Design and Implementation of PCI express to Serial Rapid IO Bridge

A Project Report

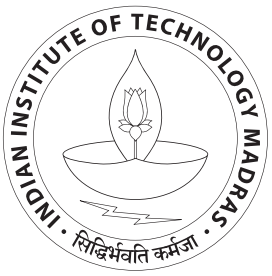
submitted by

ANAND SAI VERMA

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

under the guidance of
Dr. V. Kamakoti



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS
May 2015

THESIS CERTIFICATE

This is to certify that the thesis entitled **Design and Implementation of PCI express to Serial Rapid IO Bridge**, submitted by **Anand Sai Verma**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr.V.Kamakoti
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to express my gratitude in the form of this acknowledgement to all the people who have helped me in one way or other through the journey towards producing this thesis.

I would like to express my most sincere thanks to **Prof. V. Kamakoti** , Department of Computer Science and Engineering, who gave me the opportunity to work on this wonderful project under his guidance. His constant directions and encouragement throughout the project duration inspired me to learn a lot of new things, that will go a long way in my professional career.

In addition, a big thanks to **Sh G. S. Madhusudan**, for providing me some of the seed ideas of the project and sharing his vast experience with me. He provided valuable inputs that were instrumental in steering the project in the right direction.

I would like to thank my Co-Guide, **Prof. Nitin Chandrachoodan**, Department of Electrical Engineering, and my Faculty Advisor, **Prof. Nagendra Krishnapura**, Department of Electrical Engineering, for facilitating in all the required matters.

A bouquet of thanks to my lab mates for all the support. The atmosphere at the RISE lab has been a perfect source of motivation. Thanks in particular to Gopinathan, Ajoy, Laxmeesha and Neel.

I would like to specially thank my parents for always giving me strength and my family members for their support and cooperation throughout the course. This work would not have been possible without their best wishes and understanding.

And last but not the least; I would like to thank Director LRDE, DRDO for providing me an opportunity to pursue M. Tech. as a sponsored student at this wonderful institute called IIT Madras.

ABSTRACT

The PCI express (PCIe) and the Serial RapidIO (SRIO) architectures are high-performance, industry-standard, packet-based technologies that provide high speed interconnects. PCIe technology makes possible to connect a multitude of I/O devices on the desktop computers. The SRIO on the other hand enables chip-to-chip, board-to-board, and system-to-system communications. Both these technologies are standards-based, reliable interconnects, which are used in networking, embedded and storage applications.

Since there are a growing number of SRIO and PCIe applications, there is a need to have a bridging functionality that converts PCIe transactions to SRIO, and vice versa.

This project work involves design and implementation of bridging functionality at the logical layer level for applications requiring bridging between PCIe and SRIO devices. The project work is implemented in Bluespec System Verilog and synthesized in Xilinx ISE.

Keywords: PCI express, Serial RapidIO, Bridge

ABBREVIATIONS

AGP	Accelerated Graphics Port
ASIC	Application Specific Integrated Circuit
BSV	Bluespec System Verilog
BSW	Bluespec Development Workstation
CPU	Central Processing Unit
DSP	Digital Signal Processor
DW	Double Word
EISA	Extended Industry Standard Architecture
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IREQ	Initiator Request
IRESP	Initiator Response
ISA	Industry Standard Architecture
PCI	Peripheral Component Interconnect
PCI-X	Peripheral Component Interconnect- eXtended
PCIe	PCI express
SRIO	Serial Rapid IO
TLP	Transaction Layer Packets
TREQ	Target Request
TRESP	Target Response
VESA	Video Electronics Standards Association

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

This chapter covers the background and motivation to design the Bridge between PCI express (PCIe) and Serial RapidIO (SRIO), the design objective and the organization of the thesis.

1.1 Motivation

Over the last couple of years, the performance of processors has grown phenomenally. With this rise in the speed of processors, the need for using high speed interconnects has also increased. Typical interconnect speeds today are touching almost 40 Gbps and the speeds are further increasing. Just as we have different types of processors for different types of applications, like general purpose processors, graphics processors, DSPs etc, different types of interconnects have evolved for different applications. Two such interconnects which are of interest for the purpose of the subject Bridge are PCIe and SRIO.

PCIe was mainly designed and developed for interconnecting peripheral devices to the main processor. It was derived from the earlier PCI bus and was basically a serialised version of the same. Processor to processor connectivity was not supported by PCIe. Also, like the ethernet, the PCIe protocol is not a routable one i.e. there exists one big address space onto which mapping of all devices is done. A PCIe switch compares the address in the packet with the address space and detects the target device. PCIe is a good connectivity choice if we have a single host device and multiple slaves.

SRIO was mainly designed and developed as an embedded system interconnect for the processor fabrics. It offers high reliability operation with a low latency. The SRIO operations are highly deterministic and support various processor architectures. SRIO supports memory to memory transactions which can be carried out without any software intervention. Peer to peer transactions are supported by SRIO. Multiple processors communicate through shared memory to each other. In SRIO, the receiving end of the packet has a destination ID and the sender is identified with a soured ID. An SRIO switch identifies the port to which the packet needs to be sent with the help of the destination ID.The SRIO and PCIe follow different routing techniques.

With these basic features of both the standards and recognising the fact that the need for multi core pro-

cessing has increased like never before in today's time,there is a necessity to use these two types of interconnects together, with necessary bridging between them and that is the motivation behind this project.

1.2 Objective

The objective of this project is to develop a Bridge between PCIe and SRIO at the logical layer level.

1.3 Organization of the Thesis

Chapter 2 describes the Bridge prerequisites of the first peripheral protocol, the PCIe. In this chapter, the necessary features of the PCIe protocol including its packet structure and various fields, that have been used in the Bridge are described.

Chapter 3 describes the Bridge prerequisites of the second peripheral protocol, the SRIO. In this chapter, the necessary features of the SRIO protocol including its packet structure and various fields, that have been used in the Bridge are described.

Chapter 4 describes the architecture and implementation aspects of the Bridge. In this chapter the mapping logic to translate the PCIe packet fields to the SRIO packet fields and the reverse, has been described in detail.

Chapter 5 covers the verification aspects and results. In this chapter, the summary of test cases is given along with the simulation and synthesis reports.

Chapter 6 is about conclusion and possible future work on the project.

CHAPTER 2

Prerequisites of the Peripheral Protocol : PCI express

PCI express (PCIe) is the third generation high performance, general purpose I/O interconnect standard defined for a wide variety of computing and communication platforms. Before this standard, there were first generation buses like ISA, EISA and VESA and the second generation buses like PCI, AGP and PCI-X.

The main requirements and features for this third generation I/O interconnect were as follows:

- To have a uniform I/O architecture for desktop, mobile, workstation, server, communications platforms, and embedded devices
- To reduce cost of PCI systems
- To support multiple platform interconnections
- To provide modular form factors
- To provide high bandwidth
- To provide aggregated lanes and scalable performance
- To provide hot plug and hot swap features etc.

2.1 PCIe Link

A typical dual-simplex channel between two PCIe components consists of two, low-voltage, differentially driven signal pairs. One pair is for transmit and the other one for receive signals as shown in Figure 2.1

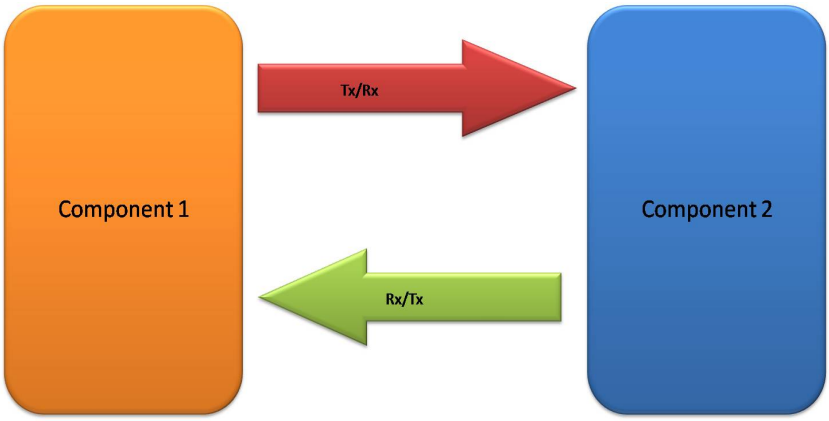


Figure 2.1: A Typical PCI express Link

The main features of a PCIe link are:

- There are two unidirectional links both of which are differential. One is a transmit pair and the second one is receive pair. High data rates are achieved using encoding schemes on embedded clocks.
- After initialisation, each link operates at the supported signalling frequency, the values for which are given as in Table 2.1 for the three PCIe generations

PCIe Generation	Raw Bandwidth(Gigabits/second/Lane/Direction)
First	2.5
Second	5.0
Third	8.0

Table 2.1: PCIe Generations and Bandwidths

- Every link must have at least one lane.
- Multiple lanes can be used to increase bandwidth. Number of lanes should be same on both sides.

2.2 PCIe Fabric Topology

There are point-to-point links to interconnect a set of components in a PCIe fabric. Generally, the following components are present in a single instance of the fabric: a Root Complex (RC), multiple endpoints (I/O devices), a Switch, and a PCIe to PCI/PCI-X Bridge. A root complex is a device that connects CPU and memory subsystem to PCIe fabric and it can communicate with an endpoint. Endpoints are devices other than root complex and switches, that are requesters or completers of PCIe transactions and can communicate with a root complex. PCIe links connect all these components together. An example hierarchy is shown in Figure 2.2

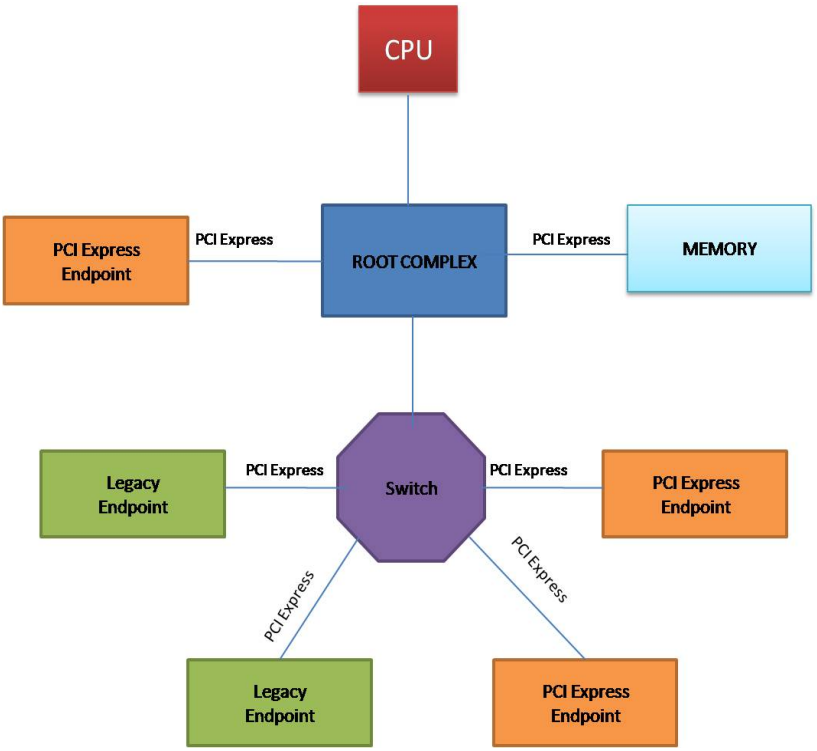


Figure 2.2: PCIe Fabric Topology

2.3 PCIe Layers

In PCIe, to communicate information between the transmitting component and the receiving component, packets are used. These packets are generated at the Transaction Layer. Additional information is appended to the packets as they pass through the Data Link Layer and the Physical Layer.The reverse process takes place at the receiving side i.e. the additional information appended at the transmit side is stripped at the Physical and Data Link layers and the Transaction Layer Packet is retrieved. Figure 2.3 illustrates this.

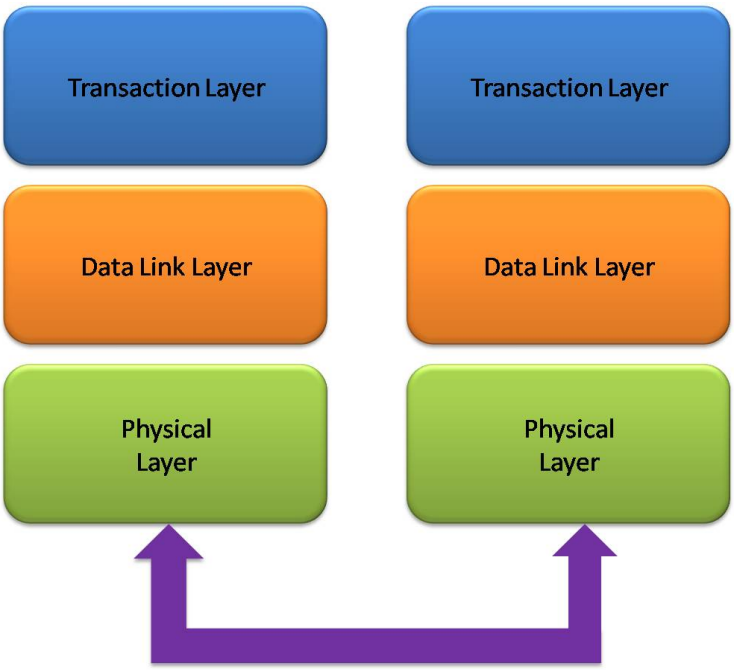


Figure 2.3: PCIe Layers

The layers are described in brief below.

