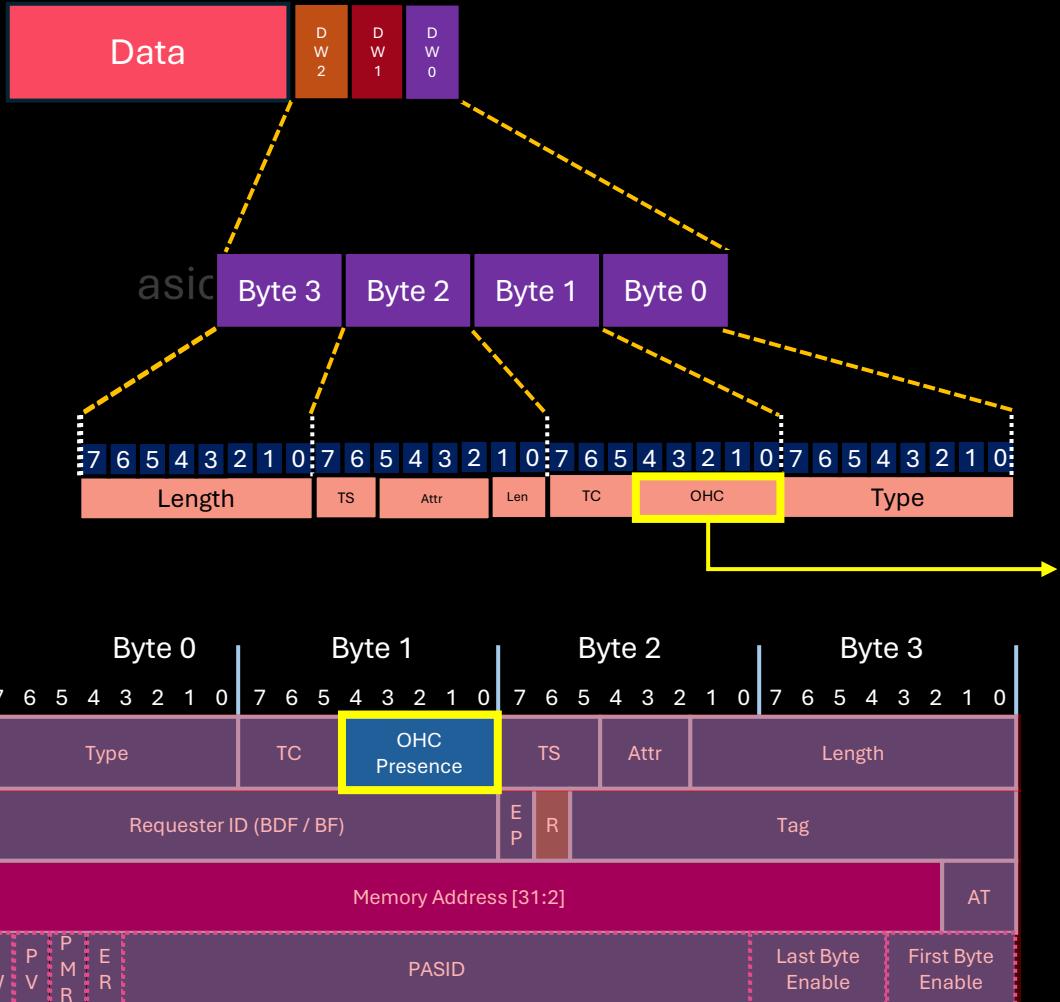
These two encodings have a different meaning in FM vs NFM

Encoding	Description	Hdr Base Size (DW)
0000_0000b	NOP	1
0000_0011b	Memory Read (32-bit addr)	3
0010_0000b	Memory Read (64-bit addr)	4
0100_0000b	Memory Write (32-bit addr)	3
0110_0000b	Memory Write (64-bit addr)	4
0000_0001b	Memory Read Lock (32-bit addr)	3
0010_0001b	Memory Read Lock (64-bit addr)	4
0000_0010b	IO Read	3
0100_0010b	IO Write	3
0000_0100b	Configuration Read Type 0	3
0000_0101b	Configuration Read Type 1	3
0100_0100b	Configuration Write Type 0	3
0100_0101b	Configuration Write Type 1	3
0000_1010b	Completion (no data)	3
0000_1011b	Completion Lock (no data)	3
0100_1010b	Completion (with data)	3
0100_1011b	Completion Lock (with data)	3
0011_0rrrb	Message (no data)	4
0111_0rrrb	Message (with data)	4
0101_1011b	Deferable Memory Write (32-bit addr)	3
0111_1011b	Deferable Memory Write (64-bit addr)	4
0100_1100b	AtomicOp: Fetch and Add (32-bit addr)	3
0110_1100b	AtomicOp: Fetch and Add (64-bit addr)	4
0100_1101b	AtomicOp: Unconditional Swap (32-bit addr)	3
0110_1101b	AtomicOP: Unconditional Swap (64-bit addr)	4
0100_1110b	AtomicOP: Compare and Swap (32-bit addr)	3
0110_1110b	AtomicOP: Compare and Swap (64-bit addr)	4
1000_1101b	FM Local TLP Prefix	1
1000_1110b	Vendor Defined Local 0	1
1000_1111b	Vendor Defined Local 1	1

Flit format

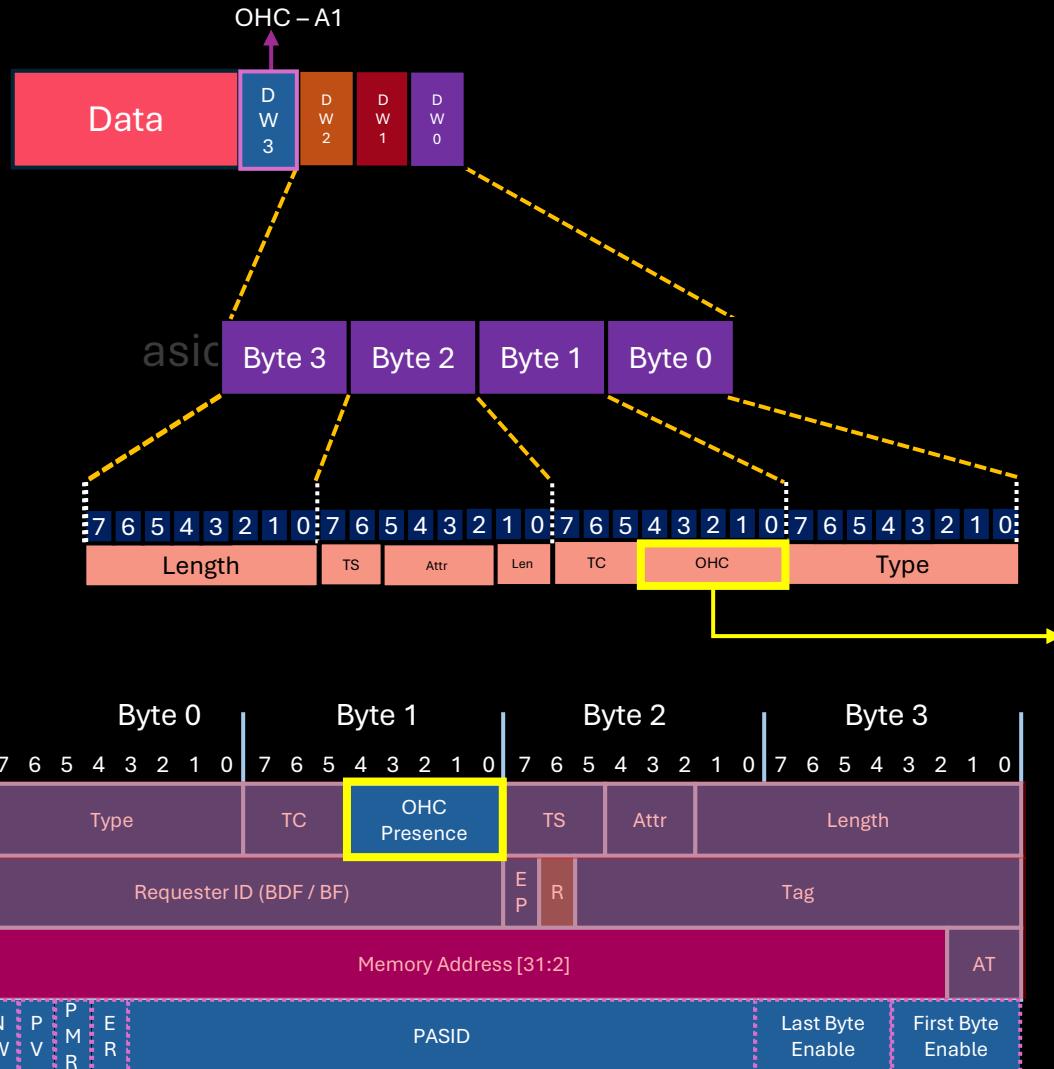


Flit format

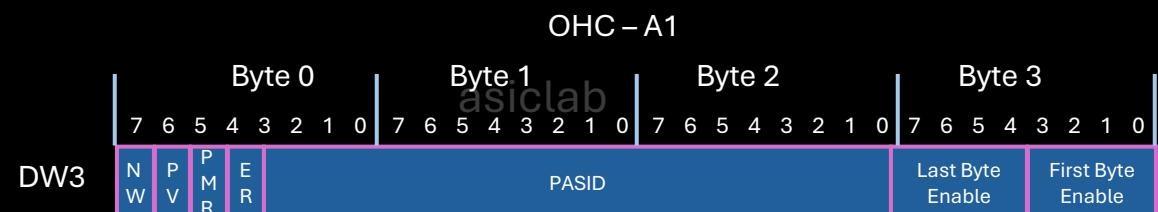


Type	Required For	Byte Enables	PV, PASID	PMR,ER	NW	DSV, Dest Segment	Cmpltr Segment	Cmpl Status	Lower Addr
OHC-A1	Memory Requests with explicitly Byte Enables and/or PASID Translation Requests, Addr Routed Messages and Route to Root Complex Messages with PASID	Yes	Yes	Yes	Yes				
OHC-A2	IO Requests	Yes							
OHC-A3	Configuration Requests	Yes				Yes			
OHC-A4	ID-Routed Messages that require Destination Segment and/or PASID		Yes			Yes			
OHC-A5	Unsuccessful Completions Completions with Lower Address [1:0] ≠ 00b Completions that require a Destination Segment					Yes	Yes	Yes	Yes

Flit format

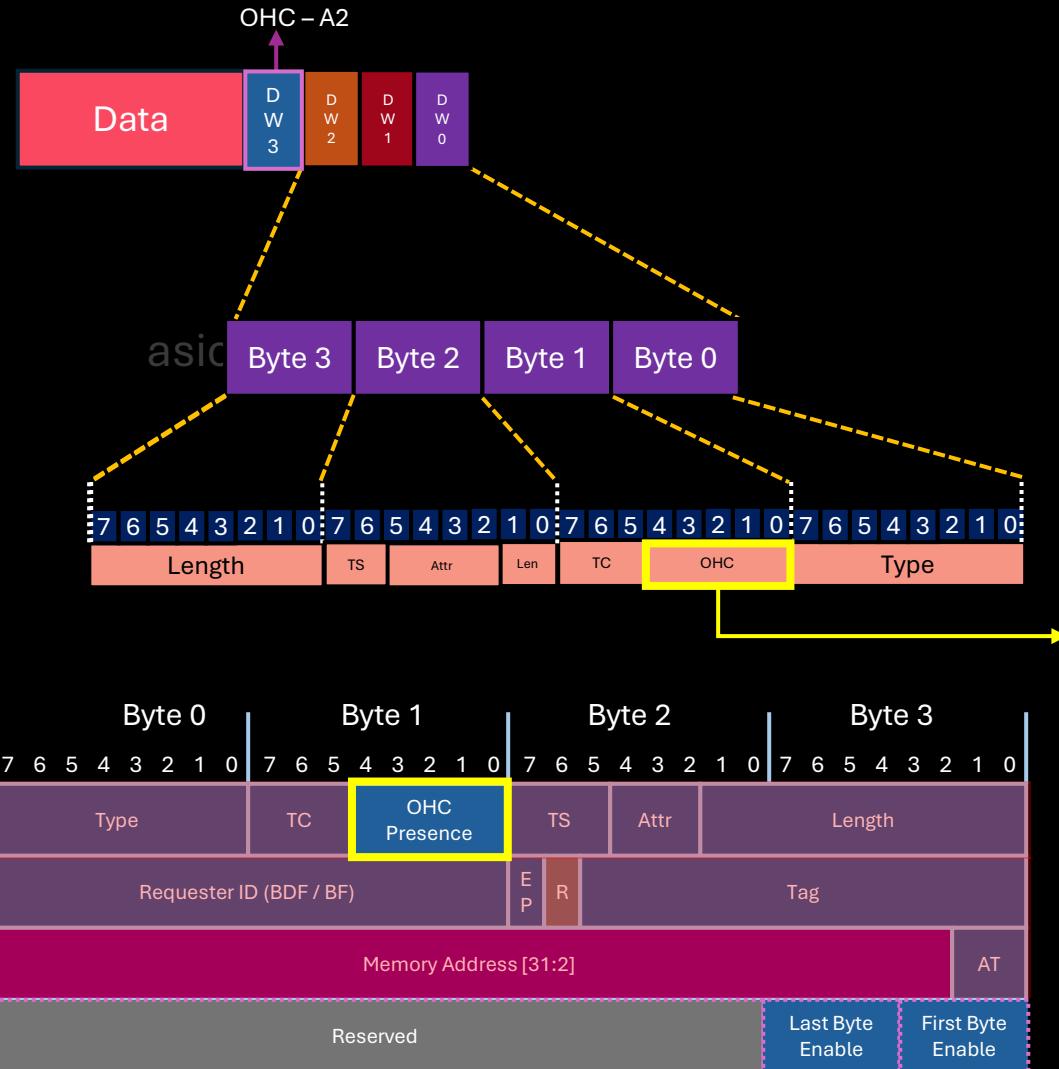


Type	Required For	Byte Enables	PV, PASID	PMR,ER	NW	DSV, Dest Segment	Cmpltr Segment	Cmpl Status	Lower Addr
OHC-A1	Memory Requests with explicit Byte Enables and/or PASID Translation Requests, Addr Routed Messages and Route to Root Complex Messages with PASID	Yes	Yes	Yes	Yes				
OHC-A2	IO Requests	Yes							
OHC-A3	Configuration Requests	Yes				Yes			
OHC-A4	ID-Routed Messages that require Destination Segment and/or PASID		Yes			Yes			
OHC-A5	Unsuccessful Completions Completions with Lower Address [1:0] ≠ 00b Completions that require a Destination Segment					Yes	Yes	Yes	Yes



ER – Execute Requested
 PMR – Privileged Mode Requested
 PV – PASID Valid
 NW – No Write

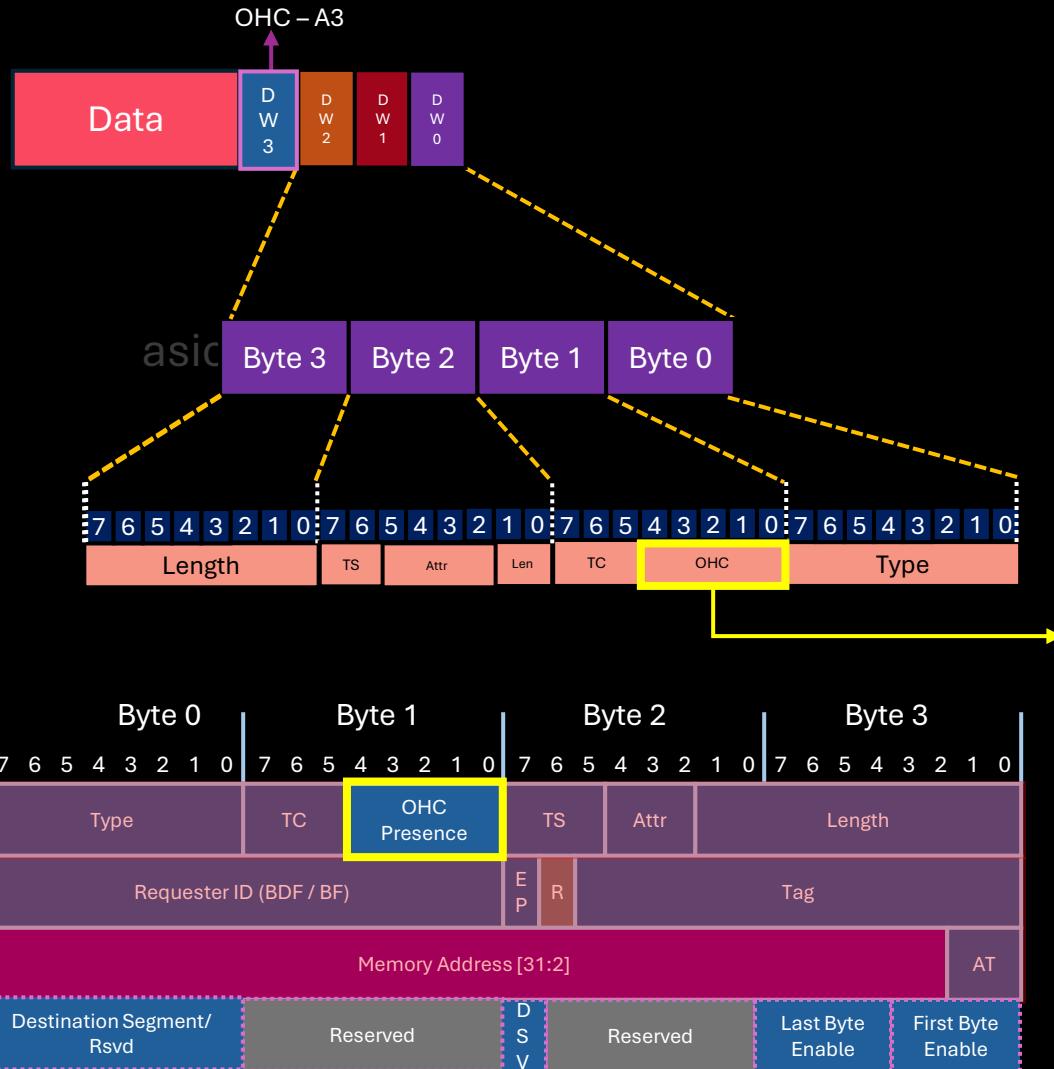
Flit format



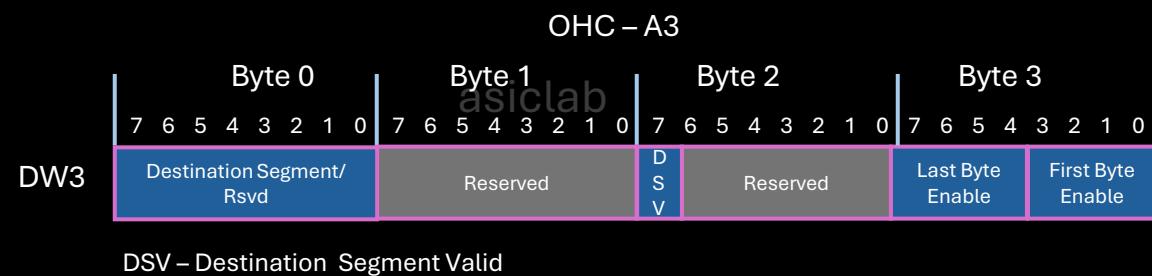
Type	Required For	Byte Enables	PV, PASID	PMR,ER	NW	DSV, Dest Segment	Cmpltr Segment	Cmpl Status	Lower Addr
OHC-A1	Memory Requests with explicitly Byte Enables and/or PASID Translation Requests, Addr Routed Messages and Route to Root Complex Messages with PASID	Yes	Yes	Yes	Yes				
OHC-A2	IO Requests	Yes							
OHC-A3	Configuration Requests	Yes				Yes			
OHC-A4	ID-Routed Messages that require Destination Segment and/or PASID		Yes			Yes			
OHC-A5	Unsuccessful Completions Completions with Lower Address [1:0] ≠ 00b Completions that require a Destination Segment					Yes	Yes	Yes	Yes



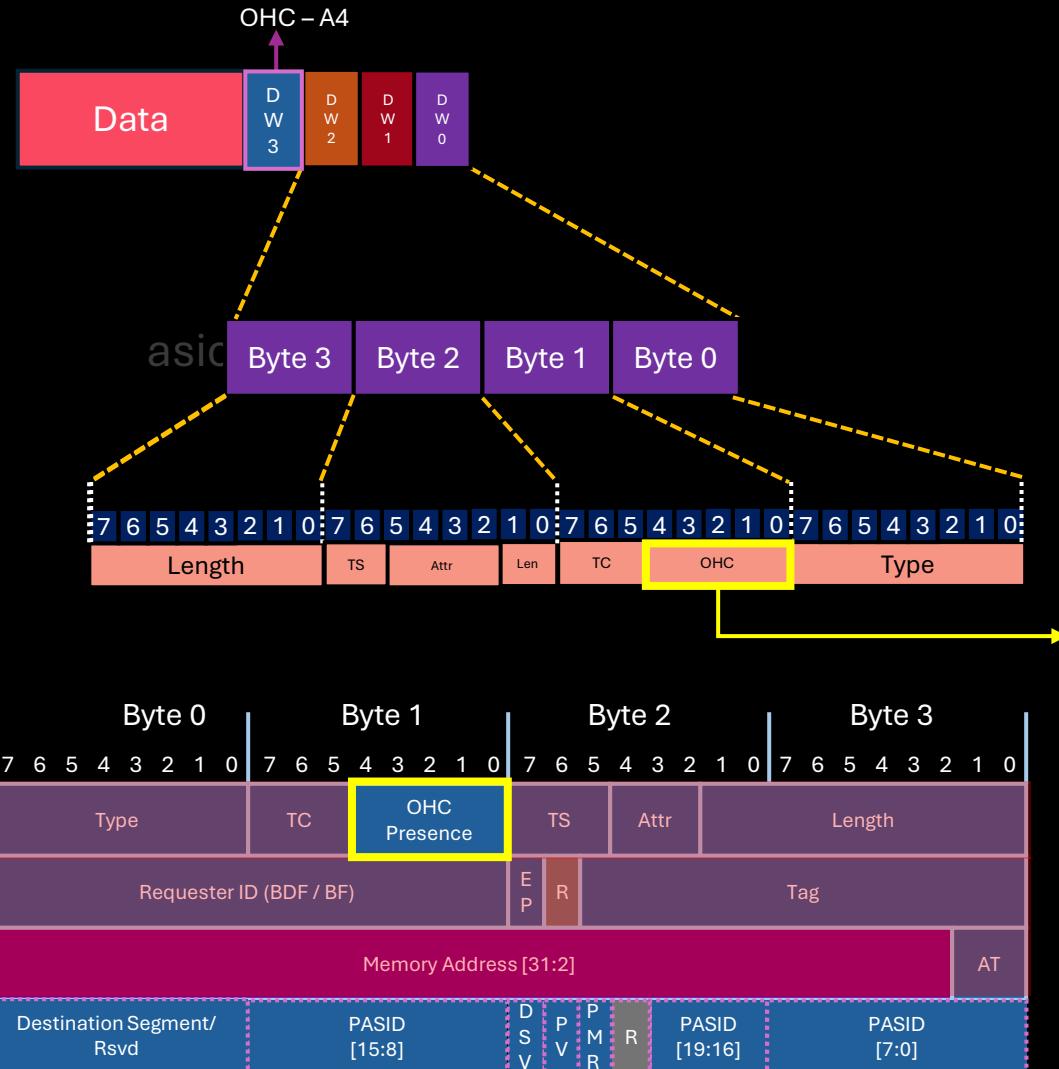
Flit format



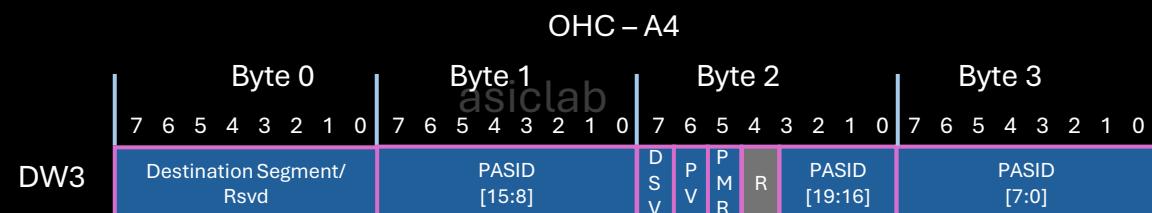
Type	Required For	Byte Enables	PV, PASID	PMR,ER	NW	DSV, Dest Segment	Cmpltr Segment	Cmpl Status	Lower Addr
OHC-A1	Memory Requests with explicitly Byte Enables and/or PASID Translation Requests, Addr Routed Messages and Route to Root Complex Messages with PASID	Yes	Yes	Yes	Yes				
OHC-A2	IO Requests	Yes							
OHC-A3	Configuration Requests	Yes				Yes			
OHC-A4	ID-Routed Messages that require Destination Segment and/or PASID		Yes			Yes			
OHC-A5	Unsuccessful Completions Completions with Lower Address [1:0] ≠ 00b Completions that require a Destination Segment					Yes	Yes	Yes	Yes



Flit format



Type	Required For	Byte Enables	PV, PASID	PMR,ER	NW	DSV, Dest Segment	Cmpltr Segment	Cmpl Status	Lower Addr
OHC-A1	Memory Requests with explicit Byte Enables and/or PASID Translation Requests, Addr Routed Messages and Route to Root Complex Messages with PASID	Yes	Yes	Yes	Yes				
OHC-A2	IO Requests	Yes							
OHC-A3	Configuration Requests	Yes				Yes			
OHC-A4	ID-Routed Messages that require Destination Segment and/or PASID		Yes			Yes			
OHC-A5	Unsuccessful Completions Completions with Lower Address [1:0] ≠ 00b Completions that require a Destination Segment					Yes	Yes	Yes	Yes

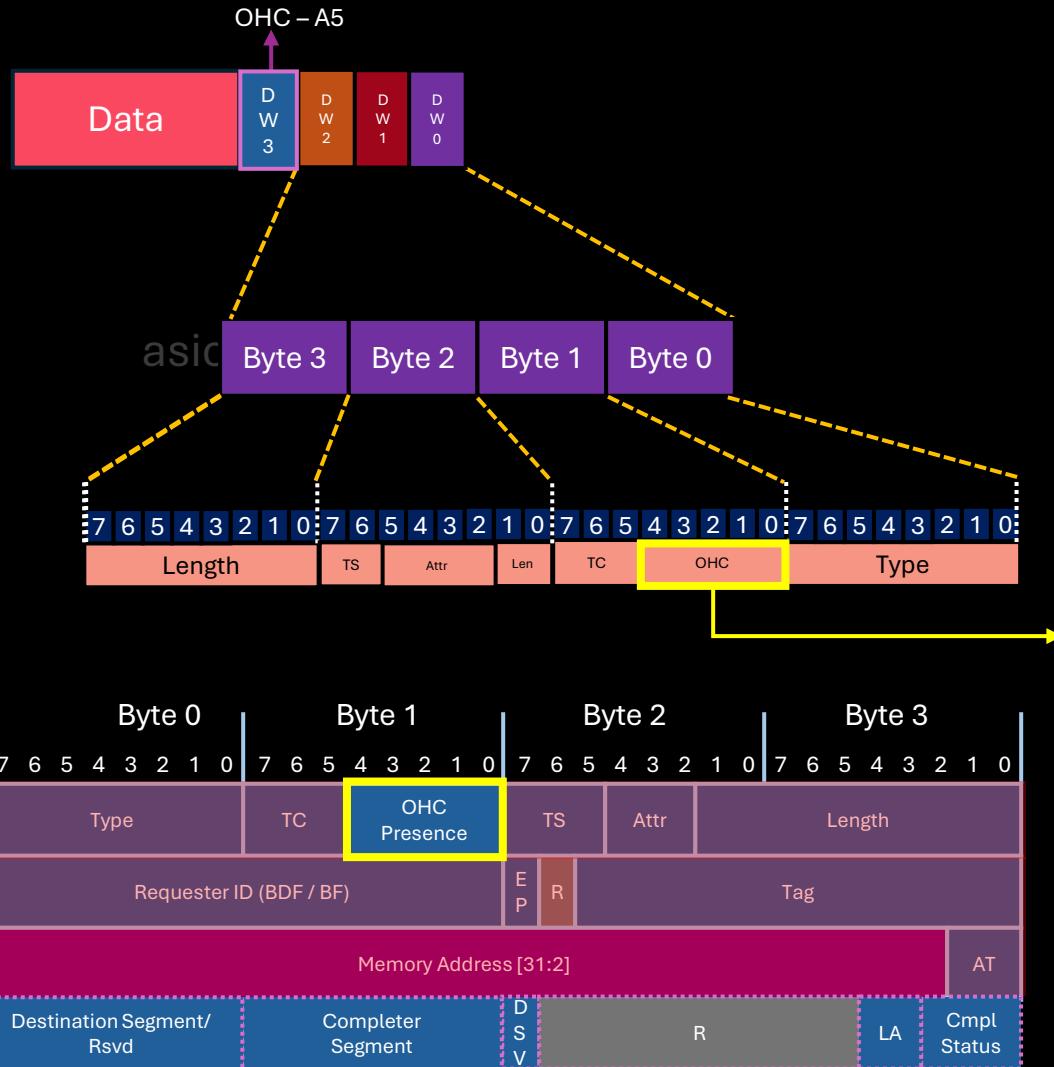


DSV – Destination Segment Valid

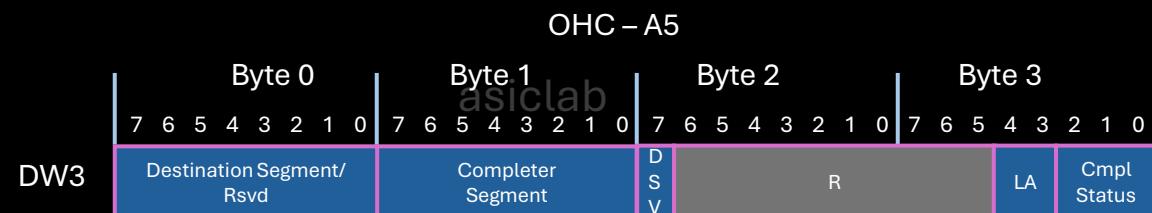
PV – PASID Valid

PMR - Privileged Mode Requested

Flit format

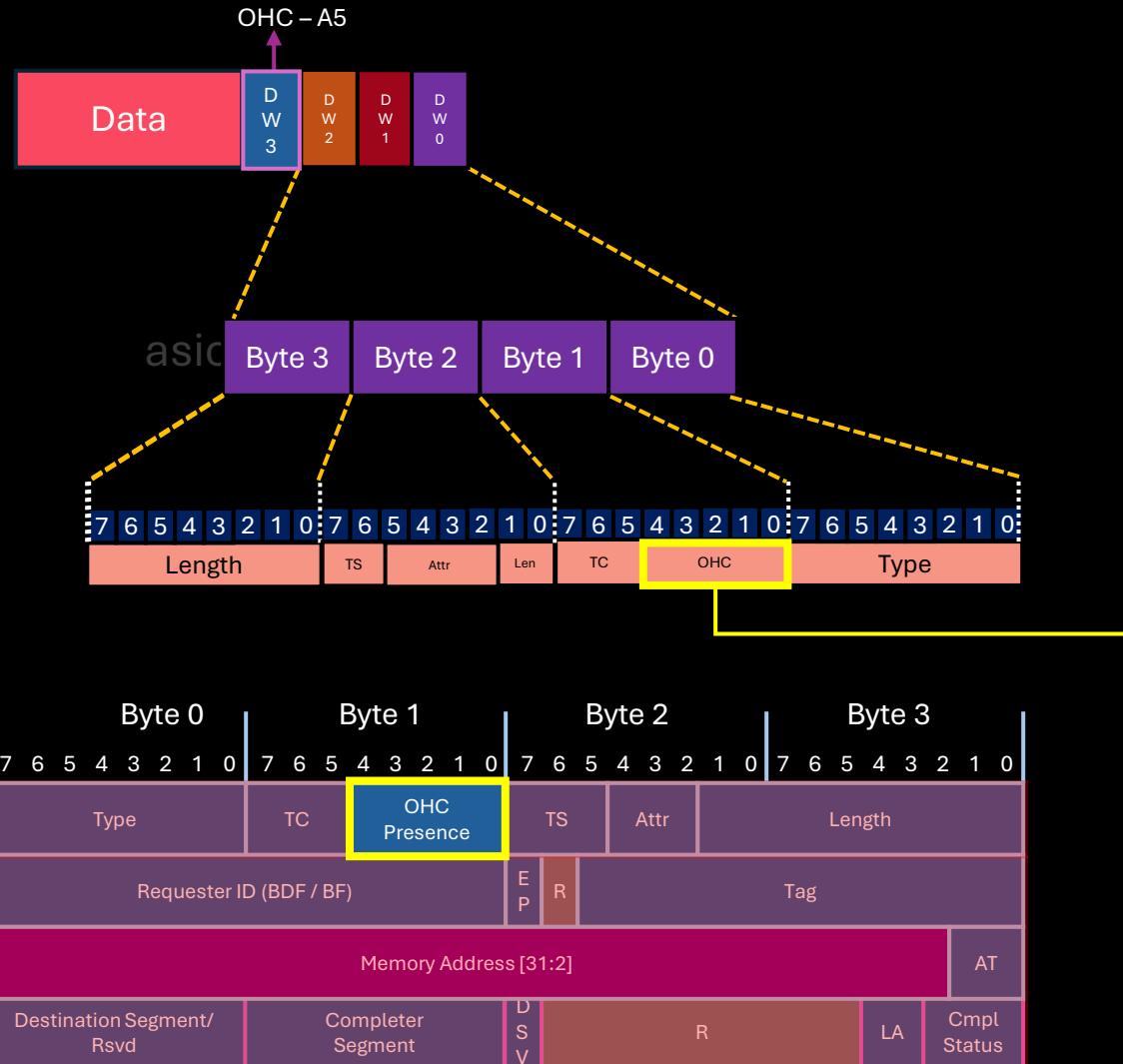


Type	Required For	Byte Enables	PV, PASID	PMR,ER	NW	DSV, Dest Segment	Cmpltr Segment	Cmpl Status	Lower Addr
OHC-A1	Memory Requests with explicit Byte Enables and/or PASID Translation Requests, Addr Routed Messages and Route to Root Complex Messages with PASID	Yes	Yes	Yes	Yes				
OHC-A2	IO Requests	Yes							
OHC-A3	Configuration Requests	Yes				Yes			
OHC-A4	ID-Routed Messages that require Destination Segment and/or PASID		Yes			Yes			
OHC-A5	Unsuccessful Completions Completions with Lower Address [1:0] ≠ 00b Completions that require a Destination Segment					Yes	Yes	Yes	Yes



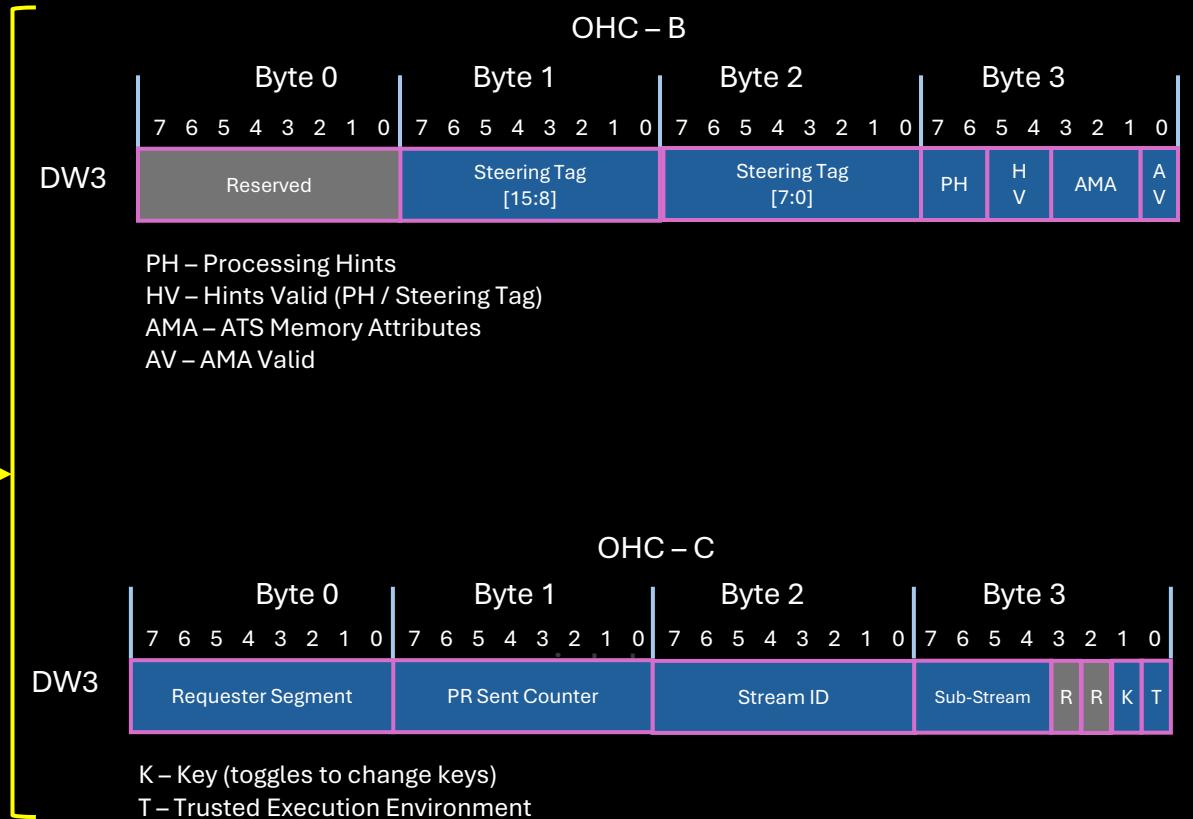
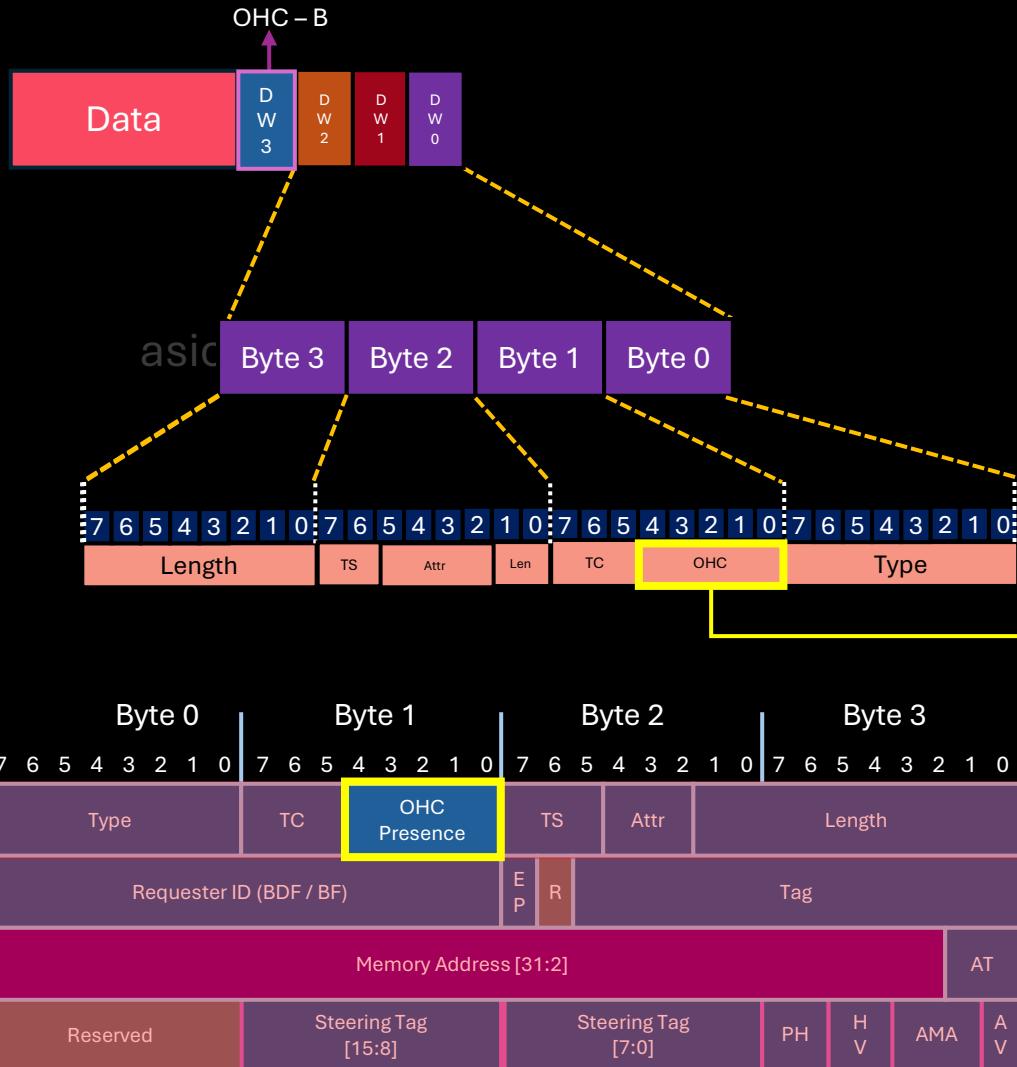
DSV – Destination Segment Valid
LA – Lower Address

Flit format

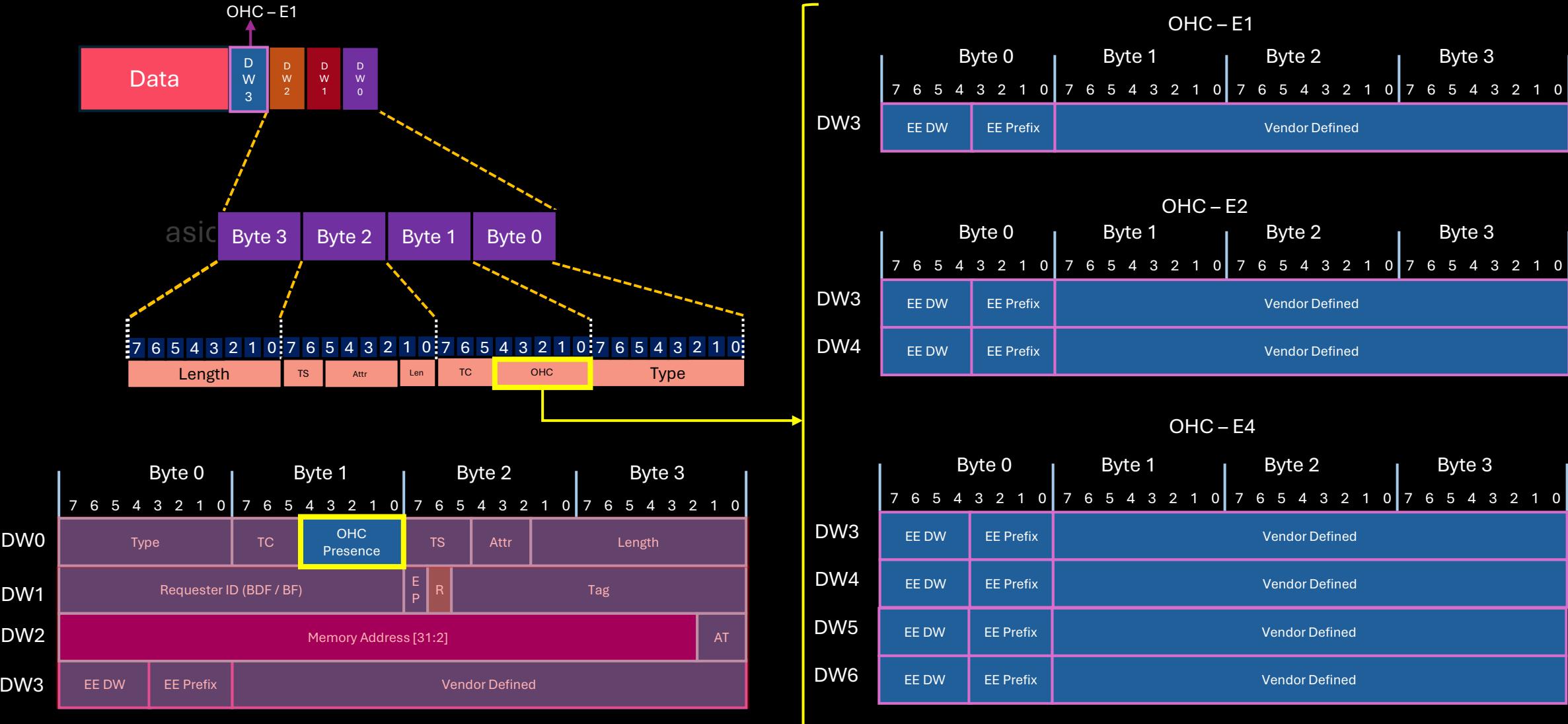


Encoding	Description	Hdr Base Size (DW)	OHC-A Type
0000_0000b	NOP	1	OHC-A1 (If needed)
0000_0011b	Memory Read (32-bit addr)	3	
0010_0000b	Memory Read (64-bit addr)	4	
0100_0000b	Memory Write (32-bit addr)	3	
0110_0000b	Memory Write (64-bit addr)	4	
0000_0001b	Memory Read Lock (32-bit addr)	3	
0010_0001b	Memory Read Lock (64-bit addr)	4	
0000_0010b	IO Read	3	
0100_0010b	IO Write	3	
0000_0100b	Configuration Read Type 0	3	
0000_0101b	Configuration Read Type 1	3	OHC-A3 (If needed)
0100_0100b	Configuration Write Type 0	3	
0100_0101b	Configuration Write Type 1	3	
0000_1010b	Completion (no data)	3	
0000_1011b	Completion Lock (no data)	3	OHC-A5 (If needed)
0100_1010b	Completion (with data)	3	
0100_1011b	Completion Lock (with data)	3	
0011_0rrrb	Message (no data)	4	OHC-A1 (If needed for addr routed messages)
0111_0rrrb	Message (with data)	4	OHC-A4 (If needed for ID routed messages)
0101_1011b	Deferable Memory Write (32-bit addr)	3	OHC-A1 (If needed)
0111_1011b	Deferable Memory Write (64-bit addr)	4	
0100_1100b	AtomicOp: Fetch and Add (32-bit addr)	3	
0110_1100b	AtomicOp: Fetch and Add (64-bit addr)	4	
0100_1101b	AtomicOp: Unconditional Swap (32-bit addr)	3	
0110_1101b	AtomicOP: Unconditional Swap (64-bit addr)	4	
0100_1110b	AtomicOP: Compare and Swap (32-bit addr)	3	
0110_1110b	AtomicOP: Compare and Swap (64-bit addr)	4	

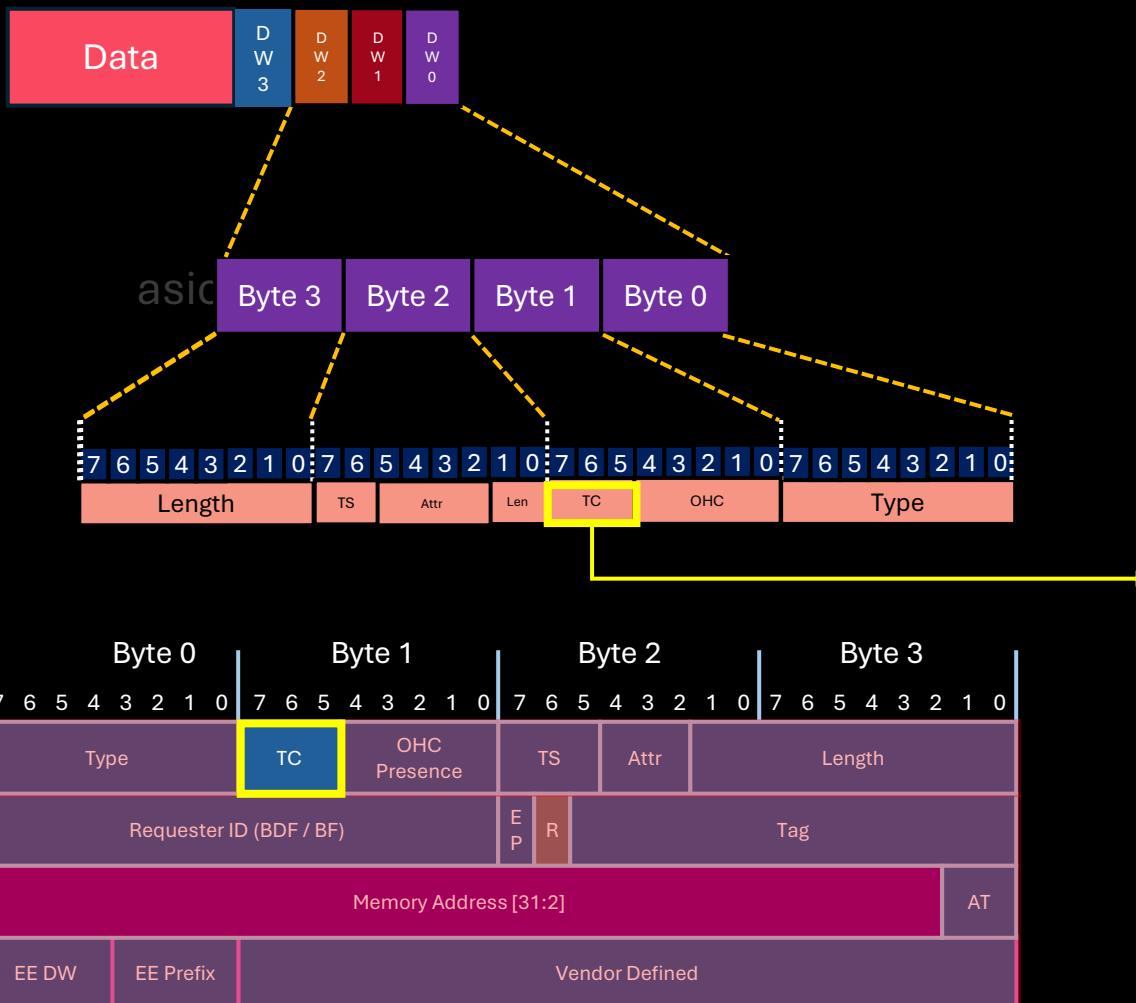
Flit format



Flit format

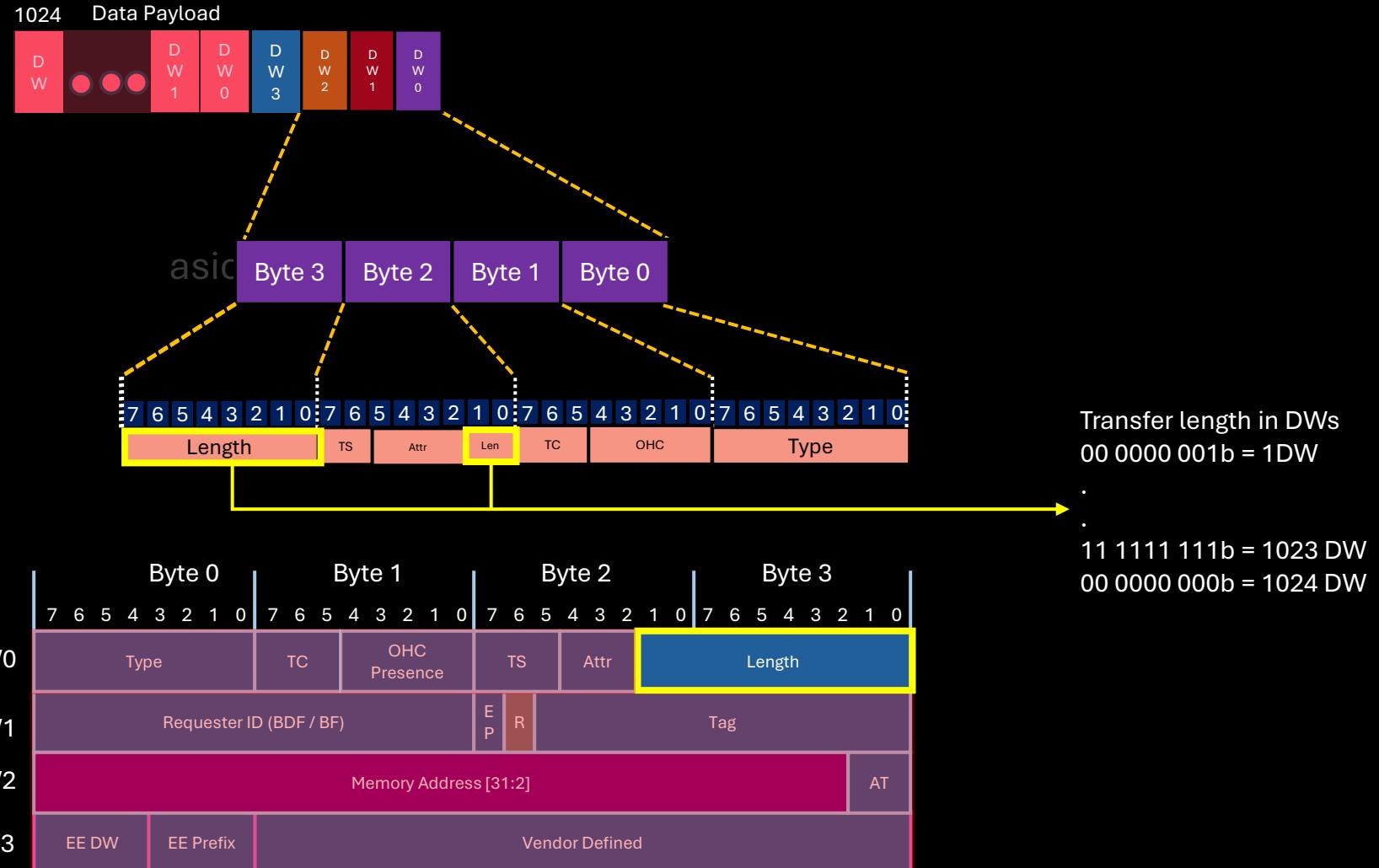


Flit format

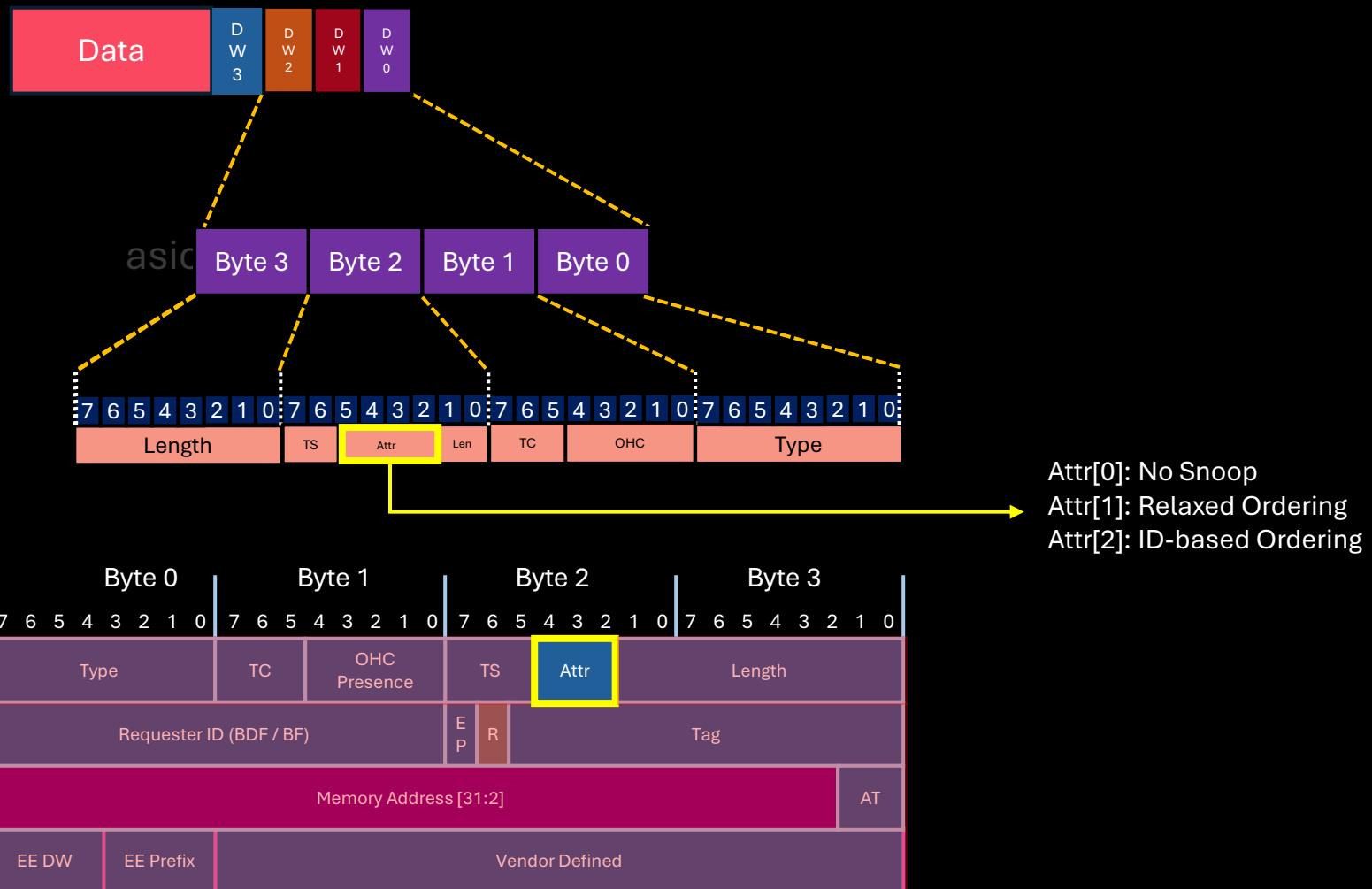


Encoding	Description
000b	TC0: Best effort service class (General Purpose I/O) Default TC
001b	Traffic Class 1
010b	Traffic Class 2
011b	Traffic Class 3
100b	Traffic Class 4
101b	Traffic Class 5
110b	Traffic Class 6
111b	TC7: Differentiated traffic class (Based on weighted-Round-Robin(WRR) and/or priority)

Flit format

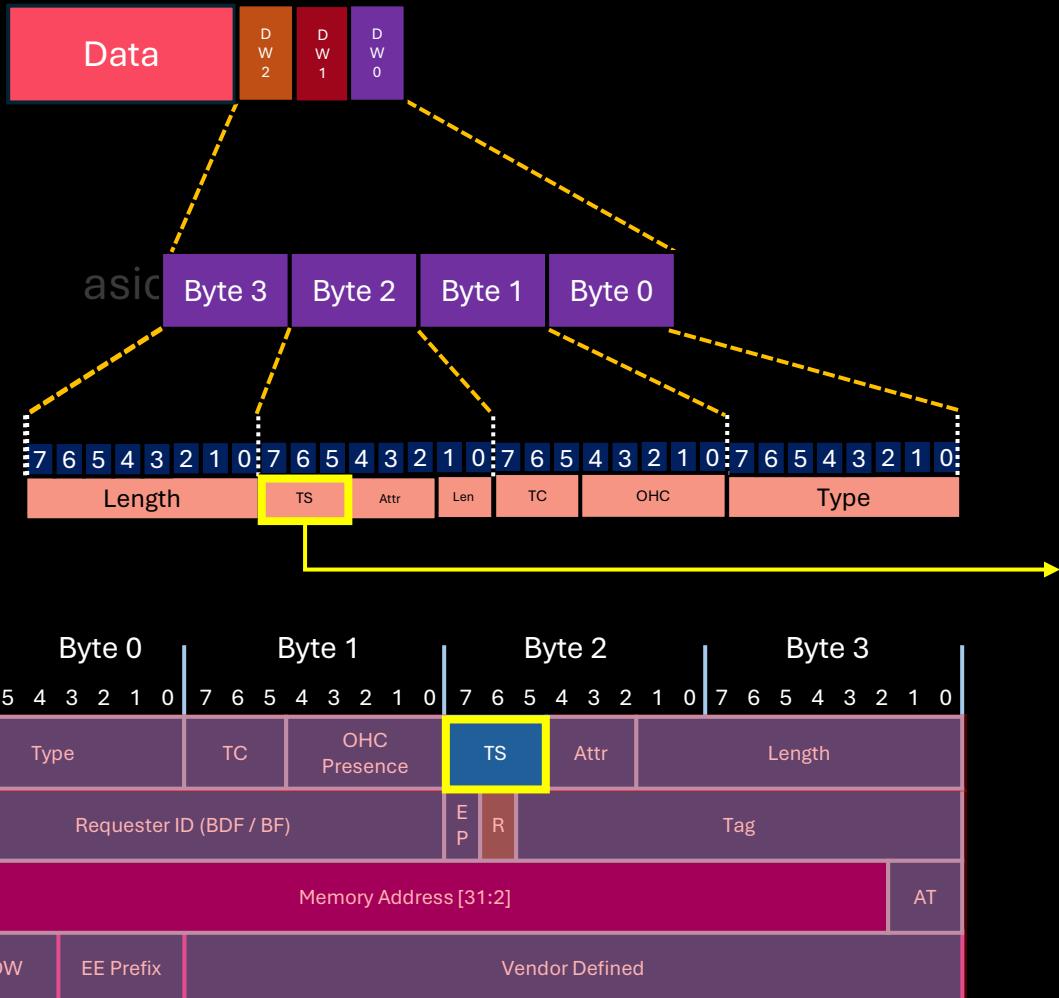


Flit format



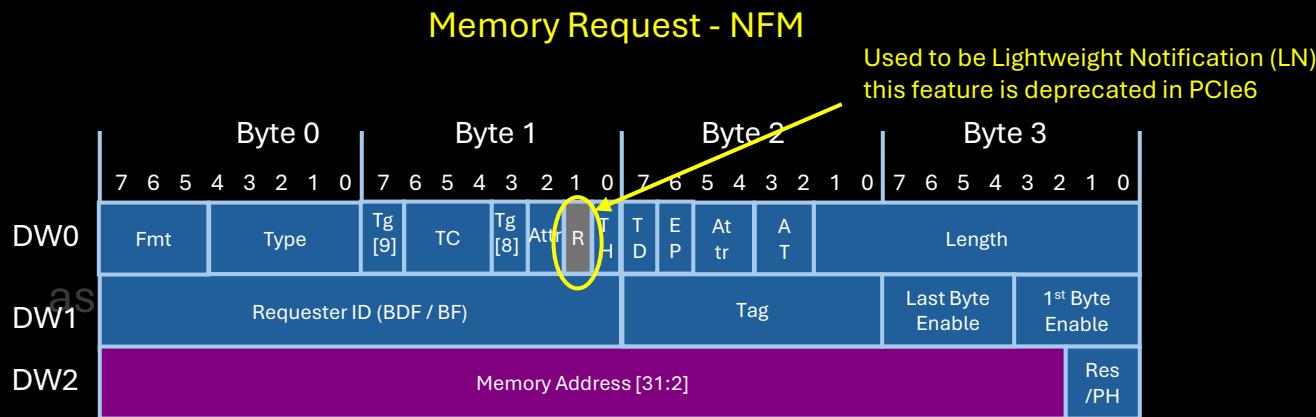
asiclab

Flit format

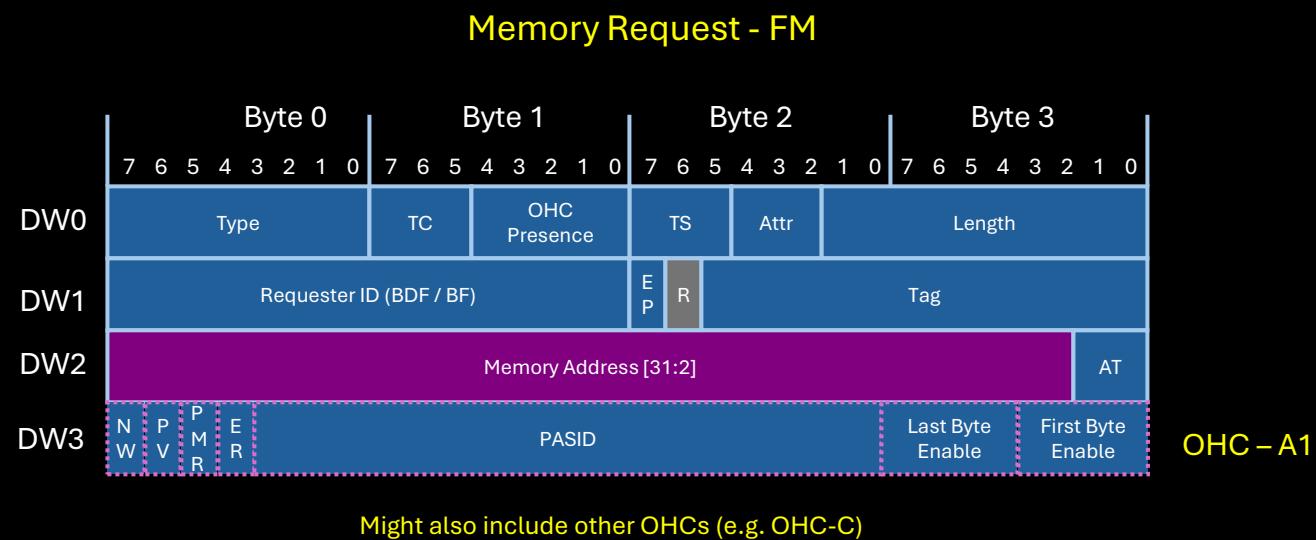


Encoding	Description
000b	No trailer present
001b	1DW trailer, with ECRC
010b	1DW trailer, content currently reserved
011b	1DW trailer, content currently reserved
100b	1DW trailer, content currently reserved
101b	1DW trailer, content of trailer depend on: IF this TLP has OHC-C and indicates IDE TLP, trailer holds IDE MAC. Else, trailer content currently reserved.
110b	4DW trailer, content of trailer depend on: If this TLP has OHC-C and indicates IDE TLP, trailer holds IDE MAC and PCRC. Else, trailer content currently reserved.
111b	5DW trailer, content currently reserved

Flit format



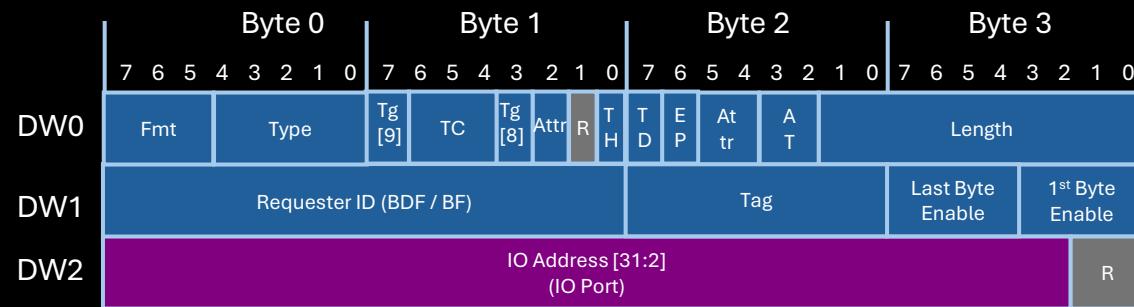
- AT moved to new location
- Res/PH processing Hint is moved to OHC B type
- EP => moved to second DW
- TD => TLP Digest now merged with TLP Trailer
- TH => TLP Processing Hint moved to OHC B type
- Light weight Notify is deprecated
- 10 Bit tag is Gen4 and later -> Now in Gen6 we have 14 bit tag field. We can enable the different tag size via cfg space
- For using Byte enable we need to use OHC A1
asiclab
- If we receive a Memory read without OHC A1 then we assume all the Byte enables are 1111b. For any Memory access that is not DW aligned then we need to use the OHC A1



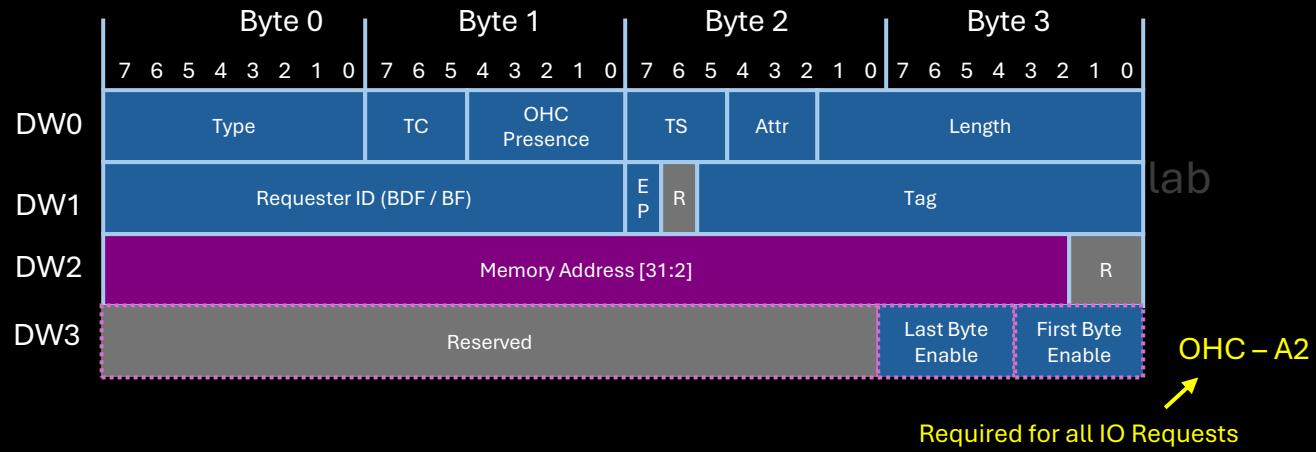
Flit format

asiclab

IO Request - NFM



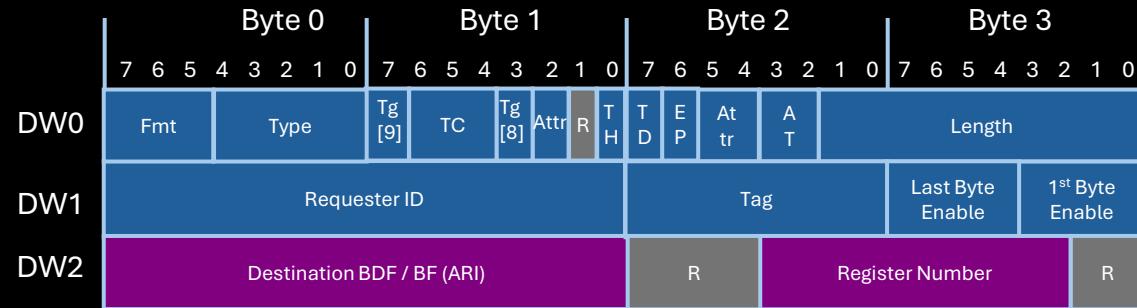
IO Request - FM



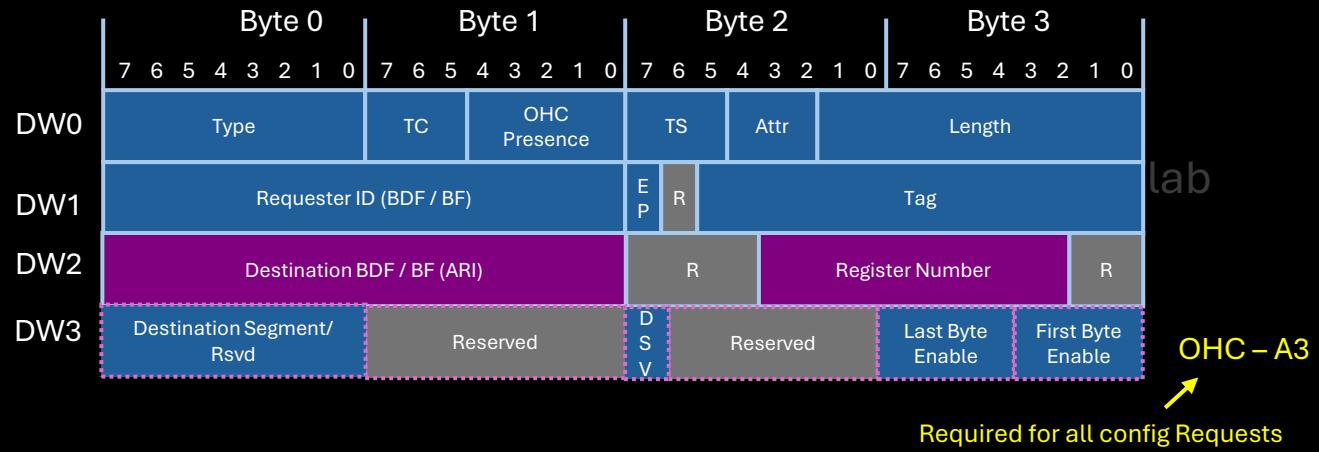
Flit format

asiclab

Configuration Request - NFM



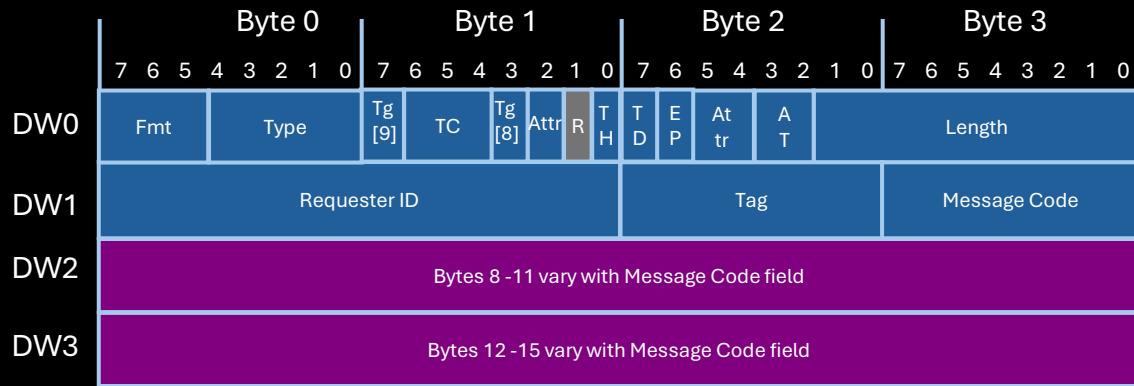
Configuration Request - FM



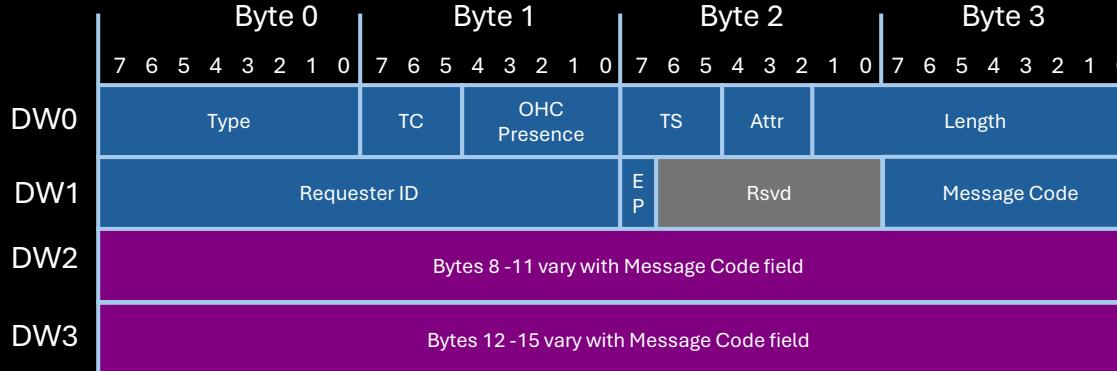
Flit format

asiclab

Message Request - NFM



Message Request - FM

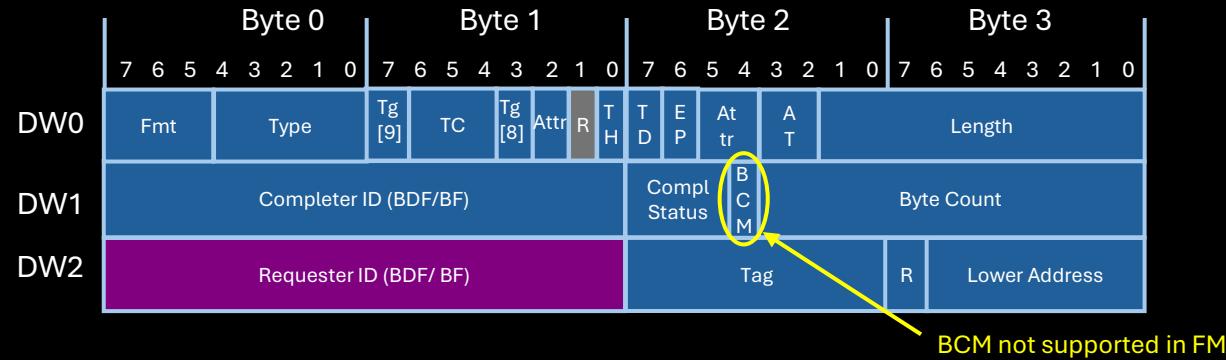


Message is a posted request, hence no need for Tag since it is used for steering the completions

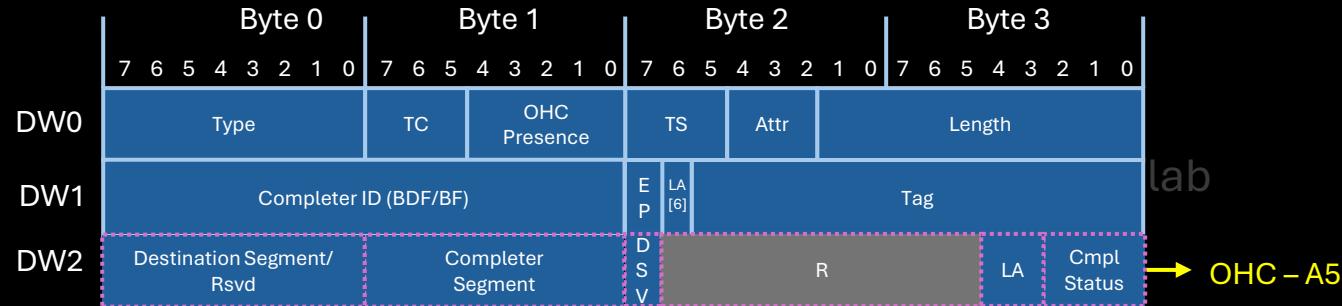
Flit format

asiclab

Completion - NFM



Completion- FM



asiclab

Flit Packing/Unpacking

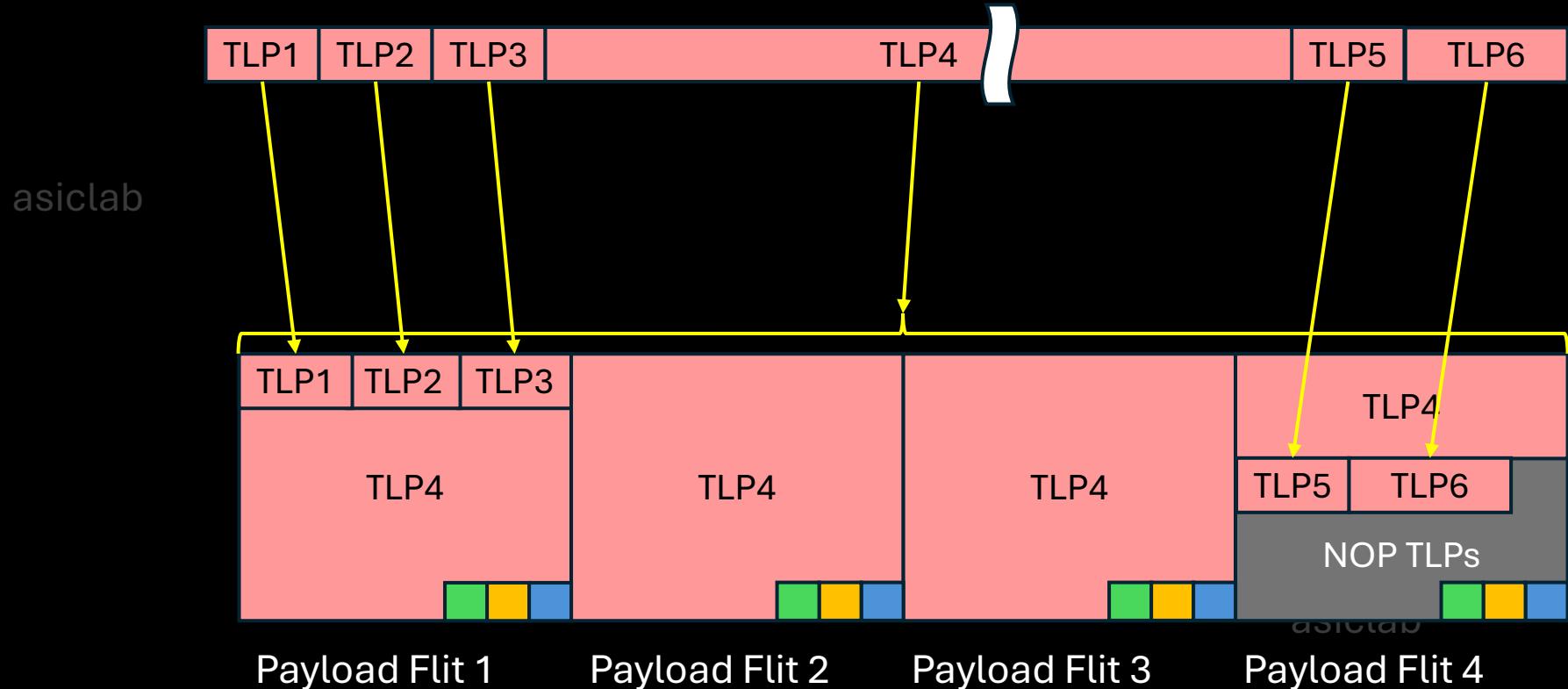
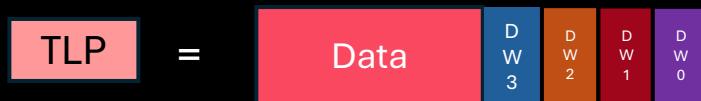
asiclab

Types of Flits

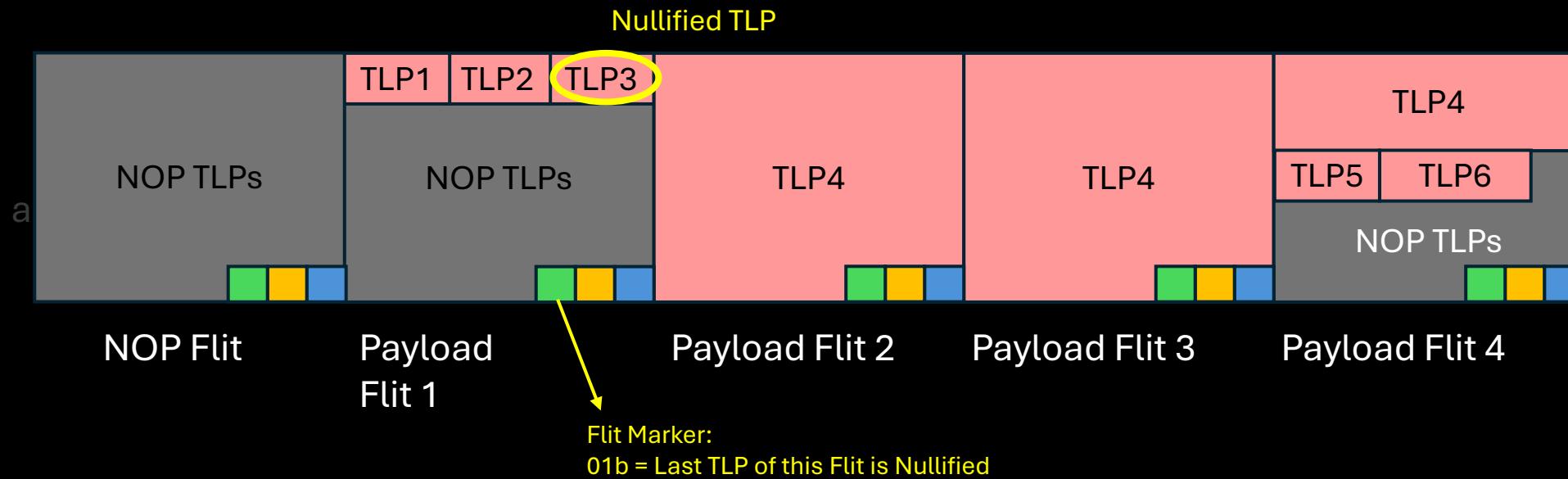
- ❖ **IDLE Flit** -> all the contents are 0 in the TLP portion and the DLP portion will have the sequence number 0. they are only sent just before entering to L0 Entering L0 can be done from config and from recovery, in both these states just before entering L0 there is a state called config.idle and recovery.idle. In Non flit mode we will be sending Logical idles in that state before entering L0. In case of FM this is not available , instead we will send IDLE Flit
- ❖ **NOP Flit** ->in case of this all the content of TLP portion is 0, but DLP can be nonzero, this is sent when there are no TLP to send so these are sent as a filler for the TLP portion
- ❖ **Payload Flit** -> TLP portion is carrying nonzero, and the sequence number is nonzero.

asiclab

Flit Packing



Flit Packing



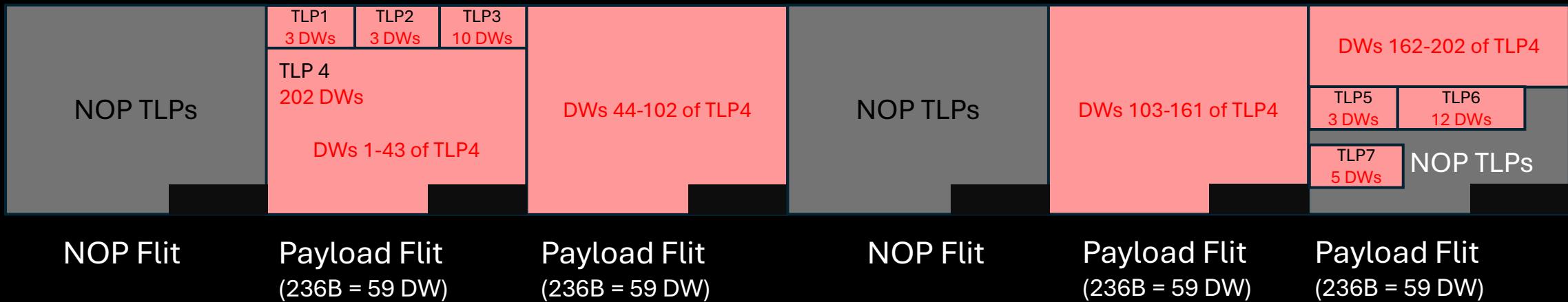
- Each NOP TLP is 4 bytes (1 DW), and it should continue for a 16-byte 4DW boundary.
- It is legal to begin a Flit with NOP TLP.
- There cannot be more than 8 Valid TLP including a partial TLP in each flit half.

Flit Packing

asiclab

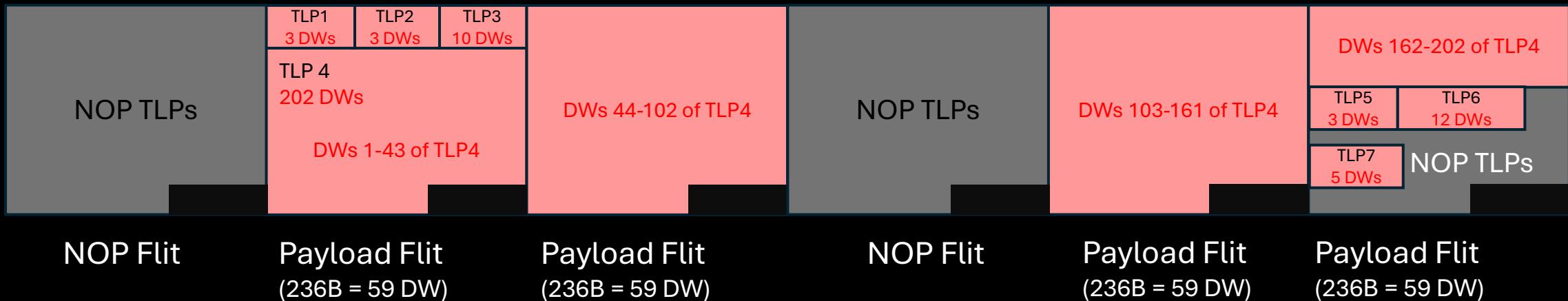
| Lane |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| T0 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T1 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T2 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T3 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T4 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T5 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T6 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T7 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T8 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T9 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T10 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T11 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T12 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T13 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T14 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| T15 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

RX finding TLP boundaries from the Flits that it received



asiclab

RX finding TLP boundaries from the Flits that it received

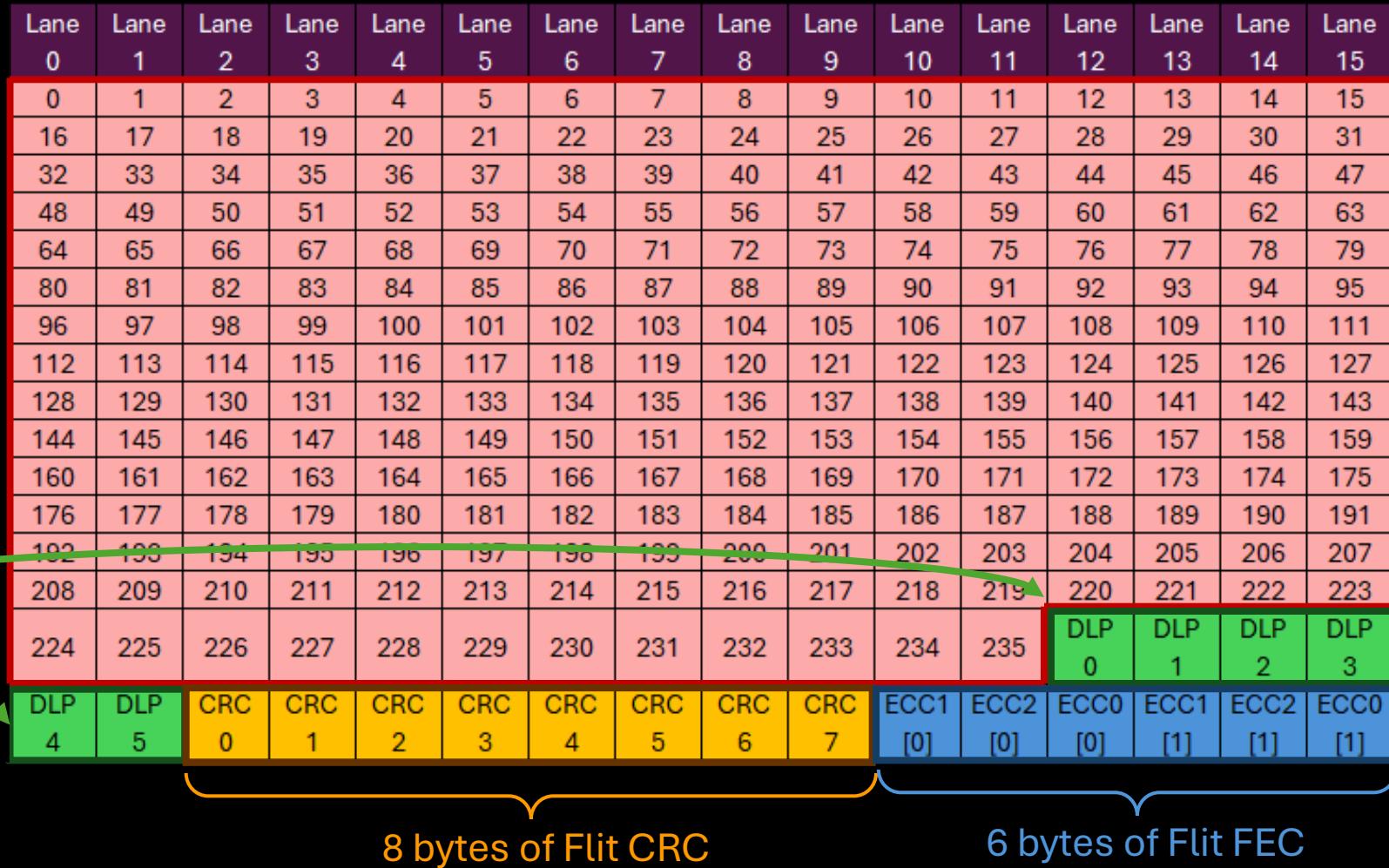


- If in the Flit if the beginning is a NOP TLP then it will continue for 16 Bytes, so RX will skip the 16Bytes and look for a valid flit
- If it is valid flit then Rx should know if there is a prefix, size of that TLP and the various OHC components that might be part of the TLP
- Once when the First TLP is identified the RX will look at the header to check for the size and presence of any Prefix and OHC based on which the Rx will fetch sequential bytes from the Flit until the calculated boundary of the TLP is reached.
- It will do the same for the remaining TLPs

Flit Usage 16 Lanes

asiclab
236 bytes of
TLP traffic

6 bytes of
DLP traffic



asiclab

Flow Control

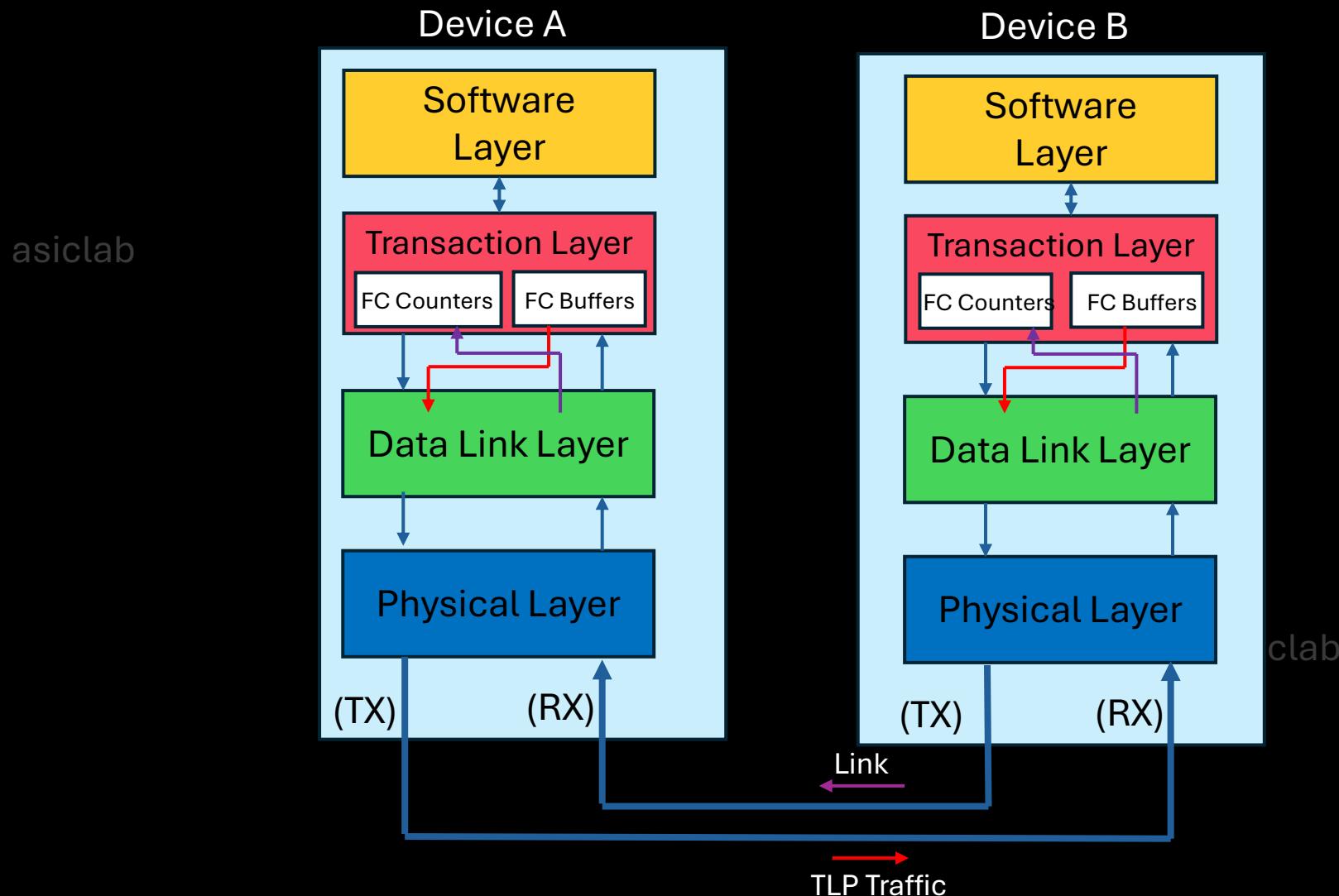
asiclab

Do Not Distribute

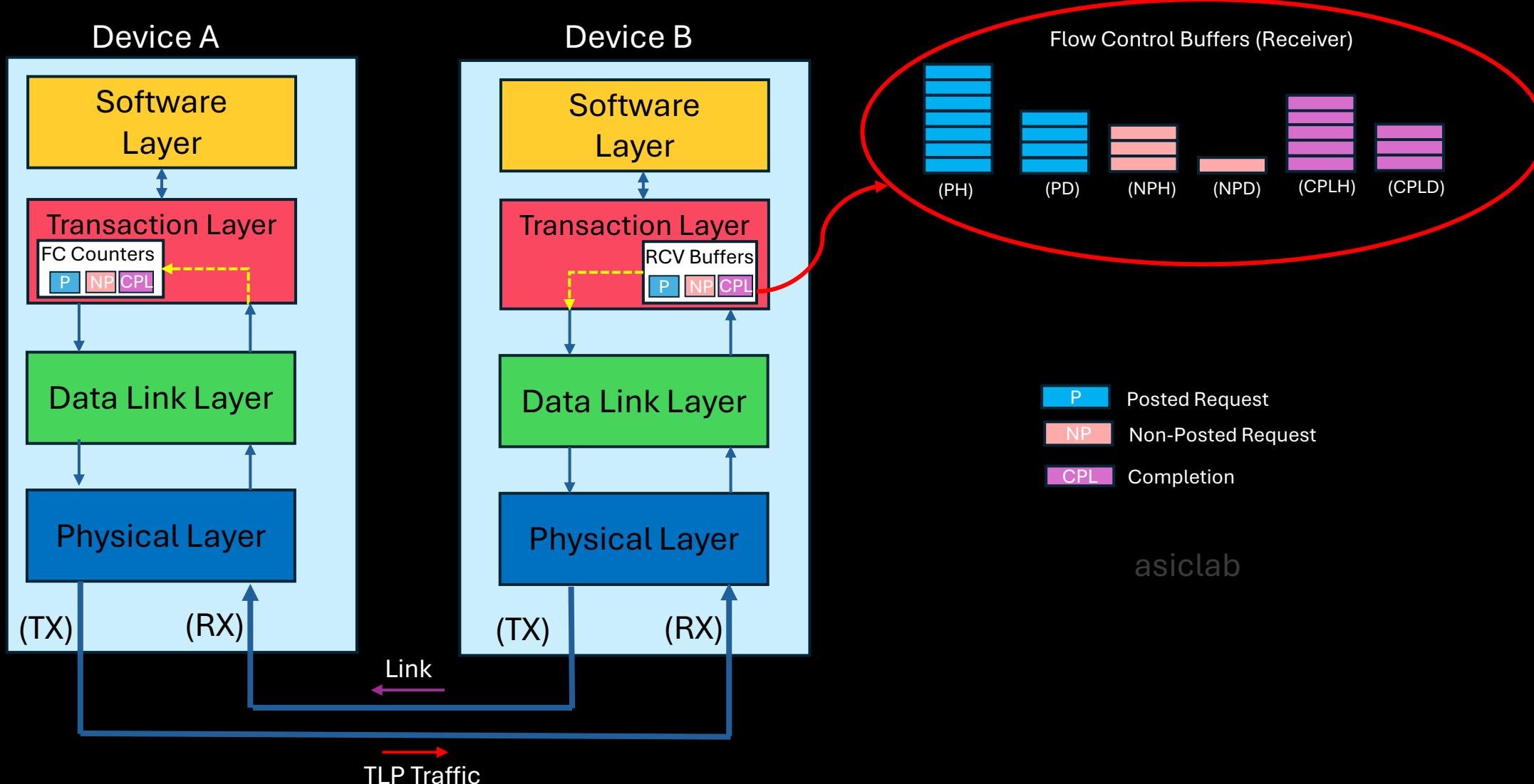
© Copyright protected 2024

asiclab
Learn. Thrive

Location of Flow Control Logic



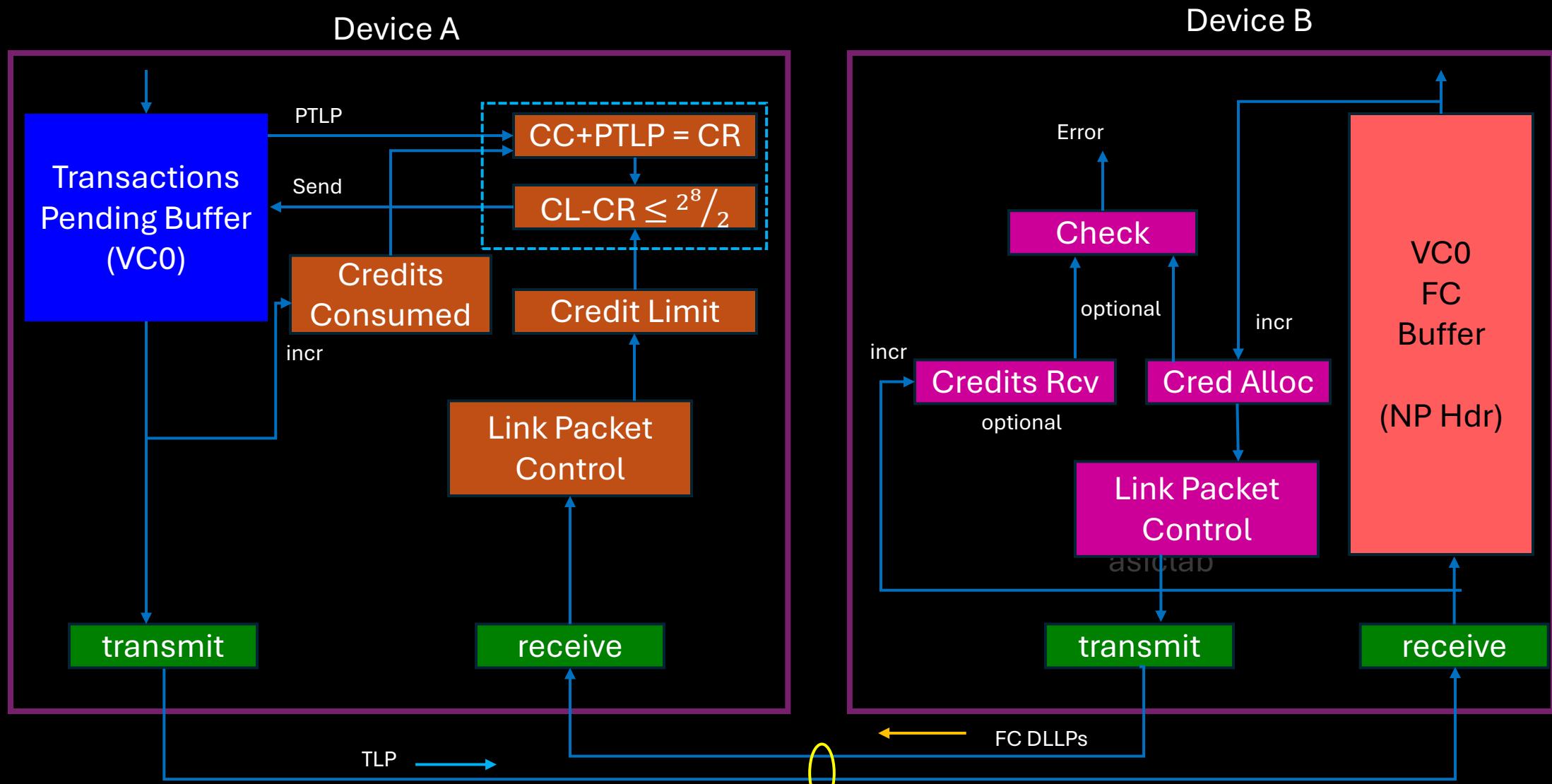
Flow Control Buffer Organization



Credit Units

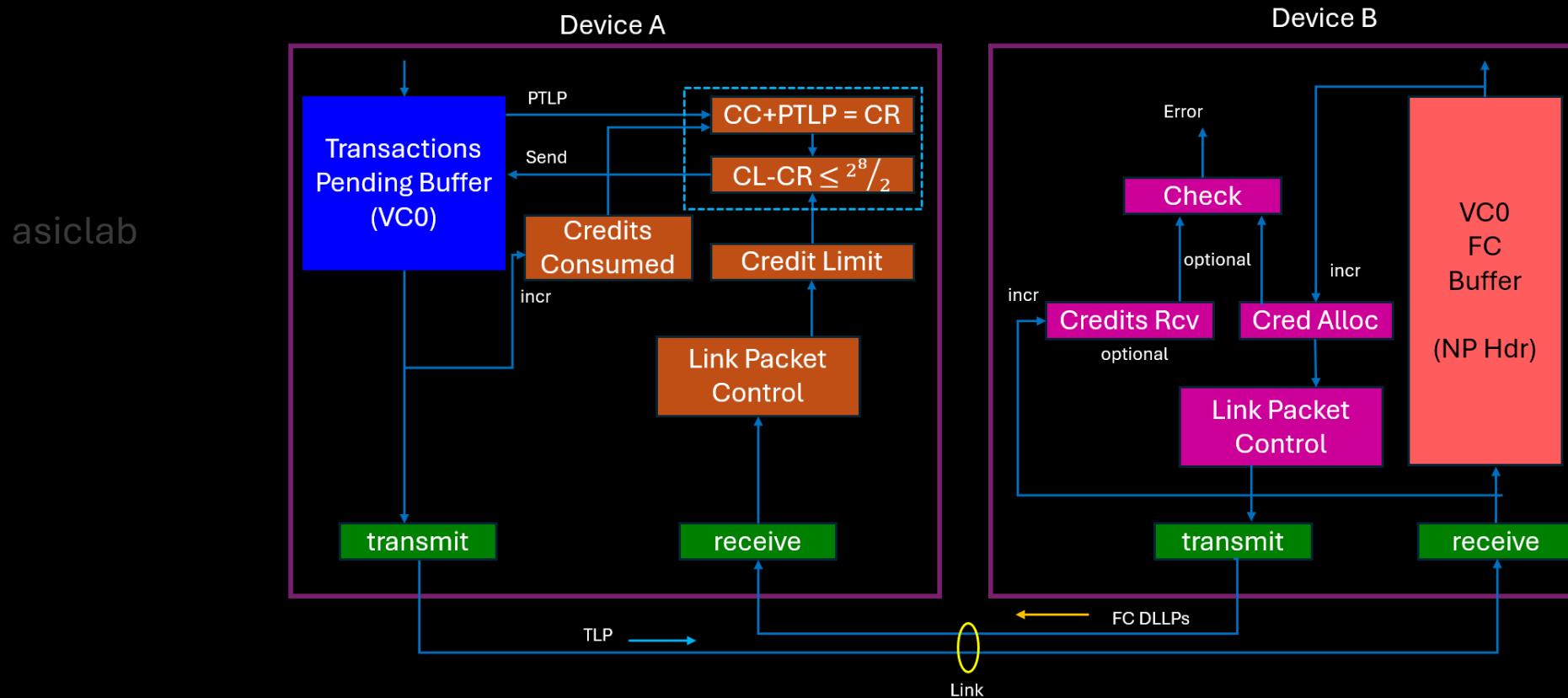
- One FC credit for data buffers = 4 DWs
- One FC credit for headers = 1 max-sized header plus the optional digest
 - 5DW for request headers (posted and non-posted)
 - 4DW for completion headers
- Without sufficient credits a TLP type can't be sent, through other types may be sent if they have enough credits
- For TLPs that include data (writes and completions with data), the transmitter must check credits for both header and data

Flow Control Elements in DLL



Flow Control Elements in DLL

*Assumes flow control scale factor = 1



The Credits Allocated counter is hardware initialized with the largest credit value supported by the receive buffer.

