

**OPTIMIZATION OF RECEIVER FIFO FOR IEEE
802.3ba 40GBASE PCS SUBLAYER**

Anuradha Nirmala Nanayakkara

(118411J)



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Degree of Master of Science

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

June 2015

**OPTIMIZATION OF RECEIVER FIFO FOR IEEE
802.3ba 40GBASE PCS SUBLAYER**

Anuradha Nirmala Nanayakkara

(118411J)



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Dissertation submitted in partial fulfillment of the requirements for the degree
Master of Science in Electronics and Automation

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

June 2015

Declaration of the Candidate and the Supervisor

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date:

The above candidate has carried out research for the Master's dissertation under my supervision.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Signature of the supervisor:

Date:

Abstract

Keywords: FIFO, IEEE802.3, PCS Sub layer, 40GBASE-R, 10GBASE-X

Local Area Networks (LAN) are based on Ethernet technology. Commonly used 10 and 40 Gigabit Ethernet systems are adopting IEEE 802.3 standards.

The aim of this dissertation is to optimize the FIFO design for the receiver of Physical Coding Sub layer (PCS) specified by IEEE 802.3 standards. This dissertation is having two phases. In the first phase, optimal FIFO for IEEE 802.3ae 10GBASE-X PCS receiver is designed and implemented. Proper operation of the proposed design is verified with simulation results. In the second phase, possible optimization for receiver FIFO of IEEE 802.3ba 40GBASE-R PCS layer is identified. Potential implementation for 40GBASE-R PCS is simulated with proposed FIFO design, to verify the proper functionality.

Proposed designs will save gate count, power and the silicon area of ASIC design considerably. As future work it is suggested to emulate the proposed design with a suitable hardware.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Acknowledgements

First and foremost I would like to express my sincere thanks to my supervisor, Dr. S. Thayaparan, Senior Lecturer, Department of Electronic and Telecommunication Engineering, University of Moratuwa, for his continuous guidance and support provided to me throughout this research. His encouragement to tackle difficulties encountered and the patience with my mistakes should always be appreciated.

I also wish to acknowledge Prof. S.R. Munasinghe, Senior Lecturer, Department of Electronic and Telecommunication Engineering, University of Moratuwa, for the guidance provided at different stages of the research.

My parents, brother and sister too should be remembered for their continuous support and encouragement. Finally I feel I should dedicate a single sentence for my little kid Poo, (who is just turning two) for bearing up her amma frowning in front of the laptop prolonged hours, in spite of the fact that she is missing the care and attention of her farther as well.

This piece of work is dedicated to all those, who helped me in numerous ways to make this a success.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

TABLE OF CONTENTS

Declaration of the Candidate and the Supervisor	i
Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	vii
List of Tables	viii
List of Abbreviations	ix
1. Introduction	1
1.1 Introduction	1
1.2 Problem Statement	1
1.3 The Aims and Objectives of the Research	2
1.4 Research Methodology	2
1.5 Resource Requirements	3
1.6 Scope of the Project	3
2. Analysis	4
2.1 10GBASE-X Systems	4
2.1.1 8B/10B encoding	5
2.1.2 Special code groups [5]	8
2.1.3 Lane alignment	8
2.1.4 Clock rate compensation	9
2.1.5 Optimal receiver FIFO parameters for 10GBASE-X PCS sub layer [11]	10
2.2 40GBASE-R Systems	11
2.2.1 Idle Control Character /I/	13
2.2.2 Lane Alignment Consideration	14
2.2.3 Clock rate compensation in worst case scenario	15

2.2.4	Optimization of receiver FIFO parameters for 40GBASE-R PCS sub layer	17
3.	Modeling PCS Sub layer of 40GBASE-R systems	18
3.1	40GBASE-R PCS Sub Layer Modeling	18
3.2	40GBASE-R PCS Sub Layer Transmitter	18
3.2.1	64B/66B Encoder [13]	19
3.2.2	Scrambler [5: Clause 82.2.5, 14]	20
3.2.3	Block distribution [5: Clause 82.2.6]	20
3.2.4	Alignment marker insertion [5: Clause 82.2.7]	21
3.2.5	Serializer	23
3.3	40GBASE-R PCS Sub Layer Receiver	23
3.3.1	Deserializer	23
3.3.2	Block synchronization [5: Clause 82.2.11]	23
3.3.3	Alignment marker lock [5: Clause 82.2.12]	23
3.3.4	Lane deskew FIFO	24
3.3.5	Lane reorder [5: Clause 82.2.13]	25
3.3.6	Alignment marker removal [5: Clause 82.2.14]	25
3.3.7	Clock rate compensation FIFO	25
3.3.8	Descrambler [5: Clause 82.2.15]	25
3.3.9	Decoder [15]	26
4.	Simulation, Results and Achievements	27
4.1	Simulation Environment	27
4.2	10GBASE-X Simulation and Results	27
4.2.1	Waveforms captured for FIFO Full viable situation	27
4.2.2	Waveforms captured for FIFO Empty viable situation	28
4.3	40GBASE-R Simulation and Results	29

4.3.1	FIFO Full viable situation	29
4.3.2	FIFO Empty viable situation	29
4.4	Publication List	31
5.	Conclusions and Future Works	32
5.1	Discussion and Conclusions	32
5.1.1	For 10 GBASE-X	32
5.1.2	For 40 GBASE-R	32
5.2	Recommendation for Future Work	33
	Reference List	34
	Appendix A: Verilog test bench for 10GBASE-X PCS Sublayer FIFO design	37
	Appendix B: Verilog testbench for 40GBASE-R PCS sub layer Model	40



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

LIST OF FIGURES

	Page
Figure 2.1: Positioning of XGXS and PCS Sub layers in IEEE 802.3 10G Model	5
Figure 2.2: Functional block diagram of the 10GBASE-X physical layer	6
Figure 2.3: XGMII character stream to PCS code group mapping example	7
Figure 2.4: Positioning of 40G Ethernet	11
Figure 2.5: Functional block diagram 40GBASE-R physical layer	12
Figure 2.6: 64B/66B block formats	13
Figure 2.7: Formation of maximum data packet	16
Figure 3.1: 40GBASE-R Transmitter Model	19
Figure 3.2: Encoder Output	20
Figure 3.3: Scrambler	20
Figure 3.4: PCS Block Distribution	21
Figure 3.5: Alignment marker format	22
Figure 3.6: Alignment Marker Insertion	22
Figure 3.7: Alignment Marker insertion period	22
Figure 3.8: 40GBASE-R Receiver Model	24
Figure 3.9: Descrambler	26
Figure 3.10: Decoder Output	26
Figure 4.1: Signals captured from ModelSim Wave simulation in a FIFO Full condition viable scenario for 10GBASE-X	28
Figure 4.2: Signals captured from ModelSim Wave simulation in a FIFO Empty condition viable scenario for 10 GBASE-X	28
Figure 4.3: Signals captured from ModelSim Wave simulation in a FIFO Full condition viable scenario for 40GBASE-R	29
Figure 4.4: Signals captured from ModelSim Wave simulation in a FIFO Empty condition viable scenario for 40GBASE-R	30

LIST OF TABLES

	Page
Table 2.1: XGMII characters to PCS code-group mapping	6
Table 2.2: PCS code-group to XGMII character mapping	7
Table 2.3: Skew Budget for 10GBASE-X	9
Table 2.4: Control codes	14
Table 2.5: Maximum skew for PCS	14



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

LIST OF ABBREVIATIONS

Abbreviation	Description
ASIC	Application Specific Integrated Circuit
BIP	Bit Interleave Parity
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DTE	Data Terminal Equipment
FCS	Frame Check Sequence
FIFO	First In First Out
HDL	Hardware Description Language
IP	Internet Protocol
IPG	Inter Packet Gap
LAN	Local Area Network
MAC	Media Access Control
OSI	Open System Interconnection
PCS	Physical Coding Sub layer
PHY	PHysical Layer
PMA	Physical Medium Attachment
RS	Reconciliation Sub layer
RXC	Receive Control signals
TXC	Transmit Control signals
UI	Unit Interval
XAUI	10 Gigabit Attachment Unit Interface
XGMII	10 Gigabit Media Independent Interface
XGXS	Extender Sub layer
XLGMII	40Gb/s Media Independent Interface



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mru.ac.lk

1. INTRODUCTION

Chapter 1, Introduction is organized into five subsections. Section 1.1 provides introduction to the research. Problem statement is defined in section 1.2. Section 1.3 gives the objective of the research work. Research Methodology, section 1.4 describes the how the research had been conducted. Resource requirements for the project and the scope of project are coming under section 1.5 and 1.6 respectively.

1.1 Introduction

Nowadays consumer Internet Protocol (IP) traffic demand is bolstering. High definition video, high speed broadband access, growth of network aggregation applications, growing number of server and computer applications with significant bandwidth need, are some contributors for the huge data traffic demands. Ethernet is the ubiquitous connectivity technology for Local Area Networks (LAN) due to its low cost, known reliability and simplicity. Rising volume of IP traffic demands for high speed LAN interfaces, 10 gigabits per second (Gbps) or even something beyond that. Commonly used 10 Gigabit and 40 Gigabit Ethernet systems are adopting IEEE 802.3 standards [1]-[10]. IEEE802.3ae for 10Gbps and IEEE802.3ba supporting 40Gbps data transfer rate were standardized in order to drive this rapid growth in IP data traffic.

1.2 Problem Statement

A family of 10 Gb/s physical layer implementations, consists of 10GBASE-CX4 [5: Clause 54] and 10GBASE-LX4 [5: Clause 53] is referred as 10GBASE-X. These are based upon 8B/10B data coding method. As defined by the IEEE802.3 standards, these designs are having independent four serial lanes at their, receive and transmit data paths [5: Clause 48] of the PCS and XGXS sub layer.

The 40GBASE-R refers to a family of Physical Layer implementations based upon 64B/66B data coding method. The 40GBASE-R Physical Coding Sub layer (PCS) performs encoding (decoding) of data from (to) the 40Gb/s Media Independent Interface (XLMII) to 64B/66B code blocks, distribute data to multiple lanes, and

transfer the encoded data to the PMA. 40GBASE-R PCS is using four encoded bit streams to communicate with PMA.

Hence in both scenarios lane synchronization has to be carried out at the receiver side of the PCS. Therefore at the receiver, one FIFO is required by each lane to store the data received. Since there are four serial lanes, a bank of four FIFOs has to be employed at the PCS receiver of each physical layer implementation.

In order to implement the FIFO banks at the PCS layer of the receiver either RAM or register arrays are used. Both RAM and the register arrays are power hunting and in order to accommodate them considerable area has to be allocated in the ASIC design. Therefore an optimal FIFO design for IEEE802.3ae 10G PCS sub layer receiver as well as optimization for IEEE802.3ba 40G PCS layer receiver FIFO, can save a significant amount of power and silicon area.

1.3 The Aims and Objectives of the Research

The aim of this research project is to study and analyze IEEE 802.3 standard specifications, clause 48 and clause 82 of [5] respectively for 10G and 40G Ethernet. With the analysis possible optimization for receiver FIFO parameters are identified. Proposed FIFO designs are simulated and desired signals are monitored to verify the proper functionality. Analysis in the 10G is applicable for both 10GBASE-CX4 and 10GBASE-LX4 systems. Also the particular FIFO design proposed is applicable for all 40GBASE-R systems.

1.4 Research Methodology

The research methodology adhered is as follows. Relevant standard clauses from [5] were studied, and data transmission was analyzed in order to identify optimization for PCS sub layer receiver FIFO designs. Proposed designs are modeled and implemented in a simulation environment. The functionalities of proposed designs are verified using simulation results.

1.5 Resource Requirements

PCS layer for the 10G is specified in clause 48 of section 4 of [5]. Clause 82 of section 6 of [5] specifies PCS layer for 40G. The proposed designs are simulated using ModelSim Simulation software [22]. Verilog, Hardware Description Language [23] (HDL) is used to model the systems. The contribution of the proposed design for the ASICs is quantified considering the general values being used in the industry.

1.6 Scope of the Project

This research project is in the scope of Ethernet for LAN based on IEEE802.3 standard. It introduces optimization for PCS receiver FIFO of 10G and 40G Ethernet systems.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2. ANALYSIS

Chapter 2 Analysis is divided in to two parts. This includes analysis of IEEE802.3 clause 48 for 10GBASE-X systems followed by analysis of IEEE802.3 clause 82 for 40GBASE-R systems. Under each type of systems first the functional block diagram is investigated. After that there is an independent flow of subsections dedicated to following discussions. Subsection 1 type of encoding used, subsection 2 special code groups. Determination of FIFO depth based on lane alignment and cock rate compensation comes under subsections 3 and 4. As the last subsection proposed optimization for FIFO parameters will be discussed for each case.

2.1 10GBASE-X Systems

Figure 2.1 taken from [5] shows the positioning of different sub layers of IEEE802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) LAN model and their relationship to Open System Interconnection (OSI) reference model.

