Mentor®
A Siemens Business

# Hardware Specification
for
PCIe Physical Layer Gen5

Prepared by Design Team

February 2021

Contents

List of Figures

List of Tables

Revision history

Revision Number	Date	Description
0.1	13/1/2021	Initial Draft
1.0	25/1/2021	First resealed of specification
1.5	8/2/2021	Second resealed of specification

Chapter 1

Introduction

1.1 Overview

PCIe (peripheral component interconnect express) is an interface standard for connecting high-speed components. Every desktop PC motherboard has a number of PCIe slots you can use to add GPUs (aka video cards aka graphics cards), RAID cards, Wi-Fi cards or SSD (solid-state drive) add-on cards. The types of PCIe slots available in the PC will depend on the motherboard that we bought. PCIe slots come in different physical configurations: x1, x4, x8, x16, x32. The number after the x tells how many lanes (how data travels to and from the PCIe card) that PCIe slot has. A PCIe x1 slot has one lane and can move data at one bit per cycle. A PCIe x2 slot has two lanes and can move data at two bits per cycle (and so on).

PCIe x1 card can be inserted into a PCIe x16 slot, but that card will receive less bandwidth. Similarly, a PCIe x8 card can be inserted into a PCIe x4 slot, but it'll only work with half the bandwidth compared to if it was in a PCIe x8 slot.

PCIe has undergone several large and smaller revisions, improving on performance and other features. So there are several generations and in this spec we will design and implement the physical layer of gen 5. The official PCIe 5.0 standard came out in May 2019. It will bring 128 GB/s of throughput. The specification is backwards compatible with previous PCIe generations and also includes new features, including electrical changes to improve signal integrity and backward-compatible CEM connectors for add-in cards. Before designing and implementing the architecture of gen 5, there is an overview to discuss the general basic concepts of PCIe.

1.2 PCIe

Definition 1.2.1 *PCIe is an interface standard for connecting high-speed components. It is a serial bus model.*

There were many problems limiting the performance of the parallel bus:

- Flight time must be less than the clock period or the model won't work.
- Clock skew
- Signal skew

But the serial transport got over these problems for example flight time becomes nonissue as the clock that will latch the data into the receiver is built into the data stream and no external reference clock, so for the same reason no clock skew. Also, signal skew is eliminated within a lane because there is only one bit of data being sent.

1.3 Lane

Definition 1.3.1 *A lane is composed of two differential signaling pairs, one pair for receiving data and the other for transmitting. Thus, each lane is composed of four wires.*

Number of lanes is called "link width" and is represented as x1, x2, x4, x8, x16 and x32. The tradeoff regarding the number of lanes: more lanes increase the bandwidth of the link but it also increases the cost, space requirement and power consumption.

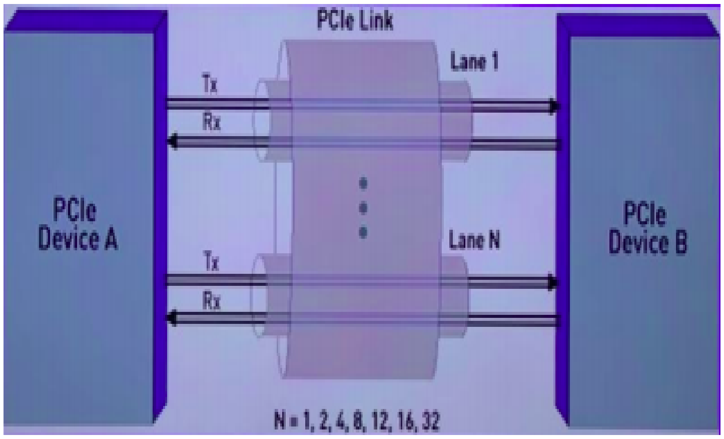

Figure 1.1: PCIe Link

In the receiver, PLL circuit (phase locked loop) 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 frequency is increased or decreased until a match is obtained, so the output clock frequency precisely matches the clock that was used to transmit the data.

Each lane uses differential signaling, this improves noise immunity and reduced signal voltage. Moreover, anything that will affect the signal will also affect the other by about the same amount and in the same direction so the receiver won't be affected by the noise that affects the signals and will be able to distinguish the bits.

1.4 Generations Speeds

Table 1.1: PCIe Generations Speeds for link width x16

Version	Bandwidth	Gigatransfer	Frequency
PCIe 1.0	8 GB/s	2.5 GT/s	2.5 GHz
PCIe 2.0	16 GB/s	5 GT/s	5 GHz
PCIe 3.0	32 GB/s	8 GT/s	8 GHz
PCIe 4.0	64 GB/s	16 GT/s	16 GHz
PCIe 5.0	128 GB/s	32 GT/s	32 GHz

Bandwidth calculations for link width x1:

$$PCIe_{B.W_{Gen1}} = (2.5Gb/s{\times}2 \quad directions)/10 \quad bits \quad per \quad symbol = 0.5 \quad GB/s \tag{1.1}$$

$$PCIe_{B.W_{Gen2}} = (5Gb/s \times 2 \quad directions)/10 \quad bits \quad per \quad symbol = 1 \quad GB/s \tag{1.2}$$

$$PCIe_{B.W_{Gen3}} = (8Gb/s \times 2 \quad directions)/8 \quad bits \quad per \quad byte = 1 \quad GB/s \tag{1.3}$$

1.5 Topology

A Topology is composed of point-to-point Links that interconnect a set of components. This figure 1.2 illustrates a single fabric instance referred to as a hierarchy – composed of a Root Complex (RC), multiple Endpoints (I/O devices), a Switch, and a PCI Express to PCI/PCI-X Bridge, all interconnected via PCI Express Links.

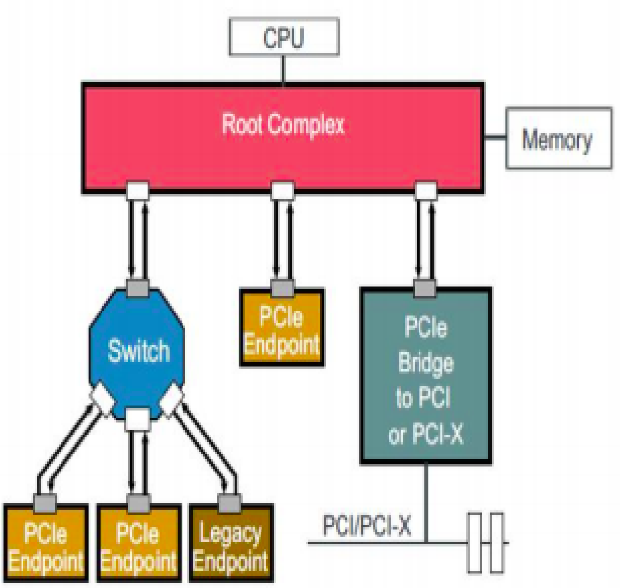

Figure 1.2: Topology

- Root complex: it is the interface between the system CPU and the PCIe topology
  - Root complex may support one or more PCI Express Ports. Each interface defines a separate hierarchy domain. Each hierarchy domain may be composed of a single Endpoint or a sub-hierarchy containing one or more Switch components and Endpoints.
  - The capability to route peer-to-peer transactions between hierarchy domains through a Root Complex is optional and implementation dependent.
- Switch: allows more devices to be attached to a single PCIe port, they act as a packet routers and recognize which path a packet will need to take based on its address or other routing information (may have several downstream ports but only one upstream port).

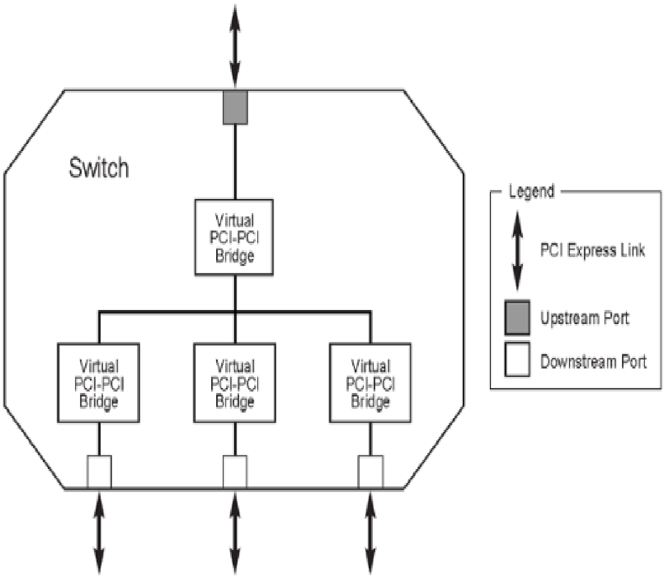

Figure 1.3: switch

All Switches are governed by the following base rules:

- Switches appear to configuration software as two or more logical PCI-to-PCI Bridges.
- Switch must forward all types of Transaction Layer Packets between any set of Ports.
- Switch is not allowed to split a packet into smaller packets.

