DESIGN, IMPLEMENTATION AND PERFORMANCE STUDY OF GIGABIT ETHERNET PROTOCOL USING EMBEDDED TECHNOLOGIES

A Thesis submitted to Goa University for the Award of the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRONICS

By

Vinaya Rajendra Gad

Research Guide

Prof. G. M. Naik

Goa University

Taleigao Goa

2014

Certificate

This is to certify that the thesis entitled "**Design, Implementation And Performance Study Of Gigabit Ethernet Protocol Using Embedded Technologies**", submitted by Mrs. Vinaya Rajendra Gad, for the award of the degree of Doctor of Philosophy in Electronics, is based on her original and independent work carried out by her during the period of study, under my supervision. The thesis or any part thereof has not been previously submitted for any other degree or diploma in any University or institute.

Place : Goa University
Date: 15th April 2014

(G. M. Naik)
Research Guide

Statement

I state that the present thesis entitled "**Design, Implementation And Performance Study Of Gigabit Ethernet Protocol Using Embedded Technologies**" is my original contribution and the same has not been submitted on any occasion for any other degree or diploma of this University or any other University / Institute. To the best of my knowledge, the present study is the first comprehensive work of its kind in the area mentioned. The literature related to the problem investigated has been cited. Due acknowledgements have been made wherever facilities and suggestions have been availed of.

Place: Goa University **(V. R. Gad)**
Date: 15th April 2014 **Candidate**

Dedicated with love to my parents

Shri. Anand Puttu Gaonkar and Smt. Medha Anand Gaonkar

Acknowledgements

"No great Discovery was ever made without a bold guess."

Isaac Newton

I express my sincere appreciation to my supervisor, Dr. Gourish M. Naik, Professor, Department of Electronics , Goa University, for his endless support, encouragement, his great inspiration, ideas, comments and an endless stream of stories and articles during our interaction.

Thanks are due to Dr. Jyoti Powar, Associate Professor, Department of Computer Science and Technology, for advising me during my Faculty research committee meetings on the various goals to be achieved during course of research works and critical comments on the data analysis I presented.

I acknowledge the advice and the assistance received from Prof. J.A.E. Desa, Former Head, Department of Physics & Dean, Faculty of Natural Science; Prof. J. B. Fernandes , Dean, Faculty of Natural Science during the various phases of the thesis. My special thanks go to Mr. P. M. Bhende, Principal at GVM's College Ponda Goa for encouragements and support throughout the duration of my research.

Thanks to Dr. Rajendra S. Gad, Associate Professor, Dr. Jivan S. Parab, Assistant Professor, Department of Electronics, Mr. Mahesh Salgaonkar, Researcher at National Institute of Oceanography, Miss Ingrid Nazareth and Mr. Narayan Vetrekar, Research Students, for helping me in compiling various examples in embedded system design. Also thanks goes to Mr. Ranjeet L. Rane for advising me on SFP modules for the SGMII interface of Ethernet. Also thanks are due for Dr. S. V. Gopalaiah, ECE IISc, and Dr. Kuruvilla Varghese, DESE, IISc; for advising me

on WireShark and critical comments on the performance analysis platforms establishments. Thanks are due for Mr. R. Narashimhan, India Channel Manager at Altera Semiconductor India, for his expert advise on the SOPC software for compiling the Ethernet IP core on FPGA.

I acknowledge Dr. Vithal Tilvi (Texas A & M University, USA), Mr. Saish Amonkar, Mr. Udaysing V. Rane, Mrs. Meera Mayekar , Miss Raina Pinto, Mrs. Sulaxana Vernekar , Smt. Deepa G. Naik , Mr. Noel Tavares and Mr. Ishara Fernandes for keeping me in good state of mind and high morale during the compilation of the thesis work.

Special thanks to Mr. Jayprakash Kamat, Mr. Agnelo Lopez , for machining the fiber optics coupler with three axis displacements. My thanks to Mr. William D'souza, Mr. Ghanasyam Kerkar and Mrs. Veronica Fernandez.

I am indebted to funding agencies like University Grants Commission (UGC) for providing me Faculty Improvement Program (FIP) for three years and ALTERA InC. USA for supporting the University Program and establishing SoC laboratory at the Department of Electronics Goa University and providing the IP cores for studies.

I express my deep sense of gratitude to my in-laws and all my family members for their patience and understanding as well as taking much of the responsibilities at home. I shall forever remain indebted to my daughter Kaivalya and son Aditya. The long hours that I owed to them, I had spent for thesis.

Vinaya R. Gad, April 2014.

TABLE OF CONTENTS PAGE NO.

I PREFACE XIX

1. Introduction to Ethernet Technologies

1.1 Introduction 2

1.2 The OSI Reference Model 3

1.3 TCP/IP Protocol suite 7

1.4 IEEE Project 802 7

1.5 History of Ethernet 12

1.6 Gigabit Ethernet 16

1.7 Emerging trends in Ethernet technologies 19

1.8 Literature Review 25

1.8.1 Review of Performance Analysis of Gigabit Ethernet 25

1.8.2 Review of CRC Error Detection Algorithm Development 28

1.8.3 Review of LDPC Error Correction Codes 31

1.9 Organisation of the thesis 37

2 Gigabit Ethernet Protocol 39

2.1 Objectives of Research 40

2.2 Gigabit Ethernet Protocol Architecture 41

2.2.1 Media Acess Control Layer 42

2.2.2 The Physical Layer 43

2.3 Ethernet Media Access layer Implementation 46

2.4 Methodology of Implementation 47

2.5 Evaluation Techniques 49

3 Hardware Implementation using FPGA Technologies and IP Cores 50

3.1 Platform and Resources 51

3.2 System Organization 54

3.3 FPGA Implementation 61

3.3.1 Programmable 10/100/1000 Mbps Ethernet operation 61

3.3.2 Transmit Operation 63

3.3.3 Receive Operation 63

3.3.4 Collision Detection 65

3.3.5 SGMII Converter 65

3.3.6 Interface on the Stratix II GX development board 67

3.3.7 Compilation Report 69

3.4 Command Line Interface 75

3.4.1 Test Menu 75

3.4.2 TSE MAC Menu 76

3.4.3 Report Menu 76

3.4.4 Console Menu 77

3.5 Experimental Setup 79

4 Modeling and Performance Analysis 83

4.1 Errors in a Digital Communication System 84

4.1.1 Shannon Capacity Formula 84

4.1.2 Bit Error Rate 86

4.1.3 Packet Error Rate 86

4.1.4 Binary Symmetric Channel 87

4.1.5 Additive White Gaussian Noise Channel 87

4.1.6 Bit Error rate of Binary Phase Shift Keying 88

4.2 Error Detection using CRC 88

4.2.1 Software Solution 90

4.2.2 Traditional Hardware Solution 91

4.2.3 Parallel Solution 91

4.2.4 Configurable Hardware 92

4.3 Simulation Model for Error Detection using CRC 92

4.4 Error Correction using LDPC codes 93

4.4.1 Representation of LDPC Codes 94

4.4.2 LDPC decoding algorithms 96

4.5 Simulation Model for Error Correction using LDPC code 101

5 Discussions and Conclusions 103

5.1 Performance Analysis of Gigabit Ethernet protocol design 104

5.2 Simulation Results for Error Detection 108

5.3 Results of Performance Studies of Error Correction using LDPC codes 110

5.4 Conclusion 119

5.5 Scope for future work 120

ANNEXURE

APPENDIX I 121

APPENDIX II 124

REFERENCES 132

LIST OF TABLES

Table 1.5.1 Evolution of Ethernet Standards from 10 Mbps to 400Gbps

Table 1.7.1 gives a detailed specification of the Gigabit Ethernet technologies from 1Gbps to 100Gbps

Table 3.2.1 Memory Map of the TSE design

Table 3.3.1 Clock and Reset Signals

Table 3.3.2 Triple-Speed Ethernet 0 Signals

Table 3.3.3 Triple-Speed Ethernet 1 Signals

Table 3.3.4 SFP Interface Signals

Table 3.3.5 Utilization of Resources of the TSE design

Table 3.4.1 List of Console commands and their purpose

Table 5.1.1 List of SFP Transceivers used for the different Gigabit Ethernet standards

Table 5.1.2 Line Rate and Throughput of different frame lengths

Table 5.1.3. Throughput of the network obtained by Duan and Han

Table 5.2.1 List of CRC codes and their generator polynomials

LIST OF FIGURES

Figure 1.2.1 Structure and functions of the OSI architecture

Figure 1.3.1 Comparison of TCP/IP and OSI models

Figure 1.4.1 The IEEE-802 LAN Model

Figure 1.5.1 World's first LAN

Figure 1.5.2 Timeline of Ethernet evolution

Figure 1.6.1 IEEE 802.3 Ethernet Frame Format

Figure 1.6.2 Enterprise LAN Topology

Figure 1.6.3 Gigabit Ethernet protocol architecture

Figure 1.7.1 Use of 10Gbps Ethernet has grown steadily among internet exchange providers

Figure 1.7.2 Dominating Gigabit Ethernet links

Figure 1.7.3 Impact of the 100Gbps Ethernet

Figure 2.2.1 Gigabit Ethernet layer diagram

Figure 2.2.2 Gigabit Ethernet Frame Structure with Carrier Extension

Figure 2.2.3 1000Base-X encapsulation

Figure 2.2.4 Gigabit Ethernet application environments and link distances

Figure 2.2.5 Gigabit Ethernet Protocol Stack

Figure 2.3.1 Proposed MAC Implementation Block Diagram

Figure 2.4.1 Various stages associated with QUARTUS-II IDE

Figure 3.1.1 Stratix II GX PCI Express Development Board Block Diagram

Figure 3.1.2 Photograph of Top View of the Stratix II GX PCIeDevelopment Board

Figure 3.2.1 Block Diagram of Triple Speed Ethernet Design

Figure 3.2.2 Parameter settings for Nios II processor

Figure 3.2.3 Parameter settings of Triple Speed Ethernet Megacore function

Figure 3.2.4 Components used in SOPC builder setup for the TSE design

Figure 3.3.1 Programmable 10/100/1000Mbps Ethernet operation

Figure 3.3.2 Block diagram of the PCS function with an embedded PMA

Figure 3.3.3 Marvell 88E1111 Gigabit Ethernet PHY Layer and GMII Interface to the FPGA

Figure 3.3.4 SFP Module Interface

Figure 3.3.5 Block Symbol file of TSE design

Figure 3.3.6 Pin Assignment for TSE design

Figure 3.3.7 Compilation Report Summary

Figure 3.5.1 Testbed setup for Performance Analysis

Figure 3.5.2 Setup for introducing errors in the Ethernet frames and Error Detection

Figure 3.5.3 Number of packet errors vs lateral displacement for 1 million packets-64,128 bytes

Figure 3.5.4 Number of packet errors vs lateral displacement for 1 million packets -1518 bytes

Figure 4.1.1 Graphical representation of BSC

Figure 4.1.2 AWGN channel model

Figure 4.3.1 Matlab Simulation model for BER performance for BSC channel

Figure 4.4.1 Tanner graph corresponding to the parity check matrix in equation

Figure 4.5.1 Matlab Simulation model for Error Correction Analysis

Figure 4.5.2 Matlab Simulation model for Error Correction interfaced with Gigabit Ethernet protocol design

Figure 5.1.1 Top view photograph of the different SFP modules

Figure 5.1.2 Line rate vs Frame length

Figure 5.1.3 Throughput vs Frame length

Figure 5.1.4 Line Rate vs Frame length

Figure 5.1.5 Throughput vs Frame length

Figure 5.1.6 Transmission time vs Frame length a) doubling data bytes b) multiplying N data bytes with a factor of 10

Figure 5.2.1 BER Performance for BSC

Figure 5.2.2 PER Performance for BSC

Figure 5.2.3 BER vs Signal to Noise Ratio (Eb/No) in dB

Figure 5.3.1 a) to e) Showing BER vs SNR for iteration= 5 for various number of frames

Figure 5.3.2 a) to e) Showing BER vs SNR for iteration= 10 for various number of frames

Figure 5.3.3 a) to d) Showing BER vs SNR for iteration= 5 and 10 for different block lengths and SPA log domain & SPA-Min Sum algorithms

Figure 5.3.4 a) to c) BER vs SNR for iteration= 220 to 300 and block length 256 bits

Figure 5.3.5 a) to c) Showing BER vs SNR for iteration= 20 to 300

Figure 5.3.6 a) to d) BER vs SNR for different block lengths

Figure 5.3.7 a) Comparison of LDPC codes n=10^6, R=1/2; b) Comparison of LDPC with Turbo codes for N=10^3,10^4,10^5,10^6 ;c) Mackay's Results

Figure 5.3.8 a) Showing Error correction performance of LDPC codes for block length =512 bits, SPA-logdomain algorithm

b) Showing Error correction performance of LDPC codes for block length =512 bits and SPA-Min-Sum algorithm – logdomainSimple.

c) Showing Error correction performance of LDPC codes for block length =1024 bits, SPA-logdomain algorithm

d) Showing Error correction performance of LDPC codes for block length =1024 bits and SPA-Min-Sum algorithm – logdomainSimple.

e) Showing Error correction performance of LDPC codes for block length =2048 bits, SPA-logdomain algorithm

f) Showing Error correction performance of LDPC codes for block length =2048 bits and SPA-Min-Sum algorithm – logdomainSimple.