The purpose of the XGMII (10 Gigabit Media Independent Interface) is to provide a simple, inexpensive, and easy-to-implement interconnection between the MAC (Media Access Control) sub layer and the Physical Layer (PHY). XAUI (10 Gigabit Attachment Unit Interface) is an optional layer, used to extend the operational distance of the XGMII. XGMII Extender sub layer (XGXS), is comprised of two XGXS layers: one at the RS (Reconciliation Sub layer) end (DTE XGXS), and other XGXS at the PHY end (PHY XGXS) and a XAUI between them. The purpose of the XGMII Extender is to extend the operational distance of the XGMII and to reduce the number of interface signals.

The transmitter and the receiver functional block diagram for the 10GBASE-X PCS layer is shown in, Figure 2.2.

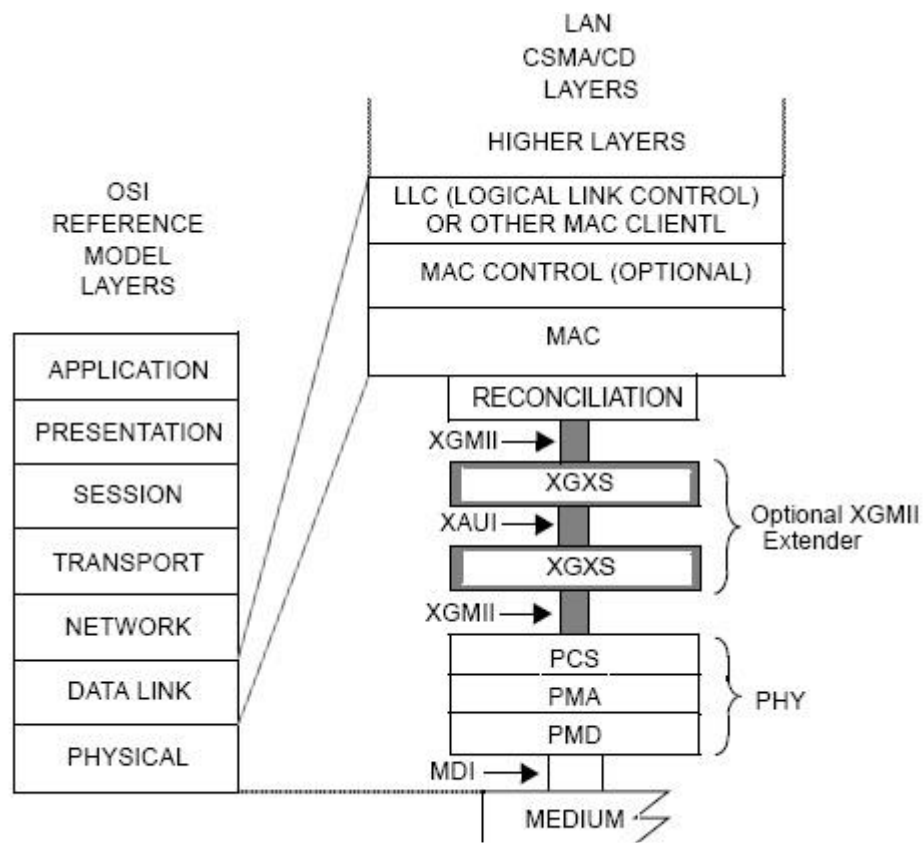


Figure 2.1: Positioning of XGXS and PCS Sublayers in IEEE 802.3 10G Model
 Source: Clause 47 [5] www.lib.mrt.ac.lk

2.1.1 8B/10B encoding

The 10GBASE-X systems are using 8B/10B encoding and 10B/8B decoding algorithms [5: Clause 48.2.3]. The PCS maps 8 bit XGMII characters into 10-bit code-groups, and vice versa, using the 8B/10B block coding scheme. The mapping of XGMII characters to PCS code-groups is specified in Table 2.1.

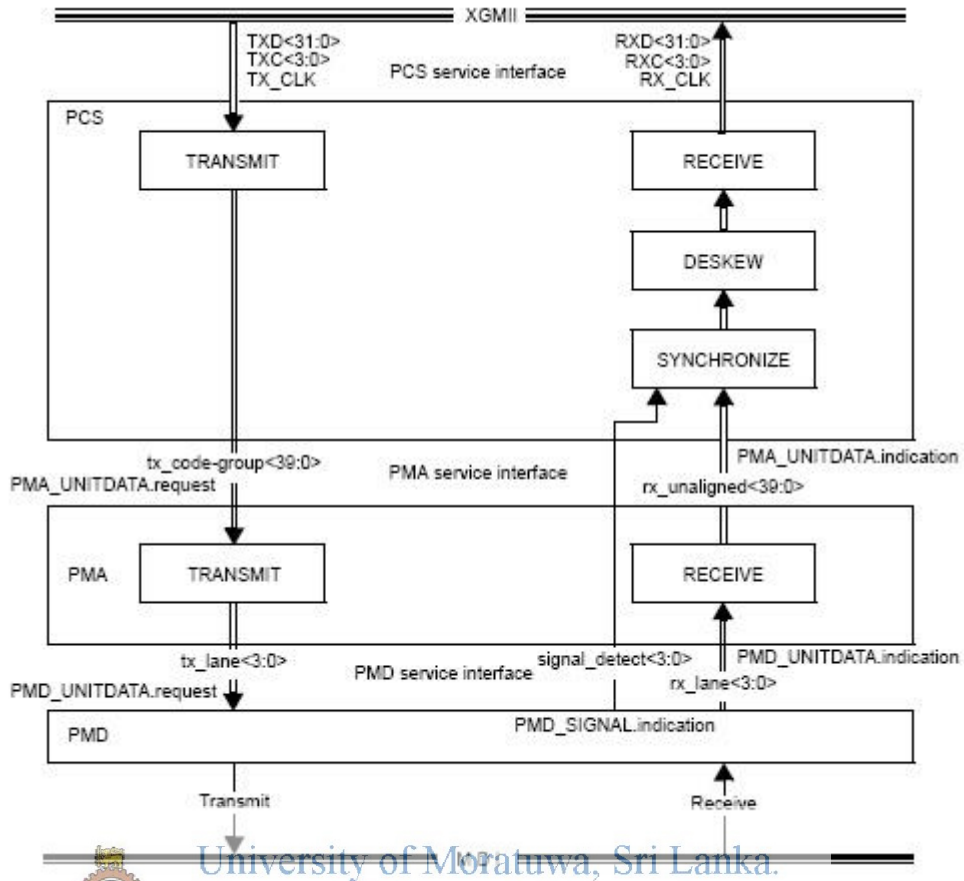


Figure 2.2: Functional block diagram of the 10GBASE-X physical layer
 Source: Clause 48.1.6 [5]

Table 2.1: XGMII characters to PCS code-group mapping

XGMII TXC	XGMII TXD	PCS code-group	Description
0	00 through FF	Dxx.y	Normal data transmission
1	06	K28.0 or K28.3 or K28.5 or D20.5	Assert LPI
1	07	K28.0 or K28.3 or K28.5	Idle in I
1	07	K28.5	Idle in T
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error

Source: Clause 48.2.3 [5]

The mapping of PCS code-groups to XGMII characters is specified in Table 2.2.

Table 2.2: PCS code-group to XGMII character mapping

XGMII RXC	XGMII RXD	PCS code-group	Description
0	00 through FF	Dxx.y	Normal data transmission
1	06	K28.0 or K28.3 or K28.5 or D20.5a	Assert LPI
1	07	K28.5	Idle in I
1	9C	K28.4	Sequence
1	FB	K27.7	start
1	FD	K29.7	Terminate
1	FE	K30.7	Error

Source: clause 48.2.3

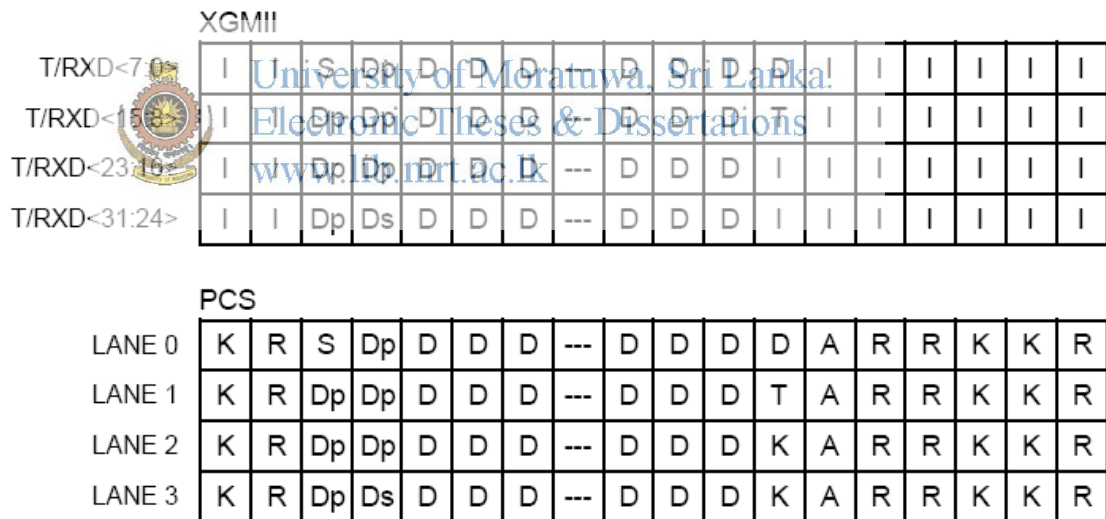


Figure 2.3: XGMII character stream to PCS code group mapping example

Source: Clause 48.2.3 [5]

Hence at the PCS transmitter an 8 bit XGMII character is mapped into 10 bit PCS code group. In order to accommodate this 8B to 10B encoded data, the width of the receiver FIFO at the PCS sub layer should be 10 bits. Further data received on data path consist of four independent serial lanes, and these data should be stored

independently at the receiver. So at the receiver, there should be a bank of four FIFOs, one per each lane. The recovered clock from the serial lane data is used to store the received data in the FIFOs. It means the FIFO write pointer is controlled by the recovered clock. The local clock is used to read the stored data and hence controlling the FIFO read pointer.

2.1.2 Special code groups [5]

Idle ordered sets (||I||) are transmitted in all four serial lanes continuously and repetitively whenever the XGMII is idle. This is denoted as TXD <31:0>=0x07070707 and TXC <3:0>=0xF. ||I|| provides a continuous fill pattern to establish and maintain lane synchronization, perform lane-to-lane deskew and perform PHY clock rate compensation.

A sequence of ||I|| ordered sets consists of one or more consecutively transmitted ||K||, ||R|| or ||A|| ordered sets. The purpose of randomizing the ||I|| sequence is to reduce 10GBASE-X electromagnetic interference (EMI) during idle state.

The idle character ||I|| on the XGMII side is converted into ||A||, ||K|| and ||R|| characters within the transmitters of the PCS and XGXS sub layers. The ||A||, ||K|| and ||R|| characters are respectively used for the lane alignment, code group alignment and clock rate compensation purposes at the receiver side of the PCS and XGXS sub layers. Similarly, ||A||, ||K|| and ||R|| characters received by the receiver of the PCS and XGXS sub layer are converted into idle character ||I|| within the receiver of the PCS and XGXS sub layers.

The depth of the receiver FIFO is determined based on two factors. One is the lane alignment at the receiver side. The other is the clock rate compensation in the worst case scenario at the receiver.

2.1.3 Lane alignment

Skew is introduced between lanes by both active and passive elements of a 10GBASE-X link. Allowable skew for all link elements are specified in the budget. The skew budget shown in

Table 2.3 is given at the IEEE Std 802.3 Clause 48.2.4.2.2 [5].

Table 2.3: Skew Budget for 10GBASE-X

Skew Source	Occurrences	Skew	Total Skew
PMA Tx	1	1 UI	1 UI
PCB	2	1 UI	2 UI
Medium	1	< 18 UI	< 18 UI
PMA Rx	1	20 UI	20 UI
Total			< 41 UI

Source: Clause 48.2.4.2.2 [5]

Unit Interval (UI) is the period of time allocated for the transmission of one symbol on one channel. Clock rate of 312.5 MHz implies 3200 ps for one code group, which is a transfer of 10 bits. Therefore 320 ps would be the time interval for one bit transfer.

As per the budget, the maximum possible skew is 40 bits in serial lanes. Since 10 bits are stored in a single location at the receiver side, 10 bits represent one write pointer. It means the $\{A\}$ code group in all four lanes can be misaligned by maximum of four write pointers at the FIFO. Therefore the FIFO depth required by the lane alignment is 5.

2.1.4 Clock rate compensation

The worst case scenario is when the data packets have their maximum size and back to back with minimum Inter Packet Gap (IPG). In addition, the maximum clock rate difference between the read and write pointers has to be considered. The read and write pointer difference should be in its maximum of ± 100 PPM at the worst case operating condition [5: Clause 48.1.4], [12]. It means the read pointer clock can be +100PPM and the write pointer can be at -100PPM or vice versa. In both cases, the difference between read and write pointers is 200PPM in its worst scenario. It can be translated into one read-write pointer difference out of 5000 clock cycles.