- Bridge: provides interface to other buses such as PCI or PCI-X or even another PCIe bus.
  Forward bridge ⟶ allows older card to be plugged into a new system.
  Reverse bridge ⟶ allows a new PCIe card to be plugged into an old PCI system.
- End points: devices that act as initiators and completers of transactions on the bus (they only implement a single upstream port). Endpoints are classified as either legacy, PCI Express, or Root Complex Integrated Endpoints.

Root complex will appear to configuration software as PCI bus number zero and the PCIe ports will appear as PCI to PCI bridges. In a similar way, a switch will appear to software simply as a collection of bridges sharing a common bus. The advantage of this approach is that it allows the transaction routing to take place in the same way it did for PCI.

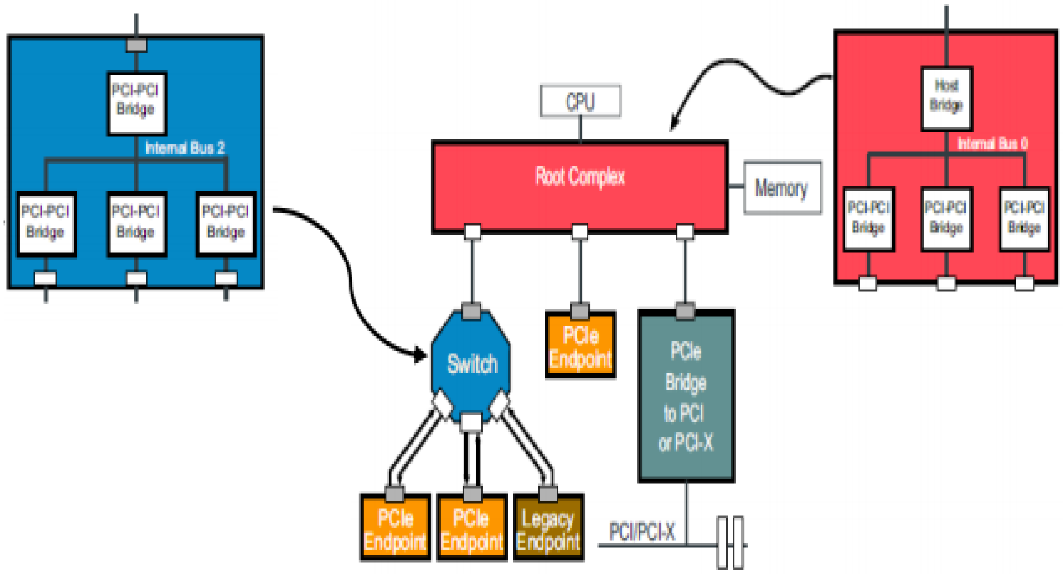

Figure 1.4: configuration software for PCIe

1.6 Device layers

The architecture of PCIe device is divided into three discrete logical layers: Transaction Layer, Data Link Layer and Physical Layer. Each of these layers is divided into two sections: one that processes outbound (to be transmitted) information and one that processes inbound (received) information.

- Transaction layer: creation of transaction layer packet (TLP) on the transmit side and decoding on the receiver side. Also responsible for other 3 functions which are flow control, quality of service and transaction ordering functionality.
- Data link layer: creation of data link layer packet (DLLP) on the transmit side and decoding on the receiver side. Also, responsible for link error detection and correction.
- Physical layer: creation of ordered set packet on transmit side and decoding on the receiver side. It processes all 3 types of packets to be transmitted on the link and processes packets received from the link also. Then packets are encoded and serialized.
- Then link training and status state machine (LTSSM) of the physical layer is responsible for link initialization and training.

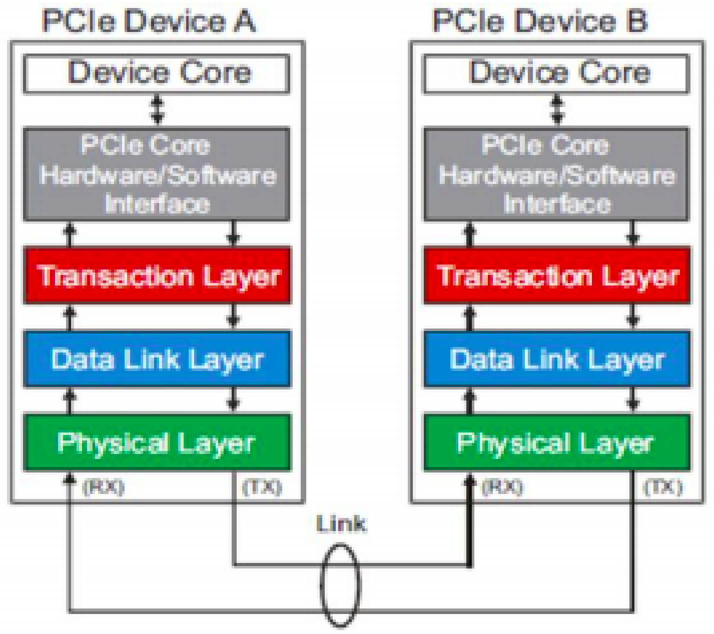

Figure 1.5: Device A and B connect by Link

- Note: switch port needs to implement all the layers as it evaluates the contents of packets, to determine their routing requires looking into the internal details of a packet and that takes place in the transaction layer.

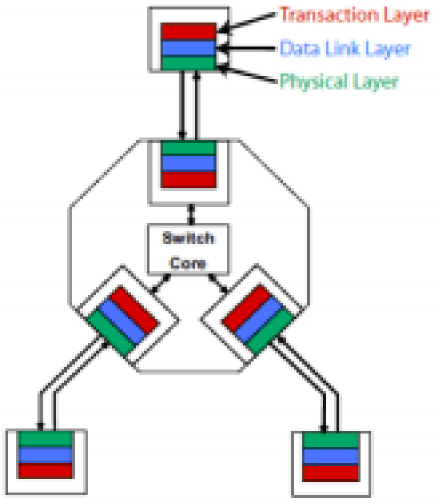

Figure 1.6: Switch port

1.7 layers interaction

- The contents of an outgoing request or completion packet from the device are assembled in the transaction layer based on information presented by device core logic. That information would usually include the type of command desired, the address of the target device and amount of data to transfer.

- The newly created packet is then stored in a buffer called a virtual channel until it is ready for passing to the next layer. When the packet is passed down to data link layer, additional information is added to the packet for error checking at the neighboring receiver and a copy is stored locally so we can send it again if a transmission error occurs.

- When the packet arrives at the physical layer, it is encoded and transmitted differentially using all available lanes of the link.

- When the packet arrives at the receiver, it decodes the incoming bits in the physical layer and check for errors that can be seen at this level.

- if there are no errors, then it forwards the resulting packet up to the data link layer.

- Again the resulting packet is checked, if no errors, then it will be forwarded up to the transaction layer.

- The packet is buffered, checked for errors and disassembled into the original information so the contents can be delivered to the device core of the receiver.

1.8 TLP packet

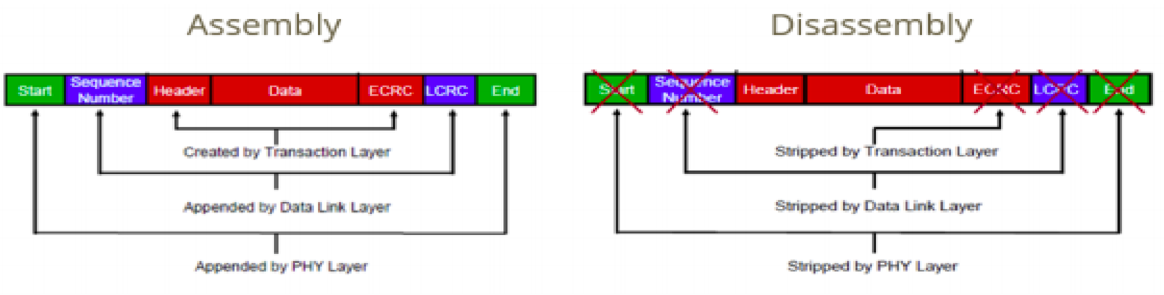

Figure 1.7: TLP

1.9 DLLP packet

They are transferred between data link layers of 2 neighboring devices on a link and the transaction layer isn't aware of these packets. Its size is 8 bytes (very small compared to TLPs).