CC = Credits Consumed

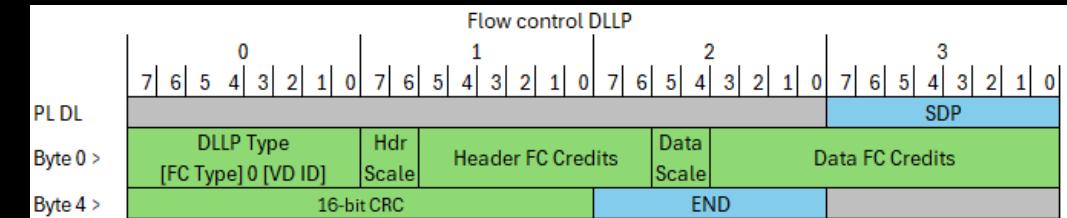
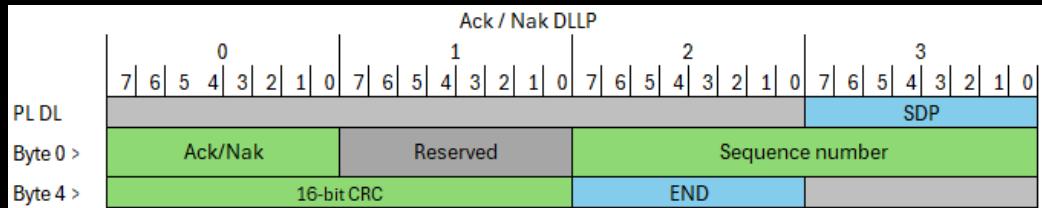
PTLP = Pending TLP (# of credits needed for this TLP)

CR = Credit Required

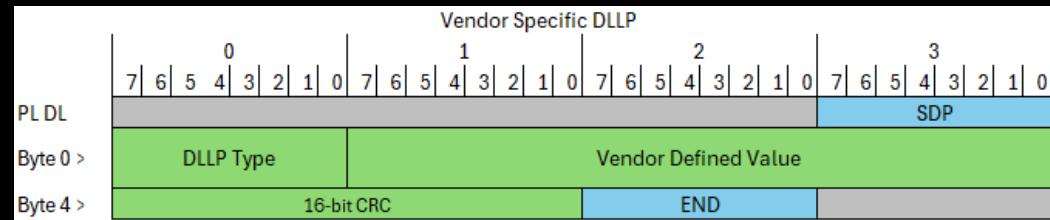
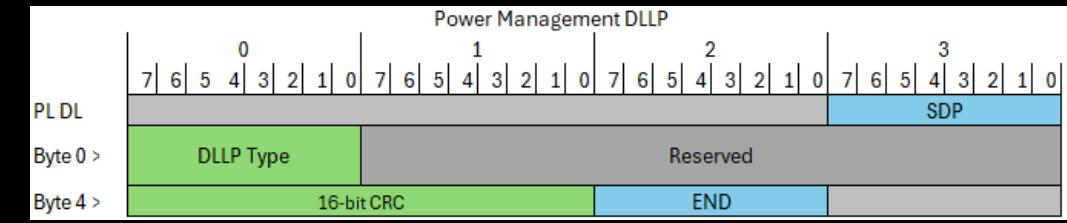
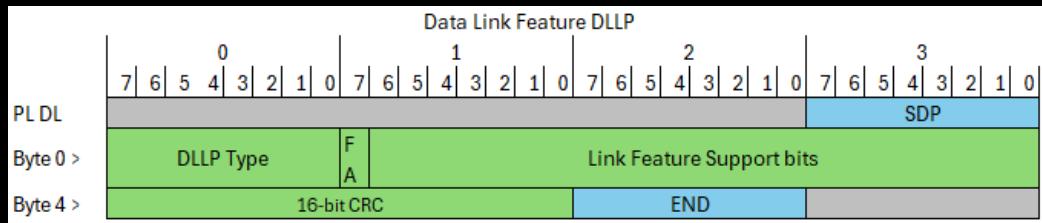
CL = Credit Limit

© Copyright protected 2024

DLL Packet format



asiclab



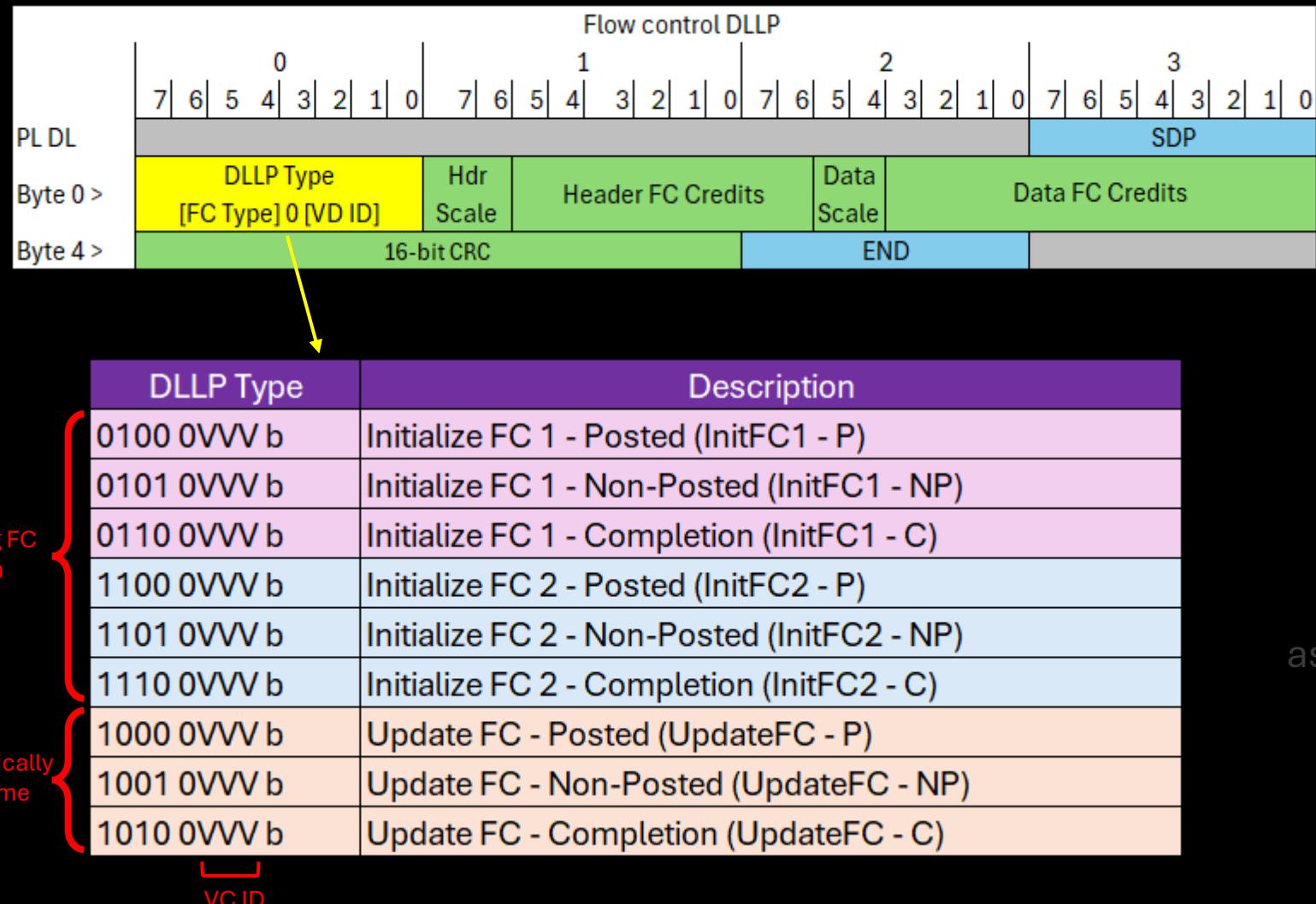
asiclab

Data Link Layer Header
Physical Layer Header
DO NOT Distribute

© Copyright protected 2024

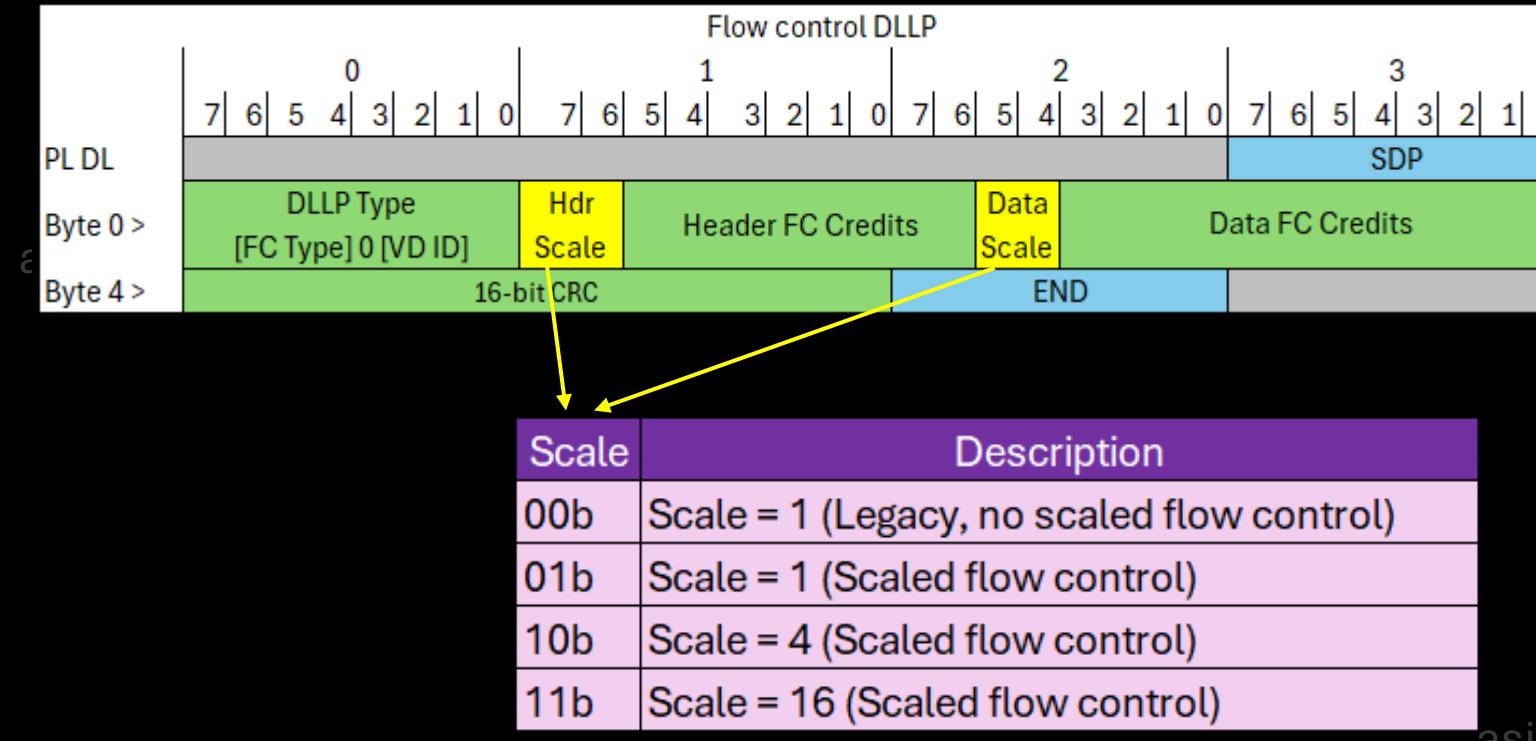
asiclab
Learn. Thrive

Flow Control DLLPs



asiclab

Flow Control DLLPs



asiclab

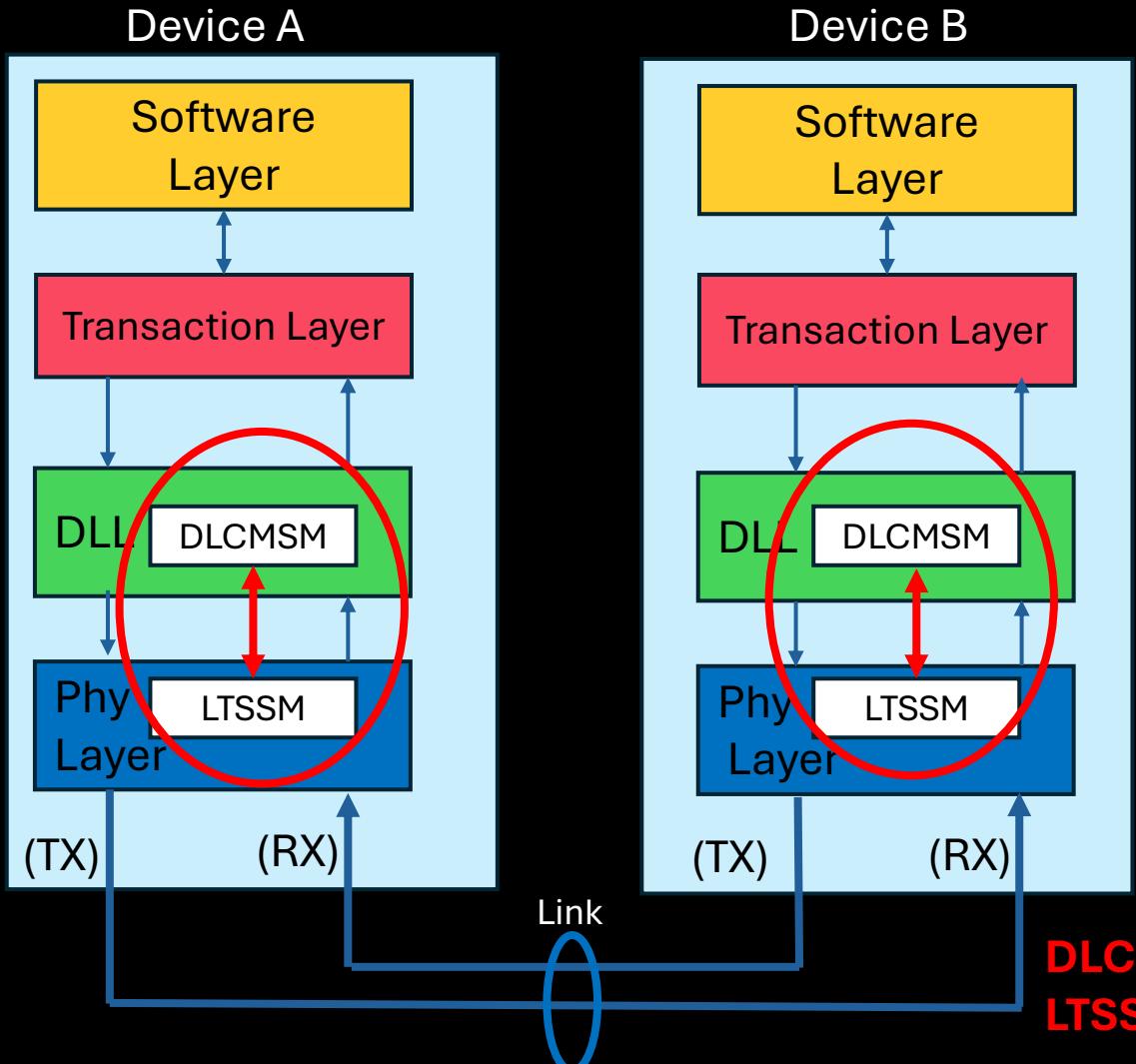
Scaled FC – Credit Calculation

- Scaled Flow Control can change the meaning of the value ADVERTISED in flow control DLLPs (InitFC1, InitFC2 and UpdateFC).
- The meaning of a flow control credit within a device DOES NOT CHANGE
 - 1 data credit always equals 16 bytes
 - 1 header credit always equals 16 or 20 bytes (depending on type of header)
- Scaled Flow Control DOES change the size of counters within the devices.
 - Larger counters allow a larger max number of credits to be advertised

Scale factor	Header Credit Counter Size	Data Credit Counter Size
1	8-bit	12-bit
4	10-bit	14-bit
16	12-bit	16-bit

asiclab

Preparing Link for TLP Transmission

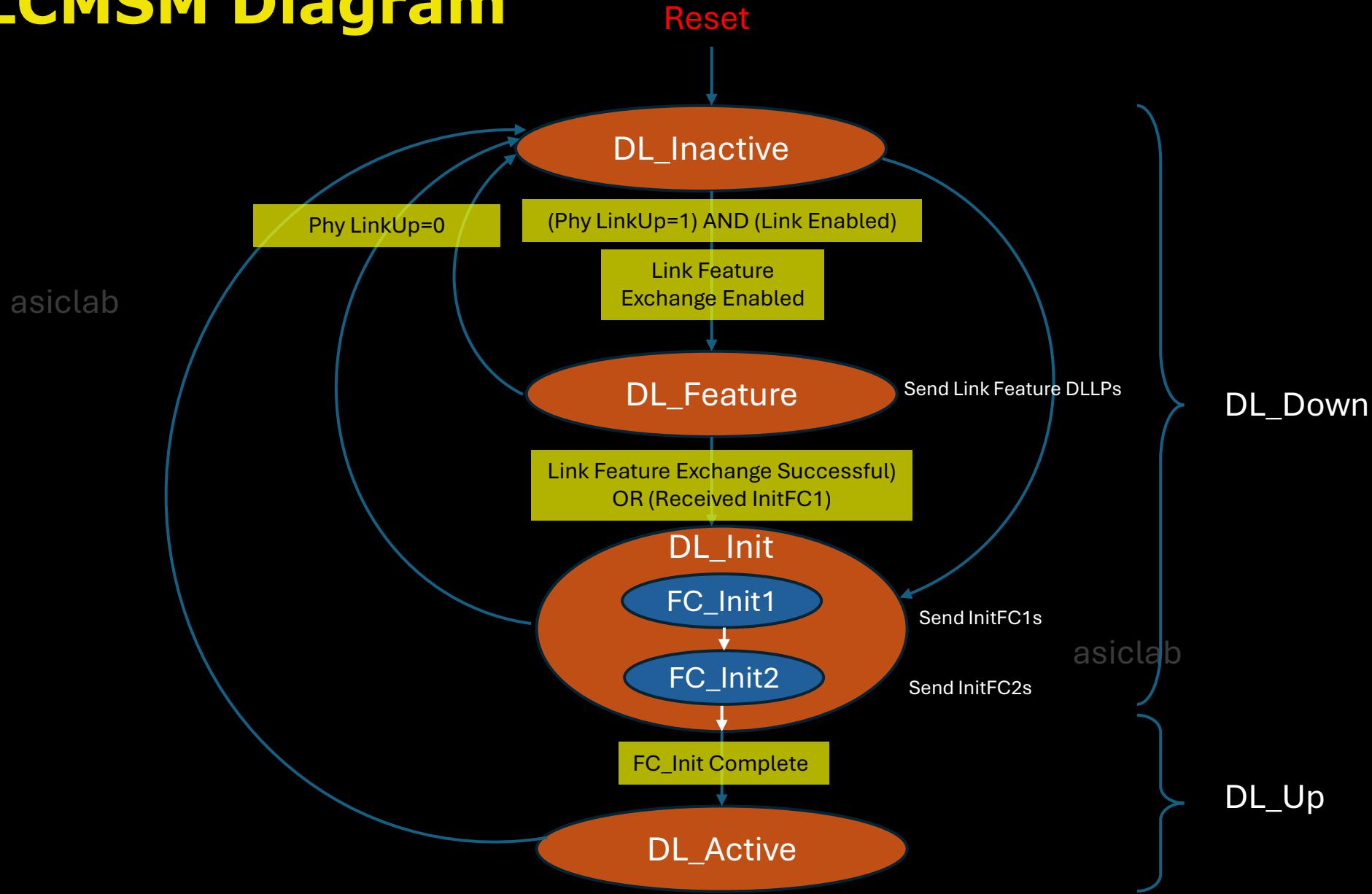


- Following Reset, two state machines interact to prepare the PCIe interface to send and receive TLPs:
 - DLCMSM tracks the state of the Link and initializes Flow Control for VC0
 - LTSSM trains and initializes the Link

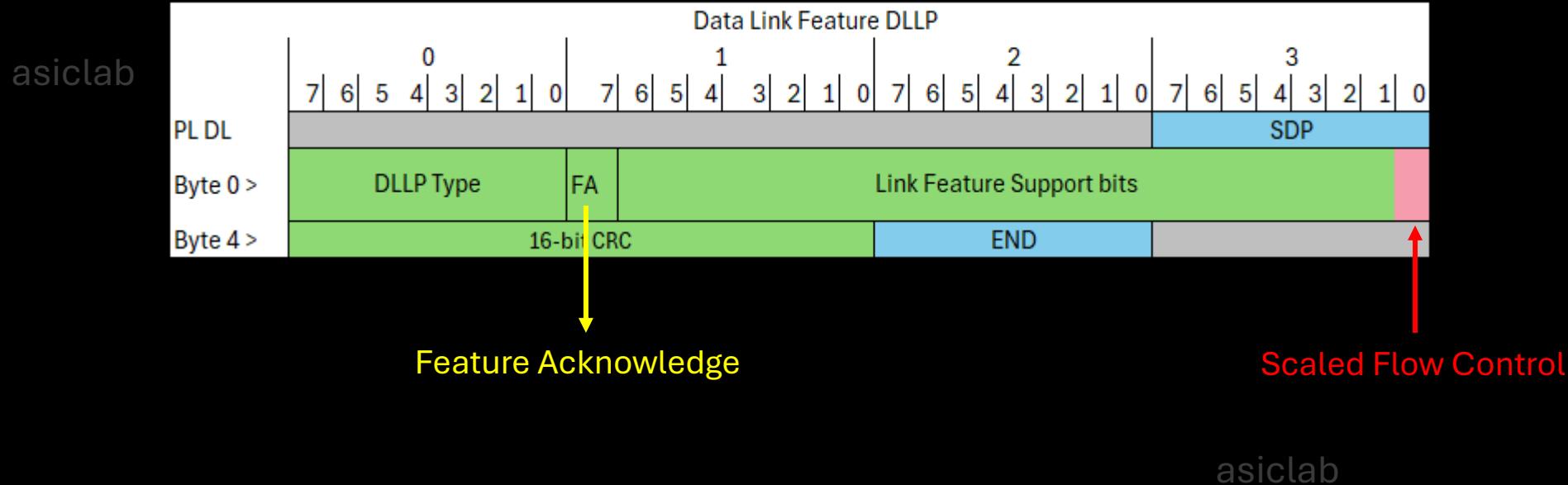
asiclab

DLCMSM: Data Link Control & Management State Machine
LTSSM: Link Training Status State Machine

DLCMSM Diagram

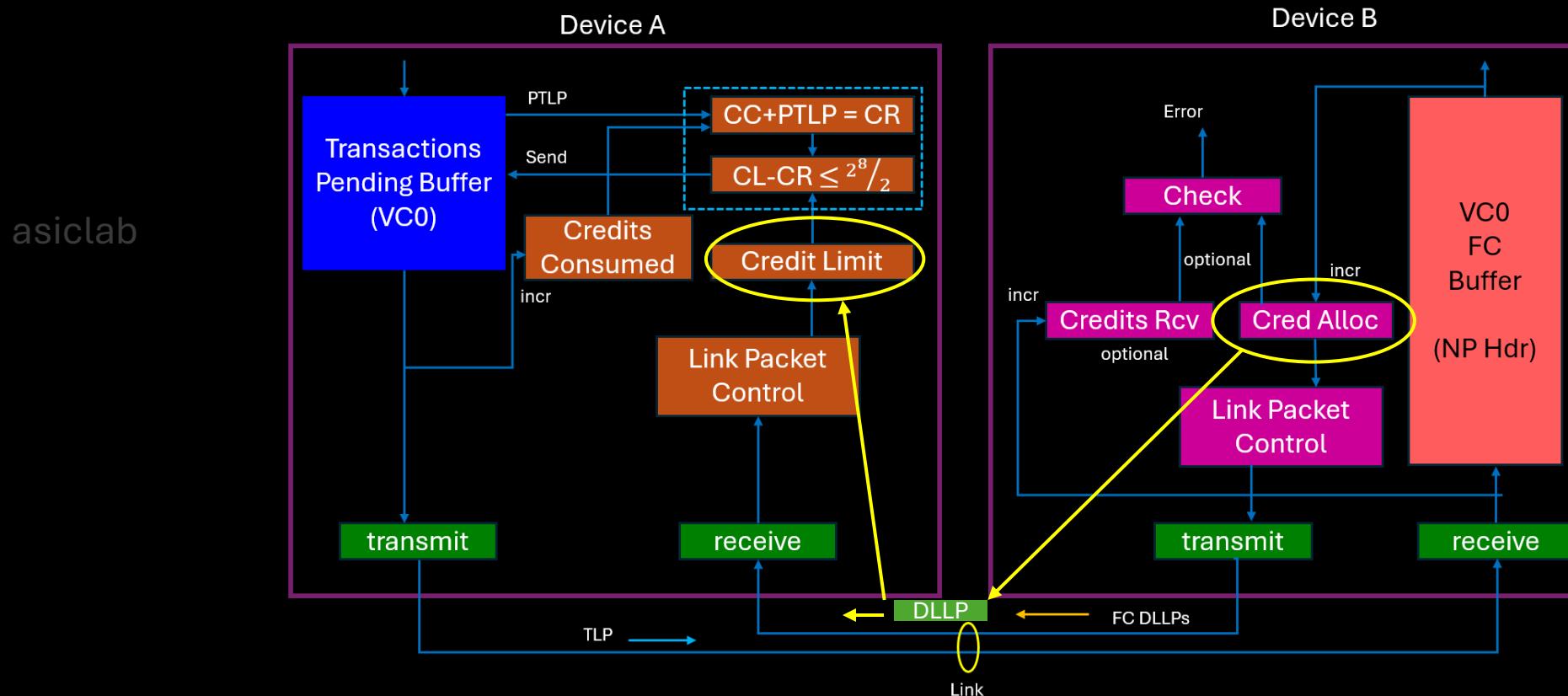


Data Link Feature DLLPs



FC Initialization in DLL

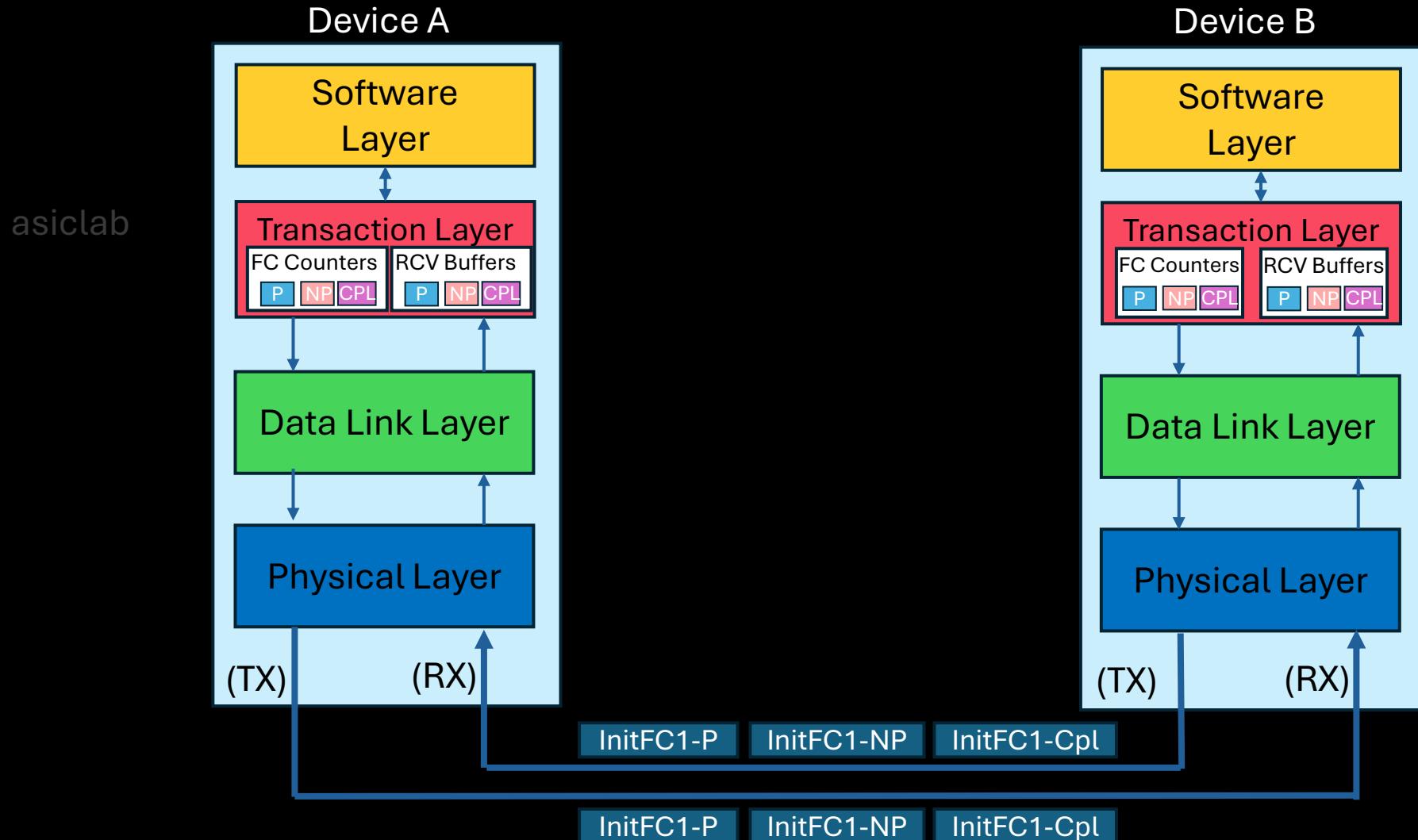
*Assumes flow control scale factor = 1



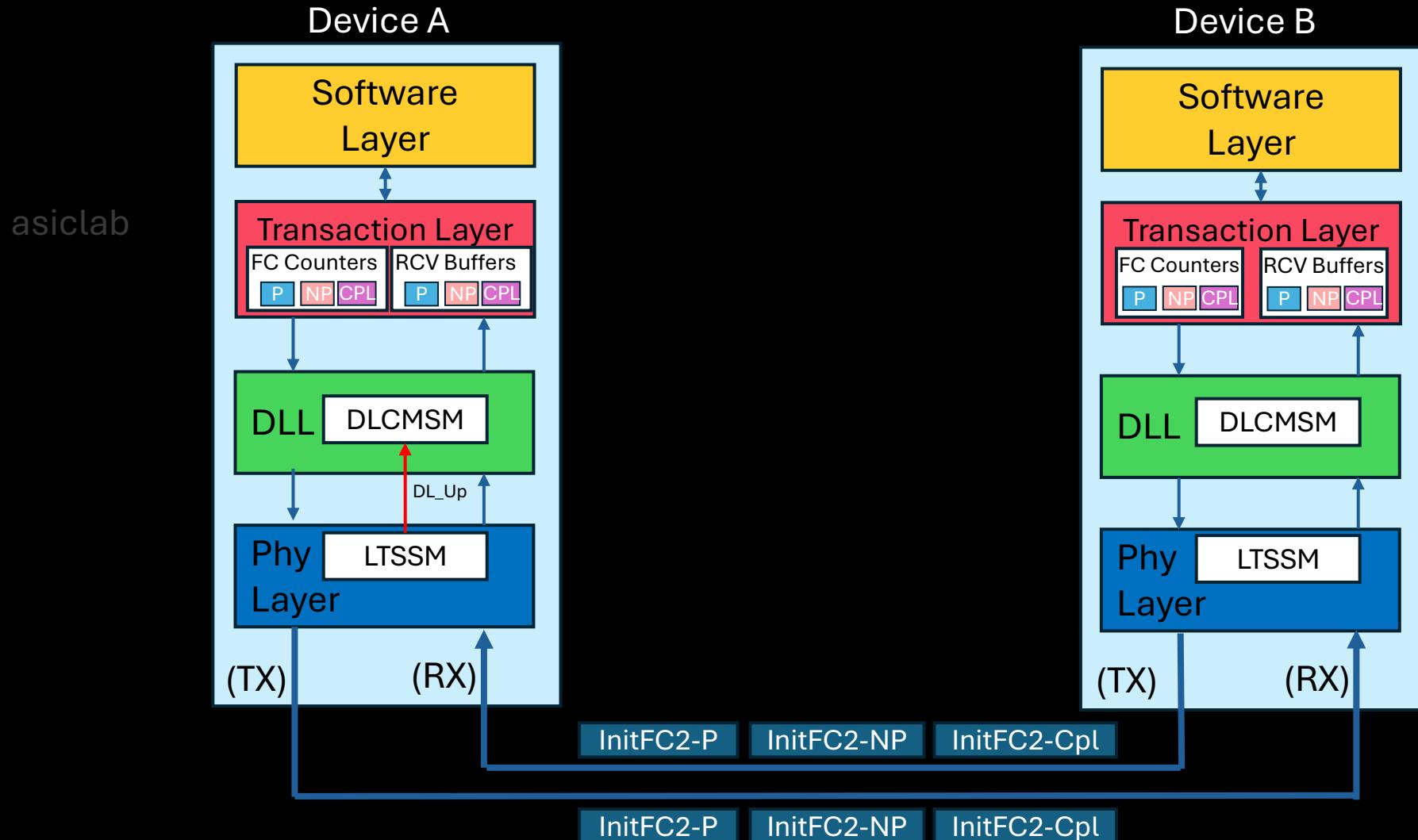
The Credits Allocated counter is sent across the Link in a FC DLLP and updates the Credit Limit counter

InitFC1/ InitFC2 DLLPs are used during FC initialization while UpdateFC DLLPs are used during runtime

InitFC1 DLLP Exchange



InitFC2 DLLP Exchange



Flow Control Initialization Sequence

- Flow Control credits for VC0 are automatically initialized after Link Training because VC0 can't be disabled.
 - Other virtual channels may be enabled later by software, triggering Flow Control initialization for those channels at that time.
- PCI Express defines two flow control initialization states, FC_INIT1 and FC_INIT2
 - FC_INIT1: FC credits are exchanged
 - FC_INIT2: FC credit exchange is confirmed

asiclab

Minimum Initial Flow Control Advertisement

Credit Type	Minimum Advertisement (scale factor of 1)
Posted Request Header (PH)	1 credit (4 DW Hdr + 1 DW Digest = 5 DW)
Posted Request Data (PD)	Enough credits to accommodate biggest possible Max_Payload_Size of all Functions in the Device. For a value of 1024 bytes, $1024/16 = 64$ credits needed.
Non-Posted Req. Header (NPH)	1 credit (4 DW Hdr + 1 DW Digest = 5 DW)
Non-Posted Req. Data (NPD)	1 credit or 2 credits if AtomicOps are supported
Completion Header (CPLH)	1 credit (3 DW Hdr + 1 DW Digest = 4 DW) for Switch Ports of Root Ports that support peer-to-peer transfers. For Endpoints or Root Ports that don't support peer-to-peer, infinite credits must be advertised (indicated by value of 0 during initialization)
Completion Data (CPLD)	Enough credits to accommodate biggest possible Max_Payload_Size of all Functions in a Switch Port or Root Port that supports peer-to-peer transfers. For Endpoints or Root Ports that don't support peer-to-peer infinite credits must be advertised.

Minimum Initial Flow Control Advertisement, cont'd

Credit Type	Minimum Advertisement	
	Scale factor 4	Scale factor of 16
Posted Request Header (PH)	1 credit (4 units*)	1 credit (16 units*)
Posted Request Data (PD)	Enough credits to accommodate biggest possible Max_Payload_Size of all Functions in the Device plus one. Max_Payload_Size/ (FC Unit Size * 4) + 1 For a Max_Payload_Size of 1024 bytes, $1024/(16*4)+1 = 17d.$	Max_Payload_Size/(FC Unit Size * 16) + 1 For a Max_Payload_Size of 1024 bytes $1024/(16*16)+1 = 5d.$
Non-Posted Req. Header (NPH)	1 credit (4 units)	1 credit (16 units)
Non-Posted Req. Data (NPD)	1 credit (4 units) or 2 credits (8 units) if AtomicOps are supported	1 credit (16 units) or 2 credits (32 units) if AtomicOps are supported
Completion Header (CPLH)	1 credit (4 units) for Switch Ports and Root Ports that support peer-to-peer transfers. 0 credits (infinite) for Endpoints and Root Ports that don't support peer-to-peer	1 credit (16 units) for Switch Ports and Root Ports that support peer-to-peer transfers. 0 credits (infinite) for Endpoints and Root Ports that don't support peer-to-peer
Completion Data (CPLD)	Same as Posted Request Data (PD) requirements for Switch Ports and Root Ports that support peer-to-peer transfers. 0 credits (infinite) for Endpoints and Root Ports that don't support peer-to-peer	

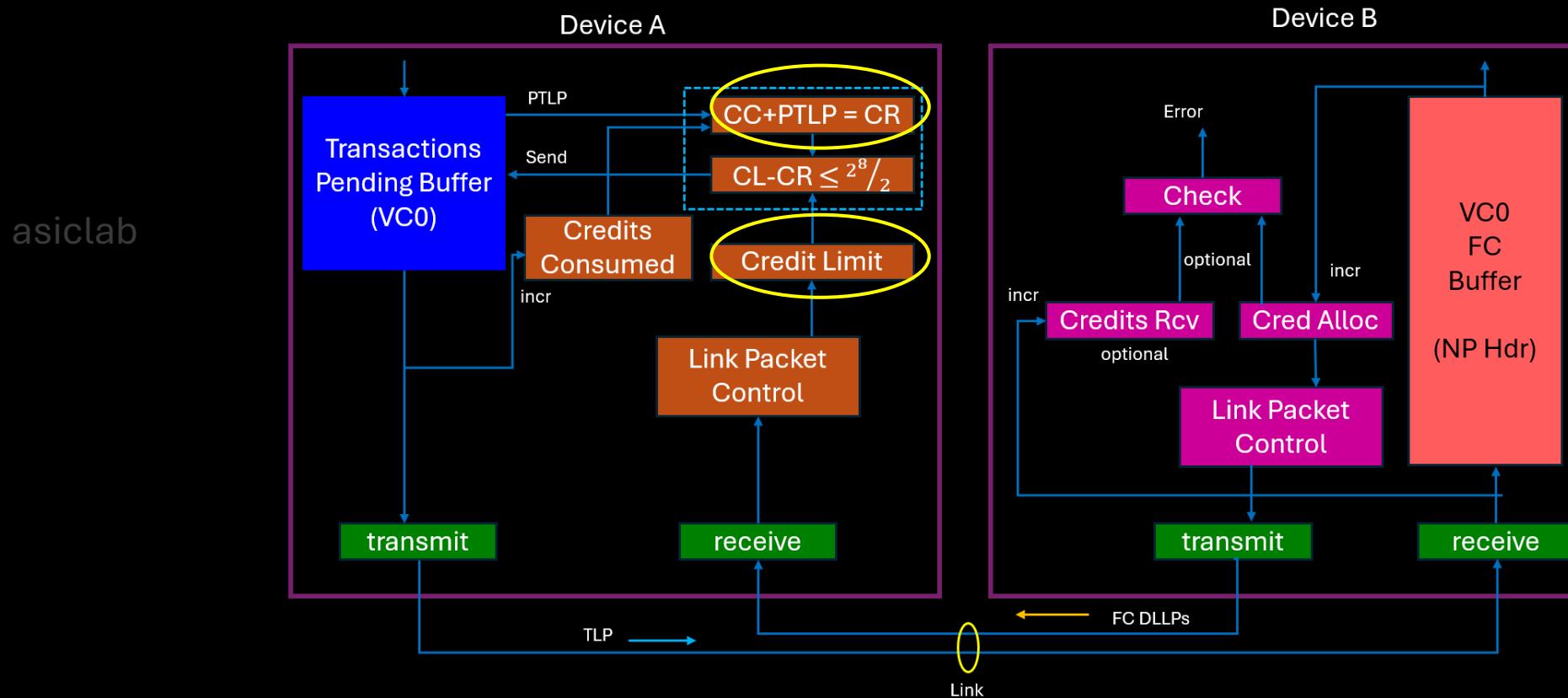
Infinite Buffer Advertisement

- Allows transmitter to send any number of TLPs without checking credits
- Indicated at initialization by advertising a credit value of zero in InitFC1 and InitFC2 DLLPs
- No UpdateFC DLLP are needed if a receiver advertised infinite header AND data credits

asiclab

Flow Control After Initialization

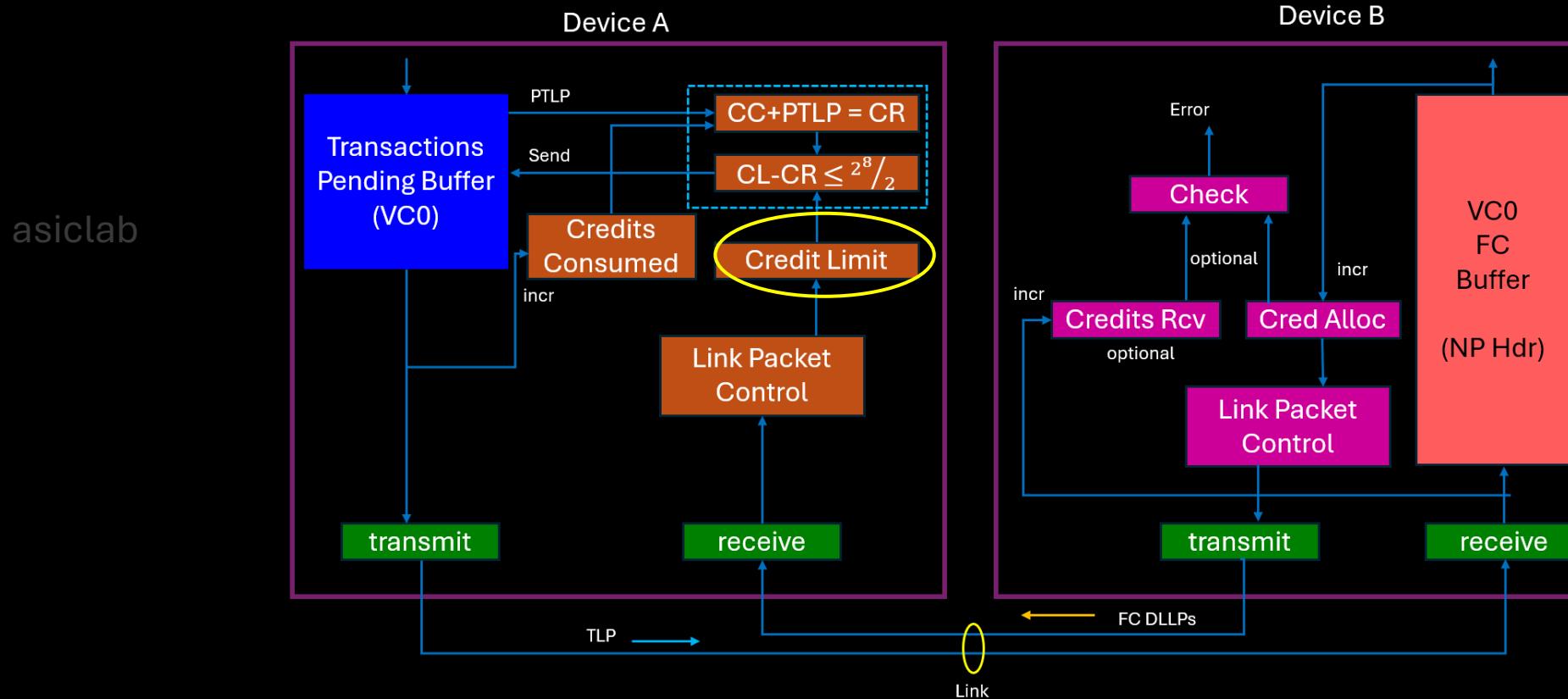
*Assumes flow control scale factor = 1



- Prior to sending TLP, transmitter checks Credits Required (CR) against the Credit Limit (CL) to verify buffer space for the next TLP.
- CR is the sum of Credit Consumed (CC) plus the credits required to send the Pending TLP (PTLP)

Two's Complement Check Before Tx

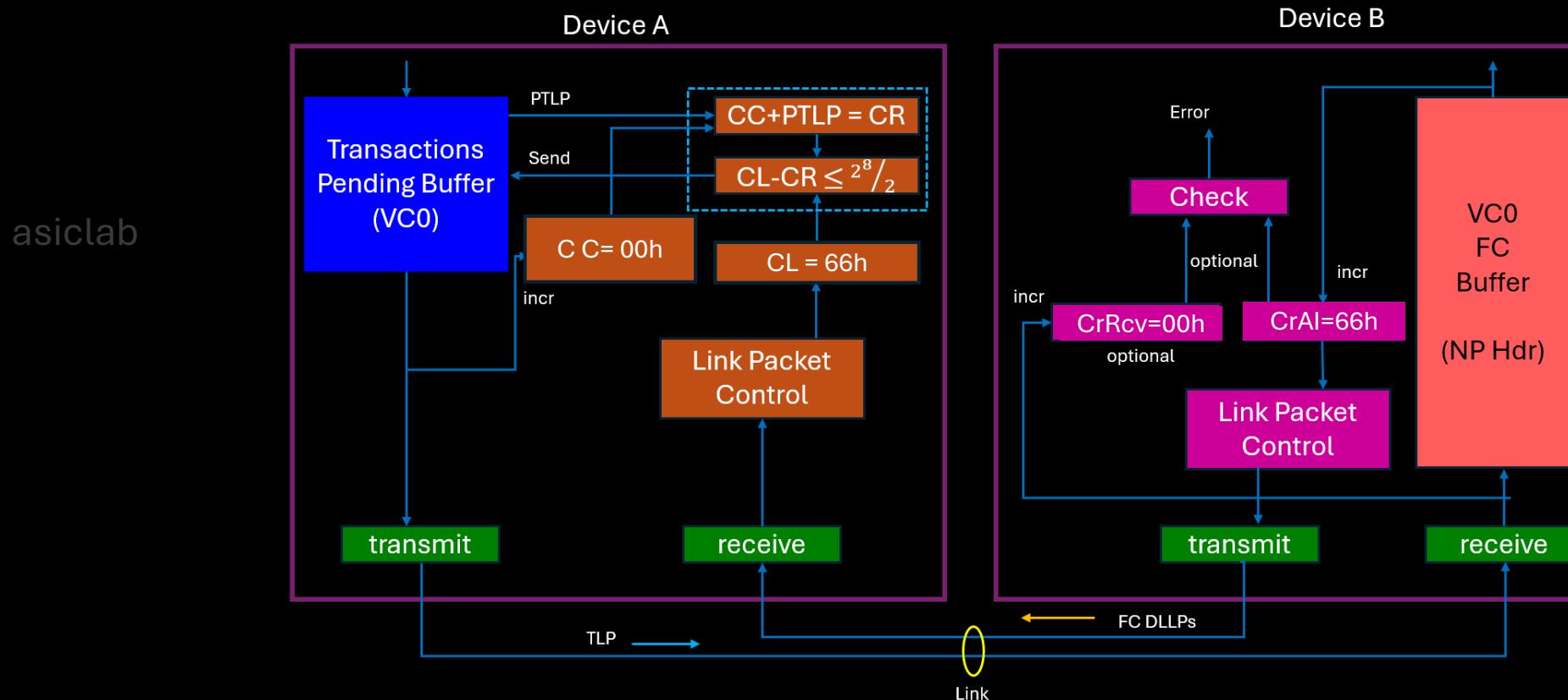
*Assumes flow control scale factor = 1



- For Header Credit Check
 $[CL - (CC + PTLP)] \bmod 256 \leq 128$
- For Data Credit Check
 $[CL - (CC + PTLP)] \bmod 4096 \leq 2048$

Example Stage 1: Initialization Complete

*Assumes flow control scale factor = 1



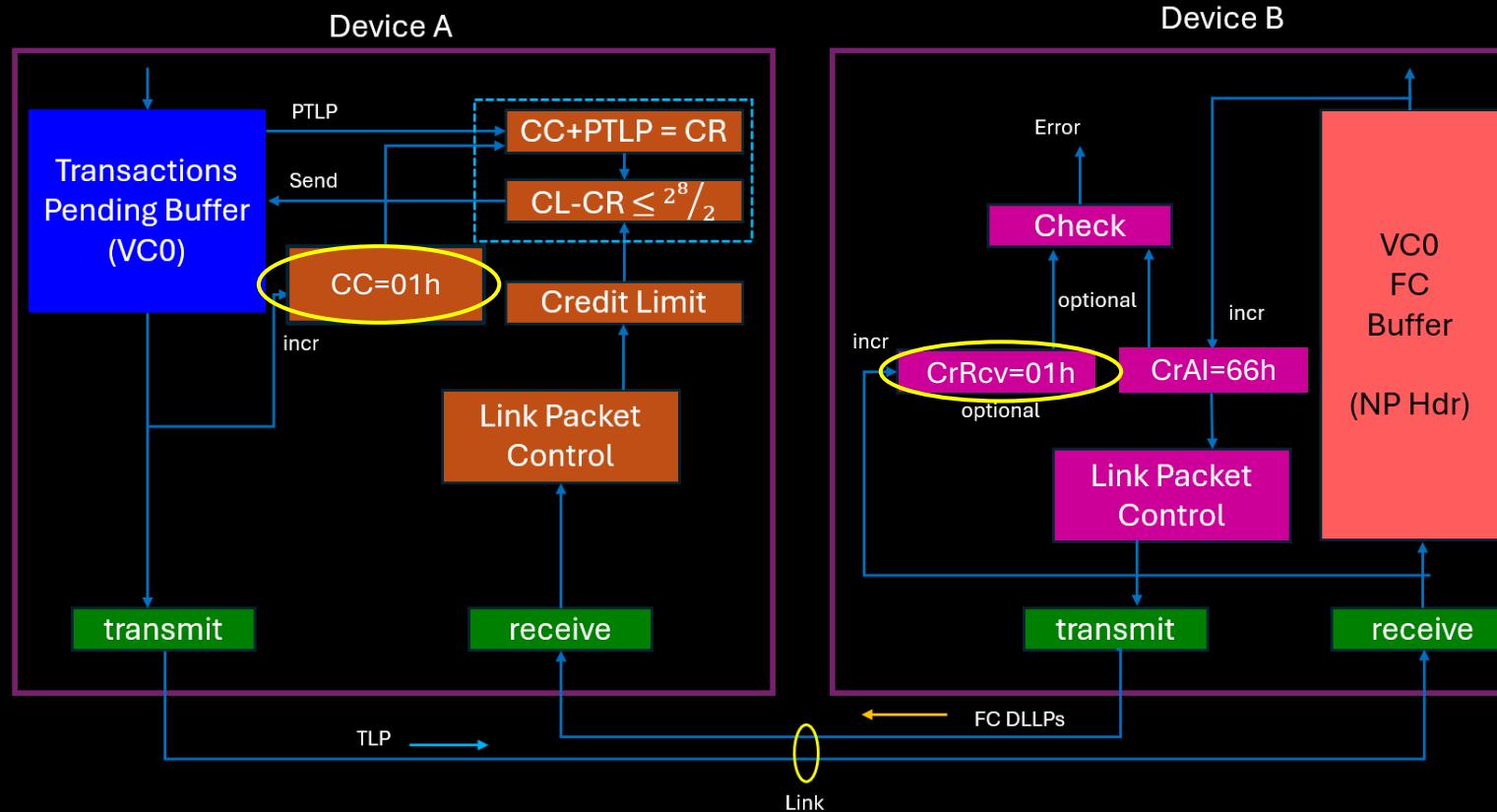
CC = Credits Consumed
CL = Credit Limit
PTLP = Pending TLP

CrAI = Credits Allocated
CrRcv = Credits Received

Example Stage 2: FC First TLP Sent

*Assumes flow control scale factor = 1

asiclab

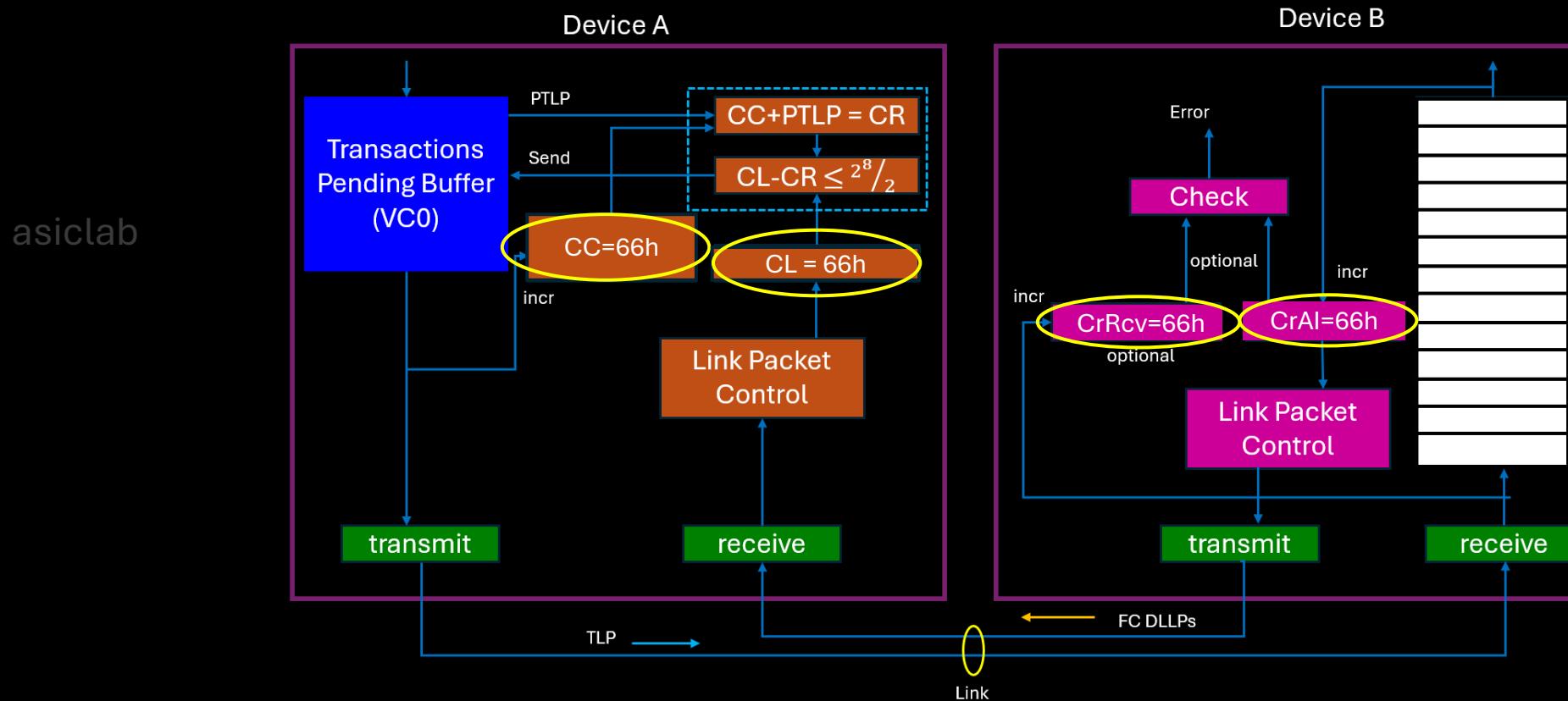


CC = Credits Consumed
CL = Credit Limit
PTLP = Pending TLP

CrAI = Credit sAllocated
CrRcv = Credits Received

Example Stage 3: Rx FC Buffer Fills Up

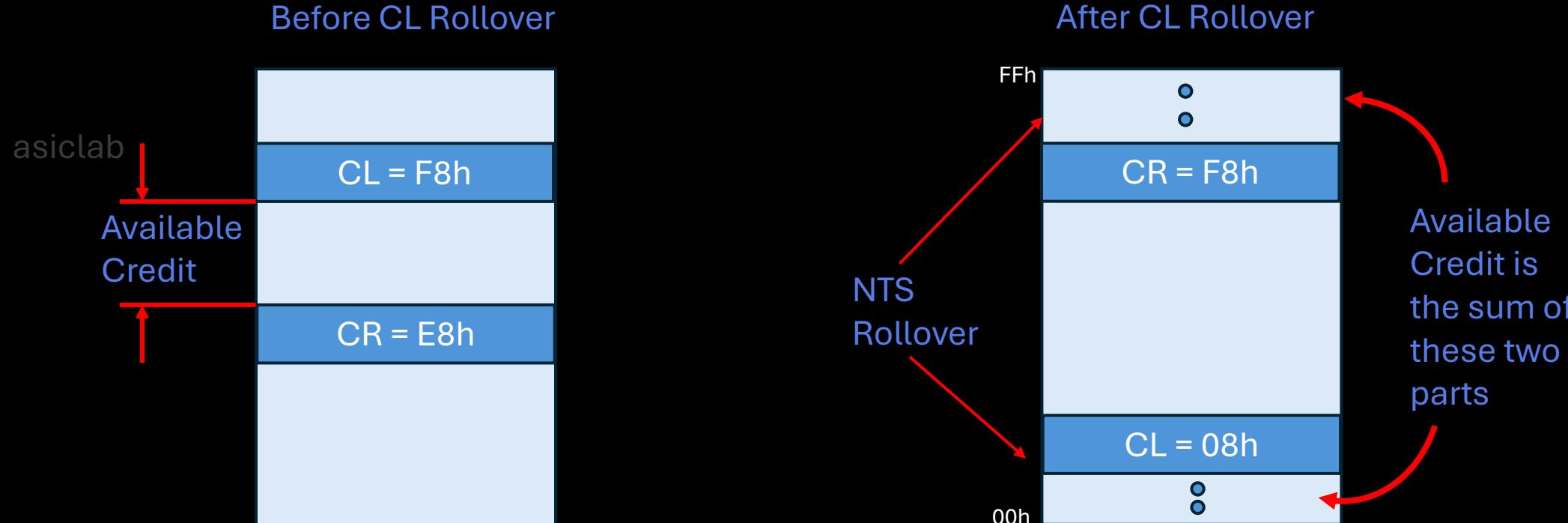
*Assumes flow control scale factor = 1



- In Device A, CC = CL = 66h; Transmitter has no credits
- In Device B, CrRcv = CrAI = 66h; Receiver buffer full

FC Counter Rollover

*Assumes flow control scale factor = 1



Using 2's complement:

$$\text{CL } 11111100 (\text{F8h})$$

$$\begin{aligned} &+ \text{CR } 00011000 (\text{E8h } 2\text{'s complement}) \\ &= 00010000 (0\text{Fh}) \end{aligned}$$

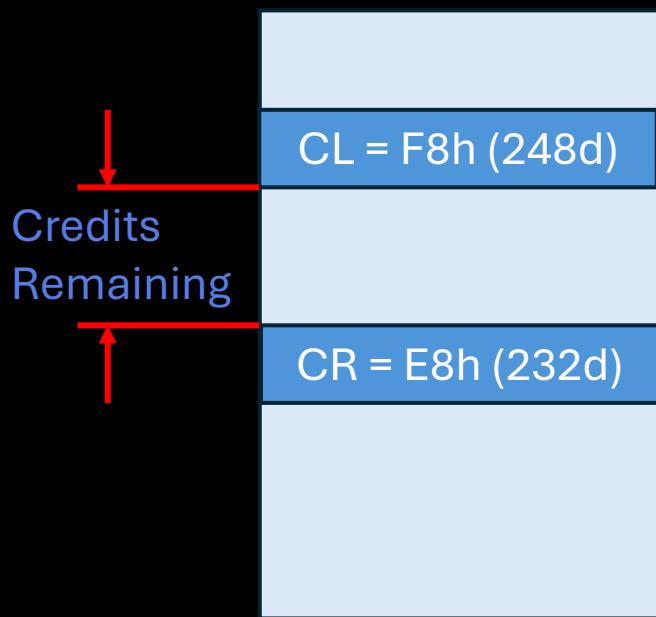
Using 2's complement:

$$\text{CL } 00001000 (08h)$$

$$\begin{aligned} &+ \text{CR } 00001000 (\text{F8h } 2\text{'s complement}) \\ &= 00010000 (0\text{Fh}) \end{aligned}$$

FC Counter Rollover Problem

- Counters have a potential problem: they can't tell whether CL stayed ahead of CR as it should, or if CR passed CL by mistake.
- If such a mistake happened, the subtraction result would be a large value. To protect again this, it will be considered an error if the difference between pointers ever exceeds half the counter value.
- Consequently, the biggest buffer allowed can only use half the counter max. value.



Did Credit Limit stay ahead as it should or did Credits required pass it by mistake?
Unsigned subtraction result won't tell us unless we restrict the options.

CL = Credit Limit
CR = Credits Required

Max Advertised Receive Buffer Size

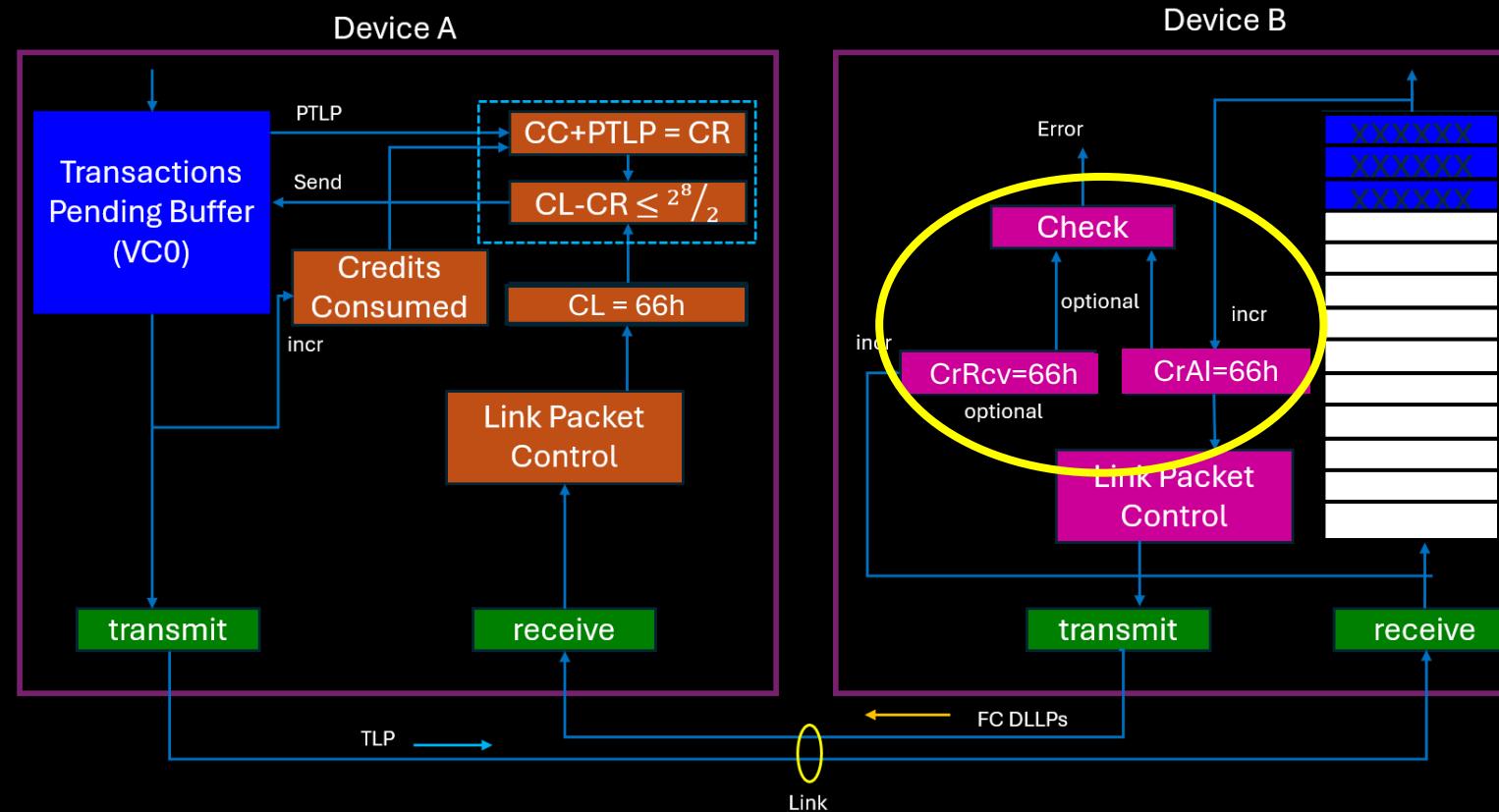
- Two's complement arithmetic means a device can only advertise a max of half the counter spec.
- The maximum number of credits that can be advertised by a receiver are:
asiclab

Scale factor	Header Credit Counter Size	Max Advertised Header Credits	Data Credit Counter Size	Max Advertised Data Credits
1	8-bit	127	12-bit	2047
4	10-bit	508	14-bit	8188
16	12-bit	2032	16-bit	32752

Example Stage 4: Buffer Overflow Check

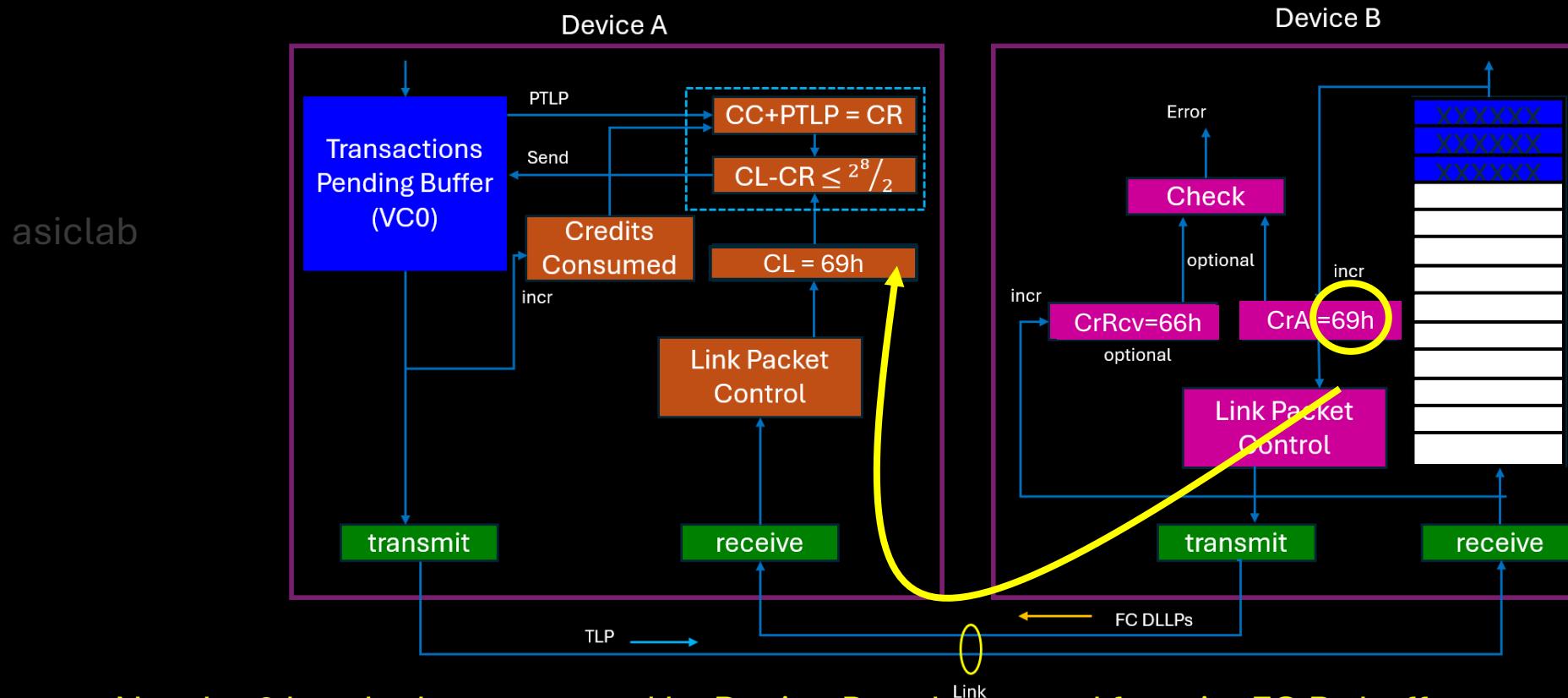
*Assumes flow control scale factor = 1

asiclab



Example Stage 5: FC Update

*Assumes flow control scale factor = 1



- Now let 3 header be consumed by Device B and removed from its FC Rx buffer
- Credits Allocated counter increments from 66h to 69h
- FC Update DLLP delivered from Device B to A and CL counter updated
- Next check for sending TLP will succeed

Update FC Frequency

- An UpdateFC DLLP for each packet type (P, NP, Cpl) must normally be scheduled within every 30µS (-0% /+50%). Exceptions:
 - If Link is in a state other than L0 or L0s, no updates are sent
 - If Extended Sync bit (within Link Control register) is set the limit becomes 120 µS (-0%/+50%)
- The PCIe specification recommends that a receiver send UpdateFC packets at the rates specified the following table (based on equation below)
 - This is NOT a requirement, only “recommendation”
$$\text{Update Rate} = \frac{(\text{Max_Payload_Size} + \text{TLP Overhead}) * \text{UpdateFactor} + \text{Internal Delay}}{\text{LinkWidth}}$$

asiclab

Transmitter Checks for FC Updates

- Transmitter may optionally check that FC updates are received at a minimum frequency (at least every 30µs)
 - Only check when Link is in L0 or L0s
 - Use timer with a limit of 200µs (-0% / +50%)
 - Timer is reset with receipt of FC DLLP, or with receipt of any DLLP
 - Timer expiration causes PHY Layer to retrain the Link via LTSSM Recovery state
 - Timeout flow control mechanism is disabled if infinite credits were advertised

asiclab

Posted Request Acceptance Rule

- A Posted Request can't be delayed for more than $10\mu s$ (Posted Request Acceptance Limit).
- Consequently, the device must either:
 - (a) be able to process received Posted Requests and return FC credits within $10\mu s$, or
 - (b) depend on a restricted programming model to ensure that a Posted Request is never sent to the device when it's unable to service the request within $10\mu s$.
- The $10\mu s$ limit does not apply under certain conditions (e.g: just after a reset)

asiclab

asiclab

Transaction Ordering

asiclab

Transaction Ordering

- Benefits
 - Satisfy the Producer Consumer model
 - Improves performance by allowing for some transactions to be posted
 - Prevents Deadlock conditions
 - Help in Completion associations
- Bridge rules
 - Why transaction type may/may NOT/must be allowed to bypass other transaction type
- End point rules
 - What type of service dependencies are not allowed

asiclab

Producer Consumer Model

- Model Sequence
 - Produce creates the DATA and sets a FLAG
 - Consumer upon seeing the FLAG (poll, interrupt or doorbell), consumes the DATA and writes the completion STATUS update
 - Produces upon seeing the STATUS update (poll, interrupt, doorbell), clears the status and the sequence repeats
- Model Enforces
 - Consumer never sees the FLAG set before the DATA has been written
 - DATA is consumed before the producer can see the STATUS update
- The Producer, consumer, DATA, FLAG, and STATUS can reside anywhere in the system
- Multiple Producer – Consumers can operate at the same time
- Need to consider P/C chain in which A tells B to tell C that data is Valid.

PCI Ordering rules

asiclab

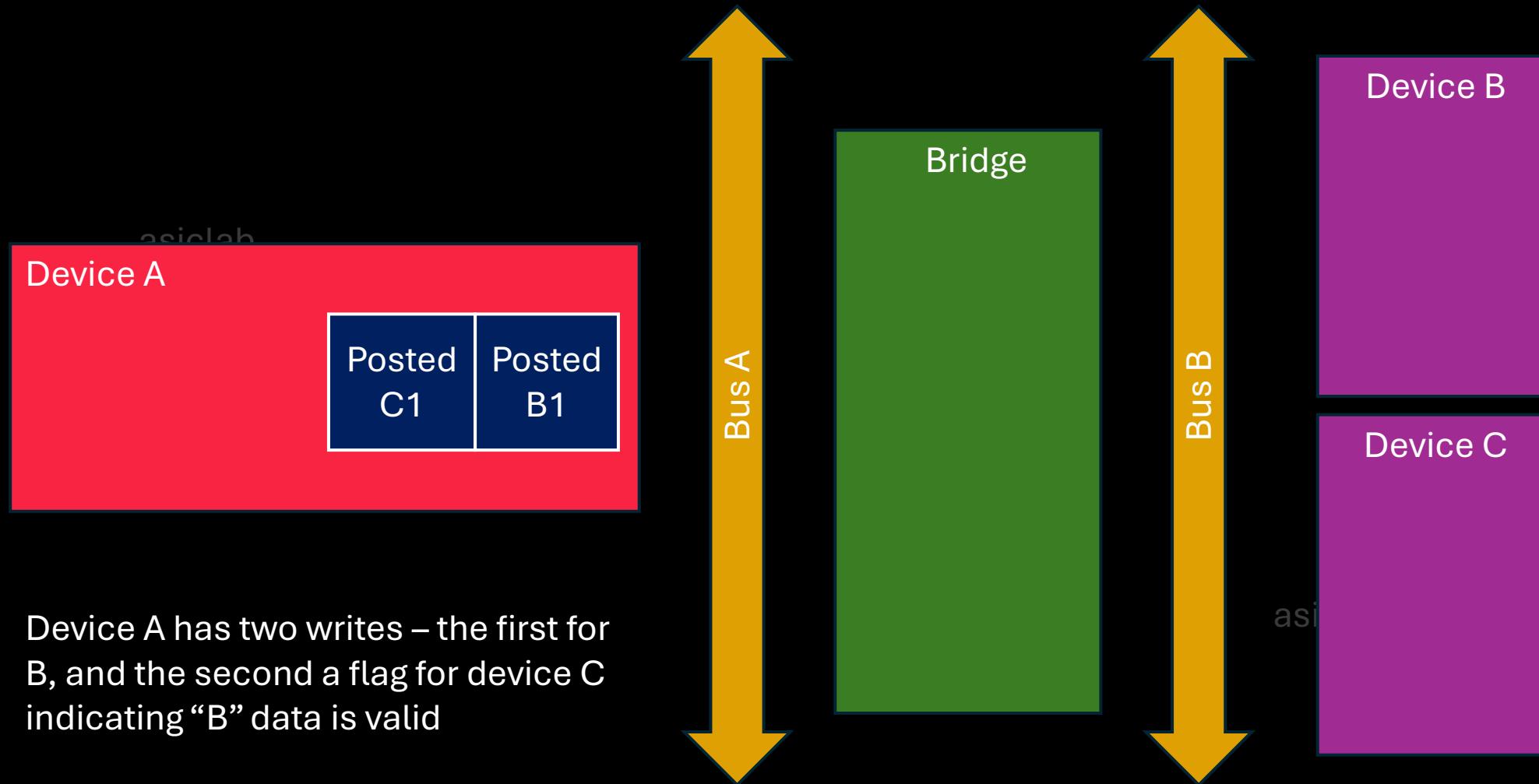
Row Pass Column		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Column 2)	Read Request (Column 3)	I/O or Cfg Write Request (Column 4)	Read Completion (Column 5)	I/O Cfg Write Completion (Column 6)
Posted Request	Memory Write Request (Row A)	No	Yes	Yes	Y/N	Y/N
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (Row D)	No	Yes	Yes	Y/N No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

PCI Ordering rules – Producer Consumer

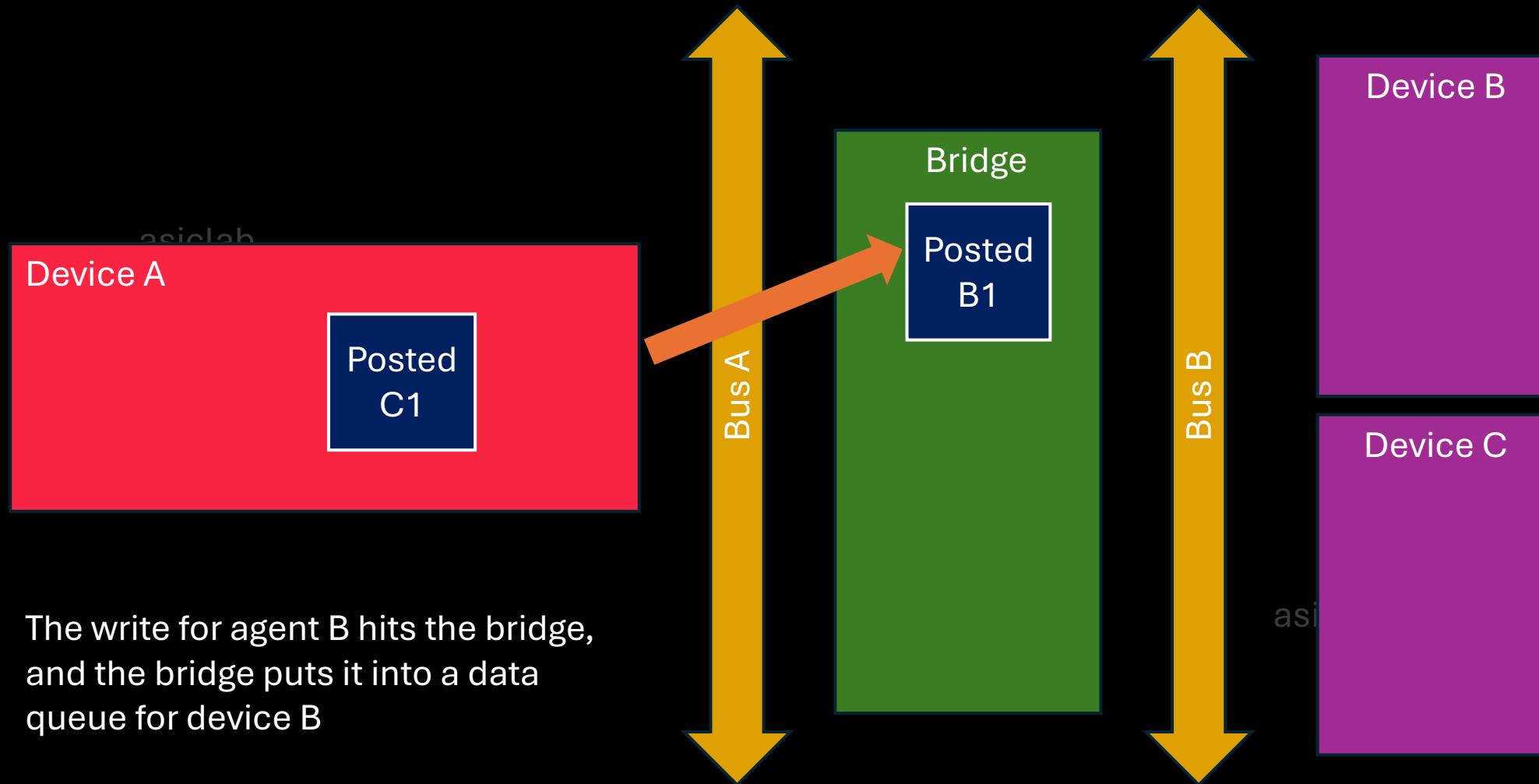
asiclab

Row Pass Column		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Column 2)	Read Request (Column 3)	I/O or Cfg Write Request (Column 4)	Read Completion (Column 5)	I/O Cfg Write Completion (Column 6)
Posted Request	Memory Write Request (Row A)	No	Yes	Yes	Y/N	Y/N
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (Row D)	No	Yes	Yes	Y/N No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

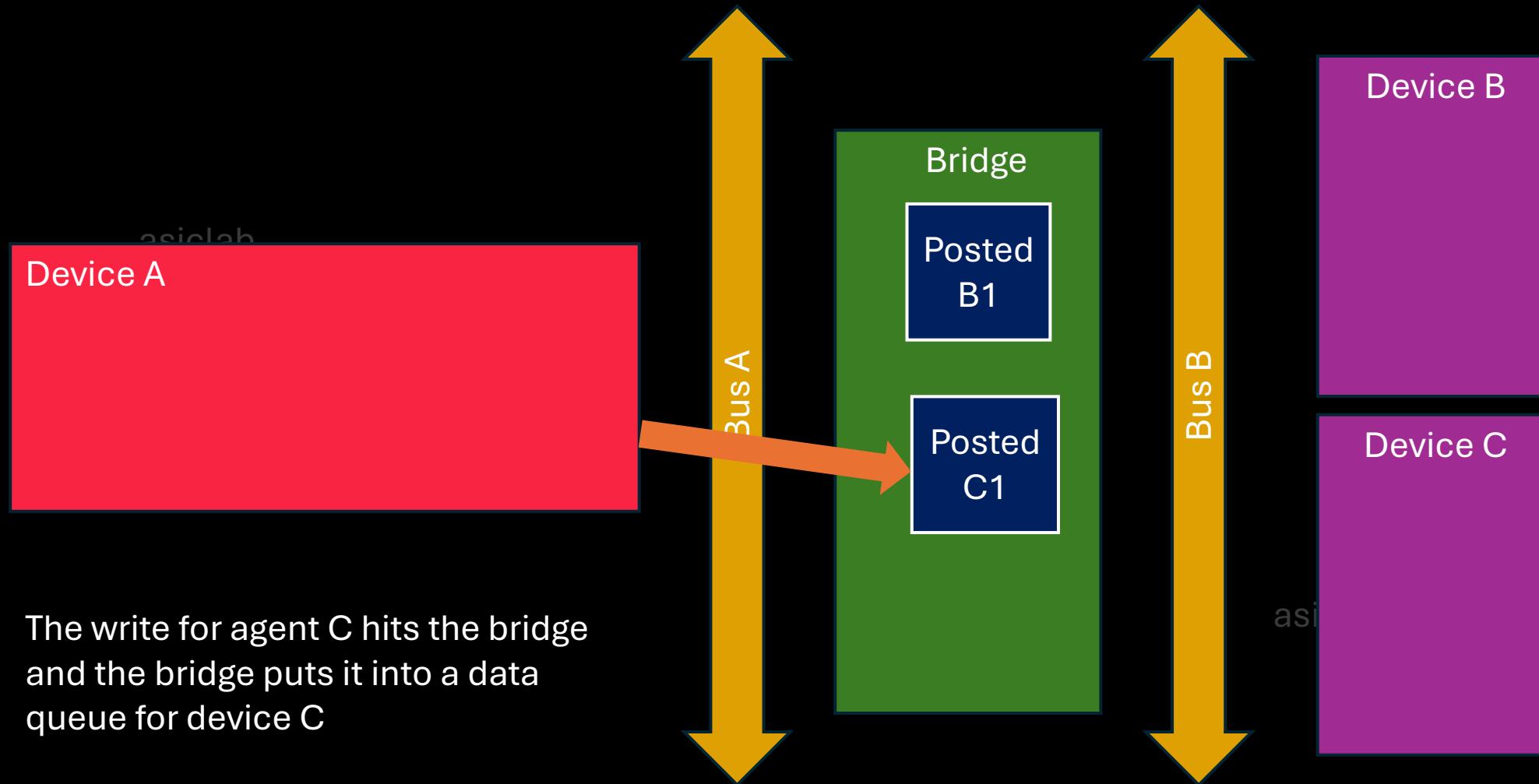
Ordering Violation Example



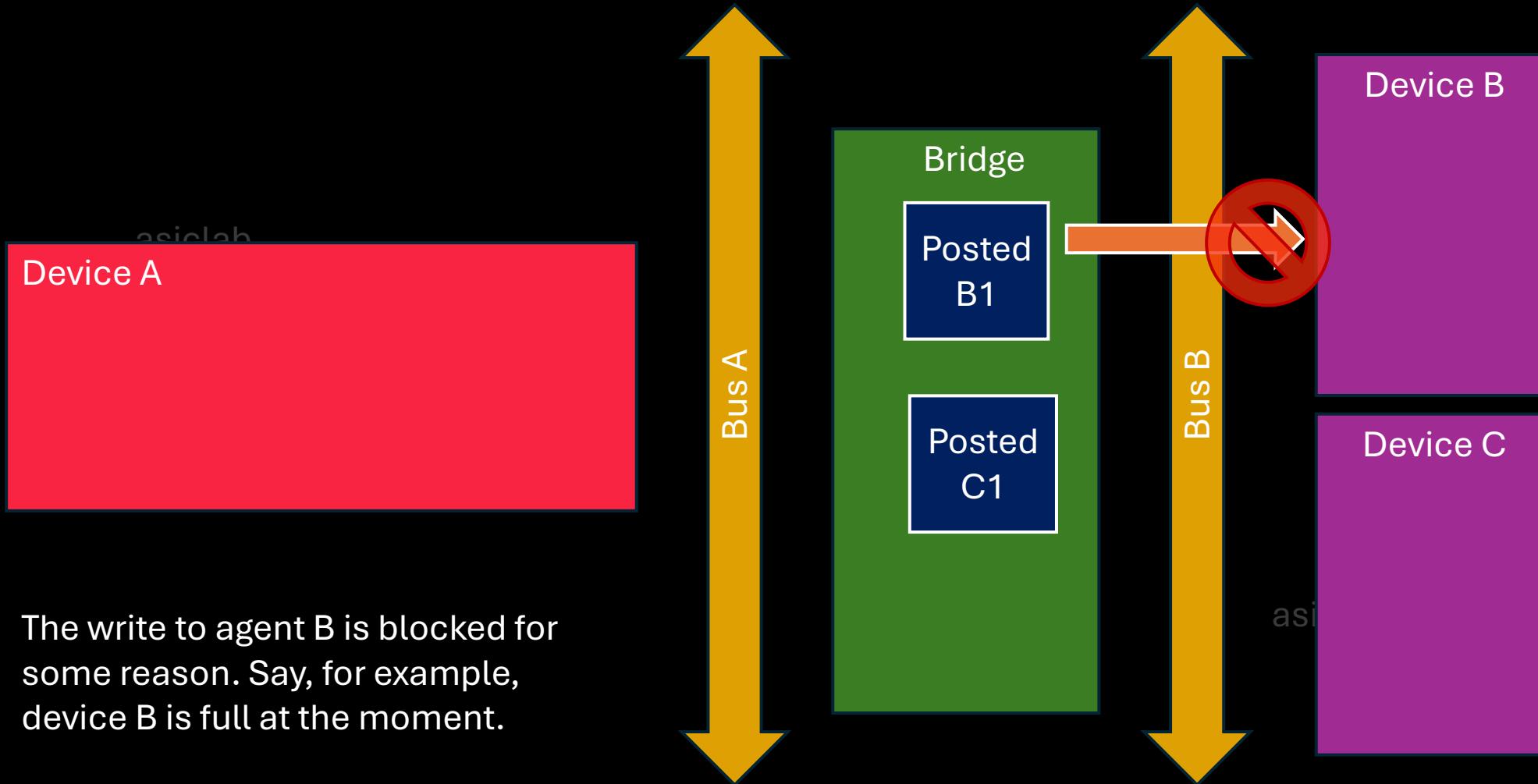
Ordering Violation Example



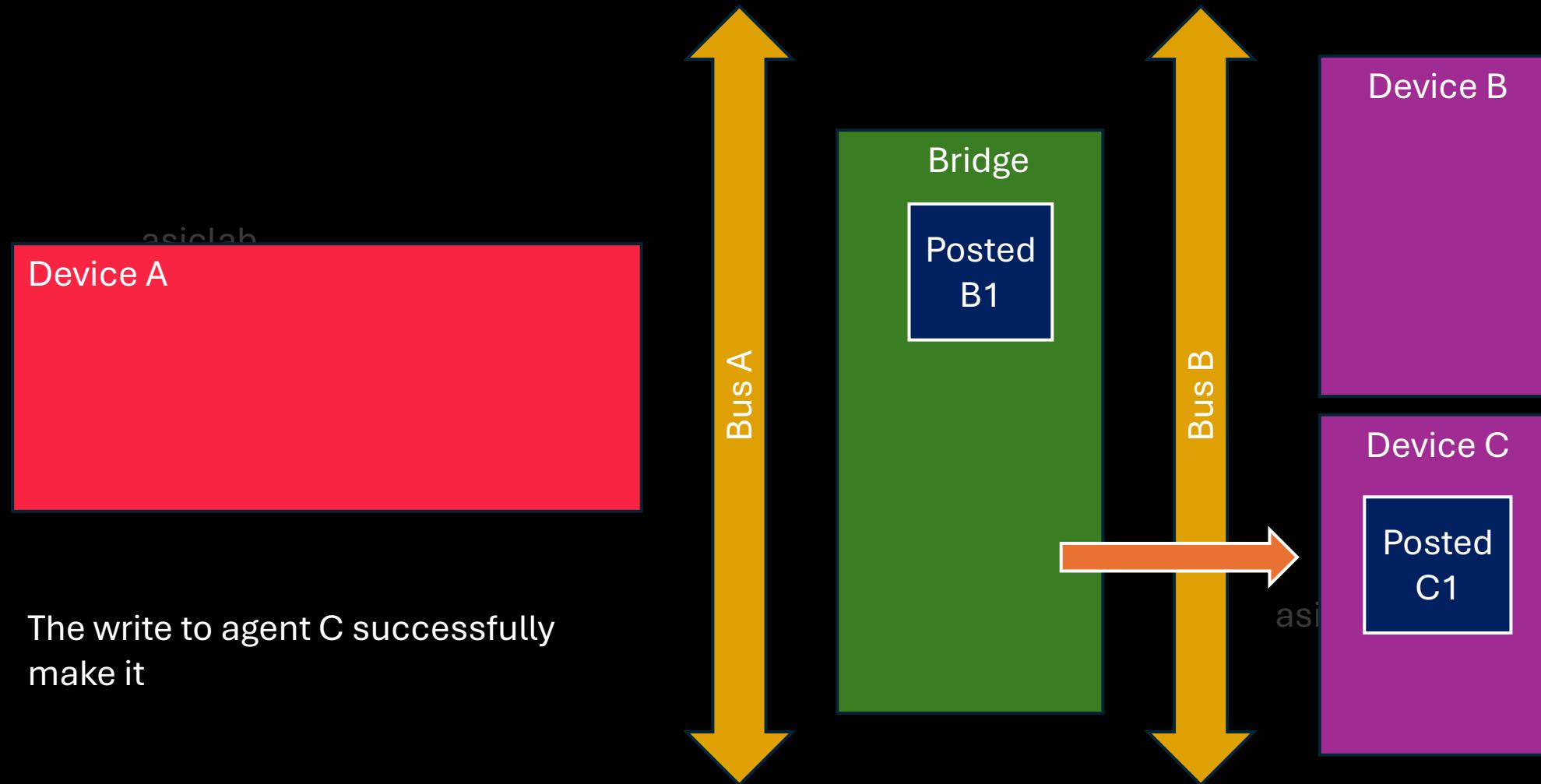
Ordering Violation Example



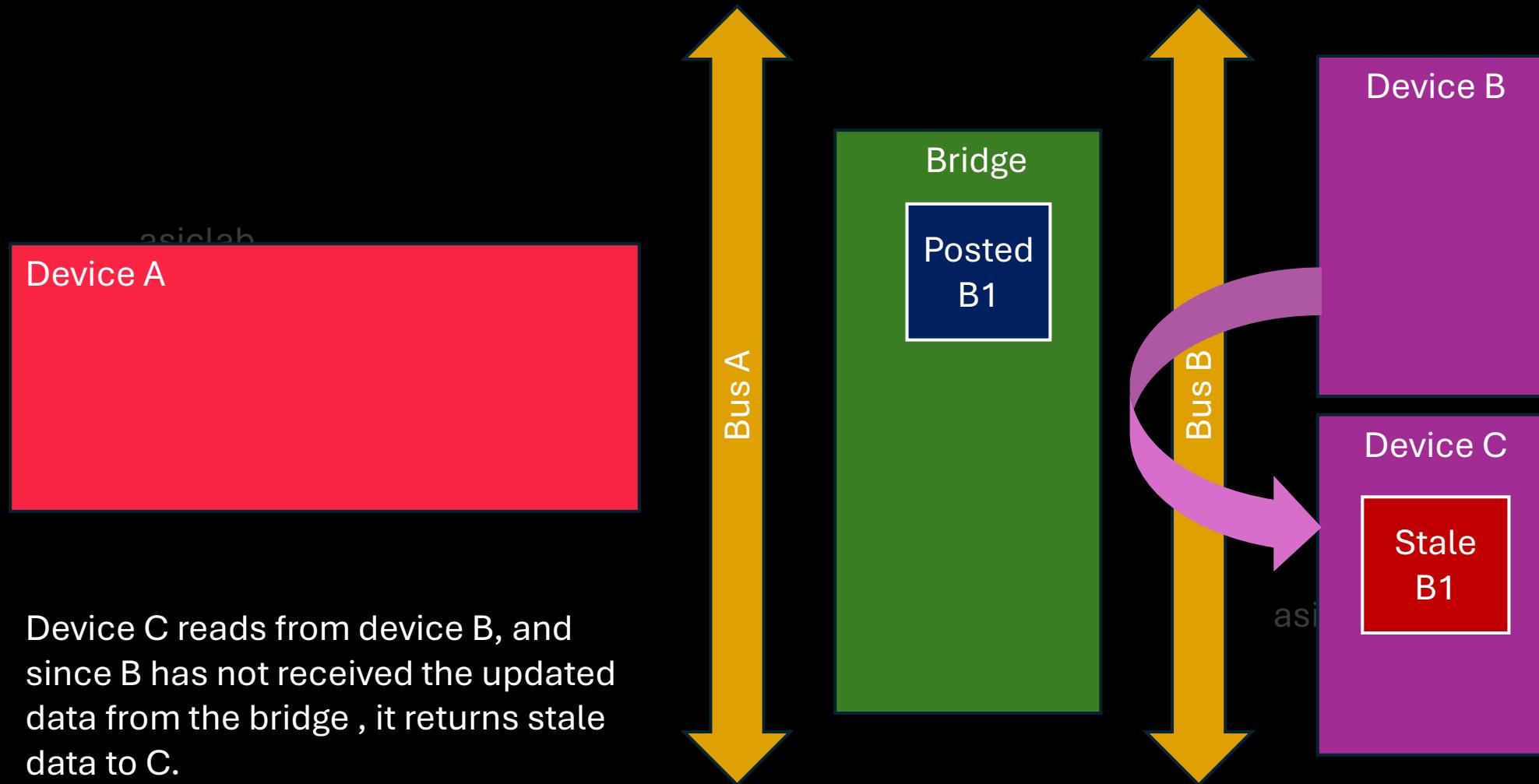
Ordering Violation Example



Ordering Violation Example



Ordering Violation Example



PCI Ordering rules – Producer Consumer

asiclab

Row Pass Column		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Column 2)	Read Request (Column 3)	I/O or Cfg Write Request (Column 4)	Read Completion (Column 5)	I/O Cfg Write Completion (Column 6)
Posted Request	Memory Write Request (Row A)	No	Yes	Yes	Y/N	Y/N
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (Row D)	No	Yes	Yes	Y/N No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

PCI Ordering rules – Dead lock avoidance

asiclab

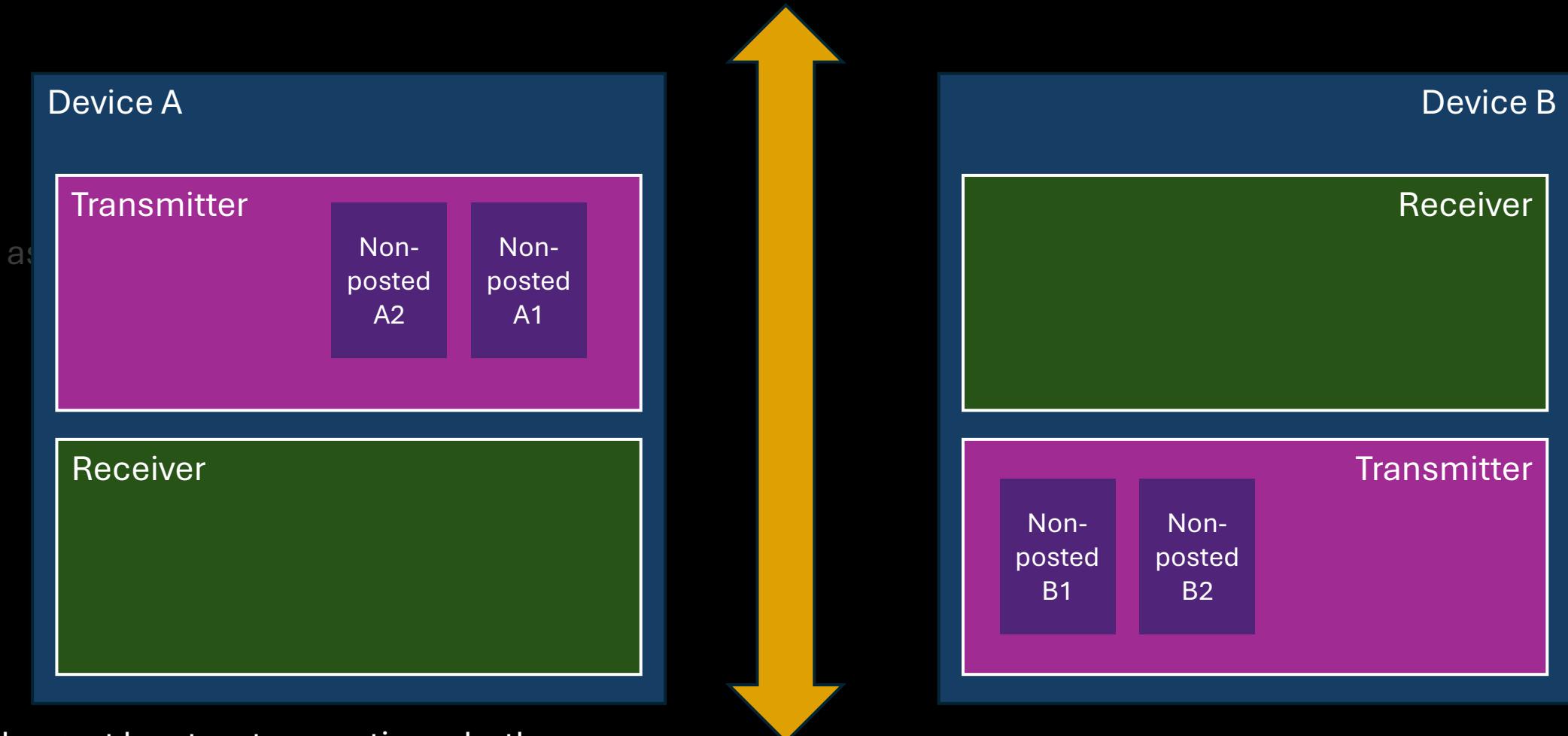
Row Pass Column		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Column 2)	Read Request (Column 3)	I/O or Cfg Write Request (Column 4)	Read Completion (Column 5)	I/O Cfg Write Completion (Column 6)
Posted Request	Memory Write Request (Row A)	No	Yes	Yes	Y/N	Y/N
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (Row D)	No	Yes	Yes	Y/N No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

Deadlock avoidance

- A deadlock occurs when two or more agents who are communicating to each other will not finish outstanding transactions until their previously issued transactions are complete
 - It takes two agents to violate this rule to create the deadlock
 - If only one agent violates, the deadlock will not occur
- As a result, strict rules have been established
 - “If you cannot do a read or non-posted write, you must allow posted and completion cycles behind the read/non-posted write to pass”.
- Example:
 - An agent does not want to generate a completion to another agent who had given it a request, until it sees that its outstanding request is serviced
 - If both agents violate, you will get a deadlock

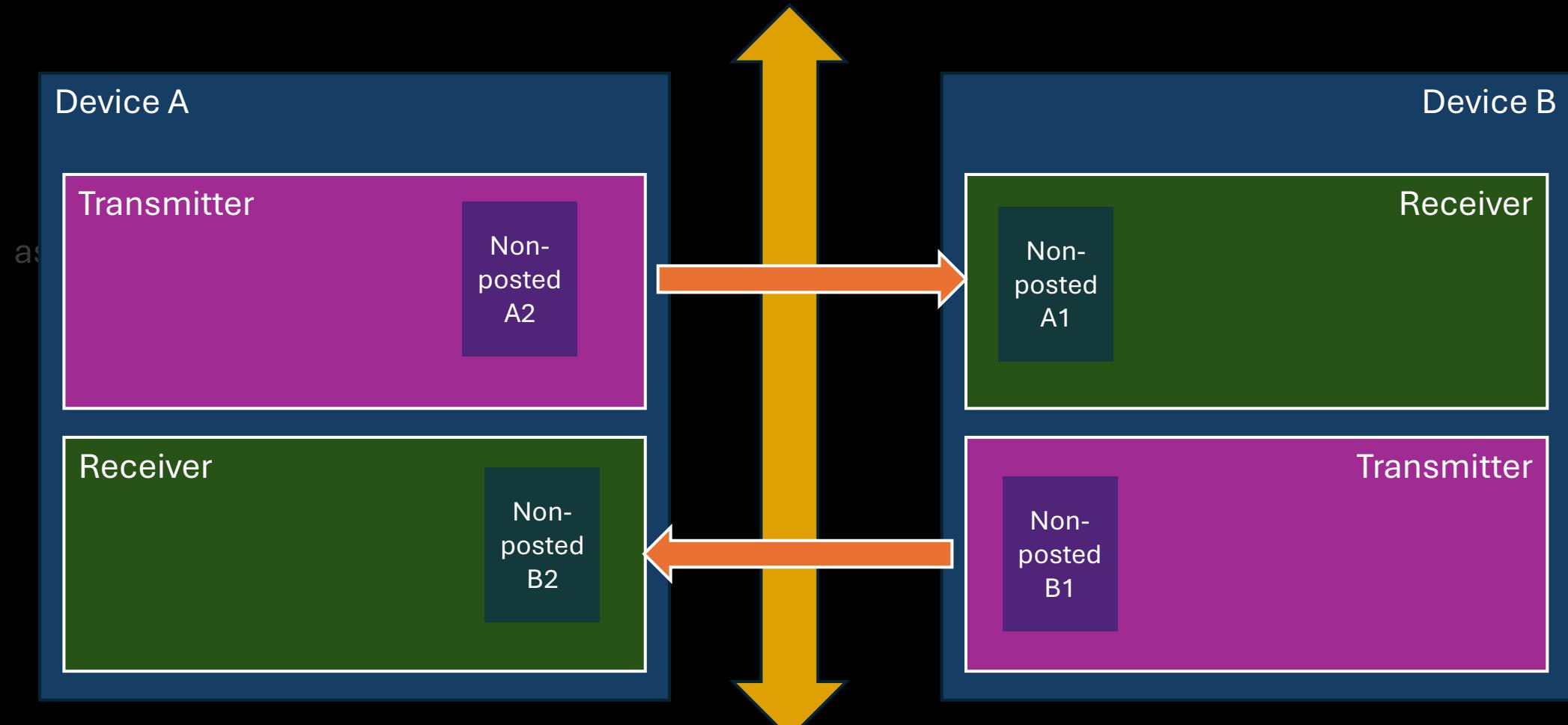
asiclab

Deadlock Example

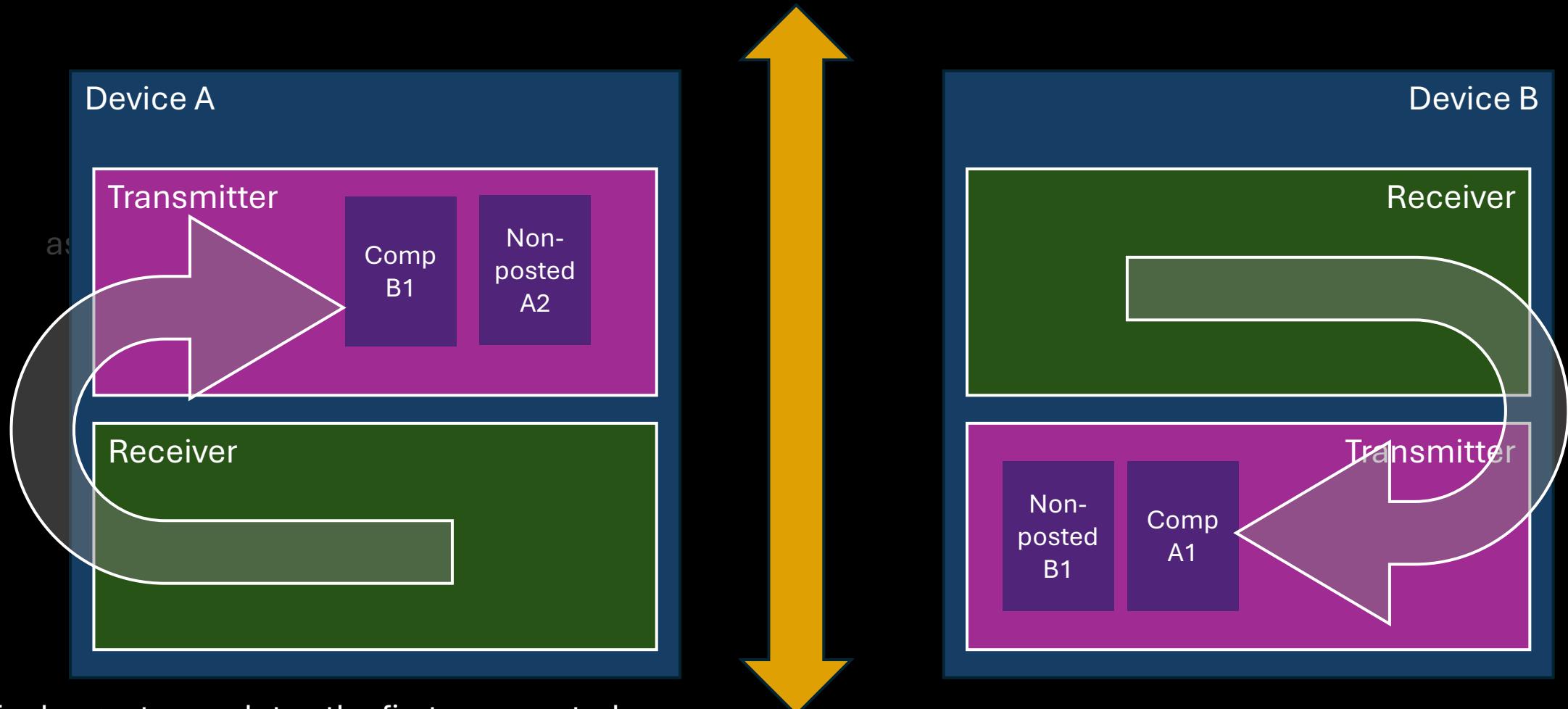


Each agent has two transactions, both transactions directly targeting the other agent.

Deadlock Example



Deadlock Example

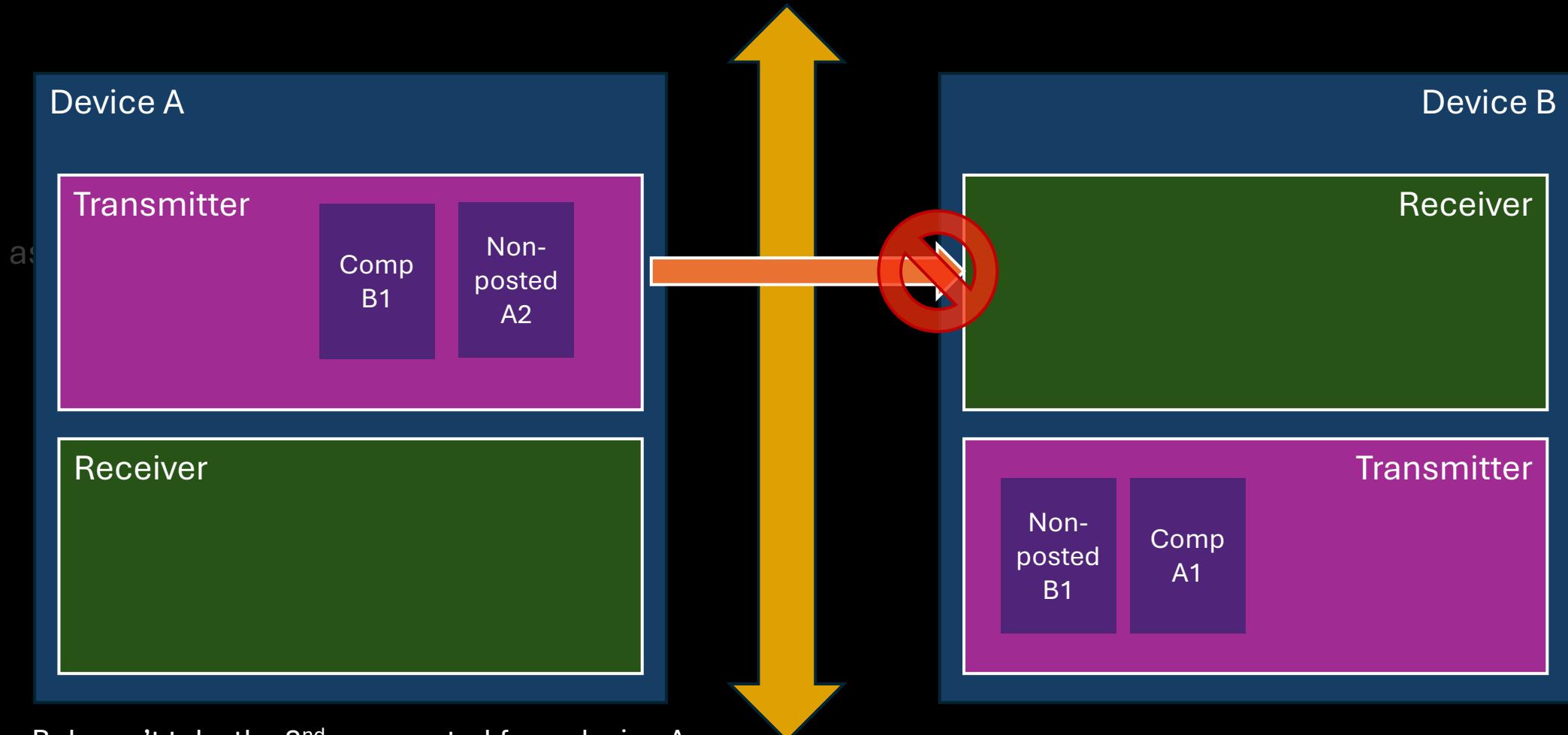


Each agent completes the first non-posted transaction and places the completion for that transaction into its transmitter, behind its 2nd non-posted transaction.