Next, we consider the worst case operation in terms of data packet size with minimum IPG. Even though the IEEE std. is limiting the max frame size to 1518 bytes [5: Clause 4.4.2], some applications may use jumbo data packets which may

consist up to 9 Kbytes of data [17] – [19]. For this analysis we consider the maximum jumbo data packet size of 10 Kbytes. In order to transfer 10 Kbytes of data packet by four lanes, we need 2500 clock cycles. It means every 2500 cycles, we have an opportunity to insert or delete one $\|R\|$ code group in the IPG to compensate the clock rate difference.

Since one read-write pointer difference can occur only after 5000 clock cycles at ± 100 PPM, there will be at least two IPGs before one read-write pointer difference occurs. One pointer location is required for the cross-domain synchronization of the FIFO control signals. Therefore it can be concluded that the minimum of two FIFO locations are sufficient between the add request of the $\|R\|$ code group in IPG and the FIFO empty condition. Similarly the minimum of two FIFO locations are sufficient between the delete request of the $\|R\|$ code group in IPG and the FIFO overflow condition.

2.1.5 Optimal receiver FIFO parameters for 10GBASE-X PCS sub layer [11]

A FIFO bank consists of four FIFOs with the width of 10 bits is required by the design. The FIFO depth of 8, is sufficient to handle the ± 100 PPM clock rate difference. The FIFO read and write pointers have to be operated in a round-robin method. If the FIFO depth is selected to be 8, the following 2 rules are applied:

1. When the read-write pointer difference is lesser than three, the insert request should be raised to add a $\|R\|$ code group. The FIFO empty condition is when the current pointer difference becomes zero and previous pointer difference is one.
2. When the read-write pointer difference is greater than five, the delete request should be raised to remove one $\|R\|$ code group. When the previous pointer difference is seven and current pointer difference becomes zero, the FIFO overflow condition is met.

The FIFO is on a free run when the read write pointer difference is between three and five. The initial setting for read pointer can be at four. It means the system can start reading the data from the FIFO bank after writing four locations.

2.2 40GBASE-R Systems

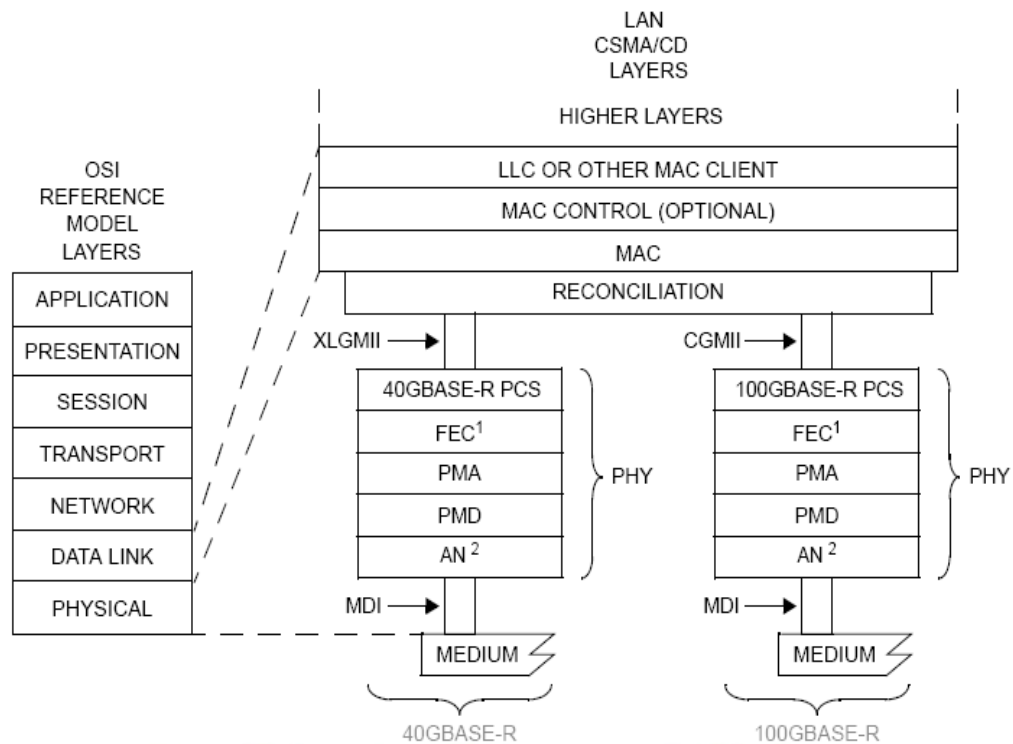


Figure 2.4: Positioning of 40Gb Ethernet

Source: Clause 80.13 [5]
www.lib.mrt.ac.lk

Figure 2.4 shows the positioning of different sub layers of 40GBASE-R. The 40GBASE-R refers to a family of Physical Layer implementations based upon 64B/66B data coding method. The 40GBASE-R PCS performs encoding (decoding) of data from (to) the 40Gb/s Media Independent Interface (XLGMII) to 64B/66B code blocks, distribution of data to multiple lanes and transmission the encoded data to the PMA. 40GBASE-R PCS is using four encoded bit streams to communicate with PMA [5]. Therefore lane synchronization has to be carried out at the receiver side and at the receiver, one FIFO is required by each lane to store data received. Hence a bank of four FIFOs needs to be employed at PCS of the receiver. XLGMII sub layer is the corresponding counterpart of XGMII for 10GBASE-X [15].

Functional block diagram of 40GBASE-R PCS is shown in **Error! Reference source not found.** PCS uses eight octet wide data path (RXD <63:0>, TXD<63:0>)

and TXC<7:0>, RXC<7:0> signals to communicate with XLGMII side. When communicates with PMA, PCS uses four encoded serial bit streams.

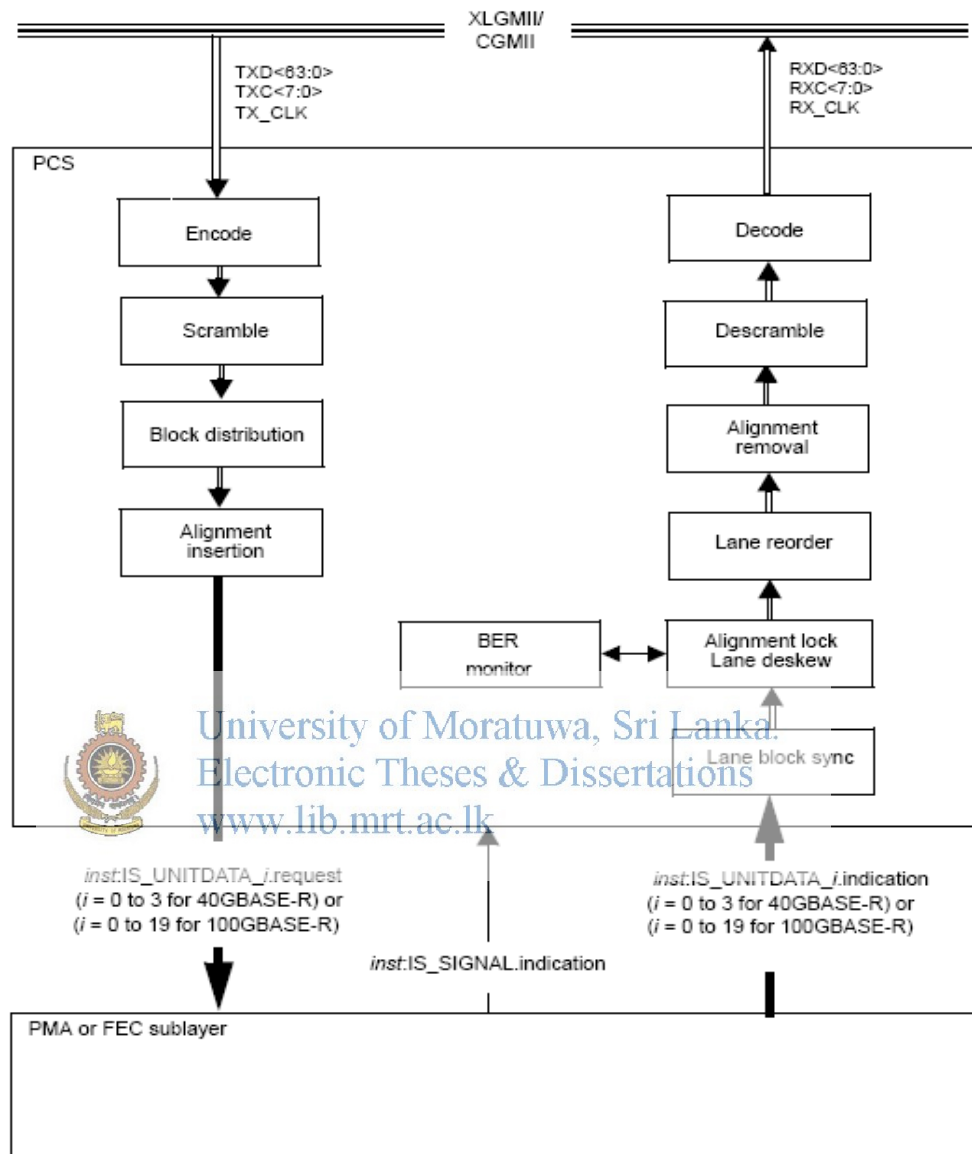


Figure 2.5: Functional block diagram 40GBASE-R physical layer

Source: Clause 82.1.5 [5]

Input Data	S y n c	Block Payload							
Bit Position:	0 1 2	65							
Data Block Format:									
D ₀ D ₁ D ₂ D ₃ D ₄ D ₅ D ₆ D ₇	01	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇
Control Block Formats:		Block Type Field							
C ₀ C ₁ C ₂ C ₃ C ₄ C ₅ C ₆ C ₇	10	0x1E	C ₀	C ₁	C ₂	C ₃	C ₄	C ₅	C ₆ C ₇
S ₀ D ₁ D ₂ D ₃ D ₄ D ₅ D ₆ D ₇	10	0x78	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇
O ₀ D ₁ D ₂ D ₃ Z ₄ Z ₅ Z ₆ Z ₇	10	0x4B	D ₁	D ₂	D ₃	O ₀	0x000_0000		
T ₀ C ₁ C ₂ C ₃ C ₄ C ₅ C ₆ C ₇	10	0x87		C ₁	C ₂	C ₃	C ₄	C ₅	C ₆ C ₇
D ₀ T ₁ C ₂ C ₃ C ₄ C ₅ C ₆ C ₇	10	0x99	D ₀		C ₂	C ₃	C ₄	C ₅	C ₆ C ₇
D ₀ D ₁ T ₂ C ₃ C ₄ C ₅ C ₆ C ₇	10	0xAA	D ₀	D ₁		C ₃	C ₄	C ₅	C ₆ C ₇
D ₀ D ₁ D ₂ T ₃ C ₄ C ₅ C ₆ C ₇	10	0xB4	D ₀	D ₁	D ₂		C ₄	C ₅	C ₆ C ₇
D ₀ D ₁ D ₂ D ₃ T ₄ C ₅ C ₆ C ₇	10	0xCC	D ₀	D ₁	D ₂	D ₃		C ₅	C ₆ C ₇
D ₀ D ₁ D ₂ D ₃ D ₄ T ₅ C ₆ C ₇	10	0xD2	D ₀	D ₁	D ₂	D ₃	D ₄		C ₆ C ₇
D ₀ D ₁ D ₂ D ₃ D ₄ D ₅ T ₆ C ₇	10	0xE1	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	C ₇
D ₀ D ₁ D ₂ D ₃ D ₄ D ₅ D ₆ T ₇	10	0xFF	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆

Figure 2.6: 64B/66B block formats

Source: Clause 82.2.3.3 [5]



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2.2.1 Idle Control Character /I/

Idle control characters (/I/) are transmitted when idle control characters are received from the XLGMII. Idle control characters may be added or deleted by the PCS to adapt between clock rates. In order to support deskew and reordering of individual PCS lanes at the receive PCS, alignment markers are added periodically to each PCS lane. The transmit process must delete idle control characters or sequence ordered sets to accommodate the transmission of alignment markers. The PCS receive process insert /I/ characters in order to accommodate any rate differences due to the removal of alignment markers.

Table 2.4: Control codes

Control character	Notation	XLGMII/ CGMII control code	40/100GBASE-R O code	40GBASE-R and 100GBASE-R control code
idle	/I/	0x07		0x00
start	/S/	0xFB		Encoded by block type field
terminate	/T/	0xFD		Encoded by block type field
error	/E/	0xFE		0x1E
Sequence ordered_set	/Q/	0x9C	0x0	Encoded by block type 0x4B plus O code, control codes are set to 0x00
Signal ordered_set ^a	/Fsig/	0x5C	0xF	Encoded by block type 0x4B plus O code, control codes are set to 0x00

Source: Clause 82.2.3.4 [5]

Here Also, the depth of the FIFO is decided based on two factors; Lane alignment at the receiver side and the clock rate compensation at the receiver in the worst case scenario.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2.2.2 Lane Alignment Consideration

Data distribution over multiple lanes, four in the case of 40GBASE-R, is supported in the Physical layer. Data from multiple lanes need to be aligned at the receiver. Periodic insertion of an alignment marker allows the lane alignment at the receiver.

