IMPLEMENTATION OF PCS OF PHYSICAL LAYER FOR PCI EXPRESS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

VLSI Design and Embedded System

By

M. DINESH KUMAR

ROLL No: 207EC207

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2007-2009

IMPLEMENTATION OF PCS OF PHYSICAL LAYER FOR PCI EXPRESS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

VLSI Design and Embedded System

By

M. DINESH KUMAR

ROLL No: 207EC207

Under the Guidance of

Prof. D. P. ACHARYA

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2007-2009

National Institute Of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, **“Implementation of PCS Physical Layer for PCI Express”** submitted by **M. DINESH KUMAR** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics & Communication Engineering** with specialization in **“VLSI Design and Embedded System”** at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. D. P. Acharya
Dept. of Electronics & Communication Engg.
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people (especially my family) who supported me and believed in me.

I am thankful to**,** Professor **D. P. Acharya** in the dept. of Electronics and Communication Engineering, NIT Rourkela for giving me the opportunity to work under him and lending every support at every stage of this project work. I truly appreciate and value him esteemed guidance and encouragement from the beginning to the end of this thesis. I am indebted to his for having helped me shape the problem and providing insights towards the solution. His trust and support inspired me in the most important moments of making right decisions and I am glad to work with him.

I want to thank all my teachers **Dr. K. K. Mahapatra, Prof. G.Panda, Prof. G.S. Rath, Prof. S.K. Patra,** and **Prof. S. Meher** for providing a solid background for my studies and research thereafter.

I also very thankful to all my classmates and seniors of VLSI lab-I especially K Sudeendra, Sushant, Dr. Jitendra K Das, and Ayas Kanta my friends Somyakant, Deepak Krishnankutty, Prashanth Reddy, Arun Kumar and George Tom V who always encouraged me in the successful completion of my thesis work.

DINESH KUMAR M

ROLL No: 207EC207

Abstract:

PCI Express is third generation Computer Bus to inter connect peripherals in a Computer, Servers, Mobile sets and systems. PCS is the sublayer of the physical layer of PCI Express 1.0. The major constituents of this layer are transmitter and receiver.

Transmitter comprises of 8b/10b encoder. The Primary purpose of this scheme is to embed a clock into the serial bit stream of transmitter lanes. No clock is transmitted along with the serial data bit stream. This eliminates EMI noise and provides DC balance.

Receiver comprises of special symbol detector, elastic buffer and 8b/10b decoder. Elastic buffer is nothing but a FIFO operated with two clocks. While a transaction, at one device Recovered Clock from the received data and the clock transmitted at another device may slightly differ. So, Recovered clock and the receiver clock will differ. In this case data corruption will occur. To avoid this situation elastic buffers are used and the data recovered through special symbols. When ever recovered clock is faster than system clock, there is overflow in the buffer. And when recovered clock is slower than system clock underflow in the elastic buffer will occur.

8b/10b decoder gives 8bit character and data/control signals. Disparity error and Decode error can be known though this module. If any error is present in the received data then loopback signal is generated.

This work uses VHDL to model different blocks of the PCS of physical layer of PCI Express. The RTL code is simulated, synthesized and implemented using the ISE 10.1 from Xilinx and the Spartan 3E FPGA was targeted for implementation.

Contents:

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Introduction:

Buses are designed to connect more than two devices together in a system. Devices may include the CPU, main memory, and I/O devices. They consist of three types of signals: Data, Address, and Control.

In addition to these signals, there may be utility lines defined as part of the bus to supply power and ground to the devices connected to it.

Different types of buses that are common in practice are SPI (Serial Peripheral Interface) , I2C (Inter Integrated Circuit), ISA (Industry Standard Architecture), VESA (Video Electronics Standards Association, VL Bus), PCI (Peripheral Component Interconnect), USB (Universal Serial Bus), AGP (Advanced Graphics Port), PCI Express.

ISA :

ISA is the first system bus was developed by a team lead by Mark Dean at IBM as part of the IBM PC project in 1981. It originated as an 8-bit system ran at 4.77 MHz and was extended in 1983 for the XT system architecture. The newer 16-bit standard operated at 6 or 8 MHz,

The IBM AT bus was introduced in 1984. EISA (Extended ISA) is of bit operated at 8.33MHz introduced in 1988.

By the time there was a strong market need for a bus of these speeds and capabilities, the VESA Local Bus and later PCI filled this niche and EISA vanished into obscurity.

VESA :

VESA is an international standards body for computer graphics founded in the late 1980s by NEC Home Electronics and eight other video display adapter manufacturers. Pronounced VESA - As in Mesa, AZ not as VISA. This is a common misconception amongst many.

VESA's initial goal was to produce a standard for 800x600 SVGA resolution video displays. Since then VESA has issued a number of standards, mostly relating to the function of video peripherals in IBM PC compatible computers.

PCI :

The PCI bus clock is 33 MHz the address bus width is 32-bits (4GB memory address space), although PCI optionally supports 64-bit address bus. The data bus width is implemented as either 32-bits or 64-bits depending on bus performance requirement. The address and data bus signals are multiplexed on the same pins (AD bus) to reduce pin count. PCI is introduced by INTEL, PCI–SIG in 1993.

Different Computer buses:

Table 1-1: Different Buses and their speeds

Device	Speed (bit/s)	Speed (byte/s)
I2C	3.4 Mbit/s	**425 kB/s**
ISA 8-Bit/4.77 MHz[	9.6 Mbit/s	**1.2 MB/s**
ISA 16-Bit/8.33 MHz	42.4 Mbit/s	**5.3 MB/s**
EISA 8-16-32bits/8.33 MHz	320 Mbit/s	**32 MB/s**
PCI 32-bit/33 MHz	1,067 Mbit/s	**133.33 MB/s**
PCI Express 1.0 (x1 link)	2,000 Mbit/s	**250 MB/s**
PCI 64-bit/33 MHz	2,133 Mbit/s	**266.7 MB/s**
PCI 32-bit/66 MHz	2,133 Mbit/s	**266.7 MB/s**
AGP 1x	2,133 Mbit/s	**266.7 MB/s**
PCI Express 1.0 (x2 link)	4,000 Mbit/s	**500 MB/s**
AGP 2x	4,266 Mbit/s	**533.3 MB/s**
PCI 64-bit/66 MHz	4,266 Mbit/s	**533.3 MB/s**
PCI-X DDR 16-bit	4,266 Mbit/s	**533.3 MB/s**
PCI 64-bit/100 MHz	6,399 Mbit/s	**800 MB/s**

PCI Express 1.0 (x4 link)	8,000 Mbit/s	**1,000 MB/s**
AGP 4x	8,533 Mbit/s	**1,067 MB/s**
PCI-X 133	8,533 Mbit/s	**1,067 MB/s**
PCI-X QDR 16-bit	8,533 Mbit/s	**1,067 MB/s**
PCI Express 1.0 (x8 link)	16,000 Mbit/s	**2,000 MB/s**
AGP 8x	17,066 Mbit/s	**2,133 MB/s**
PCI-X DDR	17,066 Mbit/s	**2,133 MB/s**
PCI Express 1.0 (x16 link)	32,000 Mbit/s	**4,000 MB/s**
PCI Express 2.0 (x8 link)	32,000 Mbit/s	**4,000 MB/s**
AGP 8x 64-bit	34,133 Mbit/s	**4,266 MB/s**
PCI Express (x32 link)	64,000 Mbit/s	**8,000 MB/s**
PCI Express 2.0 (x16 link)	64,000 Mbit/s	**8,000 MB/s**
PCI Express 2.0 (x32 link)	128,000 Mbit/s	**16,000 MB/s**

1.2. What is PCI Express:

PCI **Express** (**Peripheral Component Interconnect Express**), officially abbreviated as **PCIe**, is a computer expansion card standard designed to replace the older PCI, PCI-X, and AGP standards. Introduced by Intel in 2004, PCIe (or **PCI-E**, as it is commonly called) is the latest standard for expansion cards that is available on mainstream personal computers.

PCI Express is used in consumer, server, and industrial applications, both as a motherboard-level interconnect (to link motherboard-mounted peripherals) and as an expansion card interface for add-in boards. A key difference between PCIe and earlier PC buses is a topology based on point-to-point serial links, rather than a shared parallel bus architecture.

The PCI Express architects have carried forward the most beneficial features from previous generation bus architectures and have also taken advantages of new developments in computer architecture.

For example, PCI Express employs the same usage model and load-store communication model as PCI and PCI-X. PCI Express supports familiar transactions such as memory read/write, IO read/write and configuration read/write transactions. The memory, IO and configuration address space model is the same as PCI and PCI-X address spaces. By maintaining the address space model, existing OSs and driver software will run in a PCI Express system without any modifications. In other words, PCI Express is software backwards compatible with PCI and PCI-X systems. In fact, a PCI Express system will boot an existing OS with no changes to current drivers and application programs. Even PCI/ACPI power management software will still run.

Like predecessor buses, PCI Express supports chip-to-chip interconnect and board-to-board interconnect via cards and connectors. The connector and card structure are similar to PCI and PCI-X connectors and cards. A PCI Express motherboard will have a similar form factor to existing FR4 ATX motherboards which is encased in the familiar PC package.

1.3. Why PCI Express:

The predecessor Bus PCI has some limitations and to overcome those limitations PCI Express has been introduced.

Limitations of PCI: The maximum frequency achievable with the PCI architecture is 66 MHz this is a result of the static clock method of driving and latching signals and because reflected-wave signaling is used. PCI bus efficiency is in the order of 50% or 60%. Some of the factors that contribute towards this reduced efficiency are listed below.

The PCI specification allows master and target devices to insert wait-states during data phases of a bus cycle. Slow devices will add wait-states which reduces the efficiency of bus cycles. PCI bus cycles do not indicate transfer size. This makes buffer management within master and target devices inefficient. Delayed transactions on PCI are handled inefficiently. When a master is retried, it guesses when to try again. If the master tries too soon, the target may retry the transaction again. If the master waits too long to retry, the latency to complete a data transfer is increased. Similarly, if a target disconnects a transaction the master must guess when to resume the bus cycle at a later time.

All PCI bus master accesses to system memory result in a snoop access to the CPU cache. Doing so results in additional wait states during PCI bus master accesses of system memory. The North Bridge or MCH must assume all system memory address space is cachable even though this may not be the case. PCI bus cycles provide no mechanism by which to indicate an access to non- cachable memory address space.

PCI architecture observes strict ordering rules as defined by the specification. Even if a PCI application does not require observation of these strict ordering rules, PCI bus cycles do not provide a mechanism to allow relaxed ordering rule. Observing relaxed ordering rules allows bus cycles (especially those that cross a bridge) to complete with reduced latency.

PCI interrupt handling architecture is inefficient especially because multiple devices share a PCI interrupt signal. Additional software latency is incurred while software discovers which device or devices that share an interrupt signal actually generated the interrupt. The processor's NMI interrupt input is asserted when a PCI parity or system error is detected. Ultimately the system shuts down when an error is detected. This is a severe response. A more appropriate response might be to detect the error and attempt error recovery.

PCI does not require error recovery features, nor does it support an extensive register set for documenting a variety of detectable errors. These limitations above have been resolved in the next generation bus architectures, PCI Express.

Improvements in PCI Express:

To improve bus performance, reduce overall system cost and take advantage of new developments in computer design, the PCI Express architecture had to be significantly re-designed from its predecessor buses. PCI and PCI-X buses are multi-drop parallel interconnect buses in which many devices share one bus.

PCI Express on the other hand implements a serial, point-to-point type interconnect for communication between two devices. Multiple PCI Express devices are interconnected via the use of switches which means one can practically connect a large number of devices together in a system. A point-to-point interconnect implies limited electrical load on the link allowing transmission and reception frequencies to scale too much higher numbers. Currently PCI Express transmission and reception data rate is 2.5 Gbits/sec. A serial interconnect between two devices results in fewer pins per device package which reduces PCI Express chip and board design cost and reduces board design complexity. PCI Express performance is also highly scalable. This is achieved by implementing scalable numbers for pins and signal Lanes per interconnect based on communication performance requirements for that interconnect.

PCI Express implements switch-based technology to interconnect a large number of devices. Communication over the serial interconnect is accomplished using a packet-based communication protocol. Quality Of Service (QoS) features provide differentiated transmission performance for different applications. Hot Plug/Hot Swap support enables "always-on" systems. Advanced power management features allow one to design for low power mobile applications. RAS (Reliable, Available, and Serviceable) error handling features make PCI Express suitable for robust high-end server applications. Hot plug, power management, error handling and interrupt signaling are accomplished in-band using packet based messaging rather than side-band signals. This keeps the device pin count low and reduces system cost.

The configuration address space available per function is extended to 4KB, allowing designers to define additional registers. However, new software is required to access this extended configuration register space.

Serial Bus Advantages:

The bonded serial format was chosen over a traditional parallel format due to the phenomenon of timing skew. Timing skew is a direct result of the limitations imposed by the speed of an electrical signal traveling down a wire, which it does at a finite speed. Because different traces in an interface have different lengths, parallel signals transmitted simultaneously from a source arrive at their destinations at different times. When the interconnection clock rate rises to the point where the wavelength of a single bit is less than this difference in path length, the bits of a single word do not arrive at their destination simultaneously, making parallel recovery of the word difficult. Thus, the speed of the electrical signal, combined with the difference in length between the longest and shortest trace in a parallel interconnect, leads to a naturally imposed maximum bandwidth. Serial channel bonding avoids this issue by not requiring the bits to arrive simultaneously. PCIe is just one example of a general trend away from parallel buses to serial interconnects. Other examples include Serial ATA, USB, SAS, FireWire and RapidIO. The multichannel serial design also increases flexibility by allowing slow devices to be allocated fewer lanes than fast devices.

1.4. Genesis of PCI Express:

While in development, PCIe was initially referred to as *HSI* (for ***H****igh* ***S****peed* ***I****nterconnect)*, and underwent a name change to *3GIO* (for ***3****rd* ***G****eneration* ***I/O***) before finally settling on its PCI-SIG name *PCI Express*. It was first drawn up by a technical working group named the *Arapaho Work Group* (AWG) which for initial drafts consisted of an Intel only team of architects. Subsequently the AWG was expanded to include industry partners.

PCIe is a technology under constant development and improvement. In 2004, Intel introduced PCIe 1.0, with a data rate of 250 MB/s and a transfer rate of 2.5 GT/s. Later, version 1.1 offered minor revisions to the specification. On 15 January 2007, the PCI-SIG announced the availability of the PCI Express Base 2.0 specification. This doubled the data rate of each lane from 250 to 500, and the transfer rate from 2.5 GT/s to 5 GT/s. PCIe 2.0 is backward compatible with PCIe 1.1 as a physical interface slot and from within software, so older cards will still be able to work in machines fitted with PCIe 2.0. The proposed PCIe 3.0 is scheduled for release around 2010.

PCI Express 2.0:

PCI-SIG announced the availability of the PCI Express Base 2.0 specification on 15 January 2007. PCIe 2.0 doubles the bus standard's bandwidth from 0.25 GByte/s to 0.5 GByte/s, meaning a x32 connector can transfer data at up to 16 GByte/s for both video cards (SLI 2x etc). PCIe 2.0 has two 32 bits channels for each GPU (2x16), while the first version only has 1x16 and is operating at 2 GHz.

PCIe 2.0 motherboard slots are backward compatible with PCIe v1.x. PCIe 2.0 cards have good backwards compatibility, new PCIe 2.0 graphics cards are compatible with PCIe 1.1 motherboards, meaning that they will run on them using the available bandwidth of PCI Express 1.1. Overall, graphic cards or motherboards designed for v2.0 will be able to work with the other being v1.1 or v1.0.

The PCI-SIG also said that PCIe 2.0 features improvements to the point-to-point data transfer protocol and its software architecture. .

PCI Express 3.0:

In August 2007 PCI-SIG announced that PCI Express 3.0 will carry a bit rate of 8 gigatransfers per second. The spec will be backwards-compatible with existing PCIe implementations and a final spec is due in 2009. New features for PCIe 3.0 specification include a number of optimizations for enhanced signaling and data integrity, including transmitter and receiver equalization, PLL improvements, clock data recovery, and channel enhancements for currently supported topologies.

Following a six-month technical analysis of the feasibility of scaling the PCIe interconnect bandwidth, PCI-SIG's analysis found out that 8 gigatransfers per second can be manufactured in mainstream silicon process technology, and can be deployed with existing low-cost materials and infrastructure, while maintaining full compatibility (with negligible impact) to the PCIe protocol stack.

PCIe 2.0 delivers 5 GT/s but employed an 8b/10b encoding scheme which took 20 percent overhead on the overall raw bit rate. By removing the requirement for the 8b/10b encoding scheme (relying solely on the still-used scrambler), PCIe 3.0's 8 GT/s bit rate effectively delivers double PCIe 2.0 bandwidth. According to an official press release by PCI-SIG on August 8, 2007: "The final PCIe 3.0 specifications, including form factor specification updates, may be available by late 2009, and could be seen in products starting in 2010 and beyond." PCI-SIG expects the PCIe 3.0 specifications to undergo rigorous technical vetting and validation before being released to the industry. This process, which was followed in the development of prior generations of the PCIe Base and various form factor specifications, includes the corroboration of the final electrical parameters with data derived from test silicon and other simulations conducted by multiple members of the PCI-SIG.

1.5. PCI Express Specifications:

As of (May 2009) the following are specifications released by the PCISIG.