Transaction Layer

This is the topmost layer of the hierarchy. The Transaction Layer Packets (TLPs) are created and disassembled here. Split transactions are implemented for any request packet that requires a response. A unique identifier is attached to each packet with the help of which the response packets reach the correct originator.

Data Link Layer

This is the intermediate layer between the Transaction Layer and the Physical Layer. It is mainly responsible for error correction and detection and link management functions. On the receiving side, The Data Link Layer checks the integrity of received TLP and forwards it to the Transaction Layer for further processing.

Physical Layer

The information received from Data Link Layer is serialized and transmitted at a device compatible frequency with the help of the physical layer. This layer has necessary circuitry of this operation like the driver buffers, parallel-to-serial and serial-to-parallel converters etc.

2.4 Address Spaces, Transaction Types and Usage

The information transfer is based on the TLPs formed at the Transaction Layer. There are mainly four address spaces and different types of transactions can occur within each as listed below in Table 2.2

Address Space	Transaction Types	Basic Usage
Memory	Read,Write	Transfer data to/from memory mapped locations
I/O	Read,Write	Transfer data to/from I/O mapped locations
Configuration	Read,Write	Device Function /Configuration setup
Messsage	Baseline	General purpose messaging

Table 2.2: Transaction Types

A brief description of the transaction types is given below

Memory Transactions

Following types of Memory Transactions are possible:

- Read Request
- Read Request Completion
- Write Request
- Atomic Request/Completion

Two different address formats are possible i.e. 32-bit address (Short address format) or 64-bit address (Long address format)

I/O Transactions

I/O transaction have a 32 bit (Short) format. Following I/O Transactions are possible:

- Read Request/Read Completion
- Write Request/Write Completion

Configuration Transactions

These are used to discover device capabilities, program features and check status in the PCIe configuration space. These are of the following types:

- Read Request/Read Completion
- Write Request/Write Completion

Message Transactions

Messages are used to communicate events between devices. Transactions are packet transmissions to exchange information between requester and completer. PCIe transactions can be categorised into Non Posted and Posted Transactions.

For Non Posted Transactions, a requester transmits a TLP request packet to a complete. Later, the completer returns a TLP completion packet back to requester. The purpose of the completion TLP is to confirm to the requester that the completer has received the request TLP. Non Posted write TLPs contain data in the write request TLP.

For Posted Transaction, a requester transmits a TLP request packet to a completer. The completer however does not return a completion TLP back to a requester. Posted transactions may or may not contain data in the request TLP. The list of transactions categorised into posted and non posted transaction is as given in Table 2.3.

Transaction Type	**Posted or Non Posted**
Memory Read	Non Posted
Memory Write	Posted
Memory Read Lock	Non Posted
IO Read	Non Posted
IO Write	Non Posted
Configuration Read	Non Posted
Message	Posted

Table 2.3: Transaction Type Categories

2.5 PCIe Packet Format

A typical PCIe TLP is shown in Figure 2.4. It consists of a TLP header, a data payload (for some types of packets), and an optional TLP digest field.

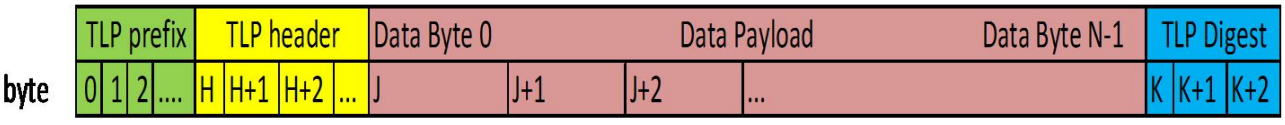


Figure 2.4: TLP Serial View

Figure 2.5 shows a more detailed view of the TLP.

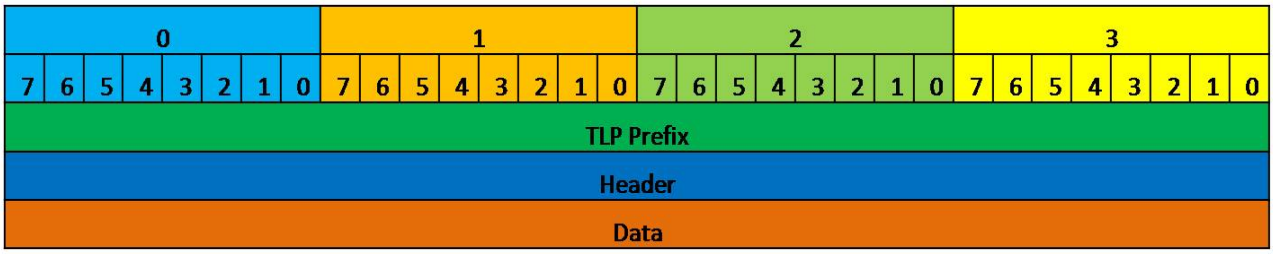


Figure 2.5: TLP Format

The information is transferred as a serialized stream of bytes as shown in Figure 2.4. Information is transmitted/received at the byte level over the interconnect with the leftmost byte of the TLP transmitted/received first.

Depending on the type of packet, the header for that packet includes some of the following types of fields:

- Format of the packet
- Type of the packet
- Length for any associated data
- Transaction descriptor
- Address/routing information
- Byte Enables
- Message encoding
- Completion status

2.6 Packet Header Fields

The structure of TLP and its various fields are shown in Figure 2.6 and Figure 2.7 respectively.

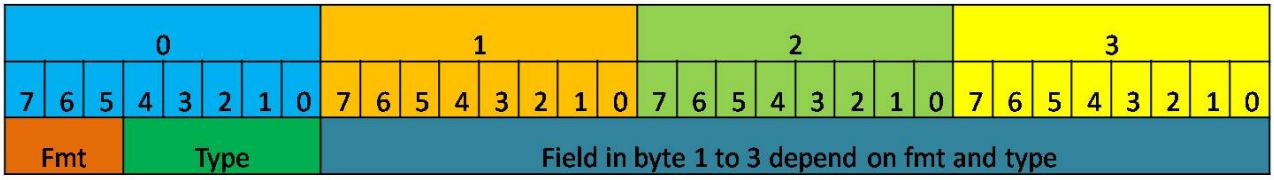


Figure 2.6: TLP Structure

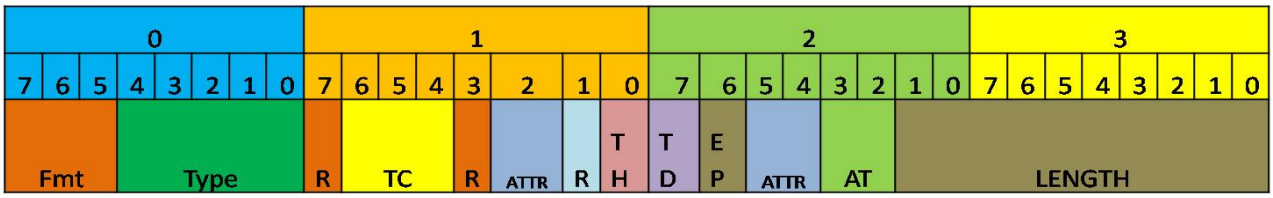


Figure 2.7: TLP Header Fields

As shown in Figure 2.6 , the Format (Fmt) and type fields which are bits 7:5 and bits 4:0 of byte 0 respectively, are used to find out the remaining fields of the packet.

The relation between the Fmt field and the TLP format is shown in Table 2.4

Fmt[2:0]	**TLP Format**
000	3DW header, no data
001	4DW header, no data
010	3DW header, with data
011	4DW header, with data

Table 2.4: Fmt field values

The size of the TLP Header is determined with the help of the Fmt and Type field.

In addition to Fmt and Type, the TD and length fields indicate the data payload length (for data packets) and with the help of these fields, the overall size of the packet can be calculated. The values of Fmt[2:0] and Type[4:0] fields are shown in Table 2.5

TLP Type	**Fmt**	**Type**	**Description**
MRd	000,001	00000	Memory Read Request
MWr	010,011	00000	Memory Write Request
IO Rd	000	00010	IO Read request
IO Wr	010	00010	IO Write Request
Cpl	000	01010	Completion without data
CplD	010	01010	Completion with Data

Table 2.5: Fmt and Type Field Encodings

The other fields of the packet are as follows:

- Traffic Class (TC[2:0]), are the bits [6:4] of byte 1
- TLP Processing Hints(TH field (1b)), indicates the presence of TH in the TLP header and is present at bit 0 of byte 1
- Attribute field (Attr[1:0]), present at bits-5, 4 of byte 2 and Attr[2] present at bit 2 of byte 1
- TLP digest (TD(1b)), indicates presence of TD in the form of a single DW at the end of the TLP and is present at bit 7 of byte 2
- Error Poisoned (EP), indicates the TLP is poisoned and is present at bit 6 of byte 2
- Length of data payload (Length[9:0]), are bits 1:0 of byte 2 concatenated with bits 7:0 of byte 3, The length of the data payload is associated with the length field value as shown in Table 2.6

Length[9:0]	TLP Data Payload Size(DW)
00 00000001	1
00 00000010	2
.	.
.	.
11 11111111	1023
00 00000000	1024

Table 2.6: Length Field and the Payload Length

2.7 Address Based Routing Rules

Memory and I/O Requests use Address routing.There can be either a 64-bit format that has 4 DW header (Figure2.8) or a 32-bit format that has 3 DW header (Figure 2.9).

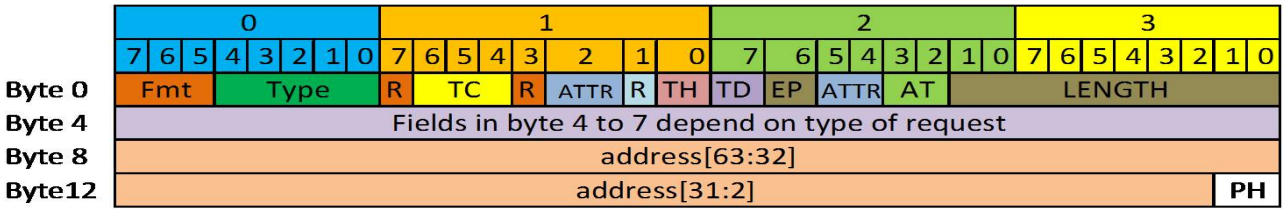


Figure 2.8: 64 Bit Addressing

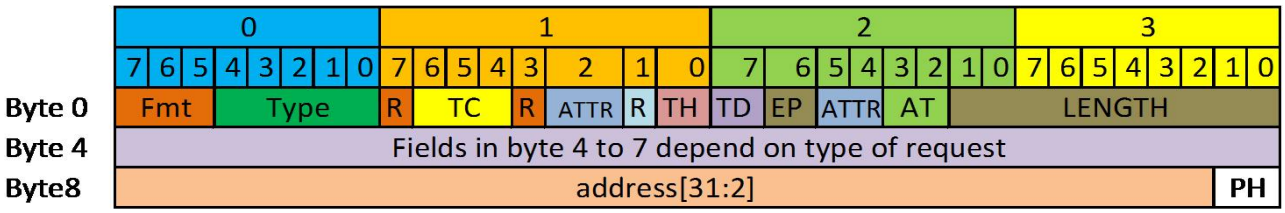


Figure 2.9: 32 Bit Addressing

The Address field is mapped to the TLP header as shown in Table 2.7

Address Bits	32 bit addressing	64 bit addressing
63:56	Not applicable	Bits 7:0 of Byte 8
55:48	Not applicable	Bits 7:0 of Byte 9
47:40	Not applicable	Bits 7:0 of Byte 10
39:32	Not applicable	Bits 7:0 of Byte 11
31:24	Bits 7:0 of Byte 8	Bits 7:0 of Byte 12
23:16	Bits 7:0 of Byte 9	Bits 7:0 of Byte 13
15:8	Bits 7:0 of Byte 10	Bits 7:0 of Byte 14
7:2	Bits 7:2 of Byte 11	Bits 7:2 of Byte 15

Table 2.7: Address Field Mapping

2.8 First/Last DW Byte Enables Rules

Byte 7 of the header contains information about the Byte Enables, as shown in Figure 2.10