Table 2.5: Maximum skew for PCS

PCS	Maximum Skew	Maximum Skew variation
40GBASE-R	180 ns (~1856 bits)	4 ns (~41 bits)
100GBASE-R	180 ns (~928 bits)	4 ns (~21 bits)

Source: Clause 82.2.12 [5]

Error! Reference source not found. showing the skew requirements for 40GBASE-R PCS is taken from IEEE Std 802.3 Section 6, clause 82.2.12 [5]. According to Table 2.5 maximum possible skew at PCS receive is 180 ns and the maximum skew variation [16] at PCS receive is 4 ns. Hence the maximum skew at PCS receive is

expected to be a value lying in the range of 176 ns to 184 ns. For the design optimization purpose we can consider the maximum possible skew at the PCS receive to be 184 ns.

For PMA sub layers supporting 40GBASE-R interfaces, the number of PCSLs z is 4, and the nominal signaling rate R of each PCSL is 10.3125 GBd. [5: Clause 83.5.2]. Signaling rate of 10.3125 GBd results in a Unit Interval (UI) time of,

$$\frac{1}{10.3125} = 96.9697 \text{ ps} \quad (1)$$

Therefore 184 ns corresponds to,

$$\frac{184 \text{ ns}}{96.9697 \text{ ps}} = 1897.49 \text{ UI} \quad (2)$$

In the PCS sub layer 64B/66B encoding is used so 66 bits are stored in a single FIFO location. Therefore in order to accommodate 1897.49 UIs,

$$\frac{1897.49}{66} = 28.7498 = 29 \text{ locations} \quad (3)$$

29 FIFO locations are required. This means alignment markers at the four serial lanes can be misaligned by a maximum of 29 FIFO locations/ FIFO write pointers. Therefore the minimum FIFO depth required by the lane alignment is 29. Hence we can implement a bank of four FIFOs one per each lane, with a depth of 32 locations.

2.2.3 Clock rate compensation in worst case scenario

The worst case scenario is met when the data packets have their maximum size and back to back with minimum IPG. In addition to that, the maximum clock rate difference between the read and write pointers has to be considered. Maximum signal rate, per lane for 40GBASE-R is 10.3125 ± 100 PPM [5: Clause 85.8.4], [12]. That is the read - write pointer difference should be in its maximum of ± 100 PPM at the worst case operating condition. It means the read pointer clock can be +100PPM and the write pointer can be at -100PPM or vice versa. In both cases, the difference between read and write pointers is 200PPM in its worst scenario. 200 PPM implies

200 pointer differences per 1000 000 clocks. Therefore one read write clock difference is anticipated for 5000 clock cycles. In order to compensate for this potential single clock difference, there should be at least one IPG for every 5000 clock cycles.

Therefore maximum data packet may span across 5000 clock cycles. Each data packet starts with eight bit start character. Data packet will be terminated with terminator character and idle characters will follow. Hence the maximum amount of effective data that can be supported, while conforming the requirements of maximum clock rate difference is 39.994 Kbytes (Figure 2.7Figure 2.1).

<p>Clk1: S0 D1 D2 D3 D4 D5 D6 D7</p> <p>Clk2- Clk4999:8 data bytes on each</p> <p>Clk5000-D0 D1 D2 T3 I4 I5 I6 I7</p> <p>7 + (8 * 4998) + 3 = 39,994 Kbytes</p>

Figure 2.7: Formation of maximum data packet



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Idle control characters may be added / deleted by PCS to adapt between clock rates. /I/ insertion / deletion shall occur in groups of eight [5: Clause 82.2.3.6]. As per the specification minimum IPG is 96 bits or 12 octets [5: Clause 4.4.2]. Minimum IPG guarantee full column of /I/ characters for only one lane per IPG. Therefore clock rate compensation needed to be carried out at the upper level of the receiver, after all the lanes are aggregated.

For the clock rate compensation purposes a separate FIFO will be implemented at the upper level of the receiver model. The width of this FIFO should be 66 bit in order to accommodate 64B/66B encoded data. One pointer location is required for the cross-domain synchronization of the FIFO control signals. Therefore it can be concluded that the minimum of two FIFO locations are sufficient between the add request of the Idle /I/ code group in IPG and the FIFO empty condition. Similarly the minimum of two FIFO locations are sufficient between the delete request of the /I/ code group in IPG and the FIFO overflow condition. For uninterrupted free running of data we are proposing a FIFO depth of eight locations.

2.2.4 Optimization of receiver FIFO parameters for 40GBASE-R PCS sub layer

Potential optimization for the 40GBASE-R PCS sub layer receiver FIFO can be summarized as follows. The proposed FIFO scheme consists of FIFOs at two levels of the receiver model. A FIFO bank consists of four FIFOs with the width of 66 bits is required by the design for deskew at each lane. The FIFO depth of 32 is sufficient to handle lane misalignment introduced by skew parameters. A separate FIFO employed at the upper level of the receiver model is used for clock rate compensation. Here again width of 66 is needed to accommodate 64B/66B encoded data. Depth of eight locations is sufficient to handle ± 100 PPM clock rate difference. In both cases received data will be stored in to the FIFO using the recovered clock from the serial lane; that is the write pointer is controlled by the recovered clock. Local system clock at the receiver is used to read stored data, hence to control the read pointer. The read and write pointers of the FIFO operate in round robin method. The following two rules, same as in the case of 10G are adhered with the clock rate compensation FIFO with depth of 8 locations.

1. The insert request to add a /I/ code group should be raised, whenever the read-write pointer difference is lesser than three. FIFO empty condition is met when the current pointer difference is zero and the previous pointer difference is one.
2. The delete request to remove /I/ code group should be raised whenever the pointer difference is exceeding five. When the previous pointer difference is seven and the current pointer difference is zero the FIFO overflow condition is met.

The FIFO is in free run condition when the read-write pointer difference is between three and five.

3. MODELING PCS SUB LAYER OF 40GBASE-R SYSTEMS

This chapter describes the proposed implementation for 40GBASE-R PCS sub layer. Transmitter and receiver models that have been implemented for the simulation are detailed separately in the forthcoming sections.

3.1 40GBASE-R PCS Sub Layer Modeling

In order to verify proper functionality of the FIFO design 2.2.4 , 40GBASE-R PCS sub layer transmitter and receiver models conforming IEEE 802.3ba specification need to be identified and implemented. Also we have to recognize the proper location in the receiver model in which the FIFO scheme should be inserted [15].

3.2 40GBASE-R PCS Sub Layer Transmitter

Figure 3.1 depicts the adopted transmitter model for 40GBASE-R PCS sub layer. The transmitter was modeled referring and conforming IEEE802.3, clause 82 [5]. PCS sub layer transmitter, transmits data received from the XLGMII interface to the PMA sub layer (refer Figure 2.4 for the positioning of different sub layers of the 40G systems). The communication between XLGMII and PCS during transmit process is done using eight octet wide data path (TXD <63:0>) and by the transmit control signals (TXC <7:0>) delimiting data octets. The other end communication with the PMA is carried out using four encoded serial bit streams. The different blocks within the transmitter are used to provide the packet mapping between XLGMII format and PMA service interface format.

The transmit clock (TX_CLK) which provides timing reference for the transfer of TXC and TXD signals should be one-sixty-fourth of the MAC transmit data rate [5: Clause 81.3.1.1] which is 40 Gb/s for 40G [5: Clause 80.1.2].

Hence,

$$TX_{CLK} = \frac{40\text{ G}}{64} = 625\text{ MHz} \quad (4)$$

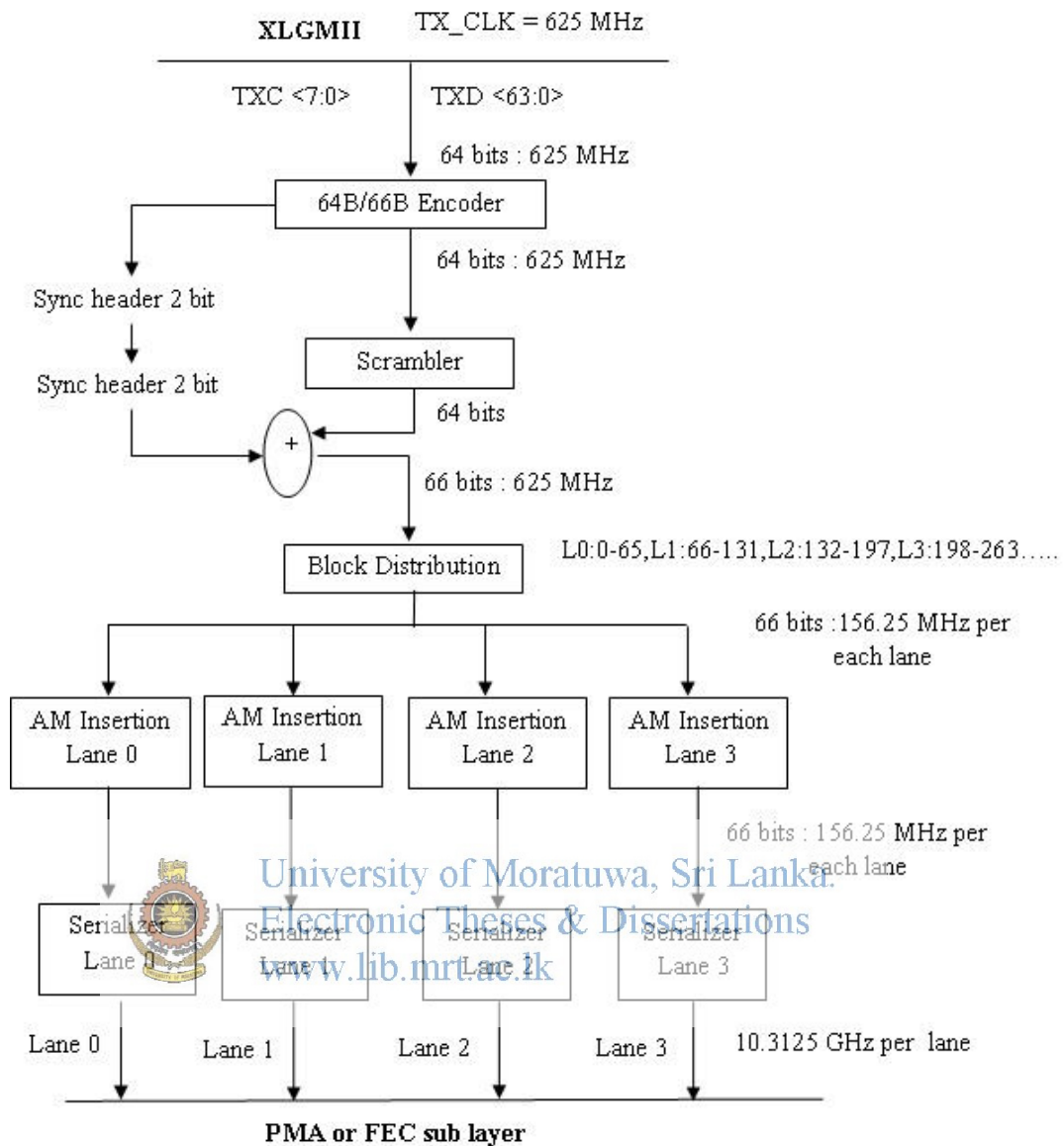


Figure 3.1: 40GBASE-R Transmitter Model

3.2.1 64B/66B Encoder [13]

The encoder maps one XLGMII data transfer, that is TXD<63:0> and TXC<7:0> to one 66 bit block (Figure 3.2). The sync header field of the 66 bit block is derived based on corresponding TXC signal. Depending on the content, remaining portion of the 66bit block is derived based on TXD, TXC or both (

Figure 2.6).

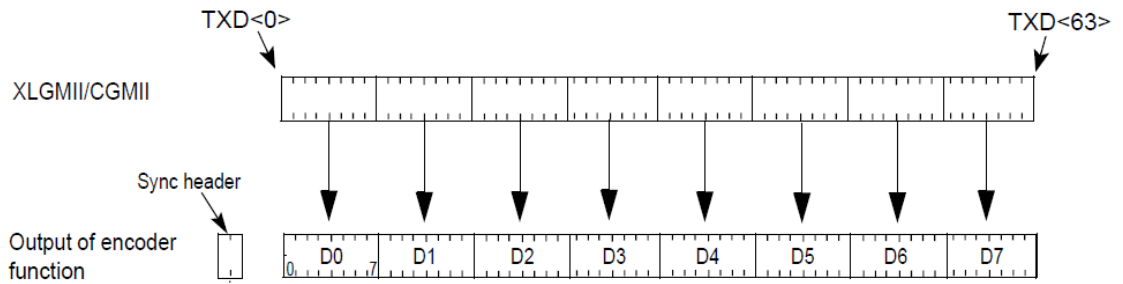


Figure 3.2: Encoder Output

Source: Clause 82.2.3.2 [5]

3.2.2 Scrambler [5: Clause 82.2.5, 14]

The pay load of the 66 bit block (bit 2: bit 65) is scrambled with a self synchronizing scrambler. The sync header bypasses the scrambler. The scrambler polynomial is given by,

$$G(x) = 1 + x^{39} + x^{58} \quad (5)$$

The scrambler implementation is shown in Figure 3.3.

