جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY

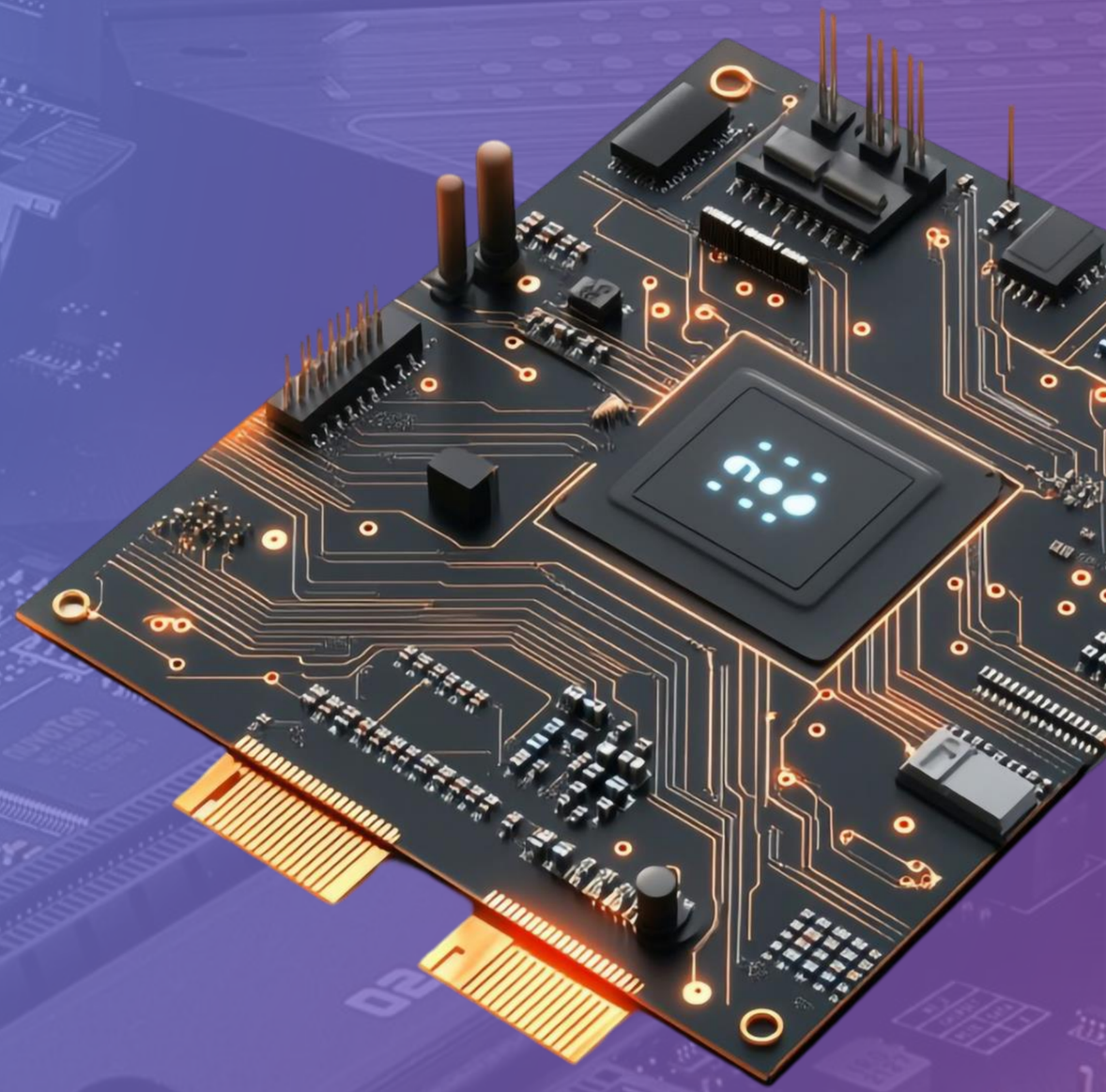
Functional Verification of PCIe 5.0 MAC Layer

Under the Supervision
of
Dr. Mohamed Elbanna

Content of presentation

01

PCIe introduction



Introduction

PCI Express Overview

- PCI Express (Peripheral Component Interconnect Express), officially abbreviated as PCIe, is a high-speed serial communication standard used to connect hardware components inside computers.
- It is designed to replace older expansion bus standards such as PCI, PCI-X
- Developed and maintained by the PCI-SIG (PCI Special Interest Group)

Advantages

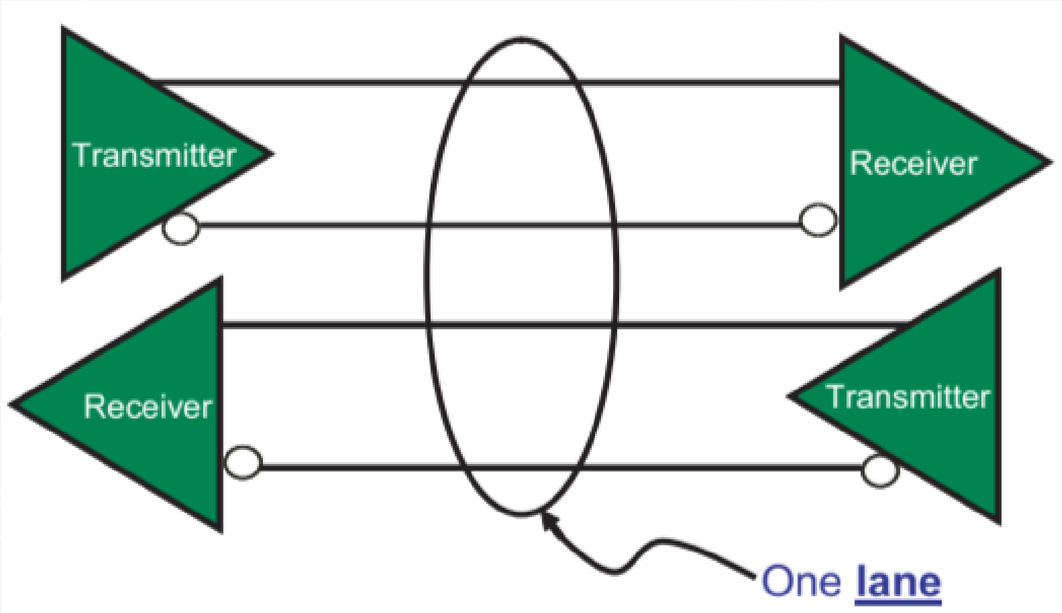
- supports **faster** data transfer.
- uses fewer pins, takes up less space
- **hot swapping**
- **better error detection**.
- supports **I/O virtualization** (IOV)
- **Backward Compatibility**

PCI Express Application

- **Graphics Cards**
  - Nivedia Geforce
  - AMD Radeon
- **Virtual Reality** (VR) and **Augmented Reality** (AR)
- **Networking**
  - Network Interface Cards (NICS)
  - Wi-Fi Adapters
- **High-Performance Computing** (HPC) and **Data Centers**

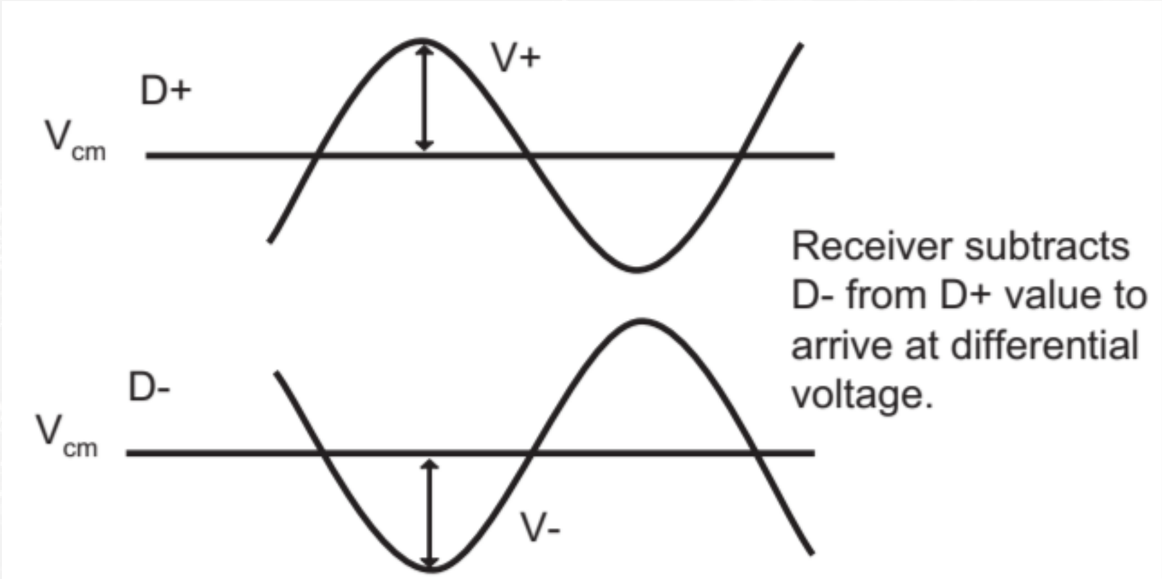
PCI Express Connection Type

- PCI Express is considered a **dual simplex connection**; each interface has a simplex transmit path and a simplex receive path
- The term for this path between the devices is a **Link**, and is made up of one or more transmit and receive pairs. One such pair is called a **Lane**, and the spec allows a Link to be made up **1, 2, 4, 8, 12, 16, or 32 Lanes**. The number of lanes is called the **Link Width**



Differential Signaling

Differential signaling is a method of transmitting data using two complementary signals: a positive version (D+) and a negative version (D-).



Advantages

- **Improved Noise Immunity**
- **Reduced Signal Voltage**
  - reducing power consumption
  - enabling higher frequencies

Disadvantages

- **Increased Pin Count**

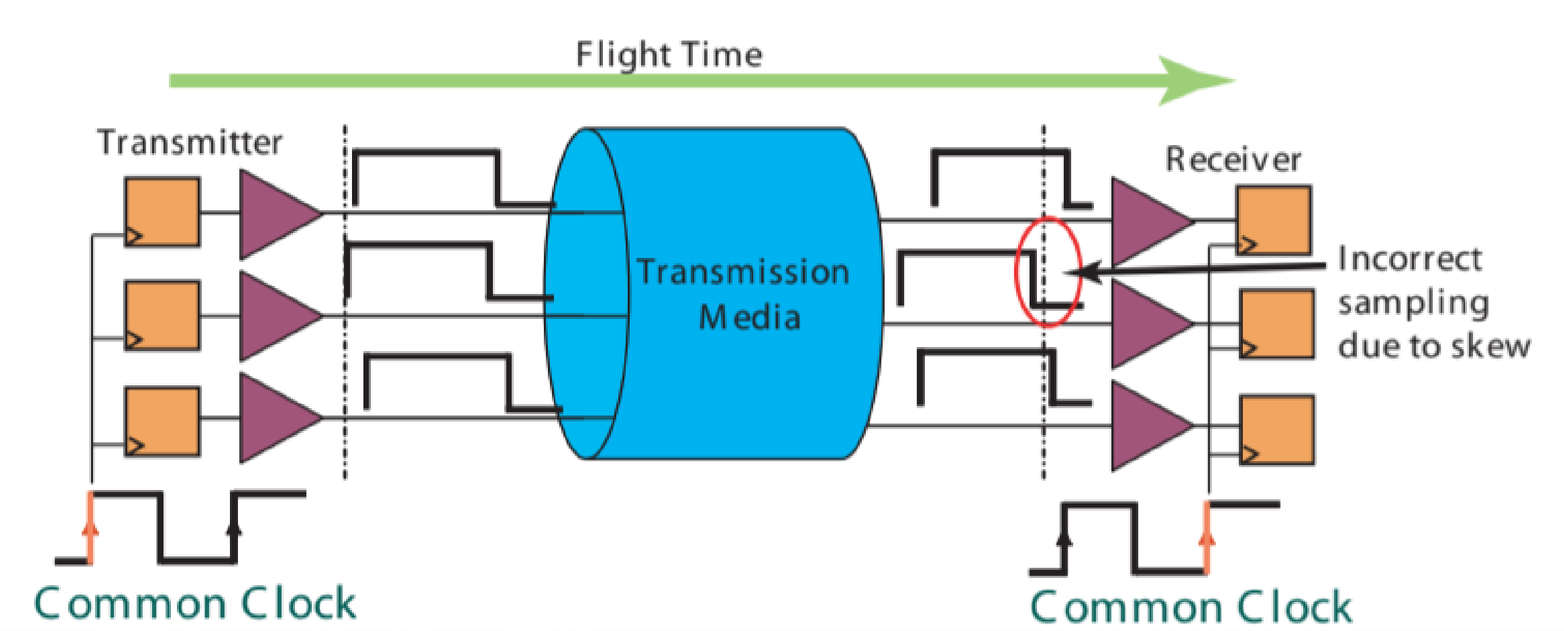
Introduction

Parallel buses Issues

- Flight Time
- Clock Skew
- Signal Skew

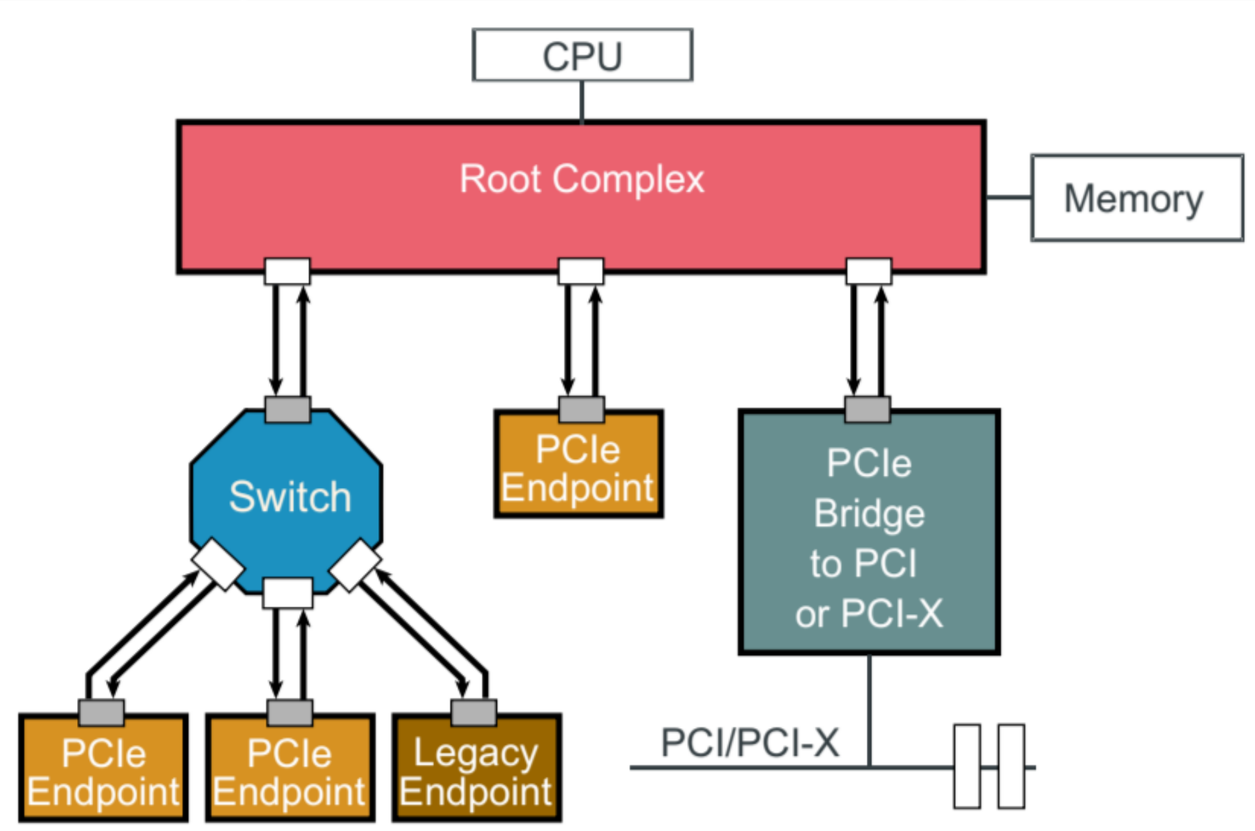
Serial Links Overcome Issues

- Flight Time Nonissue
- No Clock Skew
- No Signal Skew



PCI Express Connection

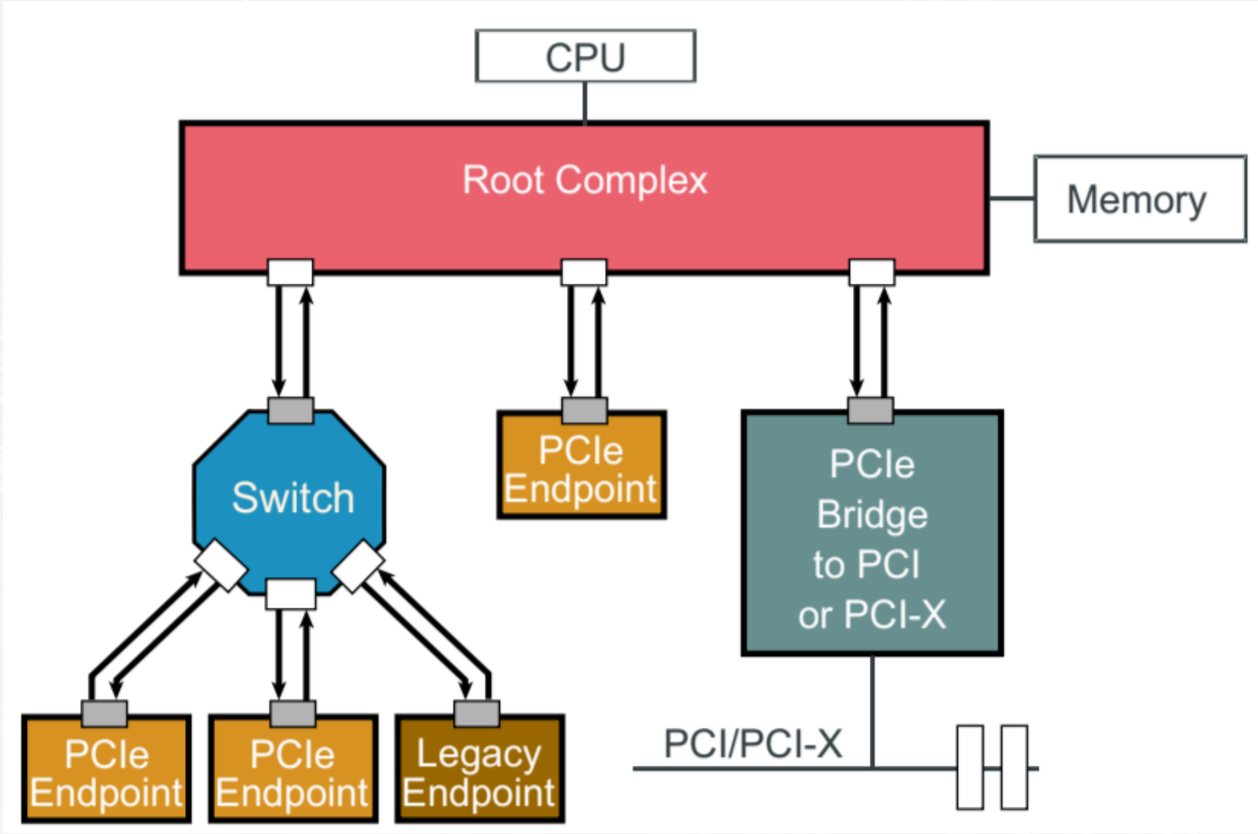
- **Downstream Port**
- **Upstream Port**



PCI Express Topology

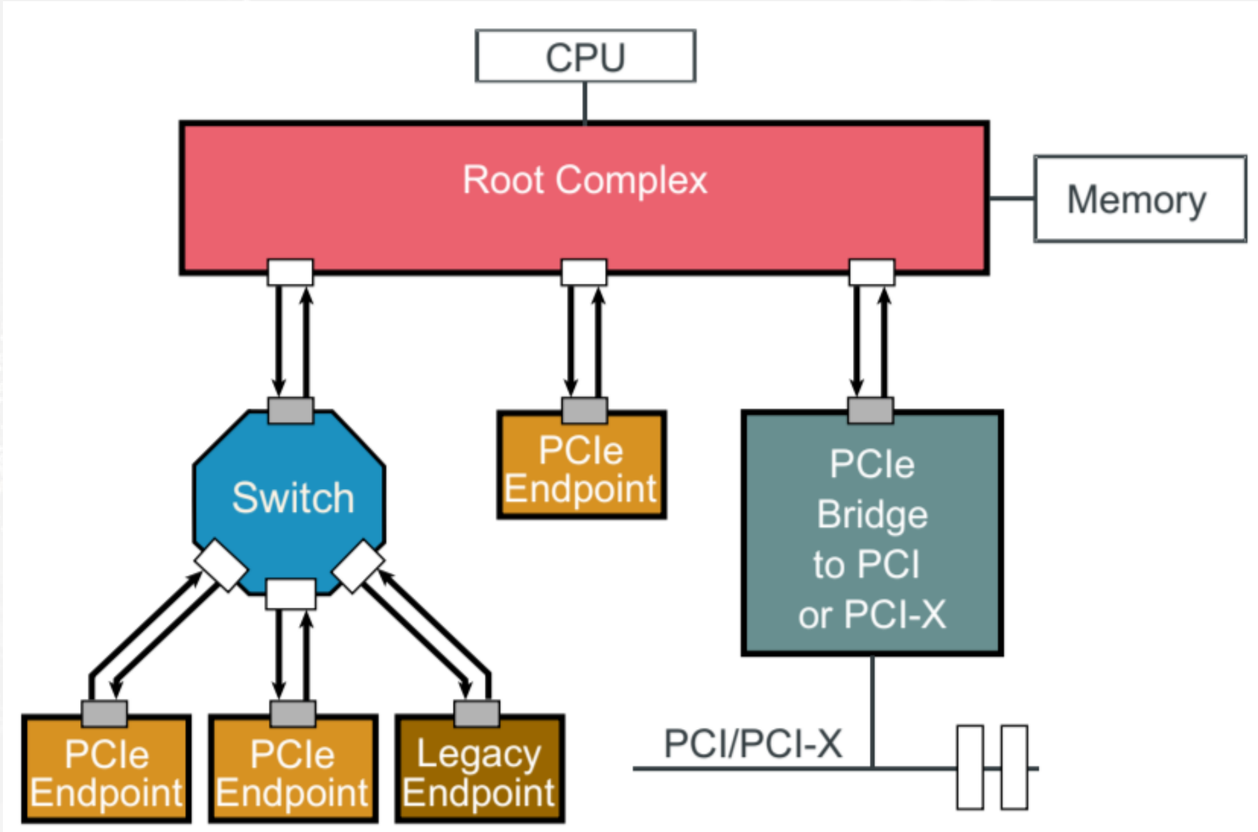
- **CPU:** The CPU is considered the top of the PCIe hierarchy.

- **Root Complex:**
  - The interface between the CPU and the PCIe buses
  - May contain several components (processor interface, DRAM interface, etc.)
  - Acts on behalf of the CPU to communicate with the rest of the system.



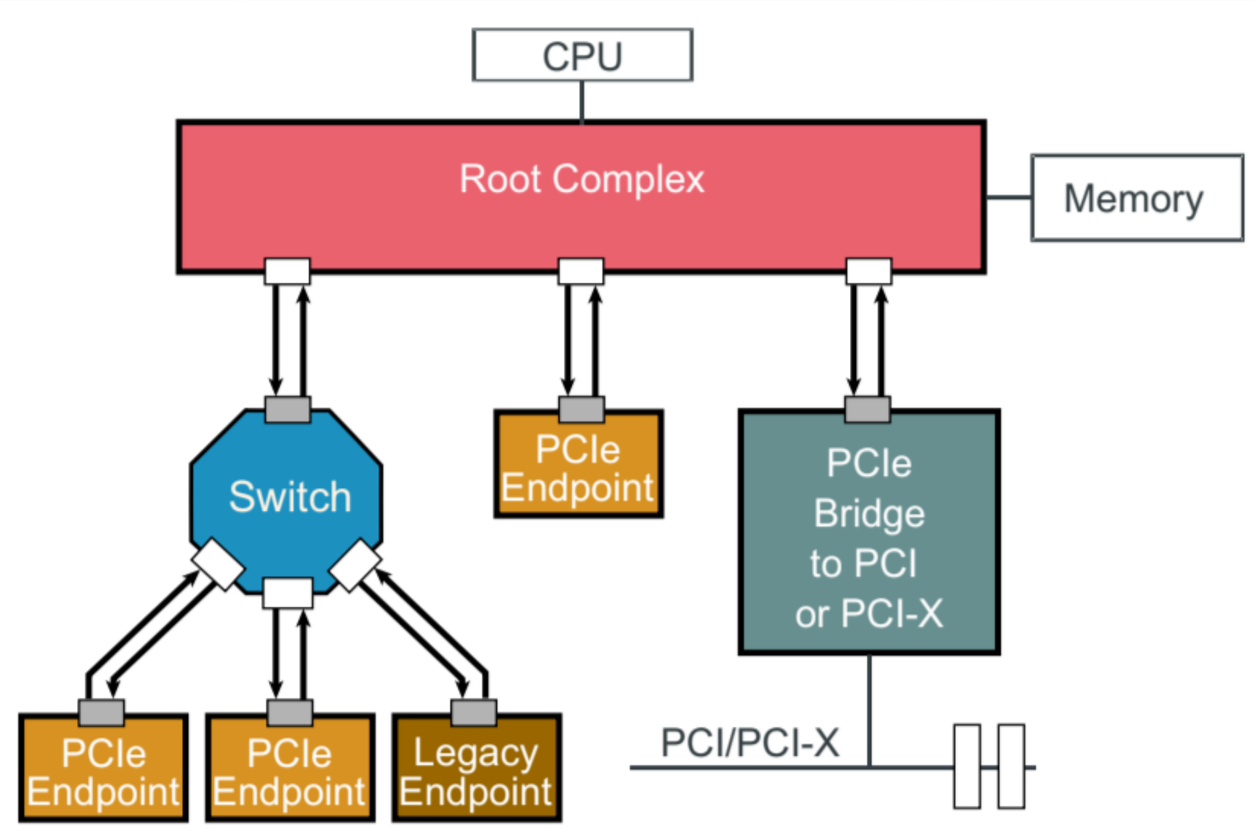
PCI Express Topology

- **Switches:**
  - Provide fanout or aggregation capability, enabling more devices to connect to a single PCIe Port.
  - Act as packet routers, determining the path for a packet based on its address or routing information.
- **Bridges:**
  - Serve as an interface to other buses, such as PCI, PCI-X, or another PCIe bus.
  - The “forward bridge” allows older PCI or PCI-X cards to be used in new PCIe systems.
  - The “reverse bridge” enables new PCIe cards to be used in older PCI systems.



PCI Express Topology

- **EndPoints:**
  - Endpoints are devices in a PCIe topology that are neither Switches nor Bridges.
  - Reside at the bottom of the tree topology branches.
  - Implement a single Upstream Port (facing toward the Root).
- **Legacy PCIe Endpoints:**
  - Devices that were designed for the operation of an older bus like PCI-X but now have a PCIe interface
- **Native PCIe Endpoints:**
  - PCIe devices designed from scratch as opposed to adding a PCIe interface to old PCI device designs.



Introduction

PCI Express Link Performance

Version	Year	Line Code		Transfer rate	Throughput (GB/s)					Frequency
					x1	x2	x4	x8	x16	
1.0	2003	NRZ	8b/10b	2.5 GT/s	0.25	0.5	1	2	4	2.5 GHz
2.0	2007			5 GT/s	0.5	1	2	4	8	5 GHz
3.0	2010		128b/130b	8 GT/s	0.985	1.969	3.938	7.877	15.754	8 GHz
4.0	2017			16 GT/s	1.969	3.938	7.877	15.754	31.508	16 GHz
5.0	2019			32 GT/s	3.938	7.877	15.754	31.508	63.015	32 GHz
6.0	2022	PAM-4 FEC	242B/256B FLIT	64 GT/s	7.563	15.125	30.25	60.5	121	32 GHz
7.0	2025			128 GT/s	15.125	30.25	60.5	121	242	64 GHz

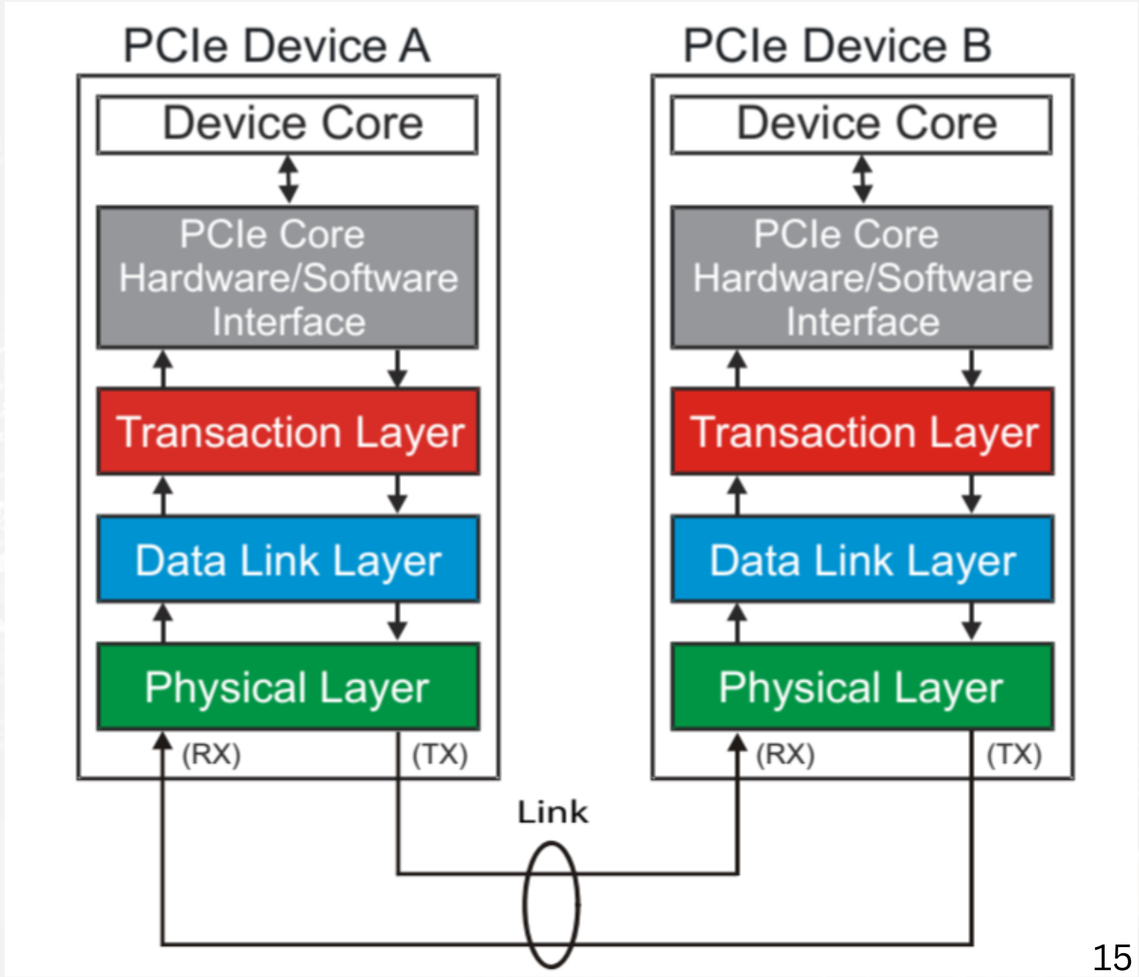
- NRZ: Non-Return to Zero
- PAM: Pulse Amplitude Modulation
- FEC: Forward Error Correction
- FLIT: Flow Control Unit

Throughput Calculation

$$Throughput\,(GB/s) = \frac{GT/s \times Number of Lanes \times Data Bits Per Cycle}{Encoding Overhead \times 8}$$

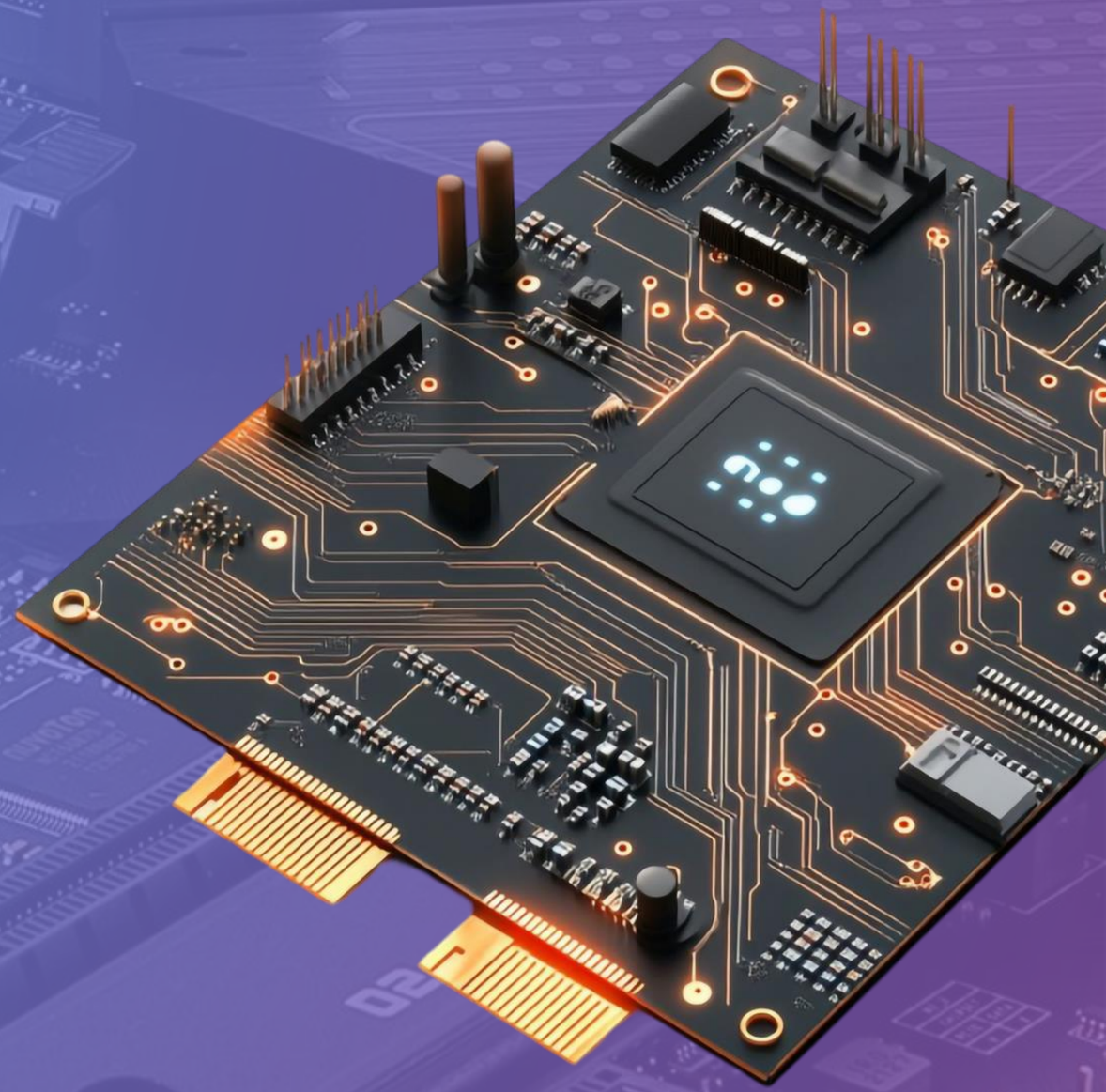
PCI Express Device Layers

- Transaction Layer
- Data Link Layer
- Physical Layer (our focus)



02

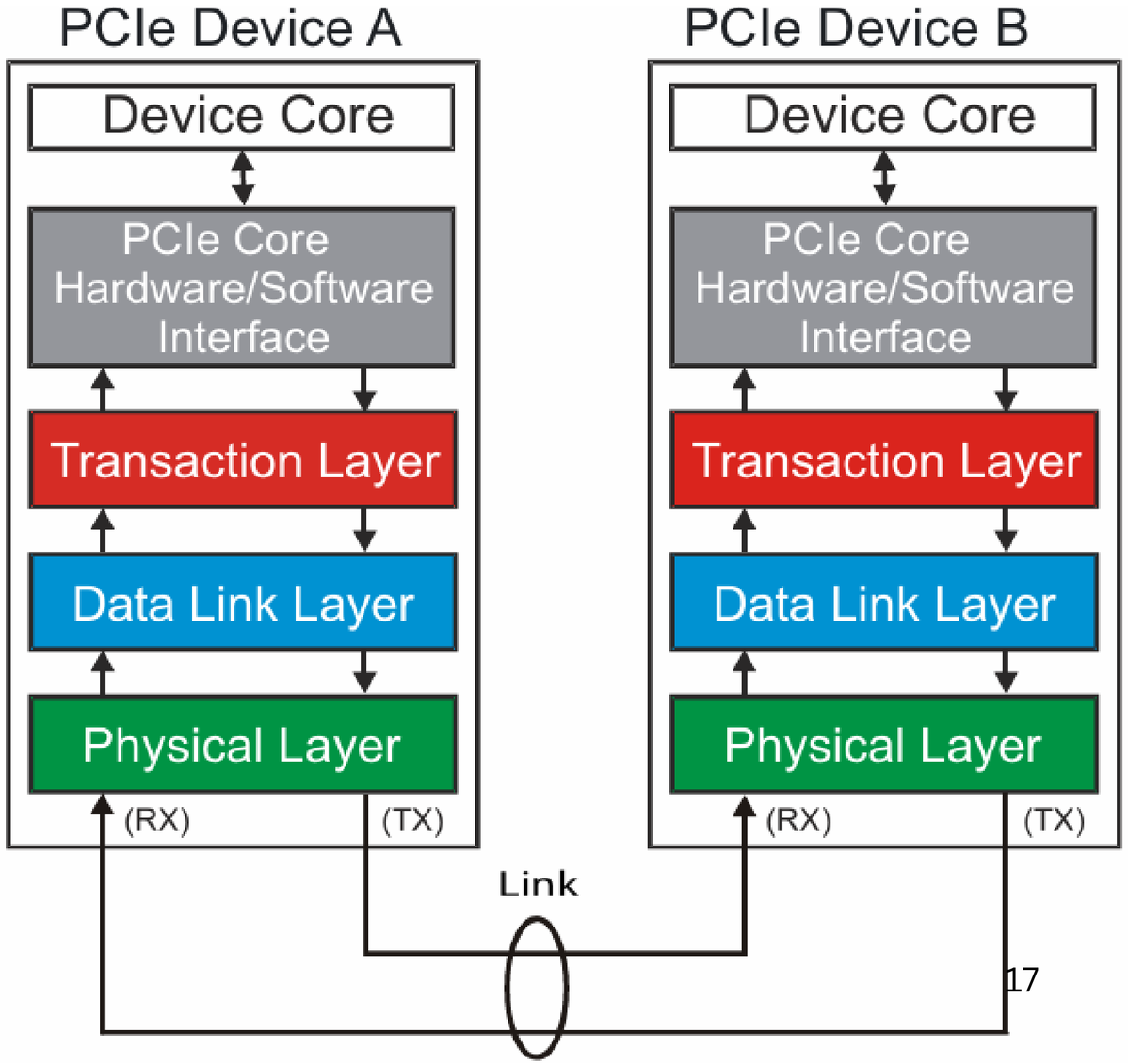
Device Layers Interaction



Device Layers Interaction

Device Layer Interaction

- Each layer on one device communicates with its corresponding layer on the device at the other end of the link. The upper layers (Transaction and Data Link) handle this by formatting data into structured packets that follow known patterns recognizable by the receiving peer layer. These packets pass through intermediate layers during transmission.
- The Physical Layer also communicates with its counterpart on the other device, but unlike the upper layers, it does so through direct electrical signaling rather than a packet-based communication manner.

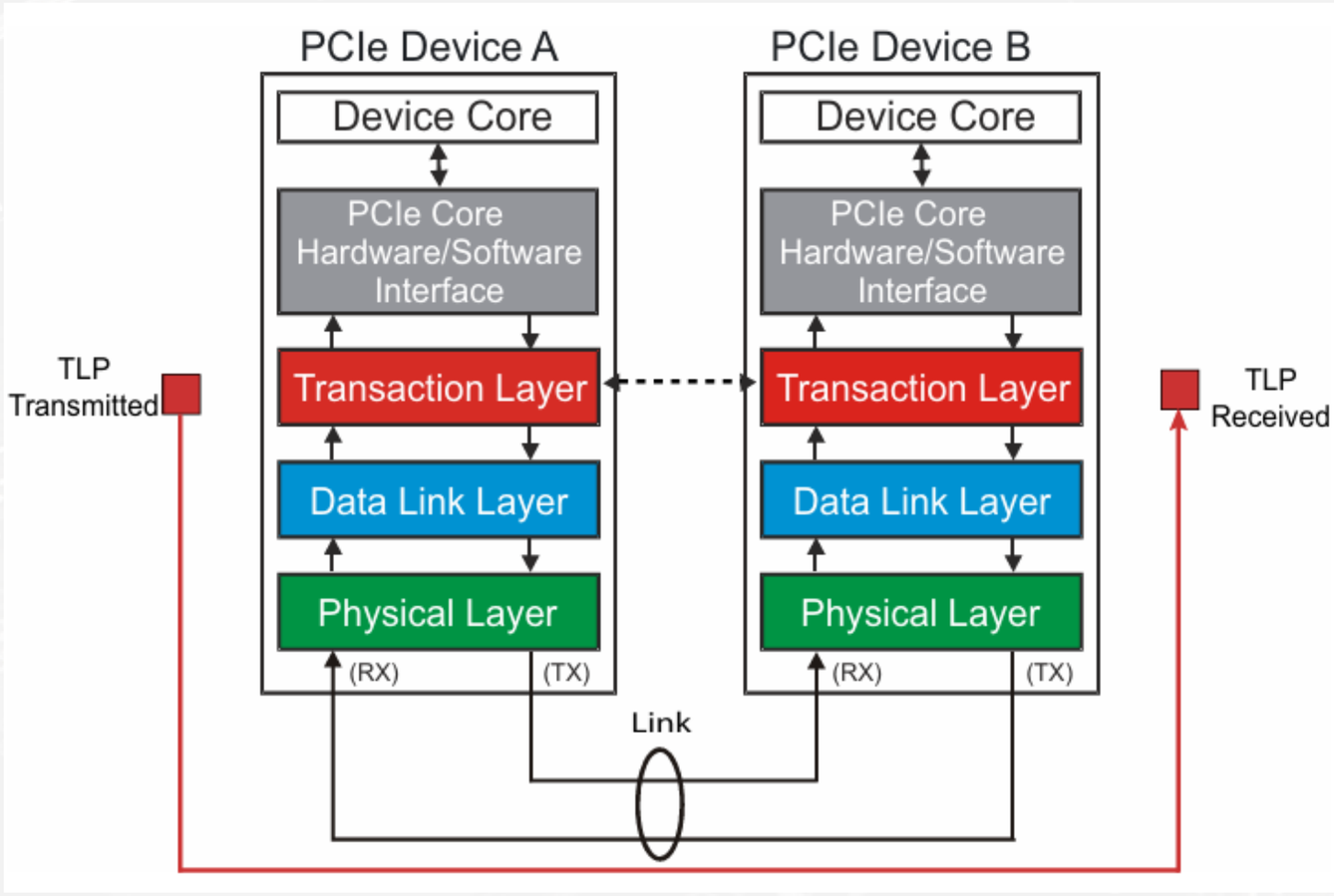


Transaction Layer Interaction in Details

- TLPs originate at the Transaction Layer of a transmitter and terminate at the Transaction Layer of a receiver, as shown in the figure
- Device core sends required information to assemble the core of the TLP

TLP Categories:

- Memory, I/O, and Configuration: Inherited from PCI/PCI-X
- Messages: Introduced in PCIe for signaling between components

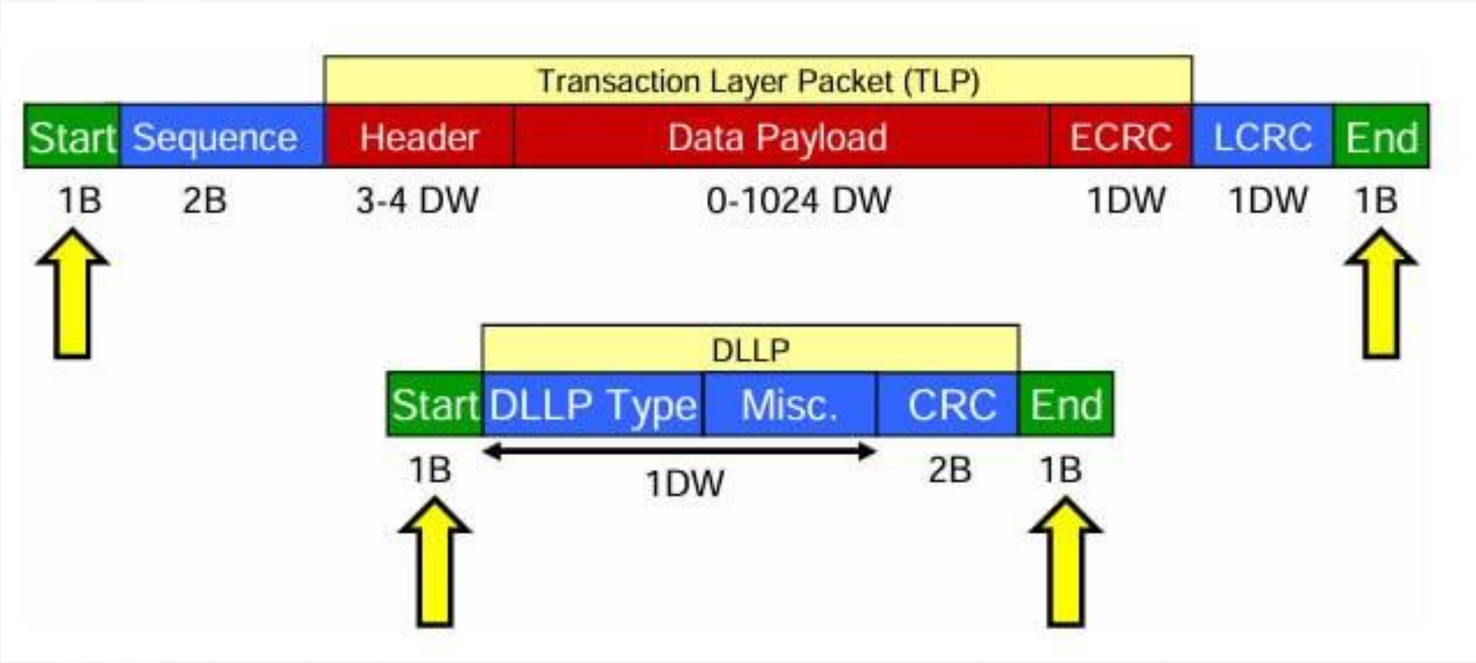


Device Layers Interaction

Transaction Layer Packet Assembly

Each TLP contains a

- **header** (Always present)**:**
  - Routing
  - QoS and ordering
  - Error handling
- **ECRC (End-to-End CRC)** (Optional)**:**
  - 32-bit CRC provides robust error detection, including burst errors.
  - Passed unchanged through service points. Validated at the destination only

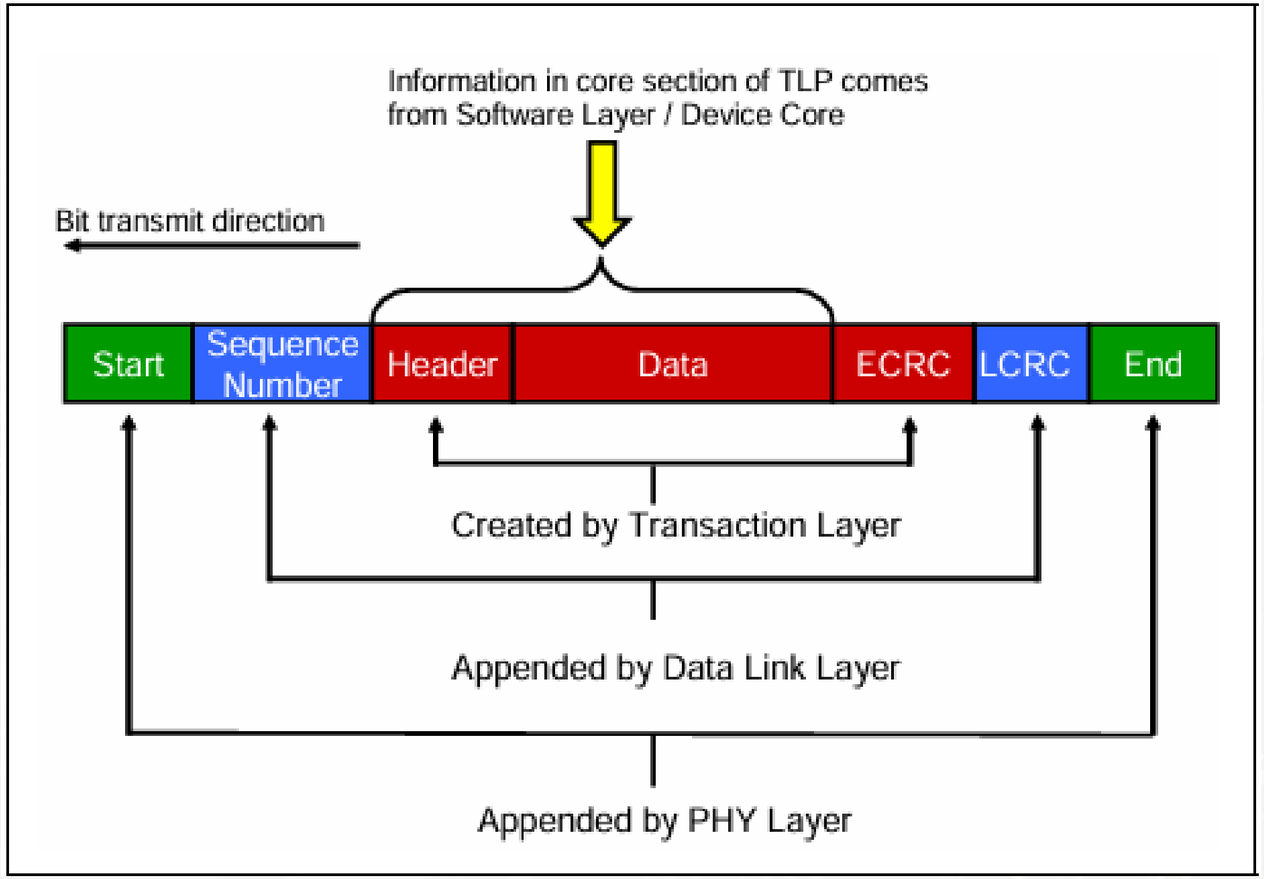


  - **CRC:** stands for Cyclic Redundancy Check (or Code) and is employed by almost all serial architectures for the simple reason that it's simple to implement and provides very robust error detection capability.

