PCI Express-based Ethernet Switch

by

Caiyi Chen

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2012 by the
Graduate Supervisory Committee:

Joseph Hui, Chair
Martin Reisslein
Yanchao Zhang

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

A new type of Ethernet switch based on the PCI Express switching fabric is being presented. The switch leverages PCI Express peer-to-peer communication protocol to implement high performance Ethernet packet switching.

The advantages and challenges of using the PCI Express as the switching fabric are addressed. The PCI Express is a high-speed short-distance communication protocol largely used in motherboard-level interconnects. The total bandwidth of a PCI Express 3.0 link can reach as high as 256 gigabit per second (Gb/s) per 16 lanes. Concerns for PCI Express such as buffer speed, address mapping, Quality of Service and power consumption need to be considered.

An overview of the proposed Ethernet switch architecture is presented. The switch consists of a PCI Express switching fabric and multiple adaptor cards. The thesis reviews the peer-to-peer (P2P) communication protocol used in the switching fabric. The thesis also discusses the packet routing procedure in P2P protocol in detail. The Ethernet switch utilizes a portion of the Quality of Service provided with PCI Express to ensure guaranteed transmission.

The thesis presents a method of adapting Ethernet packets over the PCI Express transaction layer packets. The adaptor card is divided into the following two parts: receive path and transmit path. The commercial off-the-shelf Media Access Control (MAC) core and PCI Express endpoint core are used in the adaptor. The output address lookup logic block is responsible for converting Ethernet MAC addresses to PCI Express port addresses. Different methods of providing Quality of Service in the adaptor card include classification, flow control, and error detection with the cooperation of the PCI Express switch are discussed.

The adaptor logic is implemented in Verilog hardware description language. Functional simulation is conducted in ModelSim. The simulation results show that the Ethernet packets are able to be converted to the corresponding PCI Express transaction layer packets based on their destination MAC addresses. The transaction layer packets are then converted back to Ethernet packets. A functionally correct FPGA logic of the adaptor card is ready for implementation on real FPGA development board.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Joseph Hui. His enthusiasm, vision, challenge, encouragement, and financial support helped me through my research. Thanks to Dr. David Daniel, Haojun Luo, Deepthi Sai, Bala Aditya, and Veera Papirla who gave me technical advice, guidance and suggestion on my research. I would also like to acknowledge the contribution from my family and friends who gave me unconditional support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Ethernet, the dominant computer network, has become a well-established standard for Local Area Network (LAN). It is widely used for connecting machines in a limited area such as a building, company, or campus. The simplicity, cost-effectiveness, scalability, and increasing throughput of Ethernet have enabled other types of network applications such as Storage Area Networks (SAN), Internet Small Computer System Interface (iSCSI), and Fibre Channel over Ethernet (FCoE). The throughput of Ethernet has evolved from 10 megabits per second (Mbps) to 10 gigabits per second (Gbps) in approximately three decades. 40 Gigabit Ethernet (40GbE) and 100 Gigabit Ethernet (100GbE) have been proposed and implemented recently in 2010 [1][2].

The Ethernet switch is a critical component in computer networking. The switch is used for exchanging and forwarding Ethernet frames. An Ethernet switch is largely used for connecting different types of networks such as Ethernet, Fibre Channels, and Asynchronous Transfer Mode (ATM). Different applications of Ethernet include campus LAN access network and core network, data center network, and service provider aggregation. The switches have limited capacities and incur latency. The delay increases when Ethernet frames are forwarded through a cascade of switches, which results in performance degradation of the entire network. Thus, an efficient high throughput Ethernet switch with low latency is highly desirable.

1.1. Types and Issues of Ethernet Switches

There are three types of switching fabrics that currently used as Ethernet switching: shared bus, shared memory, and crossbar.

A typical shared bus switch structure is shown below in Figure 1. The switch consists of a shared bus medium with N input/output (IO) devices attached. Only one packet can traverse the bus at a time between two devices [3]. A centralized arbiter is used for determining the device to send packet on the bus. The total capacity and number of devices are bounded by bus capacity and performance of the centralized arbiter, respectively [4].

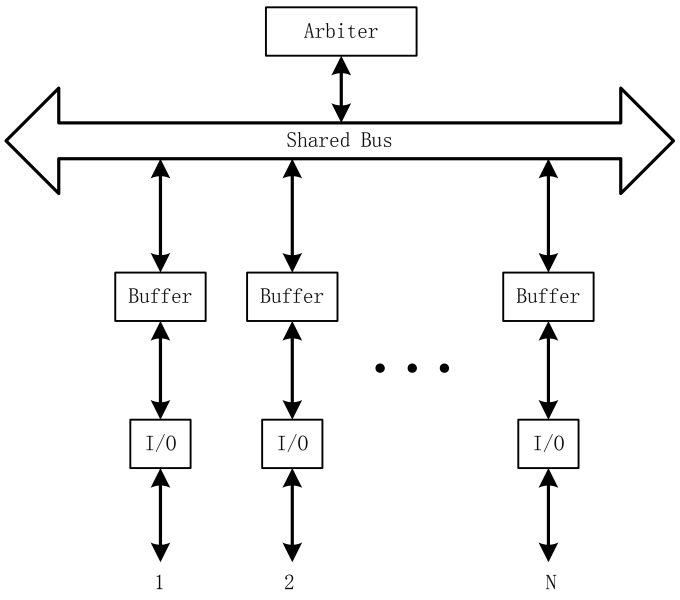


Figure 1: A Typical Shared Bus Switch

A typical shared memory switch is shown in Figure 2. It consists of a large capacity of memory, managed by memory controllers connected to the IO devices. The maximum input and output capacity of the shared memory switch is determined by the memory write and read bandwidth. Maximum capacity ranges from 4 Gbps to 10 Gbps. One major disadvantage of using a shared memory switch is the complexity and cost increase dramatically as bandwidth increases. A large shared memory switch also requires very wide memory access paths and very complex arbitration schemes.

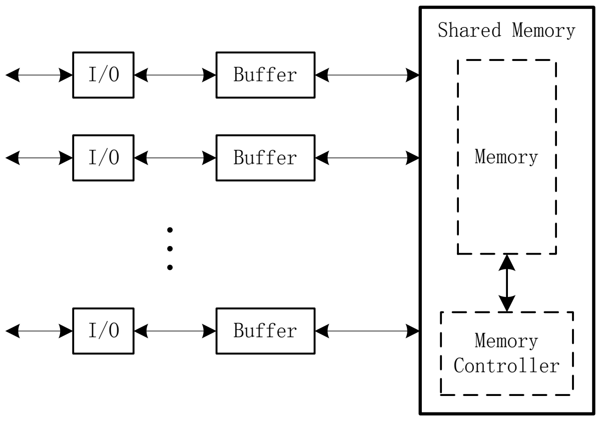


Figure 2: A Typical Shared Memory Switch

Another type of switching fabric is a crossbar. Instead of using time division in shared bus switch, a crossbar switch uses space division method. The crossbar switch can pass all the input data of multiple ports in parallel through the switching fabric, as shown in Figure 3. Each connection from an input port to an output port passes through a dedicated path through the crossbar switch. Total throughput increases linearly with the number of ports. Space division switch creates a relatively easier way to scale bandwidth than using higher capacities for shared bus or shared memory switch. However, one disadvantage of the crossbar switch is requiring contention resolution, which occurs when multiple inputs request connection to the same output. Output link utilization can fall as low as 60 percent of its raw capacity due to head-of-line (HOL) blocking [5].

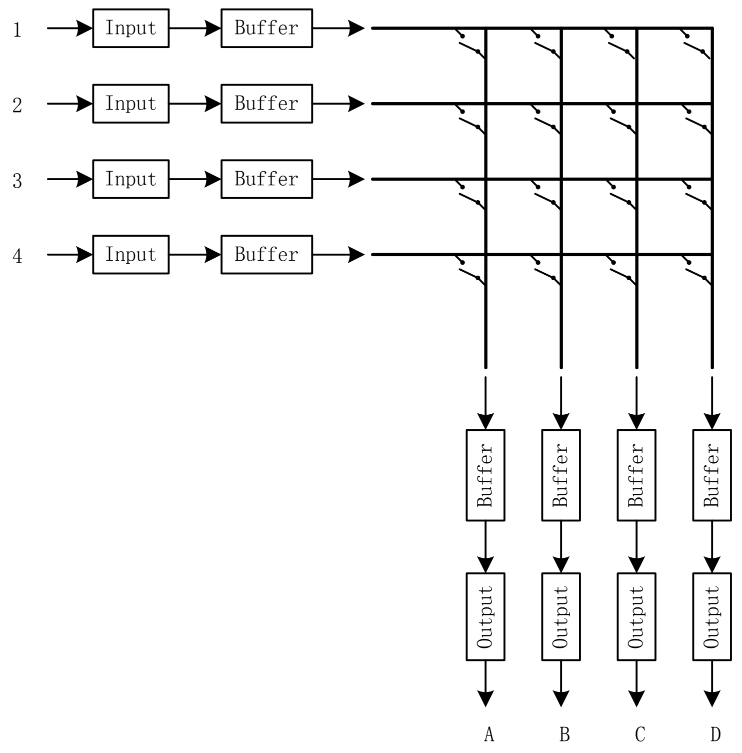


Figure 3: A Typical Crossbar Switch

1.2. A New Approach: PCI Express Switching for Ethernet

PCI Express (PCIe), the successor to AGP, PCI and PCI-X, is widely used as motherboard-level interconnect, passive backplane interconnect, and expansion card interface. PCIe is a high-speed point-to-point serial connection between devices. It uses packets as a basic transmission unit with a compatible layered protocol for different services. As of the latest technological advances, PCI Express Base Specification Revision 3.0 [6] defines a raw bit rate of 8 gigatransfers per second (GT/s), resulting in a 256 gigabit per second (Gb/s) total bandwidth for a link of 16 lanes. PCI-SIG recently announced PCI Express 4.0 would evolve into a 16 GT/s bandwidth; essentially doubling the throughput of PCI Express 3.0 [7].

The reasons for choosing PCI Express as a switching architecture for Ethernet are as follows. Firstly, a rich set of PCI Express protocols enables packets to be routed with a guaranteed bandwidth and excellent Quality of Service (QoS). In physical transmission, PCIe adopts point-to-point interconnect with differential signaling. The bandwidth of 256 Gb/s per link (with 16 lanes, per PCIe Specification 3.0) is a solid foundation for high throughput switching. PCI Express defines a packet routing scheme to regulate traffic to be switched in a fast and efficient manner. Rather than using bus cycles in PCI, PCIe uses packet-based transaction model to define packets format and boundaries, which leads to better data integrity. PCI Express provides Quality of Service such as traffic prioritization, flow control, and error detection/handling. These features provide reliable transmission, deterministic performance, and differentiated services.

Secondly, the crossbar switching fabric is a structural entity widely used in PCIe switching. Crossbar switch meets the growing need of highly-scalable and high-capacity Ethernet switches. An efficient Ethernet switching architecture is the crossbar switch with a contention resolution. The contention resolution issue in the PCIe switching can be solved by an arbitration infrastructure [8]. This functionality is able to support a set of arbitration policies that resolve traffic contention for an egress port from multiple ingress ports. Different arbitration policies can be applied to different applications for classified services.

Thirdly, peer-to-peer (P2P) communication protocol in PCIe enables traffic to be switched without involving the Central Processing Unit (CPU). Similar to Direct Memory Access (DMA) in modern computers, CPU is only responsible for initiating and terminating the P2P transaction between two devices in P2P communication. Peer devices have fully control of the communication procedure during the transaction. If the transaction is completed, one of the peer devices will report to the CPU. If an error occurs during the transaction, one of the peer devices will send an interrupt to the CPU. The CPU is only occupied at certain points of the entire transaction. The CPU is free to proceed to other unrelated tasks. Hence, P2P communication reduces switching latency, process overhead, and power consumption.

Finally, PCI Express switches are much more competitive than Ethernet switches in terms of their market prices. The cost of a traditional Ethernet switch is relatively high, especially for throughput of over 10 Gbps/port. For example, the price of an Ethernet 20-port Switch with a total capacity of 200 Gbps (IBM 46C7191 10Gb) is $9,543, while a 10-port PCI Express Gen2 switch with a total capacity of 416 Gbps (OSS-PCIe-1U-SW-x4-2.0) is $2,499 [9]. Low cost large capacity commercial off-the-shelf PCI Express switch chips are massively produced for board-level interconnect applications. It is possible to build an affordable Ethernet switch based on the PCI Express switch architecture.

1.3. Challenges of PCI Express Switching

Several challenges exist in the PCI Express switching.

a) <u>Buffer speed</u>. There could be multiple buffers in the PCI Express switching fabric and adaptor card for smoothing traffic flow from links of different speed. The total bandwidth of a PCIe link can reach as high as 256 Gb/s (with 16 lanes per PCIe 3.0). Thus, for a 40-byte packet passing through a channel without classification, the cycle time of the buffer memory at the port is around 1.25ns. Considering arbitration, address mapping, and error detection overhead, the actual requirement may be much higher. This buffering speed is challenging with today's memory technology [10].

b) <u>Arbitration</u>. Arbitration logic in PCI Express switching fabric is used for determining packets to be routed from a specific virtual channel (input buffers) or port at a certain time. Consider two 40-byte packets to be routed from two 256

Gb/s links to a single 256 Gb/s link. The arbitration logic has only 1.25ns to solve the contention. The contention resolving time becomes smaller when the number of ports increases [10].

c) Address Mapping. When an Ethernet packet with new source MAC address is received by the adaptor card, the source address must be stored associated with the receiving port number of the PCIe switch. The header fields are analyzed and reported to the address mapping logic. The address mapping logic translates the destination MAC address to the corresponding target port number of the PCIe switch. The address mapping procedure and the address mapping table management incur overhead and latency.

d) Quality of Service. The Quality of Service feature becomes difficult to implement at a higher link speed. First, the classification mechanism divides traffic into different categories. The Ethernet priority indicator to traffic class mapping as well as the traffic class to virtual channel mapping require a much higher speed. Second, the flow control algorithm needs to be perfectly selected and tuned to meet variable types of traffic. Last, error detection involves considerable computational overhead, especially when packet sizes are very small.

e) Power consumption. PCI Express provides different device and link power states for efficient power management. There is always a tradeoff among exit latency (the time to switch between different power states), power saving, and operational robustness [11]. Level zero (L0) power state has good exit latency, but often leads to link failure and large power consumption. Level one (L1) power state saves more power but has poor exit latency. More testing is needed when using lower power state while ensuring switches are working robustly.

Power consumption becomes more critical for larger capacities of PCIe switching systems.

f) Testing and debugging. PCI Express adopts serial bus instead of parallel bus topology for PCI. In a parallel bus, all signals in a data transfer can be probed simultaneously. In a serial bus like PCI Express, data is conveyed by a differential pair. The clock and header information are embedded in the signal. Interpreting those signals into a meaningful representation of different layers requires advanced tools [12].

1.4. Thesis Overview

The demand is to design a unique high performance and cost-effective multi-gigabit Ethernet switch by leveraging recent advances in PCI Express switching technologies. The thesis proposes a PCI Express-based Ethernet switch. This allows Ethernet traffic to be switched over the PCI Express architecture with a high throughput and low cost.

The thesis makes the following contributions. First, research is conducted for using peer-to-peer communication protocol to switch Ethernet traffic via PCI Express. PCI Express is widely used for interconnection between hosts and devices, while peer-to-peer communication is seldom used for personal computers or servers. Although the PCI Express Specifications define P2P communication, P2P packets routing methods are not fully discussed. In this thesis, device initialization process is reviewed and P2P packets routing procedure is presented. The Quality of Service considerations are reviewed for further discussion in Ethernet over PCIe adaptation.

Secondly, essential adaptation protocols and logics are presented. Ethernet packets use MAC addresses to indicate the target host, while PCI Express transaction layer packets use routing IDs or routing addresses to indicate target endpoint device or function. Thus, address mapping methods are discussed in this thesis, regarding how the mapping table is created and maintained. Output address lookup logic is presented for determining the correct outgoing PCIe routing ID/address based on the mapping table. A certain way to decapsulate and encapsulate Ethernet packets to form PCIe transaction layer packets is presented. Quality of service is discussed, which enables Ethernet packets to be transmitted efficiently outside the switch as well as cooperating with PCIe Quality of Service methods inside the switch to provide reliable data exchange.

Thirdly, the thesis presents a possible way to implement the PCIe-based Ethernet switch adaptor. A commercial 10G MAC Intellectual Property (IP) core is reviewed for part of the implementation. A universal interface of IP blocks called AXI4 is reviewed. Each individual logic block conforms to AXI4 interface specification. The output address lookup logic is implemented by an Ethernet parser logic, a Content-Addressable Memory (CAM), and an array. The parser logic is responsible for parsing Ethernet packets and providing source/destination MAC address, length/type, etc. The CAM is responsible for content matching and address lookup in an efficient way. In addition, First-In First-Out (FIFO) logic is customized based on a commercial universal AXI4 FIFO IP. FIFO is used for safely pass data from one clock domain to another asynchronous clock domain. A commercial PCI Express endpoint IP core is customized. PCIe interface logic blocks are presented for interfacing Ethernet MAC layer signals to PCIe transaction layer signals.

Finally, functional simulations are conducted for the logic implementation of the adaptor. The simulations demonstrate Ethernet packets to PCI Express transaction layer packets conversion and vice versa. The simulation uses a loop back configuration to verify the function of the adaptation logic blocks. The stimuli of Ethernet frames is fed into the 10G MAC core. The frames then pass through the receive path, loop back to the transmit path, and return to the MAC core. A possible functional simulation method for peer-to-peer switching using the PCIe endpoint core is discussed. The performance and issues of the proposed switch design is evaluated by these simulation results.

In summary, the PCIe-based Ethernet switch utilizes PCI Express peer-to-peer communication, transaction layer packet routing protocol, and internal crossbar switching fabric to implement Ethernet packet switching. The Ethernet over PCIe adaptor converts Ethernet packets to PCIe packets and feeds them to the PCIe switch seamlessly based on Field Programmable Gate Array (FPGA) logic blocks. The entire design leverages traffic class, virtual channel, flow control, and error detection mechanisms in PCIe along with new protocols defined in the adaptor to provide Quality of Service (QoS) for Ethernet packets switching.

1.5. Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 reviews the background of PCI Express and its expansion backplane. Chapter 3 presents an overview of PCI Express switching architecture. Chapter 4 gives possible methods of Ethernet over PCIe adaptation. Chapter 5 presents a possible design of the PCIe-based switch adaptor. Chapter 6 shows the experimental results. Chapter 7 concludes the entire thesis.

CHAPTER 2

BACKGROUND

2.1. PCI Express Architecture

The background knowledge of PCI Express is introduced here, especially that related to the proposed switching structure. PCI Express is a high-speed, high-performance, point-to-point, dual simplex, and differential signaling interconnection architecture among various devices. A PCI Express “link” is a physical connection between two devices. A “lane” is a pair of differential lines for transmitting and receiving signals. A PCI Express interconnect consists of either x1, x2, x4, x8, x12, x16 or x32 lanes of point-to-point link. For example, a x1 link consists of 1 lane or 1 differential signal pair in each direction for a total of 4 wires.

A typical PCI Express system is shown in Figure 4. PCI Express applies a hierarchical tree structure with a “Root Complex” located on the top level. Endpoint devices can be connected directly to the Root Complex or through a “Switch”. A Switch can have multiple endpoints connected to it, for switching traffic between a processor and an endpoint, or among endpoints. The internal switch consists of multiple virtual PCIe bridges, which connect different ports. Each endpoint device may implement up to 8 “functions”. A system could have up to 256 PCI Express links.

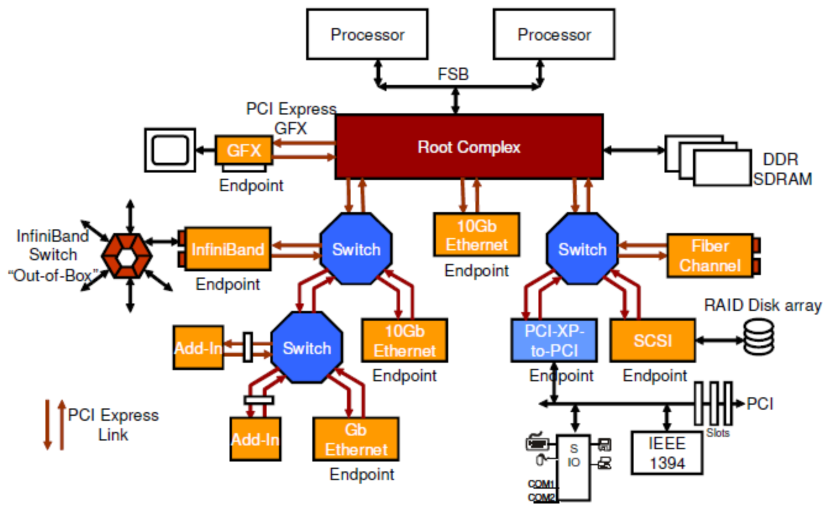


Figure 4: A Typical PCI Express System [12]

PCI Express uses transactions for data exchange. PCIe system supports memory read and memory write, I/O read and I/O write, configuration read and configuration write and message transactions. PCI Express encodes transactions using packet based protocols, which is different from previous PCI and PCI-X architectures using bus cycles. The PCI Express Specification defines a layered architecture for device design. Figure 5 shows the three layers structure, comprising Transaction Layer, Data Link Layer, and Physical layer.

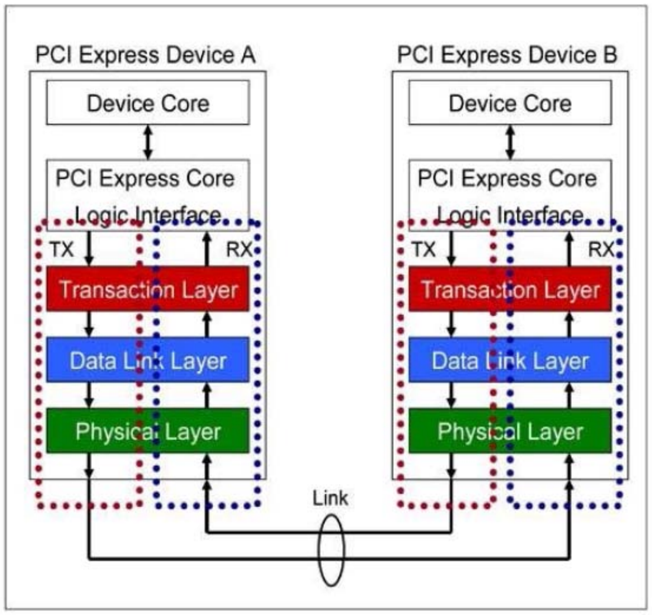


Figure 5: Three Layers Structure in PCI Express System

PCI Express 3.0 can achieve 8Gbit/s per lane per direction and 512Gbit/s total bandwidth for a x32 link. PCI Express also supports peer-to-peer communication which allows data to be directly transferred between different endpoints without involving a processor. This feature reduces the overhead of CPUs and increases the switching throughput. PCI Express also provides QoS control which divides traffic into traffic classes with different priorities. PCI Express is extensively used in the personal computer (PC) industry.

2.2. Motherboard and PCI Express Expansion Backplane

The motherboard is the most important circuit board in a PC, workstation, or server system. A motherboard holds most of the crucial components including the CPU in the system and provides connectors for other peripherals. It can be as simple as a backplane with multiple expansion cards, or as complex as a highly integrated circuit board supporting a full range of multimedia functions.

The PCI Express architecture is extensively used in computer systems due to its high speed board-level interconnect. Figure 6 shows one type of system on a motherboard, in which PCI Express buses are connected to a Northbridge (NB). Some of Intel® Northbridge chipsets are called I/O Hub (IOH) [13]. The Northbridge or IOH coordinates and connects the CPU, PCIe or AGP graphic interface, RAM and Southbridge (SB). Other types of systems are designed so that PCI Express buses are connected to CPUs directly. Some functions of the Northbridge are migrated into CPU to lower cost and improve performance.