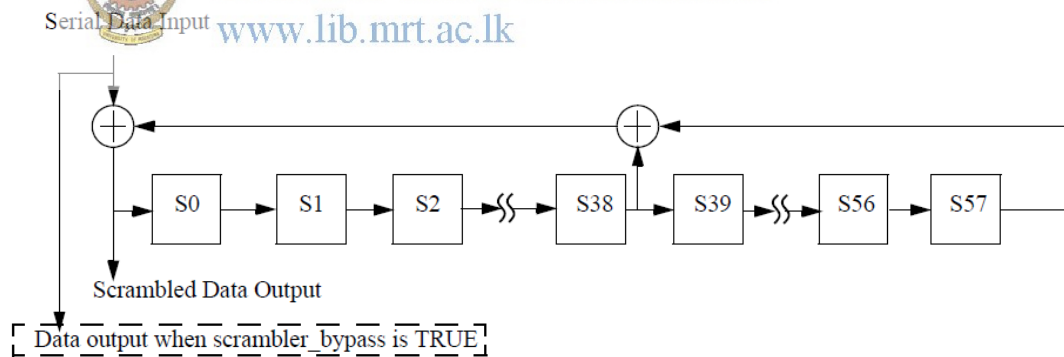


Figure 3.3: Scrambler

Source: Clause 49.2.6 [5]

3.2.3 Block distribution [5: Clause 82.2.6]

Block distribution allows support of multiple physical lanes. 66 bit data blocks resulted after encoding and scrambling processes are distributed on four lanes by the

40GBASE-R PCS. This is a round robin distribution, one block at a time from the lowest PCS lane to the highest as depicted in Figure 3.4.

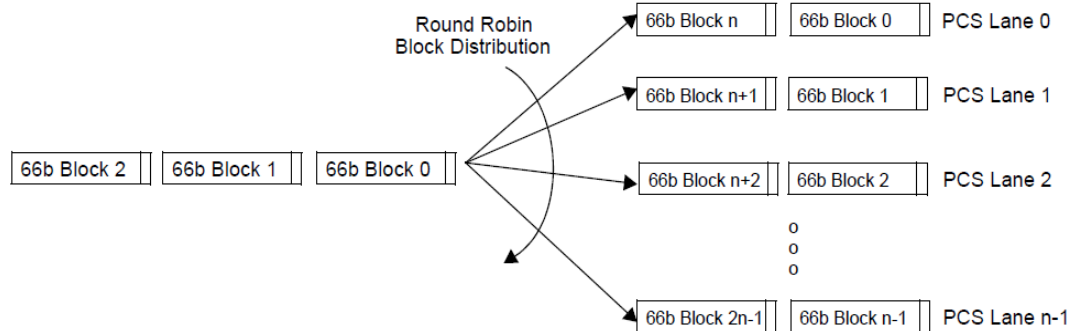


Figure 3.4: PCS Block Distribution

Source: Clause 82.2.6 [5]

In the case of 40GBASE-R systems number of lanes is four so n is equal to three. 66 bit blocks are input to the distributor at a rate of 625MHz. Therefore the resulting serial lanes will have,



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

$$625 \text{ M} / 4 = 156.25 \text{ M} \quad (6)$$

156.25 MHz of 66 bit blocks at each of the serial lanes.

3.2.4 Alignment marker insertion [5: Clause 82.2.7]

Alignment marker is an especially defined 66 bit block with sync header being ‘10’ that for a control block (Figure 3.5). Since insertion process takes place after encoding and scrambling process, at the transmitter PCS alignment markers are neither encoded nor scrambled. The content of the alignment marker block depends on the PCS lane number. M0, M1, M2 are defined specifically for each PCS lane. M4, M5 and M6 are bit wise inversion of M0, M1 and M2 respectively. Method of calculating bit interleaved parity (BIP3) field is detailed in Clause 82.2.8 [5]. BIP7 is the inversion of BIP3. This field is used as a fast measure of bit error ratio of a given PCS lane.

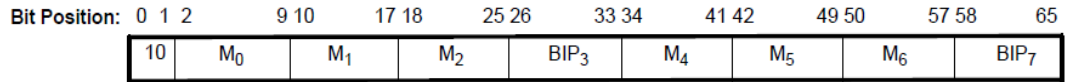


Figure 3.5: Alignment marker format

Source: Clause 82.2.7 [5]

Alignment Markers are added to each PCS lane in order to support deskew and reordering PCS lanes at the receive PCS. These are inserted to all PCS lanes at the same time and periodically (Figure 3.6 and Figure 3.7). On each lane, after every 16383 66 bit blocks, the lane specific alignment marker block is inserted.

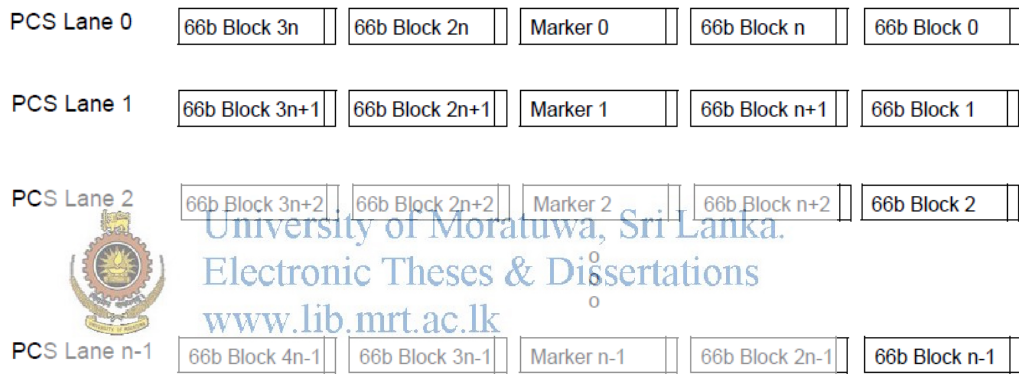


Figure 3.6: Alignment Marker Insertion

Source: Clause 82.2.7 [5]

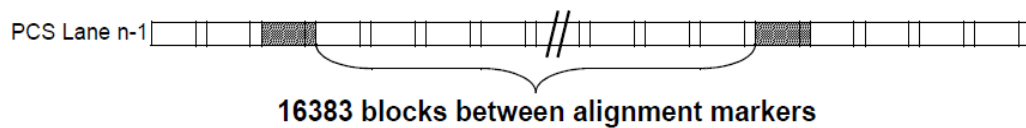


Figure 3.7: Alignment Marker insertion period

Source: Clause 82.2.7 [5]

As shown in Figure 3.1 alignment marker insertion will be carried out independently for each PCS lane.

3.2.5 Serializer

Transmitter serializes the 66 bit blocks prior transmitting. Serializer is having input of 66 bit blocks at the rate of 156.25 MHz.

$$156.25 M * 66 = 10.3125 G \quad (7)$$

Hence the serializer is outputting bits at a rate of 10.3125 GHz. Our model requires one module of this type per each lane (Figure 3.1).

3.3 40GBASE-R PCS Sub Layer Receiver

Adopted Receiver implementation is illustrated in Figure 3.8. Receiver PCS decodes the serial data stream received from PMA sub layer to produce RXD and RXC signals to be transmitted to XLGMII interface.

3.3.1 Deserializer

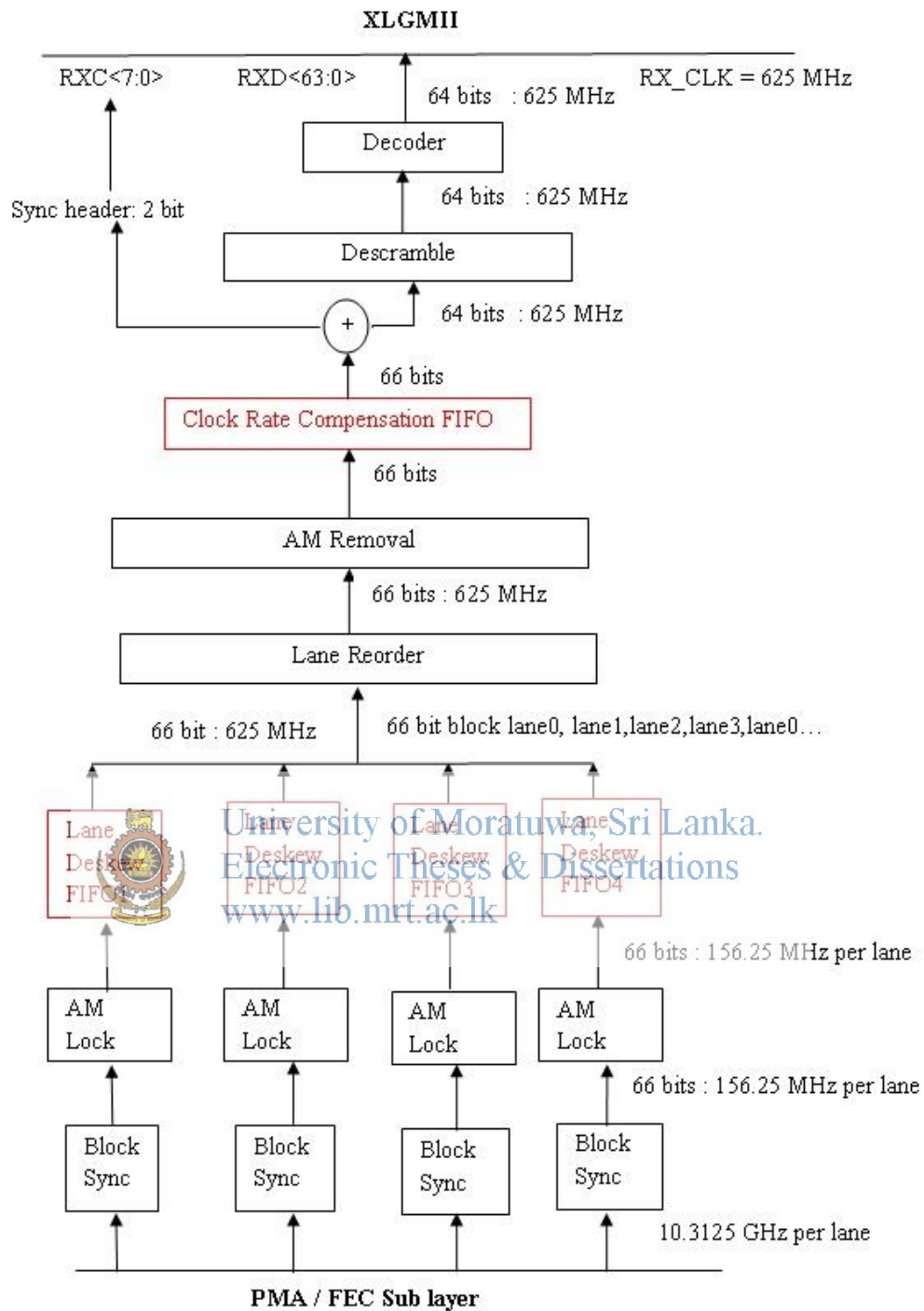
Deserializer block group the incoming bit stream to generate 66 bit parallel data. 10.3125 GHz serialize clock will produce 66 bit blocks at a rate of 156.25 MHz As Figure 3.8 shows one module per each lane is implemented separately.

3.3.2 Block synchronization [5: Clause 82.2.11]

Block synchronization process monitors the incoming data to identify block lock condition. It waits for 66 valid sync headers to obtain the block lock state. A module operating at 156.25 MHz at each lane is required.

3.3.3 Alignment marker lock [5: Clause 82.2.12]

Alignment marker lock waits for two valid alignment markers to obtain the lock state. The process is started only after the block lock state is achieved on a particular PCS lane. In alignment marker lock stage the PCS lane number received can be identified, due to the fact that alignment makers are lane specific.



3.3.4 Lane deskew FIFO

We are introducing the lane deskew FIFO at this stage of the receiver PCS. FIFO per each lane is implemented with the identified parameter values. Optimization for

parameters, width of 66 bits and depth of 32 locations are derived such that the maximum skew for 40GBASE-R as specified in **Error! Reference source not found.** is tolerated. FIFO read and write operations are carried out in 156.25 MHz lane clocks.

3.3.5 Lane reorder [5: Clause 82.2.13]

PCS reorders the received PCS lanes according to the PCS lane number. This was identified at the alignment marker lock stage.

3.3.6 Alignment marker removal [5: Clause 82.2.14]

In this stage PCS lanes are multiplexed together in the proper order to obtain the original block stream that was at the transmitter. Alignment markers are deleted from the block stream since they are not needed further. The BIP field in the alignment markers can be monitored to obtain error ratio related parameters.