Device Layers Interaction

Transaction Layer Packet Assembly in DLL

- Adds a Sequence Number and LCRC field to the packet.
- LCRC (Link CRC) checks for transmission errors between adjacent nodes.
- Why use LCRC and ECRC?
- Receiver reports any error detection to the transmitter.



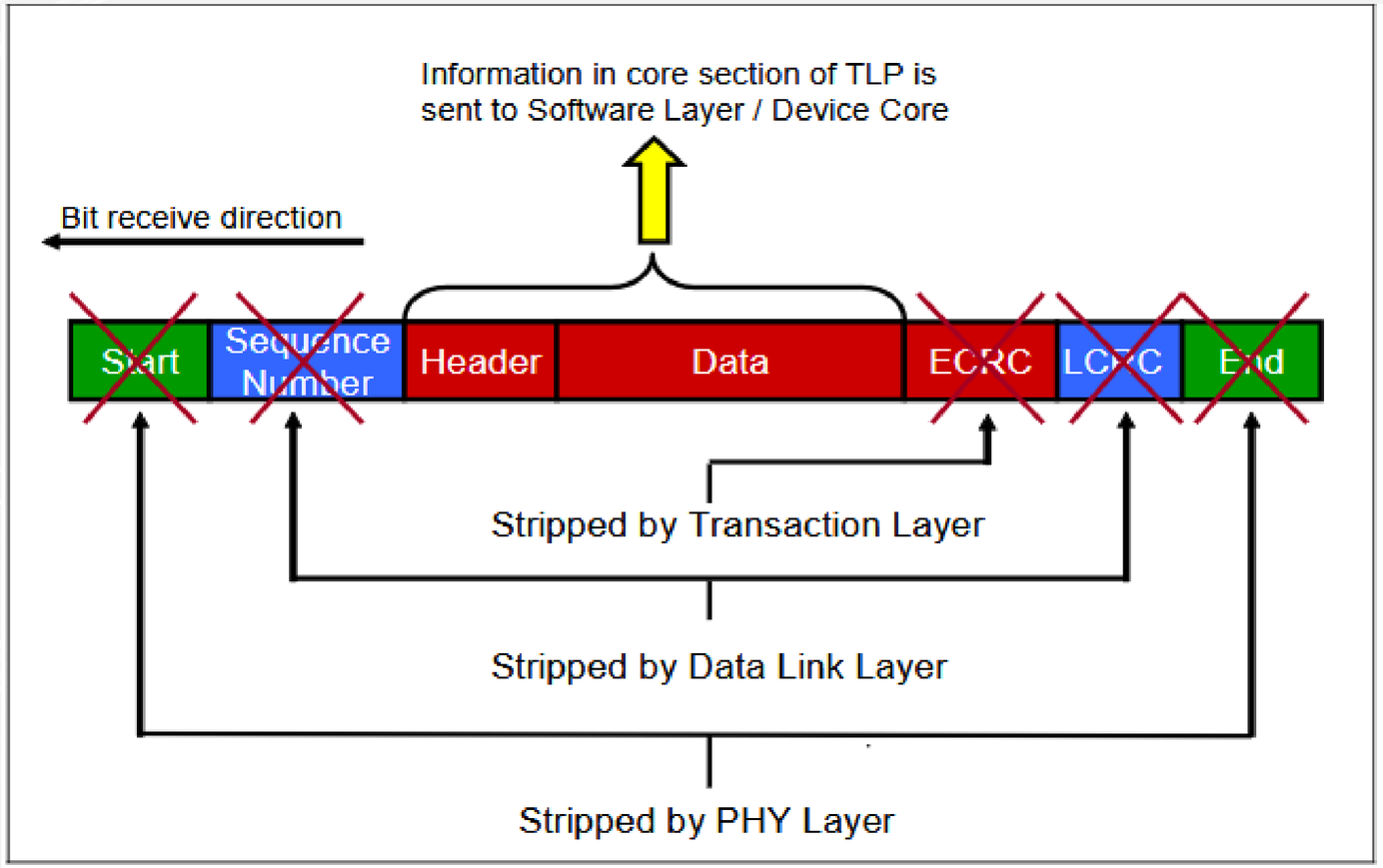
PHY Layer Assembly

- Adds framing characters to define packet boundaries.
- Gen1/Gen2: Use control characters at start and end.
- Gen3/Gen4/Gen5: Control characters no longer used.

Transaction Layer Packet Disassembly

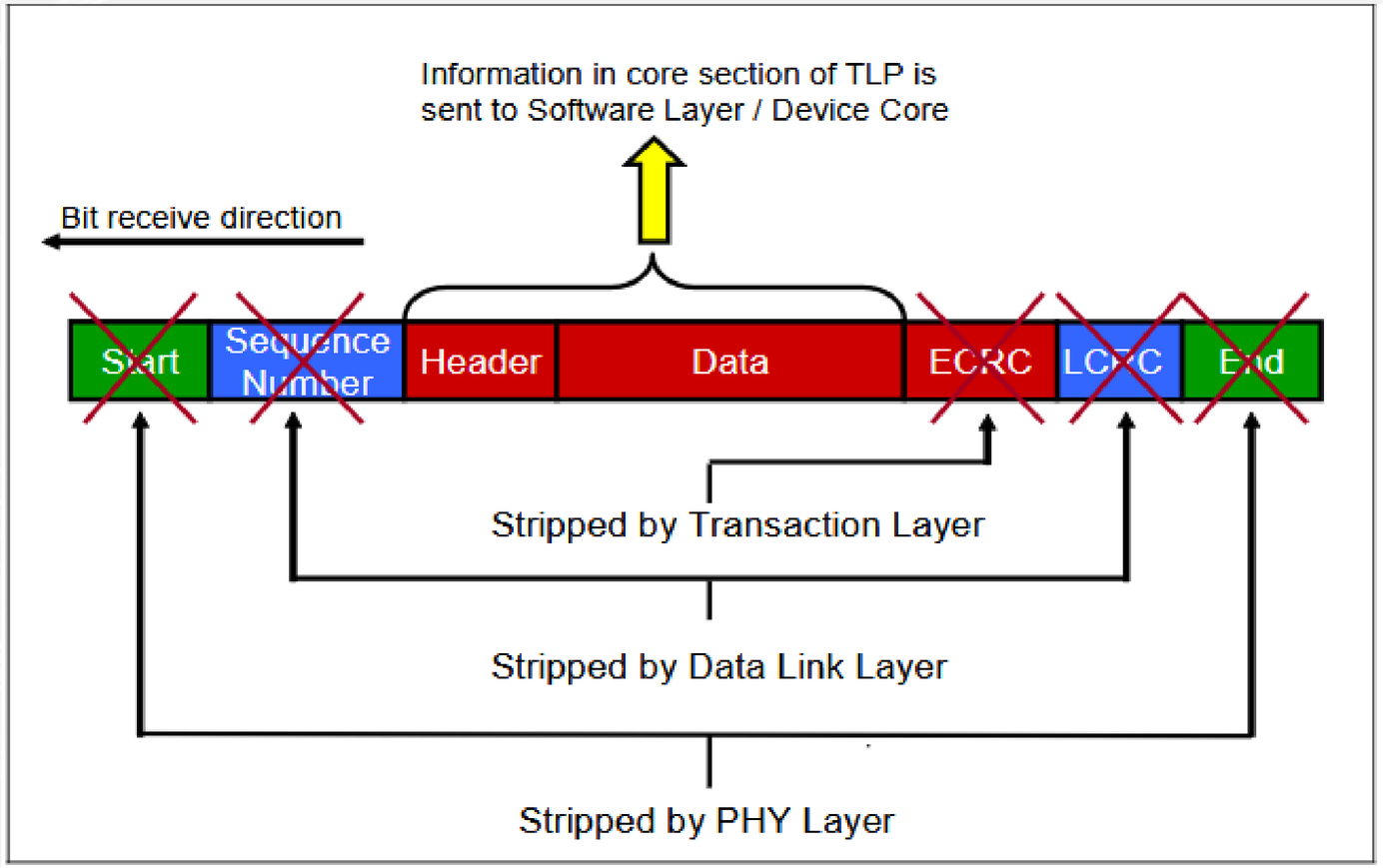
Upon receiving a TLP bitstream, the receiver begins disassembly to recover the original data as shown in the Figure.

- **Physical Layer:**
  - Verifies the presence of Start and End characters (applicable to Gen1 and Gen2).
  - Removes framing characters and forwards the packet to the Data Link Layer.
- **Data Link Layer:**
  - Checks for LCRC and sequence number errors
  - If valid, it removes these fields and forwards the packet to the Transaction Layer.



Transaction Layer Packet Disassembly

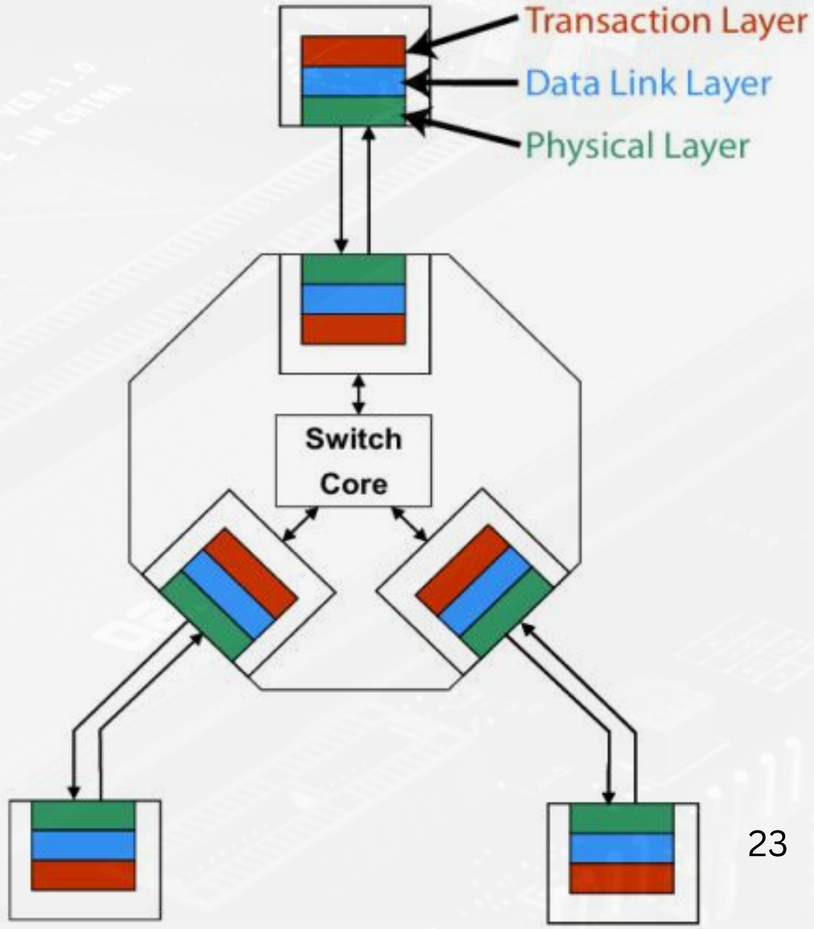
- **Transaction Layer:**
  - **If the receiver is the target device:**
    - Checks the ECRC field for errors.
    - Forwards the validated packet to the core network.
  - **If the receiver is a switch:**
    - Analyzes the TLP header for routing information
    - Forwards the packet to the appropriate port.
    - May check and report ECRC errors but cannot modify the ECRC field.



Device Layers Interaction

Device Layer Interaction Overview (Sender)

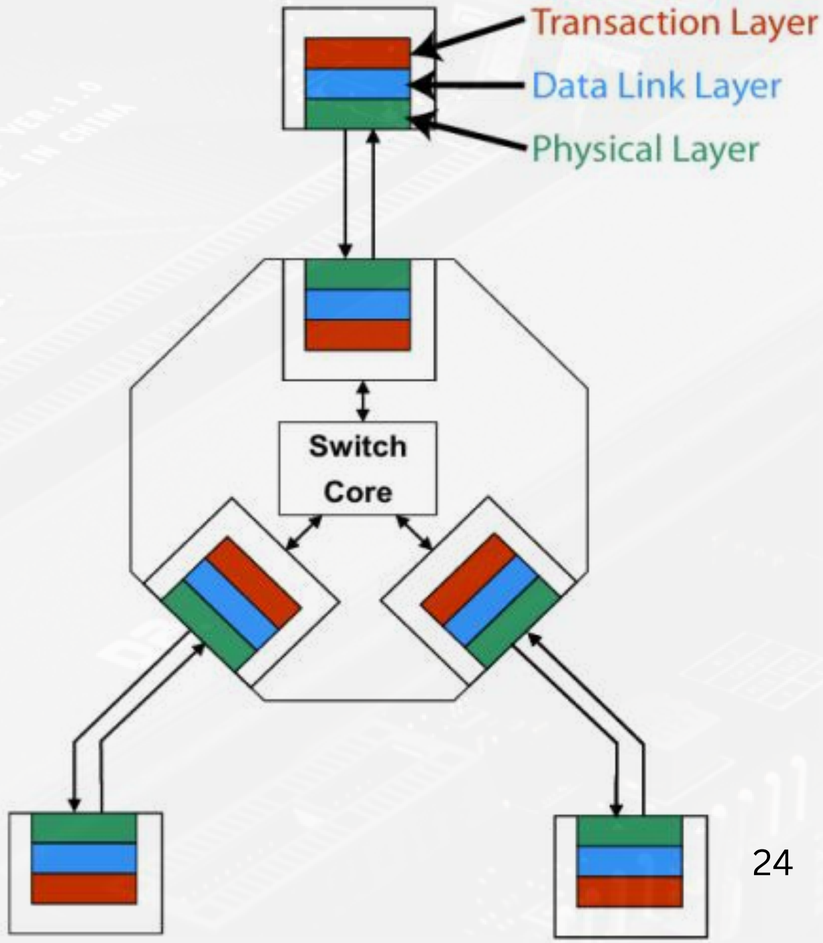
- **Transaction Layer (Sender):**
  - Constructs Transaction Layer Packet (TLP) using command, address, and metadata.
  - Stores TLP in Virtual Channel buffer.
- **Data Link Layer (Sender):**
  - Appends sequence number and CRC.
  - Retains a copy for retransmission if errors occur.
- **Physical Layer (Sender):**
  - Encodes and transmits TLP over differential signaling lanes.



Device Layers Interaction

Device Layer Interaction Overview (Receiver)

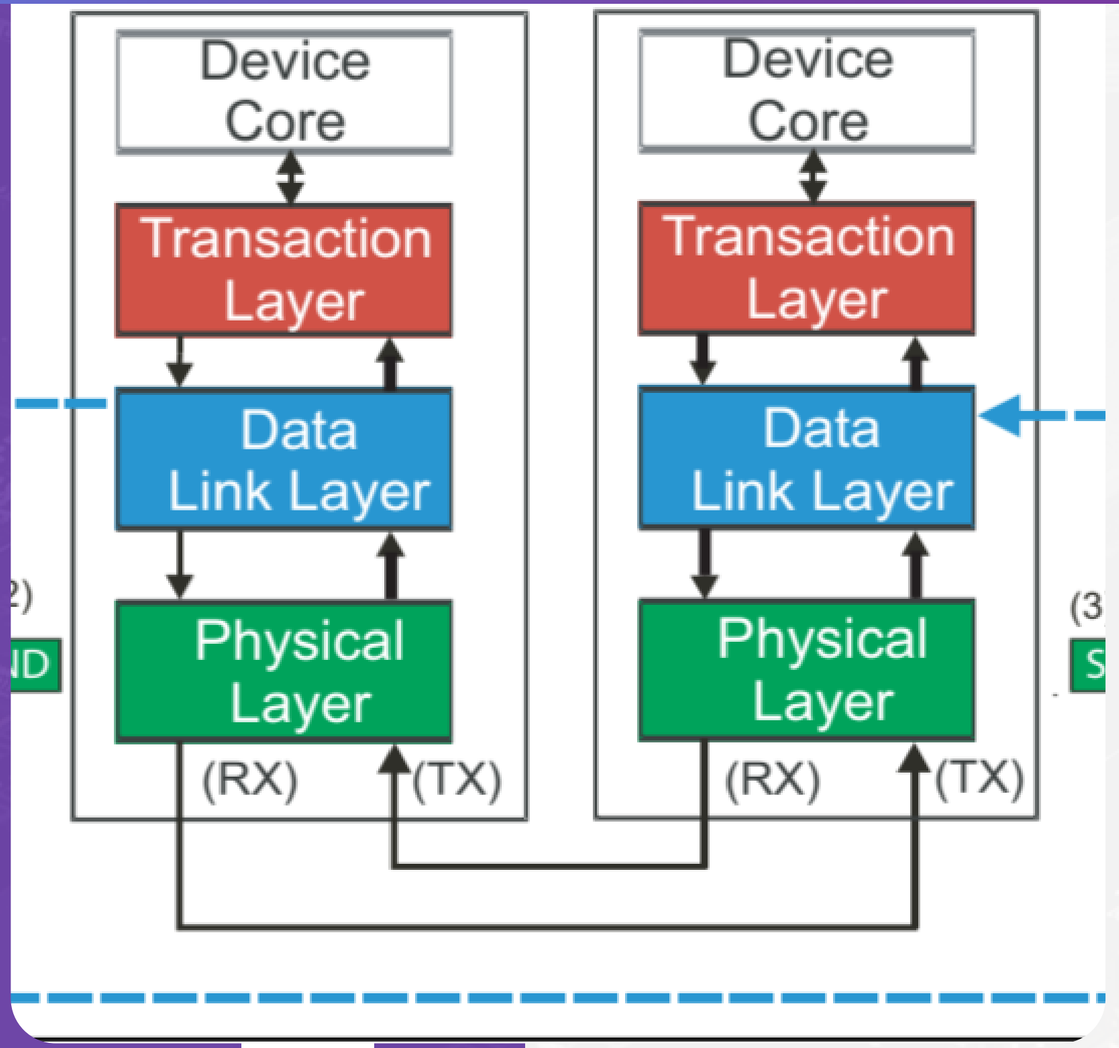
- **Physical Layer (Receiver):**
  - Decodes bitstream and performs initial error checks.
- **Data Link Layer (Receiver):**
  - Validates sequence and CRC.
  - Forwards verified packets to the Transaction Layer.
- **Transaction Layer (Receiver):**
  - Buffers, checks, and disassembles TLP.
  - Passes data and commands to the device core.



Device Layers Interaction

Data Link Layer in Details

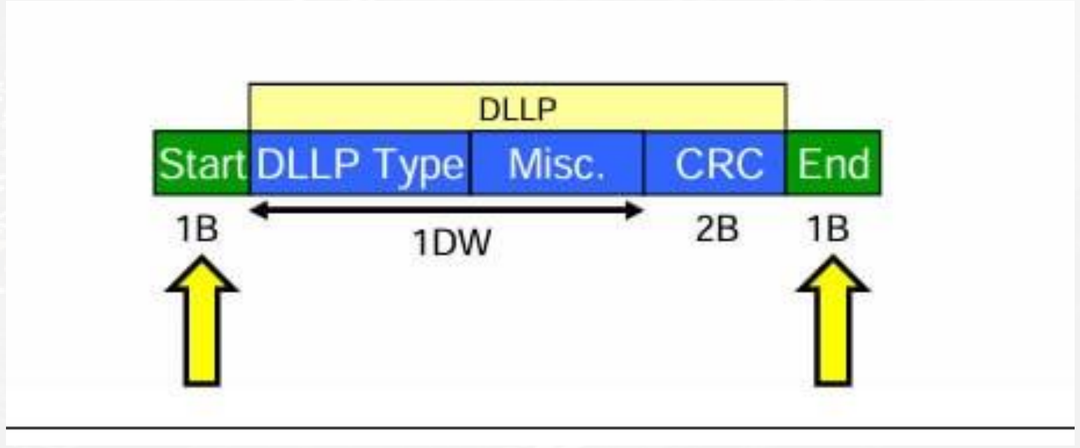
- **DLLP** is an abbreviation for **Data Link Layer Packets**, which are packets transferred between the Data Link Layers of neighboring devices.
- DLLPs are not routed beyond these neighboring devices. Their size is fixed at approximately 8 bytes, which is considered small compared to TLPs.
- This small size is beneficial, as it reduces the overhead of these control packets.



Data Link Layer in Details

DLLP Assembly and Disassembly

- DLLPs are produced by the Data Link Layer of the transmitter.
- Initially, a DLLP consists of two main parts: the **DLLP core** and a **16-bit CRC** used for error checking.
- The packet is then forwarded to the Physical Layer of the transmitter, which adds start and end fields to the packet and transmits it differentially across all available lanes.
- At the receiver side, this process is reversed.

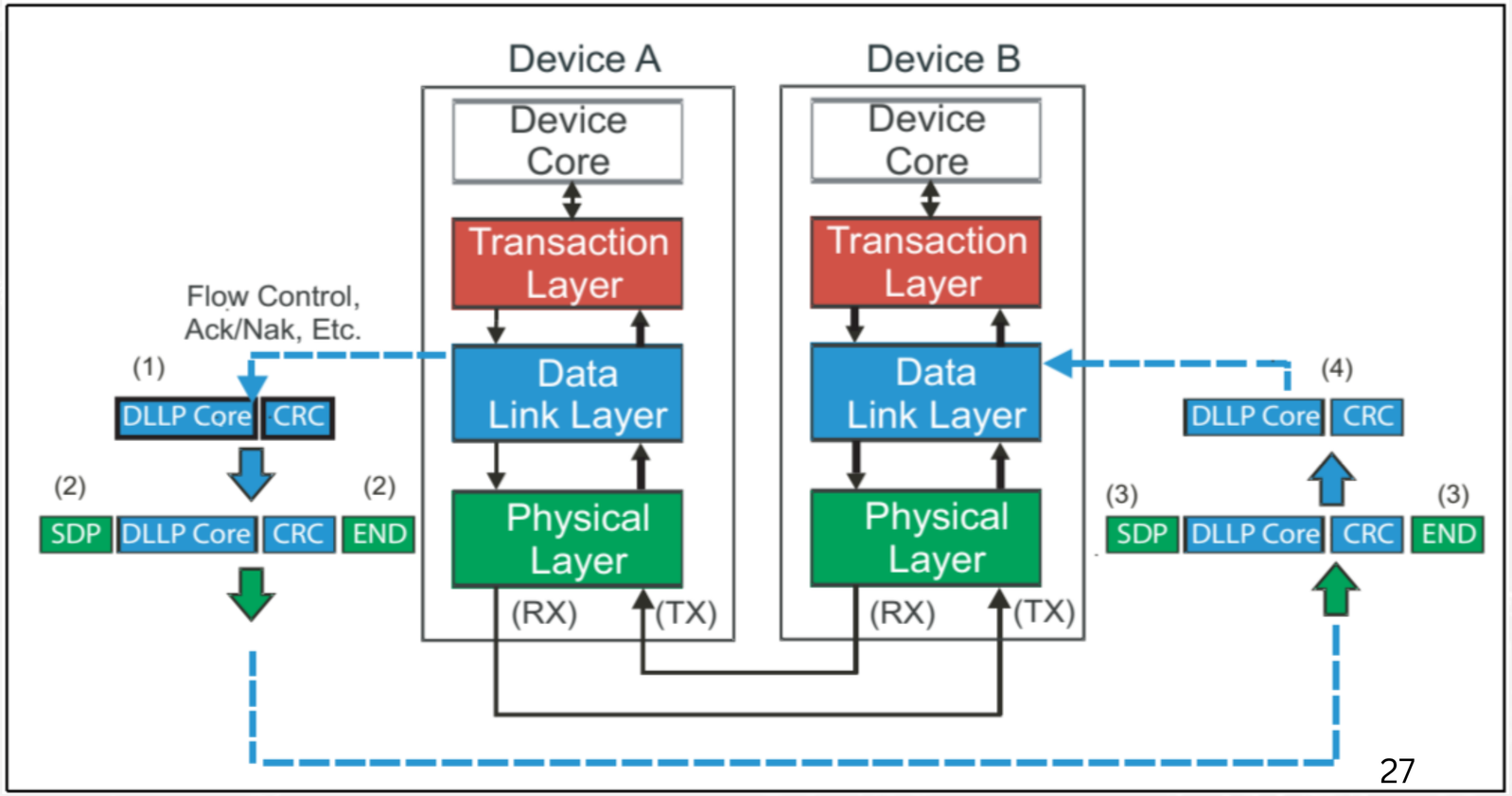


Device Layers Interaction

Data Link Layer Interaction in Details

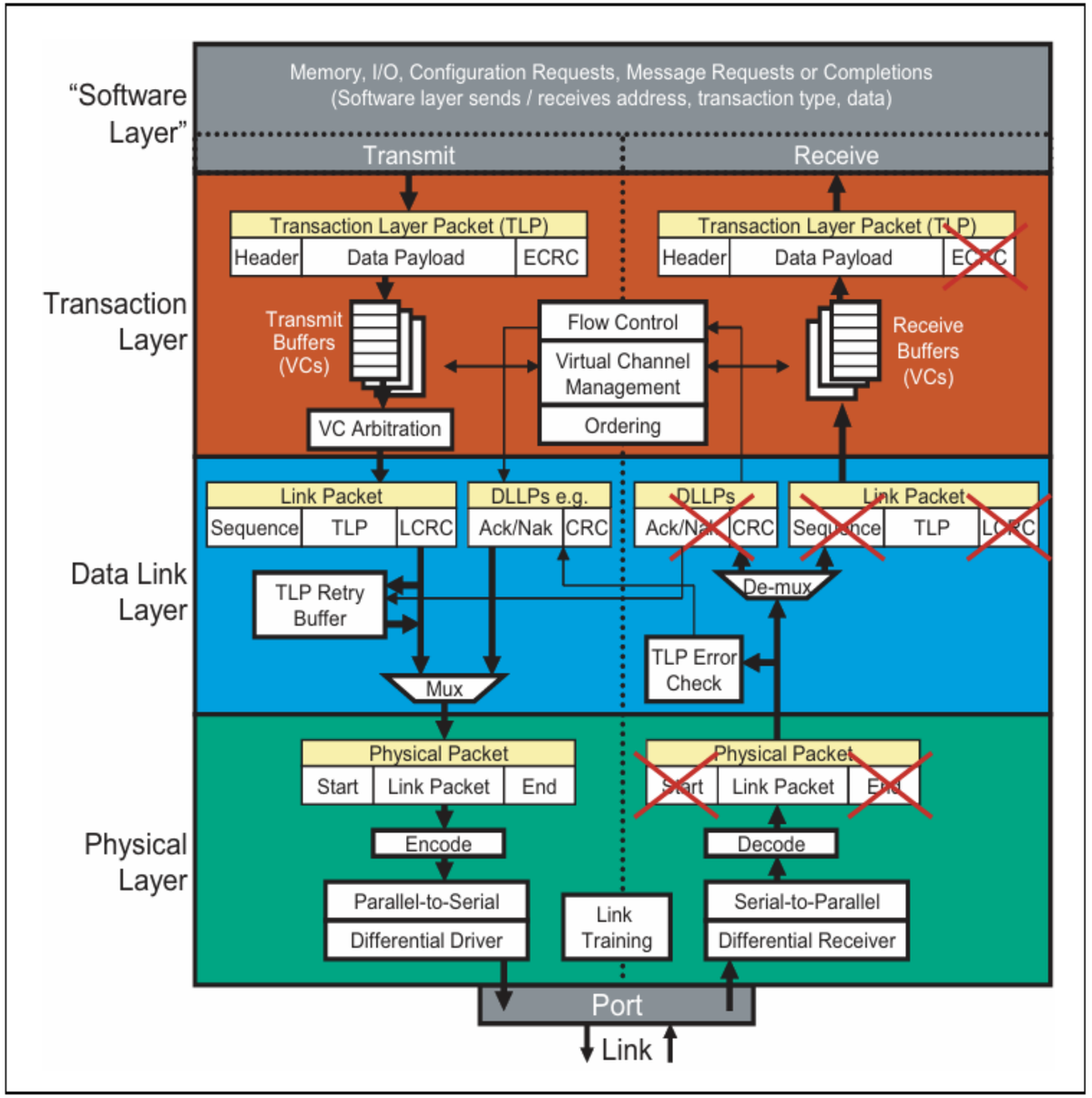
Ack/NAK Protocol

- Essential for error correction and primarily used in a hardware-based automatic retry mechanism. Certain fields are added to the TLP to assist the receiver in performing error checking, such as the LCRC and Sequence Number.
- **How does it work?**



Device Layers Interaction

Device Layers Interaction

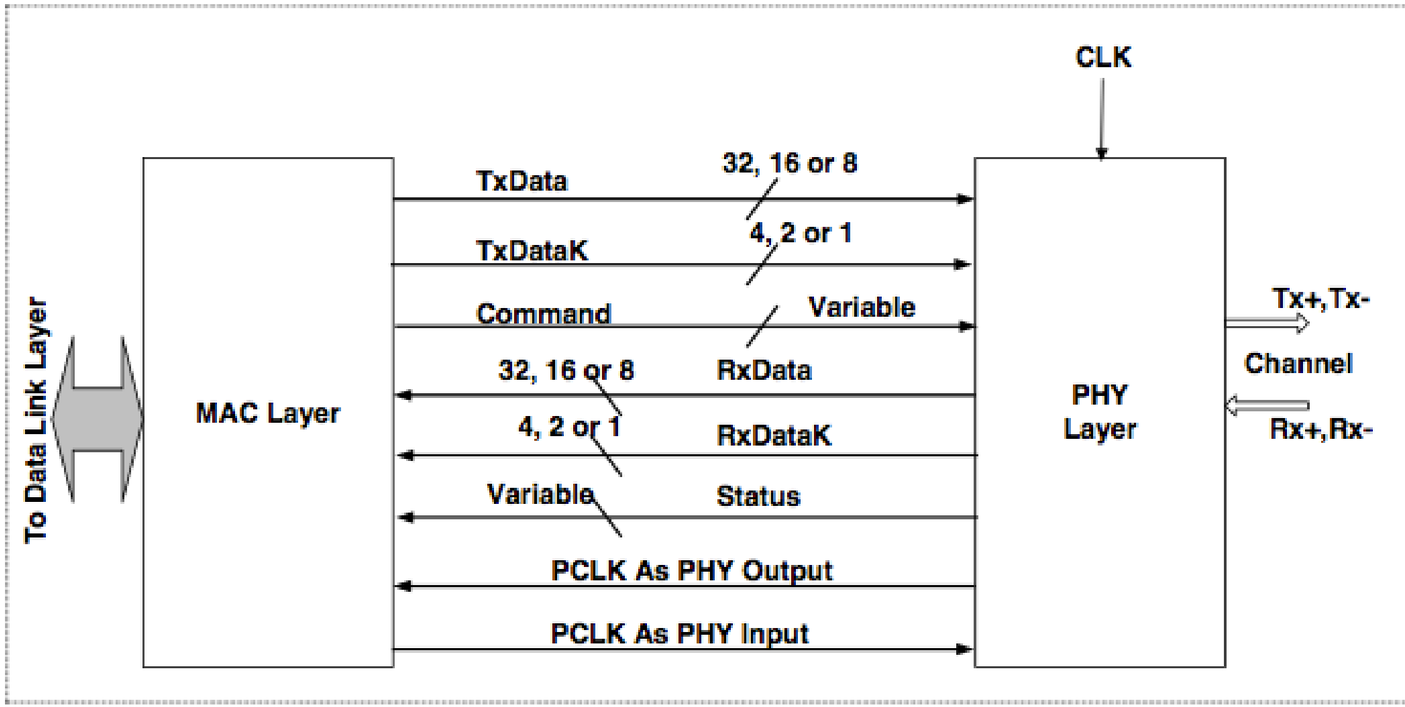


Device Layers Interaction

Physical Layer in Details

The Physical Layer is the lowest hierarchical layer. Both TLP and DLLP packets are forwarded down to it at the transmitter and up from it at the receiver.
This layer is divided into two parts:

- **Logical Part**: which prepares packets for serial transmission and reverses the process for received packets.
- **Electrical Part**: which is the analog interface that connects directly to the Link and contains differential drivers and receivers for each lane.

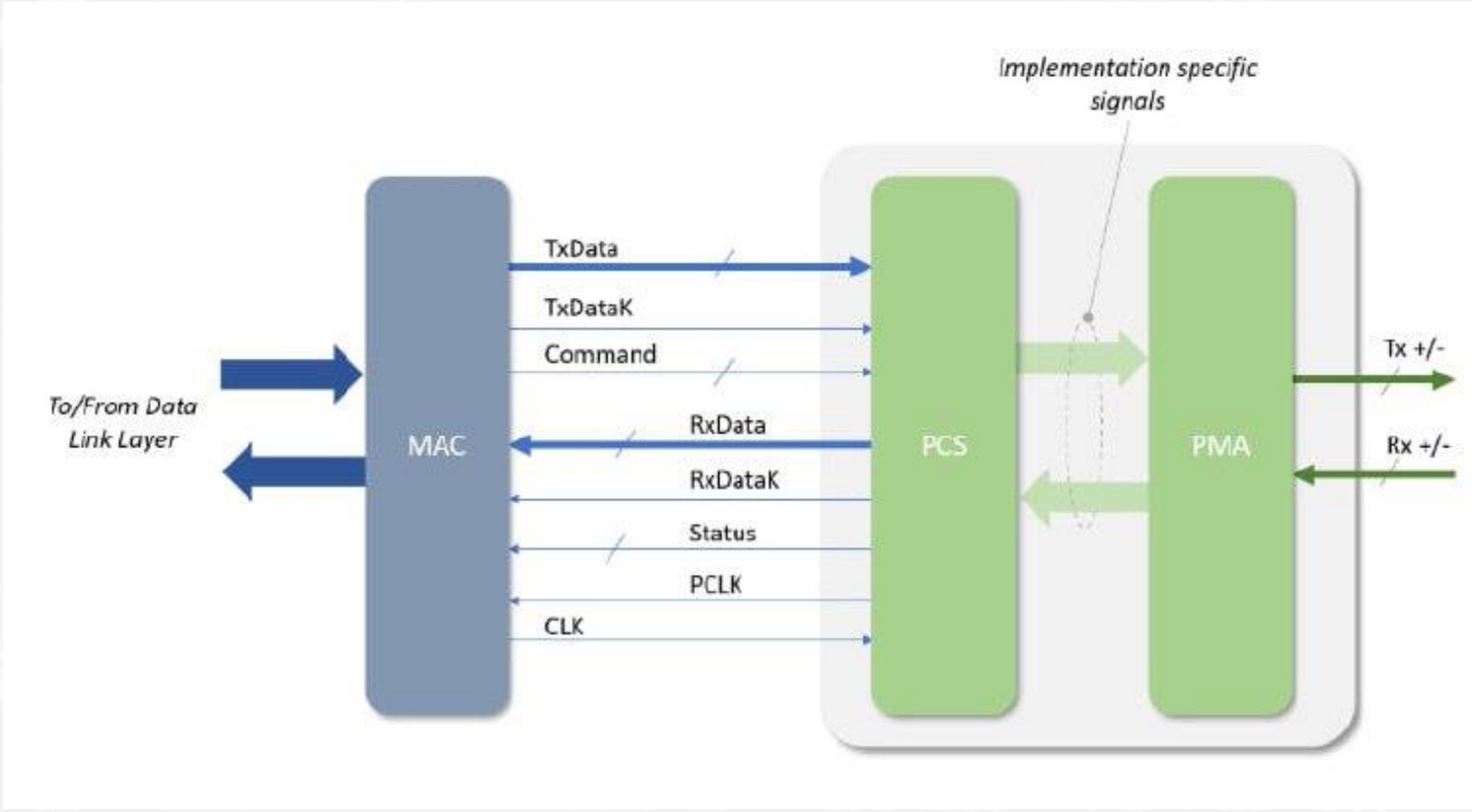


Device Layers Interaction

Physical Layer Overview

PIPE: PHY **I**nterface for the **P**CI **E**xpress (The big picture)

- Media Access Layer (MAC)
  - Handles state machines (especially LTSSM - Link Training and Status State Machine).
  - Performs scrambling/descrambling to reduce EMI.
  - Provides elastic buffers for lane-to-lane deskew and SerDes PIPE.

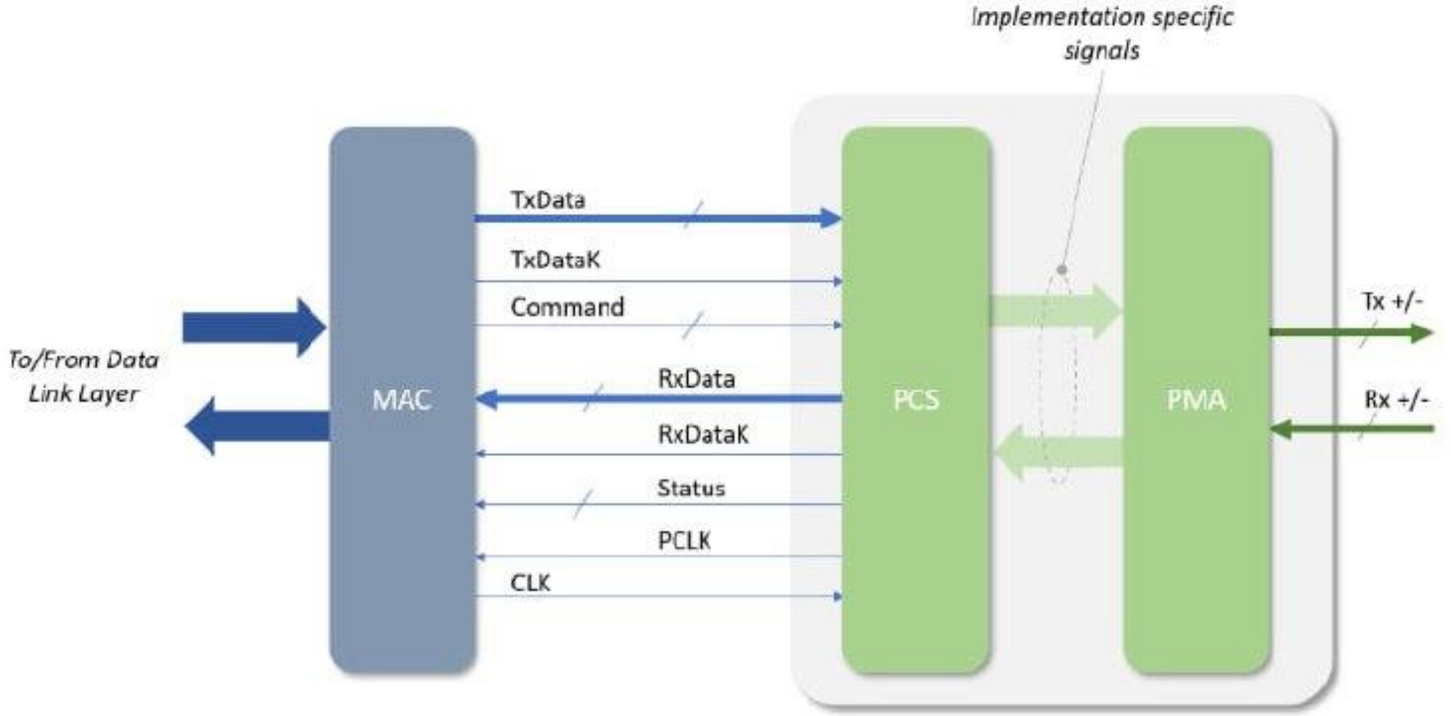


Device Layers Interaction

Physical Layer Overview

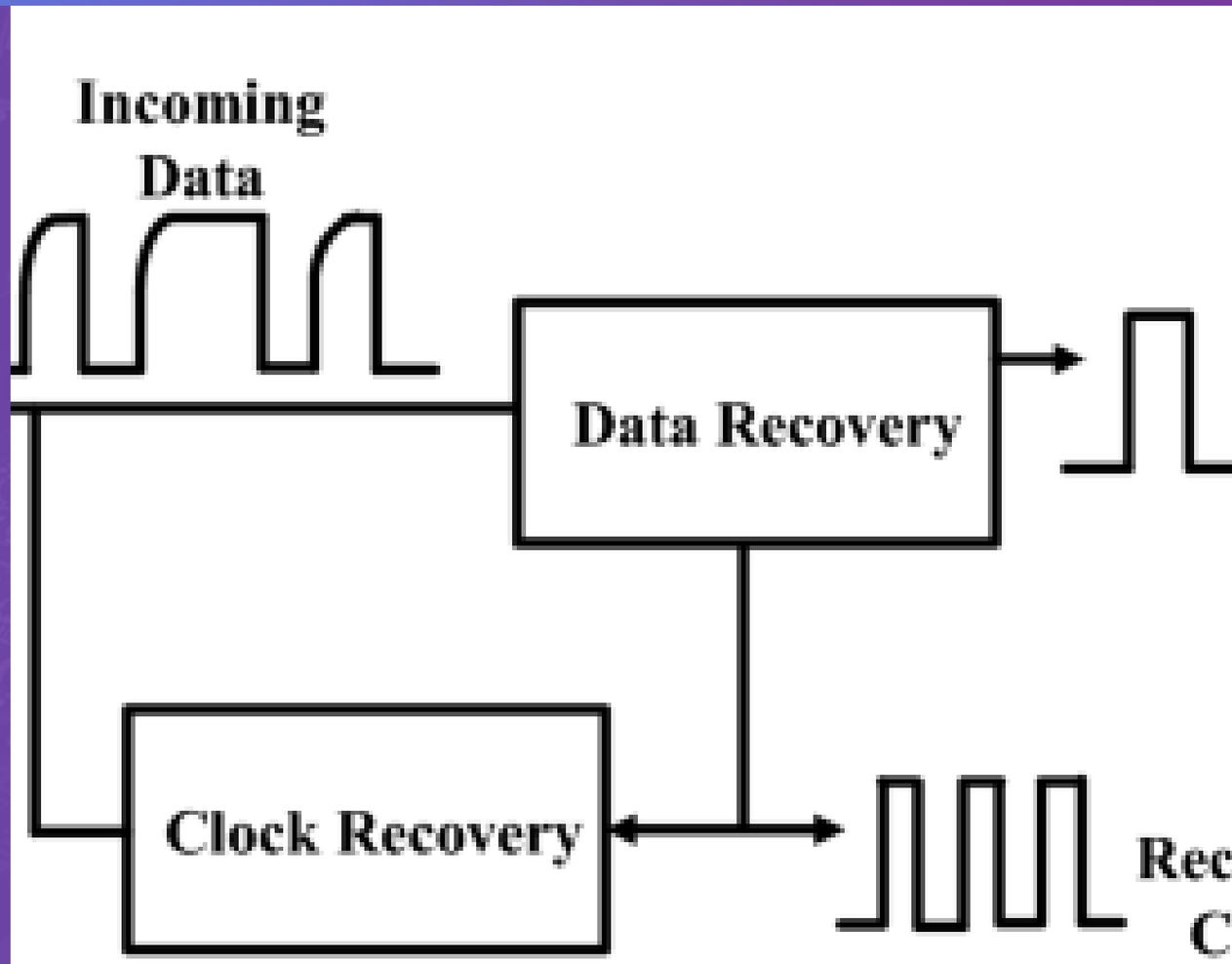
- Physical Coding Sublayer (PCS)
  - Responsible for data encoding/decoding (8b/10b or 128b/130b).
  - Handles elastic buffer for rate matching between the transmitter and receiver.
  - Detects receiver presence (Rx detection).