Typically, PCs have two to six PCI Express slots; workstations and servers tend to have more. If more PCIe slots are needed, a PCI Express expansion backplane can be used for expanding PCIe connectivity outside the PC, workstation or server. The backplane uses a Host Bus Adaptor (HBA) to connect the host computer to the expansion backplane through a PCIe cable. The host treats devices on the backplane the same as devices connected directly to the motherboard.

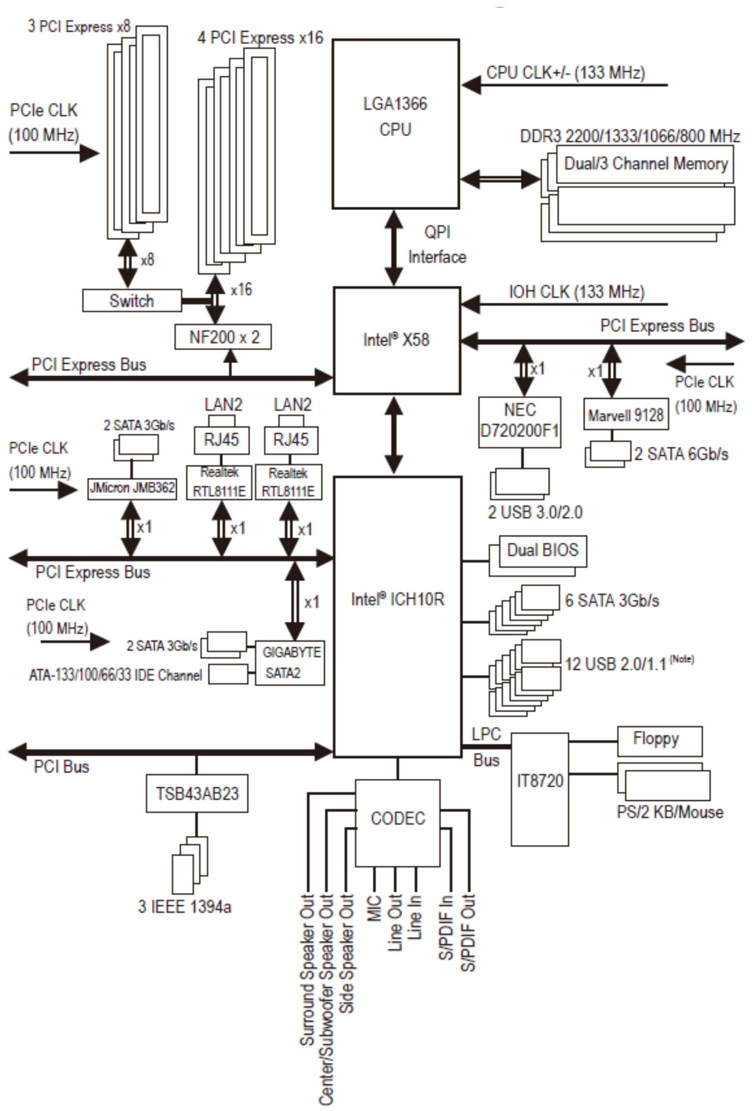


Figure 6: GIGABYTE™ GA-X58A-UD9 Motherboard Block Diagram [14]

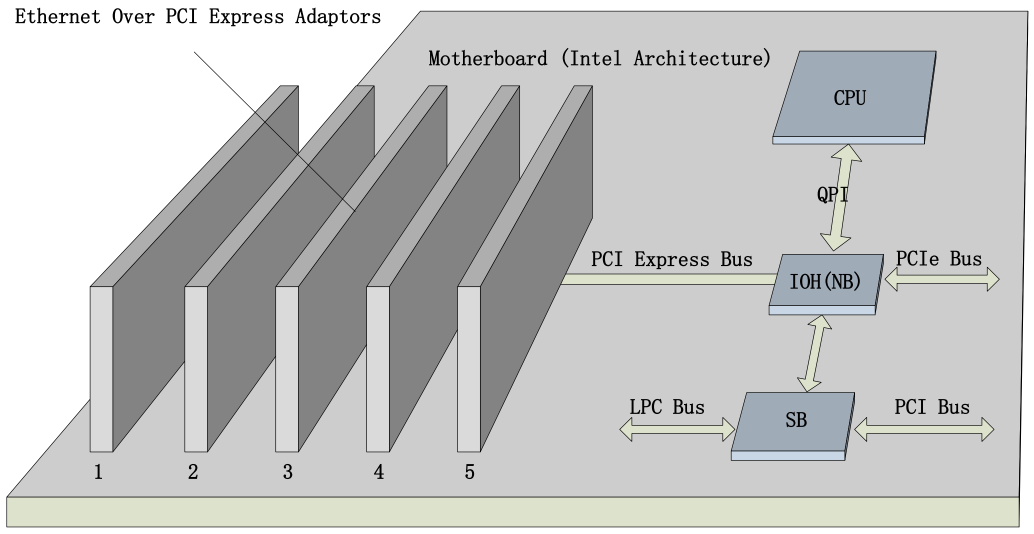
CHAPTER 3

PCI EXPRESS SWITCHING BY PEER-TO-PEER COMMUNICATION PROTOCOLS

3.1. Overview

3.1.1. Overview of the Proposed PCI Express-based Ethernet Switch

The proposed PCIe-based Ethernet switch consists of a motherboard and several Ethernet over PCIe adaptor cards as shown in Figure 7. The adaptors are plugged into PCIe slots of the motherboard. PCIe slots are connected to a PCI Express switch (inside the IOH, in this case). The IOH is controlled by the CPU through a QPI bus. Other peripherals (such as AGP bus) and chips (such as Southbridge) are connected to the IOH as well. Ethernet cables are connected to the adaptors through RJ-45 connectors or Small Form-factor Pluggable (SFP) connectors. Received Ethernet packets are processed by conversion logics in the adaptors. The adaptors can either adopt Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuit (ASIC) for packets processing. In a multi-port Ethernet switch application, users can connect a PCIe backplane to the motherboard through a Host Bus Adaptor (HBA) and PCIe cable (not drawn in this figure). The adaptors plugged into the backplane should be treated the same as local adaptors.



Ethernet Over PCI Express Adaptors
Motherboard (Intel Architecture)
CPU
QPI
PCI Express Bus
IOH(NB)
PCIe Bus
LPC Bus
SB
PCI Bus
1
2
3
4
5

Figure 7: Overview of the Proposed PCIe-based Ethernet Switch

3.1.2. Overview of Peer-to-Peer Communication

PCI Express peer-to-peer (P2P) communication enables data direct transfer between two PCI Express devices without involving many CPU cycles. Figure 8 illustrates the procedure of a P2P communication between two endpoint devices (adaptors in case of an Ethernet switch). Peer-to-peer traffic can be routed between multiple functions within an endpoint device, or be routed between devices through switches, even crossing hierarchy domains through a Root Complex as long as the Root Complex supports P2P.

Figure 8 illustrates the process of TLPs routing between adaptor 2 (requester) and adaptor 5 (completer) through a PCIe switch. First, Memory Read (MRd) TLP is sent by the requester to the PCIe switch. The ingress port of PCIe switch forwards the TLP to the correct egress port. The switch sends MRd to completer. The completer processes the request and sends back a Completion with data (CplD) to the switch ingress port. Finally, the switch forwards CplD to the egress port and then to the requester.

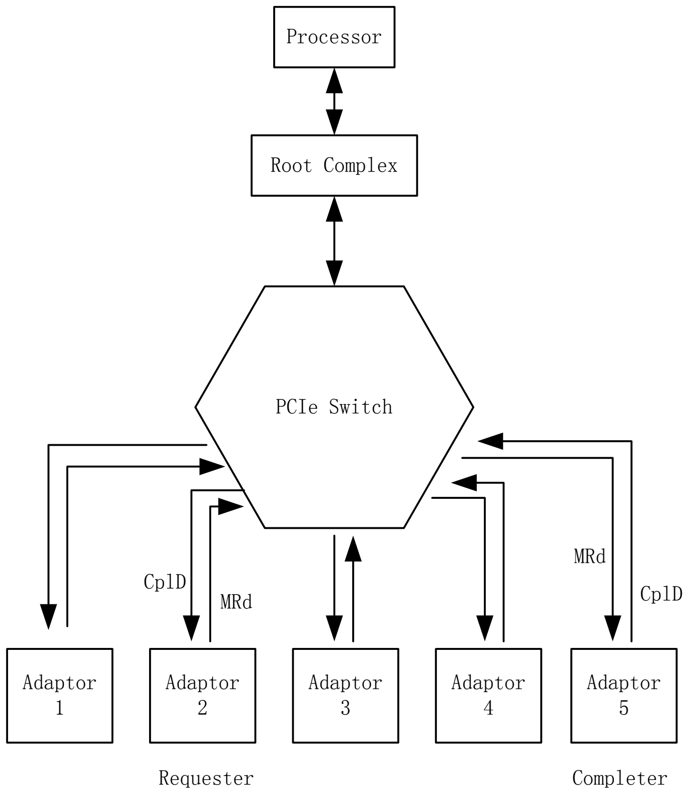


Figure 8: A Simple Demonstration of P2P Communication

3.2. Packet Routing Procedure

3.2.1. Enumeration and Initialization

Address information of PCI Express buses such as Bus, Device, Function (BDF) number and routing address are important for packet routing. When a system powers up, the configuration software (in the operating system) does not contain any address information of the PCI Express buses. Hence, the enumeration process is needed for discovering all the buses, devices, and functions in the system.

In the enumeration process, the configuration software attempts to read the header register (also called configuration space) in each device starting with device 0, bus 0 (the device attached to CPU directly) based on depth-first search. The device refers to both the endpoint and the switch/root complex port. Type 0 configuration space exists in endpoint devices and type 1 configuration space exist in switch or bridge devices. Different values returned from the header register indicate different types of devices. The configuration software also performs a series of configuration writes to set or update the bridge's bus number in each header register. The Primary Bus refers to the bus connected to the upstream side of the device. The Secondary Bus refers to the bus connected to the downstream side and the Subordinate Bus refers to the fastest downstream bus number.

After enumeration, the system software will have a complete list of BDF numbers in the system. Figure 9 shows part of the BDF information of an enumerated PCI Express system.

The configuration space in each device (including the endpoint and switch port) contains Base Address Registers (BARs). BARs are used for indicating the address range that a device can be accessed. An initialization process occurs after enumeration. During the initialization time, the startup configuration software tries to access the configuration space in each device. After the device receives the request from the software, the device sends back the amount of address space it needs (i.e. the size of BARs). The software then writes to the BARs in that device indicating the exact address range the device was assigned to. After the initialization process, address routing can be used in transactions. Transactions with addresses contained in the BARs will be accepted by that device.

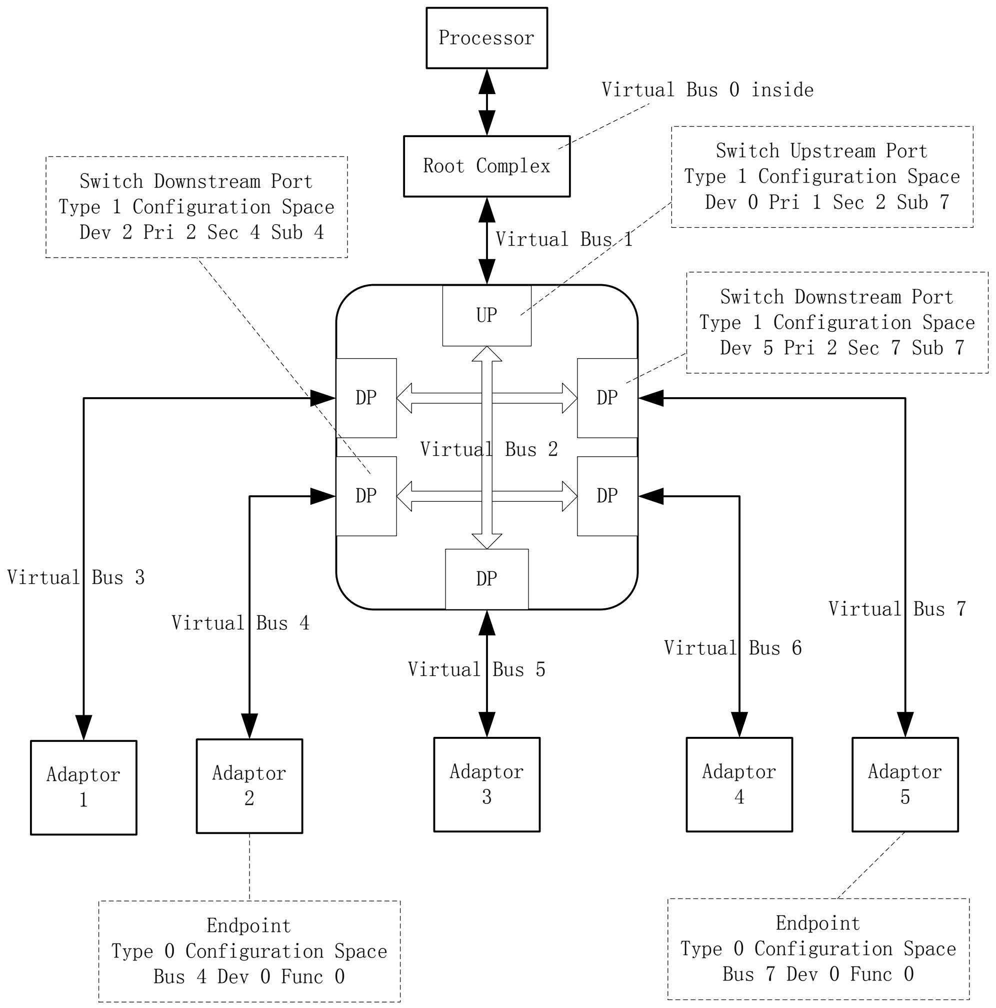


Figure 9: Enumerated PCIe System

3.2.2. Routing Scheme

There are three routing schemes in PCI Express: Address Routing, ID Routing, and Implicit Routing. The Address Routing scheme is mainly used for Memory Read (MRd), Memory Write (MWr), IO Read (IORd), and IO Write (IOWr) Transaction Layer Packets (TLPs). ID Routing scheme is mainly used for Configuration Read (CfgRd), Configuration Write (CfgWr) and Completion (Cpl) TLPs. Implicit Routing is mainly used for Message (Msg) TLPs. Note that although all three routing schemes are supported in peer-to-peer communication, configuration read and write TLPs using ID Routing are not allowed.

Take the following Memory Read (MRd) TLP routing as an example. Detailed routing procedures are given in Figure 9 and described as follows:

a) Addressing Routing scheme is used for MRd TLP routing. Endpoint device Adaptor 2 (Requester) obtains the routing address of Adaptor 5 (Completer) either from the OS or from an address table inside the device. Adaptor 2 then generates a MRd request TLP directed to Adaptor 5 with the routing address in the TLP header.
b) The downstream port with Device #2 of the switch receives the request. The port determines that the address lies outside its address range and forwards the transaction to its Primary Bus (Virtual Bus 2).
c) The downstream port with Device #5 determines that the address lies within its address range and forwards the transaction to Virtual Bus 7 through its Secondary Bus.
d) The Adaptor 5 (Completer) receives the transaction, determines that the address lies within one of the BARs, accepts the packet, and responds to the requester by generating one or more completion transaction(s) with data (CplD) using ID Routing. The Bus, Device, Function (BDF) number is obtained from the request transaction. In this case, Bus # is 4, Device # is 0, and Function # is 0.
e) The downstream port with device #5 receives the CplD, determines that the Bus #4 lies outside its Secondary Bus # 7and Subordinate Bus #7, and then forwards the CplD to its Primary Bus – Virtual Bus 2.

f) The downstream port with Device #2 receives the transaction, determines that the Bus #4 lies within its Secondary Bus #4 and Subordinate Bus #4, and then forwards the transaction to its Secondary Bus – Virtual Bus 4.

g) The Adaptor 2 receives the CplD, verifies the BDF and recognizes itself as the destination component. The Adaptor 2 then starts to process the data.

3.3. Quality of Service (QoS) and QoS-related Protocols

In general, Quality of Service (QoS) refers to the ability to allow traffic to be transferred with certain requirements. In the context of PCI Express switching fabric, QoS refers to the ability to provide a guaranteed bandwidth and latency for different types of transactions. Performance factors are taken into account of QoS, including transmission rate, effective bandwidth, latency, error rate, etc. Particularly in the PCI Express Specification 2.1, QoS focuses on providing differentiated services. The following sections discuss mechanisms that are critical to provide QoS and QoS-related features for the PCIe-based Ethernet switch.

3.3.1. Reliable and Efficient Switching

The transaction layer in PCI Express assumes that the lower layers provide a reliable transportation channel. The transaction layer is only responsible for transmitting and receiving transaction layer packets (TLPs) when informed by the application layer. The layer is not aware of the status of transmitted TLPs (e.g. whether the TLPs have arrived at the other end, and if some errors have occurred or not).

PCI Express utilizes ACK/NAK protocol to provide reliable transportation of transaction layer packets. Before a TLP is transmitted, the transmitter buffers the TLP to a replay buffer for later use. After the data link layer of a receiver receives a TLP from the transmitter, the data link layer first checks the cyclic redundancy check (CRC) field of the packet. The CRC field is calculated by the transmitter based on an error checking algorithm. The receiver calculates the CRC of the packet based on the same algorithm. If the CRC calculated by the receiver is the same as the CRC field received, the receiver will send an ACK data link layer packet (DLLP) to the transmitter. The TLP backed up in the replay buffer is then dropped. If the CRC calculated by the receiver is different than the CRC field received. The receiver will send a NAK DLLP to the transmitter. The transmitter will then resend the packet from the replay buffer. Hopefully, the TLP will be received correctly. If so, an ACK DLLP is send back and the replay buffer is cleared. If not, a NAK DLLP is sent back and the TLP will be resend again.

The ACK/NAK protocol ensures TLP to be transmitted in a reliable way without error or loss. However, efficiency issues arise when a large TLP is transmitted. We assume that a large TLP is transmitting from a transmitter of a network device to an ingress port of the PCIe switch. The ingress port is not able to check the CRC field until it receives the whole packet. Thus, the switch cannot forward the packet until receiving the entire packet and checking the CRC field. The latency of the switch at minimum is the time required to pass the packet through the ingress port and the processing time. If the packet needs to pass through multiple switches, the overall latency will increase.

One way to reduce latency is to use cut-through mode in PCI Express. In cut-through mode, the egress port of the switch starts forwarding the TLP immediately after the first bit arrives at the ingress port. When the entire packet is received, the switch will calculate and check the CRC field. If no errors occur, the switch will send the ACK DLLP to the transmitter. If an error occurs, the switch will attach an End Bad Packet (EDB) symbol to the end of the packet instead of the original END symbol. Meanwhile, the CRC field is inverted. The switch then sends a NAK DLLP to the transmitter for retransmission. When the target device receives the corrupted packet, the device detects the EDB symbol and inverted CRC field, and discard the packet. The receiver cannot send the NAK DLLP to the transmitter since the switch has already sent the NAK.

3.3.2. Classification

The Ethernet over PCI Express switch allows for different types of traffic that requires differentiated services. For example, normal Ethernet traffic should take higher priority than mass storage transactions, while taking a lower priority than video data for high performance video cards. PCI Express utilizes Traffic Class (TC) and Virtual Channel (VC) as part of a solution for providing differentiated services for different applications.

Different numbers of TCs represent different transmission priorities. TC values can be assigned from TC0 to TC7, which is carried in packet header. Larger values indicate higher priorities. TC0 must be implemented by every PCI Express device. VC identifies the physical transmission line into different virtual channels. VC0 to VC7 can be used for one link. VC0 must be implemented by every device.

Device drivers and firmware are able to configure the number of TCs and VCs that the device can support. Configuration software is responsible for setting up the VCs for each link in the PCI Express system. The number of VCs in each link is the smallest number of VCs each device attached to that link can support. For example, device A and switch port B is attached to the same link. Device A supports up to 2 VCs while port B supports up to 5 VCs. Hence, a number of 2 VCs are assigned to that link.

VC numbers or IDs are assigned by configuration software, except that VC0 is hardwired. Before transmission, different TCs must be mapped to VCs by software via VC Resource Control Register. Note that one or multiple TCs can be mapped to a single VC but one TC can’t be mapped to multiple VCs.

Figure 10 illustrates the classification structure of a PCI Express switch. When switching is performed, the receive paths of Adaptor 1 and Adaptor 2 convert Ethernet packets to PCIe Transaction Layer Packets (TLPs). The TLPs with different TCs are mapped to different VCs by a TC/VC mapping logic. Before routing through a crossbar fabric, TLPs from different VCs are scheduled to be sent by a VC arbitration logic (VARB). After proper routing, another TC/VC mapping logic resolves different TLPs from the link and sends them to multiple VCs of different ports. Port arbitration logics (PARBs) schedule the transmission of TLPs from different ports to aggregate to one VC. TLPs from different VCs are then arbitrated to the egress port by a VARB. Finally, TLPs are converted back to Ethernet packets by the transmit path of Adaptor 3.

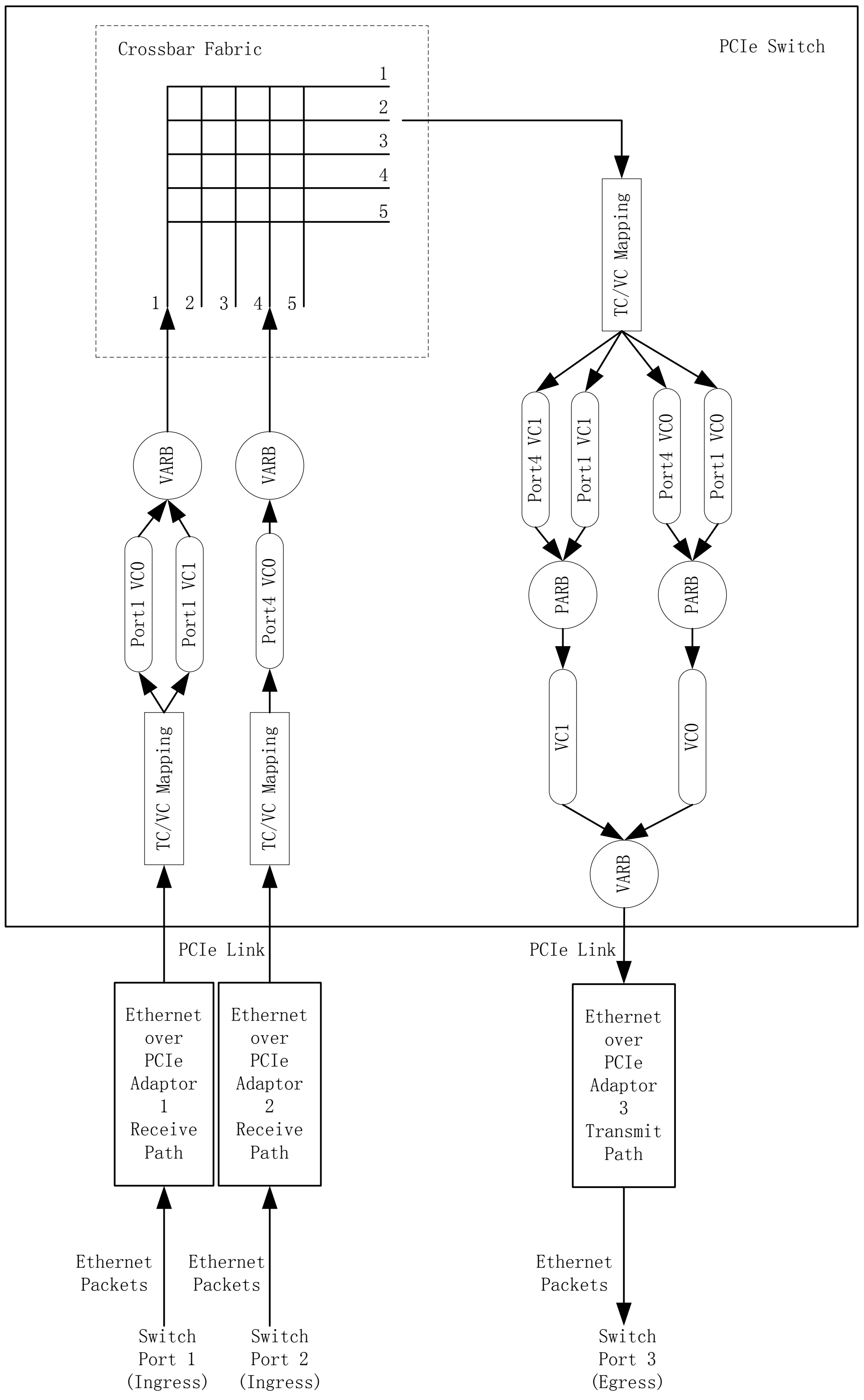


Figure 10: Classification in the PCIe Switch

3.3.3. Scheduling

Scheduling solves the issue of traffic contention, which is caused by packets from multiple ingress ports or channels aggregating to one egress port or channel. The scheduling mechanism in the PCIe-based Ethernet switch is implemented by Virtual Channel (VC) arbitration and port arbitration protocol in PCI Express.