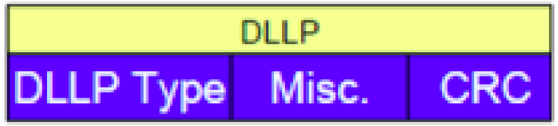

Figure 1.8: DLLP

Note: this DLLP differs from the structure of TLP at the data link layer

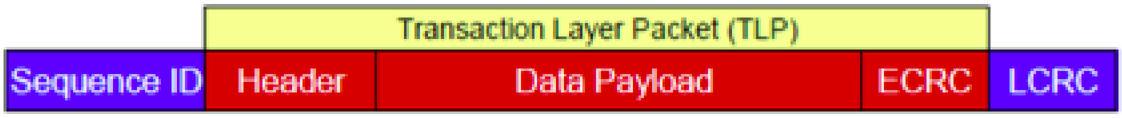

Figure 1.9: TLP at data link layer

1.10 Physical layer

It is divided into 2 portions:

1. Logical ⟶ contains the digital logic associated with preparing the packets for serial transmission on the link and reversing the process for inbound packets.
2. Electrical ⟶ the analog interface that connects to the link and consists of differential drivers and receivers for each lane.

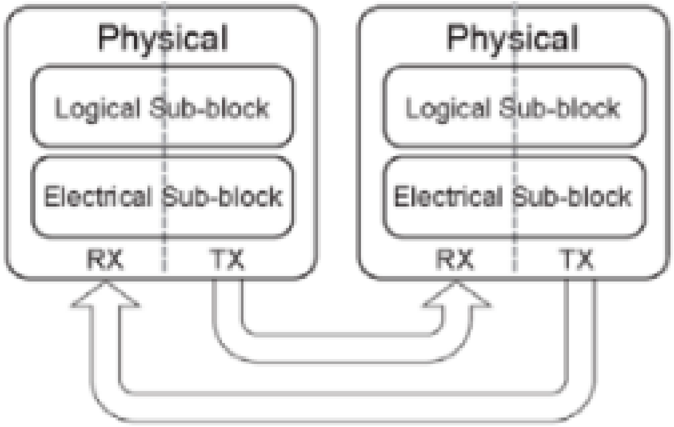

Figure 1.10: Physical Layer

1.11 Logical sub-block

It has two main sections: Transmit section that prepares outgoing information passed from the Data Link Layer for transmission by the electrical sub-block, and Receiver section that identifies and prepares received information before passing it to the Data Link Layer. The logical sub-block and electrical sub-block coordinate the state of each Transceiver through a status and control register interface or functional equivalent. The logical sub-block directs control and management functions of the Physical Layer. PCI Express uses 8b/10b encoding when the data rate is 2.5 GT/s or 5.0 GT/s. For data rates greater than or equal to 8.0 GT/s, it uses a per-lane code along with physical layer encapsulation.

Logical sub-block contains mainly two logical blocks: MAC layer and PHY.

1.12 PIPE Architecture

- Physical Interface for PCI Express Specification (PIPE) developed by Intel, has the stated intent of providing a standard interface between the internal logic of a PCIe design and the analog and high-speed circuitry required to implement the serial link. The purpose of this functional separation is to allow ASIC and integrated circuit designers to focus on the PCI Express device core, Transaction, Data Link and logical Physical Layers, while relying on the PIPE-compliant physical design (PHY) for the electrical interface of the design.
- The PIPE spec defines standard functionality that a PIPE-compliant PHY needs to implement, as well as a standard parallel interface between the PHY and the internal logic referred to in the spec as MAC.

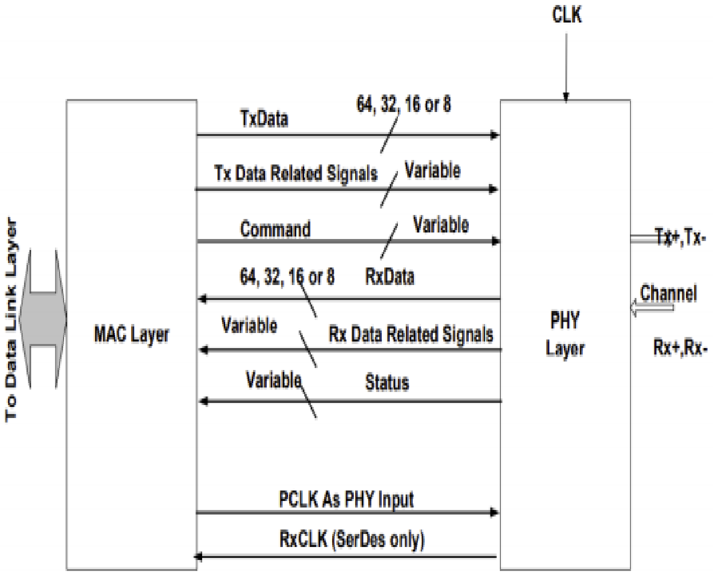

Figure 1.11: PHY/MAC Interface

1.12.1 MAC Architecture

The MAC contains many of the PCIe logical Physical Layer circuits (such as the Link Training and Status State Machine (LTSSM), data scrambling, byte striping and functions as the bridge between the DLL and the PHY/MAC interface)

1.12.2 PHY/MAC Interface

It is a parallel interface for transferring data to be transmitted on the PCIe bus. The width of this parallel interface for bytes of data is shown as either be 8, 16, 32 or 64 bits in each direction.

1.12.3 PHY Architecture

- It contains the 8b/10b encoder and decoder, elastic buffer, serializer and deserializer
- It contains the logic that controls the receiver detection and reports the detect status to the MAC via the PHY/MAC Interface.
- It contains a Phase Lock Loop (PLL) to generate the internal, high speed clocks used for the PHY based on the CLK input.

1.12.4 PIPE PLL

It generates the PCLK used in synchronizing the parallel PHY/MAC Interface based on the CLK input.

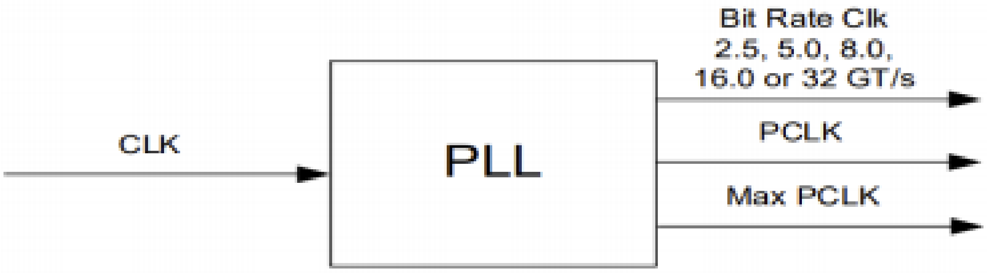

Figure 1.12: PLL

1.12.5 LPIF Architecture

The LPIF specifications defines common interface between the Link Layer and the logical physical layer to facilitate interoperability, design and validation re-use between Link Layers and Physical layers.

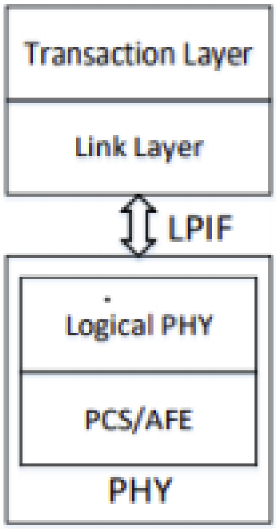

Figure 1.13: LPIF

Chapter 2

Features

In this chapter, we will discuss the blocks and the states that we are going to support, design and implement.

2.1 Supported Features

Table 2.1: supported states

Supported States
Detect
Polling
Configuration
Recovery
L0
L0s

Table 2.2: supported Blocks

Supported Blocks
LPIF error handler
LPIF control
TxBuffer
Ordered sets
LTSSM
Byte Stripping / Unstripping
Scrambler / Descrambler
PIPE register
Arbiter/Mux

The LTSSM is composed of the following 11 states: Detect, Polling, Configuration, Recovery, L0, L0s, L1, L2, Hot Reset, Loopback, Disable. So here is a quick overview about the supported states:

1. Detect
   It is the initial state of the physical layer, only used at Gen1 2.5 GT/s rate, or converted from the data link layer, or after reset, or from other states (Disable, Polling, Configuration, Recovery, etc.) Conversion. In short, the Detect state is the beginning of PCIe link training. In addition, Detect, as the name suggests, needs to implement detection work. Because in this state, the transmitting end TX needs to detect whether the receiving end RX exists and can work normally, if the detection is normal, it can enter other states. The Detect state mainly includes two sub-states: Detect.Quiet and Detect.Active.

2. Polling
   The purpose of this state is to "pair the cipher" and achieve barrier-free communication. After entering this state, the TS1 and TS2 OS sequences are sent between TX and RX to determine Bit Lock, Symbol Lock and solve the problem of Lane polarity reversal. The Polling state mainly includes three sub-states: Polling.Active, Polling.Configuration, Polling.Compliance.