- Physical Media Attachment Layer (PMA)
  - Converts digital bits to analog signals using SERDES (Serializer/Deserializer).
  - Manages analog signal buffers and drivers.
  - Provides electrical interface for Tx (transmit) and Rx (receive) through the channel (PCIe lanes).



Device Layers Interaction

Clock and Data Recovery (CDR) Logic



Clock and data recovery

Device Layers Interaction

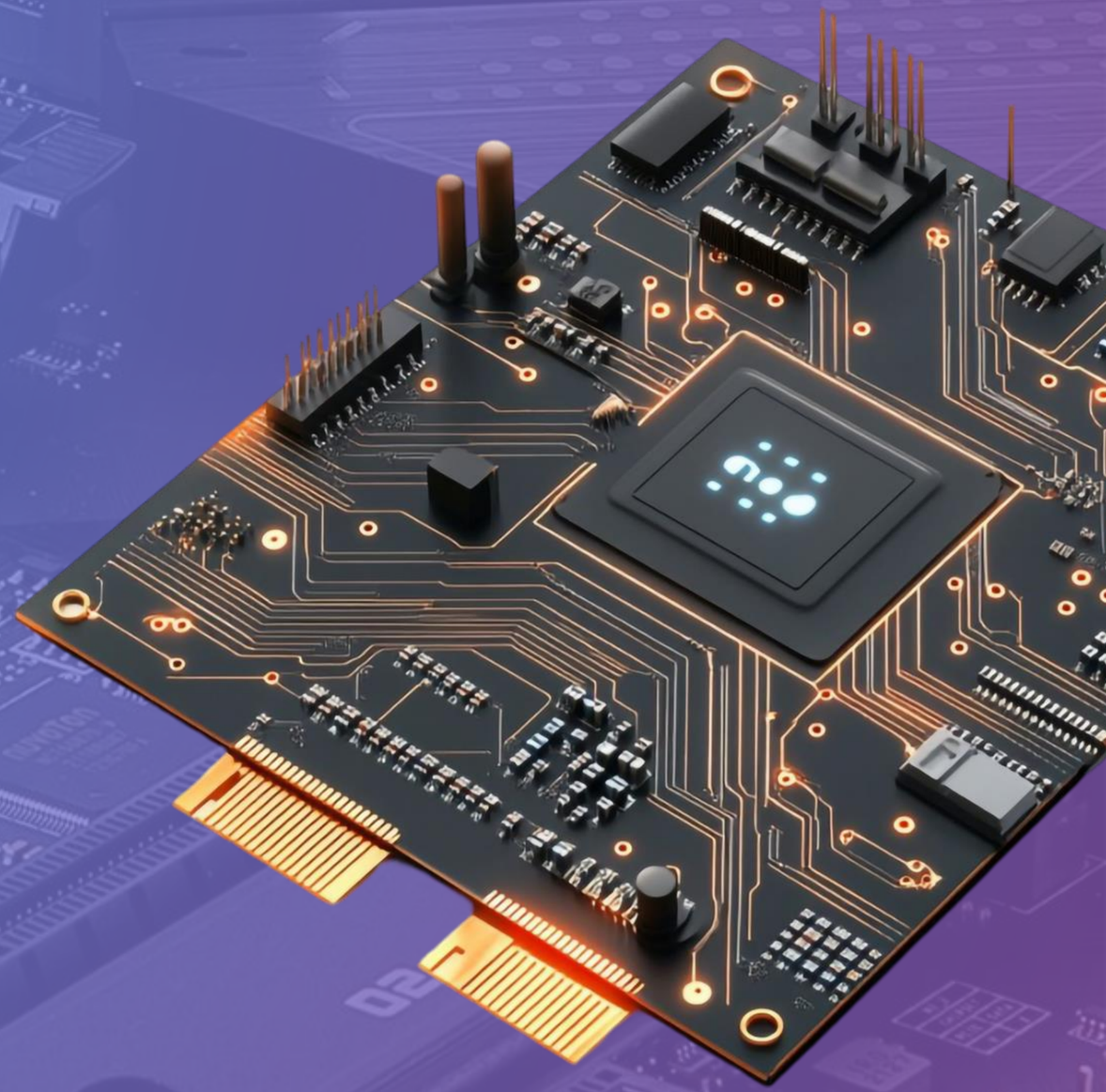
CDR

A common clock is not required for a PCIe Link because it uses a source-synchronous model, meaning the transmitter supplies the clock to the receiver to use in latching the incoming data.

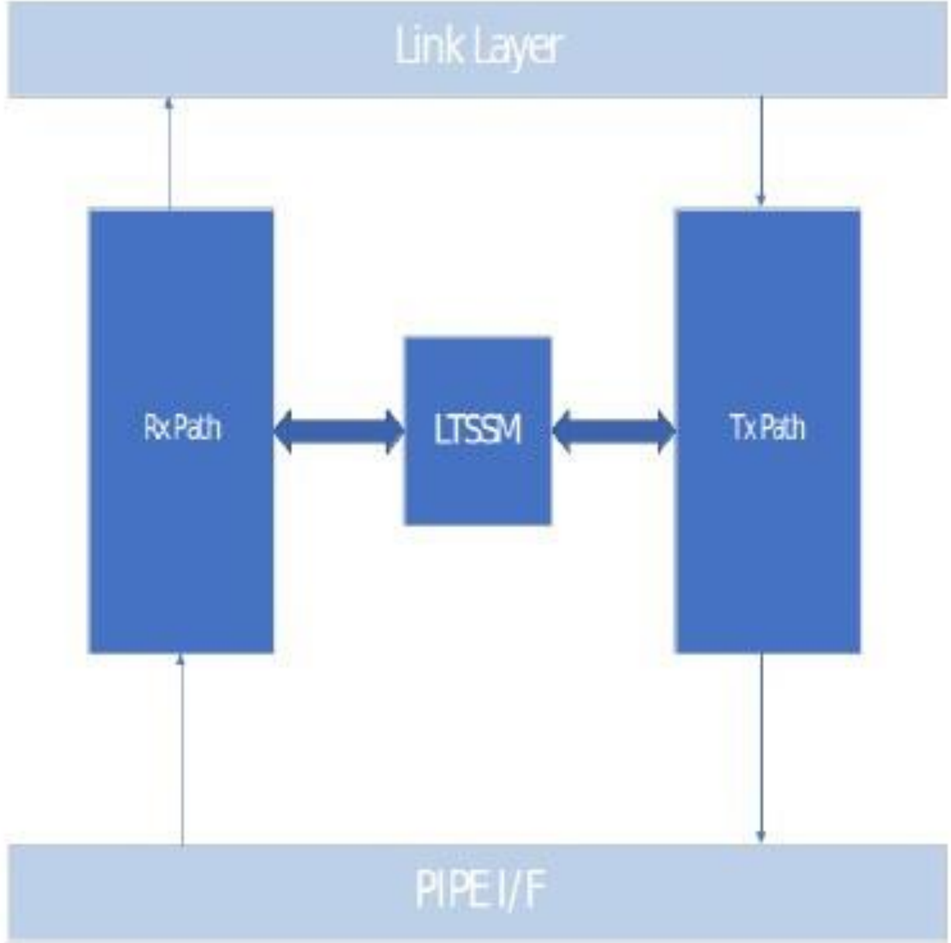
1. The transmitter embeds the clock into the data stream using 8b/10b encoding.
2. In the receiver, a PLL circuit takes the incoming bit stream as a reference clock and compares its timing, or phase, to that of an output clock that it has created with a specified frequency. Based on the result of that comparison, the output clock's frequency is increased or decreased until a match is obtained.

03

MAC Layer Architecture



MAC Layer Architecture

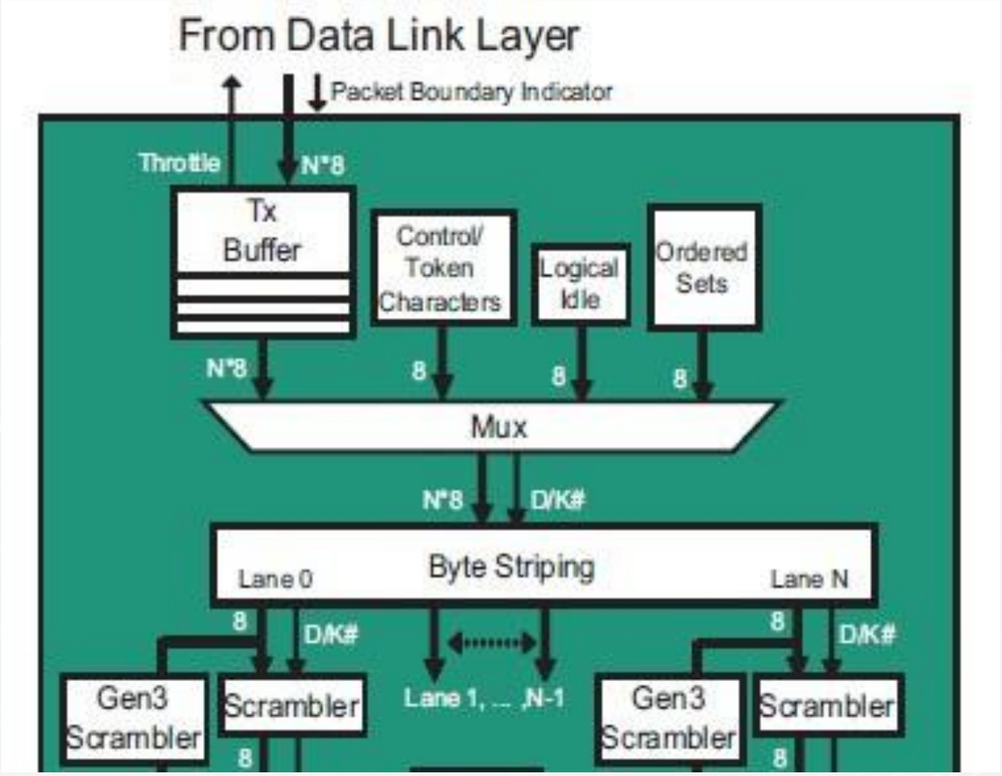


MAC Layer Architecture

TX Path

- **TX Buffer** : the buffer accepts TLPs and DLLPs from the Data Link Layer, along with ‘Control’ information that specifies when a new packet begins . ‘throttle’ signal is also shown going back up to the Data Link Layer to stop the flow of characters if the buffer should become full.

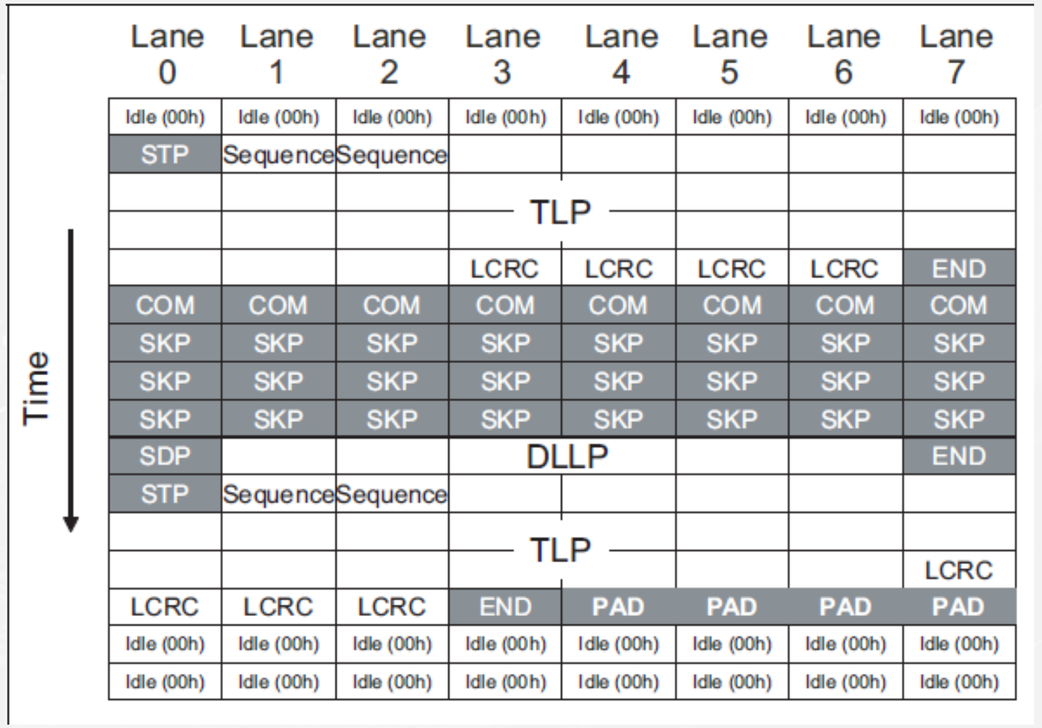
- **Mux and Control Logic**: Insertion of start and end control characters . Insertion of logical idles if no data is ready for transmission . Adding ordered sets for communication between physical layers .



MAC Layer Architecture

TX Path

- **Stripping** : Dividing data on all lanes . OS appears on all lanes at the same time . PADs as fillers for Gen1,2 and logical idle for higher gens.



MAC Layer Architecture

TX Path

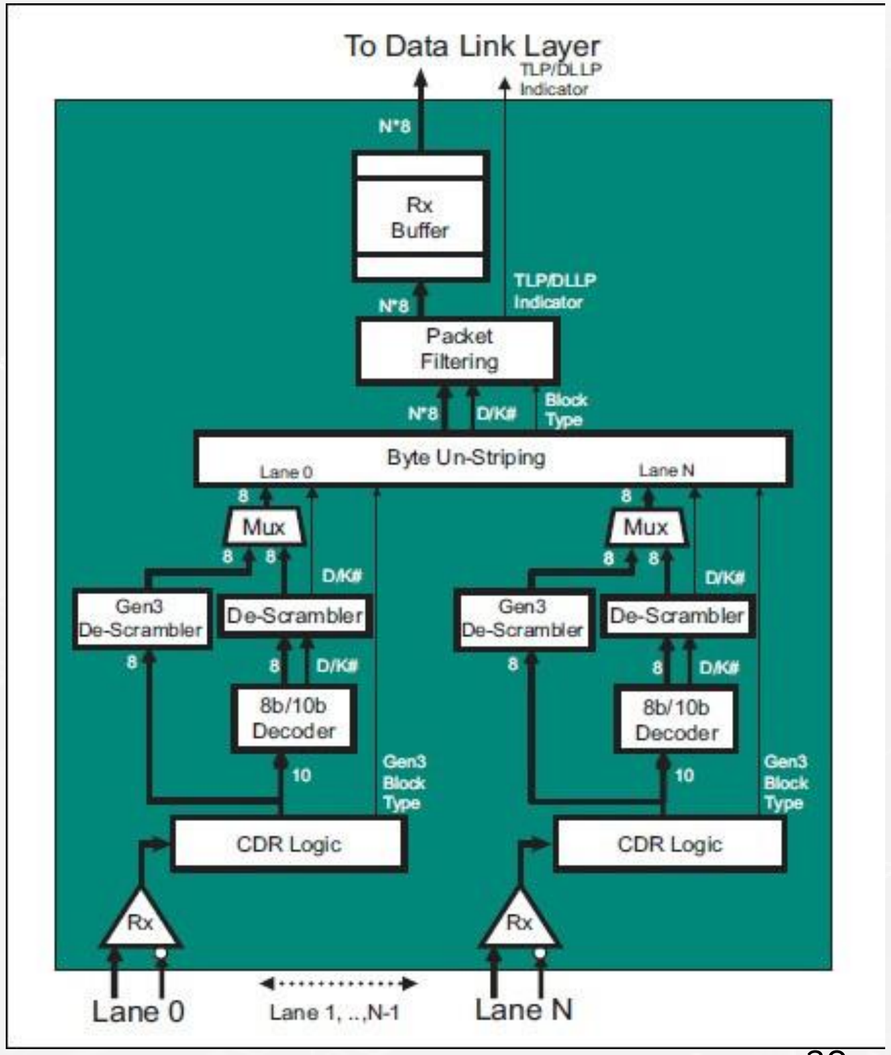
- **Scrambler(LFSR)** : Done to data not control . better for noise and EMI immunity . less correlation with neighboring links . can be disabled by 2 or more TS1,TS2 with the disable scrambling bit set . COM reinitialize the LFSR to FFFF.

- **Gen3,4,5 Scrambler :** EIEOS is used for synchronization and reinitialization of LFSR not COM . only symbol 0 of TS bypass scrambler . Cannot be disabled **.**

MAC Layer Architecture

RX Path

- **Descrambler** : Does the inverse function of scrambler to restore the real data before unstripping .
- **Unstripping** : Collects the data on all lanes to a single link before enterring the filteration process .
- **Packet filtering** : Removes TLP and DLLP tokens before forwarding them to Data Link Layer and removes logical idles and ordered sets .
- **RX Buffer** : The Rx Buffer holds received TLPs and DLLPs after the start and end characters have been eliminated. The received packets are ready to send to the Data Link Layer.



Ordered Sets

Ordered Sets are used for communication between the Physical Layers of Link partners and may be thought of as Lane management packets.

Types :

- **TS1 , TS2** : They are used for Link initialization and training. They allow the Link partners to achieve bit lock and Symbol lock and negotiate the link speed

- **Electrical Idle Ordered Set (EIOS)** : A Transmitter that wishes to go to a lower-power link state sends this before ceasing transmission. Upon receipt, Receivers know to prepare for the low power state. The EIOS consists of four Symbols: the COM Symbol followed by three IDL Symbol

Ordered Sets

- **Electrical Idle Exit Ordered Set (EIEOS)** : Added in the PCIe 2.0 spec, this Ordered Set was defined to provide a lower-frequency sequence required to exit the electrical idle Link state.

- **SKP Ordered Set (SOS)** : It's transmitted at regular intervals and is used for Clock Tolerance Compensation . Receiver evaluates the SOS and internally adds or removes SKP Symbols as needed to prevent its elastic buffer from under-flowing or over-flowing. It consists of four Symbols: the COM Symbol followed by three SKP Symbols.

- **Start of Data Stream Ordered Set (SDS)** : The Link enters a Data Stream by sending an SDS Ordered Set and transitioning to the L0 Link state

MAC Layer Architecture

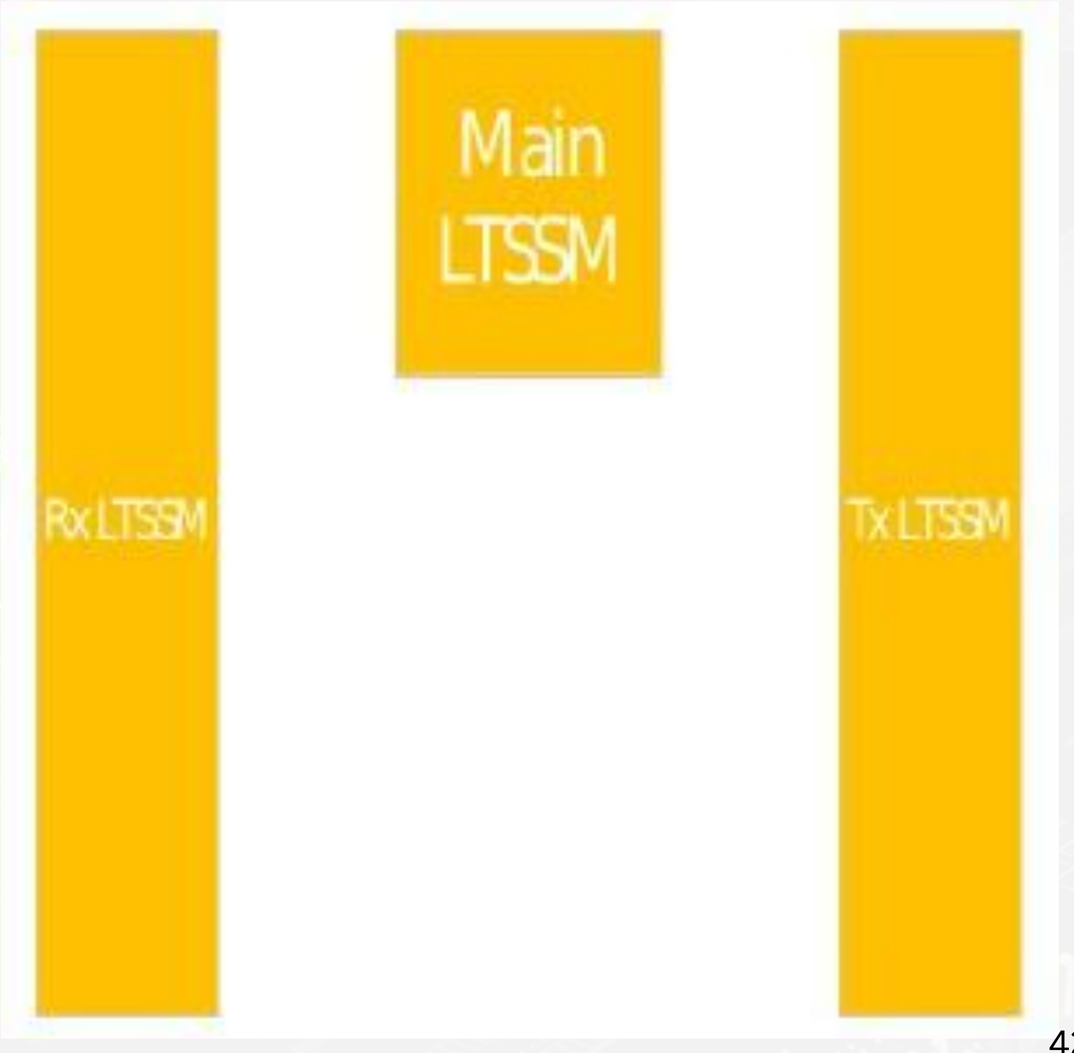
LTSSM

LTSSM is divided into two main parts:

-? TX LTSSM

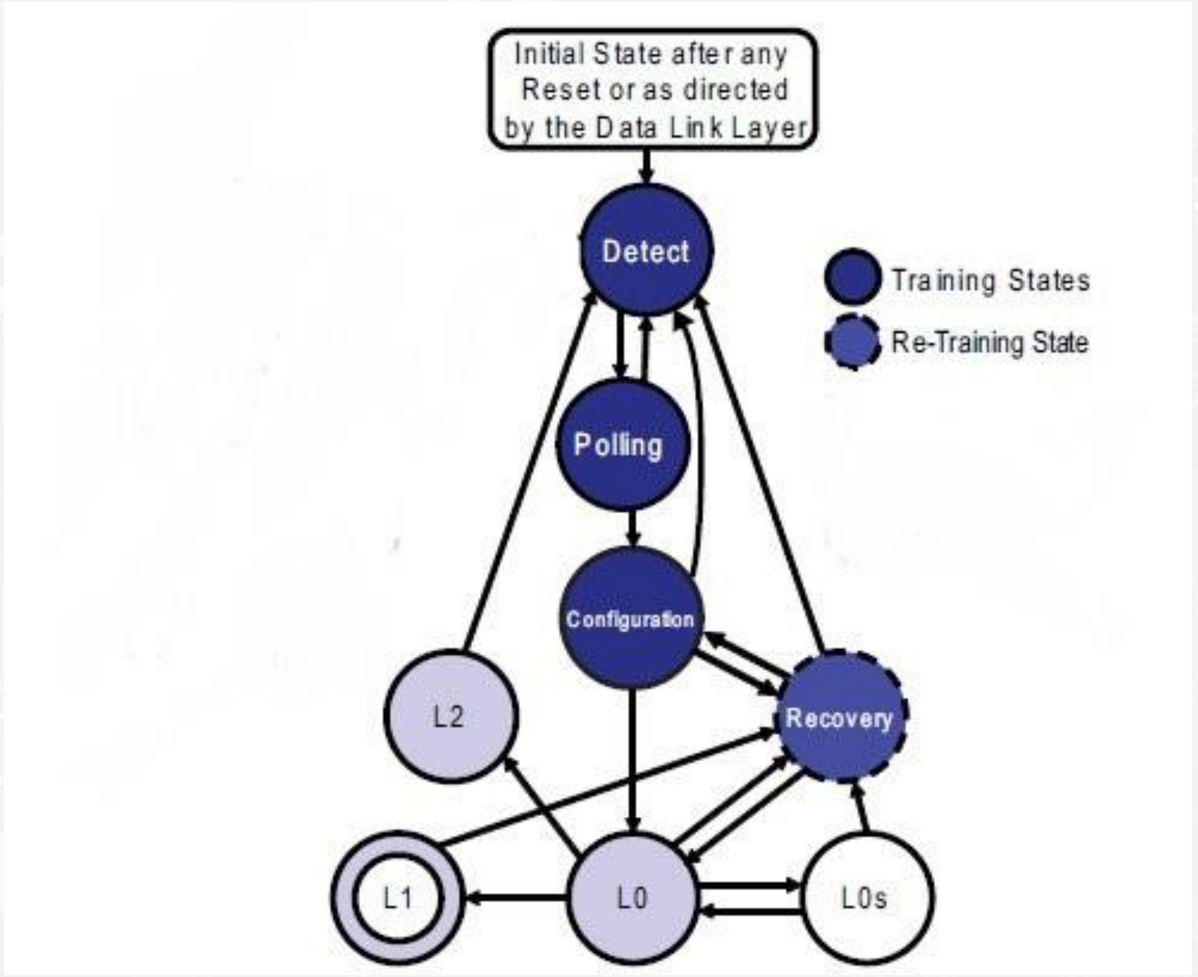
-? RX LTSSM

Main LTSSM : Represents spec states and Controls sub-states (Detect - Polling - Configuration - L0 -Recovery)



MAC Layer Architecture

LTSSM



LTSSM

- **Detect** : The initial state after reset. In this state, a device electrically detects a Receiver is present at the far end of the Link. Detect may also be entered from a number of other LTSSM states as described later.

- **Polling** : In this state, Transmitters begin to send TS1s and TS2s so that Receivers can use them to Achieve Bit Lock , Acquire Symbol Lock or Block Lock and Learn available Lane data rates .

LTSSM

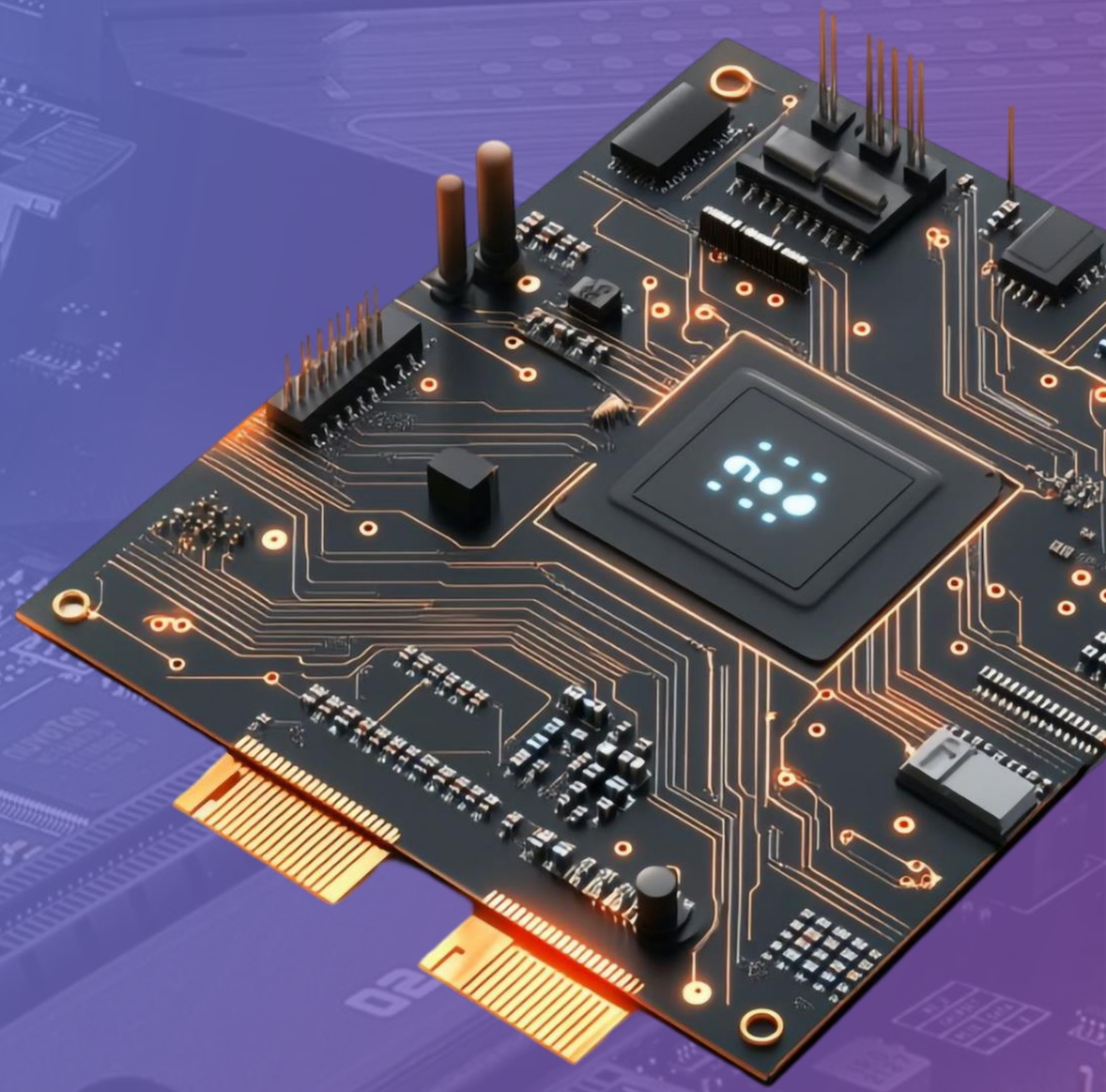
- **Configuration** : Upstream and Downstream components now play specific roles as they continue to exchange TS1s and TS2s to Determine Link width , Assign Lane numbers and Deskew Lane-to-Lane timing differences .

- **Recovery** : This state is entered when the Link needs re-training. This could be caused by errors in L0, recovery from lower power states and seeking higher speeds . In Recovery, Bit Lock and Symbol/Block Lock are re-established in a manner similar to that used in the Polling state but it typically takes much less time.

- **L0** : This is the normal, fully-active state of a Link during which TLPs, DLLPs and Ordered Sets can be exchanged

04

PCIe Verification Process



Why Is Verification So Important?

Pentium FDIV bug

In 1994, Professor Thomas Nicely found a calculation error caused by a flaw in Intel's Pentium processor's FDIV unit. Although Intel already knew about it, they didn't make it public. When Nicely shared the bug, it spread quickly online and in the media.

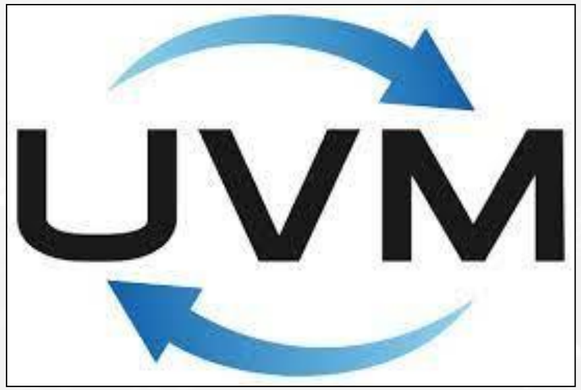
After public pressure and IBM halting Pentium sales, Intel finally agreed on December 20 to replace all faulty chips. The incident cost Intel $475 million and pushed the company to take verification more seriously in future processor designs.

PCIe Verification Process

How do we efficiently verify today's complex hardware?

In the past, hardware verification relied on simple Verilog testbenches, which were fine for small designs. But as chips became more complex, this approach no longer scaled.

Today, SystemVerilog and UVM are used to build efficient, reusable, and scalable verification environments. SystemVerilog brings essential features like object-oriented programming, constrained randomization, assertions, and functional coverage — all aimed at improving quality and reducing effort.

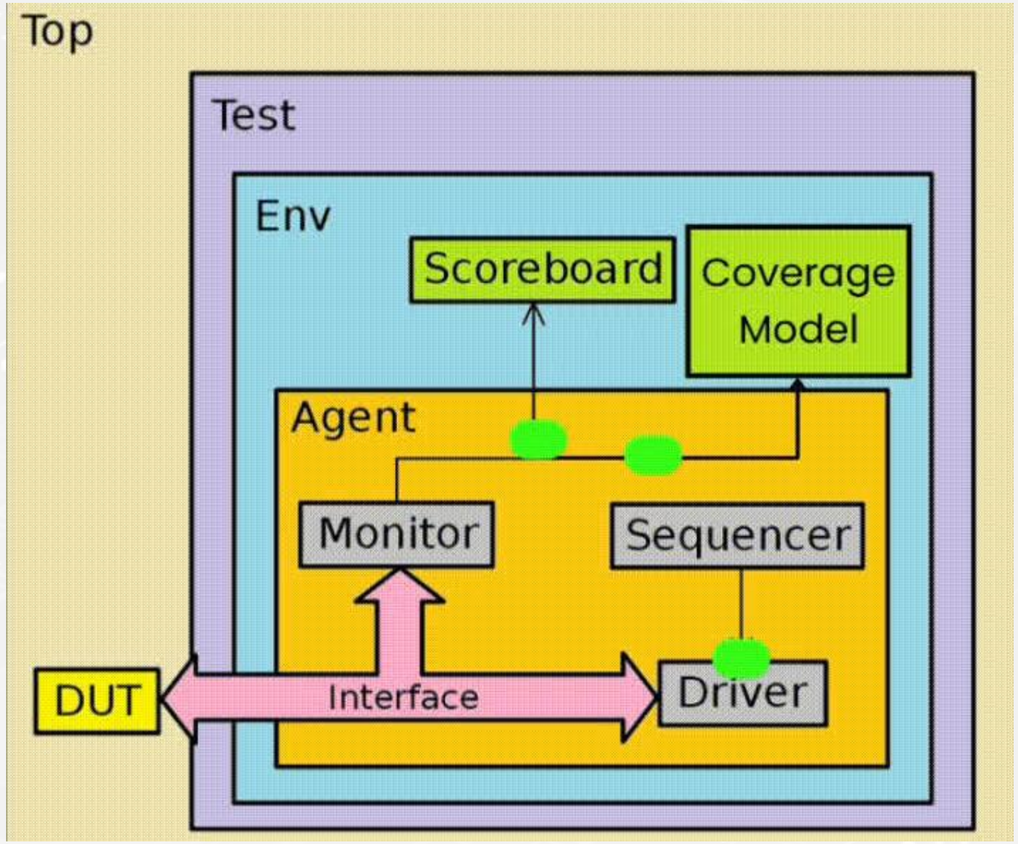


Intro To UVM

- **UVM Transaction / Sequence Item**
  Data structure used to represent inputs/outputs in the testbench.

- **UVM Sequencer**
  Generates and sends transactions to the driver based on test scenarios.

- **UVM Driver**
  Converts sequence items into pin-level signals and drives them to the DUT.

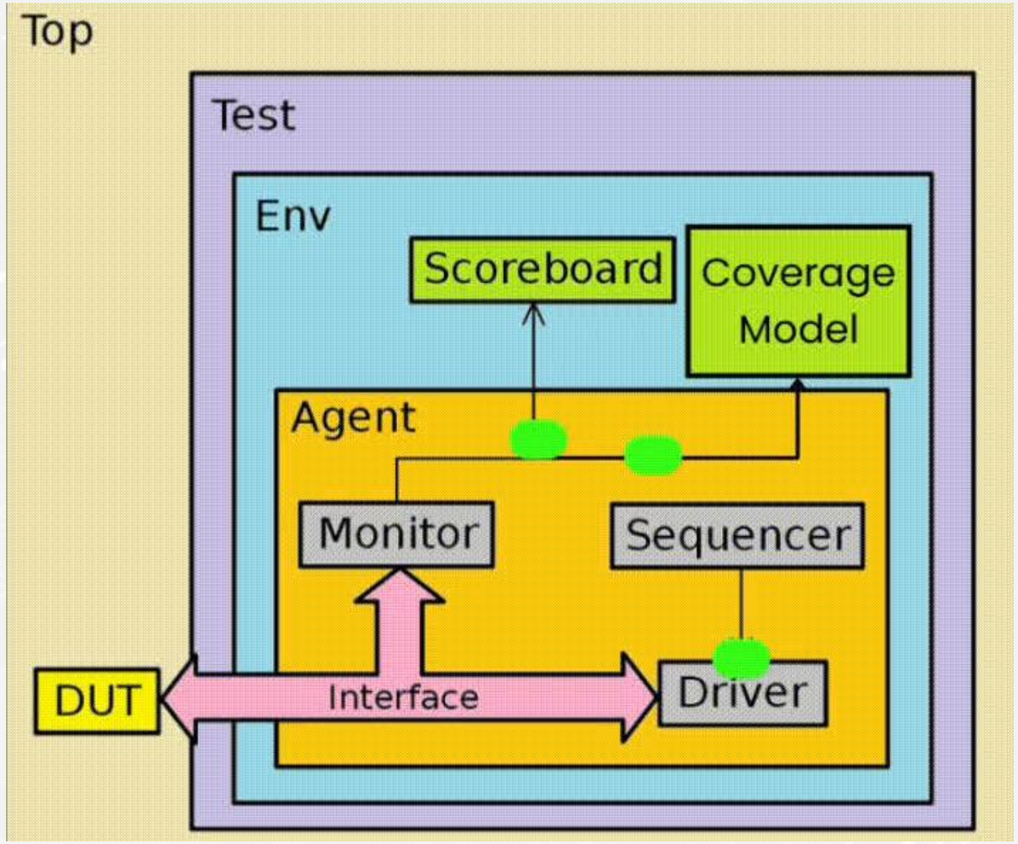


Intro To UVM

- **UVM Monitor**
  Observes DUT signals and converts them into transactions for checking and coverage.

- **UVM Scoreboard**
  Compares expected and actual outputs to check DUT correctness.

- **Coverage Model**
  Tracks which scenarios have been exercised using functional coverage (covergroups).

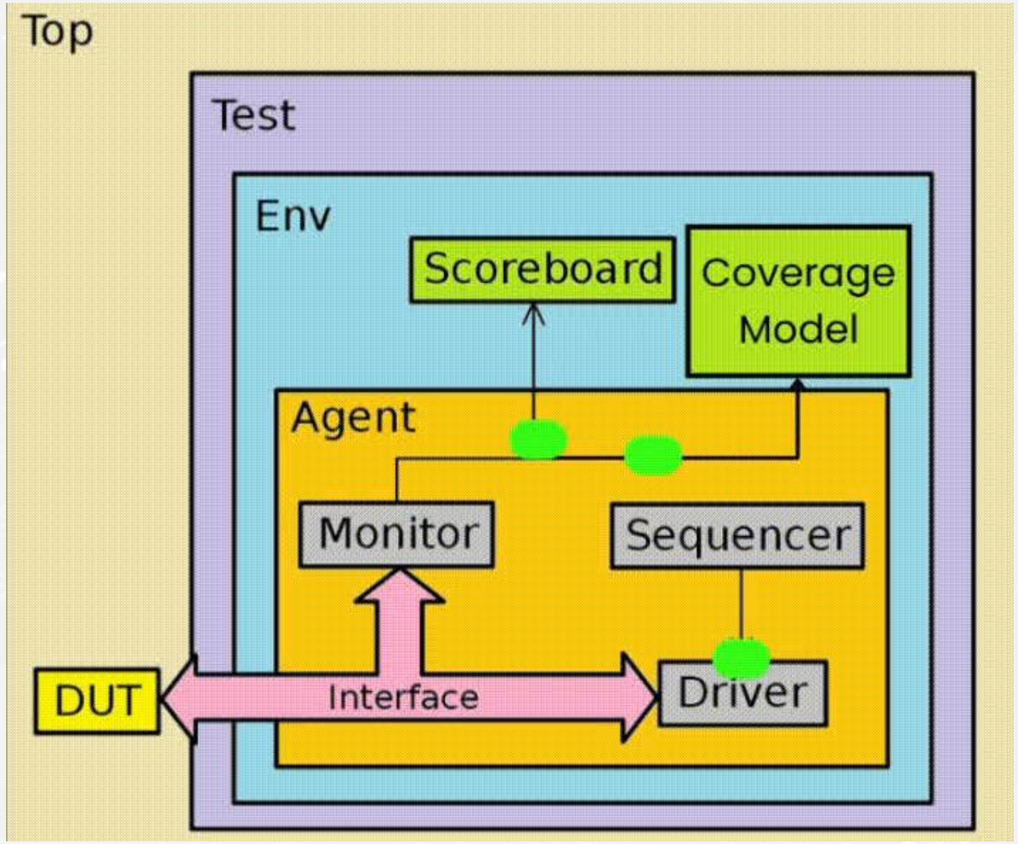


Intro To UVM

- **UVM Agent**
  Groups the driver, sequencer, and monitor. Can be active or passive.

- **UVM Environment (env)**
  Container for agents, scoreboards, coverage models, and other components.

- **UVM Test**
  Top-level component that starts the simulation and selects the test scenario.