- PCI Express 1.0a Base Specification released Q2, 2003
- PCI Express 1.0a Card Electomechanical Specification released Q2, 2002
- PCI Express 1.0 Base Specification released Q2, 2002
- PCI Express 1.0 Card Electomechanical Specification released Q2, 2002
- Mini PCI Express 1.0 Specification released Q2, 2003
- PCI Express 2.0 Base Specification released Q2, 2004
- PCI Express 3.0 Base Specification released Q2, 2007

Chapter 2

PCI Express Features

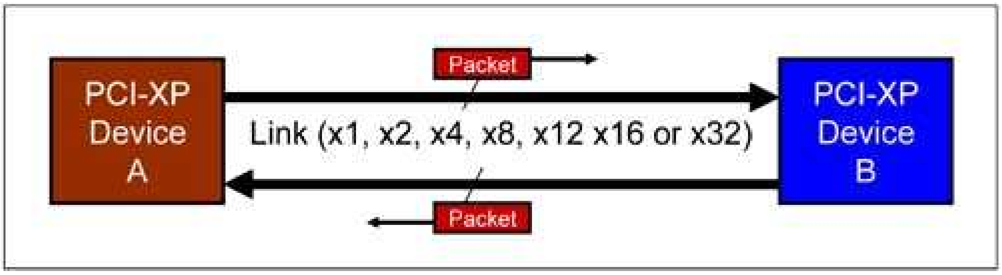
2.1. PCI Express Features:

PCI Express provides a high-speed, high-performance, point-to-point, dual simplex, differential signaling Link for interconnecting devices. Data is transmitted from a device on one set of signals, and received on another set of signals.

The Link - A Point-to-Point Interconnect:

As shown in Figure 2-1, a PCI Express interconnect consists of either a x1, x2, x4, x8, x12, x16 or x32 point-to-point Link. A PCI Express Link is the physical connection between two devices. A Lane consists of signal pairs in each direction. A x1 Link consists of 1 Lane or 1 differential signal pair in each direction for a total of 4 signals. A x32 Link consists of 32 Lanes or 32 signal pairs for each direction for a total of 128 signals. The Link supports a symmetric number of Lanes in each direction. During hardware initialization, the Link is initialized for Link width and frequency of operation automatically by the devices on opposite ends of the Link. No OS or firmware is involved during Link level initialization.

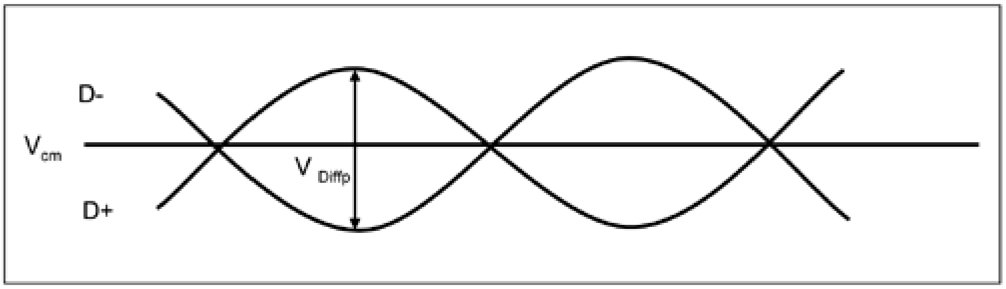
Figure 2-1: PCI Express Link



Differential Signaling:

PCI Express devices employ differential drivers and receivers at each port. Figure 2-2 shows the electrical characteristics of a PCI Express signal. A positive voltage difference between the D+ and D- terminals implies Logical 1. A negative voltage difference between D+ and D- implies a Logical 0. No voltage difference between D+ and D- means that the driver is in the high-impedance tristate condition, which is referred to as the electrical-idle and low-power state of the Link.

Figure 2-2: PCI Express Differential Signal



Switches Used to Interconnect Multiple Devices:

Switches are implemented in systems requiring multiple devices to be interconnected. Switches can range from a 2-port device to an n-port device, where each port connects to a PCI Express Link. The specification does not indicate a maximum number of ports a switch can implement. A switch may be incorporated into a Root Complex device (Host bridge or North bridge equivalent), resulting in a multi-port root complex.

Packet Based Protocol:

Rather than bus cycles we are familiar with from PCI and PCI-X architectures, PCI Express encodes transactions using a packet based protocol. Packets are transmitted and received serially and byte striped across the available Lanes of the Link. The more Lanes implemented on a Link the faster a packet is transmitted and the greater the bandwidth of the Link. The packets are used to support the split transaction protocol for non-posted transactions. Various types of packets such as memory read and write requests, IO read and write requests, configuration read and write requests, message requests and completions are defined.

Bandwidth and Clocking:

The aggregate bandwidth achievable with PCI Express is significantly higher than any bus available today. The PCI Express 1.0 specification supports 2.5 Gbits/sec/lane/direction transfer rate.

No clock signal exists on the Link. Each packet to be transmitted over the Link consists of bytes of information. Each byte is encoded into a 10-bit symbol. All symbols are guaranteed to have one-zero transitions. The receiver uses a PLL to recover a clock from the 0-to-1 and 1-to-0 transitions of the incoming bit stream.

Address Space:

PCI Express supports the same address spaces as PCI: memory, IO and configuration address spaces. In addition, the maximum configuration address space per device function is extended from 256 Bytes to 4 KBytes. New OS, drivers and applications are required to take advantage of this additional configuration address space. Also, a new messaging transaction and address space provides messaging capability between devices. Some messages are PCI Express standard messages used for error reporting, interrupt and power management messaging. Other messages are vendor defined messages.

PCI Express Transactions:

PCI Express supports the same transaction types supported by PCI and PCI-X. These include memory read and memory write, I/O read and I/O write, configuration read and configuration write. In addition, PCI Express supports a new transaction type called Message transactions. These transactions are encoded using the packet-based PCI Express protocol described later.

PCI Express Transaction Model:

PCI Express transactions can be divided into two categories. Those transactions that are non-posted and those that are posted. Non-posted transactions, such as memory reads, implement a split transaction communication model similar to the PCI-X split transaction protocol. For example, a requester device transmits a non-posted type memory read request packet to a completer. The completer returns a completion packet with the read data to the requester. Posted transactions, such as memory writes, consist of a memory write packet transmitted uni-directionally from requester to completer with no completion packet returned from completer to requester.

Error Handling and Robustness of Data Transfer:

CRC fields are embedded within each packet transmitted. One of the CRC fields supports a Link-level error checking protocol whereby each receiver of a packet checks for Link-level CRC errors. Packets transmitted over the Link in error are recognized with a CRC error at the receiver. The transmitter of the packet is notified of the error by the receiver. The transmitter automatically retries sending the packet (with no software involvement), hopefully resulting in auto-correction of the error.

In addition, an optional CRC field within a packet allows for end-to-end data integrity checking required for high availability applications.

Quality of Service (QoS), Traffic Classes (TCs) and Virtual Channels (VCs):

The Quality of Service feature of PCI Express refers to the capability of routing packets from different applications through the fabric with differentiated priorities and deterministic latencies and bandwidth. For example, it may be desirable to ensure that Isochronous applications, such as video data packets, move through the fabric with higher priority and guaranteed bandwidth, while control data packets may not have specific bandwidth or latency requirements.

PCI Express packets contain a Traffic Class (TC) number between 0 and 7 that is assigned by the device's application or device driver. Packets with different TCs can move through the fabric with different priority, resulting in varying performances. These packets are routed through the fabric by utilizing virtual channel (VC) buffers implemented in switches, endpoints and root complex devices.

Each Traffic Class is individually mapped to a Virtual Channel (a VC can have several TCs mapped to it, but a TC cannot be mapped to multiple VCs). The TC in each packet is used by the transmitting and receiving ports to determine which VC buffer to drop the packet into. Switches and devices are configured to arbitrate and prioritize between packets from different VCs before forwarding. This arbitration is referred to as VC arbitration. In addition, packets arriving at different ingress ports are forwarded to their own VC buffers at the egress port. These transactions are prioritized based on the ingress port number when being merged into a common VC output buffer for delivery across the egress link. This arbitration is referred to as Port arbitration.

Flow Control:

A packet transmitted by a device is received into a VC buffer in the receiver at the opposite end of the Link. The receiver periodically updates the transmitter with information regarding the amount of buffer space it has available. The transmitter device will only transmit a packet to the receiver if it knows that the receiving device has sufficient buffer space to hold the next transaction. The protocol by which the transmitter ensures that the receiving buffer has sufficient space available is referred to as flow control. The flow control mechanism guarantees that a transmitted packet will be accepted by the receiver, baring error conditions. As such, the PCI Express transaction protocol does not require support of packet retry (unless an error condition is detected in the receiver), thereby improving the efficiency with which packets are forwarded to a receiver via the Link.

Power Management:

The PCI Express fabric consumes less power because the interconnect consists of fewer signals that have smaller signal swings. Each device's power state is individually managed. PCI/PCI Express power management software determines the power management capability of each device and manages it individually in a manner similar to PCI. Devices can notify software of their current power state, as well as power management software can propagate a wake-up event through the fabric to power-up a device or group of devices. Devices can also signal a wake-up event using an in-band mechanism or a side-band signal.

PCI Express also supports the following Link power states: L0, L0s, L1, L2 and L3, where L0 is the full-on Link state and L3 is the Link-Off power state.

Hot Plug Support:

PCI Express supports hot plug and surprise hot unplug without usage of sideband signals. Hot plug interrupt messages, communicated in-band to the root complex, trigger hot plug software to detect a hot plug or removal event. Rather than implementing a centralized hot plug controller as exists in PCI platforms, the hot plug controller function is distributed to the port logic associated with a hot plug capable port of a switch or root complex. 2 colored LEDs, a Manually-operated Retention Latch (MRL), MRL sensor, attention button, power control signal and PRSNT2# signal are some of the elements of a hot plug capable port.

PCI Compatible Software Model:

PCI Express employs the same programming model as PCI and PCI-X systems described earlier in this chapter. The memory and IO address space remains the same as PCI/PCI-X. The first 256 Bytes of configuration space per PCI Express function is the same as PCI/PCI-X device configuration address space, thus ensuring that current OSs and device drivers will run on a PCI Express system. PCI Express architecture extends the configuration address space to 4 KB per functional device. Updated OSs and device drivers are required to take advantage and access this additional configuration address space.

PCI Express configuration model supports two mechanisms:

1. PCI compatible configuration model which is 100% compatible with existing OSs and bus enumeration and configuration software for PCI/PCI-X systems.
2. PCI Express enhanced configuration mechanism which provides access to additional configuration space beyond the first 256 Bytes and up to 4 KBytes per function.

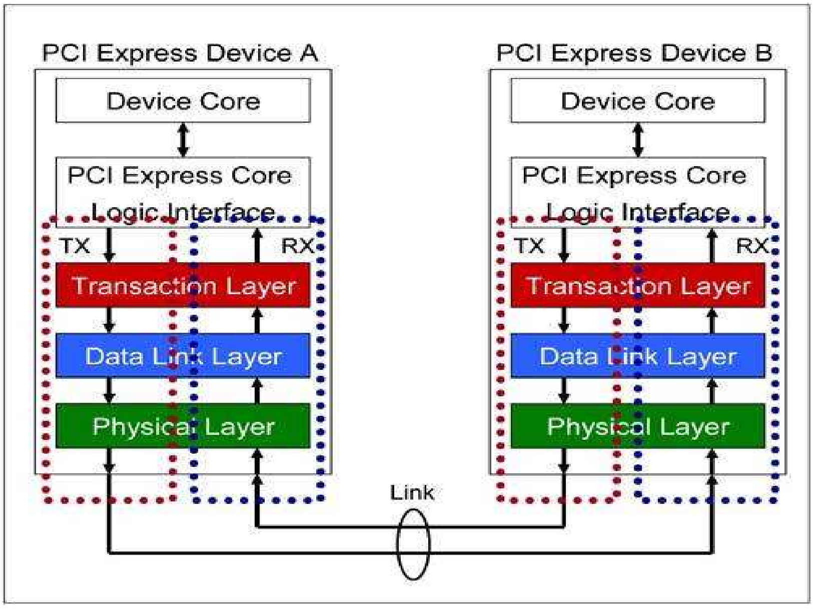
Chapter 3

PCI Express Device Layers

3.1. PCI Express Device Layers:

Overview: The PCI Express specification defines a layered architecture for device design as shown in Figure 3-1. The layers consist of a Transaction Layer, a Data Link Layer and a Physical layer. The layers can be further divided vertically into two, a transmit portion that processes outbound traffic and a receive portion that processes inbound traffic. However, a device design does not have to implement a layered architecture as long as the functionality required by the specification is supported.

Figure 3-1: PCI Express Device Layers



The goal of this section is to describe the function of each layer and to describe the flow of events to accomplish a data transfer. Packet creation at a transmitting device and packet reception and decoding at a receiving device are also explained.

Transmit Portion of Device Layers

Consider the transmit portion of a device. Packet contents are formed in the Transaction Layer with information obtained from the device core and application. The packet is stored in buffers ready for transmission to the lower layers. This packet is referred to as a Transaction Layer

Packet (TLP) described in the earlier section of this chapter. The Data Link Layer concatenates to the packet additional information required for error checking at a receiver device. The packet is then encoded in the Physical layer and transmitted differentially on the Link by the analog portion of this Layer. The packet is transmitted using the available Lanes of the Link to the receiving device which is its neighbor.

Receive Portion of Device Layers

The receiver device decodes the incoming packet contents in the Physical Layer and forwards the resulting contents to the upper layers. The Data Link Layer checks for errors in the incoming packet and if there are no errors forwards the packet up to the Transaction Layer. The Transaction Layer buffers the incoming TLPs and converts the information in the packet to a representation that can be processed by the device core and application.

3.2. Function of Each PCI Express Device Layer:

Figure 3-2 is a more detailed block diagram of a PCI Express Device's layers. This block diagram is used to explain key functions of each layer and explain the function of each layer as it relates to generation of outbound traffic and response to inbound traffic. The layers consist of Device Core/Software Layer, Transaction Layer, Data Link Layer and Physical Layer.

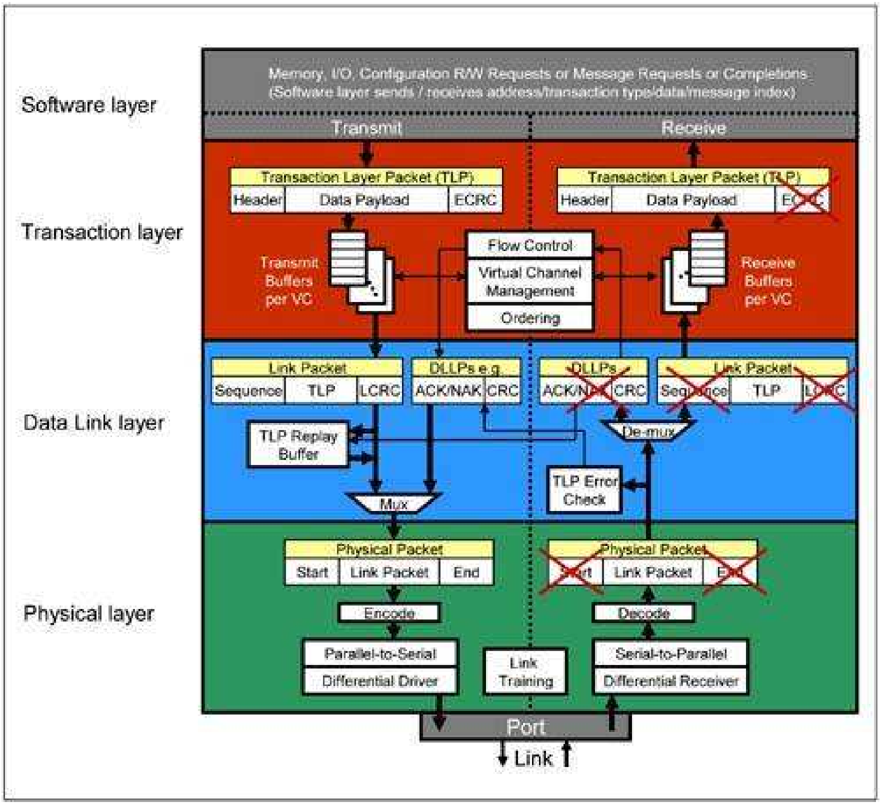
A.Device Core / Software Layer:

The Device Core consists of, for example, the root complex core logic or an endpoint core logic such as that of an Ethernet controller, SCSI controller, USB controller, etc. To design a PCI Express endpoint, a designer may reuse the Device Core logic from a PCI or PCI-X core logic design and wrap around it the PCI Express layered design described in this section.

Transmit Side

The Device Core logic in conjunction with local software provides the necessary information required by the PCI Express device to generate TLPs. This information is sent via the Transmit interface to the Transaction Layer of the device. Example of information transmitted to the

Figure 3-2: Detailed Block Diagram of PCI Express Device's Layers



Transaction Layer includes: transaction type to inform the Transaction Layer what type of TLP to generate, address, amount of data to transfer, data, traffic class, message index etc.

Receive Side

The Device Core logic is also responsible to receive information sent by the Transaction Layer via the Receive interface. This information includes: type of TLP received by the Transaction Layer, address, amount of data received, data, traffic class of received TLP, message index, error conditions etc.

B. Transaction Layer:

The transaction Layer shown in Figure 3-2 is responsible for generation of outbound TLP traffic and reception of inbound TLP traffic. The Transaction Layer supports the split transaction protocol for non-posted transactions. In other words, the Transaction Layer associates an inbound completion TLP of a given tag value with an outbound non-posted request TLP of the same tag value transmitted earlier.

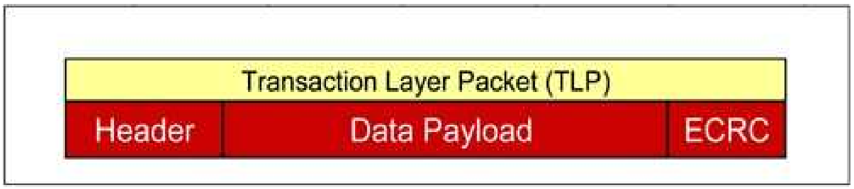
The transaction layer contains virtual channel buffers (VC Buffers) to store outbound TLPs that await transmission and also to store inbound TLPs received from the Link. The flow control protocol associated with these virtual channel buffers ensures that a remote transmitter does not transmit too many TLPs and cause the receiver virtual channel buffers to overflow. The Transaction Layer also orders TLPs according to ordering rules before transmission. It is this layer that supports the Quality of Service (QoS) protocol.

The Transaction Layer supports 4 address spaces: memory address, IO address, configuration address and message space. Message packets contain a message.