3. Configuration
   The content of this state is very simple. It is to determine the Link/Lane number by sending TS1 and TS2. Configuration contains 6 sub-states: Configuration.LinkWidth.Start, Configuration.LinkWidth.Accpet, Configuration.Lanenum.Wait, Configuration. Lanenum.Accpet, Configuration.Complete, Configuration.Idle

4. Recovery

When the PCIe link needs to be retrained, it enters the Recovery state. There are mainly the following situations:

- When the PCIe link signal finds an error, the Bit Lock and Symbol Lock need to be adjusted
- Exit from L0s state
- Speed Change. Because the first time you enter the L0 state, the rate is 2.5GT/s. When you need to adjust the rate to 5.0GT/s or 8.0GT/s, you need to enter the Recovery state for Speed Change. At this stage, Bit Lock, Symbol Lock, etc. Need to reacquire
- Need to readjust the Width of PCIe link
- Need to readjust the Width of PCIe link
- Only in Gen3 and Gen4, Equalization needs to be performed again.

5. L0
   This is the normal state (Normal and Full-Active State) of the link, and all TLP, DLLP and Ordered Sets can be sent and received normally. In this state, the rate can be 2.5GT/s or 5GT/s or higher (if the devices at both ends of the link support it and have undergone Re-Training).

6. L0s
   The ASPM (active-state power management) state is mainly used to reduce power consumption, and can enter this state when the bus is idle, and can quickly switch back to the L0 state from this state. When in the L0 state, when EIOS appears on the link, it means that it is about to enter the L0s state. When in the L0s state, when FTS appears on the link, the link will quickly complete bit lock and symbol lock, and enter the L0 state.

2.2 Supported Blocks

2.2.1 TxBuffer

The ASPM (active-state power management) state is mainly used to reduce power consumption, and can enter this state when the bus is idle, and can quickly switch back to the L0 state from this state. When in the L0 state, when EIOS appears on the link, it means that it is about to enter the L0s state. When in the L0s state, when FTS appears on the link, the link will quickly complete bit lock and symbol lock, and enter the L0 state.

2.2.2 Byte Stripping

Bytes are striped across multiple lanes ⟶ distribution of each byte to different lanes in turn to avoid different lanes with different data lengths.

2.2.3 Scrambler

PCIe uses pseudo random data scrambling to spread off the RF energy in the frequency spectrum ⟶ less electromagnetic interference (EMI). This is done by using a linear feedback shift registers on both the sender and receiver side. lfsr maintains an internal state machine which is the top flip-flops. An XOR operation is done between the state of the lfsr and the data ⟶ resulting in a predictable and repeatable sequence of pseudo random data. A parallel Scrambler performs 2 things: advance and apply. Advance means that the pattern changes, while apply means the XORing operation performed with this pattern.

2.2.4 Ordered sets

They are physical layer packets that only exists between Tx and Rx's physical layer.

- TS1OS and TS2OS (Training Sequence 1 and 2)
- FTSOS (Fast Training Sequence)
- EIEOS (Electrical Idle Exit)
- EIOS (Electrical Idle)
- SOS (SKIP)

For more details about each block and how all the blocks interact together, see chapter 4.

Chapter 3

Signal Interface

3.1 PIPE Interface

Table 3.1: TX Related signals

Name	I/O	Active Level	Description
TxData [MAXPIPEWIDTH* LANESNUMBER-1:0]	OUT	N/A	Parallel data input bus.
TxDataValid [LANESNUMBER-1:0]	OUT	N/A	This signal allows the MAC to instruct the PHY to ignore the data interface for one clock cycle. A value of one indicates the phy will use the data, a value of zero indicates the phy will not use the data.
TxElecIdle[LANESNUMBER-1:0]	OUT	High	Forces Tx output to electrical idle when asserted except in loopback. Note: The MAC must always have TxDataValid asserted when TxElecIdle transitions to either asserted or deasserted; TxDataValid is a qualifier for TxElecIdle sampling.
TxDataK [(MAXPIPEWIDTH/8)* LANESNUMBER-1:0]	OUT	N/A	A value of zero indicates a data byte, a value of 1 indicates a control byte.
TxStartBlock [LANESNUMBER-1:0]	OUT	N/A	This signals allow the MAC to tell the PHY the starting byte for a 128b block when value is 1. The starting byte for a 128b block must always start with byte 0 of the data interface.
TxSyncHeader [2*LANESNUMBER -1:0]	OUT	N/A	Provides the sync header for the PHY to use in the next 130b block. 01 ⟶control byte 10 ⟶data byte
TxDetectRx/ Loopback [LANESNUMBER-1:0]	OUT	High	Used to tell the PHY to begin a receiver detection operation or to begin loopback

Table 3.2: RX Related signals

<table>
<tr><th>Name</th><th>I/O</th><th>Active Level</th><th>Description</th></tr>
<tr><td>RxData [MAXPIPEWIDTH* LANESNUMBER-1:0]</td><td>IN</td><td>N/A</td><td>Parallel data output bus.</td></tr>
<tr><td>RxDataValid [LANESNUMBER-1:0]</td><td>IN</td><td>N/A</td><td>This signal allows the PHY to instruct the MAC to ignore the data interface for one clock cycle. A value of one indicates the mac will use the data, a value of zero indicates the mac will not use the data.</td></tr>
<tr><td>RxDataK [(MAXPIPEWIDTH/8)* LANESNUMBER-1:0]</td><td>IN</td><td>N/A</td><td>A value of zero indicates a data byte, a value of 1 indicates a control byte.</td></tr>
<tr><td>RxStartBlock [LANESNUMBER-1:0]</td><td>IN</td><td>N/A</td><td>This signals allow the PHY to tell the MAC the starting byte for a 128b block when value is 1. The starting byte for a 128b block must always start with byte 0 of the data interface.</td></tr>
<tr><td>RxSyncHeader [2*LANESNUMBER -1:0]</td><td>IN</td><td>N/A</td><td>Provides the sync header for the MAC to use in the next 130b block.

01 ⟶control byte
10 ⟶data byte</td></tr>
<tr><td>RxValid [LANESNUMBER-1:0]</td><td>IN</td><td>High</td><td>Indicates symbol lock and valid data on RxData and RxDataK and further qualifies RxDataValid when used.</td></tr>
<tr><td>RxStatus [3*LANESNUMBER -1:0]</td><td>IN</td><td>N/A</td><td>Encodes receiver status and error codes for the received data stream when receiving data.
<table>
<tr><td>[0]</td><td>[1]</td><td>[2]</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Received data OK</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 SKP added</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1 SKP removed</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Receiver detected</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Both $8B/10B$ ($128B/130B6$) decode error and (optionally) Receive Disparity error</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Elastic Buffer overflow</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Elastic Buffer underflow.</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Receive disparity error (Reserved if Receive Disparity error is reported with code 0b100)</td></tr>
</table>
</td></tr>
</table>

RXElecIdle	IN	High	Indicates receiver detection of an electrical idle. While deasserted with the PHY in P2 (PCI Express mode) or the PHY in P0, P1, P2, or P3 (USB Mode), indicates detection of either: PCI Express Mode: a beacon.

Table 3.3: Clk and reset

Name	I/O	Active Level	Description
Reset#	IN	Low	Resets the transmitter and receiver. This signal is asynchronous.
CLK	IN	Edge	This differential Input is used to generate the bit-rate clock for the PHY transmitter and receiver.
phy_reset	OUT	Low	Resets phy.

Table 3.4: Commands and Status signals

<table>
<tr><th>Name</th><th>I/O</th><th>Active Level</th><th>Description</th></tr>
<tr><td>PowerDown[3:0]</td><td>OUT</td><td>N/A</td><td>Power up or down the transceiver. Power states PCI Express Mode:
<table>
<tr><td>[0]</td><td>[1]</td><td>[2]</td><td>[3]</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>P0, normal operation</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>P0s, low recovery time latency, power saving state</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>P2, lowest power state</td></tr>
</table></td></tr>
<tr><td>Rate[3:0]</td><td>OUT</td><td>N/A</td><td>Control the link signaling rate.
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 2.5 GT/s signaling rate</td></tr>
<tr><td>1</td><td>Use 5 GT/s signaling rate</td></tr>
<tr><td>2</td><td>Use 8 GT/s signaling rate</td></tr>
<tr><td>3</td><td>Use 16 GT/s signaling rate</td></tr>
<tr><td>4</td><td>Use 32 GT/s signaling rate</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>PhyStatus</td><td>IN</td><td>High</td><td>Used to communicate completion of several PHY functions including stable PCLK and/or Max PCLK (depending on clocking mode) after Reset# deassertion, power management state transitions, rate change, and receiver detection.</td></tr>
<tr><td>Width[1:0]</td><td>OUT</td><td>N/A</td><td>Controls the PIPE data path width.
<table>
<tr><td>Value</td><td>Datapath Width</td></tr>
<tr><td>0</td><td>8 bits</td></tr>
<tr><td>1</td><td>16 bits</td></tr>
<tr><td>2</td><td>32 bits</td></tr>
</table></td></tr>
<tr><td>PclkChangeAck</td><td>OUT</td><td>High</td><td>Only used when PCLK is a PHY input. Asserted by the MAC when a PCLK rate change or rate change or, if required, width change is complete and stable.
After the MAC asserts PclkChangeAck the PHY responds by asserting PhyStatus for one cyle and deasserts PclkChangeOk at the same time as PhyStatus. The controller shall deassert PclkChangeAck when PclkChangeOk is sampled low.</td></tr>
<tr><td>PclkChangeOk</td><td>IN</td><td>High</td><td>Only used when PCLK is a PHY input. Asserted by the PHY when it is ready for the MAC to change the PCLK rate or Rate.</td></tr>
</table>

Table 3.5: Message Bus Interface Signals

Name	Direction	Description
M2P_MessageBus [7:0]	Input	The MAC multiplexes command, any required address, and any required data for sending read and write requests to access PHY PIPE registers and for sending read completion responses and write ack responses to PHY initiated requests.
P2M_MessageBus [7:0]	Output	The PHY multiplexes command, any required address, and any required data for sending read and write requests to access MAC PIPE registers and for sending read completion responses and write ack responses to MAC initiated requests.