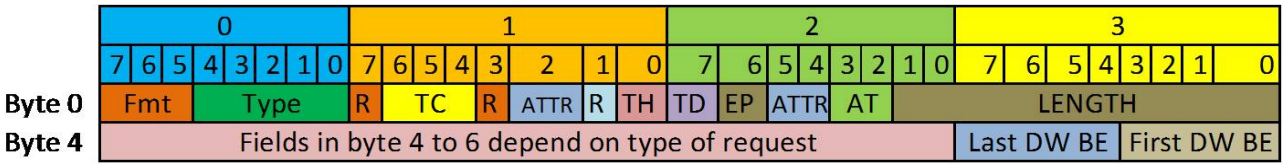


Figure 2.10: Byte Enable location in the header

The Byte Enable rules are as follows

- If the Length field value is 1 DW, then the 1st DW Byte Enable field implies 1111 and the Last DW Byte Enables implies 0000
- If the Length field is more than 1 DW, then the 1st DW and the Last DW Byte Enables imply 1111.
- Byte Enables for the first DW are indicated by the 1st DW BE[3:0]
- Byte Enables for the last DW are indicated by the last DW BE[3:0]
- For each bit of the Byte Enables fields:
  - A byte of data is not written/read if the corresponding bit of the byte enable field is 0
  - A byte of data is written/read if the corresponding bit of the byte enable field is 1

The location of the Byte Enable affects the data byte as shown in Table 2.8

Byte Enables	Header Location	Affected Data Byte
1st DW BE[0]	Bit 0 of Byte7	Byte 0
1st DW BE[1]	Bit 1 of Byte7	Byte 1
1st DW BE[2]	Bit 2 of Byte7	Byte 2
1st DW BE[3]	Bit 3 of Byte7	Byte 3
Last DW BE[0]	Bit 4 of Byte7	Byte N-4
Last DW BE[1]	Bit 5 of Byte7	Byte N-3
Last DW BE[2]	Bit 6 of Byte7	Byte N-2
Last DW BE[3]	Bit 7 of Byte7	Byte N-1

Table 2.8: The Effect of Byte Enable Bits

2.9 The Completion Packet

When a device receives a Read Request TLP, it responds with a Completion TLP. The response packet looks like Figure 2.11

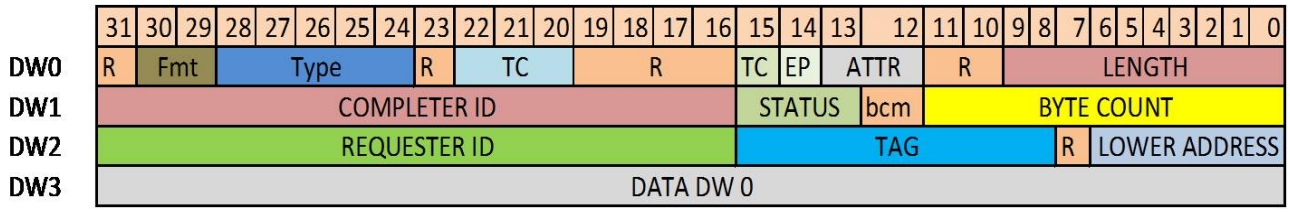


Figure 2.11: Typical Response Packet

The Fmt and Type fields are as for earlier packet types. The Byte Count field tells the number of valid payload bytes in the packet. The Completer ID field identifies the sender of the packet and the Requester ID identifies the receiver of this packet. The length field tells the length of data present in the packet.

CHAPTER 3

Prerequisites of the Peripheral Protocol: Serial RapidIO

RapidIO interconnect standard was primarily developed for embedded systems applications. It is being used these days to interconnect processors, general purpose computing platforms, memories, storage devices. The standard promises performance levels upto 60 Gbit/s for board to board and chip to chip communications. There are two variants of the RapidIO architecture: Parallel and Serial. The parallel interface is used for processor and system connectivity and the serial one is used for serial control, DSP and serial backplane applications. Both variants have the same addressing mechanisms, transactions and programming models. Serial variant has been used for the current project.

Layered architecture approach is followed in RapidIO, to keep it backward compatible and to allow scalability for future requirements. Analogous to the PCIe, there are three layers in the SRIO hierarchy viz. the Logical Layer, the Transport Layer and the Physical Layer. The packet formats and the protocols are defined in the Logical Layer. The Transport Layer is the middle layer and provides necessary routing information for the movement of the packet. The bottom most layer i.e. the Physical Layer is responsible for handling the electrical characteristics at the device level and error control etc.

3.1 SRIO Transaction Flow

Request and response transactions form the basis of SRIO operation. The communication between endpoints happens through packets. A request transaction is generated by the initiator and is transmitted towards the target. A response packet is generated by the target and is sent to the initiator to complete the transaction.

Packets contain important fields that are responsible for ensuring the delivery of required information to the target in a reliable manner. Typically, there exists a SRIO fabric that is a group of switches which provides system connectivity to link all the endpoints together. There are a group of control symbols that manage the transaction flow in the interconnect.

The flow of transctions in a SRIO system can be shown with the help of Figure 3.1

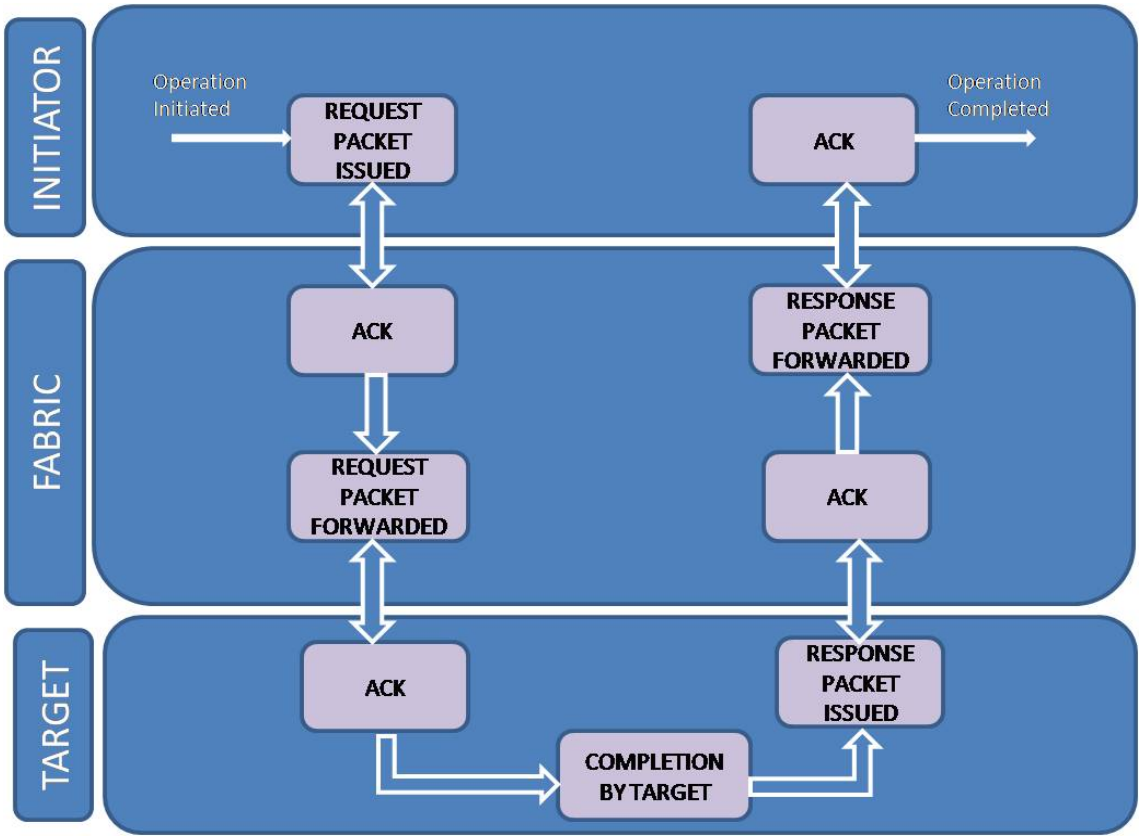


Figure 3.1: SRIO Transaction Flow

The operation begins when a request transaction is generated by the initiator. The fabric receives the packet and acknowledges it with a control symbol and forwards it towards the target device.

A response transaction is then generated by the target a complete the request which is passed onto the fabric and this response reaches the initiator with the help of the control symbols. The operation is complete when the response is acknowledged at the initiator.

3.2 Read Operations

The read operation is used by a processing element to read data from a specified address. It consists of the NREAD and RESPONSE transactions as shown in Figure 3.2. Data of requested size is returned in the response packet.

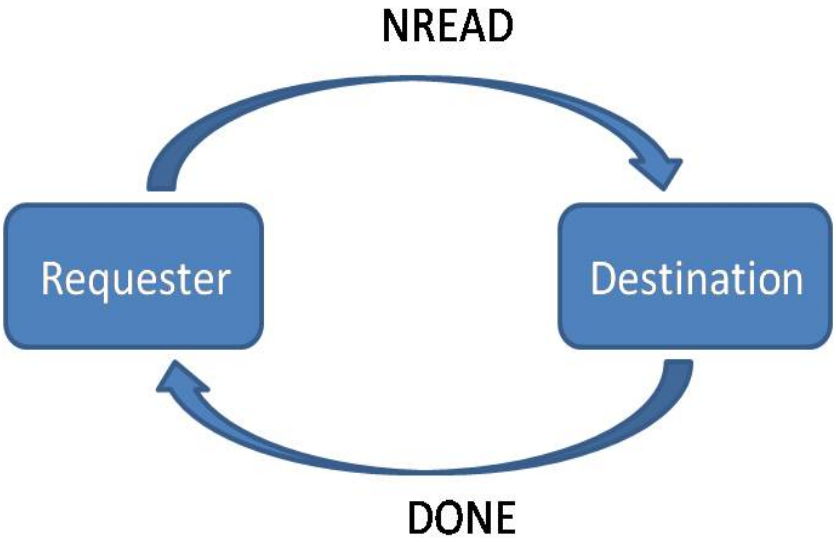


Figure 3.2: NREAD Operation

3.3 Write and Streaming-Write Operations

Whenever data is required to be written to a specified address, write and streaming-write(SWRITE) operations are used. These comprise of the NWRITE and SWRITE transactions. In an NWRITE transaction, multiple double-word, word, half-word and byte writes are allowed with properly padded and aligned data payload. The SWRITE transaction has less header overhead and is a double-word-only version of the NWRITE. There are no responses for NWRITE and SWRITE transactions, and hence sender has no information as to when the transaction has completed at the destination. These transaction can be shown as in Figure 3.3

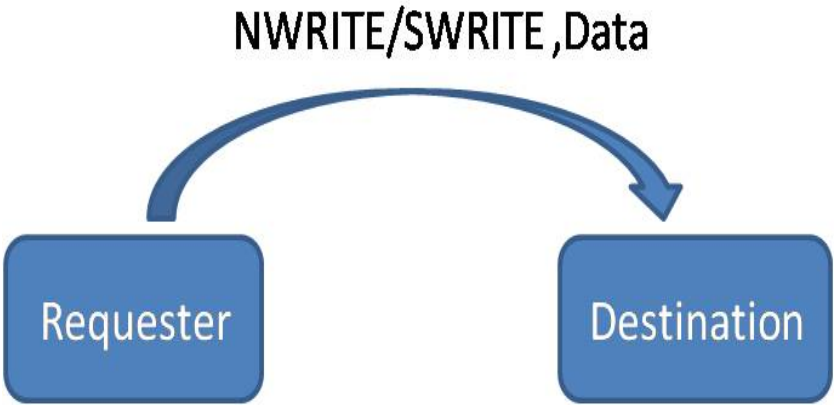


Figure 3.3: NWRITE and SWRITE Operation

3.4 Write with Response Operations

The write with response operation is identical to the write operation with only difference that it receives a response to notify the sender that the write has completed at the destination. These transaction comprise of the NWRITE(R) and RESPONSE transactions. This is shown in Figure 3.4