Transmit Side

The Transaction Layer receives information from the Device Core and generates outbound request and completion TLPs which it stores in virtual channel buffers. This layer assembles Transaction Layer Packets (TLPs). The major components of a TLP are: Header, Data Payload and an optional ECRC (specification also uses the term Digest) field as shown in Figure 3-3.

Figure 3-3: TLP Structure at the Transaction Layer



The Header is 3 doublewords or 4 doublewords in size and may include information such as;

Address, TLP type, transfer size, requester ID/completer ID, tag, traffic class, byte enables, completion codes, and attributes (including "no snoop" and "relaxed ordering" bits).

The transfer size or length field indicates the amount of data to transfer calculated in doublewords (DWs). The data transfer length can be between 1 to 1024 DWs. Write request TLPs include data payload in the amount indicated by the length field of the header. For a read request TLP, the length field indicates the amount of data requested from a completer. This data is returned in one or more completion packets. Read request TLPs do not include a data payload field. Byte enables specify byte level address resolution.

Request packets contain a requester ID (bus#, device#, function #) of the device transmitting the request. The tag field in the request is memorized by the completer and the same tag is used in the completion.

A bit in the Header (TD = TLP Digest) indicates whether this packet contains an ECRC field also referred to as Digest. This field is 32-bits wide and contains an End-to-End CRC (ECRC). The ECRC field is generated by the Transaction Layer at time of creation of the outbound TLP. It is generated based on the entire TLP from first byte of header to last byte of data payload (with the exception of the EP bit, and bit 0 of the Type field. These two bits are always considered to be a 1 for the ECRC calculation). The TLP never changes as it traverses the fabric (with the exception of perhaps the two bits mentioned in the earlier sentence). The receiver device checks for an ECRC error that may occur as the packet moves through the fabric.

Receiver Side

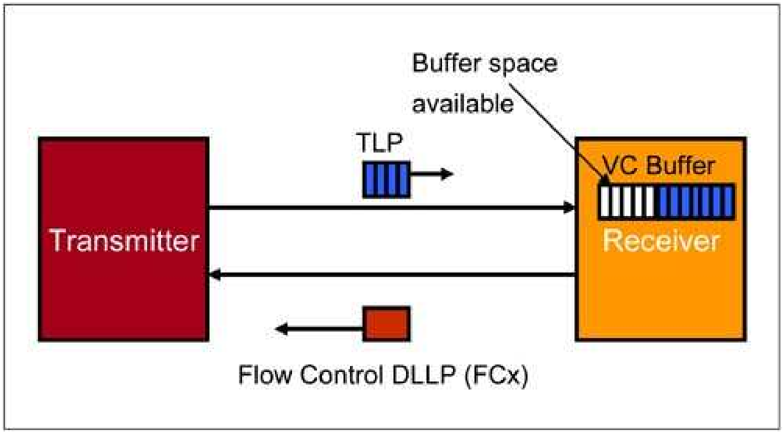
The receiver side of the Transaction Layer stores inbound TLPs in receiver virtual channel buffers. The receiver checks for CRC errors based on the ECRC field in the TLP. If there are no errors, the ECRC field is stripped and the resultant information in the TLP header as well as the data payload is sent to the Device Core.

Flow Control:

The Transaction Layer ensures that it does not transmit a TLP over the Link to a remote receiver device unless the receiver device has virtual channel buffer space to accept TLPs (of a given traffic class). The protocol for guaranteeing this mechanism is referred to as the "flow control" protocol. If the transmitter device does not observe this protocol, a transmitted TLP will cause the receiver virtual channel buffer to overflow. Flow control is automatically managed at the hardware level and is transparent to software. Software is only involved to enable additional buffers beyond the default set of virtual channel buffers (referred to as VC 0 buffers). The default buffers are enabled automatically after Link training, thus allowing TLP traffic to flow through the fabric immediately after Link training. Configuration transactions use the default virtual channel buffers and can begin immediately after the Link training process.

Refer to Figure 3-4 for an overview of the flow control process. A receiver device transmits DLLPs called Flow Control Packets (FCx DLLPs) to the transmitter device on a periodic basis. The FCx DLLPs contain flow control credit information that updates the transmitter regarding how much buffer space is available in the receiver virtual channel buffer. The transmitter keeps track of this information and will only transmit TLPs out of its Transaction Layer if it knows that the remote receiver has buffer space to accept the transmitted TLP.

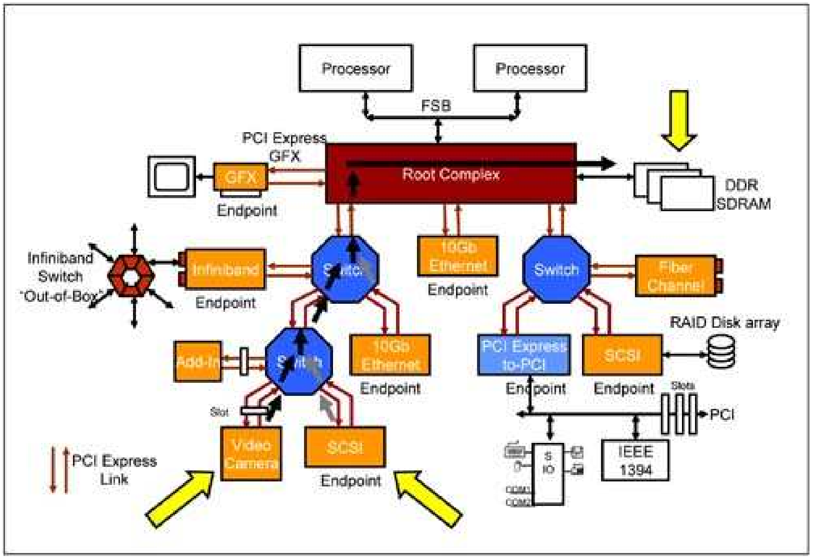
Figure 3-4: Flow Control Process



Quality of Service (QoS):

Consider Figure 3-5 in which the video camera and SCSI device shown need to transmit write request TLPs to system DRAM. The camera data is time critical isochronous data which must reach memory with guaranteed bandwidth otherwise the displayed image will appear choppy or unclear. The SCSI data is not as time sensitive and only needs to get to system memory correctly without errors. It is clear that the video data packet should have higher priority when routed through the PCI Express fabric, especially through switches. QoS refers to the capability of routing packets from different applications through the fabric with differentiated priorities and deterministic latencies and bandwidth. PCI and PCI-X systems do not support QoS capability.

Figure 3-5: Example Showing QoS Capability of PCI Express



Consider this example. Application driver software in conjunction with the OS assigns the video data packets a traffic class of 7 (TC7) and the SCSI data packet a traffic class of 0 (TC0). These TC numbers are embedded in the TLP header. Configuration software uses TC/VC mapping device configuration registers to map TC0 related TLPs to virtual channel 0 buffers (VC0) and TC7 related TLPs to virtual channel 7 buffers (VC7).

As TLPs from these two applications (video and SCSI applications) move through the fabric, the switches post incoming packets moving upstream into their respective VC buffers (VC0 and VC7). The switch uses a priority based arbitration mechanism to determine which of the two incoming packets to forward with greater priority to a common egress port. Assume VC7 buffer contents are configured with higher priority than VC0. Whenever two incoming packets are to be forwarded to one upstream port, the switch will always pick the VC7 packet, the video data, over the VC0 packet, the SCSI data. This guarantees greater bandwidth and reduced latency for video data compared to SCSI data.

Traffic Classes (TCs) and Virtual Channels (VCs):

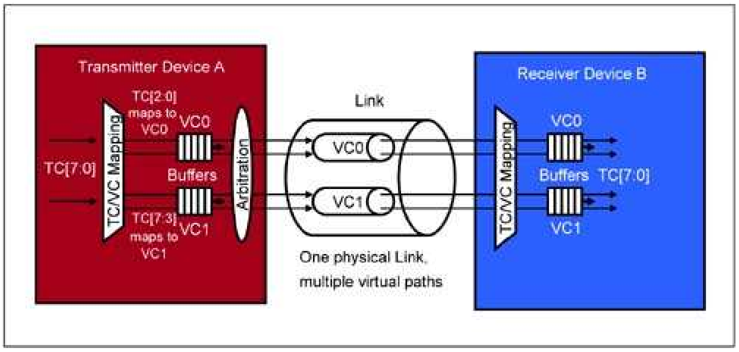
TC is a TLP header field transmitted within the packet unmodified end-to-end through the fabric. Local application software and system software based on performance requirements decides what TC label a TLP uses. VCs are physical buffers that provide a means to support multiple independent logical data flows over the physical Link via the use of transmit and receiver virtual channel buffers.

PCI Express devices may implement up to 8 VC buffers (VC0-VC7). The TC field is a 3-bit field that allows differentiation of traffic into 8 traffic classes (TC0-TC7). Devices must implement VC0. Similarly, a device is required to support TC0 (best effort general purpose service class). The other optional TCs may be used to provide differentiated service through the fabric. Associated with each implemented VC ID, a transmit device implements a transmit buffer and a receive device implements a receive buffer.

Devices or switches implement TC-to-VC mapping logic by which a TLP of a given TC number is forwarded through the Link using a particular VC numbered buffer. PCI Express provides the capability of mapping multiple TCs onto a single VC, thus reducing device cost by means of providing limited number of VC buffer support. TC/VC mapping is configured by system software through configuration registers. It is up to the device application software to determine TC label for TLPs and TC/VC mapping that meets performance requirements. In its simplest form TC/VC mapping registers can be configured with a one-to-one mapping of TC to VC.

Consider the example illustrated in Figure 3-6. The TC/VC mapping registers in Device A are configured to map, TLPs with TC[2:0] to VC0 and TLPs with TC[7:3] to VC1. The TC/VC mapping registers in receiver Device B must also be configured identically as Device A. The same numbered VC buffers are enabled both in transmitter Device A and receiver Device B.

Figure 3-6: TC Numbers and VC Buffers



If Device A needs to transmit a TLP with TC label of 7 and another packet with TC label of 0, the two packets will be placed in VC1 and VC0 buffers, respectively. The arbitration logic arbitrates between the two VC buffers. Assume VC1 buffer is configured with higher priority than VC0 buffer. Thus, Device A will forward the TC7 TLPs in VC1 to the Link ahead of the TC0 TLPs in VC0.

When the TLPs arrive in Device B, the TC/VC mapping logic decodes the TC label in each TLP and places the TLPs in their associated VC buffers.

Port Arbitration and VC Arbitration:

The goals of arbitration support in the Transaction Layer are:

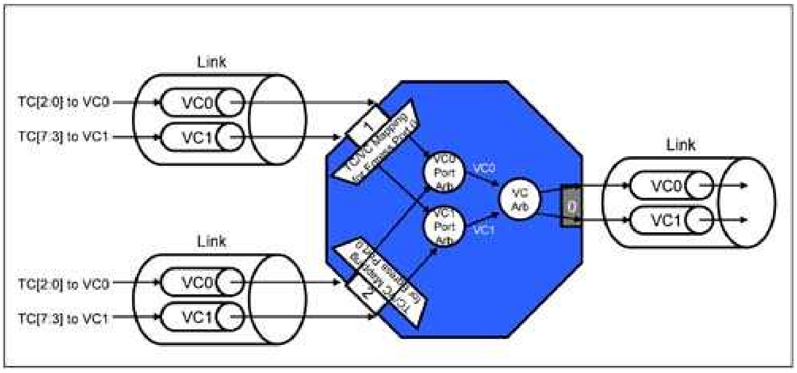
- To provide differentiated services between data flows within the fabric.
- To provide guaranteed bandwidth with deterministic and smallest end-to-end transaction latency.

Packets of different TCs are routed through the fabric of switches with different priority based on arbitration policy implemented in switches. Packets coming in from ingress ports heading towards a particular egress port compete for use of that egress port.

Switches implement two types of arbitration for each egress port: Port Arbitration and VC

Arbitration. Consider Figure 3-7.

Figure 3-7: Switch Implements Port Arbitration and VC Arbitration Logic



Port arbitration is arbitration between two packets arriving on different ingress ports but that map to the same virtual channel (after going through TC-to-VC mapping) of the common egress port. The port arbiter implements round-robin, weighted round-robin or programmable time-based round-robin arbitration schemes selectable through configuration registers.

VC arbitration takes place after port arbitration. For a given egress port, packets from all VCs compete to transmit on the same egress port. VC arbitration resolves the order in which TLPs in different VC buffers are forwarded on to the Link. VC arbitration policies supported include, strict priority, round-robin and weighted round-robin arbitration schemes selectable through configuration registers.

Independent of arbitration, each VC must observe transaction ordering and flow control rules before it can make pending TLP traffic visible to the arbitration mechanism.

Endpoint devices and a root complex with only one port do not support port arbitration. They only support VC arbitration in the Transaction Layer.

Transaction Ordering:

PCI Express protocol implements PCI/PCI-X compliant producer-consumer ordering model for transaction ordering with provision to support relaxed ordering similar to PCI-X architecture.

Transaction ordering rules guarantee that TLP traffic associated with a given traffic class is routed through the fabric in the correct order to prevent potential deadlock or live-lock conditions from occurring. Traffic associated with different TC labels has no ordering relationship.

The Transaction Layer ensures that TLPs for a given TC are ordered correctly with respect to other TLPs of the same TC label before forwarding to the Data Link Layer and Physical Layer for transmission.

Power Management:

The Transaction Layer supports ACPI/PCI power management, as dictated by system software. Hardware within the Transaction Layer autonomously power manages a device to minimize power during full-on power states. This automatic power management is referred to as Active State Power Management and does not involve software. Power management software associated with the OS power manages a device's power states though power management configuration registers.

Configuration Registers:

A device's configuration registers are associated with the Transaction Layer. The registers are configured during initialization and bus enumeration. They are also configured by device drivers and accessed by runtime software/OS. Additionally, the registers store negotiated Link capabilities, such as Link width and frequency.

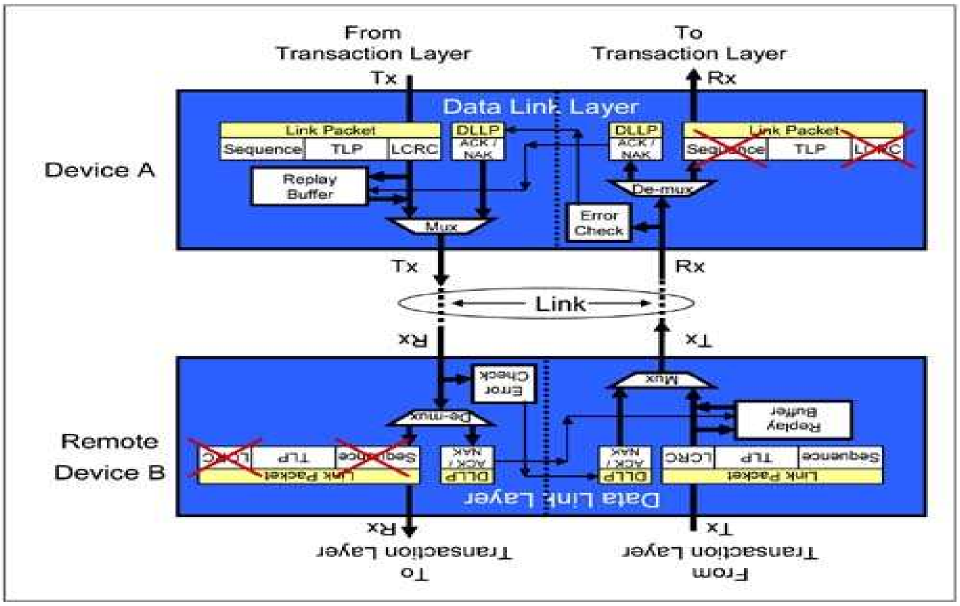
C.Data Link Layer:

Refer to Figure 3-2 for a block diagram of a device's Data Link Layer. The primary function of the Data Link Layer is to ensure data integrity during packet transmission and reception on each Link. If a transmitter device sends a TLP to a remote receiver device at the other end of a Link and a CRC error is detected, the transmitter device is notified with a NAK DLLP. The transmitter device automatically replays the TLP. This time hopefully no error occurs. With error checking and automatic replay of packets received in error, PCI Express ensures very high probability that a TLP transmitted by one device will make its way to the final destination with no errors. This makes PCI Express ideal for low error rate, high-availability systems such as servers.

Transmit Side

The Transaction Layer must observe the flow control mechanism before forwarding outbound TLPs to the Data Link Layer. If sufficient credits exist, a TLP stored within the virtual channel buffer is passed from the Transaction Layer to the Data Link Layer for transmission.

Figure 3-8: Data Link Layer Replay Mechanism



Consider Figure 3-8 which shows the logic associated with the ACK-NAK mechanism of the Data Link Layer. The Data Link Layer is responsible for TLP CRC generation and TLP error checking. For outbound TLPs from transmit Device A, a Link CRC (LCRC) is generated and appended to the TLP. In addition, a sequence ID is appended to the TLP. Device A's Data Link Layer preserves a copy of the TLP in a replay buffer and transmits the TLP to Device B. The Data Link Layer of the remote Device B receives the TLP and checks for CRC errors.

If there is no error, the Data Link Layer of Device B returns an ACK DLLP with a sequence

ID to Device A. Device A has confirmation that the TLP has reached Device B (not necessarily the final destination) successfully. Device A clears its replay buffer of the TLP associated with that sequence ID.

If on the other hand a CRC error is detected in the TLP received at the remote Device B, then a NAK DLLP with a sequence ID is returned to Device A. An error has occurred during TLP transmission. Device A's Data Link Layer replays associated TLPs from the replay buffer. For a given TLP in the replay buffer, if the transmitter device receives a NAK 4 times and the TLP is replayed 3 additional times as a result, then the Data Link Layer logs the error, reports a correctable error, and re-trains the Link.

Receive Side

The receive side of the Data Link Layer is responsible for LCRC error checking on inbound TLPs. If no error is detected, the device schedules an ACK DLLP for transmission back to the remote transmitter device. The receiver strips the TLP of the LCRC field and sequence ID.

If a CRC error is detected, it schedules a NAK to return back to the remote transmitter. The TLP is eliminated.