KEYWORDS

ATM	Asynchronous Transmission Mode
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BEC	Binary Erasure Channel
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CFP	C Form Factor Pluggable
CLB	Configurable Logic Block
CRC	Cyclic Redundancy Code
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
ECC	Error Correction Code
FDDI	Fiber Distributed Data Interface
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
GF	Galoisfield Field
GMII	Gigabit Media Independent Interface
HIPPI	High Performance Parallel Interface
HPC	High Performance Computing
ICMP	Internet Control Message Protocol
IP	Intellectual Property
IXP	Internet Exchange Point
ISO	International Organization For Standardization
JTAG	Joint Test Action Group
LAN	Local Area Network
LDPC	Low Density Parity Check
LFSR	Linear Feedback Shift Register

LHGBIC	Long Haul Gigabit Interface Convertor
LLC	Logical Link Control
LUT	Look Up Table
LC	Lucent Connectors
MAC	Media Access Control
MAN	Metropolitan Area Network
MPA	Message Passing Algorithm
NIC	Network Interface Card
OSI	Open Systems Interconnection
PAN	Personnel Area Network
PCS	Physical Coding Sublayer
PLS	Physical Signaling
PMA	Physical Medium Attachment
PMD	Physical Medium Dependent
PHY	Physical
PIO	Parallel Input Output
RAN	Rural Area Network
RGMII	Reduced GMII
RS	Reconciliation Sublayer
SC	Subscriber Connector
SDRAM	Synchronous Dynamic Random Access Memory
SERDES	Serializer Deseralizer
SFP	Small Form Factor Pluggable
SNAP	Subnetwork Access Protocol
SPA	Sum Product Algorithm
SRAM	Static Random Access Memory
SONET	Synchronous Optical Network
SOPC	System on Programmable Chip
SGMII	Serial GMII

TCP/IP	Transmission Control Protocols/ Internet Protocol
TAG	Technical Advisory Group
TWSI	Two Wire Serial Interface
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless LAN
WMAN	Wireless MAN

PREFACE

This thesis focuses on the design and implementation of Gigabit Ethernet protocol tested on Altera FPGA platform to generate Gigabit Ethernet frames and study its performance. The Simulation model for Error Detection using Cyclic Redundancy Code (CRC) and Error Correction using LDPC codes in a noisy channel is developed in MATLAB 7.0, with emphasis on Gigabit Ethernet protocol. To the best of our knowledge, no such elaborate studies have been found related to the Gigabit Ethernet protocol in the past literature.

The thesis describes the designing and implementation of Gigabit Ethernet protocol testbed using FPGA platform and it's Performance Analysis. The system is designed on Altera's Stratix II GX PCI Express Development Kit using Altera's TSE (Triple Speed Ethernet) Megacore function. The throughput of Gigabit Ethernet protocol is found to be approaching 1Gbps for 9600 frame length, which is tested for different physical media. The thesis also addresses the complexities involved in the implementation of the said design.

Also a Simulation model in Matlab is developed for Error Detection using CRC code and Error Correction Analysis using LDPC codes. The Gigabit Ethernet testbed using Altera FPGA is interfaced with Matlab Simulation model, which is used to study the BER performance in a noisy Gigabit Ethernet channel. A testbed is developed for introducing optical attenuation for optic fibre channel. The results for CRC32 Error Detection algorithm for different Gigabit Ethernet frame lengths is presented. Also the results for BER performance using LDPC codes is presented for Gigabit Ethernet frame lengths of 64 , 128 and 256 bytes respectively.

Vinaya Rajendra Gad

April, 2014

LIST OF RESEARCH RELATED COMMUNICATED PAPERS

INTERNATIONAL /NATIONAL JOURNAL

1. **V. R. Gad**, R. S. Gad, G. M. Naik, "Performance Analysis of Gigabit Ethernet Standard for various physical media using Triple Speed Ethernet IP Core on FPGA", published in the book "Advances in Computer Science, Engineering and Applications" ,Eds. David C. Wyld et. al. by Springer Verlag, 2012.
2. **V. R. Gad**, R. S. Gad, G. M. Naik, "Implementation of Gigabit Ethernet Standard using FPGA" published in International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol.2, No.4, August 2012 DOI : 10.5121/ijmnct.2012.2404 31.
3. Gad.R.S, Parab.J.S, **V.R.Gad**, Naik.G.M, "Embedded Platform Development: A SoC TCP/IP Control Platform for Ethernet-Enabled Devices", Circuit Cellar , Vol. 255 December 2011.
4. **V. R. Gad**, R. S. Gad, G. M. Naik , "Gigabit Ethernet Implementation of CRC-32 in noisy channels" , International Journal of VLSI Design , Vol.1, 2011.

Communicated

1. **Vinaya R. Gad**, Rajendra S. Gad, Gourish M. Naik , Testbed For Performance Evaluation of Low Density Parity Check (LDPC) Codes Over Gigabit Ethernet Protocol For Optical Fibre Physical Media , Computer Networks, Elsevier.
2. **Vinaya R. Gad**, Rajendra S. Gad, Gourish M. Naik Configurable CRC Error Detection Model For Performance Analysis of Polynomial: Case Study for the 32-bits Ethernet Protocol, Computer Networks, Elsevier.

INTERNATIONAL / NATIONAL CONFERENCES

1. A Saldanha, I. A. Nazareth, R. Veluskar, **V. R. Gad**, R. S. Gad, G. M. Naik, "Study of WiMAX and Simulation of Viterbi Decoder",WIRCON 2009, IEEE & L&T Institute of Technology Mumbai,October 2009.
2. **V. R. Gad**, R. S. Gad, G. M. Naik,"Implementation of Gigabit Ethernet using Double CRC-32 technique", National symposium on "VLSI & Embedded System" , Goa University & VSI ,Goa Chapter,Feb 2010.
3. **V. R. Gad**, R. S. Gad, G. M. Naik , "BER Performance for noisy channels in Gigabit Ethernet" , National symposium on "VLSI & Embedded System" , Goa University & VSI-Goa Chapter , March 2012.

CHAPTER 1

INTRODUCTION TO ETHERNET TECHNOLOGIES

1.1 Introduction

There has been a rapid development in the global information infrastructure and also a growing increase in the bandwidth requirements. Gigabit Ethernet is used not only for interconnection of computers, but also in high-speed network equipment as well as user access to metropolitan area network. It is also a popular and robust networking protocol for computer networks, backbone and industrial applications [1]. Ethernet has been successful for over 30 years because the Ethernet standards have progressed along with networking and bandwidth requirements. This progression of standards provides an obvious and straightforward migration path for companies as their bandwidth requirements increase[2]. Gigabit Ethernet has evolved from the original 10Mbps Ethernet standard, 10BASE-T, and the 100Mbps Fast Ethernet standards, 100BASE-TX and 100BASE-FX. The IEEE adopted Gigabit Ethernet over fiber optic cabling, IEEE 802.3z in June 1998 and its implementation was extensively supported by networking vendors. This standard helped the companies to improve traffic flow in congested areas. The IEEE standardized IEEE 802.3ab Gigabit Ethernet over copper as 1000BASE-T in June 1999, which enabled Gigabit speeds to be transmitted over Cat-5 cable.

Ethernet can be used with fiber optic cable in 550M and 5Km lengths and Cat-5 copper cable up to 100 meters. Gigabit connections can also be established for lengths of up to 70Km through use of long-haul Gigabit Interface Connectors (LH GBICs) in switches. Fiber is normally used between buildings or vertical connections between floors, where distances more than 100- meters are required. Copper cable is susceptible to electromagnetic interference, which can corrupt data whereas fiber provides immunity from environmental noise and security from unauthorized access to information. Multiple applications are simultaneously used on the desktops in today's enterprise. Certain applications such as online backup and recoveries of data and applications, large file transfers from data center servers and storage plants have an effect on the traffic related with all other applications that are using network resources at the same time.Thus deployment of Gigabit Ethernet to the desktop has improved the productivity

and performance of the enterprise. Similarly, data and information used in the decision-making process is extracted, delivered, and analyzed at significantly faster rates. However, as these applications become more advanced, they also become more bandwidth-intensive.

As more traffic is generated, network performance decreases, and so does employee productivity. Today, users may have one application in the foreground and several applications active in the background. Each user has his own computing requirement which is a combination of different applications requiring computer processor power and network bandwidth. The aggregate of the application traffic patterns should be applied to the available bandwidth,which reflects the true bandwidth and network use of the users and their applications. Large file transfers, multicast or on-demand video, e-mail with large attachments, on-demand backups and recovery, enterprise resource planning, customer relationship management and Web and Java-based tools, are among the list of applications that users run every day, without realizing much about it. Each user has unique traffic patterns and bandwidth requirements. Now, add the traffic of all the applications and multiply that by the number of users in the workgroup, the department, the floor, and the wiring closet. Thus there is an ever increasing requirement for processing speed as well as bandwidth and the demand for Gigabit Ethernet has been intensifying.

1.2 The OSI Reference Model

Network models are structured into layers, with each layer representing a specific networking function. These functions are controlled by protocols, which are rules that govern end-to-end communication between devices. The Open Systems Interconnection (OSI) model was developed by the International Organization for Standardization (ISO), and formalized in 1984. It provided the first framework governing how information should be sent across a network. It gives a platform for a common technical language and has led to the standardization of communications protocols and the functions of a protocol layer. The structure and functions of the OSI architecture is given in the Figure 1.2.1 [3].

User Application Programs

RESPONSIBILITY OF HOST SYSTEM	Layer 7	Application	Provides general services related to application (e.g. file transfer, user access)	SUPPORT USER APPLICATIONS
	Layer 6	Presentation	Formats data (Encodes, encrypts, compresses)	
	Layer 5	Session	Maintains dialog between communication devices	
	Layer 4	Transport	Provides reliable end-to-end data transmission	
RESPONSIBILITY OF NETWORK	Layer 3	Network	Switches and routes information units	GOVERNS THE COMMUNICATION FACILITIES
	Layer 2	Data Link	Provides data exchange between devices on the medium	
	Layer 1	Physical	Transmits bit stream to physical medium	

Figure 1.2.1 Structure and functions of the OSI architecture

The services provided by each Protocol Layer is summarized below: [4,5].

Physical Layer

The physical layer is the lowest layer of the OSI model. It deals with the transmission and reception of the raw bit stream over a physical medium. It describes the mechanical and electrical or optical specifications and functional interfaces to the transmission medium, and carries the signals for all of the higher layers. It provides data encoding, physical medium attachment, transmission technique, physical medium transmission, data rate, synchronization of bits, line configuration, physical topology and transmission mode.

Data Link Layer

The data link layer provides error-free transmission of data frames from one node to another over the physical layer It allows the above layers to assume virtually error-free transmission over the link. It provides link establishment and termination, Physical addressing, Frame traffic control, Frame sequencing, Frame acknowledgment, Detects and recovers from errors that occur in the physical layer, Frame delimiting, Frame error checking and Media access management. The data link layer is partitioned into the Logical Link Control (LLC) and the media access control(MAC) sublayers. The LLC is common to all LANs and handles functions such as connection setup, initialization, data formatting, address recognition, error control, flow control and connection termination. Thus this sublayer supervises the transmission of a packet between nodes. The MAC layer handles access to the shared medium and is specific to the type of LAN that is implemented.

Network Layer

The network layer deals with the delivery of packets from source-to-destination, possibly across multiple links. It provides routing, Subnet traffic control, Frame fragmentation, Logical-physical address mapping, Subnet usage accounting.

Transport Layer

The transport layer ensures that the data frames delivered are error-free, in sequence, and there are no losses or duplications. The higher layer protocols need not be concerned with the transfer of data between them and their peers. The type of service that is obtained from the network layer determines the size and complexity of a transport protocol. A minimal transport layer is required for a reliable network layer with virtual circuit capability. The transport protocol should include extensive error detection and recovery for unreliable network layers. The transport layer provides message segmentation, message acknowledgment, message traffic control and session multiplexing.

Session Layer

The session layer is responsible for session establishment between different communicating stations. It provides, session establishment, maintenance and termination and session support.

Presentation Layer

The presentation layer deals with the syntax and semantics of the information exchanged. It formats the data to be presented to the application layer and can be viewed as the translator for the network. The presentation layer provides, character code translation, data conversion, data compression and data encryption.

Application Layer

The application layer enables the users and application processes to access the network. It provides functions such as resource sharing, remote file access, remote printer access, inter-process communication, network management, directory services, electronic messaging and network virtual terminals.

1.3 TCP/IP Protocol suite

Transmission Control Protocol/Internet Protocol (TCP/IP) is a hierarchical protocol suite, [5] made up of interactive modules. Each module provides a specific functionality and the modules may not be interdependent. The layers of the TCP/IP Protocol suite contain relatively independent protocols that can be mixed and matched depending on the needs of the system. Figure 1.3.1 gives a comparison of TCP/IP and OSI models.