PCIe Verification Process

Verification Planning

PIPE Interface						LBIF Interface		
TX Related Signals	RX Related Signals	Equalization Signals	Commands and Status Signals	Message Bus Interface Signals	System Signals	TX Related Signals	RX Related Signals	LTSSM Signals
TxData	RxData	LocalTxPresetCoefficients	PowerDown	M2P_MessageBus	phy_reset	lp_data	pl_data	lp_state_req
TxDataValid	RxDataValid	TXDeemph	Rate	P2M_MessageBus	CLK	lp_valid	pl_valid	pl_state_sts
TxElecIdle	RxDataK	LocalFS	PhyStatus	RxStandby		lp_irdy	pl_trdy	lp_force_detect
TxDataK	RxStartBlock	LocalLF	Width			lp_dlpstart	pl_dlpstart	pl_speedmode
TxStartBlock	RxSyncHeader	GetLocalPresetCoefficients	PclkChangeAck			lp_dlpend	pl_dlpend	pl_linkUp
TxSyncHeader	RxValid	LocalTxCoefficientsValid	PclkChangeOk			lp_tlpstart	pl_tlpstart	lpreset
TxDetectRx_ Loopback	RxStatus	LF				lp_tlpend	pl_tlpend	
	RXElecIdle	RxEqEval				lp_tlpedb	pl_tlpedb	
		LocalPresetIndex						
		InvalidRequest						
		LinkEvaluationFeedback DirectionChange						

Parameter Name	Default Value	Description	Legal Values
MAXPIPEWIDTH	32	the maximum supported PIPE width	8 or 16 or 32
DEVICETYPE	0	device type is upstream (1) or downstream (0)	0 or 1
LANESNUMBER	16	the supported link width	1 or 2 or 4 or 8 or 16
GEN1_PIPEWIDTH	8	the supported PIPE width for gen1	8 or 16 or 32
GEN2_PIPEWIDTH	8	the supported PIPE width for gen2	8 or 16 or 32
GEN3_PIPEWIDTH	8	the supported PIPE width for gen3	8 or 16 or 32
GEN4_PIPEWIDTH	8	the supported PIPE width for gen4	8 or 16 or 32
GEN5_PIPEWIDTH	8	the supported PIPE width for gen5	8 or 16 or 32
MAX_GEN	1	the maximum supported PCIe gen	1 or 2 or 3 or 4 or 5

PCIe Verification Process

Verification Planning

ID	Scope/Interface	Feature Name	Associated Signals	Design Requirement/Feature Description	Reference (Section/Chapter & Page)
LTSSM_1	LPIF	MAC Reset	lpreset	This signal is used to reset the PCIe and come from the upper layers of the	Design Specific Signal
LTSSM_2	PIPE	PHY Reset	phy_reset	When we assert lpreset signal, phy_reset should be asserted	Design Specific Signal
LTSSM_3	LPIF	Link Up & Initialization	lp_state_req & pl_linkUp	The Link Up feature in PCIe ensures that the connection between two devices is fully initialized and operational. It signifies readiness for data transfer following successful link training.When an active state is requested in PCIe, the pl_linkUp signal should be asserted if all conditions are met	Ch14 - page 506
LTSSM_4	LPIF	State Request	lp_state_req & pl_state_sts	When an a specific state is requested in PCIe, pl_state_sts signal should indicates the requested state if all conditions are met	Design Specific Signal
LTSSM_5	LPIF	Forcing Detect	lp_force_detect	Activating this signal causes the LTSSM to transition to the detect state	Design Specific Signal
LTSSM_6	PIPE	PHY Speed Controlling	Rate	During a speed change in PCIe, the MAC layer outputs a signal, Rate, to instruct the PHY layer on the speed at which to operate. This ensures synchronized operation between the MAC and PHY layers at the updated link speed.	PIPE Intel Specs
LTSSM_7	LPIF	Speed Change	lp_state_req & pl_speedmode	When the retrain state is requested using the lp_state_req signal and all conditions are met, the pl_speedmode should be updated to reflect the newly negotiated speed	Ch14 - page 619

PCIe Verification Process

Verification Planning

ID	Scope/Interface	Feature Name	Associated Signals	Design Requirement/Feature Description	Generation Plan
LTSSM_1	LPIF	MAC Reset	lpreset	This signal is used to reset the PCIe and come from the upper layers of the PCIe	Drive lpreset signal from LPIF
LTSSM_2	PIPE	PHY Reset	phy_reset	When we assert lpreset signal, phy_reset should be activated	Drive lpreset signal from LPIF
LTSSM_3	LPIF	Link Up & Initialization	lp_state_req & pl_linkUp	The Link Up feature in PCIe ensures that the connection between two devices is fully initialized and operational. It signifies readiness for data transfer following successful link training.When an active state is requested in PCIe, the pl_linkUp signal should be asserted if all	Drive lp_state_req signal with value 4'b0001(request L0 state) from LPIF
LTSSM_4	LPIF	State Request	lp_state_req & pl_state_sts	When an a specific state is requested in PCIe, pl_state_sts signal should indicates the requested state if all conditions are met	Drive lp_state_req signal from LPIF
LTSSM_5	LPIF	Forcing Detect	lp_force_detect	Activating this signal causes the LTSSM to transition to the detect state	Drive lp_force_detect signal from LPIF
LTSSM_6	PIPE	PHY Speed Controlling	Rate	During a speed change in PCIe, the MAC layer outputs a signal, Rate, to instruct the PHY layer on the speed at which to operate. This ensures synchronized operation between the MAC and PHY layers at the updated link speed.	Drive lp_state_req signal with value 4'b0001 (request L0 state) from LPIF
LTSSM_7	LPIF	Speed Change	lp_state_req & pl_speedmode	When the retrain state is requested using the lp_state_req signal and all conditions are met, the pl_speedmode should be updated to reflect the newly negotiated speed	Drive lp_state_req signal with value 4'b0001(request L0 state) from LPIF and the highest common speed between Device1 and Device2 greater than speed of Gen1

PCIe Verification Process

Verification Planning

ID	Scope/Interface	Feature Name	Associated Signals	Design Requirement/Feature Description	Checking/Assertion Plan
LTSSM_1	LPIF	MAC Reset	lpreset	This signal is used to reset the PCIe and come from the upper layers of the PCIe	Check it using assertion
LTSSM_2	PIPE	PHY Reset	phy_reset	When we assert lpreset signal, phy_reset should be activated	Check it using assertion
LTSSM_3	LPIF	Link Up & Initialization	lp_state_req & pl_linkUp	The Link Up feature in PCIe ensures that the connection between two devices is fully initialized and operational. It signifies readiness for data transfer following successful link training.When an active state is requested in PCIe, the pl_linkUp signal should be asserted if all conditions are met	Check it using the TX and RX slaves monitors and the Scoreboard
LTSSM_4	LPIF	State Request	lp_state_req & pl_state_sts	When an a specific state is requested in PCIe, pl_state_sts signal should indicates the requested state if all conditions are met	Check it using the Scoreboard
LTSSM_5	LPIF	Forcing Detect	lp_force_detect	Activating this signal causes the LTSSM to transition to the detect state	Check it using assertion
LTSSM_6	PIPE	PHY Speed Controlling	Rate	During a speed change in PCIe, the MAC layer outputs a signal, Rate, to instruct the PHY layer on the speed at which to operate. This ensures synchronized operation between the MAC and PHY layers at the updated link speed.	Check it using the Scoreboard
LTSSM_7	LPIF	Speed Change	lp_state_req & pl_speedmode	When the retrain state is requested using the lp_state_req signal and all conditions are met, the pl_speedmode should be updated to reflect the newly negotiated speed	Check it using the TX and RX slaves monitors and the Scoreboard

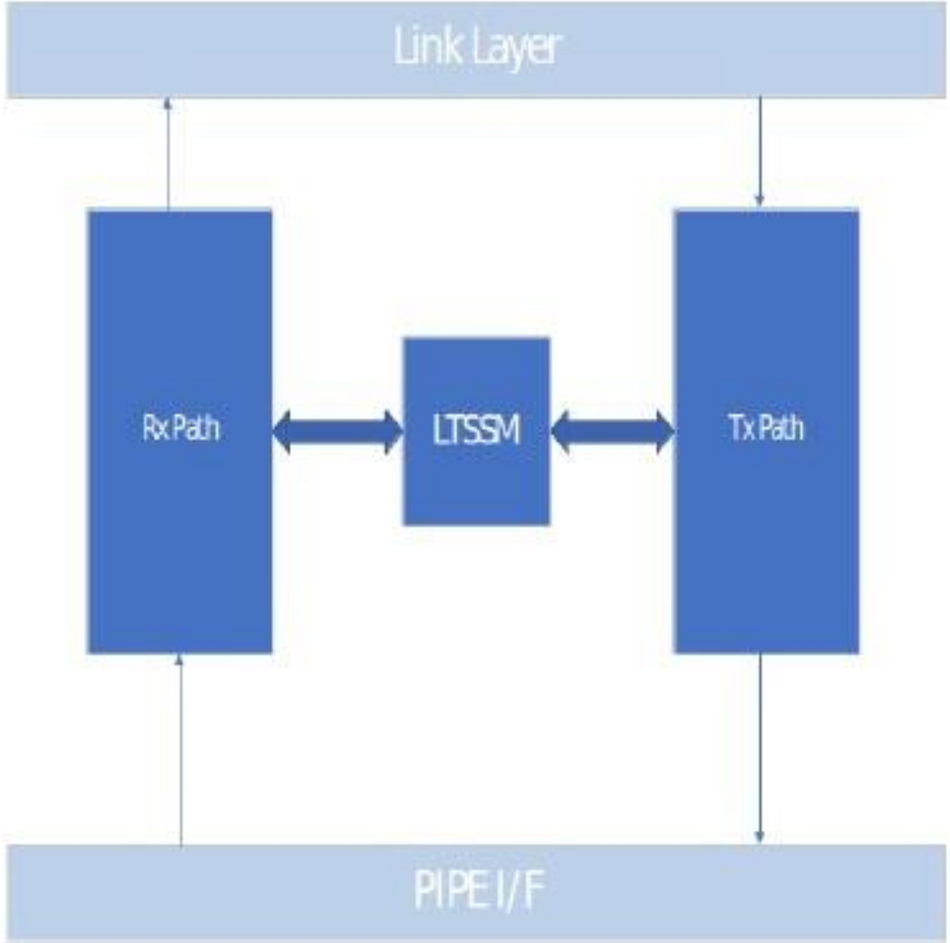
PCIe Verification Process

Verification Planning

ID	Scope/Interface	Feature Name	Associated Signals	Design Requirement/Feature Description	Coverage Plan
LTSSM_1	LPIF	MAC Reset	lpreset	This signal is used to reset the PCIe and come from the upper layers of the PCIe	Cover it using cover property
LTSSM_2	PIPE	PHY Reset	phy_reset	When we assert lpreset signal, phy_reset should be activated	Cover it using cover property
LTSSM_3	LPIF	Link Up & Initialization	lp_state_req & pl_linkUp	The Link Up feature in PCIe ensures that the connection between two devices is fully initialized and operational. It signifies readiness for data transfer following successful link training.When an active state is requested in PCIe, the pl_linkUp signal should be asserted if all conditions are met	Cover it using cover property
LTSSM_4	LPIF	State Request	lp_state_req & pl_state_sts	When an a specific state is requested in PCIe, pl_state_sts signal should indicates the requested state if all conditions are met	Cover it using coverage model for cases: lp_state_req = 4'b0000 (no request) lp_state_req = 4'b0001 (link-up request)
LTSSM_5	LPIF	Forcing Detect	lp_force_detect	Activating this signal causes the LTSSM to transition to the detect state	Cover it using cover property
LTSSM_6	PIPE	PHY Speed Controlling	Rate	During a speed change in PCIe, the MAC layer outputs a signal, Rate, to instruct the PHY layer on the speed at which to operate. This ensures synchronized operation between the MAC and PHY layers at the updated link speed.	Cover it using coverage model
LTSSM_7	LPIF	Speed Change	lp_state_req & pl_speedmode	When the retrain state is requested using the lp_state_req signal and all conditions are met, the pl_speedmode should be updated to reflect the newly negotiated speed	Cover it using cover property

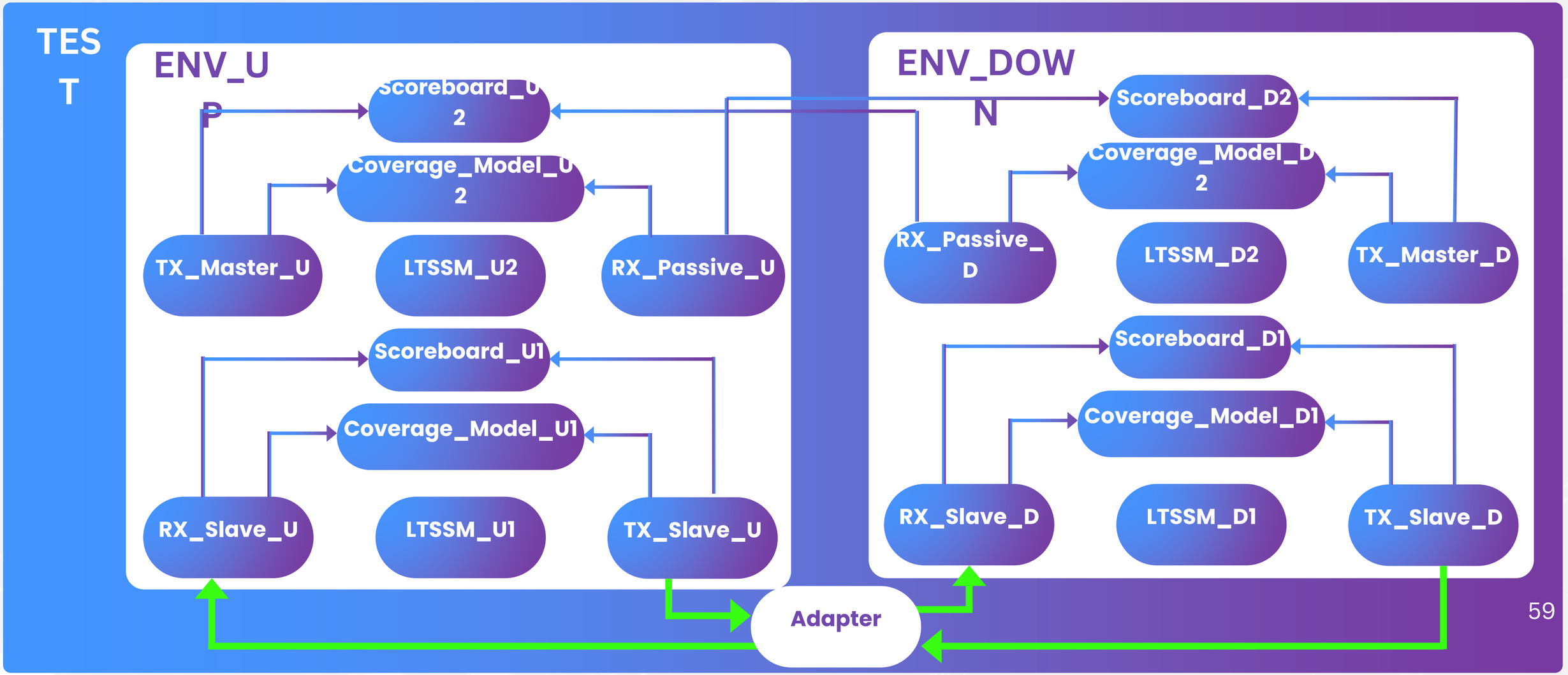
PCIe Verification Process

Design Architecture



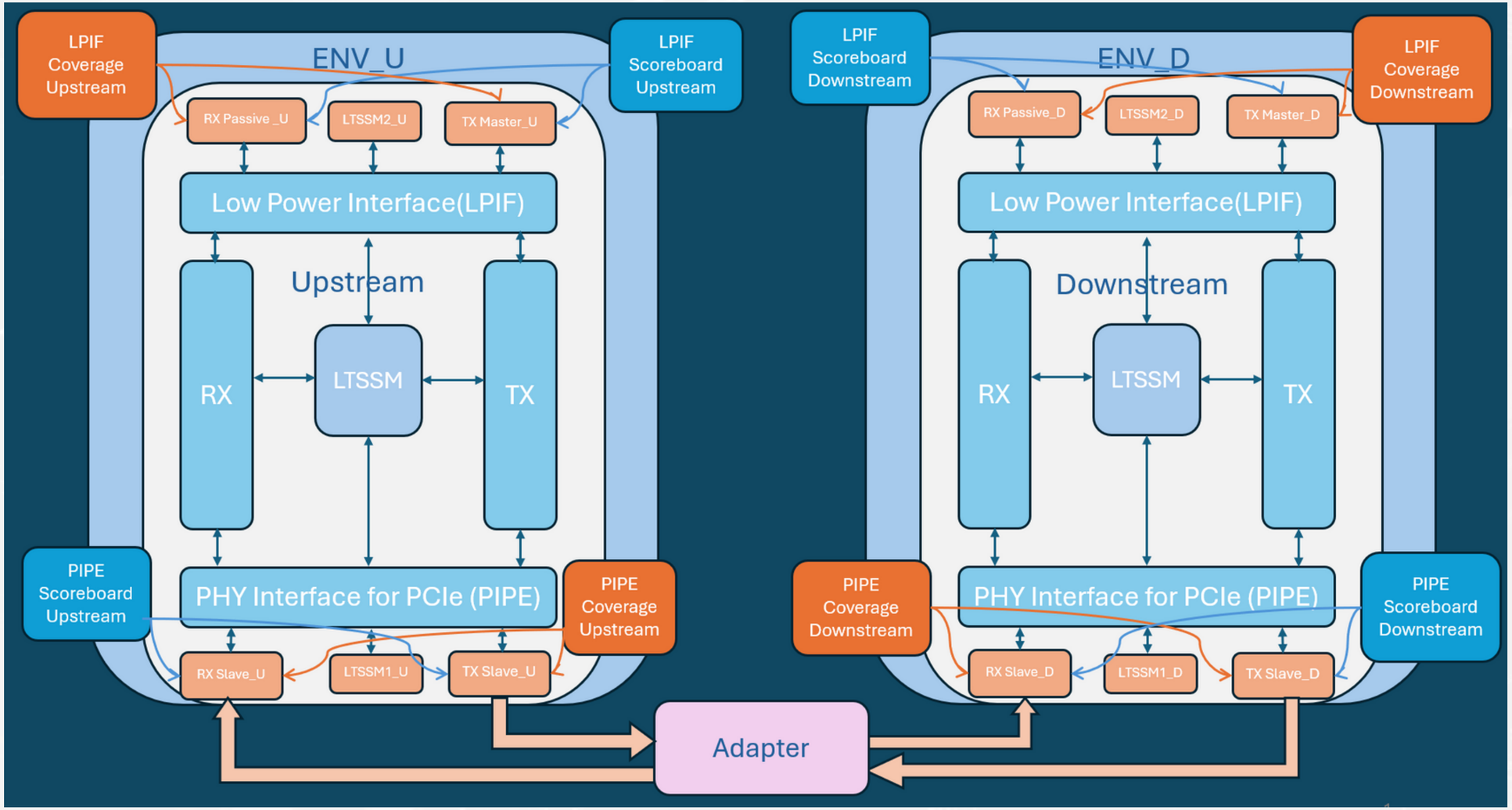
PCIe Verification Process

Testbench Architecture



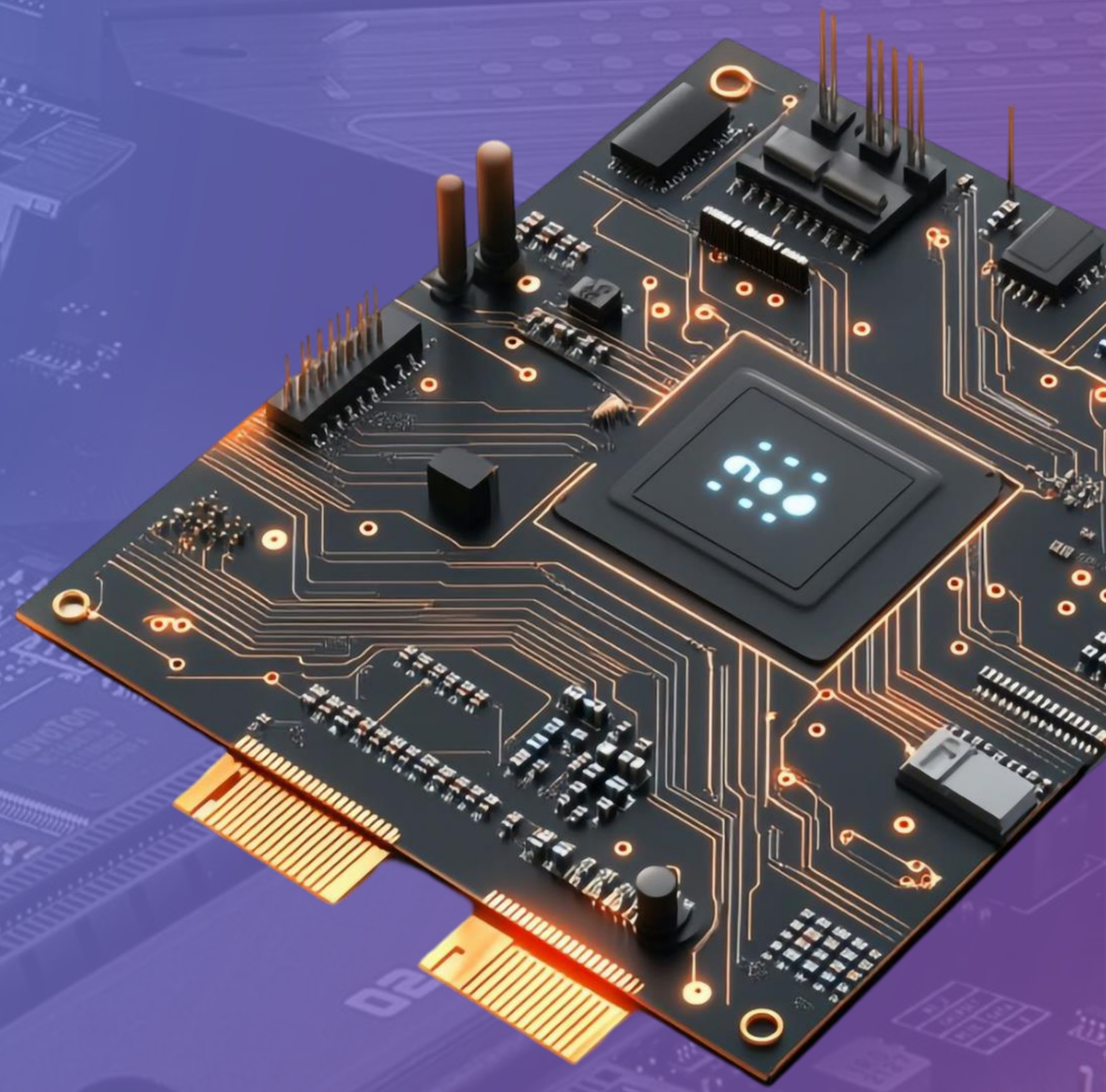
PCIe Verification Process

Testbench and Design Architectures

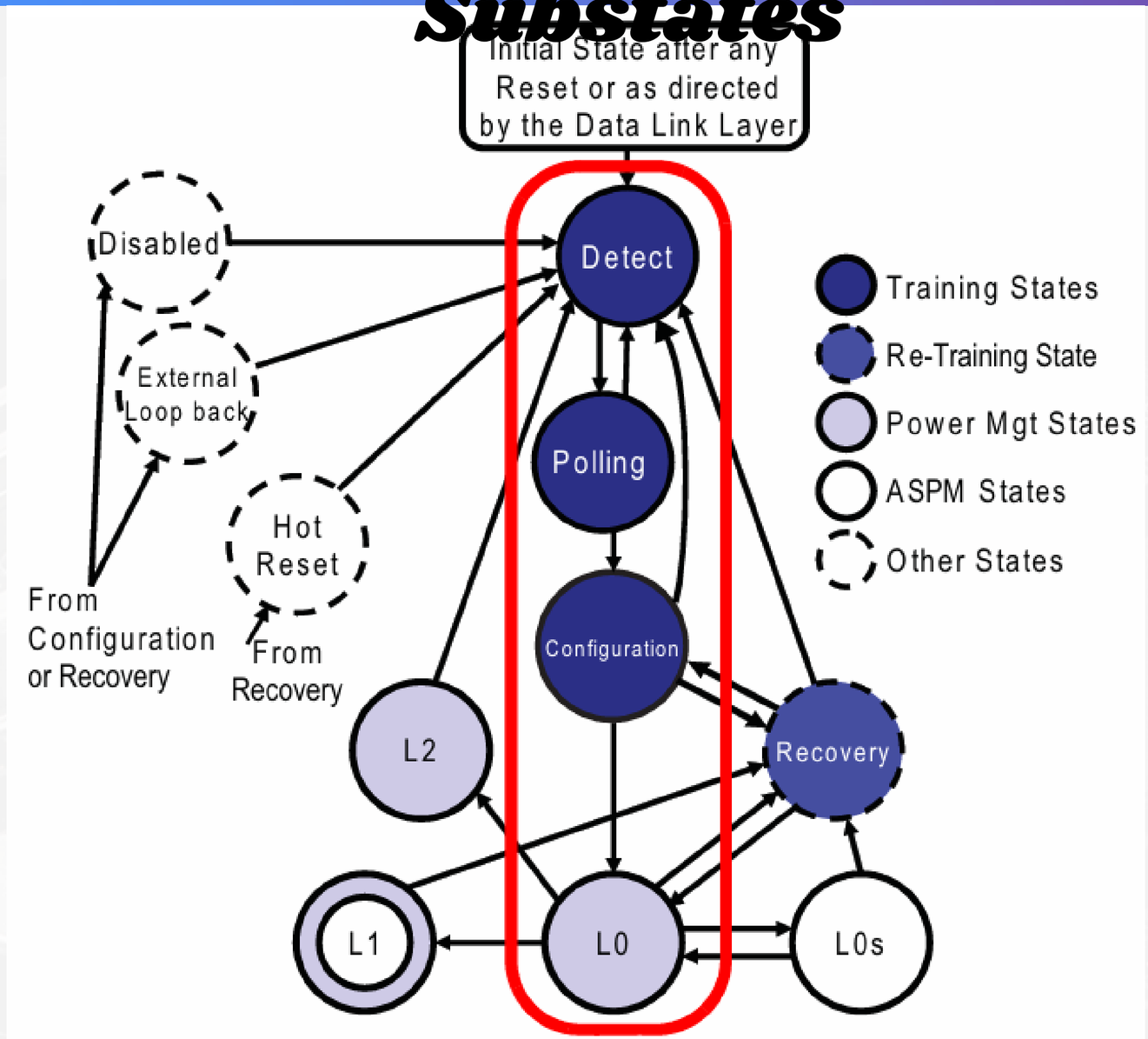


05

Link UP Substates



Link UP Substates



Link UP Substates

Main Goal

To find out if another PCIe device (receiver) is connected on the other side of the link.

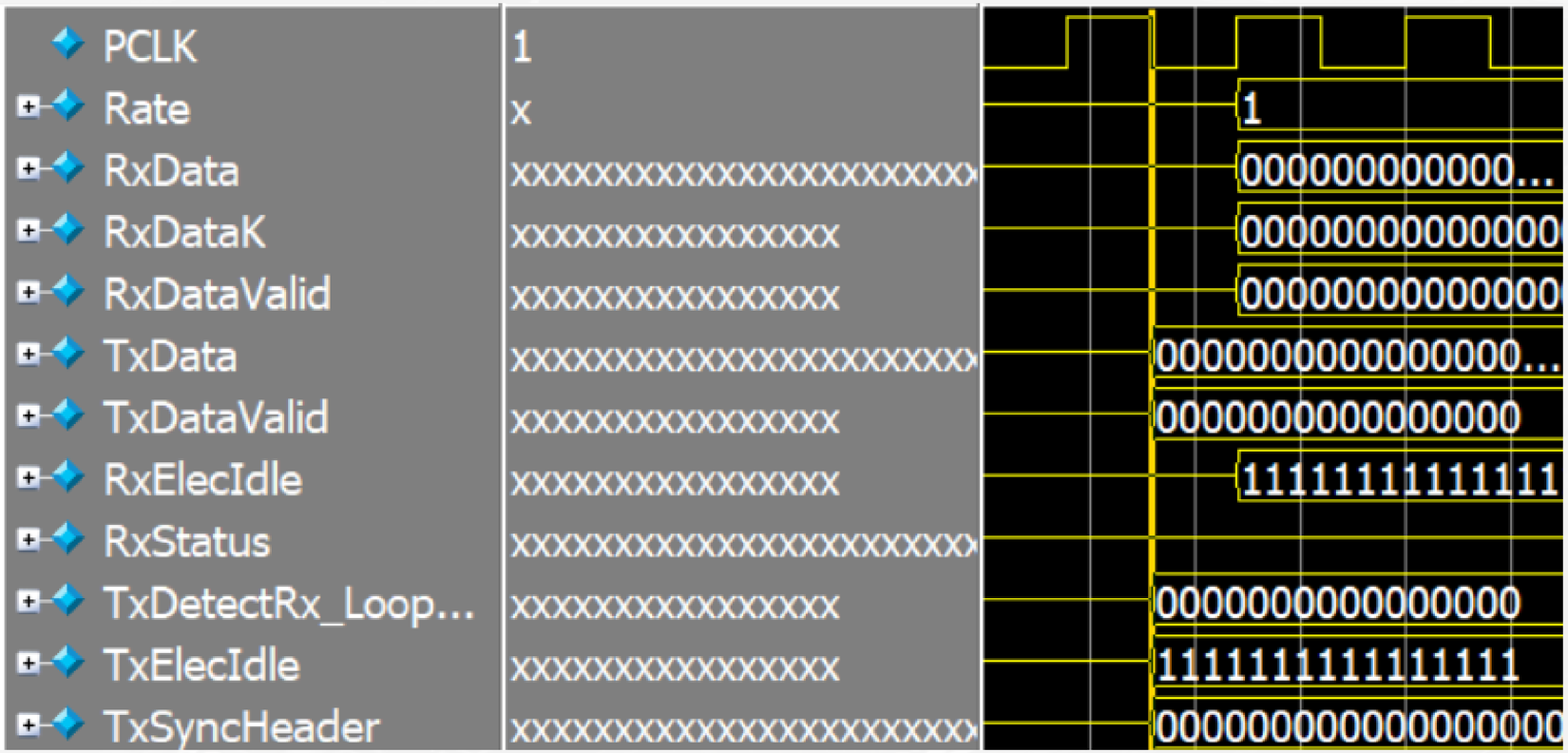
- Step 1: Detect.Quiet

  This is the starting point after power-on or reset.

  What happens here:
- The transmitter (Tx) does nothing — it stays silent, in a mode called Electrical Idle (i.e., it doesn't send any signals).
- The link is locked to Gen1 speed (2.5 GT/s) at this point.
- If nothing happens for 12 ms, we exit to Detect.Active.

Link UP Substates

Simulation




```
# UVM_INFO RX_Slave_D_Monitor.sv(1828) @ 16050: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Sla
ve_D_Monitor_h [RX_Slave_D_Monitor] Detect Quiet substate at Downstraem RX side completed success
fully
```

Link UP

Substates

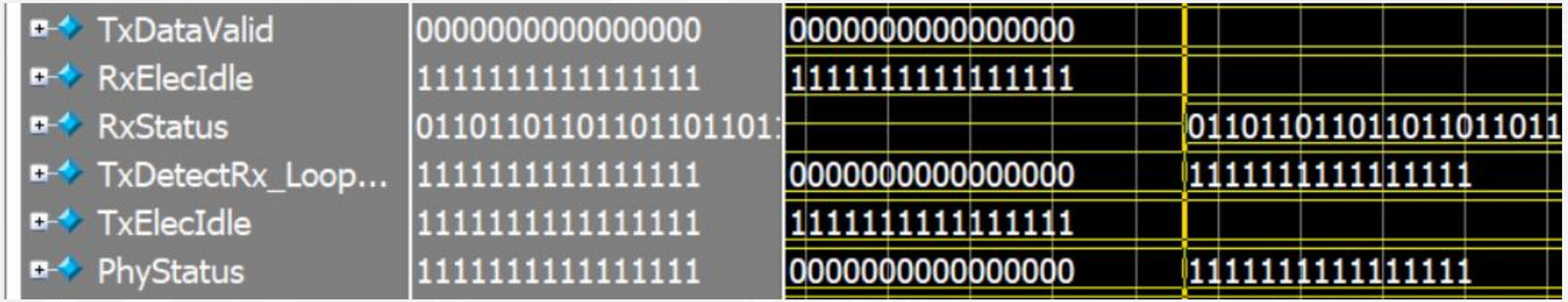
Step 2: Detect.Active

What happens here:

- The transmitter (Tx) now sends a signal and monitors the response to see if a receiver (Rx) is connected.
- It does this by sending a DC signal (common-mode voltage) and measuring how long it takes the line to charge.

Link UP Substates

Simulation

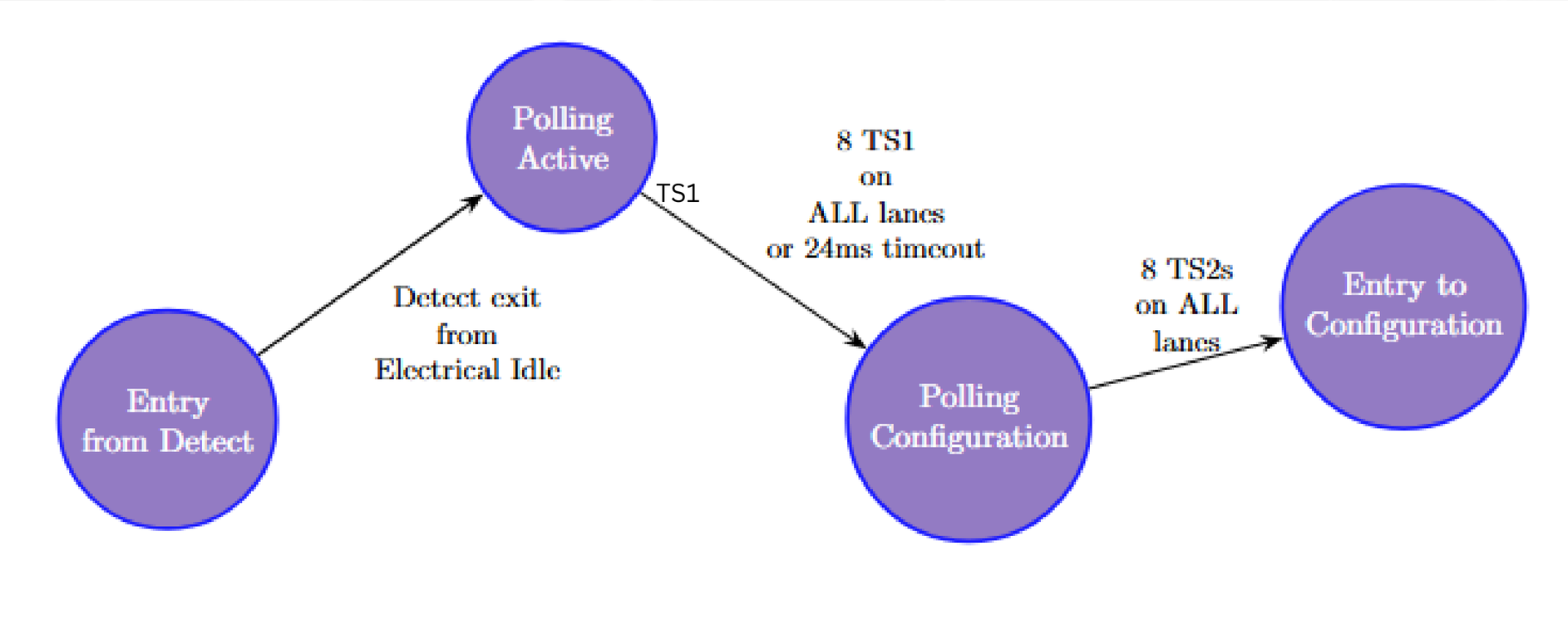



```
# UVM_INFO RX_Slave_D_Monitor.sv(1855) @ 120000: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Sl
ave_D_Monitor_h [RX_Slave_D_Monitor] Detect Active substate at Downstraem RX side completed succe
ssfully
# UVM_INFO TX_Slave_D_Monitor.sv(466) @ 120000: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX_Sla
ve_D_Monitor_h [TX_Slave_D_Monitor] Detect Active substate at Downstream TX side completed succes
sfully
```

```
# UVM_INFO PCIe_Scoreboard1_D.sv(703) @ 120000: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D_h [PCI
e_Scoreboard1_D] Downstream Detect_Active Approved
```

Link UP

Substates



Link UP Substates

Main Goal

To let both devices talk to each other and make sure they are ready for communication (bit lock, symbol lock)

Step 1: Polling.Active – “Start Talking & Locking In”

What happens here:

- Each device sends a sequence of TS1 ordered sets (Training Sequences).
- Devices must send at least 1024 TS1s on every active lane

Why?:

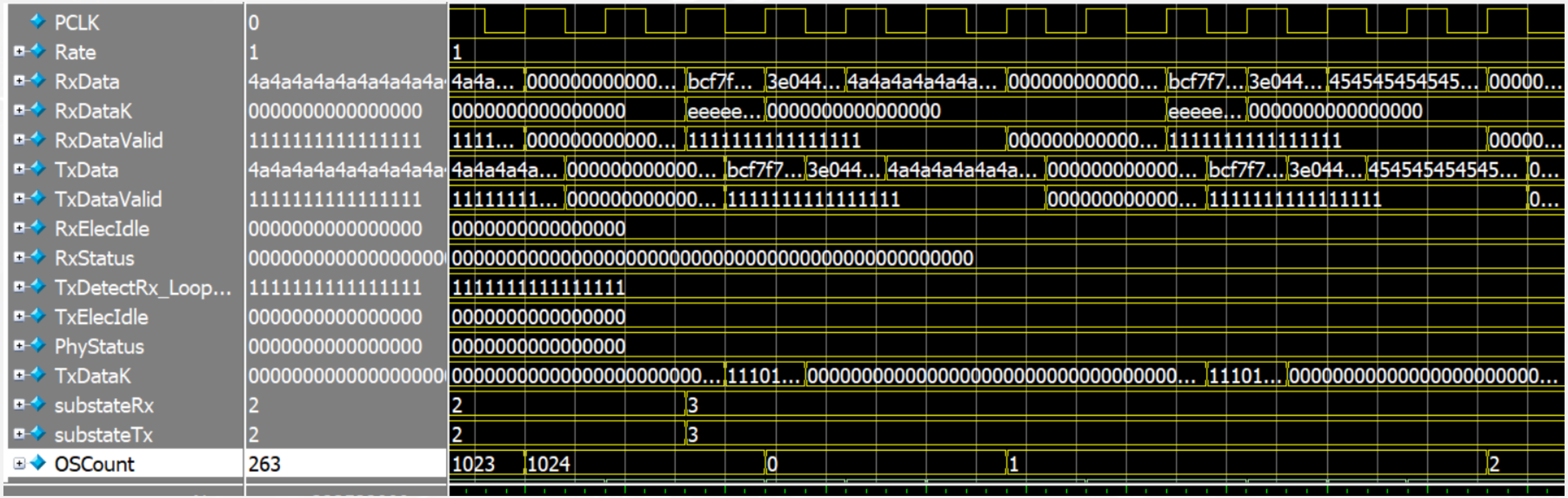
- Achieve Bit Lock (aligning to the bitstream)
- Achieve Symbol Lock or Block Alignment (understanding packet boundaries)

If the receiver gets 8 correct TS1s

Go to Polling.Configuration

Link UP Substates

Simulation




```
# UVM_INFO TX_Slave_D_Monitor.sv(939) @ 98504000: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX_S
lave_D_Monitor_h [TX_Slave_D_Monitor] Polling_Active substate at Downstream TX side completed suc
cessfully
# UVM_INFO RX_Slave_D_Monitor.sv(1884) @ 98504000: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_
Slave_D_Monitor_h [RX_Slave_D_Monitor] Polling_Active substate at Downstraem RX side completed su
ccessfully
```

Link UP

Substates

Step 2: Polling.Configuration – “Handshake Mode”

What happens here:

- The device stops sending TS1s and starts sending TS2s.
- Both devices must send AND receive TS2s to proceed.

Why?

This is a mutual handshake. You don’t go forward unless:

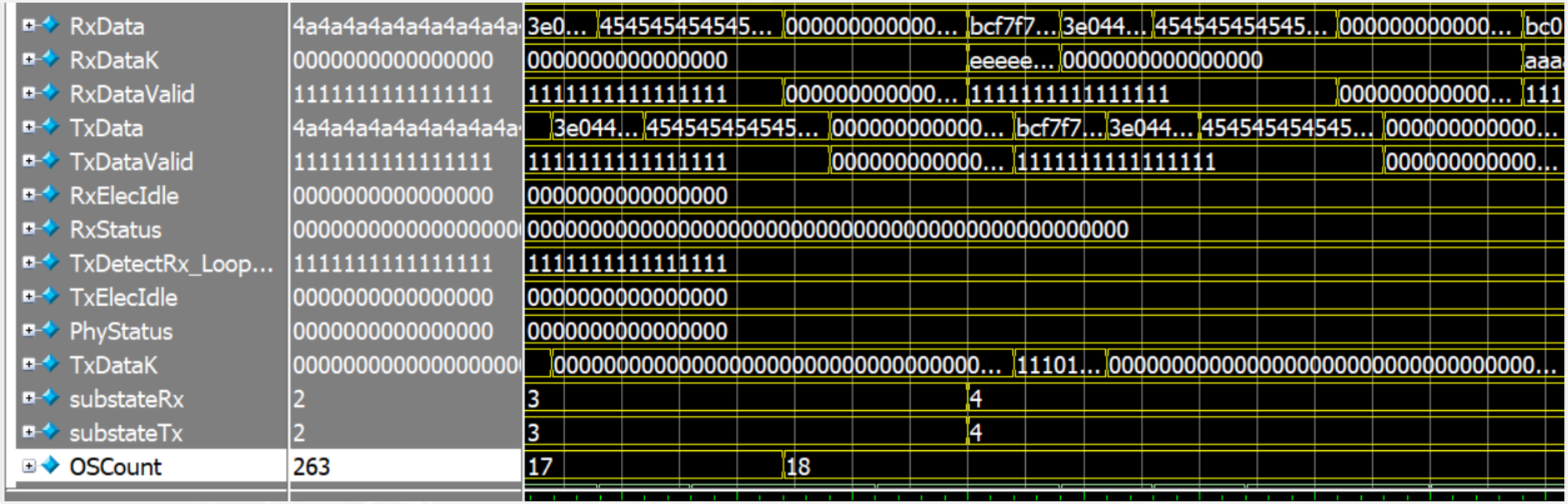
- You are ready
- And you know the other side is ready

Exit from Polling.Configuration

If a device receives 8 TS2s and has sent at least 16 TS2s

Link UP Substates

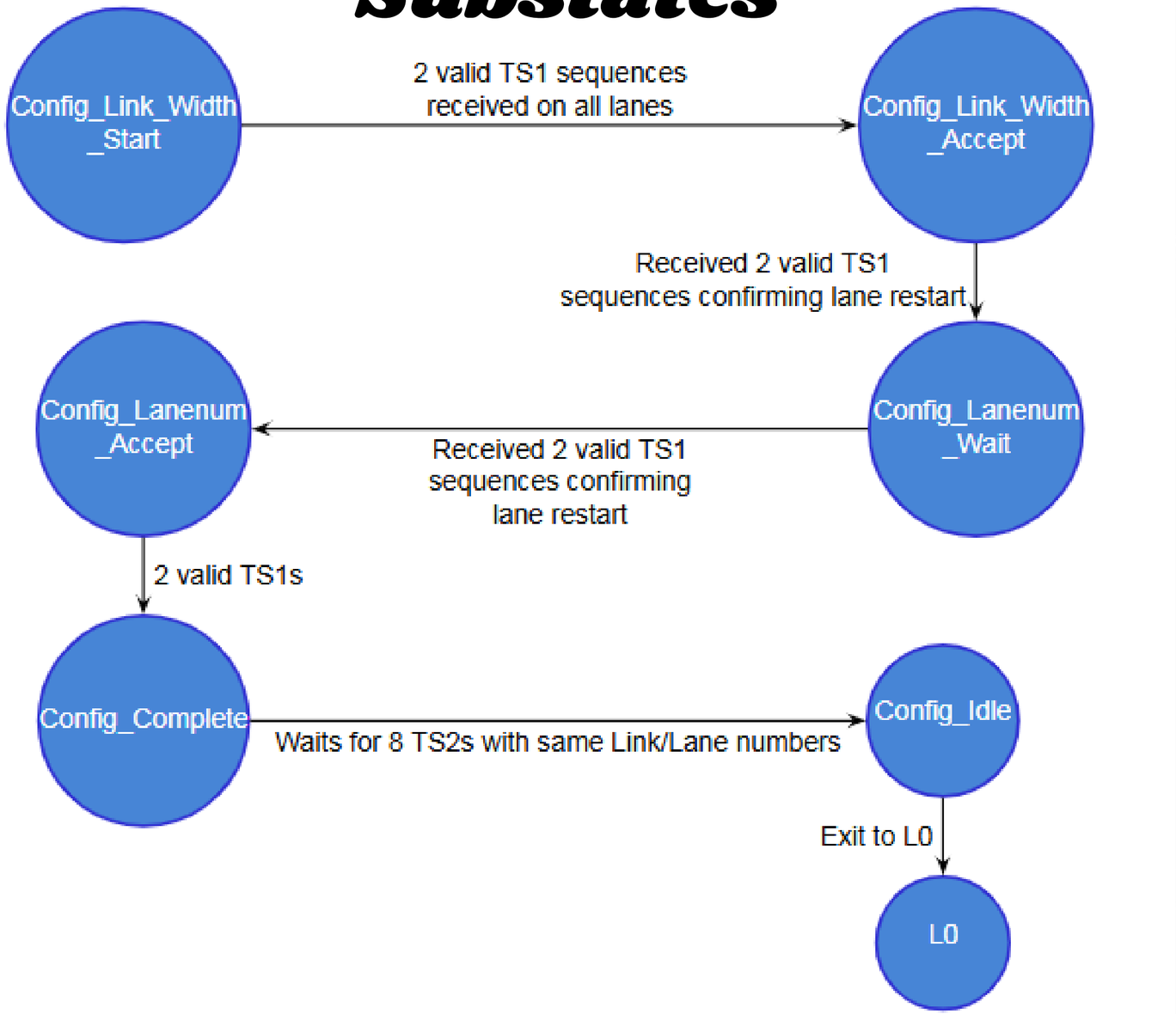
Simulation




```
# UVM_INFO TX_Slave_U_Monitor.sv(1140) @ 100248000: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_h.TX
_Slave_U_Monitor_h [TX_Slave_U_Monitor] Polling Configuration substate at Upstream TX side comple
ted successfully
# UVM_INFO PCIe_Scoreboard1_U.sv(772) @ 100248000: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U_h [
PCIe_Scoreboard1_U] Upstream Polling_Configuration Approved
```

Link UP

Substates



Link UP

Substates

Main Goal

To let both devices assign the Link number and Lane numbers and Ensure readiness to begin normal data transmission.

Step 1: **Configuration.Linkwidth.Start**

What happens here:

- Begin Link Number assignment.
- Downstream (leader) sets the Link number for all active lanes.
- Upstream (follower) keeps sending PADs until it receives proper link info.