Table 3.6: MAC Interface(in/out) Equalization signals

<table>
<tr><th>Name</th><th>I/O</th><th>Description</th></tr>
<tr><td>LocalTxPresetCoefficients
[18*LANESNUMBER -1:0]</td><td>IN</td><td>Reads the current equalization values of the transmitter port PCI Express Mode: These are the coefficients for the preset on the LocalPresetIndex[3:0] after a GetLocalPresetCoeffcients request:
[5:0] C-1
[11:6] C0
[17:12] C+1
Valid on assertion of LocalTxCoefficientsValid.
The MAC will reflect these coefficient values on the TxDeemph bus when MAC wishes to apply this preset. These signals are only used by a PHY that requires dynamic preset coefficient updates.</td></tr>
<tr><td>TXDeemph
[18*LANESNUMBER -1:0]</td><td>OUT</td><td>When rate is 2.5 or 5.0 GT/s
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>-6dB de-emphasis</td></tr>
<tr><td>1</td><td>-3.5dB de-emphasis</td></tr>
<tr><td>2</td><td>No de-emphasis</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>
PIPE implementations that only support 2.5 GT/s do not implement this signal. PIPE PHY implementations that do not support low swing are not required to support the no-de- emphasis mode. When the rate is 8.0 GT/s
[5:0] C -1
[11:6] C 0
[17:12] C +1
Note: The MAC must ensure that only supported values are used for TxDeemph.</td></tr>
<tr><td>LocalFS
[6*LANESNUMBER -1:0]</td><td>IN</td><td>PCI Express Mode:
Provides the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates.
This value shall be sampled by the MAC only when PhyStatus is pulsed after RESET or on the first PhyStatus pulse after a rate change to 8 GT/s.</td></tr>
<tr><td>LocalLF
[6*LANESNUMBER -1:0]</td><td>IN</td><td>PCI Express Mode:
Provides the LF value for the PHY. This signal is only used by a PHY that requires dynamic preset coefficient updates. This value must only be sampled by the MAC only when PhyStatus is pulsed after RESET or on the first PhyStatus pulse after a rate change to 8 GT/s.</td></tr>
</table>

GetLocalPresetCoeffcients [LANESNUMBER -1:0]	OUT	PCI Express Mode: A MAC holds this signal high for one PCLK cycle requesting a preset to co-efficient mapping for the preset on LocalPresetIndex[3:0] to coefficients on LocalTxPresetCoefficient[17:0] Maximum Response time of PHY is 128 nSec. Note. A MAC can make this request any time after reset. Note. After a local preset coefficient request a MAC could assert GetLocalPresetCoeffcients again as soon as the next PCLK after LocalTxCoefficientsValid deasserts. This signal is only used with a PHY that requires dynamic preset coefficient updates
LocalTxCoefficientsValid [LANESNUMBER -1:0]	IN	PCI Express Mode: A PHY holds this signal high for one PCLK cycle to indicate that the LocalTxPresetCoefficients[17:0] bus correctly represents the coefficients values for the preset on the LocalPresetIndex bus. This signal is only used by a PHY that requires dynamic preset coefficient updates
LF [LANESNUMBER -1:0]	OUT	PCI Express Mode: Provides the LF value advertised by the link partner. The MAC shall only change this value when a new LF value is captured during link training. A PHY may optionally consider this value when deciding how long to evaluate TX equalization settings of the link partner. These signals are only used at the 8.0 GT/s signaling rate
RxEqEval [LANESNUMBER -1:0]	OUT	PCI Express Mode: The PHY starts evaluation of the far end transmitter TX EQ settings when the MAC asserts this signal. This signal is only used at the 8.0 GT/s signaling rate.

LocalPresetIndex [4*LANESNUMBER -1:0]	OUT	PCI Express Mode: Index for local PHY preset coefficients requested by the MAC The preset index value is encoded as follows: 0000b – Preset P0. 0001b – Preset P1. 0010b – Preset P2. 0011b – Preset P3. 0100b – Preset P4. 0101b – Preset P5. 0110b – Preset P6. 0111b – Preset P7. 1000b – Preset P8. 1001b – Preset P9. 1010b – Preset P10. 1011b – Reserved 1100b – Reserved 1101b – Reserved 1110b – Reserved 1111b – Reserved. These signals are only These signals are only used with a PHY that requires dynamic preset coefficient updates.

InvalidRequest [LANESNUMBER -1:0]	OUT	PCI Express Mode: Indicates that the Link Evaluation feedback requested a link partner TX EQ setting that was out of range. The MAC asserts this signal when it detects an out of range error locally based on calculated link partner transmitter coefficients based on the last valid link equalization feedback or it receives a NACK response from the link partner. The MAC keeps the sigal asserted until the next time it asserts RxEQEval. When a MAC asserts this signal it shall subsequently ask the PHY to perform an RxEQ evaluation using the last valid setting a second time. This signal is only used at the 8.0 GT/s signaling rate.
LinkEvaluationFeedbackDirectionChange [6*LANESNUMBER -1:0]	IN	PCI Express Mode: Provides the link equalization evaluation feedback in the direction change format. Feedback is provided for each coefficient: [1:0] C -1 [3:2] C 0 [5:4] C 1 The feedback value for each coefficient is encoded as follows: 00 - No change 01 – Increment 10 – Decrement 11 - Reserved A PHY does not implement these signals if it is does not provide link equalization evaluation feedback using the Direction Change format. Note: In 8.0 GT/s mode the MAC shall ignore the C 0 value and use the correct value per the PCI Express specification. These signals are only used at the 8.0 GT/s signaling rate. Note that C -1 and C 1 are encoded as the absolute value of the actual FIR coefficient and thus incrementing or decrementing either value refers to the magnitude of the actual FIR coefficient. For example – if C -1 is 000001b the FIR coefficient is negative one and a request to increment C -1 will increase it in the direction of 000002b which decreases the FIR coefficient.

3.2 LPIF Inteface

Table 3.7: LPIF Interface Signals

Name	I/O	Active Level	Description
pl_trdy	OUT	High	LIndicates Physical Layer is ready to accept data. Data is accepted when pl_trdy, lp_valid, and lp_irdy are asserted together.
lp_irdy	IN	High	Link Layer to Physical Layer indicating Link Layer is ready to transfer data. lp_irdy must not be presented by the upper layers when pl_state_sts is RESET.
lp_valid [64 -1:0]	IN	High	Link Layer to Physical Layer indicates data valid on the corresponding lp_data bytes. 'LP_NVLD' equals the number of valid bits. The bytes of lp_data associated with a specific bit of lp_valid is implementation specific. When lp_irdy is asserted, at least one of the bits of lp_valid must be asserted.
pl_data [512 -1:0]	OUT	N/A	Physical Layer to Link Layer Data, where 'NBYTES' equals number of bytes determined by the supported data bus for the LPIF interface.
pl_valid [64 -1:0]	OUT	High	Physical Layer to Link Layer indicates data valid on pl_data. 'PL_NVLD' equals the number of valid bits. The bytes of pl_data associated with a specific bit of pl_valid is implementation specific.
lp_data[512 -1:0]	IN	N/A	Link Layer to Physical Layer Data, where 'NBYTES' equals number of bytes determined by the data width for the LPIF instance.