3.3.7 Clock rate compensation FIFO

As discussed in section 2.2.3 clock rate compensation is carried out at the upper level of receiver model. Addition/deletion of eight consecutive /I/ code blocks (a full column of /I/s) is done in order to compensate clock rate differences. Hence the FIFO should be implemented after the data from the four lanes get aggregated. Aggregated data, with the capability of addition/deletion of full column of /I/s is formed after the lane reorder and alignment removal stages. So the clock rate compensation FIFO is positioned at this stage of the receiver model. In order to accommodate encoded data the width the FIFO should be 66 bits and as deduced by the analysis in section 2.2.3 the depth of eight is sufficient.

3.3.8 Descrambler [5: Clause 82.2.15]

Descrambler processes the payload of the 66 bit block (bit 2 to bit 65) to reverse the effect of the scrambler. The descrambler polynomial is identical to the scrambler polynomial.

$$G(x) = 1 + x^{39} + x^{58} \quad (8)$$

Descrambler implementation is shown in Figure 3.9.

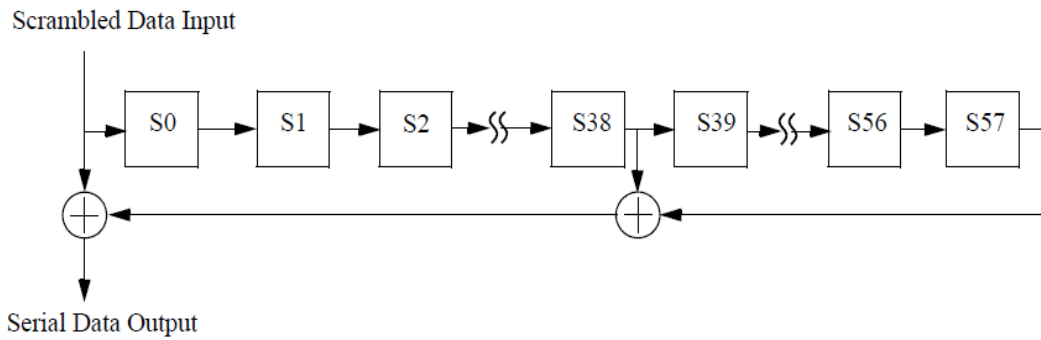


Figure 3.9: Descrambler

Source: Clause 49.2.10 [5]

3.3.9 Decoder [15]

Decoder uses the sync header two bits from the incoming 66 bit block to determine the $RXC\langle 7:0 \rangle$. Depending on the control block, to determine the corresponding RXC signal decoder may need to consider the data payload as well. One 66 bit block will be mapped into one XLGMII transfer, which is $RXC\langle 7:0 \rangle$ and $RXD\langle 63:0 \rangle$ (Figure 3.10).

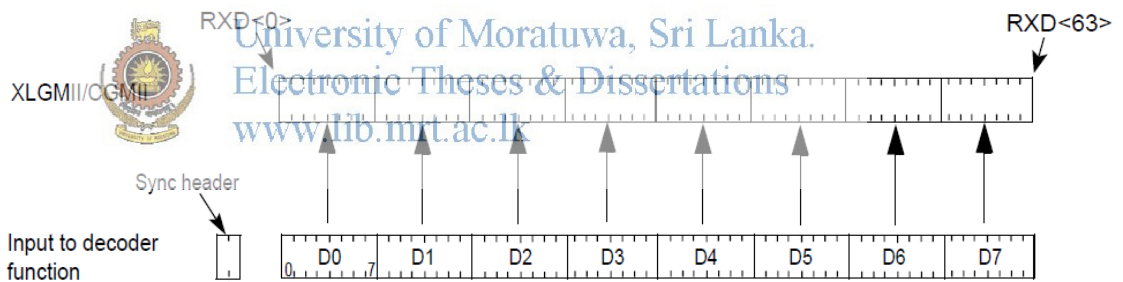


Figure 3.10: Decoder Output

Source: Clause 82.2.3.2 [5]

It should be noted that the optimized FIFO parameters suggested in 2.2.4 are valid for any implementation of 40GBASE-R PCS sub layer conforming IEEE802.3ba specification. The transmitter and receiver models described in 3.2 and 3.3 are only one possibility of such system models introduced for the purpose of functional verification of the suggested design. In other words, As far as it is abided by the IEEE802.3ba standard, PCS sub layer modeling does not put any limitations on the proposed FIFO design.

4. SIMULATION, RESULTS AND ACHIEVEMENTS

This chapter introduces the simulation criteria used for functional verification of the analysis output discussed in the previous chapter. In the latter part of the discussion, the simulation results will be analyzed to verify the functionality of the proposed designs. Discussion flows in twofold manner separately for 10GBASE-X and 40GBASE-R.

4.1 Simulation Environment

Both 10GBASE-X and 40GBASE-R systems are modeled using Verilog HDL. These systems were simulated using ModelSim advanced simulation and debugging tool from Mentor Graphics Corporation.

Two scenarios FIFO Empty and FIFO Full for both systems were simulated. FIFO Empty situation is viable when FIFO read clock is faster than FIFO write clock. FIFO can become full when FIFO write clock is faster than the FIFO read clock. Both these conditions are undesirable for the proper operation of the systems. Therefore operations at these situations were monitored.

At FIFO Empty potential scenario, the signals of interest to be monitored were, Read – Write pointer difference, FIFO Empty Flag and the Insert Request. Read – Write pointer difference, FIFO Full Flag and the Delete Request were the desired signals at the FIFO Full viable scenario.

The signals were captured using inbuilt “Wave Window” utility of ModelSim.

4.2 10GBASE-X Simulation and Results

For 10GBASE-X systems, data packets of size 10 Kbytes and 312.5 MHz \pm 200 PPM clock rate difference were used. FIFOs of 10 bit width and 8 depth locations were modeled. Desired wave forms were monitored at both FIFO Full and FIFO Empty viable situations.

4.2.1 Waveforms captured for FIFO Full viable situation

Figure 4.1 depicts FIFO Full viable simulation results. Situation is viable when Write clock is faster than the Read clock. As per the captured signals shown, whenever the pointer difference goes beyond five, the Delete Request is set high, otherwise it remains low. When the Delete Request is high, an existing $||R||$ code group in the IPG

is removed, so the Read – Write pointer difference decreases avoiding FIFO being Full. Throughout the whole simulation FIFO Full signal remains low; indicating FIFO was not full all the time. So the proper operation is achieved.

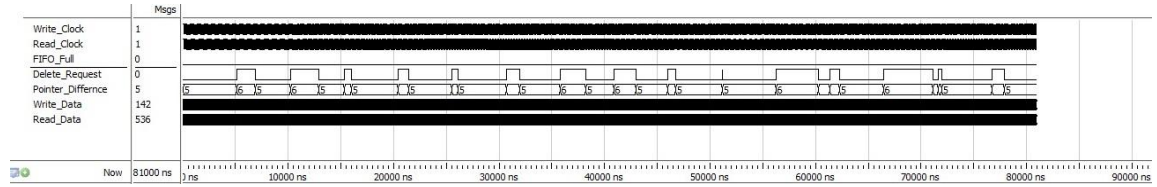


Figure 4.1: Signals captured from ModelSim Wave simulation in a FIFO Full condition viable scenario for 10GBASE-X

4.2.2 Waveforms captured for FIFO Empty viable situation

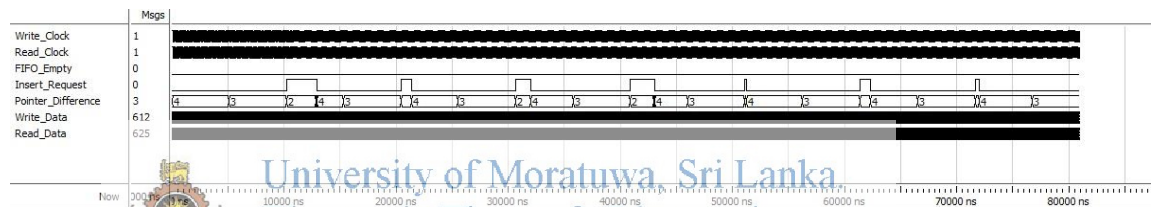


Figure 4.2: Signals captured from ModelSim Wave simulation in a FIFO Empty condition viable scenario for 10 GBASE-X

Captured waveforms for FIFO Empty viable situation is shown in Figure 4.2. As per the waveforms whenever the pointer difference goes below three, the Insert Request is set high, otherwise it remains low. When the Insert Request is high, a ||R|| code group is added to IPG, so the Read – Write pointer difference increases avoiding FIFO being Empty. Throughout the whole simulation FIFO Empty signal remains low; indicating FIFO was not empty all the time. So the proper operation is achieved.

We have published the IEEE paper, “FIFO Design for IEEE 802.3 Standard 10GBase-X PCS and XGXS Sublayers” (ISBN: 978-1-4673-5653-4) related to this piece of work in 4th International Conference on Intelligent Systems, Modelling and Simulation, 2013 [11].

4.3 40GBASE-R Simulation and Results

For 40GBASE-R systems, data packets of size 40 Kbytes (inclusive of 39.994Kbytes of effective data) were simulated with 625 MHz \pm 100 PPM clock rate difference between receiver and transmitter clocks. A bank of four FIFOs of 66 bit width and 32 depth locations were modeled as one per each lane for lane deskew purpose. A separate FIFO with width of 66 bits and depth of eight locations was implemented for clock rate compensation. Desired wave forms were monitored at both FIFO Full and FIFO Empty viable situations.

4.3.1 FIFO Full viable situation

This situation is viable when Write clock is faster than the Read clock. The results obtained are shown in Figure 4.3.

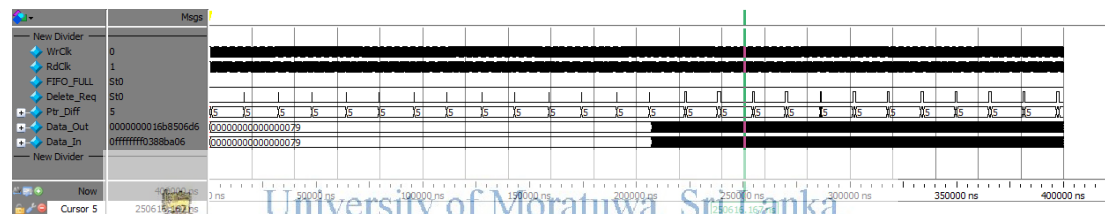


Figure 4.3: Signals captured from ModelSim & Wave simulation in a FIFO Full condition viable scenario for 40GBASE-R

Whenever the pointer difference goes beyond five, the Delete Request is set high, otherwise it remains low. When Delete Request is high, an existing /I/ idle code group in the IPG is removed, so the Read – Write pointer difference decreases avoiding FIFO being Full. As can be seen from the captured waveforms in Figure 4.3, FIFO Full signal remains low throughout simulation indicating FIFO did not meet the full condition. Hence the proper functionality is achieved.

4.3.2 FIFO Empty viable situation

When the Read clock is faster than the Write clock there is a possibility for the FIFO to be empty. Whenever the Read – Write pointer difference goes below three, the Insert Request is set high, otherwise it remains low. Figure 4.4 shows the captured waveforms of desired signals.

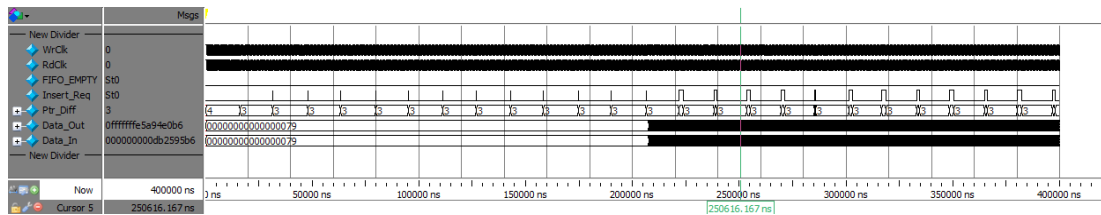


Figure 4.4: Signals captured from ModelSim Wave simulation in a FIFO Empty condition viable scenario for 40GBASE-R

When the Insert Request is high, a /I/ code group is added to IPG, so the Read – Write pointer difference increases avoiding FIFO being Empty. As seen from Figure 4.4, throughout the simulation FIFO Empty flag remains at low condition. This indicates that the FIFO did not meet the empty condition. Hence the proper operation is verified.

The 40SBASE-R PCS sub layer transmitter and receiver described respectively in sections 3.2 and 3.3 were used to model and simulate the proposed FIFO design for 40G. This optimization of FIFO parameters is valid and applicable for any other 40GBASE-R PCS sub layer receiver implemented adhering to IEEE802.3ba specification.



The IEEE paper “Optimization of Receiver FIFO for IEEE802.3ba 40GBASE PCS Sub Layer” (PID: 1570228452) will be published in 30th International Conference on Information Networking, 2016 regarding this work on 40GBASE PCS sub layer receiver.