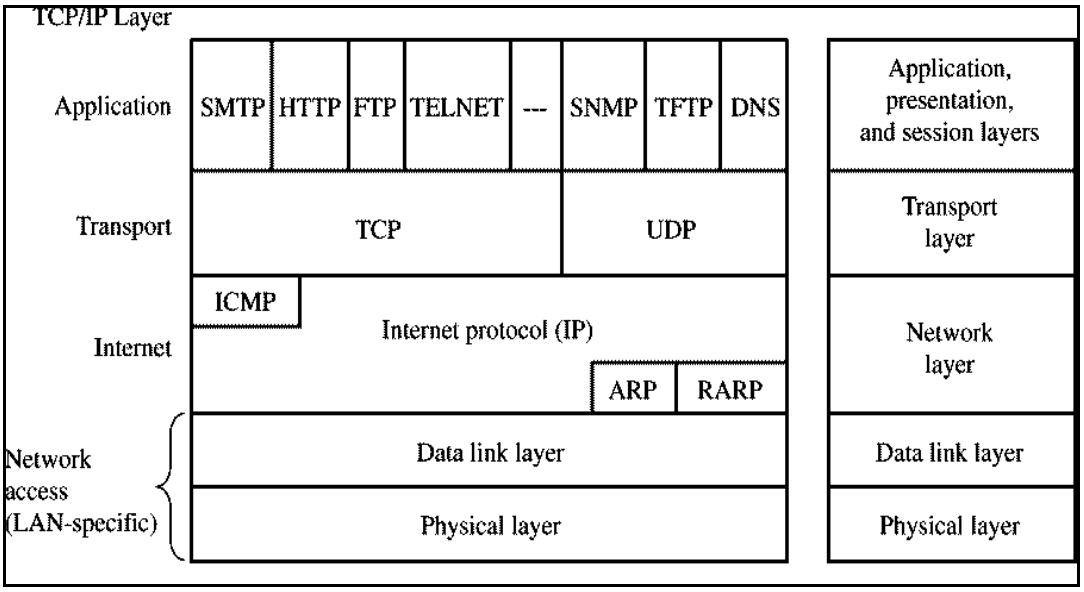


Figure 1.3.1 Comparison of TCP/IP and OSI models [3]

1.4 IEEE Project 802

The Working Group 802 of the IEEE has been a major creator of specifications for Local Area Network (LAN) architectures. The IEEE-802 LAN standards are modularly structured and consist of a powerful set of protocols. The LAN management functions are subdivided into those that can be generalized and those that must remain specific to a particular LAN architecture.

Figure 1.4.1 illustrates the IEEE-802 LAN Model and compares it with OSI model [3].

.

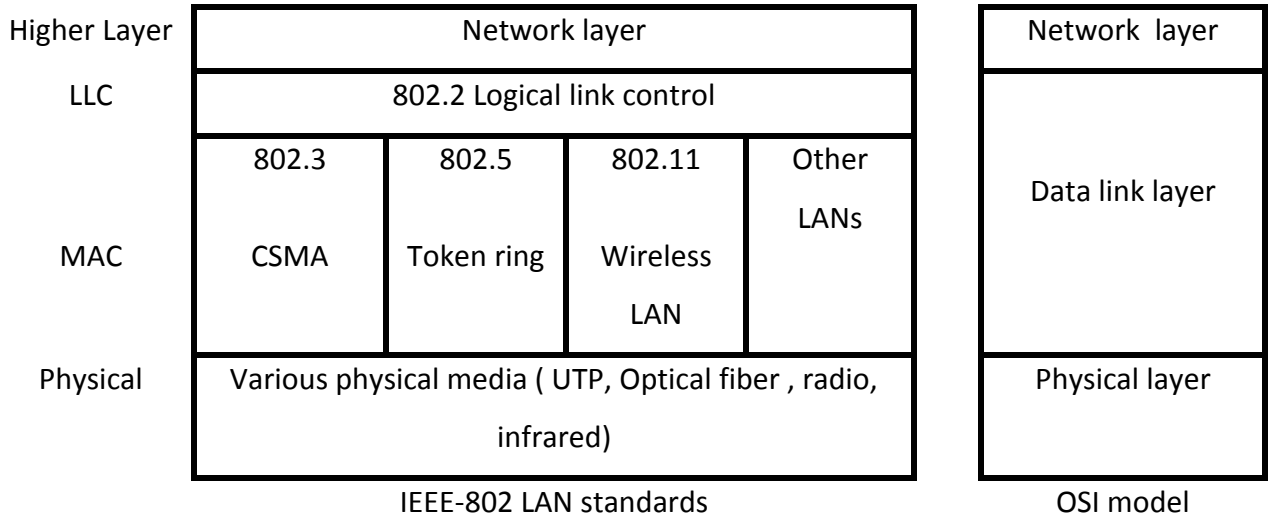


Figure 1.4.1 The IEEE-802 LAN Model

The IEEE 802 LAN/MAN (Metropolitan Area Network) Standards Committee develops and maintains networking standards and recommended practices for local, metropolitan, and other area networks, using an open and accredited process, and advocates them on a global basis. The most extensively used standards are for Ethernet, Bridging and Virtual Bridged LANs, Wireless LAN, Wireless Personal Area Network (PAN), Wireless MAN, Wireless Coexistence, Media Independent Handover Services, and Wireless Rural Area Network (RAN) [6].

Following is the list of various IEEE-802 subgroups and technical advisory groups (TAG):

IEEE 802 Working Groups and Study Groups

- 802.1 Higher Layer LAN Protocols Working Group
- 802.3 Ethernet Working Group
- 802.11 Wireless LAN Working Group
- 802.15 Wireless Personal Area Network (WPAN) Working Group

- 802.16 Broadband Wireless Access Working Group
- 802.18 Radio Regulatory TAG
- 802.19 Wireless Coexistence Working Group
- 802.21 Media Independent Handover Services Working Group
- 802.22 Wireless Regional Area Networks
- 802.24 Smart Grid TAG
- OmniRAN EC Study Group

Hibernating Working Groups

- 802.17 Resilient Packet Ring Working Group
- 802.20 Mobile Broadband Wireless Access (MBWA) Working Group

Disbanded Working Groups and Study Groups

- 802.2 Logical Link Control Working Group
- 802.4 Token Bus Working Group (material no longer available on this web site)
- 802.5 Token Ring Working Group
- 802.6 Metropolitan Area Network Working Group (material no longer available on this web site)
- 802.7 Broadband TAG (material no longer available on this web site)
- 802.8 Fiber Optic TAG (material no longer available on this web site)
- 802.9 Integrated Services LAN Working Group (material no longer available on this web site)
- 802.10 Security Working Group (material no longer available on this web site)
- 802.12 Demand Priority Working Group (material no longer available on this web site)
- 802.14 Cable Modem Working Group (material no longer available on this web site)
- 802.23 Emergency Services Working Group
- QOS/FC Executive Committee Study Group (material no longer available on this web site)

- ECSG TVWS TV Whitespace study group
- ES-ECSG Emergency Services Executive Committee Study Group

IEEE 802.3 Ethernet Working Group

List of active projects, study groups, and ad hocs as listed below:

- IEEE P802.3bj 100 Gb/s Backplane and Copper Cable Task Force.
- IEEE P802.3bm 40 Gb/s and 100 Gb/s Fibre Optic Task Force.
- IEEE P802.3bn EPON Protocol over Coax (EPoC) Task Force.
- IEEE P802.3bp Reduced Twisted Pair Gigabit Ethernet PHY Task Force.
- IEEE P802.3bq 40GBASE-T Task Force.
- IEEE P802.3br Interspersing Express Traffic Task Force.
- IEEE P802.3bt DTE Power via MDI over 4-Pair Task Force.
- IEEE P802.3bu 1-Pair Power over Data Lines (PoDL) Task Force.
- IEEE 802.3 400 Gb/s Ethernet Study Group.
- IEEE 802.3 Industry Connections NG-EPON Ad Hoc.

Archive information of completed work:

- IEEE Std 1802.3-2001Conformance test reaffirmation.
- IEEE 802.3 Trunking Study Group.
- IEEE 802.3 Higher Speed Study Group.
- IEEE 802.3 DTE Power via MDI Study Group.
- IEEE 802.3 10GBASE-CX4 Study Group.
- IEEE 802.3 10GBASE-T Study Group.
- IEEE 802.3 Backplane Ethernet Study Group.
- IEEE 802.3 10 Gb/s on FDDI-grade MM fiber Study Group.
- IEEE 802.3 Power over Ethernet Plus Study Group.
- IEEE 802.3 Residential Ethernet Study Group.

- IEEE 802.3 Energy Efficient Ethernet Study Group.
- IEEE 802.3 Higher Speed Study Group.
- IEEE 802.3 100 Gb/s Backplane and Copper Study Group.
- IEEE 802.3 Extended EPON Study Group.
- IEEE 802.3 Next Generation 40 Gb/s and 100 Gb/s Optical Ethernet Study Group.
- IEEE 802.3 EPON Protocol over a Coax (EPoC) PHY Study Group.
- IEEE 802.3 Reduced Twisted Pair Gigabit Ethernet (RTPGE) Study Group.
- IEEE 802.3 Next Generation BASE-T Study Group.
- IEEE 802.3 Distinguished Minimum Latency Traffic in a Converged Traffic Environment Study Group.
- IEEE 802.3 4-Pair Power over Ethernet (PoE) Study Group.
- IEEE 802.3 1-Pair Power over Data Lines (PoDL) Study Group.
- IEEE Std 802.3z-1998, Gigabit Ethernet.
- IEEE Std 802.3aa-1998, Maintenance #5.
- IEEE Std 802.3ab-1999, 1000BASE-T.
- IEEE Std 802.3ac-1998, VLAN TAG.
- IEEE Std 802.3ad-2000, Link Aggregation.
- IEEE Std 802.3ae-2002, 10 Gb/s Ethernet.
- IEEE Std 802.3af-2003, DTE Power via MDI.
- IEEE Std 802.3ag-2002, Maintenance #6 (Revision).
- IEEE Std 802.3ah-2004, Ethernet in the First Mile.
- IEEE Std 802.3aj-2003, Maintenance #7.
- IEEE Std 802.3ak-2004, 10GBASE-CX4.
- IEEE Std 802.3REVam-2005, Maintenance #8 (Revision).
- IEEE Std 802.3an-2006, 10GBASE-T.
- IEEE Std 802.3ap-2007, Backplane Ethernet.
- IEEE Std 802.3aq-2006, 10GBASE-LRM.
- IEEE P802.3ar, Congestion Management.
- IEEE Std 802.3as-2006, Frame Expansion.

- IEEE Std 802.3at-2009, DTE Power Enhancements.
- IEEE Std 802.3-2005/Cor 1-2006 (IEEE 802.3au) DTE Power Isolation Corrigendum.
- IEEE Std 802.3av-2009, 10 Gb/s PHY for EPON.
- IEEE Std 802.3-2005/Cor 2-2007 (IEEE 802.3aw), 10GBASE-T Corrigendum.
- IEEE Std 802.1AX-2008 (IEEE 802.3ax), Link Aggregation.
- IEEE Std 802.3-2008 (IEEE 802.3ay), Maintenance #9 (Revision).
- IEEE Std 802.3az-2010, Energy-efficient Ethernet.
- IEEE Std 802.3ba-2010, 40 Gb/s and 100 Gb/s Ethernet.
- IEEE Std 802.3-2008/Cor 1-2009 (IEEE 802.3bb) Pause Reaction Delay Corrigendum.
- IEEE Std 802.3bc-2009 Ethernet Organizationally Specific type, length, values (TLVs).
- IEEE Std 802.3bd-2011 MAC Control Frame for Priority-based Flow Control.
- IEEE Std 802.3.1-2011 (IEEE 802.3be) Ethernet MIBs .
- IEEE Std 802.3bf-2011 Ethernet Support for the IEEE P802.1AS Time Synchronization Protocol.
- IEEE Std 802.3bg-2011 40Gb/s Ethernet Single-mode Fibre PMD.
- IEEE Std 802.3-2012 (IEEE 802.3bh) Revision to IEEE Std 802.3-2008.
- IEEE Std 1802.3-2001, Conformance Test Maintenance #1.
- IEEE Std 802.3.1-2013 (IEEE 802.3.1a) Revision to IEEE Std 802.3.1-2011 Ethernet MIBs.
- IEEE Std 802.3bk-2013 Extended EPON.
- IEEE P802.3 Ethernet over LAPS liaison Ad hoc.
- IEEE P802.3 Static Discharge in Copper Cables Ad hoc.
- IEEE P802.3 100BASE-FX over dual Single Mode Fibre Call For Interest.

1.5 History of Ethernet

The University of Hawaii's ALOHA network is considered to be the ancestor of all shared media networks[7]. In 1968, Norman Abramson developed a packet radio networking system that ran at 4800bps and 9600bps. In 1973, Robert Metcalfe and David Boggs at Xerox Corporation in Palo Alto, CA applied the ALOHA network principles and created the world's first

LAN illustrated in Figure 1.5.1. The network was first named as ALTO ALOHA and later changed to Ethernet. This first version of Ethernet had a speed upto 2.94 Mbps.