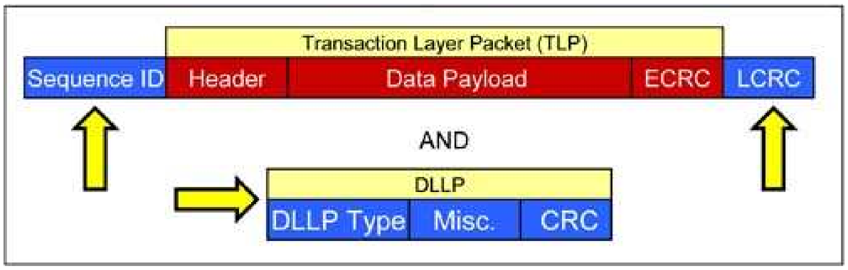
The receive side of the Data Link Layer also receives ACKs and NAKs from a remote device. If an ACK is received the receive side of the Data Link layer informs the transmit side to clear an associated TLP from the replay buffer. If a NAK is received, the receive side causes the replay buffer of the transmit side to replay associated TLPs.

The receive side is also responsible for checking the sequence ID of received TLPs to check for dropped or out-of-order TLPs.

Data Link Layer Contribution to TLPs and DLLPs:

The Data Link Layer concatenates a 12-bit sequence ID and 32-bit LCRC field to an outbound TLP that arrives from the Transaction Layer. The resultant TLP is shown in Figure 3-9 on page 90. The sequence ID is used to associate a copy of the outbound TLP stored in the replay buffer with a received ACK/NAK DLLP inbound from a neighboring remote device. The ACK/NAK DLLP confirms arrival of the outbound TLP in the remote device.

Figure 3-9: TLP and DLLP Structure at the Data Link Layer



The 32-bit LCRC is calculated based on all bytes in the TLP including the sequence ID.

Other Functions of the Data Link Layer:

Following power-up or Reset, the flow control mechanism described earlier is initialized by the Data Link Layer. This process is accomplished automatically at the hardware level and has no software involvement.

Flow control for the default virtual channel VC0 is initialized first. In addition, when additional VCs are enabled by software, the flow control initialization process is repeated for each newly enabled VC. Since VC0 is enabled before all other VCs, no TLP traffic will be active prior to initialization of VC0.

D.Physical Layer:

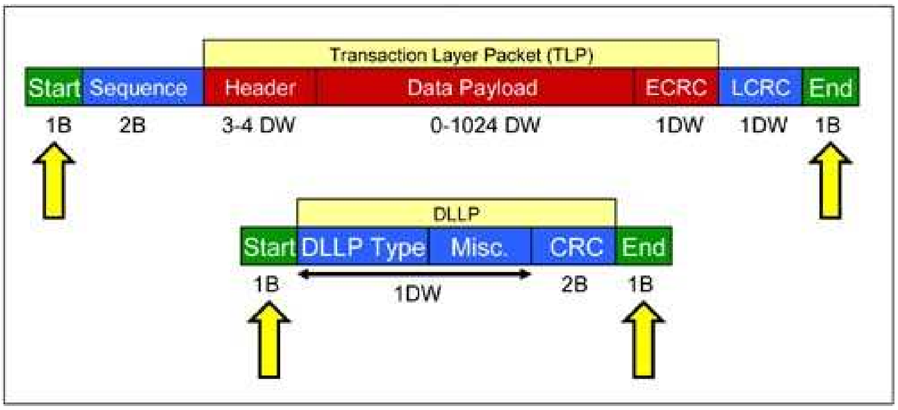
Figure 3-2 for a block diagram of a device's Physical Layer. Both TLP and DLLP type packets are sent from the Data Link Layer to the Physical Layer for transmission over the Link. Also, packets are received by the Physical Layer from the Link and sent to the Data Link Layer.

The Physical Layer is divided in two portions, the Logical Physical Layer and the Electrical Physical Layer. The Logical Physical Layer contains digital logic associated with processing packets before transmission on the Link, or processing packets inbound from the Link before sending to the Data Link Layer. The Electrical Physical Layer is the analog interface of the Physical Layer that connects to the Link.

Transmit Side

TLPs and DLLPs from the Data Link Layer are clocked into a buffer in the Logical Physical Layer. The Physical Layer frames the TLP or DLLP with a Start and End character. The symbol is a framing code byte which a receiver device uses to detect the start and end of a packet. The Start and End characters are shown appended to a TLP and DLLP in Figure 3-10. The diagram shows the size of each field in a TLP or DLLP.

Figure 3-10: TLP and DLLP Structure at the Physical Layer



The transmit logical sub-block conditions the received packet from the Data Link Layer into the correct format for transmission. Packets are byte striped across the available Lanes on the Link.

Each byte of a packet is then scrambled with the aid of Linear Feedback Shift Register type scrambler. By scrambling the bytes, repeated bit patterns on the Link are eliminated, thus reducing the average EMI noise generated.

The resultant bytes are encoded into a 10b code by the 8b/10b encoding logic. The primary purpose of encoding 8b characters to 10b symbols is to create sufficient 1-to-0 and 0-to-1 transition density in the bit stream to facilitate recreation of a receive clock with the aid of a PLL at the remote receiver device. Note that data is not transmitted along with a clock. Instead, the bit stream contains sufficient transitions to allow the receiver device to recreate a receive clock.

The parallel-to-serial converter generates a serial bit stream of the packet on each Lane and transmits it differentially at 2.5 Gbits/s.

Receive Side

The receive Electrical Physical Layer clocks in a packet arriving differentially on all Lanes. The serial bit stream of the packet is converted into a 10b parallel stream using the serial-to-parallel converter. The receiver logic also includes an elastic buffer which accommodates for clock frequency variation between a transmit clock with which the packet bit stream is clocked into a receiver and the receiver clock. The 10b symbol stream is decoded back to the 8b representation of each symbol with the 8b/10b decoder. The 8b characters are de-scrambled. The Byte unstriping logic, re-creates the original packet stream transmitted by the remote device.

Link Training and Initialization:

An additional function of the Physical Layer is Link initialization and training. Link initialization and training is a Physical Layer controlled process that configures and initializes each Link for normal operation. This process is automatic and does not involve software. The following are determined during the Link initialization and training process:

- Link width
- Link data rate
- Lane reversal
- Polarity inversion.
- Bit lock per Lane
- Symbol lock per Lane
- Lane-to-Lane de-skew within a multi-Lane Link.

Link width. Two devices with a different number of Lanes per Link may be connected. E.g. one device has x2 port and it is connected to a device with x4 port. After initialization the Physical Layer of both devices determines and sets the Link width to the minimum Lane width of x2.

Lane reversal if necessary is an optional feature. Lanes are numbered. A designer may not wire the correct Lanes of two ports correctly. In which case training allows for the Lane numbers to be reversed so that the Lane numbers of adjacent ports on each end of the Link match up. Part of the same process may allow for a multi-Lane Link to be split into multiple Links.

Polarity inversion. The D+ and D- differential pair terminals for two devices may not be connected correctly. In which case the training sequence receiver reverses the polarity on the differential receiver.

Link data rate. Training is completed at data rate of 2.5 Gbit/s. In the future, higher data rates of 5 Gbit/s and 10 Gbit/s will be supported. During training, each node advertises its highest data rate capability. The Link is initialized with the highest common frequency that devices at opposite ends of a Link support.

Lane-to-Lane De-skew. Due to Link wire length variations and different driver/receiver characteristics on a multi-Lane Link, bit streams on each Lane will arrive at a receiver skewed with respect to other Lanes. The receiver circuit must compensate for this skew by adding/removing delays on each Lane. Relaxed routing rules allow Link wire lengths in the order of 20"-30".

Reset

Two types of reset are supported:

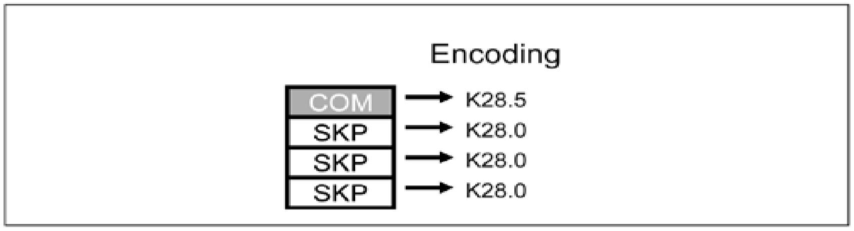
- Cold/warm reset also called a Fundamental Reset which occurs following a device being powered-on (cold reset) or due to a reset without circulating power (warm reset).
- Hot reset sometimes referred to as protocol reset is an in-band method of propagating reset. Transmission of an ordered-set is used to signal a hot reset. Software initiates hot reset generation.

On exit from reset (cold, warm, or hot), all state machines and configuration registers (hot reset does not reset sticky configuration registers) are initialized.

Inserting Clock Compensation Zones:

When the receiver logic receives a symbol stream, it sometimes needs to add or remove a symbol from the received symbol stream to compensate for transmitter verses receiver clock frequency variations. It should be obvious that the receiver logic can't arbitrarily pick a symbol to add or delete. This means that, on a periodic basis, the transmit logic must transmit a special Control character sequence that can be used for this purpose. This sequence is referred to as the SKIP Ordered-Set (see Figure 3-11) which consists of a COM character followed by three SKP characters.

Figure 3-11: SKIP Ordered-Set



SKIP Ordered-Set Insertion Rules:

A transmitter is required to transmit SKIP Ordered-Sets on a periodic basis. The following rules apply:

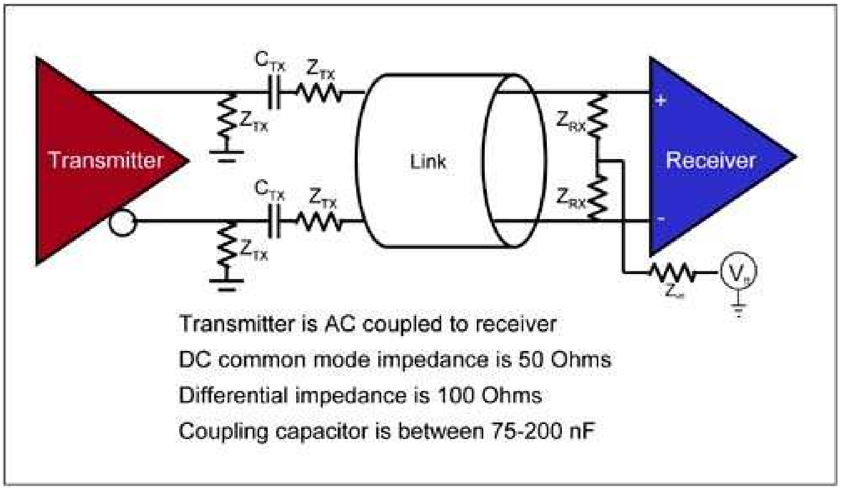
- The set must be scheduled for insertion at most once every 1180 symbol clocks (i.e., symbol times) and at least once every 1538 symbol clocks.
- When it's time to insert a SKIP Ordered-Set, it is inserted at the next packet boundary (not in the middle of a packet). SKIP Ordered-Sets are inserted between packets simultaneously on all Lanes. If a long packet transmission is already in progress, the SKIP Ordered-Sets are accumulated and then inserted consecutively at the next packet boundary.

SKIP Ordered-Sets must not be transmitted while the Compliance Pattern is in progress.

Electrical Physical Layer:

The transmitter of one device is AC coupled to the receiver of another device at the opposite end of the Link as shown in Figure 3-12. The AC coupling capacitor is between 75-200 nF. The transmitter DC common mode voltage is established during Link training and initialization. The DC common mode impedance is typically 50 ohms while the differential impedance is 100 ohms typical. This impedance is matched with a standard FR4 board.

Figure 3-12: Electrical Physical Layer Showing Differential Transmitter and Receiver



Transmitter is AC coupled to receiver

DC common mode impedance is 50 Ohms

Differential impedance is 100 Ohms

Coupling capacitor is between 75-200 nF

Chapter 4

Physical Coding Sublayer

4.1. Overview of Physical Layer:

Physical layer of PCI Express consists of three sublayer as shown in the Figure 4-1.

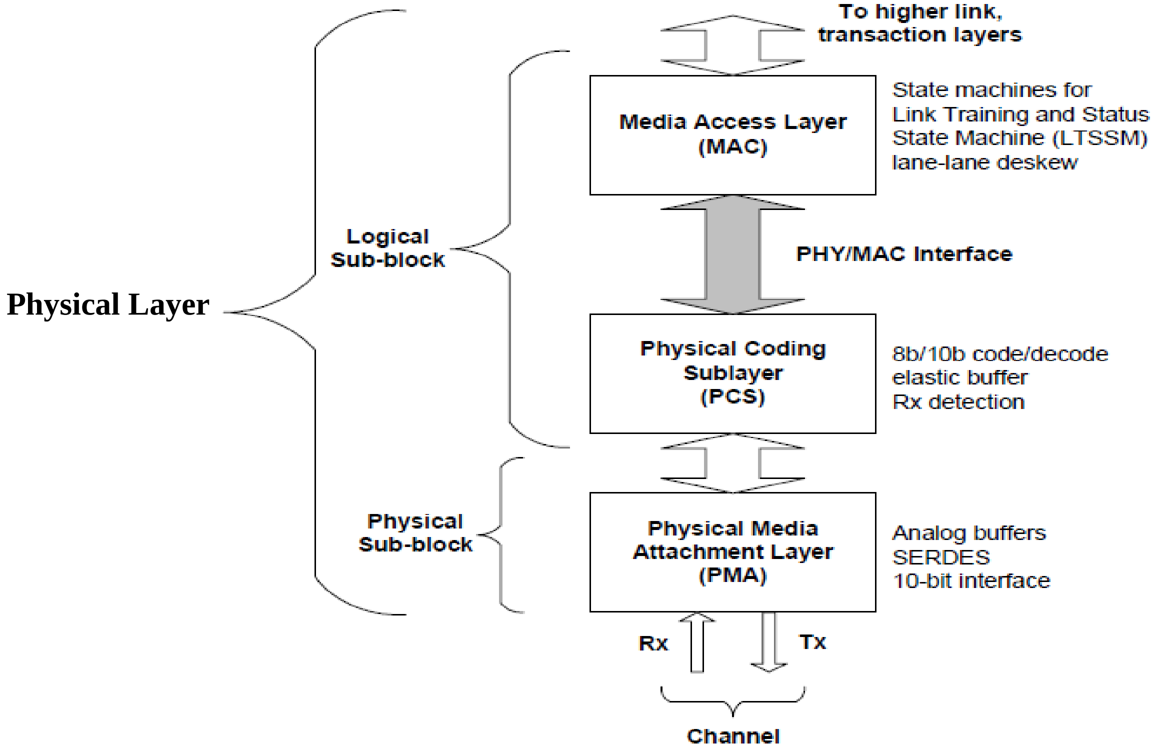
1. MAC (Media Access Control)
2. PCS (Physical Coding Sublayer)
3. PMA (Physical Media Attachment Sublayer)

MAC comprises of State Machines for Link Training and Status State Machine (LTSSM) and lane–lane deskew.

PCS comprises of 8b/10b encoder/decoder, Rx detection, and elastic buffer.

PMA comprises of Analog buffers, SEDRES, 10bit interface.

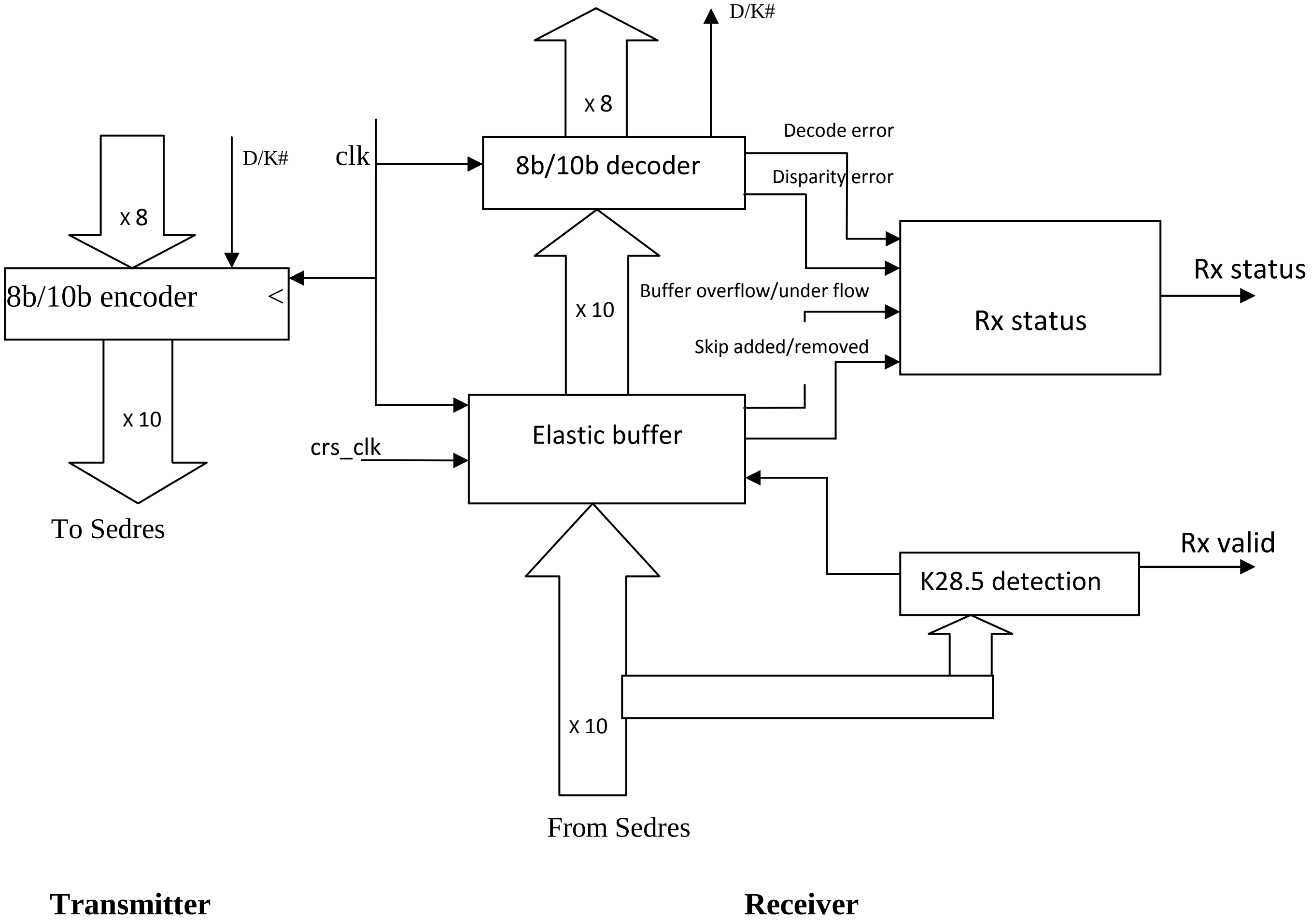
Figure 4-1: Block Diagram of Physical Layer of PCI Express


To higher link, transaction layers
Media Access Layer (MAC)
State machines for Link Training and Status State Machine (LTSSM) lane-lane deskew
Logical Sub-block
PHY/MAC Interface
Physical Layer
Physical Coding Sublayer (PCS)
8b/10b code/decode elastic buffer Rx detection
Physical Sub-block
Physical Media Attachment Layer (PMA)
Analog buffers SERDES 10-bit interface
Rx
Tx
Channel

4.2. Physical Coding Sublayer:

PCS block diagram is shown in the Figure 4-2. It has transmitter and receiver blocks. Transmitter block consist of 8b/10b encoder. Receiver Block consists of Elastic buffer, Rx detection and 8b/10b decoder.