4.4 Publication List

- [1.] *FIFO Design for IEEE 802.3 Standard 10GBase-X PCS and XGXS Sublayers* with Thayaparan, S. Proceedings of 4th International Conference on Intelligent Systems, Modelling and Simulation, IEEE Computer Society, Bangkok, 2013.
- [2.] *Optimization of Receiver FIFO for IEEE 802.3ba 40GBASE PCS Sublayer* with Thayaparan, S. Proceedings of 30th International Conference on Information Networking, Kota Kinabalu, Malaysia, 2016.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

5. CONCLUSIONS AND FUTURE WORKS

Chapter 5 Conclusions and Future Works is presented under two parts, conclusions and future works. Conclusions winds up the facts that can be formulated with the current results. Latter subsection future works suggests for potential improvements.

5.1 Discussion and Conclusions

5.1.1 For 10 GBASE-X

For 10 GBASE-X systems, a FIFO bank consists of four FIFOs with the width of 10 bits is required by the design. The FIFO depth of eight is sufficient to handle ± 100 PPM clock rate difference. With these parameters insert request to add ||R|| code group should be raised whenever the read-write pointer difference is lesser than three. When read-write pointer difference is greater than five, delete request should be raised to remove one existing ||R|| code group. The FIFO is on a free run when read-write pointer difference is between three and five. This design can also support jumbo data packets of 9K bytes. This scheme was simulated and proper functionality is verified.

In general, the XGXS and PCS sub layer ASIC designs for 10GBASE-X FIFO banks use 32 depth and 10bit wide FIFOs. A bank of four such FIFOs requires 1280 ($32*10*4$) registers (or RAM bits). The proposed FIFO size reduces the requirement to 320 ($8*10*4$) registers (or RAM bits). It saves the power and silicon area of 960 registers (or RAM bits) in the ASIC design, which is a notable contribution.

5.1.2 For 40 GBASE-R

The optimization for 40GBASE-R systems, proposed and verified by this research consists of a FIFO scheme which is having FIFOs at two levels of the PCS receiver model. For deskew purpose, a FIFO bank consists of four FIFOs with the width of 66 bits is required at each individual lane. This is the constraint driven by the design specifications; four serial lanes of data at the physical layer and use of 64B/66B encoding. Depth of 32 can tolerate the maximum possible skew specified by the standard.

The scheme employs another FIFO at the upper level of the receiver model to compensate for clock rate differences, with the width of 66 bits. As per our analysis and simulation results a FIFO depth of 8 is sufficient to handle ± 100 PPM clock rate difference. With these parameters insert request to add a /I/ code group should be raised, whenever the read-write pointer difference is lesser than three. Whenever the pointer difference is exceeding five, delete request should be raised to remove an existing /I/ code group. The proposed design was simulated and verified.

Also as per analysis in 2.2.3 this design can support super jumbo data packets up to 39.994 Kbytes of data. In order for the convenient integration with the upper layers of IEEE802.3 it is advisable to implement this maximum super jumbo data packet in two packets; as a packet of 32 Kbytes followed by a secondary packet consists of remaining 7.994 bytes. Further even though there is no formal definition for super jumbo packets and they are not standardized in IEEE std 802.3, in order to maintain same bit error rate accuracy, extended frame sizes should not extend beyond 11455 bytes [20], [21]. Both cases suggest for packet sizes lower than the analyzed maximum, thus allow IPG becoming available before 5000 clock cycle span. So in all these situations proposed FIFO scheme and parameters are capable of handling the maximum clock rate difference specified by the standard.

The analysis as well as the FIFO scheme proposed by this research is applicable and valid for any other implementation of 40GBASE-R systems conforming IEEE802.3ba standard. The suggested and verified design optimization may reduce the number of registers (or RAM bits) required. This may lead to significant savings in terms of power and silicon area of ASIC design.


5.2 Recommendation for Future Work

Currently the proposed FIFO designs for both 10GBASE-X and 40GBASE-R are simulated and functionality is verified based on the simulation results. It is recommended to emulate the systems in real hardware, such as on a FPGA (Field programmable Gate Array) board and verify the proper behavior on hardware.

Further, analysis and verification of optimized FIFO design parameters for 100GBASE systems can be addressed.

REFERENCE LIST

- [1.] R. Seifert and J. Edwards, *The All-New Switch Book, The Complete Guide to LAN Switching Technology*, 2nd ed., New York: John Wiley & Sons, 2011.
- [2.] Ethernet Task Force (2011, Mar), Home page of the IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force, [Online]. Available:
<http://grouper.ieee.org/groups/802/3/ba/public/index.html>.
- [3.] C. Cole, J. D'Ambrosia, C. DiMinico, H. Frazier, A. Healey, J. Jaeger, J. Jewell, M. Nowell, and S. Trowbridge, (2007, Nov.). An overview: The next generation of ethernet, IEEE 802.3-HSSG Meeting [Online]. Available:
<http://www.ieee802.org/3/hssg/public/nov07/index.htm>
- [4.] P. Reviriego, B. Huiszoon, V. L'opez, R. B. Coenen, J. A. Hern'andez, and J. A. Maestro, "Improving Energy Efficiency in IEEE 802.3ba High-Rate Ethernet Optical Links", *IEEE Syst. J.*, vol. 17, no 2, pp. 419-427, March-April 2011.
- [5.] IEEE Standards for Ethernet, IEEE Std 802.3™, 2012.
- [6.] Merilee Ford et al., "Chapter 7 Ethernet Technologies," in *Internetworking technology Overview*, Cisco Systems, 1999.
- [7.] Mark Gustlin et al. (2007 September). 100GE and 40GEPCS Proposal [Online]. Available:
http://grouper.ieee.org/groups/802/3/hssg/public/sept07/gustlin_01_0907.pdf
- [8.] John Ambrosia et al., *40 Gigabit Ethernet and 100 Gigabit Ethernet Technology Overview*, ethernet alliance, June 2010.

- [9.] Mark Gustlin, 40 and 100 Gigabit Ethernet PCS and PMA Overview, Cisco Systems Inc., October 2010.
- [10.] Jorg Sommer et al., “Ethernet – A Survey on its fields of Application,” in IEEE Communications Surveys and Communications, 2010, Vol. 12, No. 2.
- [11.] Subramaniam Thayaparan and Anuradha Nanayakkara, “FIFO Design for IEEE 802.3 Standard 10GBase-X PCS and XGXS Sublayers,” in 2013 4th International Conference on Intelligent Systems, Modelling and Simulation, Bangkok, 2013, pp. 589-591
- [12.] Faisal Dada and Norbert Folkens. IPG Considerations [Online]. Available: http://grouper.ieee.org/groups/802/3/ba/public/may08/folkens_01_0508.pdf#page=3
- [13.] Sowmya S Luckloor, “Introduction to 10 Gigabit 64B/66B”, October, 2001.  University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk
- [14.] Kushan B Vadawala. “Universal Scrambler by using Verilog HDL”, International Journal of Engineering Sciences and Research Technology, March, 2014.
- [15.] W.P. Ranjaula et al., “Implementation Techniques for IEEE 802.3ba 40Gbps Ethernet Physical Coding Sublayer (PCS)”, in 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2015.
- [16.] Tom Warland. (2011, December 14). Understanding Skew in 100GBASE-R4 applications [Online]. Available: http://www.eetimes.com/documnet.asp?doc_id=1279294

- [17.] Robert Winter et al., Ethernet Jumbo Frames, ethernet alliance, November 12 2009.
- [18.] Wikipedia, the free encyclopedia (2015, February 11). Jumbo frames [online]. Available: https://en.wikipedia.org/wiki/Jumbo_frame
- [19.] Sarath Pillai, (2014, July 15). What is a Jumbo Frame in Ethernet [Online]? Available: <http://www.slashroot.in/networking>
- [20.] Extended Frame Sizes for Next Generation Ethernet [Online]. Available: https://www.psc.edu/~mathis/MTU/AlteonExtendedFrames_W0601.pdf
- [21.] Doug Raid. (2007, October 26). Need To Know: Jumbo Frames in Small Networks [Online]. Available: <http://www.smallnetbuilder.com/lanwan/lanwan-features/30201-need-to-know-jumbo-frames-in-small-networks?limitstart=0>
- [22.] ModelSim User's Manual, Software version 6.6d, Mentor Graphics Corporation, Oregon, 2010.
- [23.] IEEE Standard Verilog Hardware Description Language, IEEE Std 1364-2001, 2001.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Appendix A: Verilog test bench for 10GBASE-X PCS Sublayer FIFO design

```
// -----  
// This is the implementation of TestBench for Single Asynchronous  
FIFO  
// Note: FIFO FULL :-Wr_Clk>Rd_Clk; Wr_Clk T = 3199  
//       FIFO EMPTY :- Rd_Clk>Wr_Clk; Wr_Clk T = 3200  
// -----  
// timescale unit/precision  
`timescale 1ps / 1ps // each unit is 1 ps,& simulation has 1 ps  
precision  
  
module TestLaneFIFO;  
  
parameter FIFO_width = 10; // 10 bits  
parameter ADDR_bits = 3; // 3 bits  
  
// stimuli signal generation  
regtstWr_Clock, tstRd_Clock, tstFIFOReset;  
wiretstFIFO_Full, tstFIFO_Empty;  
wiretstInsert_Req, tstDelete_Req;  
wire [2:0] tstPtrDiff;  
  
reg [FIFO_width-1:0] tstWrData; // FIFO Write In data  
reg [31:0] TestData;  
reg [11:0] ClkCounter;  
  
wire [FIFO_width-1:0] tstDataOut;  
wire [FIFO_width-1:0] tstNxtRdData;  
wire [(ADDR_bits-1):0] tstFIFO_WrPtr;  
wire [(ADDR_bits-1):0] tstFIFO_RdPtr;  
wire [(ADDR_bits-1):0] tstFIFO_NxtRdPtr;  
wire [(ADDR_bits-1):0] tstCrntWrPtrGray;  
wire [(ADDR_bits-1):0] tstCrntRdPtrGray;  
wire [(ADDR_bits-1):0] tstNxtRdPtrGray;  
wire [(ADDR_bits-1):0] tstNxtNxtRdPtrGray;  
wire [(ADDR_bits-1):0] tstSyncdWrPtr;  
wiretstReadEn;  
wire [2:0] tstCounter;  
// UUT instantiation  
  
LaneFIFOUUT (  
    .Wr_Clock(tstWr_Clock),  
    .Rd_Clock(tstRd_Clock),  
    .FIFOReset(tstFIFOReset),  
    .CrntWrData(tstWrData),  
    .InsertReq(tstInsert_Req),  
    .DeleteReq(tstDelete_Req),  
    .CrntWrPtr(tstFIFO_WrPtr),  
    .CrntRdPtr(tstFIFO_RdPtr),  
    .NxtRdPtr(tstFIFO_NxtRdPtr),  
    .PtrDiff(tstPtrDiff),  
    .FIFO_Full(tstFIFO_Full),  
    .FIFO_Empty(tstFIFO_Empty),  
    // Testing purpose only  
    /// TestClkCounter,  
    .CrntRdData(tstDataOut),  
    .NxtRdData(tstNxtRdData),  
);
```



```

        .CrntWrPtrGray(tstCrntWrPtrGray),
        .CrntRdPtrGray(tstCrntRdPtrGray),
        .NxtRdPtrGray(tstNxtRdPtrGray),
        .NxtNxtRdPtrGray(tstNxtNxtRdPtrGray),
        .SyncWrPtr(tstSyncdWrPtr),
        .ReadEn(tstReadEn),
        .Counter(tstCounter)
    );

// Define characters
`defineStartChar 10'b1101101000 // K27.7
`defineTerminateChar 10'b1011101000 // K29.7
`defineCharA 10'b0011110011 // K28.3
`defineCharK 10'b0011111010 // K28.5
`defineCharR 10'b0011110100 // K28.0

`defineWrClkCycles 12'd2500 // 10k / 4 = 2500

// -----
// Initialization
// -----
initial
begin
tstFIFOReset = 1'b0; // reset FIFO
    #16010 tstFIFOReset = 1'b1;
end

// -----
// Clock signal generation
// -----
/* Notes
    FIFO_Full & Delete request condition being testing
    tstWr_Clock: 312.5 Mhz + 100 ppm => 312531.25 kHz => 3199.68 ps
    tstRd_Clock: 312.5 Mhz - 100 ppm => 312468.75 kHz => 3200.3 ps

    tstWr_Clock: 312.5 Mhz + 200 ppm => 312562.5 kHz => 3199.3 ps
    tstRd_Clock: 312.5 Mhz - 200 ppm => 312437.5 kHz => 3200.6 ps
*/

// Write pointer to be derived on received/recovered clock
initial // Clock generator
begin
tstWr_Clock = 1'b1;
forever #1599 tstWr_Clock = !tstWr_Clock; // invert every 3199.68 /
2 = 1599.84ps
end

// Read pointer to be derived on the local clock
initial // Clock generator
begin
tstRd_Clock = 1'b1;
forever #1600 tstRd_Clock = !tstRd_Clock; // invert every 3200.3 / 2
= 1600.15ps
end

// -----
// Data packet generation
// -----