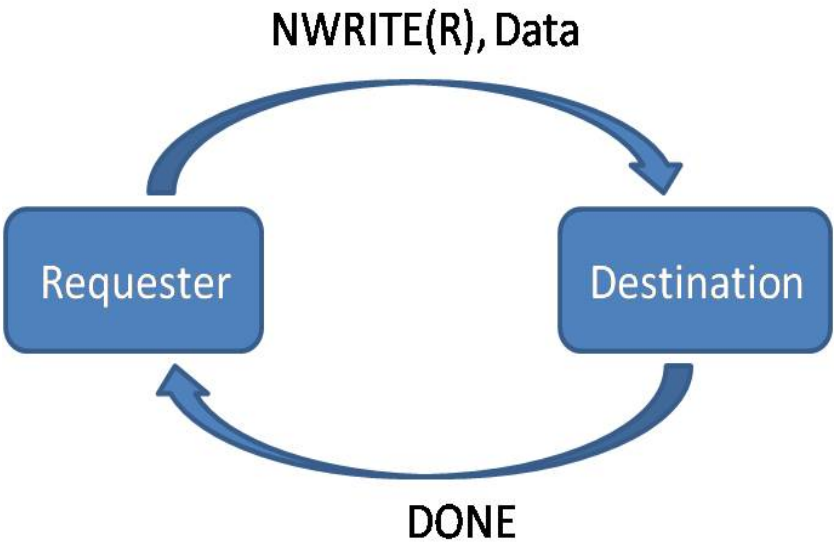


Figure 3.4: WRITE with Response Operation

3.5 Atomic (Read-Modify-Write) Operations

The read-modify-write operation is used by processing elements to carry out synchronization using non-coherent memory. It consists of the ATOMIC and RESPONSE transactions (typically a DONE response). The atomic operation is a read and write operation combined. Here, the destination reads the data at the specified address, returns the read data to the requester, performs the required operation to the data, and then writes the modified data back to the specified address. No intervening activity is allowed to that address. The operation is as shown in Figure 3.5

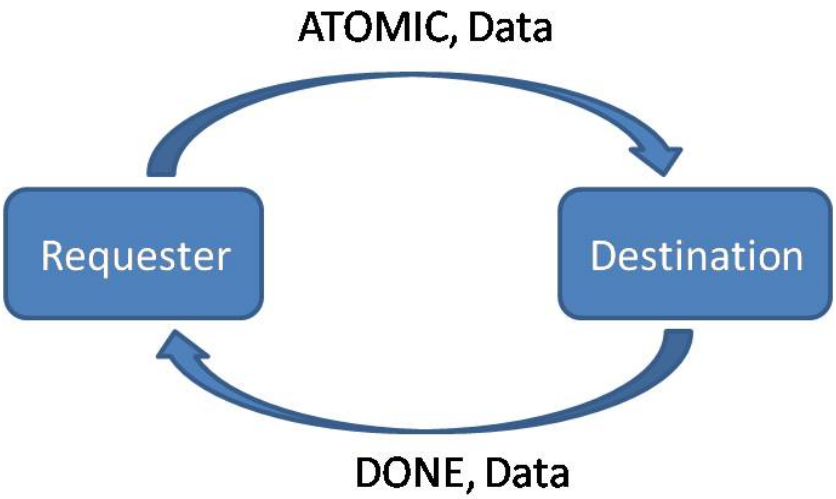


Figure 3.5: ATOMIC(Read Modify Write) Operation

3.6 Packet Formats

A typical SRIO packet can be represented as shown in Figure 3.6

Figure 3.6: Typical SRIO Packet

The packet consists of the following fields:

- **Ftype** : It is the Format type, which is as a 4-bit value. The first four bits in the logical packet stream indicate Ftype value .
- **Wdptr**: Word pointer.It is used in conjunction with the data size (rdsize and wrsize) fields
- **Rdsize** : It is the Data size for read transactions, used in conjunction with the word pointer (wdptr) bit
- **Wrsize**: It is the Write data size for sub-double-word transactions, used along with the word pointer (wdptr) bit.
- **Rsrv** : Reserved
- **SrcTID**: The packets transaction ID

- **Transaction** : It is also called type or Ttype.It indicates the specific transaction within the format class to be performed by the recipient.

- **Extended Address**: Optional. Specifies the most significant 16 bits of a 50-bit physical address or 32 bits of a 66-bit physical address.

- **Xamsbs**: These are the most significant bits of the extended address. The address specified by the address and extended address fields can be extended by 2 bits using this field.

- **Address** : Indicates the address value

The Ftype and Ttype fields of a SRIO packet determine the packet type as per the Table 3.1

Ftype	**Ttype**	**Packet Type**
0010	0100	NREAD
0101	0100	NWRITE
	0101	NWRITE_R
0110	-	SWRITE
1000	0000	MAINTENANCE(CONFIG READ REQ)
	0001	MAINTENANCE(CONFIG WRITE REQ)
	0010	MAINTENANCE(CONFIG READ RESP)
	0011	MAINTENANCE(CONFIG WRITE RESP)
1010	-	DOORBELL
1011	-	MESSAGE
1101	0000	RESPONSE WITHOUT DATA
	1000	RESPONSE WITH DATA

Table 3.1: SRIO Packet Types

3.7 Packet Details for SRIO

The bit positions for various fields for different packet types as being used for the Bridge are explained as follows:

The NREAD packet has the structure shown in Figure 3.7. The packet is identified with the help of the Ftype and Ttype fields. The address field length used for this project is 50 bits of which the first 24 bits are received in the first clock cycle and the remaining 26 bits in the second clock cycle. The other fields are as per the packet format explained in Section 3.6. Priority, Destination Id, Source Id fields of the packet belong to the Transport Layer.

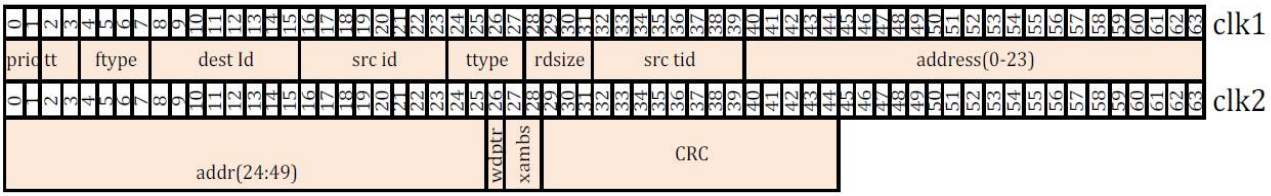


Figure 3.7: NREAD Request Packet (Ftype=2)

The NWRITE packet has the structure shown in Figure 3.8. The packet is identified with the help of the Ftype and Ttype fields. The address field length is 50 bits of which the first 24 bits are received in the first clock cycle and the remaining 26 bits in the second clock cycle. The NWRITE packet has a fixed data length of 64 bits of which the first 35 bits are received in the second clock cycle and the remaining 29 bits are received in the third clock cycle. The other fields are as per the packet format explained in Section 3.6. Priority, Destination Id, Source Id fields of the packet belong to the Transport Layer.

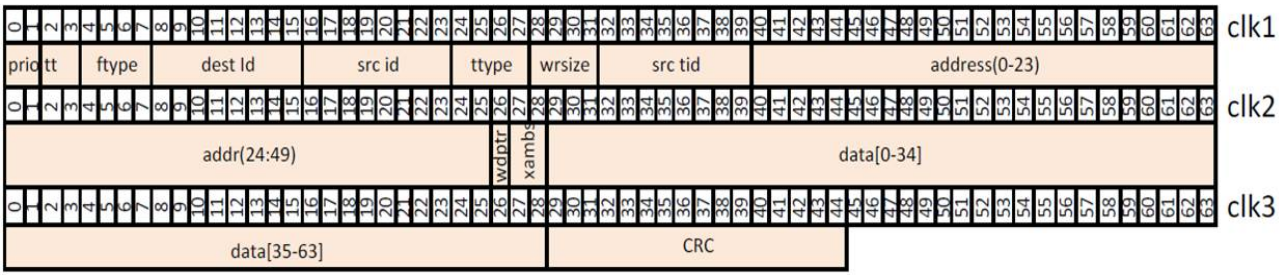


Figure 3.8: NWRITE Request Packet (Ftype=5)

The SWRITE packet has the structure similar to the NWRITE packet and is shown in Figure 3.9. Here a sample packet is shown for 3 data cycles. The difference here is that the data keeps continuously coming till the eof signal is received. Other difference here is that the address starts immediately after Srcid field. The other fields have the same meaning as explained earlier.

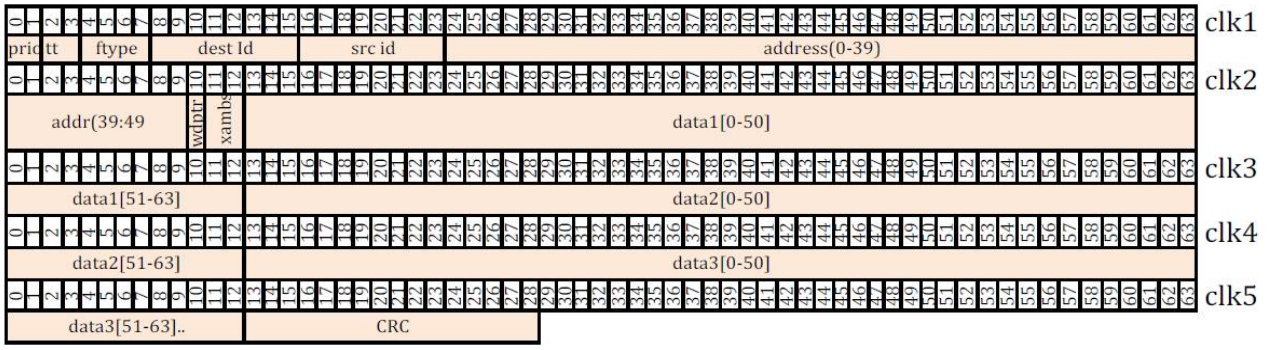


Figure 3.9: SWRITE Request Packet (Ftype=6)

The Response with data packet looks like as shown in Figure3.10 . Response with data packet has a fixed data length of 64 bits which start after the target id field and continues in the second clock cycle as well.

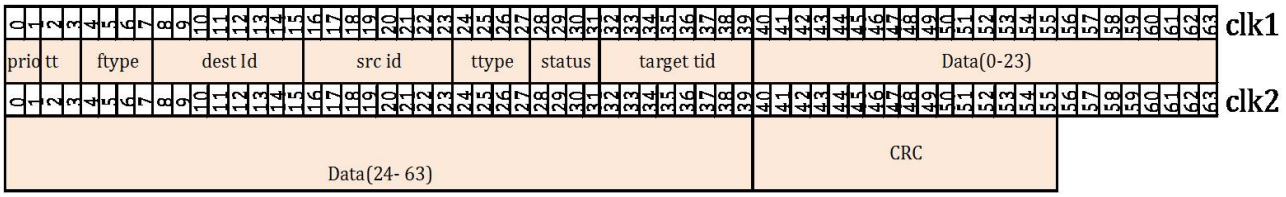


Figure 3.10: Response with Data packet (Ftype=13,Ttype=8)

Response without data packet is similar to with data case and since there is no data, the packet gets delivered in one clock cycle itself. The packet is shown in Figure 3.11

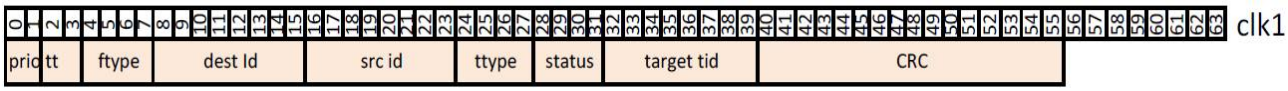


Figure 3.11: Response without Data packet (Ftype=13,Ttype=0))

CHAPTER 4

Bridge Architecture and Implementation

The Bridge was developed at the logical layers of PCIe Gen 3.0 and SRIO Gen 3.0 that have been explained in Chapter 2 and Chapter 3. The Xilinx PCIe and SRIO interfaces were used for carrying out the mapping.

The user interface of SRIO contains four ports - Initiator Request, Initiator Response, Target Request, and Target Response. A packet is issued or consumed using one of these four ports. The SRIO wrapper feeds these ports for packet generation and consumption. Similarly, the PCIe interface has two ports that are the transmit and receive ports. These two ports are responsible for the generation and consumption of packets through the PCIe interface. The PCIe wrapper feeds these two ports for packet generation and consumption.

To carry out the bridging, a mapping engine has been designed for interfacing between the four ports of SRIO and the two ports of PCIe.

The SRIO and PCIe interfaces are linked to other devices through physical lanes. The number of physical lanes can vary on both sides and that decides the bandwidth of the link. Typical signalling speed for 1 lane of PCIe Gen 3.0 are 8 GHz and that of SRIO Gen 3.0 is about 10 GHz.