Figure 4-2: Physical Coding Sublayer Block Diagram



A.8b/10b Encoder:

Each Lane of a device's transmitter implements an 8-bit to 10-bit Encoder that encodes 8-bit data or control characters into 10-bit symbols. The coding scheme was invented by IBM in 1982 and is documented in the ANSI X3.230-1994 document, clause 11 (and also IEEE 802.3z, 36.2.4) and US Patent Number 4,486,739 entitled "Byte Oriented DC Balanced 8b/10b Partitioned Block Transmission Code". 8b/10b coding is now widely used in architectures such as Gigabit Ethernet, Fibre Channel, ServerNet, FICON, IEEE1394b, InfiniBand, etc.

Purpose of Encoding a Character Stream:

The primary purpose of this scheme is to embed a clock into the serial bit stream transmitted on all Lanes. No clock is therefore transmitted along with the serial data bit stream. This eliminates the need for a high frequency 2.5GHz clock signal on the Link which would generate significant EMI noise and would be a challenge to route on a standard FR4 board. Link wire routing between two ports is much easier given that there is no clock to route, removing the need to match clock length to Lane signal trace lengths. Two devices are connected by simply wiring their Lanes together.

Below is a summary of the **advantages** of 8b/10b encoding scheme:

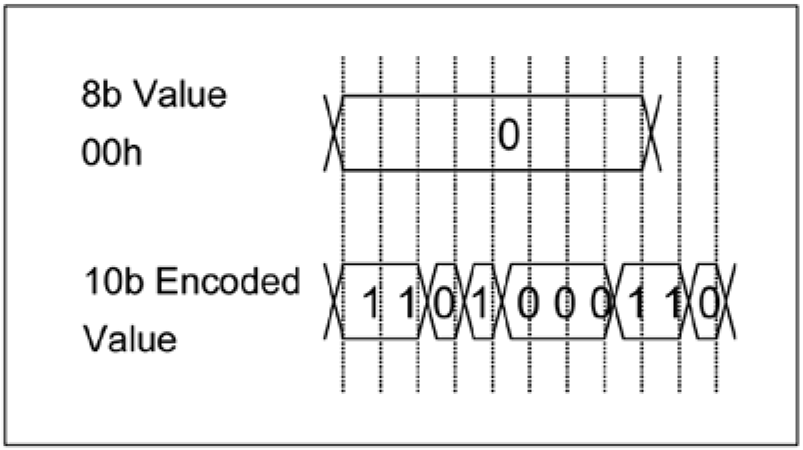
- **Embedded Clock**. Creates sufficient 0-to-1 and 1-to-0 transition density (i.e., signal changes) to facilitate re-creation of the receive clock on the receiver end using a PLL (by guaranteeing a limited run length of consecutive ones or zeros). The recovered receive clock is used to clock inbound 10-bit symbols into an elastic buffer. Figure 4.3 on page 420 illustrates the example case wherein 00h is converted to 1101000110b, where an 8-bit character with no transitions has 5 transitions when converted to a 10b symbol. These transitions keep the receiver PLL synchronized to the transmit circuit clock:

i) Limited 'run length' means that the encoding scheme ensures the signal line will not remain in a high or low state for an extended period of time. The run length does not exceed five consecutive 1s or 0s.

ii) 1s and 0s are clocked out on the rising-edge of the transmit clock. At the receiver, a PLL can recreate the clock by syncing to the leading edges of 1s and 0s.

iii) Limited run length ensures minimum frequency drift in the receiver's PLL relative to the local clock in the transmit circuit.

Figure 4-3: Example of 8-bit Character of 00h Encoded to 10-bit Symbol



- **DC Balance**. Keeps the number of 1s and 0s transmitted as close to equal as possible, thus maintaining DC balance on the transmitted bit stream to an average of half the signal threshold voltage. This is very important in capacitive- and transformer-coupled circuits.

  - Maintains a balance between the number of 1s and 0s on the signal line, thereby ensuring that the received signal is free of any DC component. This reduces the possibility of inter-bit interference. Inter-bit interference results from the inability of a signal to switch properly from one logic level to the other because the Lane coupling capacitor or intrinsic wire capacitance is over-charged.

- **Encoding of Special Control Characters**. Permits the encoding of special control ('K') characters such as the Start and End framing characters at the start and end of TLPs and DLLPs.
- **Error Detection**. A secondary benefit of the encoding scheme is that it facilitates the detection of most transmission errors. A receiver can check for 'running disparity' errors, or the reception of invalid symbols. Via the running disparity mechanism, the data bit stream transmitted maintains a balance of 1s and 0s. The receiver checks the difference between the total number of 1s and 0s transmitted since link initialization and ensures that it is as close to zero as possible. If it isn't, a disparity error is detected and reported, implying that a transmission error occurred.

The **disadvantage** of 8b/10b encoding scheme is that, due to the expansion of each 8-bit character into a 10-bit symbol prior to transmission, the actual transmission performance is degraded by 25% or said another way, the transmission overhead is increased by 25% (everything good has a price tag).

Properties of 10-bit (10b) Symbols:

- For 10-bit symbol transmissions, the average number of 1s transmitted over time is equal to the number of 0s transmitted, no matter what the 8-bit character to be transmitted is; i.e., the symbol transmission is DC balanced.
- The bit stream never contains more than five continuous 1s or 0s (limited-run length).
- Each 10-bit symbol contains:

  - Four 0s and six 1s (not necessarily contiguous), or

  - Six 0s and four 1s (not necessarily contiguous), or

  - Five 0s and five 1s (not necessarily contiguous).

- Each 10-bit symbol is subdivided into two sub-blocks: the first is six bits wide and the second is four bits wide.

  - The 6-bit sub-block contains no more than four 1s or four 0s.

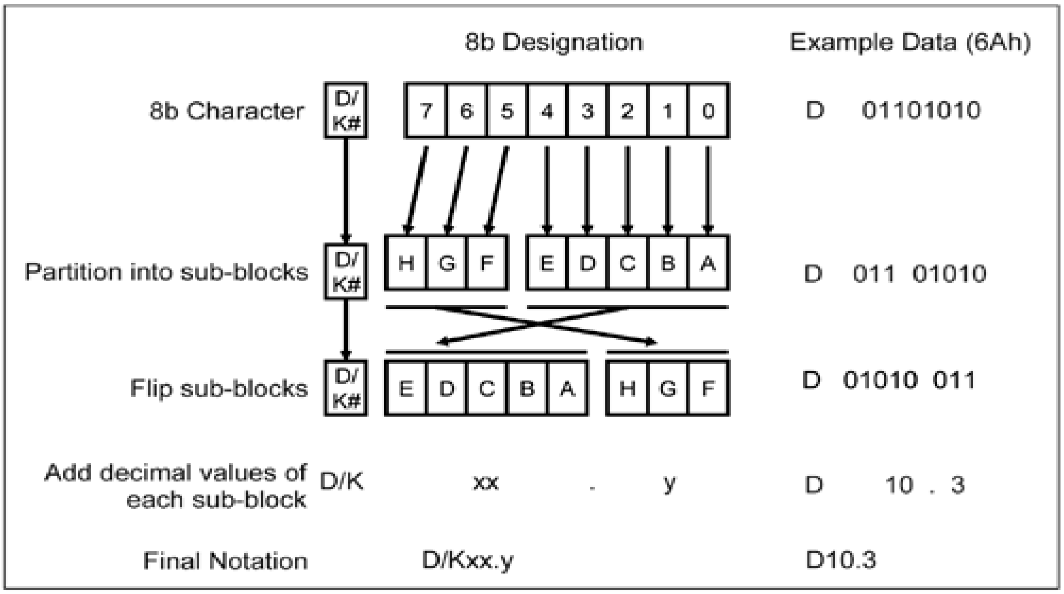
- The 4-bit sub-block contains no more than three 1s or three 0s.

- Any symbol with other than the above properties is considered invalid and a receiver consider this an error.
- An 8-bit character is submitted to the 8b/10b encoder along with a signal indicating whether the character is a Data (D) or Control (K) character. The encoder outputs the equivalent 10-bit symbol along with a current running disparity (CRD) that represents the sum of 1s and 0s for this transmission link since link initialization.
- The PCI Express specification defines Control characters that encode into the following Control symbols: STP, SDP, END, EDB, COM, PAD, SKP, FTS, and IDL.

Preparing 8-bit Character Notation:

8b/10b conversion lookup tables refer to all 8-bit characters using a special notation (represented by Dxx.y for Data characters and Kxx.y. for Control characters). Figure 4-4 illustrates the notation equivalent for any 8-bit D or K character. Below are the steps to covert the 8-bit number to its notation equivalent.

Figure 4-4: Preparing 8-bit Character for Encode



In Figure 4-4, the example character is the Data character, 6Ah.

1. The bits in the character are identified by the capitalized alpha designators A through H.
2. The character is partitioned into two sub-blocks: one 3-bit wide and the other 5-bit wide.
3. The two sub-blocks are flipped.
4. The character takes the written form Zxx.y, where:
5. Z = D or K for Data or Control,
6. xx = the decimal value of the 5-bit field,
7. y = the decimal value of the 3-bit field.
8. The example character is represented as D10.3 in the 8b/10b lookup tables.

Disparity: Definition

Character disparity refers to the difference between the number of 1s and 0s in a 10-bit symbol:

- When a symbol has more 0s than 1s, the symbol has negative (–) disparity (e.g., 0101000101b).
- When a symbol has more 1s than 0s, the symbol has positive (+) disparity (e.g., 1001101110b).
- When a symbol has an equal number of 1s and 0s, the symbol has neutral disparity (e.g., 0110100101b).
- Each 10-bit symbol contains one of the following numbers of ones and zeros (not necessarily contiguous):
  - Four 0s and six 1s (+ disparity).
  - Six 0s and four 1s (– disparity).
  - Five 0s and five 1s (neutral disparity).

Two Categories of 8-bit Characters

There are two categories of 8-bit characters:

- Those that encode into 10-bit symbols with + or – disparity.
- Those that encode into 10-bit symbols with neutral disparity.

CRD (Current Running Disparity):

The CRD reflects the total number of 1s and 0s transmitted over the link since link initialization and has the following characteristics:

- Its current state indicates the balance of 1s and 0s transmitted since link initialization.
- The CRD's initial state (before any characters are transmitted) can be + or –.
- The CRD's current state can be either positive (if more 1s than 0s have been transmitted) or negative (if more 0s than 1s).
- Each character is converted via a table lookup with the current state of the CRD factored in.
- As each new character is encoded, the CRD either remains the same (if the newly generated 10-bit character has neutral disparity) or it flips to the opposite polarity (if the newly generated character has + or – disparity).

8b/10b Encoding Procedure:

Refer to Figure 4-5. The encode is accomplished by performing two table lookups in parallel (not shown separately in the illustration):

- First Table Lookup: Three elements are submitted to a 5-bit to 6-bit table for a lookup (see Table 4-1 and Table 4-2):

  - The 5-bit portion of the 8-bit character (bits A through E).

  - The Data/Control (D/K#) indicator.

  - The current state of the CRD (positive or negative).

  - The table lookup yields the upper 6-bits of the 10-bit symbol (bits *abcdei*).

- Second Table Lookup: Three elements are submitted to a 3-bit to 4-bit table for a lookup (see Table 4-3 and Table 4-4):

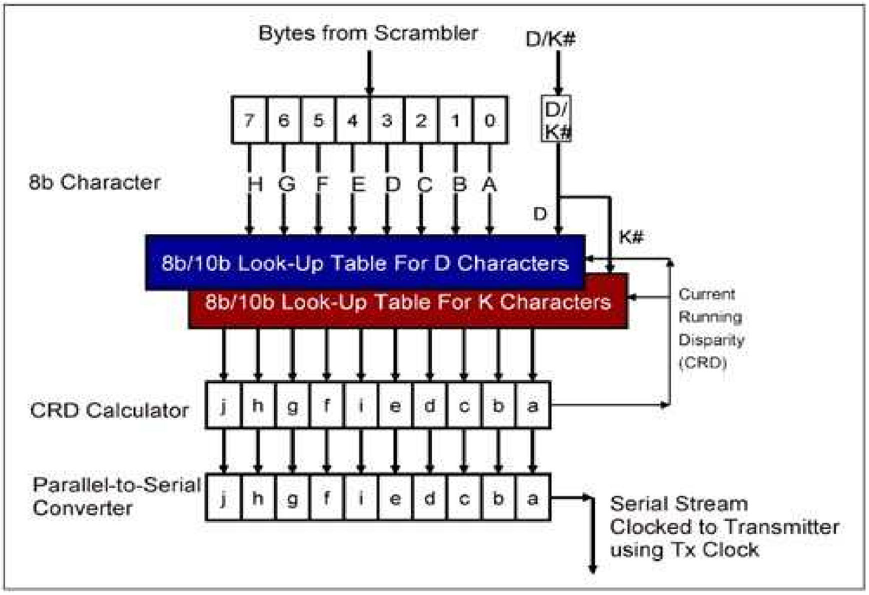
  - The 3-bit portion of the 8-bit character (bits F through H).

- The same Data/Control (D/K#) indicator.

- The current state of the CRD (positive or negative).

- The table lookup yields the lower 4-bits of the 10-bit symbol (bits *fghj*).

Figure 4-5. 8-bit to 10-bit (8b/10b) Encoder


Bytes from Scrambler
D/K#
7
6
5
4
3
2
1
0
D/K#
8b Character
H G F E D C B A
D
K#
8b/10b Look-Up Table For D Characters
8b/10b Look-Up Table For K Characters
Current Running Disparity (CRD)
CRD Calculator
j h g f i e d c b a
Parallel-to-Serial Converter
j h g f i e d c b a
Serial Stream Clocked to Transmitter using Tx Clock

The 8b/10b encoder computes a new CRD based on the resultant 10-bit symbol and supplies this CRD for the 8b/10b encode of the next character. If the resultant 10-bit symbol is neutral (i.e., it has an equal number of 1s and 0s), the polarity of the CRD remains unchanged. If the resultant 10-bit symbol is + or –, the CRD flips to its opposite state. It is an error if the CRD is currently + or – and the next 10-bit symbol produced has the same polarity as the CRD (unless the next symbol has neutral disparity, in which case the CRD remains the same).

The 8b/10b encoder feeds a Parallel-to-Serial converter which clocks 10-bit symbols out in the bit order 'abcdeifghj' (shown in Figure 4-5).

The Lookup Tables:

The following four tables define the table lookup for the two sub-blocks of 8-bit Data and Control characters.

Table 4-1: 5-bit to 6-bit Encode Table for Data Characters

Data Byte Name	Unencoded Bits EDCBA	Current RD – abcdei	Current RD + abcdei
D0	00000	100111	011000
D1	00001	011101	100010
D2	00010	101101	010010
D3	00011	110001	110001
D4	00100	110101	001010
D5	00101	101001	101001
D6	00110	011001	011001
D7	00111	111000	000111
D8	01000	111001	000110
D9	01001	100101	100101
D10	01010	010101	010101
D11	01011	110100	110100
D12	01100	001101	001101
D13	01101	101100	101100
D14	01110	011100	011100
D15	01111	010111	101000
D16	10000	011011	100100
D17	10001	100011	100011
D18	10010	010011	010011
D19	10011	110010	110010

Table 4-1: 5-bit to 6-bit Encode Table for Data Characters

Data Byte Name	Unencoded Bits EDCBA	Current RD – abcdei	Current RD + abcdei
D20	10100	001011	001011
D21	10101	101010	101010
D22	10110	011010	011010
D23	10111	111010	000101
D24	11000	110011	001100
D25	11001	100110	100110
D26	11010	010110	010110
D27	11011	110110	001001
D28	11100	001110	001110
D29	11101	101110	010001
D30	11110	011110	100001
D31	11111	101011	010100

Table 4-2: 5-bit to 6-bit Encode Table for Control Characters

Data Byte Name	Unencoded Bits EDCBA	Current RD – abcdei	Current RD + abcdei
K28	11100	001111	110000
K23	10111	111010	000101
K27	11011	110110	001001
K29	11101	101110	010001
K30	11110	011110	100001

Table 4-3: 3-bit to 4-bit Encode Table for Data Characters

Data Byte Name	**Unencoded Bits HGF**	**Current RD - fghj**	**Current RD + fghj**
--.0	000	1011	0100
--.1	001	1001	1001
--.2	010	0101	0101
--.3	011	1100	0011
--.4	100	1101	0010
--.5	101	1010	1010
--.6	110	0110	0110
--.7	111	1110/0111	0001/1000

Table 4-4: 3-bit to 4-bit Encode Table for Control Characters

Data Byte Name	**Unencoded Bits HGF**	**Current RD – fghj**	**Current RD + fghj**
--.0	000	1011	0100
--.1	001	0110	1001
--.2	010	1010	0101
--.3	011	1100	0011
--.4	100	1101	0010
--.5	101	0101	1010
--.6	110	1001	0110
--.7	111	0111	1000

Control Character Encoding:

Table 4-5 shows the encoding of the PCI Express-defined Control characters. These characters are not scrambled by the transmitter logic, but are encoded into 10-bit symbols. Because these Control characters are not scrambled, the receiver logic can easily detect these symbols in an incoming symbol stream.