lp_state_req[3:0]	IN	N/A	Link Layer Request to Logical Physical Layer to request state change. Encodings as follows: 0000: NOP 0001: Active 0010: Active.L0s 0011: Deepest Allowable PM State [L1 Substates only] 0100: L1.1 0101: L1.2 0110: L1.3 0111: L1.4 1000: L2 1001: LinkReset 1010: Reserved 1011: Retrain 1100: Disable
pl_state_sts[3:0]	OUT	N/A	Physical Layer to Link Layer Status indication of the Interface. Encodings as follows: 0000: NOP 0001: Active 0010: Active.L0s 0011: Deepest Allowable PM State [L1 Substates only] 0100: L1.1 0101: L1.2 0110: L1.3 0111: L1.4 1000: L2 1001: LinkReset 1010: Reserved 1011: Retrain 1100: Disable
pl_speedmode[2:0]	OUT	N/A	Current Link Speed as negotiated by the logPHY (3’b000=Gen1,3’b001=Gen2,3’b010=Gen3, 3’b011=Gen4, 3’b100=Gen5, rest=Rsvd) Link Layer should only consider this to be relevant when pl_state_sts=RETRAIN or ACTIVE.
lp_force_detect	IN	High	This is a level signal. It forces logPHY to shut down the receiver, drive and keep the physical LTSSM in Detect.

pl_dlpstart[64-1]	OUT	N/A	logPHY indicates the start of a Data Link Layer packet for Gen3 and above speeds. Each bit corresponds to a specific data byte depending on the configuration of the port.
pl_dlpend[64-1]	OUT	N/A	logPHY indicating the end of a Data Link Layer packet for Gen3 and above speeds. Each bit corresponds to a specific data byte depending on the configuration of the port.
pl_tlpstart[64-1]	OUT	N/A	logPHY indicating the start of a Transaction Layer packet for Gen3 and above speeds (STP).
pl_tlpend[64-1]	OUT	N/A	logPHY indicating the end of a Transaction Layer packet for Gen3 and above speeds (END).
pl_tlpedb[64-1]	OUT	N/A	logPHY indicating EDB received for Gen3 and above speeds.
lp_dlpstart[64-1]	IN	N/A	Indicates from Link Layer start of dlp
lp_dlpend[64-1]	IN	N/A	Indicates from Link Layer end of dlp
lp_tlpstart[64-1]	IN	N/A	Indicates from Link Layer start of tlp
lp_tlpend[64-1]	IN	N/A	Indicates from Link Layer end of tlp
lp_tlpedb[64-1]	IN	N/A	Indicates from Link Layer end o of bad tlp

Chapter 4

Architecture

4.1 MAC Layer Architecture

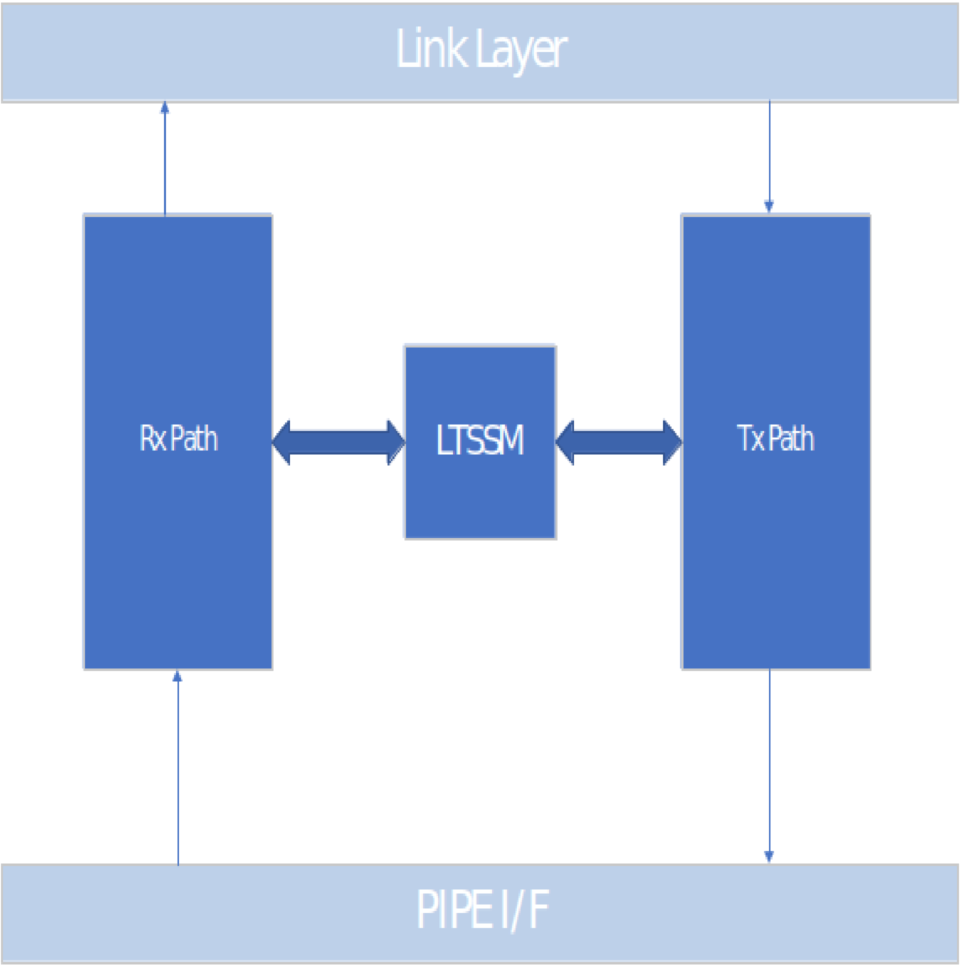

Figure 4.1: MAC Layer Architecture

Link Parameters:

- Number of Lanes: up to x16
- PIPE Width: up to 32
- Data Path: 512
- PIPE Per GEN
- The layer works on one clock which is the pclk.

4.2 LTSSM Architecture

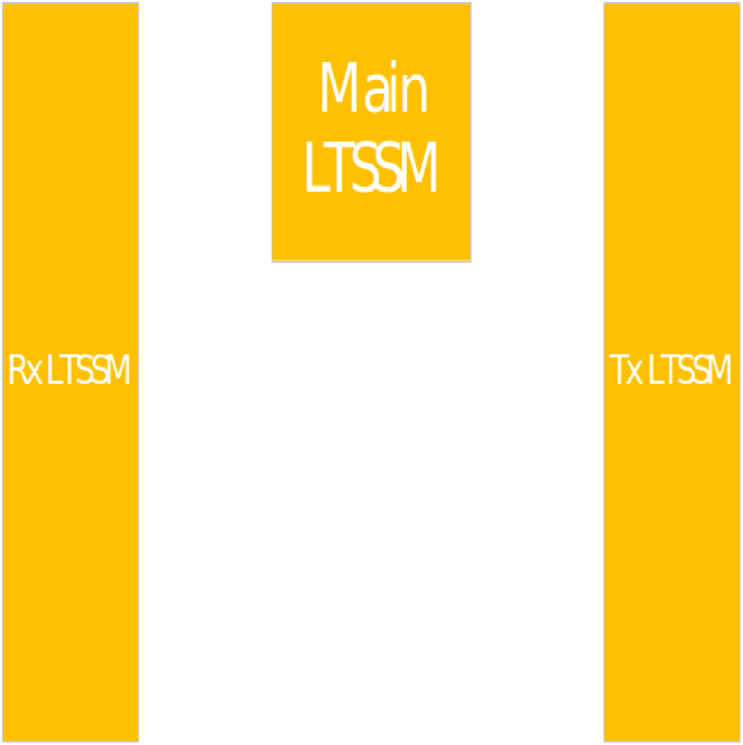

Figure 4.2: LTSSM Design

LTSSM is divided into two main parts:

- Two substates :
  - TX LTSSM
  - RX LTSSM
- Main LTSSM : Represents spec states and Controls sub-states