DO NOT DISTRIBUTE

© Copyright protected 2024

Deadlock Example

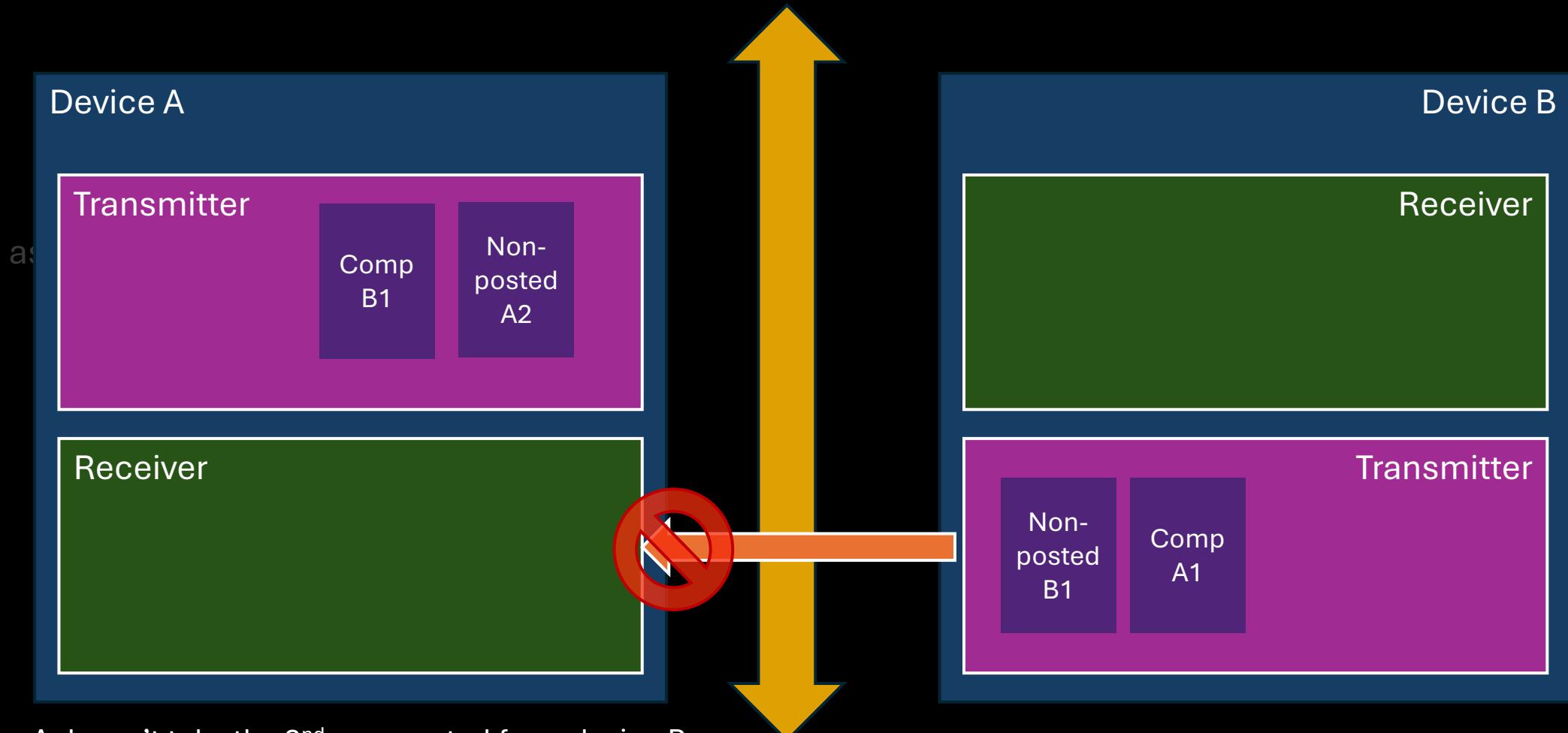


Device B doesn't take the 2nd non-posted from device A, because it is waiting for the completion for cycle B1.

In PCI, this would be a retry... In PCI-Express this would be

device B not giving a credit update for non-posted transactions.

Deadlock Example



Device A doesn't take the 2nd non-posted from device B, because it is waiting for the completion for cycle A1.

In PCI, this would be a retry... In PCI-Express this would be device A not giving a credit update for non-posted transactions.

Classic deadlock – both agents are not accepting further transactions, as they are awaiting completions from other transactions they have previously issued

PCI Ordering rules – Dead lock avoidance

asiclab

Row Pass Column		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Column 2)	Read Request (Column 3)	I/O or Cfg Write Request (Column 4)	Read Completion (Column 5)	I/O Cfg Write Completion (Column 6)
Posted Request	Memory Write Request (Row A)	No	Yes	Yes	Y/N	Y/N
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (Row D)	No	Yes	Yes	Y/N No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

PCI Ordering rules – Completion association

asiclab

Row Pass Column		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Column 2)	Read Request (Column 3)	I/O or Cfg Write Request (Column 4)	Read Completion (Column 5)	I/O Cfg Write Completion (Column 6)
Posted Request	Memory Write Request (Row A)	No	Yes	Yes	Y/N	Y/N
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (Row D)	No	Yes	Yes	Y/N No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

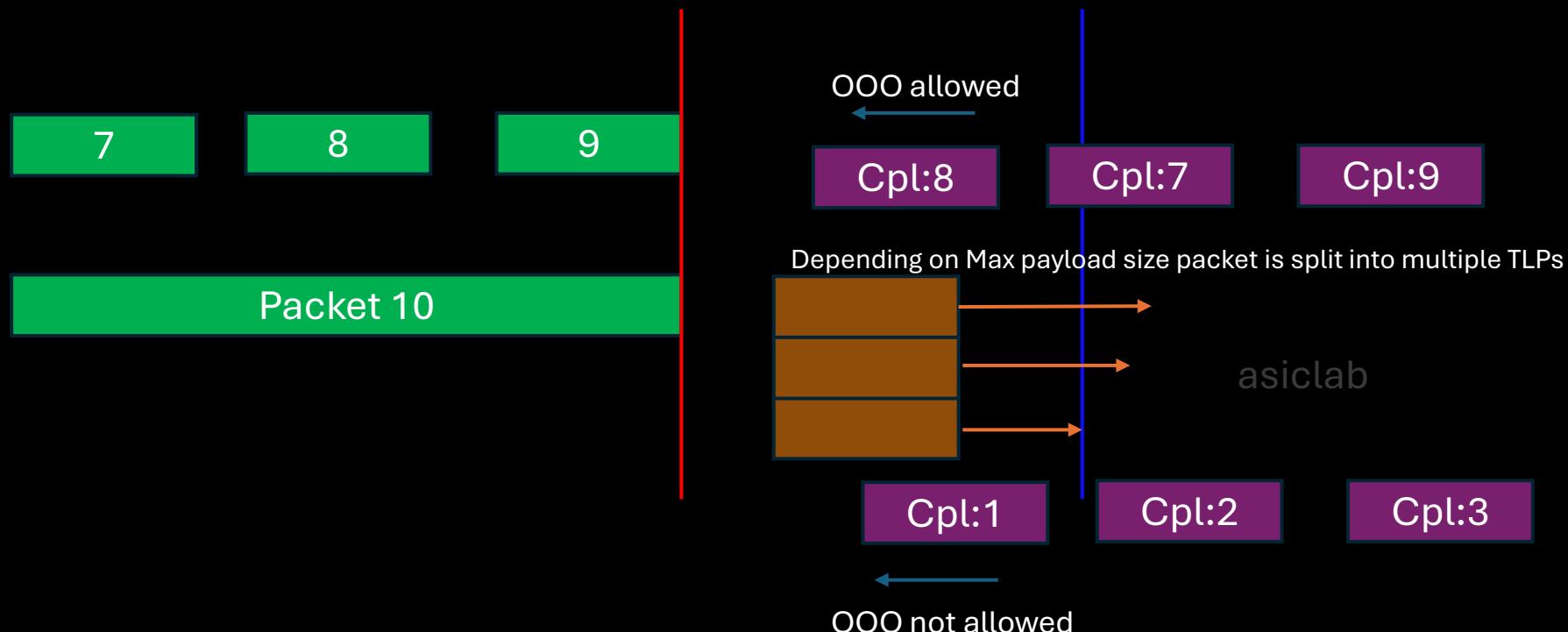
Completion association

- Completions for different requests may return OOO and are associated by transaction ID
- Multiple completions returned for single request use the same single transaction ID so they must be returned in order.

asiclab

Completion association

- Completions for different requests may return OOO and are associated by transaction ID
- Multiple completions returned for single request use the same single transaction ID so they must be returned in order.



PCI Ordering rules – Completion association

asiclab

Row Pass Column		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Column 2)	Read Request (Column 3)	I/O or Cfg Write Request (Column 4)	Read Completion (Column 5)	I/O Cfg Write Completion (Column 6)
Posted Request	Memory Write Request (Row A)	No	Yes	Yes	Y/N	Y/N
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (Row D)	No	Yes	Yes	Y/N No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

Modified Ordering Attributes

- Per transaction – Relaxed ordering hint may be used to say data ordering is not needed
- Per stream – Virtual Channels
 - asiclab Traffic Class (TC) labeling for traffic differentiations – 8 TCs
 - TCs are mapped to Virtual Channels (VCs)
 - Ordering is maintained within VCs, OOO between VCs
- Virtual Channel usage
 - Support QoS
 - Avoid deadlocks in special cases in which there is dependency between two types of streams

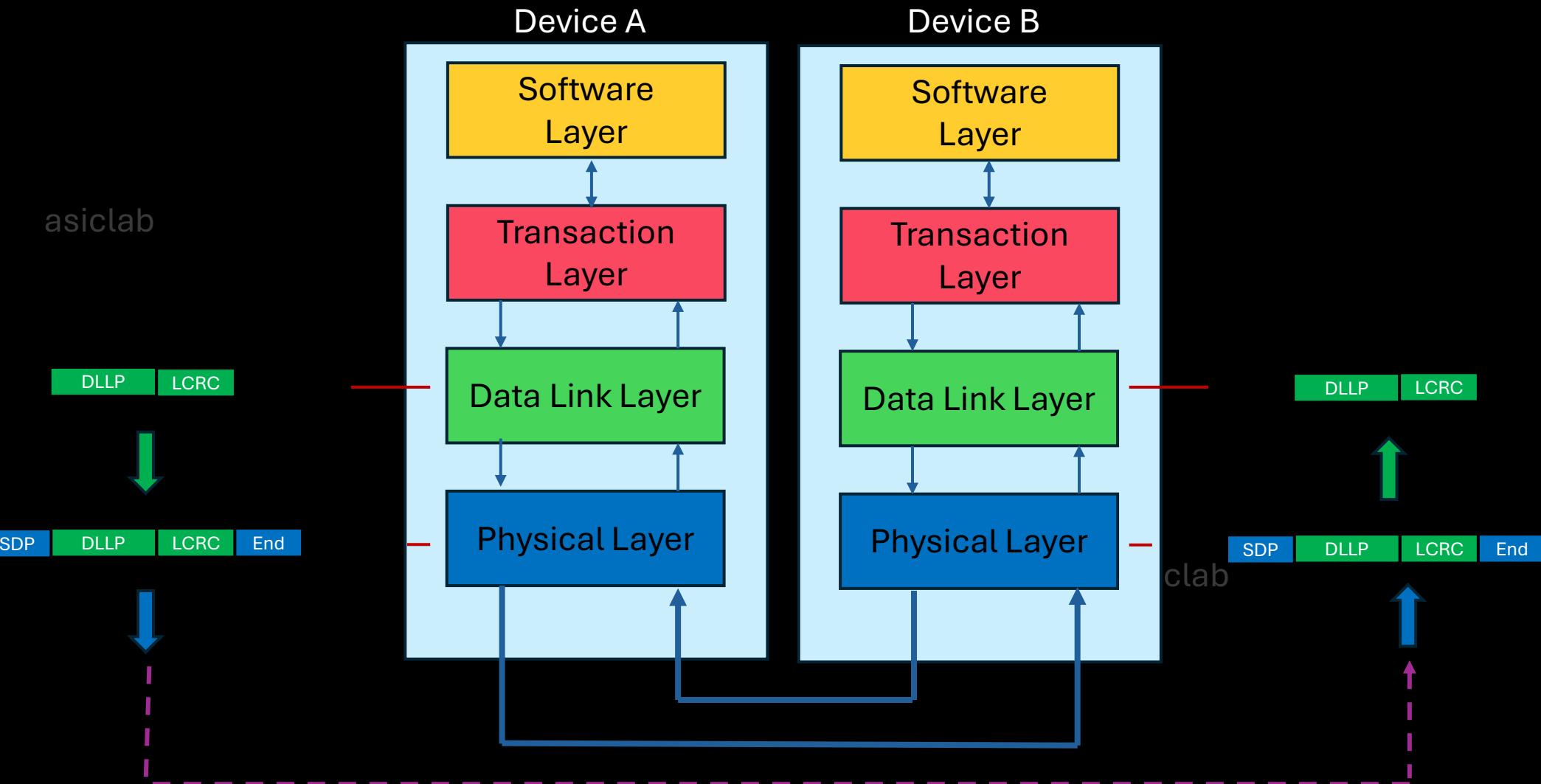
asiclab

asiclab

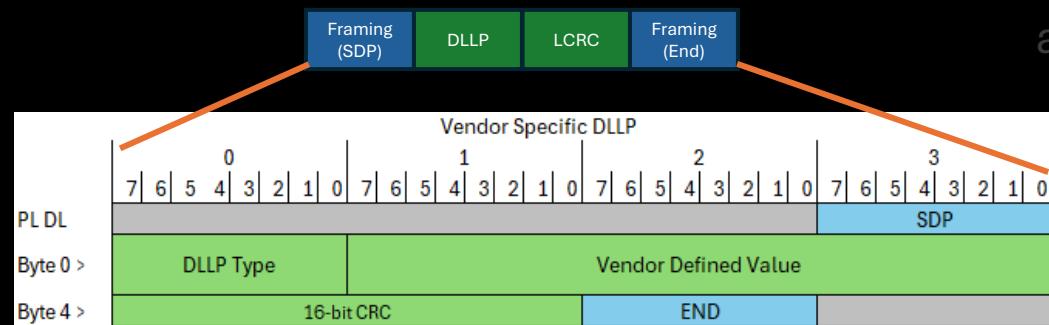
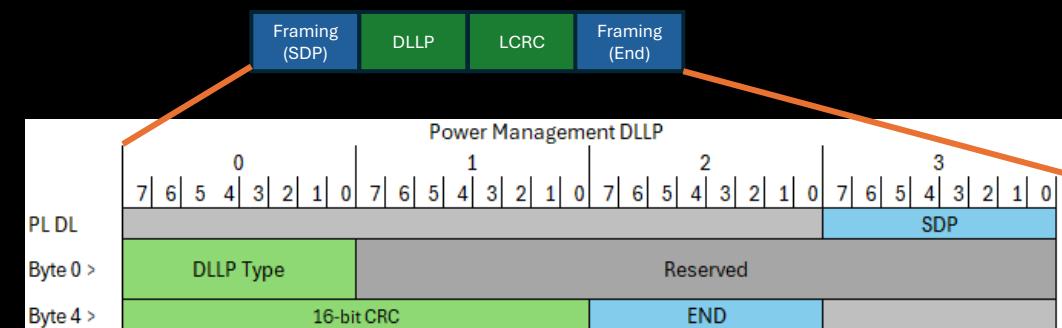
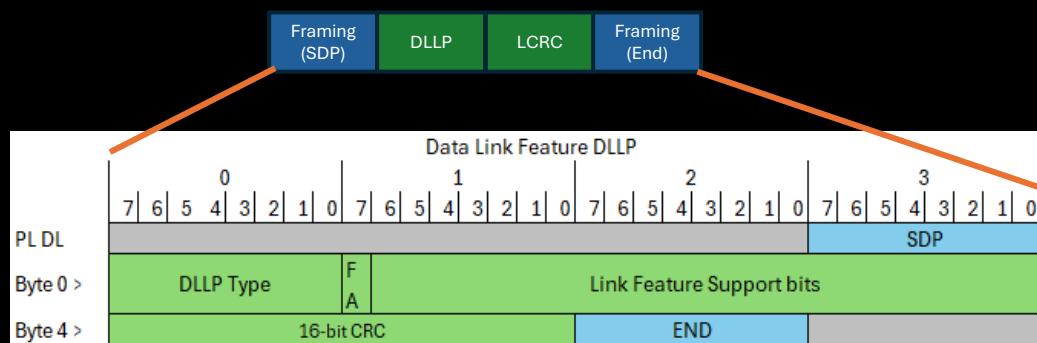
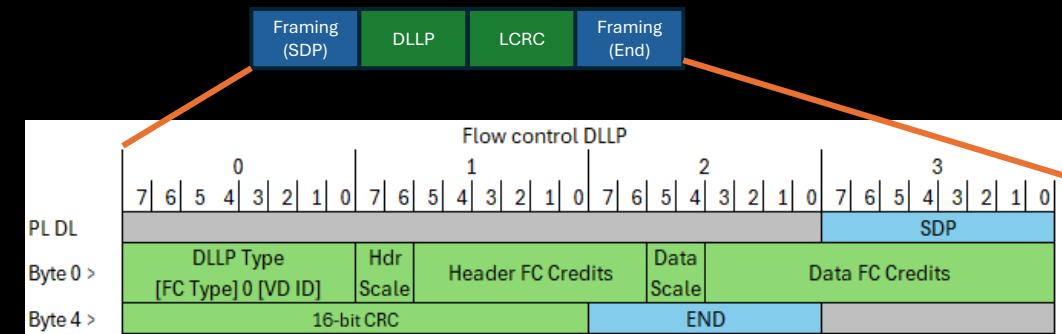
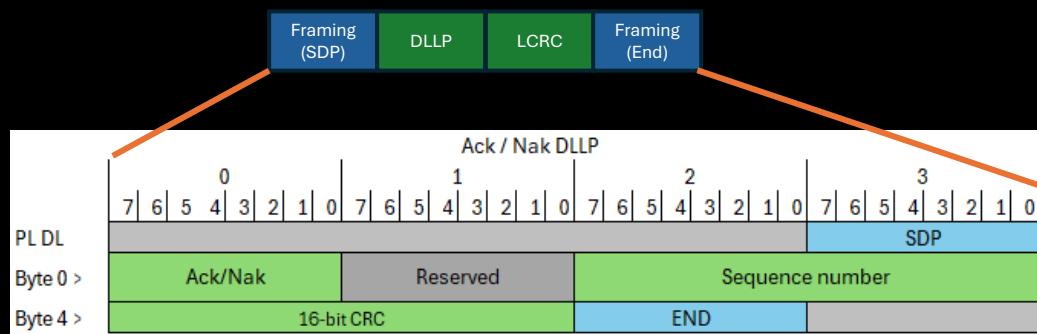
Data Link Layer

asiclab

Data Link Layer Packets(DLLP) (8b/10b) (128b/130b)



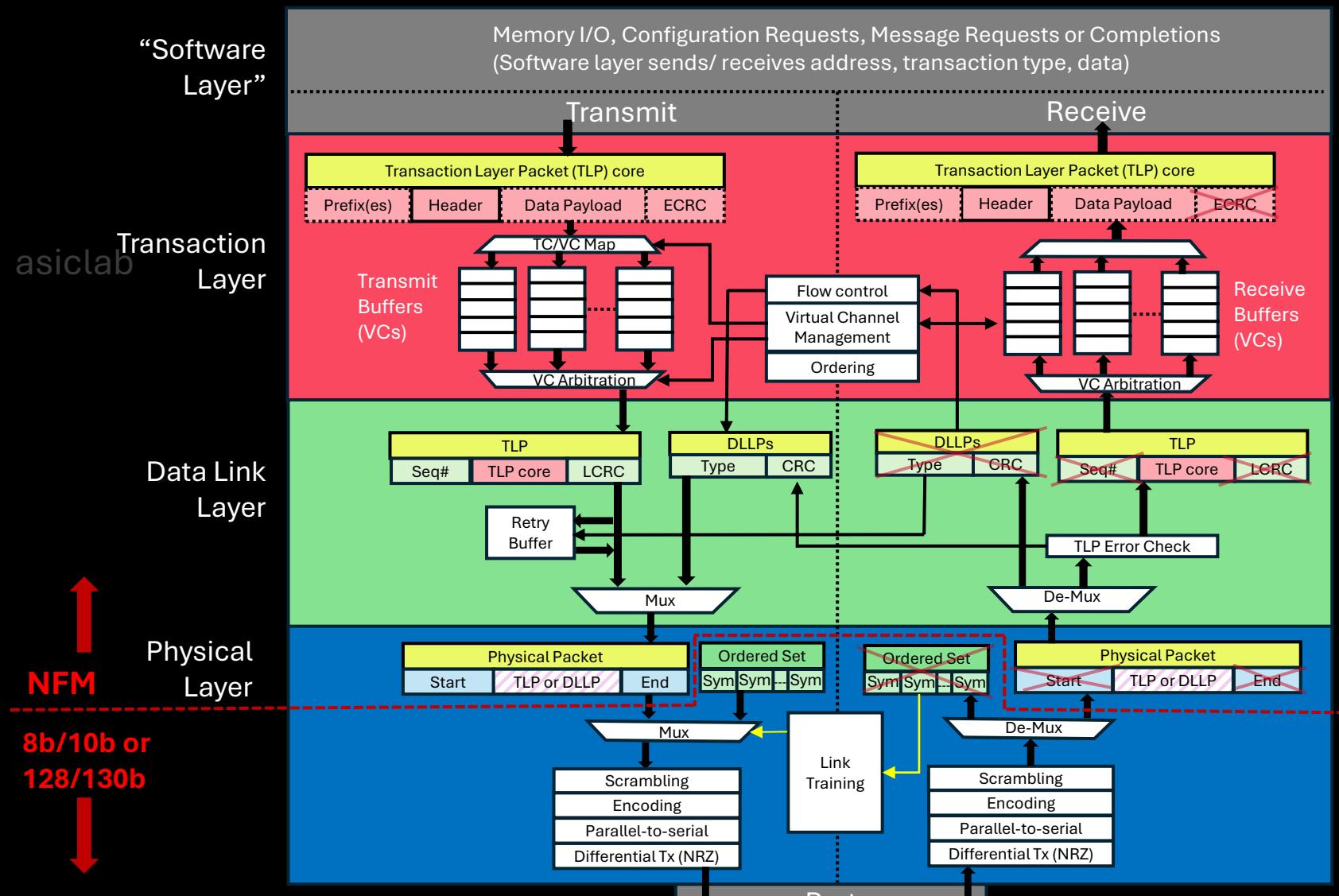
DLL Packet format



Data Link Layer Header
 Physical Layer Header
 DO NOT Distribute

asiclab

PCIe Device Layer Details 5.0 Non Flit Mode (NFM)

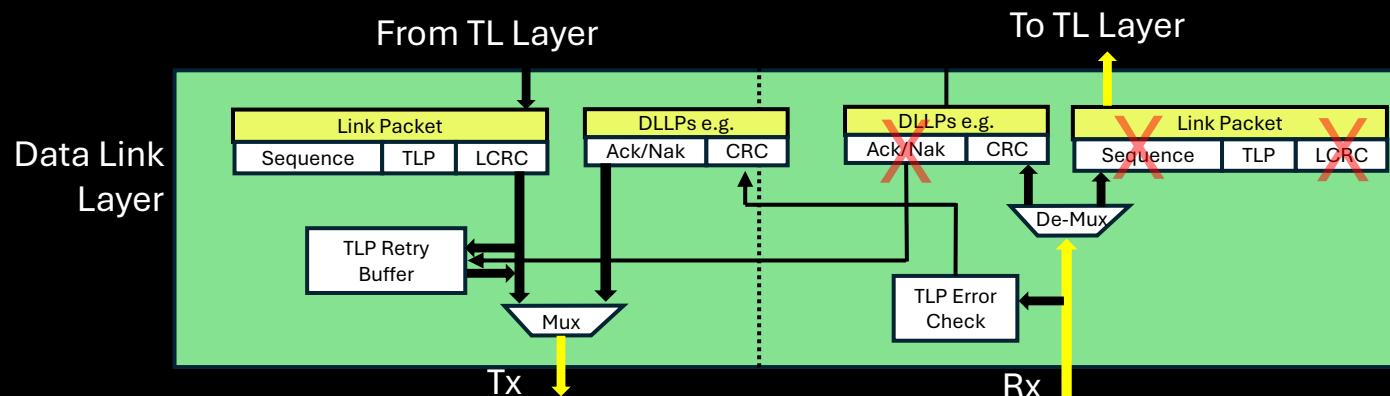


Do Not Distribute

© Copyright protected 2024
Link

Data Link Layer Replay Mechanism

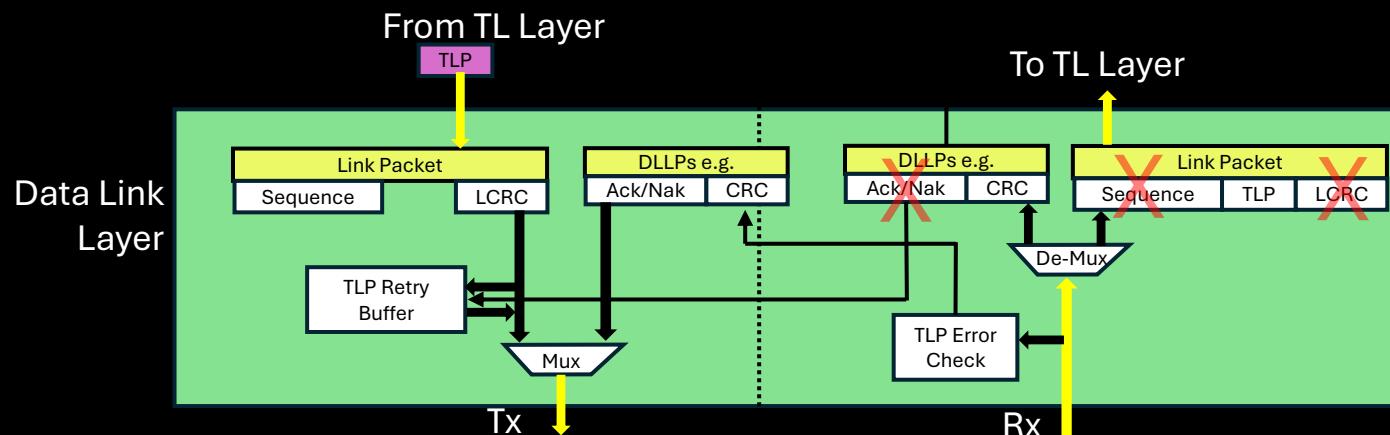
- A copy of each outgoing TLP is stored in the Retry Buffer until the neighbor acknowledges receipt
- Incoming TLPs are checked and an Ack or Nak is generated to acknowledge them



Basic Operation – Step 1

- First, TLP arrives from Transaction Layer. Sequence Number and LCRC are added to it.

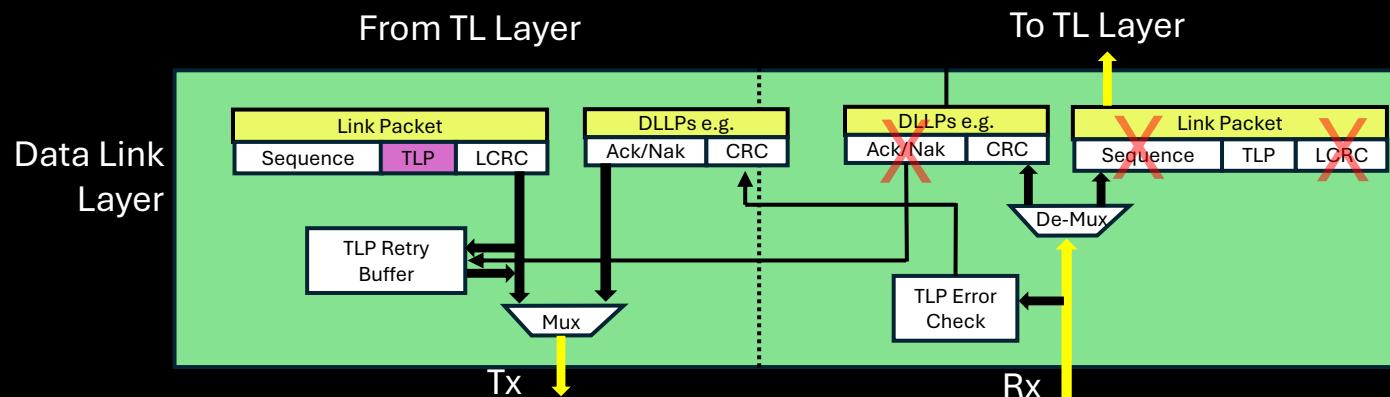
asiclab



Basic Operation – Step 2

- Second, a copy of TLP is stored in Replay Buffer until its safe arrival is acknowledged by the receiver

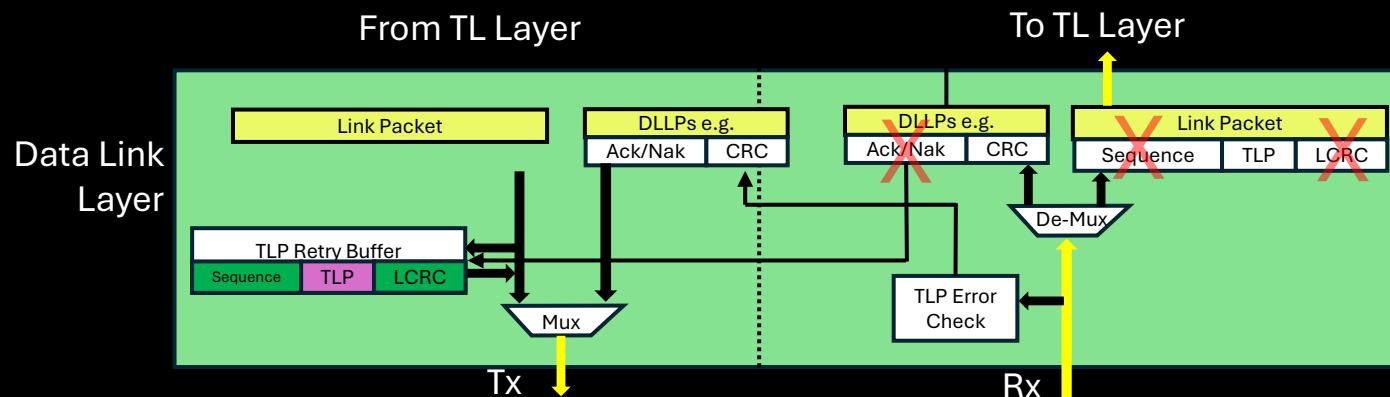
asiclab



Basic Operation – Step 3

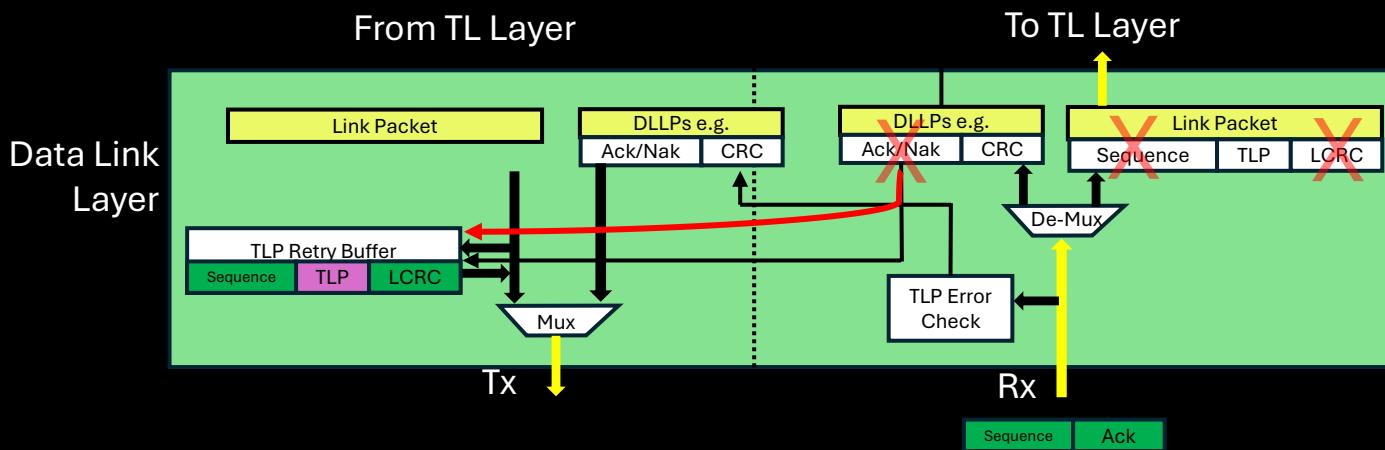
- Third, the TLP is forwarded to Physical Layer for transmission

asiclab

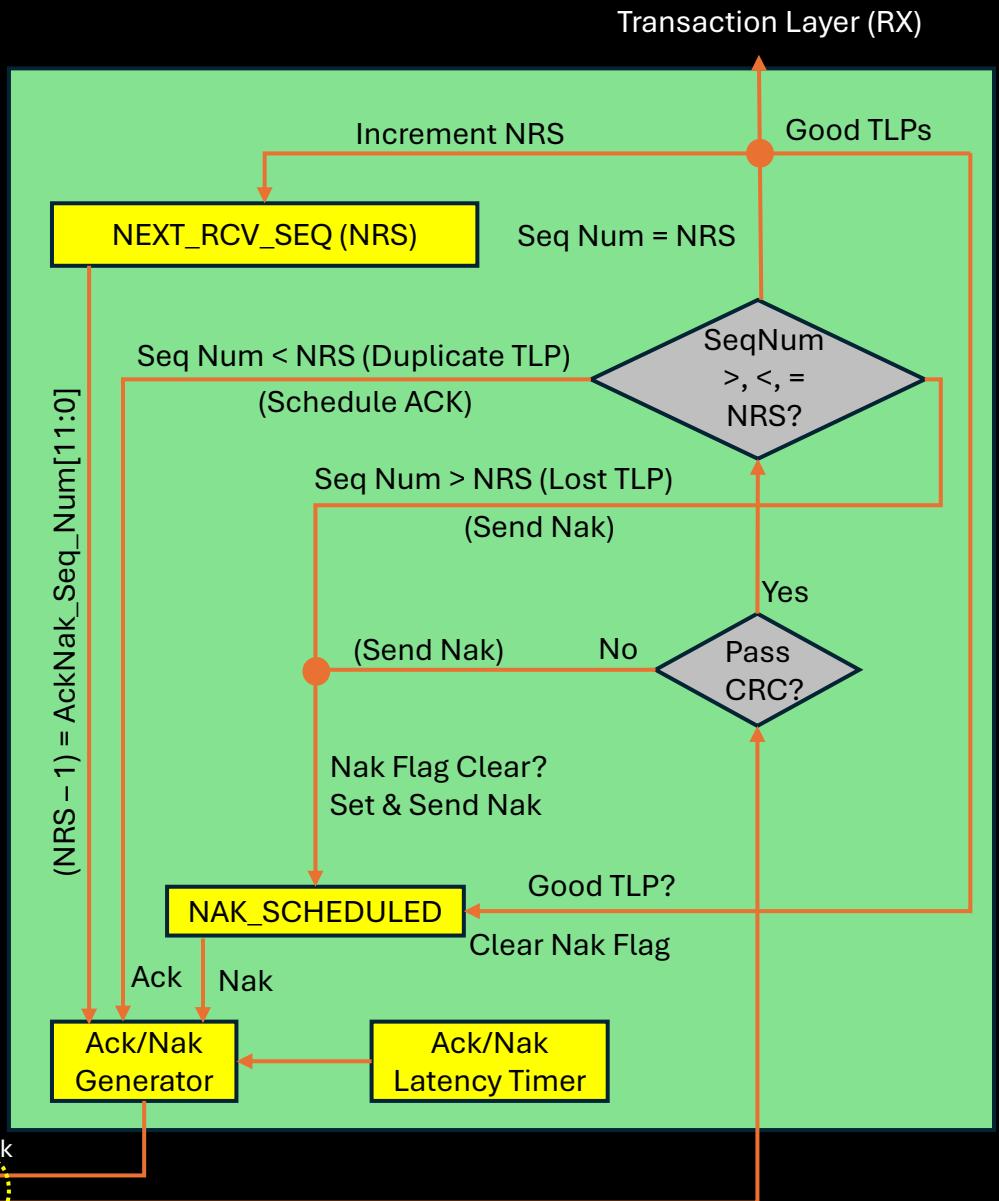
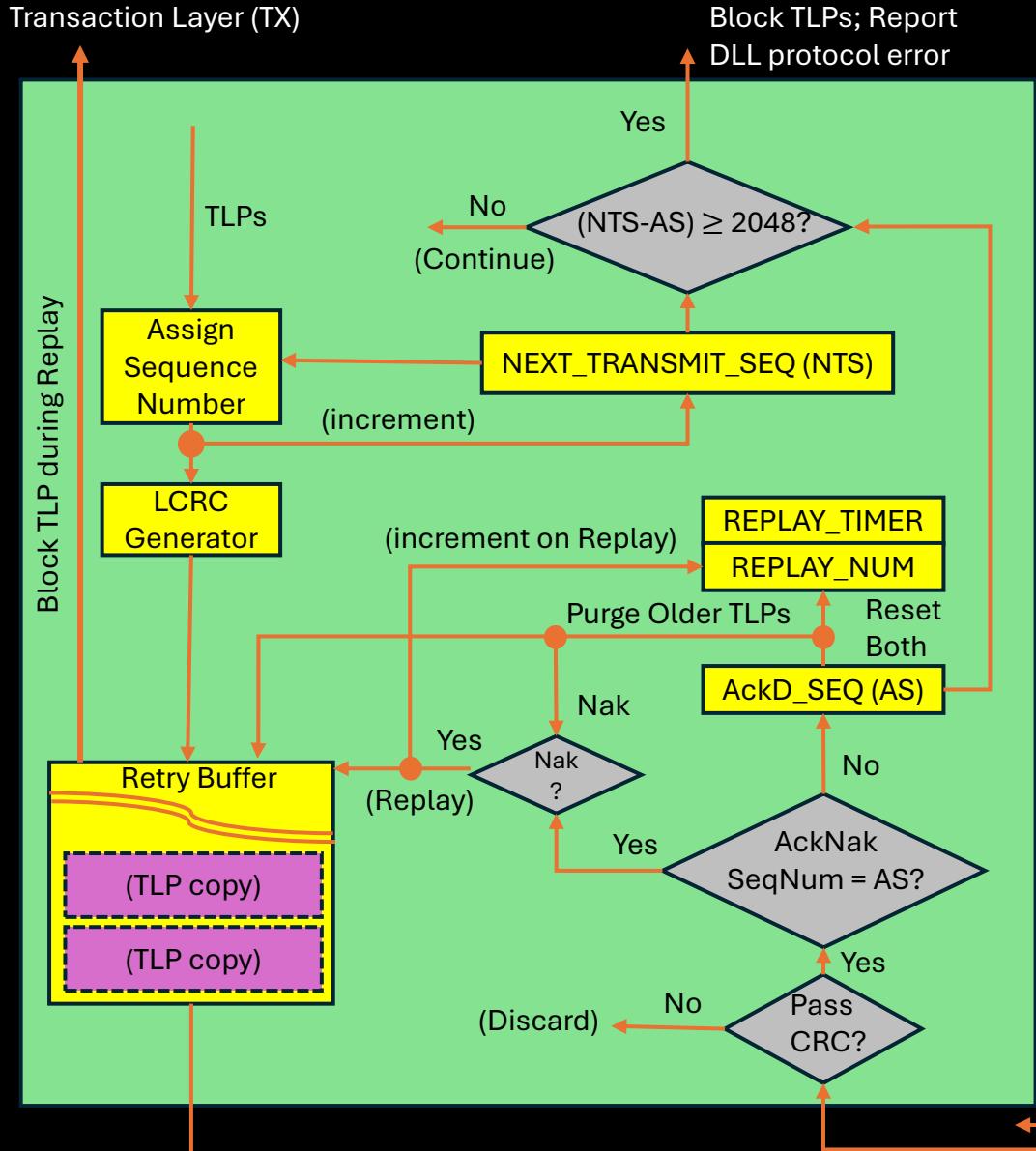


Basic Operation – Step 4

- Later, an Ack or Nak is received with a sequence number equal to or higher than this one, meaning this TLP can be removed from the buffer.
- If a Nak was received, the TLP might need to be sent again instead of being cleared out.



Elements of Ack/Nak Protocol



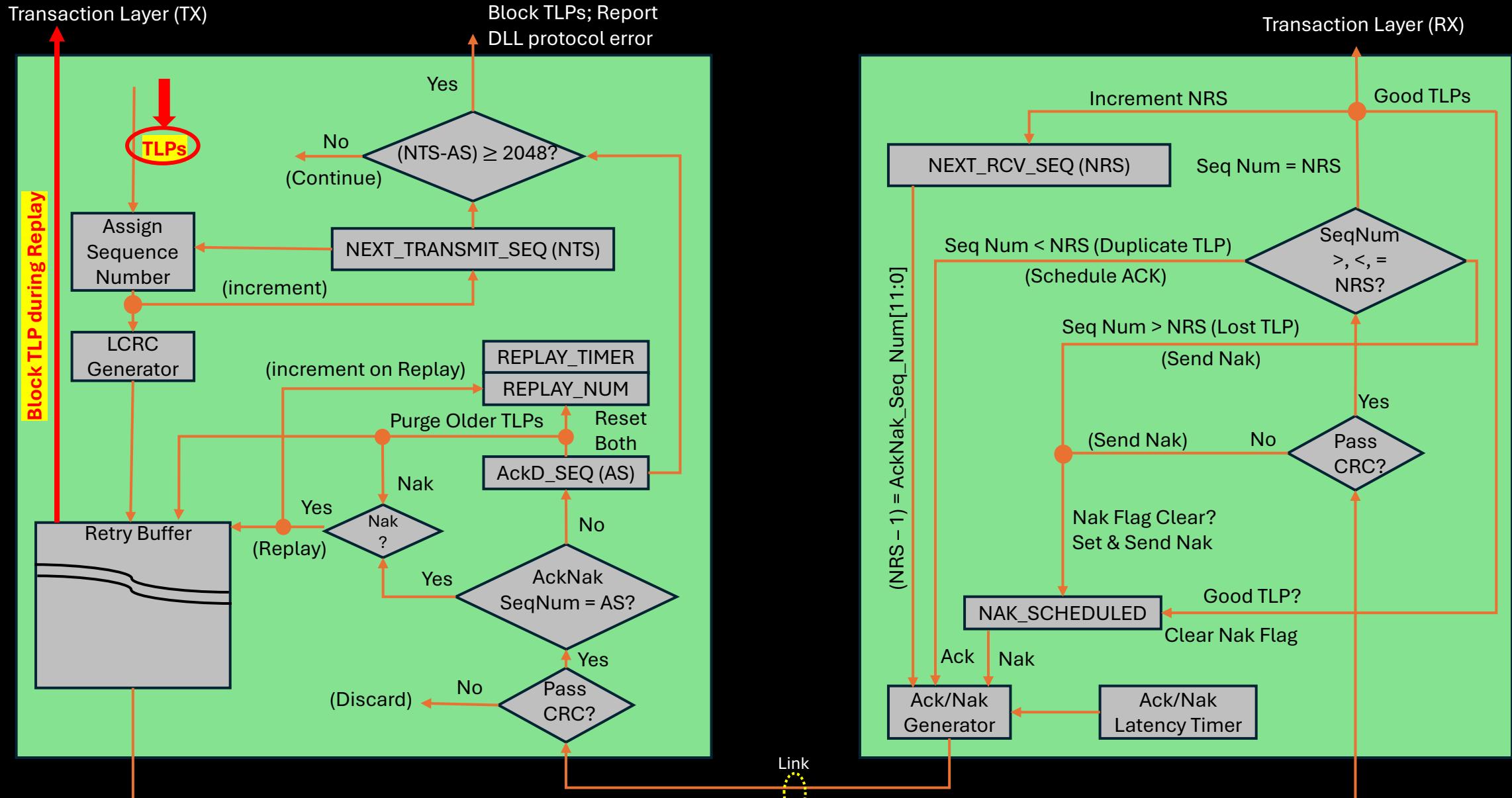
Do Not Distribute

TLP

TLP →

© Copyright protected 2024

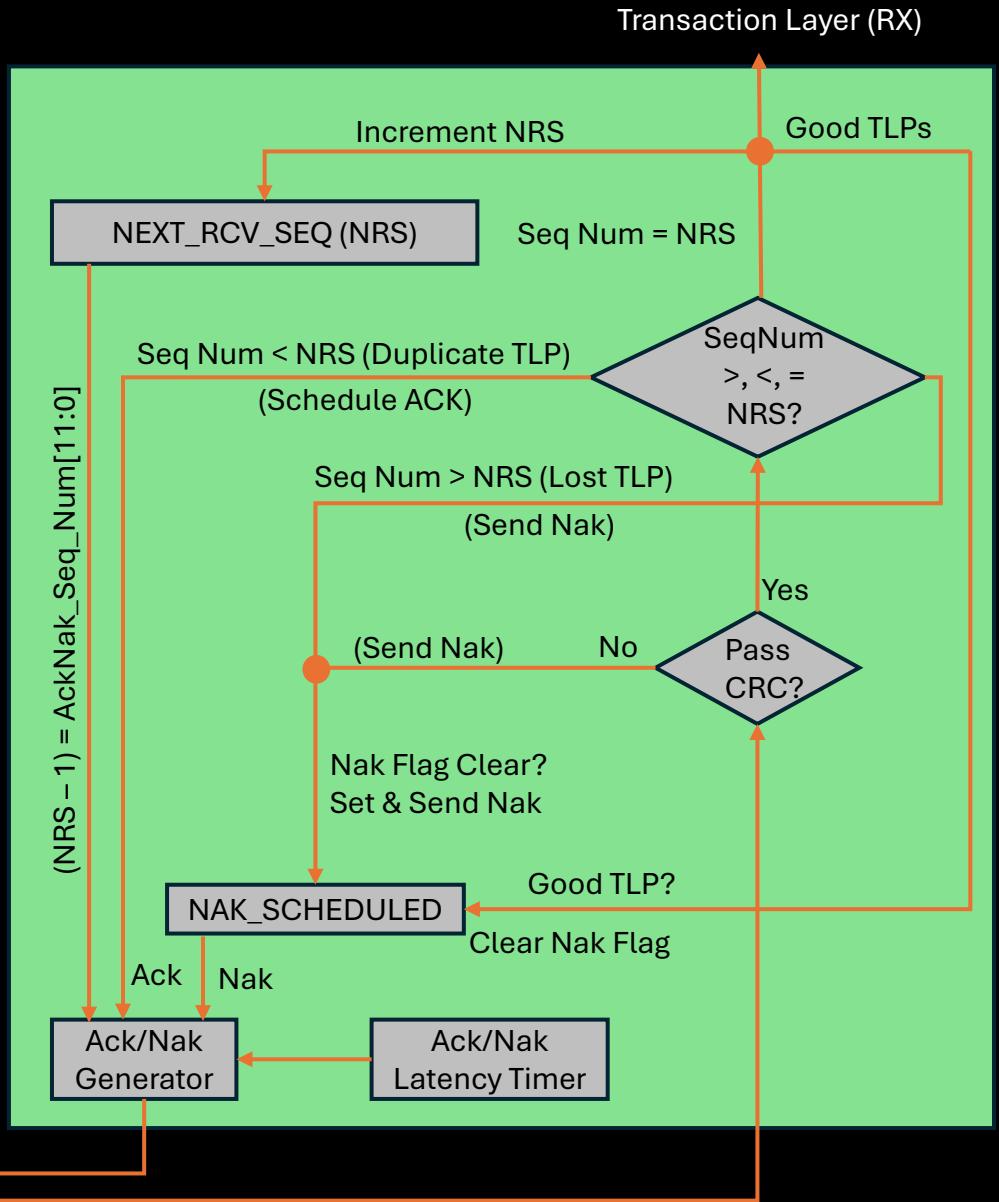
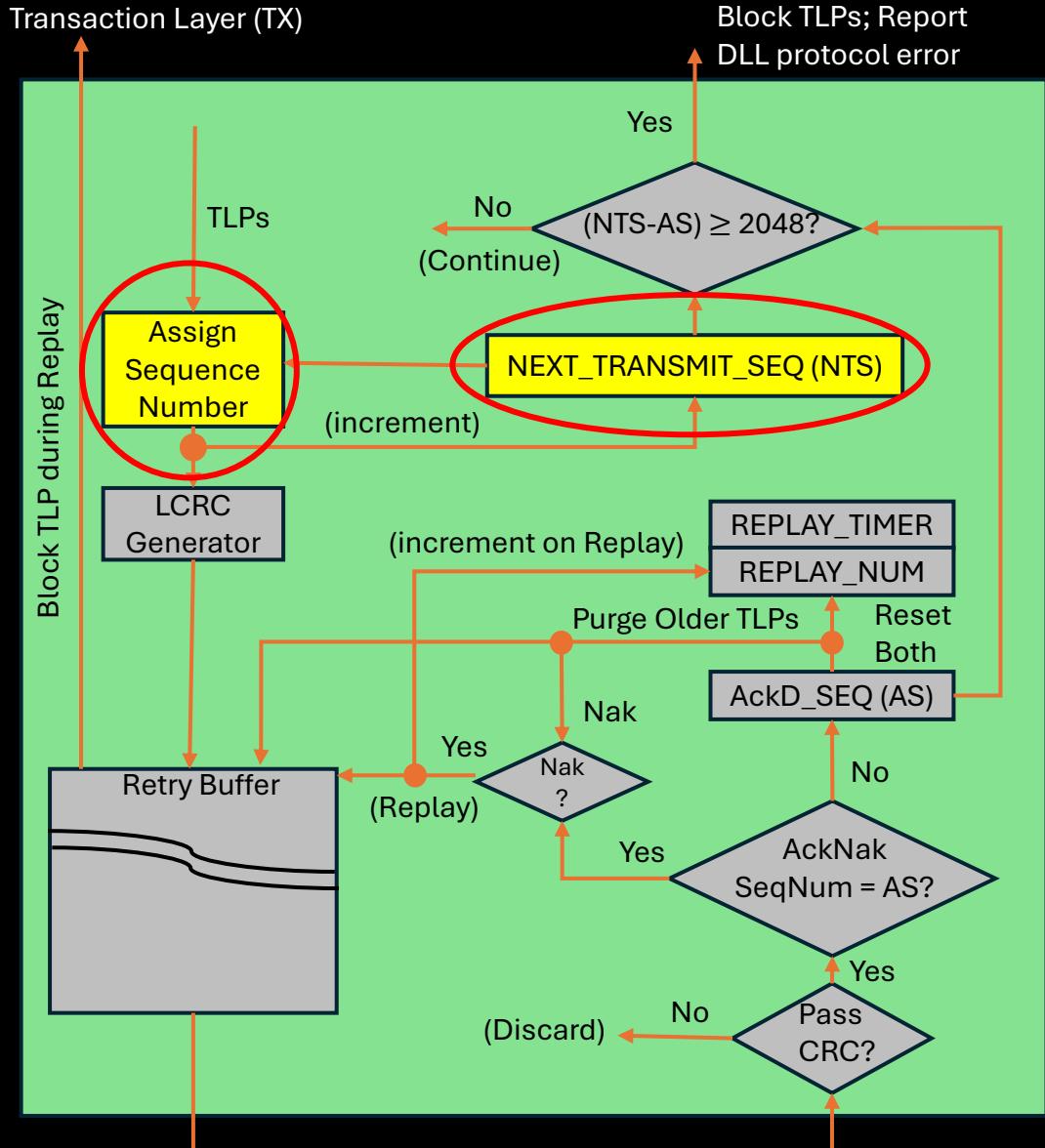
TLPs Arrive at Link Layer



Do Not Distribute

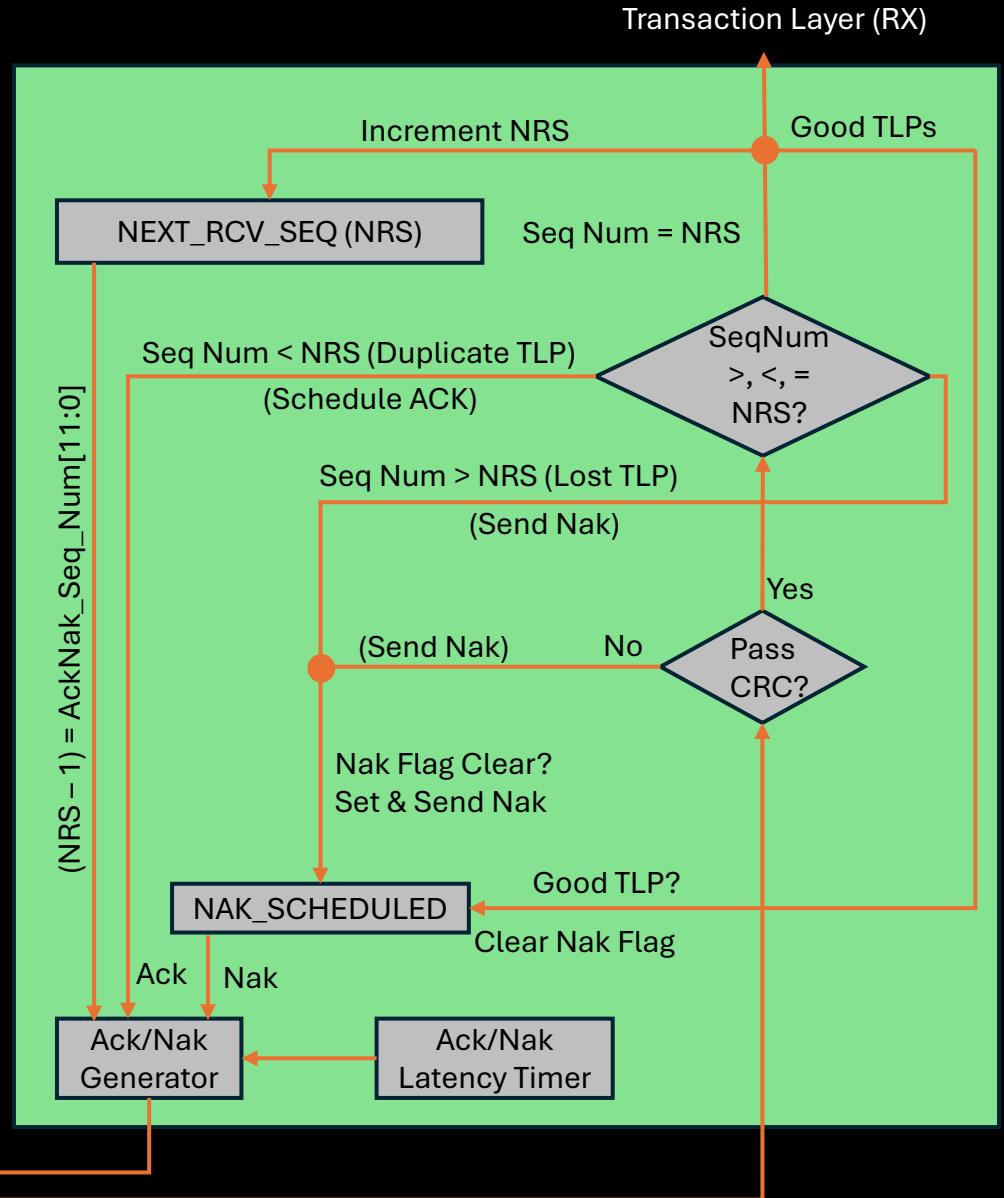
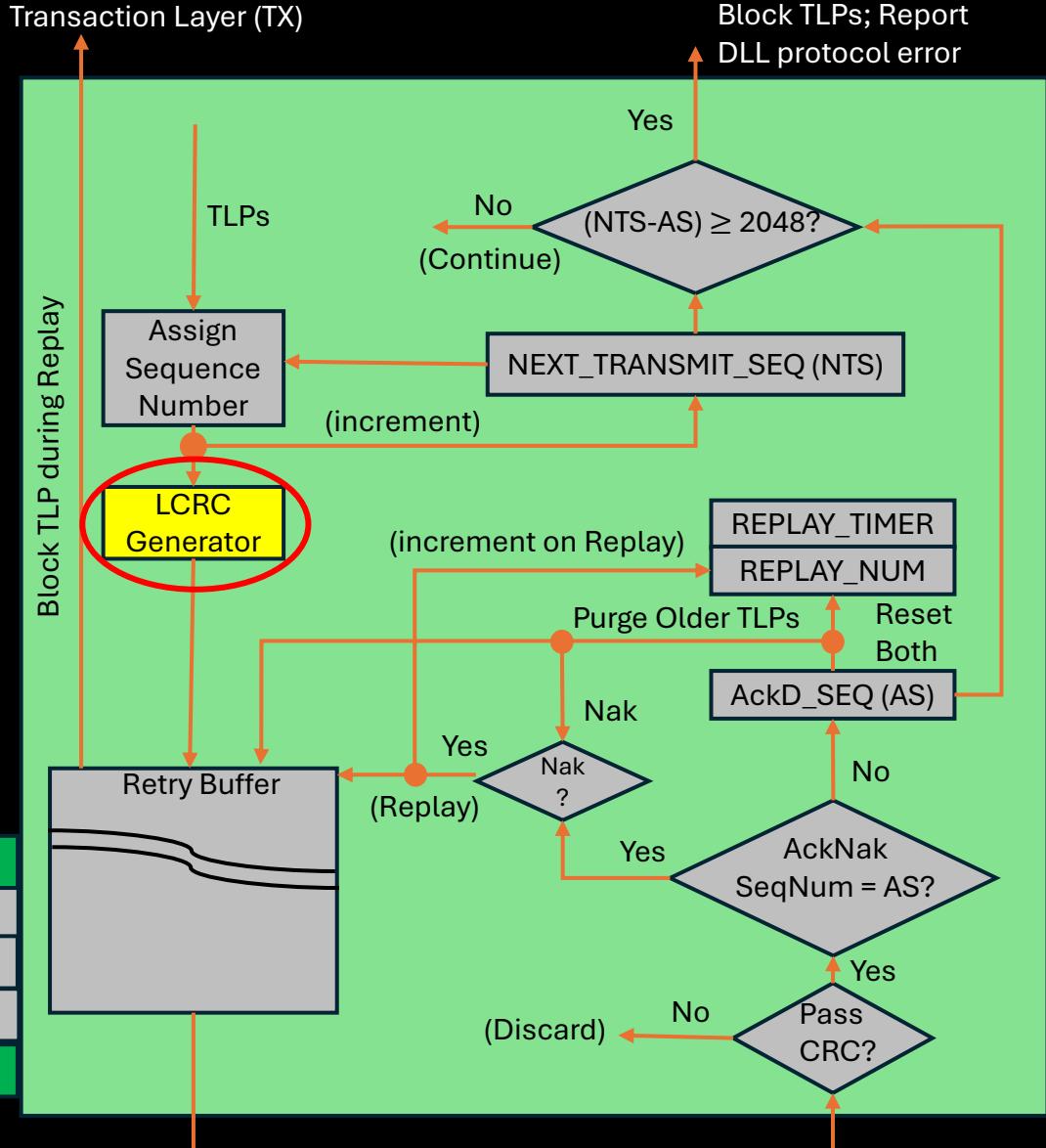
© Copyright protected 2024

Sequence Number Assigned

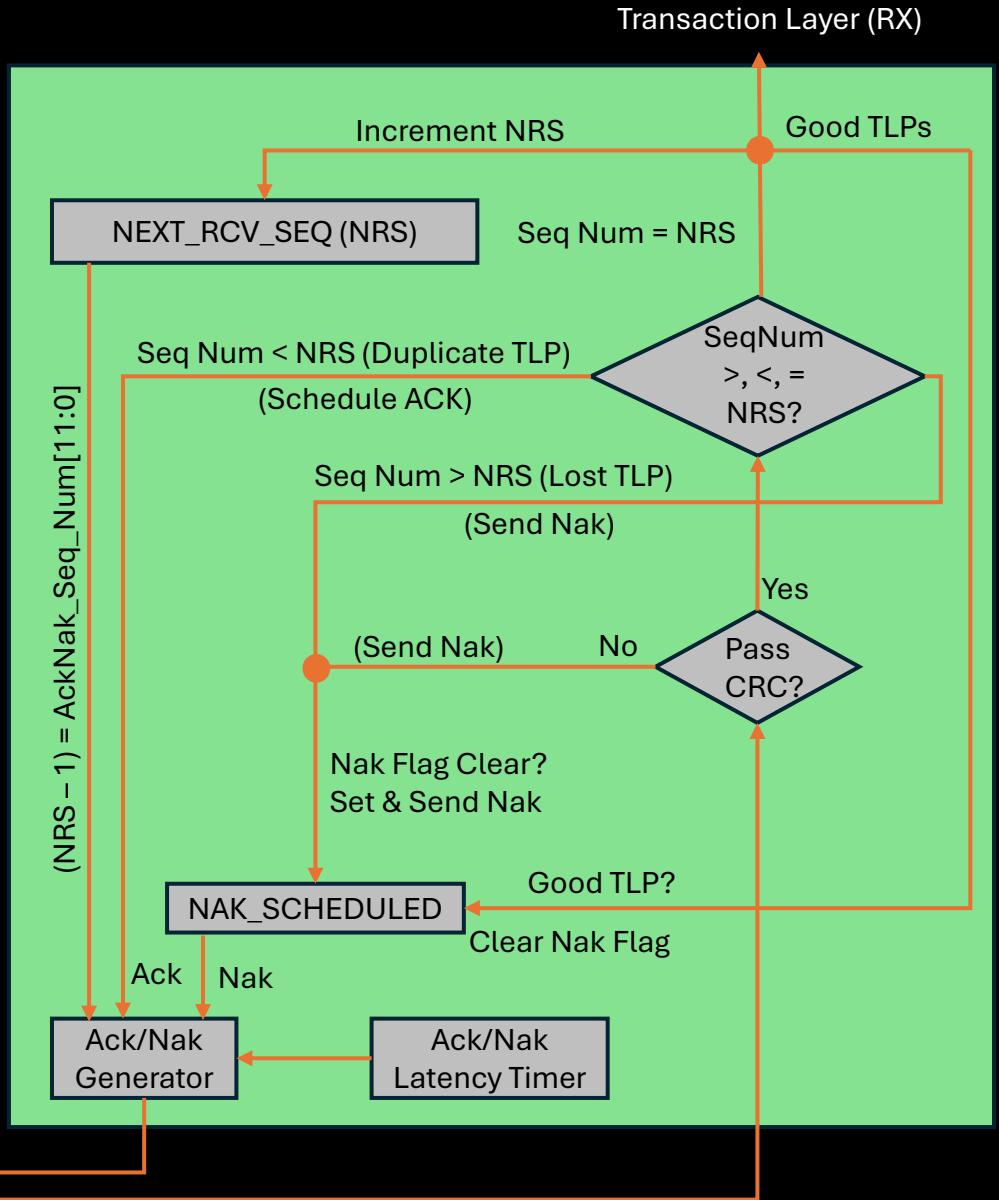
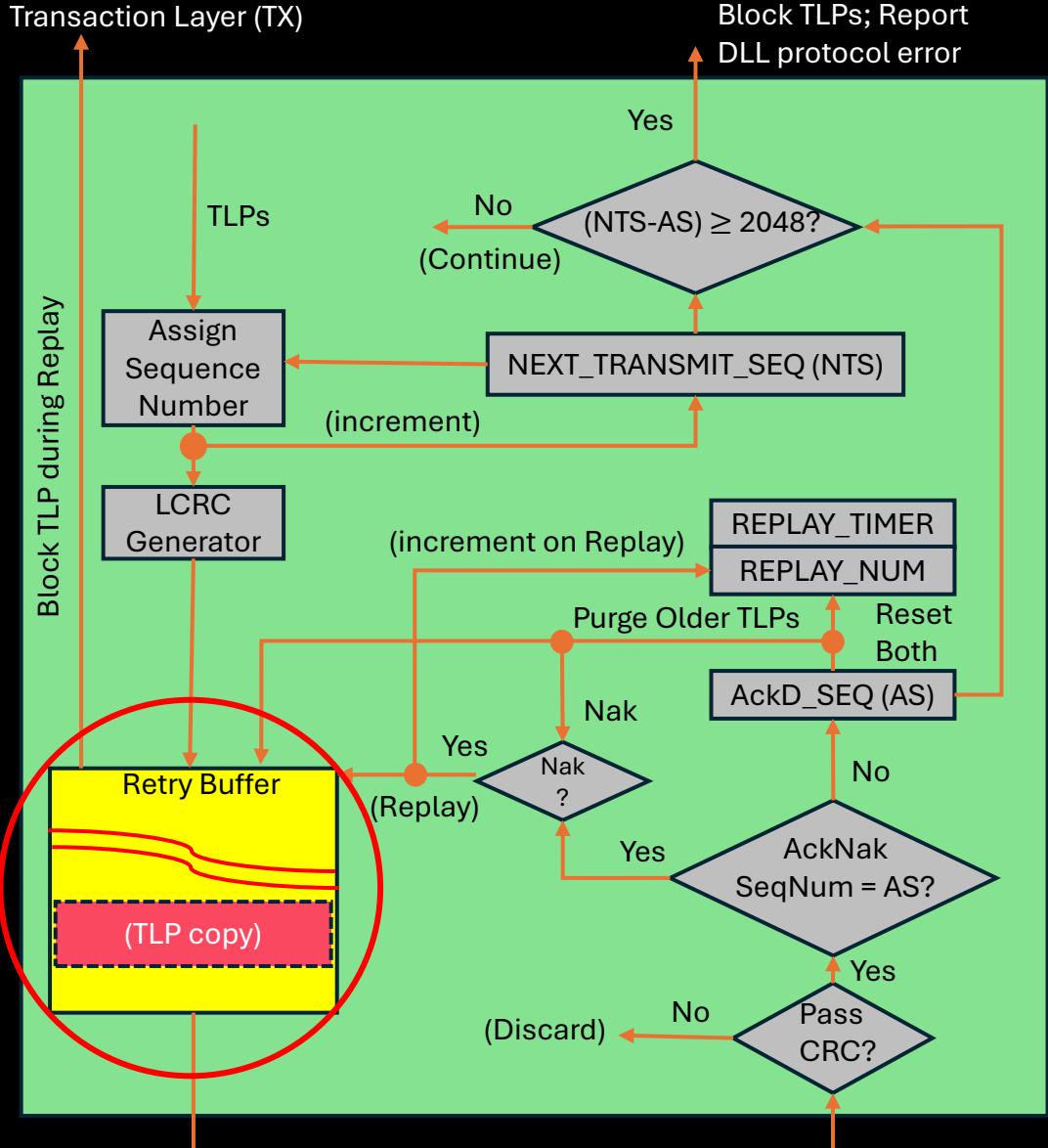


Do Not Distribute

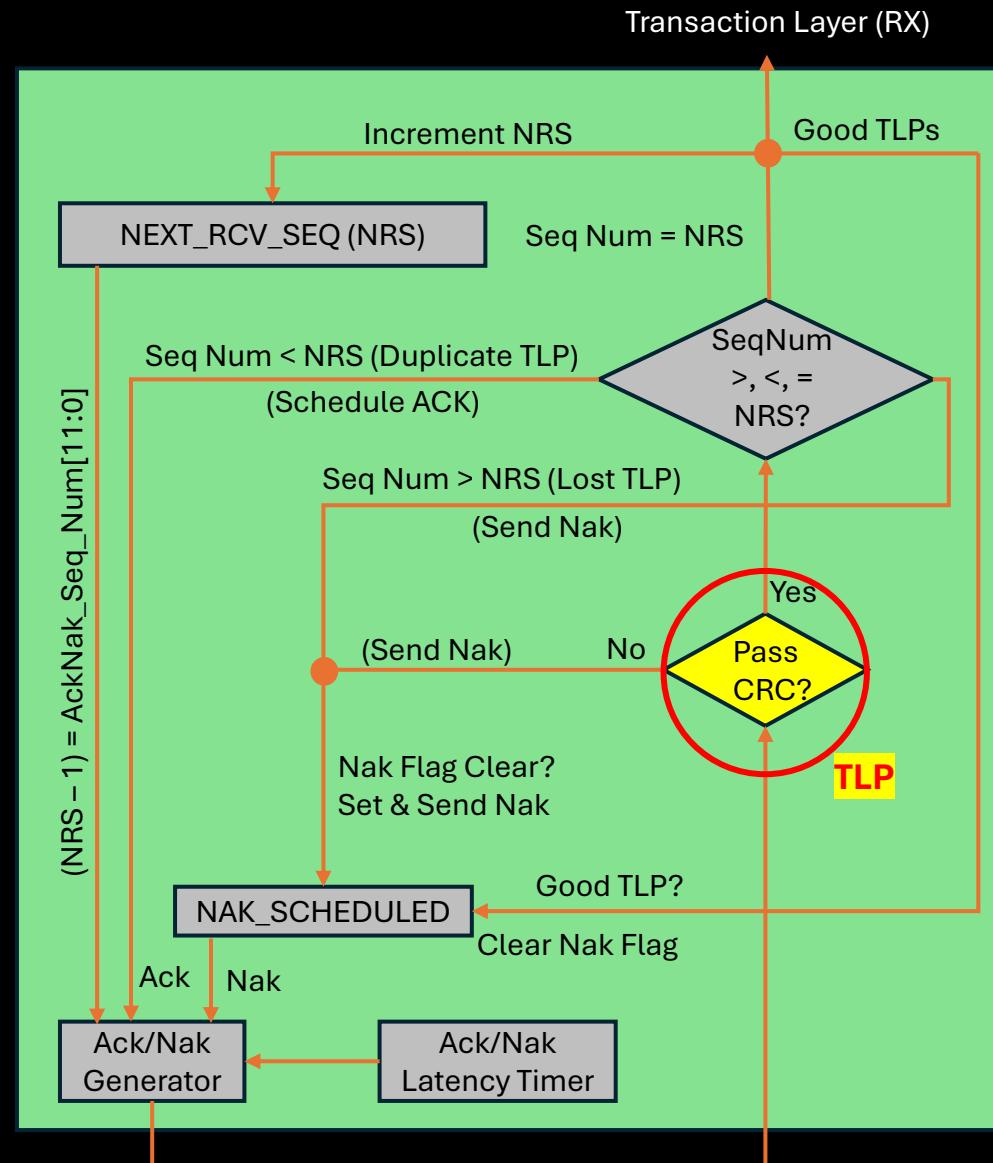
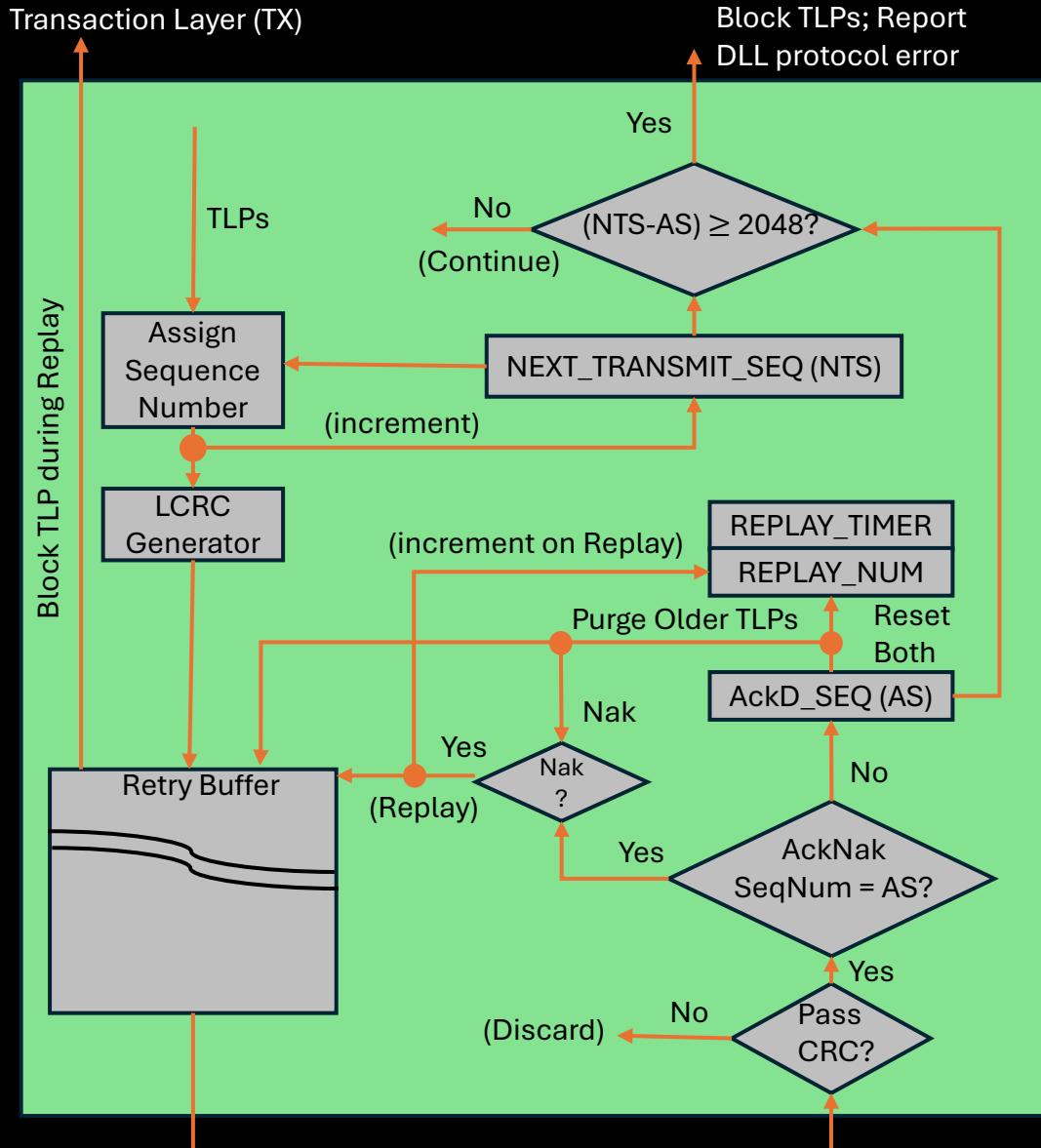
Link CRC Added



Link CRC Added



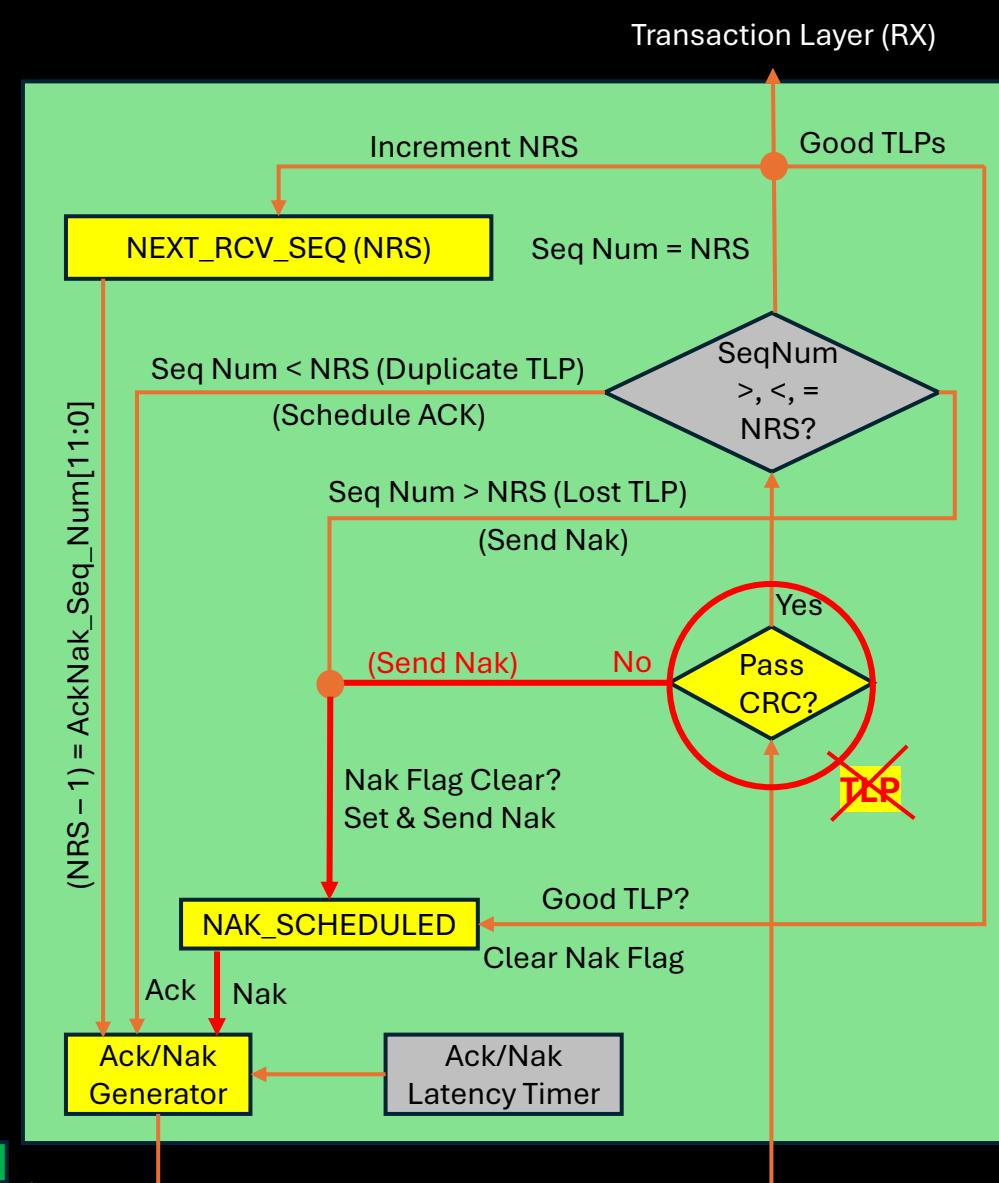
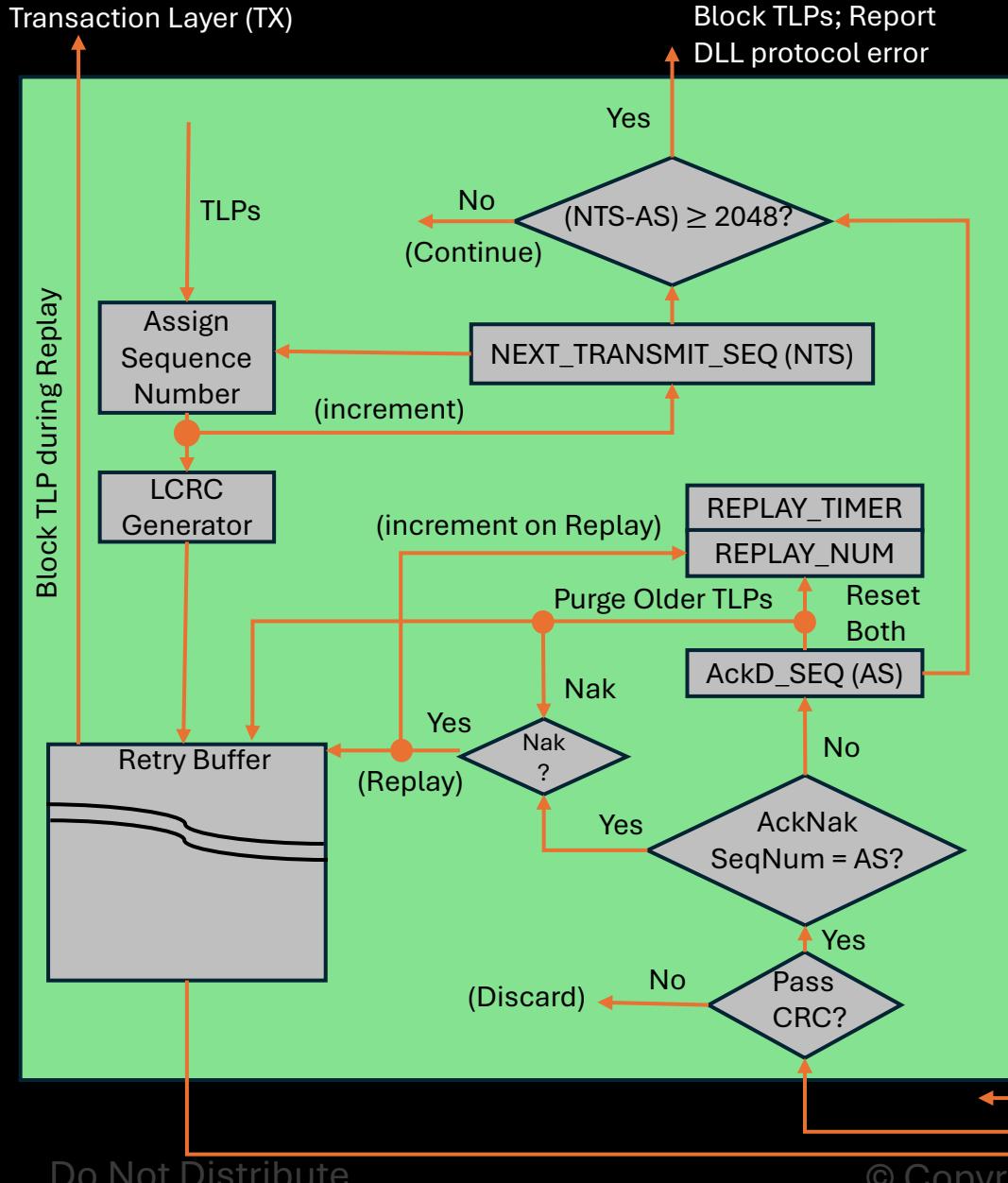
Link CRC Added



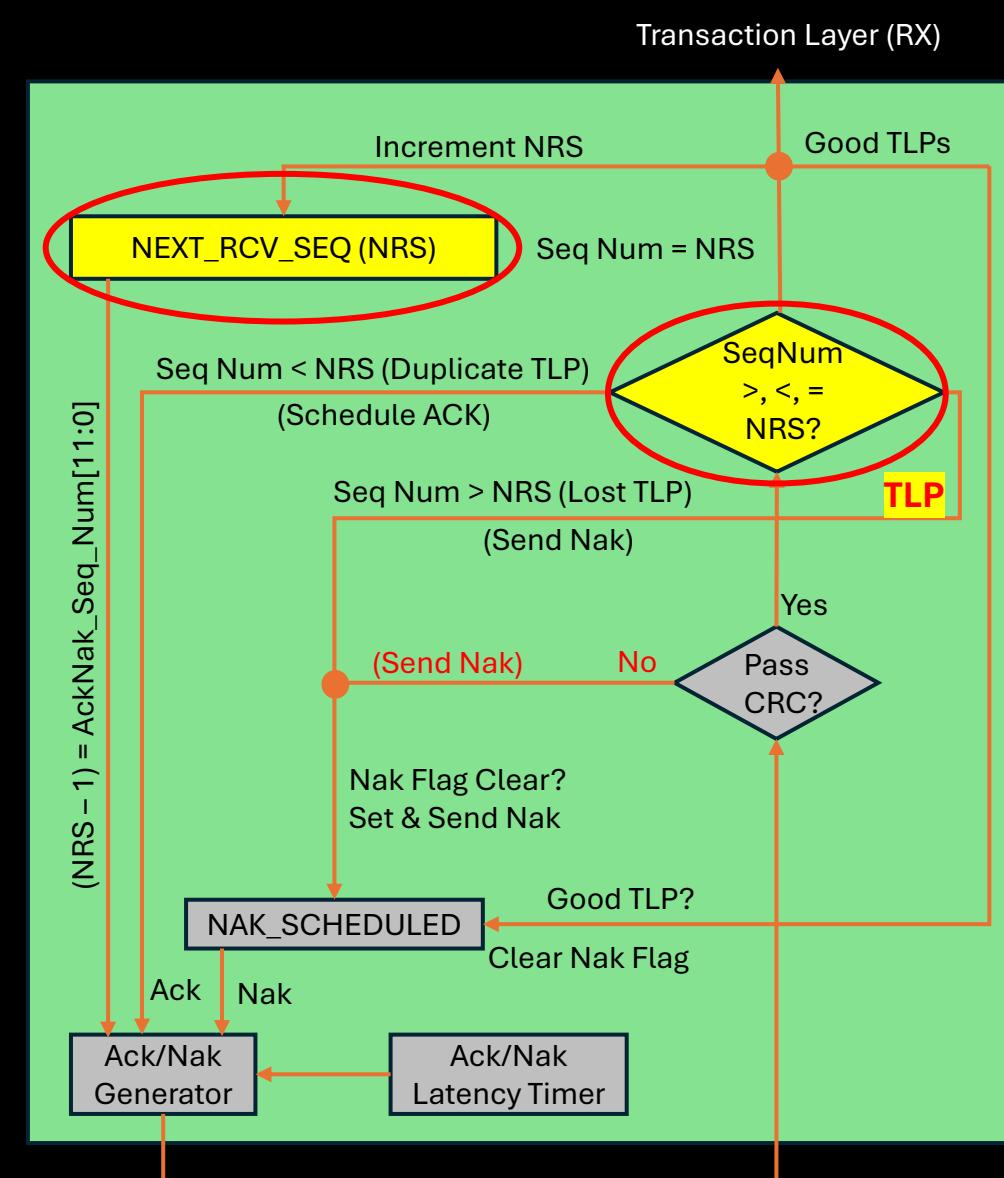
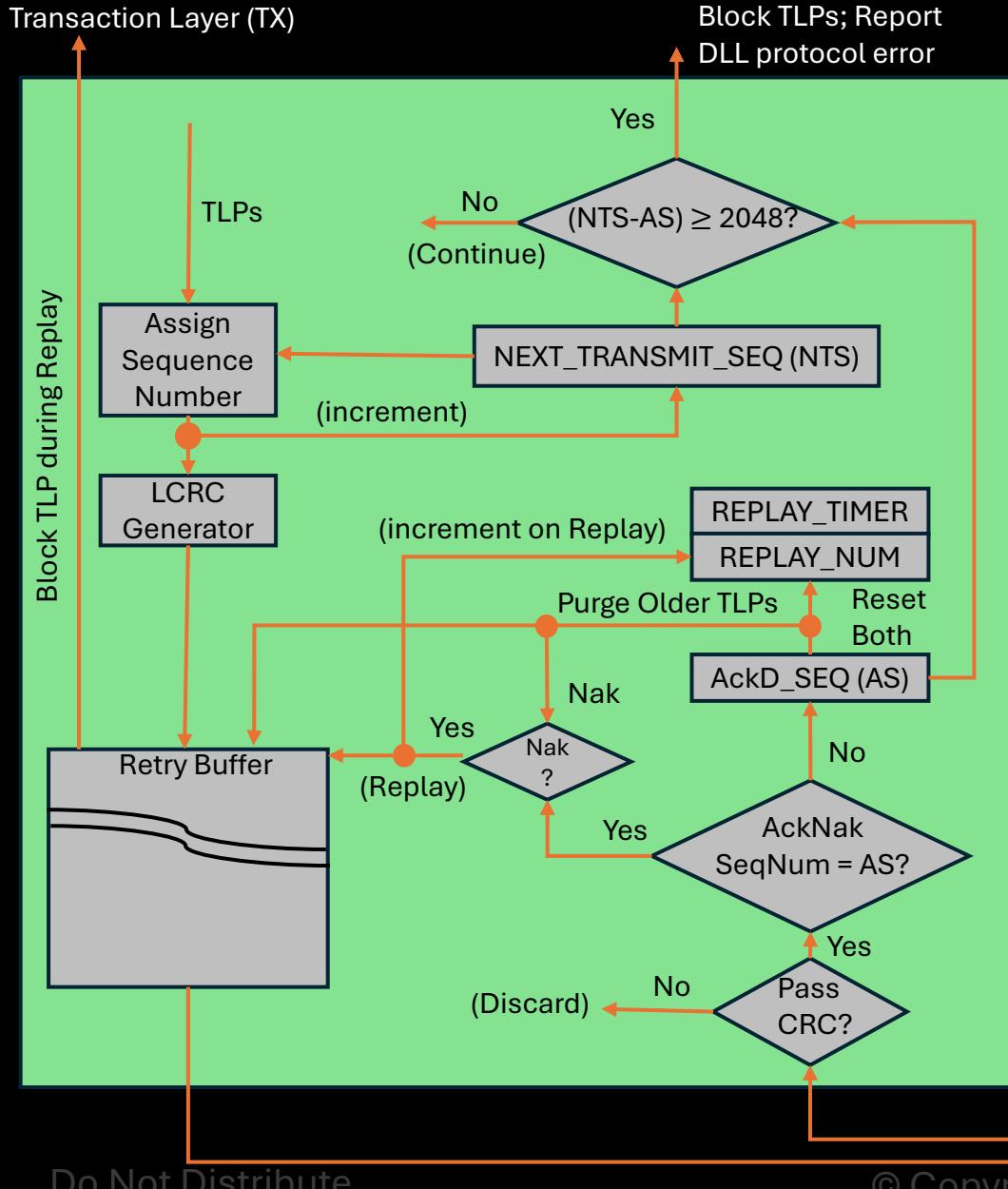
Do Not Distribute

© Copyright protected 2024

Link Check Fails

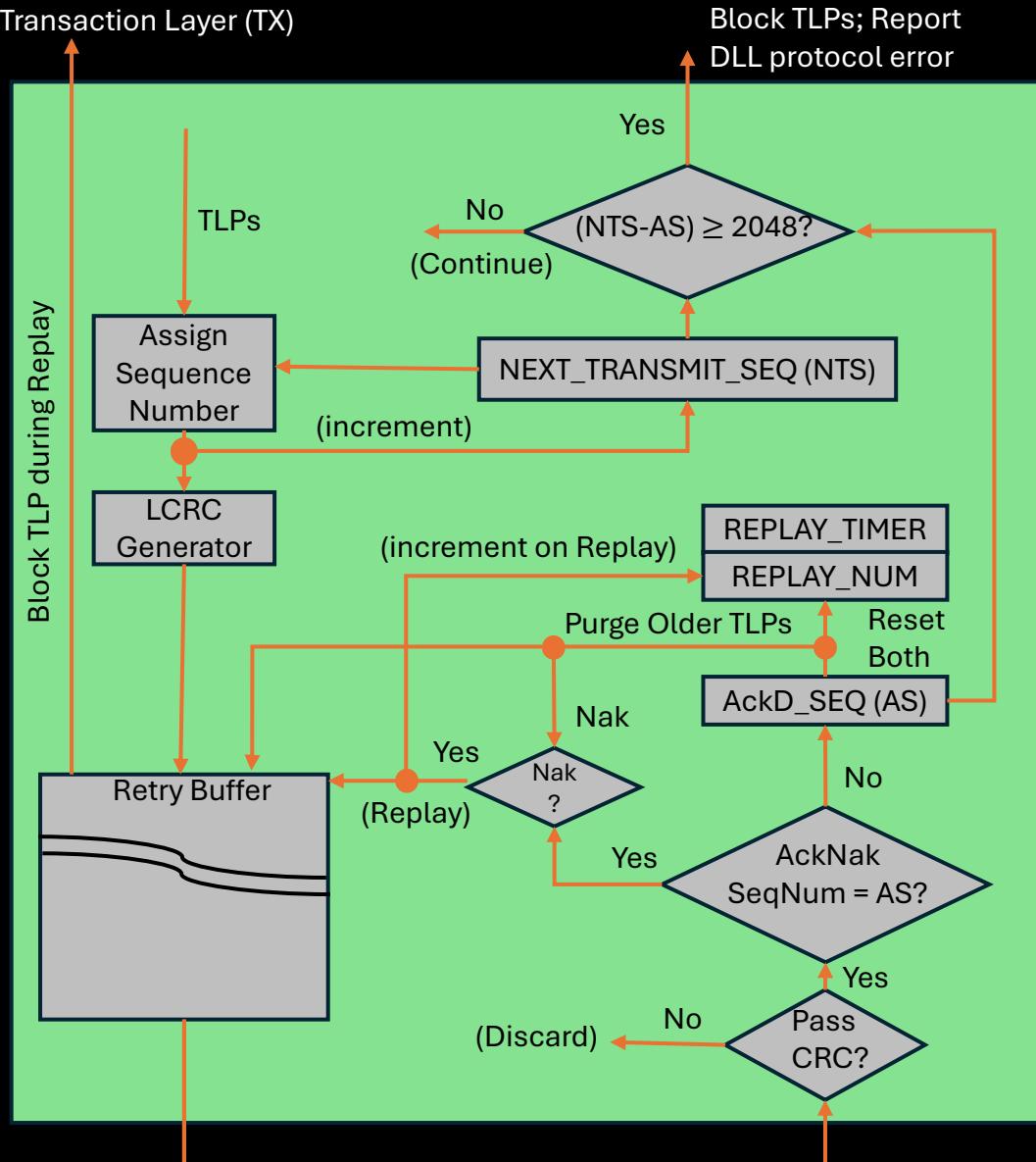


Link OK, Check Sequence Number

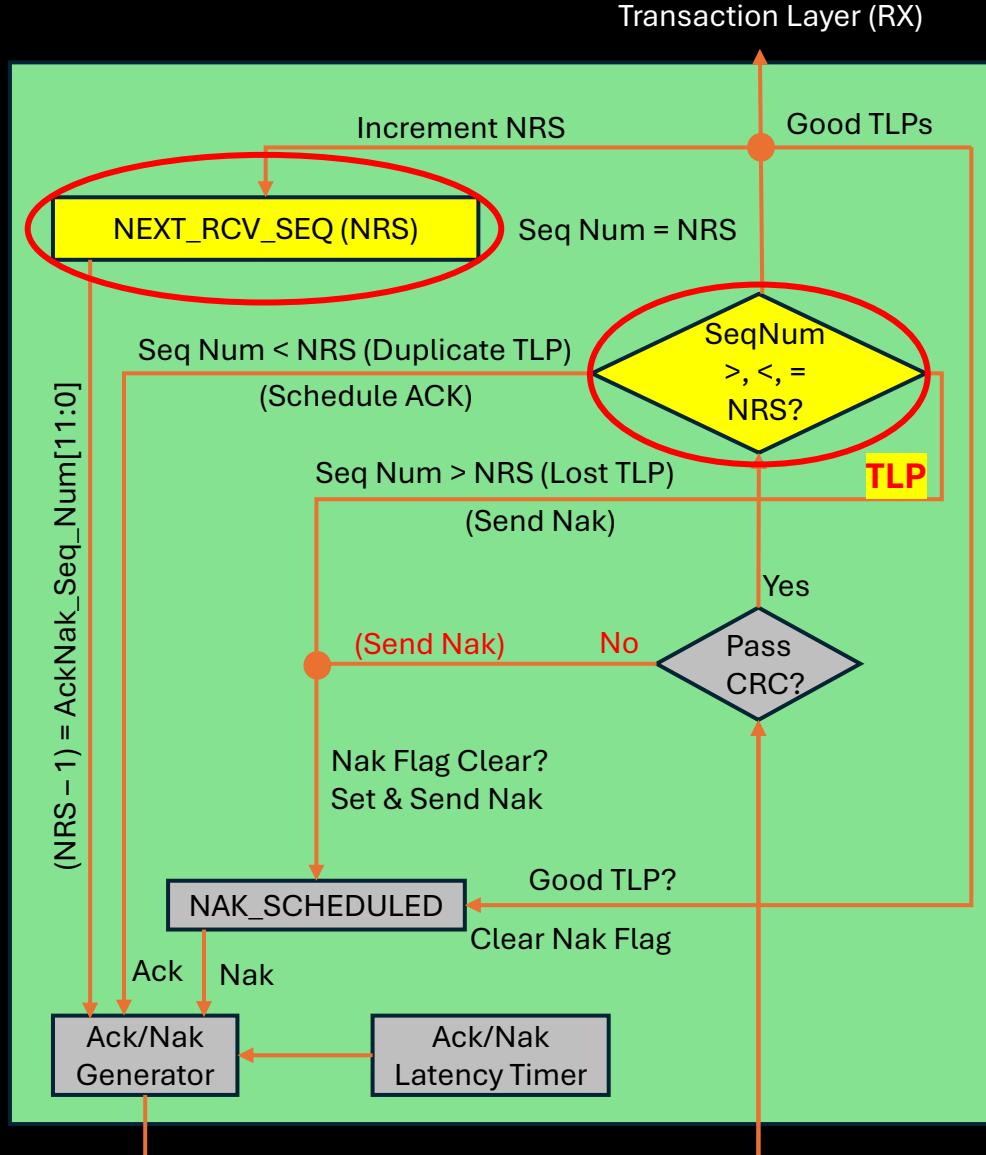


Sequence Number Check

Transaction Layer (TX)



Transaction Layer (RX)



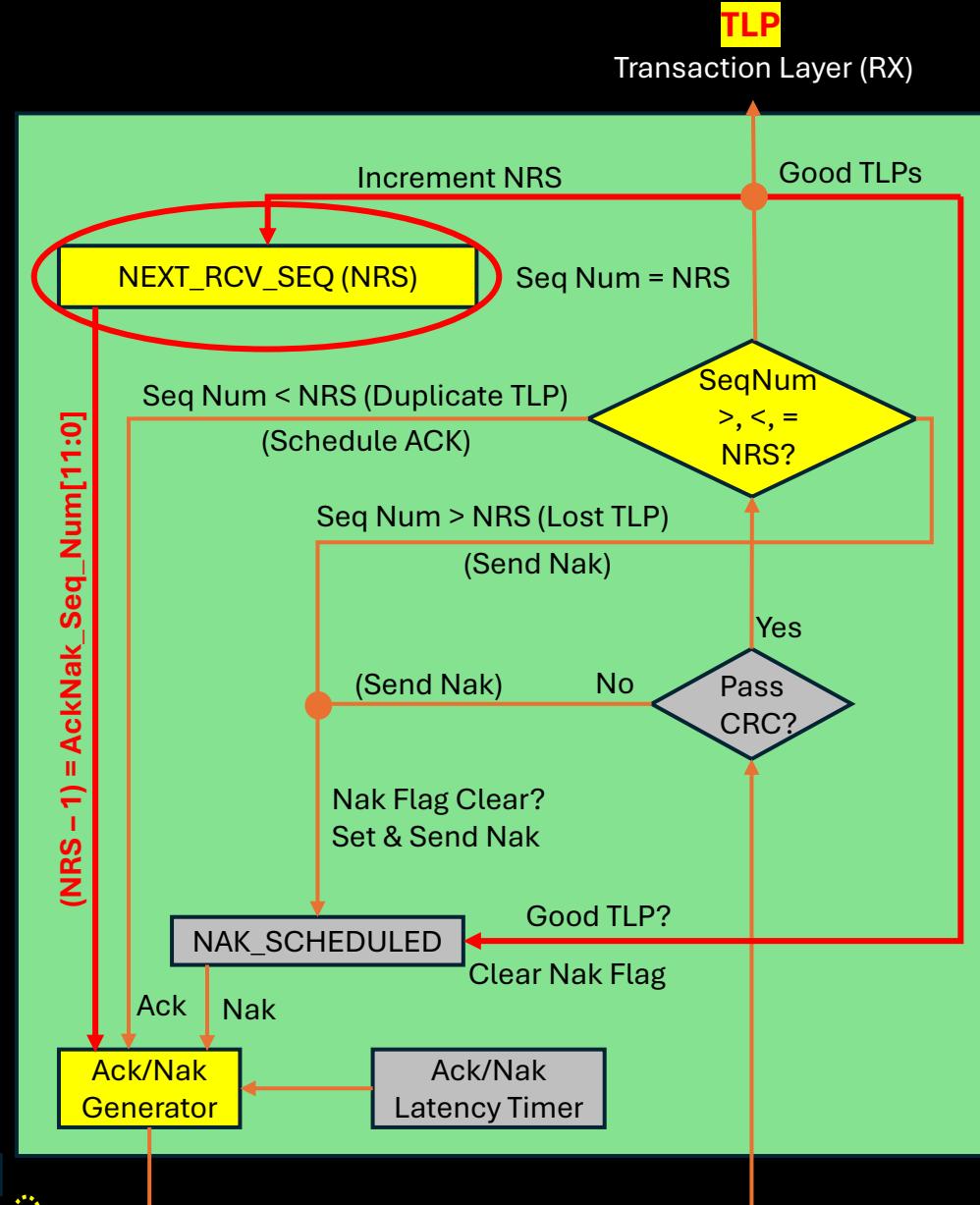
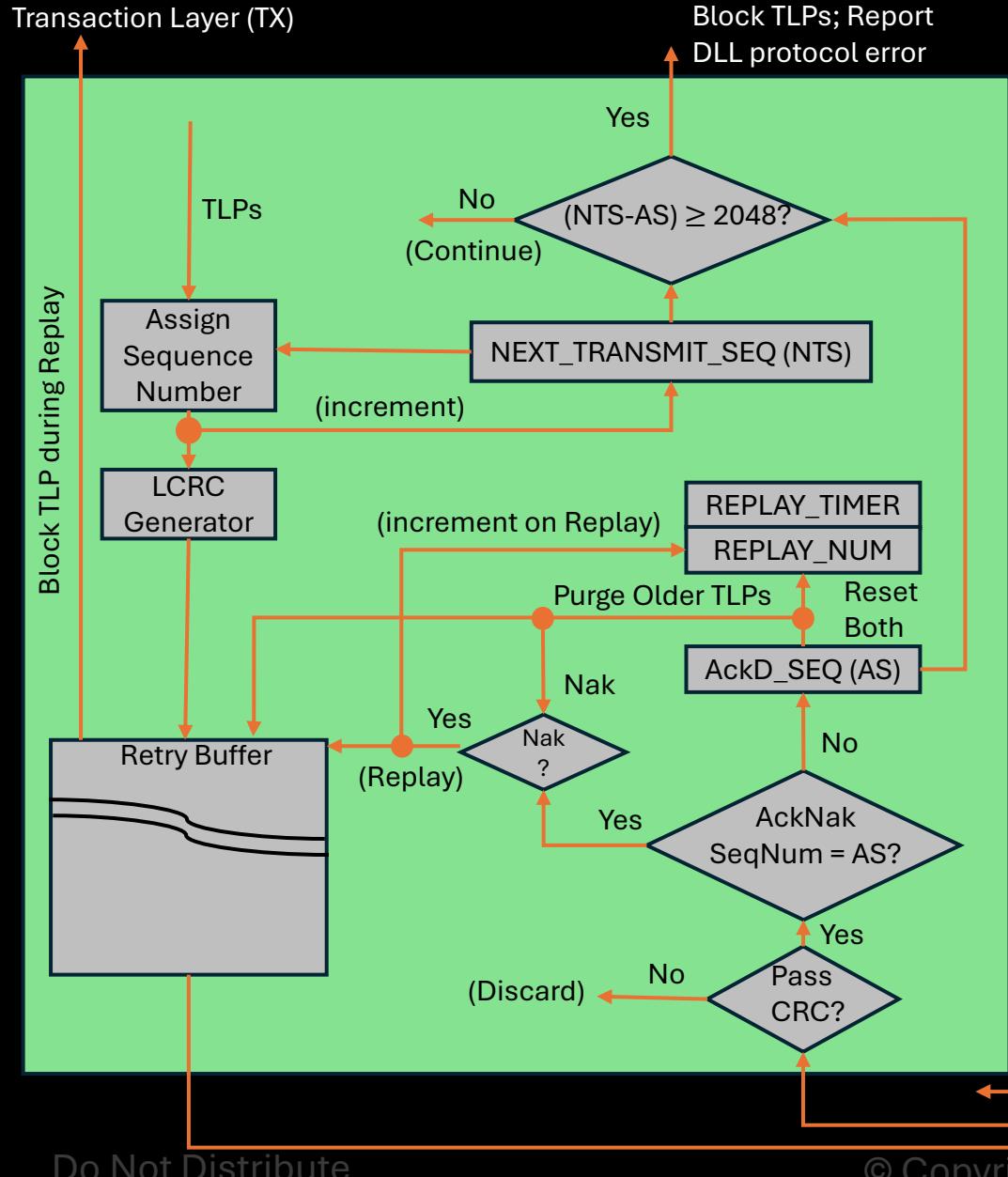
SeqNum
Header
Data
ECRC
LCRC