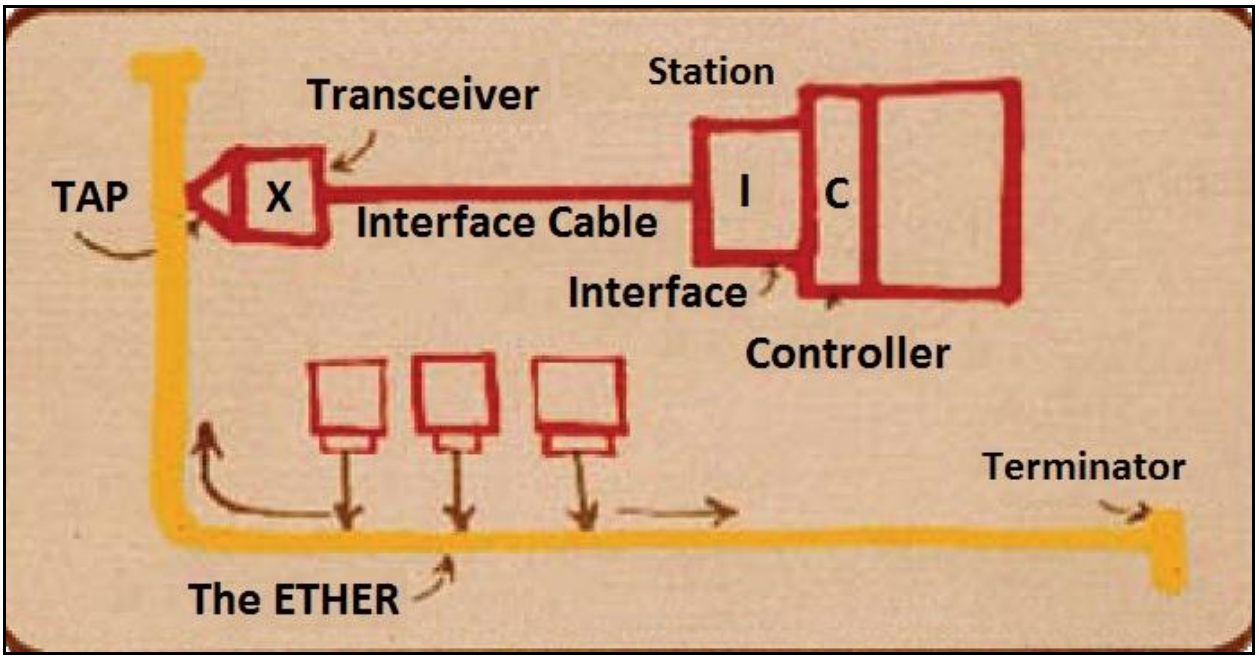


Figure 1.5.1 World's first LAN [8]

This version was used by the White House for word processing. However, this version of Ethernet was not successfully commercialized. The first commercial version of Ethernet was released by DEC, Intel, and Xerox (DIX) in 1980. It was known as Ethernet DIX 80. The second revision release, Ethernet, Version2, Ethernet DIX 82, was released in 1982, which is the standard of Ethernet technology that is in use today. The IEEE formed Project 802 to provide a framework for the standardization of LAN technology. Novell released Novell Netware'86 in 1983, that used a proprietary frame format based on a preliminary specification of the IEEE 802.3 standard. Novell software is used to manage printers and servers.

In 1983, the IEEE approved the IEEE 802.3 standard, which included IEEE 802.2 LLC. This made Novell Netware's proprietary format incompatible with the latest technology. This incompatibility was resolved by creating Sub-Network Access Protocol (SNAP) for the new IEEE 802.3 standard. The overall packet specifications were finalized and then the transmission

medium needed to be agreed upon. In the late 1980s, SynOptics Communications defined and developed a mechanism for transmitting 10Mbps Ethernet signals over twisted-pair cables. Thus, the amalgamation of a low cost transmission medium with agreed packet technology specifications heralded the wide deployment of Ethernet. The Ethernet-over-twisted-pair specification (10BASE-T) was approved by the IEEE in 1990 as the IEEE802.3i standard. It rapidly became the ideal Ethernet media type. Following Figure 1.5.2 summarizes the timeline of Ethernet evolution upto 10Gbps.

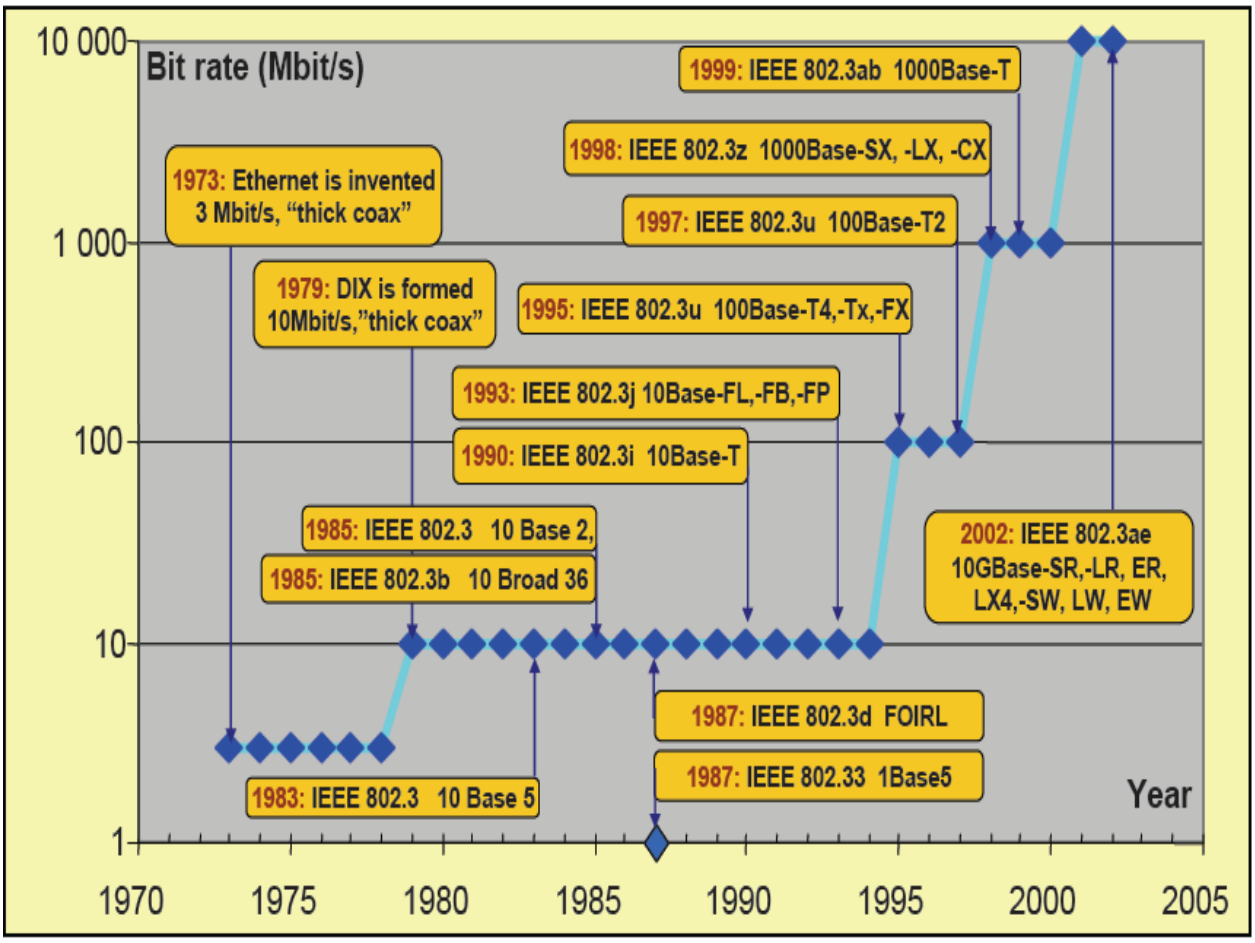


Figure 1.5.2 Timeline of Ethernet evolution [9]

10 Mbps Ethernet evolved to Fast Ethernet (100 Mbps), then Gigabit Ethernet (1000 Mbps), 10 Gigabit Ethernet (10,000 Mbps) and recently to 100Gbps and 400Gbps [144].

Ethernet has evolved rapidly over the past three decades, however, the basic Ethernet frame format and principles of operation have remained practically unaltered. As a result, networks of diverse speeds (10/100/1000 Mbps) operate uniformly without the need for packet fragmentation, packet reassembly or address translation. The advancements in Ethernet have mainly focused on speed and affordability of related cabling and transceiver technologies. It has moved from coax to twisted pair and fiber optic cables and also has been the basis for advanced wireless technologies as IEEE 802.11 Wireless Local Area Network(WLAN) and IEEE 802.16 Worldwide Interoperability For Microwave Access (WiMAX) [10] . As a result, every packet in today's networks starts and ends its existence as Ethernet. Even The next generation of cellular networks will also use this simple, ubiquitous, and extensible technology. Table 1.5.1 gives the Evolution of Ethernet from 10 Mbps to 400Gbps.

Table 1.5.1 Evolution of Ethernet Standards from 10 Mbps to 400Gbps[11].

The Evolution of Ethernet Standards				
Date	**IEEE Standards**	**Name**	**Data Rate**	**Type of cabling**
1990	802.3i	10BASE-T	10Mbps	Cat-3
1995	802.3u	100BASE-TX	100Mbps	Cat-5
1998	802.3z	1000BASE-SX	1Gbps	Multimode fiber
	802.3z	1000BASE-LX/EX		Single mode fiber
1999	802.3ab	1000BASE-T	1Gbps	Cat- 5e or higher
2003	802.3ae	10GBASE-SR	10Gbps	Laser optimized MMF
	802.3ae	10GBASE-LR/ER		Single mode fiber
2006	802.3an	10GBASE-T	10Gbps	Cat-6A
2015*	802.3bq	40GBASE-T	40Gbps	Cat-8 (Class I & II)
2010	802.3ba	40GBASE-SR4/LR4	40Gbps	Laser optimized MMF or SMF
2015*	802.3ba	100GBASE-SR10/LR4/ER4	100Gbps	Laser optimized MMF or SMF
	802.3bm	100GBASE-SR4	100Gbps	Laser optimized MMF

2016*	802.3bs	Under development	400Gbps	Laser optimized MMF or SMF
Note : *These are the proposed study groups				

1.6 Gigabit Ethernet

Gigabit Ethernet has been designed to adhere to the standard Ethernet frame format and maintains compatibility with the installed base of Ethernet and Fast Ethernet products. Thus there is no requirement of frame translation. Figure 1.6.1 describes the IEEE 802.3 Ethernet frame format.

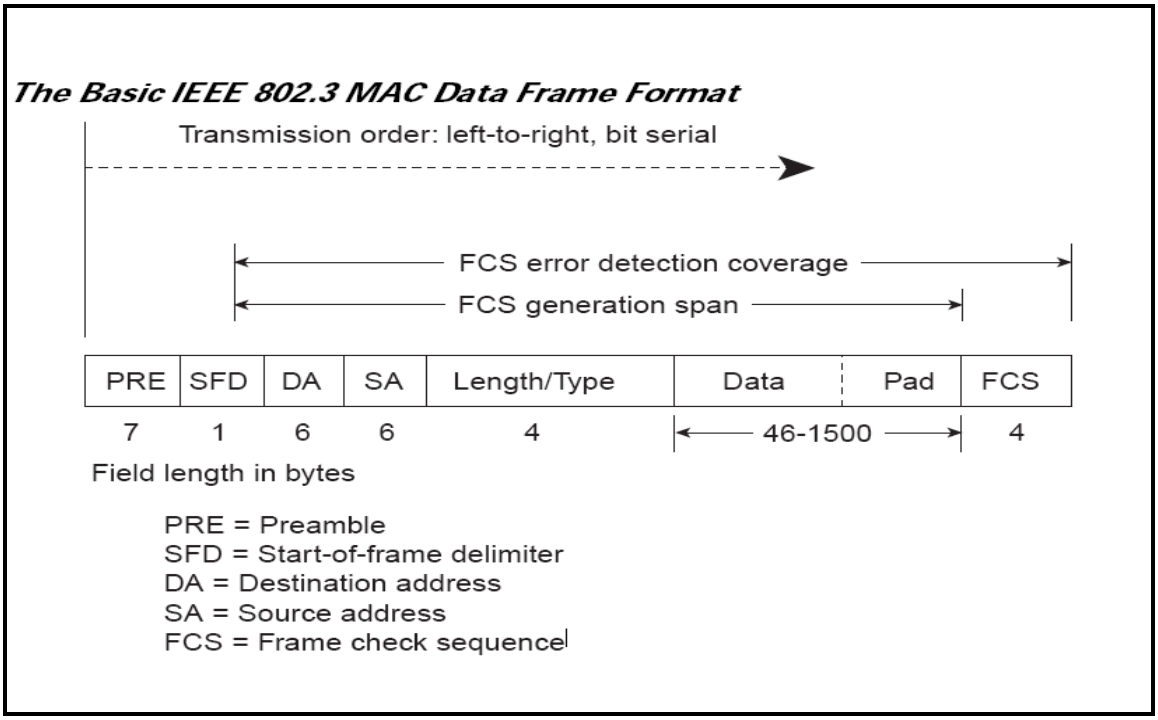


Figure 1.6.1 IEEE 802.3 Ethernet Frame Format [12]

The strategy for Gigabit Ethernet is the same as that for 100-Mbps Ethernet. It defines a new medium and transmission specification, but retains the carrier sense multiple access collision detect (CSMA/CD) protocol and frame format of its 10- and 100-Mbps predecessors [13]. Thus, it maintains compatibility with the slower Ethernet standards, providing a smooth

migration path. Figure 1.6.2 shows a distinctive application of Gigabit Ethernet [14] . A 1-Gbps LAN switch provides backbone connectivity for central servers and high-speed workgroup switches. Each workgroup LAN switch supports both 1-Gbps links, to connect to the backbone LAN switch and also supports high-performance workgroup servers, and 100-Mbps links. Thus high-performance workstations, servers, and 100-Mbps LAN switches are consistently supported.