4.3 TX Path Architecture

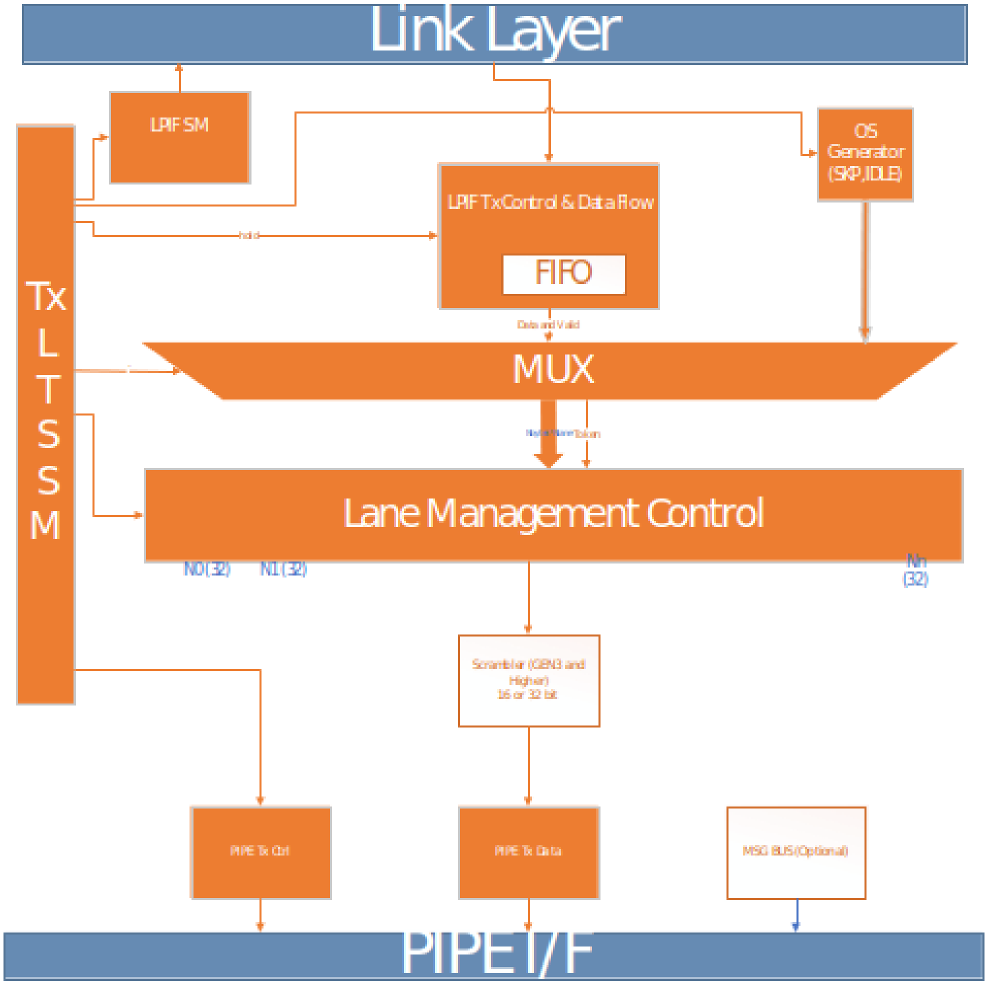

Figure 4.3: TX Path

Modules:

- Tx LTSSM : Controls all LTSSM functions for the Tx path.
- LPIF SM : Handles LPIF LTSSM.
- LPIF Tx Ctrl & DF:
    - Handles LPIF Interface.

  - STP, SDP Generation.
  - Data Valid Output.

- OS Generator:
  - Ordered Sets, Idle, SKP.
  - Data Valid Output

- MUX : multiplex between the data packets and ordered sets.

- Lane Management Control : perform byte Stripping on lanes.

- Scrambler.

- PIPE Tx Data
  - Output data.
  - Generate valid, sync header, start of block.

- PIPE Tx Ctrl : Pclk handshake, pass rate, pclk rate, width, txdetectrx

- MSG BUS

4.4 RX Path Architecture

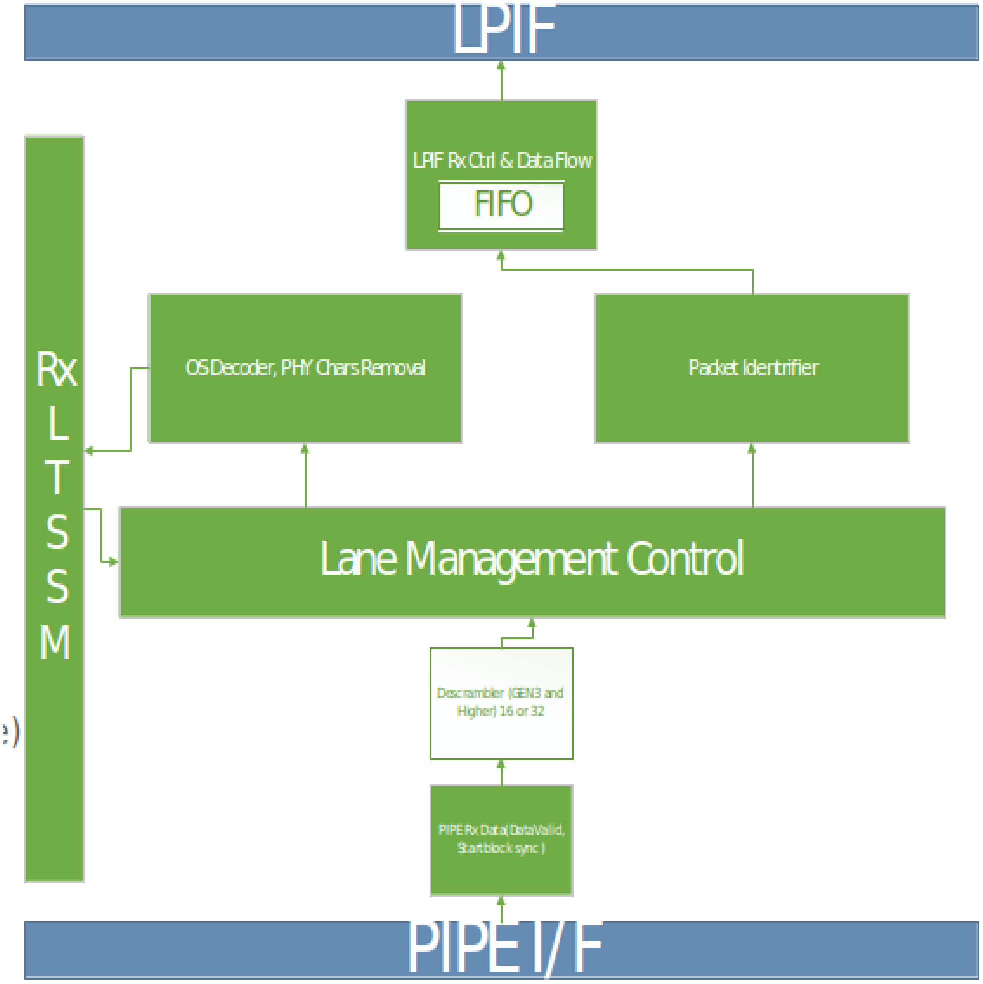
LPIF
LPIF Rx Ctrl & Data Flow
FIFO
Rx
L
T
S
S
M
OS Decoder, PHY Chars Removal
Packet Identrifier
Lane Management Control
:)
Descrambler (GEN3 and Higher) 16 or 32
PIPE Rx Data(DataValid, Startblock sync)
PIPE I/F

Figure 4.4: RX Path

RX modules:

- Rx LTSSM: Controls all LTSSM functions for the Rx path.
- PIPE Rx Data: Extract data wrt to header, valid, sb.
- Descrambler.
- LMC: Un stripe data no matter its type

- OS Decoder, Char Removal:
  - Remove SKP, PAD, IDLE
  - Decode TS and give data to LTSSM
- Packet Identifier:
  - Remove STP, SDP and add sop, eop
  - Forward packets only to Rx DF
- LPIF Rx Ctrl & DF:
  - Forwards data to LPIF in 512 bit width (if applicable)

Chapter 5

Scenarios

5.1 TX Scenarios

5.1.1 Sending TLP or DLLP

1. Data (TLP or DLLP) comes from Data Link Layer and is stored in the FIFO.
2. In GEN 1&2 FIFO Appends STP (in case of TLP), SDP (in case of DLLP) and END Symbols to TLP or DLLP and sends to Mux data, data valid and d/k signal .
3. In Gen 3&4&5 FIFO Appends STP token (in case of TLP), SDP token (in case of DLLP) and EDS token.
4. In GEN 3&4&5, if there is an order set (except skip order set) after data stream, FIFO should append EDS token.
5. Tx LTSSM states that we are in L0 state.
6. Tx LTSSM informs the MUX to take the output coming from the FIFO.
7. Tx LTSSSM informs the Lane Management Control that data coming from the output of MUX is TLP or DLLP.
8. Data is scrambled using scrambler.
9. In GEN 1&2 Pipe Tx Data outputs the data and assert data valid & datak.
10. In GEN 3&4&5, Pipe Tx Data outputs the data and assert data valid and adds sync header and start of block before sending data to the PIPE interface.

5.1.2 Sending Ordered Sets

1. Tx LTSSM based on current state will send to os generator the Type of order set it needs to send, OS fields, and count of how many times it will send it.
2. Tx LTSSM sends to FIFO to hold the data inside it.
3. Tx LTSSM informs the MUX to take the OS generator output.
4. Tx LTSSM informs Lane Management Control that data coming from the output of MUX is order set.
5. Link management control should distribute the OS on all lanes as per spec.
6. All order sets bypass scrambling except for symbols from 1 to 15 for TS order sets in GEN 3&4&5.
7. In GEN 1&2, Pipe Tx Data outputs the OS and assert data valid & datak.
8. In GEN 3&4&5, Pipe Tx Data outputs the OS and assert data valid and adds sync header and start of block before sending OS to the PIPE interface.