Check

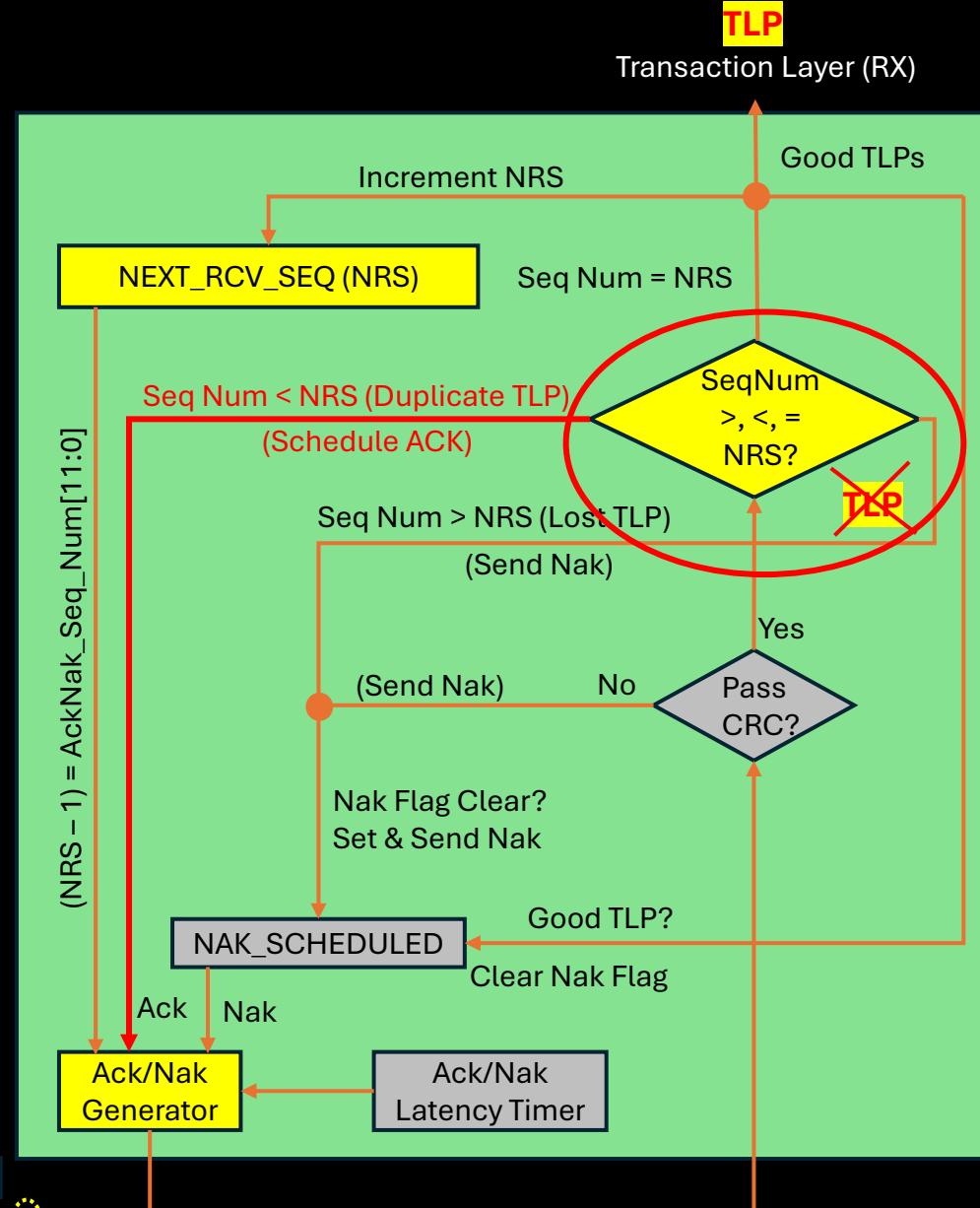
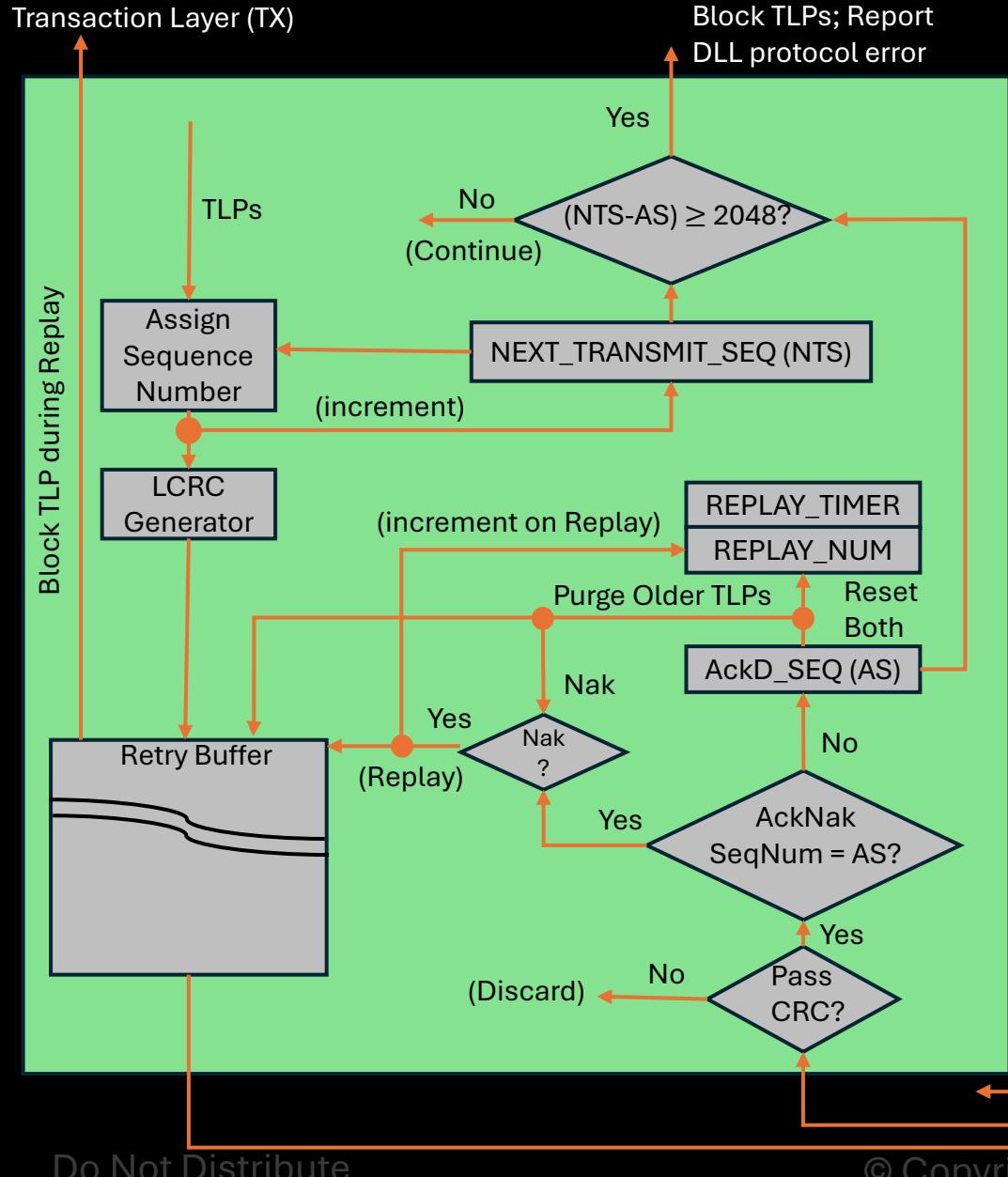
Do Not Distribute

© Copyright protected 2024

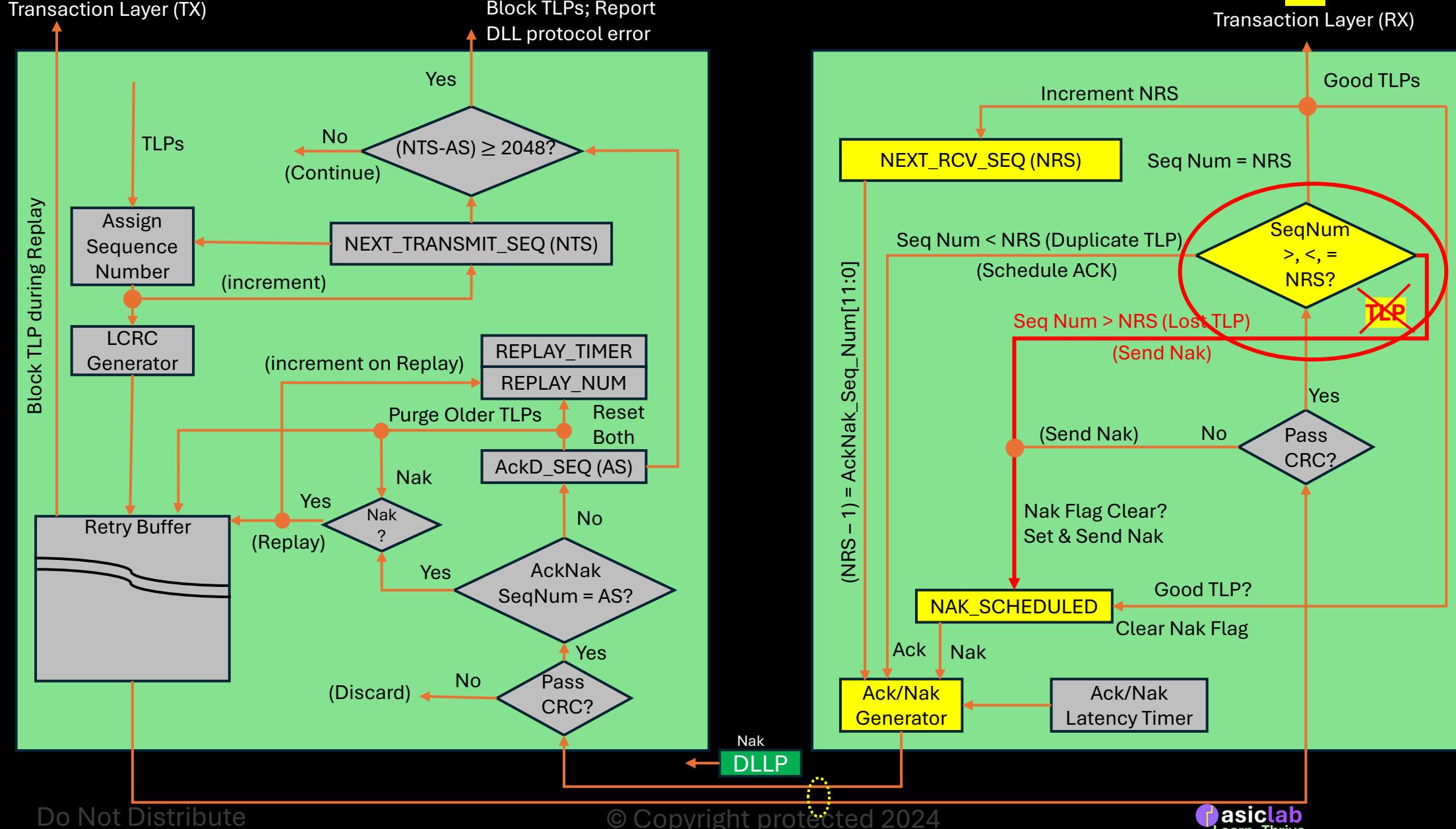
Sequence Number = NRS



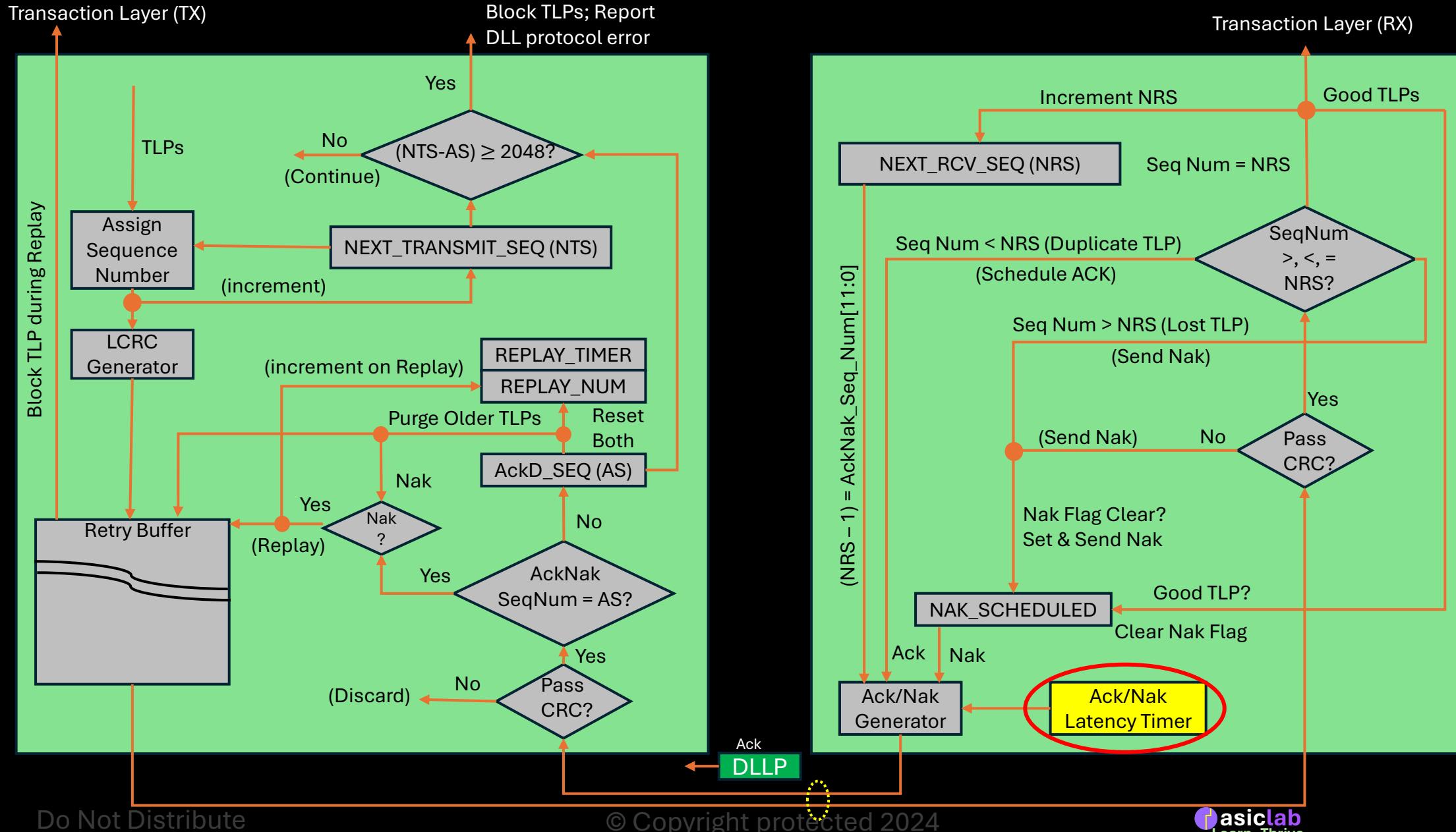
Sequence Number < NRS



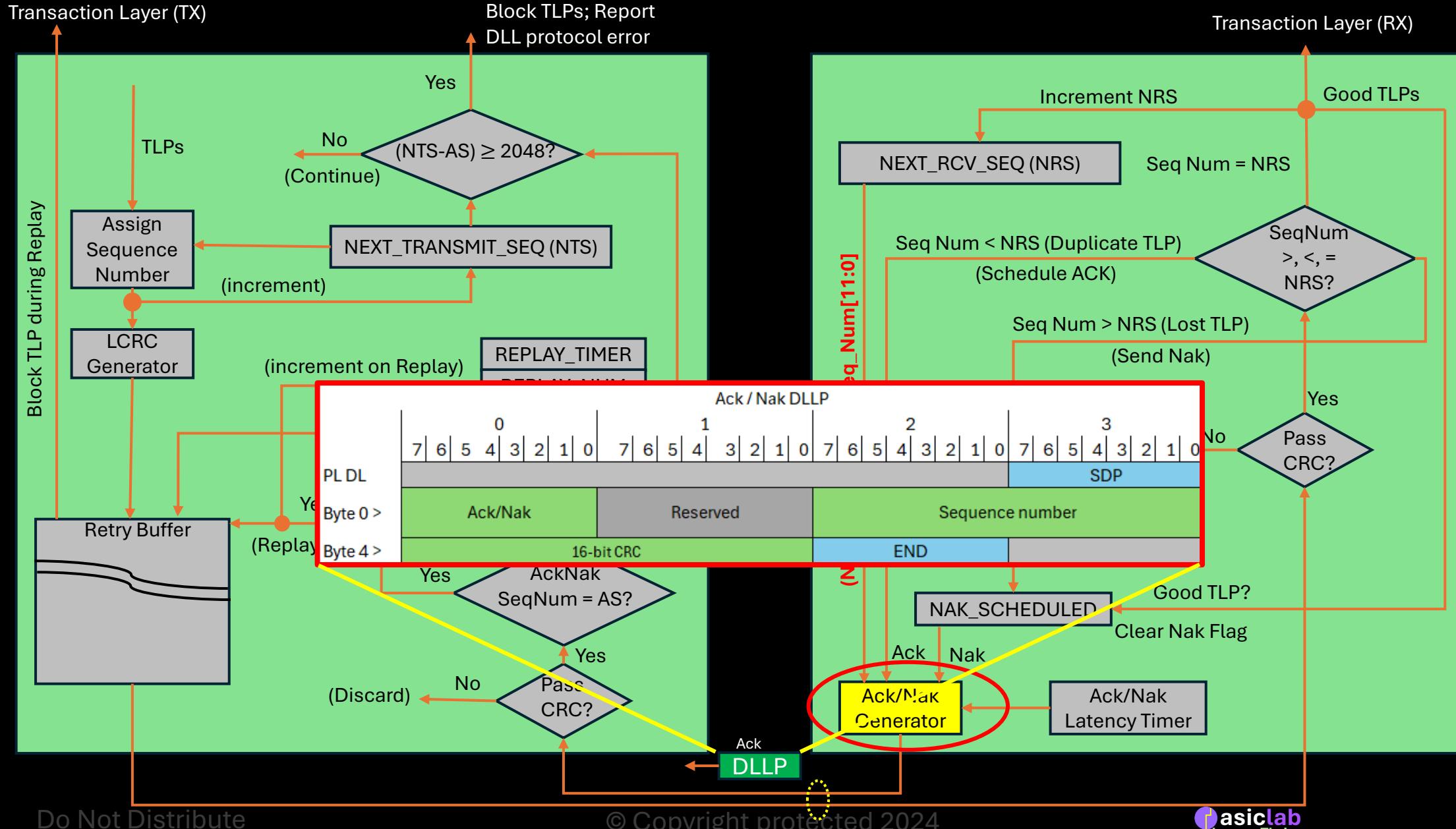
Sequence Number > NRS



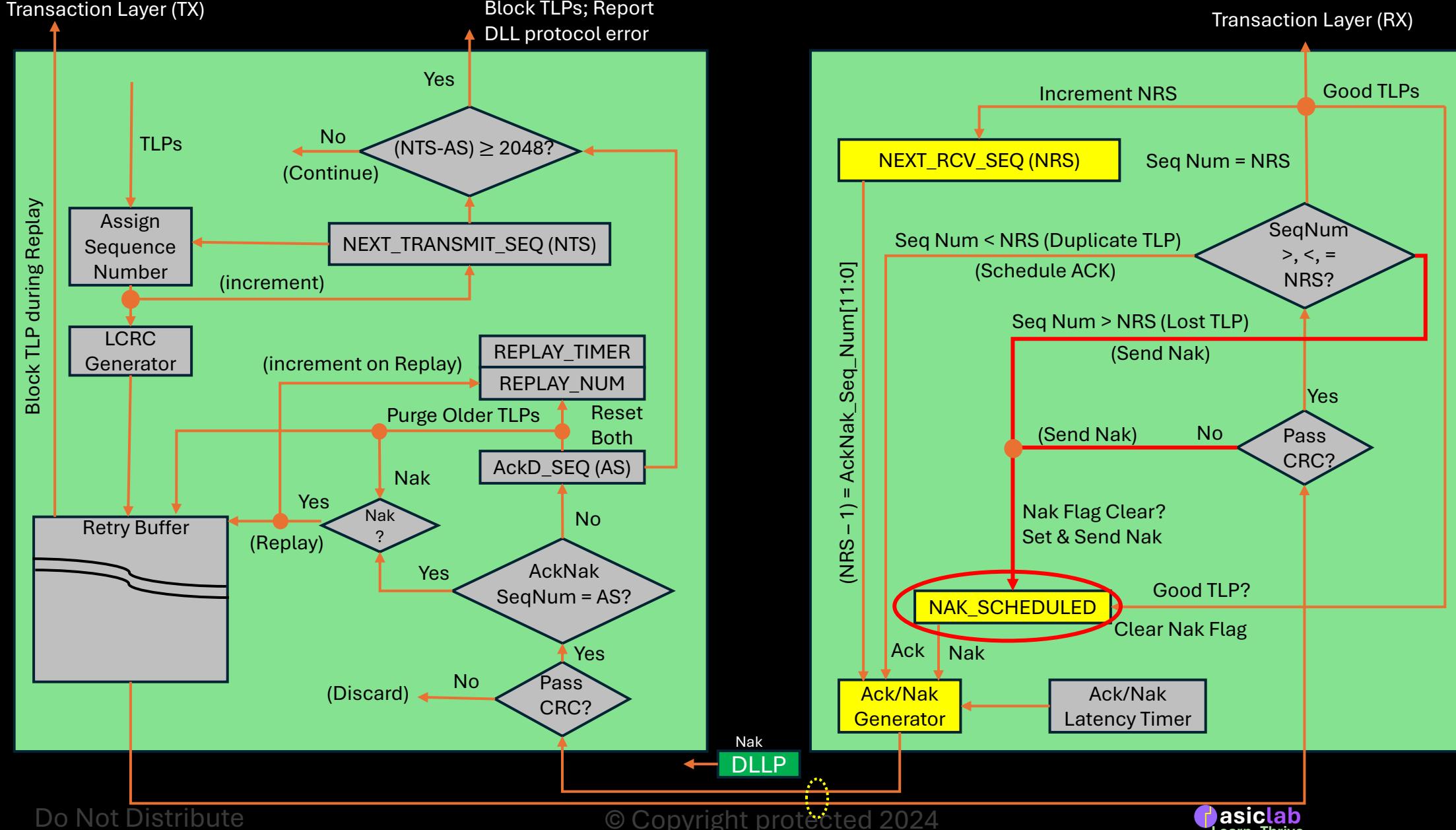
AckNak Latency Timer



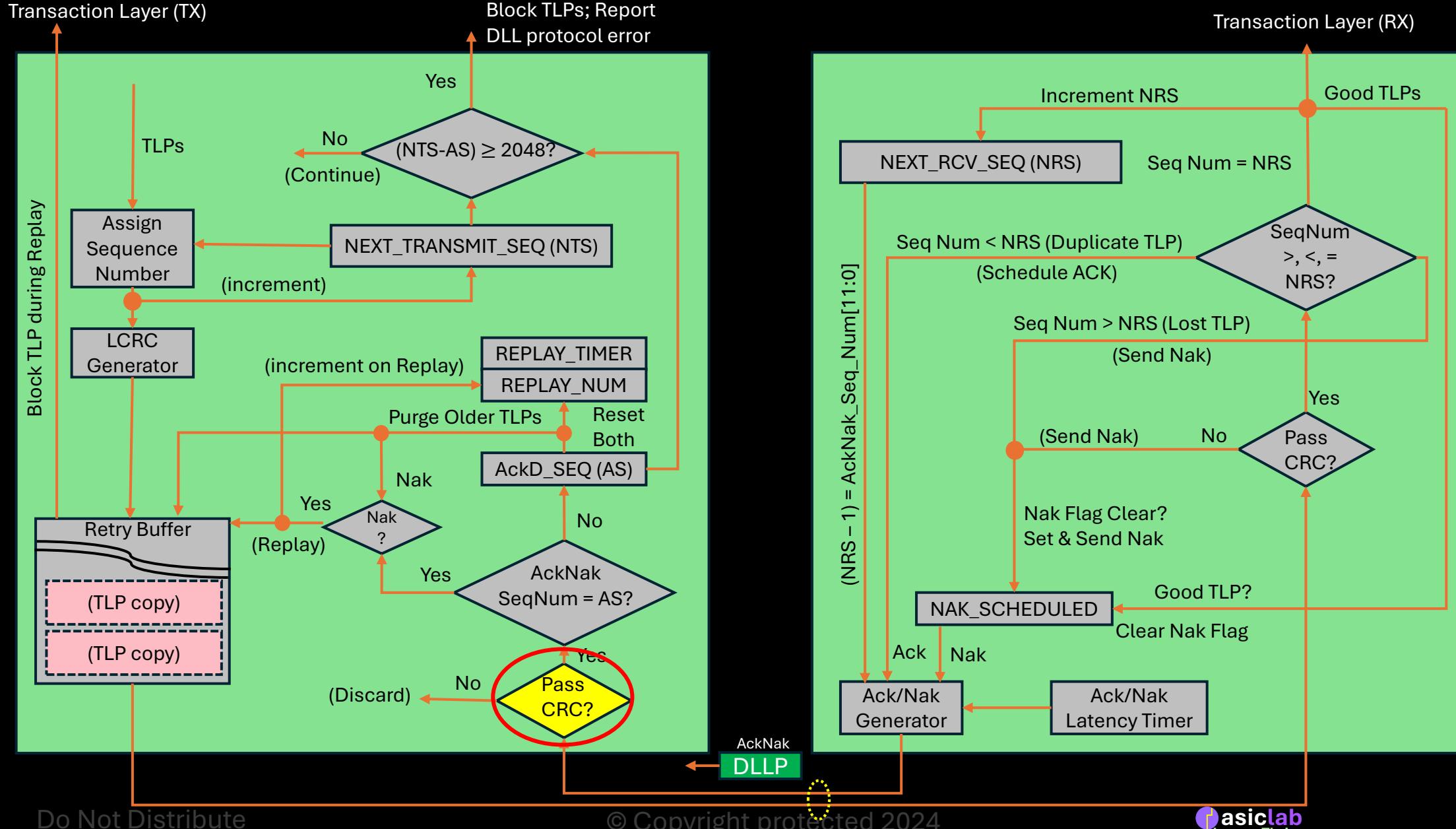
AckNak Generator



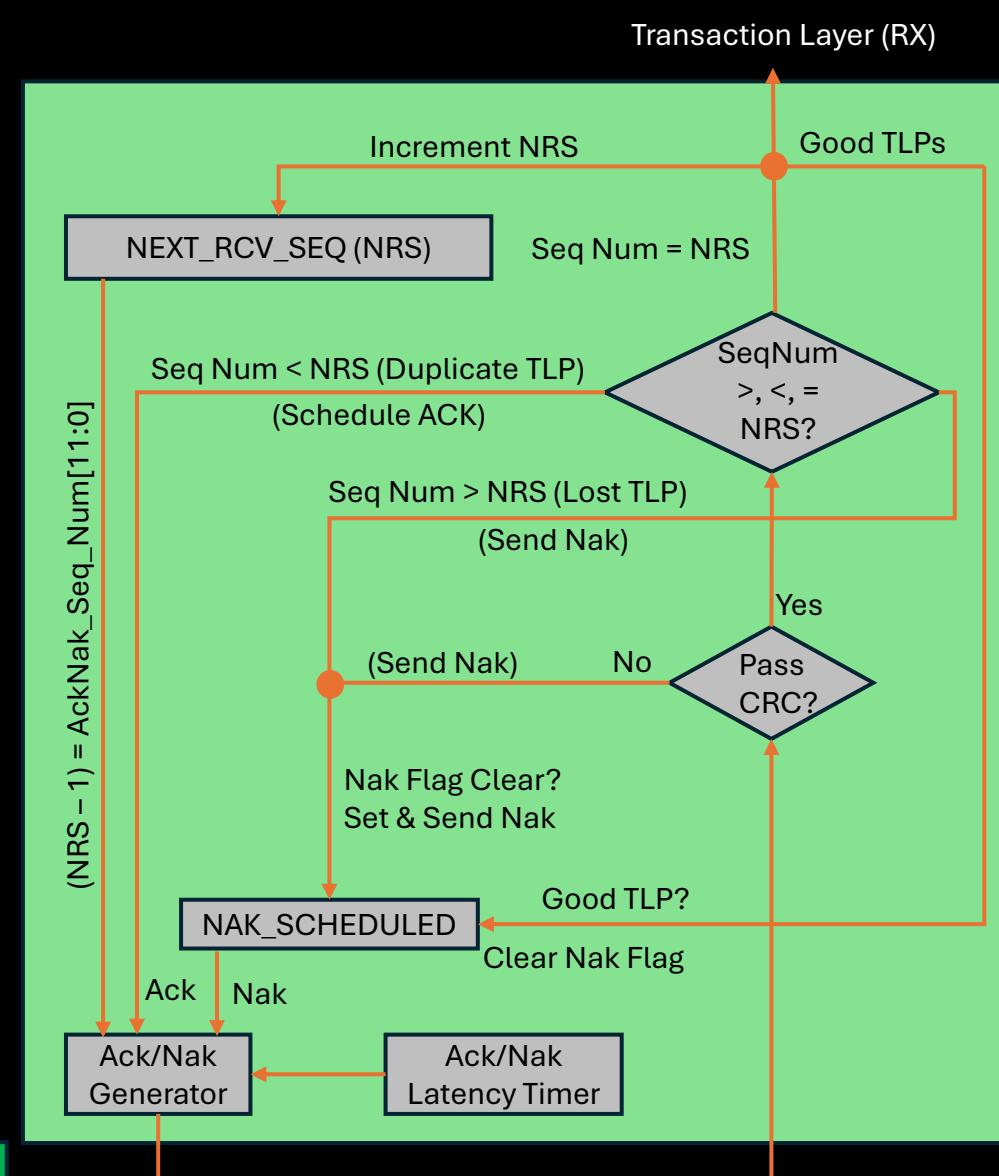
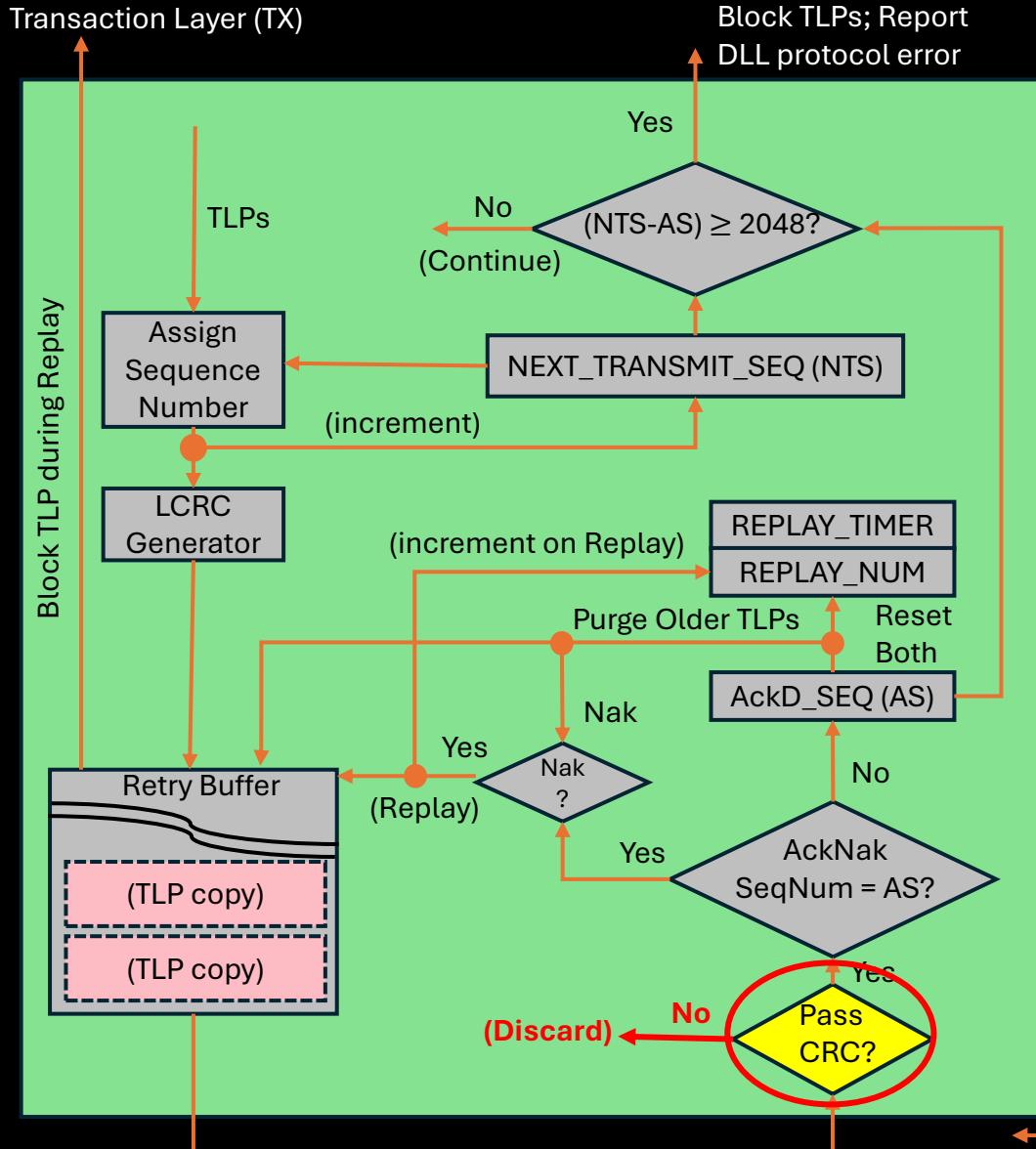
Notes: NAK_SCHEDULED Flag



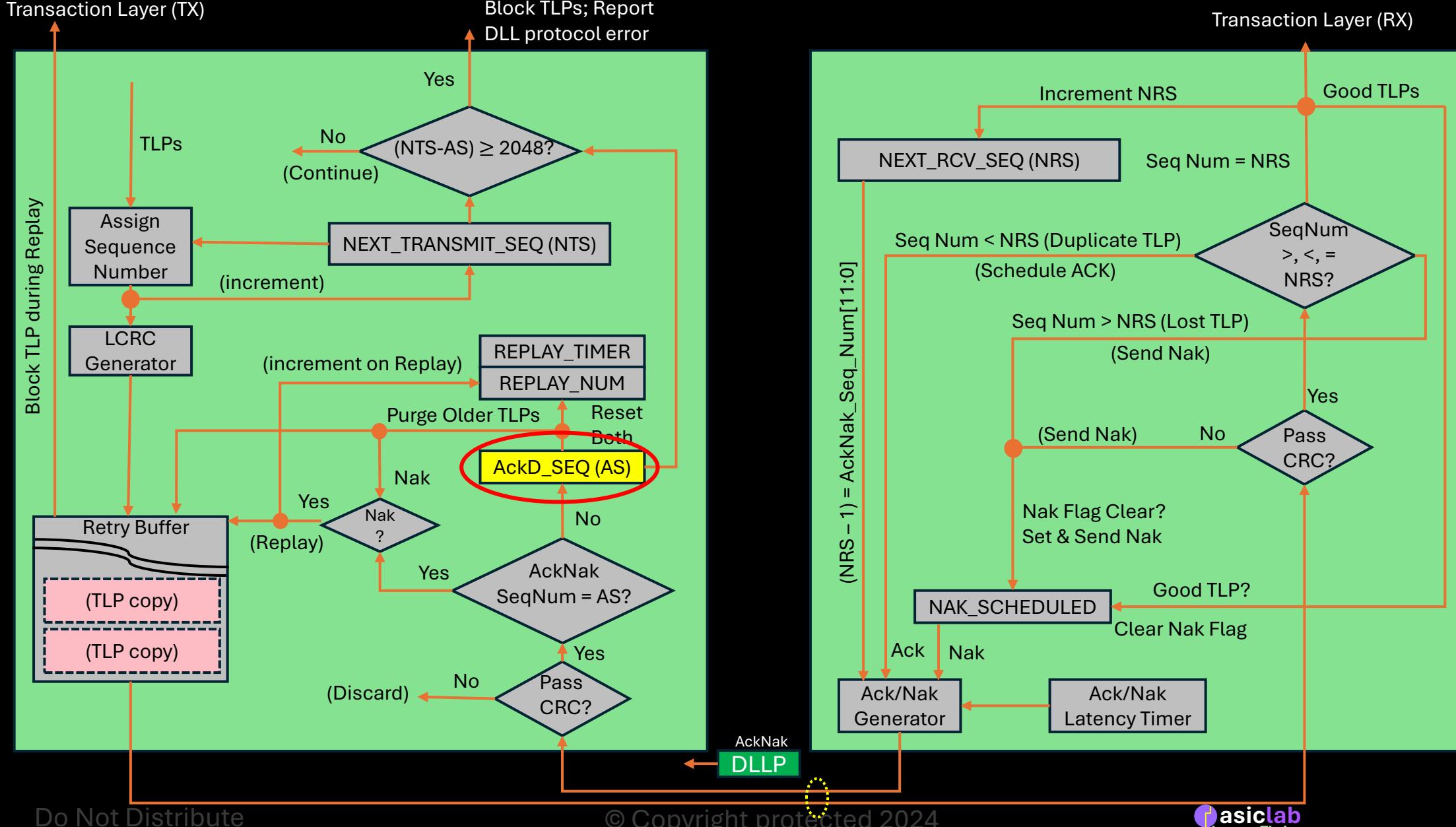
Transmitter Checks Incoming DLLPs



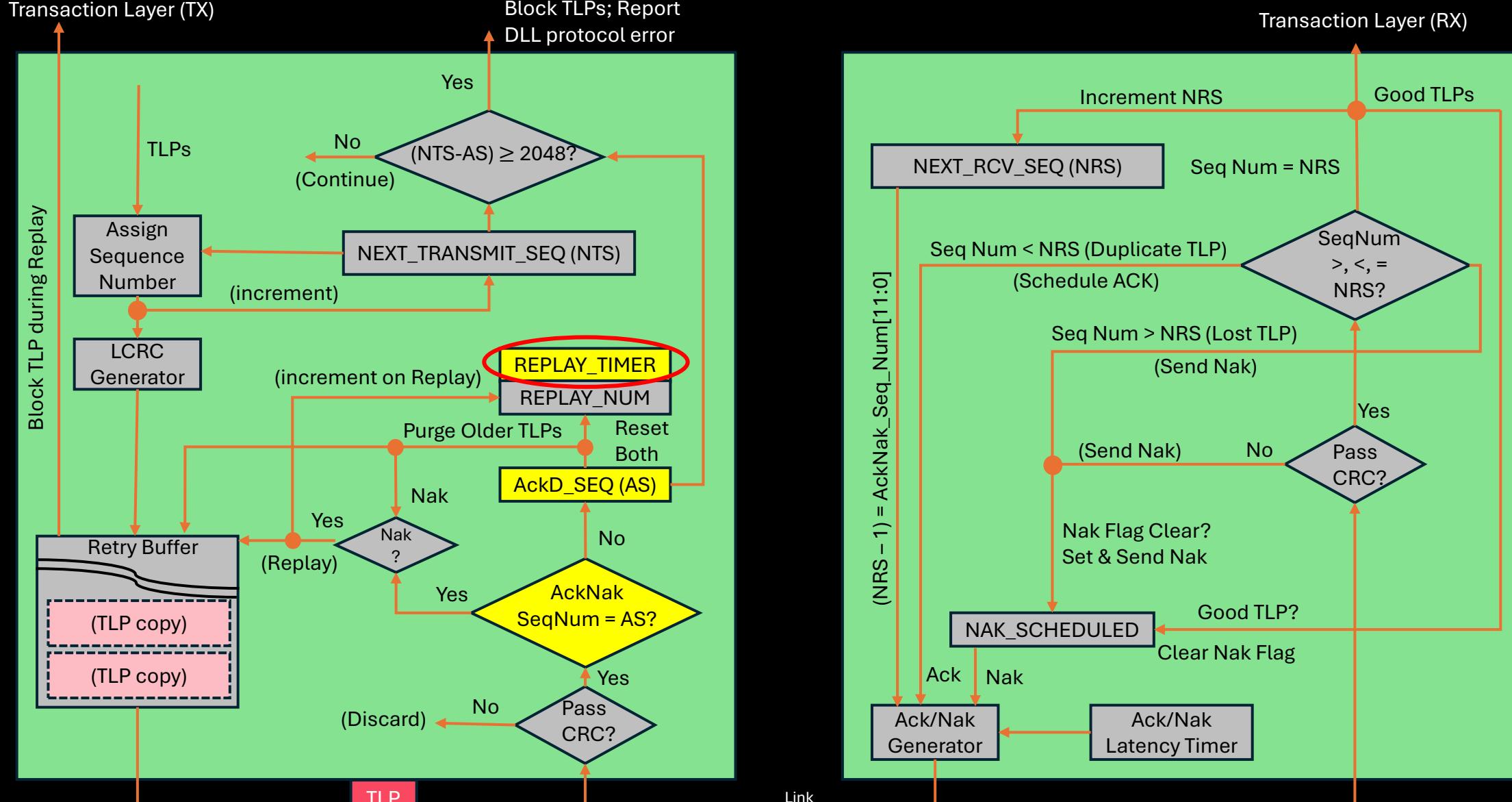
If DLLP CRC Check Fails



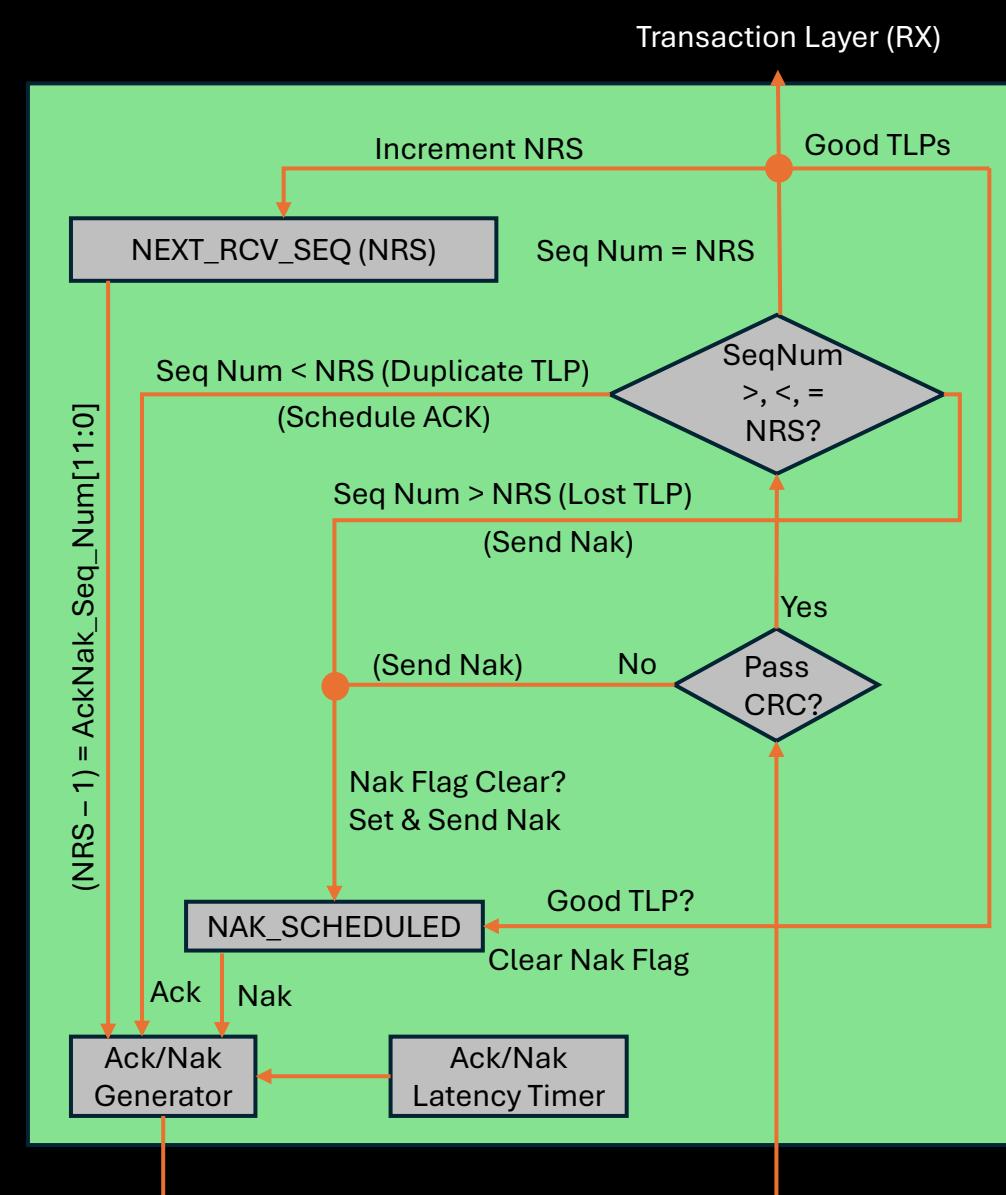
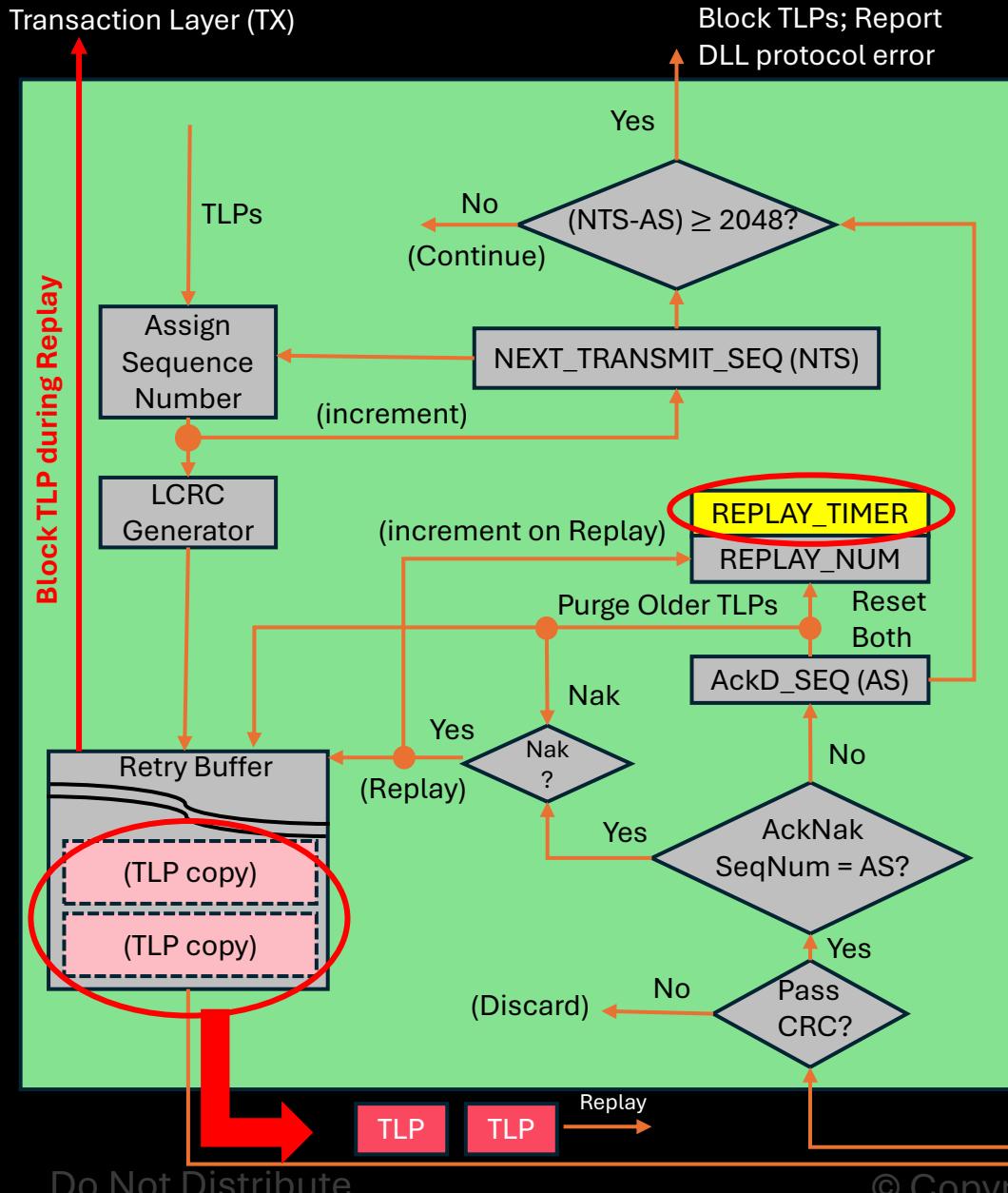
Acknowledges Sequence Number



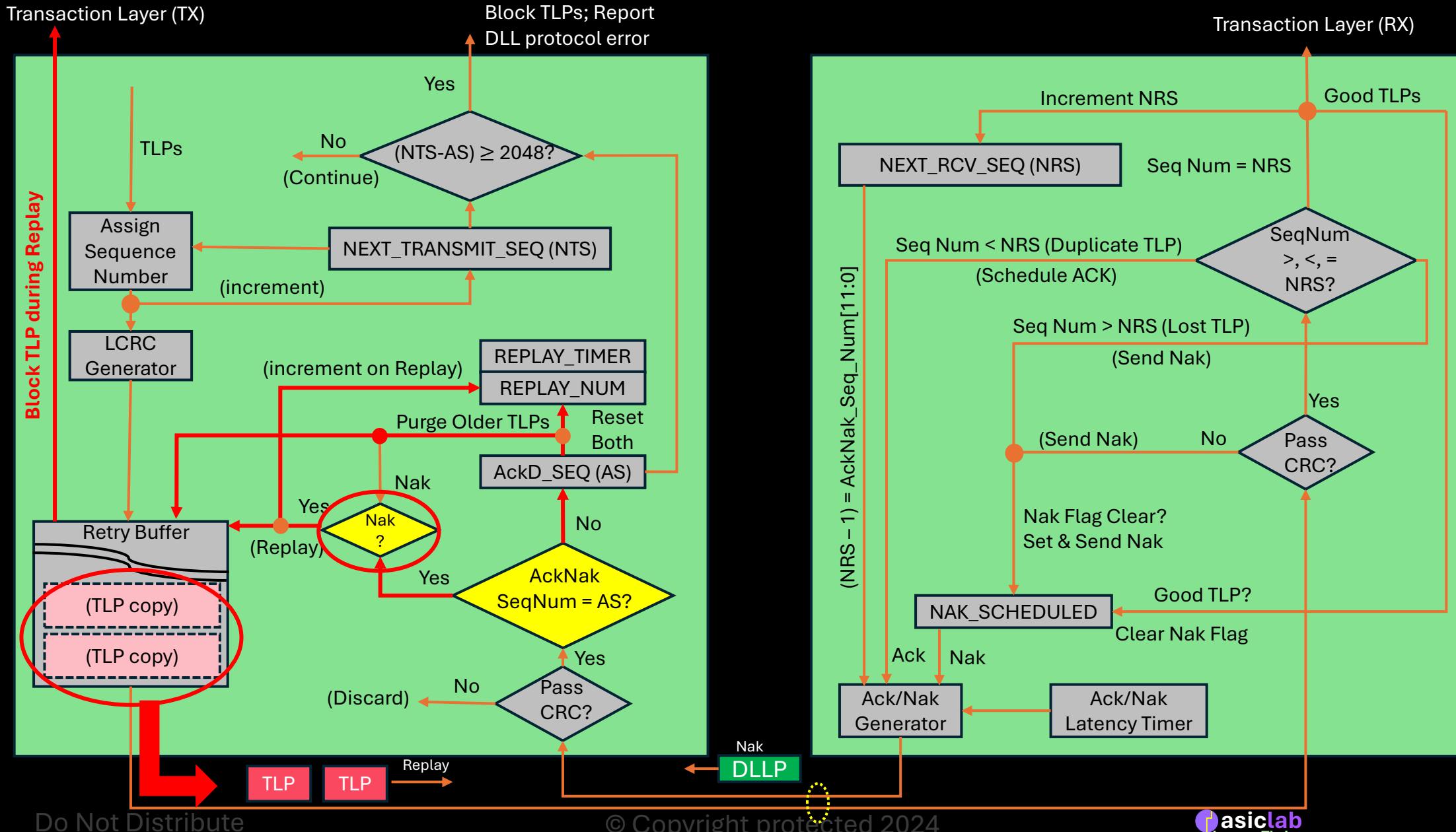
Transmitter Replay Timer Basics



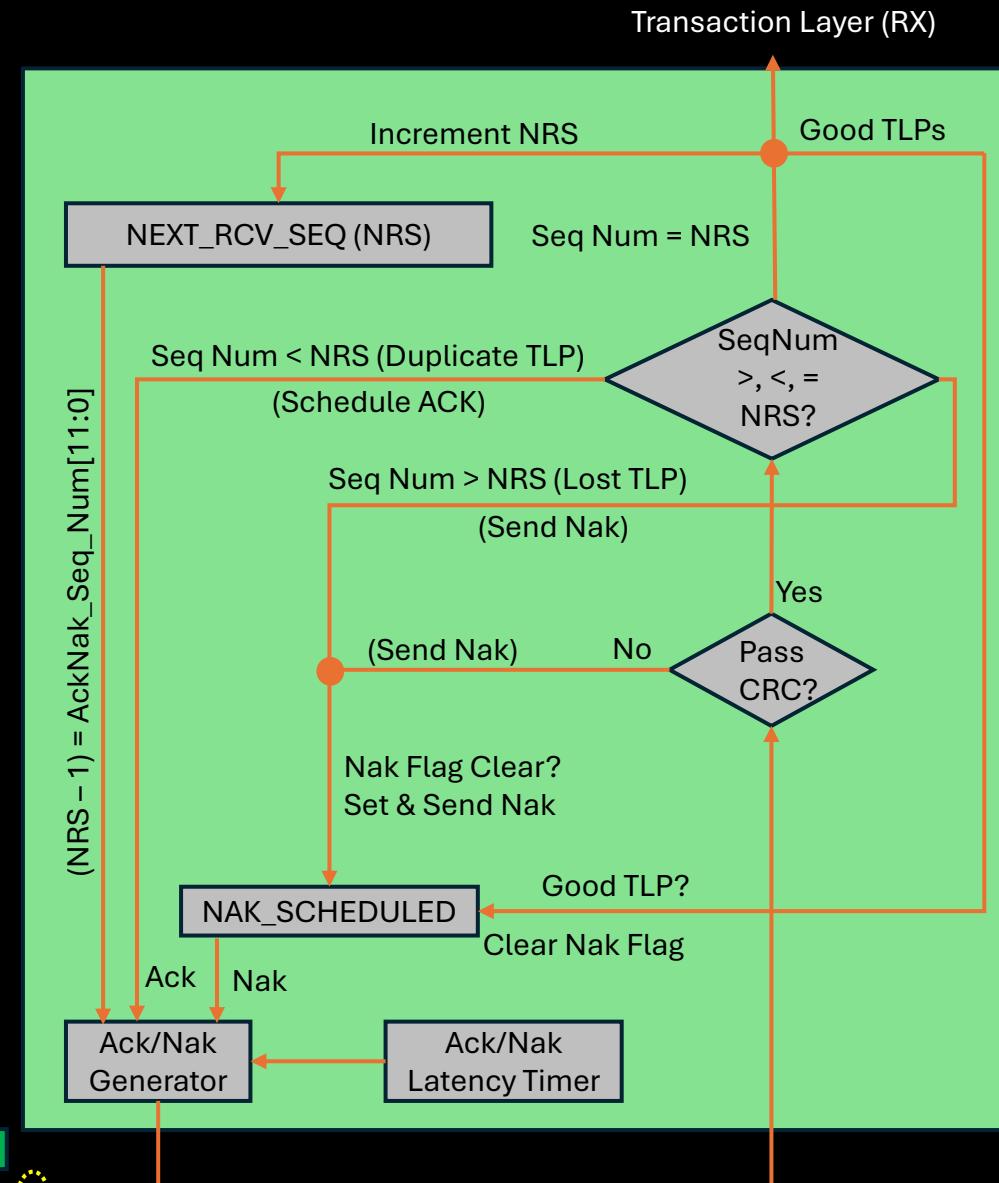
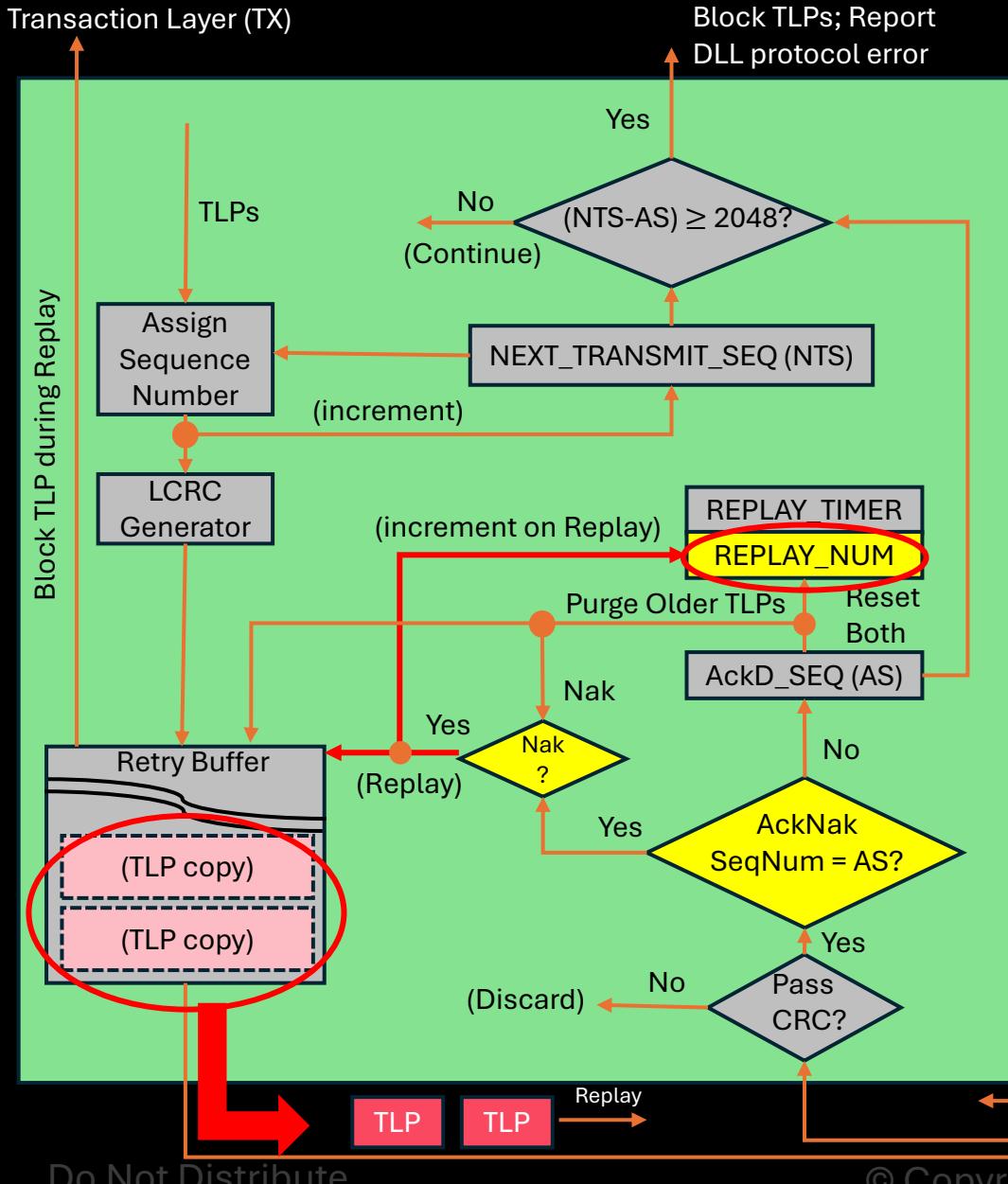
When Transmitter Replay Timer Expires



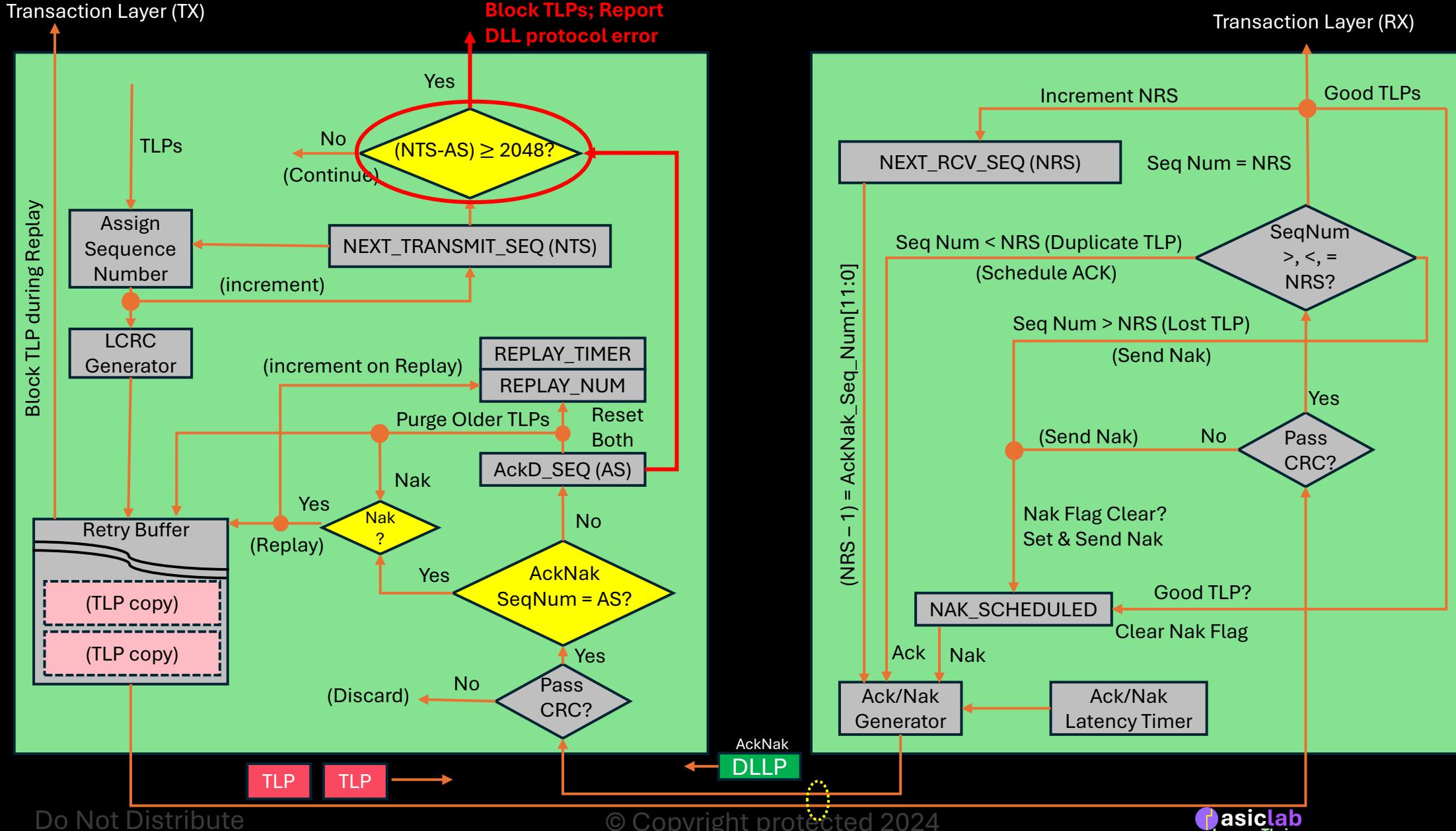
Replay Caused by Nak DLLP



Replay Number Counter



One More Transmitter Ack/Nak Check

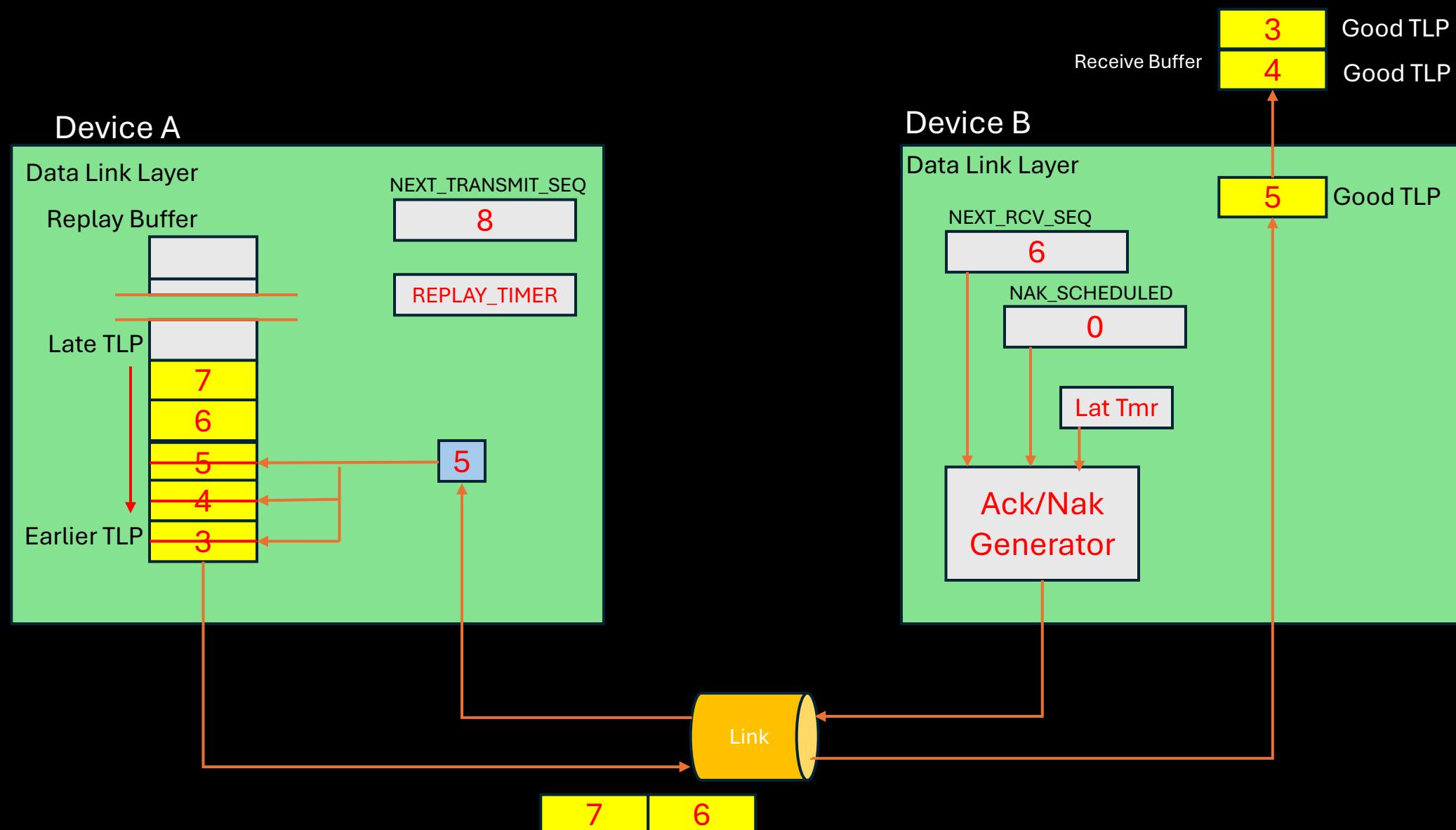


Ack/Nak Impact

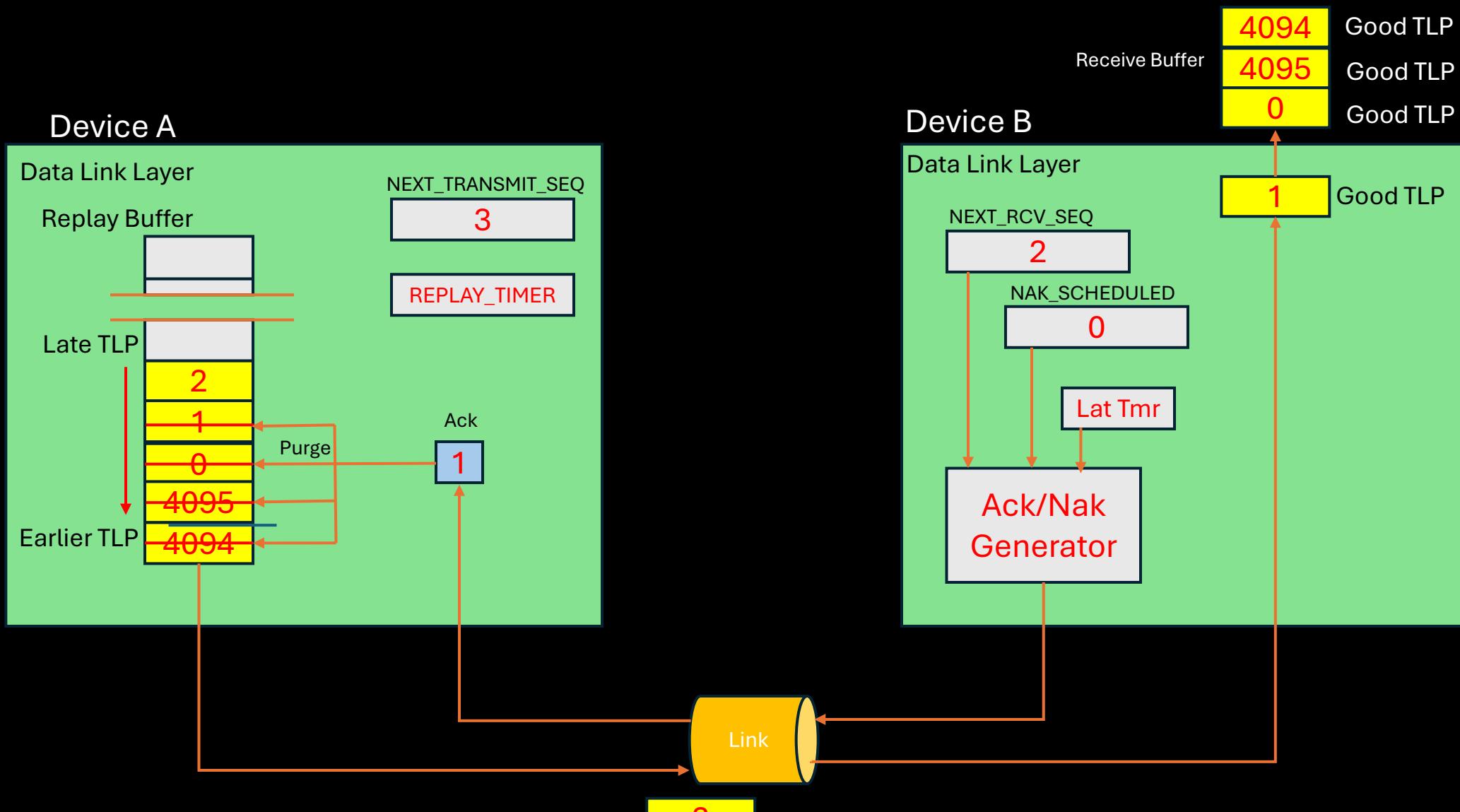
- **Hardware costs:**
 - Transmitter Retry Buffer storage, logic for Replay Timer, NTS and AS counters, CRC generation etc.
 - Receiver TLP buffers, Ack/Nak Latency Timer, NRS counter, CRC checking logic, etc.
- **Performance costs:**
 - Fixed overhead of 32-bit LCRC appended to TLPs
 - When replay is needed:
 - Bandwidth consumed
 - Replay latency penalty

asiclab

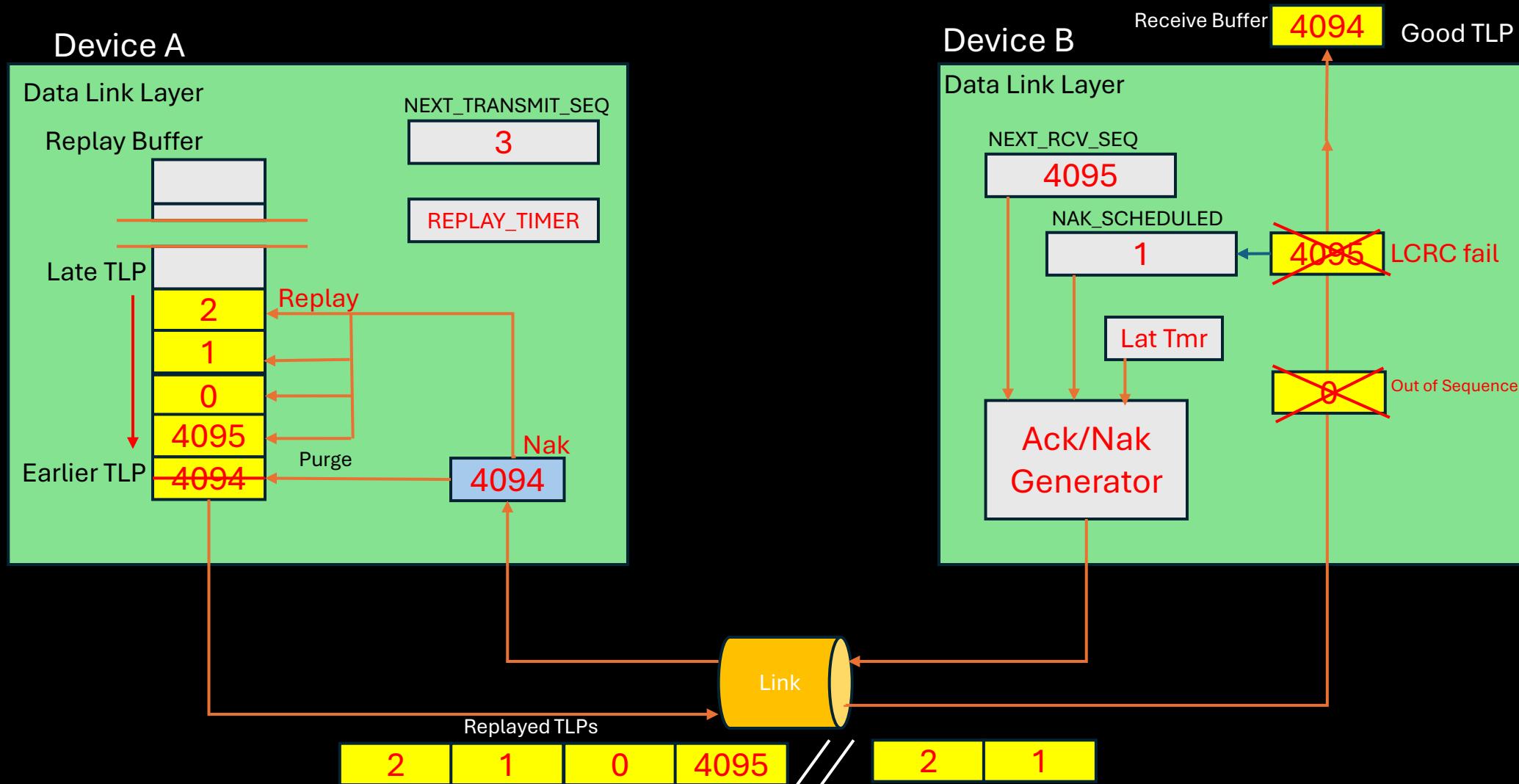
Ack Example



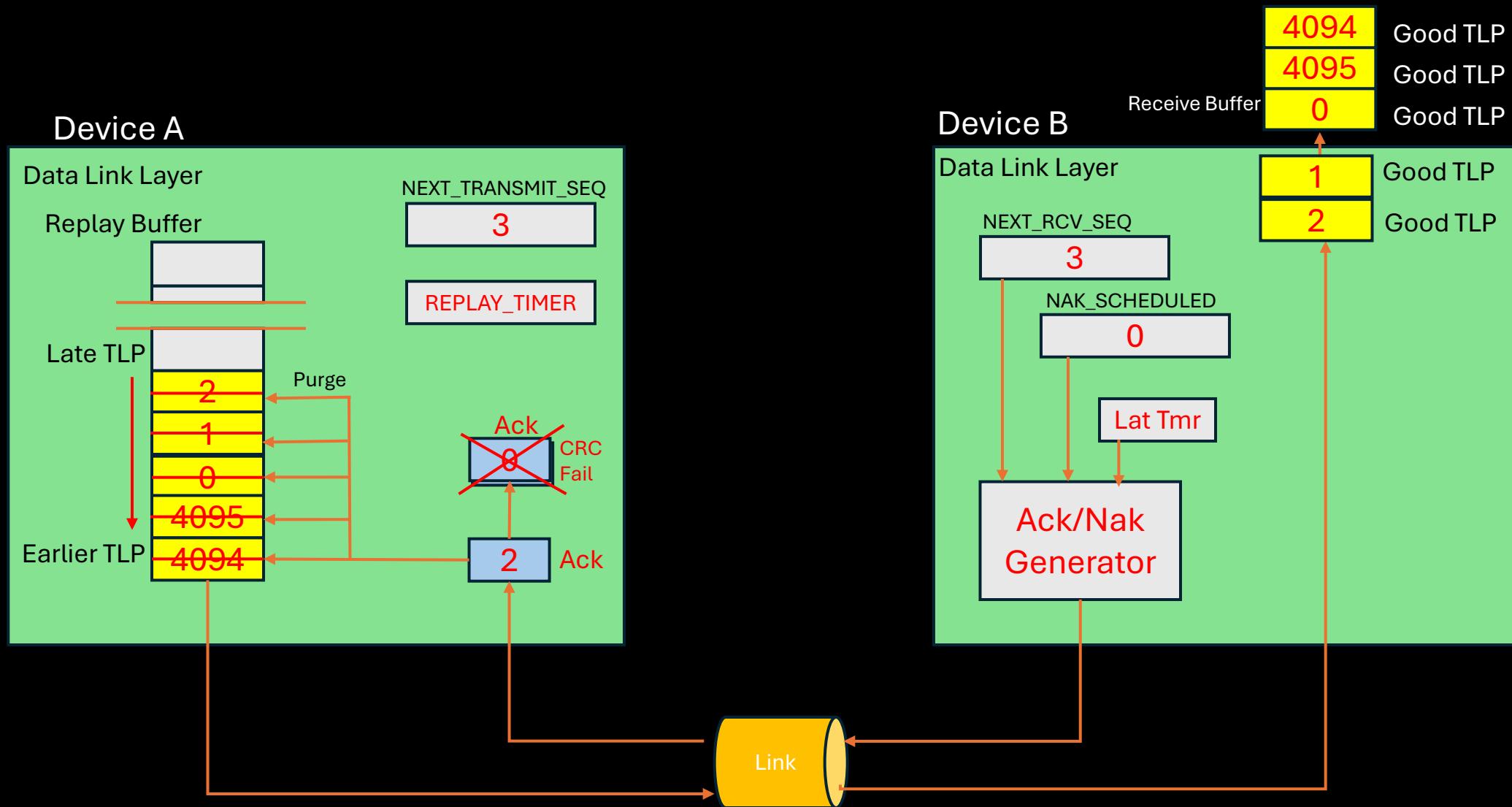
Ack with Sequence Number Rollover



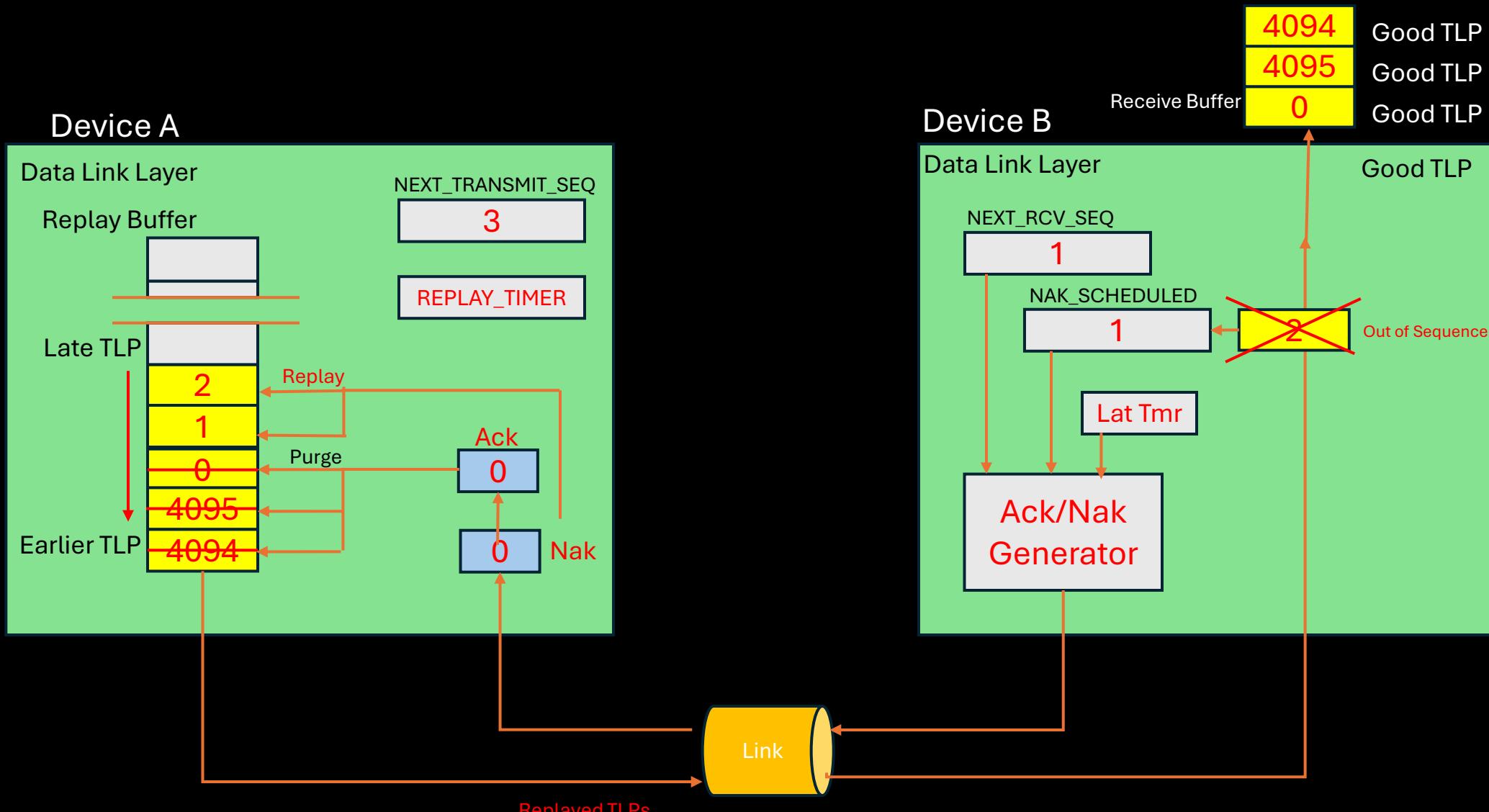
Nak Example



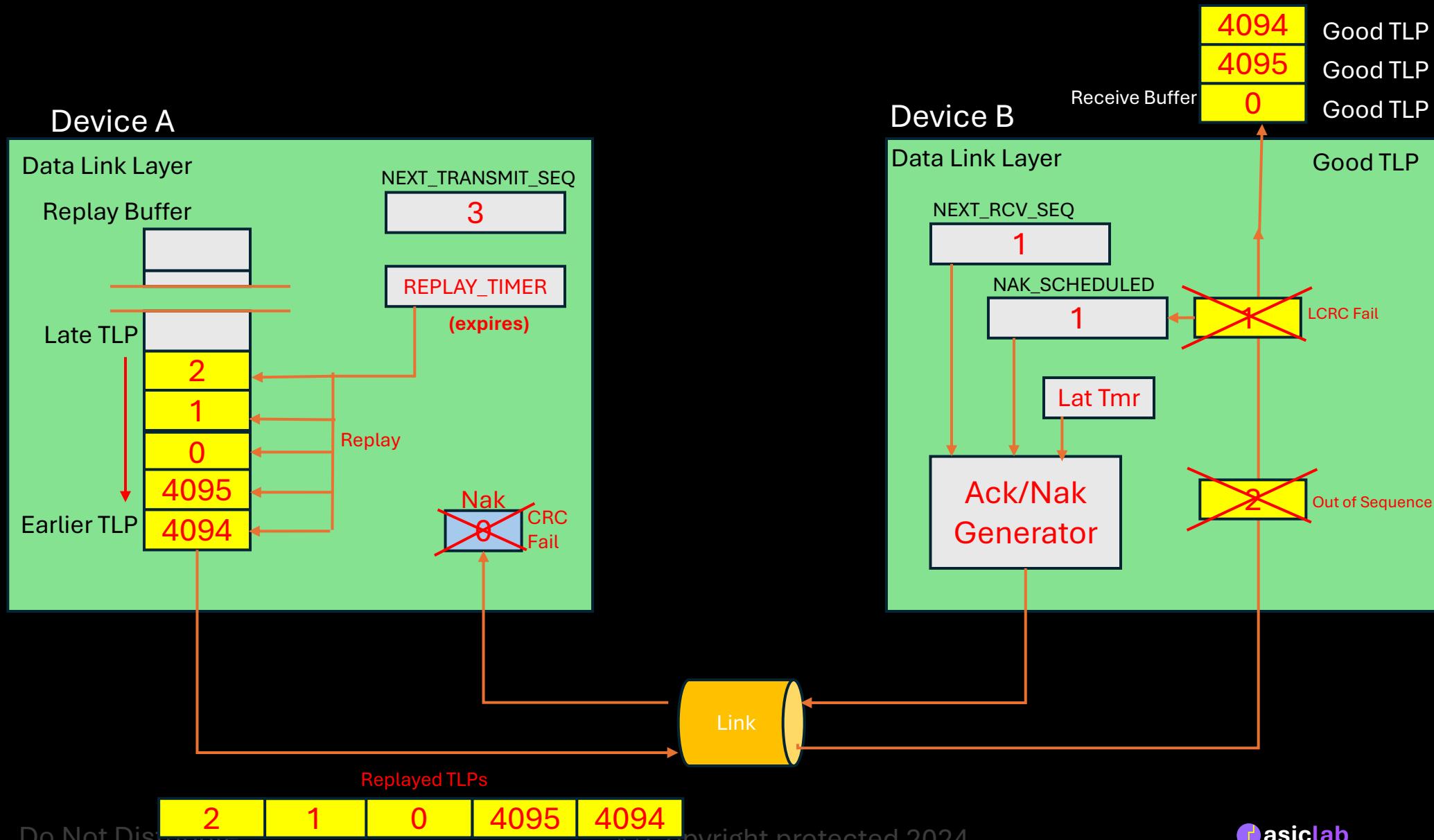
Failed ACK



Lost or Corrupt TLP



Lost or Corrupt Nak

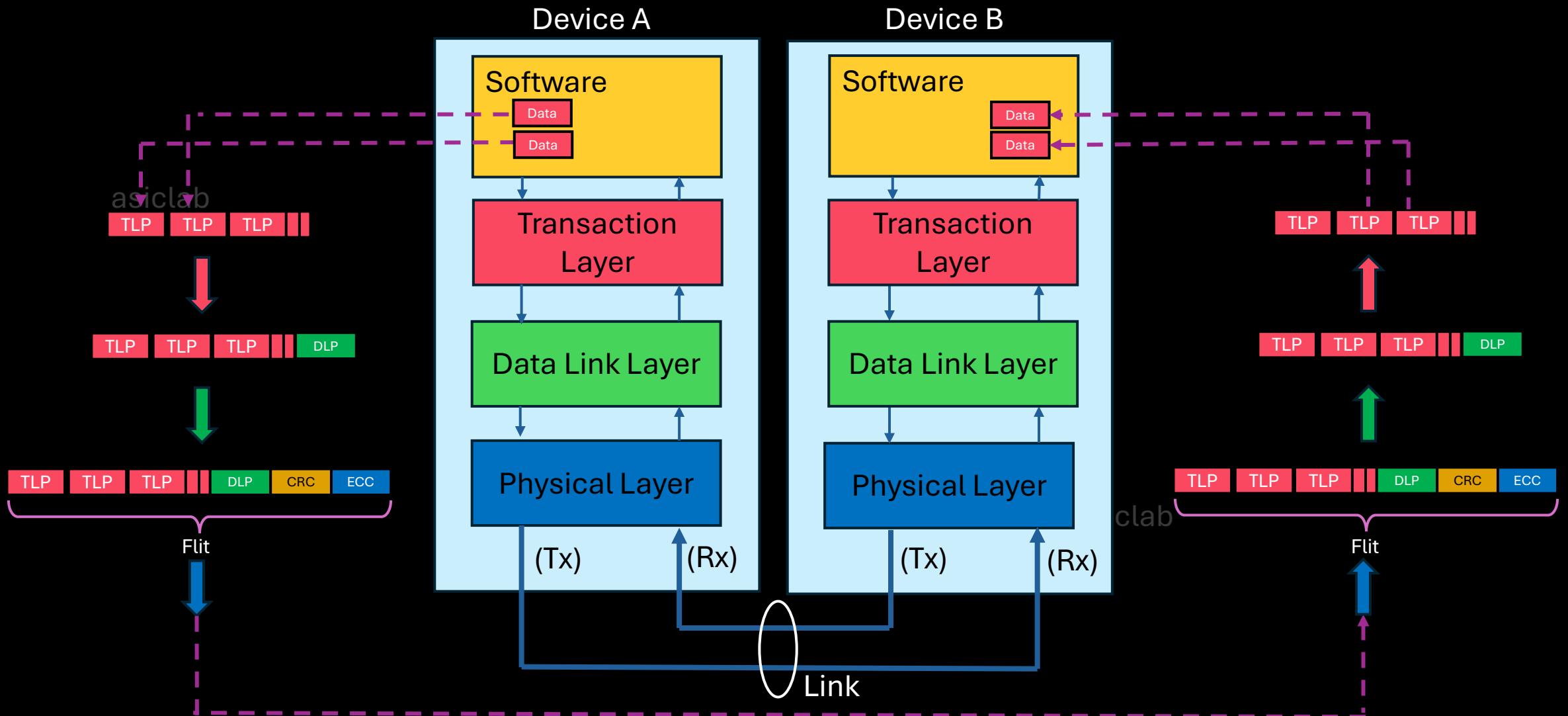


asiclab

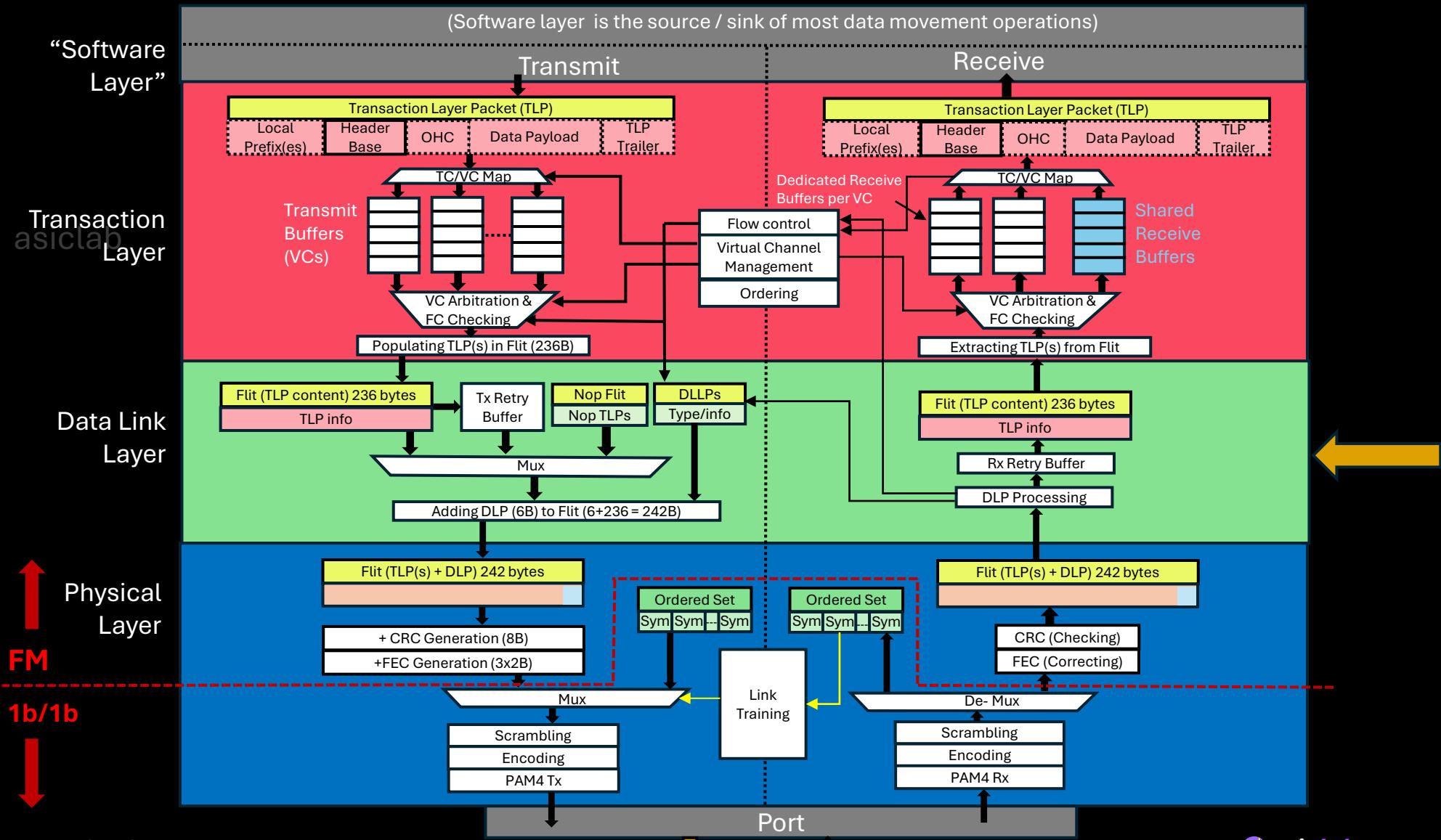
PCIe 6.0 Data Link Layer

asiclab

Flit Assembly/Disassembly (New with PCIe 6.0)



PCIe Device Layer Details 6.0 (Flit Mode – FM) (1b/1b)



Do Not Distribute

© Copyright protected 2024

Types of Flits

- ❖ **IDLE Flit** -> all the contents are 0 in the TLP portion and the DLP portion will have the sequence number 0. They are only sent just before entering to L0 Entering L0 can be done from config and from recovery, in both these states just before entering L0 there is a state called config.idle and recovery.idle. In Non flit mode we will be sending Logical idles in that state before entering L0. In case of FM this is not available, instead we will send IDLE Flit
- ❖ **NOP Flit** ->in case of this all the content of TLP portion is 0, but DLP can be nonzero, this is sent when there are no TLP to send so these are sent as a filler for the TLP portion
- ❖ **Payload Flit** -> TLP portion is carrying nonzero, and the sequence number is nonzero.

asiclab

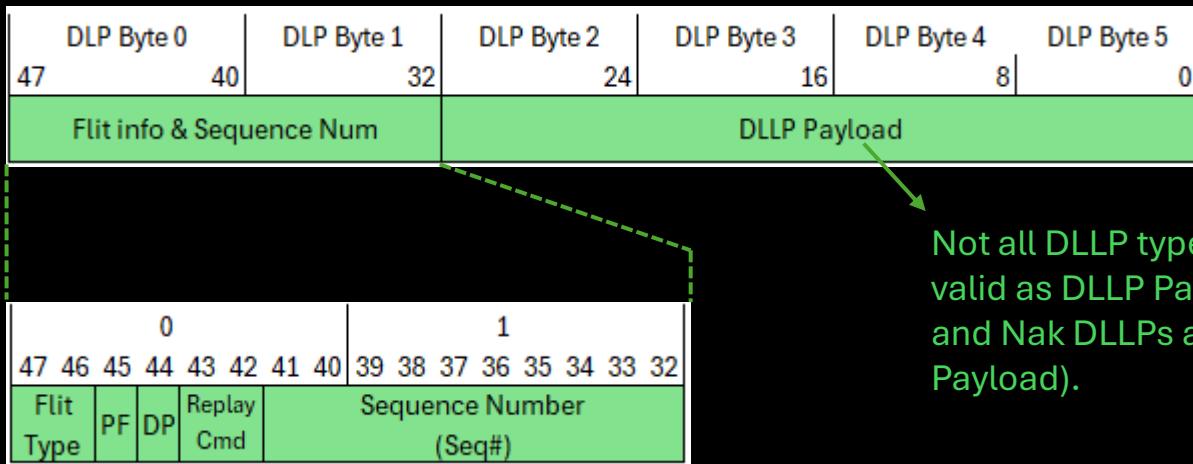
ACK NAK – Replay Mechanism

- Ack Nak mechanism usually involve generating a feedback about the last transaction and based on that either the TLP from the TX replay buffer is either flushed or replayed
- The replay can be of two types as compared to non Flit mode
 - **Standard Replay** in which the flits are replayed from the buffer based on the NAK received from the RX and the entire content of the Tx Buffer will be replayed. This may resent the TLPs that were previously send before receiving the NAK
 - **Selective Replay:** In this only the TLP that was corrupted is resent and the RX will need to implement a RX replay buffer to store the good TLPs temporarily until the lost one is received in the Replay. Selective replay can increase the bandwidth as it is not resending the good flits again

DLL Packet format – FM (New with PCIe 6.0)

- DLLP Can be packed into the DLP portion based on one of the encoding field “DP”

asiclab



Not all DLLP types from prior generations are valid as DLLP Payload when in FM (e.g. Ack and Nak DLLPs are not supported in the DLLP Payload).

asiclab

DP: DLLP Payload (0b DLLP Payload in DLP 2..5, 1b Optimized_Update_FC or Flit_Marker in DLP 2..5)

PF: Prior Flit (0b NOP of IDLE flit, 1b Payload Flit)

Flit Type: Flit Usage (00b IDLE Flit or NOP Flit, 01b Payload Flit, Others Reserved)