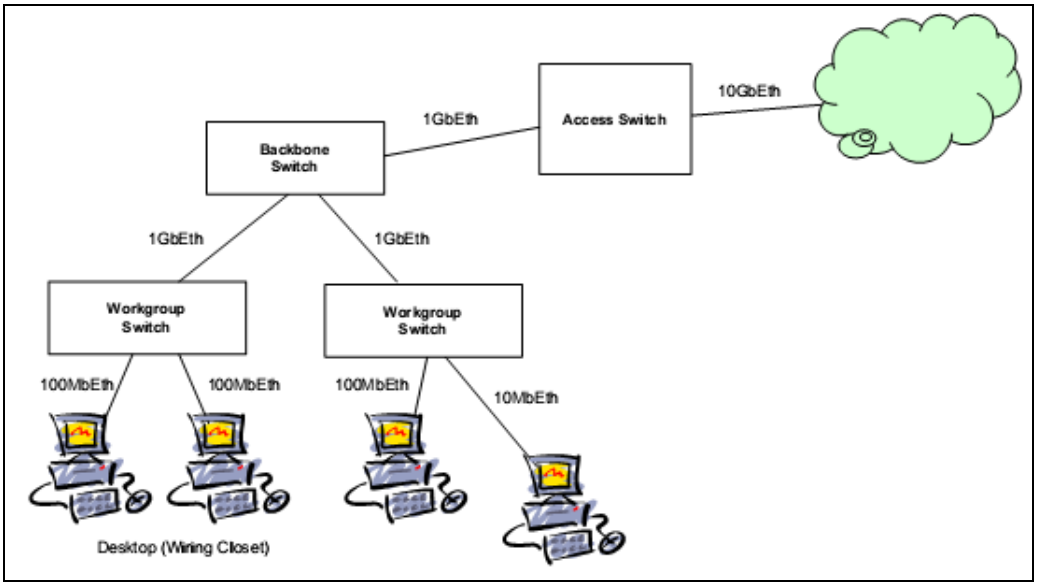


Figure 1.6.2 Enterprise LAN Topology [14]

Gigabit Ethernet Protocol Architecture

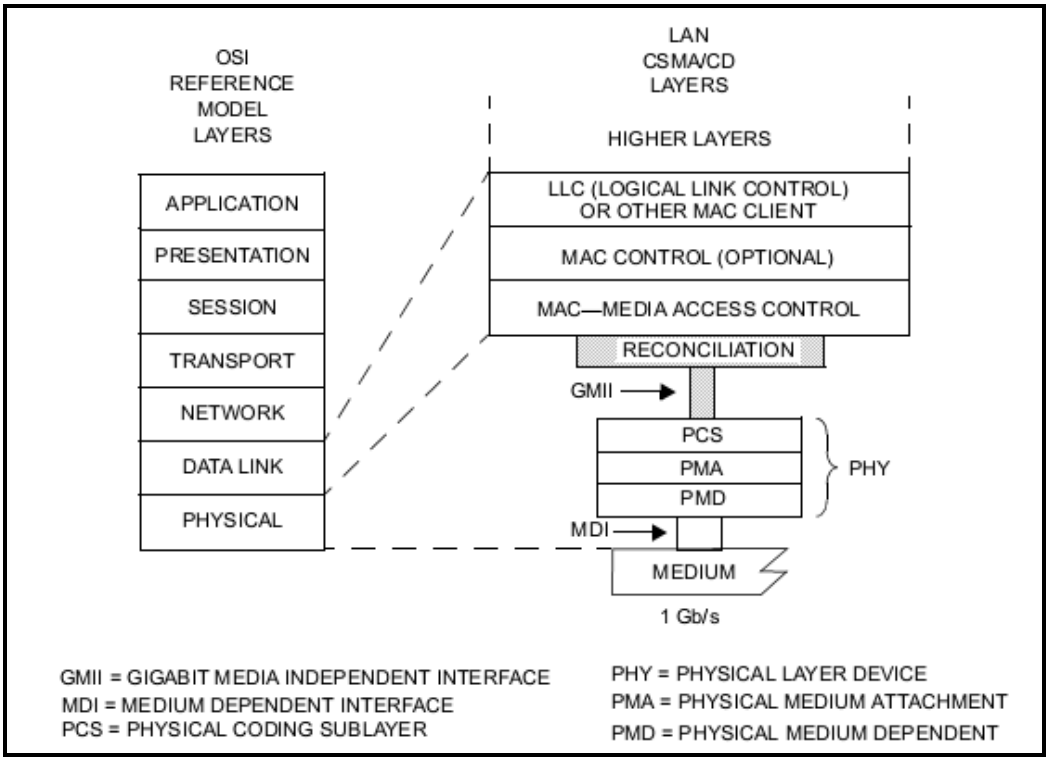


Figure 1.6.3 Gigabit Ethernet protocol architecture [12]

The various layers of the Gigabit Ethernet protocol architecture are shown in Figure 1.6.3. The GMII (Gigabit Media Independent Interface) is the interface between the MAC layer and the Physical layer. It allows any physical layer to connect with the MAC layer and is an extension of the Media Independent Interface (MII) used in Fast Ethernet. The management interface is the same as MII. It supports 10, 100 and 1000 Mbps data rates and provides separate 8-bit wide receive and transmit data paths. Thus, it supports both full-duplex as well as half-duplex operation.

The GMII has 2 media status signals : one indicates presence of the carrier, and the other indicates absence of collision. The Reconciliation Sublayer (RS) maps these signals to Physical Signalling (PLS) primitives which is understood by the MAC sublayer. The same MAC controller can be used with various media types such as shielded and unshielded twisted pair cable, single-mode and multi-mode optical fibre.

The GMII is divided into three sublayers :

Physical Coding Sublayer (PCS)

It provides a uniform interface to the Reconciliation layer for all physical media. It uses 8B/10B coding like Fibre Channel, whereby groups of 8 bits are represented by 10 bit "code groups". Some code groups represent 8 bit data symbols and others represent control symbols. Control symbols are used in Carrier Extension. It generates indications for Carrier Sense and Collision Detection and also manages the auto-negotiation process which is used to determine the network speed (10,100 or 1000 Mbps) and mode of operation (half-duplex or full-duplex).

Physical Medium Attachment (PMA)

This sublayer enables a medium-independent means for the PCS to support various serial bit-oriented physical media. This layer serializes code groups before transmission and deserializes bits received from the medium into code groups.

Physical Medium Dependent (PMD)

This sublayer maps the physical medium to the PCS and defines the physical layer signalling used for various media. The Medium Dependent Interface (MDI), which is a part of PMD is the actual physical layer interface attachments such as connectors, for different media types.

1.7 Emerging trends in Ethernet technologies

Ethernet has played a dominant role today not only for data centers and high-performance computing systems, but also for desktops and laptops. There were other less used network architectures such as, Token Ring, Fiber Distributed Data Interface (FDDI), Asyncronous Tranfer Mode (ATM), High Performance Parallel Interface (HIPPI) and Myrinet that competed with Ethernet in one or more areas. However, Ethernet has not taken over the whole market for standards-based, data networking. Most server vendors offer options for supporting Fiber Channel for attachment of high-performance storage controllers in storage area

networks, and InfiniBand for tightly coupled cluster computing and some high-end storage applications [15]. These networks provide features and capabilities which are not currently available in Ethernet. Thus, they are likely to find oppurtunities in applications beyond the focus of Ethernet vendors.

Table 1.7.1 gives a detailed specification of the Gigabit Ethernet technologies from 1Gbps to 100Gbps[15]

Ethernet Variant	Data rate (Gbps)	Min. Reach (meters)	Form Factor	Media Wavelenghth	Standard IEEE 802.3
1000BASE-					
T	1	100	RJ45, SFP, GBIC	TP Copper	ab
SX		300	GBIC, SFP	LQMF 850nm	Z
LX		5000	GBIC, SFP	SMF 1310nm	
CX		25	HSSDC/DB9	Twinax Copper	
ZX		70,000	GBIC, SFP	SMF 1550 nm	Defacto
10GBASE-					
Ethernet Variant	Data rate (Gbps)	Min. Reach (meters)	Form Factor	Media Wavelenghth	Standard IEEE 802.3
SR	10	300	XENPAK, XFP, X2, SFP+	LQMF 850nm	
LR		10,000	300-oin, XENPAK, XFP, X2, SFP+	SMF 1310nm	

LX4	4 x 2.5	300	XENPAK, X2	FDDI- grade MMF 1310nm	ae-2002/-2005
ER	10	40,000	XENPAK,XFP, X2	SMF 1550nm	
ZR		80,000	XENPAK, SFP+	SMF 1550nm	
LRM		220	XFP, SFP+	FDDI- grade MMF 1310nm	aq-2006/-2006
CX4		15	MicroGigaCN	Twinax Copper	ak-2004/-2005
T		100	RJ45 CAT6A or above	TP Copper	an-2006/-2006
CR		15	SFP+ Direct Attach Copper	Twinax Copper	SFF standard
40GBASE					
CR4	4 x 10 G	7	QSFP+ Direct attached Copper	Twinax Copper	bg
SR4		100/125	QSFP+	LQMF 850nm	
LR4		10,000	QSFP+, CFP	SMF 1310nm	
FR	40G	2000	CFP	SMF 1310nm/1550nm	bg
100GBASE					
CR10	10 x 10	7	CXP Direct attached Copper	Twinax Copper	ba
SR10		100/125	CXP, CFP	LQMF 850nm	
LR4	4 x 25	10,000	QSFP+, CFP	SMF 1310nm	

ER4		40,000	CFP	SMF 1550nm	
40GBase-					
TBD*	4 x 10	40,000	TBD/QSFP+, CFP2, CFP4	SMF 1550nm	bm
100GBase-					
CR4	4 x 25	Board – TBD Cable – 5m	PHY and Management Parameter for 100 Gbps Operation Over Backplane and copper Cables	Copper	bj
UR4*	4 x 25	20	TBD/QSFP+, CFP2, CFP4		
SR4	4 x 25	100	TBD/QSFP+, CFP2, CFP4	LOMF 850nm	bm
Note : *These are proposed and not yet in the draft standard.					

Deployment of 10 Gb/s Ethernet has followed a much slower adoption following standardization, and much lower total volumes. There may be two likely reasons for this. First, design and standardization of 10 Gb/s links over twisted-pair cable has taken a long time. Hence, more expensive optical transceivers and cables had to be used. Moreover, twisted-pair cabling at 10 Gb/s data rates required PHY circuits with sophisticated signal-processing capability. Within enterprise installations, 100 Gb/s Ethernet will be the dominant technology for interconnecting servers to each other and to storage and high-end parallel visualization systems. The IEEE has announced a new group that aims to bring wired Ethernet speeds up to 1 Tbps by 2015 and as fast as 10 Tbps by 2020. This announcement is based on the publication of the IEEE 802.3 Ethernet Bandwidth Assessment report, which found that terabit-speed networks will be essential to support overall capacity requirements by 2015 if current trends continue [16]. The report found that, in 2010, the majority of IXP (Internet Exchange Provider)

links were running at 10Gbps, while all other forms of Ethernet were trending downward in the industry as shown in Figure 1.7.1.

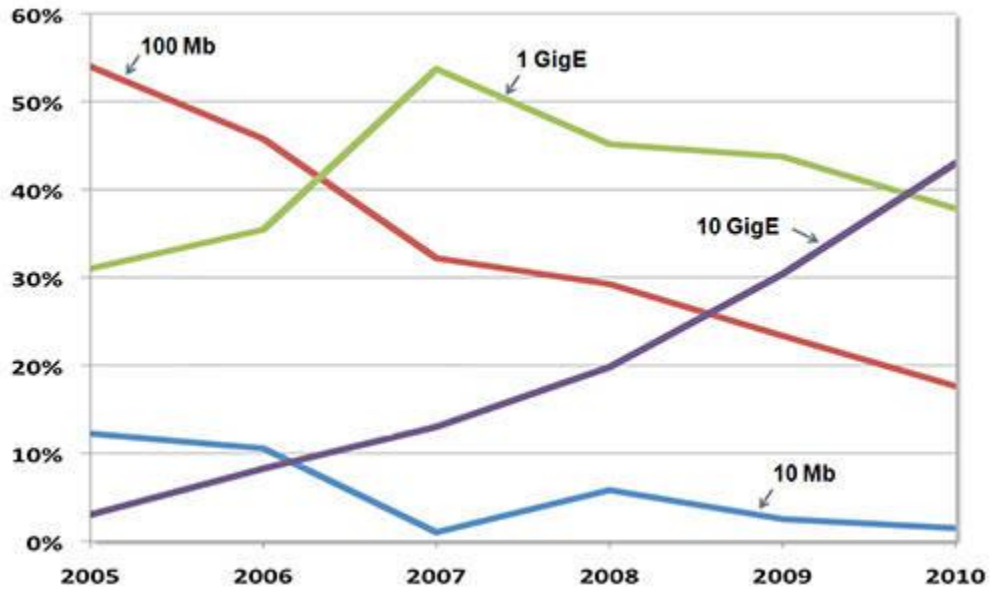


Figure 1.7.1 Use of 10Gbps Ethernet has grown steadily among internet exchange providers [6]

As of 2011, amount of bandwidth used by streaming media had more than doubled since the previous year. The most aggressive growth in network bandwidth utilization is in the financial sector and in High Performance Computing (HPC) for data-intensive science, the IEEE report found. For example, the Large Hadron Collider pushes 15 petabytes of data per year to Tier 1 storage sites around the world. Also, the high-frequency trading systems used by the finance industry, transmit data uncompressed, hence resulting in larger bandwidth consumption. Figure 1.7.2 projects the dominating Gigabit Ethernet links in 2011 and 2013.