VC arbitration determines the ordering of TLPs being transmitted from the same port. The VARB logic in Figure 10 illustrates an example of VC arbitration. In general, three methods can be used for VC arbitration.

a) Strict Priority

In this method, the priority of a TLP is uniquely determined by its VC ID. VC7 has the highest priority while VC0 has the lowest priority. Strict Priority requires that high priority VCs get precedence over low priority VCs. For example, if VC2 up to VC7 are used for strict priority method, TLPs in VC2 cannot be transmitted if any TLPs with VC3 to VC7 are pending. One problem of this method is that in some extreme cases, low priority TLPs may suffer from huge latencies due to sustained high throughput of high priority TLPs. In order to solve this problem, PCI Express requires that high priority TLPs must be regulated to avoid the starvation of low priority TLPs.

b) Split Priority

This method splits the VCs into two groups by the “VC Capability Register 1”. The high priority group keeps applying Strict Priority method while the low priority group uses alternative priority method determined by software. Either Equal Round Robin or Weighted Round Robin method can be used for the low priority group. This method avoids low priority traffic from starvation and provides more flexibility for the scheduling scheme.

c) Equal or Weighted Round Robin

This method is applied by setting the highest priority VC within the low priority group in the Split Priority method. The Equal Round Robin method gives every VC the same possibility to be transmitted, which is purely hardware-based. The Weighed Round Robin method involves a VC Arbitration Table. System software is able to configure the table by adding more entries for higher priority VCs than lower priority VCs. The VC arbiter then reads all table entries sequentially and sends out the TLP with the corresponding VC in the entry. Thus, high priority TLPs are more likely to be transmitted.

In the Ethernet over PCIe implementation, the Strict Priority method could be applied when the network is operating in relatively low throughput or with small amount of burst traffic. The Split Priority method could be applied when the throughput of certain types of traffic have to be guaranteed. The Weighted Round Robin method could be applied when priority rules are flexible and traffic types are unpredictable.

3.3.4. Flow Control

In Ethernet switching, input ports may send high throughput of data to one or more output port(s) resulting in data congestion or packet loss at the output. This problem is very significant when multiple input ports are directing large amounts of data to an output port at the same time. However, simply blocking the sending port will cause head-of-line blocking. The input buffer will overflow and the overall performance will be dramatically deceased. Hence, a proper flow control mechanism should be implemented to improve or prevent this type of issue. Since PCI Express switching fabric is leveraged for Ethernet frame switching, the flow control mechanism in PCI Express is implemented in the switch.

A credit-based flow control mechanism is used in PCI Express. In PCIe peer-to-peer switching, a peer can send a transaction packet to the other peer before the transmitter verifies the receiver has enough buffer space to store the packet. The available buffer space is called Flow Control Credits (FCCs). After initialization, the receiver starts to report the size of the receive buffer to the opposite link. The receiver should continue updating the available space in the buffer (FCCs) afterwards. Before transmission, the transmitter checks the FCCs of the data buffer in the target receiver. If a limited FCC is reported, the transmitter suspends the transmission. As soon as the receiver side reports enough FCCs, the transmitter is allowed to transmit again. Upon receipt of the transaction packets, the receiver removes the data in the buffer at a proper time. After the data is removed, the receiver adds more FCCs based on the amount of removed data. Updated FCCs are reported to the opposite side of the link.

In PCI Express, the head-of-line blocking problem is solved by using different flow control buffers for different virtual channels (VCs). If one of the receiving buffers is full and starts to report limited FCCs, the corresponding VCs at transmit side is blocked. Transaction packets in other VCs are still able to be sent out to the receivers that are reporting sufficient FCCs. Besides transmit buffers, each receiving port implements different data buffers for different VCs as well. Thus, a VC that is full on packets does not affect the transmission of other VCs. The overall possibility of overflow is decreased. As stated in 3.3.1, there are eight VCs per link in PCI Express. In addition, flow control is managed separately by six different kinds of transaction types. Therefore, there are 8 VCs x 6 types = 48 data buffers per port. Multiple data buffers ensure that the report of limited FCCs doesn’t affect much on the whole transmission.

3.3.5. Error Detection

Error detection is necessary due to unreliable transmission channels, logic block malfunctions, data source corruptions, etc. The PCI Express switch must be able to detect errors generated from Ethernet over PCIe adaptors, PCIe transmission channels, and inside the PCIe switch. Ethernet over PCIe adaptors must be able to detect errors from host devices, Ethernet transmission channels, PCI Express switches, and PCIe transmission channels. PCI Express switching utilizes the error detection mechanisms from the PCI Express Specification. The mechanisms that are related to the proposed Ethernet switch are discussed below.

The first type of error is ECRC error. This type of error can be generated by PCIe devices or PCIe transmission channels. The Ethernet over PCIe adaptor must generate a 32-bit cyclic redundancy check (CRC) that covers the header and data portions of the transaction layer packet (TLP) and attach the CRC to the end of the TLP. This type of CRC is called end-to-end CRC (ECRC), which is typically checked by the ultimate recipient of the transaction. The PCIe switch between two adaptors may optionally check and report ECRC errors. The switch must route and forward the packets intact even though errors have been detected in TLPs. If an adaptor detects an ECRC error in a request TLP, the adaptor may simply drop the packet without forwarding to the receiving logic. This will result in completion time-out in the requesting adaptor at the end. The requesting adaptor can resend this request at a certain time depending on the specific implementation. If an adaptor detects an ECRC error in a completion TLP, the adaptor may drop the packet and report the error to the adaptor driver. The driver could request the adaptor to retransmit the packet at a certain time depending on the specific implementation.

The second type of error is TC to VC mapping error. As mentioned in 3.3.1, the transmitter should consult with the receiver to decide the number of VCs which can be used for transmitting different classes of packets. Errors may happen in the meantime resulting in the receiver receiving packets that request non-supported VC numbers. A TC to VC mapping error could be reported to the device driver or OS.

The third type of error is flow control error. This type of error typically occurs when the flow control credit is reported incorrectly. For example, the specification defines the minimal flow control credit size that a VC should report after initialization. If the first flow control credit that a receiver receives is smaller than that number, a flow control error must be reported to the root complex.

3.3.6. Device Synchronization

Device drivers and system software should have full control of all the endpoint functions when data switching occurs. The system software should be aware of the switching status while all the traffic bypasses the CPU.

Specifically, the system software should be able to stop outstanding transactions for particular devices when necessary. When hot-plug occurs, BDF numbers may be changed when renumbering devices. Requests or completions from one peer still in flight may be redirected to a wrong destination. Thus, a stop mechanism should be implemented in device drivers and system software. The stop mechanism contains the following aspects.

a) Block requests generation.

The system software is able to block devices from generating new requests through drivers. The command register in each device function controls requests generation. When blocking, system software should send configuration write transactions to devices to change the command register. In particular, Bus Master Enable bit which controls the ability of the endpoint device to issue memory and I/O read/write requests should be set to 0; SERR# Enable bit which enables reporting of non-fatal and fatal errors should be set to 0; Interrupt Disable bit which controls endpoint function to generate INTx interrupts should be set to 1.

b) Block requests propagation.

Each request from peer A to peer B should go through the driver with the permission of the system software. When stopping, the system software should issue a “stop notice” to a group of device drivers. The drivers should then block the requests of their devices from propagating to the destination peers until the system software issues a “resume notice”.

c) Determine requests reached destinations.

For non-posted requests (requests requiring the completer to send back completion packets), the Transaction Pending (TP) bit in the device Status Register indicates whether the request has been completed. Endpoint functions set the TP bit to 1 after sending the request. The TP bit is set back to 0 when the requests are completed or terminated by system software.

For posted requests (requests not requiring the completer to send back completion packets), the system software should issue a “flush notice” to all endpoint devices which have sent a posted request in the last transaction. The devices should then send a non-posted read request with zero-length to the corresponding Traffic Class (TC). Consequently, determination of the read request is handed to TP bit.

CHAPTER 4

ETHERNET OVER PCI EXPRESS ADAPTATION

4.1. Overview of the Ethernet over PCI Express Adaptor Card

The Ethernet over PCI Express adaptor card implements Ethernet over PCI Express adaptation. The block diagram of the adaptor is shown in Figure 11. Ethernet packets pass through a receive path when the PCIe-based Ethernet switch receives incoming packets. The receive path consists of an Ethernet Physical (PHY) layer, an Ethernet Media Access Control (MAC) layer, Receive Path Adaptation logic blocks, a PCI Express Transaction layer, a PCI Express Data Link layer, and a PCI Express Physical (PHY) layer. Ethernet packets pass through a transmit path when the switch transmits outgoing packets. The transmit path is the reverse of the receive path, except that the packets pass through “transmit path adaptation” logic blocks instead of the “receive path adaptation” logic blocks.

The different layers of the Ethernet and PCI Express logic blocks implement encapsulation and decapsulation, encoding and decoding, collision detection and handling, error detection and handling, flow control, classification, etc. Receive and transmit path adaptation logic blocks implement output address lookup, classification, flow control, and error detection in cooperation with the layered logic blocks.

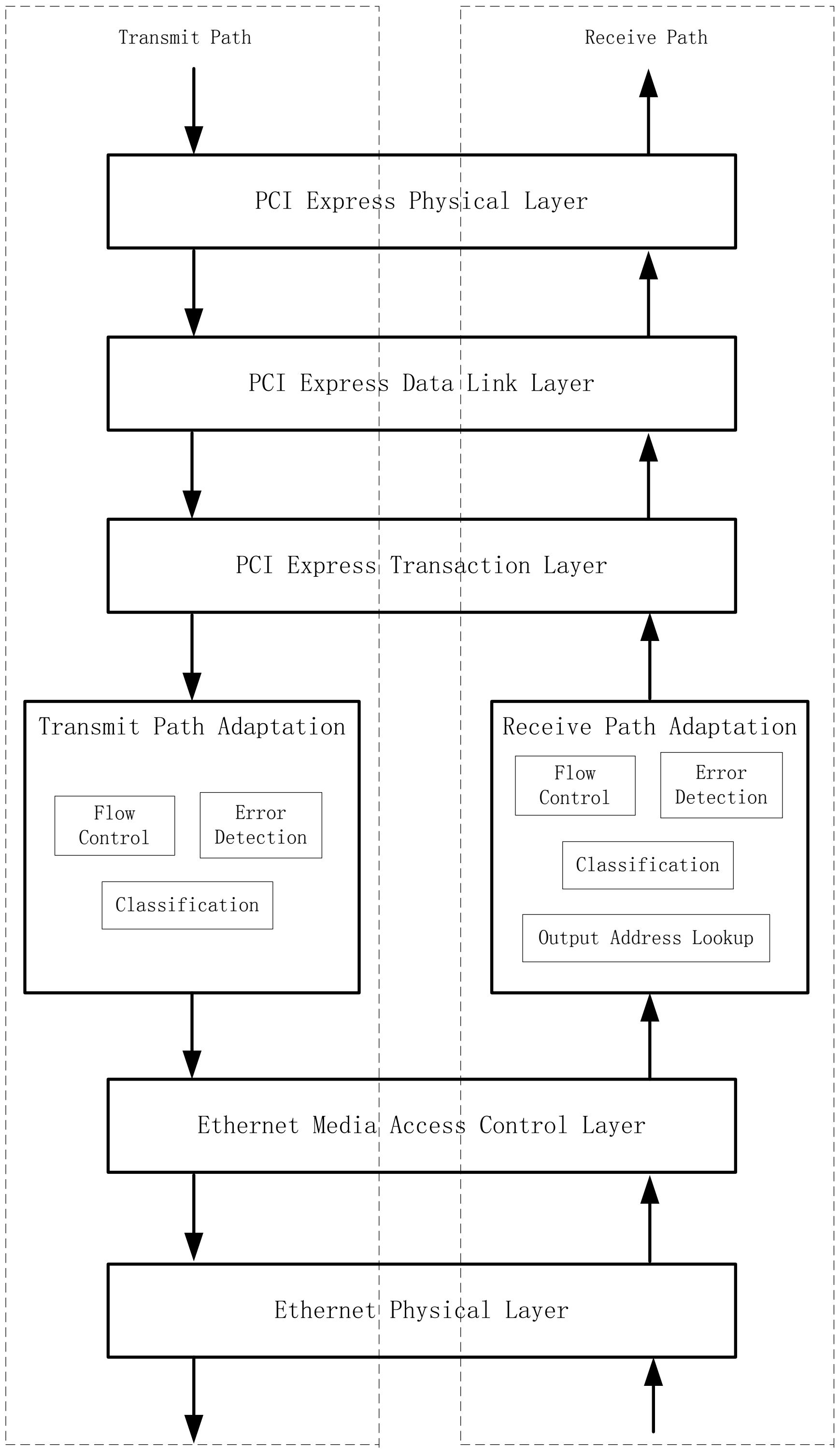


Figure 11: Overview of the Ethernet over PCI Express Adaptor Card

4.2. Encapsulation and Decapsulation

Figure 12 shows the encapsulation and decapsulation processes by the two Ethernet layers. When the Ethernet over PCIe adaptor receives Ethernet packets, the Ethernet PHY layer logic and the MAC layer logic decapsulate the Ethernet packets and send the data payload to the upper layers. When the adaptor transmits data, the MAC and PHY layer logics encapsulate the data payload from the upper layers to form Ethernet packets and send the packets to the Ethernet physical medium.

Assume the Ethernet PHY layer adopts 10GBASE-T. 10GBASE-T as a standard defined in IEEE 802.3an-2006 provides 10 Gb/s connections over unshielded or shielded twisted pair cables [29]. Ethernet signals transmit over four pairs of balanced cabling. Each of the twisted pair has a data rate of 2500 Mb/s resulting in an aggregation rate of 10 Gb/s. The physical signals use 16-level Pulse-amplitude Modulation (PAM) with a modulation rate of 800 Megasymbols per second.

XGMII and MDIO signals couple between the PHY and MAC layers. XGMII provides independent 32-bit-wide transmit and receive data paths. XGMII data stream is a sequence of bytes consisting of a preamble, a start of frame delimiter (SFD), a data payload, and an end of frame delimiter (EFD). MDIO is a bidirectional signal used for transferring control information and status between the PHY layer and Station Management Entity (STA) in the MAC layer. The structure of the MDIO frame is not drawn in Figure 12.

The MAC layer applies a media access discipline known as Carrier Sense Multiple Access with Collision Detection (CSMA/CD). The MAC layer also encapsulates and decapsulates Ethernet frames with a frame boundary delimitation, source and destination addresses, and an error detection field. A MAC frame consists of a Destination Address (DA), a Source Address (SA), a Length/Type field, MAC client data, a Pad field (if the frame is smaller than the minimum length), and a Frame Check Sequence (FCS) field.

When the adaptor receives Ethernet packets, the PHY layer logic decodes and converts 16-PAM physical signals to XGMII packets. The MAC layer logic then decapsulates the XGMII packets to MAC frames and sends the MAC frames to the upper layer logics. When the adaptor transmits Ethernet packets, the MAC layer logic encapsulates MAC frames to XGMII packets. The PHY layer logic then encodes and converts the XGMII packets to 16-PAM signals.

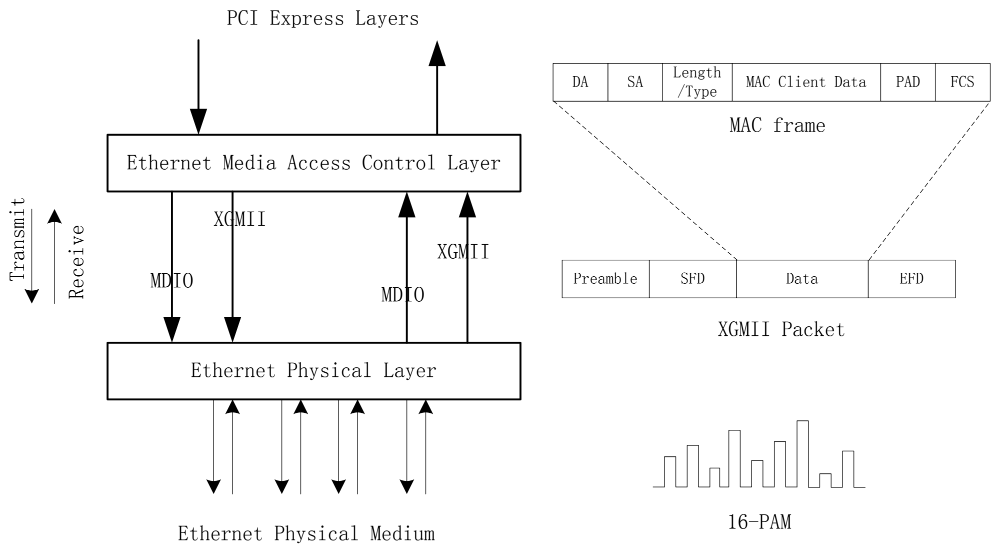


Figure 12: Ethernet Packets Encapsulation and Decapsulation

Figure 13 shows the encapsulation and decapsulation processes by the PCI Express layers. When the adaptor receives Ethernet packets, the PCI Express three-layer logics receive frames from the “receive path adaptation” logic along with PCIe routing addresses/IDs and other information. The three-layer logics encapsulate the Ethernet frames to form PCIe physical packets and send the packets to the PCIe physical medium. When the adaptor transmits data, the PCIe three-layer logics receive the PCIe packets from the PCIe physical medium. The logics then decapsulate the packets to form Ethernet frames and send the frames to the “transmit path adaptation” logics.

The PCIe transaction layer logic is responsible for encapsulation of outbound Ethernet MAC frames to form TLP packets and decapsulation of inbound TLP packets to form Ethernet MAC frames. Note that the “outbound” and “inbound” are defined in terms of PCIe packets transmitting to or from the PCIe physical medium. When encapsulating a MAC frame, the transaction layer appends a header and an optional ECRC field to the frame. The header is 3 double words or 4 double words in size and may contain information such as routing address/ID, TLP type, transfer size, tag, traffic class, byte enables, and attributes. The optional ECRC field is used for error detection, as discussed in 3.3.5. When decapsulating a MAC frame, the transaction layer checks the ECRC field and removes the header and ECRC field. The transaction layer also implements Virtual Channels (VCs) buffers and a flow control mechanism.

The PCIe data link layer logic is responsible for encapsulation of outbound TLPs to form link packets and decapsulation of inbound link packets to form TLPs. When encapsulating a TLP, the data link layer appends a sequence ID and a Link CRC (LCRC) to the TLP. The sequence ID field is used for the ACK/NAK protocol discussed in 3.3.1. The LCRC field covers the sequence ID and TLP field for the peer data link layer to check for any CRC errors. When decapsulating a TLP, the data link layer logic checks the LCRC field and removes the sequence ID and LCRC field. The data link layer logic is also responsible for generating Data Link Layer Packets (DLLPs). DLLPs are used for link management functions such as ACK/NAK protocol, power management, and exchanging of flow control information.

The PCIe physical layer logic is responsible for the encapsulation of outbound link packets and DLLPs to form physical packets and the decapsulation of inbound physical packets to form link packets and DLLPs. When encapsulating a link packet or DLLP, the physical layer appends the packet with a “start” and “end” character with the aid of a multiplexer. The characters are framing symbols used for detecting the start and end of the packet. The packet is then stripped to different lanes, scrambled, encoded, converted to serial bit stream, and finally transmitted to the PCIe physical medium. When decapsulating, the physical layer logic converts serial bit streams to parallel symbol streams, decodes, de-scrambles, and un-strips the streams to form physical layer packets. Start and end characters are then removed from the physical packets to form link packets or DLLPs. The link packets or DLLPs are handed to the data link layer.

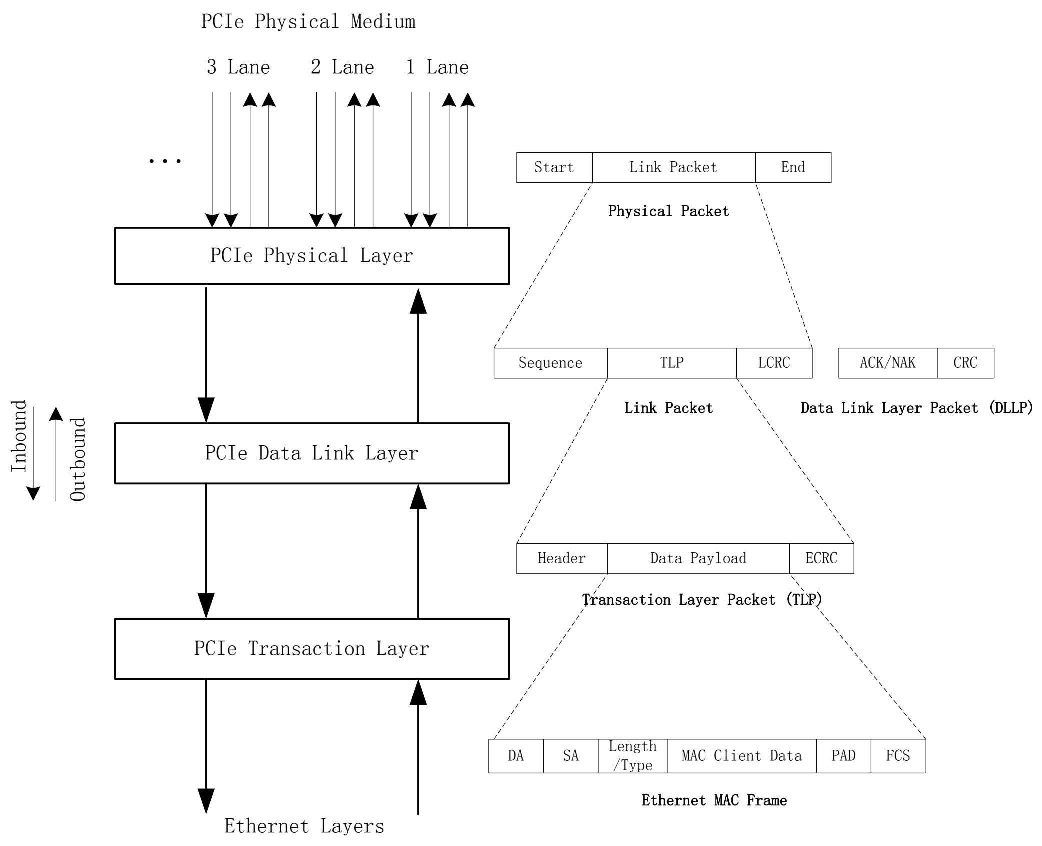


Figure 13: PCI Express Packets Encapsulation and Decapsulation

4.3. Address Mapping Table

Ethernet packets use Media Access Control (MAC) addresses to indicate destination devices. PCI Express TLPs use routing addresses or routing IDs to indicate destination endpoint devices (adaptors). A manageable mapping table between the MAC addresses in Ethernet packets and the routing addresses/IDs in PCIe TLPs is a must for Ethernet over PCIe adaptation.

The routing addresses of PCI Express endpoint devices have different representations in the PCIe system. In a system memory map, the endpoint devices can be mapped into prefetchable memory devices or memory-mapped IO (MMIO) devices. Prefetchable memory refers to the memory that does not have any side effects from read. The MMIO devices use regular memory space in the system to access IO devices. A legacy PCIe device such as a PCI or PCI-X device should be mapped into a system IO map, which is separate from the system memory map. In this case, the Ethernet over PCIe adaptor must be mapped to a prefetchable memory device or MMIO device. The adaptor can use either 32-bit or 64-bit addresses, depending on the PCI Express system.