5.2 RX Scenario

5.2.1 Receiving TLP or DLLP

1. PIPE Rx data checks data valid is asserted or not.
2. For GEN 3&4&5, PIPE Rx data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.
3. Descrambles the data using descrambler.
4. Link control management un-stripes data bytes.
5. In GEN 1&2 Packet identifier will take the output of the link management Control and will remove STP (in case of TLP), SDP (in case of DLLP) and END and set sop and eop.
6. In GEN 3&4&5 Packet identifier will take the output of the link management Control and will remove STP token (in case of TLP), SDP token (in case of DLLP)
7. In GEN 3&4&5, in case of DLLP after removing SDP token we will forward the next 6 symbols to the FIFO.
8. In GEN 3&4&5, in case of TLP after removing STP token we will forward the next (length − 1) x 4 symbols to the FIFO.
9. FIFO forwards data to datalink layer.

5.2.2 Receiving Ordered Sets

1. Order set come on all lanes simultaneously.
2. PIPE Rx data checks data valid is asserted or not.
3. For GEN 3&4&5, PIPE Rx data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.
4. All order sets bypass descrambling except for symbols from 1 to 15 for TS order sets in GEN 3&4&5.
5. Link control management un-stripes data bytes.
6. OS Decoder will decode the order set and give its information to Rx LTSSM

5.2.3 Linkup scenario

Configuration state:

1. LinkWidth.Start
   Transmitter Side

   (a) (Downstream) Tx LTSSM will specify link numbers and PAD lane numbers then pass it to OS generator to generate TS1 ordered set.

   (b) (Upstream) Tx LTSSM will send same link numbers received in the receiver side and PAD lane numbers then pass it to OS generator to generate TS1 ordered set.

   (c) Tx LTSSM will inform the MUX to pass the TS1OS.

   (d) Tx LTSSM will inform LMC to distribute the TS1OS on all lanes simultaneously.

   (e) The TS1OS will pass through Scrambler then out to the PHY through the PIPE Tx data.

   Receiver Side

   (a) The PIPE Rx Data checks data valid is asserted or not.

   (b) For GEN 3&4&5, PIPE Rx Data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.

   (c) All order sets bypass Descrambler except for symbols from 1 to 15 for TS ordered sets in GEN 3&4&5.

   (d) The LMC un-stripes the data.

   (e) The OS Decoder will decode the TS1OS and give its information to Rx LTSSM.

(f) (Upstream) Rx LTSSM takes the link and lanes numbers then it forwards it to Tx LTSSM through Main LTSSM to generate TS1OS with same values to send it to the Downstream port and after receiving 2 TS1OS with same link and lane numbers the LTSSM goes to Configuration.LinkWidth.Accept substate.

(g) (Downstream) Upon Rx LTSSM receives 2 TS1OS with same link and lane numbers the LTSSM goes to Configuration.LinkWidth.Accept substate.

2. LinkWidth.Accept
   Transmitter Side

   (a) (Downstream) Tx LTSSM initiates lane numbers with same link number then pass it to OS generator to generate TS1 ordered set.

   (b) (Upstream) Tx LTSSM will send its lane numbers and same link number then pass it to OS generator to generate TS1 ordered set.

   (c) Tx LTSSM will inform the MUX to pass the TS1OS.

   (d) Tx LTSSM will inform LMC to distribute the TS1OS on all lanes simultaneously.

   (e) The TS1OS will pass through Scrambler then out to the PHY through the PIPE Tx data.

   Receiver Side

   (a) The PIPE Rx Data checks data valid is asserted or not.

   (b) For GEN 3&4&5, PIPE Rx Data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.

   (c) All order sets bypass Descrambler except for symbols from 1 to 15 for TS ordered sets in GEN 3&4&5.

   (d) The LMC un-stripes the data.

   (e) The OS Decoder will decode the TS1OS and give its information to Rx LTSSM.

   (f) (Upstream) Rx LTSSM takes the link number and forward it to Tx LTSSM through Main LTSSM to generate TS1OS with same link numbers and with its lane numbers to send it to the Downstream port the LTSSM goes to Configuration.Lanenum.Wait substate.

   (g) (Downstream) Doesn't wait to receive TS1OS as after sending TS1OS with non-PAD link and lane numbers the LTSSM goes to Configuration.Lanenum.Wait substate.

3. Configuration.Lanenum.Wait
   Transmitter Side

(a) (Downstream) Tx LTSSM keeps the same lane numbers and link number as in previous states and pass it to OS generator to generate TS1 ordered set.

(b) (Upstream) Tx LTSSM will keep sending its lane numbers and same link number then pass it to OS generator to generate TS1 ordered set.

(c) Tx LTSSM will inform the MUX to pass the TS1OS.

(d) Tx LTSSM will inform LMC to distribute the TS1OS on all lanes simultaneously.

(e) The TS1OS will pass through Scrambler then out to the PHY through the PIPE Tx data.

Receiver Side

(a) The PIPE Rx Data checks data valid is asserted or not.

(b) For GEN 3&4&5, PIPE Rx Data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.

(c) All order sets bypass Descrambler except for symbols from 1 to 15 for TS ordered sets in GEN 3&4&5.

(d) The LMC un-stripes the data.

(e) The OS Decoder will decode the TS1OS & TS2OS and give its information to Rx LTSSM.

(f) (Upstream) After Rx LTSSM receives 2 TS2OS with same link and lane numbers the LTSSM goes to Configuration.Lanenum.Accept substate.

(g) (Downstream) After Rx LTSSM receives 2 TS1OS with same link and lane numbers the LTSSM goes to Configuration.Lanenum.Accept substate.

4. Configuration.Lanenum.Accept
   Transmitter Side

(a) (Downstream) Tx LTSSM keeps the same lane numbers and link number as in previous states and pass it to OS generator to generate TS1 ordered set.

(b) (Upstream) Tx LTSSM will keep sending its lane numbers and same link number then pass it to OS generator to generate TS1 ordered set.

(c) Tx LTSSM will inform the MUX to pass the TS1OS.

(d) Tx LTSSM will inform LMC to distribute the TS1OS on all lanes simultaneously.

(e) The TS1OS will pass through Scrambler then out to the PHY through the PIPE Tx data.

Receiver Side

(a) The PIPE Rx Data checks data valid is asserted or not.

(b) For GEN 3&4&5, PIPE Rx Data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.

(c) All order sets bypass Descrambler except for symbols from 1 to 15 for TS ordered sets in GEN 3&4&5.

(d) The LMC un-stripes the data.

(e) The OS Decoder will decode the TS1OS & TS2OS and give its information to Rx LTSSM.

(f) (Upstream) After Rx LTSSM receives 2 TS2OS with same link and lane numbers the LTSSM goes to Configuration.Complete substate.

(g) (Downstream) After Rx LTSSM receives 2 TS1OS with same link and lane numbers the LTSSM goes to Configuration.Complete substate.

5. Configuration.Complete
   Transmitter Side

   (a) (Downstream & Upstream) Tx LTSSM will send the negotiated lane numbers and link number as in previous states and pass it to OS generator to generate TS2 ordered set.

   (b) Tx LTSSM will inform the MUX to pass the TS2OS.

   (c) Tx LTSSM will inform LMC to distribute the TS2OS on all lanes simultaneously.

   (d) The TS2OS will pass through Scrambler then out to the PHY through the PIPE Tx data.

   Receiver Side

   (a) The PIPE Rx Data checks data valid is asserted or not.

   (b) For GEN 3&4&5, PIPE Rx Data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.

   (c) All order sets bypass Descrambler except for symbols from 1 to 15 for TS ordered sets in GEN 3&4&5.

   (d) The LMC un-stripes the data.

   (e) The OS Decoder will decode the TS2OS and give its information to Rx LTSSM.

   (f) (Downstream & Upstream) After Rx LTSSM receives 8 TS2OS and Tx LTSSM send 16 TS2OS with the negotiated link and lane numbers the LTSSM goes to Configuration.Idle substate.

6. Configuration.Idle
   Transmitter Side

(a) (Downstream & Upstream) Tx LTSSM will inform OS generator to send Idle symbols.

(b) Tx LTSSM will inform the MUX to pass the Idle symbol.

(c) Tx LTSSM will inform LMC to distribute the Idle symbol on all lanes simultaneously.

(d) The Idle symbol will pass through Scrambler then out to the PHY through the PIPE Tx data.

Receiver Side

(a) The PIPE Rx Data checks data valid is asserted or not.

(b) For GEN 3&4&5, PIPE Rx Data checks for the sync header to know whether packet received is data (TLP or DLLP) or order set.

(c) All order sets bypass Descrambler except for symbols from 1 to 15 for TS ordered sets in GEN 3&4&5.

(d) The LMC un-stripes the data.

(e) The OS Decoder will decode the Idle symbol then inform the Rx LTSSM that Idle symbol is received.

(f) (Downstream & Upstream) After Rx LTSSM receives 8 Idle symbols and Tx LTSSM send 16 Idle symbols then Linkup signal is asserted and training is successfully completed.