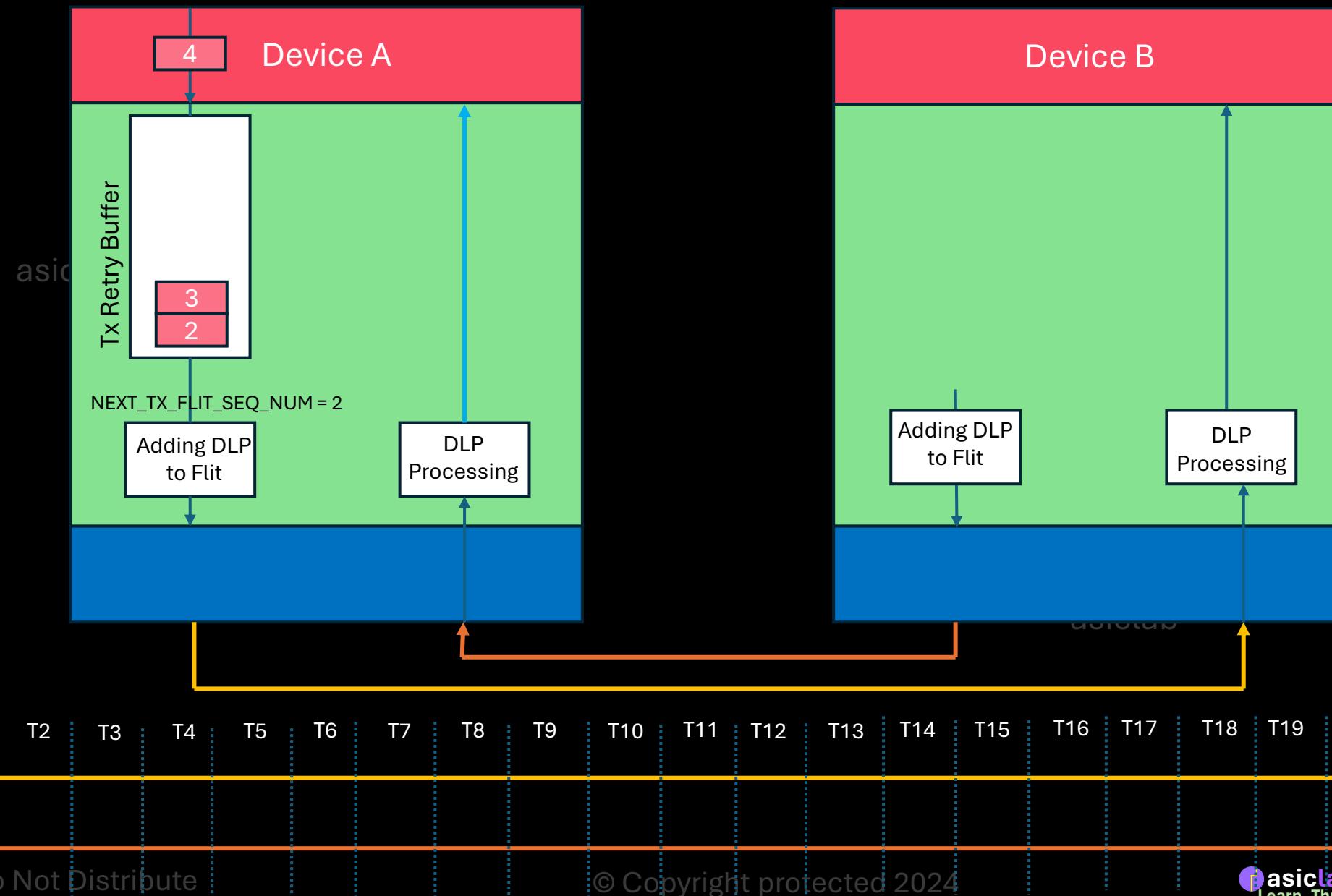
RC: Replay Command (01b ACK, 10 NAK of all unacknowledged flits, 11b NAK for single Flit)