The block diagram depicting the architecture with ports of both the interfaces is shown in Figure 4.1

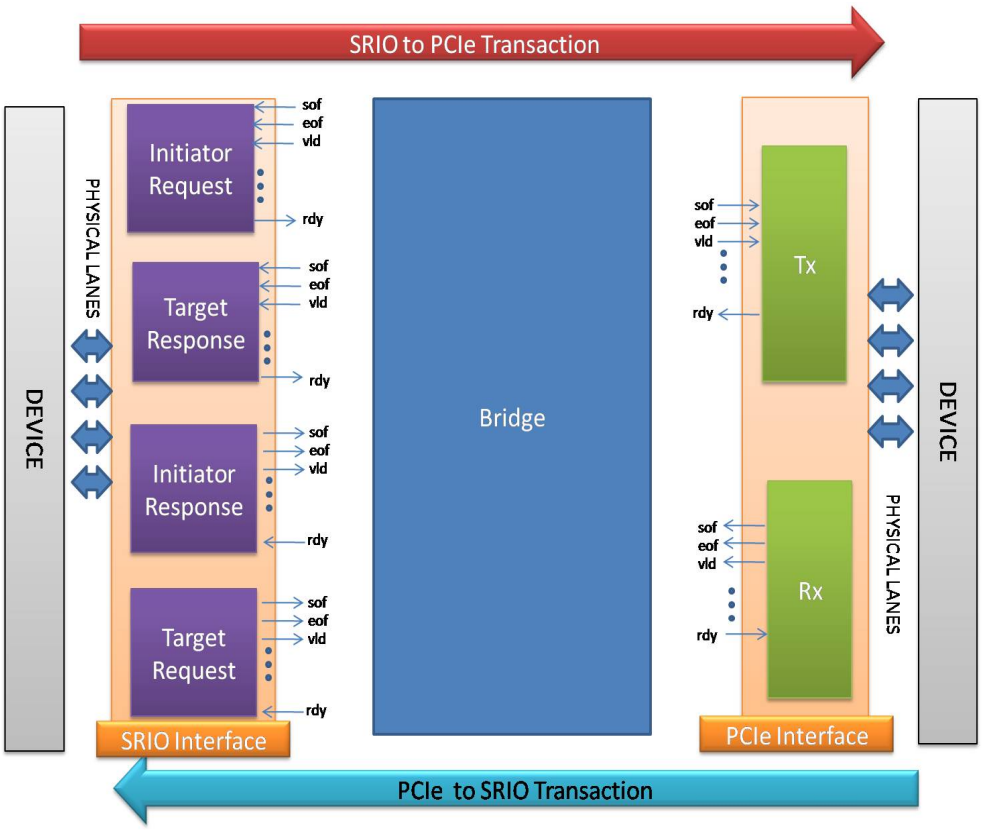


Figure 4.1: The SRIO and PCIe Interfaces of the Bridge

To carry out the mapping of the signals, the direction of signals was noted with the help of incoming and outgoing signals for both the sides.

The Logical Layer of both the sides is comprised of the following components :

- A Receiver module that accepts incoming packets.
- A Transmitter module that transmits the outbound packets.

The Initiator Request and Target Response ports on the SRIO sides form the the transmit signals to the link and the Initiator Response and Target Request form the receive signals from the link. Similarly, the PCIe has the Transmit and Receive ports. The mapping principle between the two sides is illustrated in the Figure 4.2

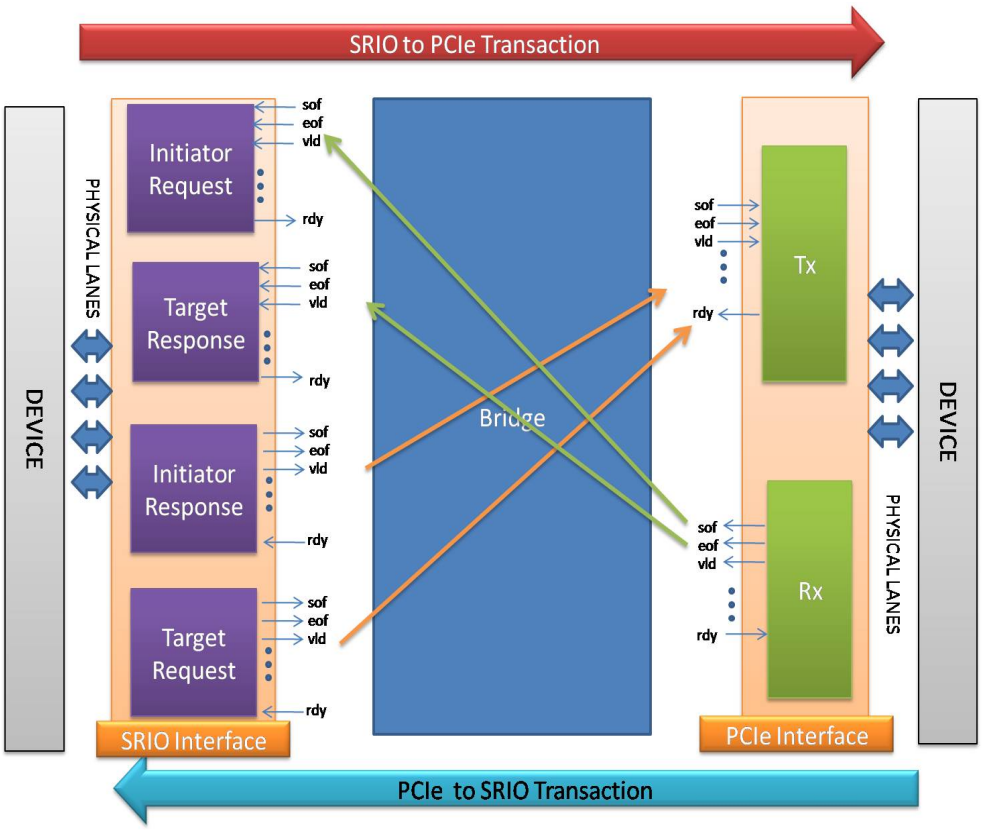


Figure 4.2: Mapping Directions of the Bridge

As shown in Figure 4.2 , the Initiator Request and Target Response ports of the SRIO interface have been mapped to the Receive port of the PCIe interface and the Initiator Response and Target Request ports of SRIO have been mapped to the Transmit port of the PCIe interface. Since the mapping function is between two ports of SRIO to one port of PCIe, a mapping logic for transmitting the signals was worked out as explained below.

At a given instance of time, the SRIO port serves either as Initiator or Target, and therefore the ready signal of only one of the two is active. Therefore, for transmitting from PCI Rx port to SRIO Initiator Request/ Target Response ports, the ready signal is checked for being active. As soon as the PCIe get a start of packet signal, it checks whether the Initiator Request or Target Response Port is ready to receive and accordingly starts transmitting the data to the relevant port on the SRIO side.

For the SRIO Initiator Response / Target Request ports to PCIe Tx port mapping, the ready from the PCI Tx port is a common signal for both the SRIO ports. However, at a given instance of time, the port is active as either Initiator Response or Target Request, so it is checked which of Initiator or Target is ready to transmit and the control is given to that port for transmission and the data can be sent as the ready signal of PCIe Tx port goes active.

Based on the directions of signals from the ports, the input output ports for the Bridge can be decided as

shown in Figure 4.3

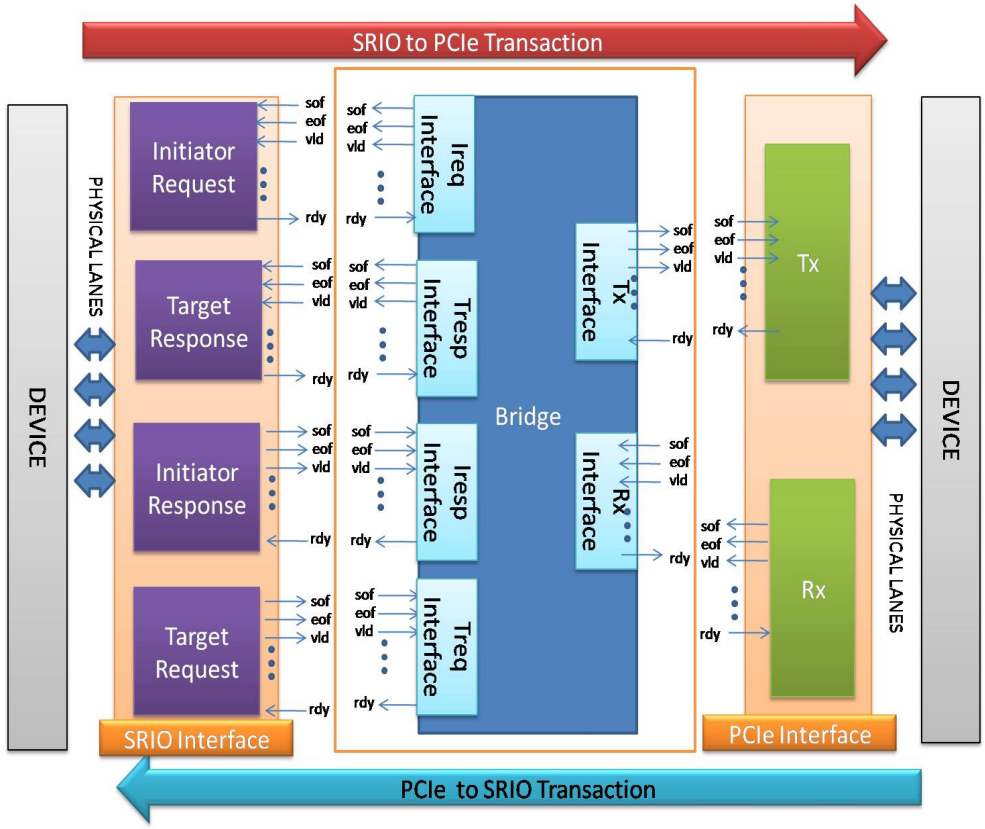


Figure 4.3: Input Output Ports of the Bridge

The signals of both the sides were identified for one to one mapping. The signals listed out under the Initiator Request and Initiator Response port of SRIO are shown in Figure 4.4

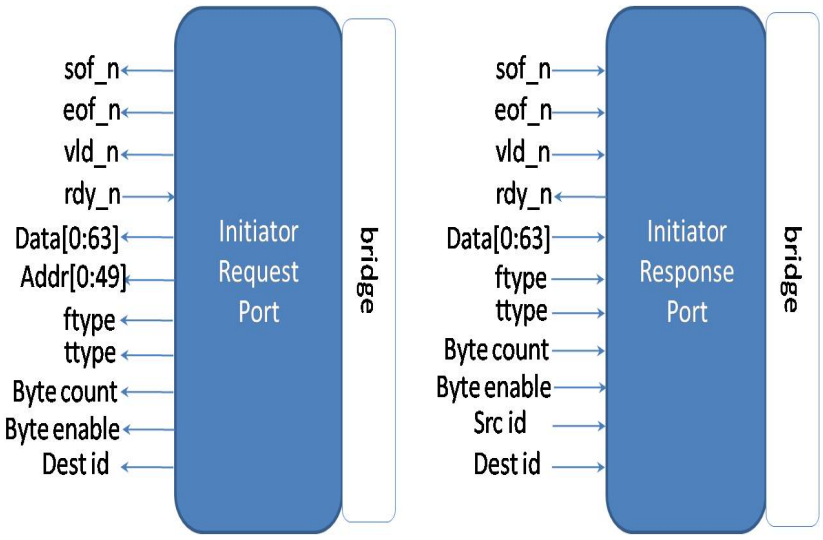


Figure 4.4: SRIO Initiator Port Signals

Similarly the signals that have been listed under the Target Request and the Target Response ports are shown in Figure 4.5

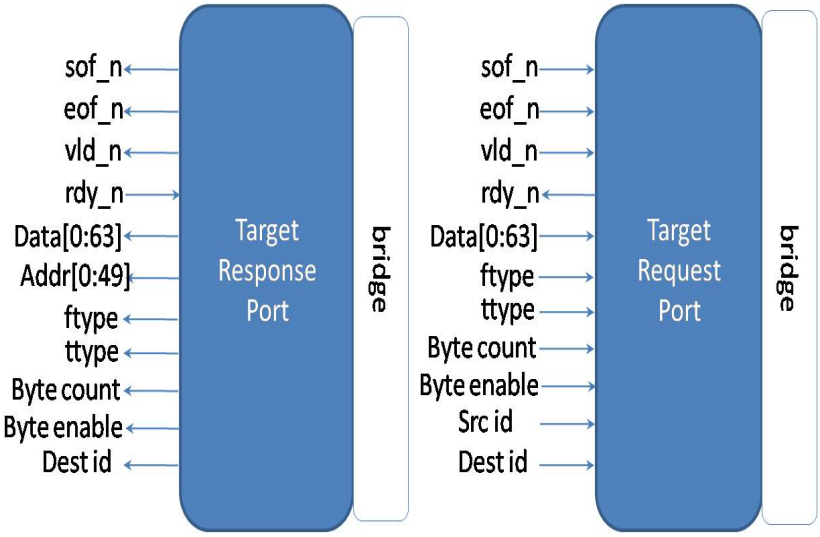


Figure 4.5: SRIO Target Port Signals

Similarly the signals identified for the PCIe ports are mentioned in Figure 4.6

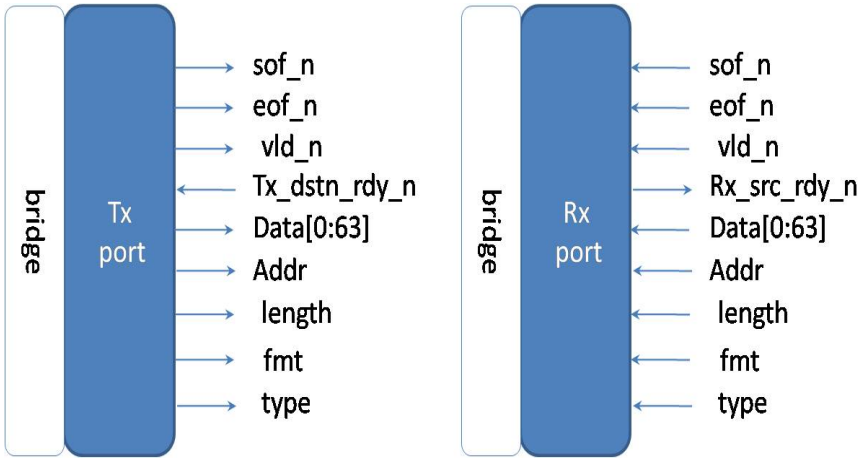


Figure 4.6: PCIe Tx and Rx Port Signals

4.1 Signal Mapping

The signal mapping logic is arrived at in the following manner.