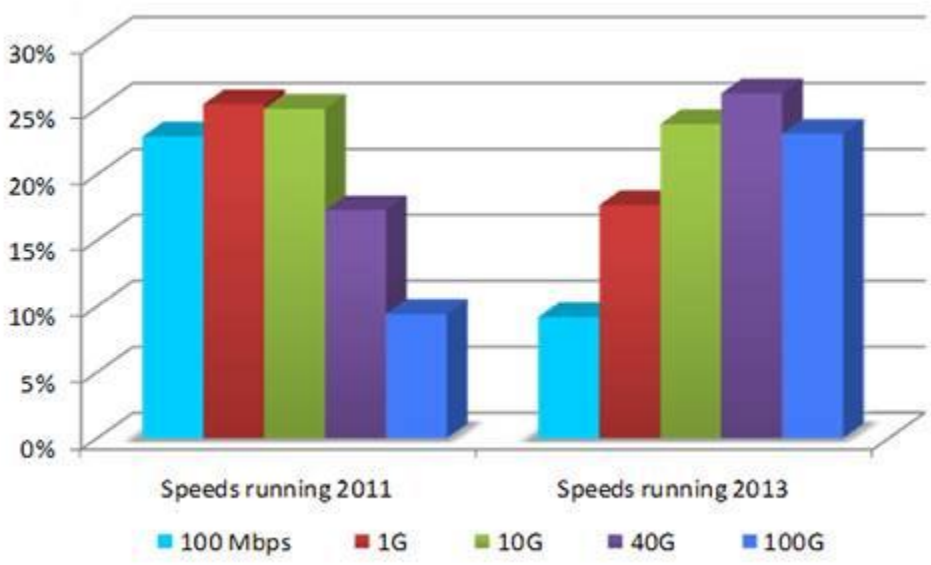


Figure 1.7.2 Dominating Gigabit Ethernet links [6]

A recent study by Intel has concluded that the Ethernet protocol is now on more than 90 percent of all networked devices globally. That percentage will grow as the "Internet of things" allows an enormous number of devices to communicate with each other and with us [17]. Figure 1.7.3 illustrates the impact of 100 Gbps Ethernet.

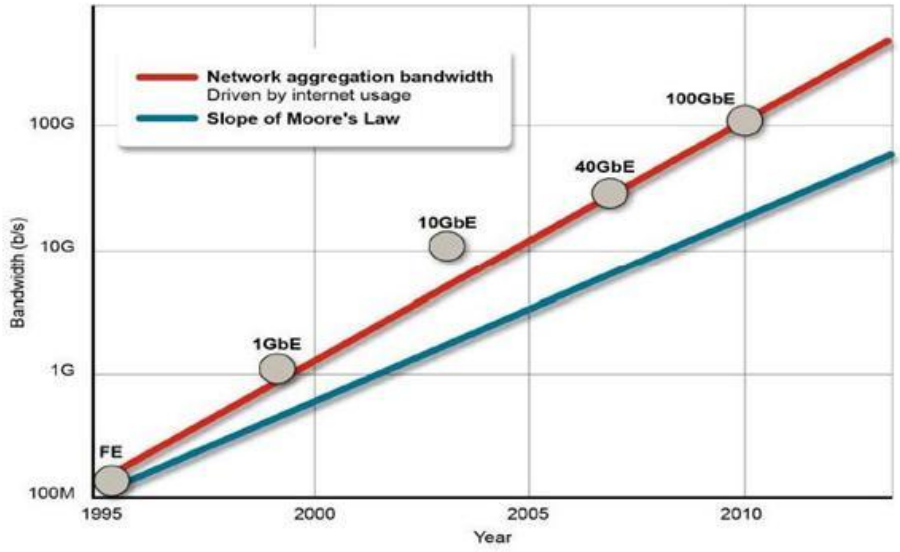


Figure 1.7.3 Impact of the 100Gbps Ethernet [18]

Even while the standards committees are debating the implementation of 400 Gbps standard for Ethernet, Metcalfe noted that it's time to start thinking about terabit Ethernet, probably with a 64-bit addressing system. Considering the rate of innovation that Ethernet seems to spawn, looks like Terabit Ethernet will appear in 10 years or less.

1.8 Literature Review

The scope of research can be broadly classified into three main categories namely the performance analysis of Gigabit Ethernet protocol, the development of Cyclic Redundancy Check (CRC) algorithms for error detection and their performance studies and the study of error correction codes used in noisy communication channels. Hence, we give a brief literature review of the areas of interest as follows:

1.8.1 Review of Performance Analysis of Gigabit Ethernet

Performance analysis of Gigabit Ethernet standard 802.3z Gigabit Ethernet was made on a simulated network with different number of stations and packet sizes by S. Finkler and D. Sidhu [19]. Gigabit Ethernet is able to maintain a throughput of 60-80% for packet sizes less than slotTime and maintains 90-98% throughput for larger packet sizes. Because of bursting, it scales very well for a large number of transmitting stations. Various experiments and analysis were carried out and Gigabit Ethernet maintained a throughput greater than 90% for upto 200 stations.

Design and evaluation of the hardware platform for the Field Programmable Gate Array (FPGA)-based Gigabit Ethernet/PCI Network Interface Card (NIC) using Avnet platform and Xilinx MAC core was studied extensively by Tinoosh M. [20] at Houston, Texas in 2004. The FPGA-based NIC uses multiple memories, including SDRAM SODIMM, for adding different network services. Experimental results at Gigabit Ethernet receive interface reveal that the NIC can receive all packet sizes and store them at SDRAM at Gigabit Ethernet line rate. The overall throughput, transfer latency and bus utilization is improved by improving the processing latency of each interface. The receive FIFO provides enough buffering space for up to 10

incoming large frames of 1518 bytes, thus improving the performance in receiving small packets. The latency in bus interface operations is reduced by pipelining and improving the RTL coding in OPB interface implementations, which results in 32% reduction in total data transfer latency and 72.5% improvement throughput for OPB single transfers.

A partially reconfigurable network controller IIM7010 was developed on a Xilinx Virtex-II XCV1000 FPGA in [21]. The functioning of this controller was then verified and also its throughput and packet loss was analysed. Tests were conducted to find the impact of partial reconfiguration on the performance of network controllers. IIM7010 network controller uses the TCP/IP family of protocols to communicate over a 10 Mbps Ethernet LAN network.

The Gigabit Ethernet platform based on a PCI card was designed and built at CERN, Geneva, Switzerland in 2005 [22]. Altera Stratix FPGA was the main component. Two Gigabit Ethernet applications were implemented, namely a network tester and a network emulator. The network tester was used to assess network equipment intended to be used in the ATLAS TDAQ network. It is used to verify whether all switching devices meet the required specifications. The network emulator is a packet processor, whose architecture comprises a packet path and a control path. It can introduce a quality degrader and the behaviour of an application can be studied under different types of network conditions.

The design of NIC using Altera Stratix II GX was discussed by Bianco [23], which uses about 33% of the 41,250 logic elements of the FPGA. The developed NICs were incorporated on a high-end PC with a SuperMicro X5DPE-G2 mainboard with one 2.8GHz Intel Xeon processor, 1 Gbyte of PC1600 DDR RAM comprising of two interleaved banks, achieving the memory bus transfer rate up to 3.2 Gbyte/s. The motherboard consists of three PCI-X 133 and three PCI-X 100 slots having peak transfer rates of 1 Gbyte/s and 0.8 Gbyte/s respectively. The board used as NIC is equipped with an Altera Stratix GX FPGA. The FPGA consists of 3 Mbit of integrated DRAM, which can be used to allocate FIFOs, buffers and storage of debug data. All queues are about 15 kbytes. The PCI core supports all frequencies of the PCI-X standard. The scheduling algorithm and the manipulation of packets can use maximum clock frequency of 66MHz. An

Agilent N2X RouterTester 900 with 8 Gigabit Ethernet ports, which could transmit and receive Ethernet frames of any size at full rate, was used as traffic source and sink. This NIC obtained a wire speed data transfer reaching up to 6 Gbit/s of aggregate throughput for 64-byte packets. This work was done in the framework of the EURO project and partly funded by the Italian Ministry of University, Education, and Research.

A reconfigurable and programmable Gigabit Ethernet network interface card RiceNIC was built upon the commercial Avnet Virtex-II Pro development card and provides a custom FPGA design, and all its raw components can function as a system. The NIC includes multiple FPGAs, two 300 MHz embedded PowerPC processors, large on-NIC memories greater than 256 MB, and a copper Gigabit Ethernet interface. Custom NIC firmware is available along with a Linux device driver. RiceNIC can saturate the Gigabit Ethernet network link with maximum-sized packets using a single PowerPC processor. This leaves significant resources on RiceNIC to use for experimental networking research [24].

A new 1-cycle pipeline microprocessor, a fast Ethernet transceiver and low cost high performance embedded network controller was established on Altera Stratix EP1S25F780C6 in [25]. TCP/IP stack is designed to access the Internet. Simulations were performed, where the throughput of Ethernet, UDP (User Datagram Protocol) and TCP packets were found to be 7 Mbps, 5.8 Mbps, and 3.4 Mbps respectively.

An attempt to design Ethernet MAC was made at Dr. M.G.R. University, Chennai, India by D. Thomas and K. S. M. Panicker. They added data compression/decompression to get the data rate more than 1Gbps. In this paper, they argue that it is time to review the MAC layer and incorporate advances made in the protocol performance field during the last twenty years [26].

Gigabit Ethernet data streams were generated over both fibre and copper links using a Xilinx Virtex 4 development system [27]. The paper presents details of the firmware developed to drive the links. Request-response protocols are used and throughput, network latency and packet loss is measured over standard Ethernet networks to PCs. Sequential request and group request DAQ data collection protocols have been implemented. The Virtex4 FPGA contains the hardware embedded MAC modules, which can be combined with the high-speed serial

communication module RocketIO to provide a complete gigabit Ethernet solution without the need for any external support logic. The only hardware required is a standard Small Form-factor Pluggable (SFP) module.

The NetFPGA platform containing Xilinx Virtex2-Pro 50 and Xilinx Spartan II FPGA, which can be programmed with user-defined logic was developed at Stanford University [28]. NetFPGA platform enables rapid prototype and development of multi-Gigabit/second line rate networking applications. The open-source gateware, hardware and software is available. A network traffic capture system and a packet generator was implemented on the NetFPGA. The NetFPGA enables rapid development of hardware-accelerated packet processing applications. The Internet packets can be transmitted at line rate on upto four Gigabit Ethernet ports simultaneously.

IP based Gigabit Ethernet connection is designed and implemented directly in hardware by Nicholas Tsakiris and Greg Knowles in [29]. It implements the ICMP, UDP and the new UDP-Lite standards. It was designed in VHDL and uses 1000 slices of the Xilinx Spartan 3 FPGA, running at full Gigabit ethernet speed.

The Gigabit network performance tester with full line rate is developed using Samsung's S3C2410 processor in [1]. The instrument can measure the performance of network across the sub-network at full line rate. Broadcom's product BCM5421S, supporting 10/100/1000 Mbps speed is used as Ethernet physical layer transceiver.

1.8.2 Review of CRC Error Detection Algorithm Development

Hardware implementation of CRC using LFSR is explained as well as five software algorithms namely Bitwise, On-the-fly, Table Lookup, Reduced Table Lookup, Bytewise and Wordwise Reduced Table Lookup Algorithms are explained along with their codes in [30]. The different algorithms are compared in terms of their speed and storage requirements.

G. Albertango and R. Sisto have presented a method of designing hardware parallel encoders for CRCs which is based on digital system theory and z-transforms in [31]. This method can be used by designers in deriving the logic equations of the parallel encoder circuit

for any generator polynomial. Pei and Zukowski investigate the use of VLSI(Very Lage Scale Integrated Circuit) technology to increase the speed of CRC circuits in [32]. It is found that CMOS technologies can achieve a speed of 1 Gb/s data rates.

Castagnoli et al. have investigated several classes of CRC codes with 24 and 32 parity bits in [33] using the method developed by T. Fujiwara et al. for computing the minimum distance of shortened Hamming codes using the weight distribution of their dual codes. The paper [34] presents the VLSI layout generation of a programmable CRC chip for 16 bit CRC using serial implementation. The chip is simulated at 600MHz speed.

A comparative study of ten different implementation strategies for computation of Cyclic Redundancy Checks has been done in [35].The paper also presents a novel architecture suitable for use as an IP in a protocol processor. The different implementation techniques have been divided into application areas based on their speed, flexibility and power-consumption.

Byte-oriented method for CRC computation is described in [36], which reduces the calculation time by a factor of four. A systematic method to calculate CRC in parallel is introduced in [37]. Galois Field property and lookahead technique is used to derive equations of two types of encoding/decoding schemes and their related hardware implementations**.**

A symbolic simulation based algorithm which derives optimized Boolean equations for a parameterizable data width CRC generator /checker is described in [38]. The equations are then used to generate the corresponding VHDL description, which is then synthesized to gates. The paper also presents the area and timing results of the implemented CRC circuit. The CRC-32 polynomial was implemented using the algorithm.

A methodology to determine an optimal CRC polynomial for applications using short messages is described by Tridib Chakravarty in [39]. This methodology has recognized an optimal 12-bit CRC that yields better error detection than CCITT 16-bit CRC, which is used for embedded networks having typical message lengths of 64 bits.