asiclab

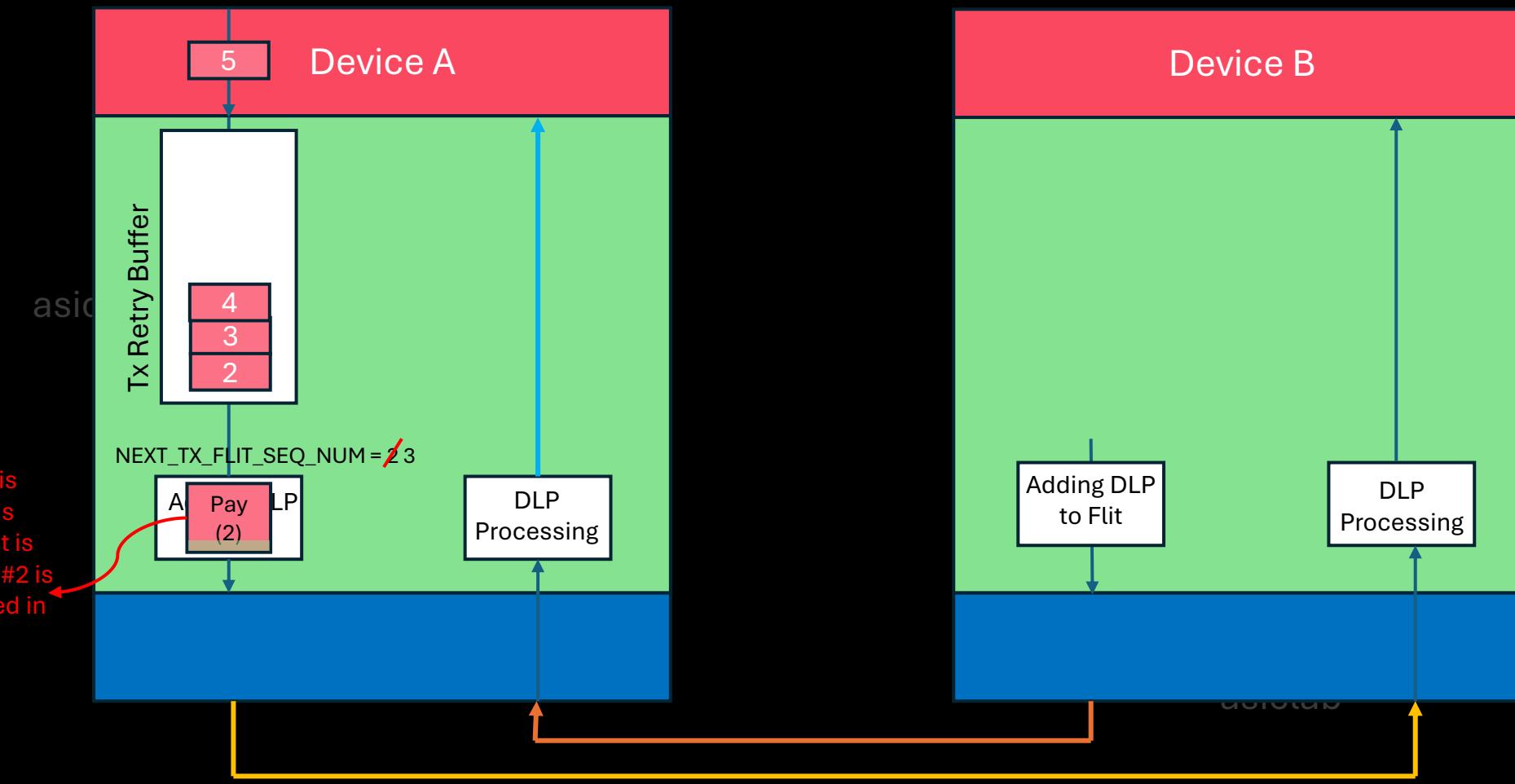
Standard ACK NAK Mechanism

asiclab

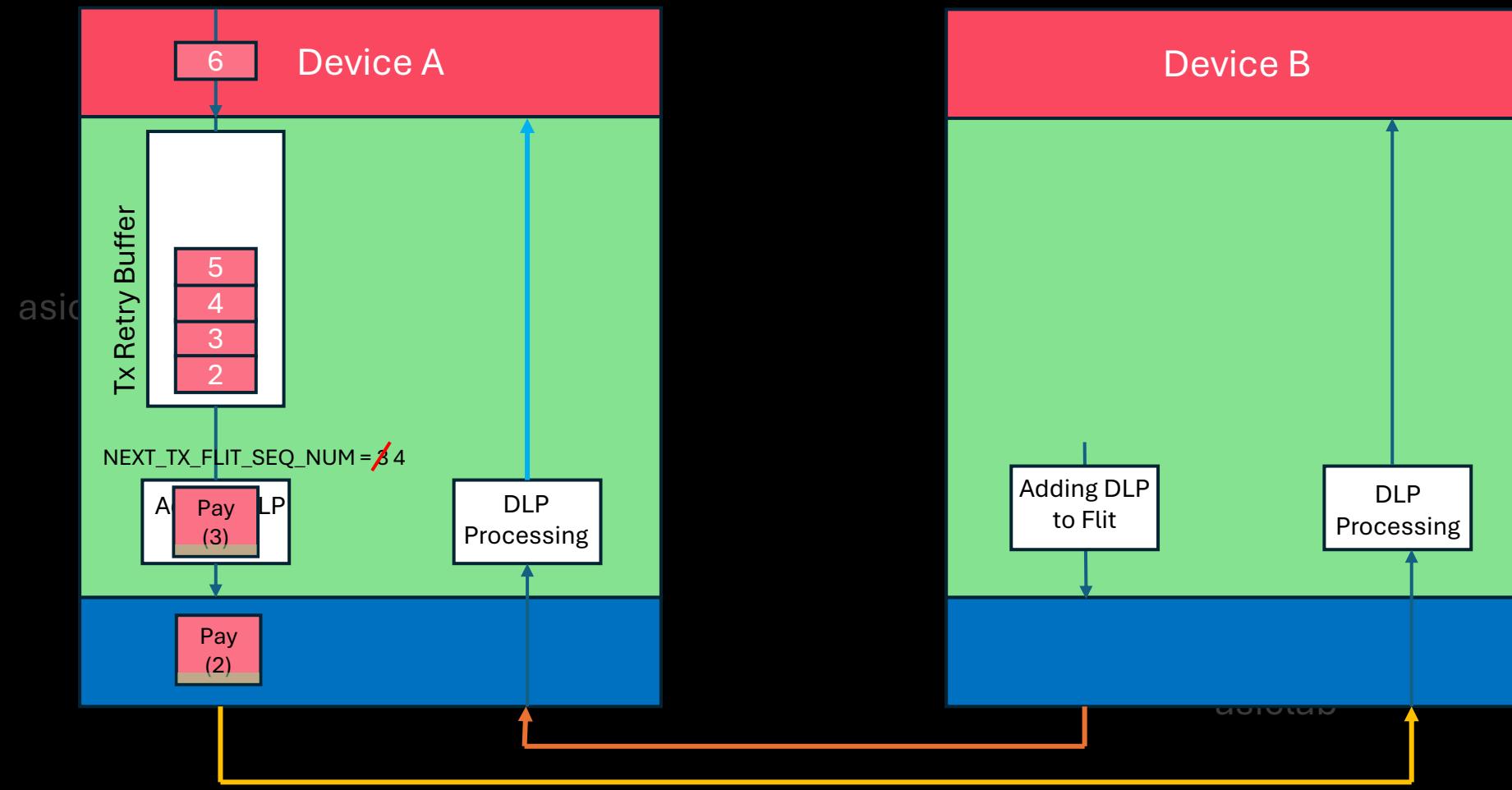
Standard ACK NAK Mechanism



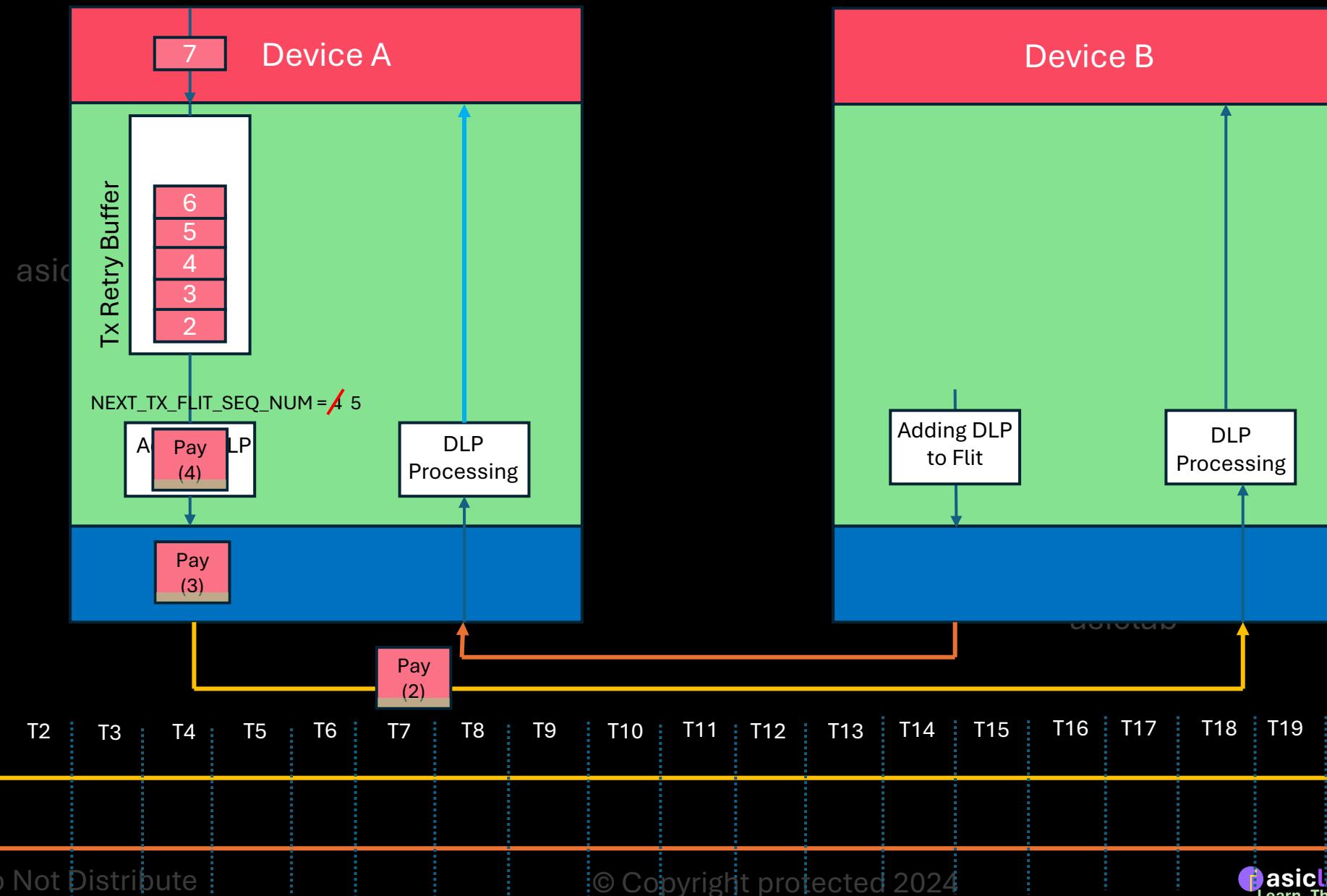
Standard ACK NAK Mechanism



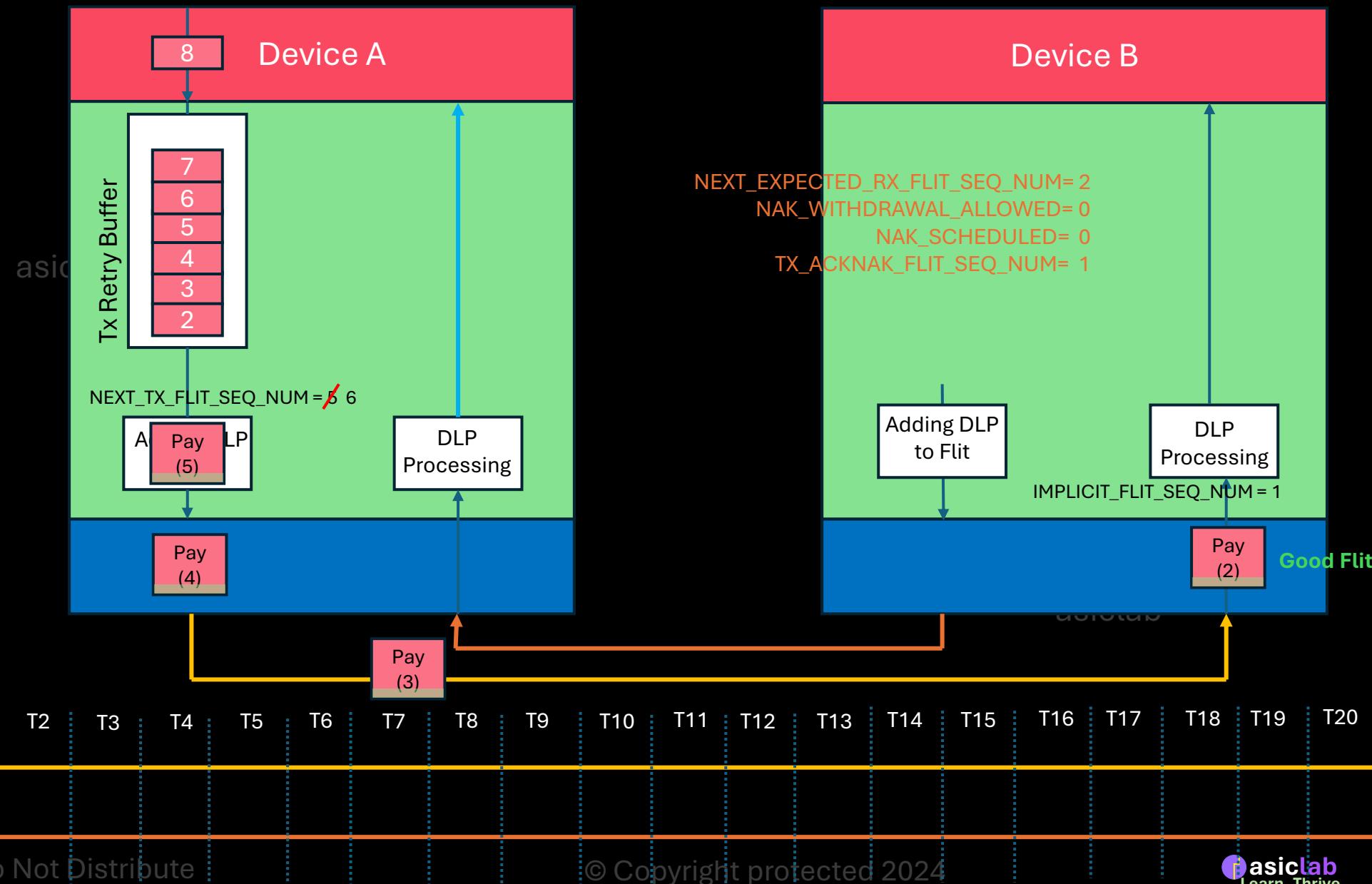
Standard ACK NAK Mechanism



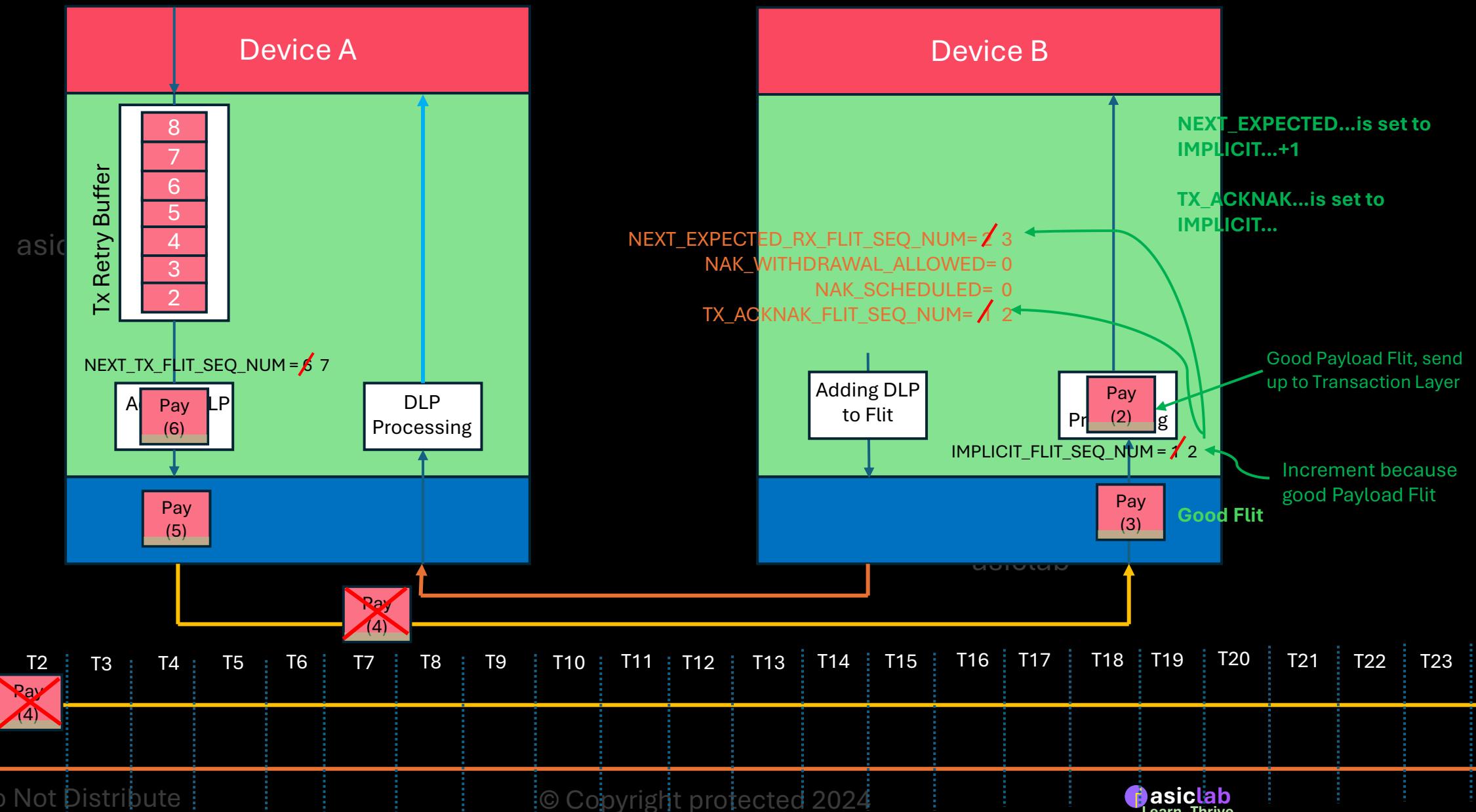
Standard ACK NAK Mechanism



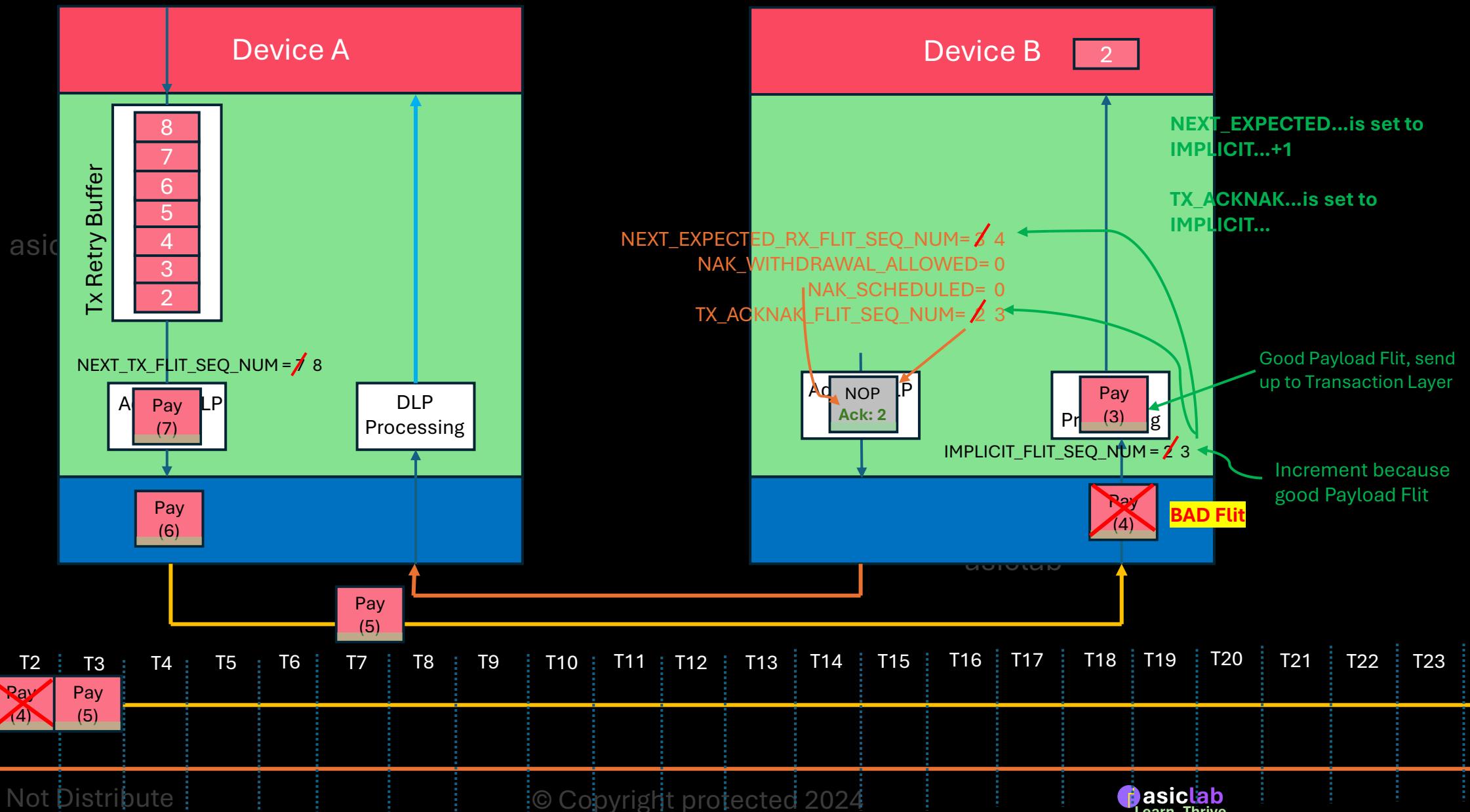
Standard ACK NAK Mechanism



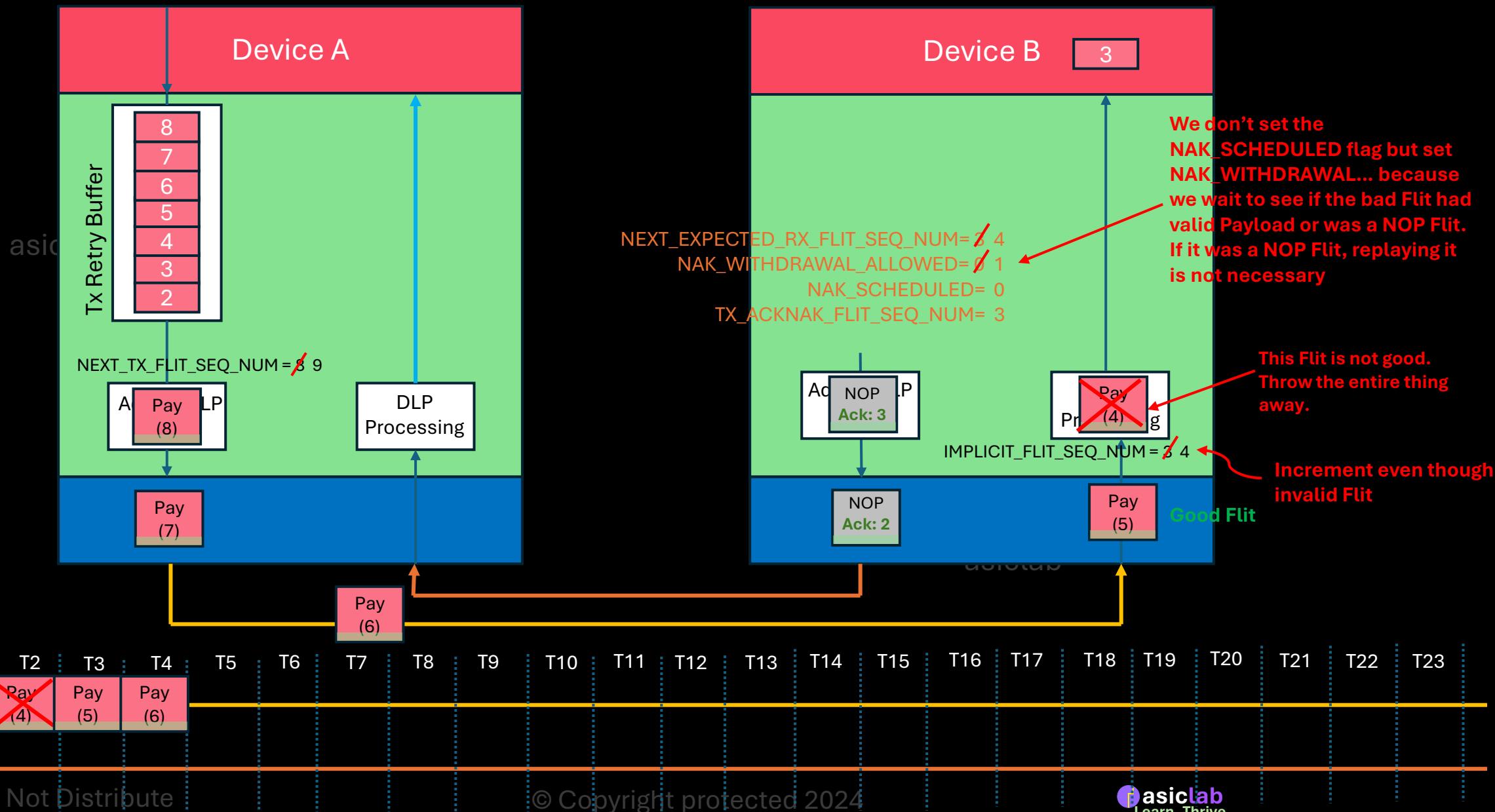
Standard ACK NAK Mechanism



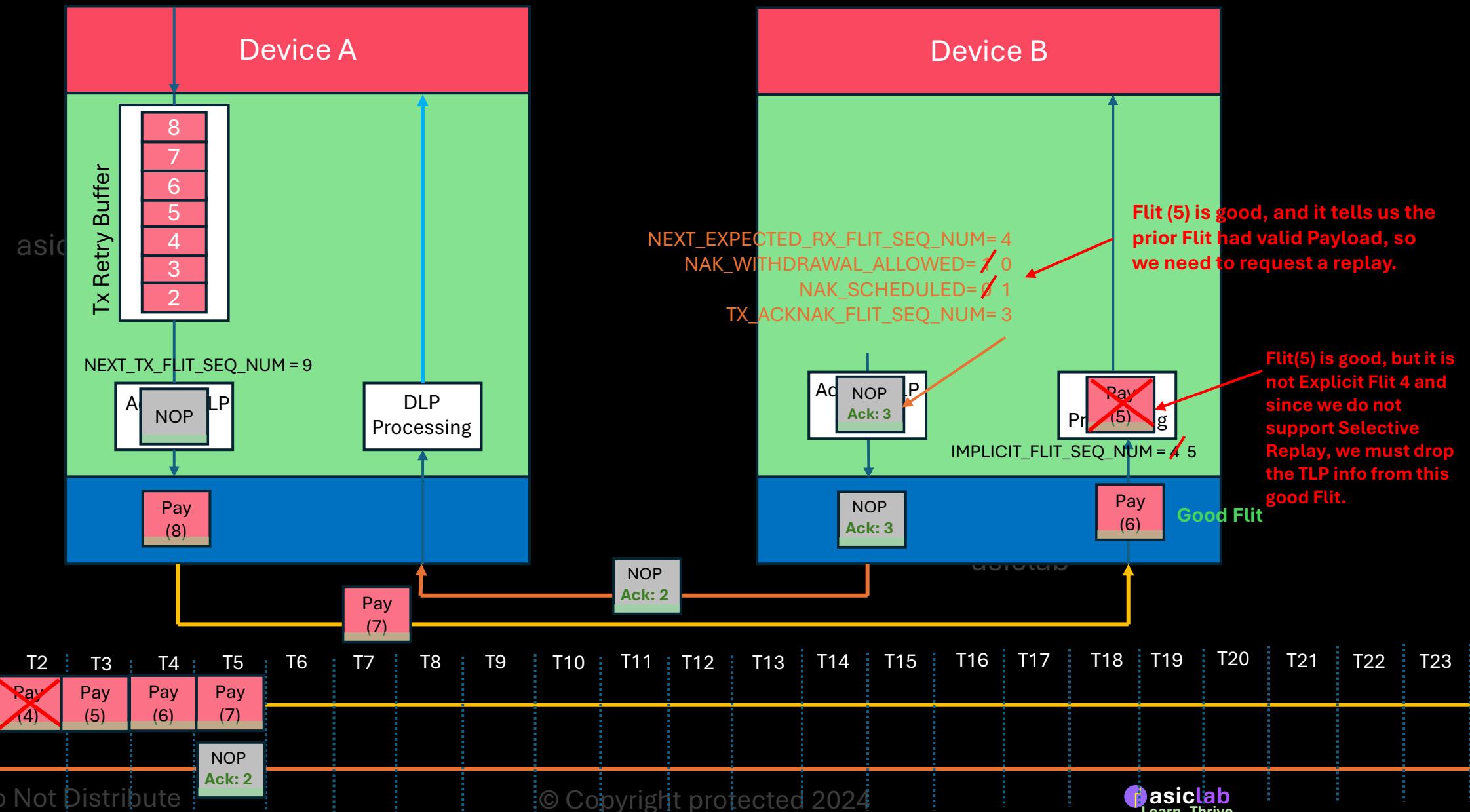
Standard ACK NAK Mechanism



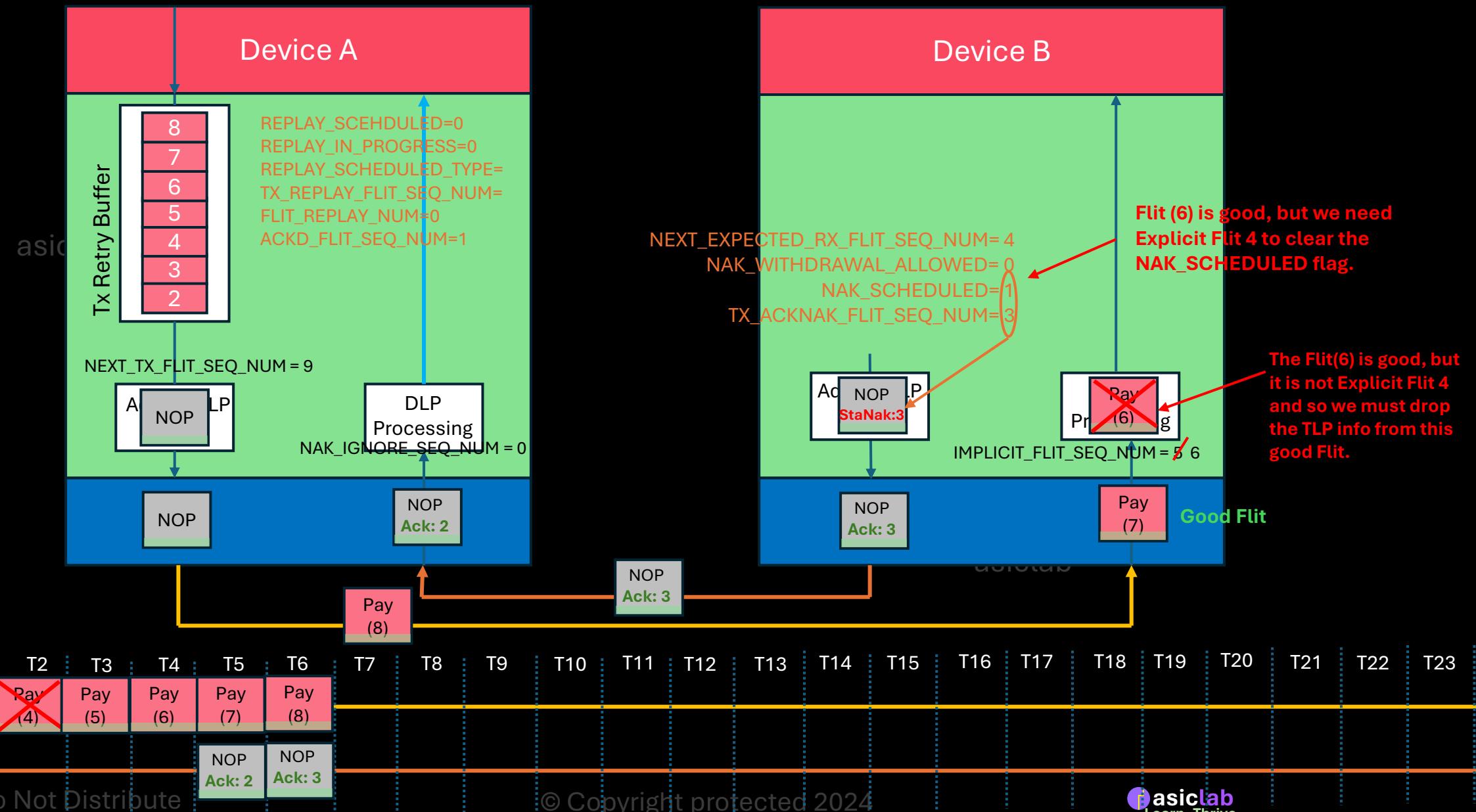
Standard ACK NAK Mechanism



Standard ACK NAK Mechanism



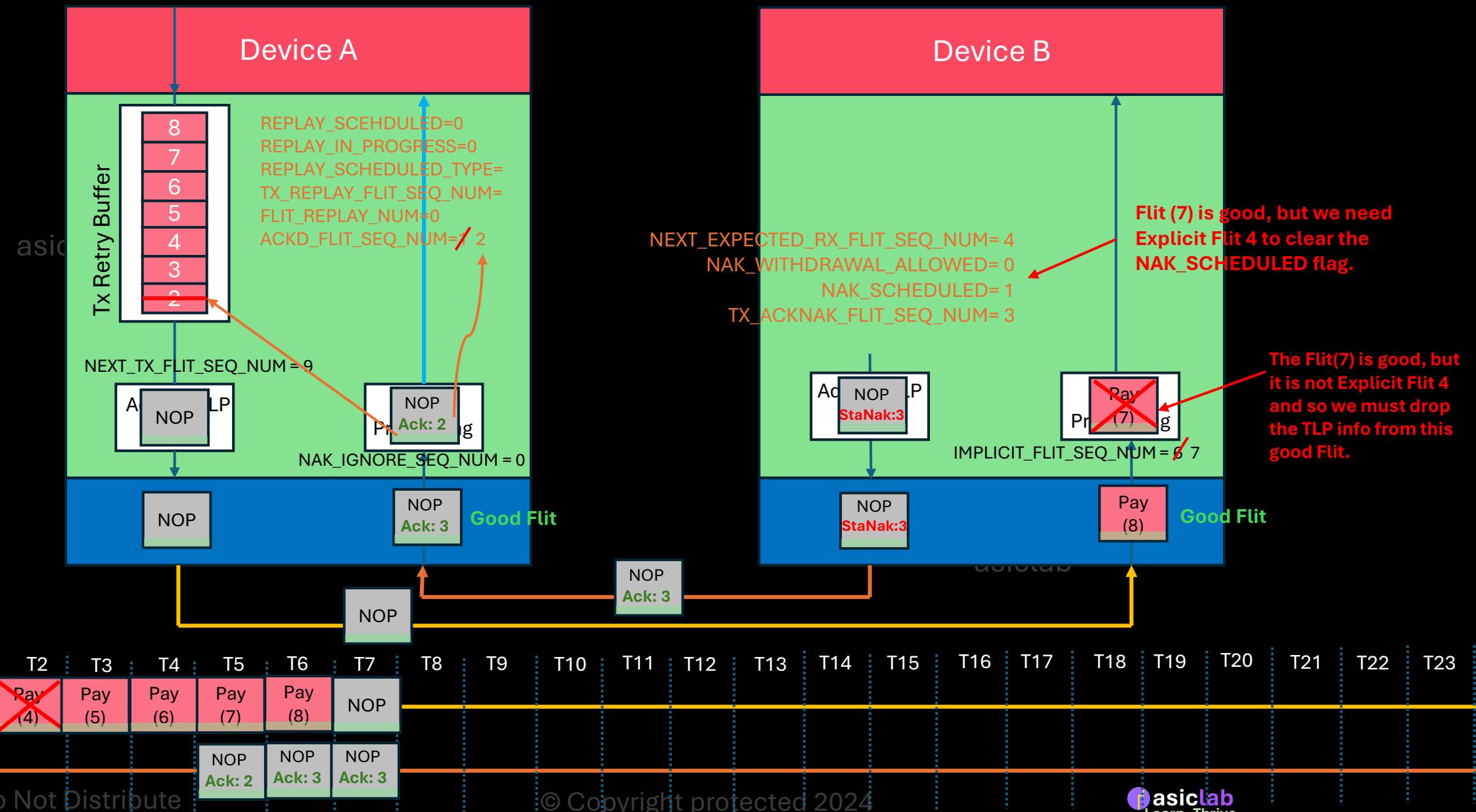
Standard ACK NAK Mechanism



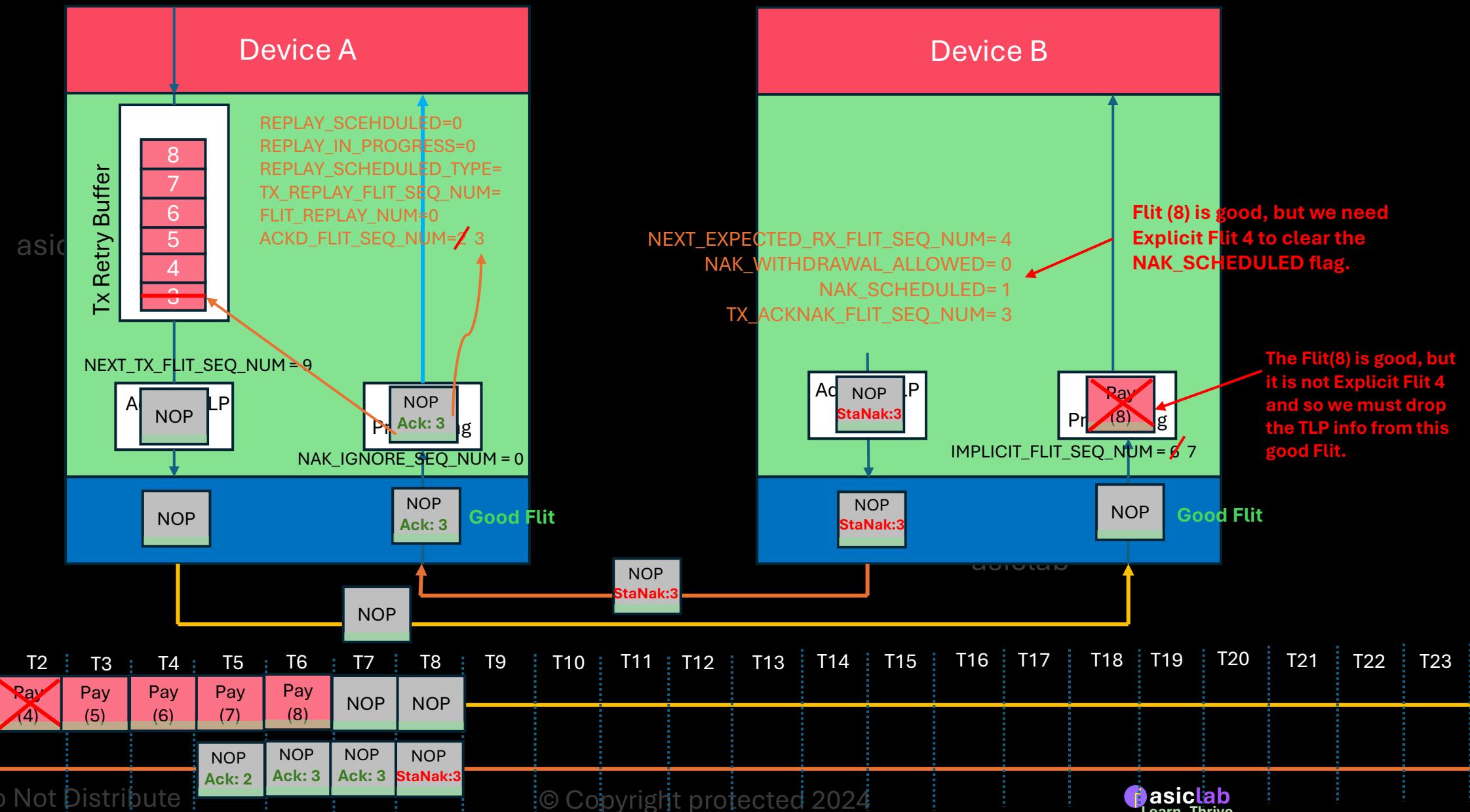
Do Not Distribute

© Copyright protected 2024

Standard ACK NAK Mechanism

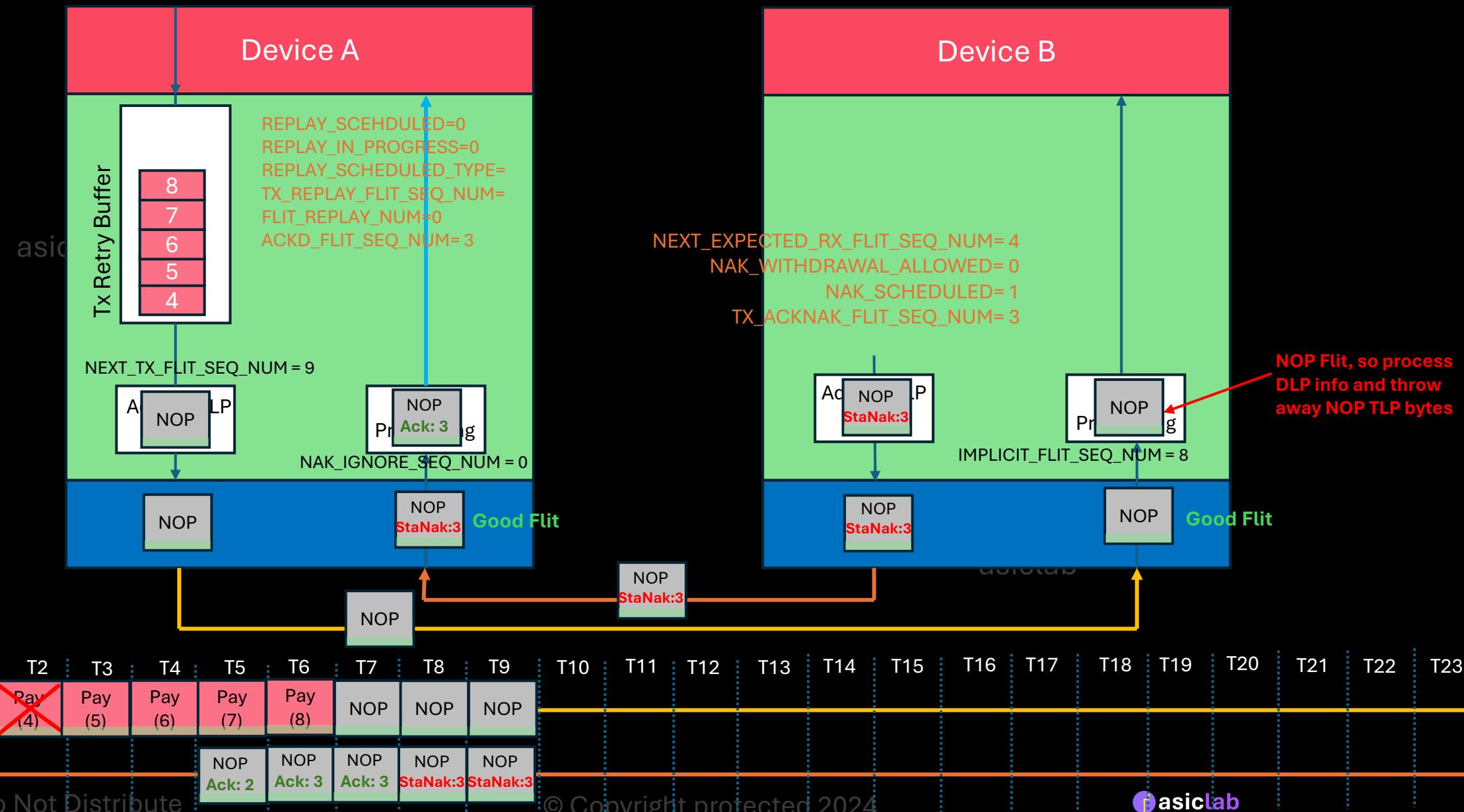


Standard ACK NAK Mechanism

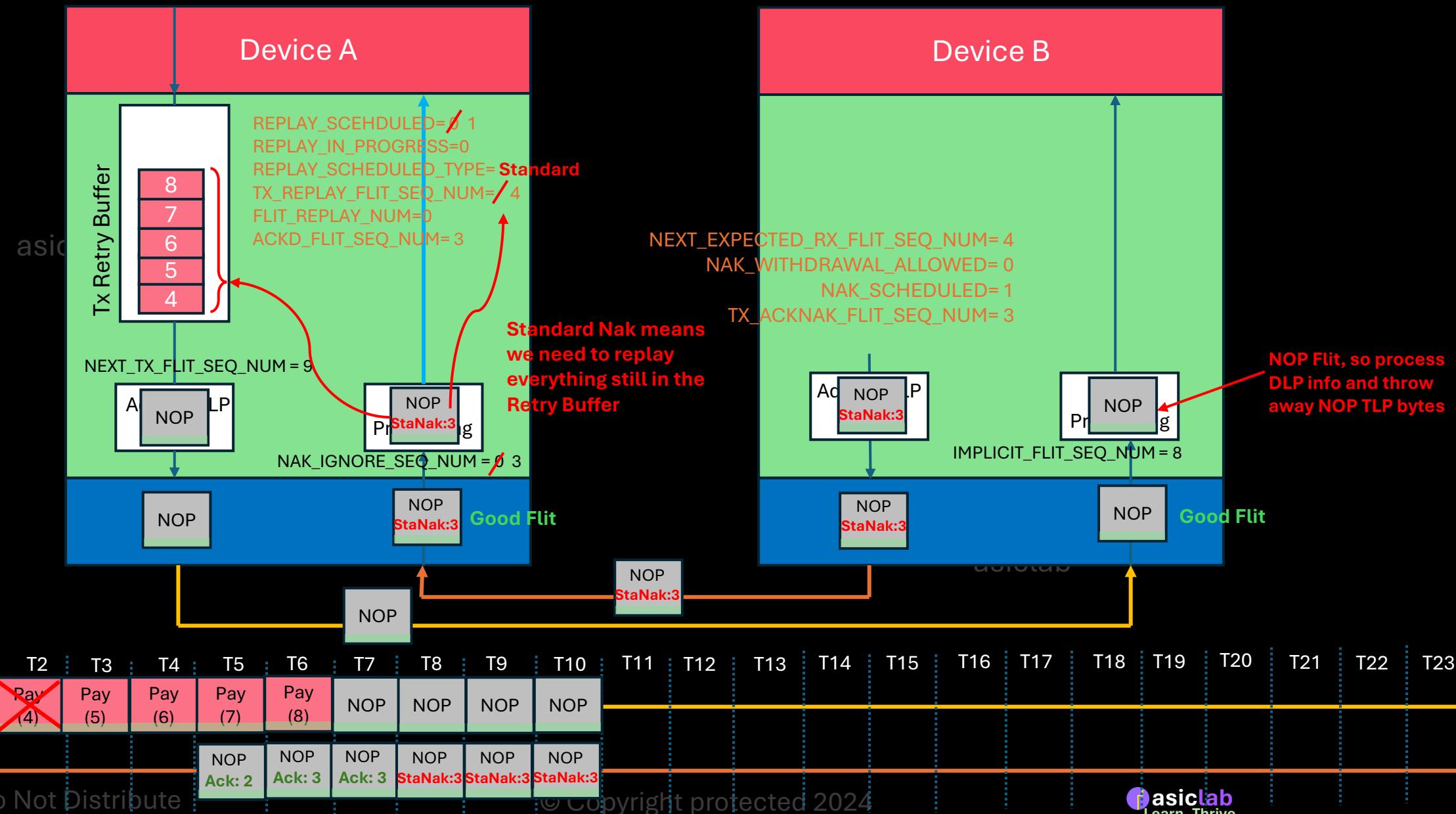


Standard ACK NAK Mechanism

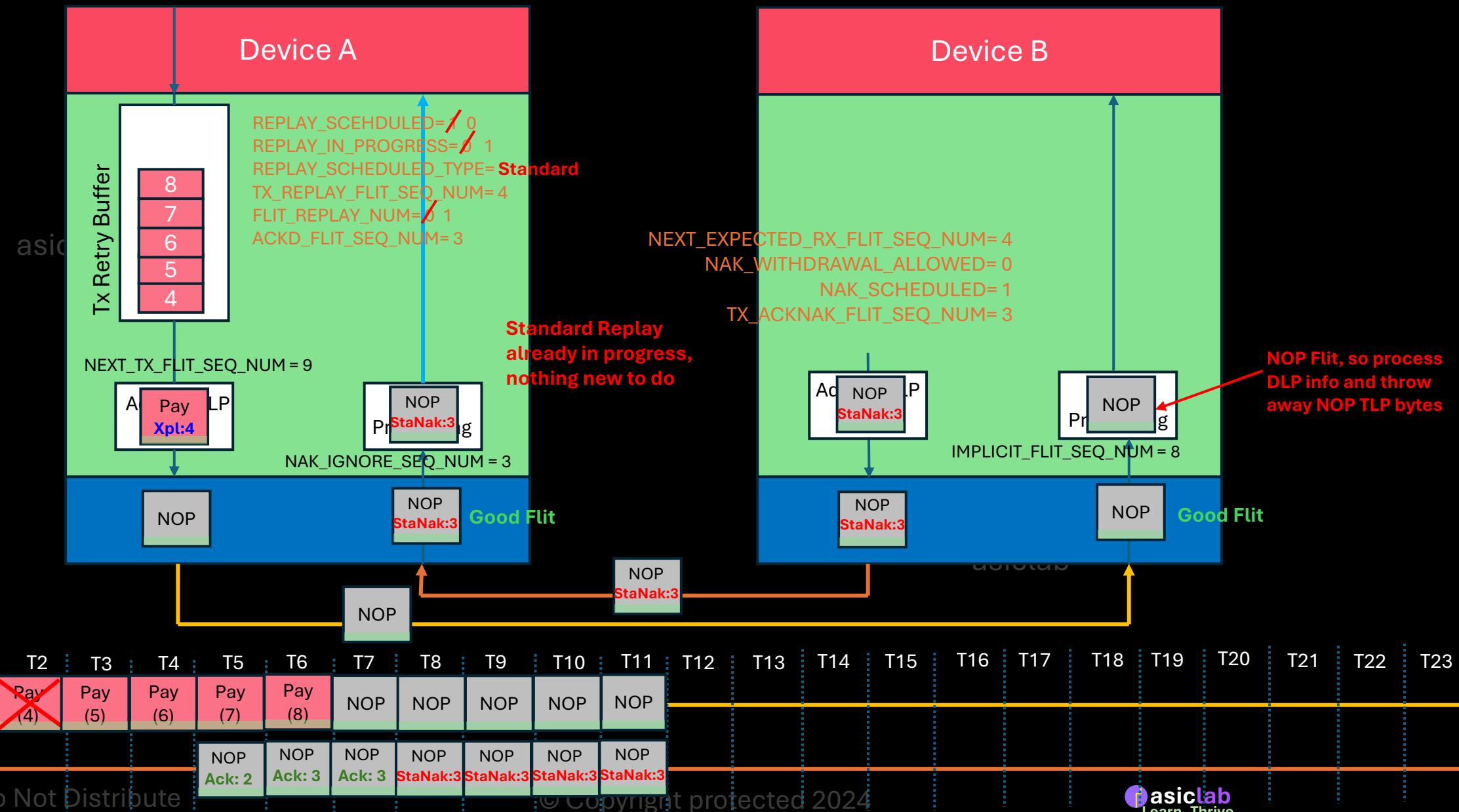
Does not support Selective Replay



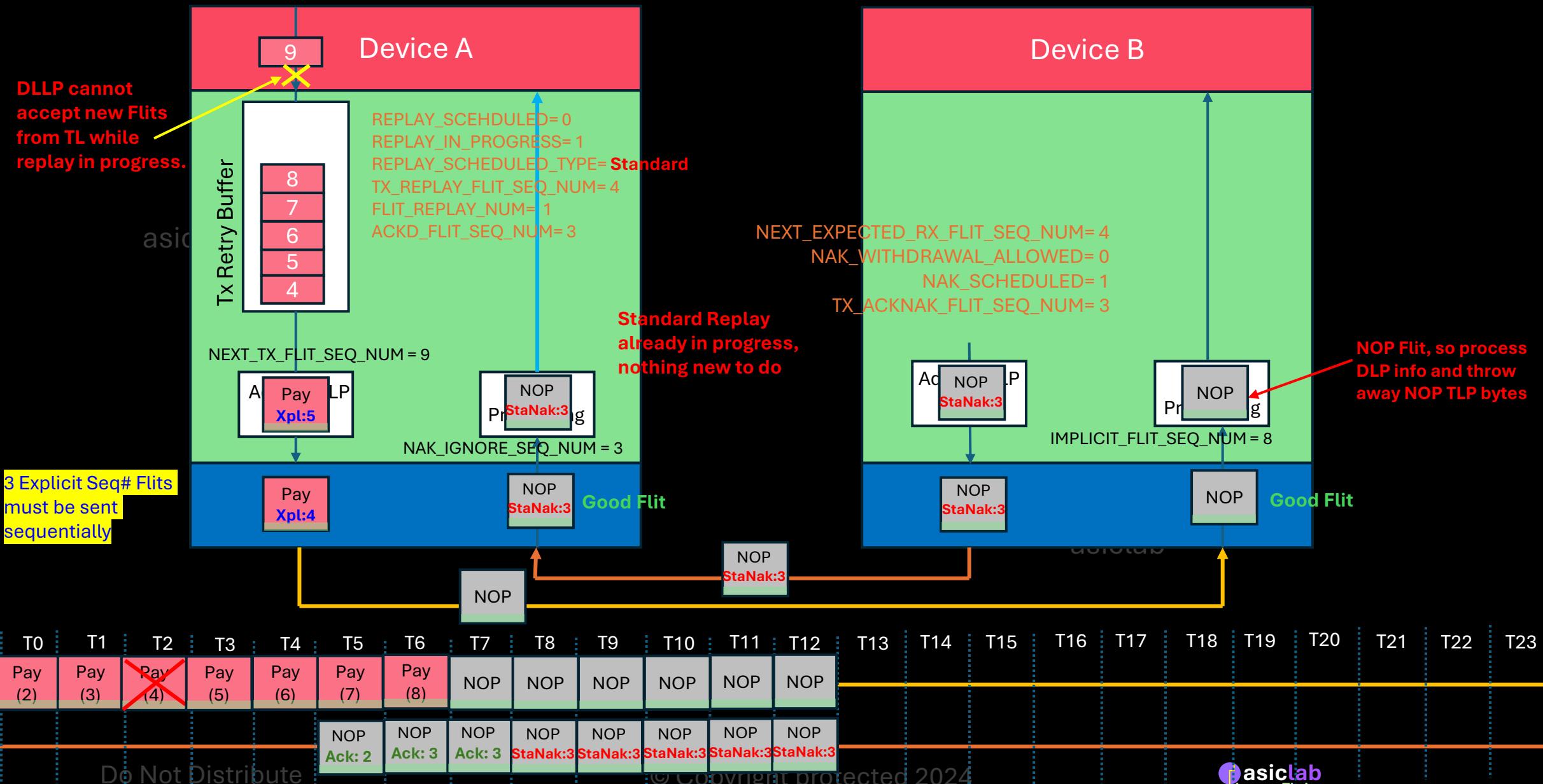
Standard ACK NAK Mechanism



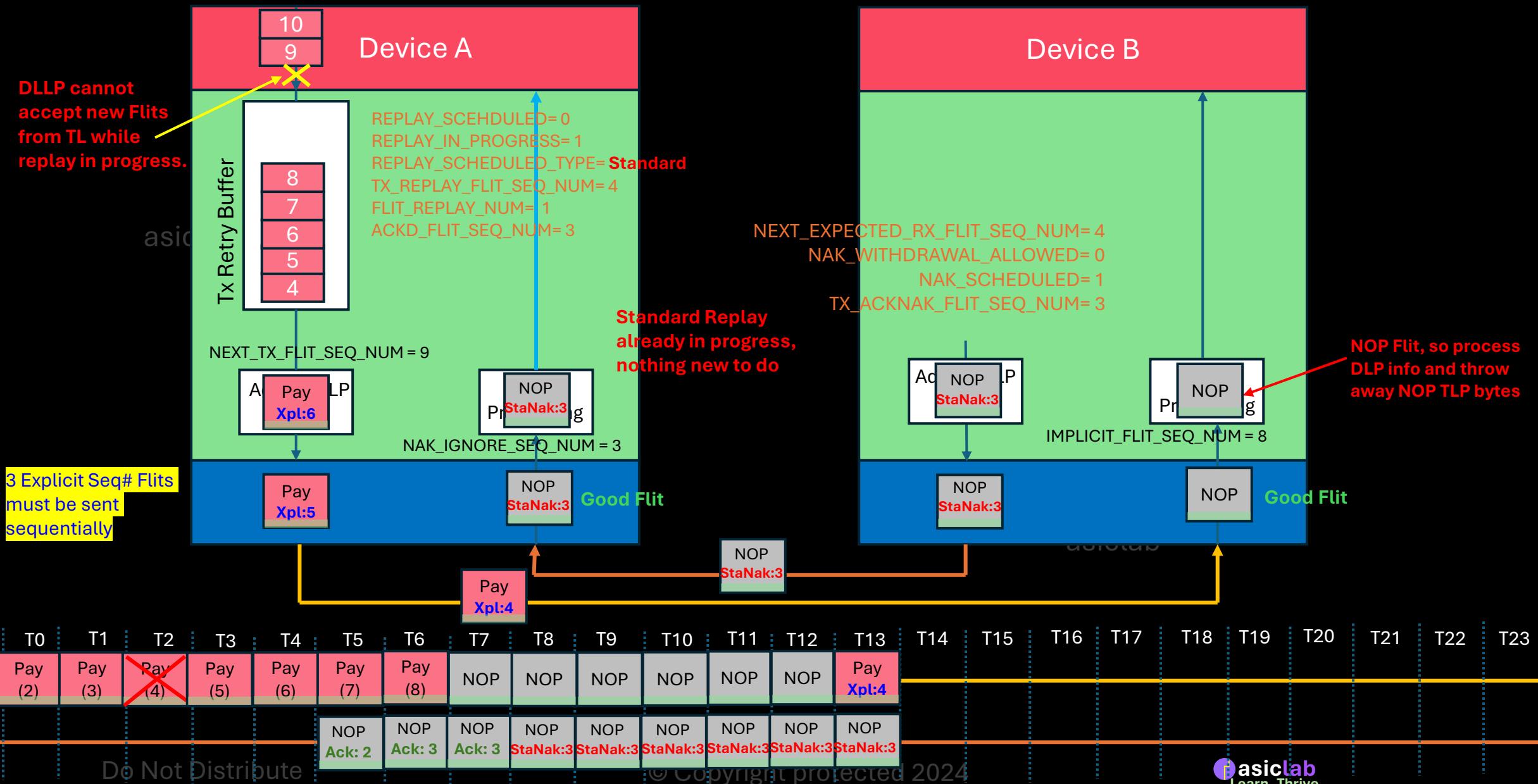
Standard ACK NAK Mechanism



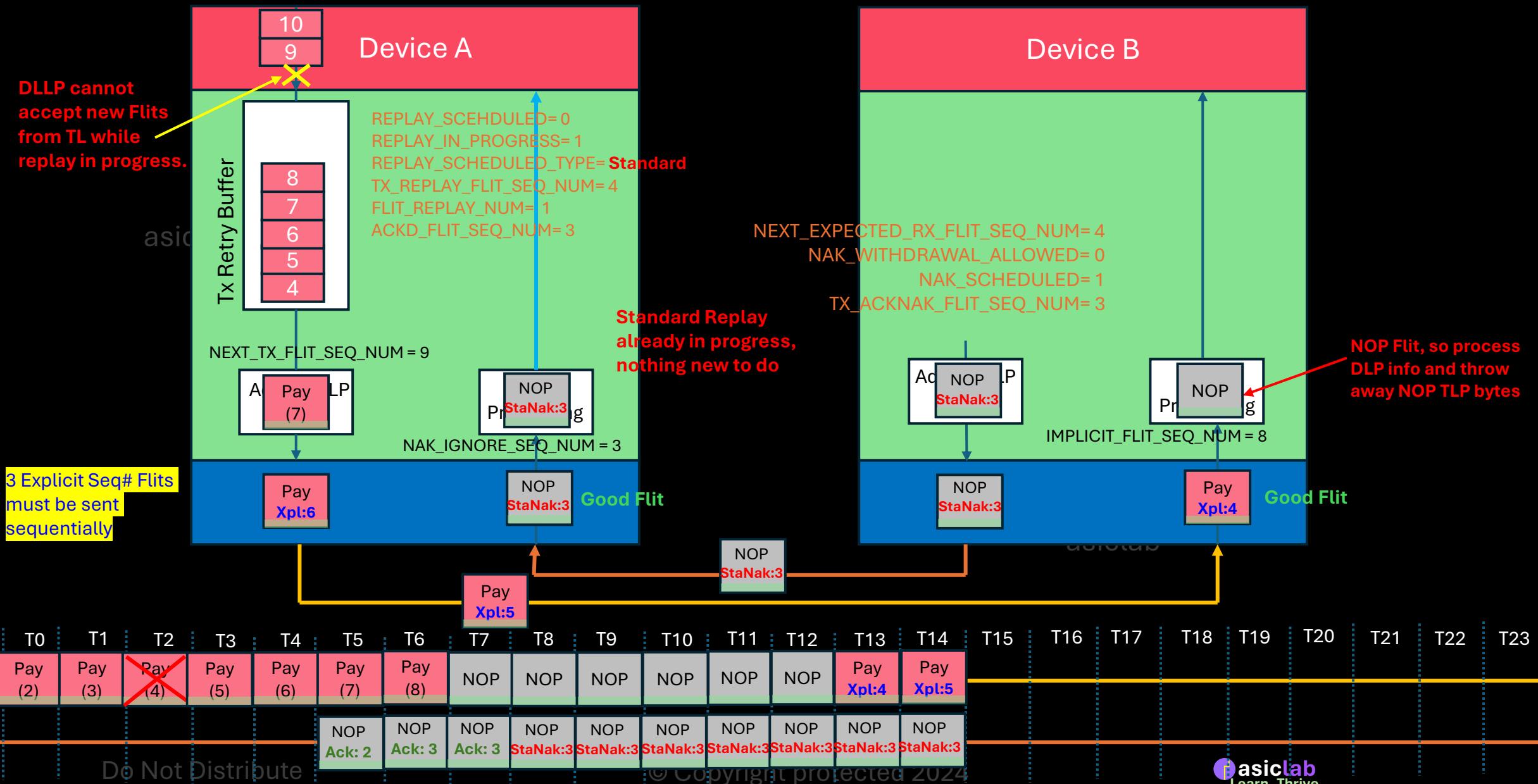
Standard ACK NAK Mechanism



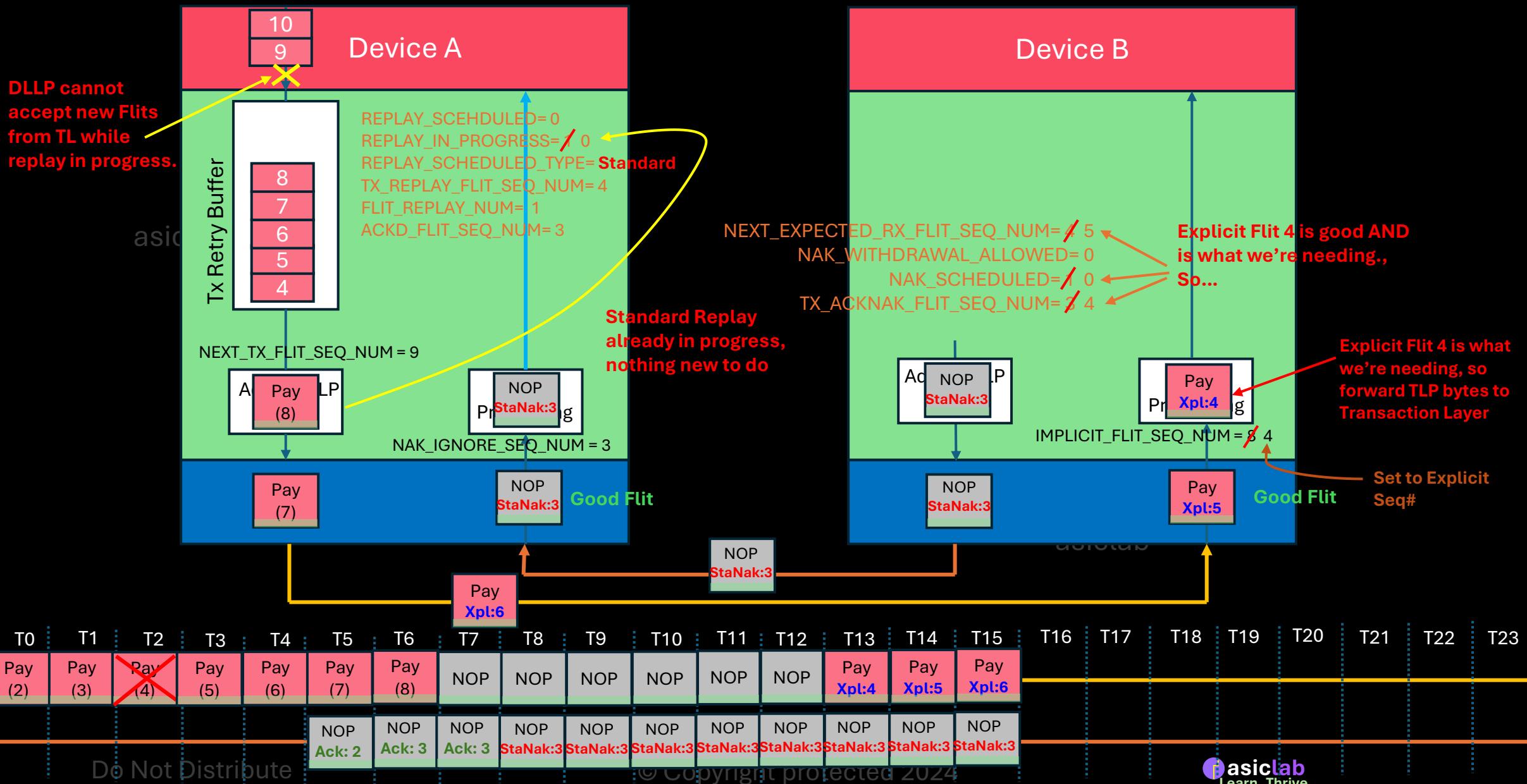
Standard ACK NAK Mechanism



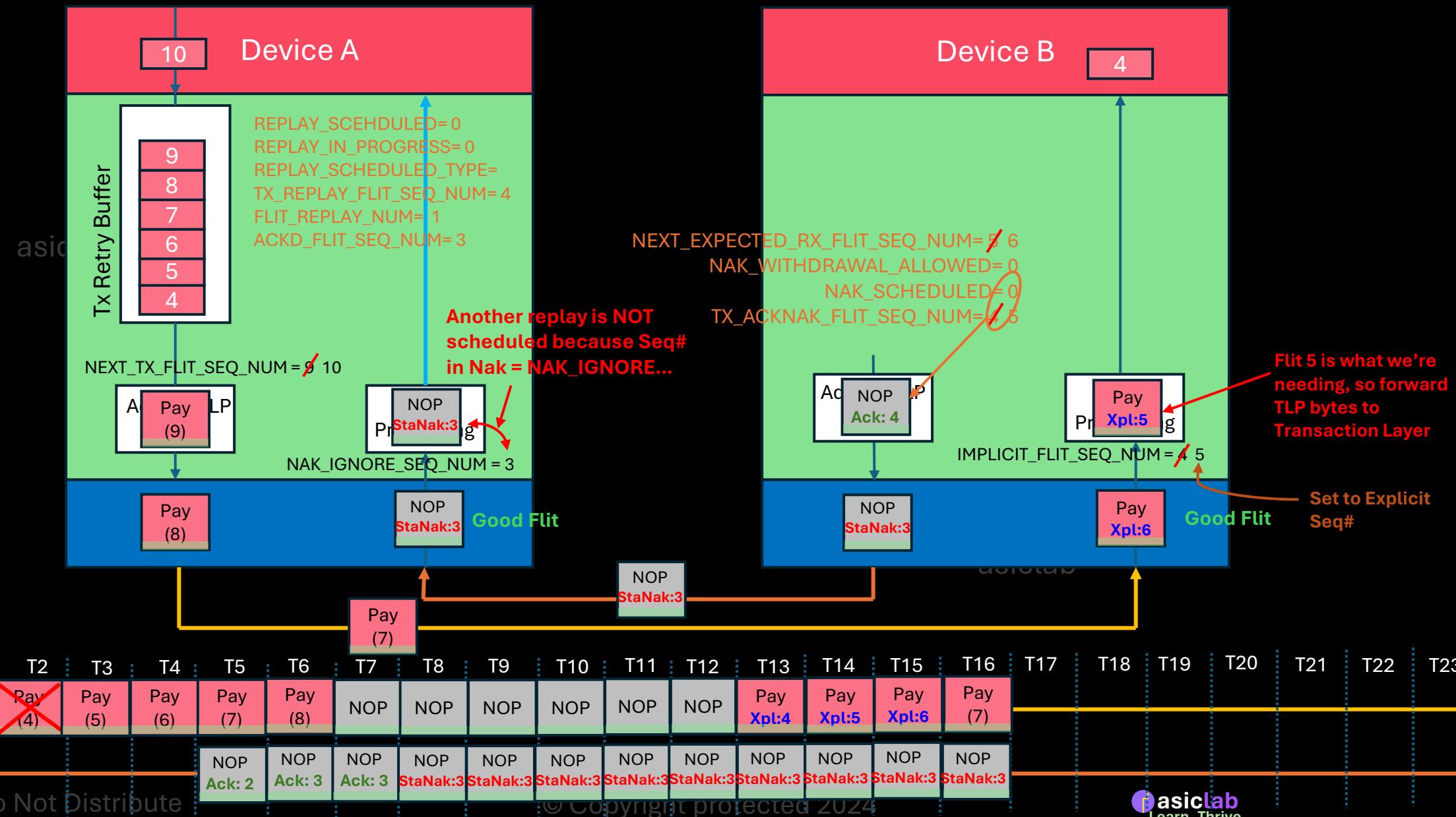
Standard ACK NAK Mechanism



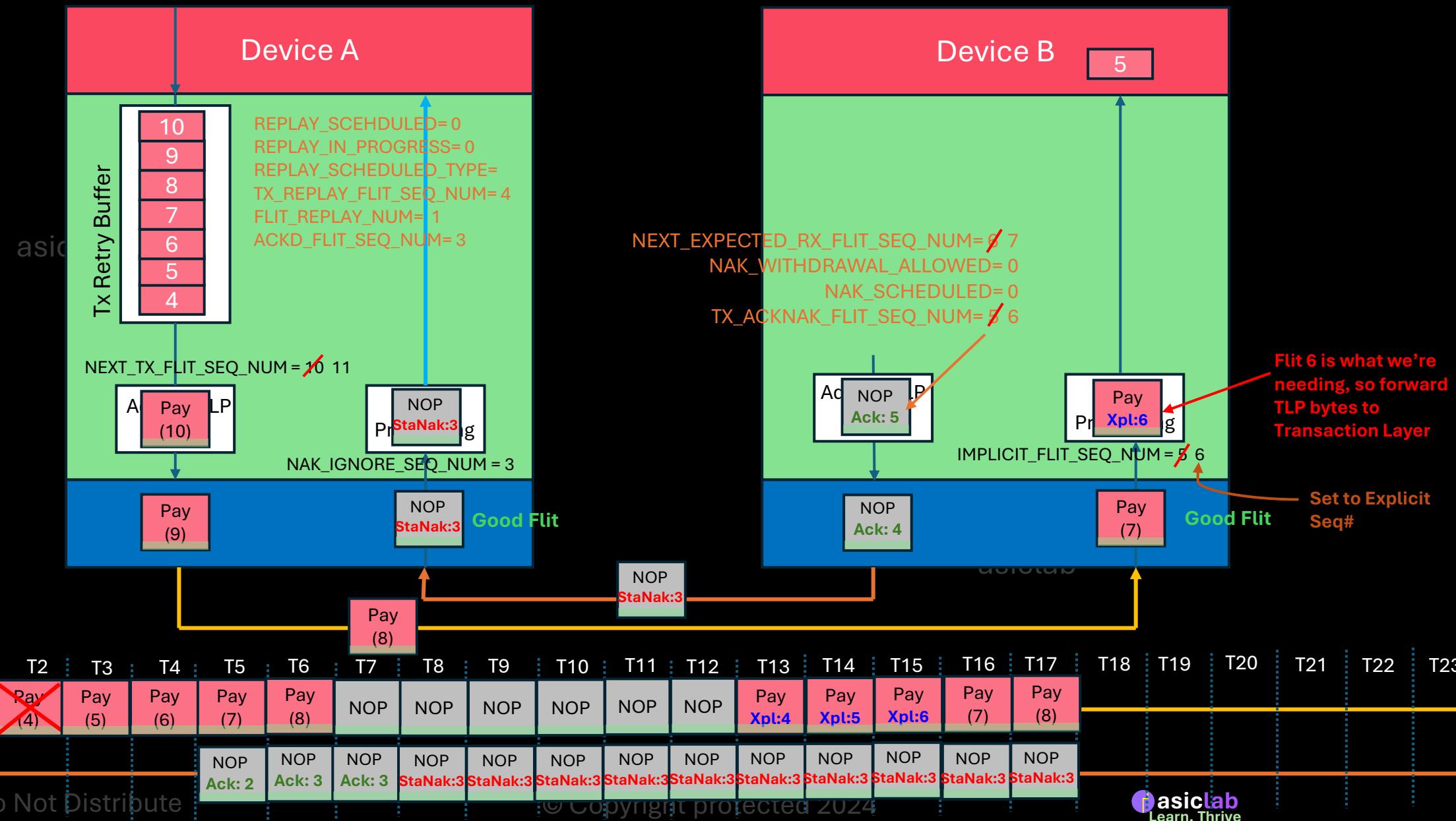
Standard ACK NAK Mechanism



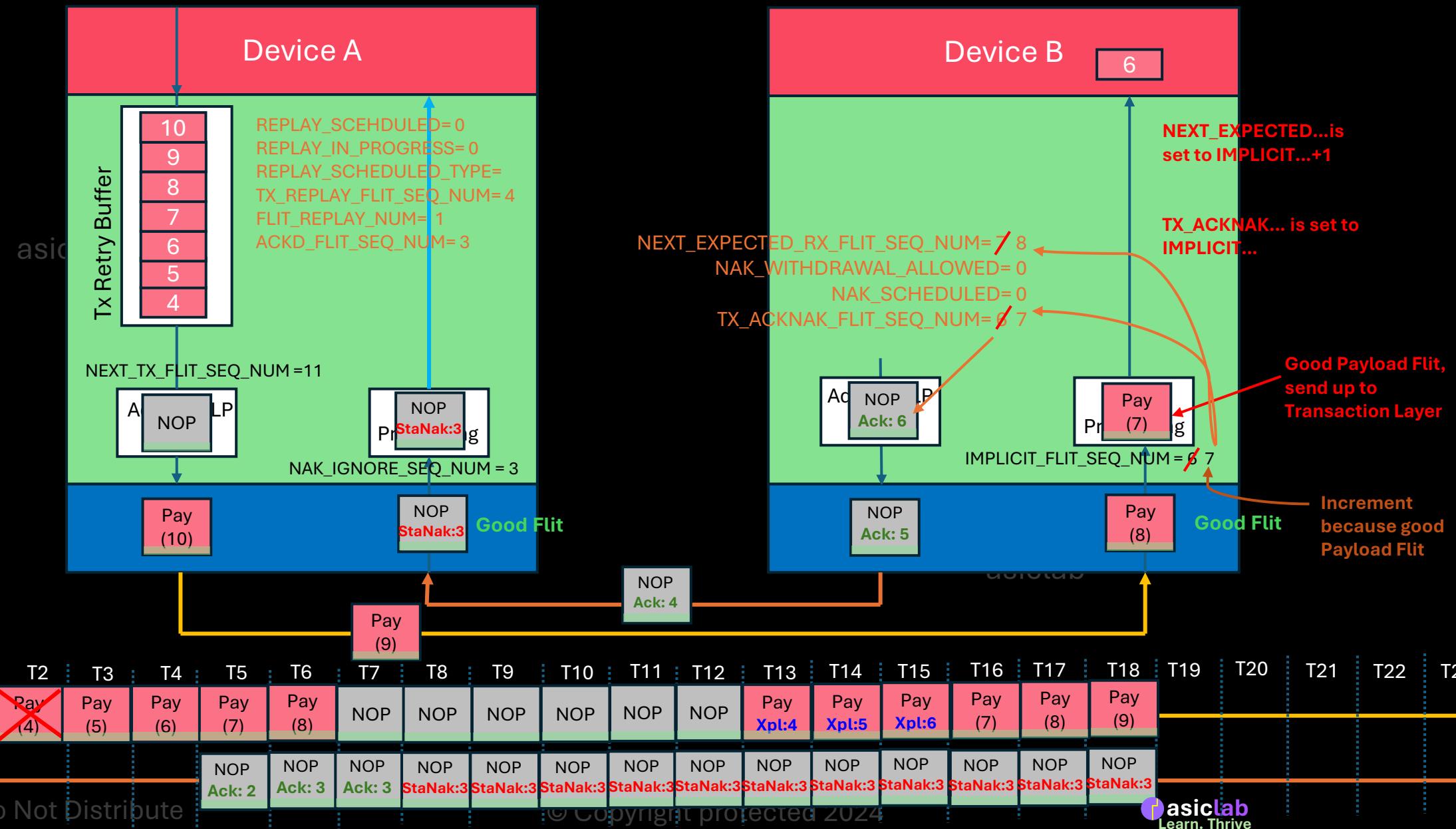
Standard ACK NAK Mechanism



Standard ACK NAK Mechanism



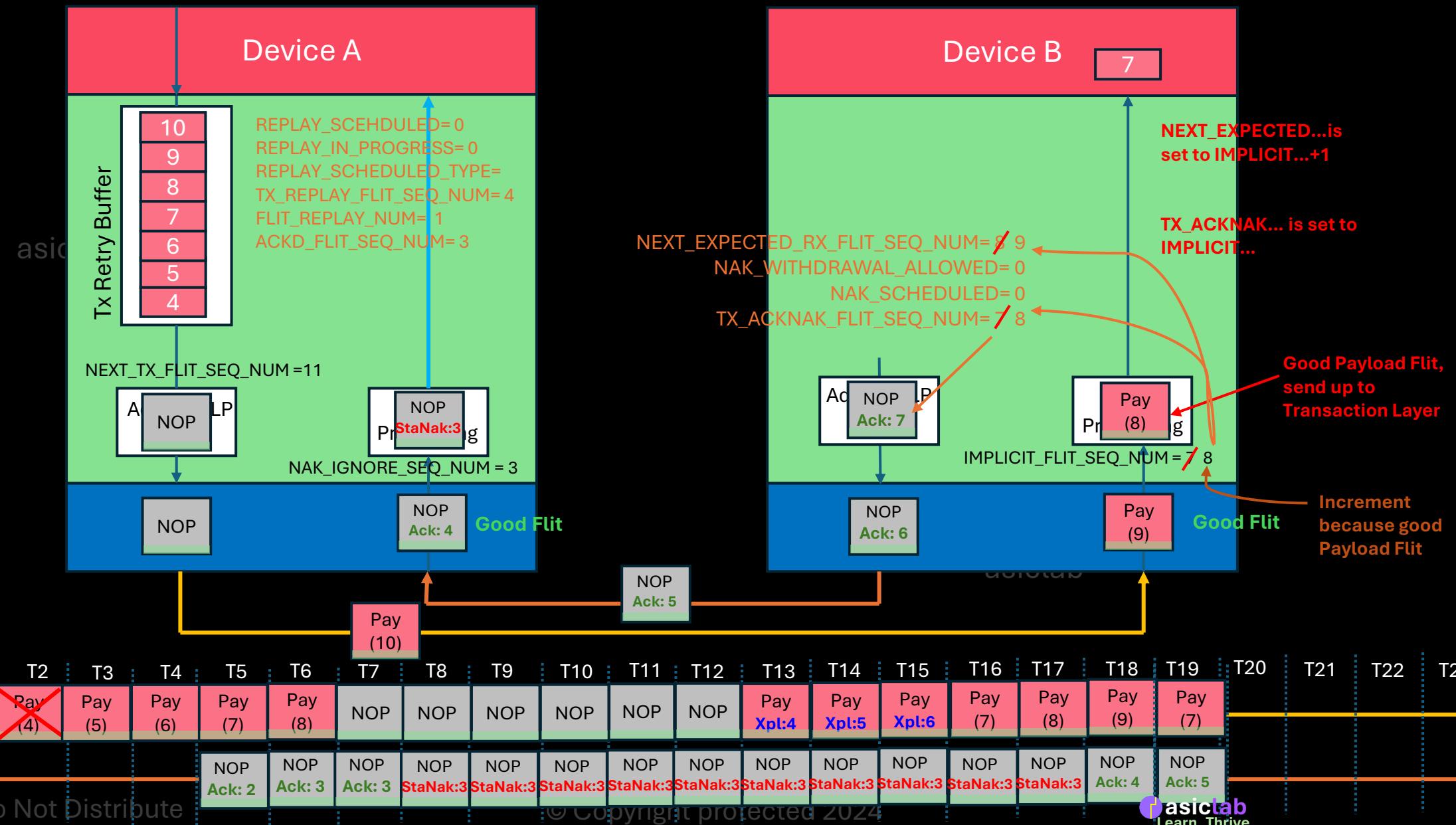
Standard ACK NAK Mechanism



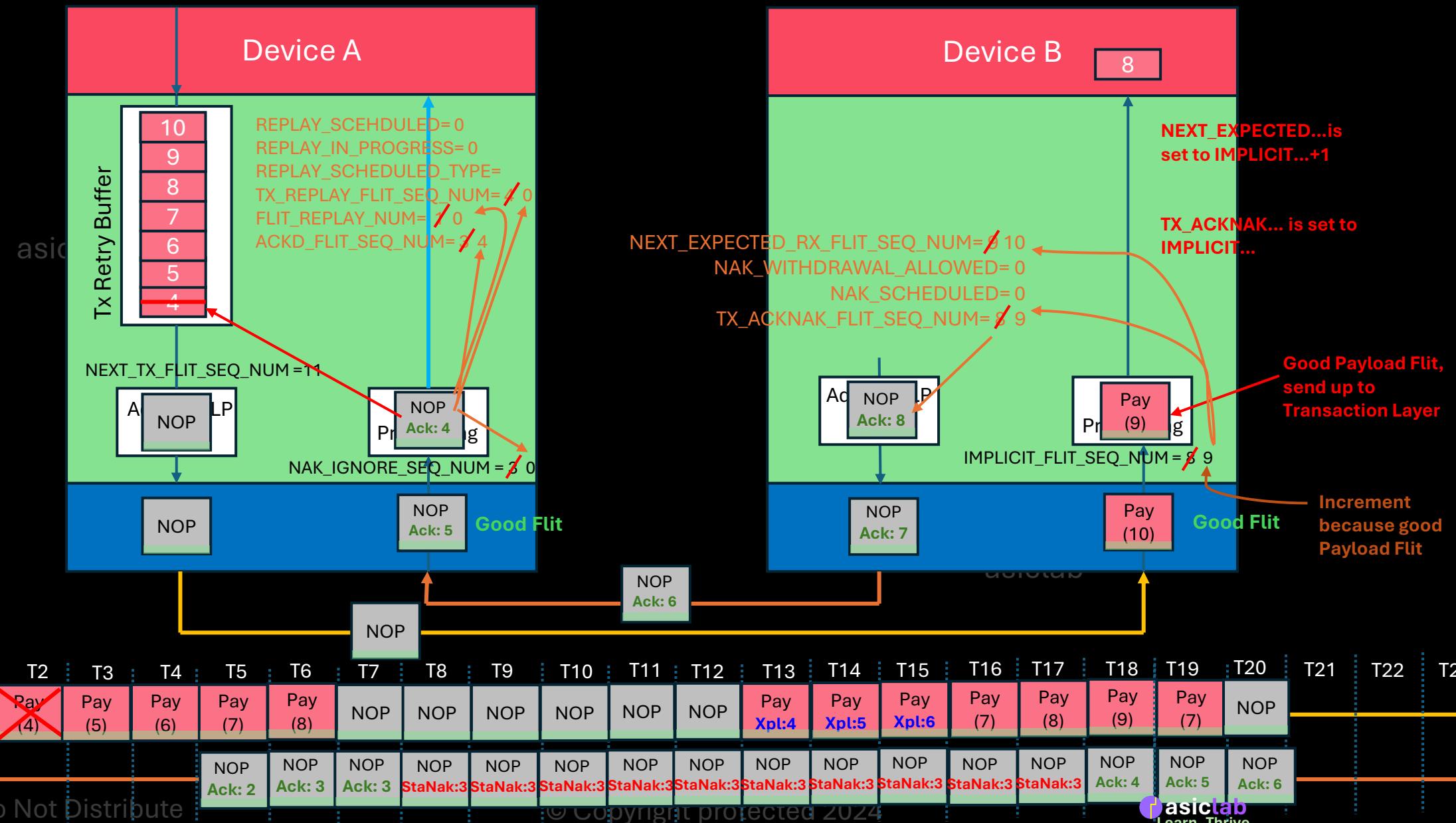
Do Not Distribute

© Copyright protected 2024

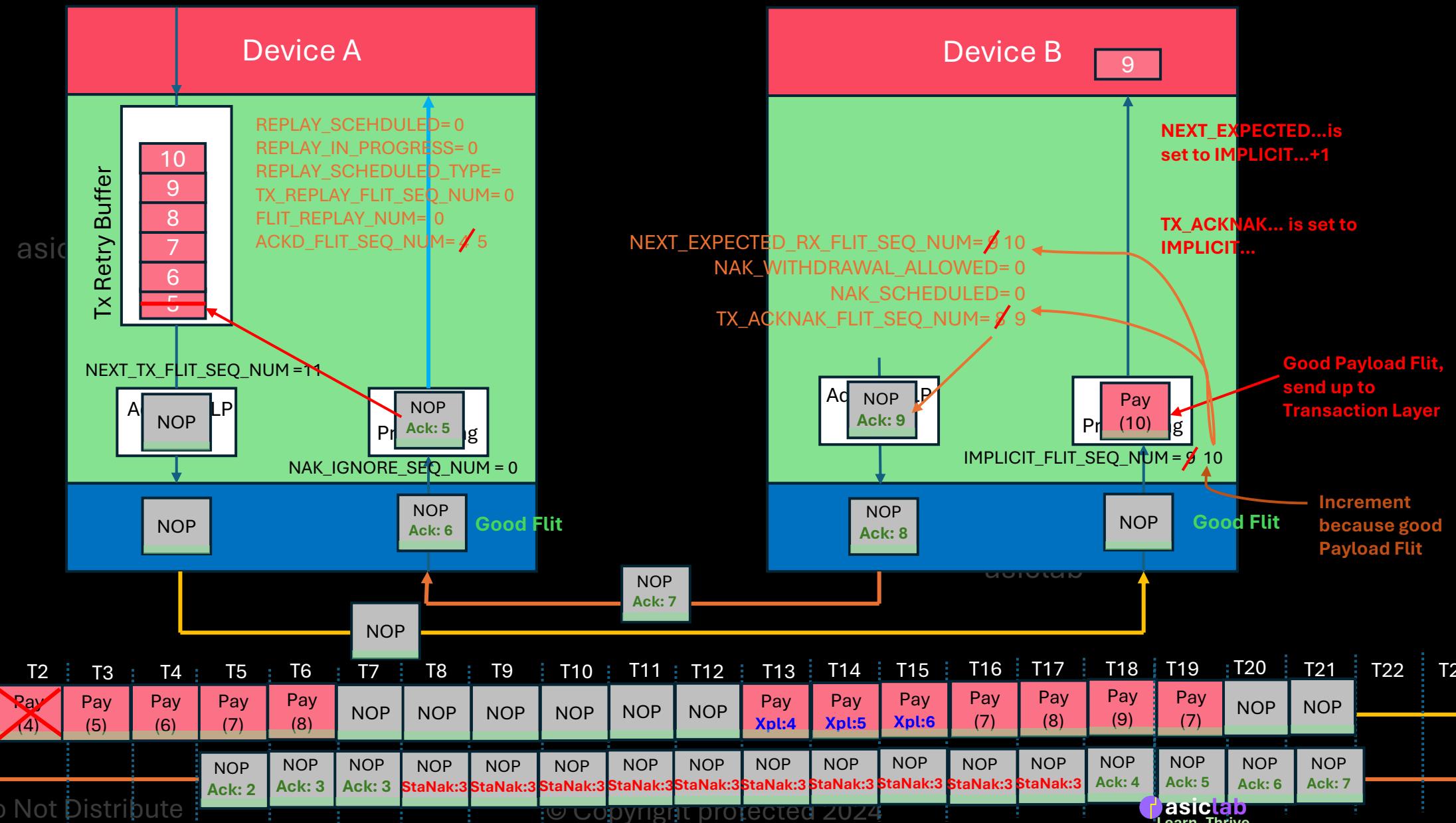
Standard ACK NAK Mechanism



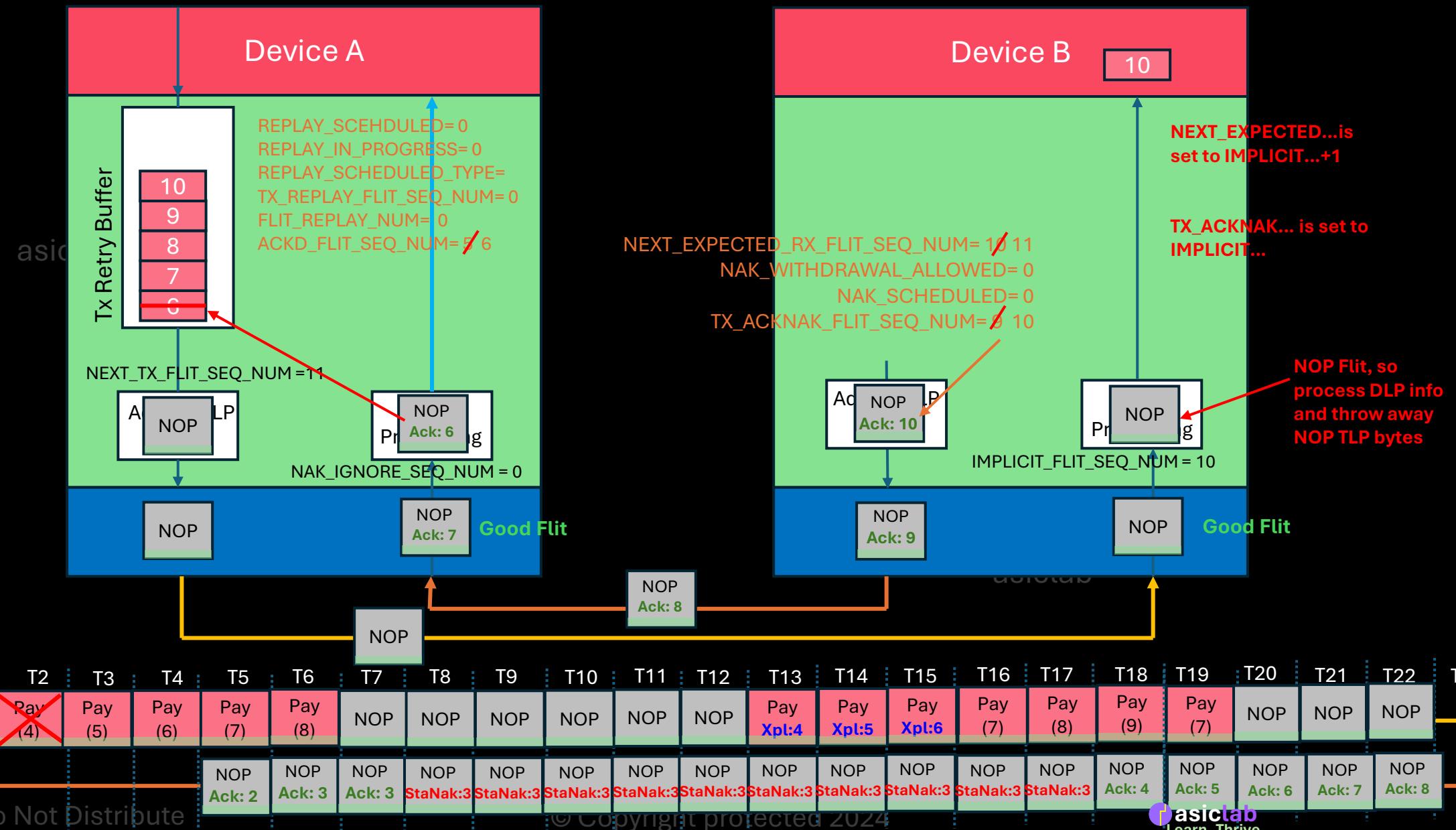
Standard ACK NAK Mechanism



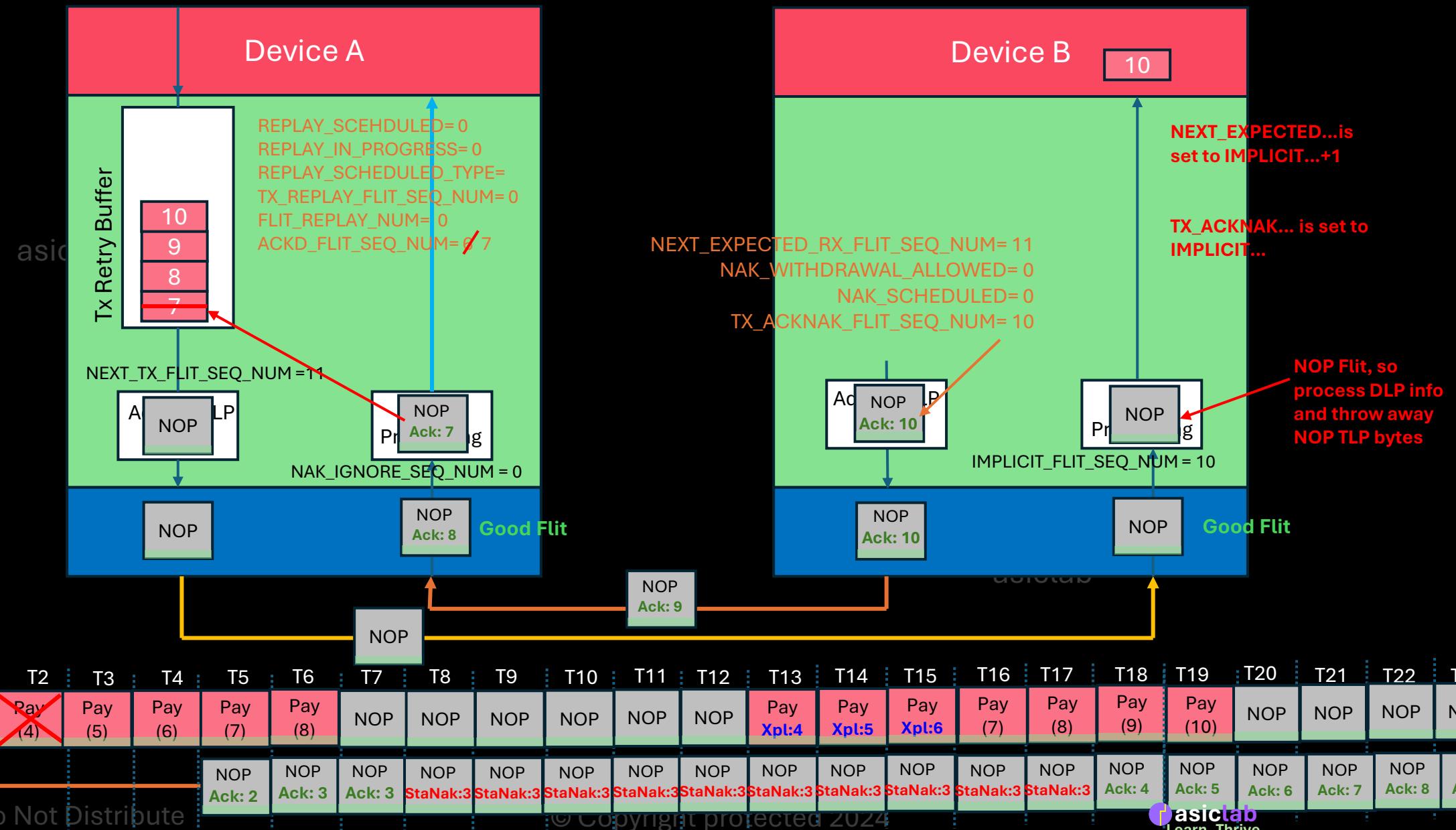
Standard ACK NAK Mechanism



Standard ACK NAK Mechanism



Standard ACK NAK Mechanism

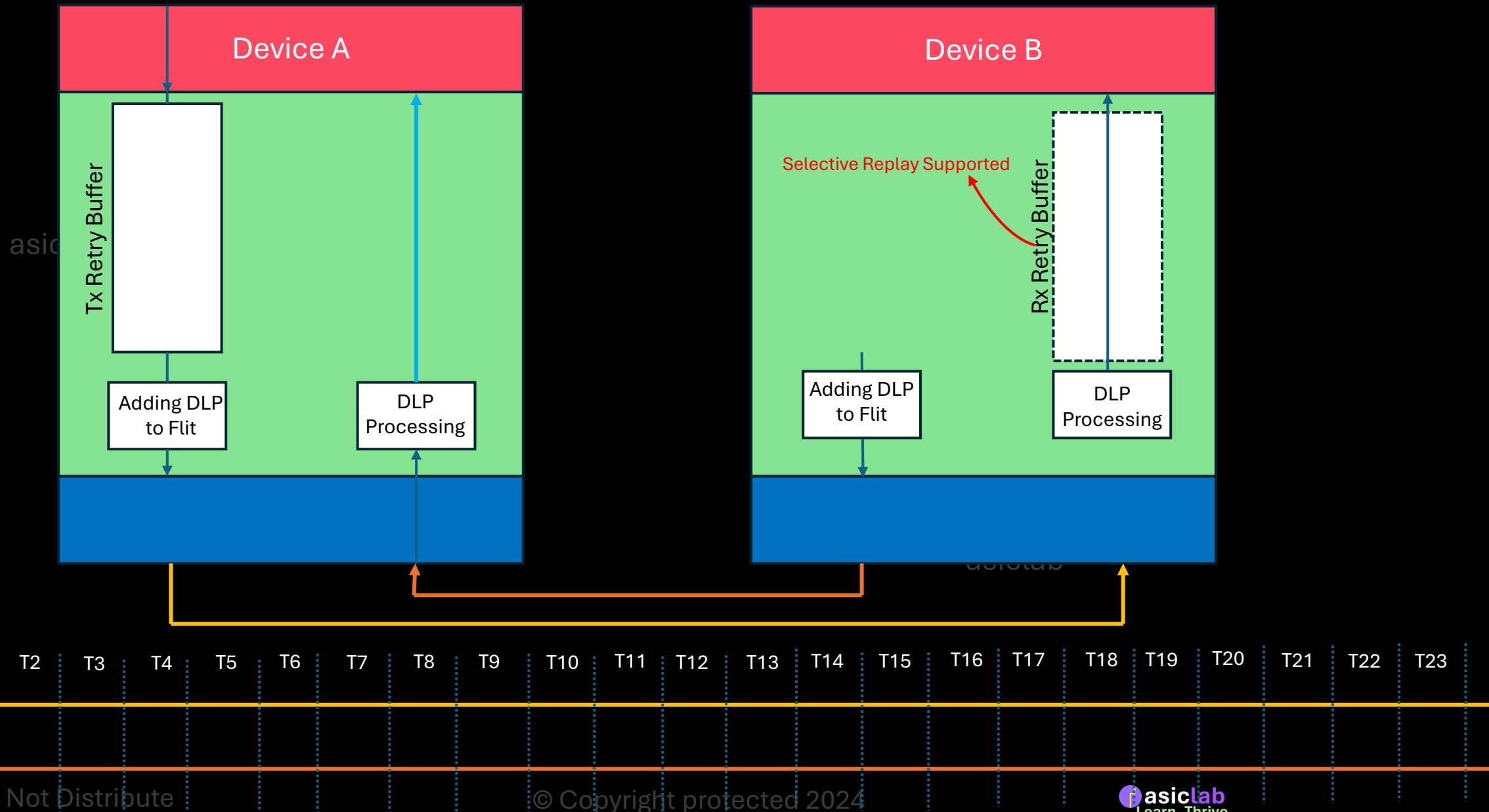


asiclab

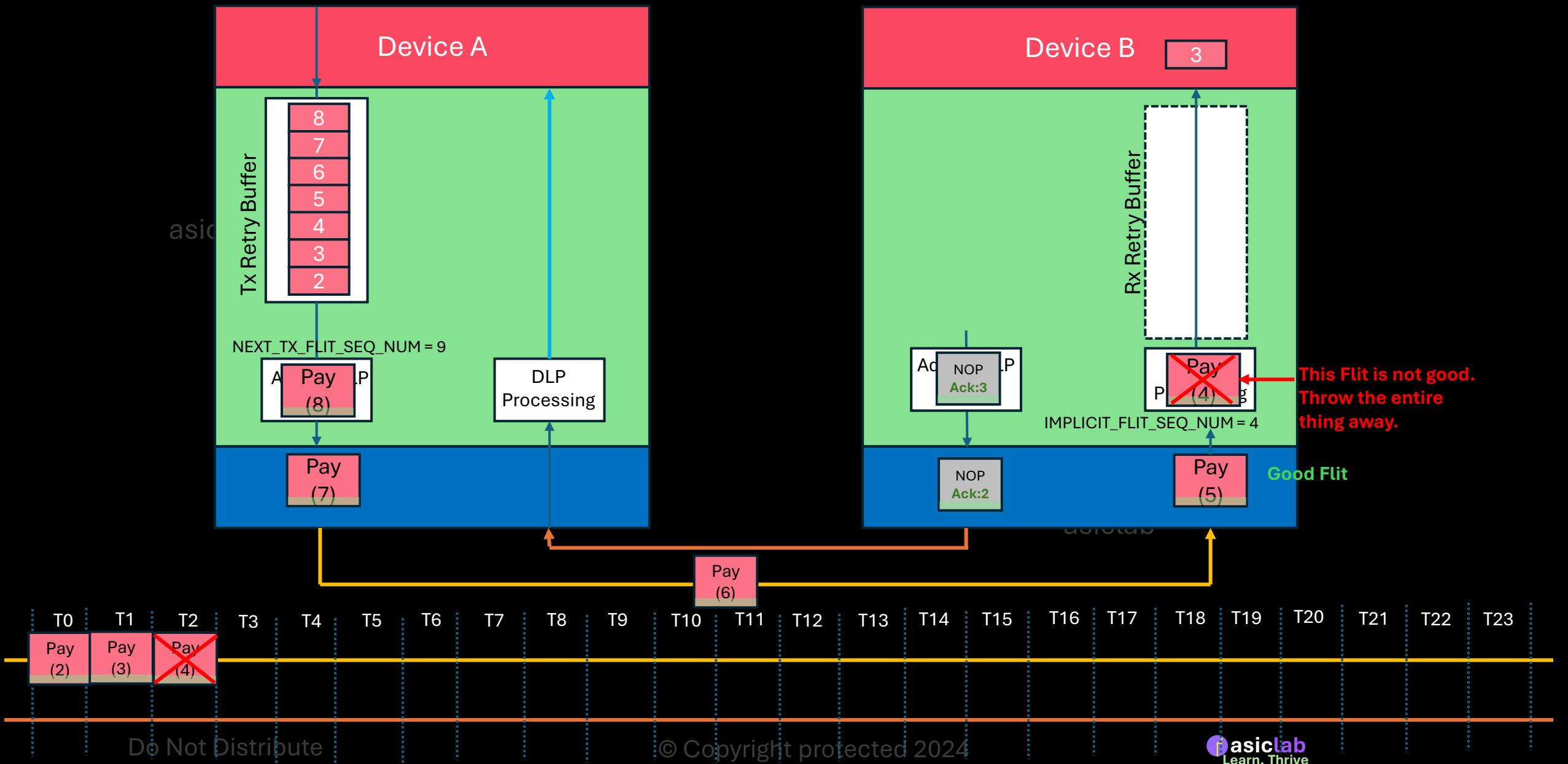
Selective ACK NAK Mechanism

asiclab

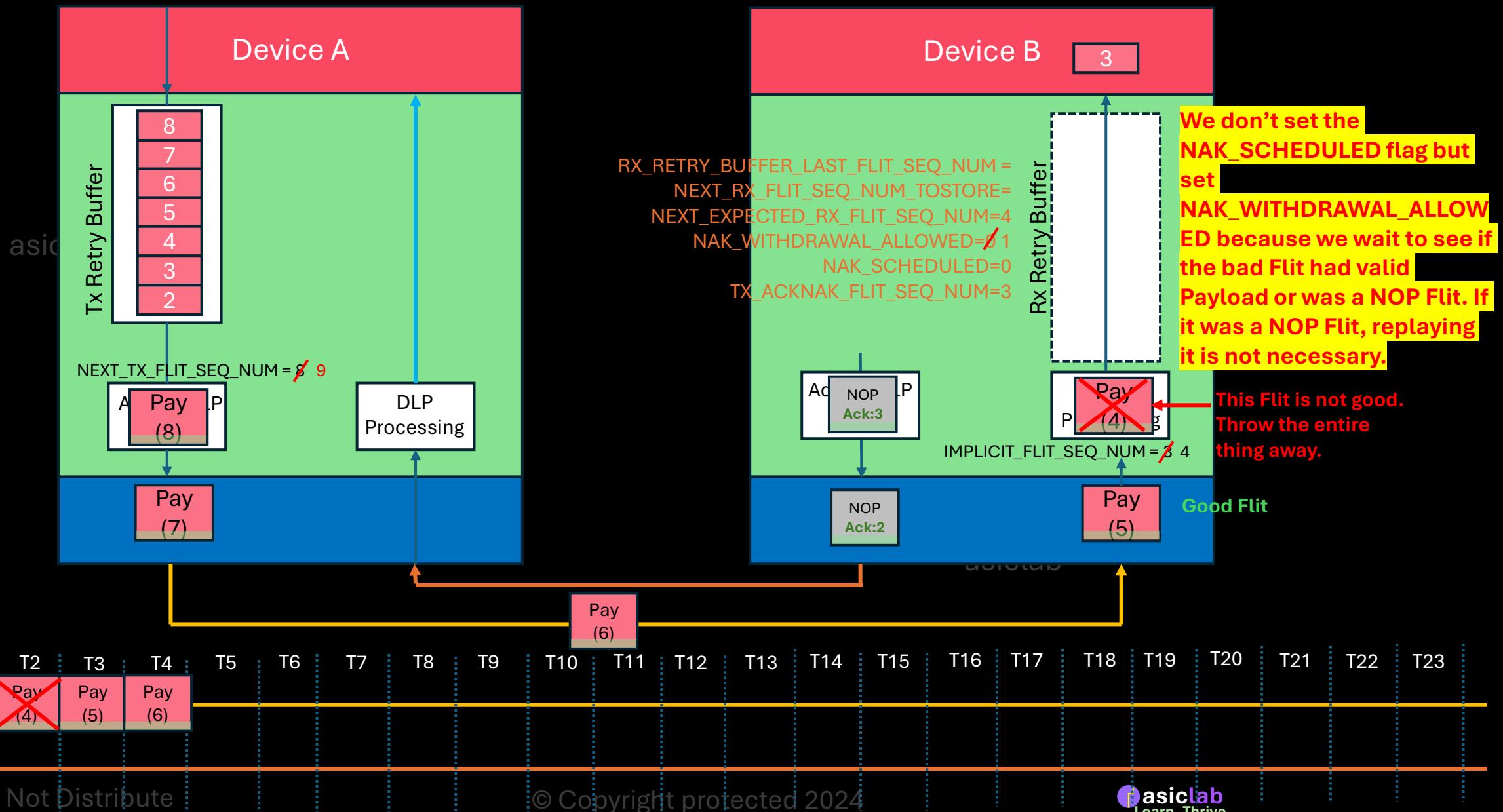
Selective ACK NAK Mechanism



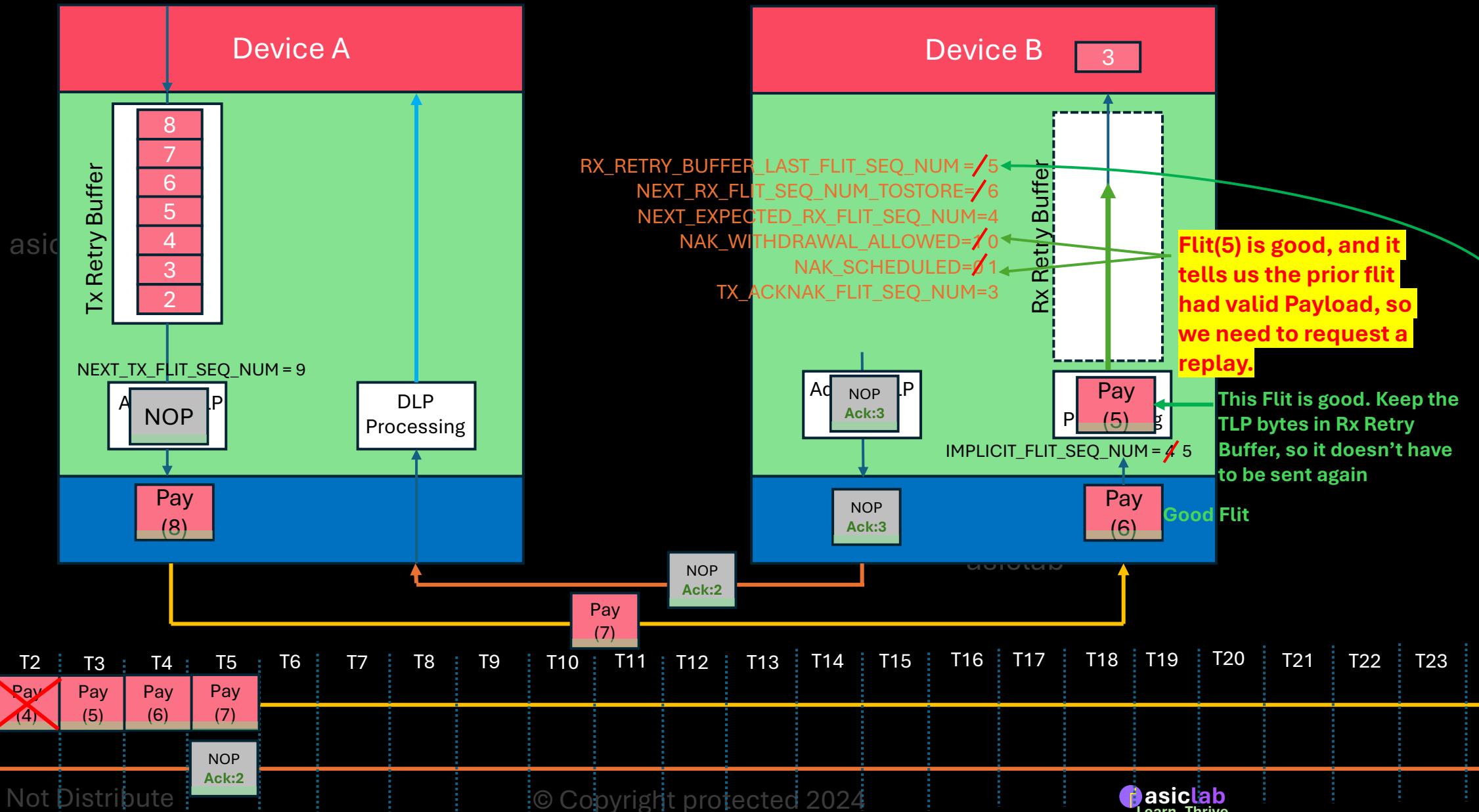
Selective ACK NAK Mechanism



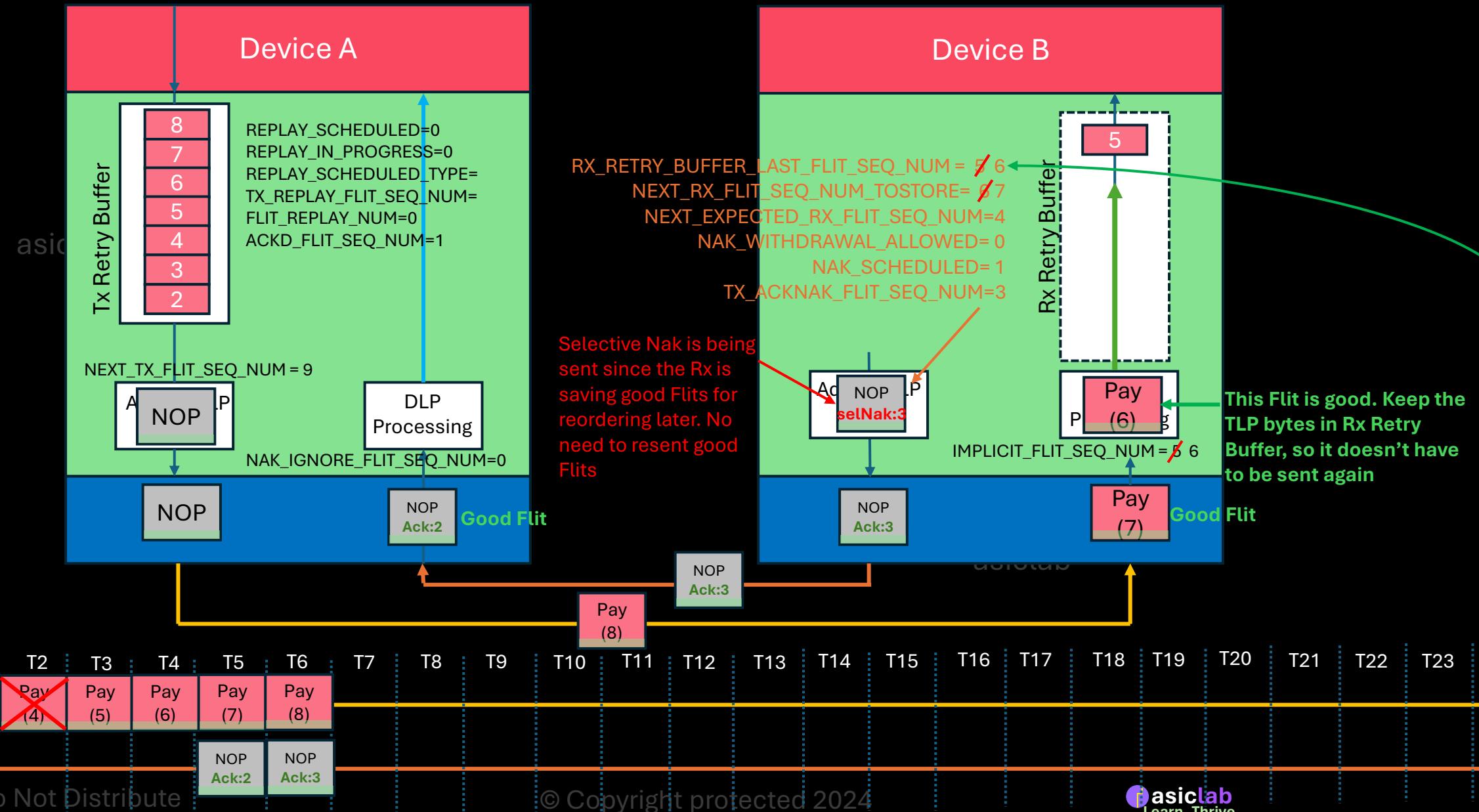
Selective ACK NAK Mechanism



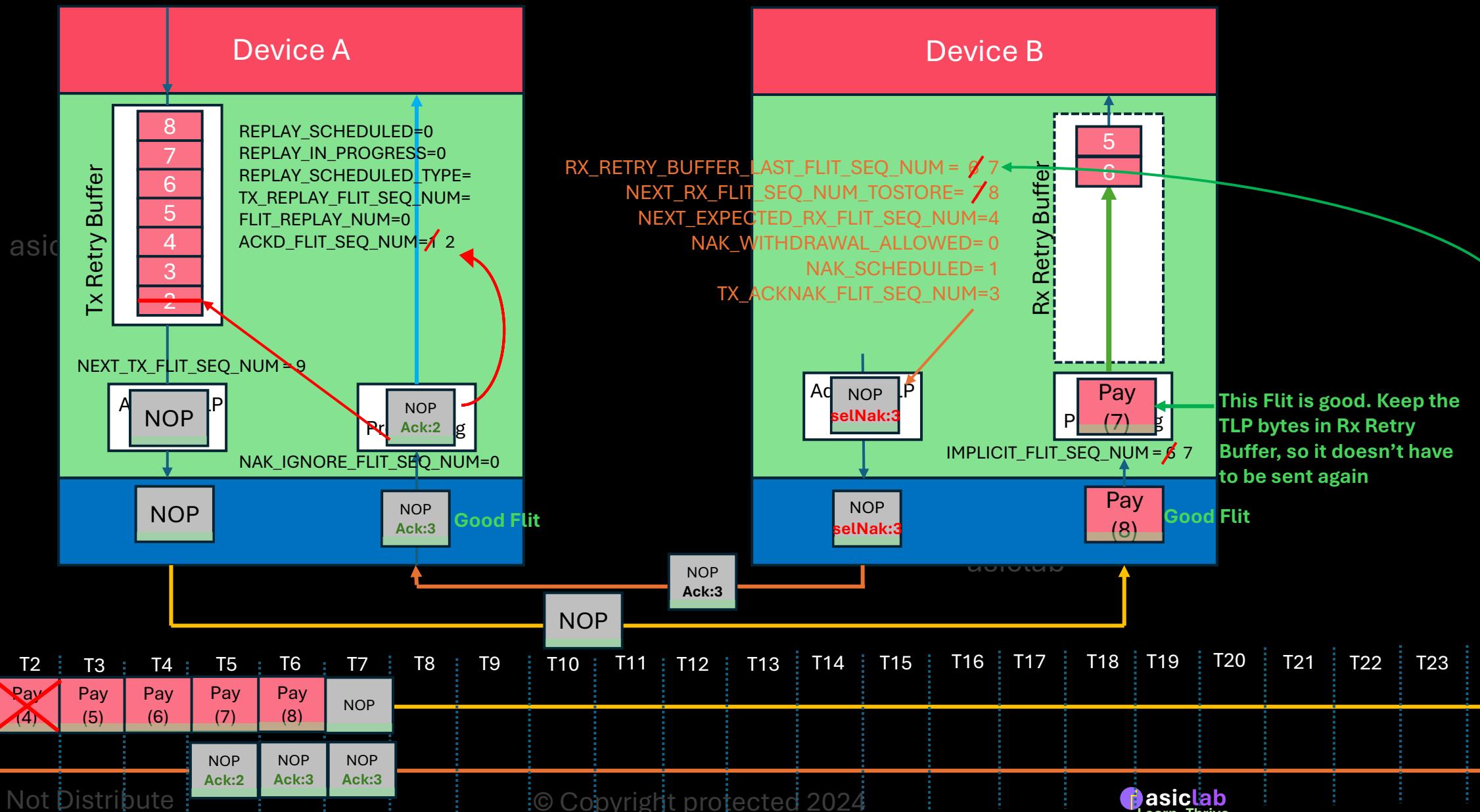
Selective ACK NAK Mechanism



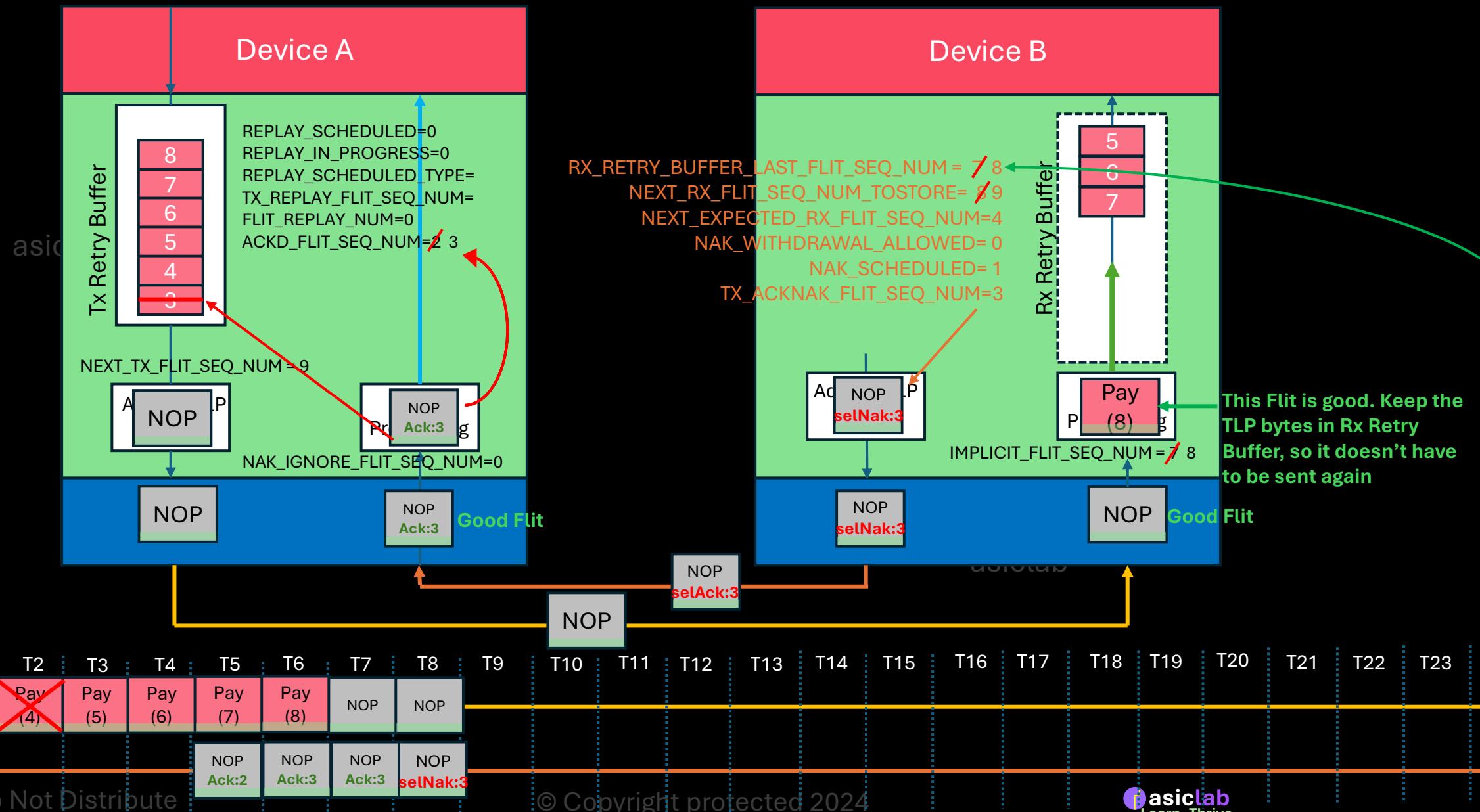
Selective ACK NAK Mechanism



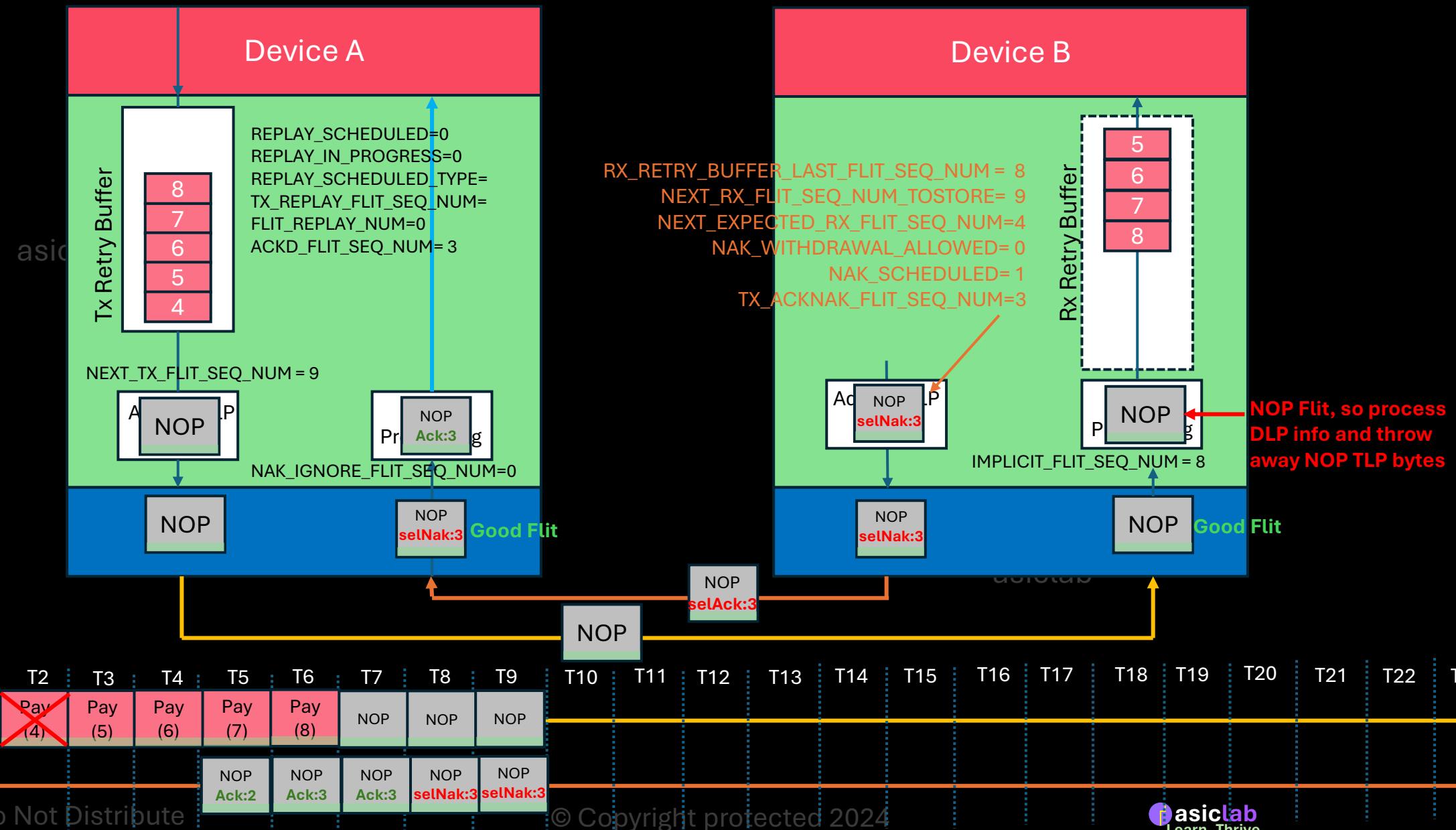
Selective ACK NAK Mechanism



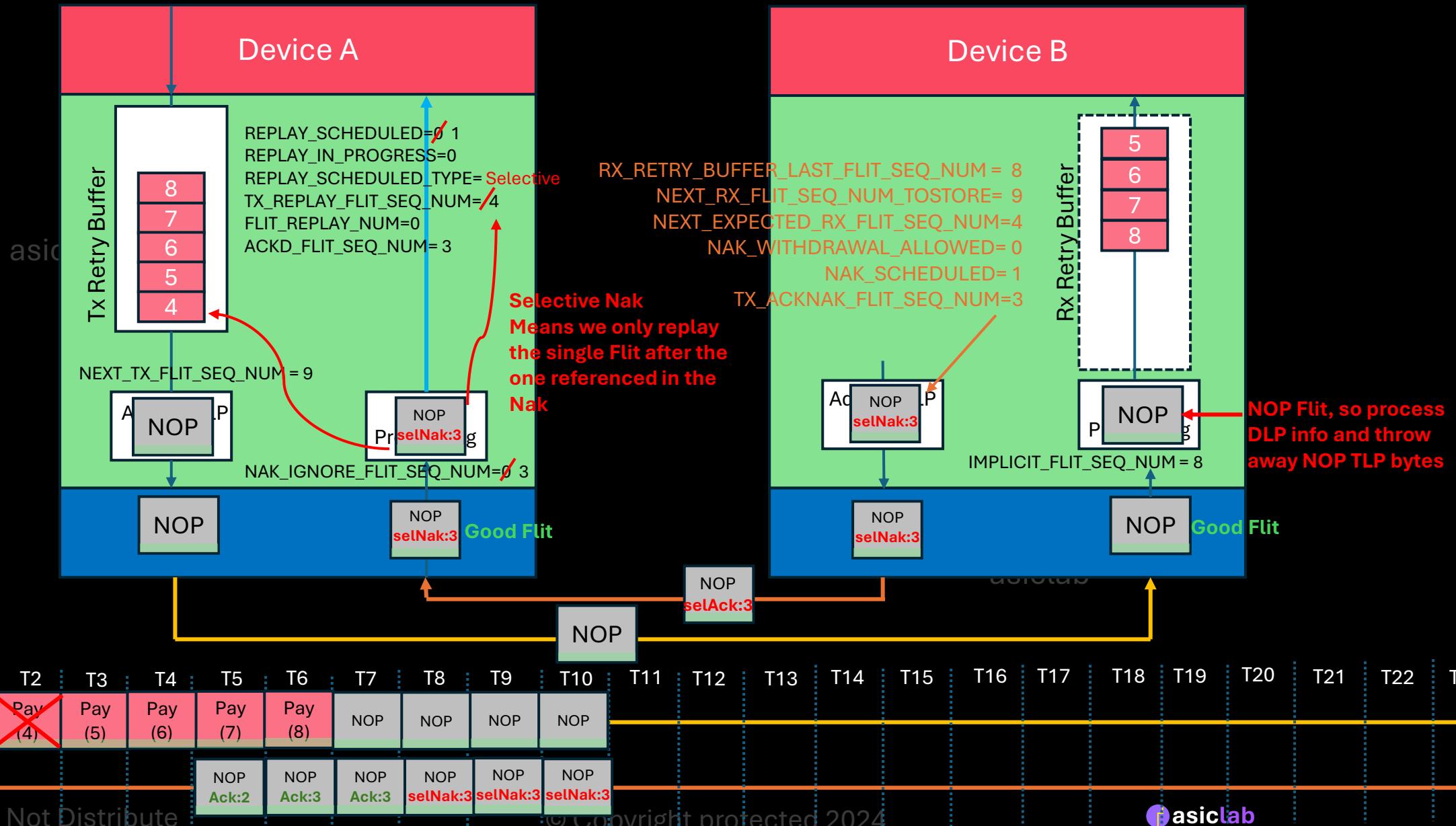
Selective ACK NAK Mechanism



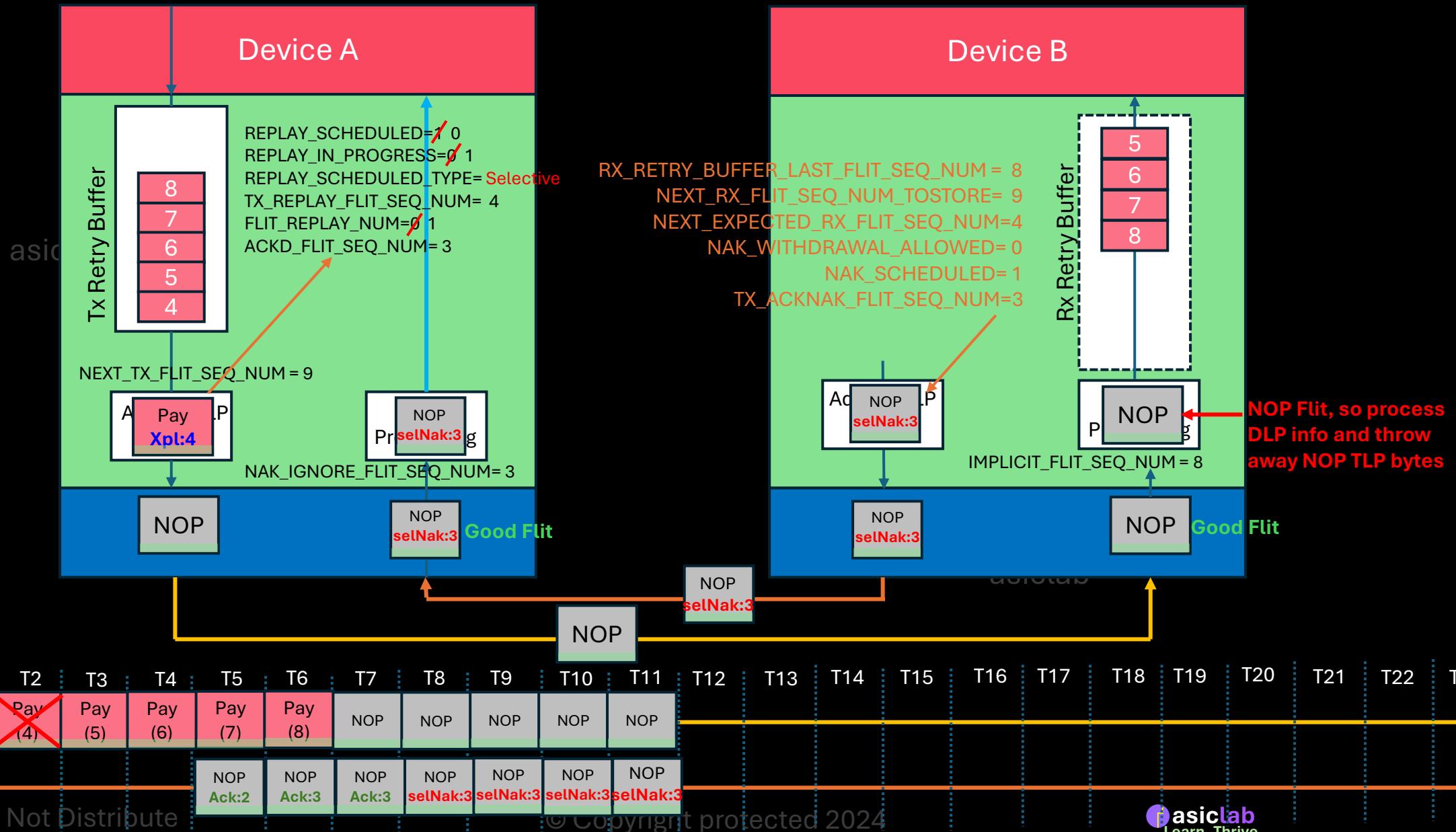
Selective ACK NAK Mechanism



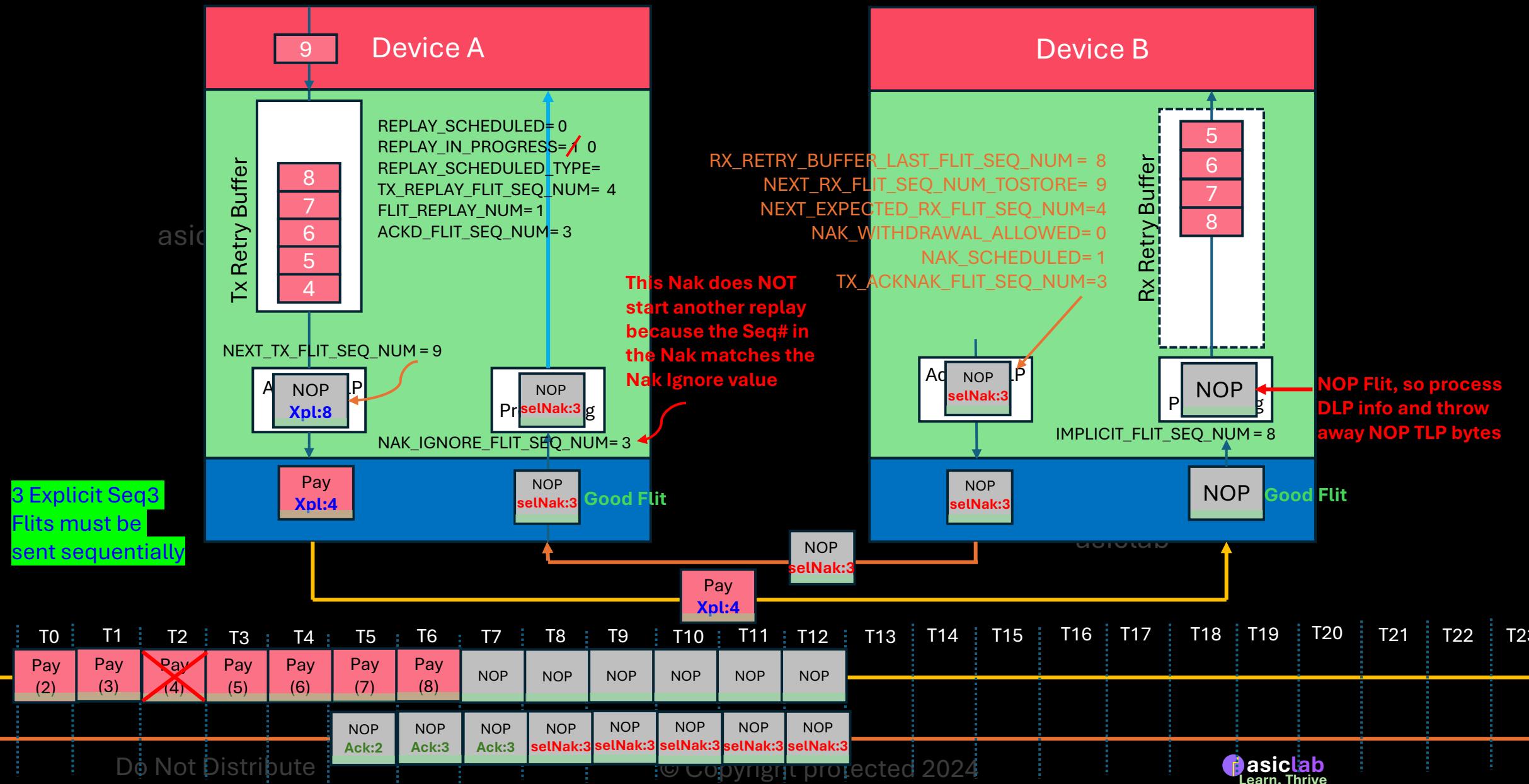
Selective ACK NAK Mechanism



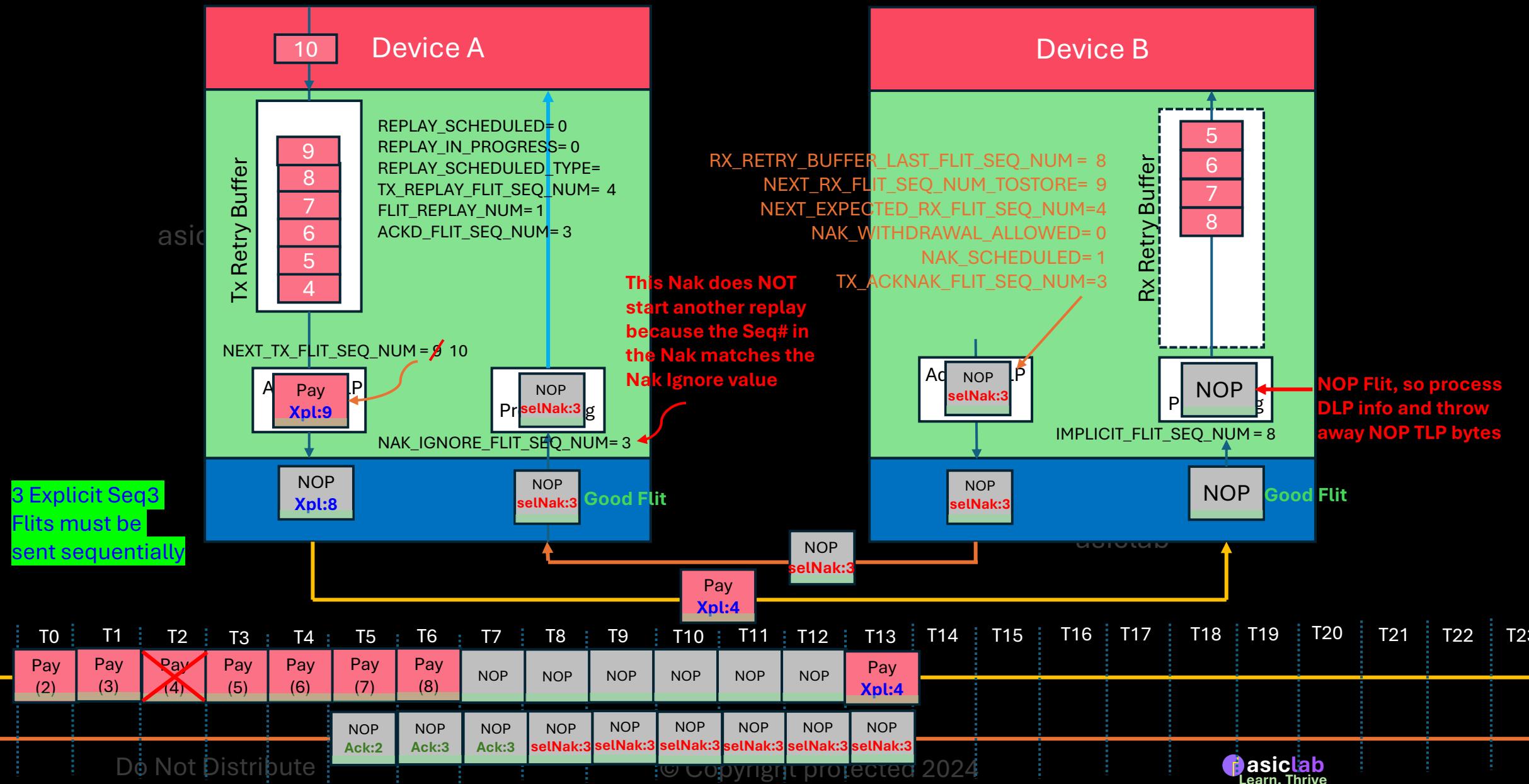
Selective ACK NAK Mechanism



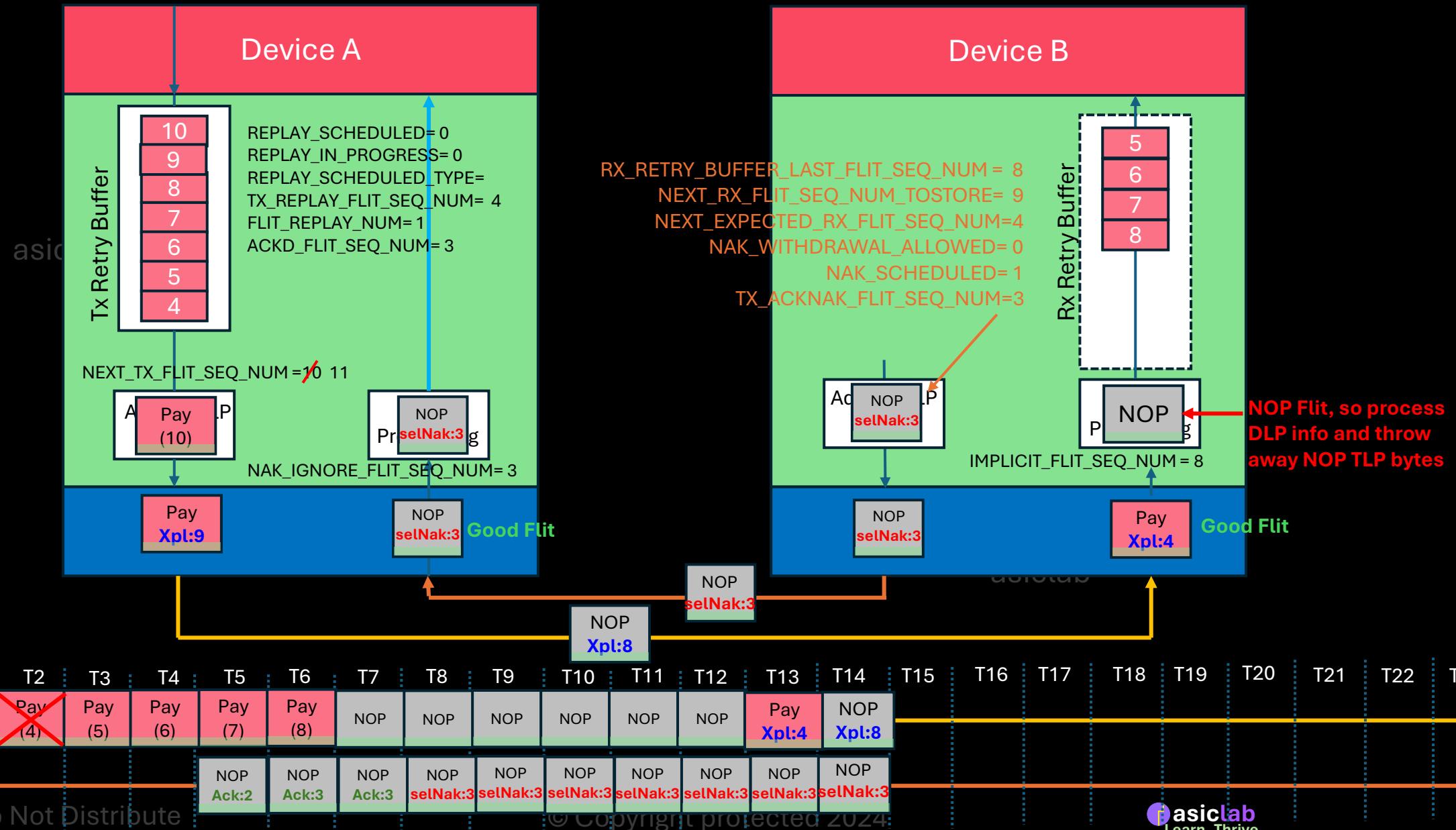
Selective ACK NAK Mechanism



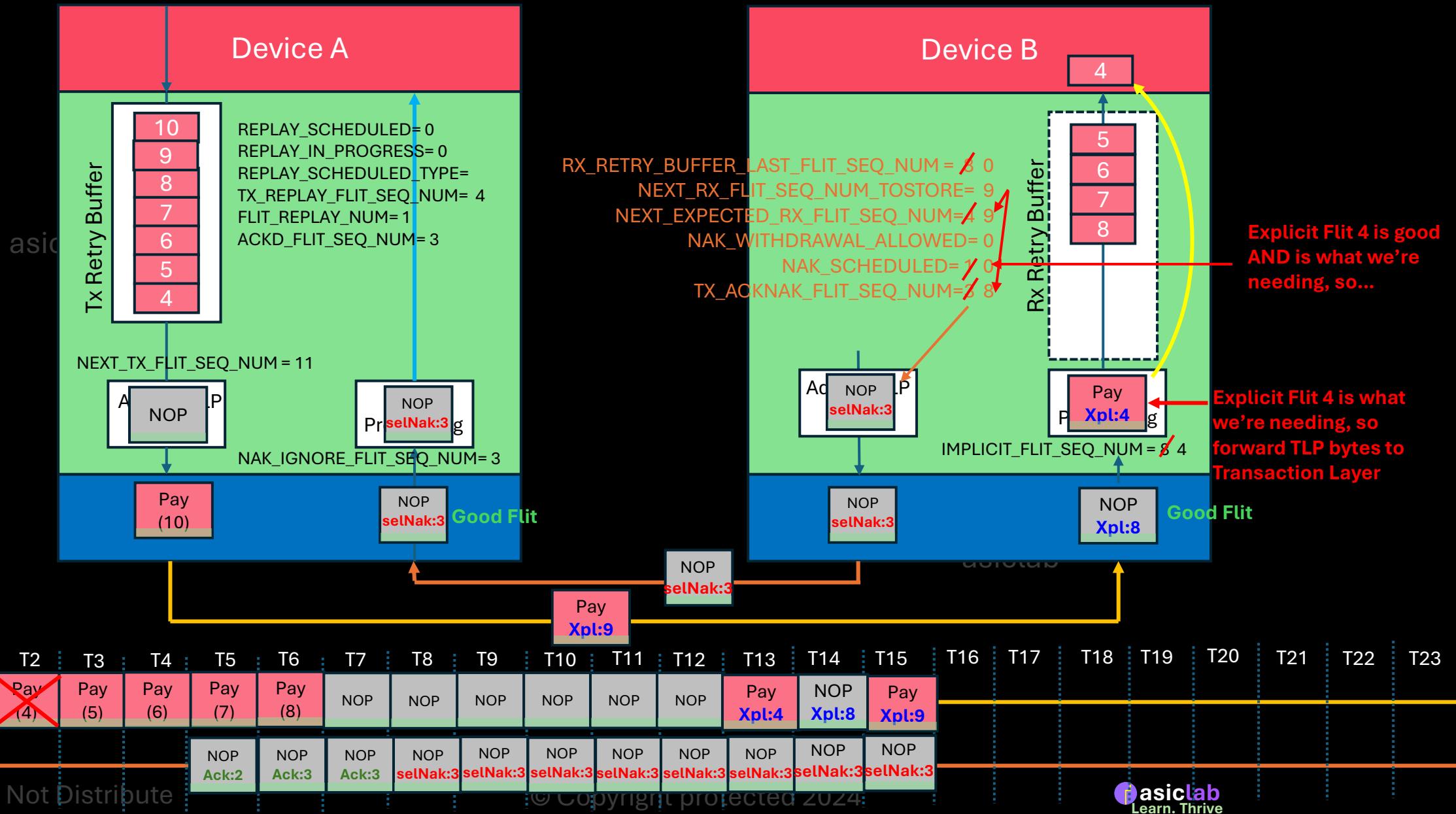
Selective ACK NAK Mechanism



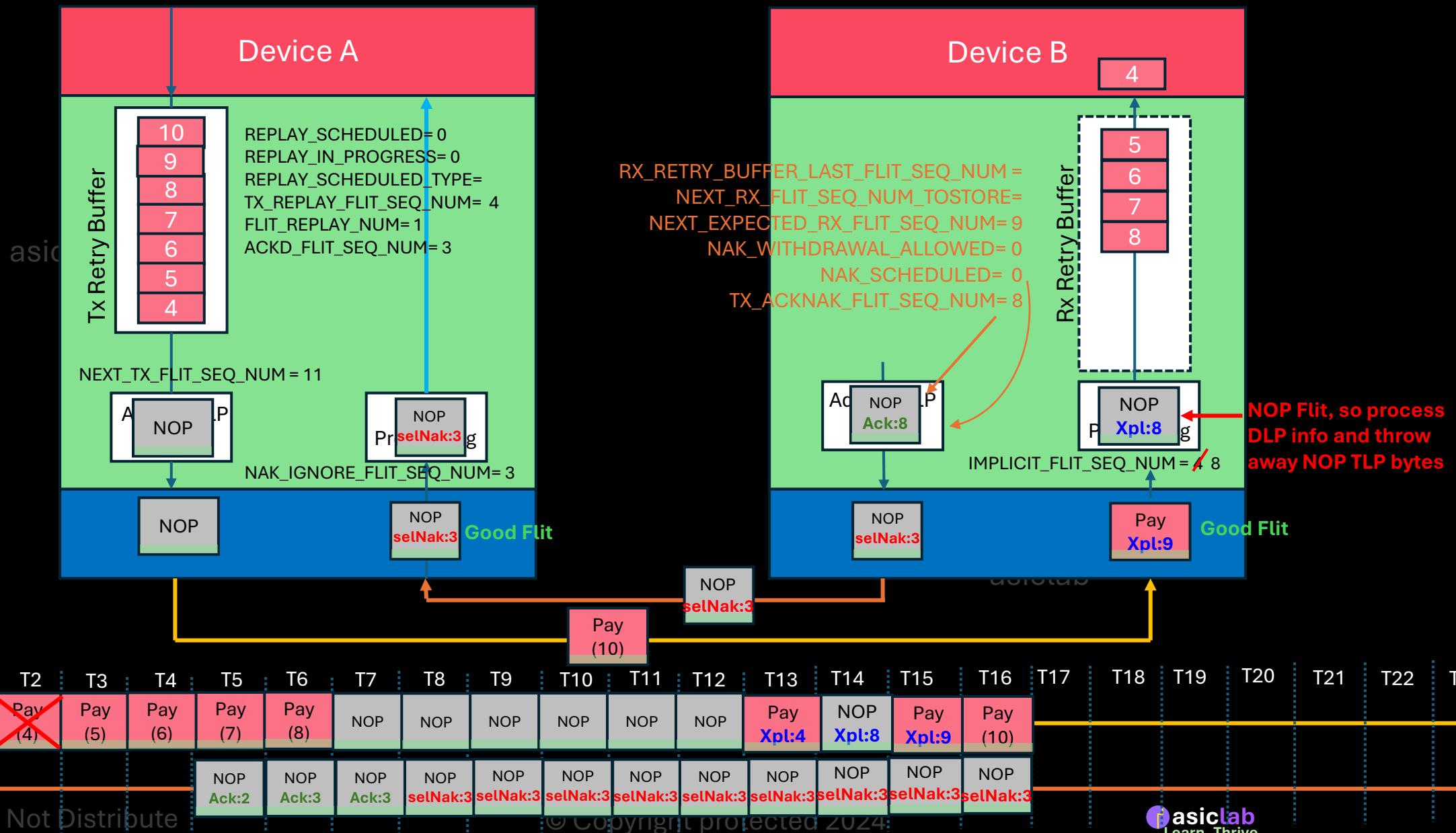
Selective ACK NAK Mechanism



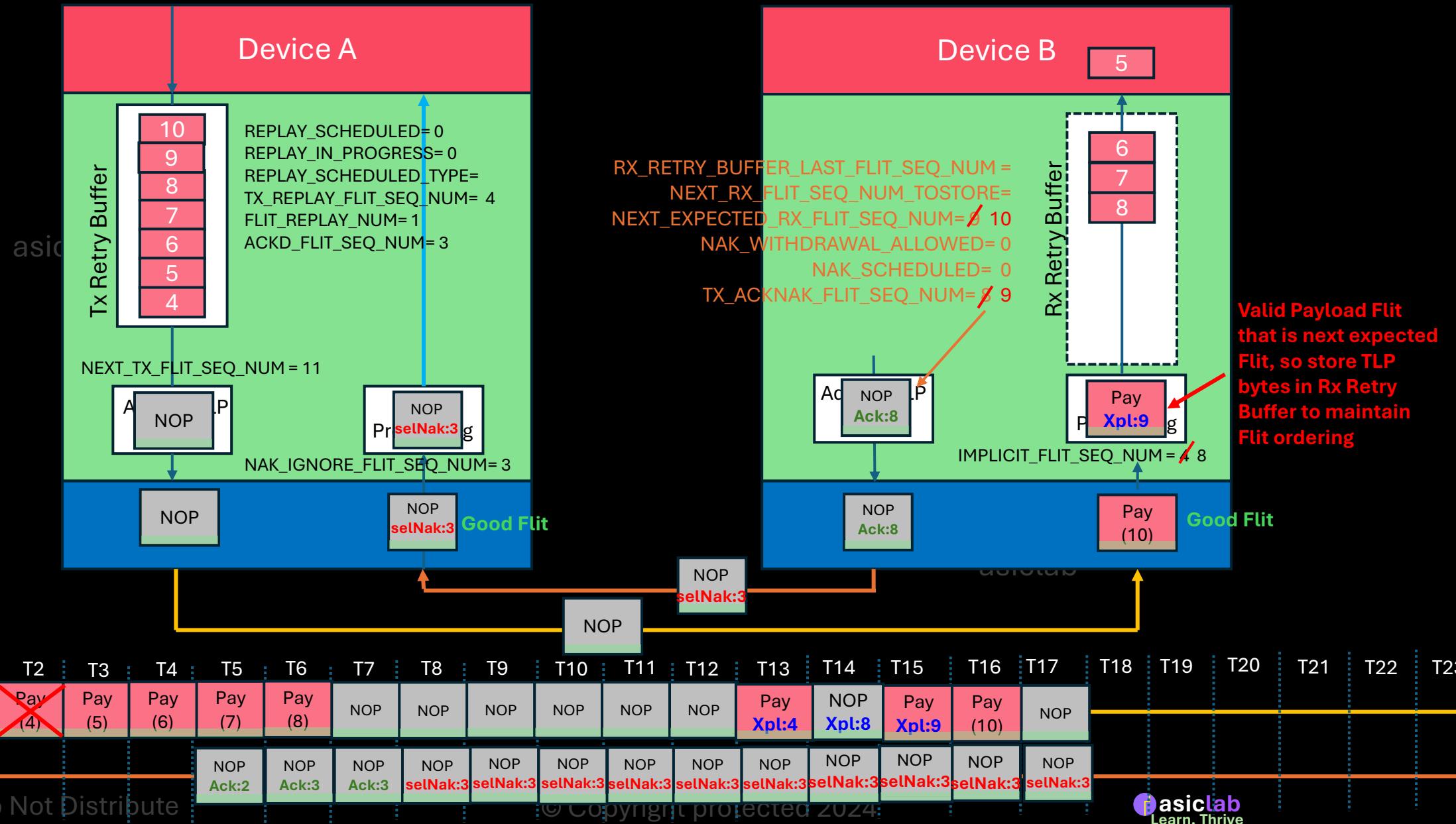
Selective ACK NAK Mechanism



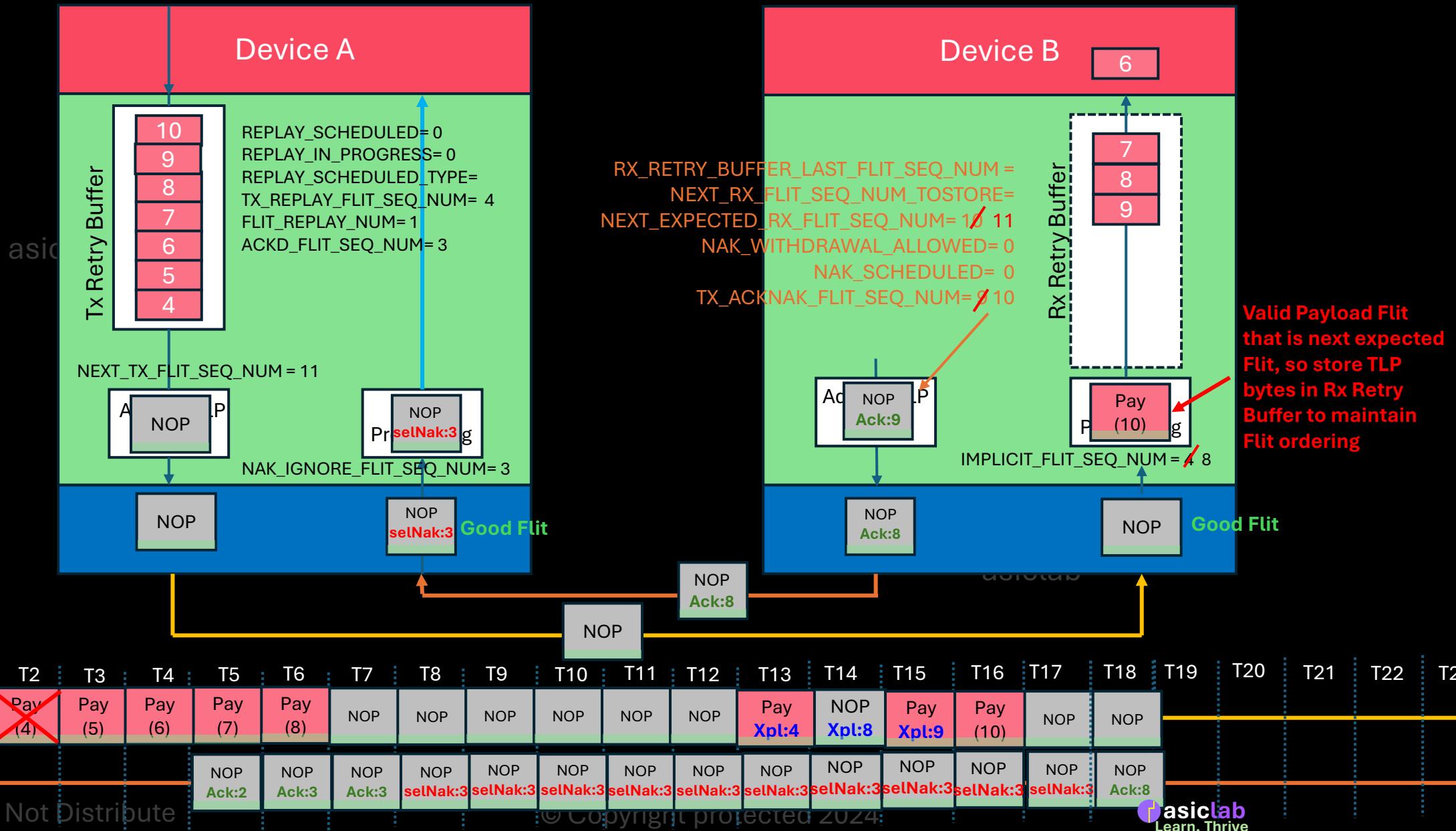
Selective ACK NAK Mechanism



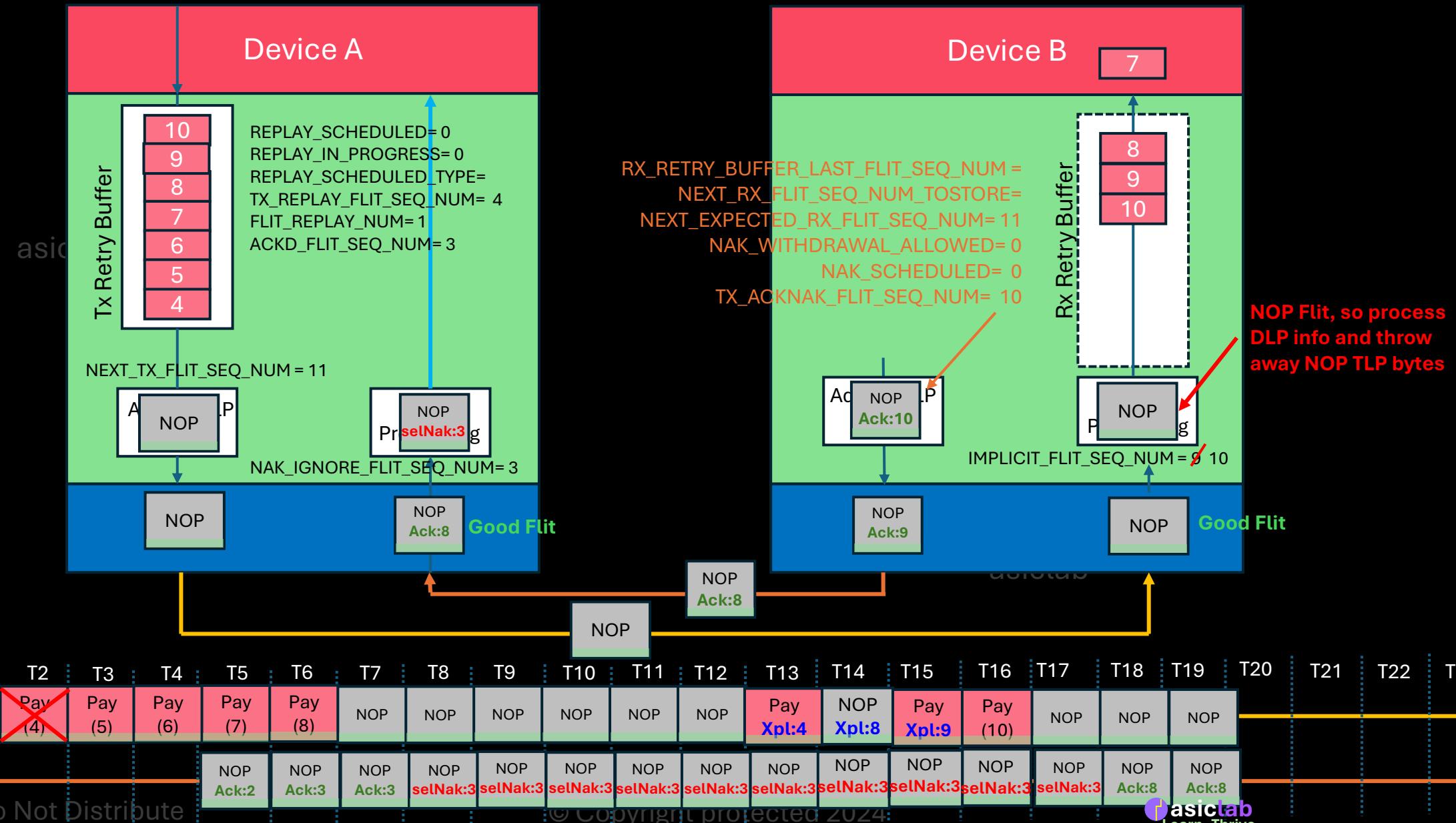
Selective ACK NAK Mechanism



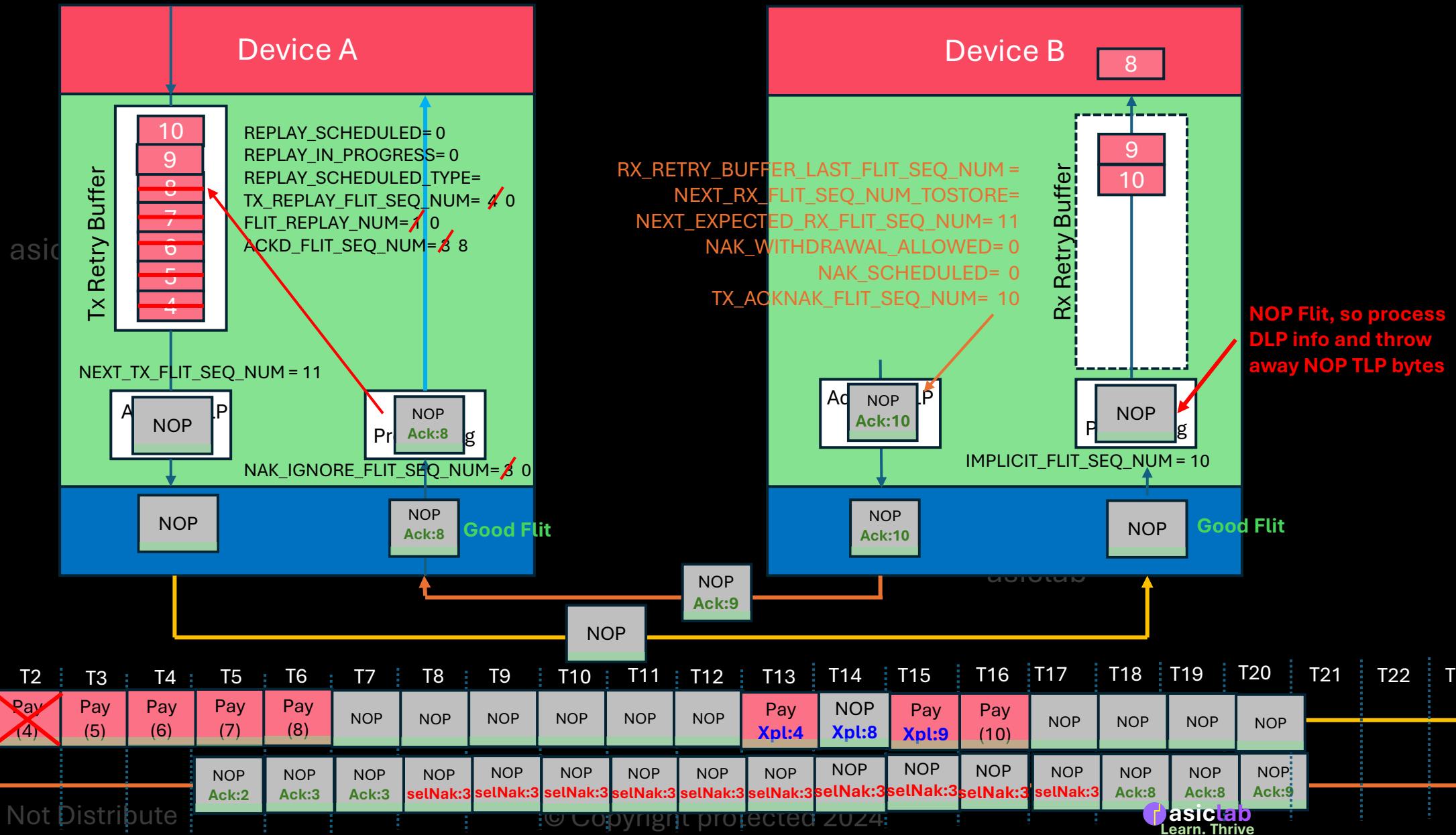
Selective ACK NAK Mechanism



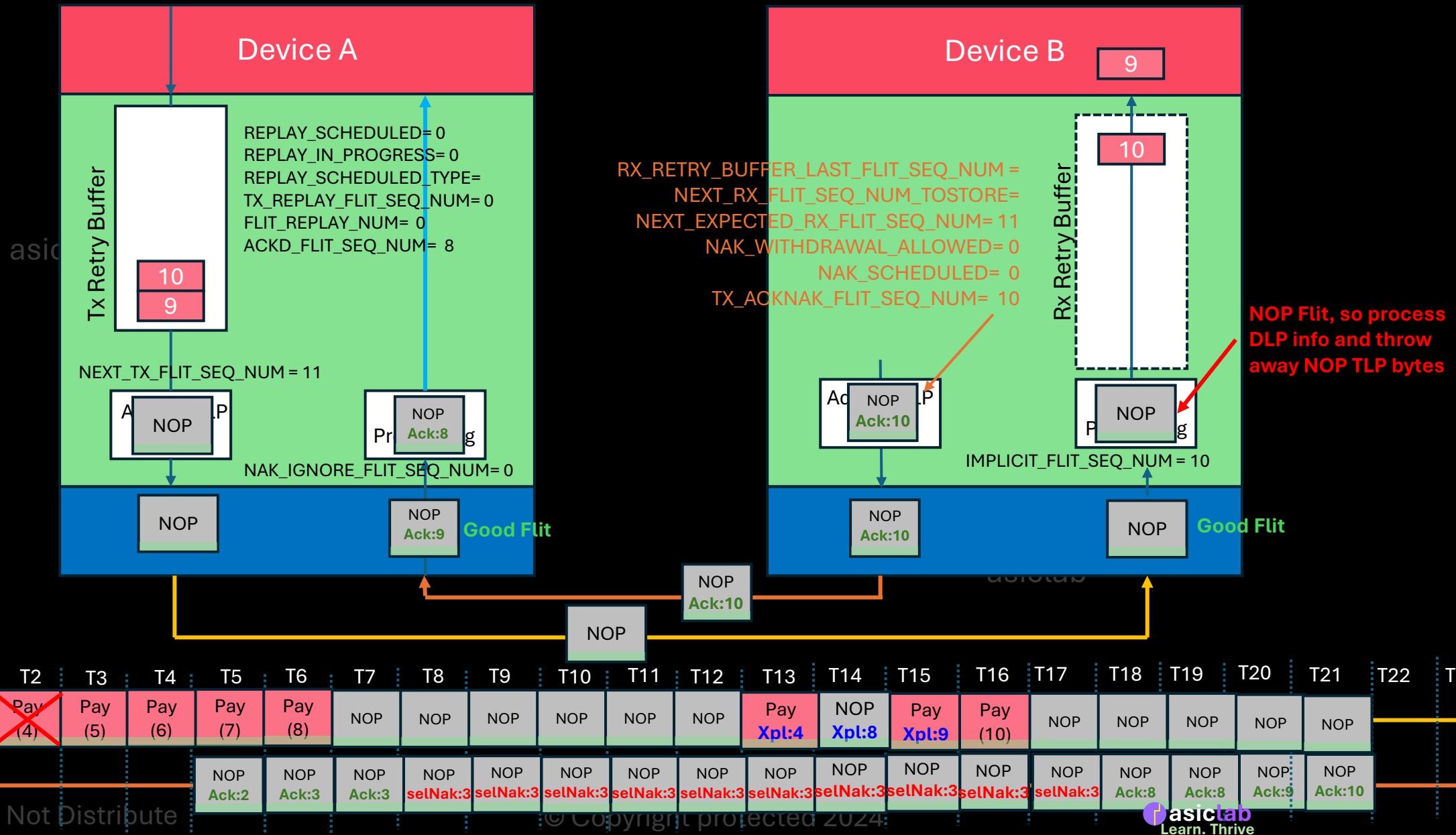
Selective ACK NAK Mechanism



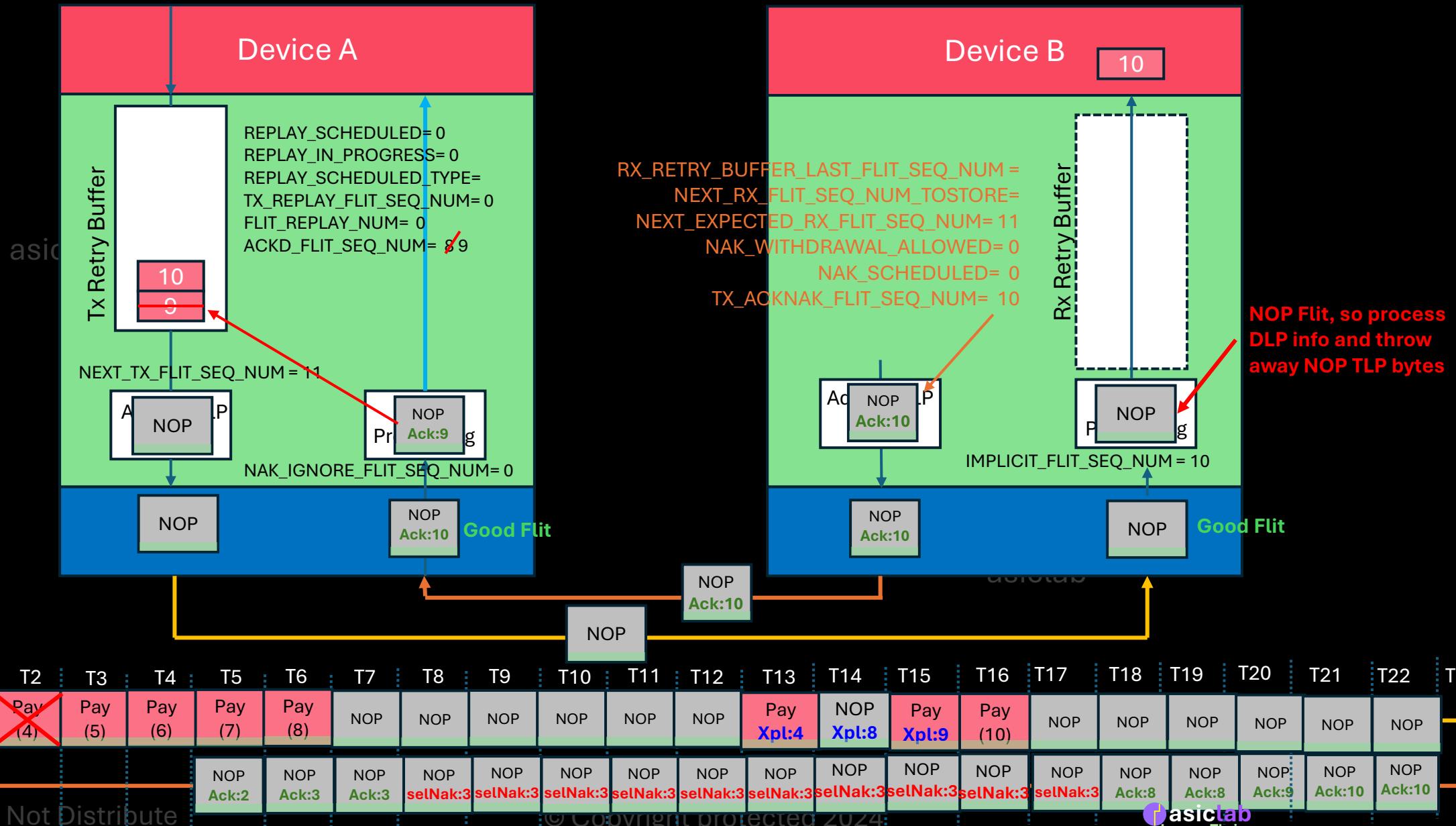
Selective ACK NAK Mechanism



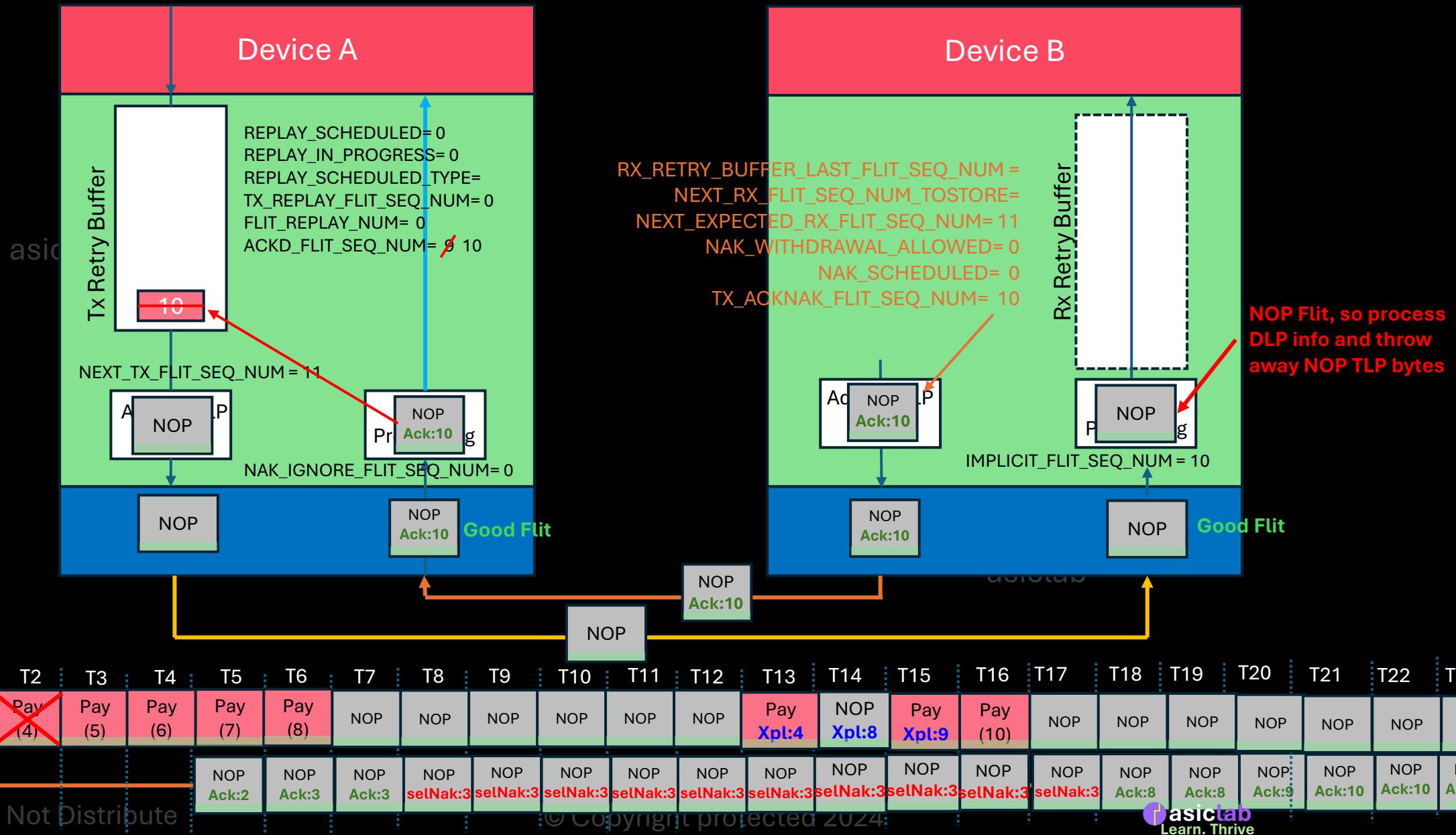
Selective ACK NAK Mechanism



Selective ACK NAK Mechanism



Selective ACK NAK Mechanism



asiclab

Physical Layer

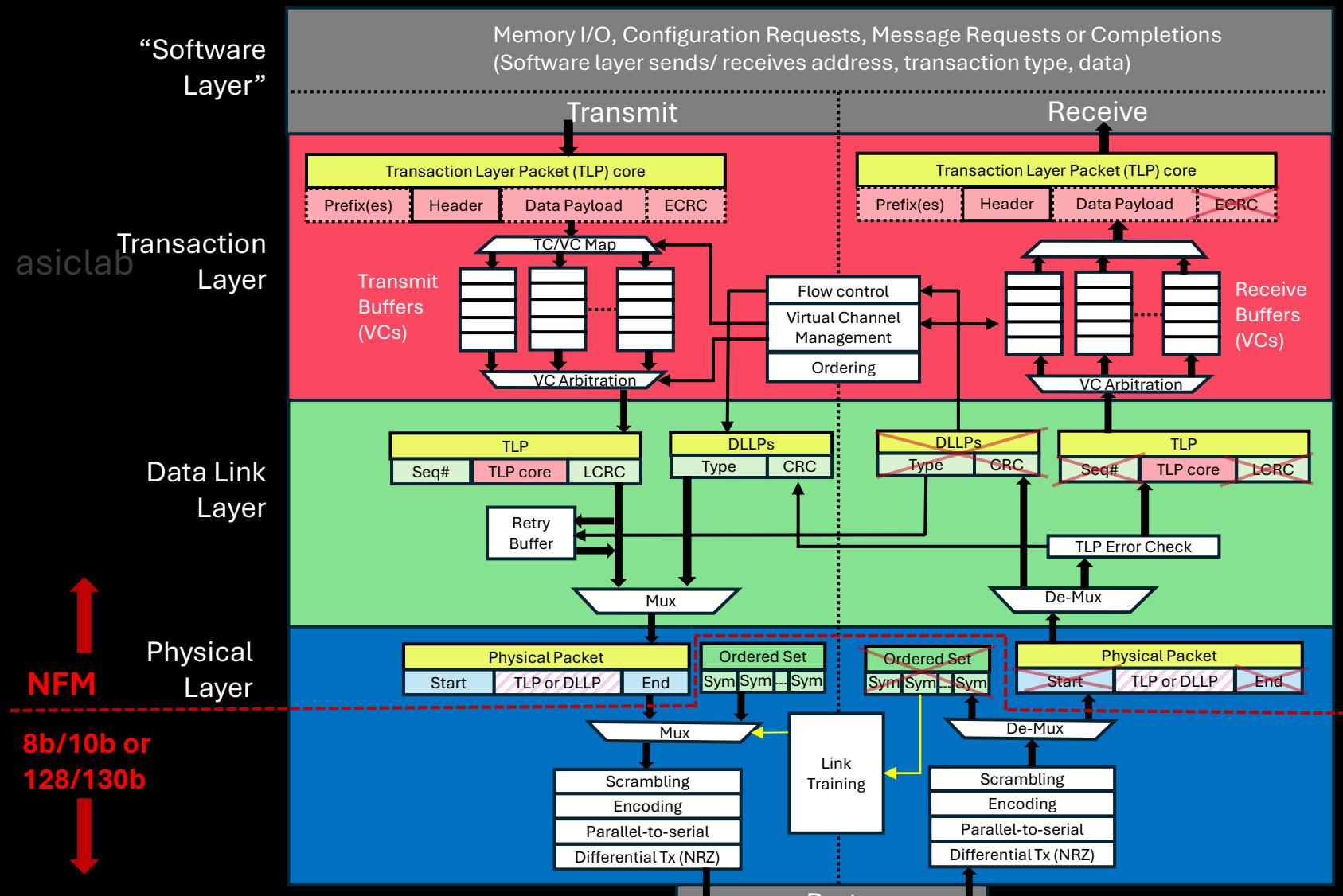
asiclab

Do Not Distribute

© Copyright protected 2024

asiclab
Learn. Thrive

PCIe Device Layer Details 5.0 Non Flit Mode (NFM)

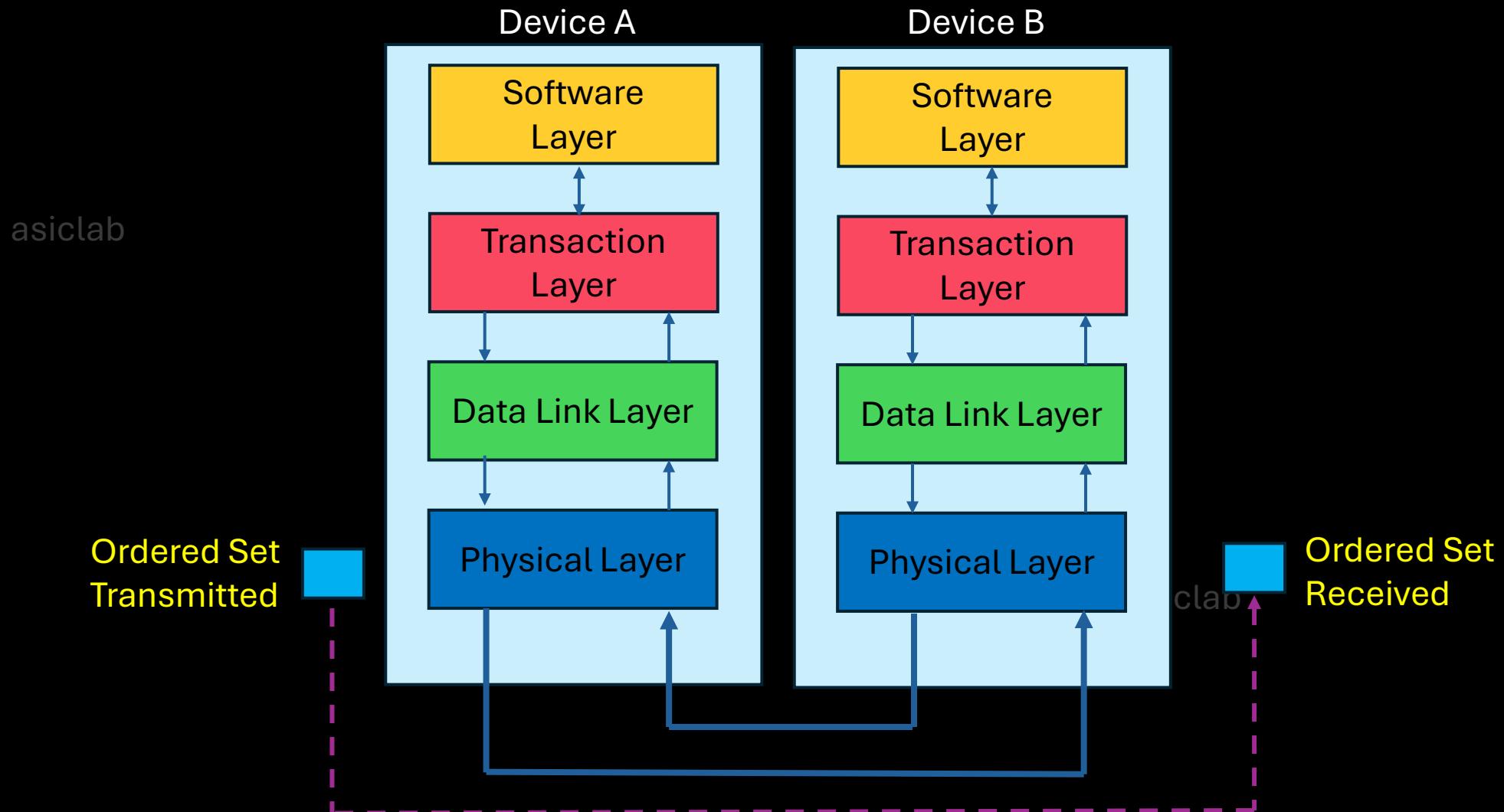


Do Not Distribute

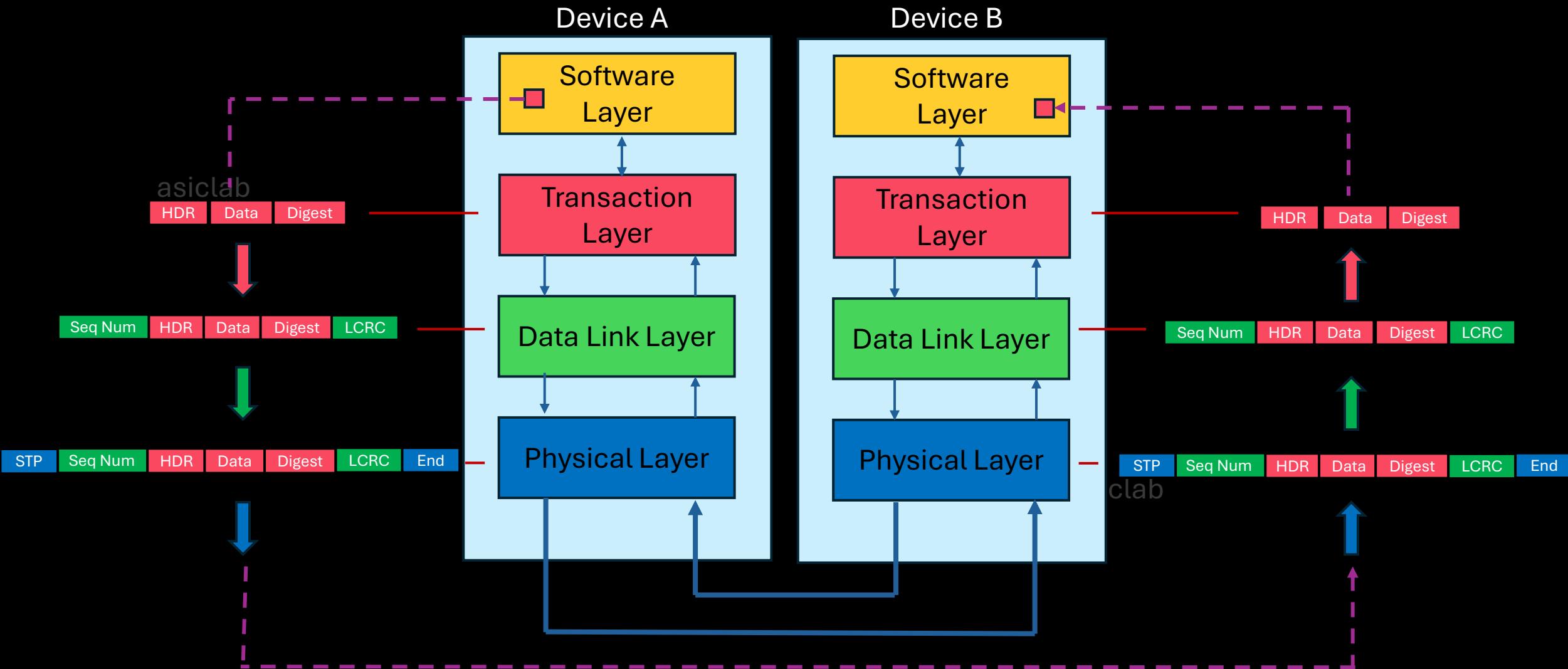
© Copyright protected 2024

Ordered Set Origin and Destination

(8b/10b) (128b/130b)



TLP Assembly/Disassembly (8b/10b) (128b/130b)



What is Link?

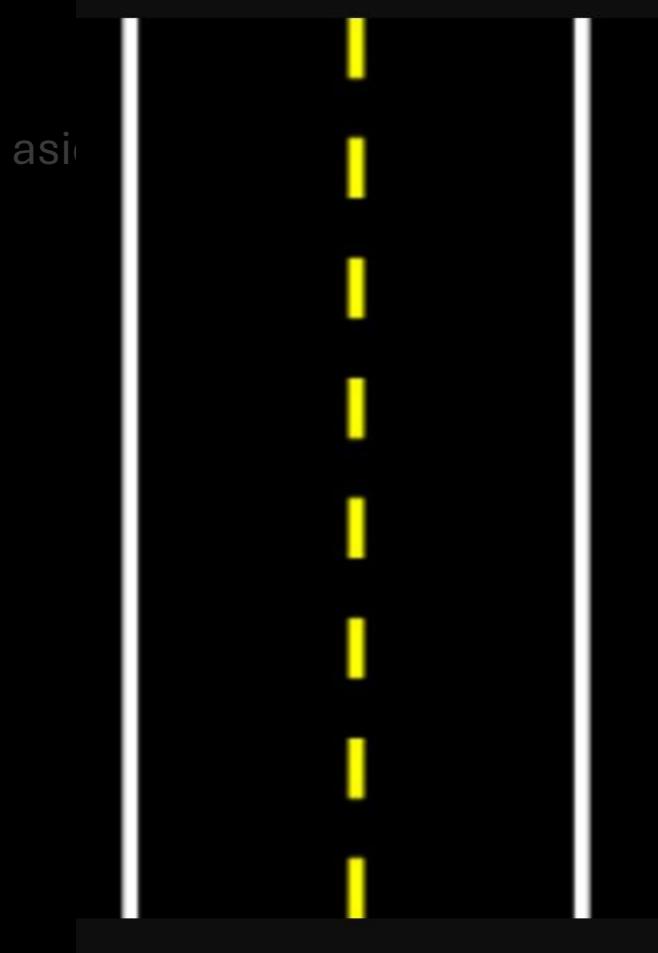
asiclab

lab

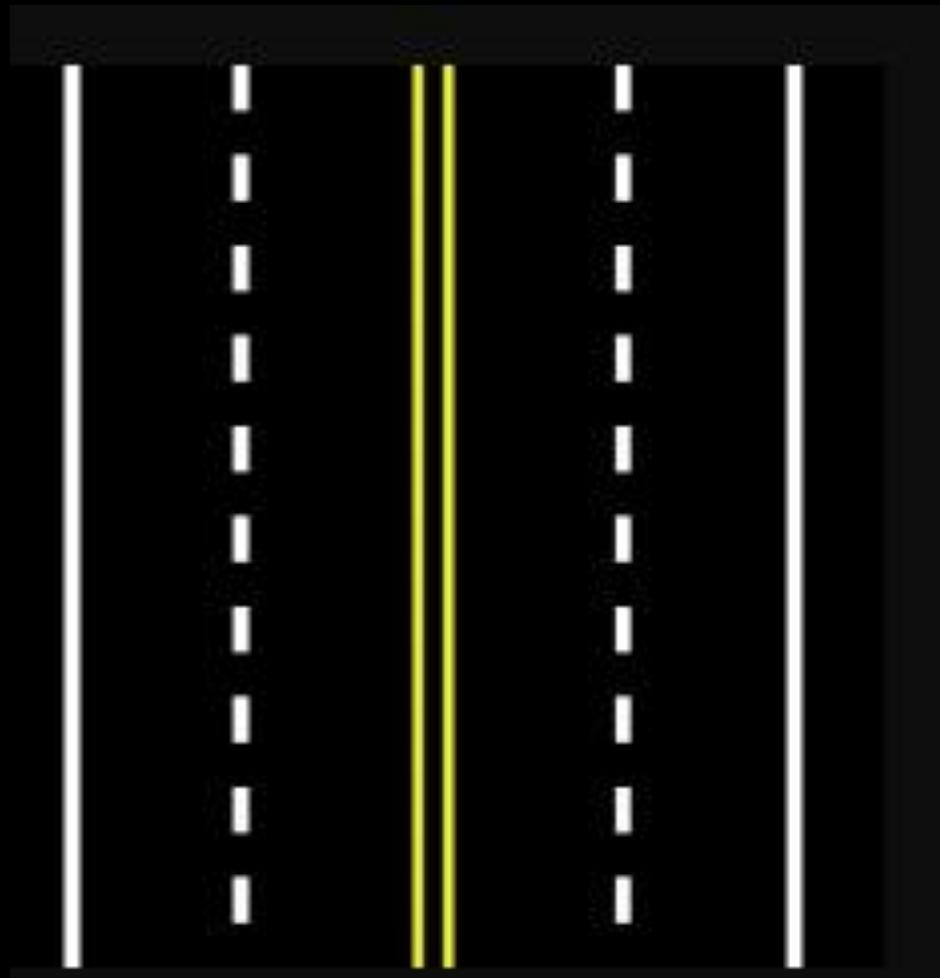


What is Lane?

x1 Lane



x2 Lanes



Lanes - Example

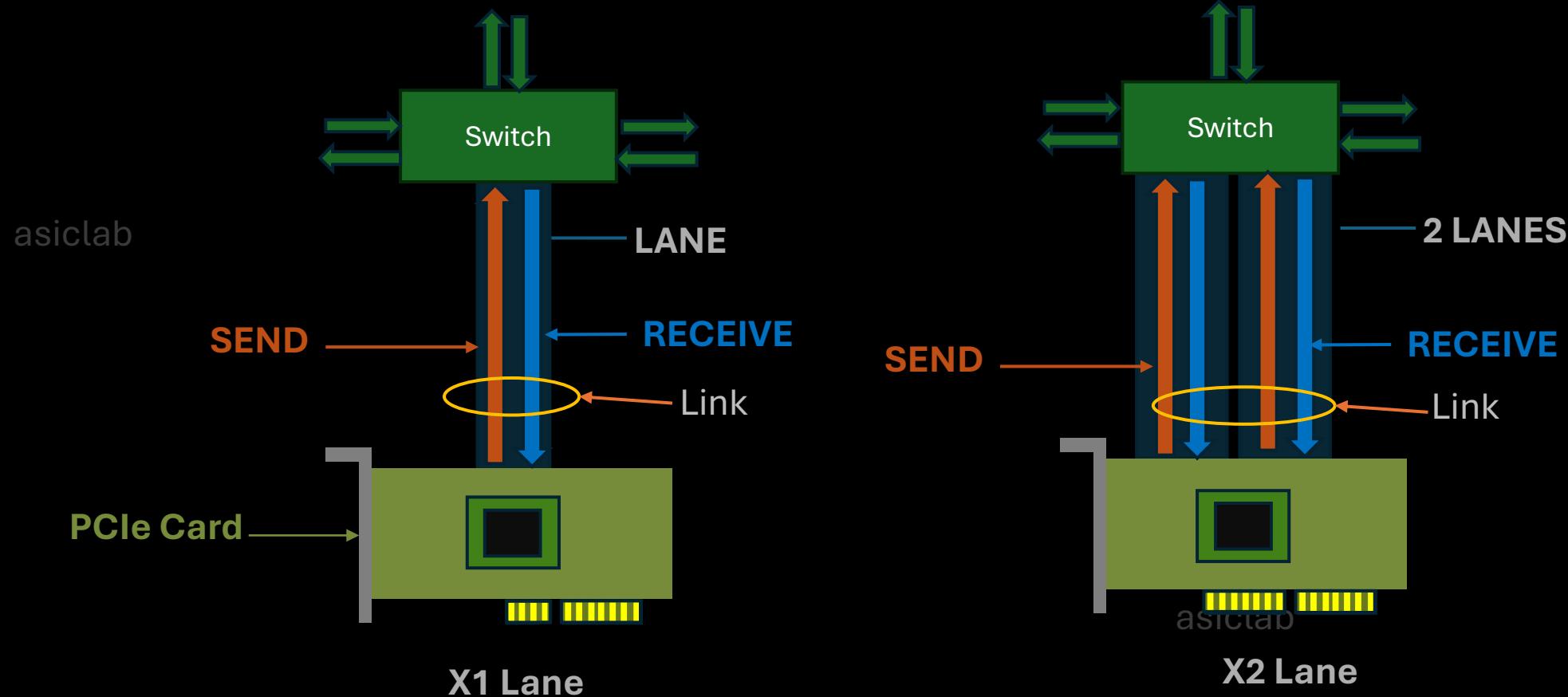
x4 Lane



x16 Lanes



Topology

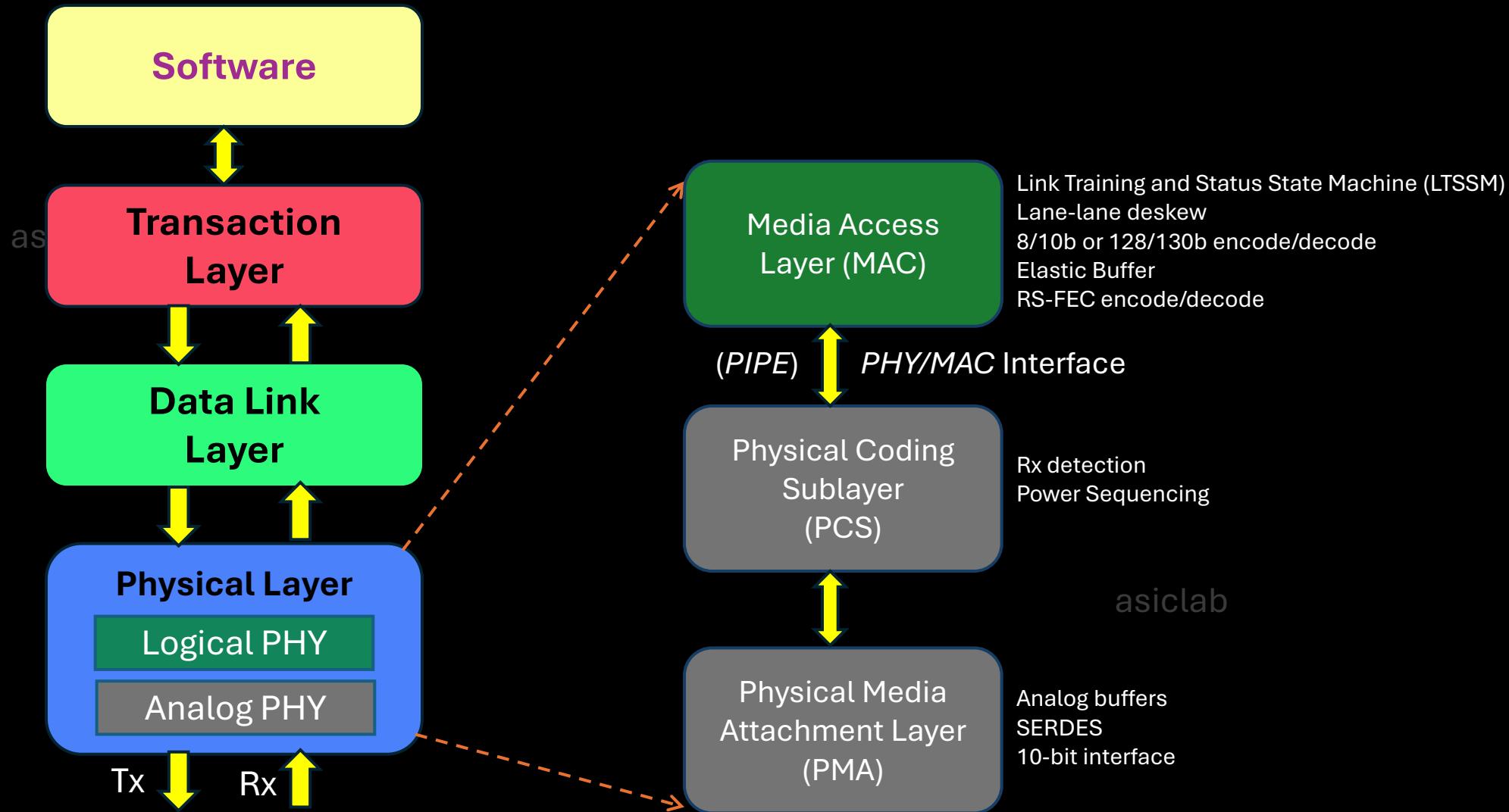


Flits - Example

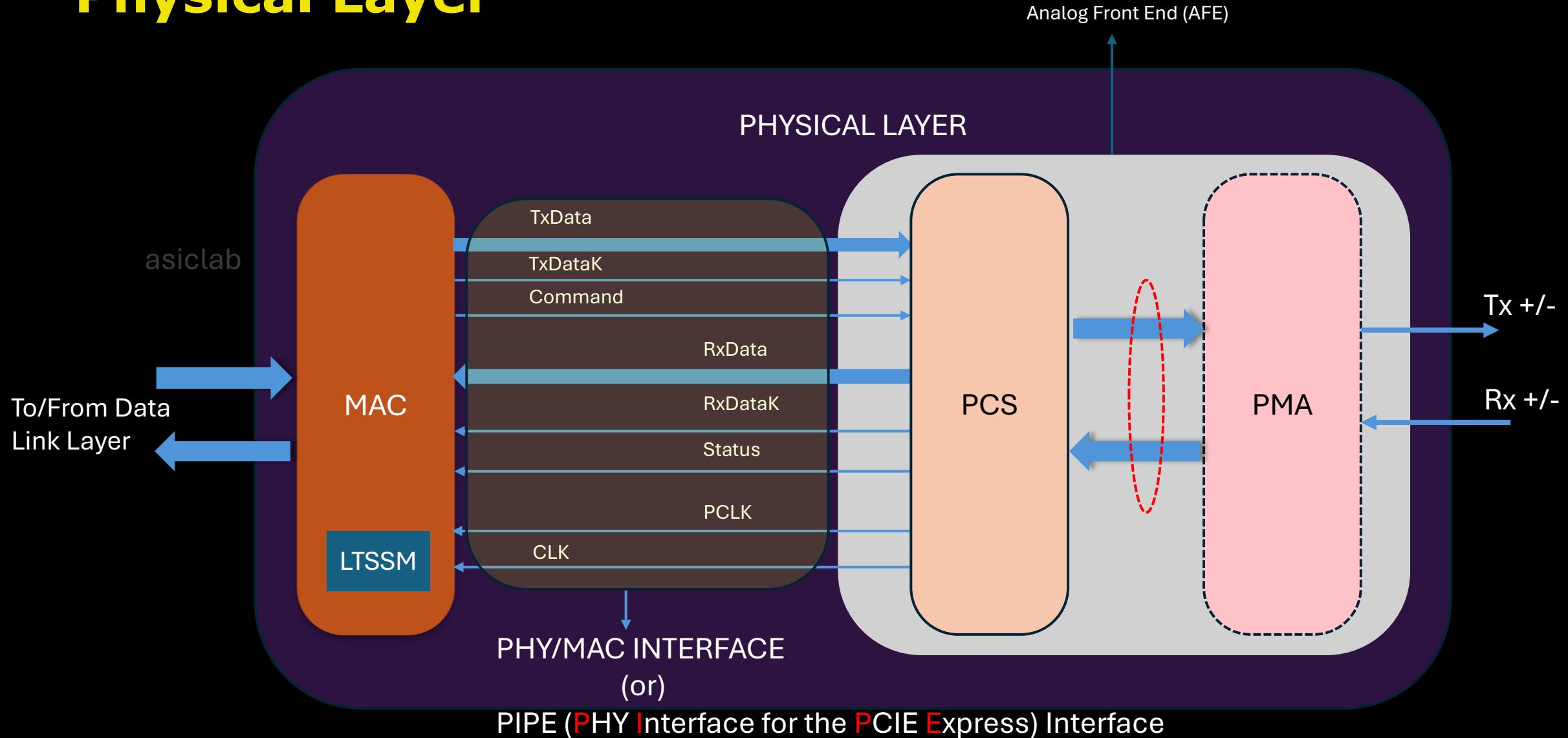
asiclab



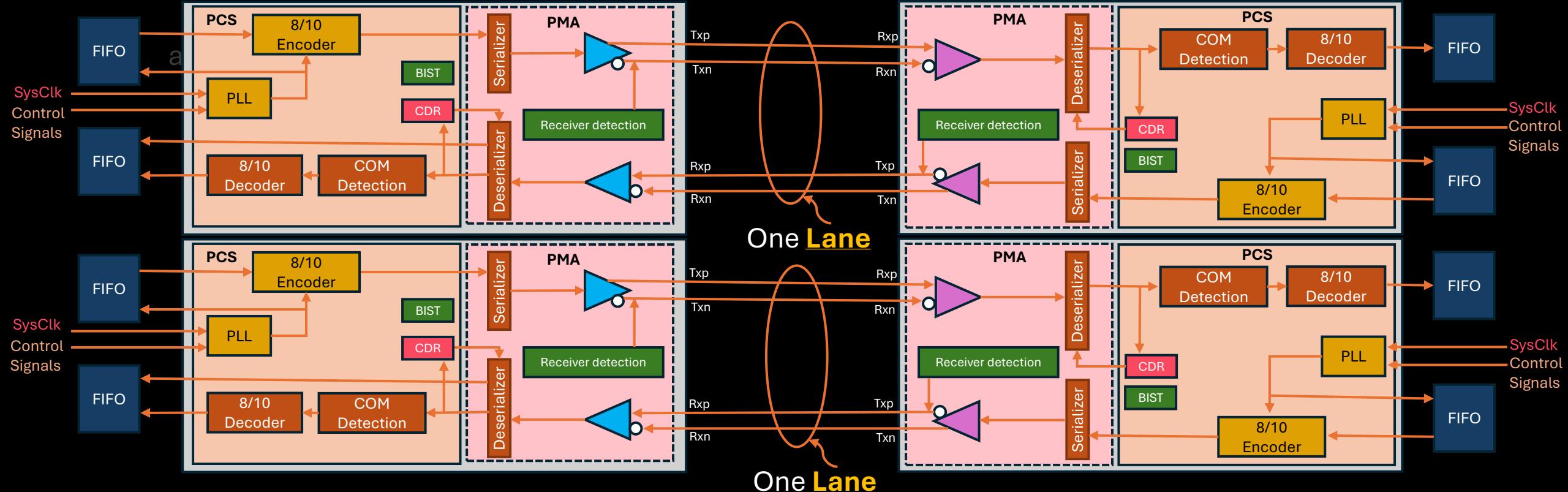
Physical Layer



Physical Layer



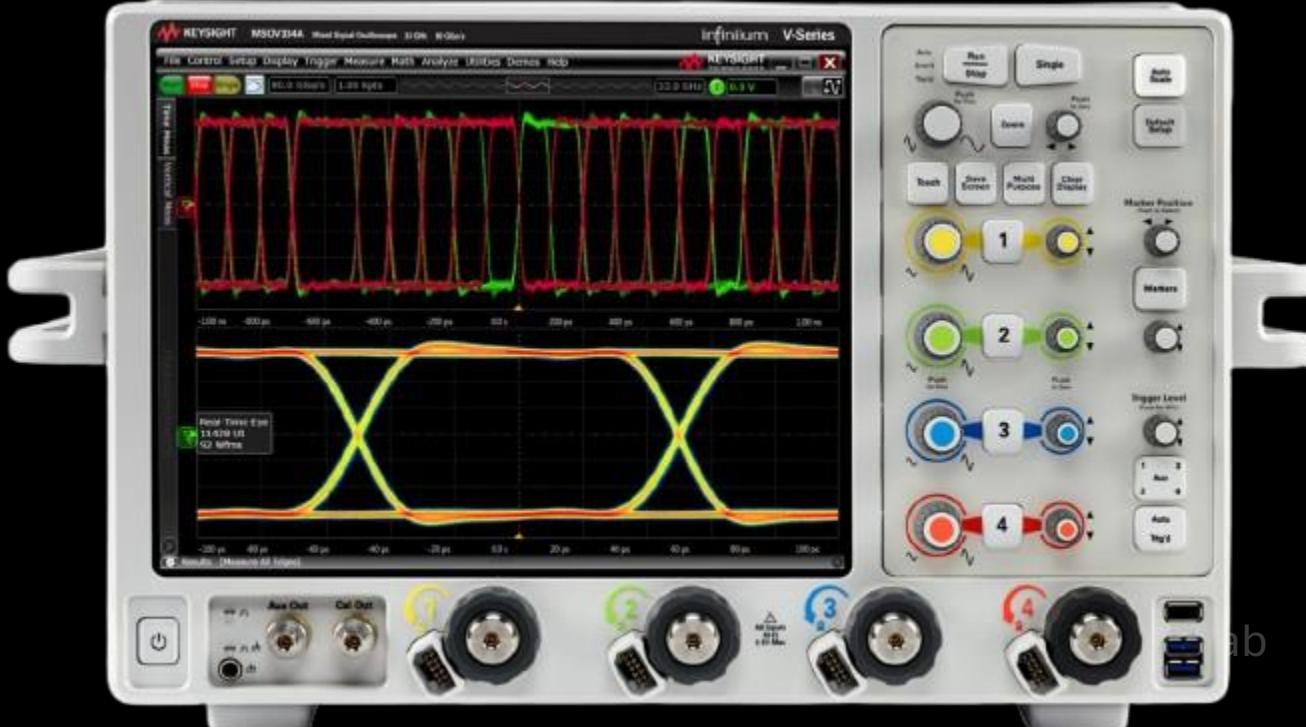
Physical Layer



Encoding

asiclab

NRZ



Eye Diagram

Reference: https://www.ti.com/content/dam/videos/external-videos/en-us/4/3816841626001/6053564299001.mp4/subassets/what_is_an_eye_diagram.pdf