The Operating System (OS) is responsible for creating the address mapping table. When the system is powered on, certain routines in the Basic Input/Output System (BIOS) perform the enumeration process. After the enumeration, Bus, Device, Function (BDF) numbers are assigned and stored in the OS. Either the memory mapping address, IO mapping address or pre-fetch address of each device is also stored in the OS [15].

When the adaptor receives Ethernet packets, the source MAC addresses of the Ethernet packets indicate the addresses of the transmitter which the packets came from. The Ethernet packets are analyzed by an “Ethernet parser” logic in the adaptor card. The logic sends the source MAC addresses to the operating system with the adaptor’s BDF number (ID) or routing address. When the OS receives the information, it creates an address mapping table with the mapping of source MAC addresses, BDF numbers, and routing addresses.

An example address mapping table is shown in Table 1. According to the PCI Express Specification Revision 3.0 [6], the maximum number of buses in a PCIe system is 256; the maximum number of devices on a bus is 32; the maximum number of functions in a device is 8. Hence, an 8-bit Bus field, a 5-bit device field, and a 3-bit function field occupy the first 16 bits of the entries in the mapping table. A 64-bit routing address field is also reserved in the table. If a 32-bit routing address is applied in the PCIe system, only the lower 32 bits of the field can be accessed.

Table 1: Address Mapping Table

[131]	[130]	[129:128]	[127:80]	[79:17]	[47:16]	[15:8]	[7:3]	[2:0]
Entry No.	Valid (1 bit)	Age (2 bits)	Source MAC Address (48 bits)	Routing Address (upper 32 bits)	Routing Address (lower 32 bits)	Bus # (8 bits)	Device # (5 bits)	Func tion # (3 bits)

A 3-bit status field is attached to each entry. The field consists of a 2-bit age field and a 1-bit valid field. According to the IEEE 802.1D standard, entries that are not being accessed for a long time should be aged out [16]. Thus, a 2-bit age field is appended to each entry to indicate the time that the entry existed. The age field is reset to “00” when a new entry is created or an existing entry is received. The age field is set to “01” if the entry has not been received for a certain amount of time. The age field is set to “10” if the entry has timed out and can be rewritten. Age “11” denotes the entry should be kept perpetually. The system software then should implement a timer to increase the age field up to “10”. The valid bit indicates whether this entry should be kept or rewritten. For example, if “10” is detected in an age field, then the valid bit should be set to “0”, indicating the entry is invalid and can be rewritten when a new entry is received.

The OS is responsible for management of the mapping table such as updating, adding and deleting entries. When a mapping entry A is received, the OS first searches the table for the source MAC address in entry A. If an existing entry B matches the source address, the OS then should rewrite the BDF field and routing address field with the source address in entry A. If no result is found, the OS must allocate an address with valid bit “0” and write the new entry to that address.

4.4. Output Address Lookup

After proper processing, every Ethernet packet must be directed to the target PCI Express endpoint adaptor according to the destination MAC address. The “output address lookup” logic in the adaptor is responsible for looking up the routing ID or address of the target PCIe adaptor based on the destination MAC address using the Address Mapping Table. The basic structure of the lookup logic is shown in Figure 14.

Each PCIe-based switch adaptor should implement the Output Address Lookup logic. The operating system is responsible for sending control signals periodically to the adaptors. The control signals include updating, adding, and deleting entries of the address mapping table. The lookup logic must create and maintain an address mapping table identical to the table in the operating system. When the adaptor receives Ethernet packets, the Ethernet parser sends the destination MAC addresses of the packets to the Output Address Lookup logic. The lookup logic looks up the corresponding routing address/ID of the target adaptor. The lookup result is sent to upper layer logic in order to generate PCI Express transactions.

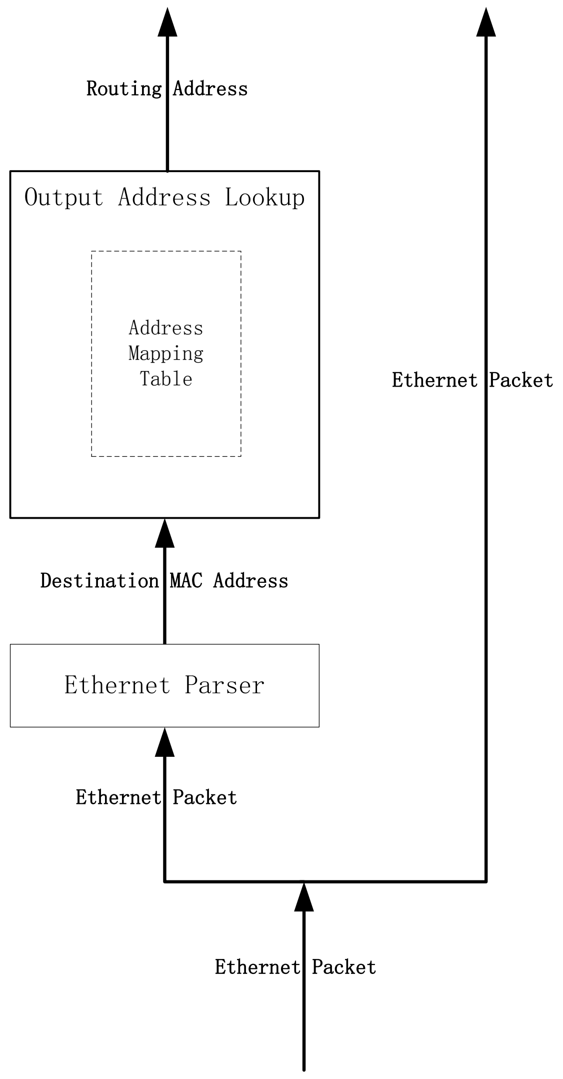


Figure 14: Output Address Lookup Logic

There are different approaches of the mapping table storage and address lookup.

The cache-based approach uses multiple caches to store a small forwarding sub-table containing only the most recent or frequent destination addresses. Other less frequently used addresses are stored in a slower but larger memory. The destination addresses of incoming packets are always searched inside the cache first. If no matching entry is found, the full mapping table is accessed. This approach can only work when the destination addresses in the Ethernet packets are not constantly changing, which will yield a high cache-hit-ratio. However, the approach becomes inefficient in networks of unpredictable and dynamic traffic [10][17].

The trie-based approach uses a multi-way tree structure to handle address strings. Each node in the tree contains zero or more pointers to its child nodes. Each address lookup process starts from the root node of the tree. For a binary tree, whether the left node or the right node is visited is based on the value of each bit in the address string. The furthest node that the address can reach returns the corresponding target address information. The trie-based approach is mainly used in the "longest prefix matching" for IP packets [18]. In Ethernet switches, the approach can still be used. The difference is that each MAC address has only one corresponding PCIe port ID/address. The node at which the searching stops is always at a leaf node and not a parent node. It is a tradeoff between searching time and storage complexity. Searching time is reduced when the destination addresses are randomly distributed, however, a wider tree needs to be built resulting in a storage capacity increase. On the other hand, concentrated ranges of addresses reduce the overall storage complexity, but the amount of memory accesses is increased when the addresses are similar (long prefixes are matched).

The hashing table approach provides a compact and well-organized way to store the address mapping table as well as an efficient way to lookup the corresponding PCIe routing IDs/addresses. Hashing is a method of transforming a set of data into preferable unique memory addresses for data storage [19]. Hashing uses the MAC address as the memory address occupies extensive memory space. The hashing function converts the source MAC address to a memory address or pointer for storing the MAC address associated with the PCIe port ID/address. However, the hashing approach introduces hash collisions, which may cause decreased address table capacity and inefficient bandwidth utilization. During the design of the switch, the hashing function needs to be carefully chosen to accommodate address distribution in a real network in order to reduce hash collisions [20].

Another approach is Content-Addressable Memory (CAM). CAM is a specialized matching memory that performs parallel comparison. MAC addresses are stored in the data field instead of the address field of the CAM. The corresponding PCIe routing ID/address is stored in the address field. When a destination MAC address is received, the CAM outputs the address field (i.e. PCIe routing ID/address) of the MAC address. The CAM approach is simpler than the hashing table and trie-based approaches. CAM has the disadvantage of high cost-to-density ratio and high power-consumption [18].

The selection among these different approaches depends on the specific application and network traffic condition. Some performance metrics [21] discussed below also need to be considered.

a) Lookup Speed: The lookup speed needs to meet the increasing demand of high bandwidth Ethernet communication. For a 10 Gbps Ethernet channel, the lookup logic has to perform destination ID/address lookup at 31.25 million times per second, when considering a minimum size of a 40-byte packet.
b) Storage Requirement: Randomly distributed Ethernet packets require large amount of storage. Proper management and organization is necessary to maintain the mapping table as concise as possible, leading to fast memory access and low power consumption.
c) Update Time: The update time is very critical when the Ethernet switch is located in a large LAN with constantly changing hosts and variable Ethernet packets. Certain updating algorithms need to be performed to avoid routing instabilities while interfering little with normal lookup operations.

d) Scalability: It is expected that the size of forwarding tables will increase at a speed of 25k entries per year. The ability of an algorithm to handle large forwarding tables is required.

e) Flexibility: Most current lookup algorithms can be implemented in either software or hardware. Some of the algorithms have the flexibility of being implemented in different ways such as ASIC, network processors, or generic processors.

4.5. Packets Routing

Before the switch routes packets, Base Address Registers (BARs) must be set specifically in the Ethernet over PCIe adaptors. BARs are used to indicate the type and start address of the device when using address routing. Figure 15 shows the bit assignment of a 64-bit BAR. Whether the memory address or IO address is stored in the BAR depends on the “memory space indicator” bit. IO transactions are only allowed for legacy devices and software (e.g. PCI and PCI-X). Hence, the “memory space indicator” bit should be set to “0” in the BAR to indicate the adaptor is a memory device. The “prefetchable attribute” bit defines the memory as prefetchable or not. The PCI Express Specification encourages that resources mapped into memory space should be designed as prefetchable whenever possible. PCI Express endpoints other than legacy endpoints must support 64-bit addressing for any BAR that requests a prefetchable memory space. In the case of the Ethernet over PCIe adaptor, the prefetchable attribute bit is set to “1”; and a 64-bit decoder is used. The range of the memory space address could be set to a minimum of 128 bits.

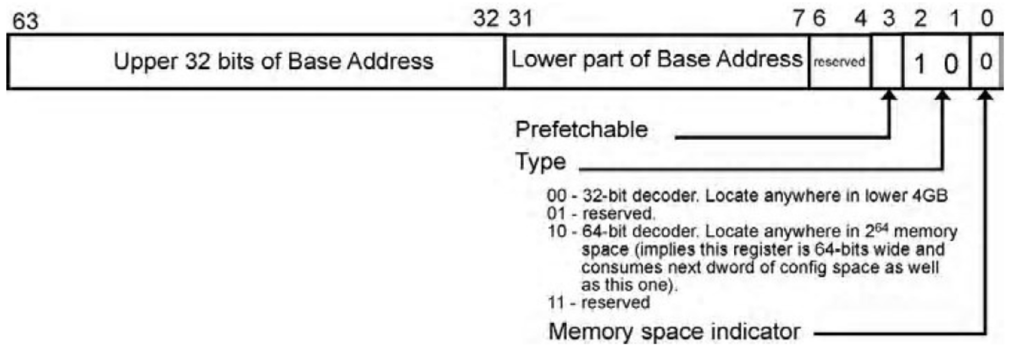

63
32 31
7 6
4 3 2 1 0
Upper 32 bits of Base Address
Lower part of Base Address
reserved
1 0
0
Prefetchable
Type
00 - 32-bit decoder. Locate anywhere in lower 4GB
01 - reserved.
10 - 64-bit decoder. Locate anywhere in 2^64 memory space (implies this register is 64-bits wide and consumes next dword of config space as well as this one).
11 - reserved
Memory space indicator

Figure 15: 64-bit Memory Base Address Register Bit Assignment [12]

As stated in 3.2.2, there are three routing schemes in PCI Express: Address Routing, ID Routing, and Implicit Routing. In the case of Ethernet over PCI Express, Ethernet packets transmission from transmitter A to receiver B can be treated as memory write from peer A to peer B in PCI Express. Ethernet packets reception from transmitter B to a receiver A can be treated as memory write from peer B to peer A. Hence, the Address Routing scheme is adopted for Ethernet packet switching since only the memory write (MWr) transaction is needed. ID Routing and Implicit Routing are only necessary for the native configurations in the PCI Express system.

The memory write routing procedure is simpler than the example discussed in 3.2.2. The transaction layer logic in an adaptor (Requester) first encapsulates an Ethernet frame with a header including the Completer’s routing address, MWr transaction type, payload size, traffic class, etc. The logic also appends an ECRC field covering the header and payload at the end of the frame. The TLP is then encapsulated in the data link layer logic and buffered in the TLP replay buffer. When the link packet is transmitted through the physical layer logic and received by the switch downstream port correctly, the switch downstream port returns an ACK DLLP to the Requester. The Requester discards the copy of the TLP from the replay buffer. Next, the switch forwards the MWr TLP to the correct egress port using the routing address. If the Completer receives the TLP without an error, it returns an ACK DLLP to the switch; and the switch discards the copy of the TLP in the replay buffer. MWr requests are posted transactions, in which Completers do not return completion packets.

4.6. QoS and QoS-related Protocols

Similar to the QoS of PCI Express as discussed in 3.3, the QoS in the context of Ethernet over PCIe adaptation refers to providing an Ethernet connection of differentiated services for the hosts connected to the adaptors as well as coordinating the QoS provided inside the PCIe switching fabric. The QoS protocol (classification) and QoS-related protocols (flow control and error detection) are discussed below.

4.6.1. Classification

Basic Ethernet packets do not contain any classification information. Solutions have been proposed in order to classify different sort of packets. For example, one solution offers higher priority for smaller frames to adapt inter-frame gap. Another solution uses a variable length of preamble field to represent different priorities [22].

The IEEE 802.1Q standard [23] defines Virtual Local Area Network (VLAN), which enables a group of hosts to communicate as if they were attached to the same broadcast domain. At the same time, the standard adds a 4-byte tag field into the basic Ethernet frame. The tag field enables priority information to be conveyed within the Ethernet frame. The tag field also allows VLAN Identifier (VLAN ID) to be conveyed with the VLAN classification information throughout the network.

The structure of a tagged frame is shown in Table 2. A Tag Protocol Identifier (TPID) field and a Tag Control Information (TCI) field are added to the basic Ethernet frame.

Table 2: Structure of Tagged Frame

No. of Bytes	7	1	6	6	2	2	2	42-1496	4
Field	Preamble	SFD	DA	SA	TPID	TCI	Type Length	Data	CRC

An EtherType value is included in the TPID field, which is used for identifying the frame as a tagged frame and selecting the corresponding tag decoding function. A regular EtherType value is 8100 in hexadecimal representing “Customer VLAN Tag” type. The TCI field contains 16-bit control information as shown in Table 3.

Table 3: Structure of TCI Field

No. of Bits	3	1	12
Field	PCP	DEI	VID

Priority and Drop Eligible parameters are encoded in the Priority Code Point (PCP) field and Drop Eligible Indicator (DEI) field, respectively. The DEI field along with the PCP field specifies the priority level and the number of bits that can be dropped. Up to 8 levels of priorities can be selected by using the PCP field. The highest priority level is 7, which can be used for network-critical traffic such as the Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) table updates. Level 5 or 6 can be used for delay-sensitive applications such as video and voice transmission. Levels 1 to 4 can be used for controlled-load applications such as business-critical traffic. Level 0 is left to the default best-effort traffic [24]. VID is the identification of VLAN, which contains 4096 possible VIDs.

The PCI Express-based switch adaptor is responsible for converting different classes of Ethernet packets to Traffic Classes (TCs) in PCI Express. As Figure 16 shows, when an Ethernet packet is received, the Ethernet Parser logic determines whether the packet is VLAN tagged or not. In other words, the switch interface should be VLAN-aware. If the packet is VLAN tagged, the parser logic must extract the Priority Code Point (PCP) field from the packet, decode the level of priority, and convert the priority level to the corresponding TC number in PCI Express. Next, the TC number is sent to the PCIe Transmit Interface logic. The logic adds the TC number into the TLP header. On the other hand, if the packet is not VLAN tagged, the parser logic must assign the packet with TC0 and send the packet to the interface logic. The interface logic then adds TC0 to the TLP header. Obviously, this type of Ethernet packets is mapped to the VC0 in the PCI Express.

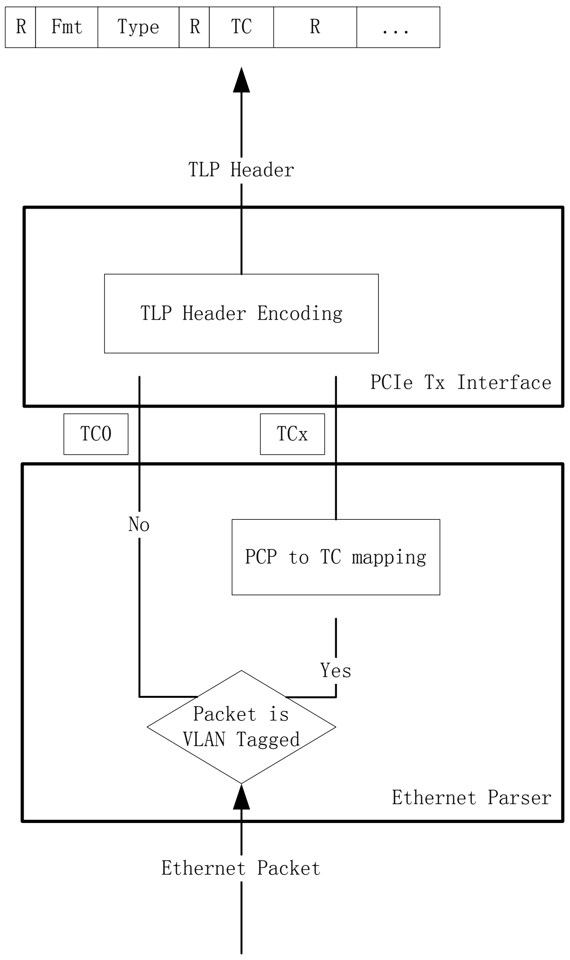


Figure 16: Ethernet Class Conversion

4.6.2. Flow Control

As stated in 3.3.4, the internal switch utilizes the PCI Express flow control mechanism. This mechanism ensures that packets are transmitted smoothly (without packet loss or buffer overflow) inside the switching fabric. However, a flow control mechanism must also be implemented outside of the switch. For example, if an input buffer of a host connected to an egress port of the switch is about to overflow, the host should report to the switch to slow down or pause transmission to avoid packet loss.

4.6.2.1. Conventional Flow Control

The IEEE 802.3x standard annex 31B [25] defines a MAC control PAUSE operation for basic flow control across Ethernet. This conventional flow control mechanism is still being used though the IEEE 802.3x standard has been withdrawn. The PAUSE operation is used for inhibiting transmission of Ethernet frames for a specified period of time to avoid buffer overflow. The receiver side normally uses a high and a low threshold to indicate receive buffer status. The transmitter side normally implements an "on and off" state machine to control data transmission. If the buffered data level reaches the high threshold, the receiver will send a PAUSE frame to the opposite side of the link. The PAUSE frame contains a timer parameter which ranges from 0 to a maximum pause time of 65535. The time unit is 512 bit time depending on the specific implementation. A globally-assigned multicast MAC address 01-80-C2-00-00-01 has been reserved for the PAUSE frame. After the transmitter receives the PAUSE frame, the state machine will be triggered to the "off" state. The transmitter then stops transmitting frames until the timer expires or receives another PAUSE frame with a zero timer. If the receiver consumes the buffered data and the buffer level reaches the low threshold, the receiver will send another PAUSE frame timer parameter equal to zero. The transmitter then turns to the "on" state and resumes transmission upon receiving the zero timer PAUSE frame.

4.6.2.2. Priority-based Dynamic Flow Control with Memory (PDFC)

The conventional flow control mechanism defined in the IEEE 802.3x standard can be applied to basic networks. However, the mechanism suffers from data congestion and starvation in unpredictable networks since only fixed PAUSE timers can be sent. New mechanisms have been put forward to solve this issue, such as Dynamic Pause Time Calculation (DPTC) [26]. The DPTC method calculates the pause time according to the network congestion to avoid buffer overflows and underflows. DPTC reduces the number of PAUSE frames needed to be sent and optimizes buffer utilization, but the head-of-line blocking issue still remains unsolved. The Priority Flow Control (PFC) protocol defined in the IEEE 802.1Qbb standard extends the basic flow control mechanism in the IEEE 802.3x standard to multiple Class of Services (CoSs) [27]. The PFC protocol implements a separate flow control method for different classes of data to solve the head-of-line blocking problem. However, the PFC protocol still uses the fix PAUSE frame timer resulting in the control of a particular class of queues not being optimized.

A Priority-based Dynamic Flow Control with Memory (PDFC) method is presented here. PDFC leverages different flow control mechanisms for multiple CoSs defined in the PFC as well as creates a new method of calculating pause time dynamically based on throughput and past events. The PDFC method is illustrated in Figure 17.

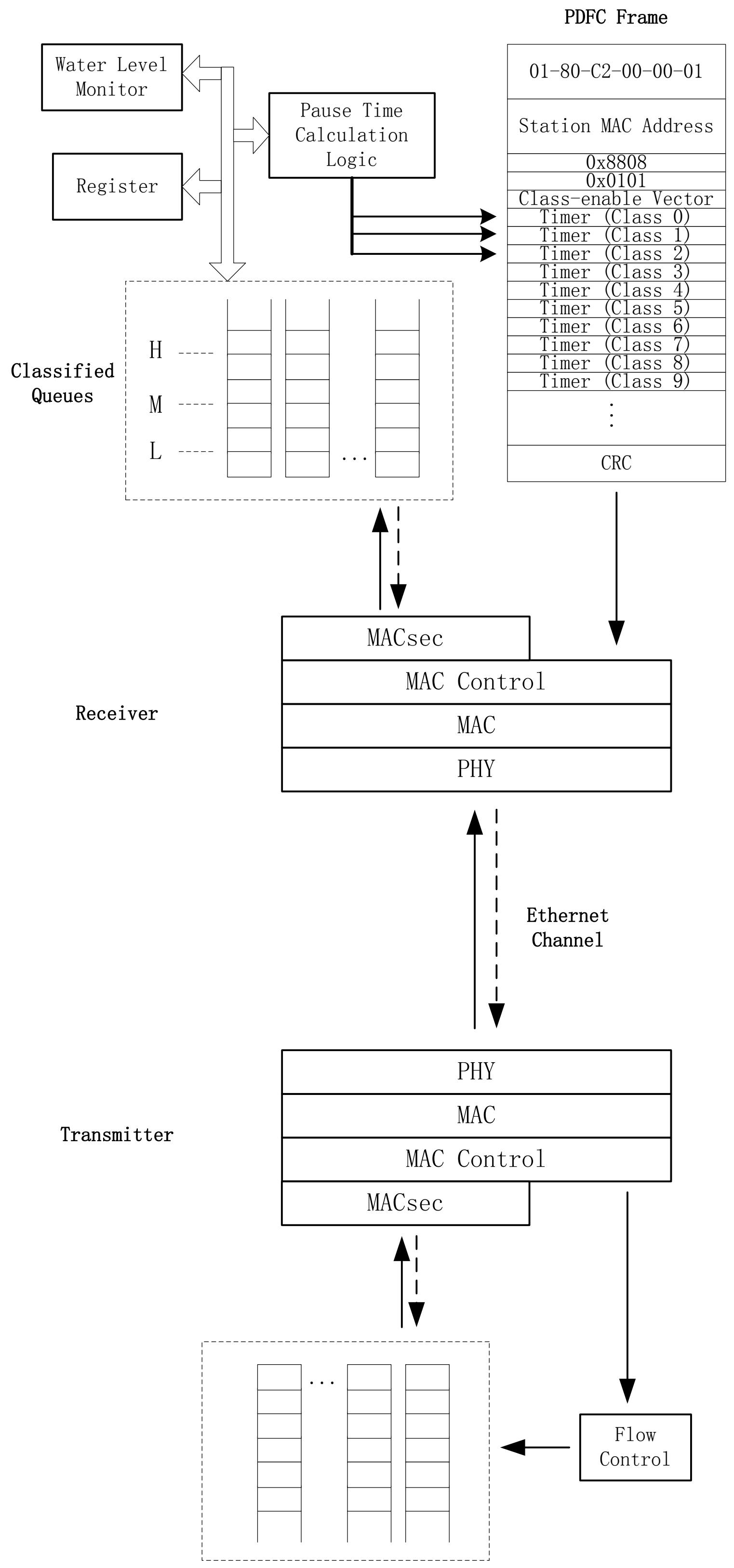


Figure 17: Priority-based Dynamic Flow Control with Memory

The PDFC method utilizes tagged frame (defined in 4.6.1) to prioritize different classes of frames. The Priority Code Point (PCP) field in the tagged frame contains the priority level of each frame. Each device has to implement classified queues for different priorities of frames. Each queue has three watermarks, High (H), Middle (M) and Low (L), which can be programmed by the operator. The water level monitor and the register are responsible for recording water levels (length of queue) and time. The pause time calculation logic calculates a proper pause time for each queue based on the recorded data in the monitor and the register. Similar to the PAUSE frame defined in the IEEE 802.3x standard, a PDFC frame contains different timers for different prioritized classes. Suppose the transmitter is sending Ethernet packets faster than the receiver can process during transmission. When one of the classified queues in the receiver reaches the middle or high watermark, the receiver generates a PDFC frame and sends the frame to the Ethernet channel. After the transmitter receives the frame, the transmitter extracts the timer field and suspends the transmission of that particular queue for a certain amount of time based on the value of the timer. The queue is allowed to resume the transmission when the timer expires.