Link UP Substates

Configuration.Linkwidth.Start

Down

Sends TS1 Ordered Sets with:

- Non-PAD Link number
- PAD Lane number
- Sends on all active lanes

Up

Keeps sending TS1s with PADs (both link and lane numbers) until:

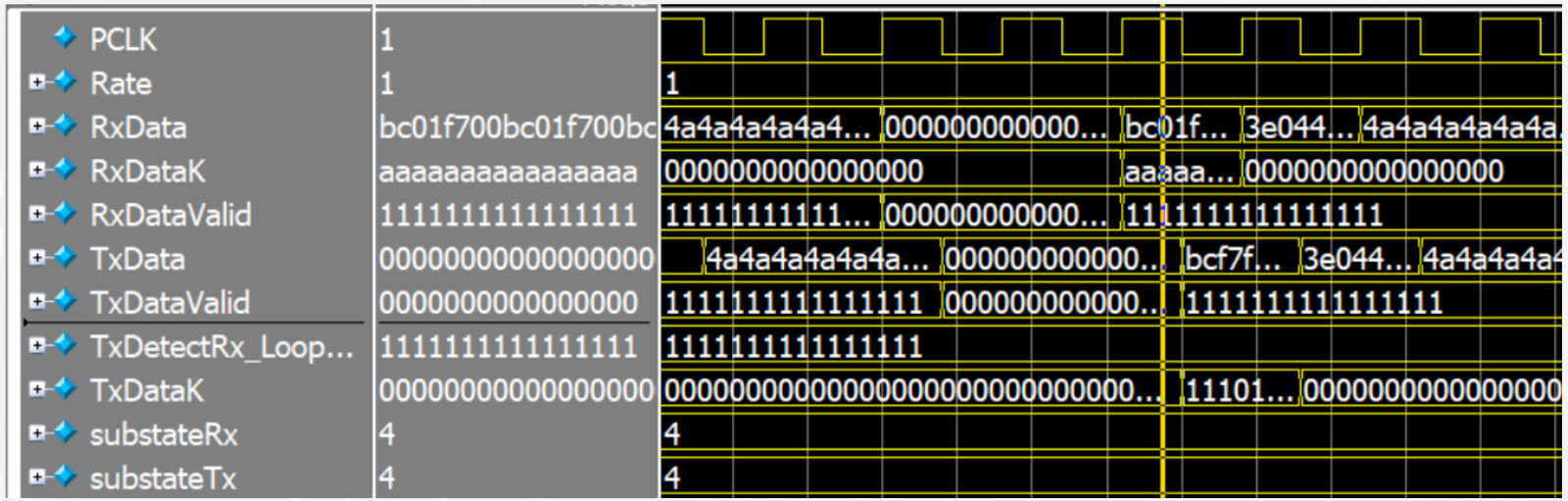
- It receives TS1s with non-PAD Link number from downstream.

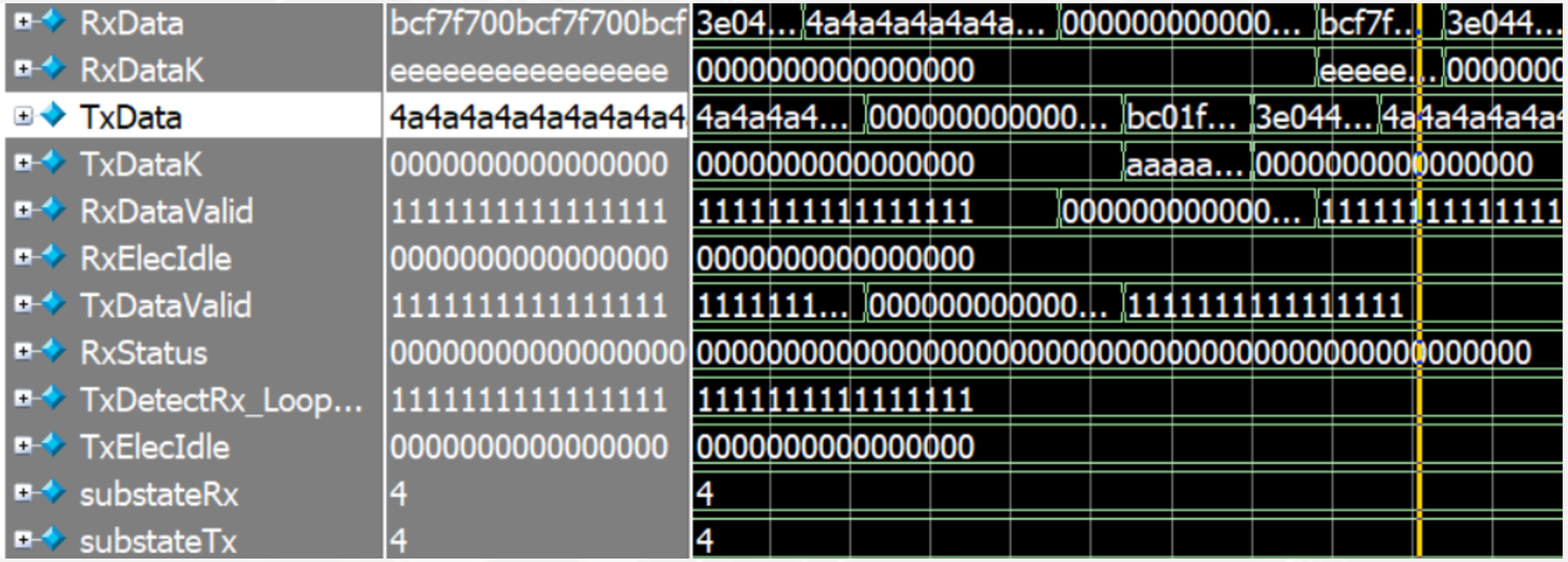
```
# UVM_INFO RX_Slave_U_Monitor.sv(2060) @ 100520001: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX
_Slave_U_Monitor_h [RX_Slave_U_Monitor] Config Link Width Start substate at Upstream RX side comp
leted successfully
# UVM_INFO TX_Slave_U_Monitor.sv(1257) @ 100576000: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_h.TX
_Slave_U_Monitor_h [TX_Slave_U_Monitor] Config_Link_Width_Start substate at Upstream TX side comp
leted successfully
```

Link UP Substates

Simulation

UP





Down

Link UP

Substates

Step 2: **Configuration.Linkwidth.Accept**

What happens here:

- In this substate, the Upstream Port (follower) reflects the Link number it received from the Downstream Port, and the Downstream Port (leader) uses that confirmation to determine the Link width (i.e., how many lanes are responding with the same Link number).
- Once this is confirmed, the Downstream Port begins assigning Lane numbers.

Link UP Substates

Configuration.Linkwidth.Accept

Down

Keeps sending TS1s:

- Same non-PAD Link number (previously sent)

Up

For each lane that received a non-PAD Link number:

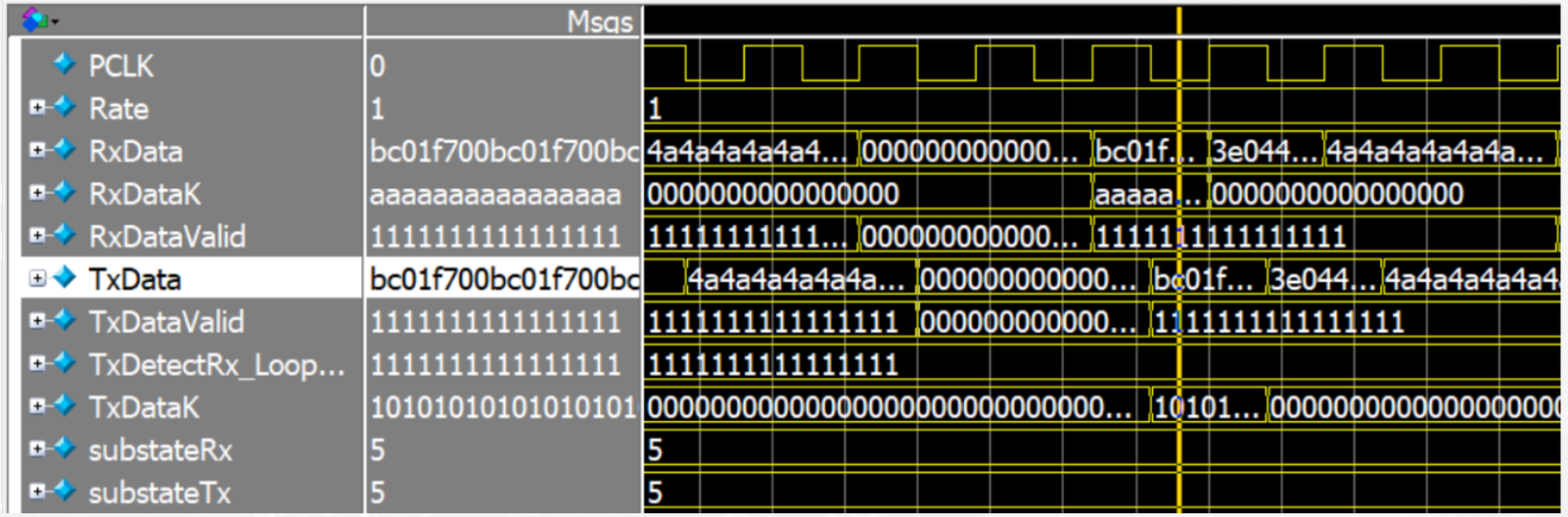
- Sends back TS1s with the same Link number, but still PAD Lane numbers.

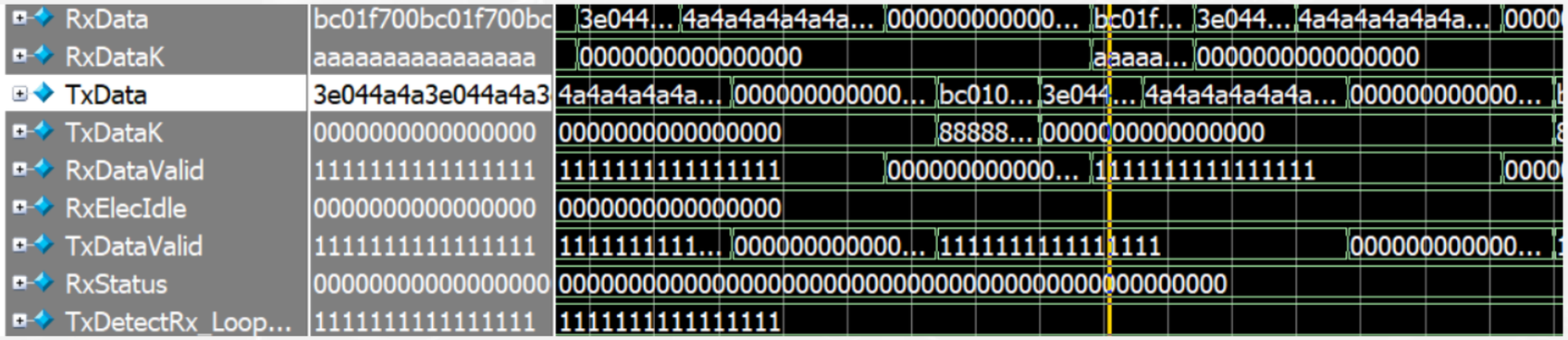
```
# UVM_INFO RX_Slave_U_Monitor.sv(2105) @ 101096001: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX
_Slave_U_Monitor_h [RX_Slave_U_Monitor] Config Link Width Accept substate at Upstream RX side com
pleted successfully
# UVM_INFO TX_Slave_U_Monitor.sv(1365) @ 101152000: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_h.TX
_Slave_U_Monitor_h [TX_Slave_U_Monitor] Config Link Width Accept substate at Upstream TX side com
pleted successfully
```

Link UP Substates

Simulation

UP





Down

Link UP

Substates

Step 3: **Configuration.Lanenum.Wait**

What happens here:

- This substate finalizes Lane numbering. Both ports (Downstream and Upstream) monitor the TS1s to confirm that they agree on both the Link number and the Lane numbers for all active lanes.

Link UP Substates

Configuration.Lanenum.Wait

Downstream

sending TS1s with:

- Same Link number
- Sequential Lane numbers

Up

Continues sending TS1s with:

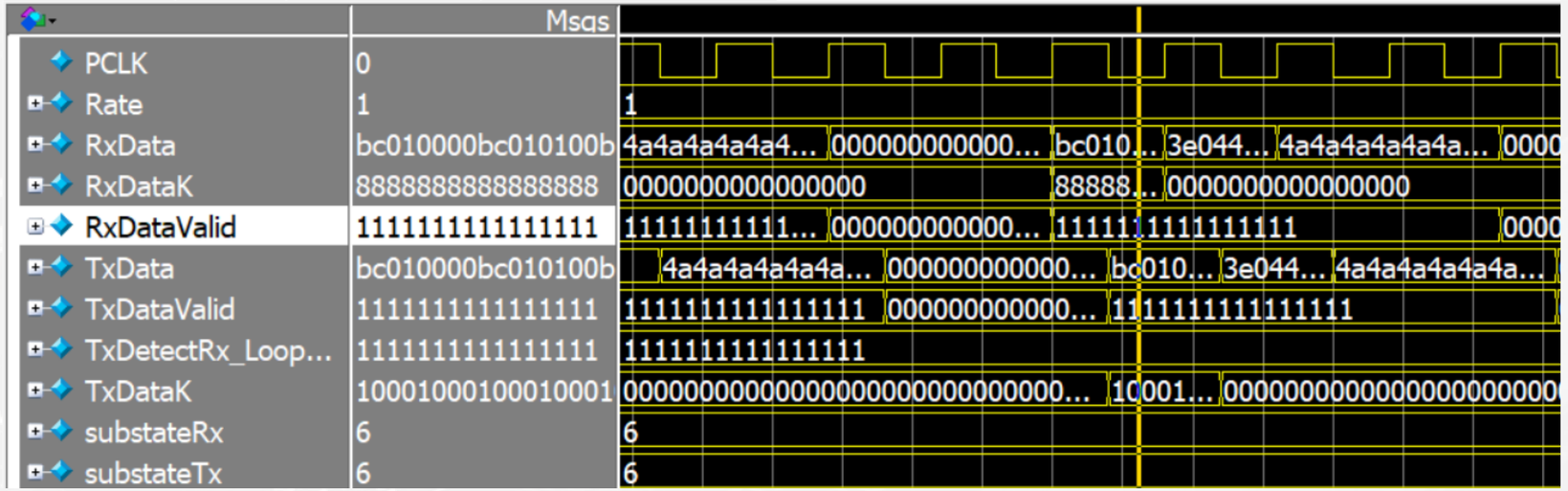
- Received Link number
- Waits for matching TS1s to confirm agreement.

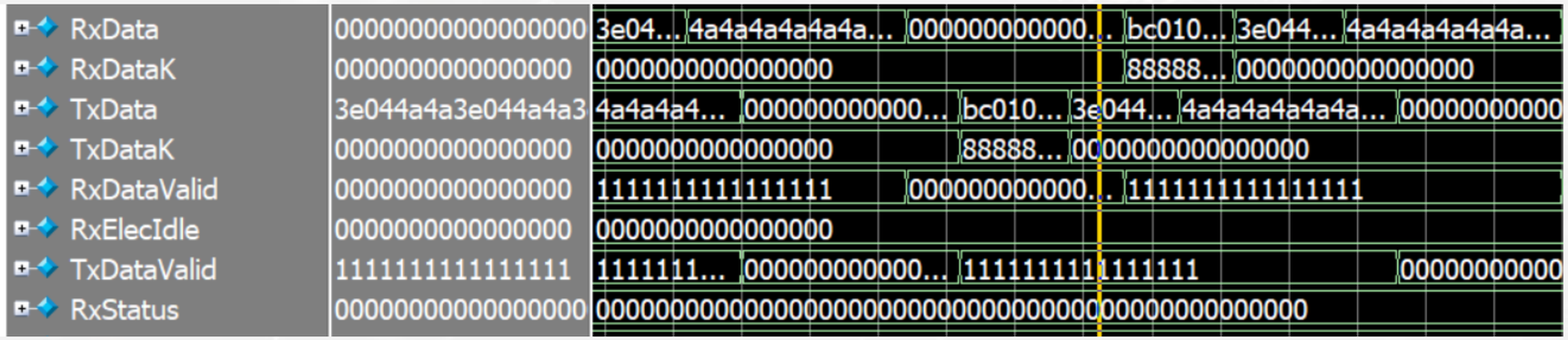
```
# UVM_INFO RX_Slave_D_Monitor.sv(2075) @ 101400001: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX
_Slave_D_Monitor_h [RX_Slave_D_Monitor]   Configuration Lanenum Wait substate at Downstraem RX si
de completed successfully
# UVM_INFO TX_Slave_D_Monitor.sv(1398) @ 101424000: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX
_Slave_D_Monitor_h [TX_Slave_D_Monitor] Config Lanenum Wait substate at Downstream TX side comple
ted successfully
```

Link UP Substates

Simulation

UP





Down

Link UP

Substates

Step 4: **Configuration.Lanenum.Accept**

What happens here:

- This substate confirms that both the Downstream and Upstream ports fully agree on the Link and Lane numbers. This is the final negotiation step before moving to Configuration.Complete, where TS2s will be exchanged.

Link UP Substates

Configuration.Lanenum.Accept

Down

Has received TS1s with non-PAD Link and Lane numbers.

Checks if:

- These values match what it was sending.

Up

- Has received TS1s with non-PAD Link and Lane numbers.

Validates that:

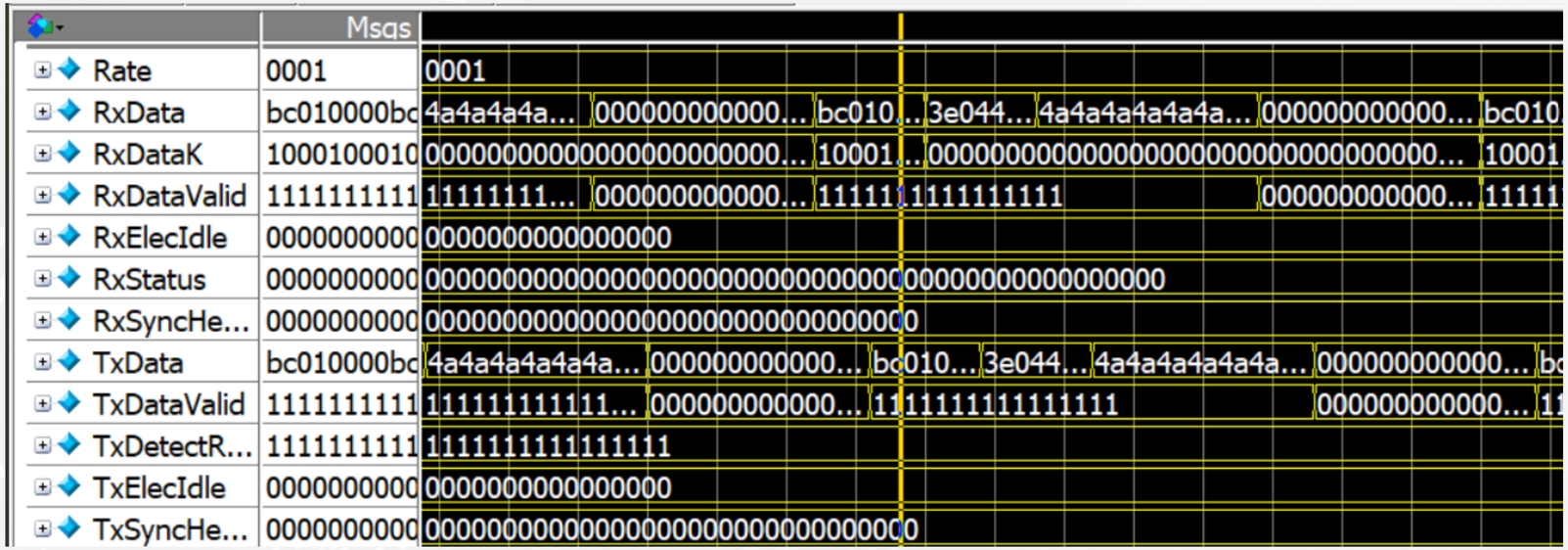
- The numbers match what it is sending.

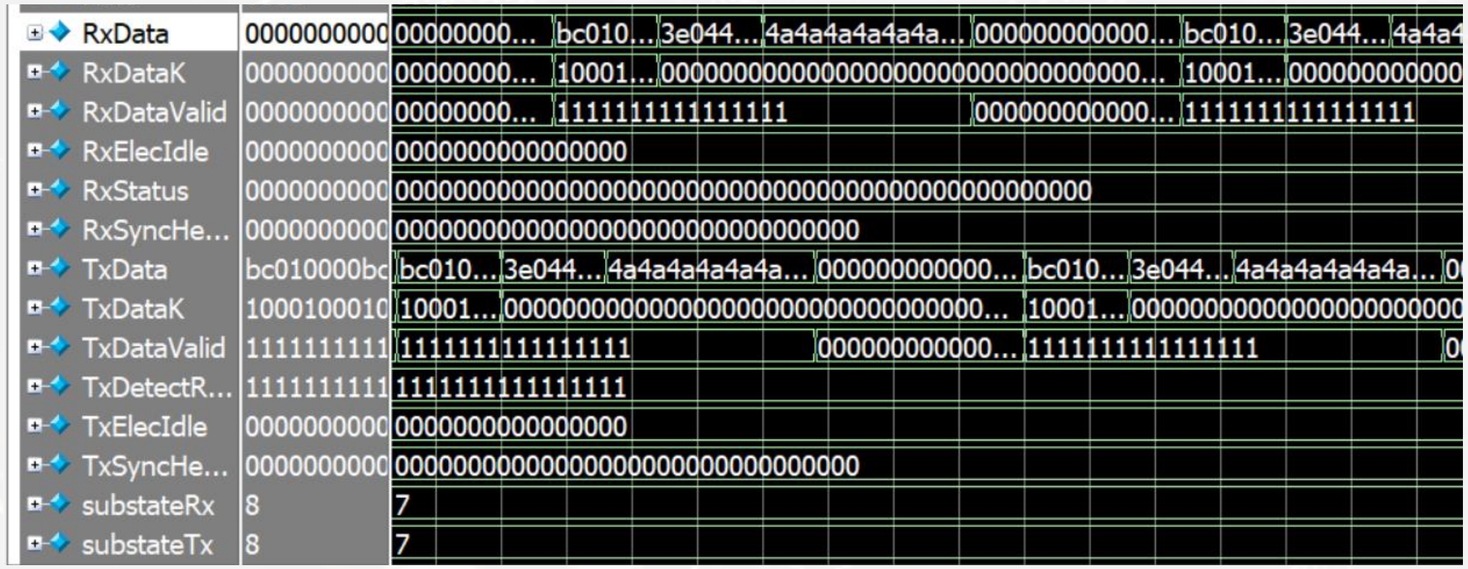
```
# UVM_INFO RX_Slave_D_Monitor.sv(2147) @ 101592001: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Slave_D_Monitor_h [RX_Slav
e_D_Monitor]  Configuration Lanenum Accept substate at Downstraem RX side completed successfully
# UVM_INFO TX_Slave_D_Monitor.sv(1505) @ 101616000: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX_Slave_D_Monitor_h [TX_Slav
e_D_Monitor] Config Lanenum Accept substate at Downstream TX side completed successfully
```

Link UP Substates

Simulation

UP





Down

Link UP

Substates

Step 5: **Configuration.Complete**

What happens here:

- This substate performs a final confirmation between the two link partners using TS2 Ordered Sets. TS2s confirm:
  - Agreed Link number
  - Agreed Lane numbers
  - Data rate capability

Only TS2s are sent and received in this phase — no TS1s anymore.

Link UP Substates

Configuration.Complet

Down

Sends TS2s with:

- Confirmed Link and Lane numbers (non-PAD)
- Optional Upconfigure Capable bit
- Waits for 8 matching TS2s on each lane.
- Must send at least 16 TS2s

Up

Sends TS2s with:

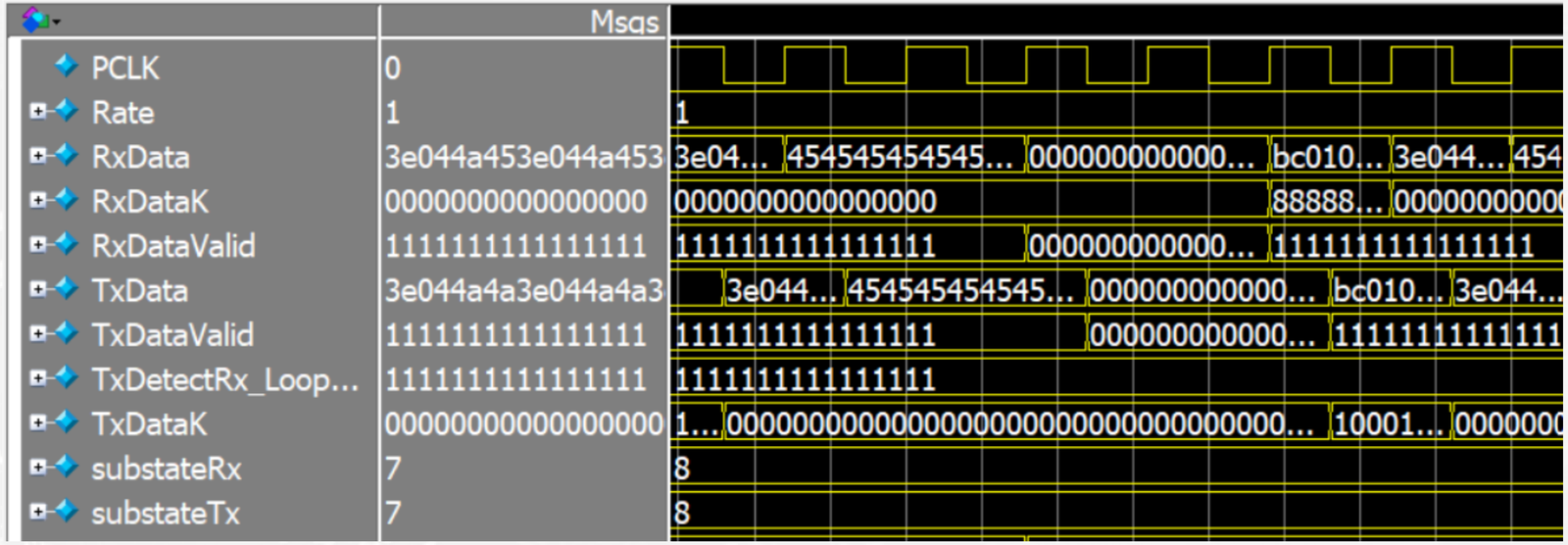
- Confirmed Link and Lane numbers
- Waits for 8 TS2s on all lanes with same Link/Lane numbers and capabilities.

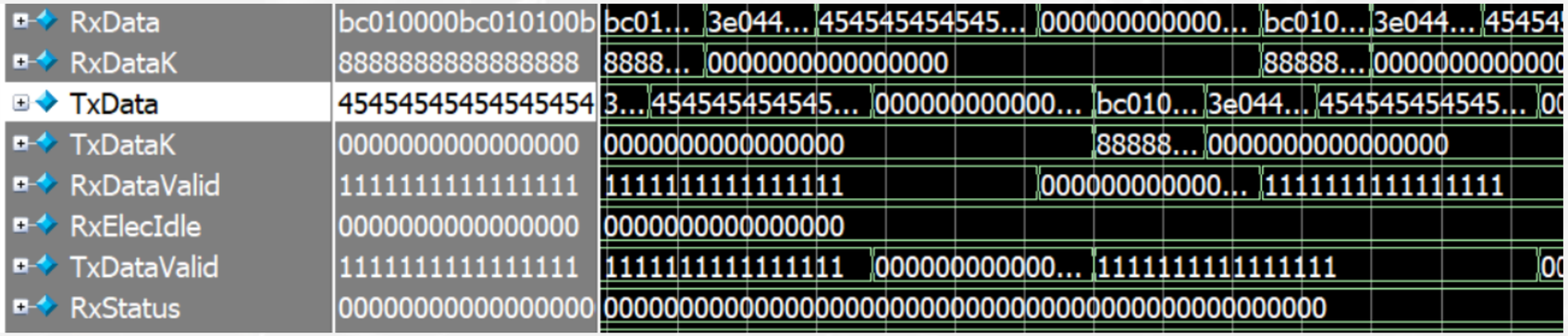
```
# UVM_INFO TX_Slave_U_Monitor.sv(1686) @ 103992000: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_h.TX_Slave_U_Monitor_h [TX_Slav
e_U_Monitor] Config Complete substate at Upstream TX side completed successfully
# UVM_INFO RX_Slave_U_Monitor.sv(2262) @ 103992000: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX_Slave_U_Monitor_h [RX_Slav
e_U_Monitor] Config Complete substate at Upstream RX side completed successfully
```

Link UP Substates

Simulation

UP





Down

Link UP

Substates

Step 5: **Configuration.Idle**

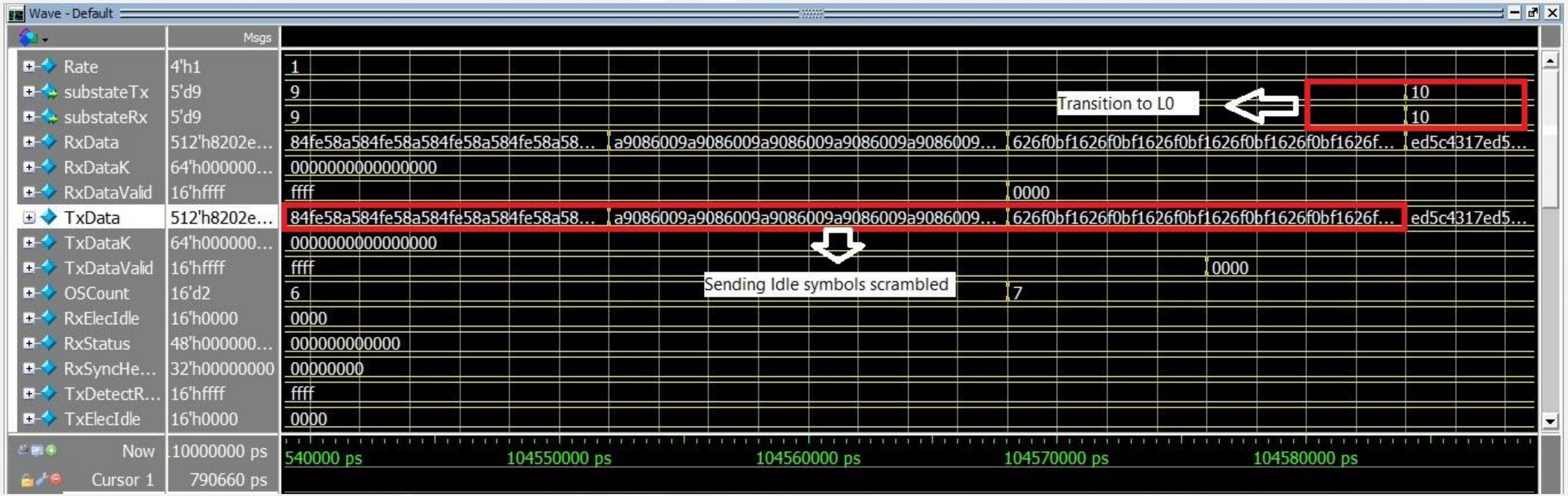
What happens here:

- This is the final holding state before entering L0 (the active data transmission state). Here, the physical layer:
  - Sends Idle symbols
  - Confirms link is stable
  - Finalizes that link-up is complete (LinkUp = 1b)

All lanes receive 8 consecutive Idle symbols and 16 Idles sent after receiving one ----------> L0

Link UP Substates

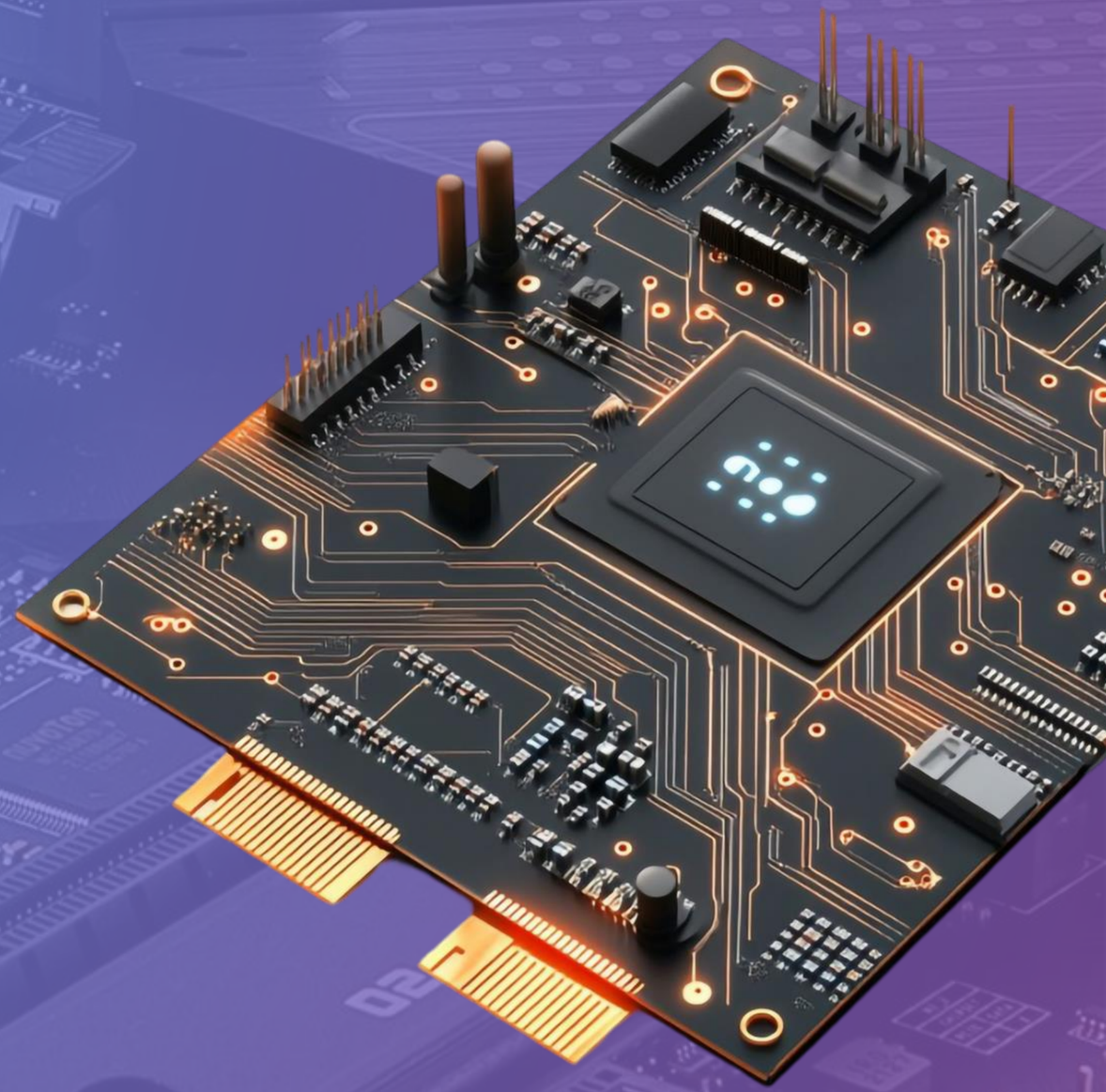
Simulation




```
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/TX_Slave_D_Monitor.sv(1656) @ 104184000: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX_Slave_D_Monitor_h [TX_Slave_D_M
onitor] Config Idle substate at Downstream TX side completed successfully
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/RX_Slave_D_Monitor.sv(2239) @ 104184000: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Slave_D_Monitor_h [RX_Slave_D_M
onitor]  Configuration Idle substate at Downstraem RX side completed successfully
```

06

Recovery States



Recovery states

Overview

Before diving into the Recovery states, let's ask a few important questions:

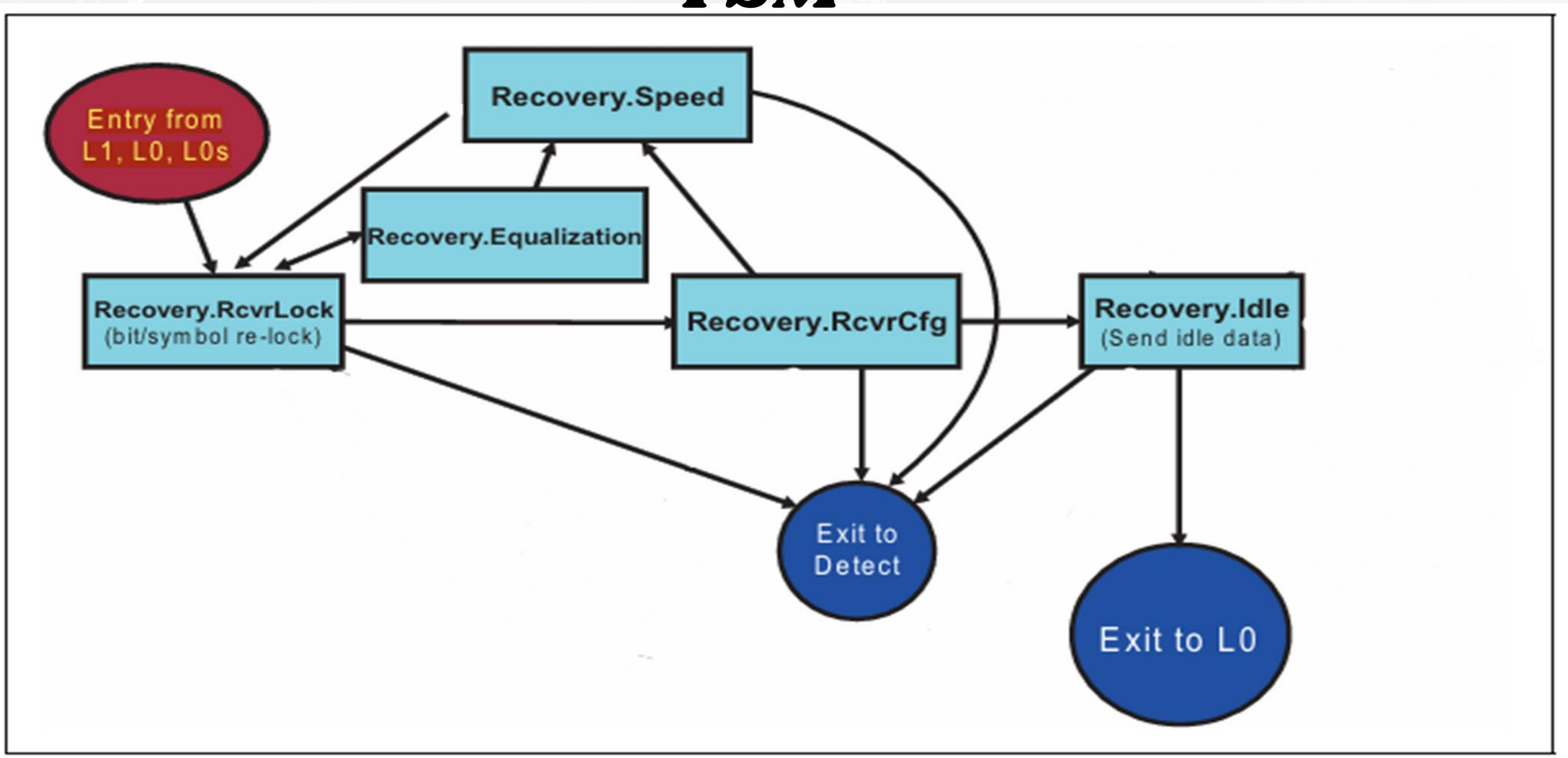
- Do all PCIe lanes always operate with perfect alignment and efficiency?
- Is there only one PCIe generation and fixed speed?
- Is the link between two devices always stable and permanent?
- Can the link maintain synchronization indefinitely under all conditions?

The answer to all these questions is: No.

Due to potential lane misalignment, speed mismatches, link instability, or changes in link conditions, PCIe requires a set of **Recovery states**

Recovery state

FSM



Recovery states

Recovery Lock

- It’s the first substate in the Recovery process — used to relock and align lanes between devices after a disruption or speed change.

- Devices exchange TS1 Ordered Sets with updated Link and Lane numbers and may signal a speed change.

Making it the foundation for verifying lane alignment, negotiating speed, and ensuring link quality.

Recovery Lock

- now we need to Initiate a link speed change during PCIe Link Training.

Requirement:

Send TS1s

Condition for successful detection:

- 8 consecutive TS1s must be received.

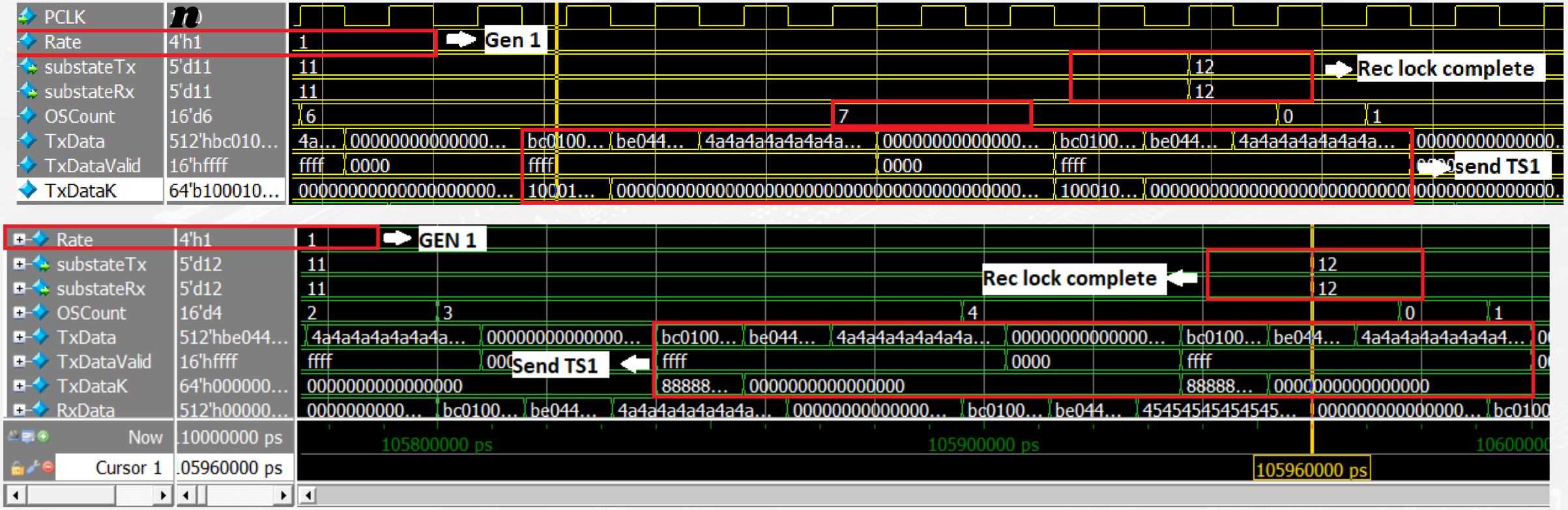
These must have:

- Matching Link and Lane numbers.
- The speed_change bit set (indicates the intention to change speed)

Recovery states

Recovery Lock

- Simulatio n



Recovery states

Recovery Lock

from the Transcript window in the simulation

we observe that the TX,Rx Monitor and Scoreboard have passed successfully indicating that 8 TS1s were sent with the correct field values and correct count.

```
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/TX_Slave_D_Monitor.sv(2009) @ 105992001: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX_Slave_D_Monitor_h [TX_Slave_D_M
onitor] Recovery.RcvrLock substate at Downstream TX side completed successfully
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/RX_Slave_D_Monitor.sv(2348) @ 105992001: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Slave_D_Monitor_h [RX_Slave_D_M
onitor] Recovery RcvrLock substate at Downstraem RX side completed successfully
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/PCIe_Scoreboard1_D.sv(737) @ 105992001: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D_h [PCIe_Scoreboard1_D] Downstream R
ecovery RcvrLock Approved
```

```
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/TX_Slave_U_Monitor.sv(2049) @ 105848001: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_h.TX_Slave_U_Monitor_h [TX_Slave_U_M
onitor] Recovery.RcvrLock substate at Upstream TX side completed successfully
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/RX_Slave_U_Monitor.sv(2397) @ 105848001: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX_Slave_U_Monitor_h [RX_Slave_U_M
onitor] Recovery.RcvrLock substate at Upstream RX side completed successfully
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/PCIe_Scoreboard1_U.sv(796) @ 105848001: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U_h [PCIe_Scoreboard1_U] Upstream_Rec
overy RcvrLock Approved
```

Recovery CFG

After achieving bit lock and symbol/block lock in **Recovery RcvrLock**, **Recovery RcvrCfg** checks why Recovery was entered and what action is needed next.

Determine the reason for entering Recovery : -

- If it's for a speed change → transition to Recovery.Speed. **(our Foucs)**
- If it's for a link width change → transition to the appropriate lane configuration substate
- If it's just to re-lock after exiting low-power (L1) → transition to Recovery.Idle.

Recovery states

Recovery CFG

- now we need to Initiate a link speed change to highest common speed.

Requirement:

- Transmit: TS2 Ordered Sets .