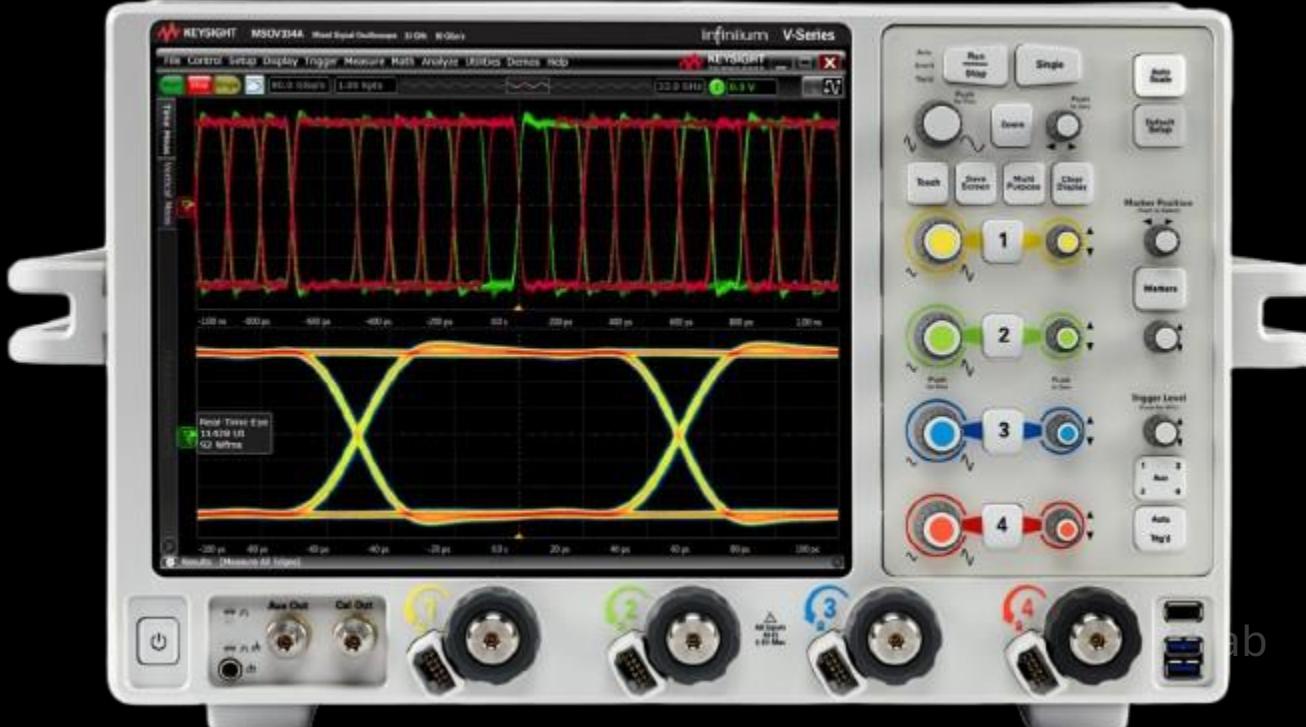
asiclab

asiclab

Encoding

asiclab

NRZ



Do Not Distribute

© Copyright protected 2024

asiclab
Learn. Thrive

Encoding

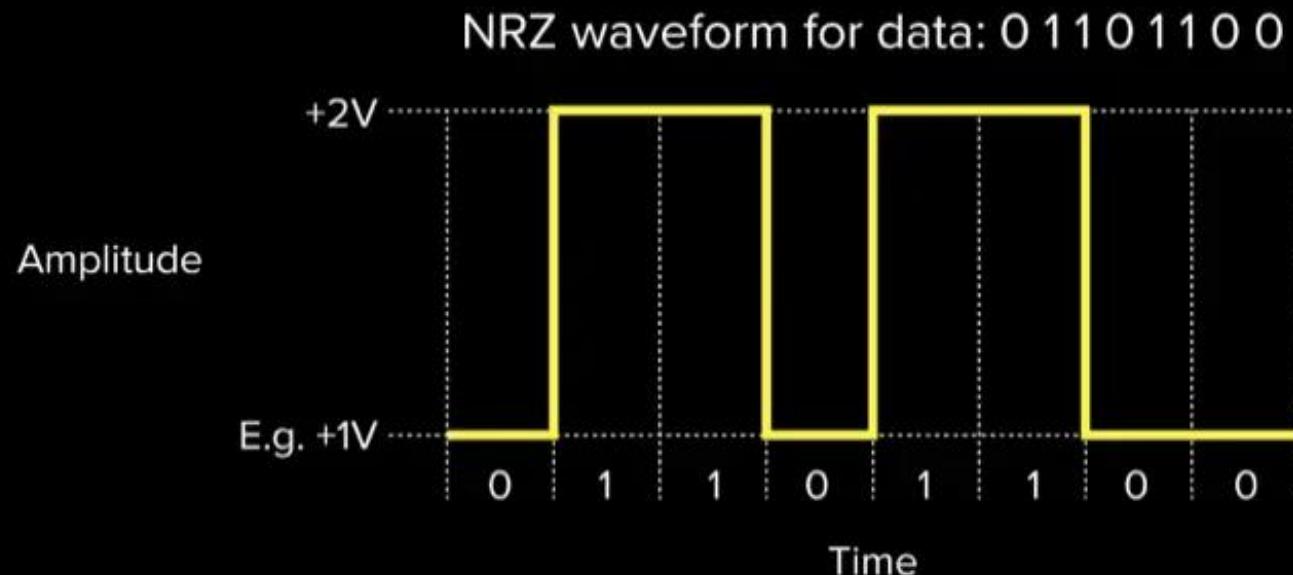
PAM4

asiclab

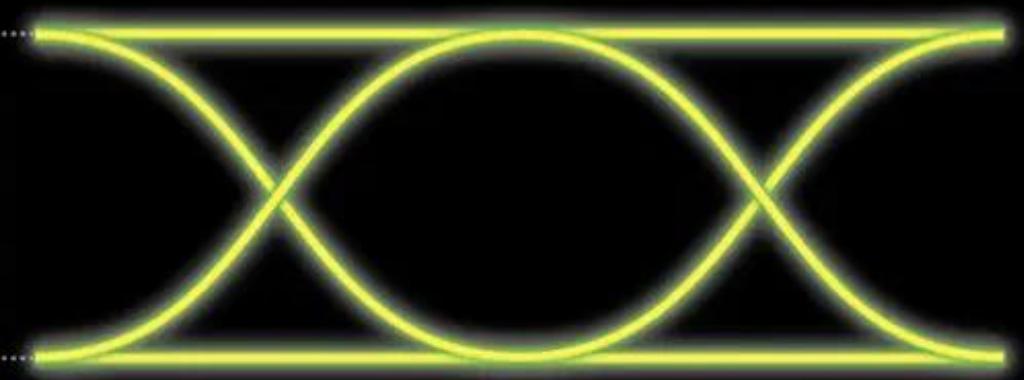


Encoding

NRZ

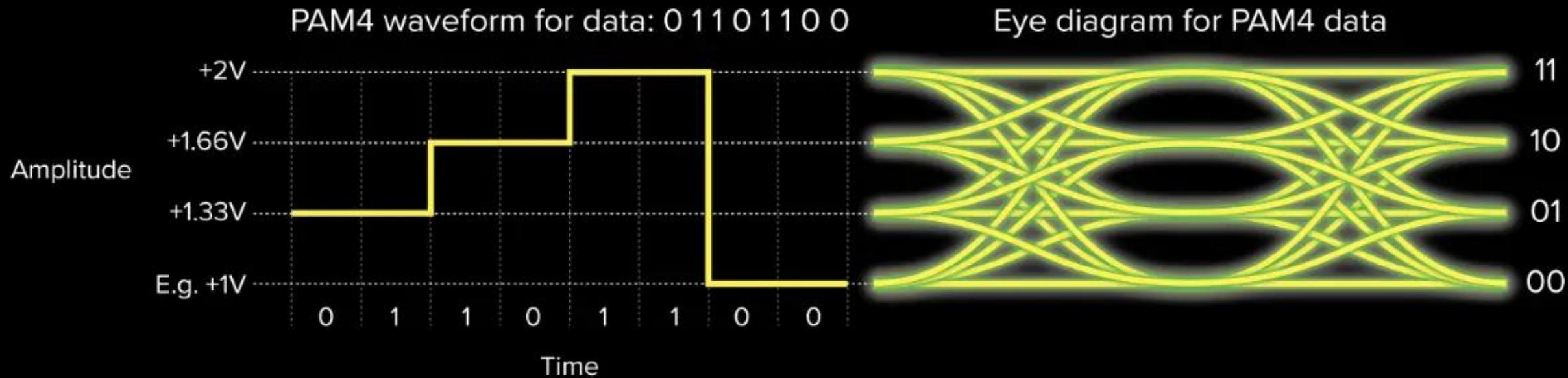


Eye diagram for NRZ data

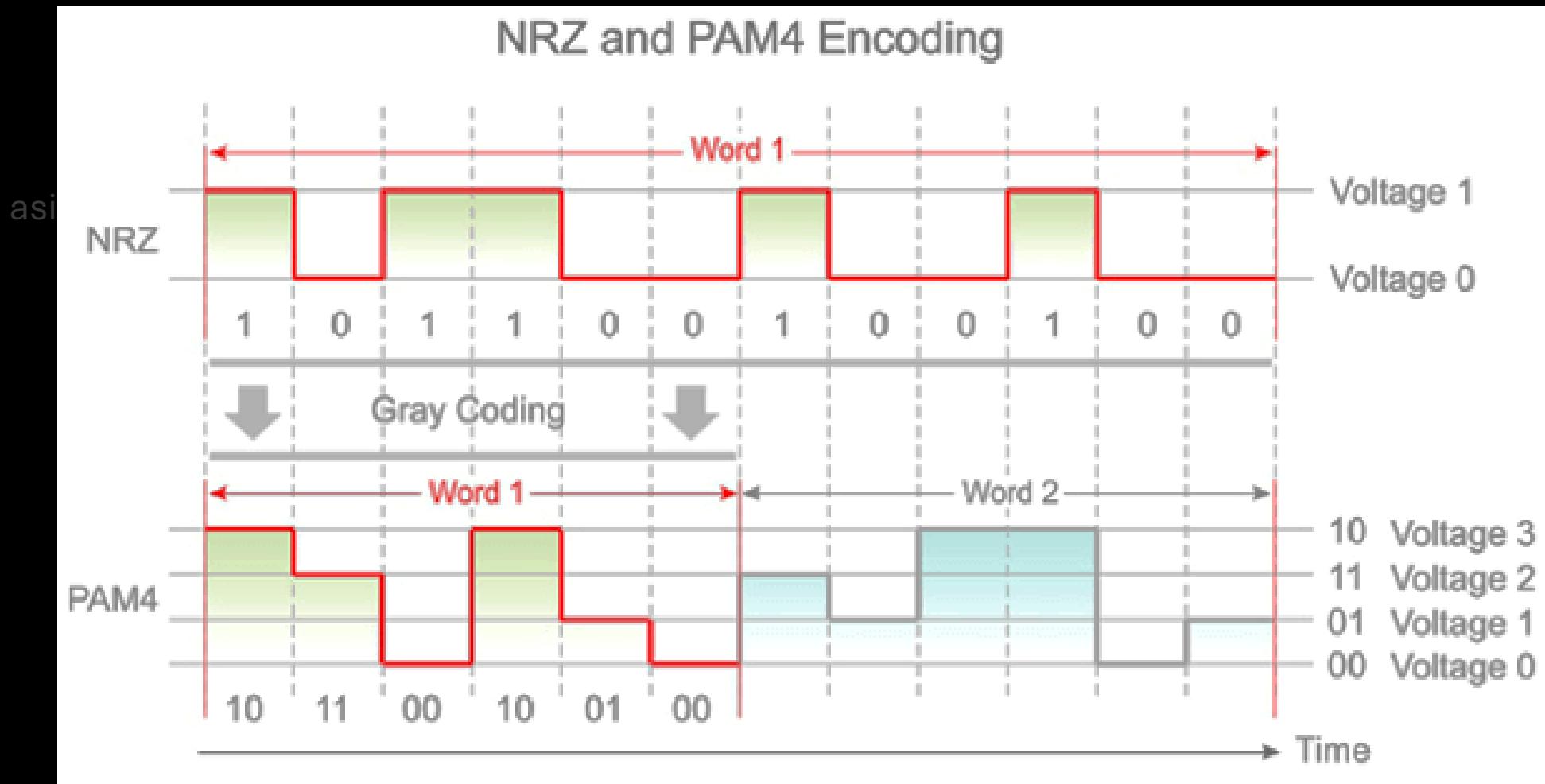


Encoding

PAM4

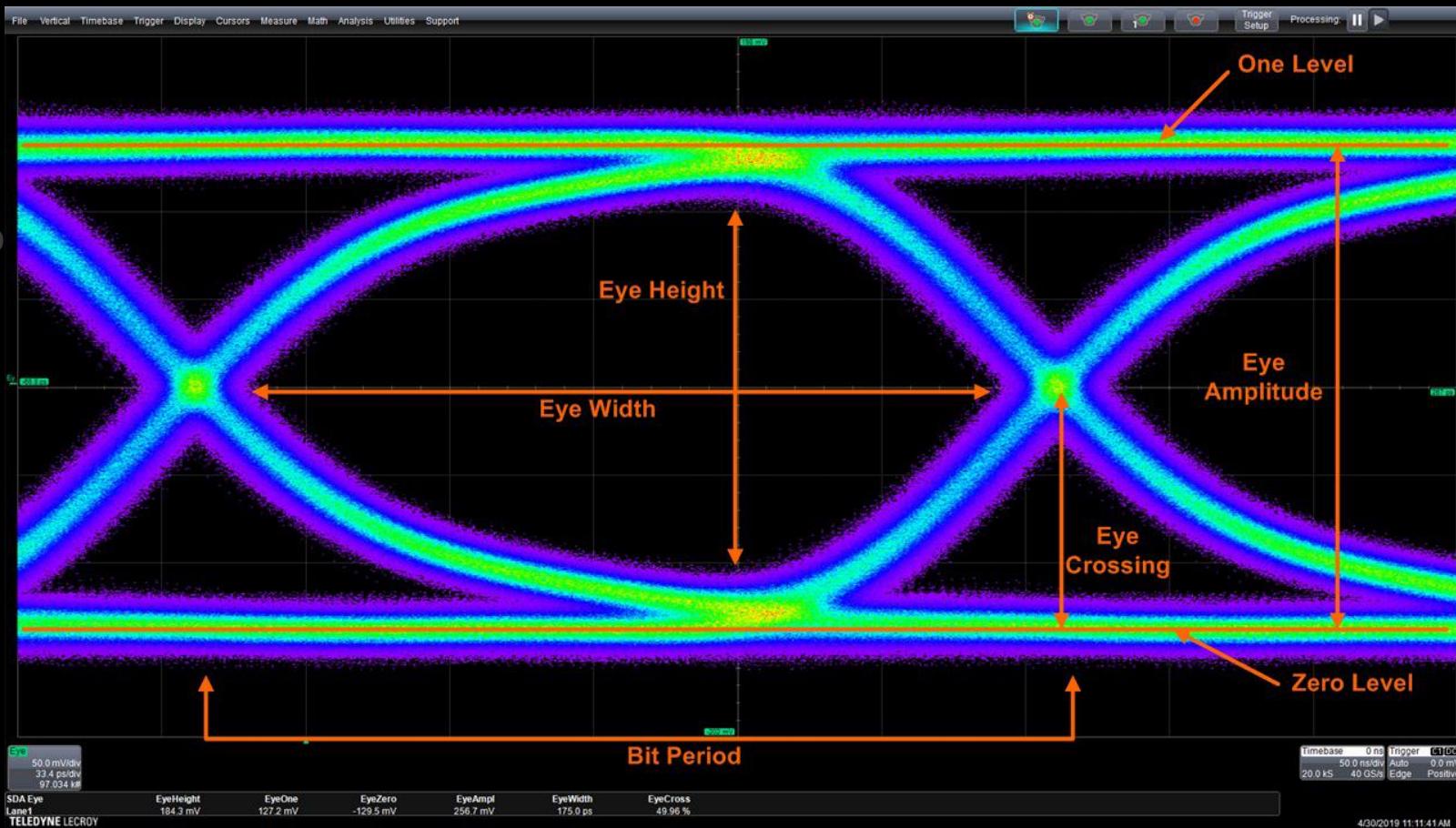


Encoding



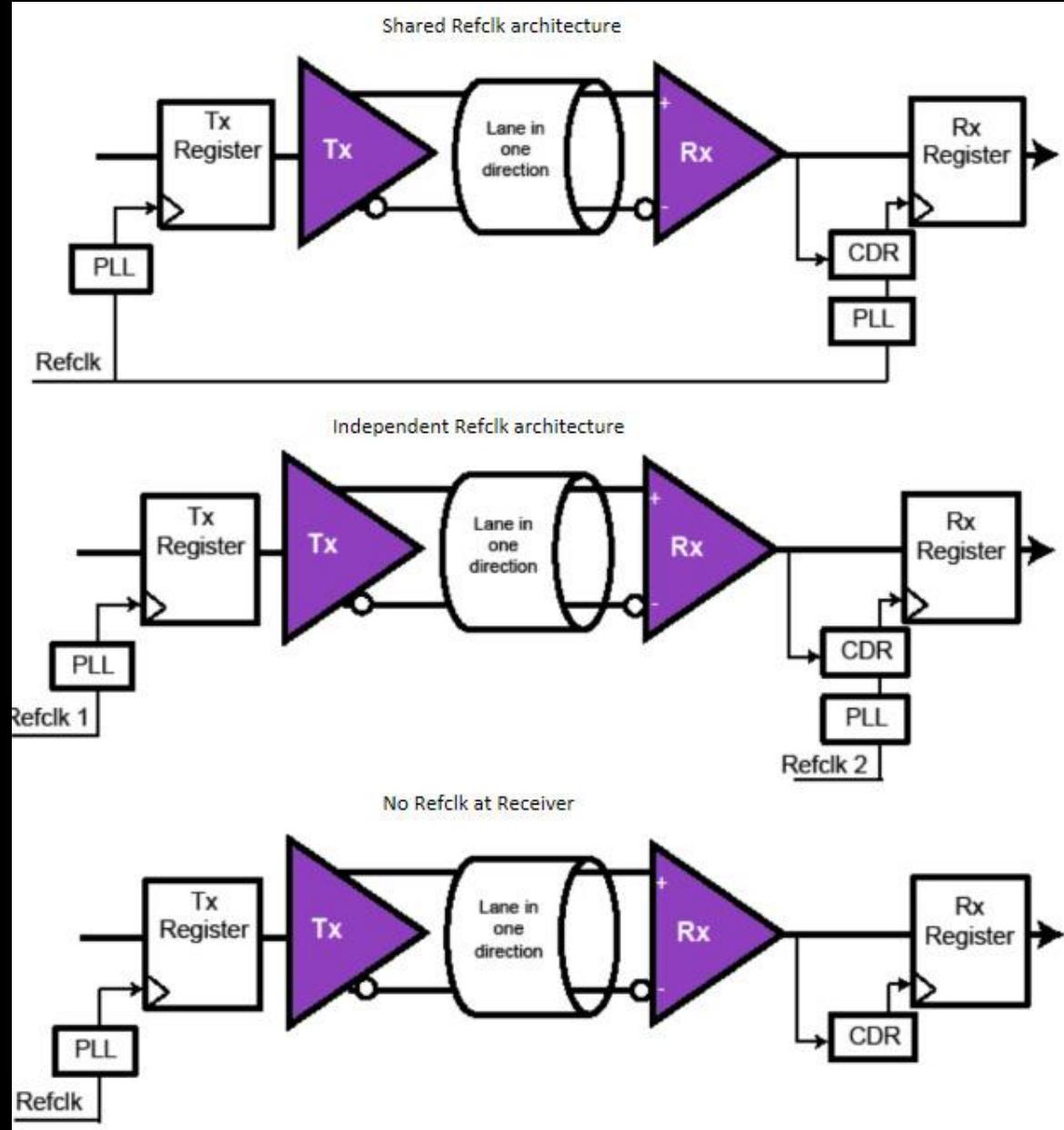
Eye Diagram Measurements

asiclab



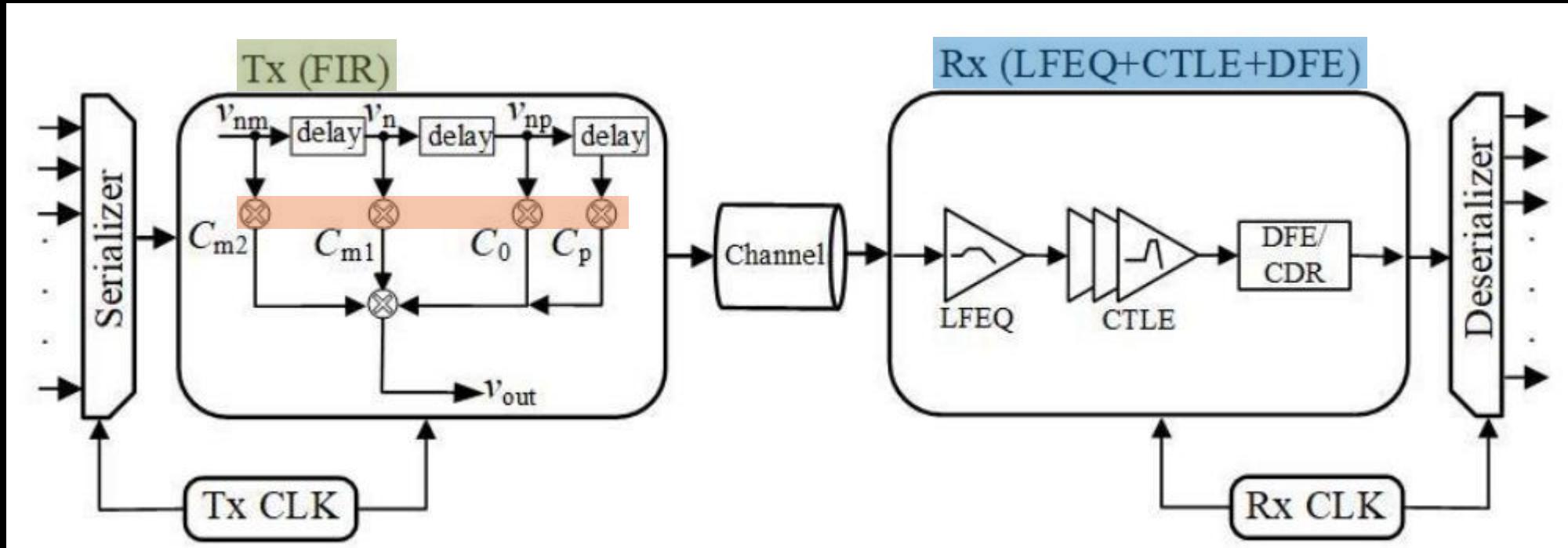
Clocking architecture

asiclab



Do Not Distribute

TX/ RX Design

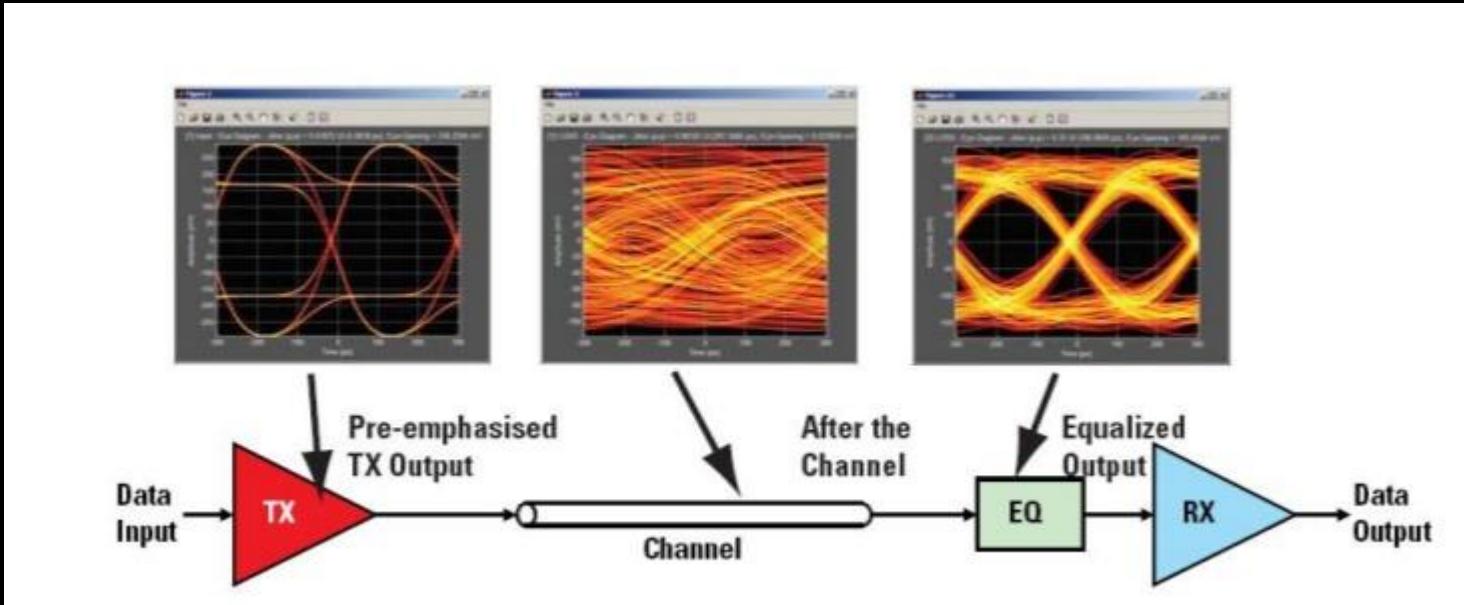


Reference: <http://journal.auric.kr/jsts/XmlViewer/f384698>



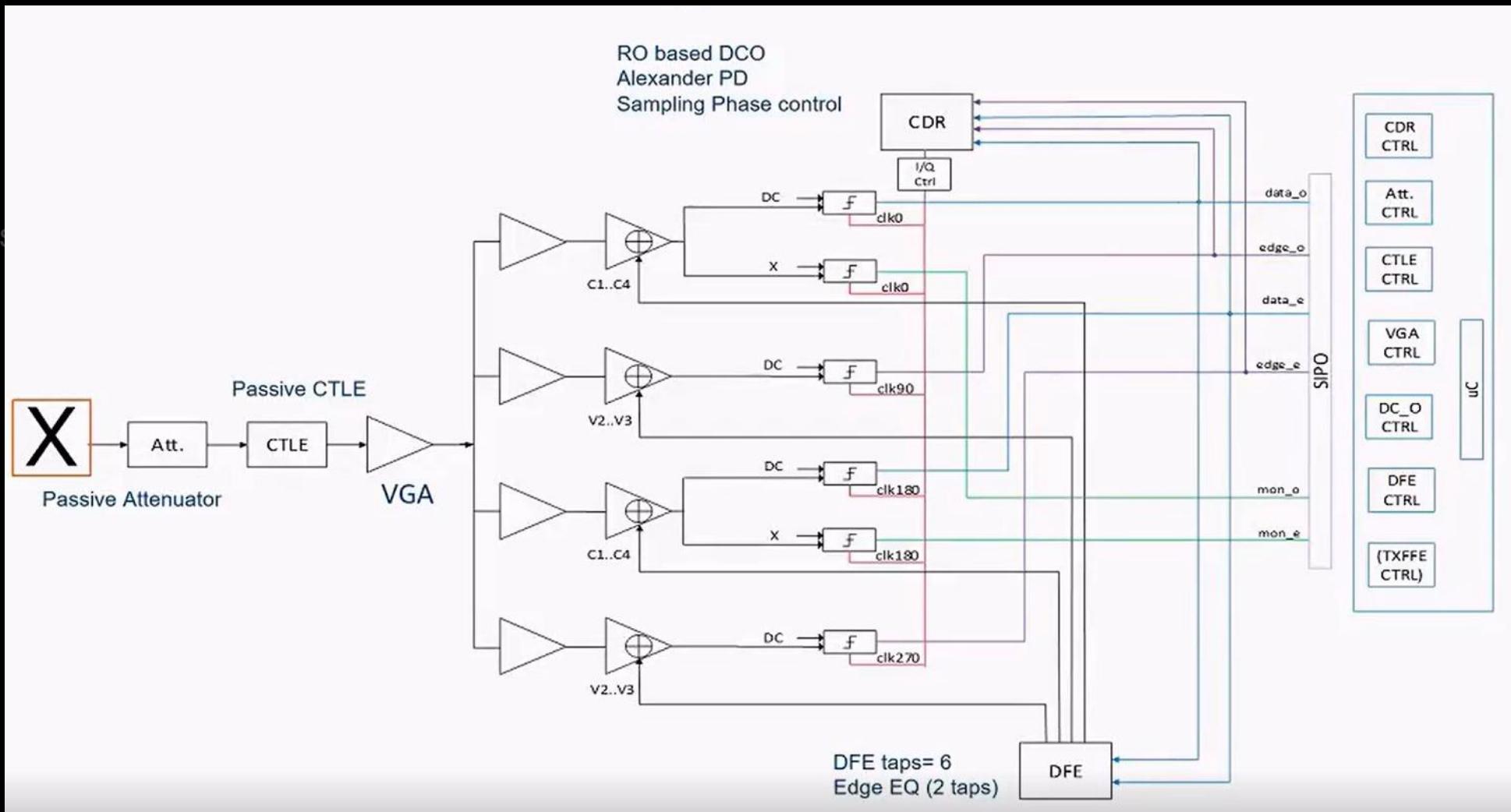
Equalization

asiclab



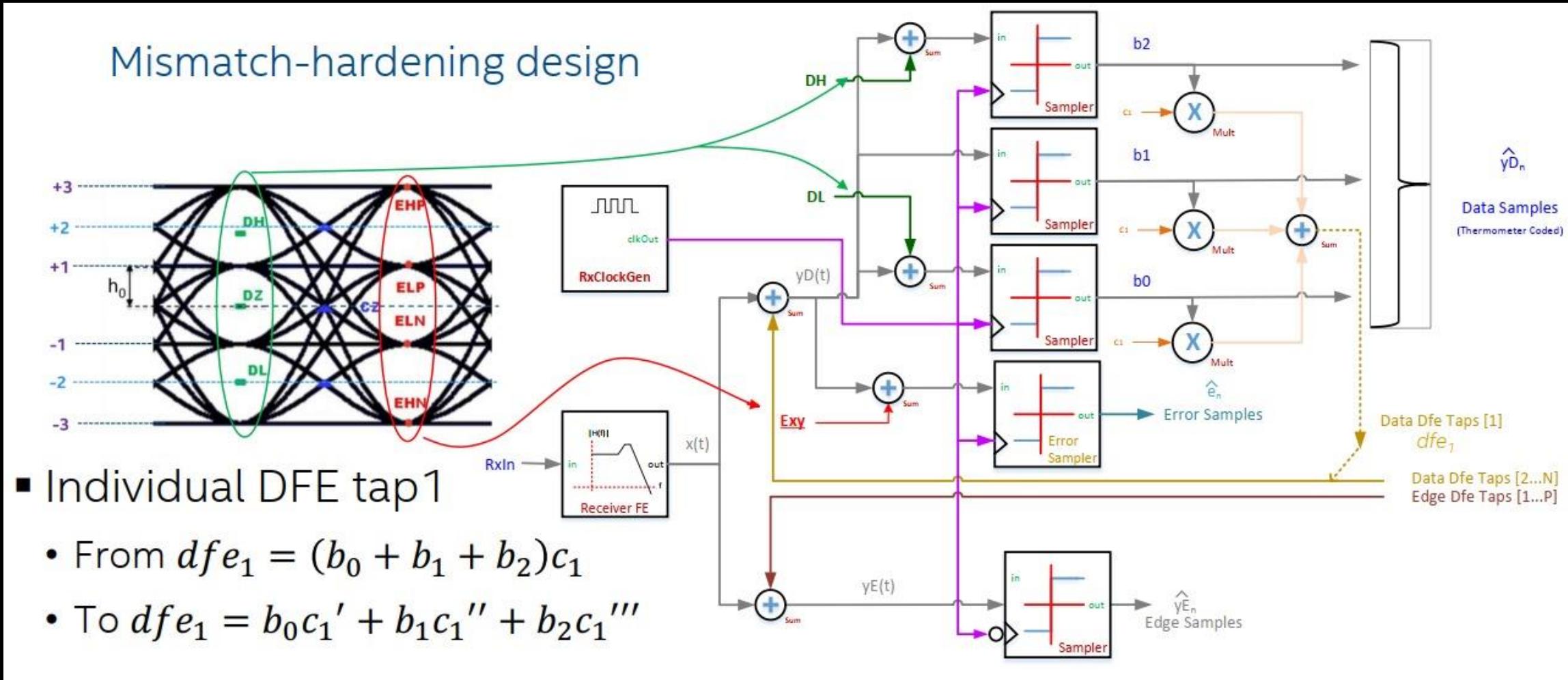
asiclab

Receiver - RX Circuit



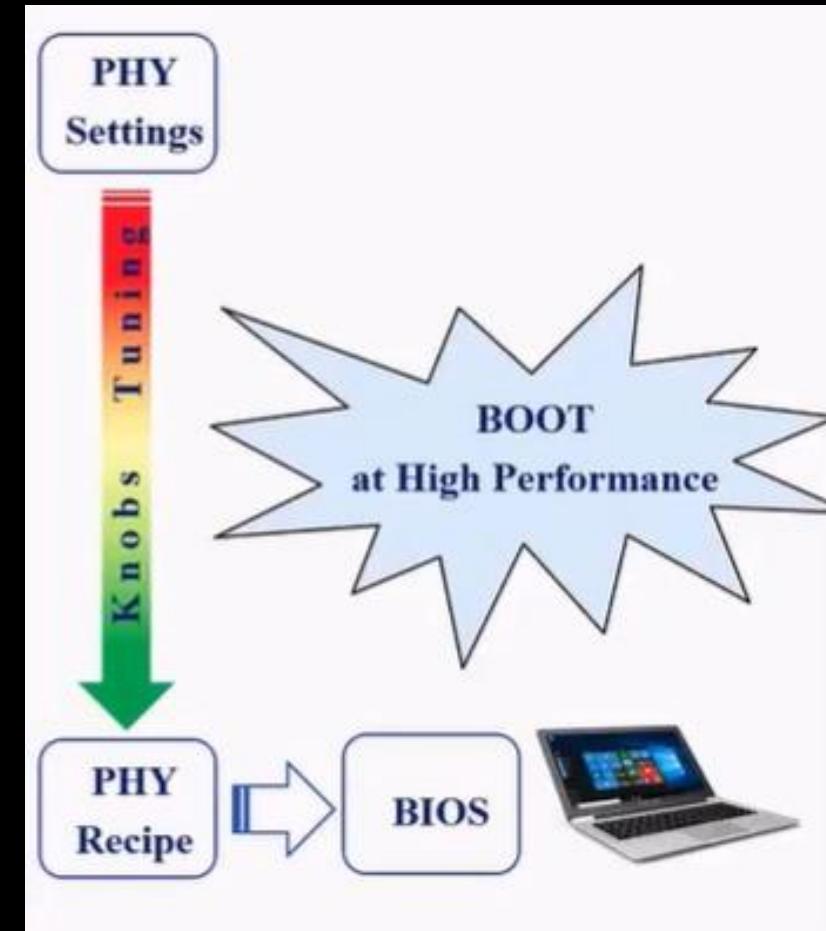
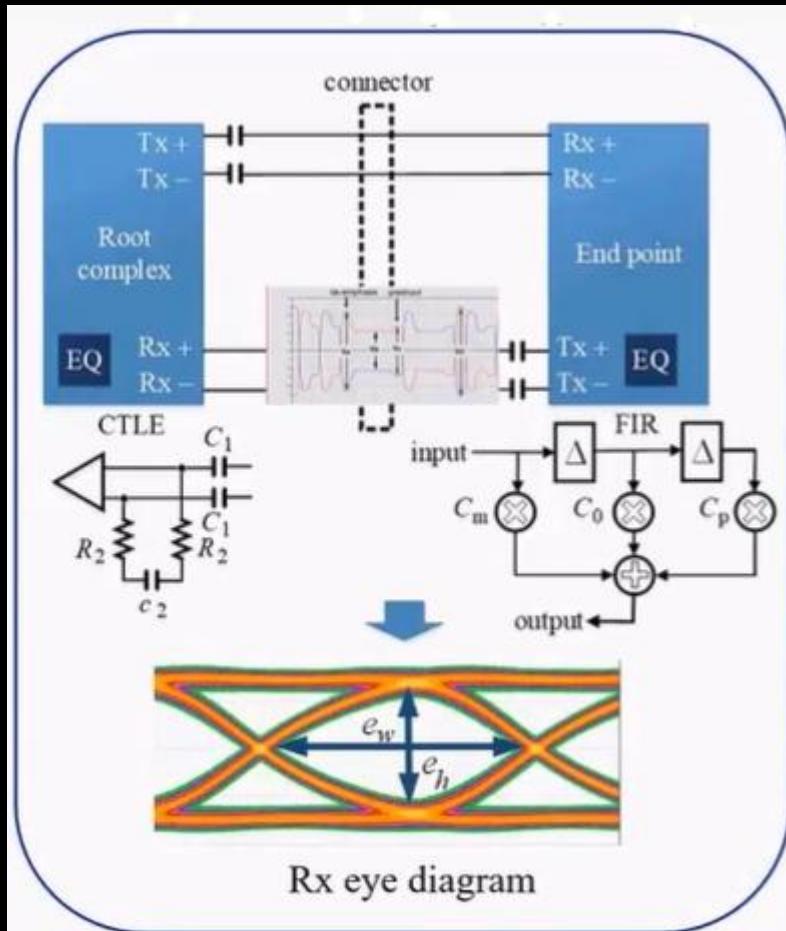
Decision Feedback Equalizer (DFE) Adaptation

Mismatch-hardening design



PCIe Post-Silicon Equalization process

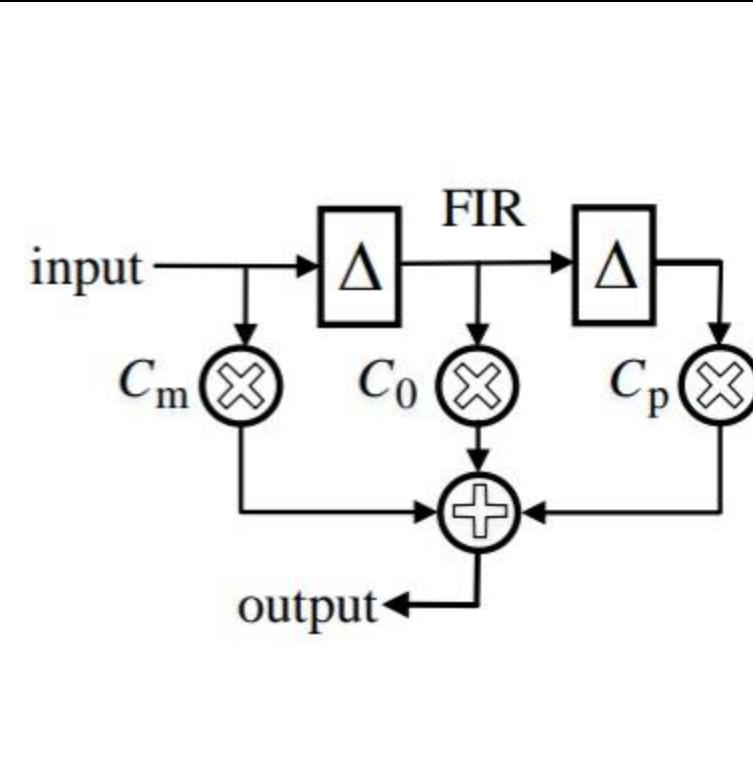
asiclab



Equalization Parameters

	C_p											
	0	2	4	6	8	10	12	14	16	18	20	22
C_m	0	0	3	0	3	0	2	1	0	0	0	0
	2	4	6	4	6	0	4	5	8	5	5	3
	4	5	5	5	6	7	9	7	7	6	7	
	6	6	8	8	9	9	11	11	10	9		
	8	10	8	10	12	10	10	10	10			
	10	10	10	9	10	9	9	9	8			
	12	10	8	8	7	8	6					
	14	8	7	5	5	5						
	16	7	7	5	4							

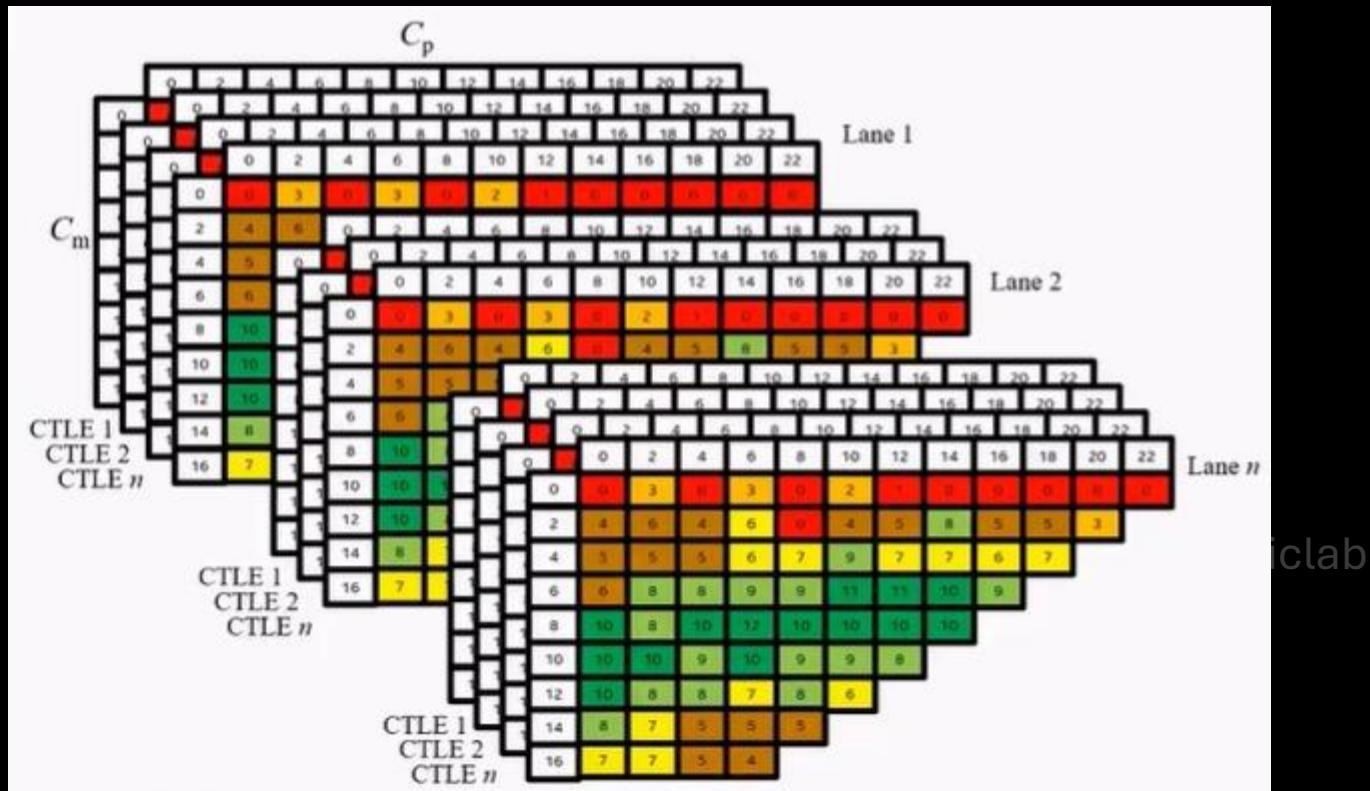
e_w / e_h measurements



Current PCIe Tuning Methodology

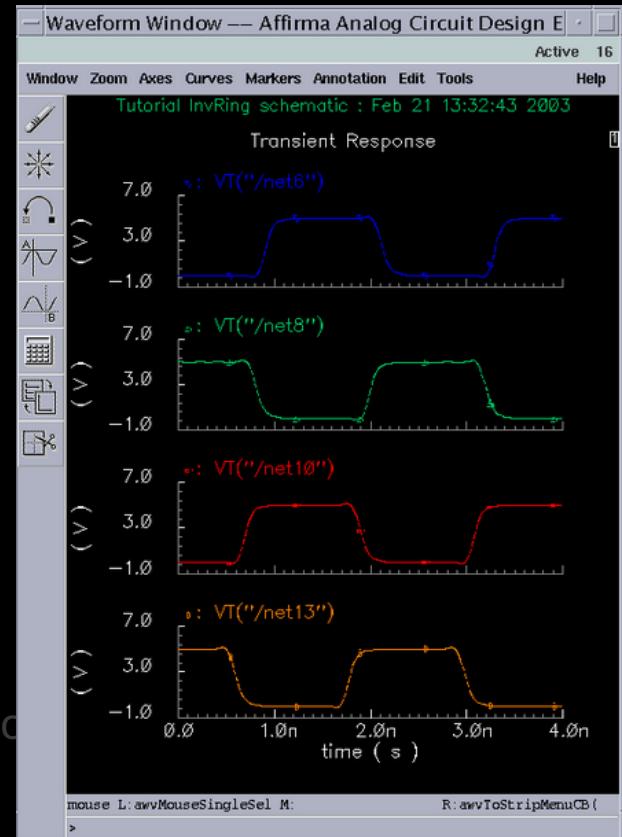
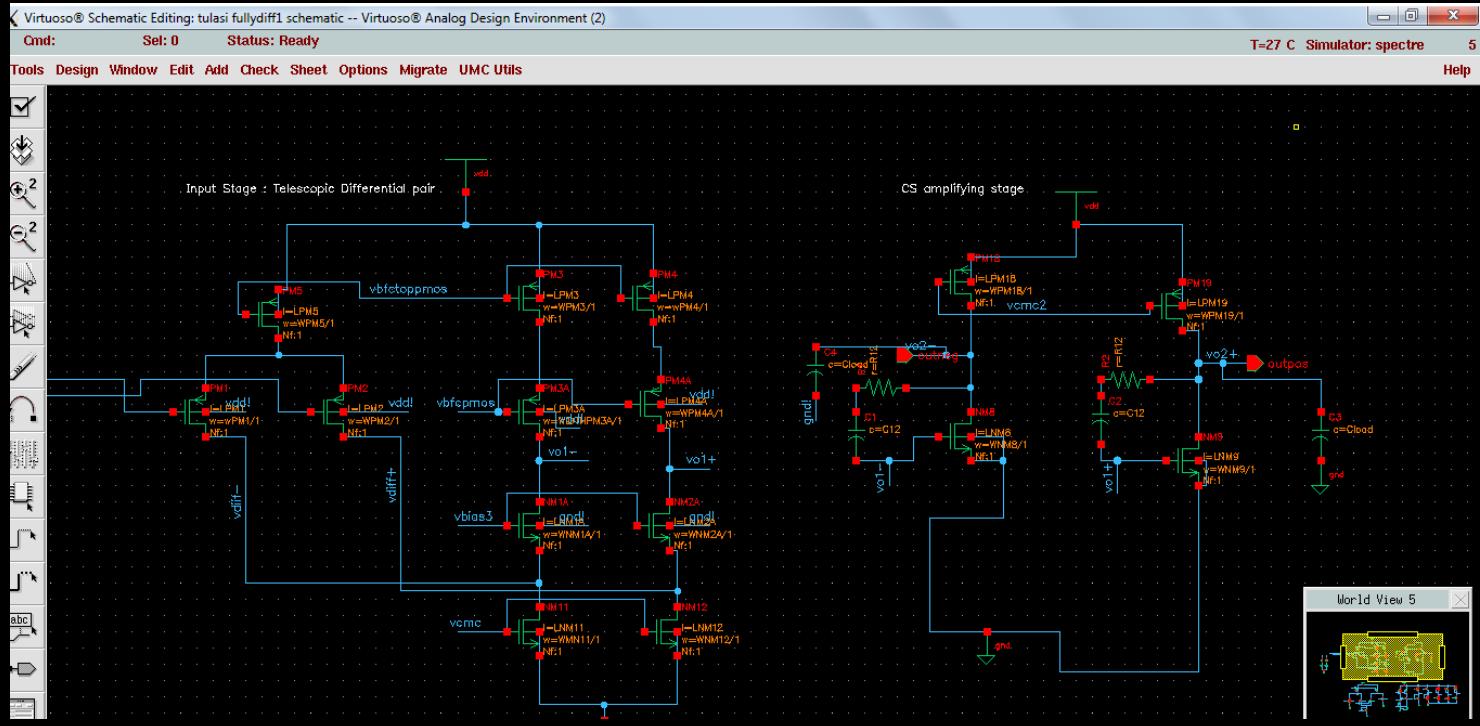
- Number of EQs Maps increase exponentially when considering CTLW values and number of lanes

asiclab



asiclab

Analog Design Example



When Equalization?

1. When link is out of detect → Gen 3 trained first time

- Out of Platform reset (cold reset / warm reset / Sx)
- Link disable / enable
- Hot Reset
- asL2 exit (RTD2 flow – exit)
- Hot Plug

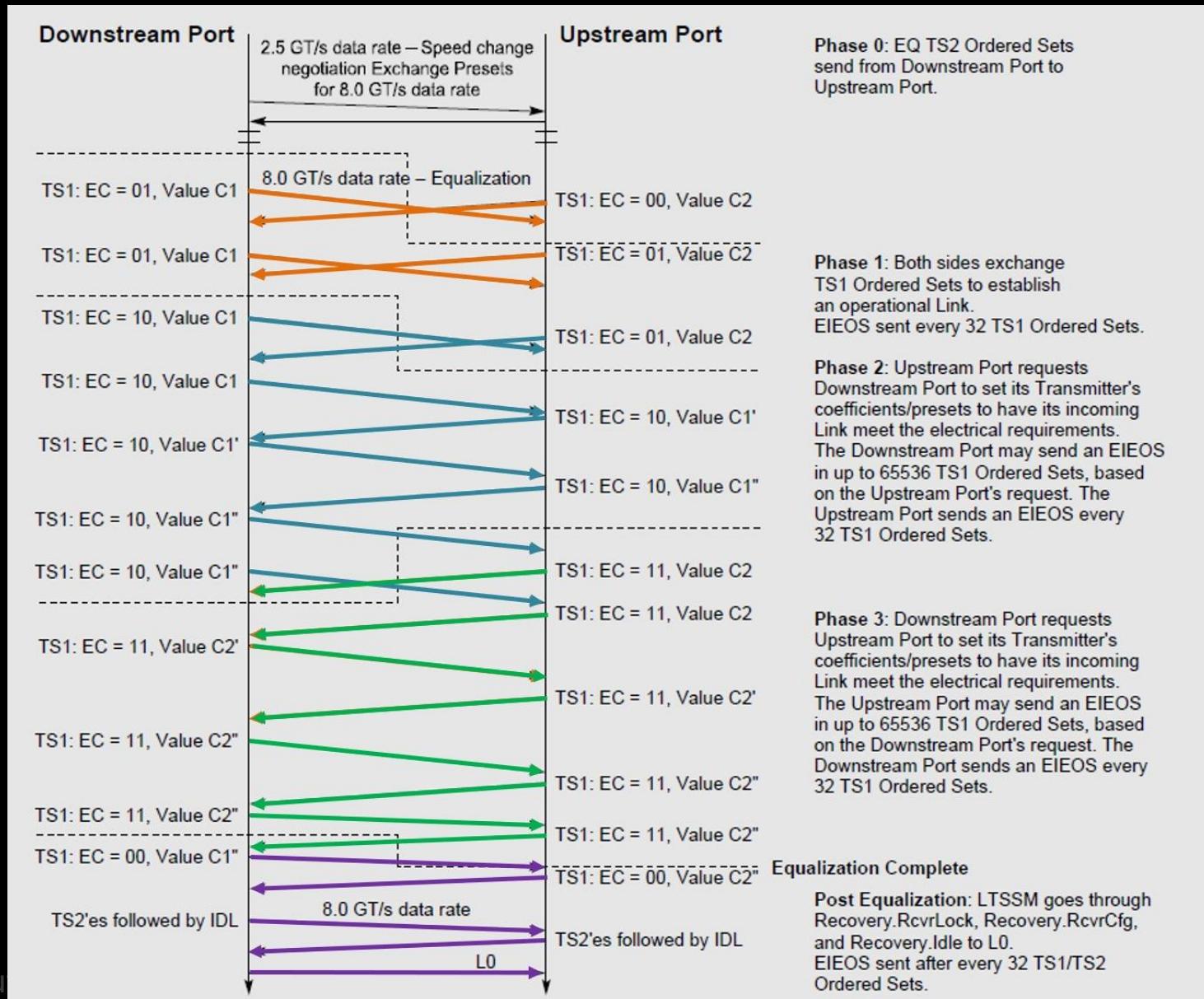
2. Redo –equalization

- Downstream port LCTL3.DOEQ=1 followed by LCTC.RL=1
- Upstream port does equalization request (Quiesce Guarantee bit set TS1(?) / TS2(?))

Detect → Polling → Config → L0 (Gen1) → Recovery (R.RcvrLock) → R.RcvrCfg(DP sends EQTS2) → Initiate EQ → Squelch(R.speed) → Recovery.Eqph0/ph1/ph2/ph3 → R.RcvrLock → R.RcvfCfg → R.idle → L0(Gen3)

Equalization

asiclab



Do Not Distribu

Equalization

PHY Equalization complexity increases as the data rate increases

Tx and Rx both have Equalizations

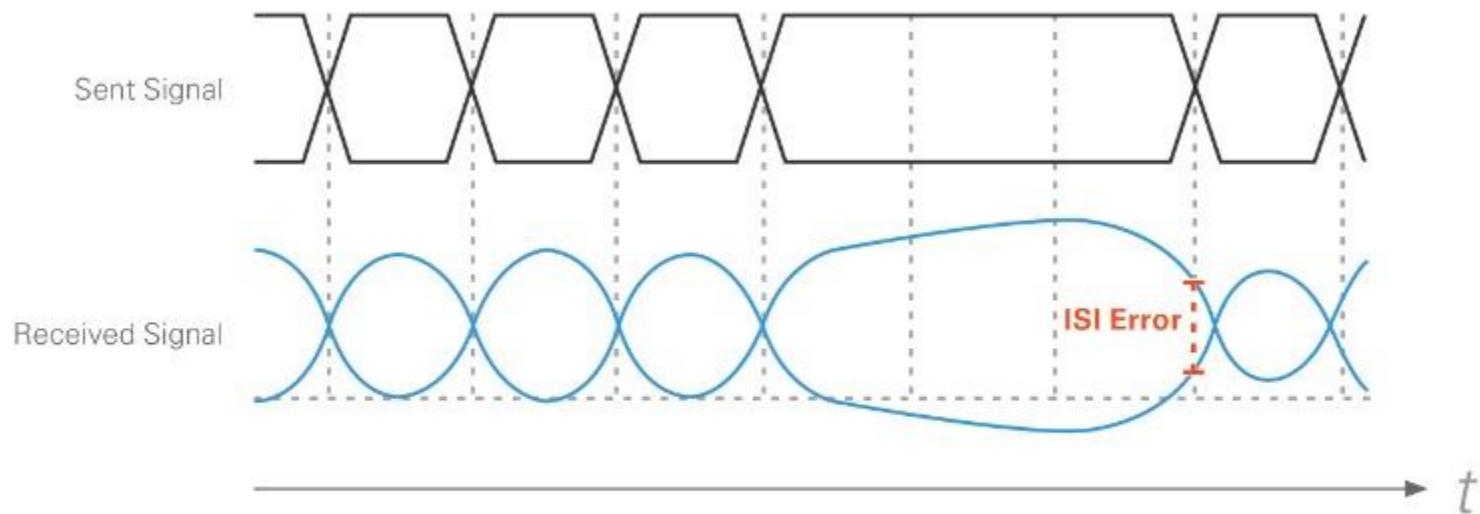
Tx is fairly fixed EQ, some amount of flexibility with phase 3 EQ to search within given set of EQ settings (~5) and choose best one

RX EQ are complex in nature with the threat of noise amplification and error passing, careful design techniques needed. Few popular RX EQ techniques are

- Continuous Time Linear Equalization with Pole – Zero
- Decision Feedback Equalization for Post Cursor cancellations
- Sampling Phase adjustment for Pre Cursor cancellation with CDR sampling position adjustments.

asiclab

Signal degradation



Signal degradation due to ISI is caused when a lack of bit transitions causes the line to build a bias, making it harder for the first transition to the opposite bit to occur quickly.

Two ways to overcome ISI

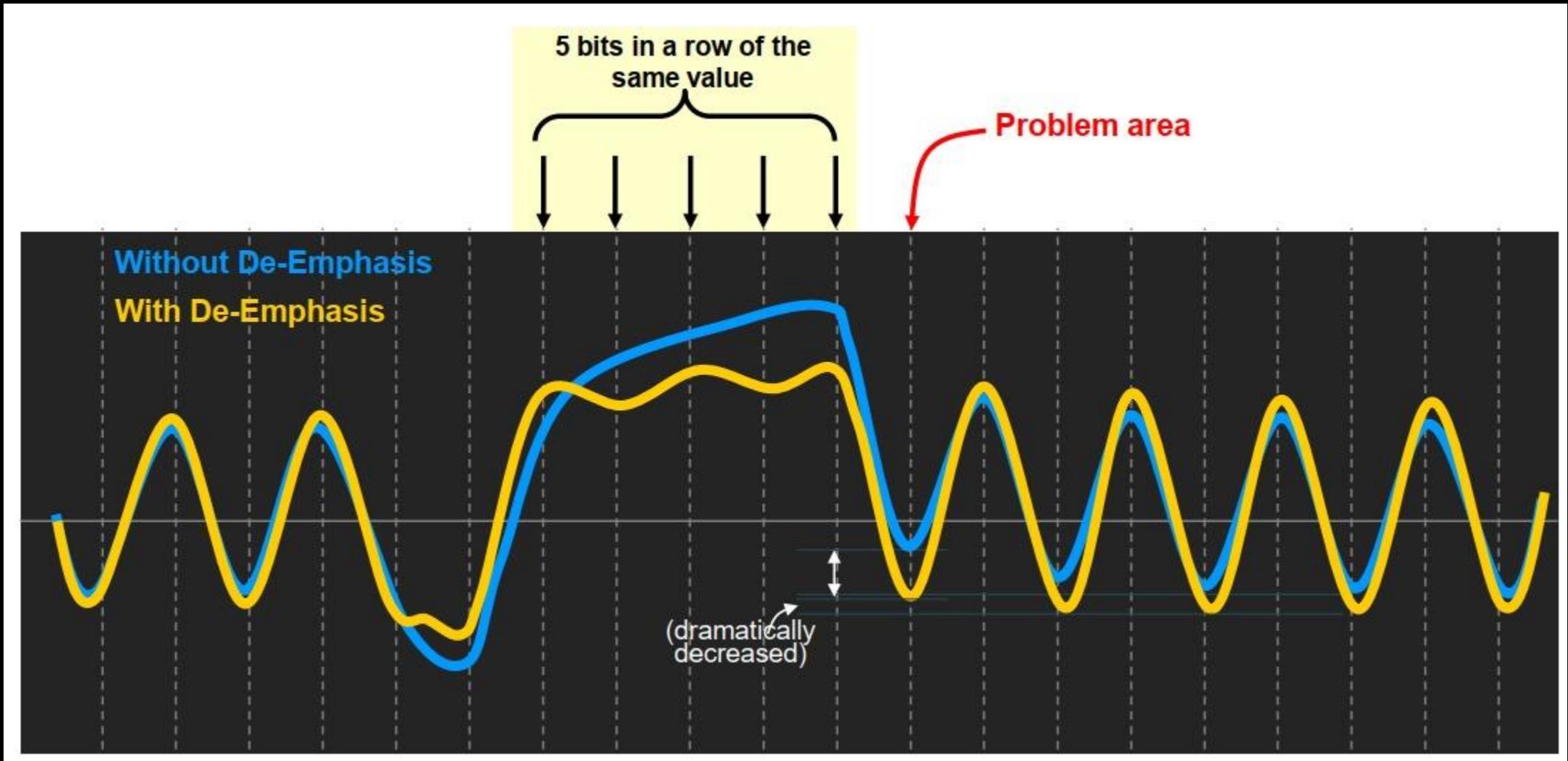
If you know that only first bit after a transition from low to high or high to low has trouble, you can amplify just the first bit after a transition.

A more common approach to achieving a similar waveform is to attenuate the bits after the first transition instead of amplifying the first bit.

High pass filtering accomplishes this because the transitioning bits (higher frequency) are attenuated less than the non transitioning bits.

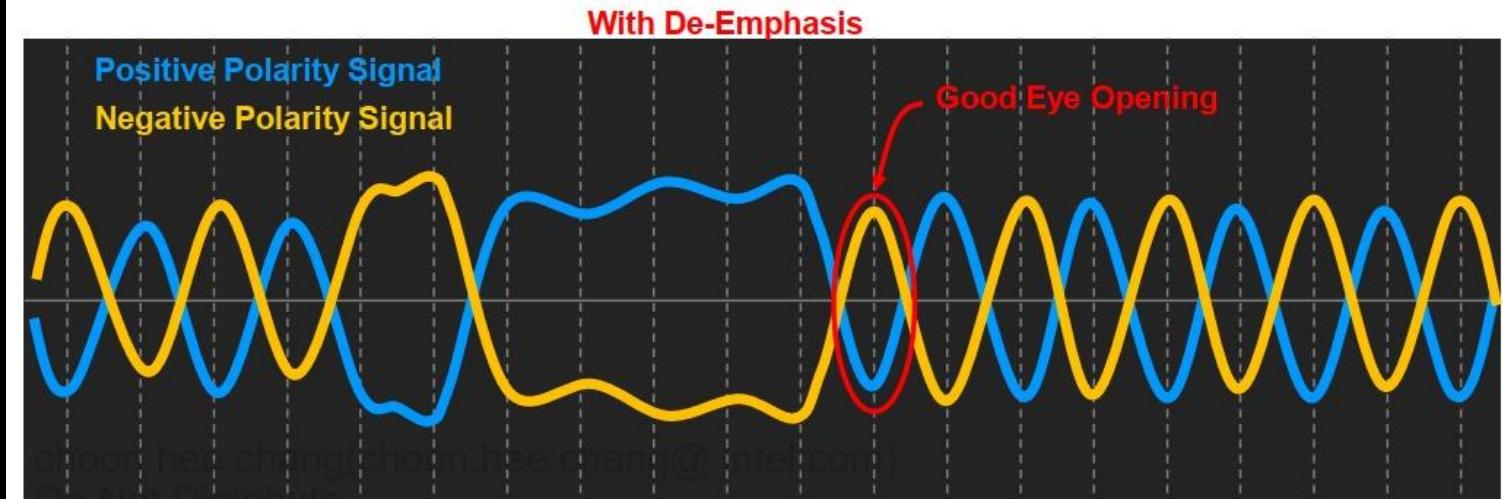
asiclab

De-Emphasis



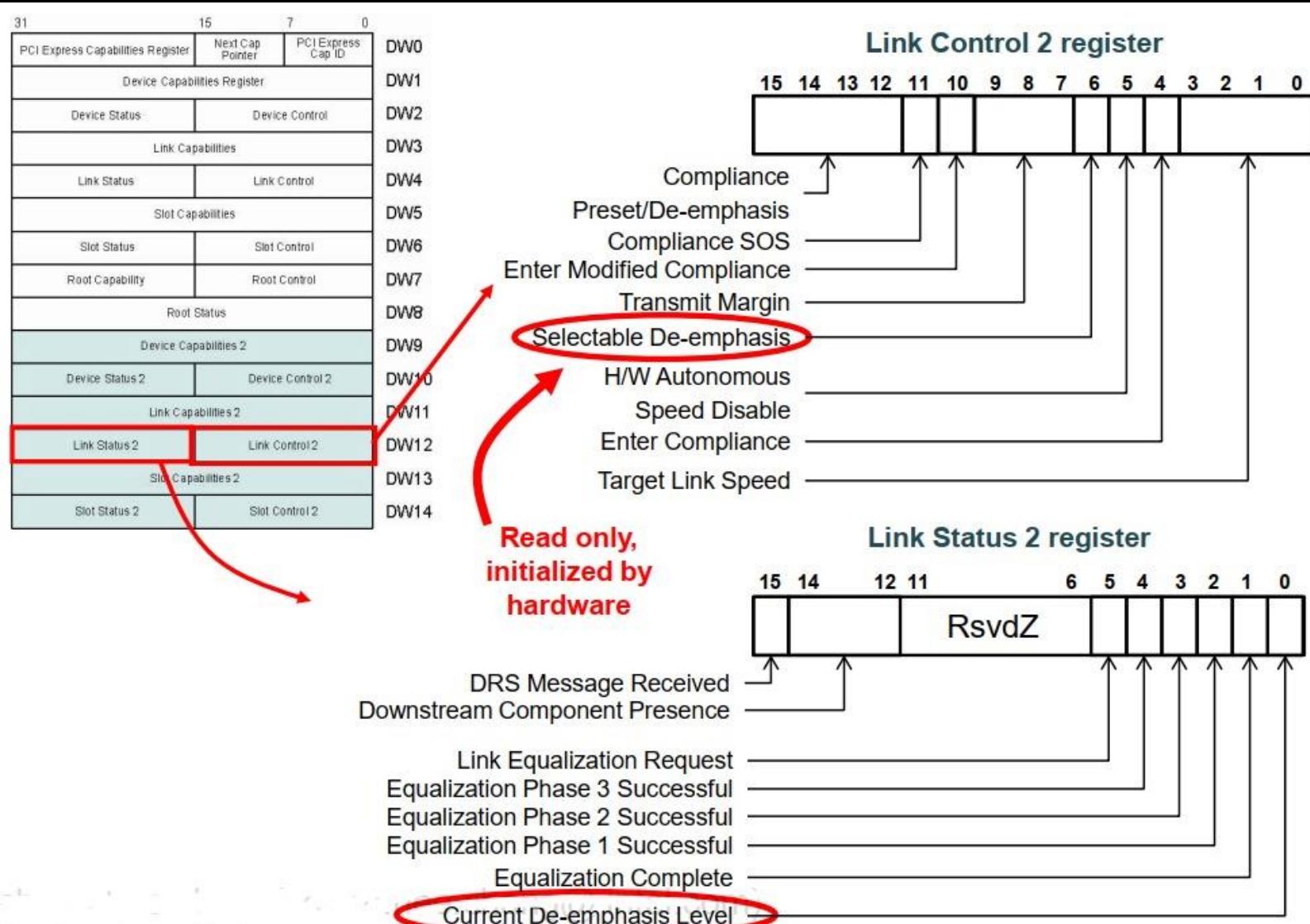
De-Emphasis

asiclab



De-Emphasis

asiclab

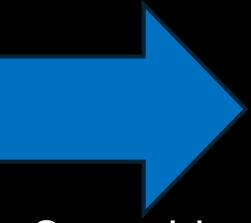


Scrambling

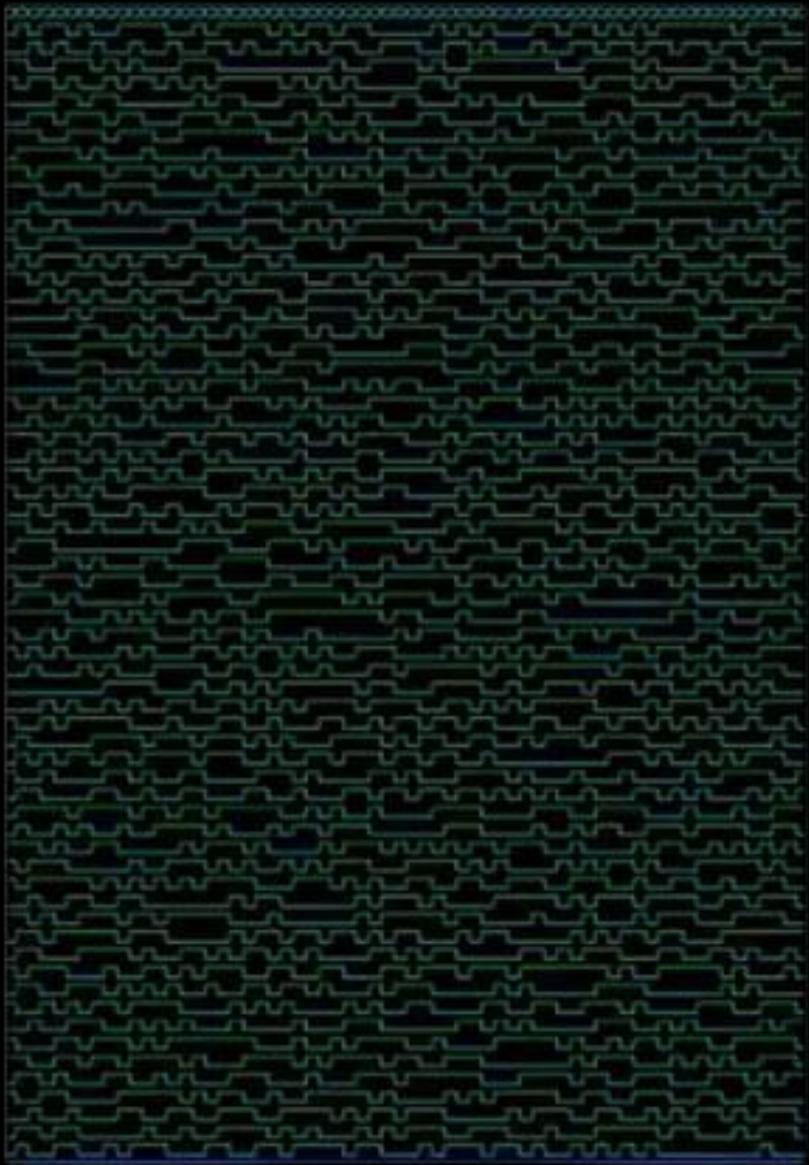
asiclab



Idles



Scrambler



Scrambled Idles

6 Types of Ordered Sets

- Training Sequence One (TS1)
 - 16-character set: 1 COM + 15 additional characters
- Training Sequence Two (TS2)
 - 16-character set: 1 COM + 15 additional characters
- Skip Ordered Set (SOS)
 - 4-character set: 1 COM + 3 SKPs
- Electrical Idle Ordered Set (EIOS)
 - 4 characters at 2.5 GT/s: 1 COM + 3 IDLs
- Fast Training Sequence (FTS)
 - 4-character set: 1 COM + 3 FTSs
- Electrical Idle Exit Ordered Set (EIEOS) [only for data rates above 2.5 GT/s]
 - 16-character set: 1 COM, 14 K28.7, TS1 Identifier

asiclab

Ordered Sets – TS1, TS2

Symbol	TS1 Ordered set											
	7	6	5	4	3	2	1	0				
0	Start: COM (1Eh)											
1	Link Number											
2	Lane Number											
3	N_FTS											
4	Speed change	Auto Change deemphasis	32.0GT/s	16.0 GT/s	8.0 GT/s	5.0GT/s	2.5GT/s	Reserved				
5	Reserved		Compliance Receive	Disable Scrambling	Loopback	Disable Link	Hot Reset					
TS1 ID (D10.2)												
6	1 (EQ)	Transmitter Preset			Receiver Preset Hint							
	User preset	Transmitter Preset			Receiver Preset Hint							
TS1 ID (D10.2)												
7	Reserved	FS										
	Reserved	Pre-cursor coefficient										
TS1 ID (D10.2)												
8	Reserved	LF										
	Reserved	Cursor coefficient										
TS1 ID (D10.2)												
9	Parity	Reject Coefficients	Post-cursor coefficient									
10	TS1 ID (8'h4A)											
11	TS1 ID (8'h4A)											
12	TS1 ID (8'h4A)											
13	TS1 ID (8'h4A)											
14	TS1 ID (8'h4A) or DC balance symbol											
15	TS1 ID (8'h4A) or DC balance symbol											

Symbol	TS2 Ordered set																				
	7	6	5	4	3	2	1	0													
0	Start: COM K28.3 (2Dh)																				
1	Link Number																				
2	Lane Number																				
3	N_FTS																				
4	Speed change	Auto Change deemphasis	32.0GT/s	16.0 GT/s	8.0 GT/s	5.0GT/s	2.5GT/s	Reserved													
5	Reserved		Compliance Receive	Disable Scrambling	Loopback	Disable Link	Hot Reset														
TS2 ID (D5.2)																					
6	1 (EQ)	Transmitter Preset			Receiver Preset Hint																
	Request EQ	Quiesce Guarantee	Reserved																		
7	TS2 ID (8'h45)																				
8	TS2 ID (8'h45)																				
9	TS2 ID (8'h45)																				
10	TS2 ID (8'h45)																				
11	TS2 ID (8'h45)																				
12	TS2 ID (8'h45)																				
13	TS2 ID (8'h45)																				
14	TS2 ID (8'h45) or DC Balance symbol																				
15	TS2 ID (8'h45) or DC Balance symbol																				

Ordered Sets

EIOS	Gen1/Gen2	Gen 3-5	Gen 6
Symbol 0	1Eh	66h	0xF0
Symbol 1	IDL	66h	0x0F
Symbol 2	IDL	66h	0xF0
Symbol 3	IDL	66h	0x0F
Symbol 4		66h	0xF0
Symbol 5		66h	0x0F
Symbol 6		66h	0xF0
Symbol 7		66h	0x0F
Symbol 8		66h	0xF0
Symbol 9		66h	0x0F
Symbol 10		66h	0xF0
Symbol 11		66h	0x0F
Symbol 12		66h	0xF0
Symbol 13		66h	0x0F
Symbol 14		66h	0xF0
Symbol 15		66h	0x0F

EIEOS	Gen 3-5	Gen 6
Symbol 0	COM	00h
Symbol 1	FFh	00h
Symbol 2	00h	FFh
Symbol 3	FFh	FFh
Symbol 4	00h	00h
Symbol 5	FFh	00h
Symbol 6	00h	FFh
Symbol 7	FFh	FFh
Symbol 8	00h	00h
Symbol 9	FFh	00h
Symbol 10	00h	FFh
Symbol 11	FFh	FFh
Symbol 12	00h	00h
Symbol 13	FFh	00h
Symbol 14	00h	FFh
Symbol 15	FFh	FFh

FTS	Gen 3-5	Gen 6
Symbol 0	55h	55h
Symbol 1	47h	47h
Symbol 2	4Eh	4Eh
Symbol 3	C7h	C7h
Symbol 4	CCh	CCh
Symbol 5	C6h	C6h
Symbol 6	C9h	C9h
Symbol 7	25h	25h
Symbol 8	6Eh	6Eh
Symbol 9	ECh	ECh
Symbol 10	88h	88h
Symbol 11	7Fh	7Fh
Symbol 12	80h	80h
Symbol 13	8Dh	8Dh
Symbol 14	8Bh	8Bh
Symbol 15	8Eh	8Eh

SKP	Gen 3-5	Gen 6
Symbol 0	AAh	AAh
Symbol 1	AAh	AAh
Symbol 2	AAh	AAh
Symbol 3	AAh	AAh
Symbol 4	AAh	AAh
Symbol 5	AAh	AAh
Symbol 6	AAh	AAh
Symbol 7	AAh	AAh
Symbol 8	AAh	AAh
Symbol 9	AAh	AAh
Symbol 10	AAh	AAh
Symbol 11	AAh	AAh
Symbol 12	E1h (SKP_END)	E1h (SKP_END)
Symbol 13	DP + LFSR[22:16]	DP + LFSR[22:16]
Symbol 14	LFSR[15:8]	LFSR[15:8]
Symbol 15	LFSR[7:0]	LFSR[7:0]

Ordered Sets

asiclab

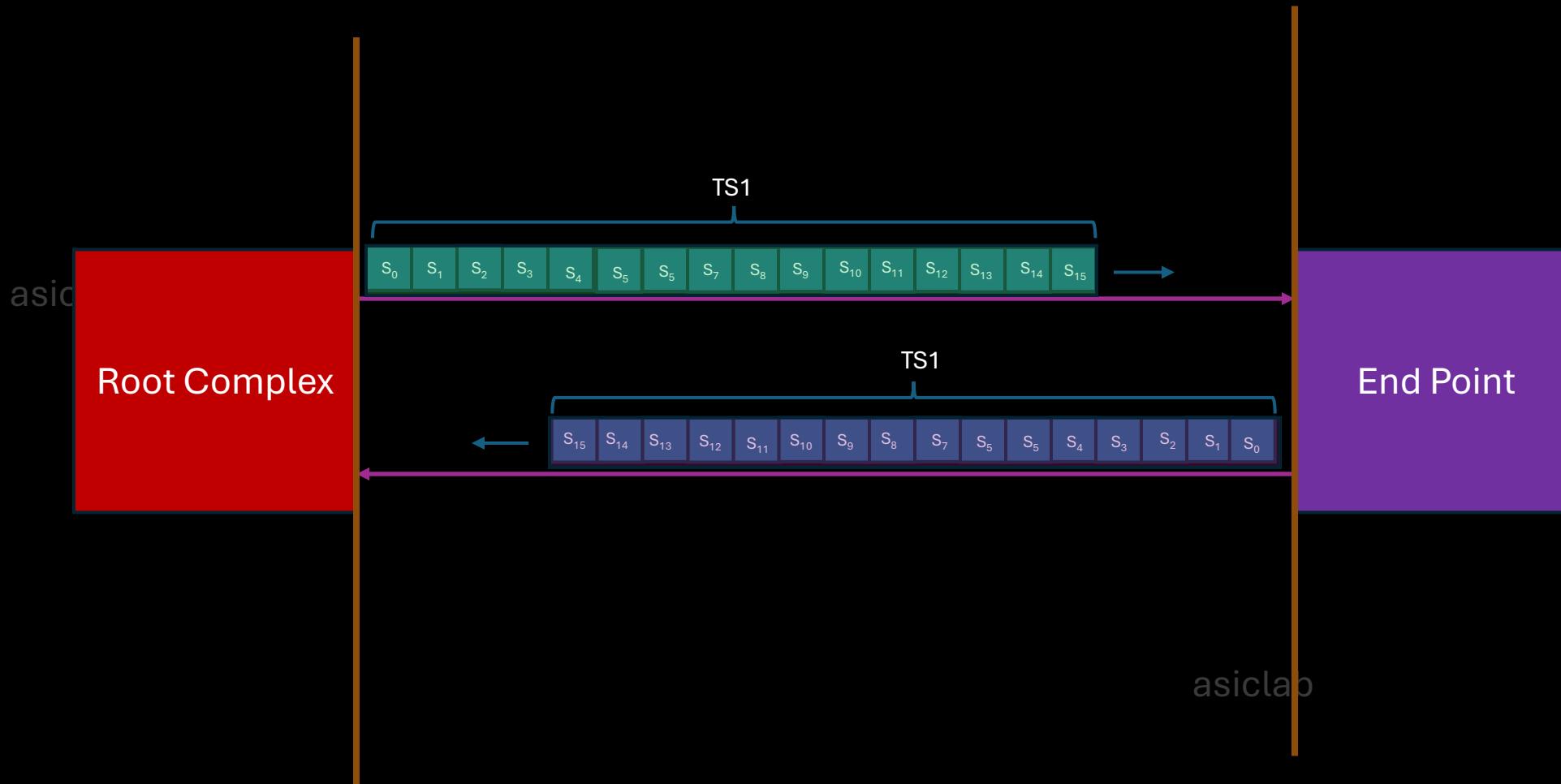
CTL SKP	Gen 3-5	Gen 6
Symbol 0	AAh	1F0h
Symbol 1	AAh	0Fh
Symbol 2	AAh	F0h
Symbol 3	AAh	0Fh
Symbol 4	AAh	F0h
Symbol 5	AAh	0Fh
Symbol 6	AAh	F0h
Symbol 7	AAh	0Fh
Symbol 8	AAh	F0h
Symbol 9	AAh	0Fh
Symbol 10	AAh	F0h
Symbol 11	AAh	0Fh
Symbol 12	78h (SKP_END_CTL)	78h (SKP_END_CTL)
Symbol 13	DP + LFSR[22:16]	DP + LFSR[22:16]
Symbol 14	LFSR[15:8]	LFSR[15:8]
Symbol 15	LFSR[7:0]	LFSR[7:0]

SDS	Gen 3-5	Gen 6
Symbol 0	E1h	B1h
Symbol 1	55h	C6h
Symbol 2	55h	C6h
Symbol 3	55h	C6h
Symbol 4	55h	B1h
Symbol 5	55h	C6h
Symbol 6	55h	C6h
Symbol 7	55h	C6h
Symbol 8	55h	B1h
Symbol 9	55h	C6h
Symbol 10	55h	C6h
Symbol 11	55h	C6h
Symbol 12	55h	B1h
Symbol 13	55h	C6h
Symbol 14	55h	C6h
Symbol 15	55h	C6h

Control Character Encodings

Character	8b Name	10b (CRD-)	10b (CRD+)	Description
COM	K28.5 (BCh)	001111 1010	110000 0101	Comma used as a character boundary alignment symbol
PAD	K23.7 (F7h)	111010 1000	000101 0111	Packet Padding Symbol
SKP	K28.0 (1Ch)	001111 0100	110000 1011	Used in SKP Ordered Set (SOS)
STP	K27.7 (FBh)	110110 1000	001001 0111	Start of TLP Symbol
SDP	K28.2 (5Ch)	001111 0101	110000 1010	Start of DLLP Symbol
END	K29.7 (FDh)	101110 1000	010001 0111	End of Good Packet Symbol
EDB	K30.7 (FEh)	011110 1000	100001 0111	End of Bad Packet Symbol, used by Switch which detects bad packet
FTS	K28.1 (3Ch)	001111 1001	110000 0110	Used in Ordered Set to exit L0s to L0 power state
IDL	K28.3 (7Ch)	001111 0111	110000 1100	Used in Electrical Idle Ordered Set
EIE	K28.7 (FCh)	001111 1000	110000 0111	Used in the Electrical Idle Exit Ordered Set (EIEOS) and sent prior to FTS at speeds other than 2.5 GT/s

Ordered Set Exchange



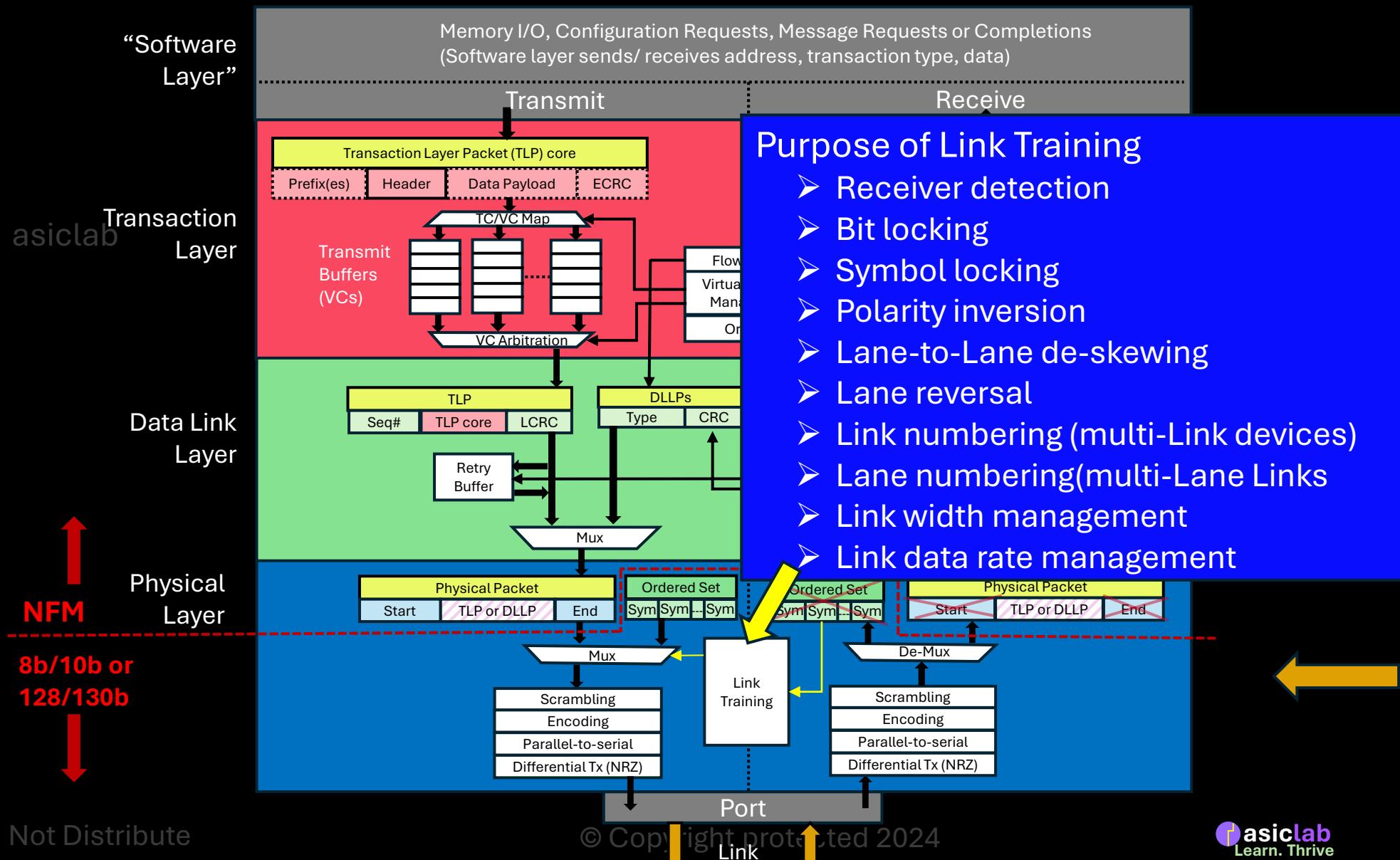
$S_0 - S_{15}$ – Symbol 0 to Symbol 15

asiclab

Link Training

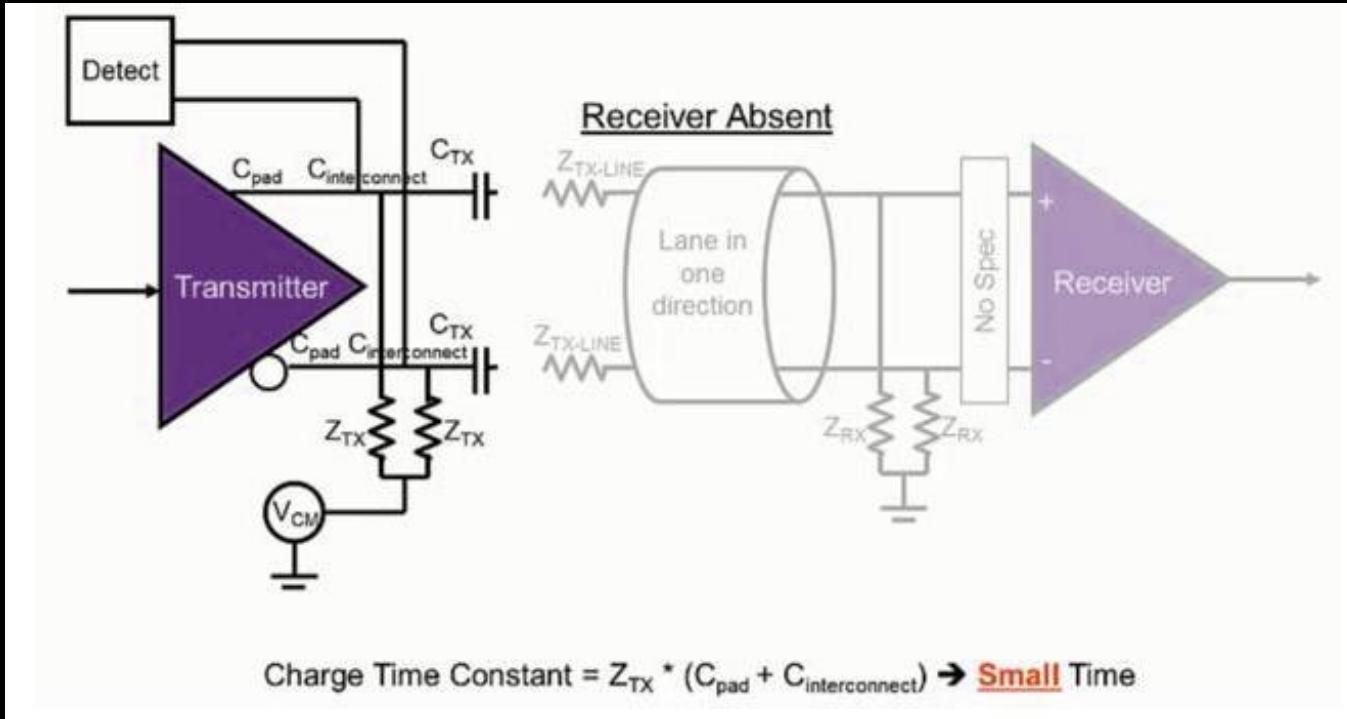
asiclab

Link Initialization and Training



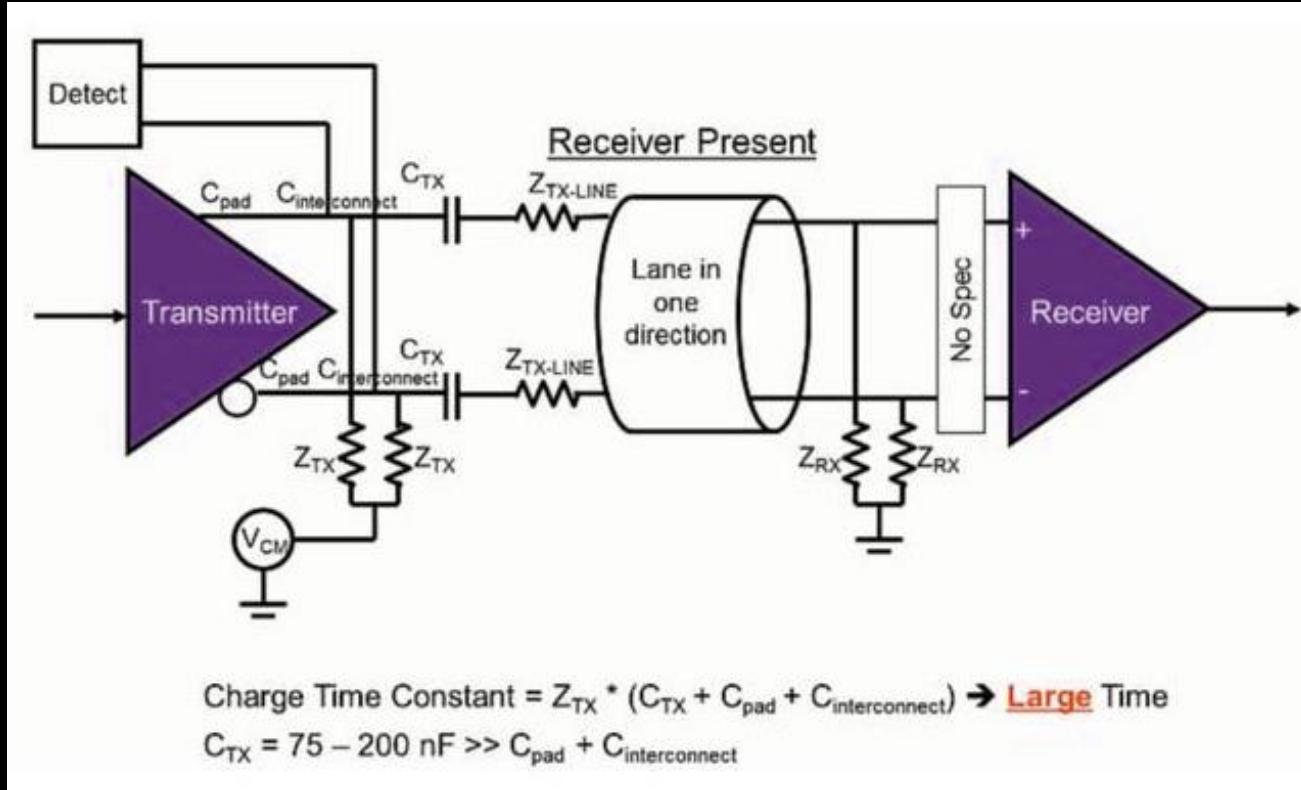
Receiver not connected

asiclab

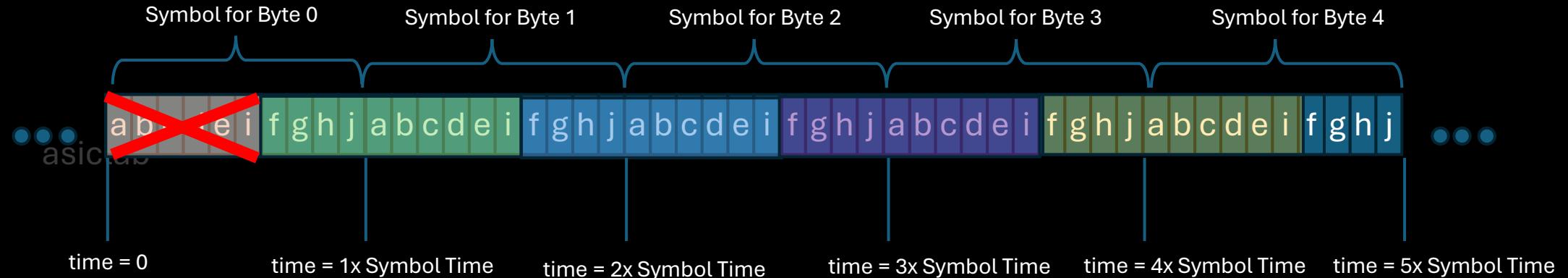


Receiver connected

asiclab



Achieving Bit lock



Assuming above data traffic is transmitted in 8b/10b on PHY RX PATH

Considering Symbol 0 {a,b,c,d,e,l} gets consumed by PHY DUT for achieving CDR lock

Serdes byte 0 will be Symbol Byte 0 {f,g,h,j} and Symbol Byte 1 {a,b,c,d,e,i}

Serdes byte 1 will be Symbol Byte 1 {f,g,h,j} and Symbol Byte 2 {a,b,c,d,e,i}

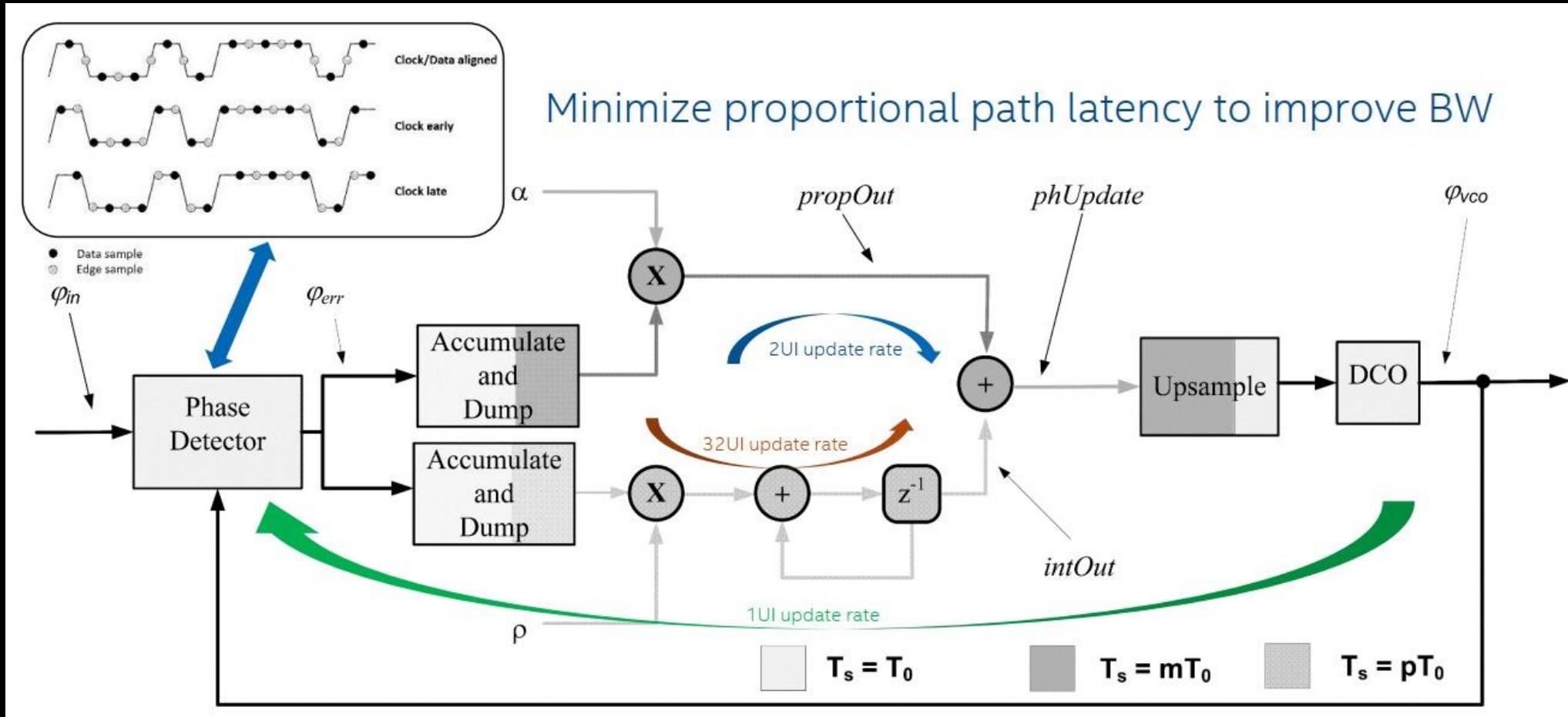
Bits follow in similar fashion for Serdes Byte 2, Serdes Byte 3 and so on ...

Achieving Bit lock

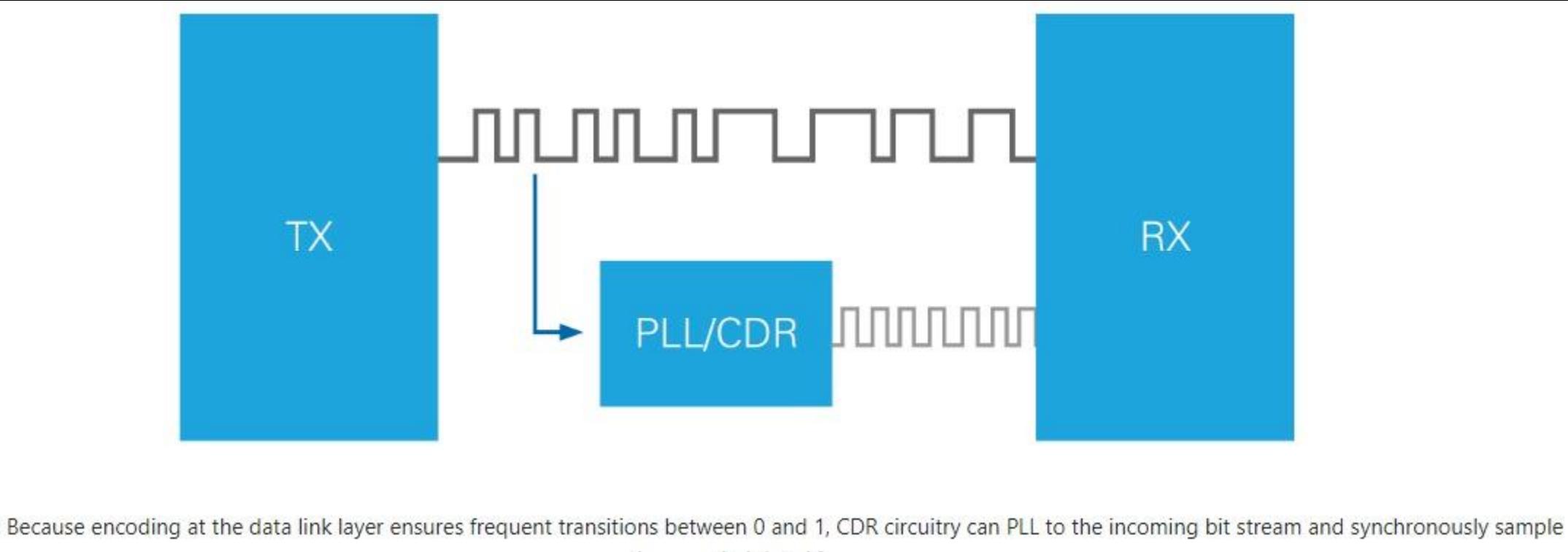
- Using PLL or other method, receiver generates a RX (Recovered) Clock and aligns the phase using input data Edges as a reference. It can legally vary from the ideal frequency by +/- 300 ppm.
- Resulting Rx Clock has same frequency and phase as the Tx clock used to send the data.
- Rx Clock latches incoming bits into deserializer and divided by symbol size, into the elastic buffer
- “Local Clock” latches data out of the elastic buffer and may be different from the ideal clock by +/- 300 ppm.

asiclab

Clock and Data Recovery (CDR)

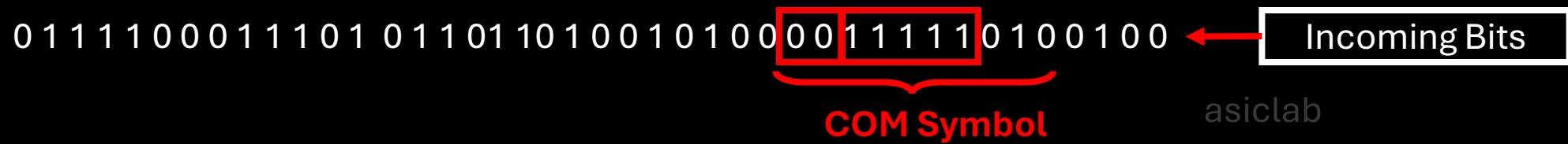


Clock and Data Recovery (CDR)

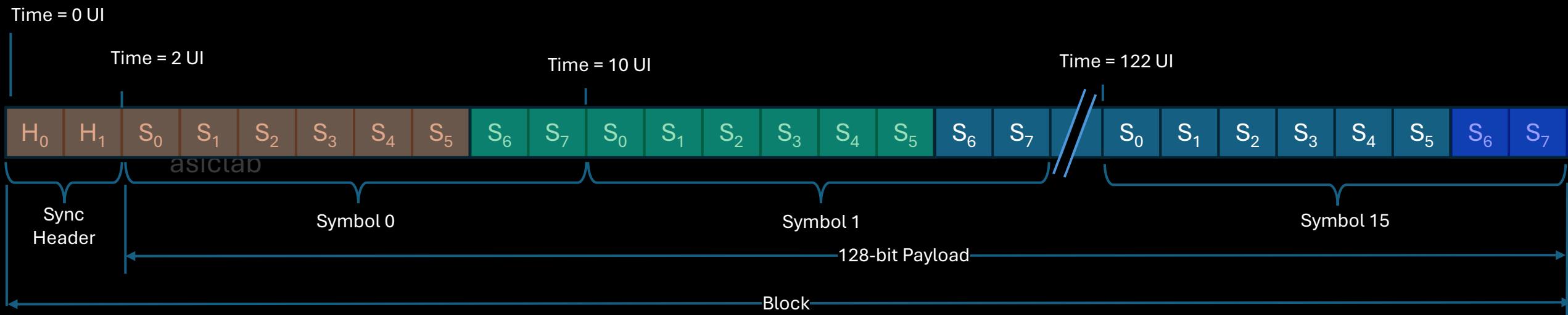


Achieving Symbol Lock

- Once PLL is locked, receiver can reliably sample incoming bits.
- To recognize what's being sent on the Link, though, the next step is to find the 10-bit symbol boundary (acquiring Symbol Lock).
- This is done by searching for the pattern of 2 bits of one polarity followed by 5 bits of the other polarity. During training this will be the COM character and finding it gives the symbol position in the bit stream.



Achieving Block Lock



Assuming above data traffic transmitted in 128/130b on PHY TX path

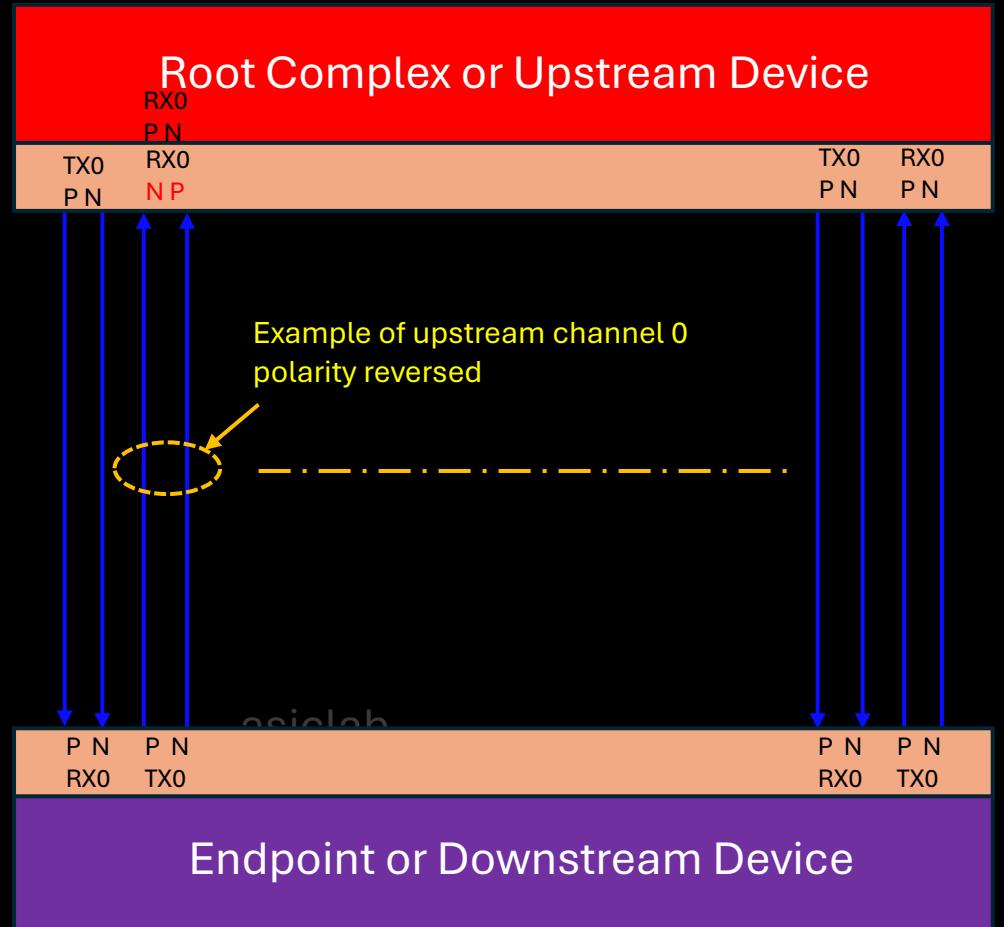
Serdes byte 0 will be Sync Header H₀, H₁, Symbol0 S₀-S₅ (with higher 2 bits reserved)

Serdes byte 1 will be Symbol0 S₆-S₇, Symbol1 S₀-S₅ (with higher two bits reserved)

At the end of 130b block, two bits will be left out to be send in upcoming Serdes byte

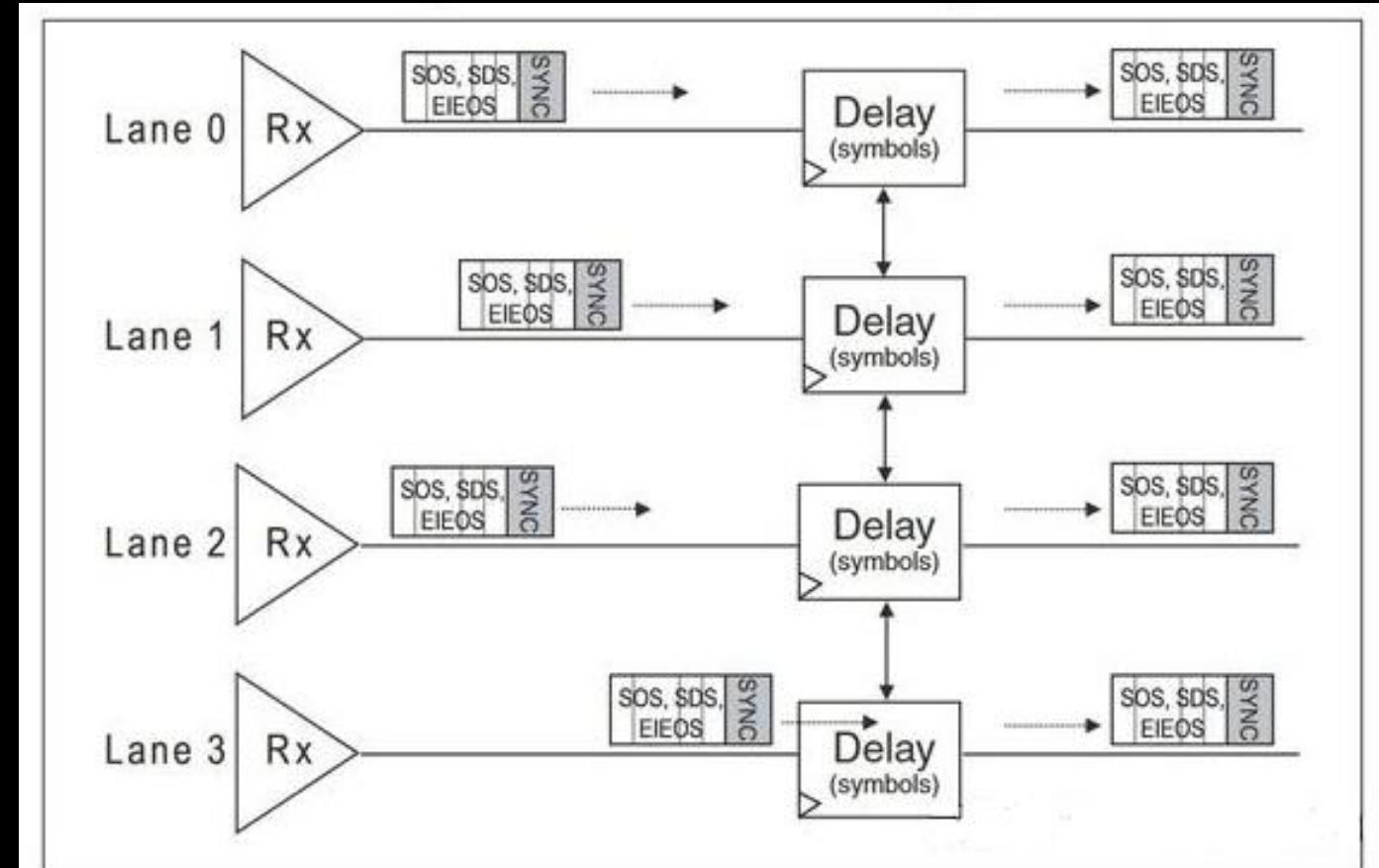
Polarity Inversion

1. Any channel can be connected with the positive and negative signals reversed.
2. Training sequence detects polarity inversion condition
asiclab
3. RX port is required to invert its signal polarity
4. Eases routing if differential signals are normally bow-tied
5. Reduced vias



Receiver Link De-Skew

Worst-case Rx skew to compensate:
2.5GT/s: 20ns = 5 symbol time delay
5.0GT/s: 8ns = 4 symbol time delay
8.0GT/s: 6ns = 6 symbol time delay
16.0GT/s: 5ns = 10 symbol time delay

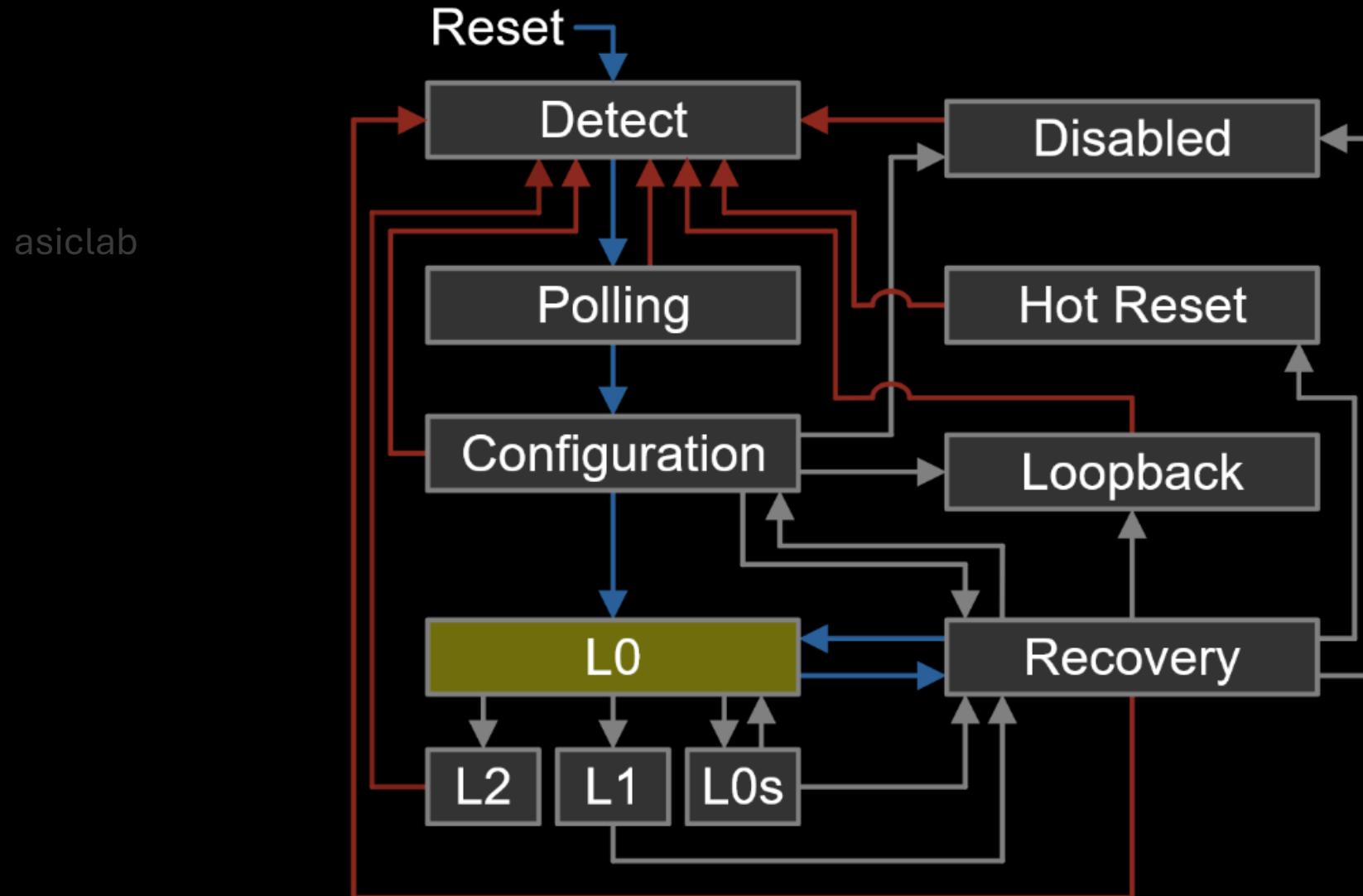


asiclab

LTSSM (Link Training Status State Machine)

asiclab

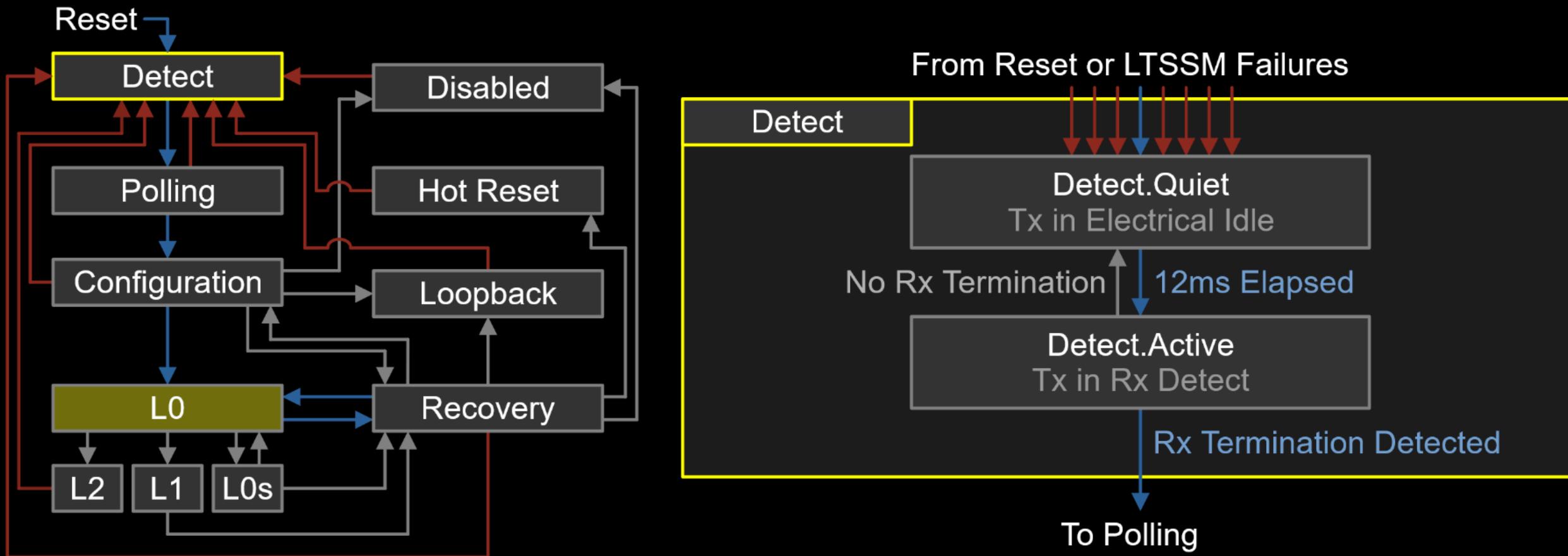
Link Training Status State Machine (LTSSM)



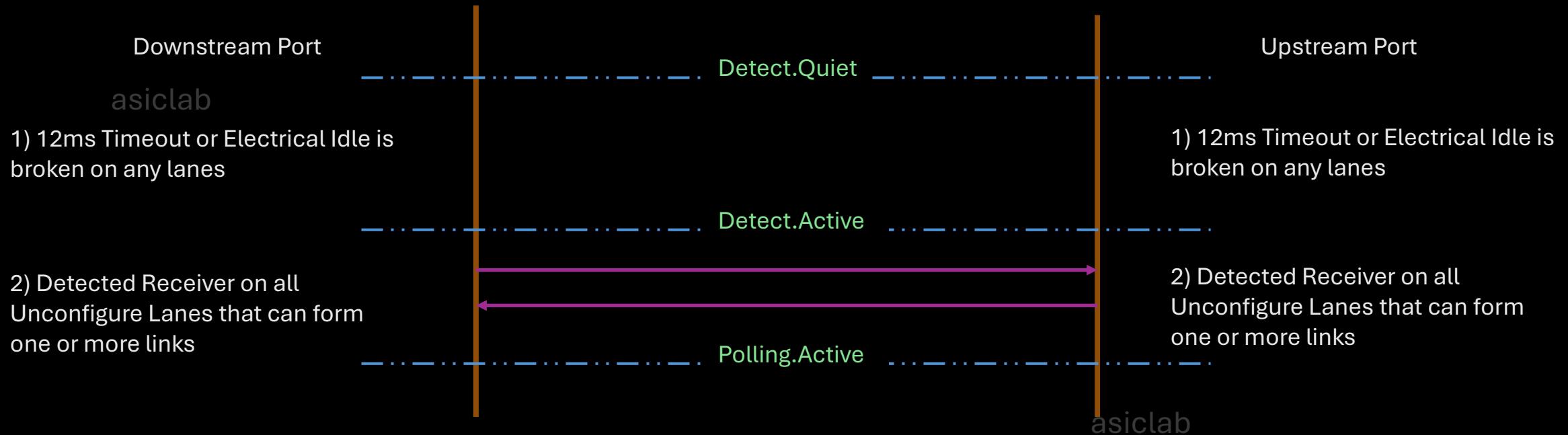
Link Training Status State Machine (LTSSM)

- PCI express is a serial link. PCI express port required go through a series of link training before can be used to transmit / receive data.
- Reasons of link training is used to train up bit lock and Symbol lock of the PCI express port.
- Besides that, link training also use to exchange information between 2 points and negotiate to a configuration that can be supported by both points.
- Link training also use to identify the healthiness of the link.
- Link training also use to recover the link when link is not stable.
- Link training is using training sequence (TS1 and TS2) to train up the PCI express ports.