The calculation method of PDFC evaluates an optimized pause time based on the changing rate of water levels and the past pause time. Let L_M and L_H denote water levels at the watermark middle and high, respectively. T_{M_last} denotes the pause time that was calculated last time when reaching the water level middle. T_{M_real} denotes the actual duration that the water level falls from middle to low. Similar denotations apply to T_{H_last} and T_{H_real}. The calculated pause time T_M when reaching water level middle is given by

$$T_M = 32768 \times F_1\left(\frac{dL_M}{dt}\right) \times F_2\left(\frac{T_{M\_real}}{T_{M\_last}}\right) \tag{1}$$

Where,

$$F_1\left(\frac{dL_M}{dt}\right) = \begin{cases} R_1\frac{dL_M}{dt} & ,when\ 0 \le R_1\frac{dL_M}{dt} < 1 \\ 1 & ,when\ R_1\frac{dL_M}{dt} \ge 1 \end{cases}$$

$$F_2\left(\frac{T_{M\_real}}{T_{M\_last}}\right) = \begin{cases} R_2\frac{T_{M\_real}}{T_{M\_last}} & ,when\ 0 \le R_2\frac{T_{M\_real}}{T_{M\_last}} < 1 \\ 1 & ,when\ R_2\frac{T_{M\_real}}{T_{M\_last}} \ge 1 \end{cases}$$

Term $F_1\left(\frac{dL_M}{dt}\right)$ is a factor to describe the increasing rate of the data in queue. The term is calculated by monitoring the water level increment around the middle level during a short period of time. The higher the ratio, the longer the pause time should be set to avoid potential overflow. R_1 is a parameter which can be tuned by the operator. The term $F_1\left(\frac{dL_M}{dt}\right)$ is limited within 1. Term $F_2\left(\frac{T_{M\_real}}{T_{M\_last}}\right)$ is a ratio to adjust the pause time T_M based on the pause time that was calculated last time and the actual time period that data falls from the middle to low water level. If T_{M_last} is larger than T_{M_real}, it indicates the pause time set last time was too long so that the queue was about to underflow. Thus, the ratio $\frac{T_{M\_real}}{T_{M\_last}}$ is less than 1; shorter pause time T_M is set. If T_{M_last} is smaller than T_{M_real}, it indicates the pause time set last time was too short so that the queue had potential of overflow. Thus, the ratio $\frac{T_{M\_real}}{T_{M\_last}}$ is more than 1; larger pause time T_M is set. R_2 is a parameter which can be tuned by the operator. The term $F_2\left(\frac{T_{M\_real}}{T_{M\_last}}\right)$ is also limited within 1. The coefficient of 32768 is set to be around the middle of the maximum pause time 65535.

Similar to the previous equation, the pause time when reaching water level high T_H is given by

$$T_H = 65535 \times F_3\left(\frac{dL_H}{dt}\right) \times F_4\left(\frac{T_{H\_real}}{T_{H\_last}}\right) \tag{2}$$

Where,

$$F_3\left(\frac{dL_H}{dt}\right) = \begin{cases} R_3\left(\frac{dL_H}{dt}\right)^2 & ,when\ 0 \leq R_3\left(\frac{dL_H}{dt}\right)^2 < 1 \\ 1 & ,when\ R_3\left(\frac{dL_H}{dt}\right)^2 \geq 1 \end{cases}$$

$$F_4\left(\frac{T_{H\_real}}{T_{H\_last}}\right) = \begin{cases} R_4\frac{T_{H\_real}}{T_{H\_last}} & ,when\ 0 \leq R_4\frac{T_{H\_real}}{T_{H\_last}} < 1 \\ 1 & ,when\ R_4\frac{T_{H\_real}}{T_{H\_last}} \geq 1 \end{cases}$$

The difference with T_H is that term $F_3\left(\frac{dL_H}{dt}\right)$ is proportional to $\left(\frac{dL_H}{dt}\right)^2$ instead of $\frac{dL_M}{dt}$. This is a punishment algorithm to increase the pause time dramatically when the data increasing rate is too high. Besides, the coefficient is set to be the maximum value 65535.

The PDFC method memorizes the pause time calculated last time and the actual draining time. The method can be improved by memorizing more records, then take an average of the past calculated pause time and the past actual draining time. This improved method can be used for networks with variable throughput.

The determination of the watermarks is also important, especially for the middle and high watermarks. The receiver must predict the potential extra packets it will receive after data reaches the middle or high watermark [28]. The PDFC frame has to be sent early enough for frame transmission and processing by the transmitter. The receiver has to reserve enough buffer space to ensure that the extra packets will not cause overflow. The determination of watermarks must consider the following aspects.

a) Transmission time of a Maximum Transmission Unit (MTU): When the receiver needs to send a PDFC frame, the receiver cannot interrupt the current transmitting frame. In the worst case scenario, a MTU will have just started transmitting the first bit when the PDFC frame needs to be sent. The receiver must wait until the MTU has completed its transmission. Similarly, the MTU transmission must be completed at the transmitter end after the transmitter decides to suspend transmission. The transmission time of MTU at both ends must be considered.

b) Transmission time in Ethernet channel: It takes some time for the PDFC frame to reach the transmitter after the frame is sent by the receiver. Certain amounts of packets are received by the receiver during that time. Similarly, after the transmitter suspends transmission, the packets in the channel should finally reach the receiver. Hence, the transmission time on both sides must be considered.

c) Transmitter respond time and transceiver latency: After the transmitter receives a PDFC frame, the processing time before the transmitter suspends transmission must be taken into account. In addition, both the transmitter and receiver suffer from transceiver latency. Extra packets are sent by the transmitter at the speed of wire during that time.

4.6.3. Error Detection

As discussed in 3.3.5, Ethernet over PCIe adaptors must be able to detect errors from host devices, Ethernet transmission channels, PCI Express switches, and PCIe transmission channels. The errors from the PCIe structure (including PCIe switches and transmission channels) are detected by the PCIe error detection mechanism. The errors from Ethernet structure (including host devices and Ethernet transmission channels) are detected by the Frame Check Sequence (FCS) field in the Ethernet frames.

The IEEE 802.3an [29] standard defines the format of Ethernet Media Access Control (MAC) frame. The FCS filed in the MAC frame consists of a 32-bit CRC, which covers all the fields in the MAC frame except the preamble, start of frame delimiter (SFD), and FCS.

When a host transmits Ethernet frames, the transmitter of the host generates a CRC based on the scheduled Ethernet frame. The frame is transmitted from the egress port of the host with the FCS field attached to the end. Errors may occur at the transceivers (both the host and switch) and the transmission channels. When the Ethernet over PCIe adaptor receives the frame, the adaptor calculates a CRC based on that frame. If the calculated CRC is the same with the FCS field in the frame, the adaptor will keep the frame and forward it to the PCIe switch. If the calculated CRC is different from the FCS field, the adaptor will drop the frame and report an error to the driver through PCI Express. The transmit path of the adaptor is not responsible for checking the FCS field in frames. However, it may need to generate a FCS field when new Ethernet frames are created.

CHAPTER 5

IMPLEMENTATION OF THE PCIE-BASED ETHERNET SWITCH ADAPTOR

5.1. Overview of the PCIe-based Ethernet Switch Adaptor Implementation

Basic Ethernet over PCI Express adaptation logics in the adaptor are implemented as shown in Figure 18. The Output Address Lookup logic, PCIe Transmit Interface logic, and PCIe Receive Interface logic are designed in Verilog HDL. Commercial IP cores including the 10G Ethernet MAC Core, PCIe Endpoint Core, PCIe PHY, and AXI FIFOs are customized. The 10G Ethernet PHY chip is included on the FPGA development board. The logic blocks in the adaptor are divided into three different clock domains: MAC Rx Clock Domain, PCIe User Clock Domain, and MAC Tx Clock Domain. The clock frequencies in the three clock domains are defined by standards. The adaptor is designed to be implemented in an FPGA platform. FPGA has a shorter and simpler development cycle compared to ASIC. The QoS features are not implemented in the adaptor at this time.

As discussed in Chapter 4, the 10G Ethernet PHY logic and 10G Ethernet MAC Core perform the functionality of the Ethernet Physical layer and Ethernet MAC layer respectively. The interface between the 10G MAC Core and 10G PHY is XGMII. The PCIe Endpoint Core performs the functionality of the PCIe Transaction Layer, PCIe Data Link Layer, and PCIe MAC Sub-layer (part of the PCIe Physical Layer). The PCIe PHY performs the functionality of Physical Coding Sub-layer (PCS) and Physical Media Attachment (PMA), which are the remaining parts of the PCIe Physical Layer. The interface between the PCIe Endpoint Core and PCIe PHY is called the PHY Interface for the PCI Express Architecture (PIPE) interface.

The MAC Transmit FIFO and MAC Receive FIFO are used for dividing the paths into three separate clock domains and passing data among the domains safely. The two MAC FIFOs and the PCIe Transmit FIFO are also used for data buffering and flow control. The Output Address Lookup logic contains an Ethernet Parser, a Content-Addressable Memory (CAM), and an array. The PCIe Transmit Interface counts the lengths of Ethernet frames sent out from the MAC Rx FIFO and then gathers the lengths, routing addresses, and other information obtained from the lookup logic to send to the PCIe Endpoint Core. The Ethernet frames are also sent to the PCIe Tx Interface by the PCIe Tx FIFO to provide the data payload of TLPs. The PCIe Rx Interface analyzes TLPs sent from the PCIe Endpoint Core and sends Ethernet frames to the MAC Tx FIFO. Note that the logic blocks in the PCIe User Clock Domain of the Receive Path are called “Transmit” logics. The logic blocks in the PCIe User Clock Domain of the Transmit Path are called “Receive” logics. The “Tx” and “Rx” are named in terms of the PCIe Endpoint Core rather than the 10G MAC Core.

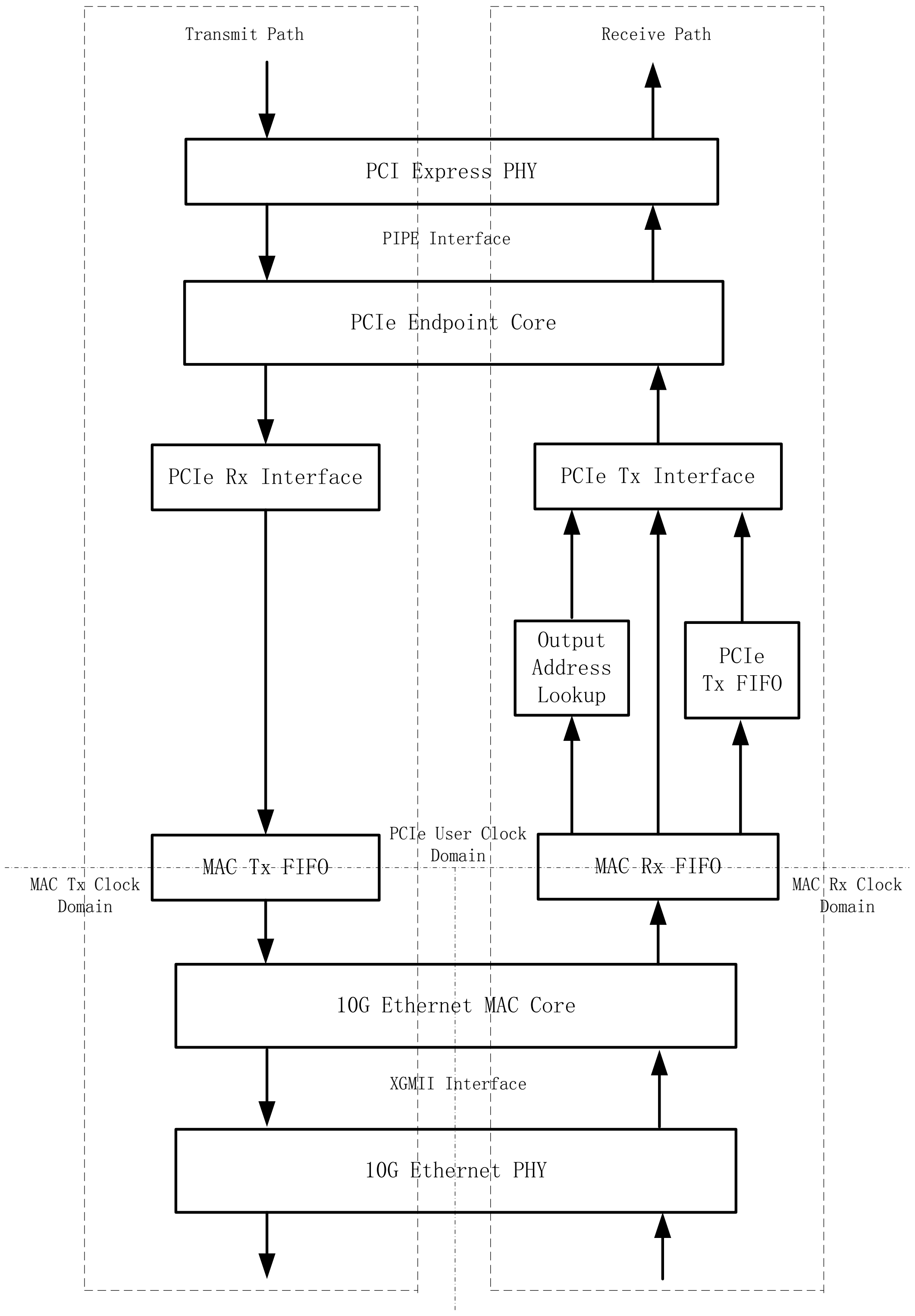


Figure 18: Implementation of the Ethernet over PCIe adaptor

5.2. Available Resources and Platform Selection

Related open source or commercial projects and IP cores can be used as reference designs. Here is a description of some of the available resources for designing the adaptor logic blocks.

Ethernet MAC IP cores are available from open source community, FPGA suppliers and IP core developers.

OpenCores organization shares a tri-mode (10/100/1000 Mbps) Ethernet MAC core which implements a MAC controller conforming to the IEEE 802.3 specification [30]. The core is designed using less than 2000 LCs/LEs to be implemented fully function. The core also provides tcl/tk script language for configuring parameters. A verification system is designed with a tcl/tk interface, by which the stimulus can be generated and the output packets can be verified by a CRC-32 checksum. The advantage of the OpenCore MAC core is that the internal logics can be modified for specific applications. The disadvantages are the lack of documentation and the bandwidth is capped up to 1 Gbps.

Both Xilinx® and Altera® provide tri-mode MAC cores and 10G MAC cores for design evaluation. The 10G MAC core provided by Xilinx supports most of the device families up to the latest Virtex-7. The core is designed to the IEEE 802.3-2008 specification, supporting flow control [31]. The client side interface uses AXI4, which is a family of micro controller buses. The PHY side uses either an external 32-bit XGMII interface or an internal 64-bit interface. The 10G MAC core provided by Altera also supports the device families up to the latest Stratix V GX [32]. The client side uses the Avalon [33] interface, which is a standard bus for Altera IP cores. The PHY side uses the XGMII interface for connecting to an internal XAUI/PHY IP or external PHY chip. Both MAC cores from the Xilinx and Altera have flexible configurations and rich features.

PCI Express endpoint IP cores are available from IP core developers and FPGA suppliers.

The XpressRICH IP core provided by PLDA® is an all-in-one soft IP including PCI Express endpoint, root port, dual-mode and switch controller IPs [34]. XpressRICH supports up to PCI Express Specification Rev 2.0 with x8 lanes. The PHY side interface is the PHY Interface for PCI Express (PIPE). The application layer interface supports 64-bit or 128-bit data path per VC. The disadvantage of XpressRICH core is that it does not support the standard AXI4 interface for Xilinx FPGAs or Avalon interface for Altera FPGAs.

Both Xilinx® and Altera® provide PCI Express endpoint IP cores. The PCIe endpoint core from Xilinx complies with PCI Express Specification Rev 2.1 [35]. The core supports one VC with a maximum of x8 lanes. The main application layer interface is an AXI4 interface for transactions. The PHY side interface uses differential transmit and receive pairs. The DMA engine is not included in the core, but the engine can be purchased from third-party companies. Altera provides the PCIe endpoint core complying with PCIe Specification Rev 2.0 [36]. The core supports up to x8 lanes with one VC. The application layer side uses Avalon interface, while the PHY side is PIPE interface. The core provides a DMA reference design.

Some open source projects are available for using as reference designs.

NetFPGA [37] is a platform for building high performance networking systems on FPGA development board. The latest version, NetFPGA-10G, supports 10G Ethernet MAC and PCI Express gen2 with x8 lanes. In the network interface card (NIC) reference design, Ethernet packets from four 10G MAC cores are directed and arbitrated to the PCI Express interface. The output port lookup logic is used for setting the destination port data field of all incoming packets to be that of the host, and sets the destination port of all outgoing packets to be that of the small form-factor pluggable (SFP+) connection. This logic can be a reference of the Output Address Lookup logic in the switch design.

An open router project, called BORA-BORA, implements multi-gigabit-per-second routing on a shared PCI bus [38]. The group developed a High-speed Enhanced Routing Operation (HERO) architecture between a MAC core and a PCI core in a programmable NIC card. Ethernet packets whose destination can be determined locally can be exchanged directly between two NICs using a fast path. However, the project uses the PCI-X bus instead of the PCI Express bus. The HERO core lies in the network layer for packet routing instead of the data link layer for switching.

RiceNIC [39] is a reconfigurable and programmable Gigabit Ethernet NIC developed in an open platform. RiceNIC provides significant computation and storage resources that are largely unutilized when performing the basic tasks of network interface. The hardware architecture can be used as a reference, but again, the design uses PCI instead of PCI Express.

The FPGA development board needs to be selected from the two largest FPGA suppliers, Altera and Xilinx. Table 4 summarizes the available resources when using FPGAs from different suppliers.

Table 4: Platform Selection

	Altera® Stratix V	Xilinx® Virtex-7
Open Source MAC IP core	OpenCore triple speed (10/100/1000 Mb/s) MAC core	OpenCore triple speed MAC core
Ethernet MAC IP core from FPGA suppliers	1G/10G/40G/100G	1G/10G/100G/400G
	10G MAC MegaCore®. Interfaces: MAC Client side uses Avalon Interface. PHY side need an Altera PHY IP core such as a soft XAUI PHY with external PHY.	10G MAC LogiCore®. Interfaces: MAC Client side uses AXI4 interface. PHY side uses XGMII or internal interfaces.
PCI Express IP core from FPGA suppliers	Gen 1/2/3	Gen 2/3
	PCI Express Compiler MegaCore Functions. Application interfaces:	7 Series Integrated Block for PCI Express with AXI4-

	Avalon-ST application interface, descriptor/data application interface or Avalon-MM application interface.	Stream interface. DMA engine is available from third-party company.
PCI Express IP core from third-party: PLDA® XpressRICH core	Application layer side: descriptor/data interface. PHY side: PIPE interface with PHYPCS layer.	
	PHY side: need to write PHYPCS layer or purchase PHY IP and modify the IP.	Application layer side: need to migrate to AXI4 interface. PHY side: need to write PHYPCS layer or purchase PHY IP and modify the IP.
PLDA® P2P core	Application layer: AXI4 interface only, no Avalon or DMA interface, support Gen 1/2/3. Both soft/hard IPs are available.	

The Xilinx Virtex-7 development board is selected to be the platform of the PCIe-based Ethernet switch adaptor. First, the current PCIe IP cores do not support P2P communication protocols. The cost of a customized P2P PCIe endpoint core is too high. PLDA® announced that it will release a P2P IP core with AXI4 interface recently. Hence, designing with the Xilinx platform with AXI4 interface IPs will be easier for migrating to the P2P core in the future. Secondly, the AXI4 interface from Xilinx is a widely used industrial standard compared to the Avalon interface from Altera. IP cores provided by ARM partners can be used in Xilinx platform as well. Third-party AXI tools are available for providing a variety of verification, system development, and performance characterization. Finally, most of the open source projects are developed based on Xilinx platforms. Reusing the IP cores shortens the developing cycle of the switch.

5.3. AXI4 Interface

AXI is part of ARM AMBA, a family of micro controller buses. AXI4 interface is widely used in Xilinx IPs [40]. AXI4 is a standardized protocol and is flexible for different applications. There are three types of AXI4 interfaces. AXI4 is used for high-performance memory-mapped requirements. AXI4-Lite is used for simple, low-throughput memory-mapped communication. AXI4-Stream is used for high-speed streaming data.

Both Ethernet and PCIe packet data transfer in the adaptor implement the AXI-Stream interface. AXI-Stream does not require an address phase to be transmitted with the data phases and allows unlimited data burst size. The Ethernet MAC addresses and PCIe routing addresses are encapsulated into packet data. Two interconnect IPs act as a master-slave pair in the AXI4 interface. The master interface is responsible for data transmission; the slave interface is responsible for data reception. Table 5 lists the AXI4-Stream signals that are mainly used in the logic cores of the adaptor.

Table 5: AXI4-Stream Interface Signals

Signal	Direction	Representation
TVALID	Master to Slave	Indicating the corresponding data phase is valid.
TREADY	Slave to Master	Indicating the slave is ready to receive data.
TDATA	Master to Slave	Conveying stream data.
TKEEP	Master to Slave	Indicating which byte(s) of the packet phase is valid
TLAST	Master to Slave	Indicating the last data phase

5.4. 10G MAC Core

The adaptor adopts the 10G MAC core IP provided by Xilinx. The MAC client side uses the AXI-Stream interface for data stream transfer and the AXI4-Lite interface for management and MDIO interfaces. The PHY side uses XGMII and MDIO interfaces. Figure 19 shows the block diagram of the 10G MAC core [31].