An exhaustive search of the 32-bit CRC design space is presented in [40]. All polynomials achieving a better HD than IEEE 802.3 CRC-32 polynomial are identified.

A theoretical result to realize high-speed hardware for parallel CRC checksums is presented in [41]. The new scheme is faster and is independent of the technology used in its realization. Here, the number of bits processed in parallel can be different from the degree of the polynomial generator. Also high-level parametric codes that are capable of generating the circuits, when only the polynomial is given, is developed.

A polynomial selection process for embedded network applications is described in [42] and a set of good general-purpose polynomials is proposed. A set of 35 new polynomials provide good performance for 3- to 16-bit CRCs for data word lengths up to 2048 bits.

Generator polynomials for CRC codes of degree eight over GF(2) are investigated by Baicheva et al. in [43]. Their minimum distance, properness and undetected error probability for binary symmetric channels (BSCs) are compared with the existing ATM standard.

A high-speed parallel CRC implementation based on unfolding, pipelining, and retiming algorithms is presented by Cheng and Parhi in [44]. The proposed design can increase the speed by 25% and control or reduce hardware implementation cost. A technique for pipelining the calculation of CRCs, such as CRC32 is introduced in [45]. It is shown how CRC throughput can scale linearly with data width and area. Also pipeline latency and operating frequency can be traded off for area.

Polynomials of degree eight over GF(2), which are used as generator polynomials for CRC codes are investigated in [46]. Their minimum distance, properness and undetected error probability for BSCs are compared with the existing ATM standard. A fast CRC algorithm that performs CRC computation for any length of message in parallel is given in [47]. First, the message is chunked into blocks of M-bits equal to the number of bits in CRC. Then CRC computation is done in parallel using Galois Field (GF) multipliers. The final CRC is obtained by accumulating the products of GF multipliers.

A circuit for CRC computation using two parallel calculation units has been implemented in a 0.35 micron process [48]. The units use 32 bits and 64 bits parallel input respectively. It can achieve I throughput higher than 5.76 Gb/s, thus indicating that 10Gb/s throughput is possible

in modern processes. A CRC calculation unit has been designed by Tomas Henriksson in [49], which handles all control symbols and alignment and the data in accordance with the specification. The unit occupies 0.51 mm2 silicon area in a 0.35 micron technology.

Nordqvist et al. [50] presents a novel architecture for CRC computation, which can be used as accelerating hardware in a protocol processor. The architecture provides high throughput and configurability for different protocols and also would reduce the workload of a host processor considerably.

Parallel computation of n-bit CRC checksum on FPGAs using logic minimization strategy is outlined in [51]. It achieves notably better performance than standard logic optimizers. A parallel fast CRC algorithm for an arbitrary length of message is presented in [52]. The algorithm first chunks the given message into blocks of fixed size equal to the degree of the generator polynomial. Then, CRC is computed in parallel for the chunked blocks using lookup tables. The results are combined using XOR operations. Simulation results show that this approach is faster and more space-efficient than previous methods.

1.8.3 Review of LDPC Error Correction codes

The empirical performance of Gallager's low density parity check codes on Gaussian channels is reported by David J.C. MacKay in [53]. It is shown the performance achieved is better than that of standard convolutional and concatenated codes. The performance achieved is almost as close to the Shannon limit similar to that of Turbo codes.

Significant improvement is shown by the analogous codes defined over finite fields GF(q) of order q>2 by in M.C. Davey and D.J.C. MacKay in [54]. The results of Monte Carlo simulations of the degree of infinite Low Density Parity Check (LDPC) codes, which can be used to attain good construction for finite codes are presented. The empirical results for the Gaussian channel of a rate ¼ code with bit error probabilities of 10^{-4} at Eb/No =.05 dB are also presented.

Low density parity-check codes known as Gallager codes, repeat-accumulate codes, and turbo codes, are reviewed by David Mackay in [55]. Results of extensive experimental studies of performance of LDPC codes, also known as “Gallager codes” [56] and “MN” (Mackay-Neal) codes [57] are presented in [58]. The results are obtained for binary symmetric channels and Gaussian channels, which demonstrate that practical performance which is better than that of standard convolutional and concatenated codes can be achieved. The performance of Gallager codes obtained is almost as close to the Shannon limit as that of turbo codes.

Turbo codes and LDPC codes are assessed in terms of their flexibility to hardware implementation by B. Levine et. al. in [59]. LDPC codes are found to be more suitable because their decoders have parallelism, the computations are easy to implement in hardware, and they are more regular than turbo decoders.

An effective general method for determining the capacity of LDPC codes using message passing decoding for any binary-input memoryless channel with discrete or continuous output alphabets is described in [60]. The capacity can be derived for any desired degree of accuracy. The general method applied to LDPC codes can also be extended to turbo codes and other concatenated coding schemes.

A joint code and decoder design is constructed for a class of (3,k) –regular LDPC codes by T. Zhang and K.K.Parhi[61]. The decoder architecture, which is partly parallel, is appropriate for efficient VLSI implementation and has very good performance.

The paper [62] presents two decoding schedules and the equivalent serialized architectures for LDPC decoders and are useful to codes with randomly generated parity-check matrices or using geometric properties of the elements in Galois fields. The staggered decoding schedule is appropriate for forward error correction applications in magnetic recording, wireless, wireline and optical communication systems.

H. Futaki and T. Ohtsuki proposed the decoding algorithm for the LDPC-COFDM(LDPC coded Orthogonal Frequency Division Multiplexing) in [63]. It used M-PSK using Gray mapping on a flat Rayleigh fading channel. LDPC-COFDM systems with M-PSK using Gray mapping have

found to have better error rate performance [64] than the systems using natural mapping on an AWGN channel. They also illustrate that, on a flat Rayleigh fading channel, the LDPC-COFDM systems with QPSK is more effective than other systems.

A 1024-b, rate-1/2, soft decision LDPC code decoder has been described by Blanksby and Howland in [65]. The decoder has a parallel architecture, supporting a maximum throughput of 1 Gb/s with 64 iterations and matches the coding gain of equivalent turbo codes. The parallel architecture enables rapid convergence resulting in a power dissipation of only 690 mW.

The paper [66] describes the improvement in Maximum Likelihood Decoding Algorithm (MLDA) proposed in [67] for the development of LDPC codes. The improved design is also applied to the decoding of Progressive-Edge-Growth (PEG) LDPC codes [68] over Binary Erasure Channels (BEC). The study demonstrates clear improvements over current LDPC decoding algorithms.

A high throughput, parallel, scalable and irregular LDPC coding and decoding system hardware is implemented on FPGA in [69]. The architecture consists twelve combinations of block lengths 648, 1296, 1944 bits and code rates 1/2, 2/3, 3/4, 5/6 based on IEEE 802.11n standard and is tested on Rice Wireless Open Access Research (WARP) Platform.

MatLab/Simulink models are developed for the Zigbee/IEEE 802.15.4 protocol in [70] and the performance evaluation is presented. The data rate and power is varied to find the effect on BER to SNR relationship. Higher the data rate, higher is the probability of error for a desired SNR. Also experiments were performed to quantify interference effect of Zigbee devices on the throughput performance of the IEEE 802.11g/WLAN and vice versa. It is found that the Zigbee interference has more effect on the IEEE 802.11g uplink rather than the downlink. Also IEEE 802.11g is greatly more affected by Bluetooth than Zigbee.

An encoder/decoder pair and code construction methodology for LDPC codes is introduced in [71] which permits the use of a wide range of code lengths and rates. The implementation was done in Virtex-4 part (XC4VLX25–12-FF668) FPGA using XST as a synthesis tool and ISE 7.1.03i. The encoder and decoder design achieved a maximum clock frequency of

278 MHz and 181 MHz respectively. Similarly the encoder design used 614 slices with 11 Block RAMs and decoder design size was 10970 slices with 46 Block- RAMs. Simulations are performed and the BER curves for an Additive White Gaussian Noise (AWGN) channel are plotted.

It was established that working in a higher order Galois field, significantly improve the performance of the LDPC code with moderate code lengths by V.S. Ganepola et.al. in [72]. Simulation studies in AWGN channel indicated that there was a significant performance gain between the codes over GF(2) , GF(4) , GF(16) , GF(64) and GF(256), reaching towards the Shannon's limit.

A method for estimating the frame error rate (FER) and bit error rate (BER) and thus the performance of LDPC codes decoded by hard-decision iterative decoding algorithms on BSCs is proposed by Hua Xiao and A. H. Banihashemi in [73]. The proposed method can be effectively used for both regular and irregular LDPC codes, a variety of hard-decision iterative decoding algorithms and also, it has a much smaller computational complexity, particularly for lower error rates compared with the conventional Monte Carlo simulation. They also proposed an algorithm in [74], which can competently find out the size and the number of the smallest error patterns that a hard-decision iterative decoder for LDPC codes over a BSC fails to correct. Thus, the error rate performance of LDPC codes over the whole range of channel crossover probabilities of interest can be estimated.

A modified min-sum decoding algorithm based on classified correction is proposed for LDPC codes by Z. Zhong et. al. in [75]. The proposed algorithm is different from the single correction in the normalized Belief Propagation (BP)-based and offset BP-based algorithms because it utilizes two corrections for minimum and sub-minimum magnitudes of input messages in check nodes. Simulation results indicate that the proposed algorithm achieves performance very close to that of the BP algorithm The decoder is implemented in VHDL and targeted on the Xilinx Virtex2p x2vp50 FPGA and achieves the maximum clock frequency of 155.8MHz.

An efficient high level approach to designing LDPC decoders is proposed in [76]. The high level design methods use Simulink, with predefined library components and with embedded Matlab codes. Performance of the LDPC decoding algorithm was evaluated by simulating the decoder over AWGN channel and plotting the BER against Signal-to-Noise-Ratio (SNR). Synthesis results for the high-level designs are compared with corresponding VHDL designs. The results were obtained using Altera's Quartus II software with Cyclone II EP2C70F672C6 FPGA device. It is found that Simulink design uses much less resources and is faster than the VHDL only design. Both designs were implemented on a Xilinx Spartan 2E FPGA.

Md. Murad Hossain et. al have introduced modified log-domain algorithm and min-sum algorithm of sum-product algorithm (SPA) for LDPC codes over GF (q) in [77]. A log-domain implementation has several advantages as far as practical implementation is concerned. The BER performance and computational complexity of the log domain algorithm, modified log-domain algorithm and modified min-sum algorithm is compared. Computer simulations verify that the modified log-domain algorithm has superior BER performance than modified min-sum algorithm for small SNR. Both modified log domain algorithm and modified min-sum algorithm requires no message multiplications. Hence they involve less computational complexity as compared to the SPA.

Yeo et al. [78] discuss architectures for LDPC decoders with methods to reduce their complexity. Area, power, and throughput constraints are of specific interest in the design of communications receivers. LDPC decoders need much less computation to achieve a similar performance, compared to turbo decoders. The choice between a serial and a parallel implementation requires the tradeoff between memory or interconnect complexity. However future practical implementations of LDPC decoders will be trading off the error correcting performance for reduced implementation complexity.

A parallel-serial, flexible, high-throughput architecture for high-performance LDPC decoder is proposed by Z. Zhang et. al. in [79]. This emulation platform can be used to capture low BER values upto 10^{-13} for a (2048,1723) RS-LDPC code and (2209,1978) array-based LDPC

code. High-order modulation formats and advanced error correcting codes are two techniques for performance improvement of ultra high-speed optical transport networks. In [80], Qi Yang et. al. present record receiver sensitivity for 107 Gb/s CO-OFDM (coherent optical OFDM) transmission through constellation expansion to 16-QAM and rate-1/2 LDPC coding.

Performance evaluation of four-dimensional nonbinary LDPC-coded modulation scheme appropriate for next-generation optical transport networks (OTNs) is presented in [81].

A novel scheme of using LDPC codes in atmospheric optical communication system is proposed by Tao Jing-jing in [82]. Receiving sensitivity is improved by employing coherent detection. The performance is evaluated by introducing atmospheric channel attenuations of 20-30 dB/km. It is found that the received power requirement is reduced by ~4 dBm at BER of 10^{-9}.

An architecture for high data rate VLF communication using Gaussian minimum shift keying (GMSK) modulation and LDPC channel coding is proposed by Arun kumar and Rajendar Bahl in [83]. The paper describes the modeling of atmospheric radio noise (ARN) that corrupts VLF signals and an algorithm for signal enhancement in presence of ARN is developed. The BER performance of the system is estimated for bit rates of 400 bps, 600 bps, and 800 bps for communication bandwidth of ~200 Hz.

A new improved construction method of LDPC code, based on the construction method of systematically constructed Gallager (SCG)(4, k) code is proposed by J. Yuan et.al. in [84]. This method has been found to save storage space and has reduced computation complexity for hardware implementation. Thereby, LDPC (5929,5624) code is constructed by the proposed construction method and has better error-correction performance, lower redundancy and lower decoding complexity than that of RS (255, 239) code and can have better suitability for optical transmission systems. Also [85] proposes novel LDPC(3969,3720) code with 6.69% redundancy and the novel LDPC(8281,7920) code with 4.56% redundancy for high speed long haul optical transmission systems and also their performance is assessed.