From PCIe to SRIO side, for a start of packet , a valid sof signal is checked. On receiving a valid sof active low signal , it is checked whether the Initiator Request or Target Response ports is ready to receive the data. This is done by checking the active low Rdy signal. Data transmission starts to the port that is ready.

Once the start of packet happens, the header fields are identified with the help of their bit positions and stored in different registers for later processing. The address and data length and their positions are determined with the help of relevant fields of the the 64 bit header explained in Section 4.1.1.

Similarly for the SRIO to PCIe direction, the sof and valid signals are checked for being enabled on Initiator Response/ Target Request port. Once these signals are found enabled, the Rdy signal of PCIe Tx port is checked for being enabled. Once these conditions are met, data transmission is started.

The relevant fields of PCIe / SRIO that indicate the packet properties are stored in separate registers for further processing and are later converted to the SRIO / PCIe format respectively.

This is done in the following manner for both the sides:

4.1.1 PCIe to SRIO side Conversion

To convert from PCIe to SRIO, two important fields that depict the packet type are the Fmt and Type. The values of these fields are available at bit positions 62:61 and 60:56 respectively in the 64 bit PCIe header that is received in the first clock cycle after start of packet.

After the Fmt and Type fields have been stored in registers, they are checked against their values and the packet type information is deduced with the help of Table 2.4. These values are stored in separate registers of appropriate size. The Fmt and Type fields together are used to find out the address length of the packet. The length of the data to be stored in PCIe is given by the Length field in Double Words(DWs). The length as explained in Chapter 2 can vary from 0 to 1024 DWs. Also the first and last Byte Enable fields are used to decide which bytes of the data have to be read from the first and last DWs. This is done as per the Byte Enable rules explained in Section 2.8

After the Address and Data are read, the translation is initiated.The address conversion is carried out before transmission from one side to the other side of the Bridge.The PCIe address of 32 or 64 bit length is converted to the required SRIO address length of 50 bits and after the converted address is ready, it is kept in a separate register and is used after the complete packet data is available for transmission.

Here it is important to note that the PCIe packet length is different than the SRIO packet Lengths. The PCIe packet maximum length is 1024 DWs as mentioned above while that of the SRIO packet is 256 bytes. Therefore, to accommodate a complete PCIe packet data to SRIO data, multiple SRIO packets may be required. This is done by splitting the packets wherever required. If after every 256 bytes, there is still some data remaining to be

transmitted from the PCIe side, a new packet is generated by transmitting eof signal of the previous packet and the sof of the next one. The other fields however remain same to identify continuing data of the same packet. Also, to indicate the end of packet at SRIO for such cases, the length field of PCIe which is in expressed in DWs is translated to bytes for SRIO format and eof is generated accordingly at the last SRIO packet.

The Fmt and Type fields of PCIe need to be mapped to the the corresponding fields of the SRIO.The Ftype and Ttype field provide packet information of SRIO as explained in Chapter 3. Therefore the Fmt and Type fields are mapped to the Ftype and Ttype fields.

The bit conversions are done as per Table 4.1 .The first 3 columns are the PCIe packet types alongwith their Fmt and Type values, The next three columns indicate the corresponding SRIO packet and their Ftype and Ttype values that are required to be set.

PCI Packet Type	**Fmt**	**Type**	**SRIO Packet Type**	**Ftype**	**Ttype**
Memory Read(MRd)	00(3DW Hdr, No Data), 01(4DW Hdr No Data)	00000-MRd,00001-MRd Locked	ATOMIC NREAD	0010(2)	0100 (NREAD)
Memory Write(MWr)	01(3DW Hdr, with Data), 11(4DW Hdr,with Data)	00000	NWRITE / SWRITE	0101(5)	0100 (NWRITE)
CPL wo data	00	01010	RESP wo data	1101(13)	0000
CPL with Data	10	01010	RESP with data	1101(13)	1000

Table 4.1: Field Values for Packet Conversion PCIe to SRIO

Table 4.1 shows two subcases each for PCIe MRd and MWr cases. 3DW Hdr reflects the 32 bit address case and 4DW hdr reflects 64 bit address case. Also, within MWr cases, different data alignment results into different subcases. To take care of this, a suitable logic was implemented in the code.

Similarly, the Completion with Data and Completion without Data cases are translated to Response with Data and Response without Data cases respectively.

Once the necessary fields along with the address and the data are stored in required registers/FIFO, the mapping table is used to find out to which corresponding packet type it needs to be translated and accordingly the bit values are set on the SRIO side and sof and valid signals are enabled to indicate the start of a valid reception.

The request packets which include the MRd and MWr are thus mapped to the Initiator Request port and the Completion packets are mapped in a similar way to the Target Response port.

4.1.2 SRIO to PCIe side Conversion

The SRIO packet is different than the PCIe packet in various ways and those conditions are required to be addressed while doing translation from SRIO to PCIe.

The first important difference is the maximum data size of the SRIO packet which is 256 bytes. Therefore, the smaller SRIO data is required to be fitted into a bigger PCIe packet here which the opposite of the previous case.

Another difference is in the use of sof and eof signals to initiate and end the packet in SRIO. Since there is no length field in SRIO packet, its Data length has to be derived with the help of sof and eof signals. This is done by counting the number of Data Double Words arrived between sof and eof signals. The 64 bit header that is received in the first cycle contains the fields that contain information on the packet type.

The Ftype and Ttype fields contain the information about the packet type as described in Table 3.1. The address filed of SRIO is taken to be of 50 bits as per requirement. The Data field length and position vary with the packet type as described in Section 3.7. These Ftype and Ttype fields are captured and stored in separate registers and the address field is translated to the 64 bit PCIe format. The data is stored in a register if it is NWRITE/RESP with Data case and in a FIFO if it is SWRITE case.

After availability of the translated address and the required data, the values of Ftype and Ttype fields are converted to the Fmt and Type fields of PCIe. This is done as per the Table 4.2

SRIO Packet Type	**Ftype**	**Ttype**	**PCIe Packet Type**	**Fmt**	**Type**
NREAD	0010(2)	0100 (NREAD)	Memory Read(MRd)	01(4DW Hdr No Data)	00000
NWRITE	0101(5)	0100 (NWRITE)	Memory Write(MWr)	11(4DW Hdr with Data)	00000
SWRITE	0110(6)	0100 (SWRITE)	Memory Write(MWr)	11(4DW Hdr with Data)	00000
RESP without Data	1101(13)	0000	CPL without Data	00	01010
RESP with Data	1101(13)	1000	CPL with Data	10	01010

Table 4.2: Field values for Packet Conversion SRIO to PCIe

After the conversions of field values have been done, the translated address and data values are transmitted from one side of the Bridge to the other side and the results are tested for correctness.The verification process and the results obtained are explained in the next chapter.

CHAPTER 5

Verification and Results

The Bridge is implemented in Bluespec System Verilog (BSV) Hardware Description Language(HDL) and the complete functionality was verified by writing a number of testcases.

5.1 Verification Setup

The Bluespec code that is written in .bsv format is given to the Bluespec compiler in the Bluespec Development Workstation (BSW). The Bluespec Compiler compiles and generates the Verilog code in .v format. The Verilog code is synthesized in Xilinx ISE to get the clock frequency and to know about the amount of hardware generated by the design. Figure 5.1 shows the verification steps in Bluespec.

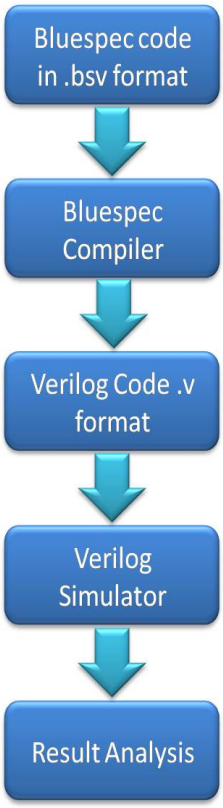


Figure 5.1: Verification Steps in Bluespec

Test cases were written to verify all the cases and subcases of Memory Read and Write and Response/Completion with and without Data operations of PCIe to SRIO side as well as SRIO to PCIe side.
For all the test cases, the necessary control and test Data signals with necessary fields were injected from a test bench file. The simulation results obtained were captured in a .vcd file and were viewed using gtkwave tool.

A brief description of the simulated test cases is given in Section 5.2 and Section 5.3

5.2 Simulation from PCIe to SRIO side

5.2.1 Mem Read 32 Bit Address No Data

The Memory Read case with 32 bit Address case was simulated by setting the Fmt field value of 000 and Type field vale of 00000. An Address value of 32 bits was set in the test bench. The Initiator Request Port was checked for being ready to transmit the Data.

The expected Ftype value was 2(0010) and Ttype value was 4(0100). The Ftype, Ttype and the translated address values were obtained at the Ireq port correctly after processing by the Bridge. The results are shown in Figure 5.2

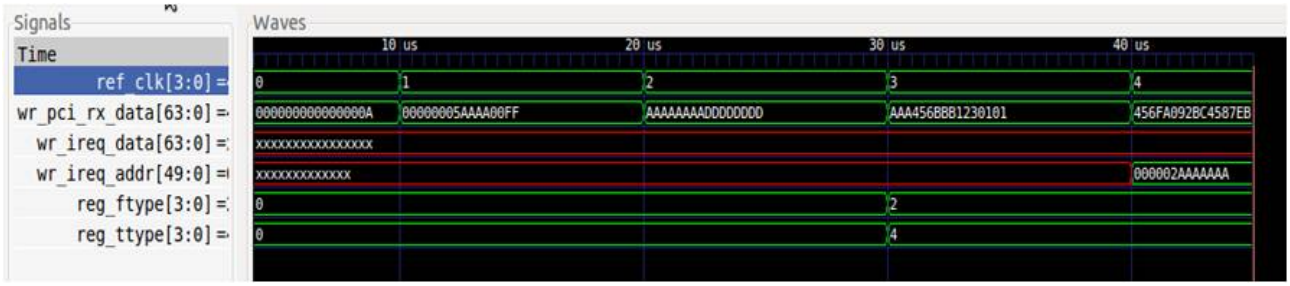


Figure 5.2: Simulation Result of Test case- MemRd 32 Bit Address

5.2.2 Mem Read 64 bit Address No Data

The Memory Read case with 64 bit Address case was simulated by setting the Fmt field value of 001 and Type field vale of 00000. An Address value of 64 bits was set in the test bench. The Initiator Request Port was checked for being ready to transmit the Data.

The expected Ftype value was 2(0010) and Ttype value was 4(0100). The Ftype, Ttype and the translated address values were obtained at the Ireq Port correctly after processing by the Bridge. The results are shown in

Figure 5.3

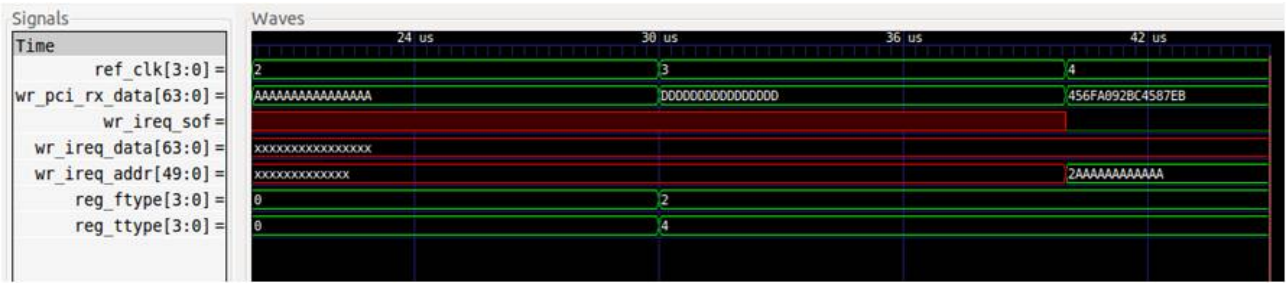


Figure 5.3: Simulation Result of Test case- MemRd 64 Bit Address

5.2.3 Mem Write 32 Bit Address 5DW Data

This subcase of the Memory Write case with 32 bit Address case was simulated by setting the Fmt field value of 010 and Type field vale of 00000. An Address value of 64 bits was set and 5DWs of Data were sent from the test bench. The Initiator Request Port was checked for being ready to transmit the Data.