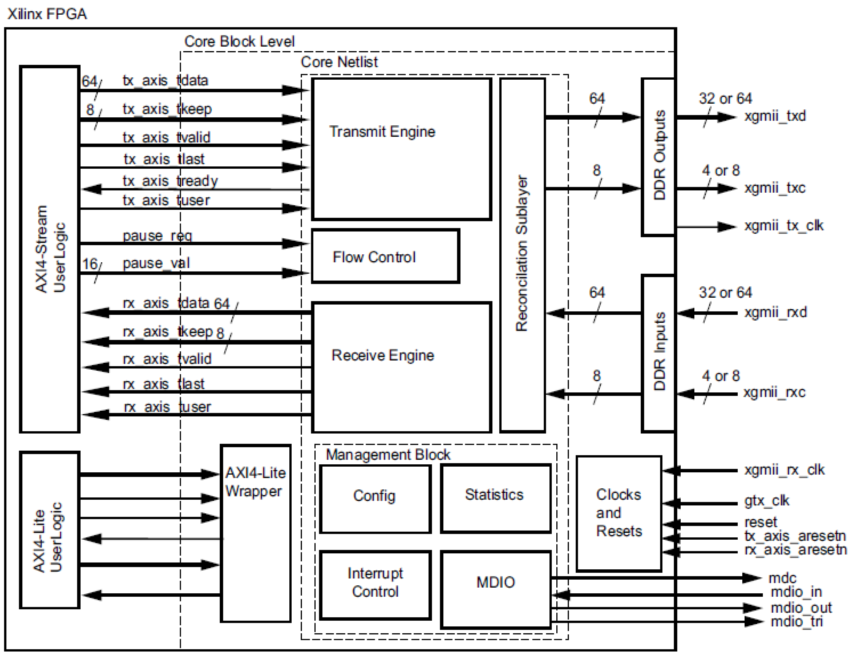


Figure 19: 10G MAC Core Block Diagram

When the MAC core receives Ethernet frames, the client must be prepared to accept data at any time. The data is transmitted on consecutive clock cycles through the “rx_axis_tdata” signal. The “rx_axis_tlast” signal asserts when the final data phase approaches. The “rx_axis_tuser” signal asserts along with the “rx_axis_tlast” signal if the frame was successfully received by the MAC core, thus indicating the frame should be analyzed by the client. The “rx_axis_tkeep” signal indicates which byte(s) of the data phase is (are) valid. The “keep” signal should be especially paid attention to the last data phase.

When the MAC core transmits Ethernet frames, the client should asserts the “tx_axis_tvalid” signal with the “tx_axis_tdata” and “tx_axis_tkeep” signals presented in the same clock cycle. After the MAC core asserts the “tx_axis_tready” signal, the client must provide the remainder of the data phases to the core. The end of the data phase is indicated by the “tx_axis_tlast” signal with an appropriate “tx_axis_tkeep” signal.

5.5. Output Address Lookup

The Output Address Lookup logic block consists of an Ethernet Parser, a Content-Addressable Memory (CAM), and an Array as shown in Figure 20. The 10G MAC Rx FIFO sends Ethernet frames and control signals to the Ethernet Parser logic. The parser logic then sends destination MAC addresses to the CAM and sends source MAC addresses to the driver or OS. The OS gathers the source MAC addresses along with the routing addresses/IDs of the corresponding adaptors in the PCIe system. The OS then creates an address mapping table in the PCIe system and sends the table to the CAM and Array in each adaptor. When the CAM and Array receive the destination MAC addresses sent from the Ethernet Parser, they look up the PCIe routing addresses/IDs by their destination addresses. The routing addresses/IDs are finally sent to the PCIe Tx Interface. The Array is necessary since the PCIe routing addresses are too wide to be stored as CAM addresses.

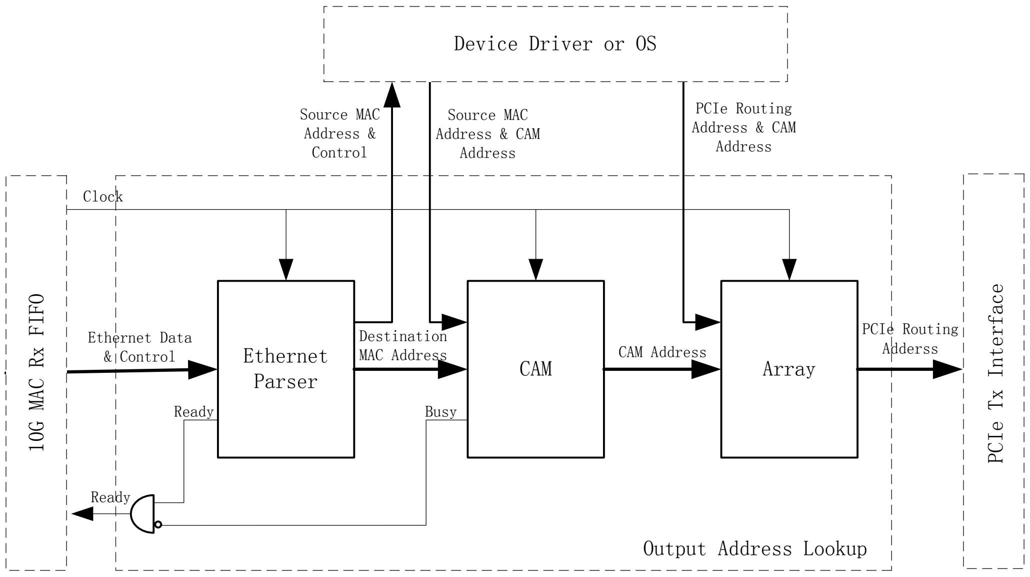


Figure 20: Output Address Lookup Block Diagram

5.5.1. Ethernet Parser

The Ethernet parser logic is responsible for analyzing the headers of Ethernet packets that are received. The parser logic extracts the destination MAC address, source MAC address, and length/type field. These fields are sent to the upper layer logic with a “complete” signal.

The structure of the Ethernet frames is shown in Figure 21. The preamble and SFD field in the frame header are disabled in this implementation. The first data phase is aligned with the Least Significant Bit (LSB) of the destination MAC address field. The parser logic reads the entire 48 bits of the destination MAC address and the first 16 bits of the source MAC address during the first data phase. The logic reads the rest of the 32 bits of the source MAC address and length/type field during the second data phase.

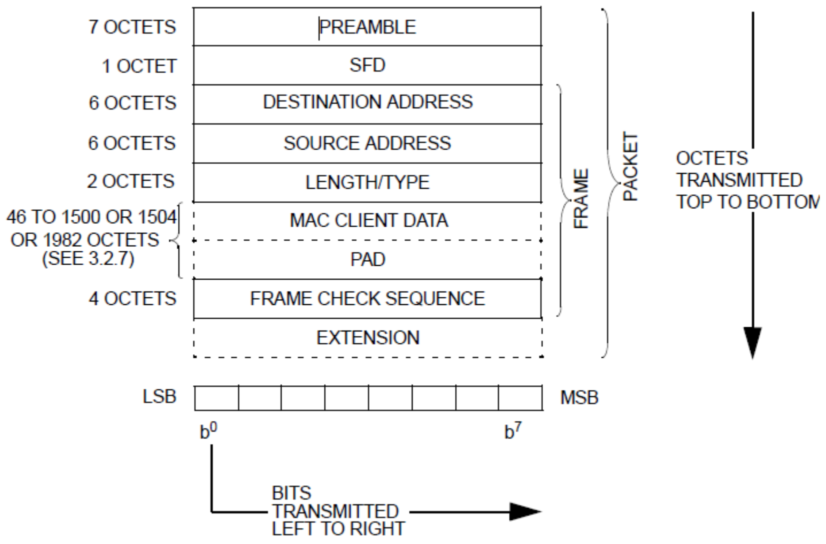


Figure 21: Ethernet Frame Structure

The implementation adopts a 4-state Finite State Machine (FSM) as shown in Figure 22. Table 6 describes details of the states transition.

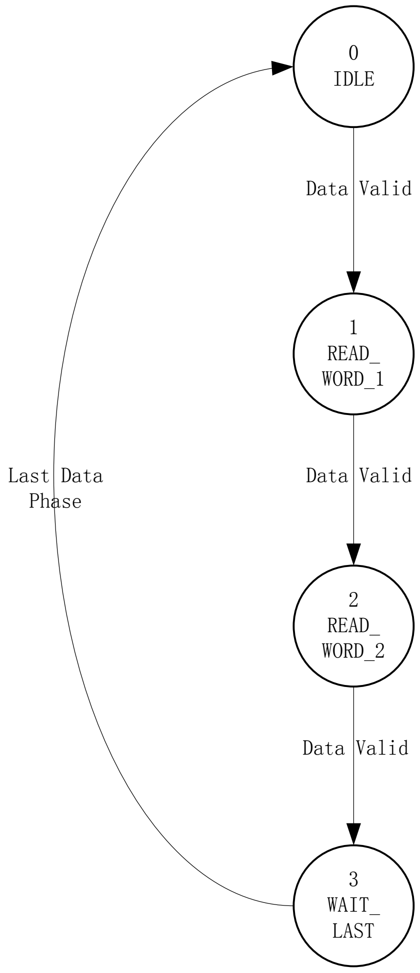


Figure 22: Ethernet Parser FSM

Table 6: Ethernet Parser FSM Description

No.	State Name	Description
0	IDLE	Wait for the first data phase. If the Ethernet frame from the MAC Rx FIFO is valid, the FSM transfers to state 1.
1	READ_WORD_1	Read the first 2DW of the Ethernet frame. The FSM receives and analyzes the first 2DW of the frame, extracts the destination MAC address and part of the source MAC address. The FSM transfers to state 2 on the next clock cycle.
2	READ_WORD_2	Read the second 2DW of the Ethernet frame. The FSM receives and analyzes the second 2DW of the frame, extracts part of the source MAC address and the entire Ethernet length/type field. The FSM also reports “parsing completed” signal to the upper layer logic. The FSM

		transfers to state 3 on the next clock cycle.
3	WAIT_LAST	Wait for the last data phase. The FSM transfers to state 0 when the “last” signal from the MAC Rx FIFO is asserted.

5.5.2. Address Mapping Table

The BARs of the adaptors in the switch are configured in the manner stated in Section 4.5. The last four bits of the BARs is “1100” in binary (“C” in hexadecimal). Each adaptor requests a minimum of 128-bit memory address storage space. Assume the OS assigns the BARs in Adaptors 2, 3, and 5 as shown in Figure 23. The BDF numbers are also assigned during the enumeration process.

As discussed in Section 4.3, the OS creates the address mapping table based on the source MAC addresses of the received Ethernet packets and the routing addresses/IDs of the adaptors. The routing addresses can be selected from the range of the Base Address Registers (BARs) of the adaptors. In this implementation, the routing addresses are set to be the same as the BARs in the adaptors. Prior to Ethernet switching, Adaptor 3 and Adaptor 5 receive Ethernet frames with two types of source MAC addresses: “060504030201h” and “010203040506h”. The two adaptors send the two MAC addresses with their corresponding BARs and BDF numbers to the OS. The OS then creates an address mapping table as illustrated in Table 7. The OS must write the mapping table in the Output Address Lookup logic in each adaptor and update the table periodically. The address mapping table is stored in the CAM and the array of the lookup logic.

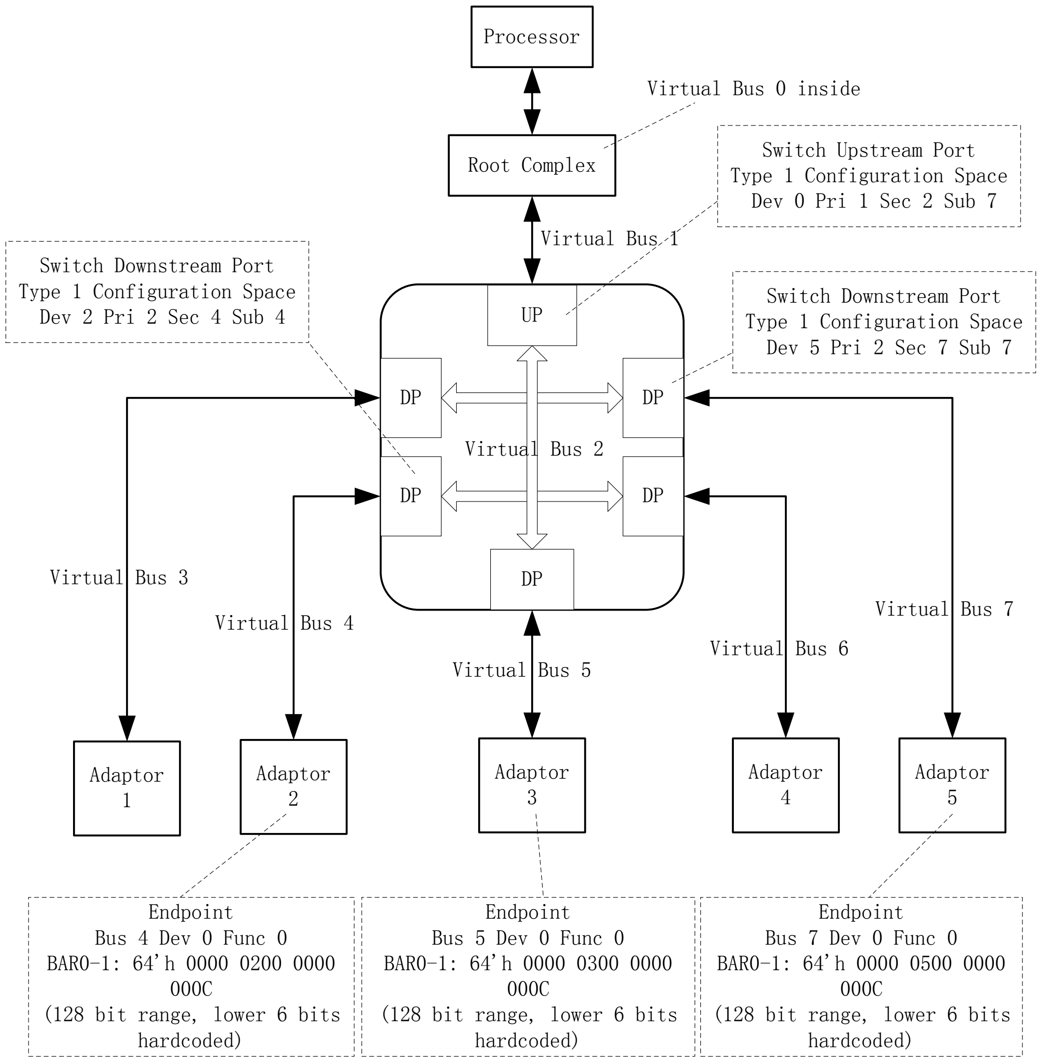


Figure 23: Address Mapping Illustration

Table 7: An Example of the Address Mapping Table

[134:131]	[130]	[129:128]	[127:80]	[79:17]	[47:16]	[15:8]	[7:3]	[2:0]
Entry No.	Valid (1 bit)	Age (2 bits)	Source MAC Address (48 bits)	Routing Address (upper 32 bits)	Routing Address (lower 32 bits)	Bus # (8 bits)	Device # (5 bits)	Func # (3 bits)
0	1	00	060504030201h	00000300h	0000000Ch	5h	0h	0h
1	1	00	010203040506h	00000500h	0000000Ch	7h	0h	0h

5.5.3. Content-Addressable Memory (CAM)

CAM enables faster data searches by performing content matching rather than address matching performed by standard memory cores. The adaptor implements a CAM inside the Output Address Lookup logic based on the block RAM from Xilinx [41]. The block RAM behaves like a two-dimensional grid with CAM data and addresses as the axes. All the possible combinations of data/address are stored in the grid. Whether a particular data is stored in the address or not is marked by "1" or "0" in the CAM. An example CAM of 4 words deep and 2 bits wide is shown in Table 8. In this example, CAM data "01", "11", and "00" are stored in CAM address "0", "2", and "3", respectively.

Table 8: Block RAM-based CAM

	RAM Data/CAM Address				
		0	1	2	3
RAM Address/ CAM Data	00	0	0	0	1
	01	1	0	0	0
	10	0	0	0	0
	11	0	0	1	0

The block RAM-based implementation of CAM has a single clock cycle latency on the read operation, and two clock cycles latency on the write operation. Table 9 shows the interface signals of the CAM. In the read operation, the Ethernet Parser logic sends a destination MAC address to the CAM by the CMP_DIN signals. The CAM asserts the MATCH signal on the next clock cycle if the MAC address matches the data in the CAM. The corresponding CAM address is also presented on the MATCH_ADD bus. In the write operation, the OS sends the source MAC address and the corresponding CAM address to the CAM through the DIN and WR_ADD buses respectively, with the WE signal asserted. The CAM asserts the BUSY signal when it is writing data. Write operations must be two clock cycles apart when executing consecutive write operations.

Table 9: CAM Interface Signals

Signal	Direction	Description
CMP_DIN [47:0]	IN	Data to look up from the CAM during a read operation. 48-bit destination MAC addresses are the input.
DIN [47:0]	IN	Data to be written to the CAM during write operation. 48-bit source MAC addresses are the input.
BUSY	OUT	Indicating a write operation is currently being executed.
WE	IN	Write enable signal, asserted when DIN is valid in a write operation.
WR_ADDR [3:0]	IN	Write address, indicating the location to which the data on DIN is written in CAM.
MATCH	OUT	Indicating at least one location in the CAM contains the data on the CMP_DIN bus.
MATCH_ADDR [3:0]	OUT	Indicating the CAM address where matching data is located.
CLK	IN	Clock signal. All CAM operations are synchronous to the rising edge of the clock.

5.6. AXI4-Stream FIFO

First-In First-Out (FIFO) is mainly used in pipeline design for safely passing data from one clock domain to another asynchronous clock domain. FIFO can also be used in the same clock domain for buffering data and flow control. The adaptor uses FIFOs for interfacing the 10G MAC clock domain and the PCI Express user clock domain as well as buffering data between different logic blocks.

Xilinx provides a customizable FIFO IP with an AXI4-Stream interface [42]. The FIFO can be implemented by either block RAM or distributed memory. The adaptor uses the block RAM implementation for larger depth and Error Correction Checking (ECC) support in the future. The FIFO is divided into a write clock domain and a read clock domain. In the write clock domain, the transmitter is master and the FIFO is slave. In the read clock domain, the FIFO is master and the receiver is slave. Table 10 shows the major signals in the write clock domain of the FIFOs implemented in the adaptor. The signals in the read clock domain are similar to the signals in the write clock domain, except that the reset signal is global.

Table 10: Major Signals in the Write Clock Domain of the AXI4 FIFO

Signal	Direction	Description
S_AXIS_TDATA [63:0]	IN	Data bus used for conveying Ethernet frames.
S_AXIS_TVALID	IN	Indicating the master is sending a valid data phase.
S_AXIS_TREADY	OUT	Indicating the slave can accept a transfer in the current cycle.
S_AXIS_TKEEP [8:0]	IN	Indicating whether the corresponding byte of the data phase is valid or not.
S_AXIS_TLAST	IN	Indicating the last data phase.
S_ARESETN	IN	Active low global reset signal.
S_ACLK	IN	Slave clock. Write operation occurs at the rising edge of the clock.

The adaptor contains a 10G MAC Receive FIFO and a 10G MAC Transmit FIFO for dividing the adaptor logics into three different clock domains as well as buffering data between the MAC layer and PCIe layers. In the Receive Path, the MAC Rx FIFO stores Ethernet frames from the 10G MAC core whenever the frames are valid. The Output Address Lookup logic then reads data from the MAC Rx FIFO while the PCIe Tx Interface counts the lengths of the frames. In the Transmit Path, the MAC Tx FIFO stores the Ethernet frames from the PCIe Rx Interface. The FIFO then forwards the Ethernet frames to the 10G MAC core when the core is ready.

The adaptor uses another AXI4 FIFO, PCIe Tx FIFO, in the PCIe User Clock Domain of the Receive Path to buffer data between the MAC Rx FIFO and the PCIe Tx Interface. The PCIe Tx FIFO is used for storing Ethernet frames when the PCIe Tx

Interface counts the lengths of the Ethernet frames sent from the MAC Rx FIFO. The PCIe Tx FIFO buffers these frames for the following transfer of the TLP data payload.

5.7. PCI Express Endpoint Core

The adaptor uses the XpressRich Core [43] provided by PLDA as the PCI Express Endpoint Core. The XpressRich core supports up to PCI Express Base Specification Revision 2.0 with x8 lanes. The core consists of PCIe Transaction Layer, Data Link Layer, and MAC sub-layer of the Physical Layer. The Transaction Layer side uses a data/descriptor interface (including a transmit interface and a receive interface); the Physical Layer side uses a PIPE interface. The major signals of the transmit interface on the Transaction Layer side are listed in Table 11. The signals of the receive interface on the Transaction Layer side are similar to the signals of the transmit interface.

Table 11: Major Signals of the Transmit Interface on the Transaction Layer Side of the XpressRich Core

Signal	Direction	Description
TX_REQ	IN	Requesting data transmission, asserted until the TX_ACK signal is asserted.
TX_DESC [127:0]	IN	Transmit descriptor bus used for transmitting TLP header. The format of the descriptor complies with the PCIe specification.
TX_ACK	OUT	Acknowledgment signal asserted by the core after receiving the request, indicating the core is ready to accept descriptor and data.
TX_DFR	IN	Transmit data phase framing signal, asserted from the time of request until proceeding to the last data phase.
TX_DATA [63:0]	IN	Transmit data bus used for sending data payload of TLPs.
TX_WS	OUT	Transmit wait states signal, asserted by the core to suspend data transmission.
TX_ERR	IN	Error signal used for discard or nullify a TLP.

When the PCIe Transmit Interface transmits a TLP to the PCIe Endpoint Core, the Tx Interface must assert the “tx_req” and “tx_dfr” signals with a descriptor presented on the “tx_desc” signal until the Endpoint Core asserts the “tx_ack” signal. TLP data payload starts to be transferred on the next clock cycle of the asserted “tx_ack” signal. The Tx Interface deasserts the “tx_dfr” signal before the last data phase. If the “tx_ws” signal is asserted by the Endpoint Core, the Tx Interface suspends the transmission until the “tx_ws” signal is deasserted. If the Endpoint Core reports an error by the “tx_err” signal, the Tx Interface cancels the current transmission and removes the remainder of the packet in the FIFOs. The receiving process is similar to the transmitting process discussed above.

5.8. PCI Express Transmit Interface

The PCI Express Transmit Interface is responsible for converting Ethernet MAC layer signals to PCIe transaction layer signals. The interface generates TLP headers (especially the descriptor fields in the headers) based on TLP type, traffic class, length of the packet, Ethernet frame “keep” signal, and routing address of the destination PCIe endpoint. The transmit interface then sends the data payload along with the TLP header and other control signals to the PCIe Endpoint Core. The implementation adopts a 10-state FSM as shown in Figure 24. Table 12 is the description of the FSM.

The PCIe Tx Interface has to count the length of each Ethernet frame. TLP headers have a length field indicating the size of the data payload. The Ethernet frames cannot be transferred until the length information is collected. On the other hand, the length/type field in the Ethernet frame indicates the length of the MAC client data if the value of this field is less than or equal to 1500 decimal. The length/type field in the Ethernet frame indicates the type of the frame if the value of the field is greater than or equal to 1536 decimal. Thus the PCIe Tx Interface has to count the lengths of the Ethernet frames in the case that the length/type field does not contain length information.