Condition for successful detection:

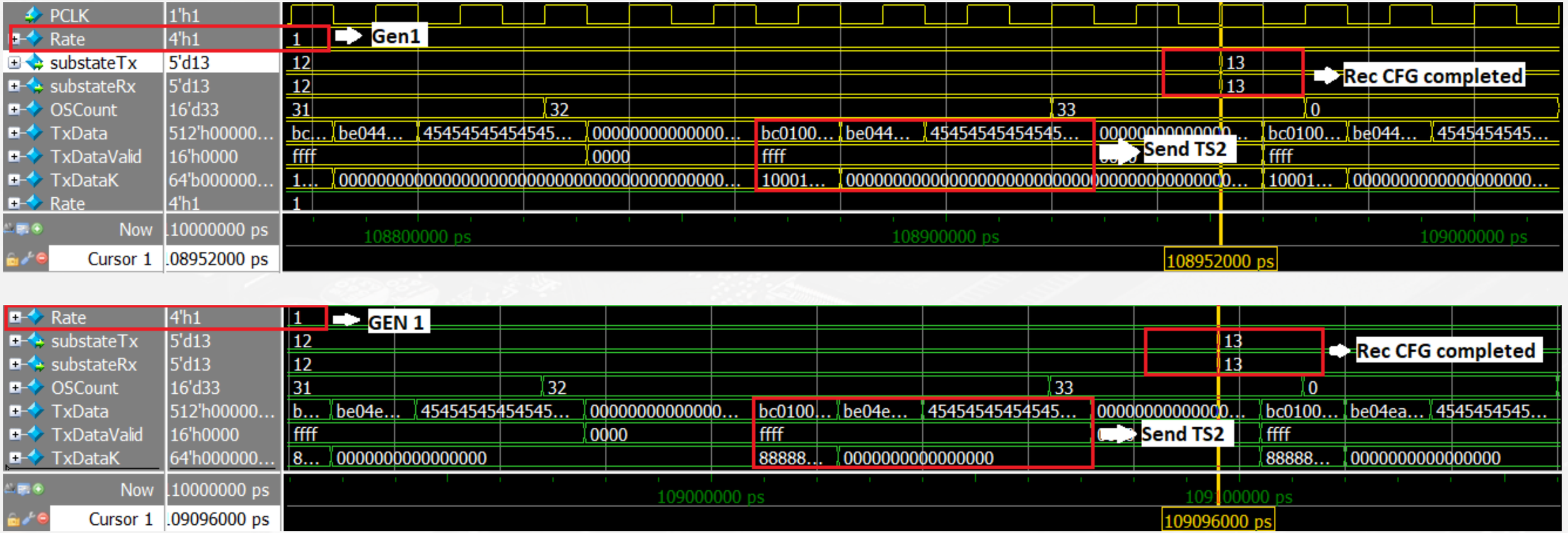
- 32 consecutive TS2s must be Sent on any configured lane.

These must have:

- Matching Link and Lane numbers
- The speed_change bit set (indicates the intention to change speed)
- Identical Rate Identifiers

Recovery states

Recovery CFG



Recovery states

Recovery CFG

from the Transcript window in the simulation

we observe that the TX ,Rx Monitors and Scoreboard have passed successfully indicating that 32 TS2s were sent with the correct field values and correct count.

```
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/TX_Slave_D_Monitor.sv(2373) @ 109064001: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX_Slave_D_Monitor_h [TX_Slave_D_M
onitor] Recovery.RcvrCfg substate at Downstream TX side completed successfully
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/RX_Slave_D_Monitor.sv(2416) @ 106616001: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Slave_D_Monitor_h [RX_Slave_D_M
onitor]  Recovery.RcvrCfg  substate at Downstraem RX side completed successfully

# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/TX_Slave_U_Monitor.sv(2377) @ 108920001: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_h.TX_Slave_U_Monitor_h [TX_Slave_U_M
onitor] Recovery.RcvrCfg substate at Upstream TX side completed successfully
# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/RX_Slave_U_Monitor.sv(2475) @ 106760001: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX_Slave_U_Monitor_h [RX_Slave_U_M
onitor] Recovery.RcvrCfg substate at Upstream RX side completed successfully

# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/PCIe_Scoreboard1_D.sv(740) @ 109064001: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D_h [PCIe_Scoreboard1_D] Downstream R
ecovery_RcvrCfg_Approved

# UVM_INFO E:/(Faculty of Engineering, Alexandria University)/Level 4/Graduation Project/Term 2/Last Env/Grad_BY_Youssef/
TB/PCIe_Scoreboard1_U.sv(799) @ 108920001: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U_h [PCIe_Scoreboard1_U] Upstream Rec
overy_RcvrCfg_Approved
```

Recovery speed

- Recovery.Speed ensures both devices safely transition to a new link speed during PCIe link training by resetting the physical layer

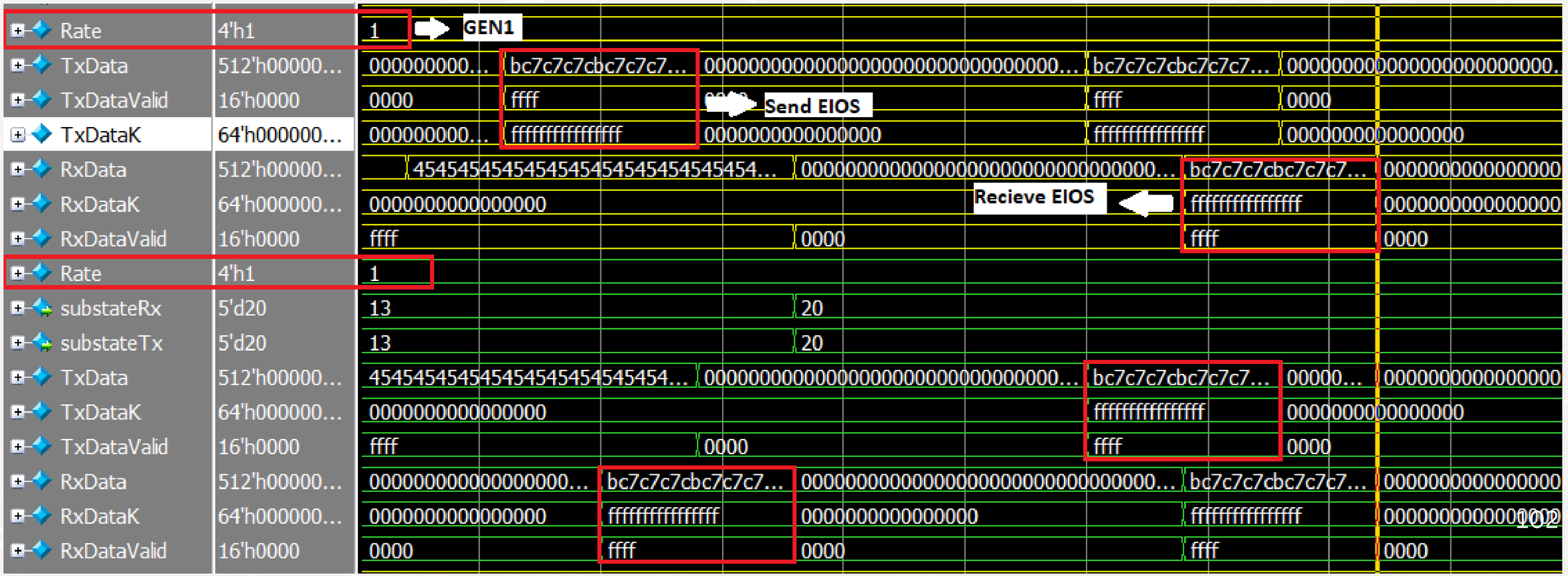
- The transmitter enters Electrical Idle and waits for the receiver to also go idle
- If the speed negotiation fails after at most 1 ms, it returns to Recovery RcvrLock without changing the speed.

- If the speed negotiation is successful, both sides send EIEOS and exit the Recovery Speed substate, transitioning back to Recovery.RcvrLock

Recovery states

Recovery speed

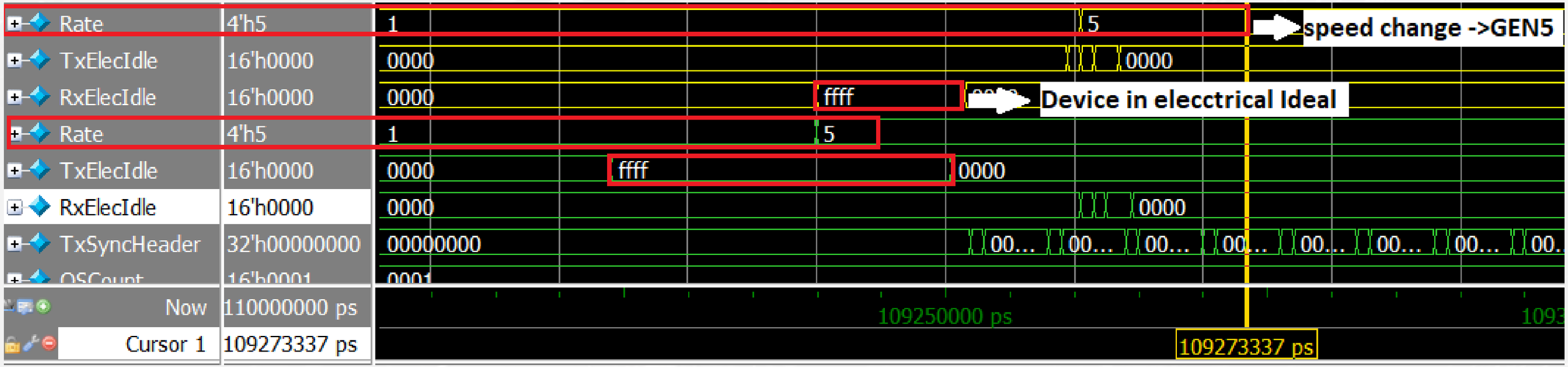
- As we can see, both devices begin sending EIOS to enter the Idle state during link training.



Recovery states

Recovery speed

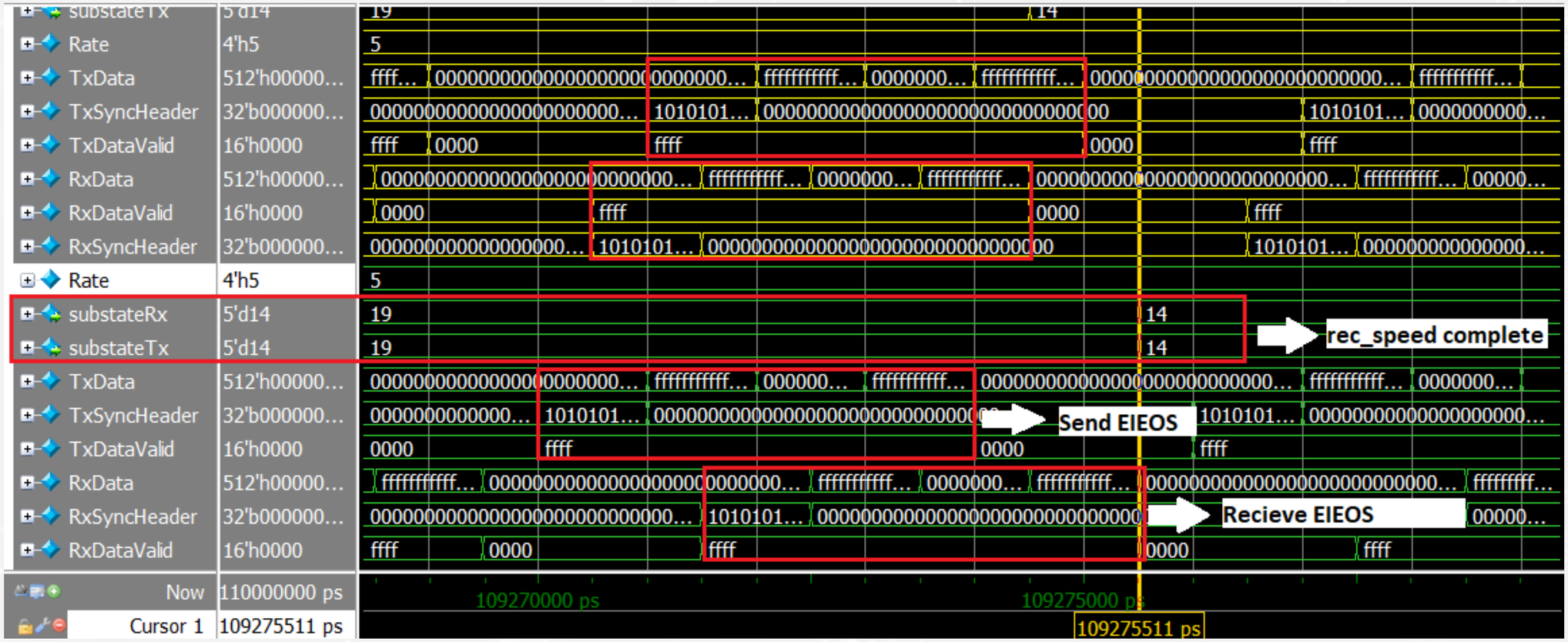
- Now, the speed change has completed successfully, as shown in the figure.



Recovery states

Recovery speed

- Now, both devices begin sending EIEOS to exit the Idle state and transition back to Recovery.RcvrLock for lane re-alignment at the new speed



Recovery states

Recovery speed

from the Transcript window in the simulation

we observe that the TX ,Rx Monitors and Scoreboard have passed successfully indicating that EIOS and EIEOSs were sent with the correct field values and count.

```
# UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_D_Monitor.sv(2563) @ 109267500: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_
h.TX_Slave_D_Monitor_h [TX_Slave_D_Monitor] Recovery.Speed substate at Downstream TX side completed successfully

# UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_D_Monitor.sv(2599) @ 109276500: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_
h.RX_Slave_D_Monitor_h [RX_Slave_D_Monitor] Recovery Speed substate at Downstraem RX side completed successfully

# UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_U_Monitor.sv(2554) @ 109274500: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_
h.TX_Slave_U_Monitor_h [TX_Slave_U_Monitor] Recovery.Speed substate at Upstream TX side completed successfully

# UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_U_Monitor.sv(2553) @ 109263500: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_
h.RX_Slave_U_Monitor_h [RX_Slave_U_Monitor] Recovery.Speed substate at Upstream RX side completed successfully
```

```
# UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_D.sv(734) @ 109276500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D
_h [PCIe_Scoreboard1_D] Downstream Recovery_Speed Approved

# UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_U.sv(808) @ 109274500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U
_h [PCIe_Scoreboard1_U] Upstream Recovery_Speed Approved
```

Recovery states

Recovery Lock

- After returning to the Recovery.RcvrLock state, the link begins the process of re-establishing communication.
- The goal is to evaluate signal quality to determine if it is sufficient for reliable data transmission.
- If the signal quality check is required, the link transitions to the Equalization states to perform fine-tuning and optimization.

Recovery states

Equalization states

- Recovery.Equalization is required for PCIe Gen3 and above, including Gen5 (8.0 GT/s), to ensure proper signal integrity.
- It allows transmitter and receiver to negotiate equalization settings to compensate for high-speed channel loss.
- This substate is not used at lower speeds like 2.5 GT/s or 5.0 GT/s.
- Even at 8.0 GT/s, Recovery.Equalization is only entered when certain conditions are met (e.g., speed change to Gen3+).
- Its main goal is to ensure the link is electrically stable before normal operation begins.

Recovery states

Phase 0

- The Upstream Port sends TS1s with Equalization Control (EC) = 00b.
- It uses the Tx Preset values received in the EQ TS2s (from the Downstream Port).
- The TS1s must include:
  - The Tx Preset value used.
  - The corresponding Pre-cursor, Cursor, and Post-cursor coefficients.

Since the Training Sequences (TS) are scrambled in Gen3 and later, we implemented a descrambler function in our environment to recover the original TS content. This allows us to accurately extract and verify fields like the EC bits for correctness.

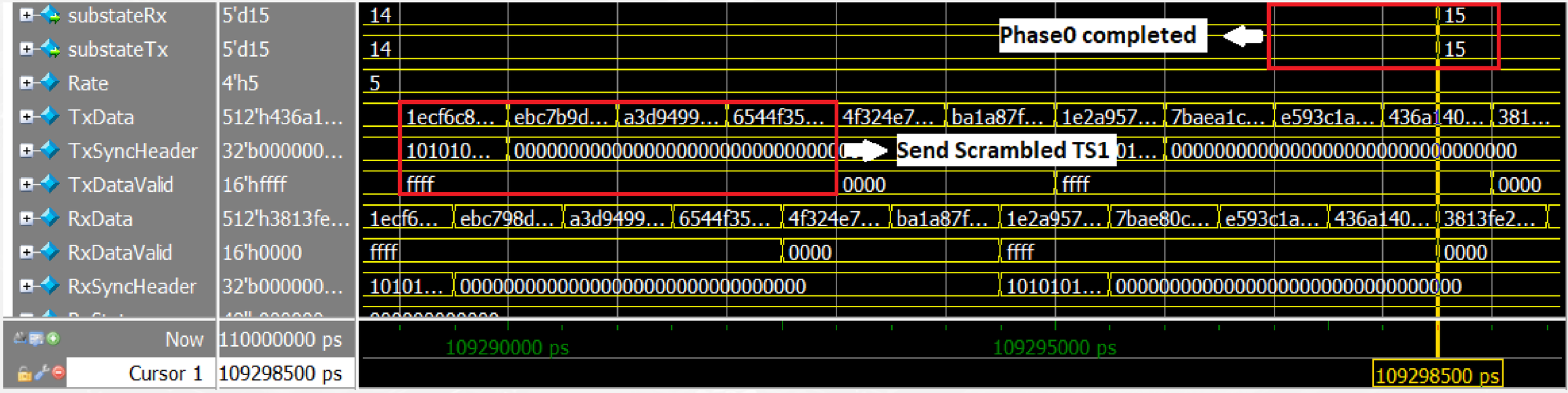
```
Lane_Datades = PIPE_vif_h.TxData[i*`MAXPIPEWIDTH +: `MAXPIPEWIDTH];

Descrambled_Data = apply_descramble(de_scrambler,Lane_Datades,i,`GEN5);

All_data = All_data ^ ( Descrambled_Data << (`MAXPIPEWIDTH * i) );
```

Recovery states

Phase 0



The Receiver receives the Tx Preset from the Downstream Port and responds with feedback to initiate the transition to Phase 1.

Recovery states

➢ Phase 0

from the Transcript window in the simulation

we observe that the TX ,Rx Monitors and Scoreboard in upstream device have passed successfully indicating that receive TS1 with EC bit =00 .

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_U_Monitor.sv(2669) @ 109295000: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_
TX_Slave_U_Monitor_h [TX_Slave_U_Monitor] Phase0 substate at Upstream TX side completed successfully
UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_U_Monitor.sv(2646) @ 109291501: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_
RX_Slave_U_Monitor_h [RX_Slave_U_Monitor] Phase0 substate at Upstream RX side completed successfully
```

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_U.sv(802) @ 109295000: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U
 [PCIe_Scoreboard1_U] Upstream Phase0 Approved
```

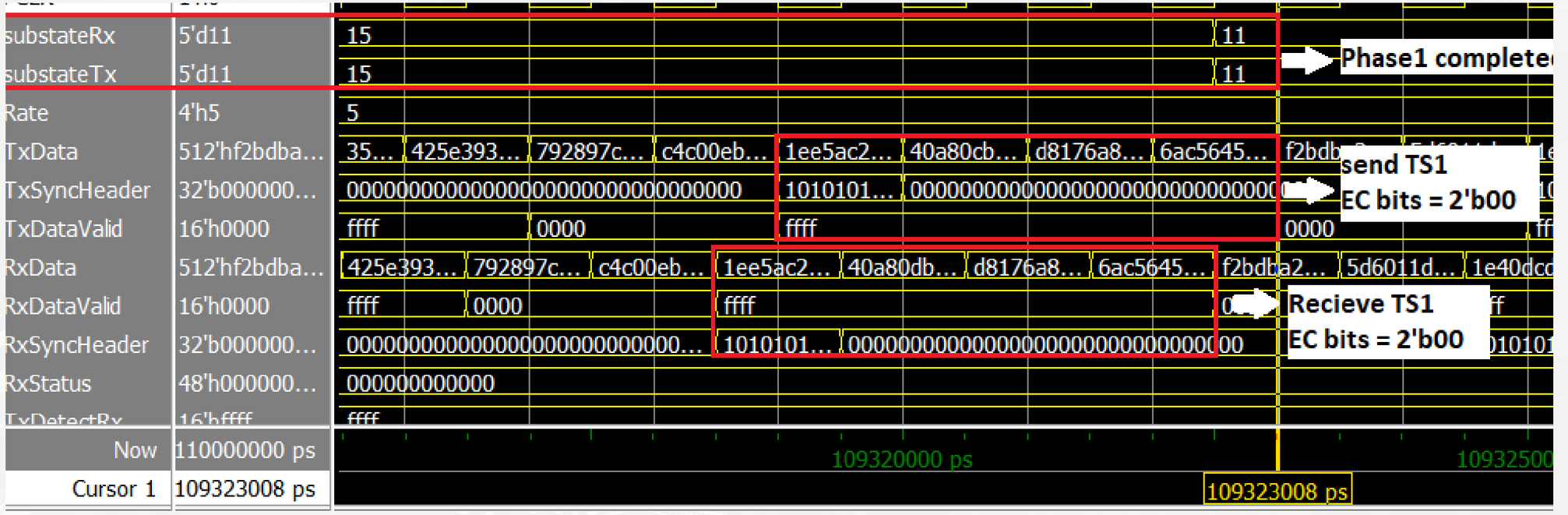
Recovery states

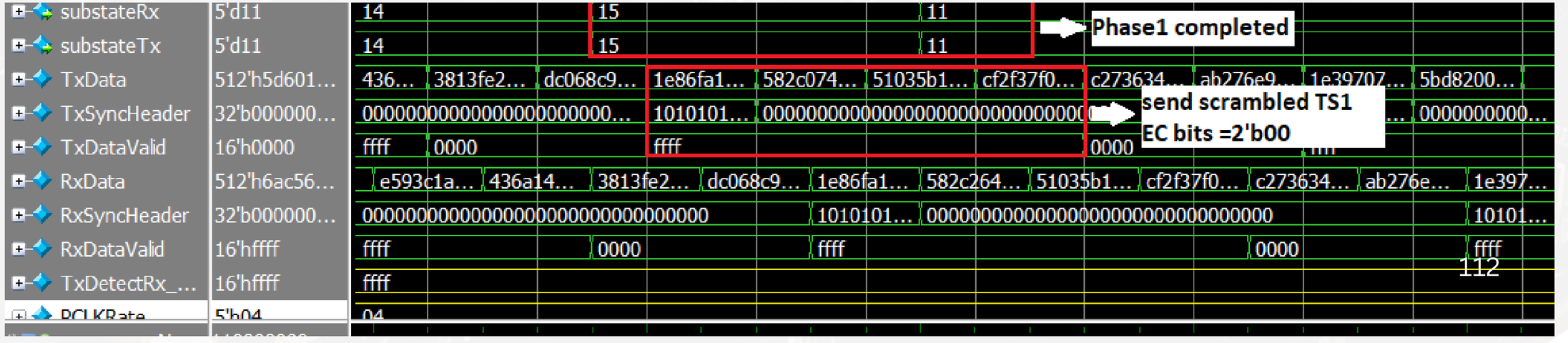
Phase 1

- Both Upstream and Downstream devices start sending TS (Training Sequences) with EC bits = 01, indicating the start of signal quality evaluation.
- If the signal quality is acceptable:
- Downstream device transitions by sending TS with EC bits = 00, signaling the end of equalization.
- This also initiates the move to the Recovery RcvrLock state.

Recovery states

Phase 1





Recovery states

➢ Phase 1

from the Transcript window in the simulation

we observe that the TX ,Rx Monitors and Scoreboard have passed successfully indicating that receive TS1 with EC bit =00 .

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_D_Monitor.sv(2721) @ 109294000: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_
TX_Slave_D_Monitor_h [TX_Slave_D_Monitor] Phase1 substate at Downstream TX side completed successfully

UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_U_Monitor.sv(2694) @ 109315501: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_
RX_Slave_U_Monitor_h [RX_Slave_U_Monitor] Phase1 substate at Upstream RX side completed successfully
UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_U_Monitor.sv(2805) @ 109325000: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_
TX_Slave_U_Monitor_h [TX_Slave_U_Monitor] Phase1 substate at Upstream TX side completed successfully

RX_Slave_D_Monitor_h [RX_Slave_D_Monitor] Phase1 substate at Downstraem RX side completed successfully
UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_D.sv(743) @ 109298501: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D
```

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_D.sv(743) @ 109298501: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D
 [PCIe_Scoreboard1_D] Downstream Phase1 Approved

UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_U.sv(805) @ 109325000: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U
 [PCIe_Scoreboard1_U] Upstream Phase1 Approved
```

Recovery Lock in GEN5

- In this substate, the goal is to transition to the L0 state.
- Devices begin sending at least 8 consecutive TS1s, as required for the transition.
- These TS1s are now scrambled, following the protocol.
- A descrambler function is used in the environment to decode the TS1s.
- This allows verification that the Lane Number, Link Number, and other critical fields remain accurate.

Recovery CFG in GEN5

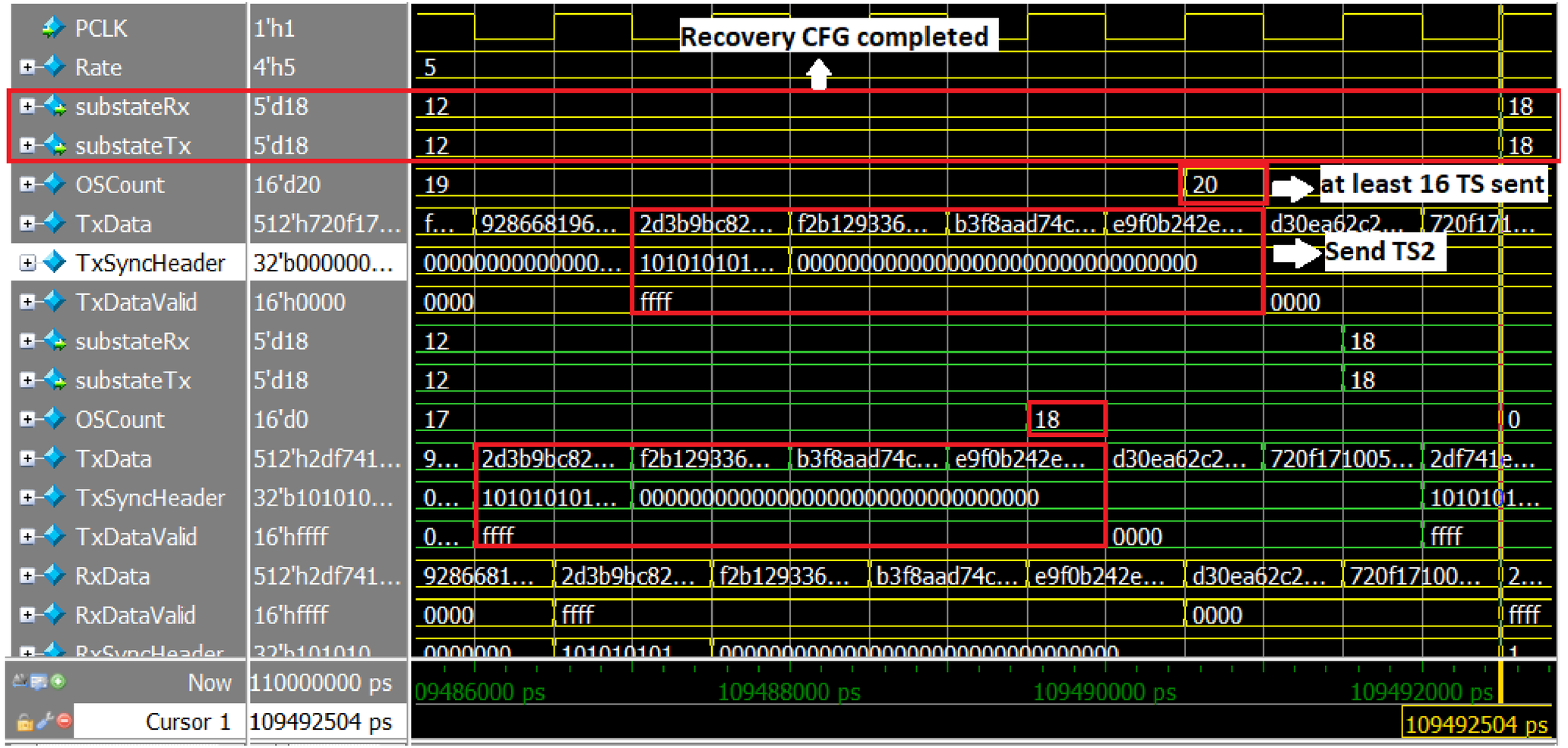
- The receiver must detect eight consecutive TS2s on any configured Lane.

- These TS2s must have:
  Matching Link and Lane numbers, and
  Matching rate identifiers to those being transmitted.

- In addition, one of the following conditions must be met:
  a) The speed_change bit in the received TS2s is cleared to 0b, or
  b) The transmitter has sent sixteen TS2s after receiving at least one valid TS2.

Recovery states

Recovery CFG in GEN5



Recovery states

Recovery CFG in GEN5

from the Transcript window in the simulation

we observe that the TX ,Rx Monitors and Scoreboard have passed successfully indicating that receive scrambled TS2 with right symbol and counts.

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_U_Monitor.sv(2377) @ 109496501: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_
TX_Slave_U_Monitor_h [TX_Slave_U_Monitor] Recovery.RcvrCfg substate at Upstream TX side completed successfully

 UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_D_Monitor.sv(2441) @ 109424501: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_
.RX_Slave_D_Monitor_h [RX_Slave_D_Monitor]  Recovery RcvrCfg  substate at Downstraem RX side completed successfully
 UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_U_Monitor.sv(2500) @ 109441501: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_
.RX_Slave_U_Monitor_h [RX_Slave_U_Monitor] Recovery.RcvrCfg substate at Upstream RX side completed successfully
 UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_D_Monitor.sv(2373) @ 109489501: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_
.TX_Slave_D_Monitor_h [TX_Slave_D_Monitor] Recovery.RcvrCfg substate at Downstream TX side completed successfully
```

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_D.sv(740) @ 109489501: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D
 [PCIe_Scoreboard1_D] Downstream Recovery_RcvrCfg Approved

UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_U.sv(799) @ 109496501: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U
 [PCIe_Scoreboard1_U] Upstream Recovery_RcvrCfg Approved
```

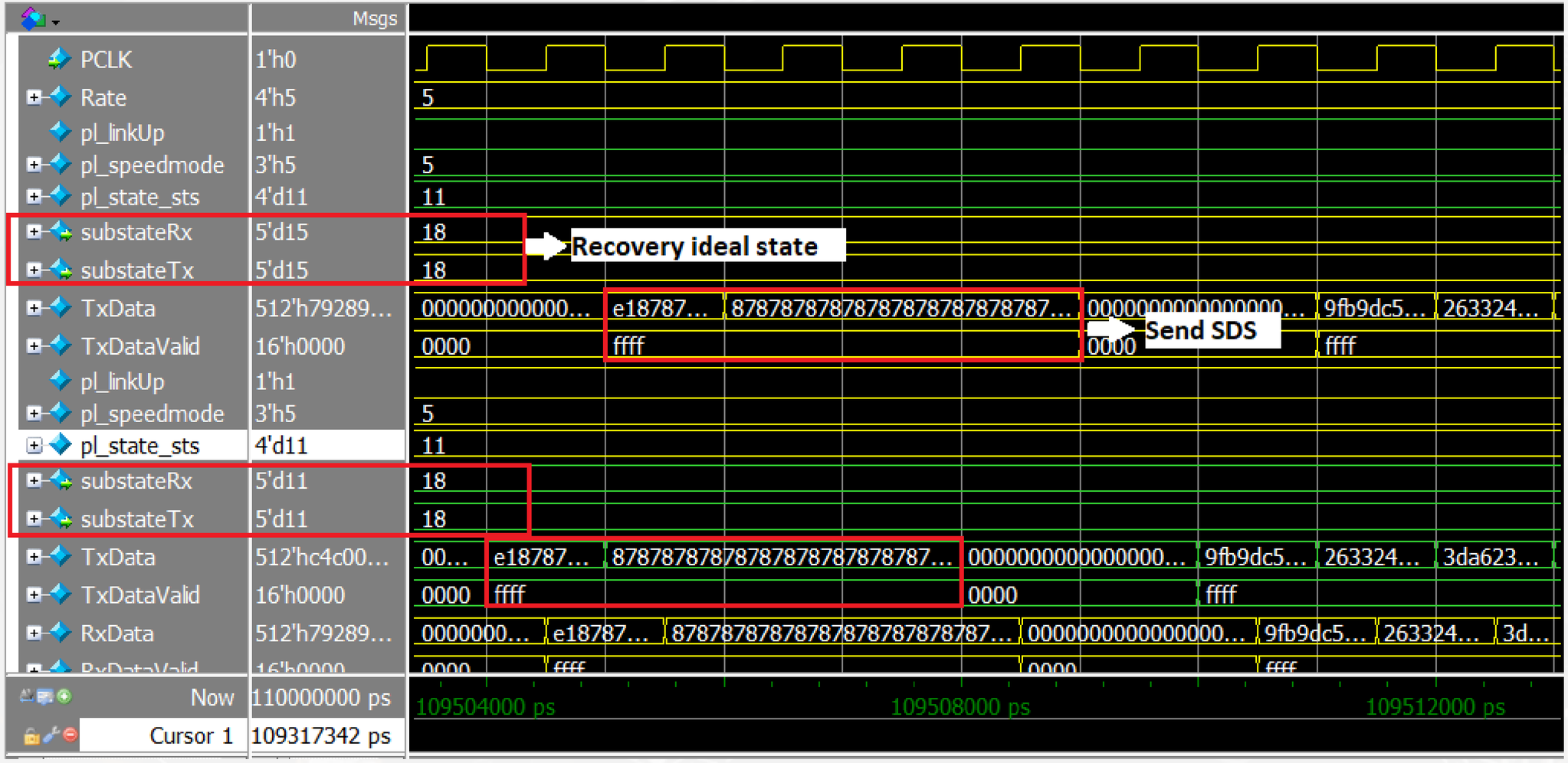
Recovery states

Recovery Idle

- Transmitters will usually send Idles in this substate as a preparation for changing to the fully operational L0 state
- start to send SDS OS that Used to mark the beginning of valid data in 128b/130b.
- send SKP OS that Used for synchronization, ensuring reliable communication despite small frequency mismatches.
- Finally, the transmitter sends approximately 8 consecutive Symbol Times of scrambled Idle data to complete the transition toward the L0 state.

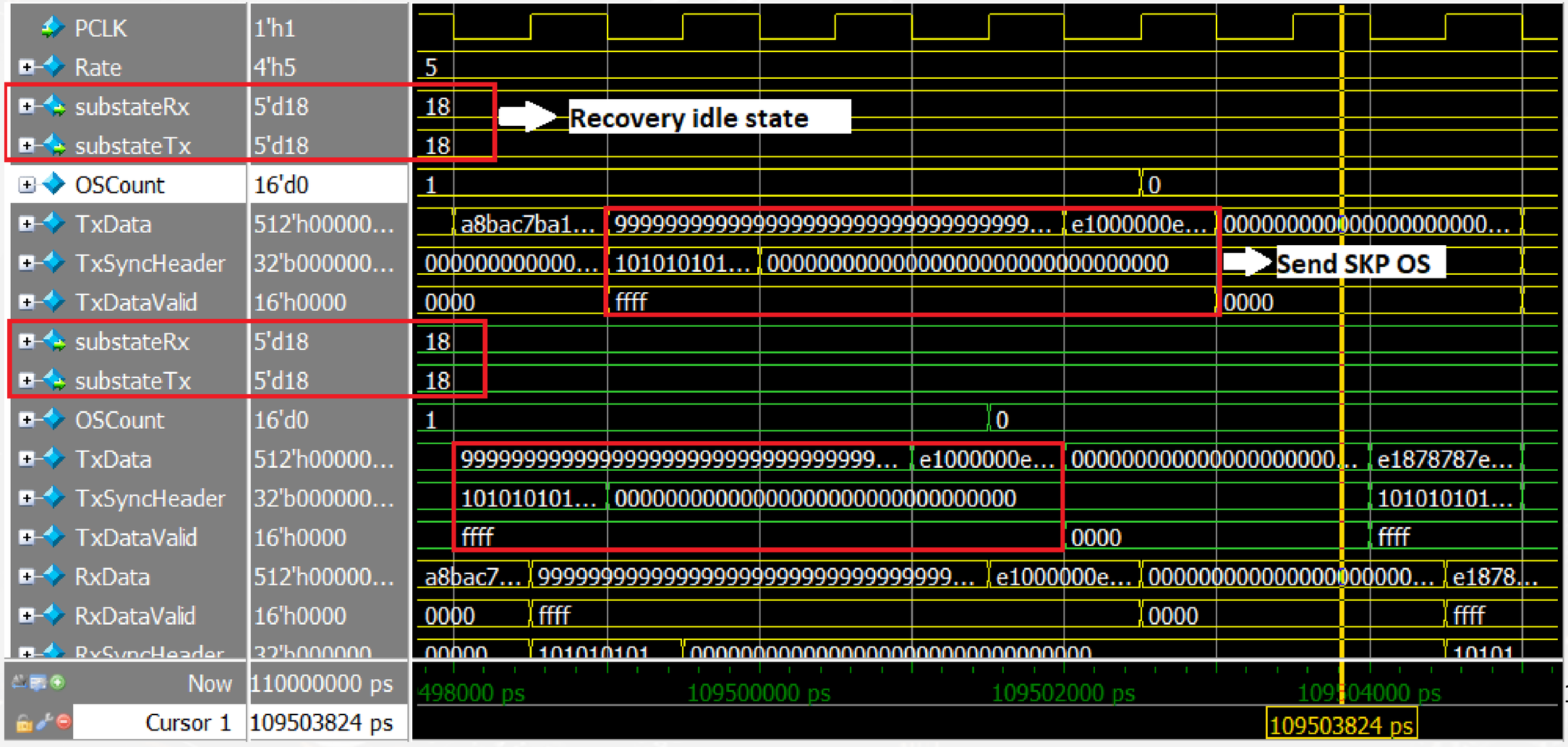
Recovery states

Recovery Idle



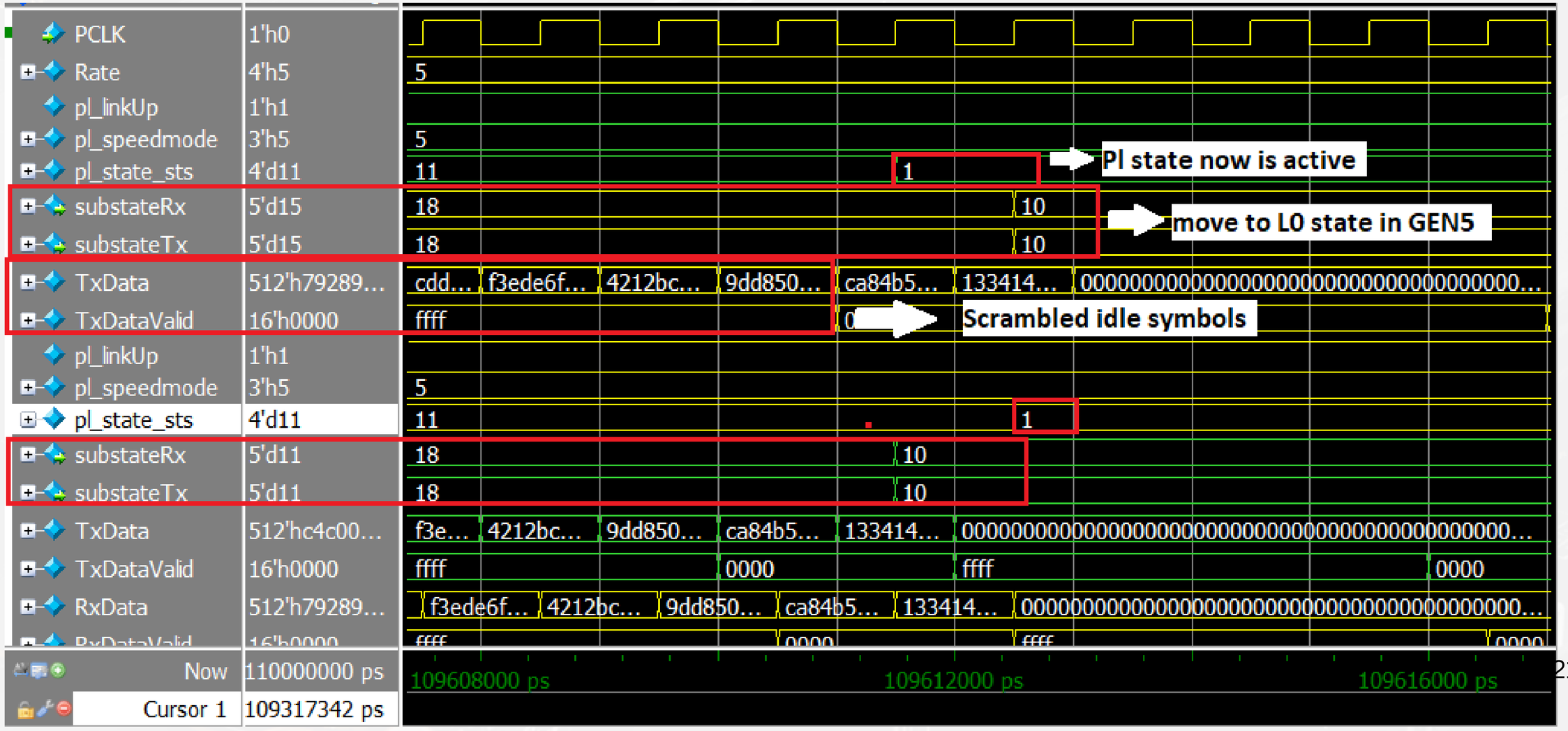
Recovery states

Recovery Idle



Recovery states

Recovery Idle



Recovery states

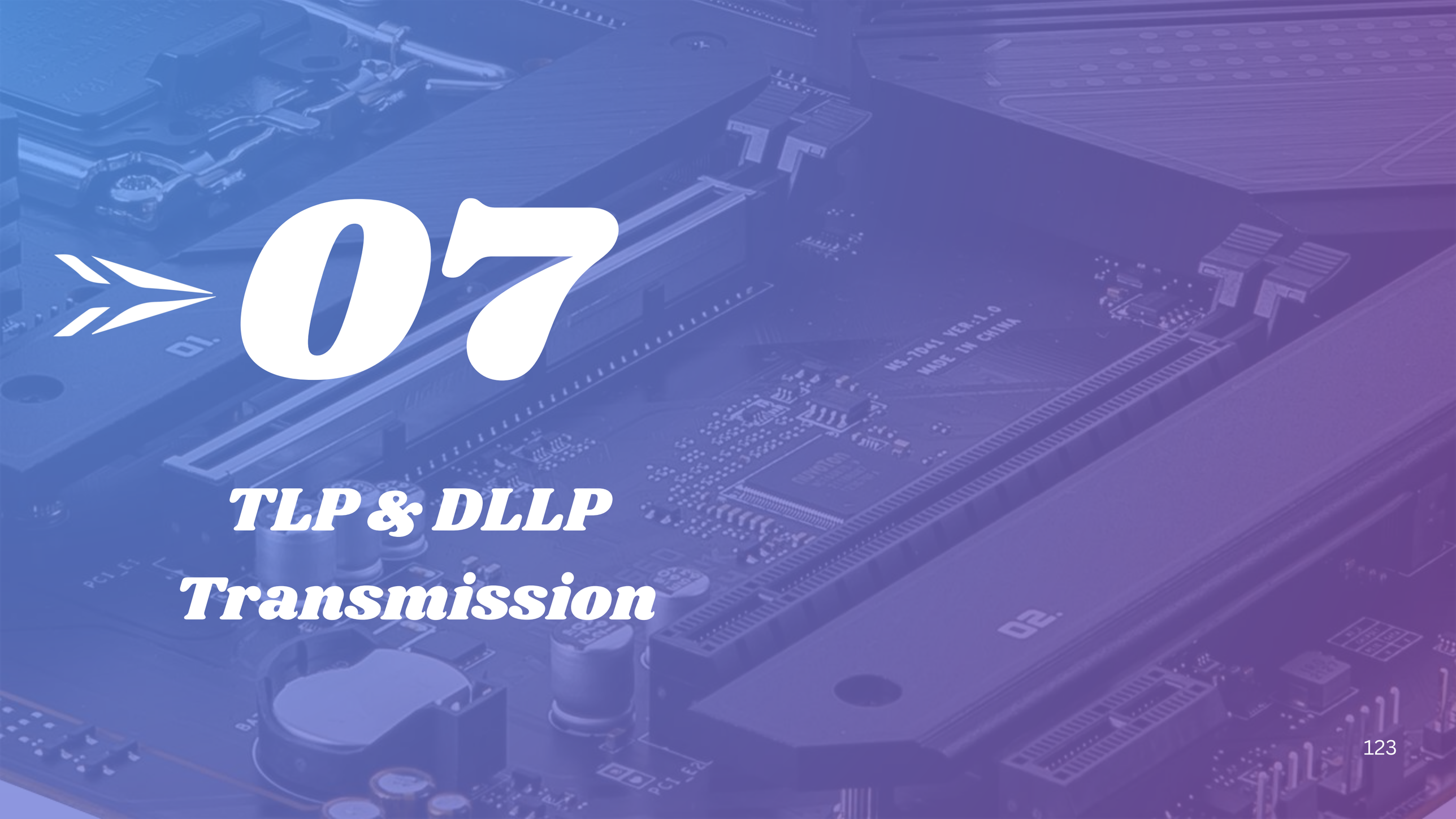
Recovery Idle

from the Transcript window in the simulation

we observe that the TX ,Rx Monitors and Scoreboard have passed successfully indicating that receive scrambled SDS and SKP OS and idle symbols with right symbol and counts.