The expected Ftype value was 5(0101) and Ttype value was 4(0100). The Ftype, Ttype and the translated Address values were obtained at the Ireq Port correctly after processing by the Bridge. The results are shown in Figure 5.4

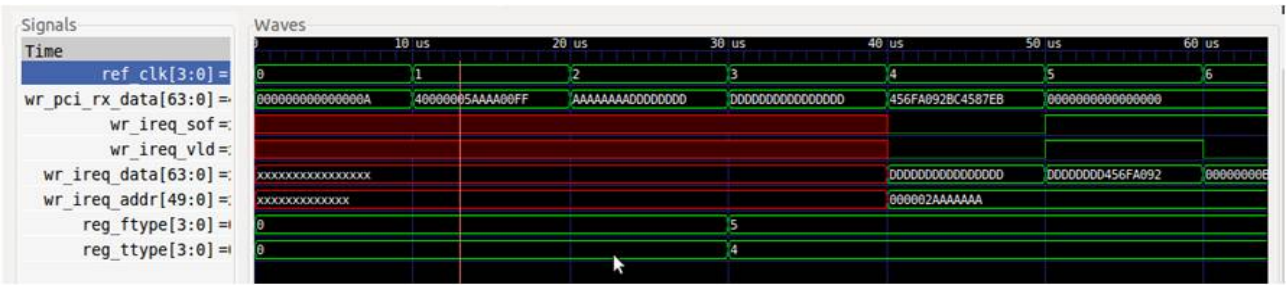


Figure 5.4: Simulation Result of Test case- MemWr 32 Bit Address 5DW Data

5.2.4 Mem Write 32 Bit Address 1024 DW Data

This subcase of the Memory Write case with 32 bit Address case was simulated by setting the Fmt field value of 010 and Type field value of 00000. An Address value of 64 bits was set and 1024 DWs of Data were sent from the test bench. The Initiator Request Port was checked for being ready to transmit the Data.

The expected Ftype value was 5(0101) and Ttype value was 4(0100). The Ftype, Ttype and the translated Address values were obtained at the Ireq Port correctly after processing by the Bridge. The results are shown in

Figure 5.5

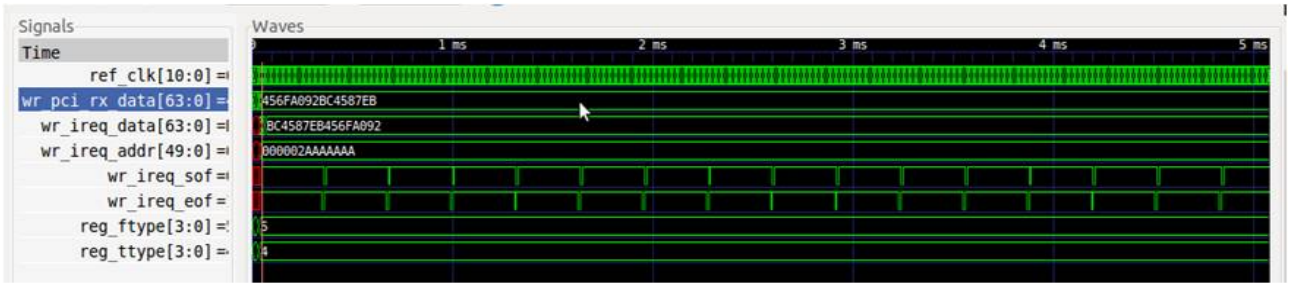


Figure 5.5: Simulation Result of Test case- MemWr 32 Bit Address 1024 DW Data

5.2.5 Mem Write 64 Bit Address 5DW Data

This subcase of the Memory Write case with 64 bit Address was simulated by setting the Fmt field value of 011 and Type field vale of 00000. An Address value of 64 bits was set and 5DWs of Data were sent from the test bench. The Initiator Request port was checked for being ready to transmit the Data.

The expected Ftype value was 5(0101) and Ttype value was 4(0100). The Ftype, Ttype and the translated Address values were obtained at the Ireq Port correctly after processing by the Bridge. The results are shown in Figure 5.6

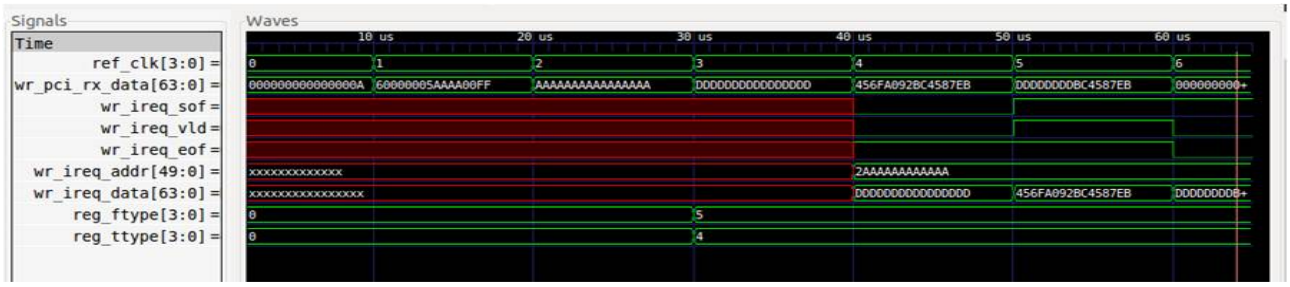


Figure 5.6: Simulation Result of Test case- MemWr 64 Bit Address 5DW Data

5.2.6 Mem Write 64 Bit Address 1024DW Data

This subcase of the Memory Write case with 64 bit Address was simulated by setting the Fmt field value of 011 and Type field value of 00000. An Address value of 64 bits was set and 1024 DWs of Data were sent from the test bench. The Initiator Request Port was checked for being ready to transmit the Data.

The expected Ftype value was 5(0101) and Ttype value was 4(0100). The Ftype, Ttype and the translated Address values were obtained at the Ireq port correctly after processing by the Bridge. The results are shown in

Figure 5.7

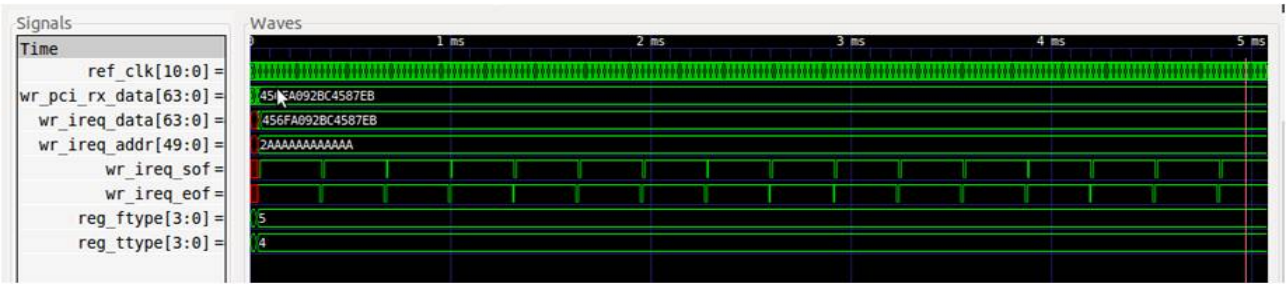


Figure 5.7: Simulation Result of Test case- MemWr 64 Bit Address 1024DW Data

5.2.7 Completion without Data(No Addr No Data)

The Completion without Data case with No Address and No Data was simulated by setting the Fmt field value of 000 and Type field value of 01010. The Target Response Port was checked for being ready to transmit the Data.

The expected Ftype value was 13(1101) and Ttype value was 0(0000). The Ftype, Ttype values were obtained at the Tresp Port correctly after processing by the Bridge. The results are shown in Figure 5.8

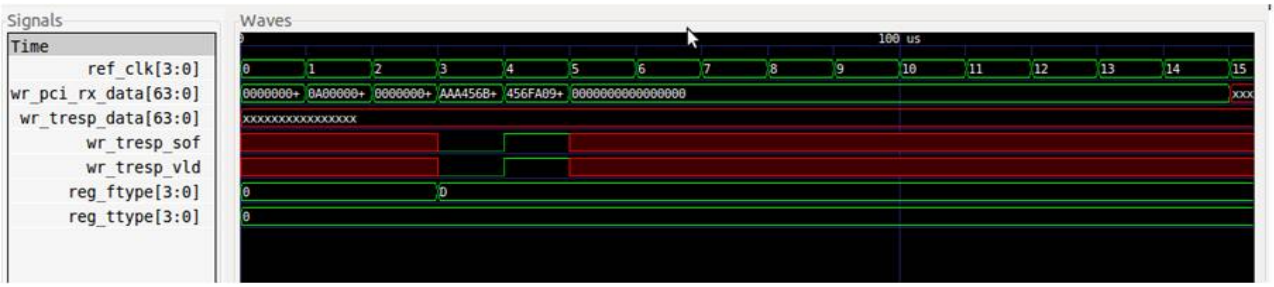


Figure 5.8: Simulation Result of Test case- Completion Without Data

5.2.8 Completion with Data (No Addr 5DW Data)

The Completion with Data case with no Address and 5DW Data subcase was simulated by setting the Fmt field value of 010 and Type field value of 01010. A 5DW Data was injected from test bench. The Target Response Port was checked for being ready to transmit the Data.

The expected Ftype value was 13(1101) and Ttype value was 8(1000). The Ftype, Ttype and the translated Address values were obtained at the Tresp Port correctly after processing by the Bridge. The results are shown in Figure 5.9

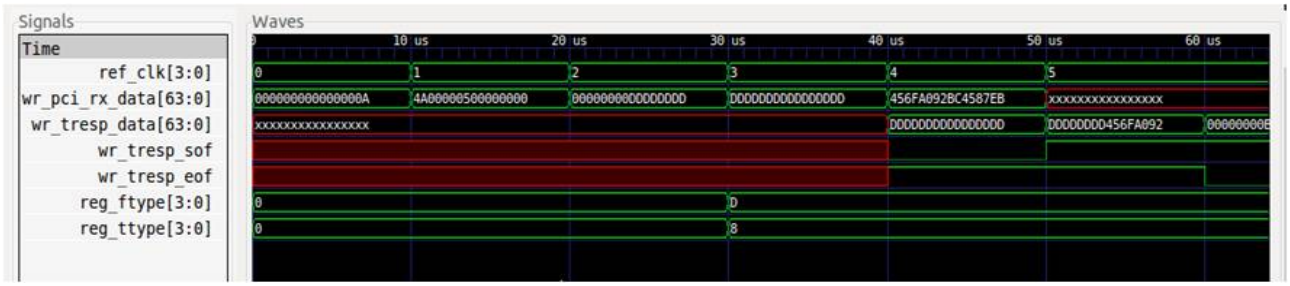


Figure 5.9: Simulation Result of Test case- Completion With 5DW Data

5.2.9 Completion with Data (No Addr 1024DW Data)

The Completion with Data case with No Address and 1024 DW Data subcase was simulated by setting the Fmt field value of 010 and Type field vale of 01010. A 1024 DW data was injected from test bench. The Target Response Port was checked for being ready to transmit the Data.

The expected Ftype value was 13(1101) and Ttype value was 8(1000). The Ftype, Type and the translated Address values were obtained at the Tresp port correctly after processing by the Bridge. The results are shown in Figure 5.10

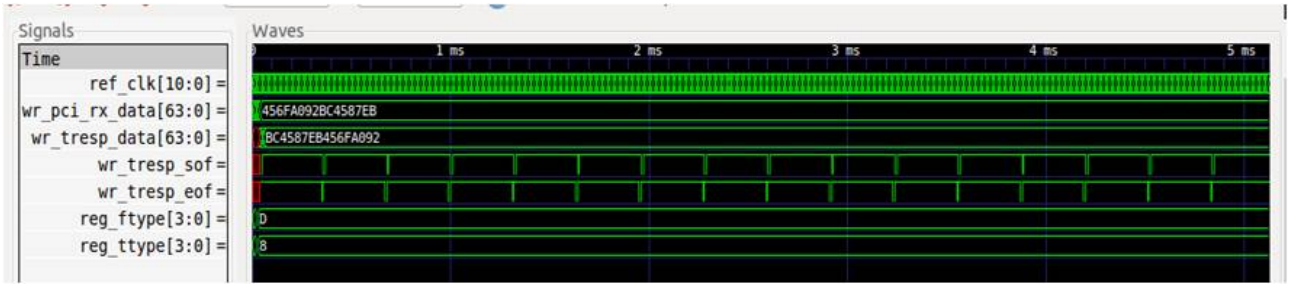


Figure 5.10: Simulation Result of Test case- Completion With 1024DW Data

5.3 Simulation from SRIO to PCIe side

5.3.1 NREAD for Iresp Port

An Ftype value of 2(0010) and Ttype value of 4(0100)was set was set at the Initiator Request Port in the test bench to simulate the NREAD packet with a 50 bit SRIO Address value.