These Control characters have the following properties

COM (comma) character. The COM character is used as the first character of any Ordered-Set. Ordered-Sets are a collection of multiples of 4 characters that are used for specialized purposes. The 10-bit encoding of the COM (K28.5) character contains two bits of one polarity followed by five bits of the opposite polarity (001111 1010 or 110000 0101). The COM (and FTS) symbols are the only two symbols that have this property, thereby making it easy to detect at the receiver's Physical Layer. A receiver detects the COM pattern to detect the start of an Ordered-Set. In particular, the COM character associated with TS1, TS2, or FTS Ordered-Sets are used by a receiver to achieve bit and symbol lock on the incoming symbol stream.

- **PAD** character. On a multi-Lane Link, assume the transmitter transmits the END character associated with a packet end on an intermediate Lane such as Lane 3 of a x8 Link. If the Link goes to the Logical Idle state after the transmission of the packet's END character, then the PAD character is used to fill in the remaining Lanes. This is done so packets as well as Logical Idle sequences always begin on Lane 0. For more information.
- **SKP** (skip) character. The SKP character is used as part of the SKIP Ordered-Set. The SKIP Ordered-Set is transmitted for clock tolerance compensation.
- **STP** (Start TLP) character. This character is inserted to identify the start of a TLP.
- **SDP** (Start DLLP) character. This character is inserted to identify the start of a DLLP.
- **END** character. This character is inserted to identify the end of a TLP or DLLP that has not experienced any CRC errors on previously-traversed links.
- **EDB** (EnD Bad packet) character. This character is inserted to identify the end of a TLP that a forwarding device (such as a switch) wishes to 'nullify'. Cut-through mode is a

mode in which the switch forwards a packet from its ingress port to an egress port with minimal latency without having to buffer the incoming packet first. A receiver that receives such a nullified packet discards it and does not return an ACK or NAK. Also see the chapter on Ack/Nak for a detailed description of the switch cut-through mode.

- **FTS** (Fast Training Sequence) character. This character is used as part of the FTS Ordered-Set. FTS Ordered-Sets are transmitted by a device in order to transition a Link from the low power L0s low power state back to the full-on L0 state.
- **IDL** (Idle) character. This character is used as part of the Electrical Idle Ordered-Set. The Ordered-Set is transmitted to inform the receiver that the Link is about to transition to the L0s low power state (also referred to as the Electrical Idle state of the Link).

Table 4-5: Control Character Encoding and Definition

Character Name	8b Name	10b (CRD-)	10b (CRD+)	Description
COM	K28.5 (BCh)	001111 1010	110000 0101	First character in any Ordered-Set. Detected by receiver and used to achieve symbol lock during TS1/TS2 Ordered-Set reception at receiver
PAD	K23.7 (F7h)	111010 1000	000101 0111	Packet Padding character
SKP	K28.0 (1Ch)	001111 0100	110000 1011	Used in SKIP Ordered-Set. This Ordered-Set is used for Clock Tolerance Compensation
STP	K27.7 (FBh)	110110 1000	001001 0111	Start of TLP character
SDP	K28.2 (5Ch)	001111 0101	110000 1010	Start of DLLP character
END	K29.7 (FDh)	101110 1000	010001 0111	End of Good Packet character

Table 4-5: Control Character Encoding and Definition

Character Name	8b Name	10b (CRD-)	10b (CRD+)	Description
EDB	K30.7 (FEh)	0111101000	1000010111	Character used to mark the end of a 'nullified' TLP.
FTS	K28.1 (3Ch)	0011111001	1100000110	Used in FTS Ordered-Set. This Ordered-Set used to exit from L0s low power state to L0
IDL	K28.3 (7Ch)	0011110011	1100001100	Used in Electrical Idle Ordered-Set. This Ordered-Set used to place Link in Electrical Idle state

B. Elastic Buffer:

Elastic Buffers (also known as Elasticity Buffers, Synchronization Buffers, and Elastic Stores) are used to ensure data integrity when bridging two different clock domains. This buffer is simply a FIFO (First-In-First-Out) where data is deposited at a certain rate based on one clock and removed at a rate derived from a different clock. Because these two clocks could (and almost always do) have minor frequency differences, there is the potential for this FIFO to eventually overflow or underflow. To avoid this situation, an Elastic Buffer has the ability to insert or remove special symbols, during specified intervals that allow the buffer to compensate for the clock differences.

The Elastic Buffer, or the concept of the Elastic Buffer, has been around since at least 1960. Maurice Karnaugh was granted a patent for this technology back in 1963 for its use in transmitting Pulse-Code Modulated (PCM) signals in telephone networks (*Pat. 3,093,815*). However, this was just the first of many applications that these buffers would be used in. The robustness of Elastic Buffers is indicated by their inclusion in modern day technologies. These buffers can be found in protocols such as USB, InfiniBand, Fibre Channel, Gigabit Ethernet, and of course PCI Express.

The Need for Elastic Buffers:

The PCI Express technology is a high-speed, serialized, source-synchronous timing (clock-forwarding), data transfer protocol. These characteristics are significantly different than the characteristics implemented by PCI Express' predecessor buses, namely PCI and PCI-X. Both PCI and PCI-X use a multi-drop, parallel bus that utilizes a synchronous timing architecture.

A synchronous timing architecture is where a common clock source supplies a clock to all the devices on the bus, and that clock is used to enable the device's transceivers to clock data in and out. This scheme requires that the clocks arrive at each device at precisely the same time. There is a very small amount of pin-to-pin skew allowed, which means that the lengths of these clock traces have to be matched to minimize the amount of skew between devices. It now becomes apparent that as the speed of the clock increases, the allowed pin-to-pin skew must decrease, which requires the matched routing of these clock traces to become increasingly difficult.

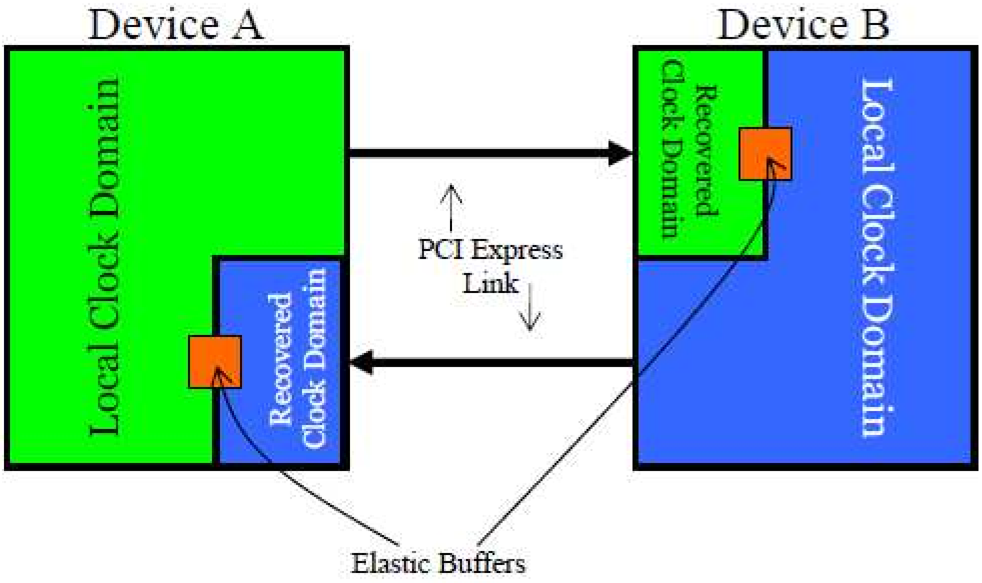
(1) The Local Clock Domain that it uses to clock all of its internal gates and to transmit data with, and

(2) The Recovered (or received) Clock Domain that it uses to latch inbound data. The data being received will have to cross the clock boundary from the Recovered Clock Domain to the Local Clock Domain in order for the device to process that information. Elastic Buffers are implemented in PCI Express devices for this exact purpose, to allow received data to transition from the Recovered Clock Domain to the Local Clock Domain.

How These Buffers Work:

All of the first generation PCI Express devices must transmit data at a rate of 2.5Gbps (Giga-bits per second) with a tolerance of +/- 300 ppm (parts per million). This translates into an allowed frequency range of 2.49925GHz – 2.50075GHz at the transmitter of a device. Due to the tolerance allowed, two devices connected to each other can (and most likely will) be running at slightly different frequencies. This creates two clock domains within one device that the received data must transition across. As shown in Figure 4-6, the function of the Elastic Buffer is to bridge these two clock domains by compensating for their phase and frequency differences, thereby allowing the received data to maintain its integrity as it flows upstream in the target device.

Figure 4-6: Elastic Buffers bridging the clock domains in each PCI Express device.



Bridging these two clock domains is accomplished by depositing the received data into the buffer using one clock domain (the recovered clock) and pulling the data out of the buffer using the other clock domain (the local clock of the device). Since these two clock domains can be running at different frequencies, the buffer has the potential to overflow or underflow. However, these error conditions are prevented by designing the buffer to monitor its own state (or fill-level) and enabling it to insert or remove special symbols at the appropriate rate (hence the name, Elastic Buffer). PCI Express defines these special symbols that can be inserted or removed as SKP symbols which are only found in SKP ordered-sets. A transmitted SKP ordered-set is comprised of a single COM symbol followed by three SKP symbols. Transmitters are required to send SKP ordered-sets periodically for the exact reason just discussed; to prevent an overflow or underflow condition in the Elastic Buffer.

The rate at which these SKP ordered-sets are transmitted is derived from the max tolerance allowed between two devices, 600ppm. This worst-case scenario would result in the local clocks of the two devices shifting by one clock cycle every 1666 cycles. Therefore, the transmitter must schedule a SKP ordered-set to be sent more frequently than every 1666 clocks. The spec defines the allowed interval between SKP ordered-sets being scheduled for transmission as being between 1180 – 1538 symbol times. (Since SKP symbols are removed and inserted an entire symbol at a time, the 1180-1538 interval is measured in symbol times, a 4ns period, and not bit times.) Upon receipt of the SKP ordered-set, the Elastic Buffer can either insert or remove up to 2 SKP symbols per ordered-set to compensate for the differences between the recovered clock and the local clock. There will be instances where a device will not be able to transmit a SKP orderedset within the defined interval of 1180-1538 symbol times. This occurs during the transmission of a large Transaction Layer Packet (TLP) that prevents the device sending a SKP ordered-set at the regularly scheduled time. Since the device cannot interrupt the transmission of the TLP, it must wait until the entire TLP is transmitted and then it can transmit the delayed SKP ordered-set(s). During the large TLP transmission, multiple symbol shifts can occur at the receiver between the Recovered Clock Domain and the Local Clock Domain. (The number shifts that can occur is described in a later section of this paper.) To ensure that the Elastic Buffer in the target device receives enough SKP ordered-sets to compensate for the multiple symbol shifts, the SKP ordered-sets that were scheduled for transmission during the TLP are accumulated. These accumulated SKP ordered-sets are sent consecutively, immediately following the end of the TLP.

Implementations and Sizes:

The sizes of these buffers are dependent on three factors: the interval at which the transmitter schedules SKP ordered-sets to be sent, the Max Payload Size supported by the device, and the Link Width. Each of these terms are used to determine the maximum amount of time between SKP orderedsets arriving at the receiver. The maximum amount of time between SKP orderedset being received may be longer than the specified 1180 – 1538 symbol times scheduling interval. For example, if the transmitter is currently in the middle of sending a TLP when it determines that another SKP ordered-set need to be sent, it must finish transmitting the current packet before sending the SKP ordered-set. This is why the Max Payload Size supported by a device and the

Link Width need to be considered, because the amount of time it takes to finish transmitting that packet affects the amount of time between SKP ordered-sets being transmitted on the link.

The worst-case scenario would be when two PCI Express devices are connected together with a x1 link, the transmitter has sent the first symbol of a maximum sized packet when it determines that it needs to send a SKP ordered-set. To properly calculate the sizes of the Elastic Buffer implementations (described later) we first must determine the number of symbols that can shift during this worst case scenario. As mentioned earlier, based on the clock tolerance allowed by the spec, two connected devices could have a symbol shift every 1666 symbol times. Therefore, the number of symbols that could shift during the worst-case scenario is described in the equation below.

$$Max\_Symbols\_Shifted = \frac{\left(\frac{Max\_Payload\_Size + TLP\_Overhead}{Link\_Width}\right) + 1538}{1666}$$

The *Max_Payload_Size* term in the equation above is the only variable and is based on the Max Payload Size parameter of the device. The *TLP_Overhead* term is a constant of 28 symbol times and is comprised of:

- Starting frame symbol (1)
- Sequence number (2)
- Header (16)
- ECRC (4)
- LCRC (4)
- End framing symbol (1)

The *Link_Width* term should be a constant of 1 and **NOT** the max link width that the device supports. This is because the smaller the link width, the longer the interval is between SKP ordered-sets, which is a component of the worst-case scenario. Since a device may not always know what the link width capability will be of the device it is connected to, the required link width of x1 should always be used in this equation. The 1538 value in the equation represents the max interval between SKP ordered-sets being scheduled by the transmitter. A sample calculation is shown below indicating the maximum number of symbols that can shift given a device's characteristics. Table 4-6 shows the results of the equation based on different Max Payload Sizes.

$$\frac{\left(\frac{4096+28}{1}\right)+1538}{1666}=3.4 Max\_Symbols\_Shifted$$

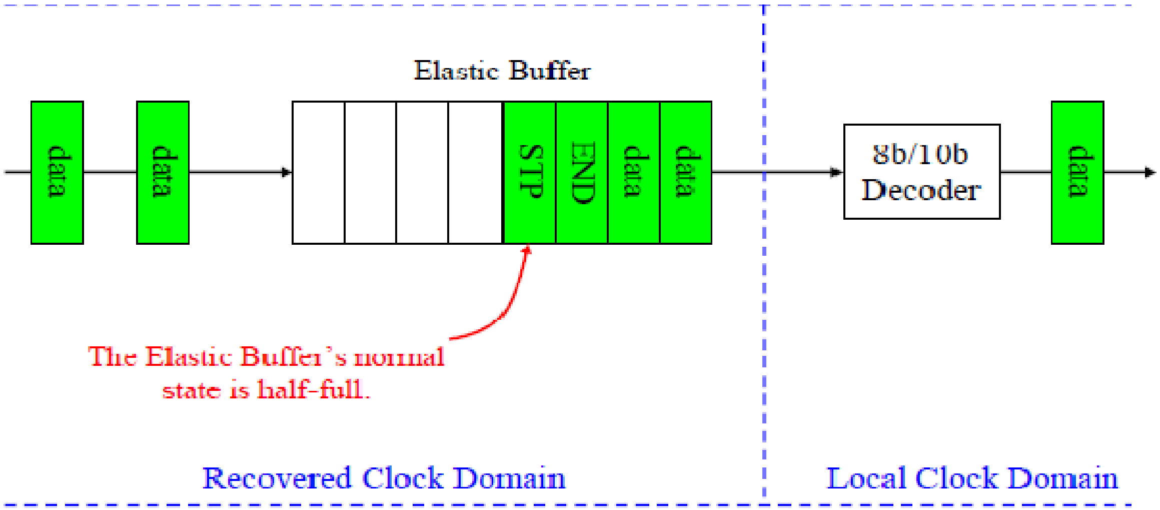
Table 4-6: Max number of shifted symbols based on payload size.

Max_Payload_Size	128	256	512	1024	2048	4096
Max_Symbols_Shifted	**1.02**	**1.09**	**1.25**	**1.55**	**2.17**	**3.40**

Procedure:

This implementation involves a buffer whose normal state is half-full (primed) with enough entries before and after the middle entry to ensure that the max clock differences can be compensated for by inserting or removing SKP symbols from SKP ordered-sets. This implementation is illustrated in Figure 4-7.

Figure 4-7: Implementation of Elastic Buffer using Primed Method



The number of entries before and after the middle entry is determined by the maximum number of symbols that could shift between received SKP ordered-sets. The number of symbol shifts is given in Table 4-6, based on the *Max_Payload_Size* parameter of the device. For example, a device which is capable of receiving a packet with a payload size of 4096 bytes should have an Elastic Buffer that is 8 entries deep. This is determined by taking the 3.40 result from Table 4-6 for a *Max_Payload_Size* of 4096 and rounding that value up to 4, then doubling that to allow for shifting in either direction. If the *Max_Payload_Size* parameter for a device is 2048, then using the result from Table 4-6, its Elastic Buffer should be 6 entries deep.

The goal of the Primed Method is to keep the Elastic Buffer half-full. Since the Figure 4-7 illustrates the scenario where the Local Clock is faster than the in Figure 4-9, the Elastic Buffer is getting close to overflowing because the Local Recovered Clock Domain and the Local Clock Domain will have slightly different frequencies, the Elastic Buffer's fill level will vary depending on the frequency difference of the two clock domains and the interval between received SKP ordered-sets. However, each time the device does receive one (or more) SKP ordered-sets, it returns the Elastic Buffer to the half-full state by inserting or removing SKP symbols from these SKP ordered-sets. (As mentioned earlier, it should be noted that not more than 2 SKP symbols can be inserted or removed from a single SKP ordered-set.) Examples of this process are shown in Figure 4-8, and also in Figure 4-9.