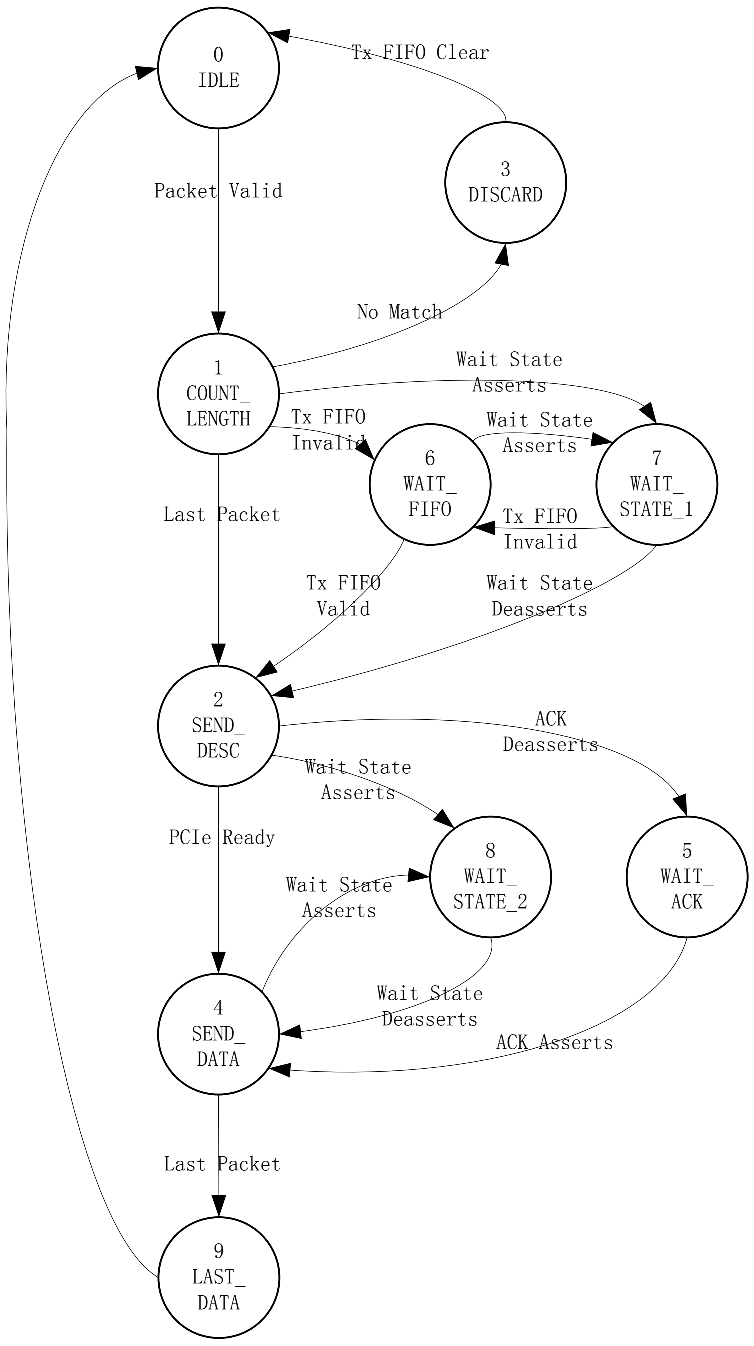


Figure 24: PCIe Tx Interface FSM

Table 12: PCIe Tx Interface FSM Description

No.	State Name	Description
0	IDLE	Waits for valid packets sent from the MAC Rx FIFO. If a valid signal is sent from the MAC Rx FIFO, the FSM transfers to state 1. If an invalid signal is sent from the

		MAC Rx FIFO, the FSM keeps the current state.
1	COUNT_LENGTH	Count the number of DWs sent from the MAC Rx FIFO. The FSM also registers the “keep” signal from the MAC Rx FIFO. At the end of the packet, if the destination MAC address matches the address mapping table while the PCIe core is ready, the FSM transfers to state 2. If no match is found, the FSM transfers to state 3. If the PCIe Tx FIFO does not have valid data, the FSM transfers to state 6. If the “wait state” signal is asserted, the FSM transfers to state 7. Otherwise, the FSM keeps in the current state.
2	SEND_DESC	Send TLP descriptor to the PCIe endpoint core. The formation of the descriptor is based on the destination routing address, Ethernet packet length, and the “keep” signal. If the PCIe core is ready for receiving, the FSM transfers to state 4 to send data payload. If the ACK signal of the PCIe core is not asserted, the FSM transfers to state 5. If the “wait state” signal is asserted by the PCIe endpoint core, the FSM transfers to state 8. If the PCIe Tx FIFO is not valid or the “last” signal is asserted (indicating the FSM still remains at the last 2DW of the previous packet), the FSM transfers to state 6.
3	DISCARD	Discard packets in the PCIe Tx FIFO. The interface sends a reset signal to the Tx FIFO to clear data in the queue. The FSM transfers to state 0 after the reset signal is sent.
4	SEND_DATA	Send Ethernet packets (including headers) to the PCIe endpoint core from the PCIe Tx FIFO. If the “wait state” signal is asserted by PCIe core, the FSM transfers to state 8. If the last packet is sent, the FSM transfers to state 0. Otherwise, the FSM keeps in the current state.
5	WAIT_ACK	Wait for the ACK signal to be asserted by the PCIe endpoint core. If the ACK signal is asserted and the PCIe Tx FIFO is valid, the FSM transfers to state 4. If the ACK signal is not asserted, the FSM keeps in the current state.
6	WAIT_FIFO	Wait for the PCIe Tx FIFO to send valid data. If the data from the FIFO is valid and the “wait state” is deasserted, the FSM transfers to state 2. If the data from the FIFO is valid and the “wait state” is asserted, the FSM transfers to state 7. Otherwise, the FSM keeps in the current state.
7	WAIT_STATE_1	Wait for the “wait state” signal to be deasserted by the PCIe endpoint core. If the “wait state” signal is deasserted and the PCIe Tx FIFO has valid data, the FSM transfers to state 2. If the “wait state” signal is deasserted and the PCIe Tx FIFO has invalid data, the FSM transfers to state 6. Otherwise, the FSM keeps in the current state.
8	WAIT_STATE_2	Wait for the “wait state” signal to be deasserted by the PCIe endpoint core. If the “wait state” signal is deasserted, the FSM transfers to state 4. Otherwise, the FSM keeps in the current state.
9	LAST_DATA	Send the last data phase. The FSM asserts the “last” signal

		at the same time. The FSM transfers to state 0 on the next clock cycle.

5.9. PCI Express Receive Interface

The PCI Express Receive Interface logic is responsible for converting PCIe transaction layer signals to Ethernet MAC layer signals. The logic analyzes the TLP headers (especially the descriptor field) of the received TLPs and decides whether the TLPs are the supported type or not. The interface logic receives and decapsulates the supported TLPs and then sends them to the MAC Tx FIFO. The logic aborts transmission when the TLP is not supported. The implementation adopts a 9-state FSM as shown in Figure 25. Table 13 describes the state diagram in detail.

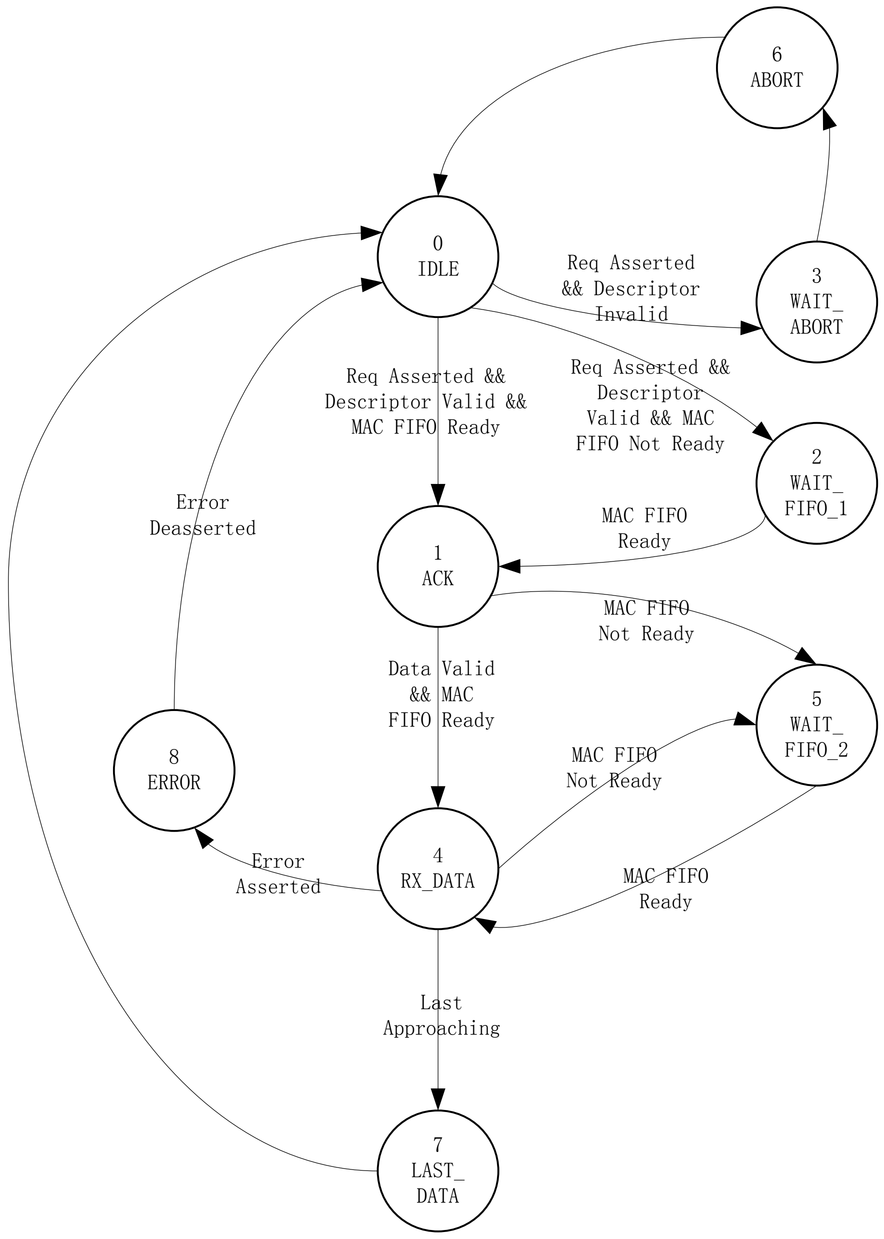


Figure 25: PCIe Rx Interface FSM

Table 13: PCIe Rx Interface FSM Description

No.	State	Description
0	IDLE	Wait for the request signal from the PCIe endpoint core to be asserted. If the core requests a TLP transmission, the descriptor field is valid, and the MAC TX FIFO is ready to receive data, then the FSM transfers to state 1. If the PCIe core requests a TLP transmission but the descriptor is

		invalid, the FSM transfers to state 3. If the PCIe core requests a TLP transmission but the MAC Tx FIFO is not ready, the FSM transfers to state 2. Otherwise, the FSM keeps in the current state.
1	ACK	Send an ACK signal to the PCIe endpoint core and receive the first DW of data if the data is valid. If the TLP data is valid and the MAC Tx FIFO is ready, the FSM transfers to state 4. If the MAC Tx FIFO is not ready, the FSM transfers to state 5.
2	WAIT_FIFO_1	Wait for the MAC Tx FIFO to be ready for receiving data. The FSM sends an ACK signal and asserts the “wait state” signal to the PCIe core as well. If the MAC Tx FIFO is ready, the FSM transfers to state 1. Otherwise, the FSM keeps in the current state.
3	WAIT_ABORT	Prepare the “abort” signal to be asserted in the next clock cycle. The FSM asserts the “wait state” signal to the PCIe core to throttle data transmission. The FSM transfers to state 6 on the next clock cycle.
4	RX_DATA	Receive data frames from the PCIe endpoint core and send the frames to the MAC Tx FIFO. If the “data frame” signal is deasserted, which indicates the last data phase is approaching, the FSM transfers to state 7. If the MAC Tx FIFO is not ready to receive data during the transmission, the FSM transfers to state 5. If the “error” signal is asserted during the transmission, the FSM transfers to state 8. Otherwise, the FSM keeps in the current state.
5	WAIT_FIFO_2	Wait the MAC Tx FIFO to be ready for receiving data. The FSM asserts the “wait state” signal to the PCIe core to throttle transmission. If the MAC Tx FIFO is ready, the FSM transfers to state 4. Otherwise, the FSM keeps in the current state.
6	ABORT	Abort transmission. The FSM sends the “abort” signal to the PCIe core to report unsupported TLP. The FSM transfers to state 0 on the next clock cycle.
7	LAST_DATA	Transmit the last data phase. The FSM receives the last data phase and sends the data to the MAC Tx FIFO. The interface also sends a “keep” signal to the MAC Tx FIFO based on the “byte enable” signal from the PCIe core to indicate which byte(s) of the data is valid. The FSM transfers to state 0 on the next clock cycle.
8	ERROR	Report error. The interface stops transmitting the current data frame and reports error to PCIe system. The FSM transfers to state 0 when the “error” signal is deasserted. Otherwise, the FSM keeps in the current state.

CHAPTER 6

EXPERIMENTAL RESULTS AND DESIGN EVALUATION

6.1. Simulation and Testbench Setup

The simulation is set up as a loopback mode as shown in Figure 26. In the Receive Path, four Ethernet frames are sent to the receive port of the 10G MAC core through the XGMII signals as stimuli. The 10G MAC Core converts the stimuli to Ethernet MAC frames; the MAC Rx FIFO buffers the MAC frames. The signal monitor probes the MAC frames from the MAC Rx FIFO interface. The MAC frames then pass through the Output Address Lookup logic and are probed by the signal monitor. The routing information and MAC frames are handed to the PCIe Tx Interface and are converted to the PCIe data/descriptor interface signals. The signal monitor probes the data/descriptor signals. The signals are then looped back to the PCIe Rx Interface as if they were sent from the PCIe Endpoint Core.

In the Transmit Path, the PCIe Rx Interface converts the data/descriptor interface signals into MAC frames; the MAC frames are probed by the signal monitor. The MAC frames are then buffered by the MAC Tx FIFO; the signal monitor probes the signals at the MAC Tx Client side. The MAC frames are finally passed through the 10G MAC core and converted to the XGMII signals. The signal monitor probes the signals transmitted from the MAC core.

The testbench generates clock signals of 156.25 MHz for the MAC Clock Domains (both the MAC Rx Clock Domain and MAC Tx Clock Domain) and clock signals of 125 MHz for the PCIe User Clock Domain. The testbench also writes the Address Mapping Table to the CAM and array after the system is reset.

The testbench generates the stimuli of XGMII frames as shown in Table 14 and Table 15. Frame 0 is a minimum length frame. Frame 1 is a regular frame with longer length. Frame 2 is a frame with error asserted. Frame 3 is less than the minimum length and is padded up to the minimum length. The bits in the control frames indicate whether the corresponding octets in the data frames are valid or not. Note that the control frame is different from the “txc” and “rxc” signals in the XGMII interface.

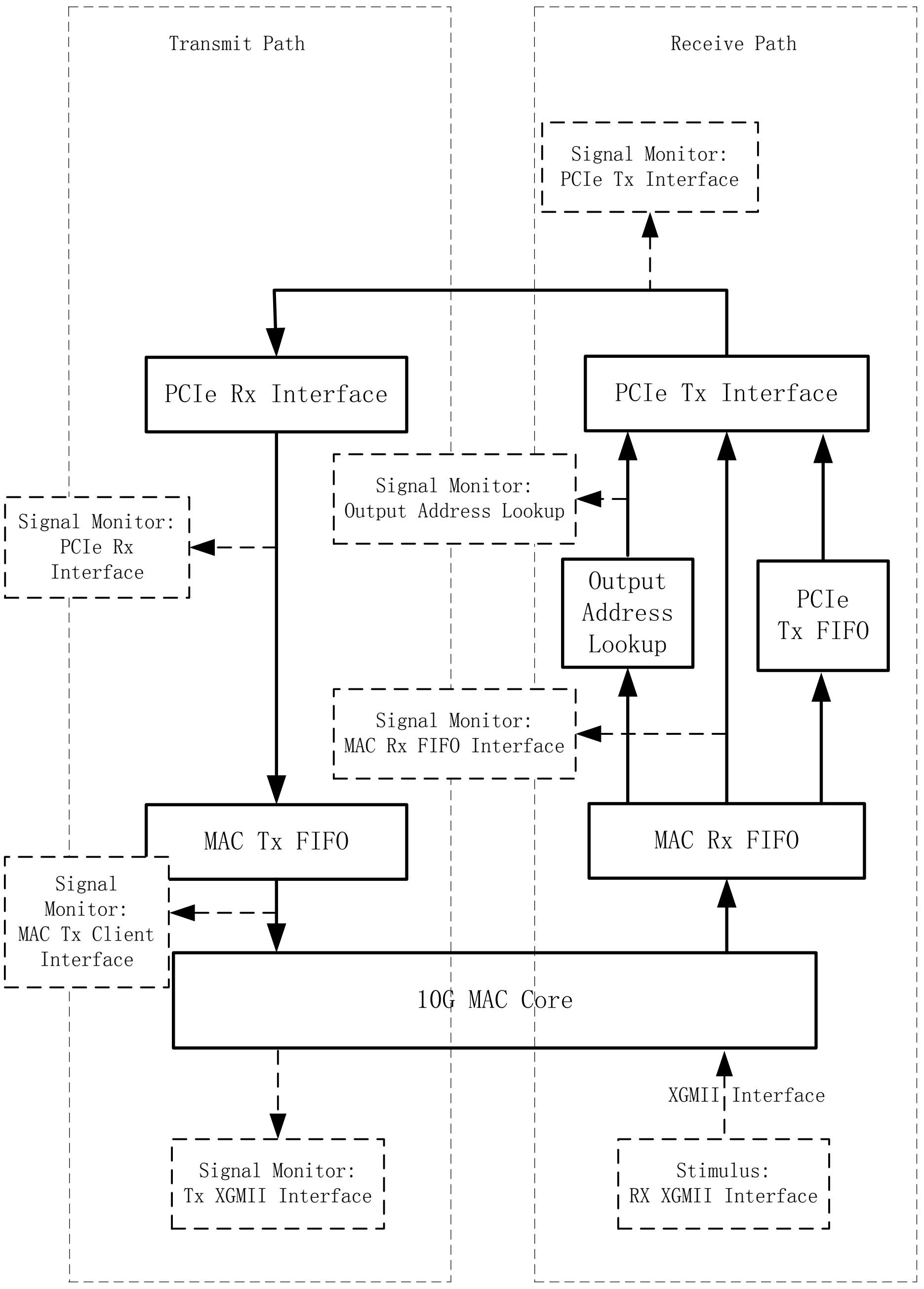

Transmit Path
Receive Path
Signal Monitor: PCIe Tx Interface
PCIe Rx Interface
PCIe Tx Interface
Signal Monitor: Output Address Lookup
Signal Monitor: PCIe Rx Interface
Output Address Lookup
PCIe Tx FIFO
Signal Monitor: MAC Rx FIFO Interface
MAC Tx FIFO
MAC Rx FIFO
Signal Monitor: MAC Tx Client Interface
10G MAC Core
XGMII Interface
Signal Monitor: Tx XGMII Interface
Stimulus: RX XGMII Interface

Figure 26: Simulation Setup Block Diagram

Table 14: Stimuli: XGMII Data Frames

	Frame 0	Frame 1	Frame 2	Frame 3
preamble 0	555555FB	555555FB	555555FB	555555FB
preamble 1	D5555555	D5555555	D5555555	D5555555

data 0	04030201	03040506	04030201	03040506
data 1	02020605	05060102	02020605	05060102
data 2	06050403	02020304	06050403	02020304
data 3	55AA2E00	EE110080	55AA2E80	EE111500
data 4	AA55AA55	11EE11EE	AA55AA55	11EE11EE
data 5	55AA55AA	EE11EE11	55AA55AA	EE11EE11
data 6	AA55AA55	11EE11EE	AA55AA55	11EE11EE
data 7	55AA55AA	EE11EE11	55AA55AA	EE11EE11
data 8	AA55AA55	11EE11EE	AA55AA55	00EE11EE
data 9	55AA55AA	EE11EE11	55AA55AA	00000000
data 10	AA55AA55	11EE11EE	AA55AA55	00000000
data 11	55AA55AA	EE11EE11	55AA55AA	00000000
data 12	AA55AA55	11EE11EE	AA55AA55	00000000
data 13	55AA55AA	EE11EE11	55AA55AA	00000000
data 14	AA55AA55	11EE11EE	AA55AA55	00000000
data 15	00000000	EE11EE11	55AA55AA	00000000
data 16	00000000	11EE11EE	AA55AA55	00000000
data 17	00000000	EE11EE11	55AA55AA	00000000
data 18	00000000	11EE11EE	AA55AA55	00000000
data 19	00000000	EE11EE11	55AA55AA	00000000
data 20	00000000	11EE11EE	00000000	00000000
data 21	00000000	0000EE11	00000000	00000000

Table 15: Stimuli: XGMII Control Frames

	Frame 0	Frame 1	Frame 2	Frame 3
ctrl 0	1111	1111	1111	1111
ctrl 1	1111	1111	1111	1111
ctrl 2	1111	1111	1111	1111
ctrl 3	1111	1111	1111	1111
ctrl 4	1111	1111	1111	1111
ctrl 5	1111	1111	1111	1111
ctrl 6	1111	1111	1111	1111
ctrl 7	1111	1111	1111	1111
ctrl 8	1111	1111	1111	1111
ctrl 9	1111	1111	1111	0000
ctrl 10	1111	1111	1111	0000
ctrl 11	1111	1111	1111	0000
ctrl 12	1111	1111	1111	0000

ctrl 13	1111	1111	1111	0000
ctrl 14	1111	1111	1111	0000
ctrl 15	0000	1111	1111	0000
ctrl 16	0000	1111	1111	0000
ctrl 17	0000	1111	1111	0000
ctrl 18	0000	1111	1111	0000
ctrl 19	0000	1111	1111	0000
ctrl 20	0000	1111	0000	0000
ctrl 21	0000	0011	0000	0000

6.2. Functional Simulation of the Receive Path

The simulation results of the Receive Path are shown from Figure 27 to Figure 30. Three Ethernet frames are received by the MAC core, passed through FIFOs and Output Address Lookup logic, and finally transmitted by the PCIe Transmit Interface. Signal monitors probe different groups of signals from different logic blocks (refer to Figure 26).

Figure 27 shows the overview of the receiving process of the three frames in the Receive Path. In the System Signals group, the “xgmii_rx_clk” signal is the clock signal for the MAC Rx Clock Domain. The “uclk” signal is the clock signal for the PCIe User Clock Domain. In the Rx XGMII Interface group, the “xgmii_rxd” signal only shows the last frame of Ethernet data due to the range of the figure. In the MAC Rx FIFO Interface group, three Ethernet frames are received during 1252000ps to 1324000ps, 1340000ps to 1500000ps, and 1516000ps to 1644000ps respectively. The three data frames are conveyed by the “rx_axis_tdata” signal. The “rx_asix_tvalid” signal indicates which phases of the data frames are valid. In the Output Address Lookup group, the “rd_dst_mac” signal represents the destination MAC addresses which are parsed from the three Ethernet frames. The “rd_tlp_addr” signal represents the TLP routing addresses that are looked up from the address mapping table. The addresses of the second and third frames are the same, so there are no transitions between the second and third addresses.

The “rd_match” signal is asserted when the destination MAC address matches the entry in the lookup table. In the PCIe Tx Interface group, the three TLP headers are transmitted by the “tx_desc” signal. The three data payloads are transmitted by the “tx_data” signal. The TLP data is valid from the next clock cycle of the asserted “tx_ack” signal pulse to the next clock cycle of the negative edge of the “tx_dfr” signal.

Figure 28 shows the waveform of Frame 0 at the Rx XGMII Interface. The “xgmii_rxd” and “xgmii_rxc” buses convey the data and control signals as set in Table 14 and Table 15. The delimiters and interframe idle characters on the “xgmii_rxd” bus are indicated by the assertion of the “xgmii_rxc” signal. An additional CRC field is appended after the last data phase. Figure 29 shows the waveform of Frame 0 at the MAC Rx FIFO Interface. Frame 0 is buffered by the FIFO and sent to the upper logic. The FIFO waits for one clock cycle after presenting the first data phase since the upper logic is not yet ready. Figure 30 shows the waveform of Frame 0 at the PCIe Tx Interface. Frame 0 is converted to the TLP header, data payload and control signals which can potentially be sent to the PCIe Endpoint Core. The TLP header is transmitted during State 2; the TLP payload is transmitted during State 4 and State 9.