LTSSM – Detect State



LTSSM – Detect State



Gen 3

1. EQ 8.0 GT/s Phase 1 Successful reset to “0”
2. EQ 8.0 GT/s Phase 2 Successful reset to “0”
3. EQ 8.0 GT/s Phase 3 Successful reset to “0”
4. EQ 8.0 GT/s Complete reset to “0”
5. EQ_done_8GT_data_rate reset to “0”

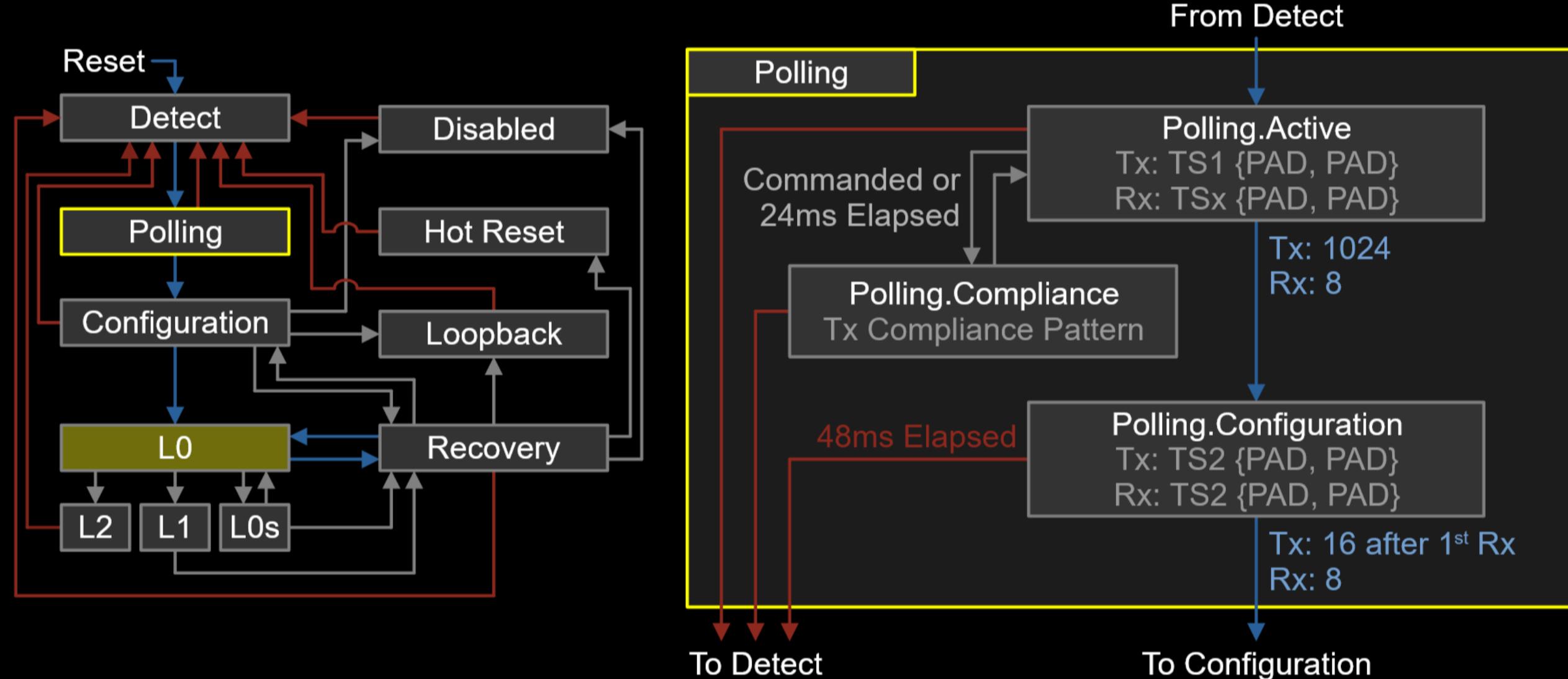
asiclab

Gen 4

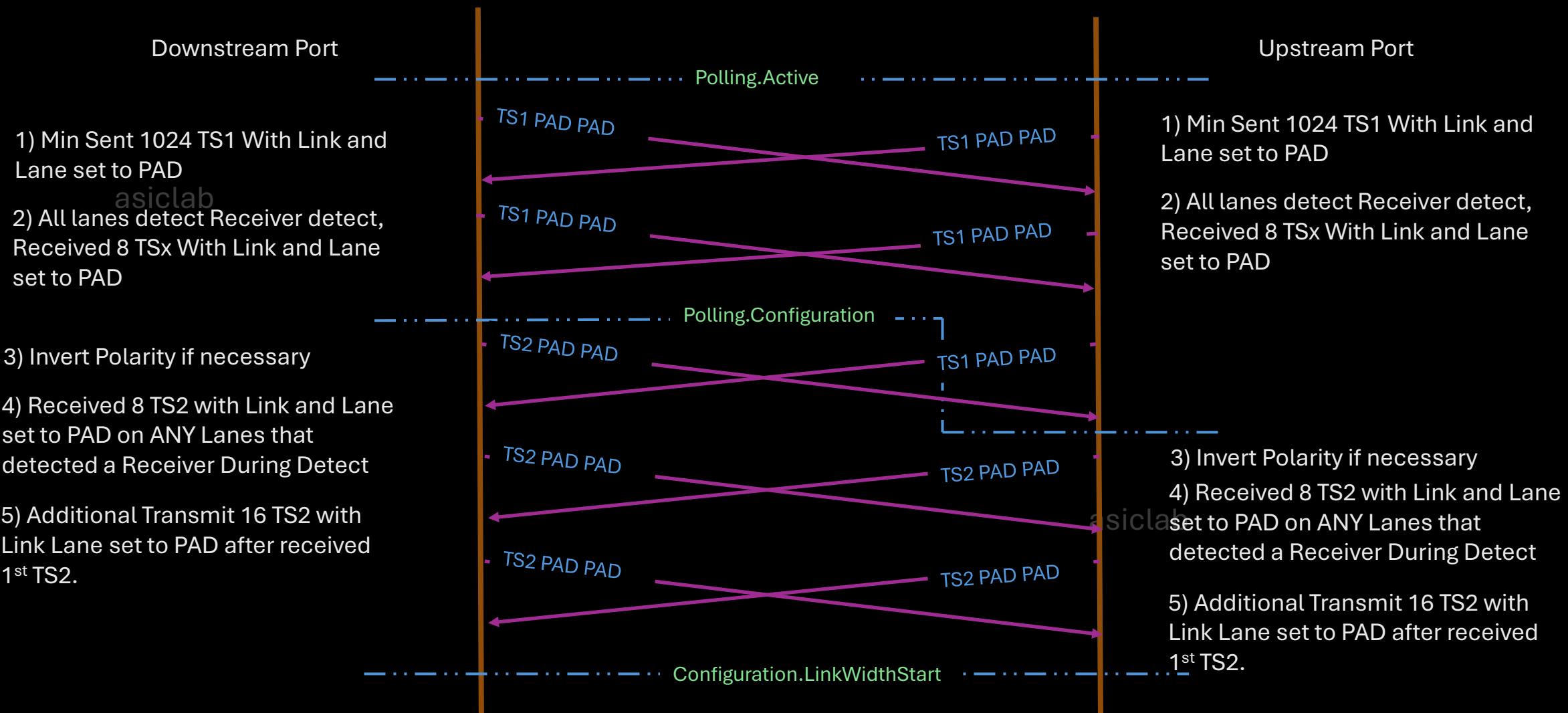
1. EQ 16.0 GT/s Phase 1 Successful reset to “0”
2. EQ 16.0 GT/s Phase 2 Successful reset to “0”
3. EQ 16.0 GT/s Phase 3 Successful reset to “0”
4. EQ 16.0 GT/s Complete reset to “0”
5. EQ_done_16GT_data_rate reset to “0”

asiclab

LTSSM – Polling State



LTSSM – Polling State



LTSSM – Polling State

1. During Polling State, Training Sequence Will Be Transmitted.
2. Phy Will Make Use Of Training Sequence Bits Toggling To Achieve Bits Lock.
Minimum of 1024 TS sequence must be transmitted to allow bit lock achievement.
3. Reasons Of “4A” Is Chosen As TS1 Identifier And “45” Is Chosen As TS2 Identifier Is Because “4A” and “45” Have Most Toggling And Is Good To Help Achieve Bits Lock.
4. After Bit Lock Achieve, K-align Logic Is Going To Make Use Of COM Of The TS Sequence To Achieve K-align Lock.
5. As Soon As K-align Lock Achieve, Init Layer Is Going To Observe The Identifier Of The TS Sequence And If Detected TS1 or TS2 Identifier In Reverse Polarity (Instead Of “4A”, Init Receive “B5” Or Instead Of “45”, Init Receive “BA”), Polarity Inversion Of Incoming Data Will Be Perform.

LTSSM – Polling State

6. Polarity Inversion will be discovered through inverted 8B10B “0x4A” (Inverted Become “0xB5”)
7. Polling Compliance State Use For Electrical Characterization Usage.
8. After complete TS1 Handshaking, LTSSM transition to Polling Configurations.
9. During Polling Configuration, both ports LTSSM handshake again with TS2, serve as acknowledgement of receiving TS1 in Polling Active State. Whichever Ports achieved bit lock, symbol lock 1st and had detected 16 TS1 / TS2, will transition to this states to wait for remote ports achieve bit lock and symbol lock

asiclab

LTSSM – Polling Compliance

1. 3 mode of Entry
 - a) Enter Compliance bit (EC)
 - i. Transmitter setting -Compliance Preset/Deemphasis register
 - b) Enter due to Received TS1 with Compliance Receive Bit set.
 - asiclab i. Gen 3/4 - TS1 could be EQTS1 – use as transmitter setting.
 - c) Compliance Load Board (CLB) – Squelch exit not detected onany lane which detected receiver
 - i. Transmitter setting is based iteration – table.
2. Two type of compliance pattern
 1. Compliance pattern
 2. Modified Compliance Pattern
3. Mode a) without Enter Modified Compliance (EMC) bit set transmitter will send Compliance pattern, with EMC bit set transmitter will send modified compliance pattern. Exit if EC bit get cleared.
4. Mode b) will send modified compliance pattern. No exit path.
5. Mode c) will send compliance pattern. Exit after detect Squelch exit

TS1 at Polling.Active

asiclab

TS2 at Polling.Configuration

asiclab

***** TS2 *****	COM	C							
Link No: PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	P
Lane Ordering: PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	P
N_FTS: 34 Dec	22	22	22	22	22	22	22	22	2
Data Rate ID: 02 Hex	02	02	02	02	02	02	02	02	0
Gen 1 rate supported	-	-	-	-	-	-	-	-	-
Training Control: 00 Hex	00	00	00	00	00	00	00	00	0
Hot Reset: De-assert	-	-	-	-	-	-	-	-	-
Disable Link: De-assert	-	-	-	-	-	-	-	-	-
Loopback: De-assert	-	-	-	-	-	-	-	-	-
Disable Scrambling: De-assert	-	-	-	-	-	-	-	-	-
Compliance Receive: De-assert	-	-	-	-	-	-	-	-	-
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
***** TS2 *****	COM	C							
Link No: PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	P
Lane Ordering: PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	PAD	P
N_FTS: 34 Dec	22	22	22	22	22	22	22	22	2
Data Rate ID: 02 Hex	02	02	02	02	02	02	02	02	0
Gen 1 rate supported	-	-	-	-	-	-	-	-	-
Training Control: 00 Hex	00	00	00	00	00	00	00	00	0
Hot Reset: De-assert	-	-	-	-	-	-	-	-	-
Disable Link: De-assert	-	-	-	-	-	-	-	-	-
Loopback: De-assert	-	-	-	-	-	-	-	-	-
Disable Scrambling: De-assert	-	-	-	-	-	-	-	-	-
Compliance Receive: De-assert	-	-	-	-	-	-	-	-	-
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4
TS2 Identifier	45	45	45	45	45	45	45	45	4

Gen 1 / 2 Compliance Pattern

asiclab

Lane 0	D	D	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5	K28.5+	D10.2
Lane 1	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5
Lane 2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 3	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 4	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 5	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 6	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 7	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 8	D	D	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5	K28.5+	D10.2
Lane 9	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5

Key:

K28.5- COM when disparity is negative, specifically: "0011111010"

K28.5+ COM when disparity is positive, specifically: "1100000101"

D21.5 Out of phase data Symbol, specifically: "1010101010"

D10.2 Out of phase data Symbol, specifically: "0101010101"

D Delay Symbol K28.5 (with appropriate disparity)

Gen 1 / 2 Modified Compliance Pattern

asiclab

An illustration of the Modified Compliance Pattern is shown below:																	
Lane0	D	D	D	D	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.7-	K28.7-	K28.7-	K28.5-	D21.5
Lane1	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D
Lane2	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D21.5
Lane3	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D21.5
Lane4	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D21.5
Lane5	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D21.5
Lane6	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D21.5
Lane7	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D21.5
Lane8	D	D	D	D	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.7-	K28.7-	K28.7-	K28.5-	D21.5
Lane9	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D

Key:

- K28.5- COM when disparity is negative, specifically: "0011111010"
- K28.5+ COM when disparity is positive, specifically: "1100000101"
- D21.5 Out of phase data Symbol specifically: "1010101010"
- D10.2 Out of phase data Symbol, specifically: "0101010101"
- D Delay Symbol K28.5 (with appropriate disparity)
- ERR error status Symbol (with appropriate disparity)
- K28.7- EIE when disparity is negative, specifically "0011111000"

Gen 3/4 Compliance Pattern

The compliance pattern consists of the following repeating sequence of 36 Blocks

1. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of 64 1's followed by 64 0's
2. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of the following:

	Lane No modulo 8 = 0	Lane No modulo 8 = 1	Lane No modulo 8 = 2	Lane No modulo 8 = 3	Lane No modulo 8 = 4	Lane No modulo 8 = 5	Lane No modulo 8 = 6	Lane No modulo 8 = 7
Symbol 0	55h	FFh	FFh	FFh	55h	FFh	FFh	FFh
Symbol 1	55h	FFh	FFh	FFh	55h	FFh	FFh	FFh
Symbol 2	55h	00h	FFh	FFh	55h	FFh	FFh	FFh
Symbol 3	55h	00h	FFh	FFh	55h	FFh	F0h	F0h
Symbol 4	55h	00h	FFh	C0h	55h	FFh	00h	00h
Symbol 5	55h	00h	C0h	00h	55h	E0h	00h	00h
Symbol 6	55h	00h	00h	00h	55h	00h	00h	00h
Symbol 7	{P,~P}							
Symbol 8	00h	1Eh	2Dh	3Ch	48h	5Ah	69h	78h
Symbol 9	00h	55h	00h	00h	00h	55h	00h	F0h
Symbol 10	00h	55h	00h	00h	00h	55h	00h	00h
Symbol 11	00h	55h	00h	00h	00h	55h	00h	00h
Symbol 12	00h	55h	0Fh	0Fh	00h	55h	07h	00h
Symbol 13	00h	55h	FFh	FFh	00h	55h	FFh	00h
Symbol 14	00h	55h	FFh	FFh	7Fh	55h	FFh	00h
Symbol 15	00h	55h	FFh	FFh	FFh	55h	FFh	00h

Key: P: Indicates the 4-bit encoding of the Transmitter preset being used.
~P: Indicates the bit-wise inverse of P.

Gen 3/4 Compliance Pattern

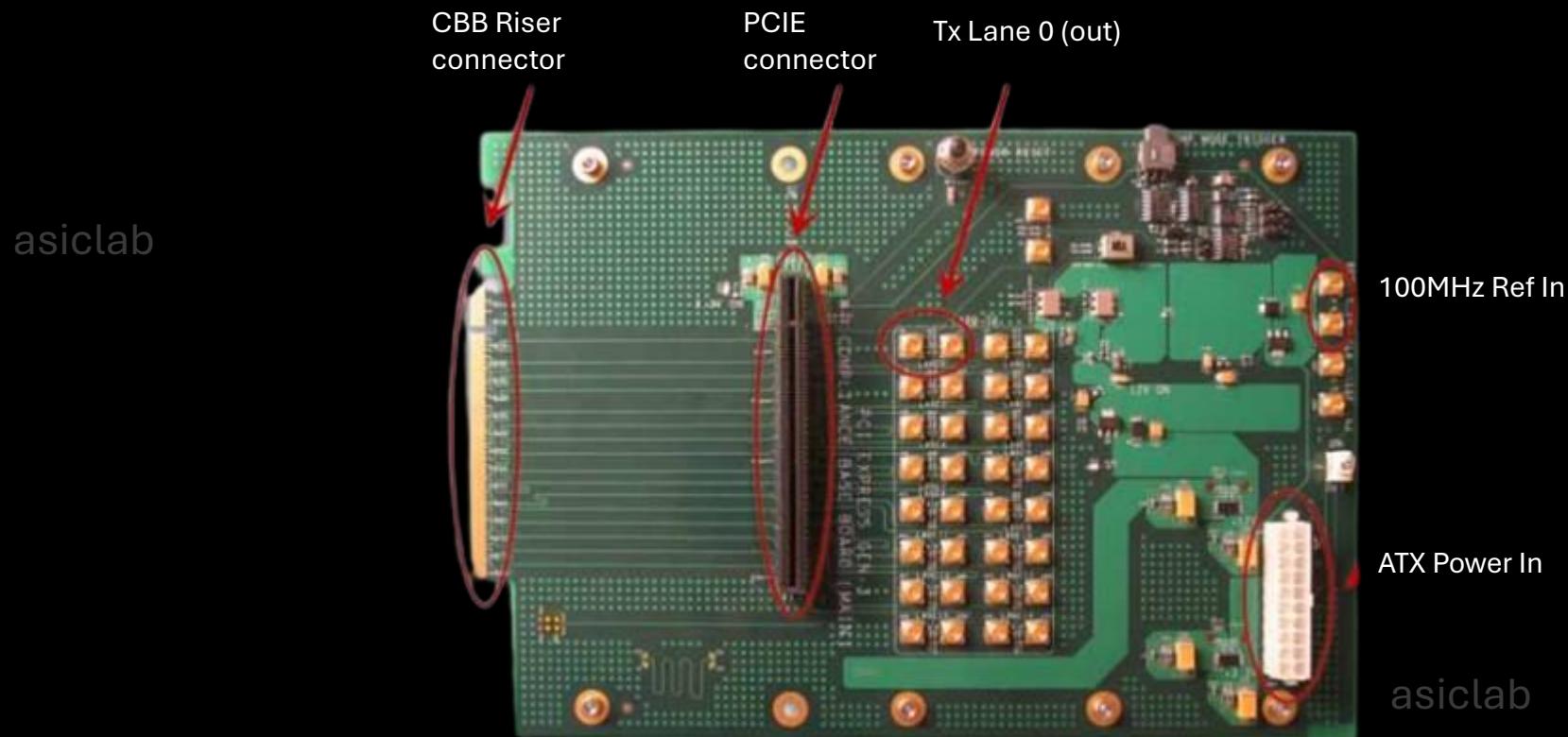
3. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of the following:

	Lane No modulo 8 = 0	Lane No modulo 8 = 1	Lane No modulo 8 = 2	Lane No modulo 8 = 3	Lane No modulo 8 = 4	Lane No modulo 8 = 5	Lane No modulo 8 = 6	Lane No modulo 8 = 7
Symbol 0	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 1	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 2	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 3	F0h	F0h	55h	F0h	F0h	F0h	55h	F0h
Symbol 4	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 5	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 6	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 7	{P,~P}							
Symbol 8	00h	1Eh	2Dh	3Ch	4Bh	5Ah	69h	78h
Symbol 9	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 10	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 11	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 12	FFh	0Fh	0Fh	55h	0Fh	0Fh	0Fh	55h
Symbol 13	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h
Symbol 14	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h
Symbol 15	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h

Key: P: Indicates the 4-bit encoding of the Transmitter preset being used.
~P: Indicates the bit-wise inverse of P.

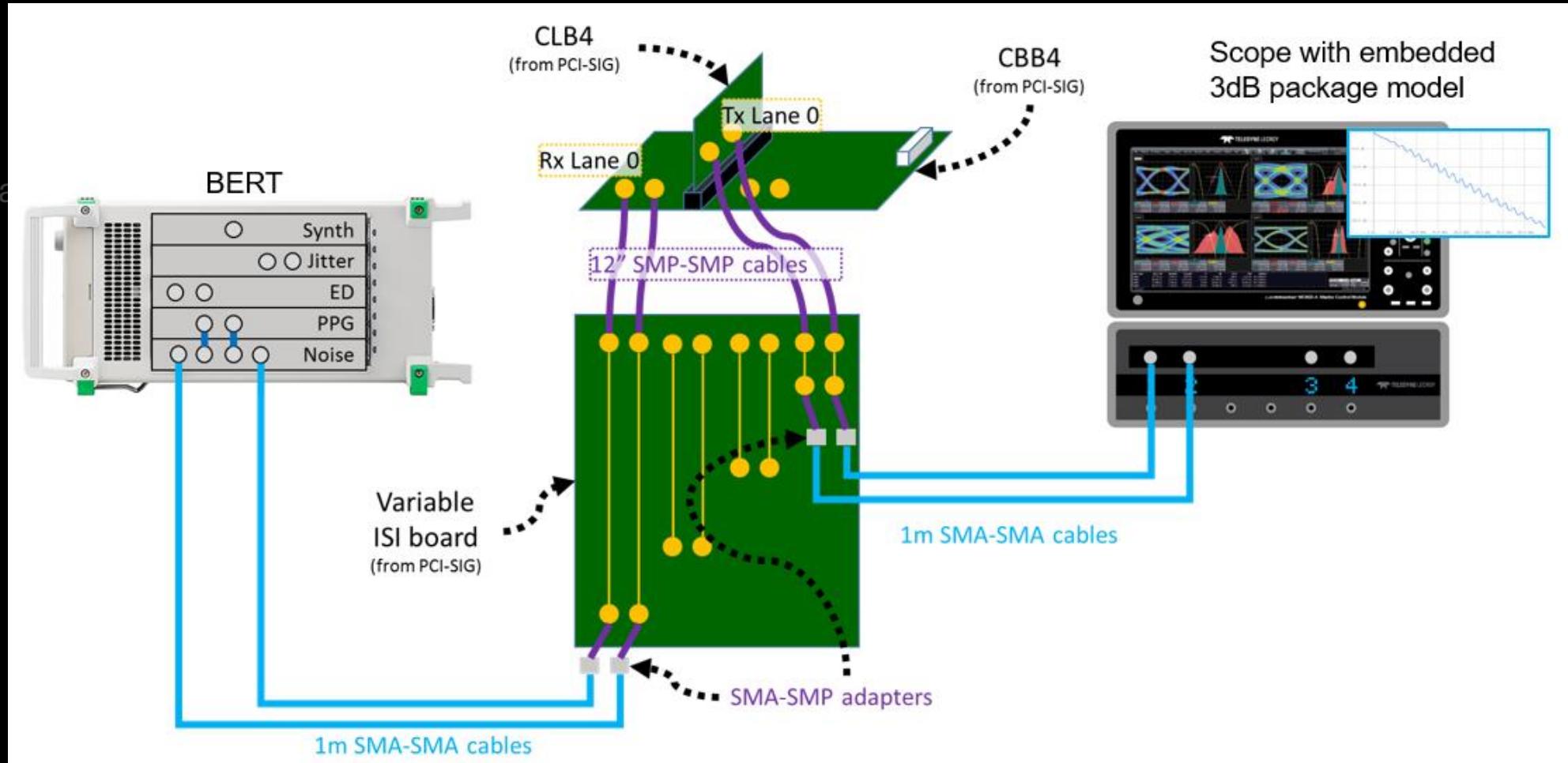
4. One EIEOS Block
5. 32 Data Blocks, each with a payload of 16 IDL data Symbols (00h) scrambled

Gen 3/4 Compliance Base Board (CBB)



Gen 3/4 Compliance Base Board (CBB)

asiclab



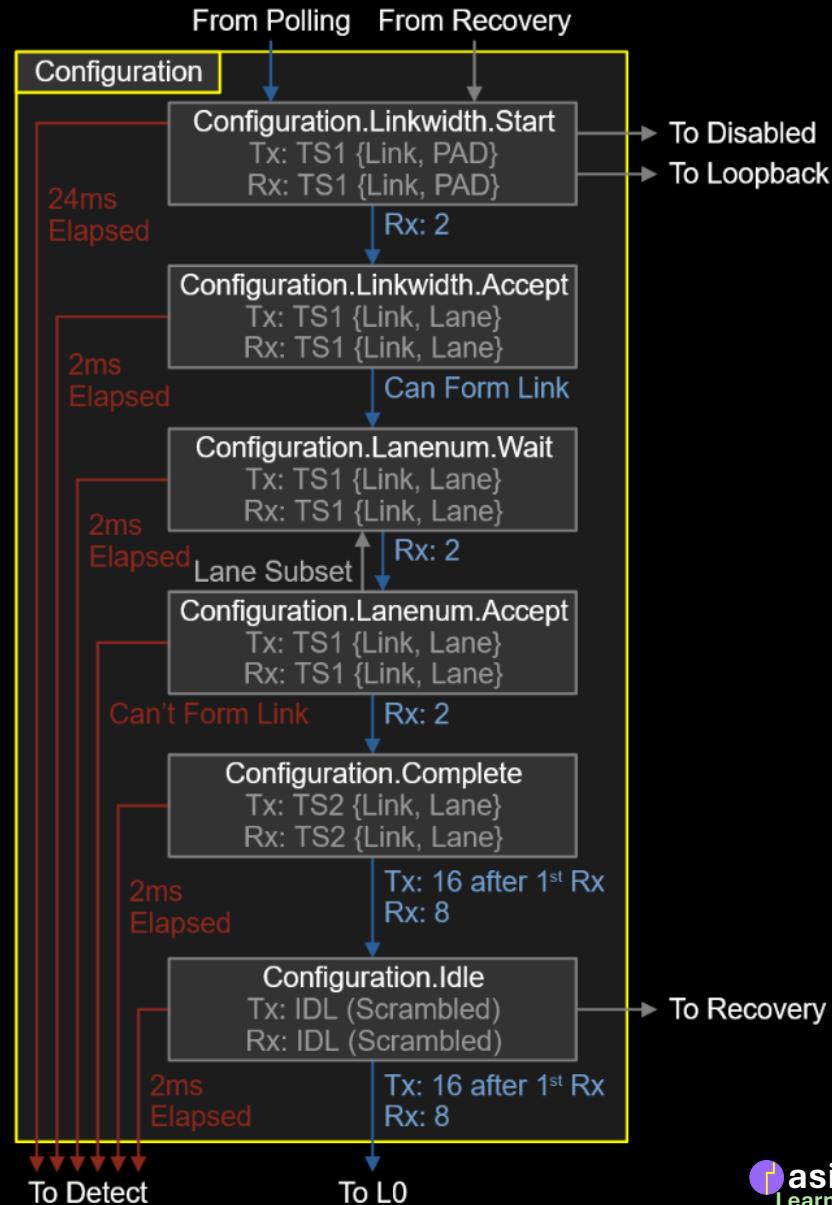
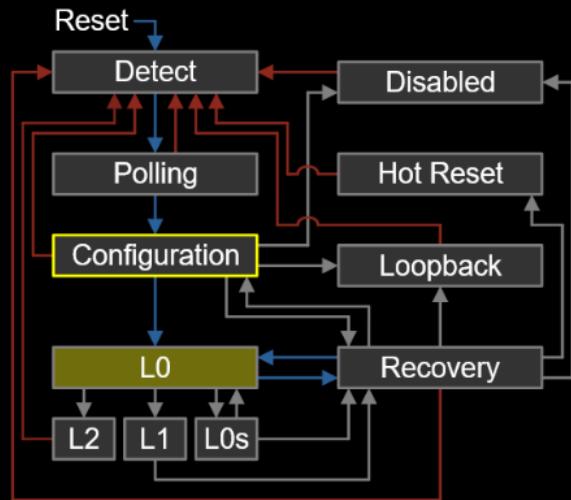
Gen 3/4 Modified Compliance Pattern

1. One EIEOS Block
2. 256 Data Blocks, each write a payload of 16 Idle data Symbols (00h), scrambled
3. 255 sets of the following sequence:
 1. One SKP Ordered Set
 2. 256 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled

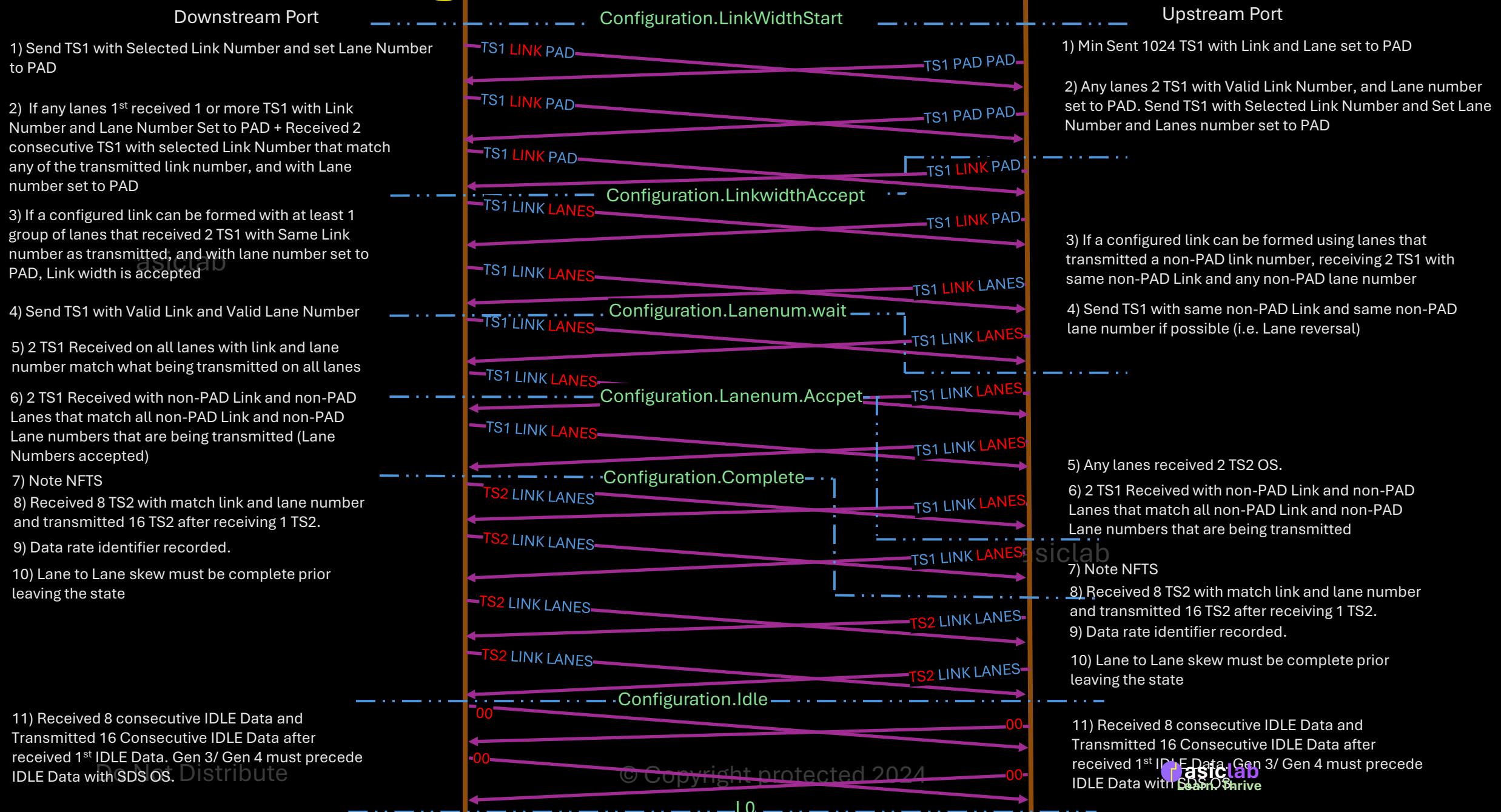
asiclab

LTSSM – Configuration State

asiclab



LTSSM – Configuration State



Link Number Negotiation

LTSSM – Configuration State

1. In Configuration State, Link Number Will Be Define.
2. Lane Number Of Each Lane Will Be Name Accordingly From “0” to “N-1” Where “N” Is Link Width Number.
3. Both Side Of The PCI Express Port Needs To Negotiate For Link Number And Lane Number That Can Be Supported And Both Port Will Try To Negotiate For Highest Common Data Width that supported
4. Width That Can Be Negotiate Is X1, X2, X4, X8, and X16.
5. During This Time, Deskew Will Be Perform Prior to transition to L0 state

LTSSM – Configuration State

6. If Dynamic Lane Reversal is supported, Link will auto negotiate to Dynamic Lane Reversal that can achieve highest common data width that support.

asiclab

7. If Dynamic Link Width negotiation is supported, link can always Transition from L0 → Recovery → Configuration to perform link width up down configurations.

8. Data Rate supported will be snap duringConfiguration.Complete if transition to L0 requirement meet

asiclab

LTSSM – Configuration State

Gen 3 Changes

1. Configuration.Link.Width.Start (Upstream Port)
 - a) Possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes.
 - b) Wait for additional two, or more, TS1 Ordered Sets, when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b encoding but not exceed 1ms.
2. Configuration.Link.Width.Accept/Lane.Num.Accept (Both)
 - a) Same as above.

LTSSM – Configuration State

Gen 3 Changes

3. Configuration.Complete

a) Disable scrambling – only applicable for Gen ½

asiclab
b) Timeout go detect – Gen ½ (2.0 Spec)

c) Timeout go to Configuration.Idle – idle_to_rlock < FF_for Gen 3 / 4

4. Configuration.Idle

a) Require to send SDS OS followed by Idle data for Gen 3

b) Rx Side – Will lock block alignment using SDS and startdetect Idle data.

asiclab

c) Block entering L0 if entered Config.Complete with timeout.

d) Timeout go to Recovery.Rcvr.Lock – idle_to_rlock increment to 1 for Gen 3/4 (Note: to FF for Gen 1 / 2)

e) Idle_to_rlock = 0 if successfully entered L0

LTSSM – Configuration State

Gen 4 Changes

1. Configuration.Lane.Num.Accept

- a) The deskew operation below should be delayed by 80 us or until the LTSSM enter Configuration.Complete. Since the retimer may take up to 40 us to change the elastic store (width changes), deskew should only be performed after that.

2. Configuration.Complete

- a) Retimer Presence Detect Supported (in LCAP2) bit set – Requires to receive 8TS2 ordered set with identical Retimer Present bit(Symbol 5 bit 4) when data rate is 2.5GT/s. If Retimer Present bit is set to 1 in the receivedTS2, Retimer Present Detected bit (LSTAT2) must be set to 1, else set to 0.

LTSSM – Configuration State

Gen 4 Changes

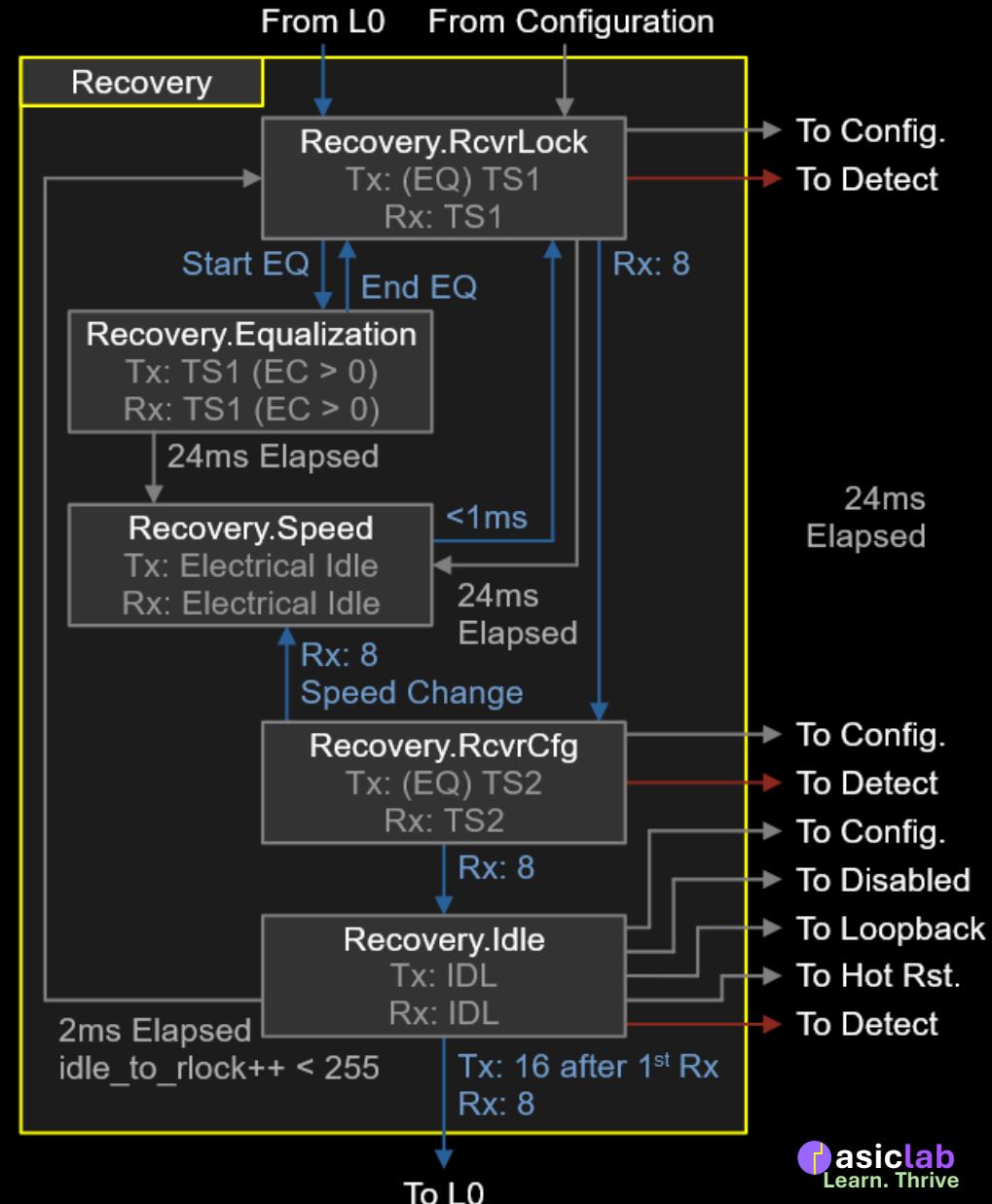
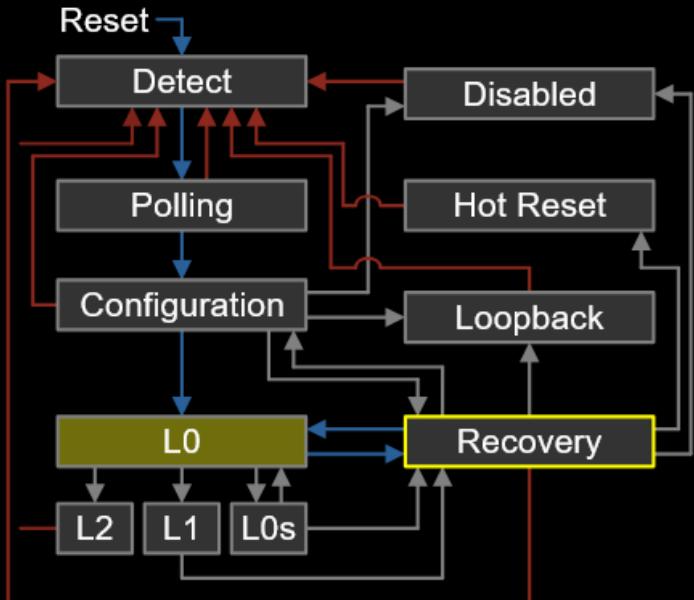
b) Two Retimer Presence Detect Supported (in LCAP2)bit set – Requires to receive 8TS2 ordered set with identical Retimer Present bit(Symbol 5 bit 5:4) when data rate is 2.5GT/s. If Two Retimer Present bit is set to 1 in the received TS2, Two Retimer PresentDetected bit (LSTAT2) must be set to 1, else set to 0.

3. Configuration.Idle

- a) Require to send SKP Control, followed by SDS OS and then Idle data for Gen 4.
- b) After enter L0, need to alternate standard SKP/ControlSKP.TxL0s exit does not require to send Control SKP.

LTSSM – Recovery State

asiclab



Do Not Dist

LTSSM – Recovery State (Physical Resync)

Downstream Port

1) Received 8 TS1/TS2 with Valid Link and Lanes
match its Transmitted for Gen 3/ Gen 4,
Received TS needs to be EC = 00

2) If Extended Sync enable, Transmitter
must send minimum 1024 TS1 before
transition to Recovery.RcvrCfg.

3) Transmitter send TS2 with Link and Lane
number were set after leaving
Configurations.

4) Received 8 TS2 on all configured lanes
with same link and lane number that
match its transmitted TS2

5) Transmit 16 TS2 after receiving 1st TS2

6) Gen speed Noted, NFTS Noted For its Gen

7) Received 8 Logical IDLE, and Transmit 16
Logicle idle after received 1st Logical Idle

Upstream Port

1) Received 8 TS1/TS2 with Valid Link and Lanes
match its Transmitted for Gen 3/ Gen 4,
Received TS needs to be EC = 00

2) If Extended Sync enable, Transmitter
must send minimum 1024 TS1 before
transition to Recovery.RcvrCfg.

3) Transmitter send TS2 with Link and Lane
number were set after leaving
Configurations.

4) Received 8 TS2 on all configured lanes
with same link and lane number that
match its transmitted TS2

5) Transmit 16 TS2 after receiving 1st TS2

6) Gen speed Noted, NFTS Noted For its Gen

7) Received 8 Logical IDLE, and Transmit 16
Logicle idle after received 1st Logical Idle

LTSSM – L0

- Upon Arrive Of L0 State, Link Is Consider As Healthy And IsReady To Transmit DLLP / TLP Packet. This Is The Only ActiveState That DLLP / TLP Is Allowed To Transmit.

asiclab

- Before TLP Can Be Transmitted, Init Flow Control Credit NeedsTo Be Exchange Amount Both Port. Both Port Needs To GetInformation Of Opposite Device Que Size Before Can PresentAny TLP Packet To Opposite Device.

- As Soon As Init Flow Control Credit Get Updated Properly, TLPCan Be Present To Opposite Device.

asiclab

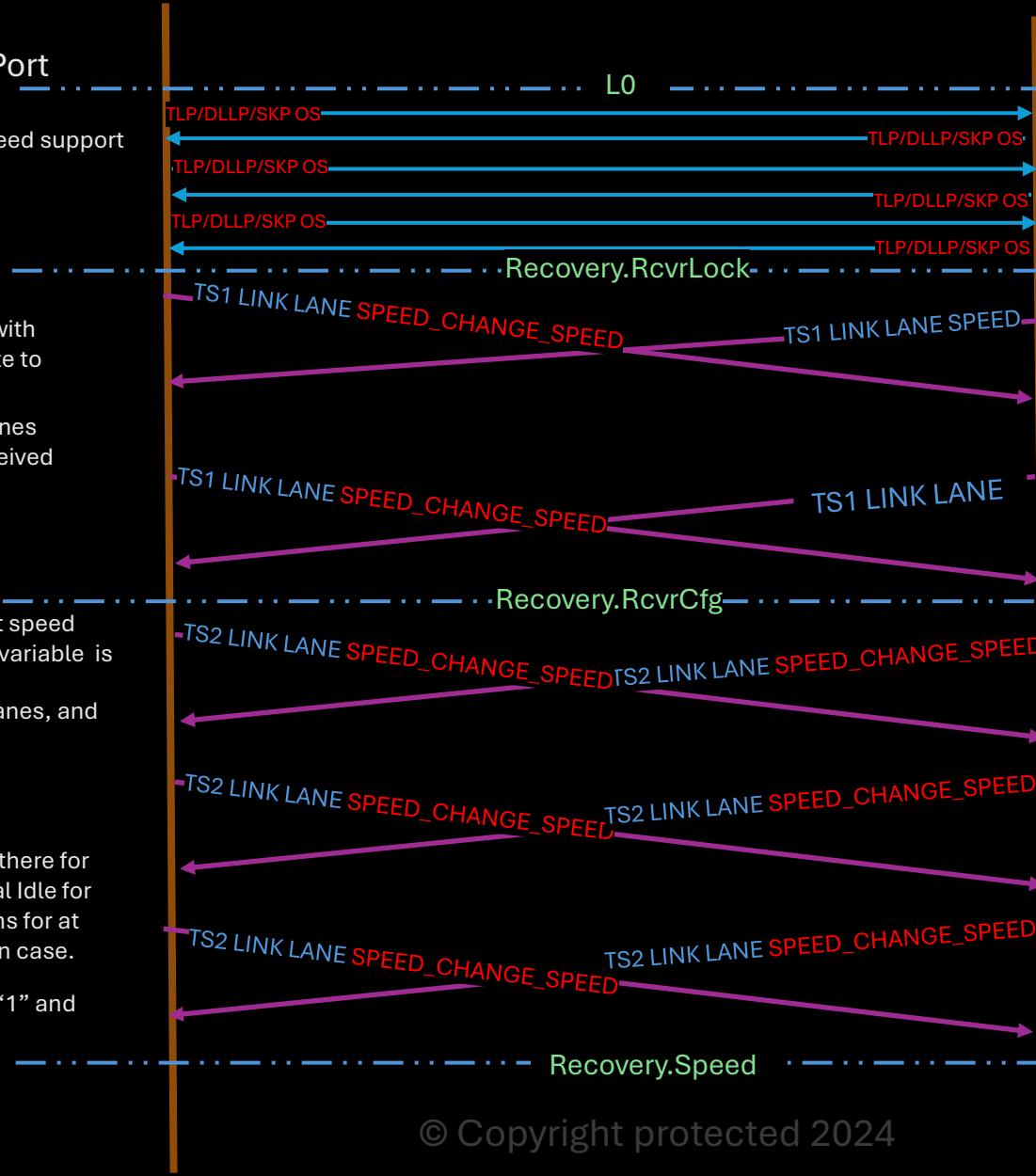
- Just In Case There Is Any Error That Cause Link Not Reliable ToTransmit TLP, Link Can Always Go Back To Recovery State ToPerform Quick Retrain.

LTSSM – Recovery State (Speed Change)

Downstream Port

- 1) Target Link Speed > Gen 1, Remote Gen speed support noted > Gen 1
- 2) Speed Change Event Occurs (Directed/Hardware Autonomous)
- 3) Set directed_speed_change variable to 1
- 4) Send TS1 with Valid Link and Valid Lanes, with Speed Change Bit set to "1". Change Data rate to targeted data rate.
- 5) Received 8 TS1 / TS2 with Valid Link and Lanes match its Transmitted For Gen3 / Gen 4, Received TS needs to be EC = 00
- 6) Send TS2 with Valid Link and Lanes and set speed change bit to "1" as directed_speed_change variable is "1"
- 7) Received 8 TS2 with Valid Link and Valid Lanes, and with speed_change_bit set to "1"
- 8) Set successful_speed_negotiation to "1"
- 9) Transmitter in Electrical Idle and remains there for additional of 800ns after receiver in Electrical Idle for success speed negotiation case, and remains for at least 6 us on unsuccessful speed negotiation case.
- 10) Change_speed_recovery variable set to "1" and directed_speed_change variable set to "0"

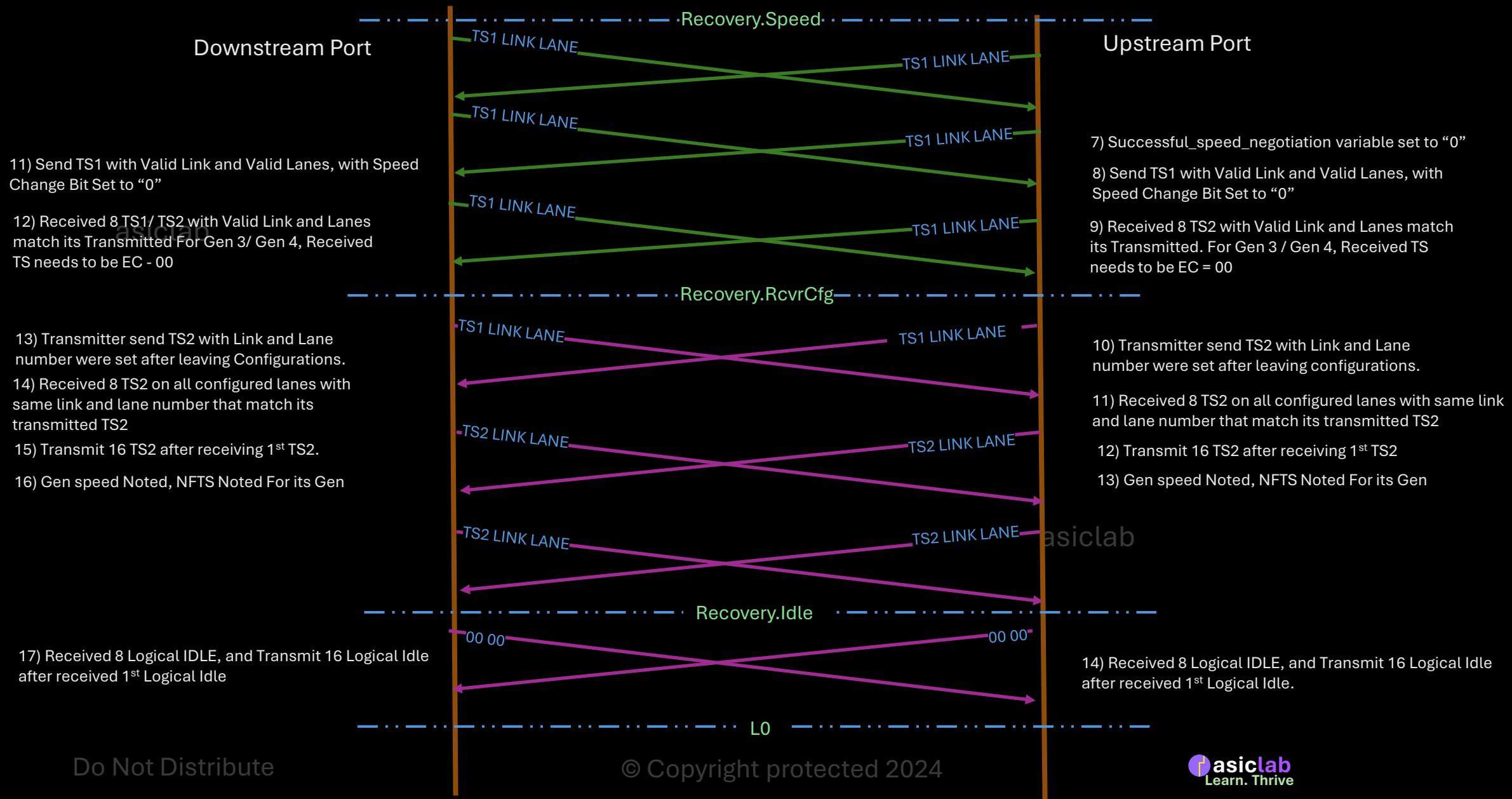
Upstream Port



1) Received 8 TS1/TS2 with Valid Link and Lanes match its Transmitted for Gen 3/ Gen 4, Received TS needs to be EC = 00. Set directed_speed_change variable when received 8 TS1 have Speed Change bit set to "!"

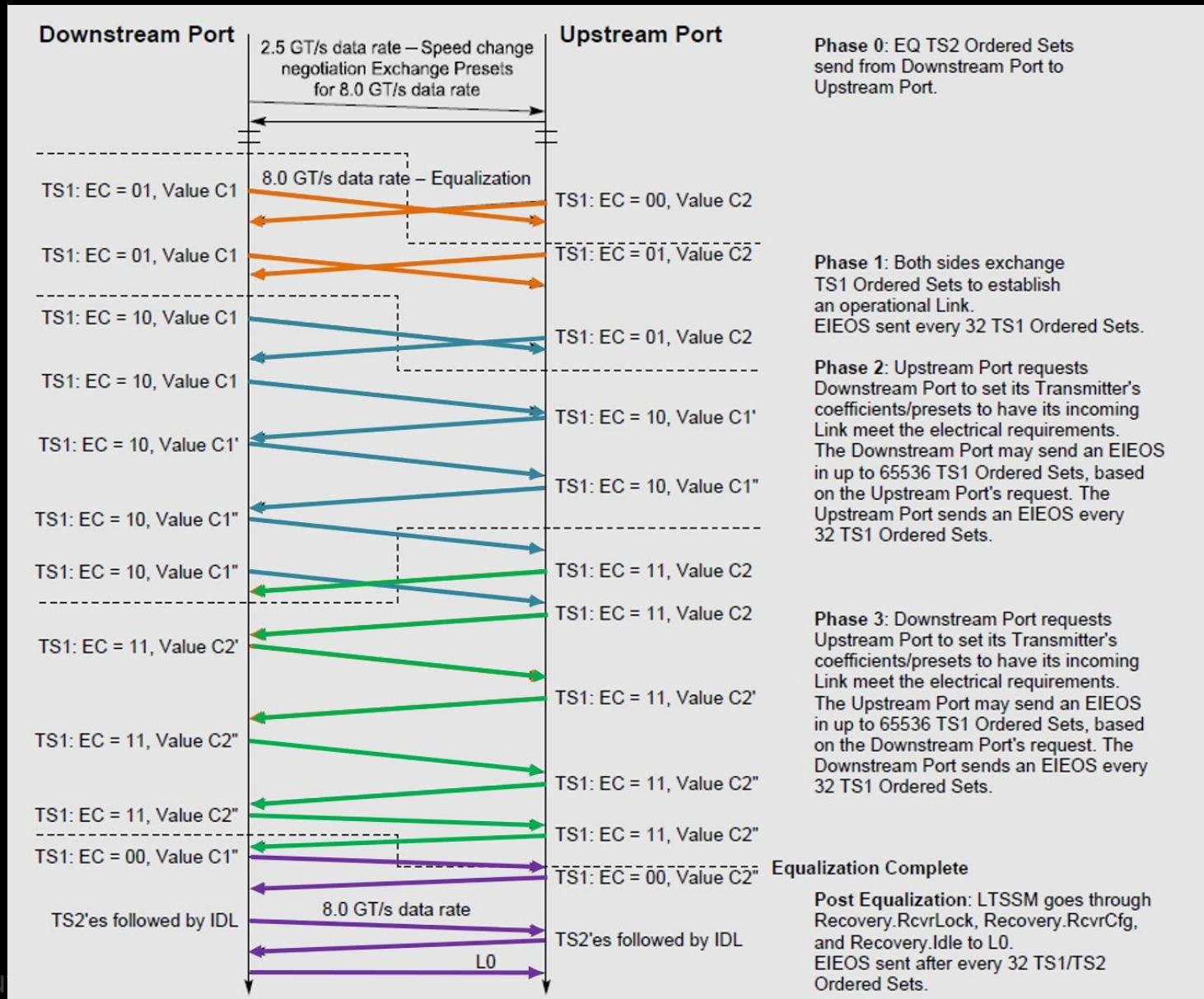
- 1) Received 8 TS1/TS2 with Valid Link and Lanes match its Transmitted for Gen 3/ Gen 4, Received TS needs to be EC = 00. Set directed_speed_change variable when received 8 TS1 have Speed Change bit set to "!"
- 2) Send TS2 with Valid Link and Lanes and set speed_change_bit to "1" as directed_speed_change variable is "1"
- 3) Received 8 TS2 with Valid Link and Valid Lanes, and with speed_change_bit set to "1" and send 32 TS2 after receiving 1st TS2 for Gen1/ Gen 2, or send 128 TS2 after receiving 1st TS2 for Gen 3/ Gen 4
- 4) Set_successful_speed_negotiation to "!"
- 5) Transmitter in Electrical Idle and remains there for additional of 800ns after receiver in Electrical Idle for success speed negotiation case, and remains for at least 6 us on unsuccessful speed negotiation case.
- 6) Change_speed_recovery variable set to "1" and directed_speed_change variable set to "0"

LTSSM – Recovery State (Speed Change)



Equalization

asiclab



Do Not Distribu

Encoding

asiclab

(Symbol Time)	Sample Number	Lane 0	Lane 1	Lane 2	Lane 3
600.016 us	15213	00	00	00	00
600.017 us	15214	5F	82	CA	C4
600.018 us	15214	4A	00	00	20
600.019 us	15214	00	00	02	00
600.020 us	15214	01	00	36	00
600.021 us	15214	F7	AF	F7	FF
600.022 us	15214	FF	F5	FF	6D
600.023 us	15214	FF	EF	FF	FF
600.024 us	15214	7F	FF	FF	EF
600.025 us	15214	7E	54	FD	9B
600.026 us	15214	3F	6D	FD	BE
600.027 us	15214	FF	F7	6F	FF
600.028 us	15214	7F	FF	FF	FF
600.029 us	15214	7F	FF	E7	BF
600.030 us	15214	FE	7F	BD	FF
600.031 us	15214	FF	FF	FF	E3
600.032 us	15214	FF	FF	DD	FF
600.033 us	15214	FF	FF	FF	FD
600.034 us	15214	DD	5D	FF	FF
600.035 us	15214	FA	FF	FF	FF
600.036 us	15214	BF	FF	FF	FF
600.037 us	15214	FF	DF	FF	FF
600.038 us	15214	DB	FF	FF	DF
600.039 us	15214	FF	FE	FF	FF
600.040 us	15214	D7	FF	FB	FD
600.041 us	15214	5F	C9	67	57
600.042 us	15214	F6	EE	FB	FF
600.043 us	15214	FF	69	FB	DF
600.044 us	15214	B7	EF	FF	FF
600.045 us	15214	FF	FF	CD	DF
600.046 us	15214	FF	B7	C1	F5
600.047 us	15214	6F	FF	FE	D7
600.048 us	15214	FF	77	3F	FF
600.049 us	15214	FF	FB	DF	DF
600.050 us	15214	DF	FF	FE	FF
600.051 us	15214	FF	DF	FF	FF
600.052 us	15214	FF	FF	FF	FF
600.053 us	15214	9D	2D	0B	72
600.054 us	15214	00	00	00	00
600.055 us	15214	00	00	00	00
600.056 us	15214	00	00	00	00
600.058 us	15215	5F	82	CA	C5

STP (4B)

TLP Header
(3DW = 12B)

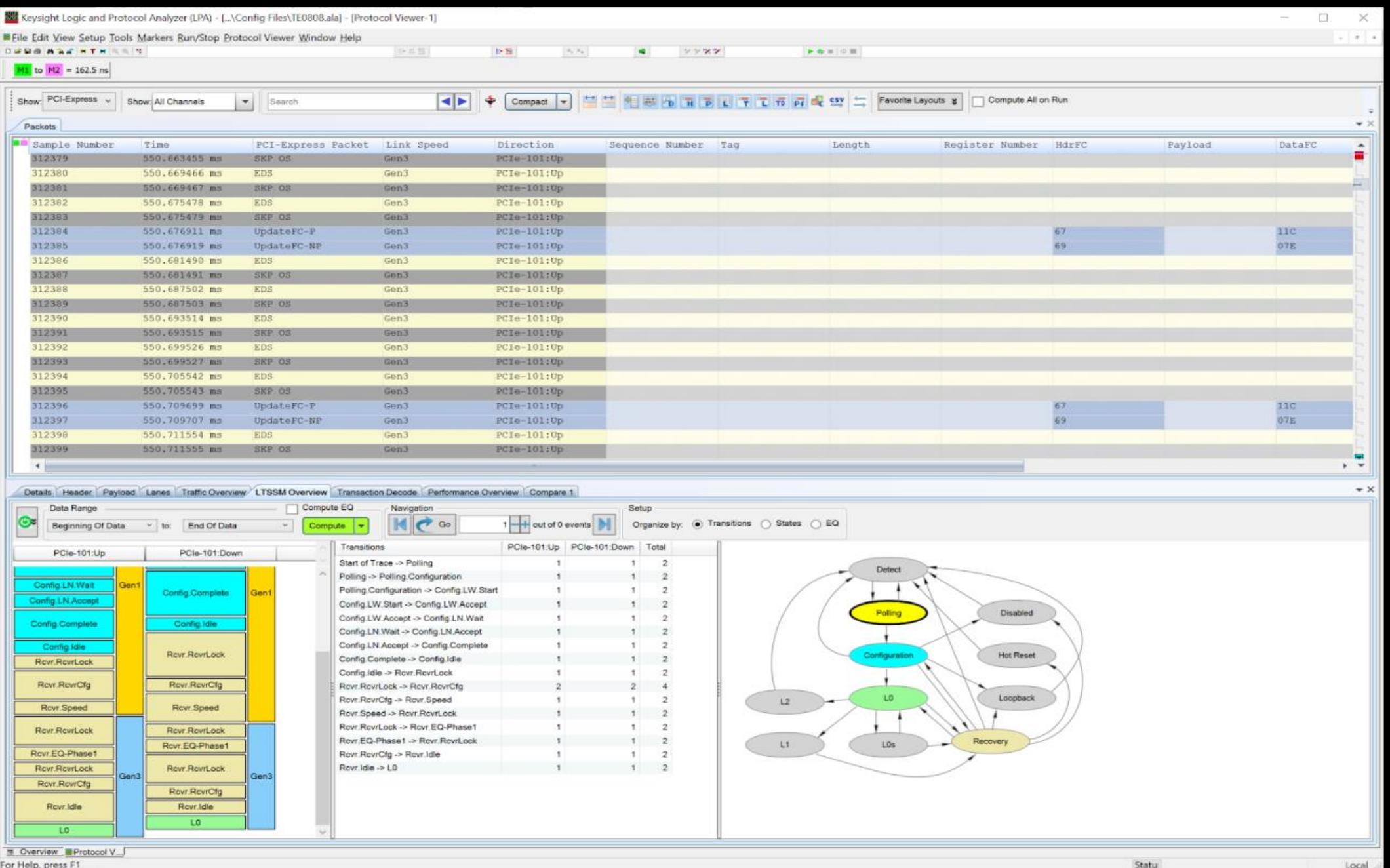
Data Payload
(32DW = 128B)

LCRC (4B)

Alignment (12B)

asiclab

Do Not Distribute



Do Not Distribute

© Copyright protected 2024

Traces

Details Header Payload Lanes Traffic Overview LTSSM Overview Transaction Decode Performance Overview Compare 1

Data Range: M1 to M2 Compute EQ Navigation: Go 1 out of 0 events Setup: Organize by: Transitions States EQ

PCIe-101:Up	PCIe-101:Down	Transitions	PCIe-101:Up	PCIe-101:Down	Total
Polling		Start of Trace -> Polling	1	1	2
Polling Configuration		Polling -> Polling.Configuration	1	1	2
Config.LW.Start		Polling.Configuration -> Config.LW.Start	1	1	2
Config.LW.Accept		Config.LW.Start -> Config.LW.Accept	1	1	2
Config.LN.Wait		Config.LW.Accept -> Config.LN.Wait	1	1	2
Config.LN.Accept		Config.LN.Wait -> Config.LN.Accept	1	1	2
Config.LW.Accept		Config.LN.Accept -> Config.Complete	1	1	2
Config.LN.Wait		Config.Complete -> Config.Idle	1	1	2
Config.LN.Accept		Config.Idle -> Rcvr.RcvrLock	1	1	2
Config Complete		Rcvr.RcvrLock -> Rcvr.RcvrCfg	2	2	4
Config Idle		Rcvr.RcvrCfg -> Rcvr.Speed	1	1	2
Rcvr.RcvrLock		Rcvr.Speed -> Rcvr.RcvrLock	1	1	2
Rcvr.RcvrCfg		Rcvr.RcvrLock -> Rcvr.EQ-Phase1	1	1	2
Rcvr.Speed		Rcvr.EQ-Phase1 -> Rcvr.RcvrLock	1	1	2
Rcvr.RcvrLock		Rcvr.RcvrCfg -> Rcvr.Idle	1	1	2
Rcvr.EQ-Phase1		Rcvr.Idle -> L0	1	1	2
Gen1	Gen1				
Gen3	Gen3				

```

graph TD
    Detect((Detect)) --> Polling((Polling))
    Polling --> Configuration((Configuration))
    Configuration --> L0s((L0s))
    Configuration --> Recovery((Recovery))
    L0s --> L0((L0))
    L0 --> L1((L1))
    L1 --> L2((L2))
    L2 --> Detect
    L0s --> Loopback((Loopback))
    Loopback --> HotReset((Hot Reset))
    HotReset --> Disabled((Disabled))
    Disabled --> Detect
    Recovery --> L0s
    Recovery --> Loopback
    Recovery --> HotReset
    Recovery --> Disabled
    L0s --> Recovery
    L1 --> Recovery
    L2 --> Recovery
    L2 --> Loopback
    L2 --> HotReset
    L2 --> Disabled
    Loopback --> L0s
    Loopback --> Recovery
    Loopback --> HotReset
    Loopback --> Disabled
    HotReset --> L0s
    HotReset --> Recovery
    HotReset --> Loopback
    HotReset --> Disabled
    Disabled --> L0s
    Disabled --> Recovery
    Disabled --> Loopback
    Disabled --> HotReset
  
```

Summary

Reference: Supplementary LTSSM Sheet



asiclab

asiclab

Do Not Distribute

© Copyright protected 2024

asiclab
Learn. Thrive

asiclab

Gen 6.0 Link Training

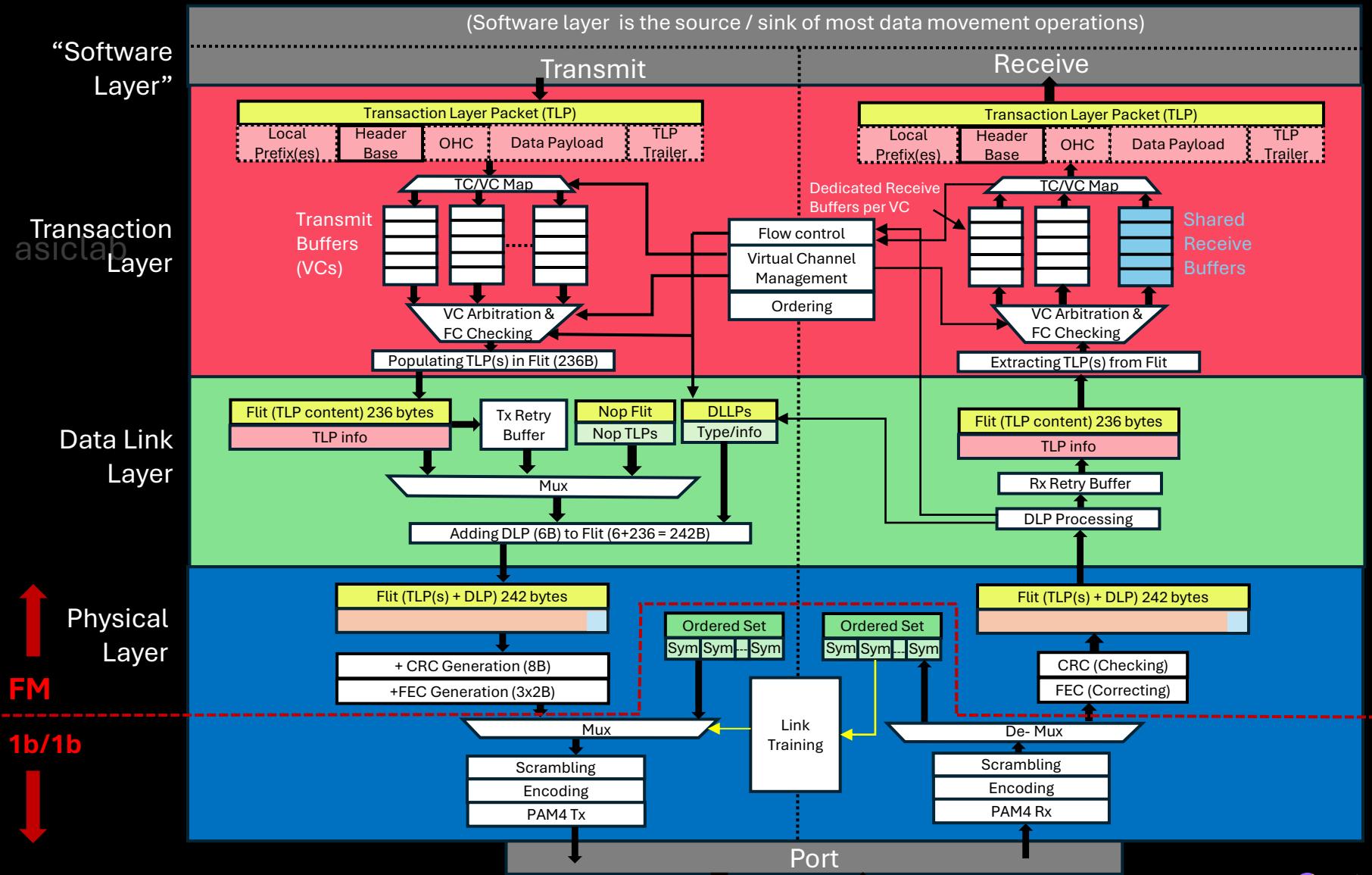
asiclab

Do Not Distribute

© Copyright protected 2024

asiclab
Learn. Thrive

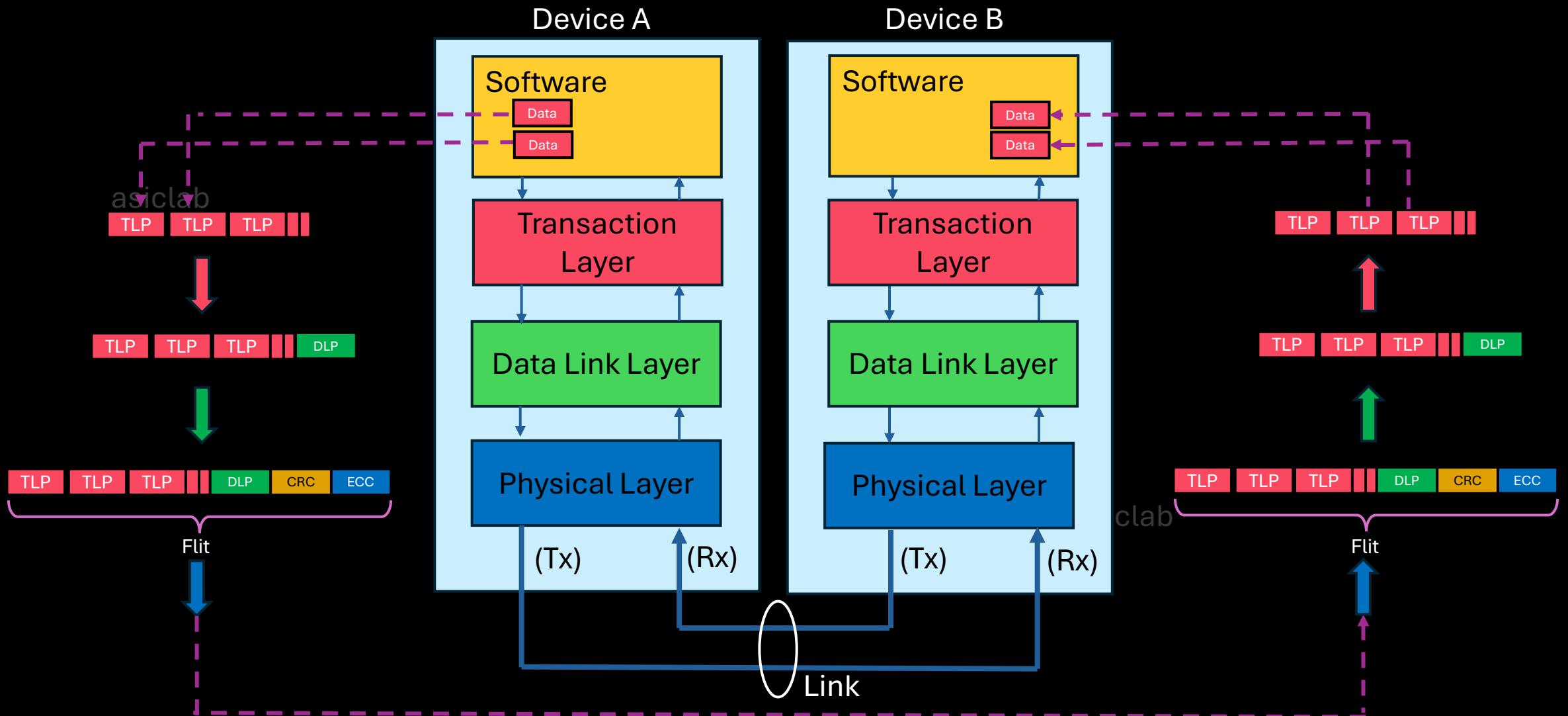
PCIe Device Layer Details 6.0 (Flit Mode – FM) (1b/1b)



Do Not Distribute

© Copyright protected 2024
asiclab

Flit Assembly/Disassembly (New with PCIe 6.0)



Gen 6.0 Flit Types

- In Flit mode there are 3 types of flits identified at the Data link layer
 - IDLE Flit -> all the contents are 0 in the TLP portion and the DLP portion will have the sequence number 0. they are only sent just before entering to L0 Entering L0 can be done from config and ~~and~~ recovery, in both these states just before entering L0 there is a state called config.idle and recovery.idle. In Non flit mode we will be sending Logical idles in that state before entering L0. In case of FM this is not available , instead we will send IDLE Flit
 - NOP Flit ->in case of this all the content of TLP portion is 0, but DLP can be non zero, this is sent when there are no TLP to send so these are sent as a filler for the TLP portion
 - Payload Flit -> TLP portion is carrying TLPs and the sequence number is non zero

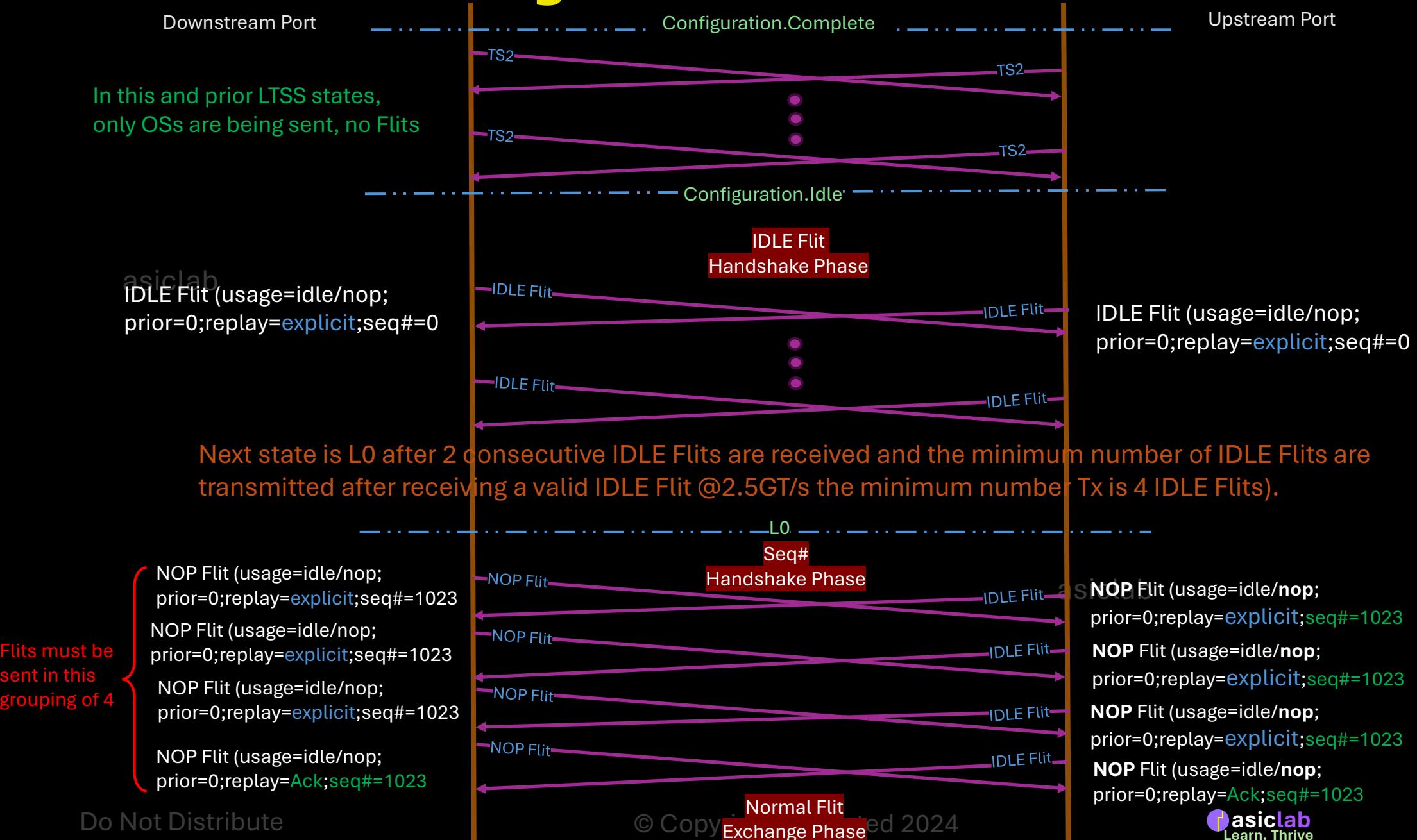
asiclab

Gen 6.0 Sequence Number

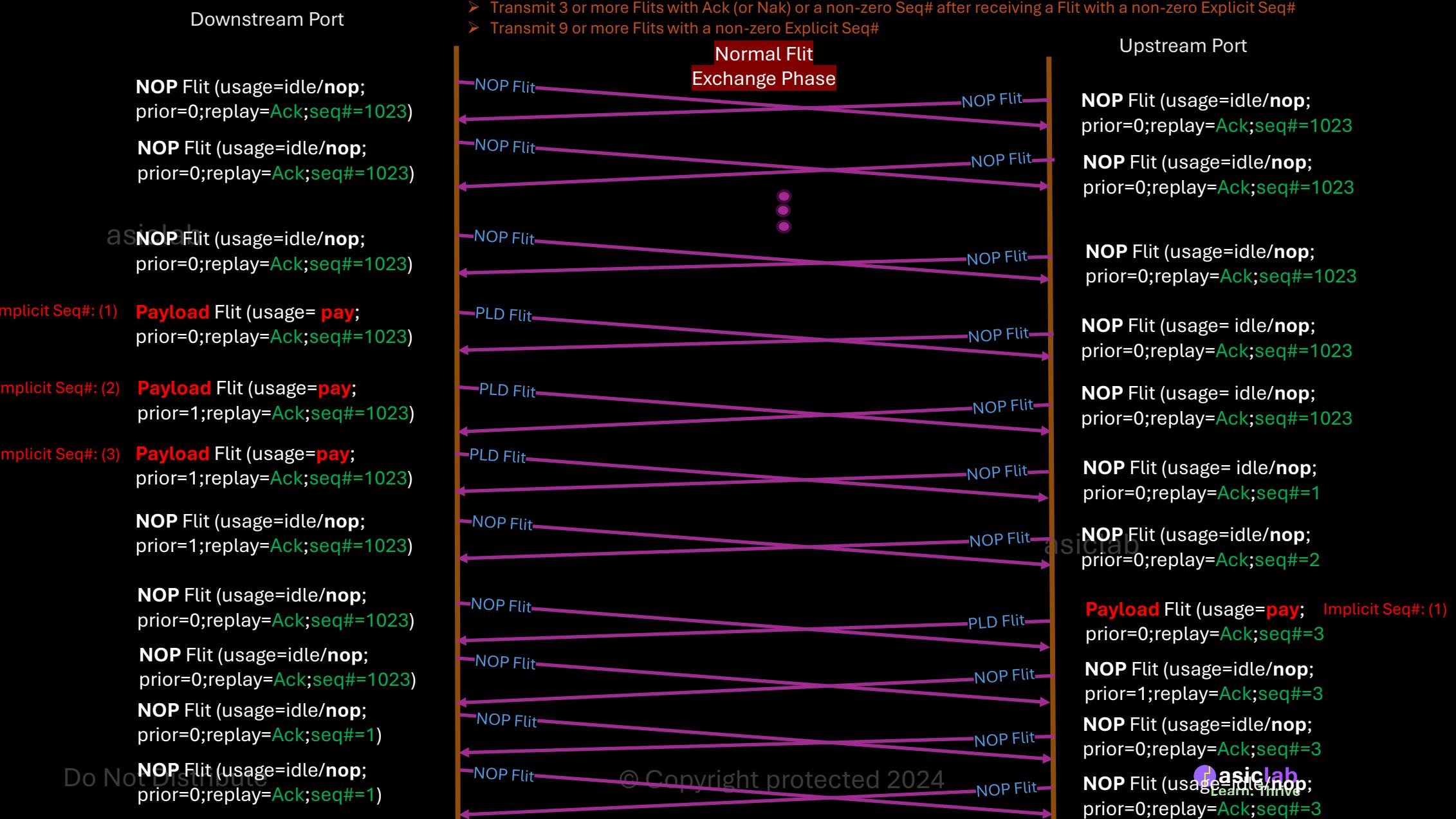
- Sequence number which is 10 bit value is less than we used to have in flit mode.
 - Which means sequence number wraps from 1023 to 1. Note that sequence 0 is reserved and cannot be used, it is only to be used with an IDLE flit
 - 10 bit value for sequence means we can have only maximum of 511 outstanding flits in the buffer
 - First valid flit will have the implicit seq number 1
 - Every payload flit has a unique sequence number but it is implicit, meaning that the sequence number is not sent across the Physical layer. The sequence number is implicitly tracked at the Tx and the RX side using local counters
 - A Sequence number is explicitly sent along with a TLP is when we are starting a replay or when we are sending ACK or NAK for a TLP that we already received

asiclab

LTSSM – 6.0 changes



LTSSM – 6.0 changes



LTSSM Debug

asiclab

6277	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	6277	4A
6281	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	6281	4A
6285	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	6285	4A
6289	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	6289	4A
6293	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	6293	4A
6297	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	6297	4A
6301	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	6301	4A
6305	COM	6305	4A														
6309	PAD	6309	4A														
6313	PAD	6313	4A														
6317	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	6317	COM
6321	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	6321	PAD
6325	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	6325	PAD
6329	4A	6329	0C														
6333	4A	6333	02														
6337	4A	6337	00														
6341	4A	6341	4A														
6345	4A	6345	4A														
6349	4A	6349	4A														
6353	4A	6353	4A														
6357	4A	6357	4A														
6361	4A	6361	4A														
6365	4A	6365	4A														
6369	COM	6369	4A														
6373	PAD	6373	4A														
6377	PAD	6377	4A														
6381	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	6381	COM
6385	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	6385	PAD
6389	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	6389	PAD
6393	4A	6393	0C														
6397	4A	6397	02														
6401	4A	6401	00														
6405	4A	6405	4A														
6409	4A	6409	4A														
6413	4A	6413	4A														
6417	4A	6417	4A														
6421	4A	6421	4A														
6425	4A	6425	4A														
6429	4A	6429	4A														
6433	COM	6433	4A														

LTSSM Debug

asicla

Configuration State

Negotiate Link Number

9121	4H	9121	4H															
9125	4A	9125	4A															
9129	4A	9129	4A															
9133	4A	9133	4A															
9137	COM	9137	4A															
9141	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	9141	4A
9145	PAD	9145	4A															
9149	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	9149	COM
9153	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	9153	00
9157	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	9157	PAD
9161	4A	9161	0C															
9165	4A	9165	02															
9169	4A	9169	00															
9173	4A	9173	4A															
9177	4A	9177	4A															
9181	4A	9181	4A															
9185	4A	9185	4A															
9189	4A	9189	4A															
9193	4A	9193	4A															
9197	4A	9197	4A															
9201	COM	9201	4A															
9205	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	9205	4A
9209	PAD	9209	4A															
9213	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	9213	COM
9217	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	9217	00
9221	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	9221	PAD
9225	4A	9225	0C															
9229	4A	9229	02															
9233	4A	9233	00															
9237	4A	9237	4A															
9241	4A	9241	4A															
9245	4A	9245	4A															
9249	4A	9249	4A															
9253	4A	9253	4A															
9257	4A	9257	4A															
9261	4A	9261	4A															
9265	COM	9265	4A															
9269	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	9269	4A
9273	PAD	9273	4A															
9277	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	9277	COM
9281	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	9281	00

LTSSM Debug

asiclab

10153	4A	10153	4A															
10157	4A	10157	4A															
10161	COM	10161	4A															
10165	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10165	4A
10169	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10169	4A
10173	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	10173	COM
10177	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	10177	00
10181	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10181	00
10185	4A	10185	0C															
10189	4A	10189	02															
10193	4A	10193	00															
10197	4A	10197	4A															
10201	4A	10201	4A															
10205	4A	10205	4A															
10209	4A	10209	4A															
10213	4A	10213	4A															
10217	4A	10217	4A															
10221	4A	10221	4A															
10225	COM	COM	COM	I	COM	10225	4A											
10229	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10229	4A
10233	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10233	4A
10237	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	10237	COM
10241	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	10241	00
10245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10245	00
10249	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10249	0C
10253	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10253	02
10257	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10257	00
10261	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10261	4A
10265	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10265	4A
10269	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10269	4A
10273	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10273	4A
10277	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10277	4A
10281	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10281	4A
10285	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	10285	4A
10289	COM	10289	4A															
10293	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10293	4A
10297	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10297	4A
10301	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	10301	COM
10305	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	10305	00
10309	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10309	00

Configuration State

Negotiated Link And Lane Number

LTSSM Debug

11557	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11557	00
11561	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11561	00
11565	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11565	00
11569	COM	11569	00																
11573	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11573	00
11577	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	11577	00	
11581	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	11581	00
11585	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	11585	00
11589	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11589	00
11593	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11593	00
11597	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11597	00
11601	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11601	00
11605	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11605	00
11609	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11609	00
11613	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11613	00
11617	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11617	00
11621	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11621	00
11625	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11625	00
11629	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	11629	00
11633	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11633	00
11637	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11637	00
11641	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11641	00
11645	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11645	COM
11649	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11649	SKP
11653	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11653	SKP
11657	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11657	SKP
11661	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11661	00
11665	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11665	00
11669	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11669	00
11673	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11673	00
11677	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11677	00
11681	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11681	00
11685	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11685	00
11689	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11689	00
11693	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11693	00
11697	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11697	00
11701	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11701	00
11705	COM	11705	00																
11709	SKP	11709	00																

Lo State
Logical IDLE

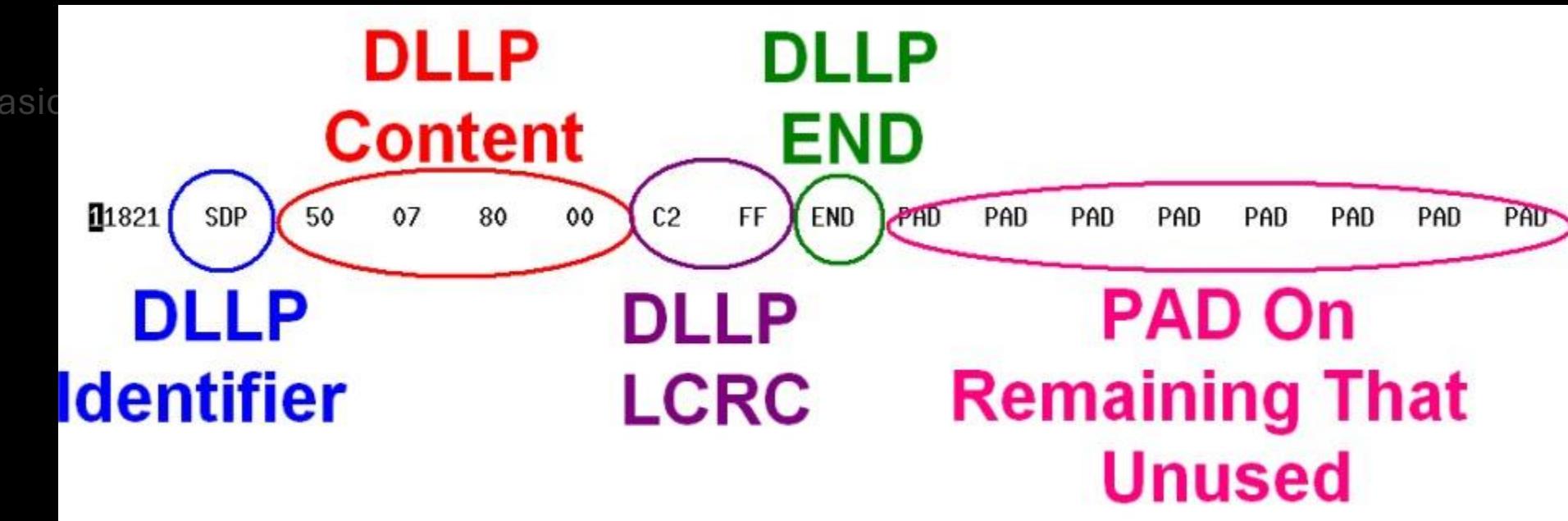
LTSSM Debug

asiclab

11661	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11661	00
11665	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11665	00
11669	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11669	00
11673	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11673	00
11677	I	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11677	00
11681	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11681	00
11685	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11685	00
11689	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11689	00
11693	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11693	00
11697	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11697	00
11701	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11701	00
11705	COM	11705	00															
11708	SKP	11709	00															
11713	SKP	11713	00															
11717	SKP	11717	SDP															
11721	SDP	40	07	80	80	21	48	END	PAD	11721	SDP							
11725	SDP	50	07	80	00	C2	FF	END	PAD	11725	SDP							
11729	SDP	60	00	00	00	D8	92	END	PAD	11729	SDP							
11733	SDP	40	07	80	80	21	48	END	PAD	11733	SDP							
11737	SDP	50	07	80	00	C2	FF	END	PAD	11737	SDP							
11741	SDP	60	00	00	00	D8	92	END	PAD	11741	SDP							
11745	SDP	40	07	80	80	21	48	END	PAD	11745	SDP							
11749	SDP	50	07	80	00	C2	FF	END	PAD	11749	SDP							
11753	SDP	60	00	00	00	D8	92	END	PAD	11753	SDP							
11757	SDP	40	07	80	80	21	48	END	PAD	11757	SDP							
11761	SDP	50	07	80	00	C2	FF	END	PAD	11761	SDP							
11765	SDP	60	00	00	00	D8	92	END	PAD	11765	SDP							
11769	SDP	40	07	80	80	21	48	END	PAD	11769	SDP							
11773	SDP	50	07	80	00	C2	FF	END	PAD	11773	SDP							
11777	SDP	60	00	00	00	D8	92	END	PAD	11777	SDP							
11781	SDP	40	07	80	80	21	48	END	PAD	11781	SDP							
11785	SDP	50	07	80	00	C2	FF	END	PAD	11785	SDP							
11789	SDP	60	00	00	00	D8	92	END	PAD	11789	SDP							
11793	SDP	40	07	80	80	21	48	END	PAD	11793	SDP							
11797	SDP	50	07	80	00	C2	FF	END	PAD	11797	SDP							
11801	SDP	60	00	00	00	D8	92	END	PAD	11801	SDP							
11805	SDP	40	07	80	80	21	48	END	PAD	11805	SDP							
11809	SDP	50	07	80	00	C2	FF	END	PAD	11809	SDP							
11813	SDP	60	00	00	00	D8	92	END	PAD	11813	SDP							
11817	SDP	40	07	80	80	21	48	END	PAD	11817	SDP							
11821	SDP	50	07	80	00	C2	FF	END	PAD	11821	SDP							

Data Link Layer Packet

LTSSM Debug

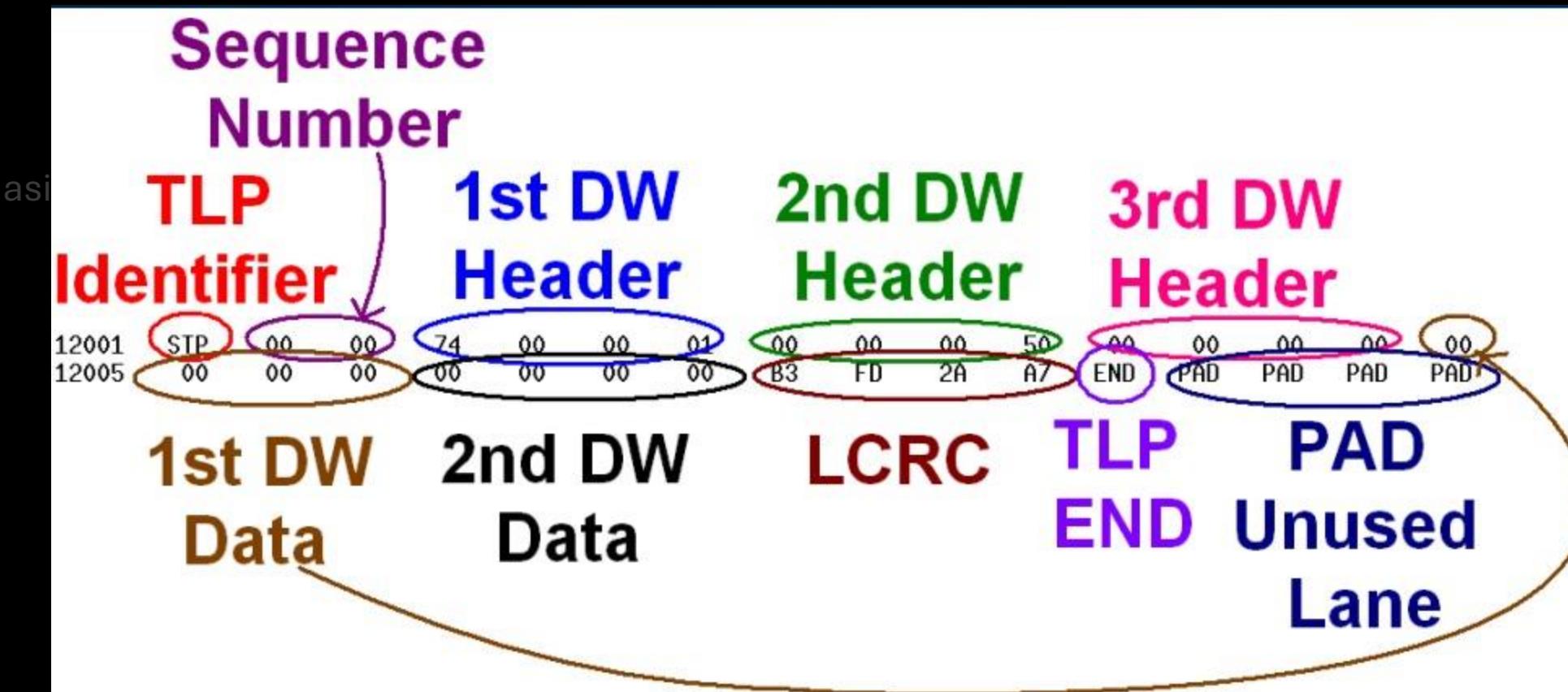


LTSSM Debug

11937	SDP	C0	07	80	80	5B	37	END	PAD	11937	SDP								
11941	SDP	D0	07	80	00	B8	80	END	PAD	11941	SDP								
11945	SDP	E0	00	00	00	A2	ED	END	PAD	11945	00								
11949	SDP	C0	07	80	80	5B	37	END	PAD	11949	00								
11953	SDP	D0	07	80	00	B8	80	END	PAD	11953	00								
11957	SDP	E0	00	00	00	A2	ED	END	PAD	11957	00								
11961	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11961	00
11965	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11965	00
11969	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11969	00
11973	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11973	00
11977	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11977	00
11981	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11981	00
11985	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11985	00
11989	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11989	00
11993	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11993	STP
11997	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11997	DE
12001	STP	00	00	74	00	00	01	00	00	00	50	00	00	00	00	00	00	12001	E4
12005	00	00	00	00	00	00	B3	FD	2A	A7	END	PAD	PAD	PAD	PAD	PAD	12005	23	
12009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12009	96
12013	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12013	3B
12017	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12017	8A
12021	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12021	B9
12025	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12025	34
12029	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12029	C6
12033	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12033	1D
12037	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12037	SDP
12041	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12041	00
12045	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12045	E6
12049	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12049	58
12053	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12053	8F
12057	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12057	E2
12061	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12061	00
12065	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12065	E8
12069	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12069	51
12073	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12073	ED
12077	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12077	53
12081	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12081	STP
12085	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12085	E2
12089	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12089	FA

Transaction Layer Packet

LTSSM Debug



LTSSM Debug

asicl

21465	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21465	00
21469	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21469	00
21473	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21473	00
21477	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21477	00
21481	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21481	00
21485	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21485	00
21489	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21489	00
21493	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21493	00
21497	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21497	00
21501	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21501	00
21505	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21505	00
21509	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21509	00
21513	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21513	00
21517	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21517	00
21521	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21521	00
21525	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21525	00
21529	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21529	00
21533	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21533	00
21537	COM	21537	00														
21541	IDL	21541	00														
21545	IDL	21545	00														
21549	IDL	21549	00														
21553	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21553	00
21557	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21557	00
21561	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21561	00
21565	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21565	00
21569	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21569	00
21573	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21573	00
21577	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21577	00
21581	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21581	00
21585	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21585	00
21589	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21589	00
21593	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	I	Z	Z	Z	21593	00
21597	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21597	00
21601	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21601	00
21605	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21605	00
21609	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21609	00
21613	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21613	00
21617	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21617	00
21621	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	21621	00

LTSSM Debug

asic

22713	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22713	00
22717	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22717	00
22721	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22721	00
22725	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22725	00
22729	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22729	00
22733	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22733	00
22737	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22737	00
22741	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22741	00
22745	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22745	00
22749	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22749	00
22753	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22753	00
22757	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22757	00
22761	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22761	00
22765	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22765	00
22769	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22769	00
22773	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22773	00
22777	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22777	00
22781	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22781	00
22785	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22785	00
22789	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22789	00
22793	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22793	00
22797	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22797	00
22801	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22801	00
22805	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22805	00
22809	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22809	00
22813	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22813	00
22817	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22817	00
22821	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22821	00
22825	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	22825	00
22829	COM	22829	00																
22833	FTS	22833	00																
22837	FTS	22837	00																
22841	FTS	22841	00																
22845	COM	22845	00																
22849	FTS	22849	00																
22853	FTS	22853	00																
22857	FTS	22857	00																
22861	COM	22861	00																
22865	FTS	22865	00																
22869	FTS	22869	00																

L0s Exit Via Fast
Training Sequence

LTSSM Debug

asicl

12469	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12469	4A
12473	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12473	4A
12477	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12477	4A
12481	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12481	4A
12485	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12485	4A
12489	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12489	4A
12493	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12493	4A
12497	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12497	4A
12501	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12501	COM
12505	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12505	00
12509	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12509	00
12513	COM	12513	0C														
12517	SKP	12517	86														
12521	SKP	12521	00														
12525	SKP	12525	4A														
12529	COM	12529	4A														
12533	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12533	4A
12537	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	12537	4A
12541	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	12541	4A
12545	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	12545	4A
12549	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12549	4A
12553	4A	12553	4A														
12557	4A	12557	4A														
12561	4A	12561	4A														
12565	4A	12565	COM														
12569	4A	12569	00														
12573	4A	12573	00														
12577	4A	12577	0C														
12581	4A	12581	86														
12585	4A	12585	00														
12589	4A	12589	4A														
12593	COM	12593	4A														
12597	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12597	4A
12601	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	12601	4A
12605	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	12605	4A
12609	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	12609	4A
12613	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12613	4A
12617	4A	12617	4A														
12621	4A	12621	4A														
12625	4A	12625	4A														

LTSSM Debug

33403313	08	00	06	08	00	00	06	00	33403687	19	29	51	01	31	39	01	31
33404313	37	30	2C	37	30	28	39	37	33404687	2C							
33405313	00	0F	8D	00	0F	17	00	08	33405687	03	03	03	03	03	03	03	03
33406313	4A	33406687	0A	0A	0A	0A	0A	8A	0A	0A							
33407313	4A	33407687	4A														
33408313	4A	33408687	4A														
33409313	4A	33409687	4A														
33410313	4A	33410687	4A														
33411313	4A	4A	4A	4A	4A	4A	4A	4A	4A								
-----	S01	33411687	4A														
33412313	TS1	33412687	4A														
-----	-----	-----	-----	-----	-----	-----	-----	-----	S01	S01	S01	S01	S01	S01	S01	S01	S01
33413313	0D	33413687	TS1														
33414313	07	06	05	04	03	02	01	00	33414687	0D							
33415313	FF	33415687	07	06	05	04	03	02	01	00							
33416313	0E	33416687	9E														
33417313	00	00	00	00	00	00	00	00	33417687	0E							
33418313	30	00	38	30	00	50	28	18	33418687	00	00	00	00	00	00	00	00
33419313	08	00	06	08	00	00	06	00	33419687	19	29	51	01	31	39	01	31
33420313	37	30	2C	37	30	28	39	37	33420687	2C							
33421313	00	0F	8D	00	0F	17	00	08	33421687	03	03	03	03	03	03	03	03
33422313	4A	33422687	0A	0A	0A	0A	0A	8A	0A	0A							
33423313	4A	33423687	4A														
33424313	4A	33424687	4A														
33425313	4A	33425687	4A														
33426313	4A	33426687	4A														
33427313	4A	4A	4A	4A	4A	4A	4A	4A	4A								
-----	S01	33427687	4A														
33428313	TS1	33428687	4A														
33429313	0D	33429687	S01														
33430313	07	06	05	04	03	02	01	00	33430687	AA							
33431313	FF	33431687	AA														
33432313	0E	33432687	AA														
33433313	00	00	00	00	00	00	00	00	33433687	AA							
33434313	30	00	38	30	00	50	28	18	33434687	AA							
33435313	08	00	06	08	00	00	06	00	33435687	AA							
33436313	37	30	2C	37	30	28	39	37	33436687	AA							
33437313	00	0F	8D	00	0F	17	00	08	33437687	AA							
33438313	4A	33438687	E1														
33439313	4A	33439687	7C	81	7D	BF	42	B7	75	89							
33440313	4A	33440687	54	E7	B3	43	F0	56	A6	F2							
33441313	4A	33441687	DC	F4	28	DB	F3	60	93	4F							
-----	-----	-----	-----	-----	-----	-----	-----	-----	S01	S01	S01	S01	S01	S01	S01	S01	S01
33442313	4A	33442687	TS1														
33443313	4A	33443687	0D														
33444313	S01	33444687	07	06	05	04	03	02	01	00							
33445313	TS1	33445687	9E														
33446313	0D	33446687	0E														
33447313	07	06	05	04	03	02	01	00	33447687	00	00	00	00	00	00	00	00
33448313	FF	33448687	19	29	51	01	31	39	01	31							
33449313	0F	33449687	2C														

asiclab

Gen 7.0 Developments

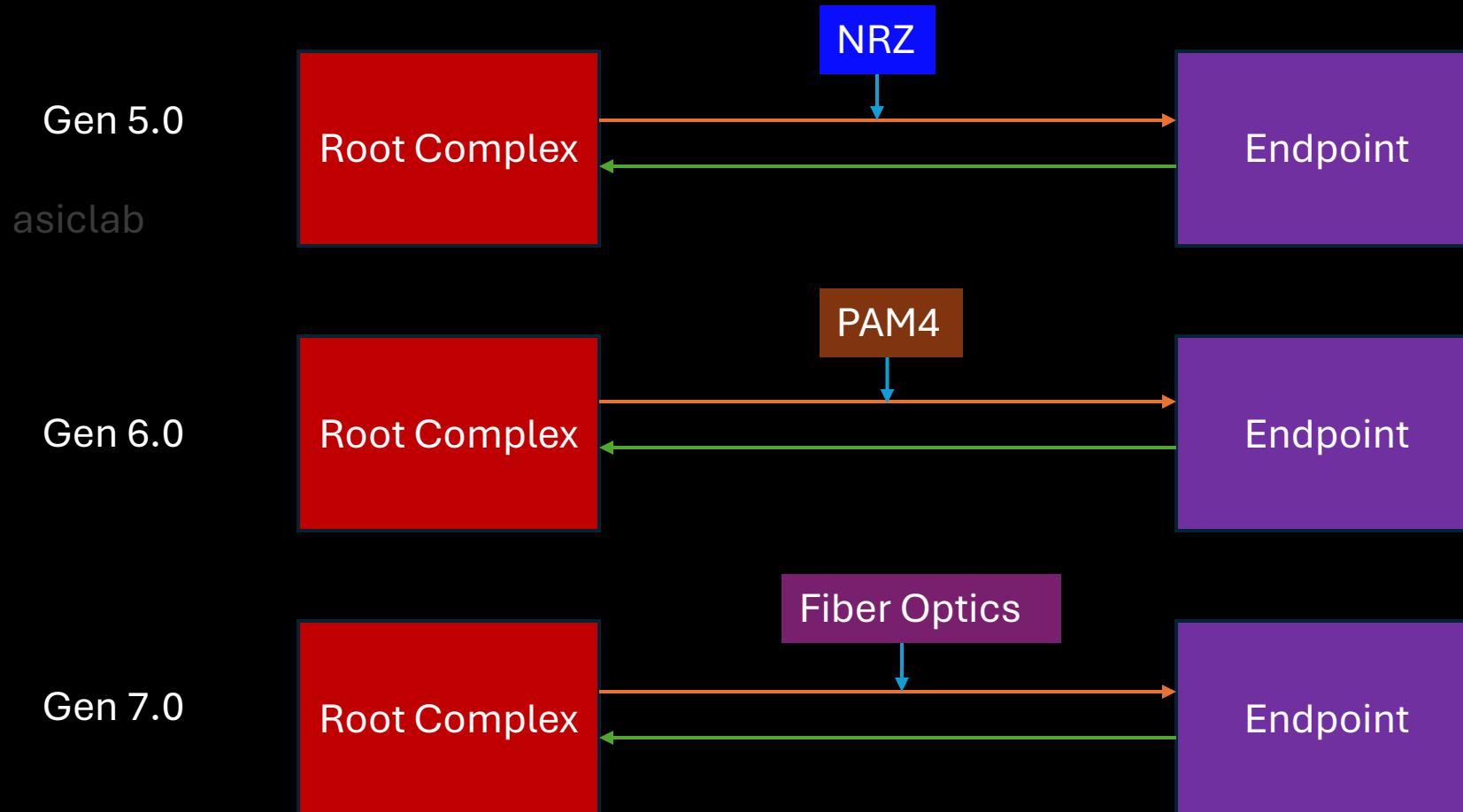
asiclab

Do Not Distribute

© Copyright protected 2024

asiclab
Learn. Thrive

PCIe Gen 5.0, 6.0 Gen 7.0 Developments



PCIe Gen 7.0 – LDO (Linear Driver Optics)

asiclab



Do Not Distribute

© Copyright protected 2024

asiclab
Learn. Thrive

PCIe Gen 7.0 – Eye Measurements

asiclab