Figure 4-8: Local Clock Faster than Recovered Clock

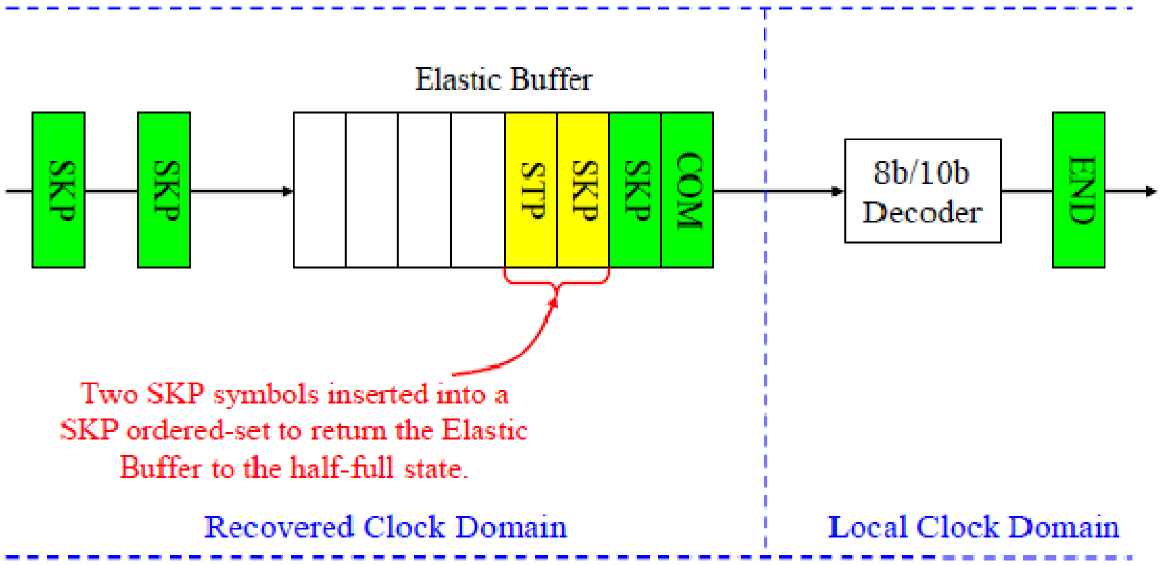
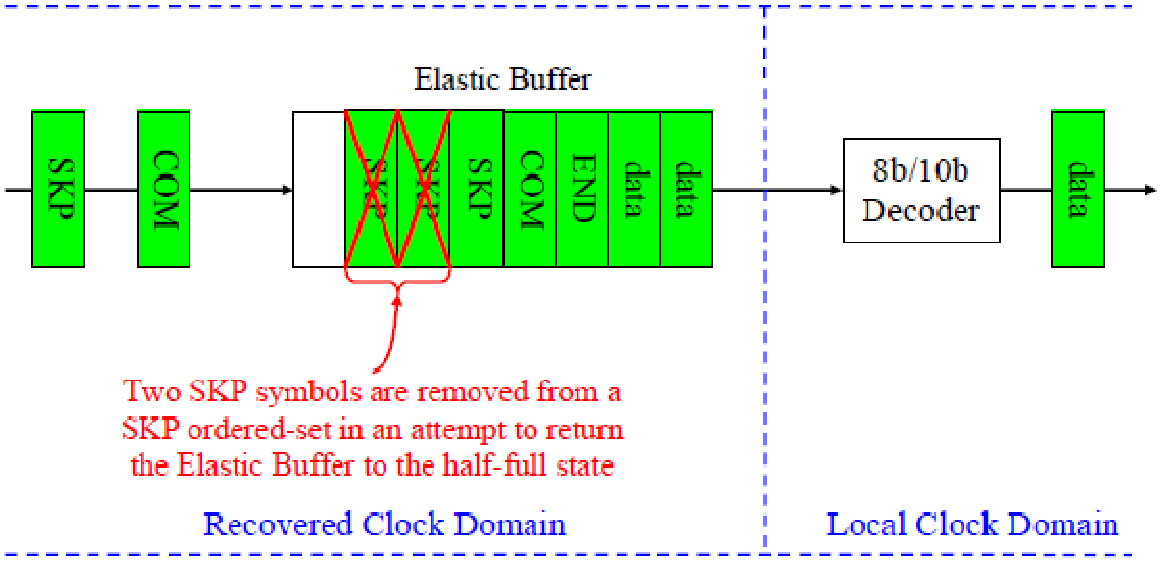
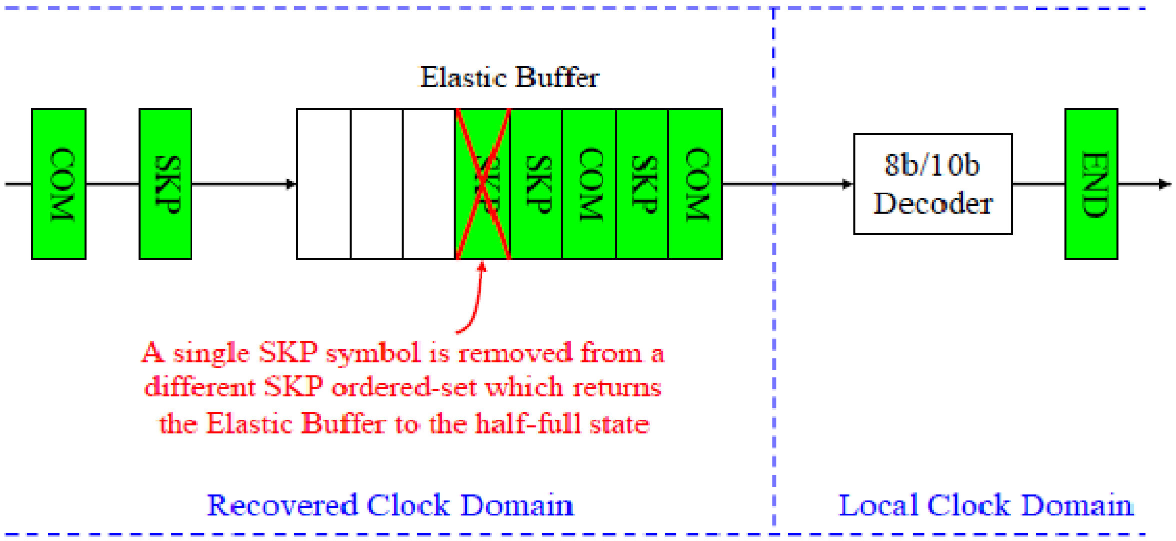

Elastic Buffer
SKP
SKP
STP
SKP
SKP
COM
8b/10b Decoder
END
Two SKP symbols inserted into a SKP ordered-set to return the Elastic Buffer to the half-full state.
Recovered Clock Domain
Local Clock Domain

Figure 4-9: Local Clock is Slower than Recovered Clock


Elastic Buffer
SKP
COM
SKP
SKP
SKP
COM
END
data
data
8b/10b Decoder
data
Two SKP symbols are removed from a SKP ordered-set in an attempt to return the Elastic Buffer to the half-full state
Recovered Clock Domain
Local Clock Domain

This is compensated for by inserting two additional SKP symbols into the received SKP ordered-set. In Figure 4-9, the Elastic Buffer is getting close to overflowing because the Local Clock is slower than the Recovered Clock and a SKP ordered-set has not been received in a long time. This situation occurs when a very large packet (possibly of max size) is being transmitted and thus SKP ordered-sets cannot be sent until the device finishes transmitting the TLP. In this case, a single SKP ordered-set is not sufficient to return the Elastic Buffer to its half-full state.

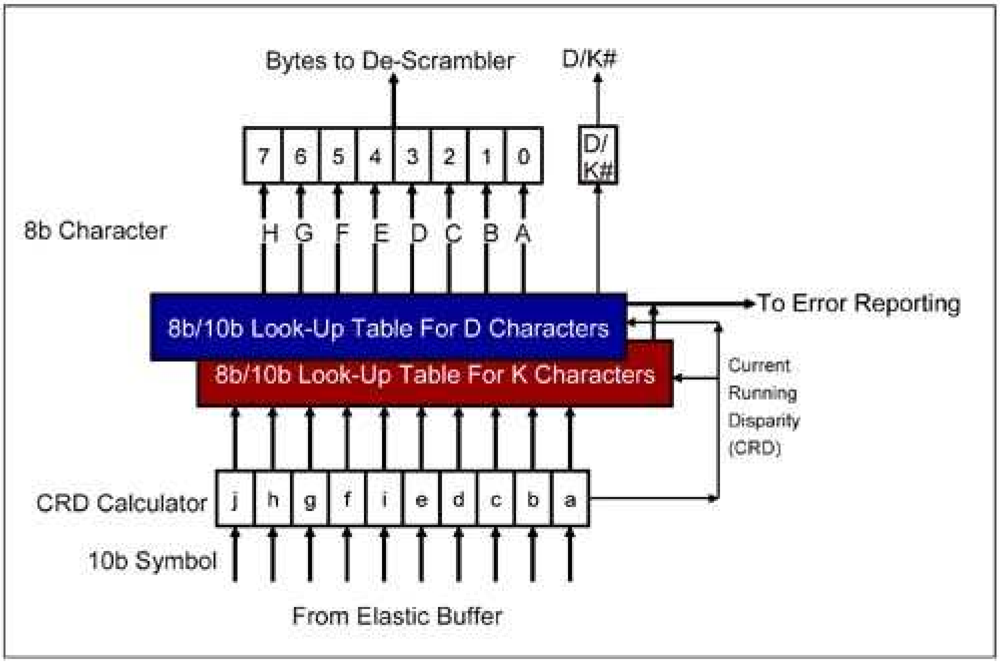
Figure 4-10: An additional SKP ordered-set must be used for clock tolerance compensation.


Elastic Buffer
COM
SKP
SKP
SKP
COM
SKP
COM
8b/10b Decoder
END
A single SKP symbol is removed from a different SKP ordered-set which returns the Elastic Buffer to the half-full state
Recovered Clock Domain
Local Clock Domain

C. 8b/10b Decoder:

Each receiver Lane incorporates a 10b/8b Decoder which is fed from the Elastic Buffer. The 8b/10b Decoder uses two lookup tables (the D and K tables) to decode the 10-bit symbol stream into 8-bit Data (D) or Control (K) characters plus the D/K# signal. The state of the D/K# signal indicates that the received symbol is:

Figure 4-11. 8b/10b Decoder per Lane



- A Data (D) character if a match for the received symbol is discovered in the D table. D/K# is driven High.
- A Control (K) character if a match for the received symbol is discovered in the K table. D/K# is driven Low.

Disparity Calculator:

The decoder determines the initial disparity value based on the disparity of the first symbol received. After the first symbol, once the disparity is initialized in the decoder, it expects the calculated disparity for each subsequent symbol received to toggle between + and - unless the symbol received has neutral disparity in which case the disparity remains the same value.

Code Violation and Disparity Error Detection:

The error detection logic of the 8b/10b Decoder detects errors in the received symbol stream. It should be noted that it doesn't catch all possible transmission errors. The specification requires that these errors be detected and reported as a Receiver Error indication to the Data Link Layer. The two types of errors detected are:

- Code violation errors (i.e., a 10-bit symbol could not be decoded into a valid 8-bit Data or Control character).
- Disparity errors.

There is no automatic hardware error correction for these errors at the Physical Layer.

Code Violations:

The following conditions represent code violations:

- Any 6-bit sub-block containing more than four 1s or four 0s is in error.
- Any 4-bit sub-block containing more than three 1s or three 0s is in error.
- Any 10-bit symbol containing more than six 1s or six 0s is in error.
- Any 10-bit symbol containing more than five consecutive 1s or five consecutive 0s is in error.
- Any 10-bit symbol that doesn't decode into an 8-bit character is in error.

Disparity Errors:

A character that encodes into a 10-bit symbol with disparity other than neutral is encoded into a 10-bit symbol with polarity opposite to that of the CRD. If the next symbol does not have neutral disparity and its disparity is the same as the CRD, a disparity error is detected.

- If two bits in a symbol flip in error, the error may not be detected (and the symbol may decode into a valid 8-bit character). The error goes undetected at the Physical Layer.

Physical Layer Error Handling:

When the Physical Layer logic detects an error, it sends a Receiver Error indication to the Data Link Layer. The specification lists a few of these errors, but it is far from being an exhaustive error list. It is up to the designer to determine what Physical Layer errors to detect and report.

Some of these errors include:

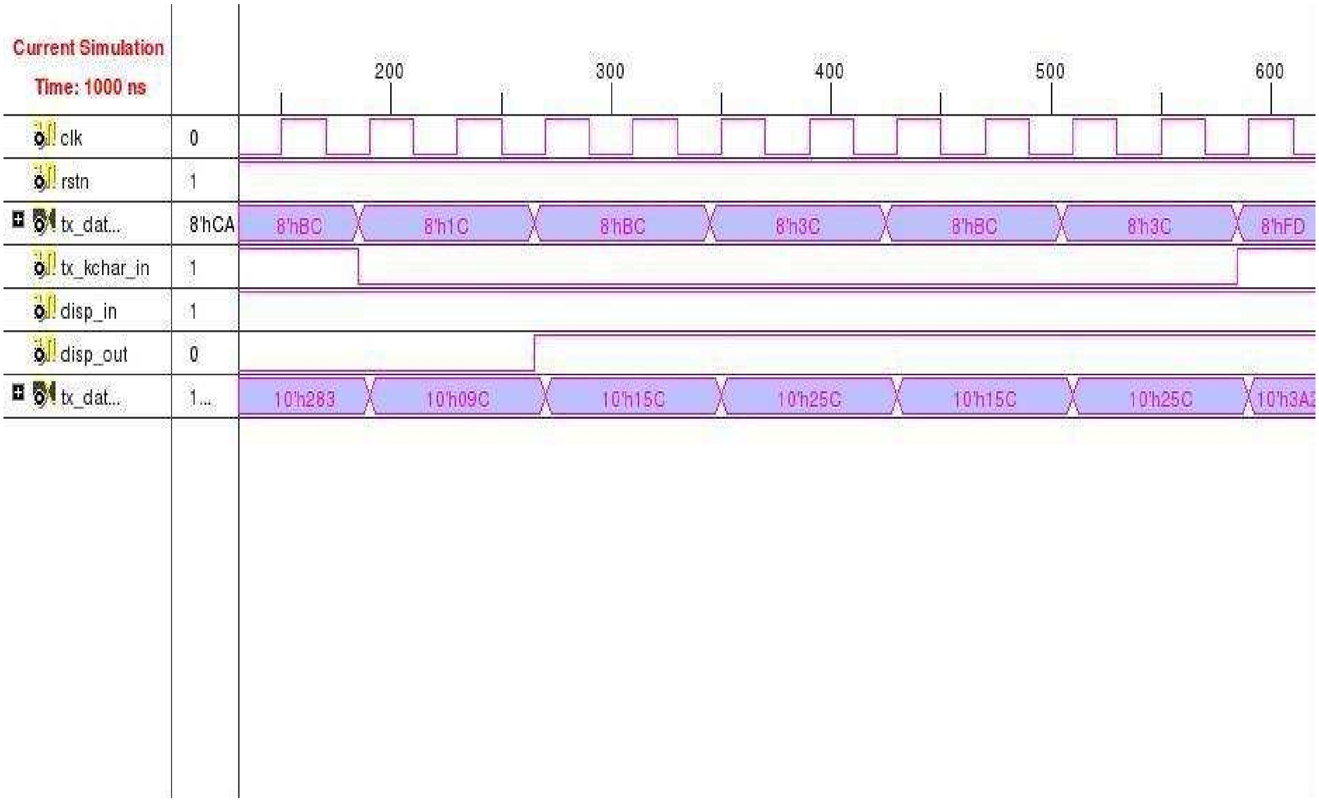
- 8b/10b Decoder-related disparity errors
- 8b/10b Decoder-related code violation errors
- Elastic Buffer overflow or underflow caused by loss of symbol(s)
- The packet received is not consistent with the packet format rules

4.3. Simulation Results & Discussions:

VHDL codes are written for 8b/10b encoder/decoder, elastic buffer blocks of PCS of Physical layer of PCI Express. These RTL codes are simulated, synthesized and implemented using the Xilinx ISE 10.1 tool. The results of simulation are reported here.

8b/10b Encoder:

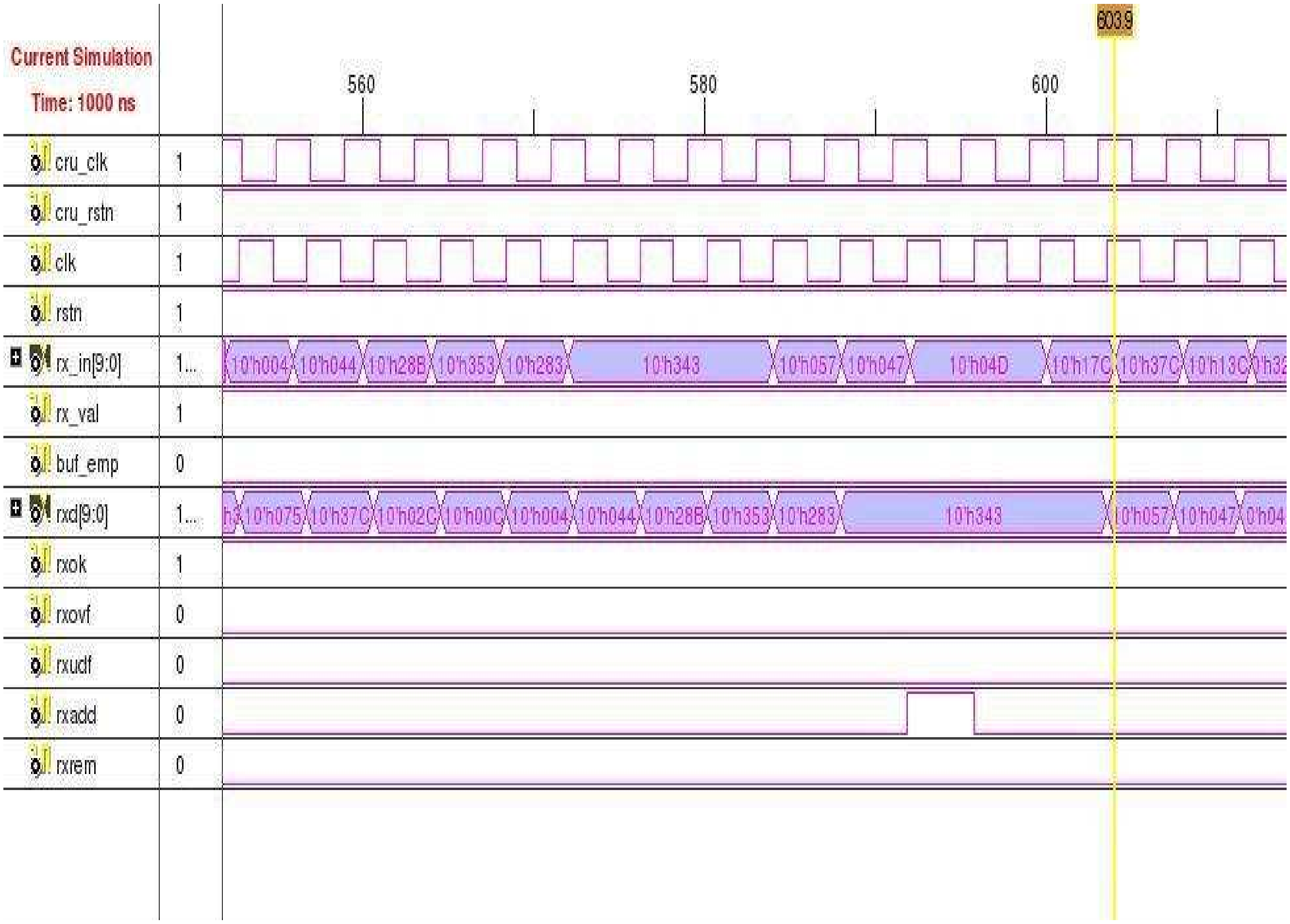
8b/10b Encoder module is operated with a clock speed of 250 MHz. The Input signals are clk, rstn, tx_data[7:0], disp_in and tx_kchar_in. Output signals are tx_data_out[9:0], disp_out. The signal rstn is active low signal. Reset is asserted when rstn signal is low. Remaining all the signals are active hing signals. This design is done using the lookup tables shown in Tables 4-1 to Table 4-5 in the previous sections. Disp_out shows the running disparity of the tx_data_out[9:0].