```

```

always @ (posedgetstWr_Clock or negedgetstFIFOReset)
if (!tstFIFOReset)
begin
ClkCounter = 0;
end
else
begin
if (ClkCounter >= `WrClkCycles)
ClkCounter = 1;
else
ClkCounter = ClkCounter + 1;
end

//always @(ClkCounter >= 0) // start character
always @ (posedgetstWr_Clock or negedgetstFIFOReset)
if (!tstFIFOReset)
begin
tstWrData = `CharA; // idle
TestData = 32'b0;
end
else
begin
TestData = $random;
if (ClkCounter == 1)
tstWrData = `StartChar;
else if (ClkCounter == (`WrClkCycles - 3)) // 2497 = T
tstWrData = `TerminateChar;
else if (ClkCounter == (`WrClkCycles - 2)) // 2498 = A
tstWrData = `CharA;
else if (ClkCounter == (`WrClkCycles - 1)) // 2499 = R is chosen as
the second following T
tstWrData = `CharR;
else if (ClkCounter == `WrClkCycles) // 2500 = K
tstWrData = `CharK;
else
// TxData = TestData[(Lane0FIFO_width - 1):0];
tstWrData = TestData[(FIFO_width - 1):0];
end
// -----

// -----
initial
#81_000_000 $stop;
// -----
endmodule

```

Appendix B: Verilog testbench for 40GBASE-R PCS sub layer Model

```
// -----  
// File Name: TestFourtyG_PCS.v  
// Description: This is the implementation of TestBench for 40G  
transmitter  
// Input: tstPCSReset - Active Low Reset  
// Output:  
// Notes:  
// TODO:  
// -----  
  
// timescale unit/precision  
`timescale 1ps / 1ps // each unit is 1 ps, & the simulation has 1 ps  
precision  
  
module TestFourtyG_PCS;  
  
reg tstTxClock, tstPCSReset, tstPCSEn, tstRxClock;  
reg tstSerialLaneClk, tstSerialLaneWrClk, tstSerialLaneRdClk,  
NxtSerialLaneWrClk, NxtSerialLaneRdClk;  
reg [14:0] SerialClkCntr;  
reg [14:0] SerialClkCntrNxt;  
reg tstLaneWrClk, tstLaneRdClk, nxtLaneWrClk, nxtLaneRdClk;  
reg [63:0] tstTxData;  
reg [7:0] tstTxCtrl;  
  
wire [63:0] tstRxData;  
wire [7:0] tstRxCtrl;  
wire tstRxRdy, tstTxRdy;  
  
`define WrClkCycles 13'd5001  
parameter TxClkPeriodbyTwo = 10'd799;  
parameter RXClkPeriodbyTwo = 10'd800;  
parameter SerialLaneWrClkPeriodbyTwo = 6'd48;  
parameter SerialLaneRdClkPeriodbyTwo = 6'd48;  
  
parameter BlockType_S = 8'h78; // PCS /S/ = 0x78  
parameter BlockType_T0 = 8'h87; // PCS /T0/ = 0x87  
parameter BlockType_T1 = 8'h99;  
parameter BlockType_T2 = 8'hAA;  
parameter BlockType_T3 = 8'hB4;  
parameter BlockType_T4 = 8'hCC;  
parameter BlockType_T5 = 8'hD2;  
parameter BlockType_T6 = 8'hE1;  
parameter BlockType_T7 = 8'hFF;  
  
reg [63:0] TestData;  
reg [12:0] ClkCounter;  
reg [7:0] CkEdgeCounterTx;  
reg [7:0] CkEdgeCounterTxNxt;  
reg [7:0] CkEdgeCounterRx;  
reg [7:0] CkEdgeCounterRxNxt;  
  
integer i;
```



```

// -----
// UUT Instantation
// -----
FourtyG_PCS UUT_PCS (
    .TxClock(tstTxClock), // input TxClock,
    .SerialLaneWrClk(tstSerialLaneWrClk),
    .LaneWrClk(tstLaneWrClk),
    .LaneRdClk(tstLaneRdClk),
    .TxData(tstTxData), // input [63:0] TxData,
    .TxCtrl(tstTxCtrl), // input [7:0] TxCtrl,
    .PCSReset(tstPCSReset), // input PCSReset,
    .PCSEn(tstPCSEn), // input PCSEn,
    .RxClock(tstRxClock), // input RxClock,
    .SerialLaneRdClk(tstSerialLaneRdClk),
    .RxData(tstRxData), // output [63:0] RxData,
    .RxCtrl(tstRxCtrl), // output [7:0] RxCtrl
    .TxRdy(tstTxRdy), // output TxRdy,
    .RxRdy(tstRxRdy) // output RxRdy

);

// -----
// Clock signal generation
// Serial Lane clock: 10.3125 G => 96.9697 ps => T/2 = 48 ps
// tstSerialLaneClk = 24 ps is used to obtain serial lane Rd Wr
// clocks of T/2 = 48 ps with +/- 100 ppm
// -----
initial // Clock Generator
begin
    tstSerialLaneClk = 1'b0;
    forever #24 tstSerialLaneClk = ~tstSerialLaneClk; // invert
every (1/(10.3125) / 4 = 24.2424
end

// -----
// Clock signal generation
// Serial Lane clock: 10.3125 G => 96.9697 ps
// 100 ppm => 100 for 10^6 => 1 for 10 000 clks
// stop the 10001 clk -> slower clk
// having the 10001 as a normal clock -> faster clk
// tstSerialLaneWrClk(47) > tstSerialLaneRdClk(49) => FIFO_FULL
// tstSerialLaneWrClk(49) < tstSerialLaneRdClk(47) => FIFO_EMPTY
// Generated period of both tstSerialLaneWrClk & tstSerialLaneWrClk
are 96 ps
// Expected period of both tstSerialLaneWrClk & tstSerialLaneWrClk
are 96.9697 ps
// -----
initial
begin
    tstSerialLaneWrClk = 1'b0;
    tstSerialLaneRdClk = 1'b0;
    SerialClkCntr = 15'd0;
end

always @(*) // Combinational logic
begin
    if (SerialClkCntr >= 15'd20001) // counts 0 : 20001
        SerialClkCntrNxt <= 15'd0;

```



```

        else
            SerialClkCntrNxt <= #1 SerialClkCntr + 15'd1;
        end
    always @ (posedge tstSerialLaneClk)
        begin
            SerialClkCntr <= #1 SerialClkCntrNxt;
        end

    always @ (posedge tstSerialLaneClk)
    begin
        if (SerialClkCntr >= 15'd20000)
            begin
                // Delete req Assert: FIFO Full possible
                tstSerialLaneRdClk <= #1 1'b0; // slow clk
                tstSerialLaneWrClk <= #1 ~tstSerialLaneWrClk; // fast clk
            //      // Insert req Assert: FIFO Empty possible
            //      tstSerialLaneWrClk <= #1 1'b0; // slow clk
            //      tstSerialLaneRdClk <= #1 ~tstSerialLaneRdClk; // fast clk
            end
        else
            begin
                tstSerialLaneWrClk <= #1 !tstSerialLaneWrClk;
                tstSerialLaneRdClk <= #1 !tstSerialLaneRdClk;
            end
        end
    end

// -----
// Clock Signal Generation: Lane Parallel input/output clock
// tstSerialLaneWrClk /66 = tstLaneWrClk; tstLaneWrClk * 4 =
tstTxClock
// clock: 156.25 MHz; period 6336 ps (expected 6400 ps)
//      when crossing boundaries having a diff of 96 ps 6330ps :
6432ps
// -----
initial
    begin
        tstLaneWrClk = 1'b0;
        CkEdgeCounterTx = 7'b0000000;
        tstLaneRdClk = 1'b0;
        CkEdgeCounterRx = 7'b0000000;
    end
// Tx parallel clock generation
always @ (posedge tstSerialLaneWrClk)
    begin
        tstLaneWrClk <= #5 nxtLaneWrClk;
    end

always @ (posedge tstSerialLaneWrClk)
    begin
        CkEdgeCounterTx <= #1 CkEdgeCounterTxNxt;
    end
always @(*) // Combinational logic
    begin
        if (CkEdgeCounterTx >= 7'd32)
            CkEdgeCounterTxNxt = 7'b0000000;
        else
            CkEdgeCounterTxNxt = #1 CkEdgeCounterTx + 7'b0000001;
        end
    end

```



```

always @(*)
begin
    if (CkEdgeCounterTx >= 7'd32)
        nxtLaneWrClk = !tstLaneWrClk;
    else
        nxtLaneWrClk = tstLaneWrClk;
    end

// Rx parallel clock generation
always @ (posedge tstSerialLaneRdClk)
begin
    tstLaneRdClk <= #5 nxtLaneRdClk;
end

always @ (posedge tstSerialLaneRdClk)
begin
    CkEdgeCounterRx<= #1 CkEdgeCounterRxNxt;
end

always @(*) // Combinational logic
begin
    if (CkEdgeCounterRx >= 7'd32)
        CkEdgeCounterRxNxt = 7'b0000000;
    else
        CkEdgeCounterRxNxt = #1 CkEdgeCounterRx + 7'b0000001;
    end

always @(*)
begin
    if (CkEdgeCounterRx >= 7'd32)
        nxtLaneRdClk = !tstLaneRdClk;
    else
        nxtLaneRdClk = tstLaneRdClk;
    end

// -----
// Clock signal generation
// RX_CLK & TX_CLK 625 MHz: period = 1591ps (1600 expected)
// Sometimes this goes for 1584: 1679 diff of 95 ps
// -----
initial
begin
    tstTxClock = 1'b0;
    tstRxClock = 1'b0;
end

// Clock Generator Tx
always @(posedge tstLaneWrClk)
begin
    repeat (8)
    begin
        tstTxClock = # ((SerialLaneWrClkPeriodbyTwo * 2 * 66) / 8)
~tstTxClock;
    end
end

// Clock Generator Rx

```



```

always @(posedge tstLaneRdClk)
begin
repeat (8)
begin
tstRxClock = # ((SerialLaneRdClkPeriodbyTwo * 2 * 66) / 8)
~tstRxClock;
end
end

// -----
// Initialization
// -----
// Reset Signal : Reset pulse width = (2 * 1584) + 5
initial
begin
tstPCSReset = 1'b0; // Active low reset pulse
@ (posedge tstLaneWrClk);
@ (posedge tstLaneWrClk);
#6
tstPCSReset = 1'b1;
end
// Enable signal: Enable is activated at(3 * 1584) + 25
initial
begin
tstPCSEn = 1'b0; // not eanbled
@ (posedge tstLaneWrClk);
@ (posedge tstLaneWrClk);
@ (posedge tstLaneWrClk);
#26
tstPCSEn = 1'b1;
end

// -----
// Test Data Generation
// -----
always @ (posedge tstTxClock)
begin
if((tstPCSReset == 1'b0) || (!tstPCSEn) || (!tstRxRdy))
begin
ClkCounter = 13'd0;
end
else
begin
if(ClkCounter >= `WrClkCycles)
ClkCounter = 13'd1;
else
ClkCounter = ClkCounter + 13'd1;
end
end

// -----
// Test Data Packet considerarions
// Minimum IPG = 96 bits = 12 bytes
// Maximum clock rate diff +/- 100 ppm => 1 clk diff for 5000 clks
// Data Pkt :-
// Clk1: S0 D1 D2 D3 D4 D5 D6 D7 - Data 7
// Clk5000: D0 D1 D2 T I0 I1 I2 I3 - Data 3 : 5000 clks

```



```

//          Clk5001: I0 I1 I2 I3 I4 I5 I6 I7
//          Size: 7 + 3 + (8 * 4998) = 39994 Bytes
// -----
always @ (posedge tstTxClock or negedge tstPCSReset)
begin
    if ((tstPCSReset == 1'b0)|| (!tstPCSEn)) // Active Low Reset
    assumed
        begin
            TestData = {64{1'b0}};
            tstTxData =
64'b00000111_00000111_00000111_00000111_00000111_00000111_00000111_0
0000111; // idle
            tstTxCtrl = 8'b11111111;
        end
    else
        begin
            TestData = $random;
            if (!tstRxRdy)
                begin
                    tstTxData =
64'b00000111_00000111_00000111_00000111_00000111_00000111_00000111_0
0000111; // idle
                    tstTxCtrl = 8'b11111111;
                end
            else if(ClkCounter == 1)
                begin
                    tstTxData = {TestData[55:0],8'hFB}; // Start: SDDD DDDD
                    tstTxCtrl = 8'b00000001;
                end
            else if(ClkCounter == (`WrClkCycles - 1)) // T3: DDDT IIII
                begin
                    tstTxData = {8'h07, 8'h07, 8'h07, 8'h07, 8'hFD,
TestData[23:0]};
                    tstTxCtrl = 8'b11111000;
                end
            else if(ClkCounter == `WrClkCycles) // I
                begin
                    tstTxData =
64'b00000111_00000111_00000111_00000111_00000111_00000111_00000111_0
0000111; // idle 8 octects
                    tstTxCtrl = 8'b11111111;
                end
            else
                begin
                    tstTxData = TestData;
                    tstTxCtrl = 8'h00; // Data
                end
        end
    end
end

// -----
initial
    #600_000_000 $stop;
// -----
endmodule

```



University of Moratuwa, Sri Lanka.
 Electronic Theses & Dissertations
www.lib.mru.ac.lk