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_D_Monitor.sv(2975) @ 109603500: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_
TX_Slave_D_Monitor_h [TX_Slave_D_Monitor] Recovery.Idle substate at Downstream TX side completed successfully
UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_D_Monitor.sv(2639) @ 109603500: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_
RX_Slave_D_Monitor_h [RX_Slave_D_Monitor] Recovery Idle substate at Downstraem RX side completed successfully
UVM_INFO E:\study\Grad_BY_Youssef\TB\TX_Slave_U_Monitor.sv(3051) @ 109604500: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_
TX_Slave_U_Monitor_h [TX_Slave_U_Monitor] Recovery.Idle substate at Upstream TX side completed successfully
UVM_INFO E:\study\Grad_BY_Youssef\TB\RX_Slave_U_Monitor.sv(2594) @ 109510500: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_
RX_Slave_U_Monitor_h [RX_Slave_U_Monitor] Recovery.Idle substate at Upstream RX side completed successfully
```

```
UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_U.sv(811) @ 109510500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U
 [PCIe_Scoreboard1_U] Upstream Recovery_Idle Approved
UVM_INFO E:\study\Grad_BY_Youssef\TB\PCIe_Scoreboard1_D.sv(746) @ 109510500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D
 [PCIe_Scoreboard1_D] Downstream Recovery_Idle Approved
```



07

TLP & DLLP Transmission

TLP & DLLP Transmission

Achieved linkup and negotiated the highest common speed between the upstream and downstream devices.

What's Next?

TLP and DLLP Transmission!

TLP & DLLP Transmission

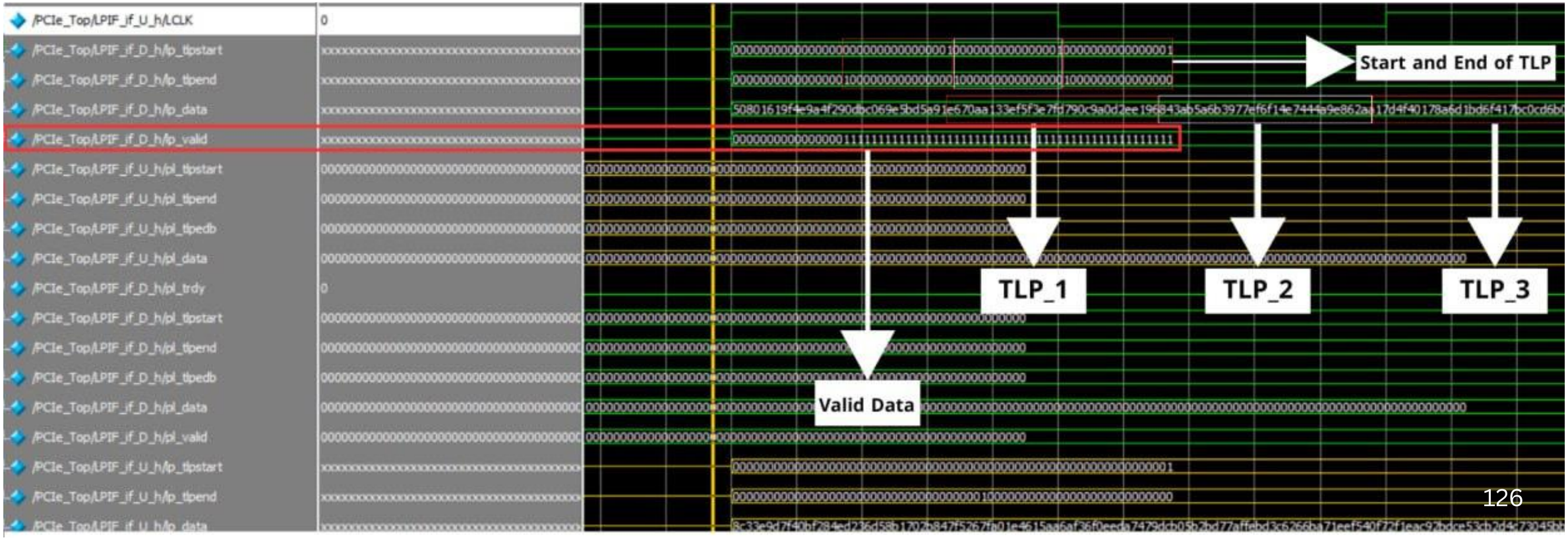
Signals Overview

- pl stands for physical to link layer (Reception path)
- lp stands for link layer to physical (Transmission path)

SIGNAL	DESCRIPTION
pl_tlpstart[63:0]	Indicating the start of TLP packet
pl_tlpend[63:0]	Indicating the end of TLP packet
pl_dlpstart[63:0]	Indicating the start of DLLP packet
pl_dlpend[63:0]	Indicating the end of DLLP packet
pl_data [511:0]	Physical Layer to Link Layer Data
pl_valid [63:0]	Physical Layer to Link Layer indicates data valid on pl_data

1- Transmission of TLPs

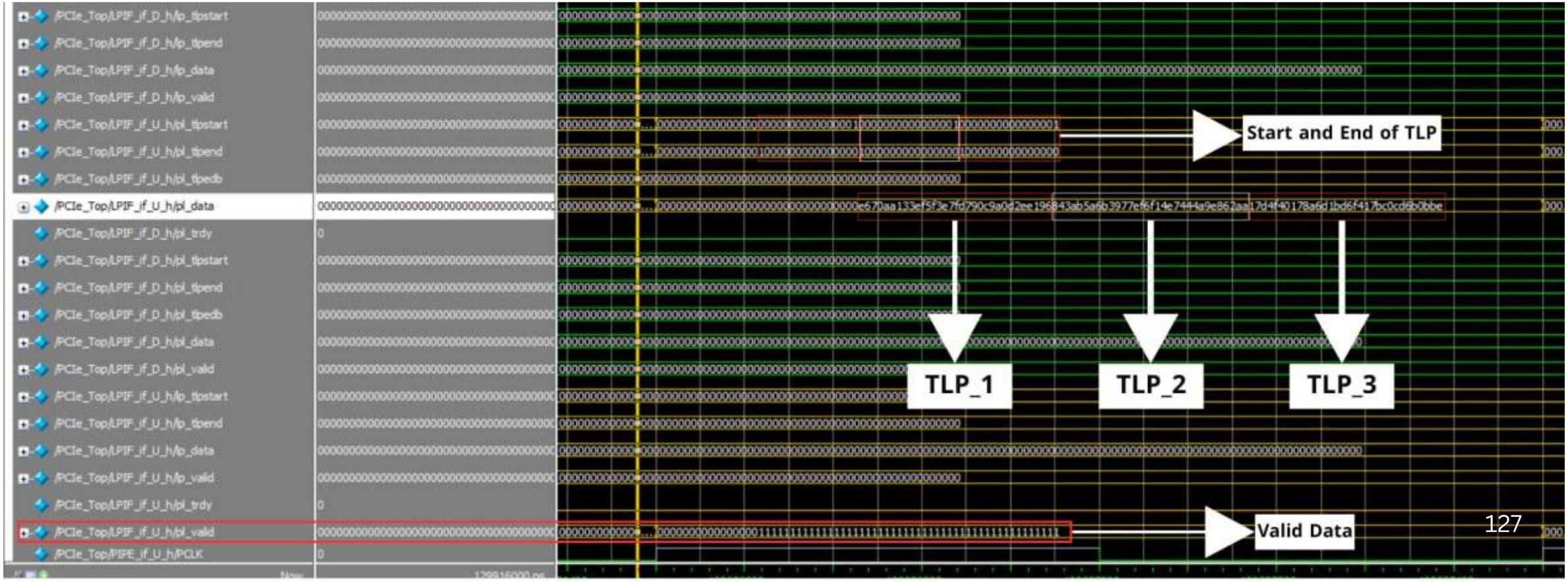
- Downstream device is transmitting three TLPs



TLP & DLLP Transmission

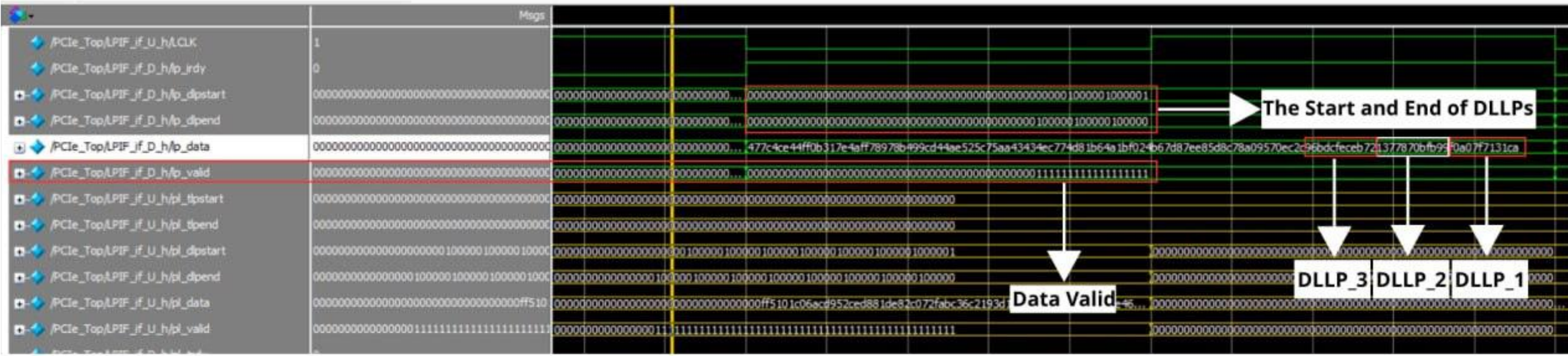
1-Reception of TLPs

- Upstream device is receiving three TLPs



2- Transmission of DLLPs

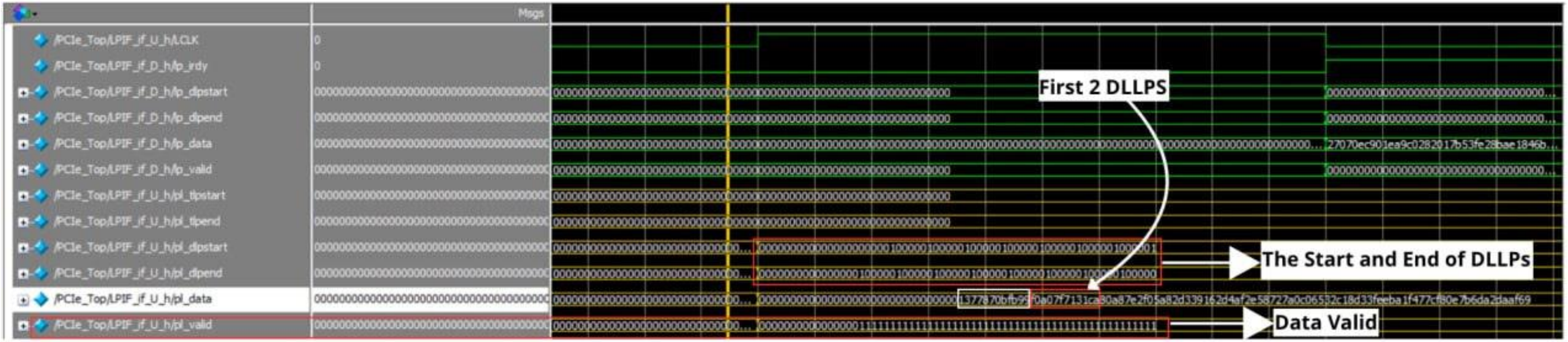
- Downstream device is sending three DLLPs



TLP & DLLP Transmission

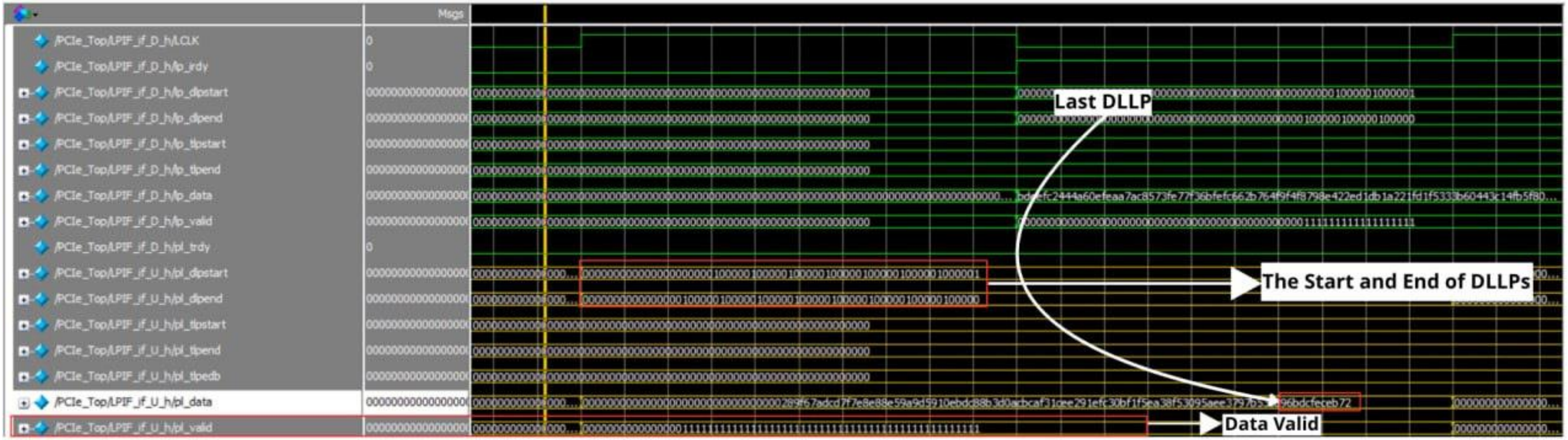
2- Reception of DLLPs

- Upstream device is receiving first two DLLPs in this transfer
- In this example, 6 DLLPs from the previous transfer were not transmitted, so they are sent along with the 3 DLLPs of the current transfer



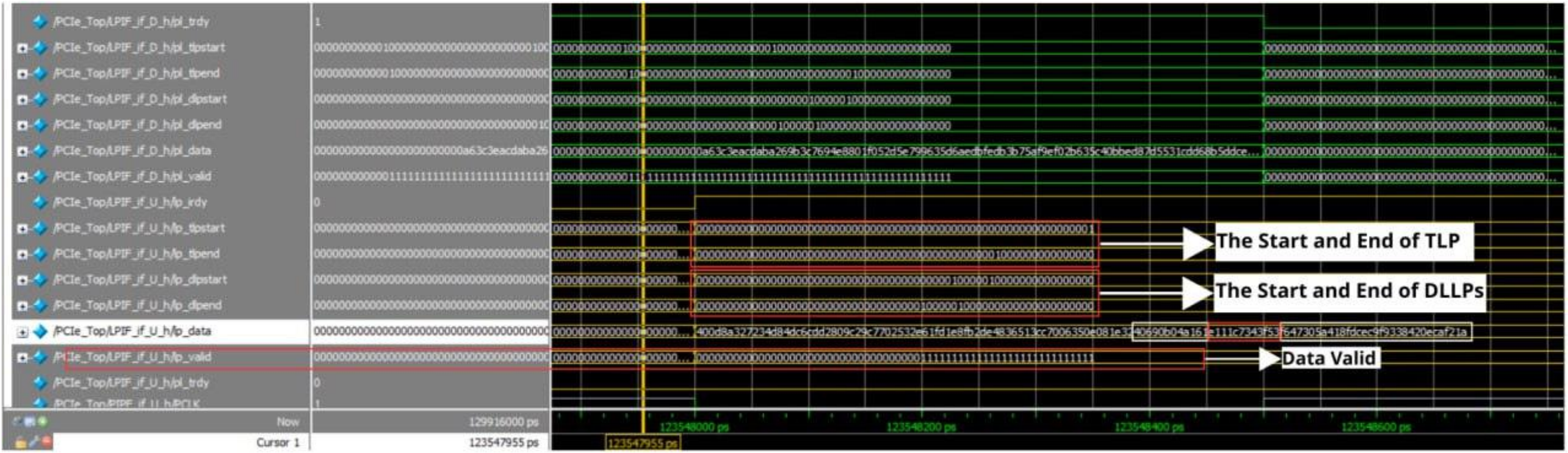
2- Reception of DLLPs

- Upstream device is receiving last DLLP in this transfer



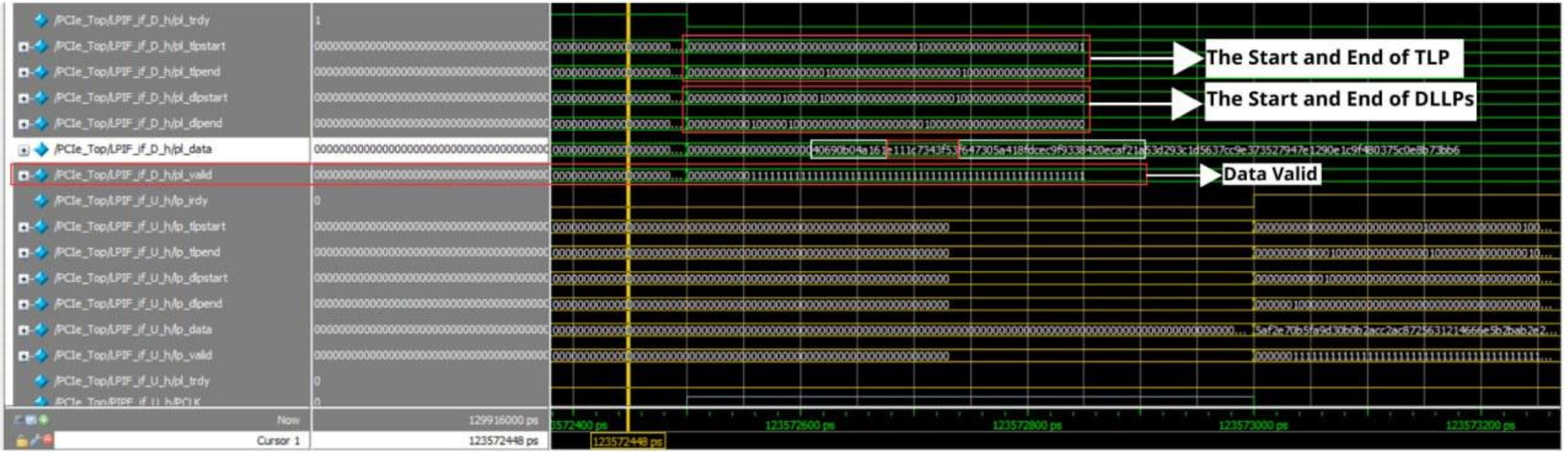
3- Transmission of TLPs and DLLPs

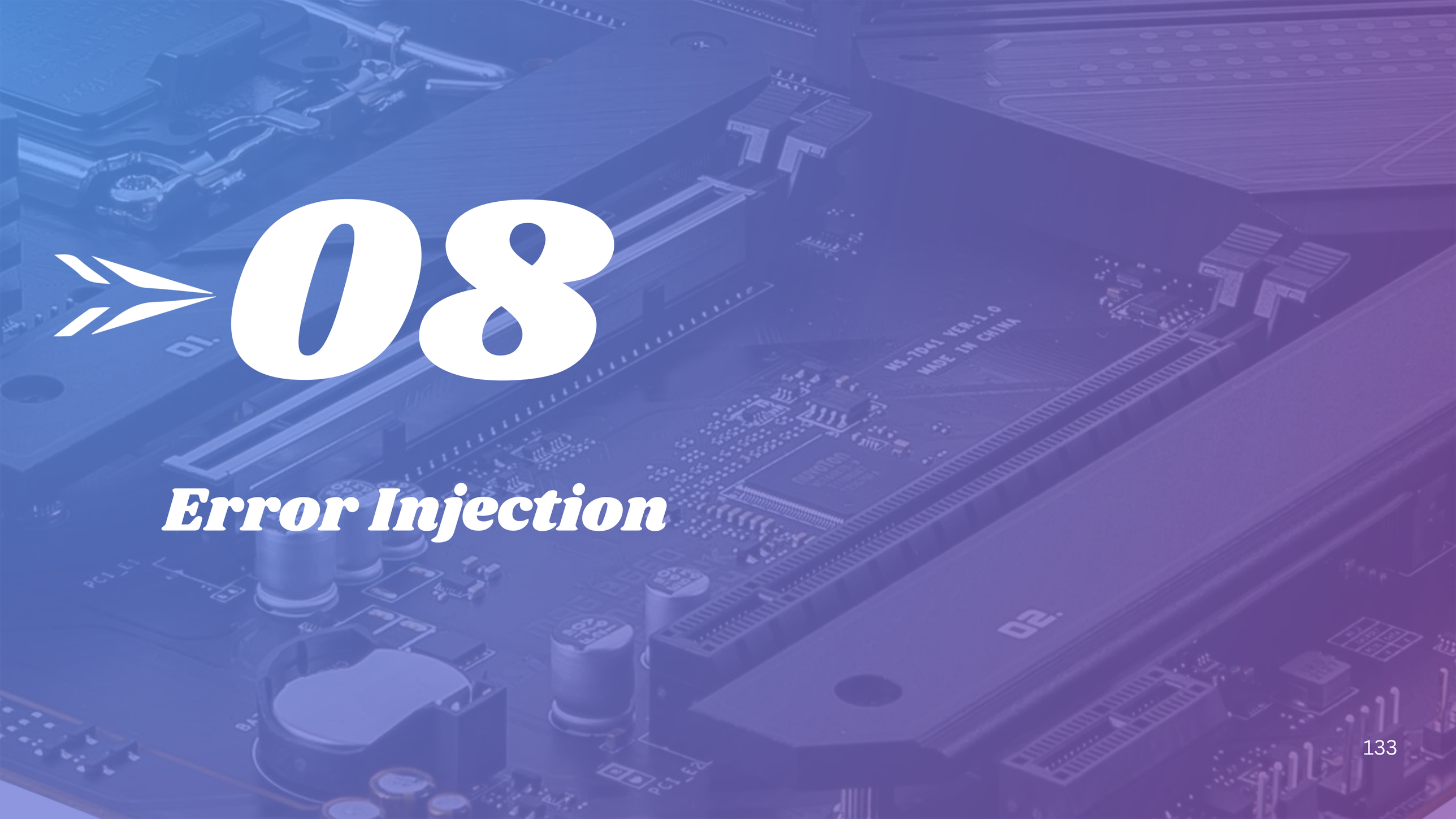
- Upstream device is transmitting one TLP and two DLLPs back-to back



3- Reception of TLPs and DLLPs

- Downstream device is receiving one TLP and two DLLPs back-to back





08

Error Injection

Error Inection

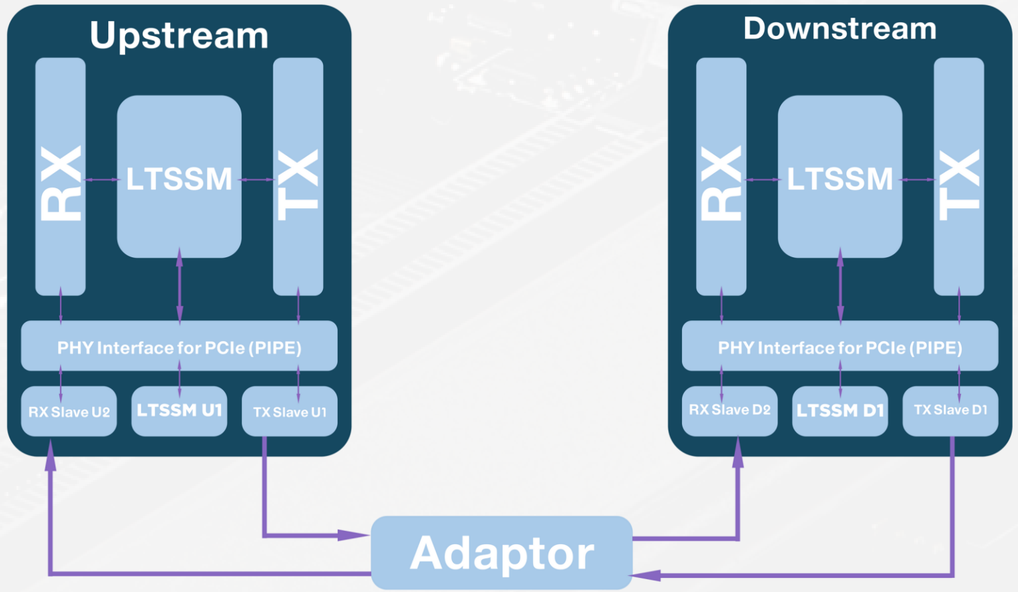
Why Error Injection?

Some LTSSM state transitions—like timeouts or error recovery—don't occur under ideal conditions. To verify design compliance and robustness, we intentionally inject errors into the transmitted data between DUTs.

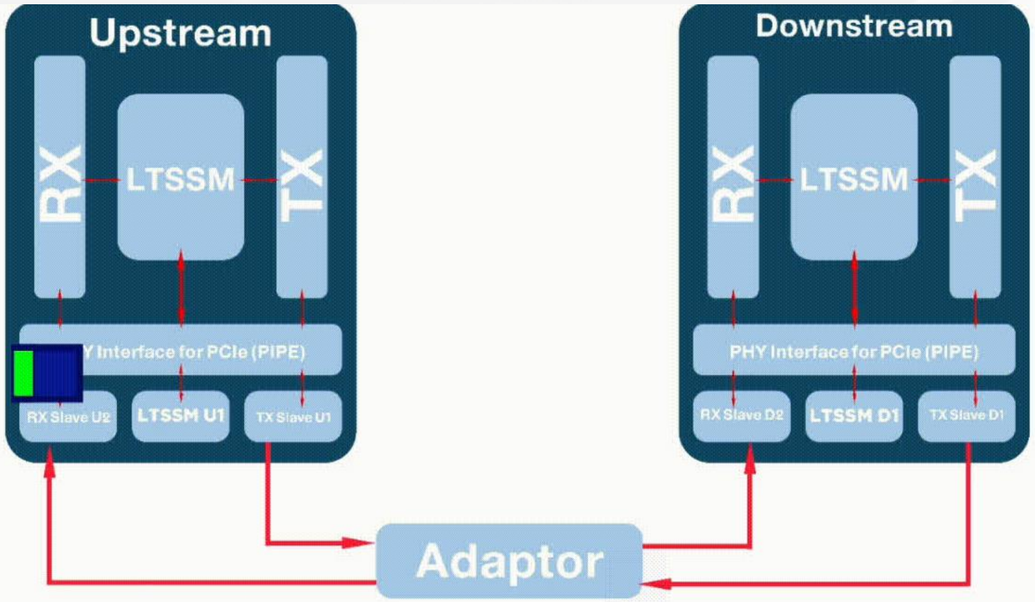
This allows us to:

- Trigger rare transitions not seen in normal operation.
- Validate recovery behavior as specified in the protocol.
- Simulate real-world faults like signal degradation or link failures.

Error injection ensures the link can detect, handle, and recover from faults reliably.



Error Injection Mechanism



Error injection is performed using a dedicated adaptor module, which modifies the content of transactions to simulate faulty conditions. This adapter can:

- Block transactions entirely
- Alter specific fields within the transaction
- Bypass or delay transmissions

The adaptor is substate-aware. It receives the current LTSSM substate information from PIPE sequence monitors connected to both the Downstream and Upstream DUTs. Upon detecting a match between the current substate and a predefined target substate for injection, the adapter begins manipulating the data stream accordingly. The altered data is then forwarded to the receiver (Rx) on the opposite DUT.

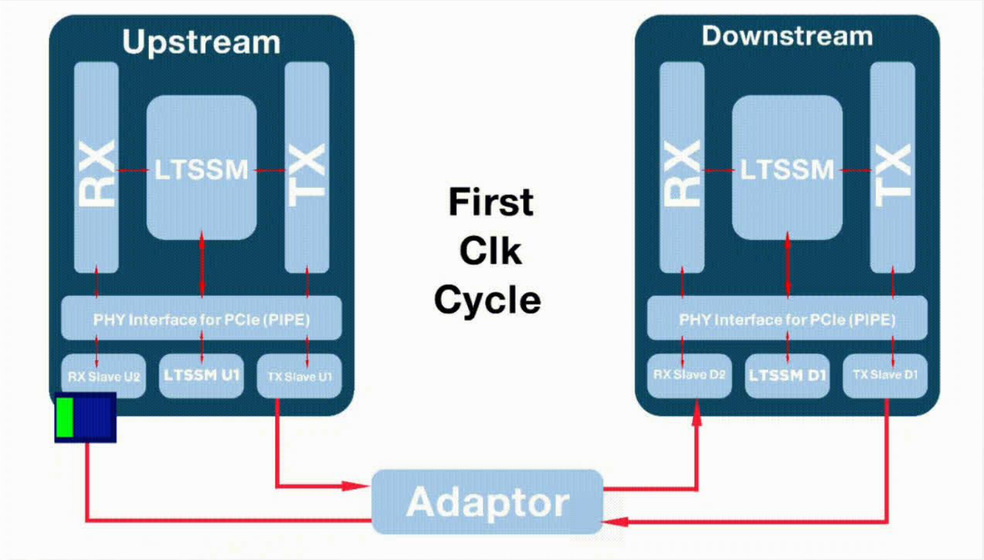
This targeted approach ensures that error scenarios are injected precisely where needed to validate robustness and substate-specific behavior.

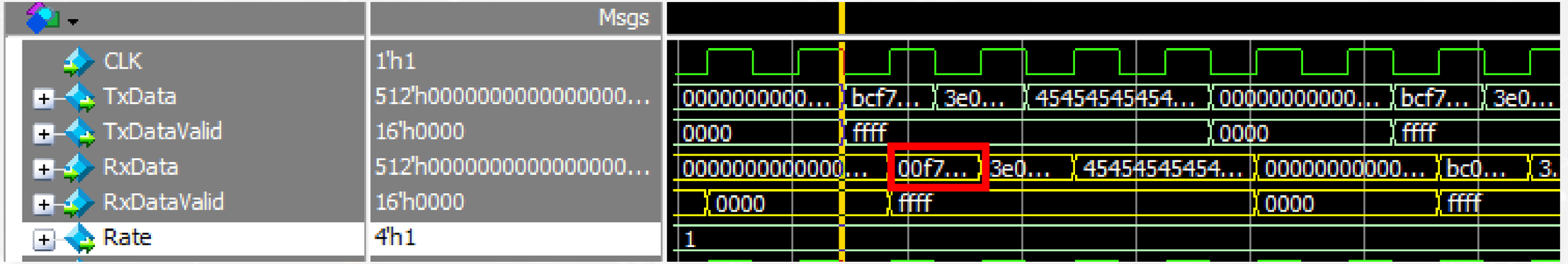
Error Injection

Adaptor in Action: COM Field Error Injection

This diagram illustrates the adapter actively injecting an error into the COM field of a Training Sequence (TS) during PCIe Gen1 communication.

- The PIPE interface operates at 32 bits per lane, while a complete TS is 128 bits, requiring 4 clock cycles to transmit the full TS.





Downstrea
m

Upstrea
m

What is Timeout?

After entring a substate a timer starts, Why ?

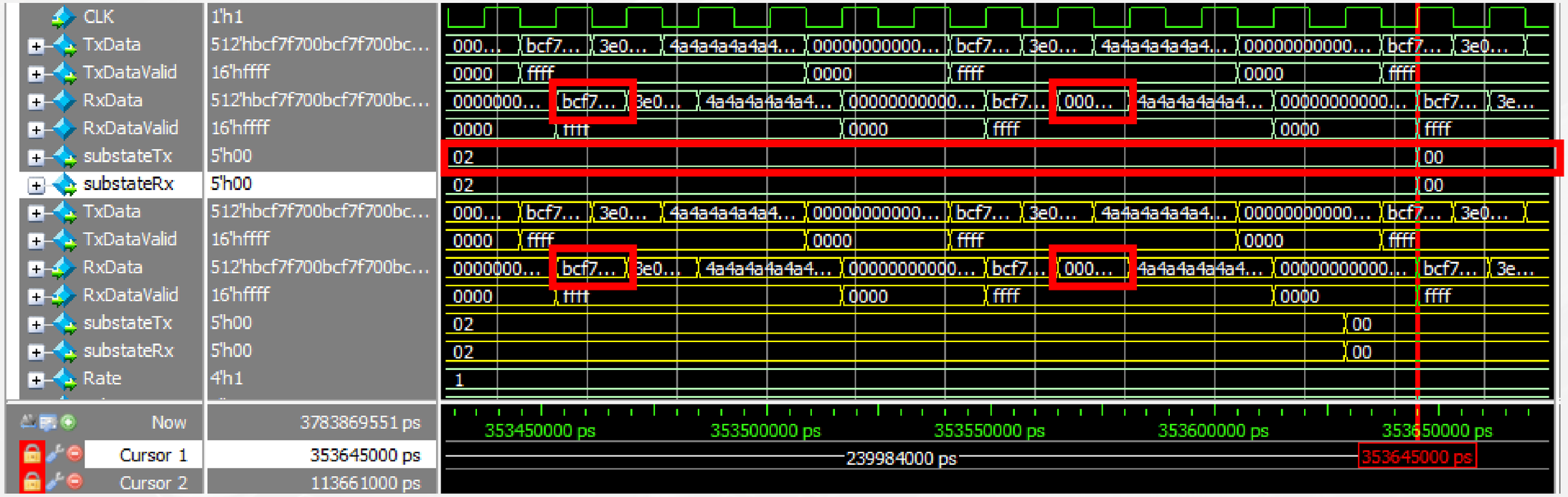
- A timer is initiated when entering each LTSSM substate to ensure progress in the link training process.
- If one device (upstream or downstream) becomes stuck due to an unreliable connection, the timer prevents indefinite stalling.
- When the timer expires without successful signaling, the LTSSM transitions back to the Detect state to re-attempt establishing a reliable link.

Quite simillar to the principle of reliable data transfer in networks (ACKs,Timeouts)

ACKs → TSs Timeout → timeout of each substate

Error Injection

Simulation: Polling.Active



Timeout of Polling.Active substate

- Both DUTs (Upstream & Downstream) transmit TS1 Ordered Sets (OS) continuously
- Receivers (Rx) check each TS1 for field correctness
- Invalid TS1s are discarded, valid ones are counted.

✔ Transition to Next Substate (Polling.Configuration): Occurs when ≥ 8 valid TS1s are received.

✖ Timeout Condition : If < 8 valid TS1s are received after 24 ms, LTSSM reverts to Detect.Quiet.

Error Injection

Simulation: Scoreboards & monitors