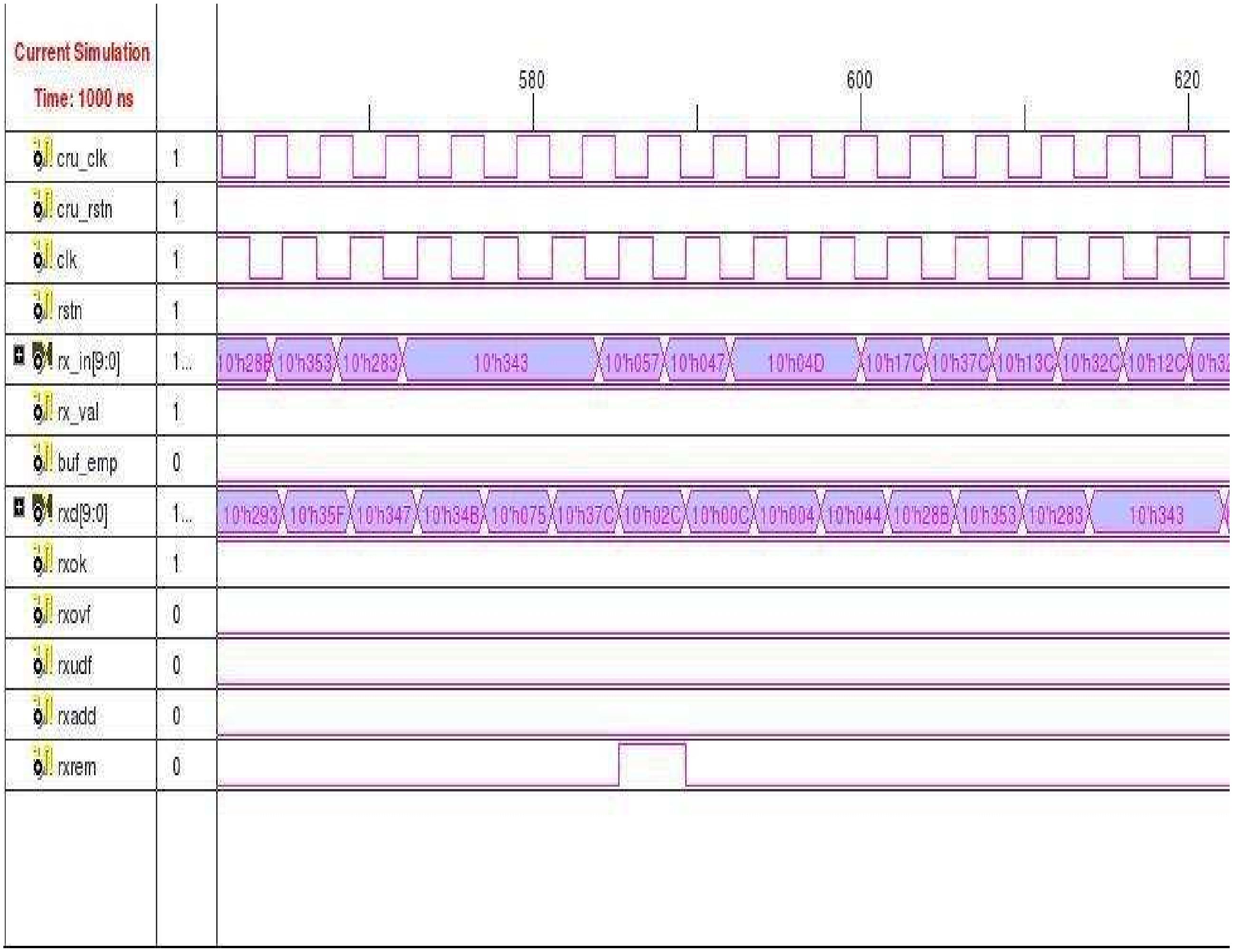
Elastic Buffer:

Elastic buffer works with two clocks clk and cru_clk. Input signals are rx_in[9:0], rstn, cru_rstn, rx_val. Output singals are rxd[9:0], rxok, rxovr, rxunf, rxadd, rxadd. The clock cru_clk is recovered clock form the data and clk is system clock. The two clocks are nearly equal to 250 MHz. cru_rstn and rstn singals are active low signals reminaing all singals are active high. Elastic Buffer uses the module dcfifo. First in first out (FIFO) is dulaport asynchoronus fifo.

Elastic Buffer when Recovered Clock speed less than System Clock:


Current Simulation
Time: 1000 ns
560
580
600
603.9
cru_clk 1
cru_rstn 1
clk 1
rstn 1
rx_in[9:0] 1...
10'h004 10'h044 10'h28B 10'h353 10'h283 10'h343 10'h057 10'h047 10'h04D 10'h17C 10'h37C 10'h13C
rx_val 1
buf_emp 0
rxd[9:0] 1...
10'h075 10'h37C 10'h02C 10'h00C 10'h004 10'h044 10'h28B 10'h353 10'h283 10'h343 10'h057 10'h047
rxok 1
rxovf 0
rxudf 0
rxadd 0
rxrem 0

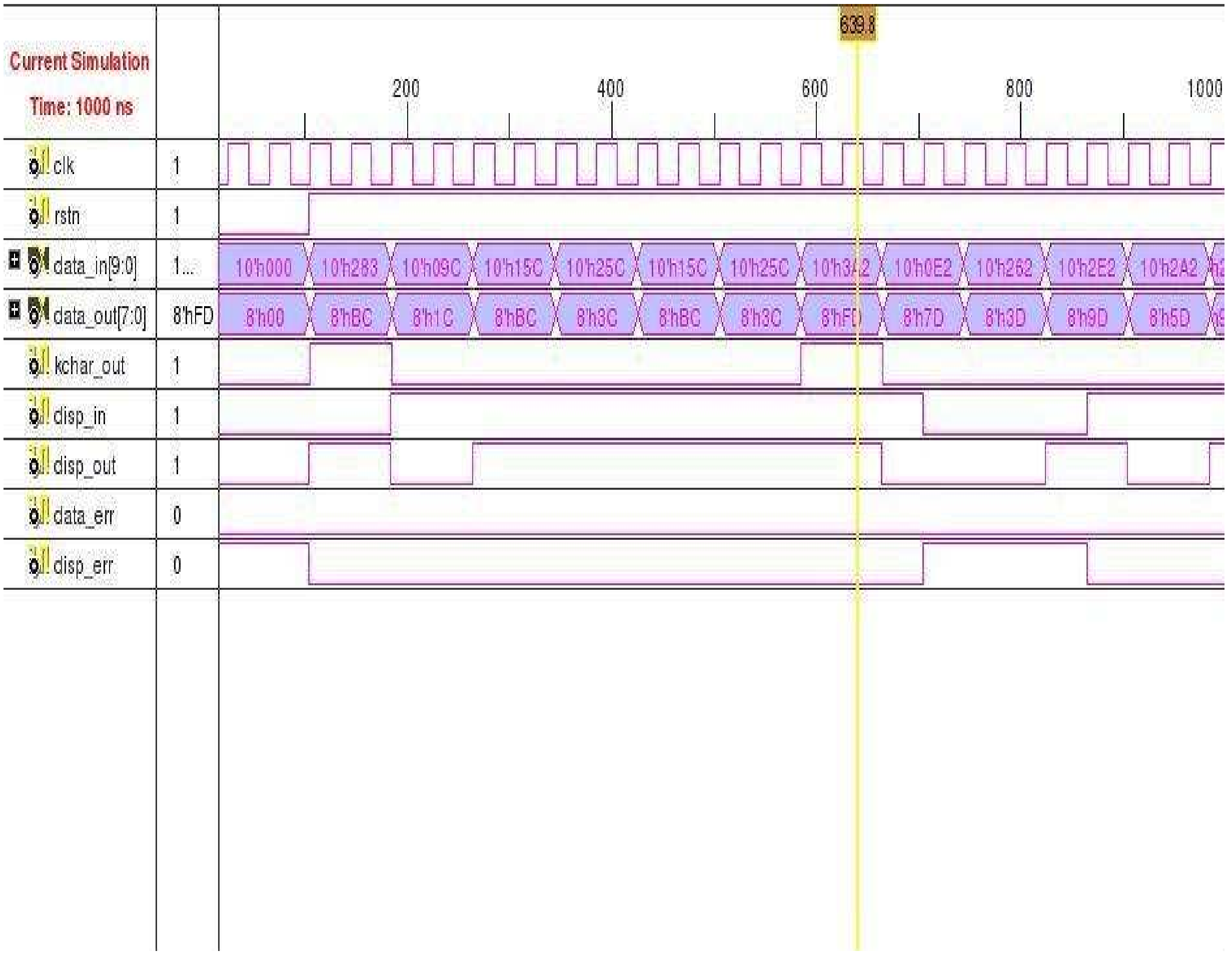
Elastic Buffer while Recovered Clock speed greater than System Clock:


Current Simulation
Time: 1000 ns
580
600
620
cru_clk 1
cru_rstn 1
clk 1
rstn 1
rx_in[9:0] 1...
10'h28B 10'h353 10'h283 10'h343 10'h057 10'h047 10'h04D 10'h17C 10'h37C 10'h13C 10'h32C 10'h12C 0'h32
rx_val 1
buf_emp 0
rxd[9:0] 1...
10'h293 10'h35F 10'h347 10'h34B 10'h075 10'h37C 10'h02C 10'h00C 10'h004 10'h044 10'h28B 10'h353 10'h283 10'h343
rxok 1
rxovf 0
rxudf 0
rxadd 0
rxrem 0

8b/10b Decoder:

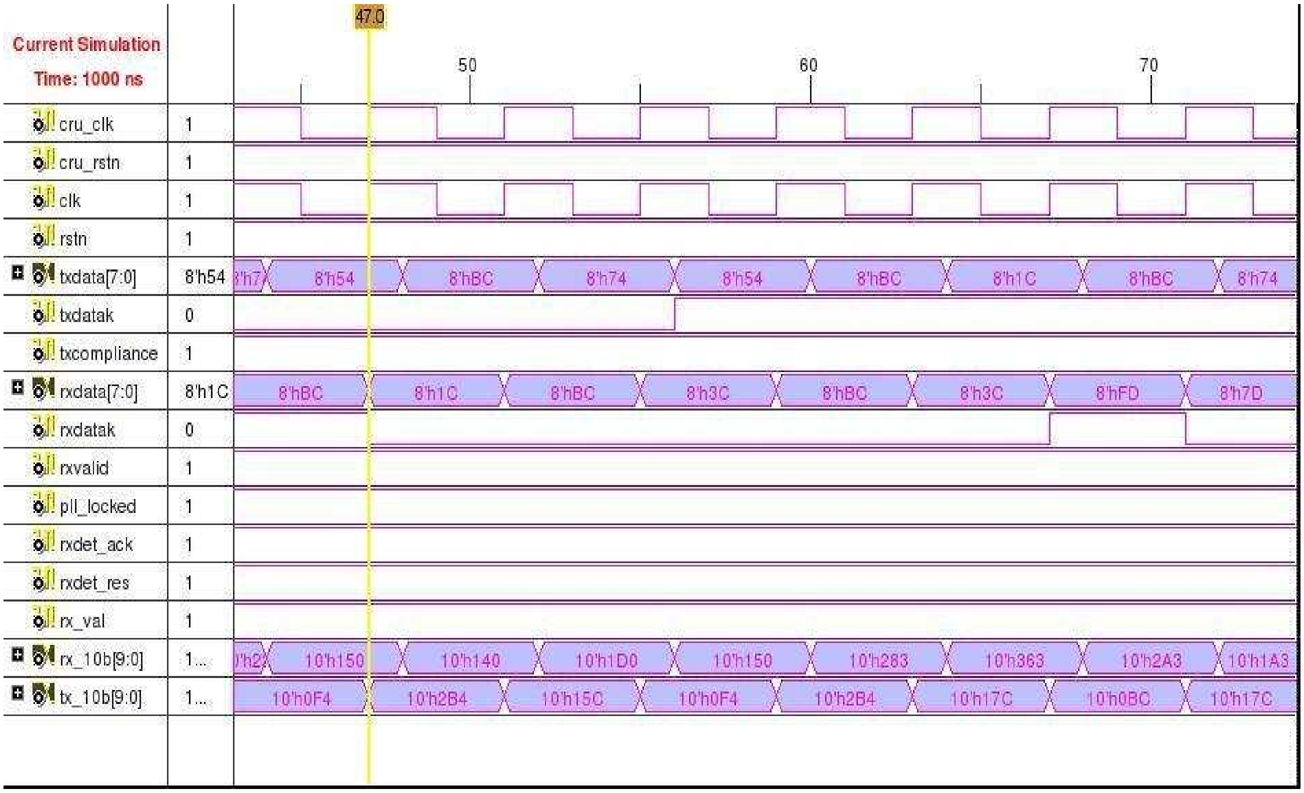
8b/10b Decoder module operates at 250 MHz. Input Siganls data_in[9:0], disp_in, rstn. Output signals data_out[7:0], kchar_out, disp_err, data_err. Rstn is active low signal. When it is logic'0'

rstn is asserted. This is modules is used lookup tables shown in the previous section. The received data is not matched with the data of the lookup table data error is dected and also disparity error also detected by the finding the number of ones and zeros of the received data.



PHY-PCS:

PCS is the top module for the 8b/10b encoder/decoder and elastic buffer. It has extra singals to 8b/10b encoder/decoder, Elastic Buffer. Those signals are pll_locked, rxvalid, rxdet_ack, rxdet_res. All these signals are active high signals. Txcompilance is input signal to say transmitter and receiver both are compliant to each other. If pll_locked is not asseted PCS module don't accept the rx_10b data.


Current Simulation
Time: 1000 ns
47.0
50
60
70
cru_clk 1
cru_rstn 1
clk 1
rstn 1
txdata[7:0] 8'h54
8'h7 8'h54 8'hBC 8'h74 8'h54 8'hBC 8'h1C 8'hBC 8'h74
txdatak 0
txcompliance 1
rxdata[7:0] 8'h1C
8'hBC 8'h1C 8'hBC 8'h3C 8'hBC 8'h3C 8'hFD 8'h7D
rxdatak 0
rxvalid 1
pll_locked 1
rxdet_ack 1
rxdet_res 1
rx_val 1
rx_10b[9:0] 1...
'h2 10'h150 10'h140 10'h1D0 10'h150 10'h283 10'h363 10'h2A3 10'h1A3
tx_10b[9:0] 1...
10'h0F4 10'h2B4 10'h15C 10'h0F4 10'h2B4 10'h17C 10'h0BC 10'h17C

Conclusion:

Physical Coding sublayer of Physical Layer of PCI Express has been designed.

8b/10b Buffer capable of encode 8 bit character and data/control signal into 10 bit symbol. If there is any invalid data, encoder detects that.

Elastic Buffer detects the underflow and overflow of the FIFO. It adds SKP symbol when the Buffer is not half filled. To maintain half-filled state it adds SKP symbol. It also removes SKP symbol when ever the buffer filled more than half-filled state. The purpose of synchronisation and to avoid the data loss is fulfilled through Elastic Buffer. This module also detects the overflow/underflow.

8b/10b decoder decodes 10 bit symbols into 8 bit character and data/control signal. It detects disparity error and decoder error through which the validity of received data is known.

PCS the major block where future developments are done. Usually in later came versions of PCI Express of 1.0, major developments were in the physical coding sublayer. The Encoding scheme has been changed in the PCI Express 3.0 as 8b/10b to 128b/130b. This almost overcomes the overhead as in PCI Express 1.0.

References:

1. *“PCI Express System Architecture”* by MindShare, Inc, Ravi Buduck, Don Anderson, Tom Shanley, Addison Wesley Publications, September 2003.
2. http://www.pcisig.com
3. *“PCI Express Base Specification”* v1.1, released by PCI-SIG, 2002.
4. *“PHY Interface for the PCI Express^{TM} Architecture”*, version 1.00, April 2003.
5. http://www.ati.amd.com
6. http://www.intel.com
7. *“PCI Express Expert Core Reference Manual”*, version 1.6.0, February 2006.
8. *“PHY Interface for the PCI Express Architecture”*, version 2.00, released by PCI-SIG, May 2008.