An expected Fmt value of 001 and Type value 00000 as well as a zero padded 64 bit Address values were obtained correctly at the PCI Tx Port after processing by the Bridge. The calculated Data length is stored in a register and the Length value is 0 in this case(no data). The results are shown in Figure 5.11

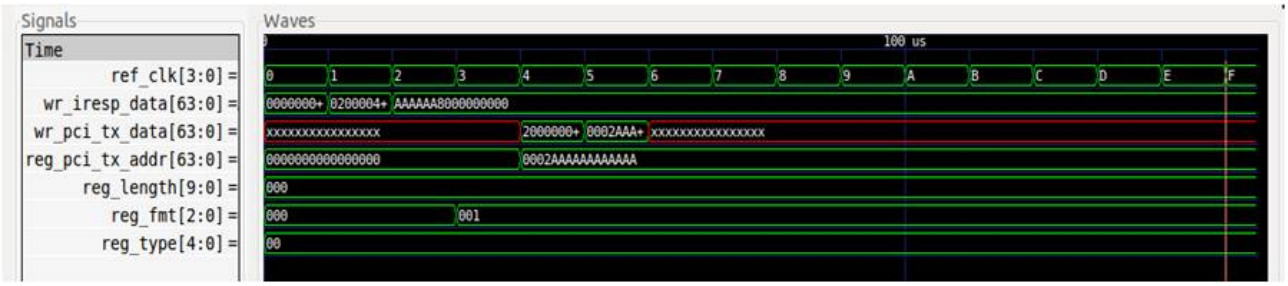


Figure 5.11: Simulation Result of Test case- NREAD (Iresp)

5.3.2 NWRITE for Iresp Port

An Ftype value of 5 (0101) and Ttype value of 4(0100) was set at the Initiator Request Port in the test bench to simulate the NWRITE packet with a 50 bit SRIO Address and 64 bit data value for single clock cycle.

An expected Fmt value of 011 and Type value 00000 as well as a zero padded 64 bit Address and the same data were obtained correctly at the PCI Tx Port after processing by the Bridge. The calculated data length in DWs is stored in a register and the Length value is 2 in this case. The results are shown in Figure 5.12

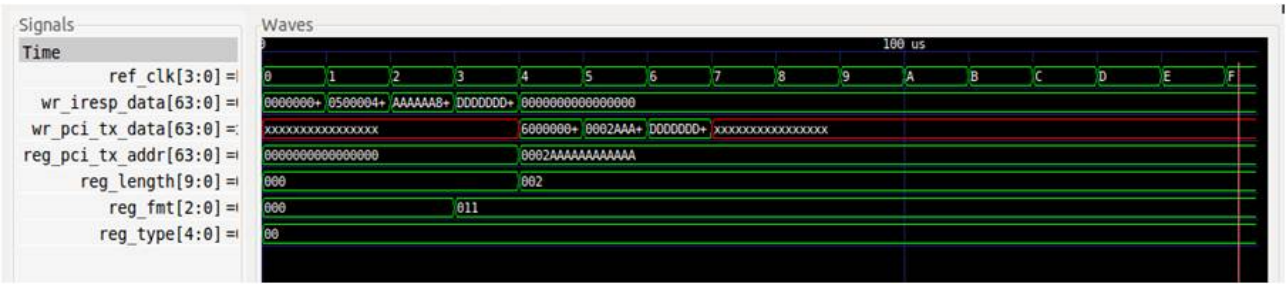


Figure 5.12: Simulation Result of Test case- NWRITE (Iresp)

5.3.3 SWRITE for Iresp Port

An Ftype value of 6 (0110) and Ttype value of 4(0100) was set was set at the Initiator Request Port in the test bench to simulate the SWRITE packet with a 50 bit SRIO Address and 64 bit Data value for 7 clock cycles.

An expected Fmt value of 011 and Type value 00000 as well as a zero padded 64 bit address and the same data were obtained correctly at the PCI Tx Port after processing by the Bridge. The calculated Data length in DWs is stored in a register and the Length value is 14 in this case. The results are shown in Figure 5.13

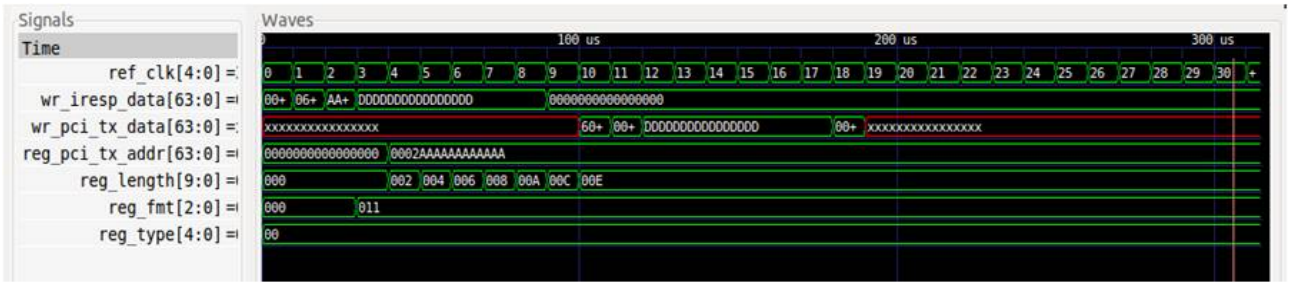


Figure 5.13: Simulation Result of Test case- SWRITE (Iresp)

5.3.4 Response without Data for Iresp Port

An Ftype value of 13 (1101) and Ttype value of 0(0000) was set at the Initiator Request Port in the test bench to simulate the Response without Data packet. An expected Fmt value of 000 and Type value 01010 (No Address No Data case) were obtained correctly at the PCI Tx port after processing by the Bridge. The results are shown in Figure 5.14

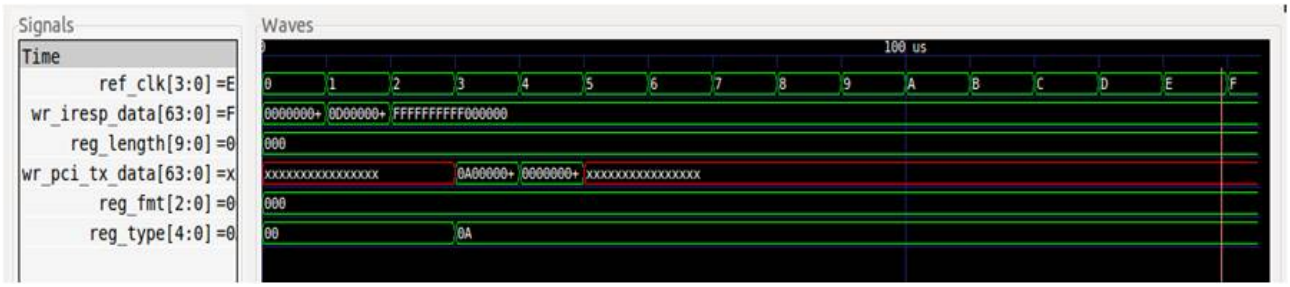


Figure 5.14: Simulation result of Test case - Resp w/o Data (Iresp)

5.3.5 Response with Data for Iresp Port

An Ftype value of 13 (1101) and Ttype value of 0(0000) was set at the Initiator Request Port in the test bench to simulate the Response with Data packet with 64 bit Data value for 1 clock cycle.

An expected Fmt value of 010 and Type value 01010 were obtained correctly at the PCI Tx Port after processing by the Bridge. The calculated Data length is stored in a register and the Length value in DWs is 2 in this case.The results are shown in Figure 5.15

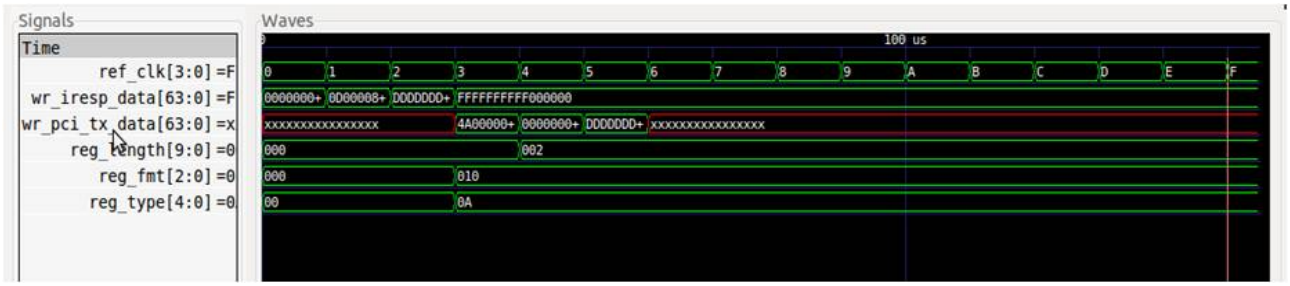


Figure 5.15: Simulation Result of Test case - Resp with Data (Iresp)

Similar to the Iresp, all the cases were simulated for Target Request (Treq) Port. The results obtained for all the cases of Treq Port were verified and were identical to the Iresp Port.

5.4 Synthesis Reports

5.4.1 FPGA Synthesis Report

The design has been synthesized using Xilinx ISE for Artix Series device 7a200tfbg676.

All default settings were used. The design strategy was set to optimization for speed. The slice utilization and timing summary are provided below.

Device Utilization Summary :

Selected Device: 7a200tfbg676-1

Slice Logic Utilization:

Number of Slice Registers : 885 out of 269200

Number of Slice LUTs : 1979 out of 134600

Number used as Logic : 1979 out of 134600

Number of LUT Flip Flop pairs used: 2038

Timing Summary :

Minimum period: 5.068ns

Maximum Frequency: 197.321MHz

Minimum input arrival time before clock: 4.619ns

Maximum output required time after clock: 7.095ns

Maximum combinational path delay: 5.856ns

5.4.2 ASIC Synthesis Report

ASIC synthesis was done in Synopsys Design Compiler with 55nm technology and the following results were obtained:

Area Utilization Summary:

Number of ports: 770

Number of nets: 120892

Number of cells: 120593

Number of combinational cells: 87070

Number of sequential cells: 33522

Number of buf/inv: 26441

Number of references: 375

Combinational area: 323278.083832

Buf/Inv area: 63390.720185

Noncombinational area: 312142.571746

Total cell area: 635420.655577

Timing Summary :

Minimum period: 1.1 ns

Maximum Frequency: 909 MHz

Power Consumption:

Cell Internal Power: 333.9417 mW

Net Switching Power: 2.7061 mW

Total Dynamic Power: 336.6478 mW

Cell Leakage Power: 672.6069 uW

CHAPTER 6

Conclusion and Future Work

In this project, a Bridge was developed at the logical layer level of PCIe and SRIO.

A mapping logic was developed to map the signals of PCIe with those of SRIO and vice versa for translating the packets.Test cases were written to simulate the Read, Write and Response Packets of both PCIe and SRIO alongwith their subcases . Conversion of corresponding field values was verified in the simulated packets for both sides and all subcases.

The Bridge was written in BSV HDL and was compiled and simulated using Bluesim BSV Compiler and Verilog compiler of Bluespec.The design was then synthesized using Xilinx ISE and a clock frequency of 197 MHz was achieved. Also, the design was synthesized for ASIC implementation using Synopsis Design Compiler with a timing constraint of 1.1 ns to yield a frequency of about 900 MHz.

Future Work:

The following additions and improvements can be done to the current design in future:

- The existing design has been made at the logical layer level.The design can be extended upto Physical Layer.
- The features of the Bridge can be enhanced by adding the configuration,maintenance modes and multiple SRIO Address length values.
- The Bridge can be implemented using Silicon-Photonics based Interconnects in future.

REFERENCES

[1] Bluespec, Inc. *Bluespec System Verilog Reference Guide*, revision: 17 edition, 2012.

[2] Sam Fuller. *RapidIO: The Embedded System Interconnect*. John Wiley and Sons Ltd, 2005.

[3] Integrated Device Technology, Inc. *Tsi721 User Manual*, version 1.0 edition, 2013.

[4] Rishiyur S. Nikhil and Kathy Czeck. *BSV by Example*. Bluespec, Inc, 2010.

[5] PCI-SIG. *PCI Express Base Specification Revision 3.0*, version 3.0 edition, 2010.

[6] Xilinx, Inc. *LogiCORE IP Serial RapidIO v5.6*, version 5.1 edition, 2011.

[7] Xilinx, Inc. *7 Series FPGAs Integrated Block for PCI Express v3.0*, version 1.0 edition, 2014.