```
# ----------------------------------------------------------------------------
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/Inject_Error_vseq.sv(112) @ 133926505: reporter@@Inject_Error_vseq_h [Inject_Error_vseq] ---------------2- start polling activ
e error Injection sequence ----------------
# ----------------------------------------------------------------------------
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/LTSSM1_U_Driver.sv(92) @ 133926505: uvm_test_top.PCIe_Env_h.LTSSM1_U_Agent_h.LTSSM1_U_Driver_h [LTSSM1_U_Driver]  before detec
tion
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/LTSSM1_U_Driver.sv(95) @ 133926505: uvm_test_top.PCIe_Env_h.LTSSM1_U_Agent_h.LTSSM1_U_Driver_h [LTSSM1_U_Driver]  after detect
ion
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/TX_Slave_U_Monitor.sv(485  @ 133926505: uvm_test_top.PCIe_Env_h.TX_Slave_U_Agent_h.TX_Slave_U_Monitor_h [TX_Slave_U_Monitor] D
etect Active substate at Upstream TX side completed successfully
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/RX_Slave_U_Monitor.sv(197 ) @ 133926505: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX_Slave_U_Monitor_h [RX_Slave_U_Monitor]
Detect Active substate at Upstream RX side completed successfully
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/TX_Slave_D_Monitor.sv(466  @ 133926505: uvm_test_top.PCIe_Env_h.TX_Slave_D_Agent_h.TX_Slave_D_Monitor_h [TX_Slave_D_Monitor] D
etect Active substate at Downstream TX side completed successfully
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/RX_Slave_D_Monitor.sv(187 ) @ 133926505: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Slave_D_Monitor_h [RX_Slave_D_Monitor]
Detect Active substate at Downstraem RX side completed successfully
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/PCIe_Scoreboard1_U.sv(765) @ 133926505: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U_h [PCIe_Scoreboard1_U] Upstream Detect_Acti
ve Approved
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/PCIe_Scoreboard1_D.sv(703) @ 133926505: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D_h [PCIe_Scoreboard1_D] Downstream Detect_Ac
tive Approved
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/RX_Slave_U_Monitor.sv(2019) @ 373926505: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX_Slave_U_Monitor_h [RX_Slave_U_Monitor]
Time out occur in Polling Active , we get back to detect state
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/RX_Slave_U_Monitor.sv(1947) @ 373926505: uvm_test_top.PCIe_Env_h.RX_Slave_U_Agent_h.RX_Slave_U_Monitor_h [RX_Slave_U_Monitor]
Detect Quiet substate at Upstream RX side completed successfully
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/RX_Slave_D_Monitor.sv(1927) @ 373926505: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Slave_D_Monitor_h [RX_Slave_D_Monitor]
Time out occur in Polling Active , we get back to detect state
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/RX_Slave_D_Monitor.sv(1848) @ 373926505: uvm_test_top.PCIe_Env_h.RX_Slave_D_Agent_h.RX_Slave_D_Monitor_h [RX_Slave_D_Monitor]
Detect Quiet substate at Downstraem RX side completed successfully
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/PCIe_Scoreboard1_U.sv(101) @ 373926505: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_U_h [PCIe_Scoreboard1_U] Upstream RX [Polling
_Active] Time Out To Detect_Quiet
# UVM_INFO C:/Users/gpscl/OneDrive/Desktop/Grad_BY_Youssef/TB/PCIe_Scoreboard1_D.sv(97) @ 373926505: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard1_D_h [PCIe_Scoreboard1_D] Downstream RX [Pollin
g_Active] Time Out To Detect_Quiet
```

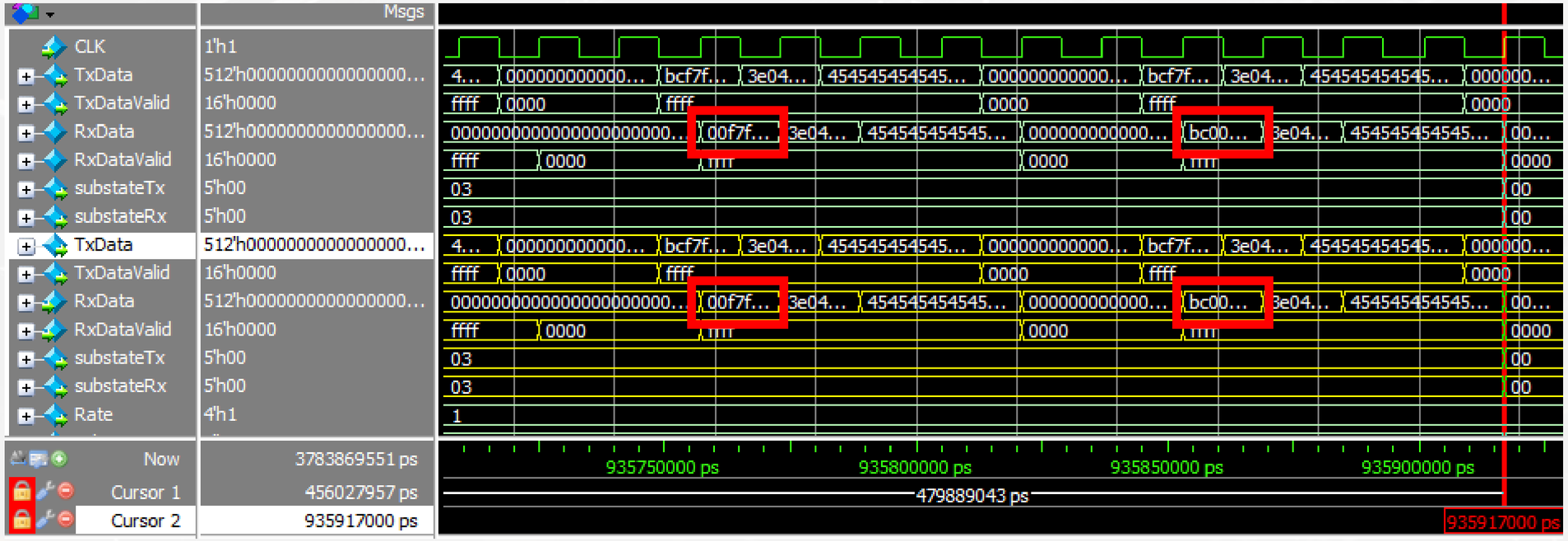
Error Injection

Simulation: Polling.Configuration

- Both DUTs transmit TS2 Ordered Sets (OS)
- Invalid TS2s are discarded, valid ones are counted.

✔ Transition to Next Substate (Configuration.LinkWidth.Start): Occurs when ≥ 8 valid TS2s are received.

✘ Timeout Condition : If < 8 valid TS2s are received after 48 ms, LTSSM reverts to Detect.Quiet.



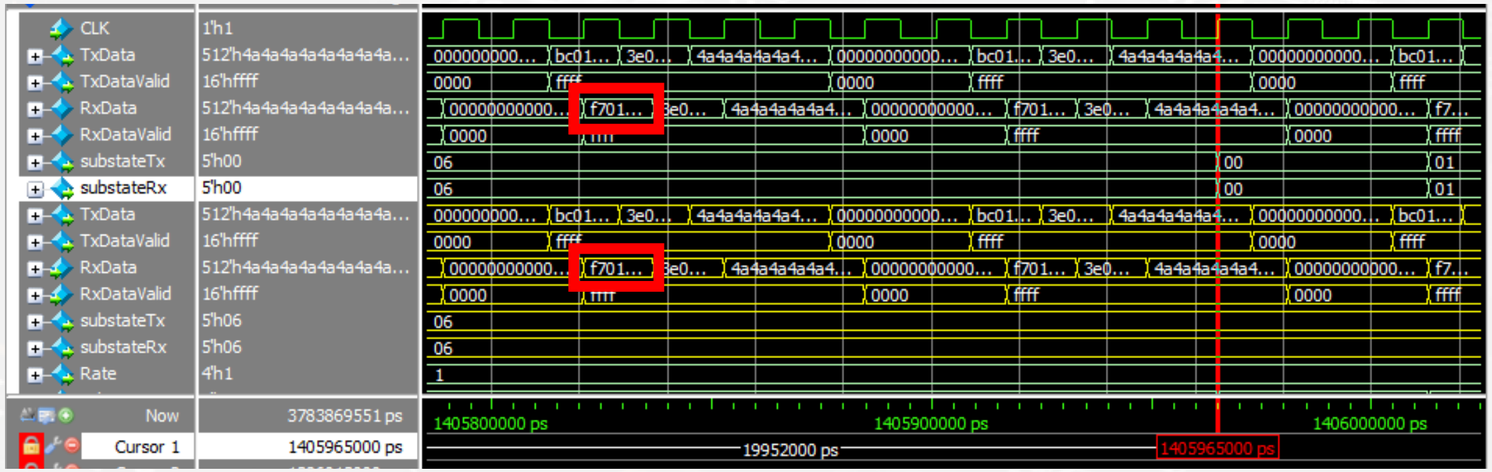
Timeout of Polling..Configuration substate

Error Injection

Simulation

- During this substate, the DUTs exchange TS1 ordered sets that include the negotiated link number.
- To transition to Configuration.Linkwidth.Accept, the receiver (RX) must detect two consecutive valid TS1s containing a non-PAD link number.
- If the expected TS1s are not received within 24 ms, a timeout occurs, and the LTSSM transitions back to the Detect state.

24ms Timeout of Configuration.Linkwidth_Start



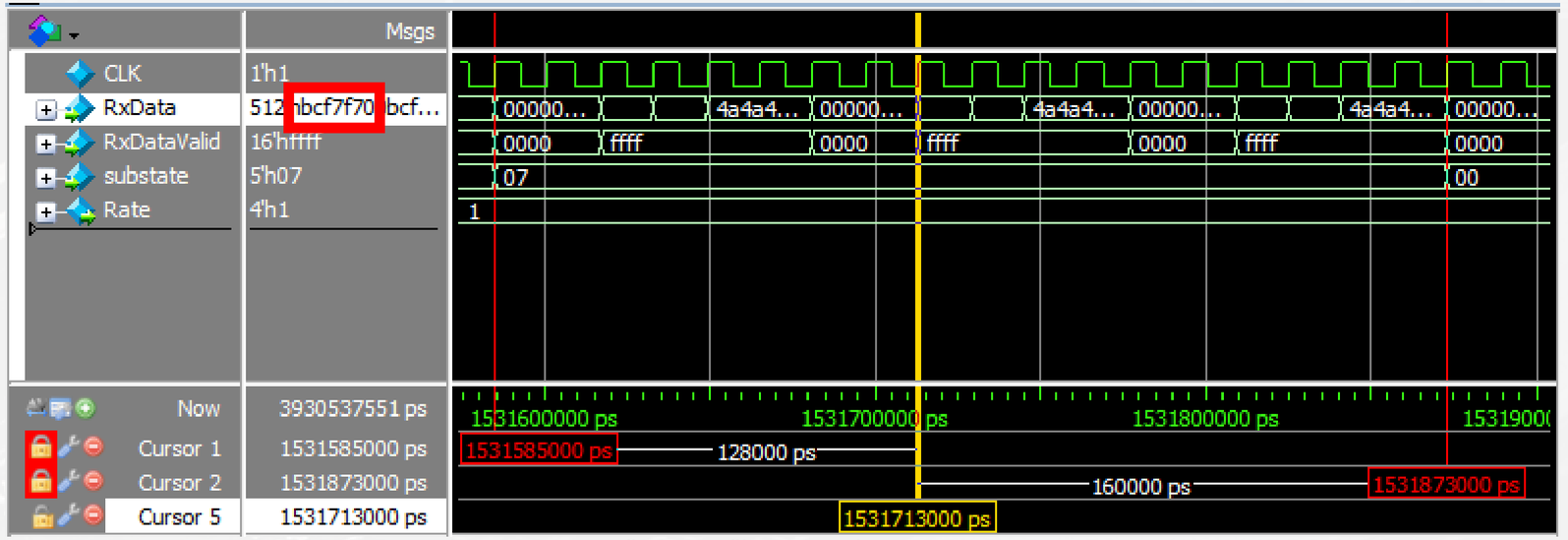
2ms Timeout of Configuration.Lanenum_Wait

- The downstream TX transmits TS2 ordered sets with the negotiated lane number.
- The upstream TX responds with TS1 ordered sets containing the same lane number.
- A transition to the next substate occurs when the DUT receives two consecutive valid TS1/TS2 ordered sets with matching lane numbers.
- If valid ordered sets are not received within 2 ms, the training is considered unsuccessful, and the LTSSM returns to the Detect state.

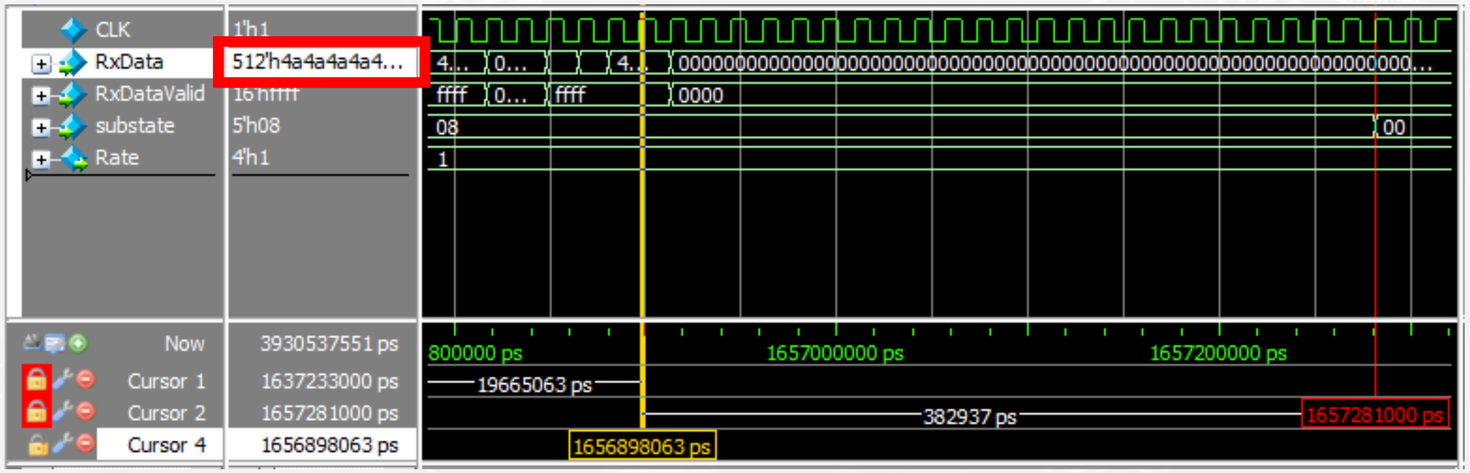
Error Injection

Simulation

- A timeout is triggered if the received TS1 ordered sets contain PAD values for the link or lane numbers.
- PAD values indicate that configuration was not successfully completed.
- As a result, the LTSSM transitions back to the Detect state to restart link training and reestablish the connection between the upstream and downstream links and lanes.



Timeout of Configuration.Lanenum_Accept



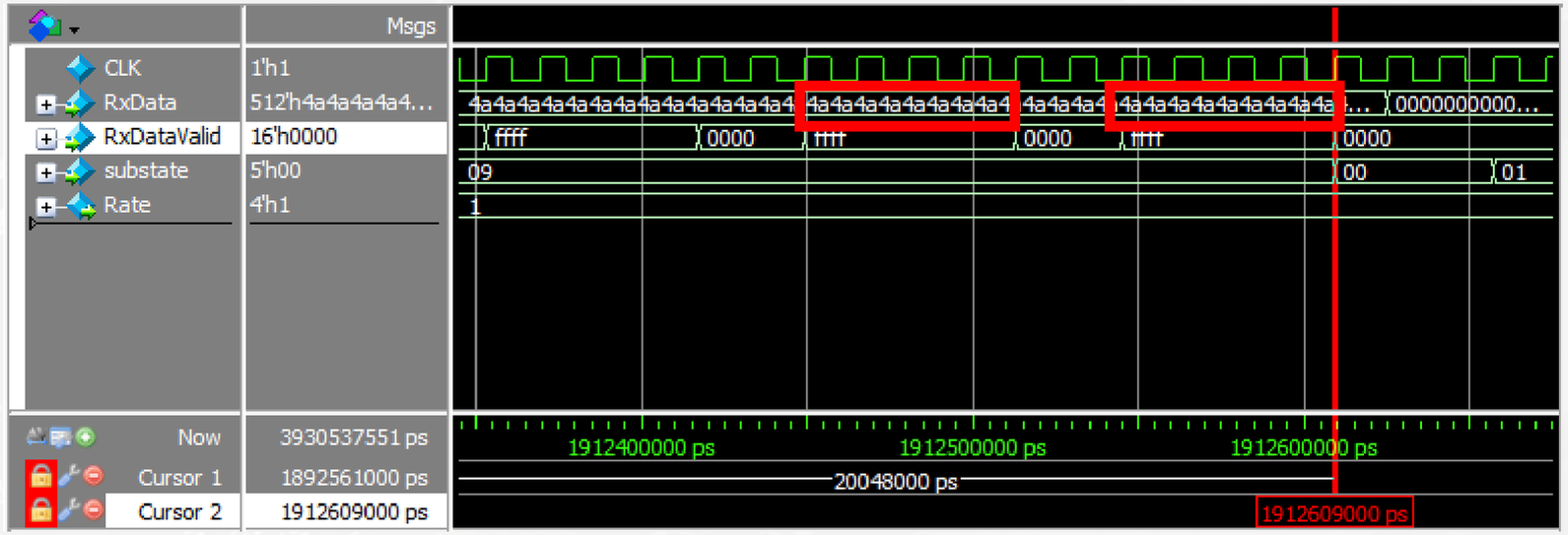
2ms Timeout of

- In Configuration.Complete, the DUTs exchange TS2 ordered sets to confirm link and lane number configuration.
- The receiver (RX) must successfully receive 8 consecutive valid TS2 ordered sets to move to the next substate.
- If this condition is not met in 2ms time frame, the LTSSM transitions back to the Detect state to reinitiate the link training process.

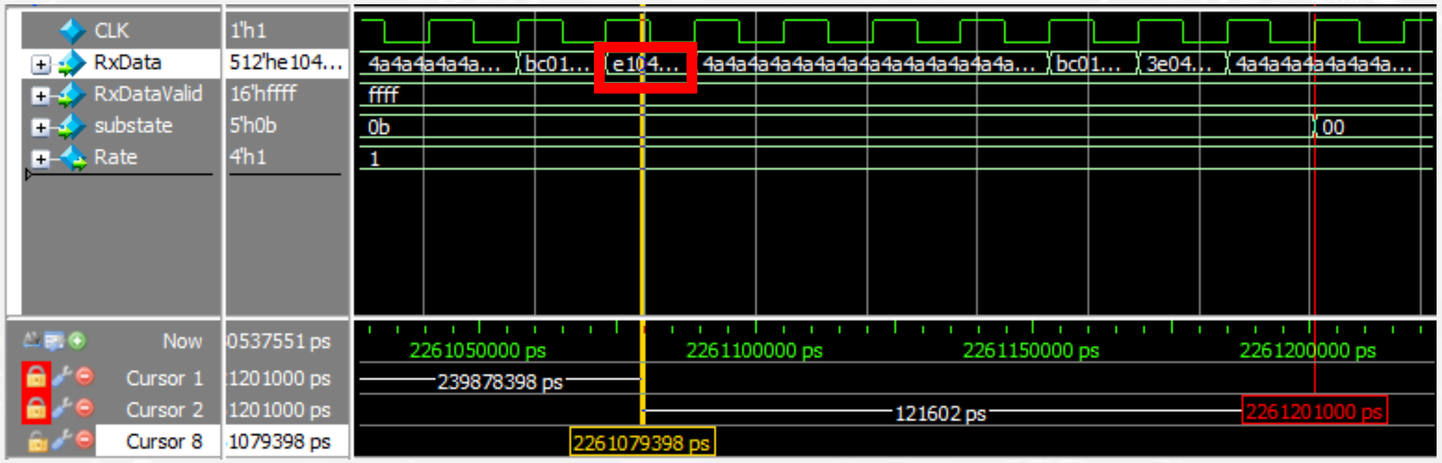
Error Injection

Simulation

- In this substate, the DUTs exchange Idle OS to prepare the link to go to L0.
- The receiver (RX) must successfully receive 8 consecutive valid scrambled EIOS to move to L0.
- If this condition is not met in 2ms time out window, the LTSSM transitions back to the Detect state to reinitiate the link training process.



2ms Timeout of Configuration.Idle



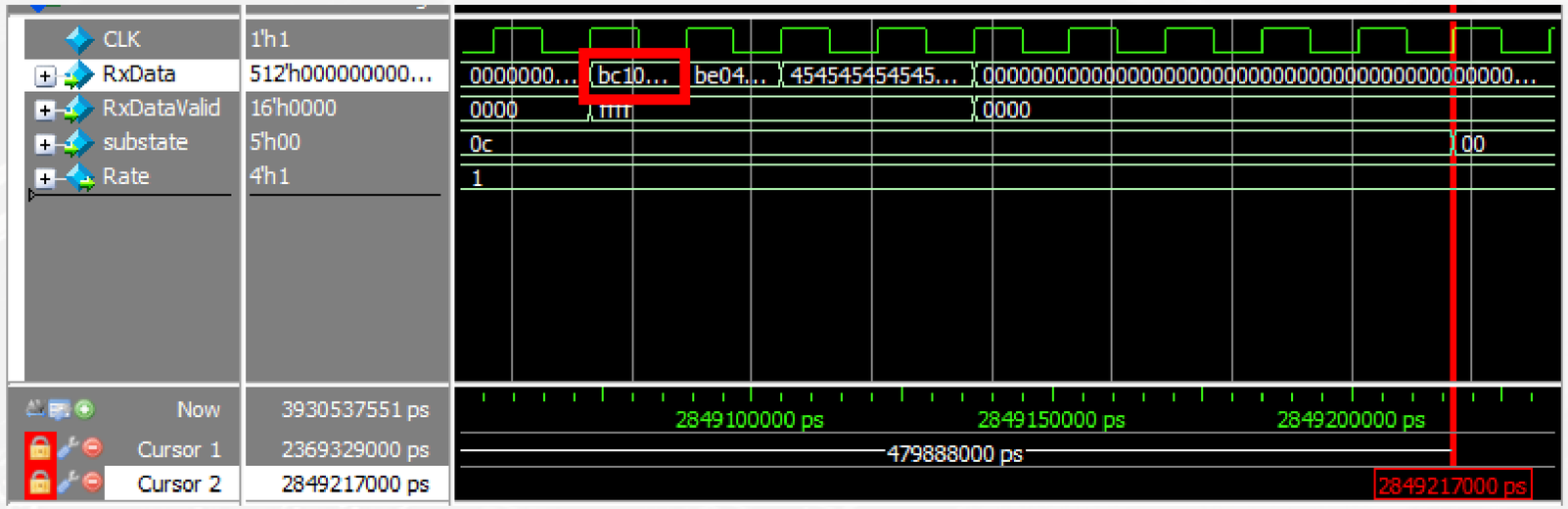
24ms Timeout of Configuration.Idle

- In this substate, TS1 and TS2 ordered sets are exchanged between DUTs to determine the next transition:
  - Equalization: Phase 1 (downstream) or Phase 0 (upstream), if the device has reached highest common speed and Start_Equalization_With_Preset bit is set to 1.
  - Speed Change, if the Speed_Change bit is set to 1.
- If the receiver fails to detect the required number of valid TS1/TS2 ordered sets within 24 ms, a timeout occurs, and the LTSSM returns to the Detect state.

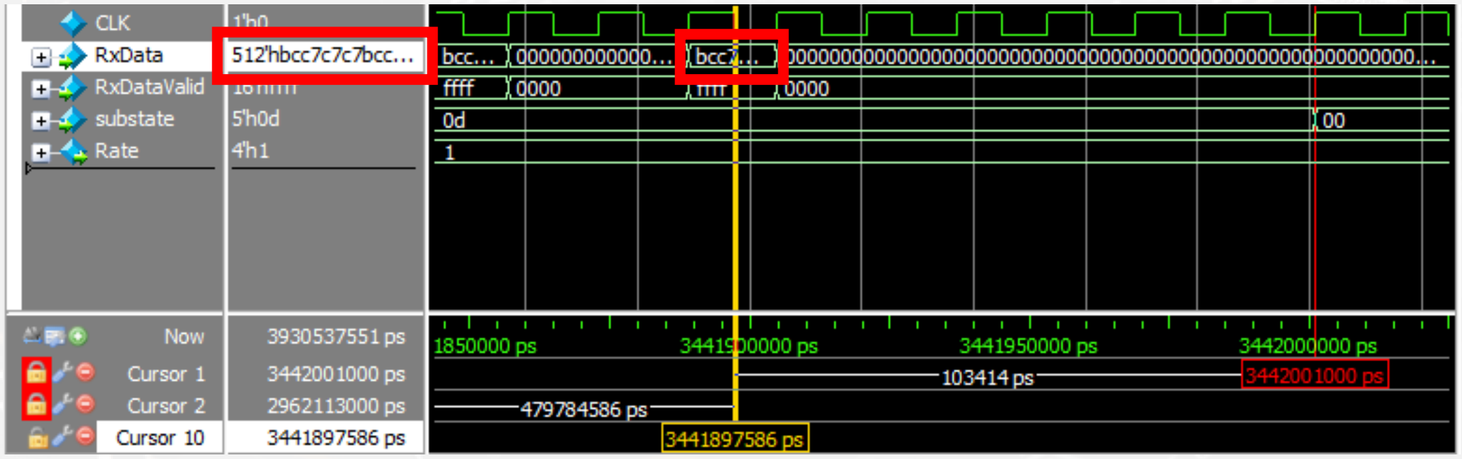
Error Injection

Simulation

- In this substate, TS2 ordered sets are exchanged between DUTs.
- If the receiver fails to detect the required number of valid TS2 ordered sets within 48ms, a timeout occurs, and the LTSSM returns to the Detect state.



48ms Timeout of Recovery.RcvrCfg



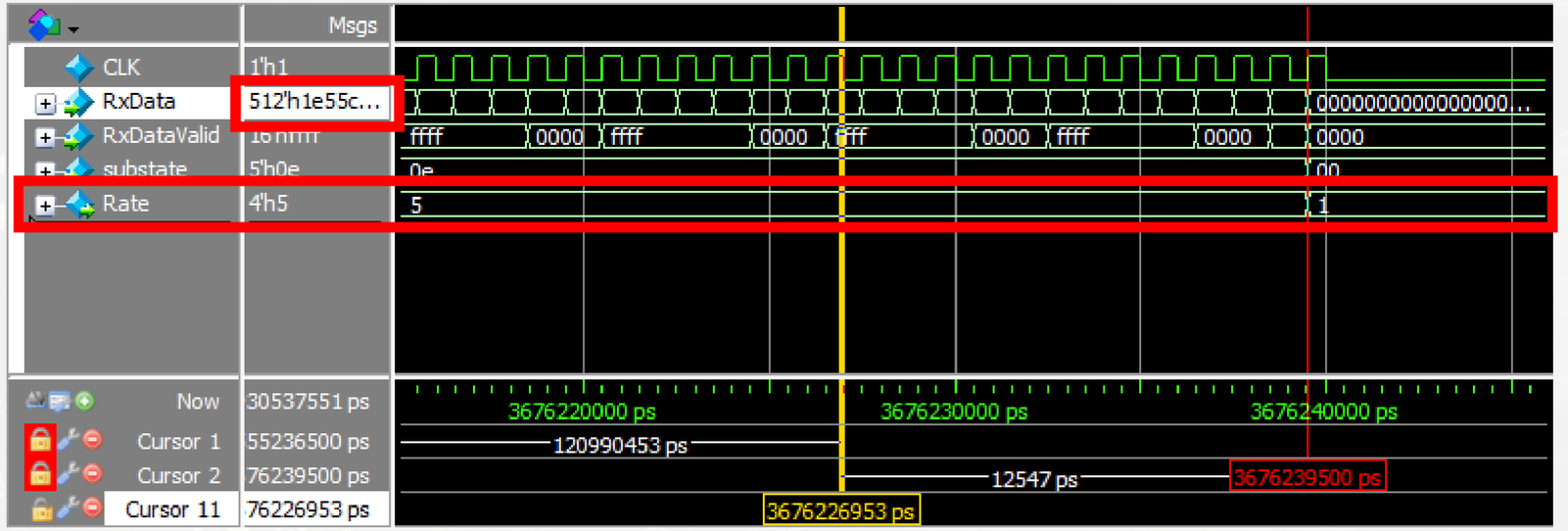
48ms Timeout of

- In this substate, EIOS are exchanged between DUTs.
- EIOS is 32'hBC7C7C7C for gen 1
- If the receiver fails to detect the required number of valid EIOS within 48ms, a timeout occurs, and the LTSSM returns to the Detect state.

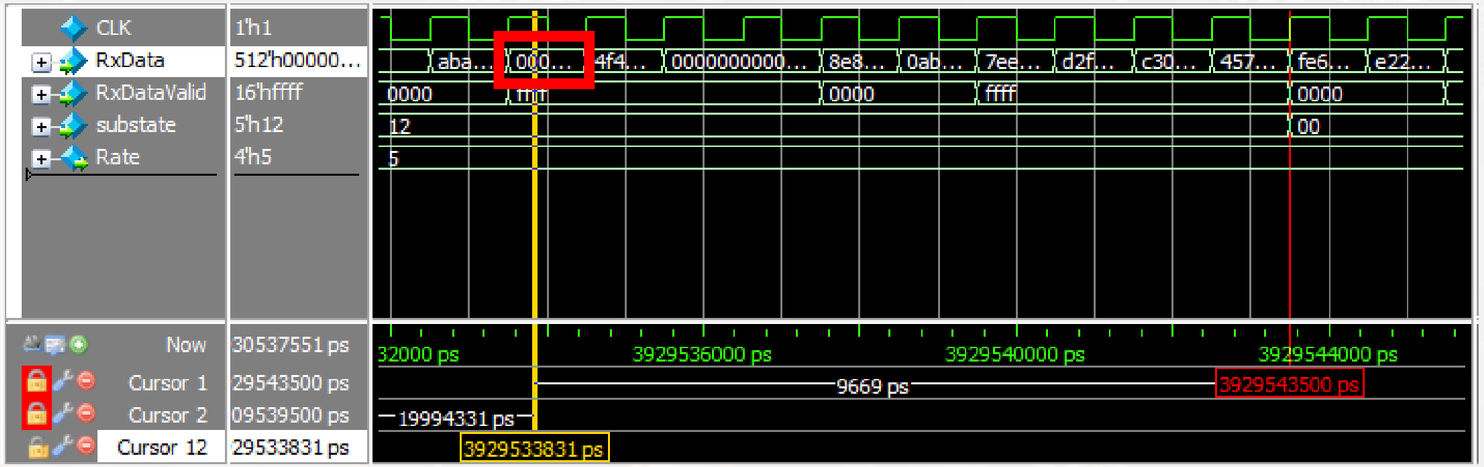
Error Injection

Simulation

- In this substate, TS1 ordered sets are exchanged between DUTs.
- Received TSs are scrambled, the COM of gen 5 is 8'h1E
- The reference clock frequency is significantly higher in Gen 5 operation compared to previous generations.
- If the receiver fails to detect the required number of valid TS1 ordered sets within 12ms, a timeout occurs, and the LTSSM returns to the Detect state.



12ms Timeout of Recovery.EQ (Phase 1)



2ms Timeout of

- In this substate, scrambled Idle OS are exchanged between DUTs.
- If the receiver fails to detect the required number of valid Idle OS within 2ms, a timeout occurs, and the LTSSM returns to the Detect state.
- After we enter GEN 5, timer module increases the number counts to reach any specific timeout, our Environment is the source of the clk for the devices so after we enter GEN5 we increase the clk frequency (clk time decreses) (acting as PLL in the procese)

09

Verification Results

Verification Results

Scoreboard Results

```
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 67  , RX_Data = 67
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 38  , RX_Data = 38
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 1e  , RX_Data = 1e
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 22  , RX_Data = 22
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = ff  , RX_Data = ff
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 6a  , RX_Data = 6a
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 73  , RX_Data = 73
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 83  , RX_Data = 83
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 18  , RX_Data = 18
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = e6  , RX_Data = e6
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = e6  , RX_Data = e6
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 20  , RX_Data = 20
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = e9  , RX_Data = e9
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 1e  , RX_Data = 1e
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 0a  , RX_Data = 0a
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = cb  , RX_Data = cb
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 2b  , RX_Data = 2b
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = 6e  , RX_Data = 6e
@ 109902500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_D_h [PCIe_Scoreboard2_D] TX_Data = ae  , RX_Data = ae
```

Verification Results

Scoreboard Results

```
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = ca  , RX_Data = ca
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 80  , RX_Data = 80
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = e7  , RX_Data = e7
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 47  , RX_Data = 47
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 2d  , RX_Data = 2d
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = c1  , RX_Data = c1
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = b4  , RX_Data = b4
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 47  , RX_Data = 47
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 26  , RX_Data = 26
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = fa  , RX_Data = fa
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 2d  , RX_Data = 2d
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = e8  , RX_Data = e8
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = c5  , RX_Data = c5
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 5a  , RX_Data = 5a
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 87  , RX_Data = 87
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 06  , RX_Data = 06
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 6e  , RX_Data = 6e
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = a8  , RX_Data = a8
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 98  , RX_Data = 98
@ 110077500: uvm_test_top.PCIe_Env_h.PCIe_Scoreboard2_U_h [PCIe_Scoreboard2_U] TX_Data = 21  , RX_Data = 21
```

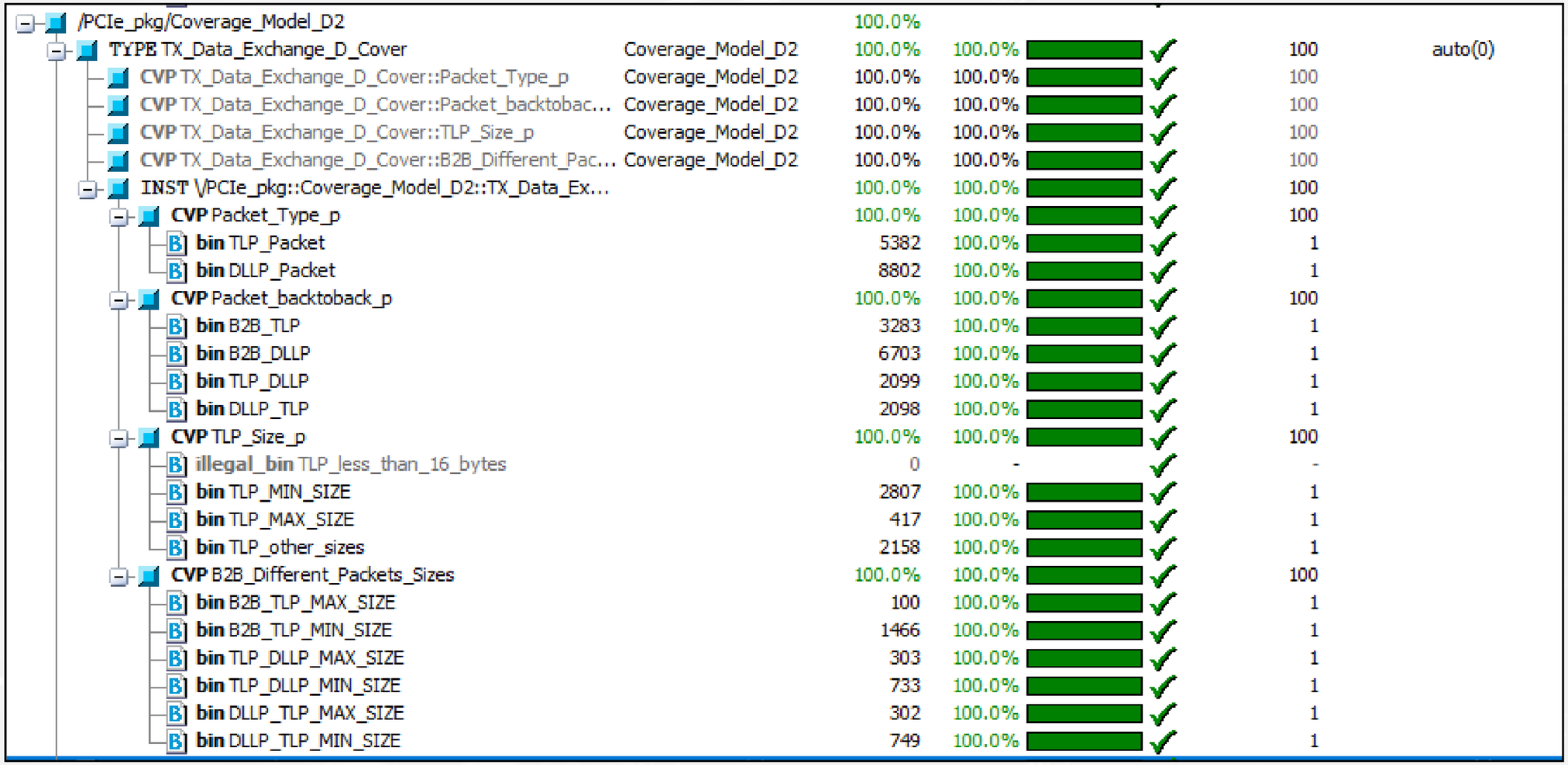
Verification Results

Scoreboard Results

```
[PCIe_Scoreboard2_D] ------------------------------------------------------------
[PCIe_Scoreboard2_D]          Scoreboard Summary Report For Data Integrity: TX Downstream -> RX Upstream
[PCIe_Scoreboard2_D] ------------------------------------------------------------
[PCIe_Scoreboard2_D]    Transaction Layer Packets (TLP) verified : 5373
[PCIe_Scoreboard2_D]    Data Link Layer Packets (DLLP) verified  : 8501
[PCIe_Scoreboard2_D]    All data matched the expected results
[PCIe_Scoreboard2_U] ------------------------------------------------------------
[PCIe_Scoreboard2_U]          Scoreboard Summary Report For Data Integrity: TX Upstream -> RX Downstream
[PCIe_Scoreboard2_U] ------------------------------------------------------------
[PCIe_Scoreboard2_U]    Transaction Layer Packets (TLP) verified : 5390
[PCIe_Scoreboard2_U]    Data Link Layer Packets (DLLP) verified  : 8597
[PCIe_Scoreboard2_U]    All data matched the expected results
```

Verification Results

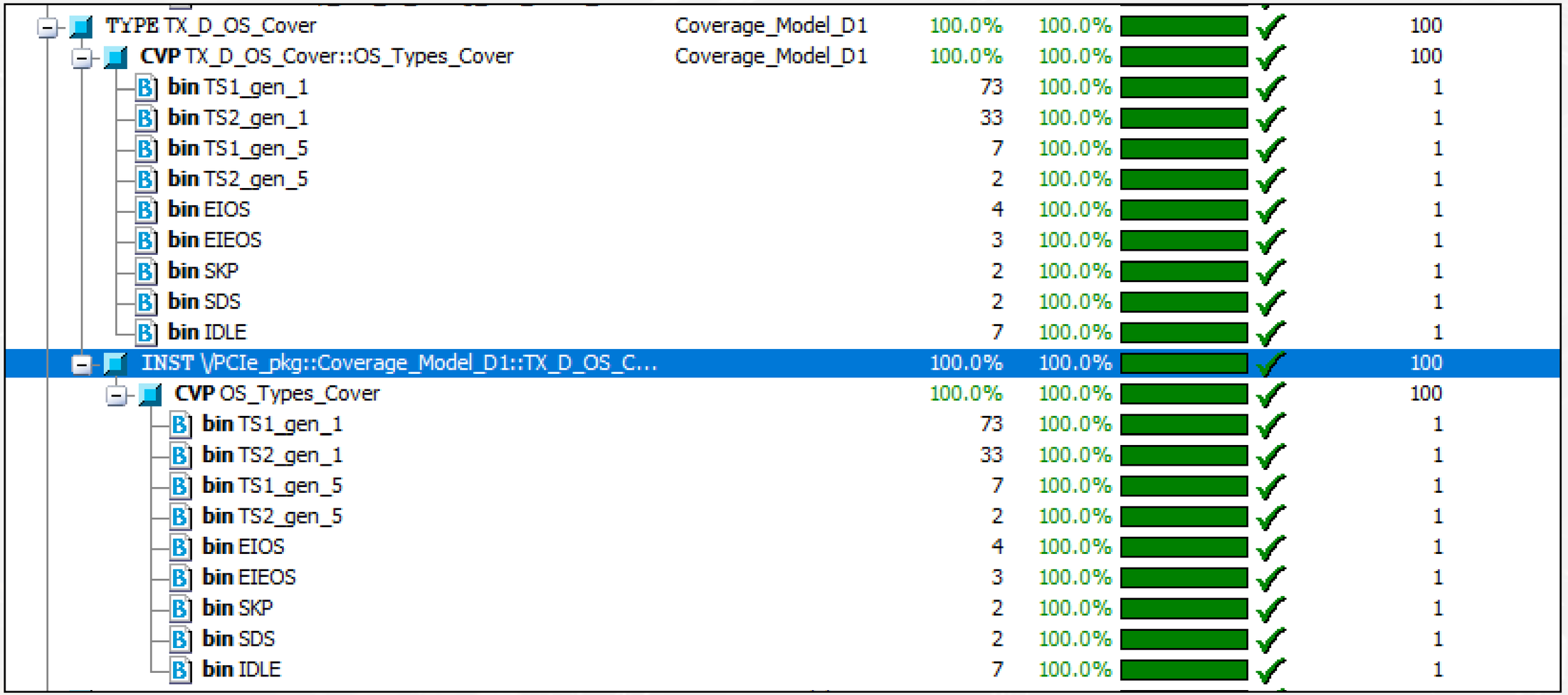
Functional Coverage



/PCIe_pkg/Coverage_Model_D2		100.0%				
TYPE TX_Data_Exchange_D_Cover	Coverage_Model_D2	100.0%	100.0%	✓	100	auto(0)
CVP TX_Data_Exchange_D_Cover::Packet_Type_p	Coverage_Model_D2	100.0%	100.0%	✓	100	
CVP TX_Data_Exchange_D_Cover::Packet_backtobac...	Coverage_Model_D2	100.0%	100.0%	✓	100	
CVP TX_Data_Exchange_D_Cover::TLP_Size_p	Coverage_Model_D2	100.0%	100.0%	✓	100	
CVP TX_Data_Exchange_D_Cover::B2B_Different_Pac...	Coverage_Model_D2	100.0%	100.0%	✓	100	
INST \/PCIe_pkg::Coverage_Model_D2::TX_Data_Ex...		100.0%	100.0%	✓	100	
CVP Packet_Type_p		100.0%	100.0%	✓	100	
bin TLP_Packet		5382	100.0%	✓	1	
bin DLLP_Packet		8802	100.0%	✓	1	
CVP Packet_backtoback_p		100.0%	100.0%	✓	100	
bin B2B_TLP		3283	100.0%	✓	1	
bin B2B_DLLP		6703	100.0%	✓	1	
bin TLP_DLLP		2099	100.0%	✓	1	
bin DLLP_TLP		2098	100.0%	✓	1	
CVP TLP_Size_p		100.0%	100.0%	✓	100	
illegal_bin TLP_less_than_16_bytes		0	-	✓	-	
bin TLP_MIN_SIZE		2807	100.0%	✓	1	
bin TLP_MAX_SIZE		417	100.0%	✓	1	
bin TLP_other_sizes		2158	100.0%	✓	1	
CVP B2B_Different_Packets_Sizes		100.0%	100.0%	✓	100	
bin B2B_TLP_MAX_SIZE		100	100.0%	✓	1	
bin B2B_TLP_MIN_SIZE		1466	100.0%	✓	1	
bin TLP_DLLP_MAX_SIZE		303	100.0%	✓	1	
bin TLP_DLLP_MIN_SIZE		733	100.0%	✓	1	
bin DLLP_TLP_MAX_SIZE		302	100.0%	✓	1	
bin DLLP_TLP_MIN_SIZE		749	100.0%	✓	1	

Verification Results

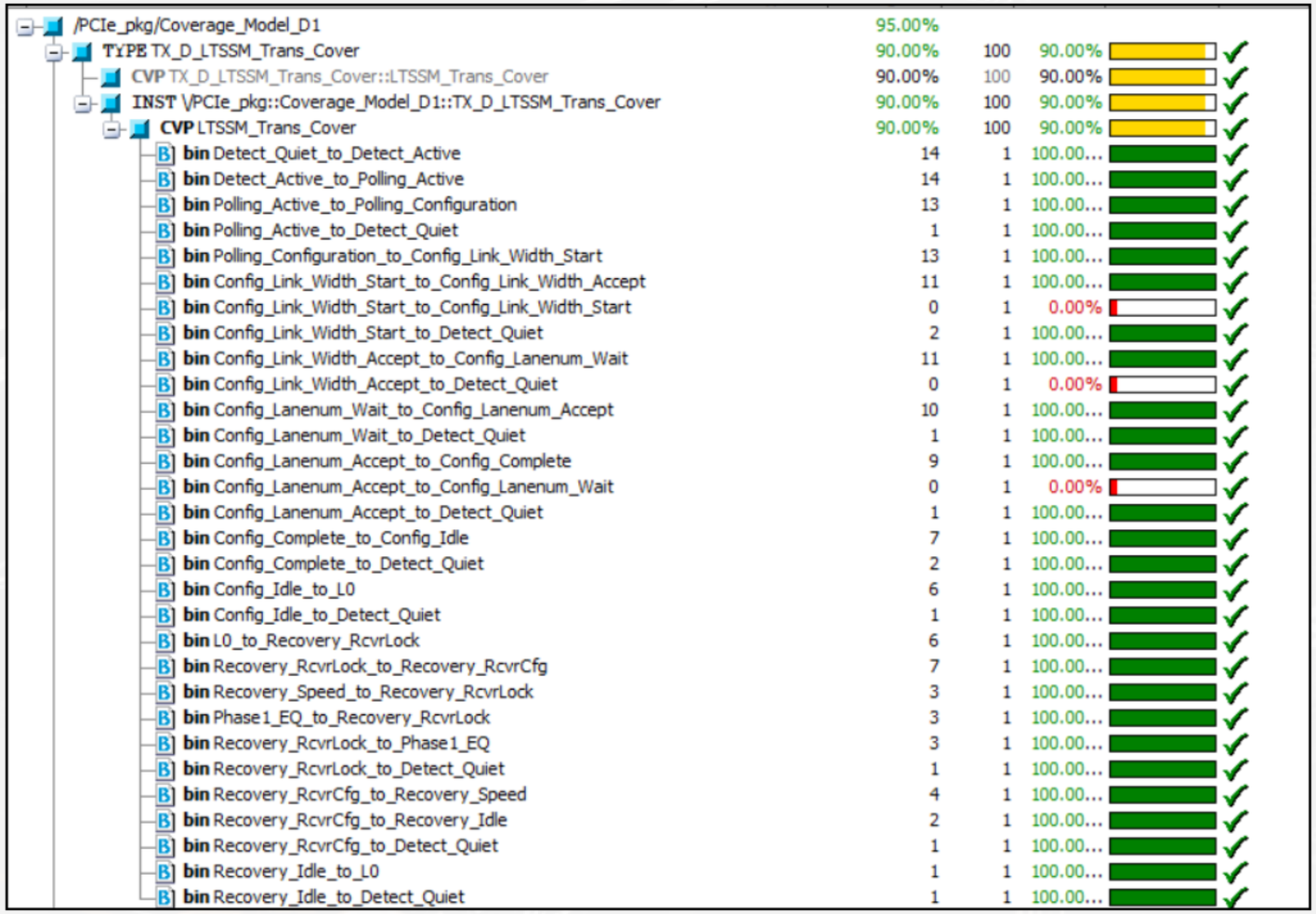
Functional Coverage



TYPE TX_D_OS_Cover	Coverage_Model_D1	100.0%	100.0%	✓	100
CVP TX_D_OS_Cover::OS_Types_Cover	Coverage_Model_D1	100.0%	100.0%	✓	100
bin TS1_gen_1		73	100.0%	✓	1
bin TS2_gen_1		33	100.0%	✓	1
bin TS1_gen_5		7	100.0%	✓	1
bin TS2_gen_5		2	100.0%	✓	1
bin EIOS		4	100.0%	✓	1
bin EIEOS		3	100.0%	✓	1
bin SKP		2	100.0%	✓	1
bin SDS		2	100.0%	✓	1
bin IDLE		7	100.0%	✓	1
INST \PCIe_pkg::Coverage_Model_D1::TX_D_OS_C...		100.0%	100.0%	✓	100
CVP OS_Types_Cover		100.0%	100.0%	✓	100
bin TS1_gen_1		73	100.0%	✓	1
bin TS2_gen_1		33	100.0%	✓	1
bin TS1_gen_5		7	100.0%	✓	1
bin TS2_gen_5		2	100.0%	✓	1
bin EIOS		4	100.0%	✓	1
bin EIEOS		3	100.0%	✓	1
bin SKP		2	100.0%	✓	1
bin SDS		2	100.0%	✓	1
bin IDLE		7	100.0%	✓	1

Verification Results

Functional Coverage



/PCIe_pkg/Coverage_Model_D1	95.00%		
TYPE TX_D_LTSSM_Trans_Cover	90.00%	100	90.00%
CVP TX_D_LTSSM_Trans_Cover::LTSSM_Trans_Cover	90.00%	100	90.00%
INST \/PCIe_pkg::Coverage_Model_D1::TX_D_LTSSM_Trans_Cover	90.00%	100	90.00%
CVP LTSSM_Trans_Cover	90.00%	100	90.00%
bin Detect_Quiet_to_Detect_Active	14	1	100.00...
bin Detect_Active_to_Polling_Active	14	1	100.00...
bin Polling_Active_to_Polling_Configuration	13	1	100.00...
bin Polling_Active_to_Detect_Quiet	1	1	100.00...
bin Polling_Configuration_to_Config_Link_Width_Start	13	1	100.00...
bin Config_Link_Width_Start_to_Config_Link_Width_Accept	11	1	100.00...
bin Config_Link_Width_Start_to_Config_Link_Width_Start	0	1	0.00%
bin Config_Link_Width_Start_to_Detect_Quiet	2	1	100.00...
bin Config_Link_Width_Accept_to_Config_Lanenum_Wait	11	1	100.00...
bin Config_Link_Width_Accept_to_Detect_Quiet	0	1	0.00%
bin Config_Lanenum_Wait_to_Config_Lanenum_Accept	10	1	100.00...
bin Config_Lanenum_Wait_to_Detect_Quiet	1	1	100.00...
bin Config_Lanenum_Accept_to_Config_Complete	9	1	100.00...
bin Config_Lanenum_Accept_to_Config_Lanenum_Wait	0	1	0.00%
bin Config_Lanenum_Accept_to_Detect_Quiet	1	1	100.00...
bin Config_Complete_to_Config_Idle	7	1	100.00...
bin Config_Complete_to_Detect_Quiet	2	1	100.00...
bin Config_Idle_to_L0	6	1	100.00...
bin Config_Idle_to_Detect_Quiet	1	1	100.00...
bin L0_to_Recovery_RcvrLock	6	1	100.00...
bin Recovery_RcvrLock_to_Recovery_RcvrCfg	7	1	100.00...
bin Recovery_Speed_to_Recovery_RcvrLock	3	1	100.00...
bin Phase1_EQ_to_Recovery_RcvrLock	3	1	100.00...
bin Recovery_RcvrLock_to_Phase1_EQ	3	1	100.00...
bin Recovery_RcvrLock_to_Detect_Quiet	1	1	100.00...
bin Recovery_RcvrCfg_to_Recovery_Speed	4	1	100.00...
bin Recovery_RcvrCfg_to_Recovery_Idle	2	1	100.00...
bin Recovery_RcvrCfg_to_Detect_Quiet	1	1	100.00...
bin Recovery_Idle_to_L0	1	1	100.00...
bin Recovery_Idle_to_Detect_Quiet	1	1	100.00...

Verification Results

Code Coverage

```
Total Coverage By File (code coverage only, filtered view): 94.2%
```

Verification Results

Reported Bugs

Bug ID	Description
Bug_1	Mismatch between Scrambler and Descrambler on the upstream and downstream devices, leading to misalignment.
Bug_2	COM character should always bypass scrambling and descrambling, regardless of whether it's a K-character or D-character.
Bug_3	In Gen1 or Gen2, when attempting to transmit TS1 or TS2 during the L0 state, the Ordered Sets (OS) are being descrambled, which is incorrect behavior.
Bug_4	Incorrect state transition from Phase 1 to Rx_Recovery.RcvrLock after equalization in the Main LTSSM.
Bug_5	Tx LTSSM issue: OS generator sends TS1 with EC=00 when the connected device is downstream, which is only valid for upstream direction.
Bug_6	At the start of Equalization, the upstream device must initiate by sending TS1 with EC=00.
Bug_7	In Phase 1, after the downstream device sends TS1 with EC=00, it should immediately exit the equalization process and notify upper layers of completion. However, the current design waits for TS1 with EC=01 from upstream, which is incorrect.
Bug_8	When in Phase 1, if the condition for transitioning to Recovery.RcvrLock is met, the design should go to that state. Instead, it transitions to an incorrect state in the Main LTSSM.
Bug_9	Missing linkup signal in the RX block: Certain RX functionalities depend on this signal, but it's not present in the current design ports.
Bug_10	In Gen3/4/5 data transfer, the asynchronous header should reference only the first symbol of the data block, but the RTL treats the en ↓ lock as part of the header.

Verification Results

Reported Bugs

Bug_11	In Gen3/4/5, the length field in the first block of the packet is not correctly encoded, which affects proper data interpretation.
Bug_12	Speed mismatch between Device1 and Device2: The link fails to retrain to the highest common speed, as expected per PCIe spec.
Bug_13	The LTSSM made transition to the config state although polling config condition is not met.
Bug_14	The LTSSM made transition to the Configuration.Idle state although Configuration.Complete condition is not met.
Bug_15	FS and LF are not assigned to output port in downstream TX_LTSSM, causing TX data to become xxxx.
Bug_16	The 32-bit LFSR for Gen3_4_5 updates on posedge, while LMC data updates on negedge, leading to outputting TX data twice per cycle.
Bug_17	Data in Recovery.Idle are not scrambled in both RX and TX.
Bug_18	TXs send incorrect SDS OS (Gen3 SDS instead of Gen5).
Bug_19	TXs send incorrect SKP OS (Gen3 SKP instead of Gen5).
Bug_20	TXs send SDS at L0.
Bug_21	The scrambler isn't synchronized with the descrambler in L0 due to an incorrect LFSR reset, impacting sequence generation.
Bug_22	Remaining data of TLP is not received; only the part containing STP is received.
Bug_23	When sending back-to-back max-sized TLPs, the remaining data of the first TLP is incorrect due to constant high valid signal affecting the length counter.

Verification Results

Reported Bugs

Bug_24	When transmitting four minimum-sized TLPs in a single transfer, the signal tlp_start incorrectly marks the start of the fourth TLP(generalize it).
Bug_25	The transition from Detect Quiet to Detect Active should occur either after 12 ms or when Rx Electrical Idle deasserts (goes low). However, Rx Electrical Idle never goes low, and the timer is incorrectly set to 0 ms, causing an immediate and invalid transition to Detect Active.
Bug_26	RX incorrectly flags a TLP as bad if its last byte matches 8'b1100_0000. This is wrong, since EDB should be four bytes of 8'b1100_0000, not just one. Only the full 4-byte EDB pattern should trigger error detection.
Bug_27	If the RX receives multiple adjacent DLLPs, it will only process the first one because the end of the first DLLP is determined when the first byte of the second DLLP is received, causing all subsequent data to be undetectable and incorrectly processed.
Bug_28	If the TX transmits a combination of TLP and DLLP in the same transfer, it mistakenly counts the DLLP as a TLP, resulting in incorrect data due to the insertion of an unintended STP.
Bug_29	At Recovery CFG, after timeout, the environment enters state "Detect" correctly, but the RTL does not.
Bug_30	Time out doesn't asserted after 48ms exactly
Bug_31	At Recovery Speed, after timeout, the environment enters state "Detect" correctly, but the RTL does not.
Bug_32	At Phase 1, after timeout, the environment enters "Recovery Speed" correctly, but the RTL does not.

Verification Results

Reported Bugs

Bug_33	RTL doesn’t go to detect after 12ms exactly
Bug_34	When we arrive detect state ,TxdetectRx still asserted to high
Bug_35	After returning to Detect from state L0 and detecting RX TS, the next state's TS contains an incorrect ID in symbol 6.
Bug_36	After returning from states beyond L0 to Detect, the system stops at the Detect state.
Bug_37	After timeout and returning to Detect to attempt link-up again, the upstream device fails to progress beyond state L0.
Bug_38	In the Polling.Active state, if the correct number of TS packets is not received within the timeout period, the DUT's main LTSSM incorrectly moves to the next state instead of returning to Detect. This issue has been fixed in the RTL.
Bug_39	In the Polling.Config state, the DUT’s LTSSM also proceeds to the next state after a timeout, even when it hasn't received the required number of TS packets. It should return to Detect, but it doesn’t. This issue has also been fixed in the RTL.
Bug_40	During the Polling.Active state, there was not enough validation on the received TS1s. As a result, the DUT could transition to the next substate even if the TS1s were corrupted. This has been corrected in the RTL.
Bug_41	Missing TS1 Validation on All Lanes. Spec requires all lanes to transmit and receive valid TS1s. Our test passed even when only Lane 0 had valid TS, others were invalid.

Future Work

- Extend the testbench to include directed tests and assertions for the newly added substates.
- Implement the full ASIC flow up to GDSII, including synthesis, place and route.
- Re-verify the post-layout netlist with the UVM environment to ensure functionality is preserved after physical design.
- Perform static timing analysis (STA) on the synthesized netlist and verify that timing constraints are met.
- Implement unsupported LTSSM substates (e.g., Loopback, Hot Reset) and verify their behavior in the UVM environment.

جامعة الإسكندرية
ALEXANDRIA UNIVERSITY

PCI EXPRESS

Thank You

- Abd-Elrahman Mohammed
- Ibrahim Awad
- Marwan Mohamed
- Mostafa Masoud Ali
- Moustafa Mohammed
- Sohaib Alaraby
- Yasser Mohamed
- Yousef Ahmed Ali