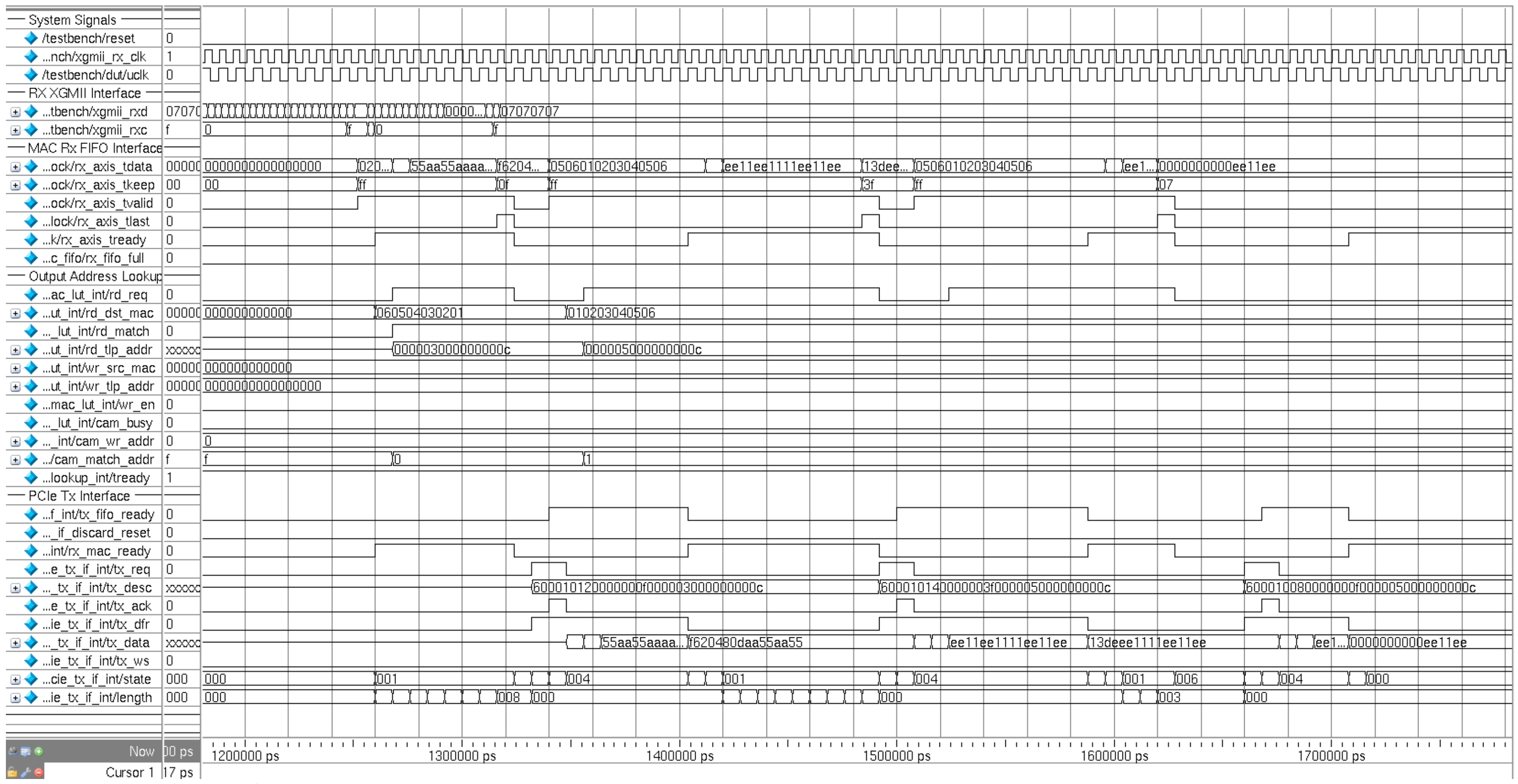


Figure 27: Waveforms of the Receive Path Signals (3 Frames)

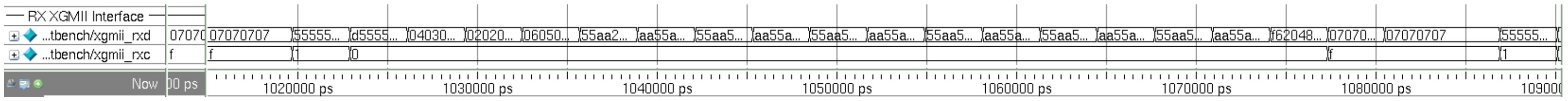


Figure 28: Waveforms of the Rx XGMII Interface Signals (Frame 0)

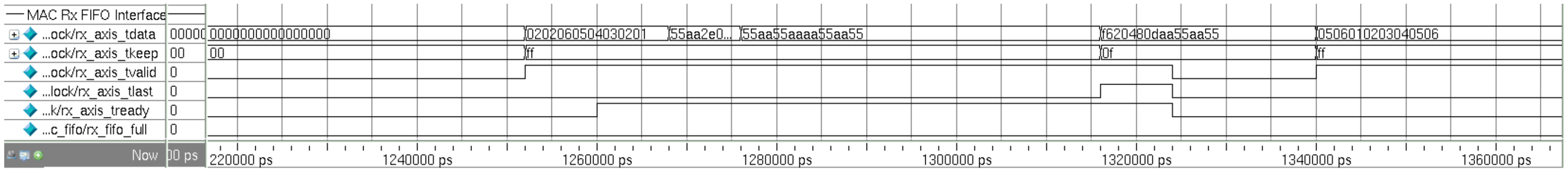


Figure 29: Waveforms of the MAC Rx FIFO Interface Signals (Frame 0)

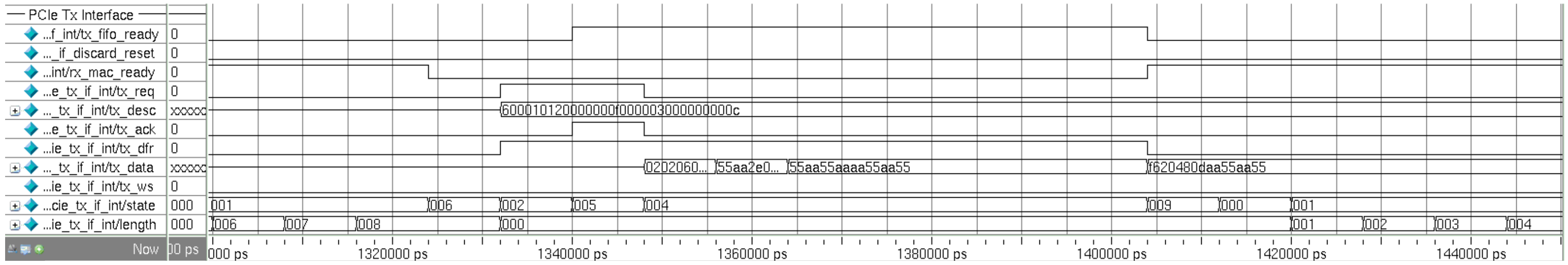


Figure 30: Waveforms of the PCIe Tx Interface Signals (Frame 0)

6.3. Functional Simulation of the Transmit Path

The simulation results of the Transmit Path are shown from Figure 31 to Figure 34. Three PCIe TLPs are received by the PCIe Receive Interface, converted to Ethernet frames, and transmitted out from the MAC core by the XGMII interface. Signal monitors probe different groups of signals from different logic blocks (refer to Figure 26 as well).

Figure 31 shows the overview of the transmitting process of the three frames in the Transmit Path. In the System Signals group, the “gtx_clk” signal is the transmit clock signal inputted from the testbench. The “xgmii_tx_clk” signal is derived from the “gtx_clk” signal and is used for transmitting the XGMII signals. In the PCIe Rx Interface group, three PCIe TLPs (with separate headers and data payloads) are received by the “rx_desc” and “rx_data” signals. The Ethernet data is conveyed by the “tx_mac_data” signal. In the MAC Tx Client Interface group, three Ethernet frames are sent from the MAC Tx FIFO to the 10G MAC core. The frame data is conveyed by the “tx_axis_tdata” signal. In the Tx XGMII Interface group, three Ethernet frames are transmitted from the MAC core through the XGMII interface. Only the first two frames are shown in the figure.

Figure 32 shows the waveform of Frame 0 at the PCIe Rx Interface. The signals with “rx_” prefix are looped back from the PCIe Tx Interface. The first TLP (with separate header and data payload) is converted to MAC Frame 0, which is represented by the signals with “tx_mac_” prefix. The frame is transmitted during State 4 and State 7. Figure 33 shows the waveform of Frame 0 at the MAC Tx Client Interface. The frame is buffered by the MAC Tx FIFO. The first data phase is repeated for two clock cycles since the MAC Core is not ready at the first data phase. Figure 34 shows the waveform of Frame 0 at the Tx XGMII Interface. The octets in frame at the Tx XGMII Interface are the same with the octets in the stimuli at the Rx XGMII Interface. The adaptor logics implement the encapsulation and decapsulation of Ethernet frames and PCIe TLPs. The simulation results show that the adaptation logics are functionally correct.

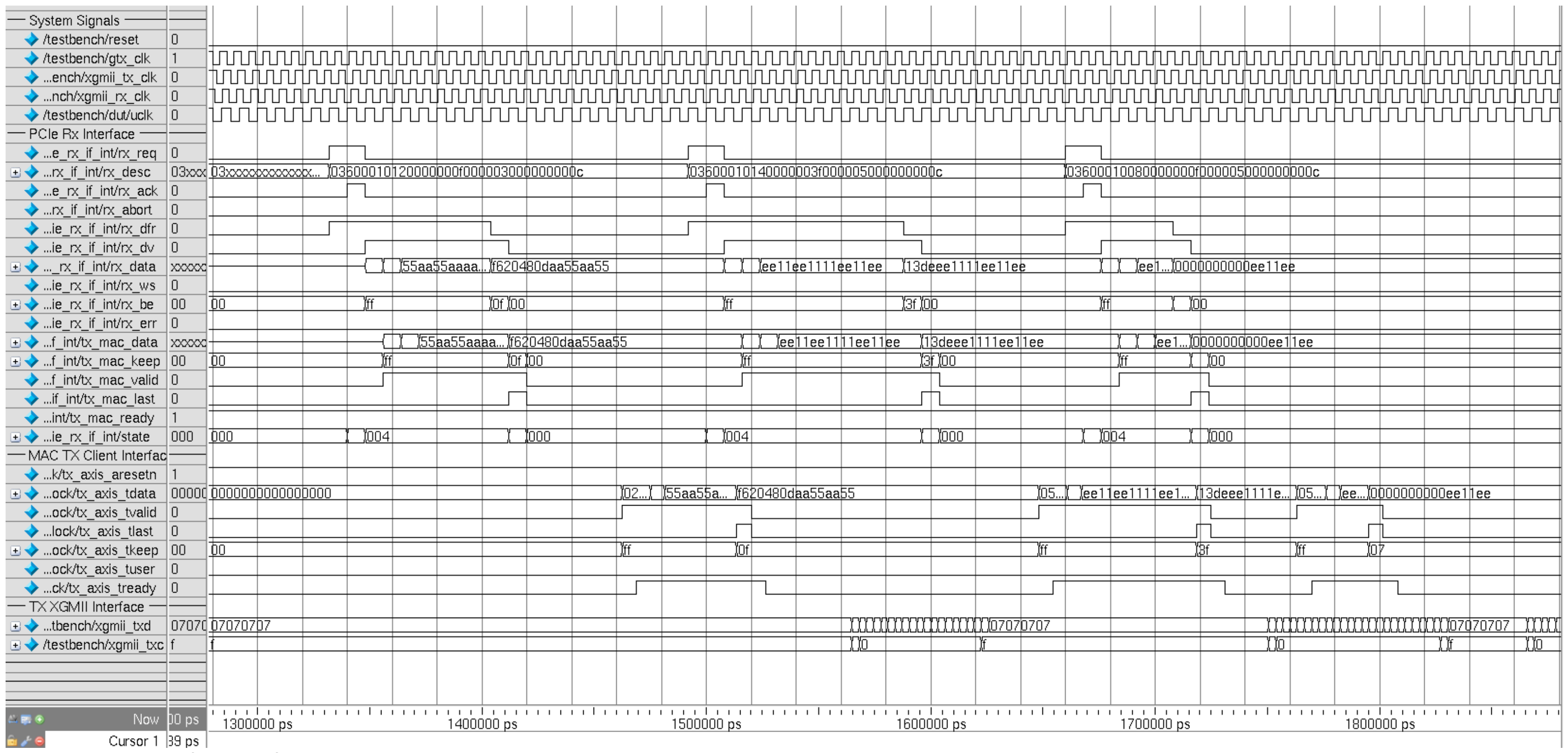


Figure 31: Waveforms of the Transmit Path Signals (3 Frames)

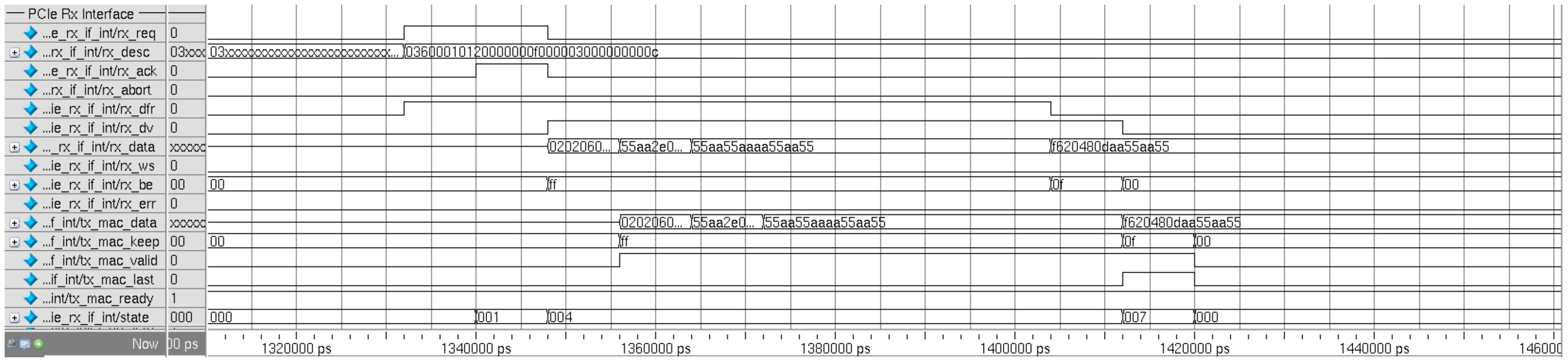


Figure 32: Waveforms of the PCIe Rx Interface Signals (Frame 0)

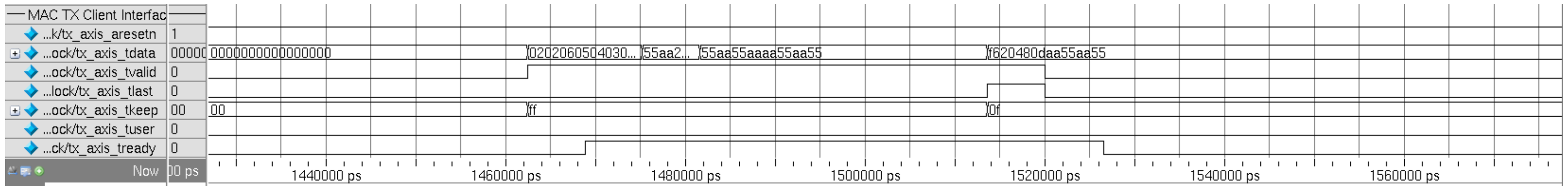


Figure 33: Waveforms of the MAC Tx Client Interface Signals (Frame 0)

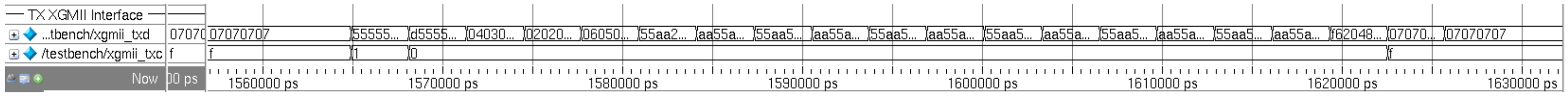


Figure 34: Waveforms of the Tx XGMII Interface Signals (Frame 0)

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The PCI Express-based Ethernet switch is conceived as a new approach of designing a low-cost and high-throughput terabit Ethernet switch. The switch leverages PCIe peer-to-peer communication protocols and off-the-shelf PCIe switch ICs to implement Ethernet switching functionality with Quality of Service (QoS).

The thesis identified the major challenges of the PCIe-based Ethernet switch solution. The issues of buffer speed, arbitration, address mapping, Quality of Service, power consumption, testing, and debugging were discussed. The thesis reviewed the basic structure of peer-to-peer communication protocols in PCI Express. Peer-to-peer communication is an efficient approach to implement high-speed packet switching within PCI Express infrastructure. Quality of service protocols and QoS-related protocols were discussed. These protocols are critical for the switch to provide guaranteed bandwidth and latency with reliable transmission. The thesis also presented possible methods of implementing Ethernet over PCI Express adaptation. The adaptor uses a layered structure to encapsulate and decapsulate Ethernet frames and PCIe packets. The Output Address Lookup logic converts the destination MAC addresses of Ethernet packets to the routing addresses/IDs of the PCIe switch. The Quality of Service in the adaptor provides differentiated services outside the Ethernet switch while coordinating with the QoS inside the PCIe switching fabric. The thesis finally presented a possible implementation of the PCIe-based Ethernet switch adaptor, which was built based on FPGA logics. The implementation enables the basic encapsulation and decapsulation of Ethernet frames and PCIe packets on the adaptor without QoS features. The interfaces and functions of the logic cores were discussed in detail.

The thesis demonstrated the functional simulation results of the adaptor using ModelSim. The results imply that the Receive Path of the adaptor is able to convert the Ethernet frames in the XGMII signals from the Ethernet Physical Layer to the PCIe headers, data payloads, and control signals to the PCIe Transaction Layer. The Receive Path is also able to create and maintain an address mapping table by the control signals from the OS (testbench). The Output Address Lookup logic is able to map the destination MAC addresses to the target PCIe routing addresses/IDs. The Transmit Path of the adaptor is able to convert the PCIe headers, data payloads, and control signals from the PCIe Transaction Layer to the Ethernet frames in the XGMII signals.

In the future, the Quality of Service features can be implemented in the adaptor. The feathers should be parameterized for the specific application and particular network condition. An embedded processor may be used in the FPGA for better memory control and data processing. When testing the adaptor logics with the PCIe Endpoint Core, the developer could write a Bus Functional Model (BFM) to simulate the adaptation logics with the PCIe bus. Furthermore, specific drivers can to be developed for the adaptor in the OS. The adaptor logics can be downloaded to the FPGA development board and plugged into a motherboard or PCIe expansion plane. Actual Ethernet traffic can be tested and evaluated in the PCIe switching system.

REFERENCES

[1] *IEEE Std 802.3ba™-2010: Part 3 Carrier Sense Multiple Access with Carrier Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Computer Society, June 22, 2010.

[2] "Huawei E2E 100G Solution," Huawei Technologies Co. Ltd., 2010.

[3] Nick McKeown, "White Paper: A Fast Switch Backplane for a Gigabit Switched Router," Business Communications Review, December, 1997.

[4] Ori Aruj, "Evolution: 20 Years of Switching Fabric," Dune Networks, last modified September 29, 2008, [Online]. Available: http://www.eetimes.com/design/communications-design/4009432/Evolution-20-years-of-switching-fabric

[5] "Performance Optimized Ethernet Switching," Cajun White Paper #1, Lucent Technologies.

[6] *PCI Express Base Specification Revision 3.0*, PCI-SIG, November 10, 2010.

[7] "PCI-SIG Announces PCI Express 4.0 Evolution to 16GT/s, Twice the Throughput of PCI Express 3.0 Technology," *PCI-SIG News Release*, November 29, 2011.

[8] *PCI Express Base Specification Revision 2.1*, PCI-SIG, March 4, 2009.

[9] "PCIe 2.0 Expansion for Low Cost GPU Acceleration, HPC and High Speed Storage," One Stop Systems (OSS).

[10] H. Jonathan Chao, "Next Generation Routers," *Proceedings of the IEEE*, vol. 90, no. 9, September, 2002.

[11] "Engineers' Guide to PCI Express Solutions," EECatalog, 2012.

[12] Ravi Budruk, Don Anderson, and Tom Shanley, *PCI Express System Architecture*, Mindshare, 2004.

[13] *Intel® X58 Express Chipset Datasheet*, Intel, November, 2009.

[14] *Gigabyte™ GA-X58A-UD9 User's Manual*, Gigabyte, 2010.

[15] Min-An Song, "System Level Assertion-Based Verification Environment for PCIPCI-X and PCIe," *International Conference on Computational Intelligence and Security*, pp. 1035-1038, December 15-19, 2007.

[16] *802.1D™-2004 IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Bridges*, IEEE Computer Society, June 9, 2004.

[17] E. Filippi , V. Innocenti , and V. Vercellone, "Address Lookup Solutions For Gigabit Switch/Router," CSELT Technical Reports, 1998.

[18] H. Jonathan Chao and Bin Liu, *High Performance Switches and Routers*, John Wiley & Sons, Inc., 2007.

[19] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, vol. 3, 2nd edition. Addison-Wesley, Don Mills, Ontario, 1998, pp. 513-558.

[20] C. Huntley, G. Antonova, and P. Guinand, "Effect of Hash Collisions on the Performance of LAN Switching Devices and Networks," *Proceedings 31st IEEE Conference on Local Computer Networks*, 2006.

[21] P. Gupta, "Routing Lookups and Packet Classifications: Theory and Practice," *Proceedings HOT Interconnects 8*, Stanford, California, August, 2000.

[22] Idriss Diouri, Jean-Philippe Georges, and Eric Rondeau, "Accommodation of Delays for Networked Control Systems Using Classification of Service," *IEEE International Conference on Networking, Sensing and Control*, Apirl 15-17, 2007.

[23] *802.1Q™-2011: IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks*, IEEE Computer Society, 2011.

[24] "IEEE 802.1p: LAN Layer 2 QoS/CoS Protocol for Traffic Prioritization," [Online]. Available: http://www.javvin.com/protocol8021P.html

[25] *IEEE Std 802.3x-1997 and IEEE Std 802.3y-1997: Specification for 802.3 Full Duplex Operation and Physical Layer Specification for 100 Mb/s Operation on Two Pairs of Category 3 or Better Balanced Twisted Pair Cable (100BASE-T2)*, IEEE Computer Society, 1997.

[26] M. Hayasaka, T. Sekiyama, S. Oshima, and T. Sakuraba, "Dynamic Pause Time Calculation Method in MAC Layer Flow Control," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2010.

[27] *802.1Qbb™-2011: IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment 17: Priority-based Flow Control*, IEEE Computer Society, 2011.

[28] "Priority Flow Control: Build Reliable Layer 2 Infrastructure," White Paper, Cisco, 2009.

[29] *IEEE 802.3an™-2006: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Computer Society, 2006.

[30] “10_100_1000 Mbps Tri-mode Ethernet MAC: Overview,” OpenCores, [Online]. Available: http://opencores.org/project,ethernet_tri_mode

[31] *LogiCORE IP 10-Gigabit Ethernet MAC v11.1 User Guide*, Xilinx, March 1, 2011.

[32] “10-Gbps Ethernet MAC MegaCore Function,” [Online]. Available: http://www.altera.com/products/ip/iup/ethernet/m-alt-10gbps-ethernet-mac.html

[33] *Avalon Interface Specifications*, Altera, May, 2011.

[34] “PCIe all-in-one (XpressRICH),” [Online]. Available: http://www.plda.com/prodetail.php?pid=202

[35] *7 Series FPGAs Integrated Block for PCI Express User Guide*, Xilinx, November 17, 2011.

[36] *IP Compiler for PCI Express User Guide*, Altera, May, 2011.

[37] John W. Lockwood, Nick McKeown, Greg Watson, Glen Gibb, Paul Hartke, Jad Naous, Ramanan Raghuraman, and Jianying Luo, “NetFPGA - An Open Platform for Gigabit-rate Network Switching and Routing,” *IEEE International Conference on Microelectronic Systems Education*, 2007.

[38] Michele Petracca, Robert Birke, and Andrea Bianco, “HERO High-speed Enhanced Routing Operation in Software Routers NICs,” *IT-NEWS*, 2008.

[39] Jeffrey Shafer and Scott Rixner, “A Reconfigurable and Programmable Gigabit Ethernet Network Interface Card,” Rice University Technical Report, 2006.

[40] *AXI Reference Guide*, Xilinx, March 7, 2011.

[41] Kyle Locke, *Parameterizable Content-Addressable Memory,* Application Note, Xilinx, March 1, 2011.

[42] *LogiCORE IP FIFO Generator v8.2 User Guide*, Xilinx, June 22, 2011.

[43] *PCI Express XpressRich Core Reference Manual*, PLDA, June 2010.