A study of the errors observed when an optical Gigabit Ethernet link is subject to attenuation is performed in [86]. The investigations are performed for 1000BASE-X, when the receiver power is sufficiently low, so as to induce errors in the Ethernet frames. A traffic generator is used to feed a Fast Ethernet link to an Ethernet switch, and a Gigabit Ethernet link is connected between the switch and a traffic sink and tester. A packet capture and measurement system is implemented using an enhanced driver for the SysKonnect SK-9844 NIC. A variable optical attenuator is placed in the fibre in the direction from the switch to the sink. It is found that some octets suffer from far higher probability of error than others, and that the distribution of errors varies depending on the type of packet transmitted. Similarly, in [87], it is shown that while a pseudo-random BER test may show low error rate, when the studies are performed using real network data, a (M,N) block code results in frequency components that cause non-deterministic error and a poorer overall result. The authors mention that, this conclusion is contradictory to the assumptions of a significant body of coding and protocol work. They also assert that the condition of low receiver power is increasingly likely as networks become more complex, with longer fibre lengths, optical switching systems and higher data rates.

A BER tester implementation based on the Altera Stratix II GX and IV GT development boards is described by Xiang et al. in [88]. The Stratix II GX tester operates at up to 5 Gbps and the Stratix IV GT tester operates at up to 10 Gbps, both in 4 duplex channels. The tester consists of a pseudo random bit sequence (PRBS) generator and detector, a transceiver controller, and an error logger. There is a computer interface for data acquisition and user configuration. A point-to-point serial optical link setup is developed to validate the functionality of the tester. The Stratix II GX tester was also used in a proton test on a custom designed serializer chip to record and analyse radiation-induced errors.

1.9 Organisation of the thesis

The Chapter 2 consists of the objectives of research and the methodology employed to implement the Gigabit Ethernet protocol design. The proposed implementation block diagram

and the architecture is explained in detail. Also, the evaluation techniques are summarized. Chapter 3 describes the details of the FPGA technologies based hardware implementation of the Gigabit Ethernet protocol design, the use of the command line interface and also the experimental setup developed for introducing errors in the Ethernet frames. Chapter 4 gives a background review of information theoretic concepts used in the Simulation models in this work. We review Shannon's noisy channel coding theorem, BER, PER, BER of BPSK, BSC, AWGN, CRC Algorithms, and LDPC error correction algorithms. Also, the simulation models developed in Matlab for error detection analysis and error correction analysis is described. The results are presented in Chapter 5 and the significance of the results is discussed. We also make important conclusions of the study in Chapter 5 and describe several possible avenues of future research indicated by this work.

CHAPTER 2

GIGABIT ETHERNET PROTOCOL

2.1 Objectives of Research

Ethernet protocol has been a robust and dominant technology and it is found to be present on more than 90 percent of all networked devices globally [17] and the percentage has been growing by leaps and bounds. Although the Ethernet protocol speed has evolved from 10Mbps to 400Gbps, the basic Ethernet Frame structure has remained the same with CRC32 error detection technique being employed for handling errors in the communication channel, which involves retransmission of the errored frames. Also, there is a growing demand for data-centric services and multimedia-rich internet applications which has led to the standardization of new Ethernet protocols with improved throughputs namely 10Gbps , 40Gbps, 100Gbps and 400Gbps. This improved throughput has also increased the probability of errors in the channel. Also, Ethernet has been the basis for transmission through fiber optic cables and also advanced wireless technologies like IEEE 802.11 (WLAN) and IEEE 802.16 WiMAX). Hence, we feel there is a need to rethink over the design of the Ethernet frame format, which requires the investigations to be performed with respect to CRC32 error detection capability and possibility of error correction for Ethernet frame which is already available in the new protocols. Therefore the said area was taken up for investigation with following objectives as

1. Development of Simulation model for Error Detection and Correction for Gigabit Ethernet protocol.
2. To implement the Gigabit Ethernet protocol (Ethernet MAC IEEE 802.3z and 802.3ab Standard) on the FPGA (Altera Stratix II GX).
3. To analyse the performance of Gigabit Ethernet protocol for optimum speed.
4. Error Detection in Gigabit Ethernet protocol for Single mode optical fibre media.
5. Error Correction Analysis of Gigabit Ethernet protocol.

2.2 Gigabit Ethernet Protocol Architecture

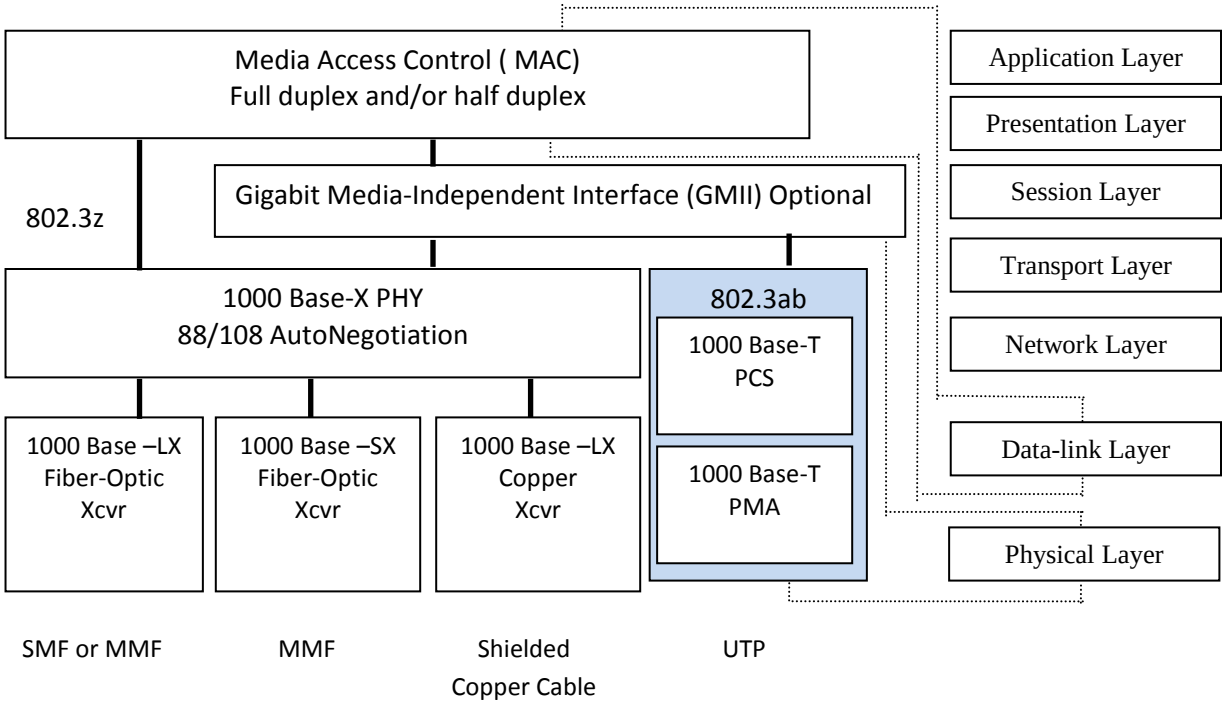


Figure 2.2.1 Gigabit Ethernet layer diagram [89]

Figure 2.2.1 shows the overall protocol architecture for Gigabit Ethernet. Gigabit Ethernet (IEEE802.3z) addresses the two lowest layers of the OSI model Layer 2, the DLL, which describes how data are organized into frames and sent over the network, and Layer 1, the Physical layer, which describes the network medium and signalling specifications [89].

The MAC layer is an enhanced version of the basic 802.3 MAC algorithm. GMII is optional for all the medium options except unshielded twisted-pair (UTP). The GMII consists of independent 8-bit-parallel transmit and receive synchronous data interfaces. It provides a chip-to-chip interface that lets system vendors mix MAC and PHY components from different manufacturers. The 8B/10B encoding scheme is used for optical fiber and shielded copper media, and the pulse amplitude modulation (PAM)-5 is used for UTP.

2.2.1 Media Access Control Layer

The 1000-Mbps specification uses the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) frame format and MAC protocol used in the 10- and 100-Mbps versions of IEEE 802.3.

The basic CSMA/CD scheme has two enhancements for traditional Ethernet hub operation:

Carrier extension

Carrier extension appends a set of special symbols at the end of short MAC frames so that the frame is at least 4096 bit-times in duration compared to the minimum 512 bit-times imposed at 10 and 100 Mbps. Thus the frame length of a transmission becomes longer than the propagation time at 1 Gbps. Figure 1.6.5 shows the Gigabit Ethernet Frame Structure with Carrier Extension.

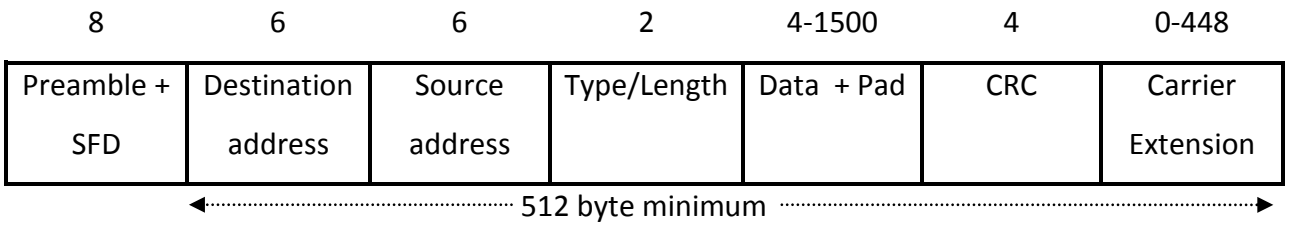


Figure 2.2.2 Gigabit Ethernet Frame Structure with Carrier Extension

Frame bursting

Frame bursting allows multiple short frames to be transmitted successively, up to a limit, without relinquishing control for CSMA/CD between frames. Frame bursting avoids the overhead of carrier extension when a number of small frames are ready to be sent on a single station.

Full Duplex operation

Full duplex operation can be employed by using a LAN switch. Here the CSMA/CD protocol, carrier extension and frame bursting are not used since data transmission and reception at a station can occur simultaneously without interference and with no contention for a shared medium. The Gigabit Ethernet specification expands on the pause protocol used for congestion control defined for 100-Mbps Ethernet by allowing asymmetric flow control. Pause protocol is

implemented by transmission of a short packet known as a pause frame. The frame contains a timer value, which contains a multiple of 512 bit-times and specifies the duration for the transmitter to stop sending frames. If the receiver becomes uncongested before the transmitter's pause timer expires, the receiver may send another Pause frame to the transmitter with a timer value of zero, and the transmitter can resume immediately. The AutoNegotiation protocol is used by a device to indicate that it may send pause frames to its link partner but will not respond to the pause frames.

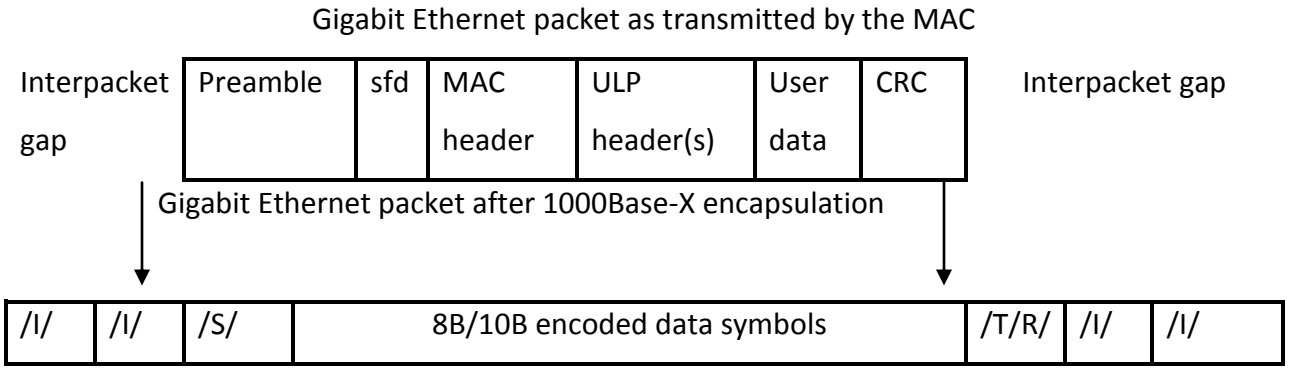


Figure 2.2.3 1000Base-X encapsulation

2.2.2 The Physical Layer

The PCS and PMA together are known as the 1000Base-X PHY. The 1000Base-X PHY performs a link configuration protocol referred to as AutoNegotiation.

Physical Coding Sublayer

The PCS encodes 8-bit data bytes from the GMII into 10-bit code groups, using a block-coding technique known as 8B/10B encoding, as mentioned earlier. The PCS follows a simple set of rules to encapsulate and encode data transmitted by the MAC. As shown in Figure 2.2.3, the /S/ code group is used to indicate the start of a packet. Then, the Preamble, start-frame-delimiter, MAC header, upper layer protocol (ULP) headers, user data and CRC are encoded using the 8B/10B code rules. A pair of special code groups, /T/ and /R/ code groups are used to terminate a packet. Packets containing an odd number of code groups are terminated with a second /R/ code group. The interpacket gap is filled with a two-symbol idle code group represented as /I/,

which is continuously between packets so that the receivers clock recovery circuits at each end of the link maintains phase and frequency lock on the recovered clock and data. Thus the 8B/10B code along with the frame encapsulation rules and a 32-bit CRC, provides Gigabit Ethernet with extraordinary robustness against undetected errors and thus ensures high data integrity.

Physical Medium Attachment

The 10-bit code-groups produced by the PCS are transmitted in a serial fashion by the PMA using a simple non-return-to-zero (NRZ) line coding.

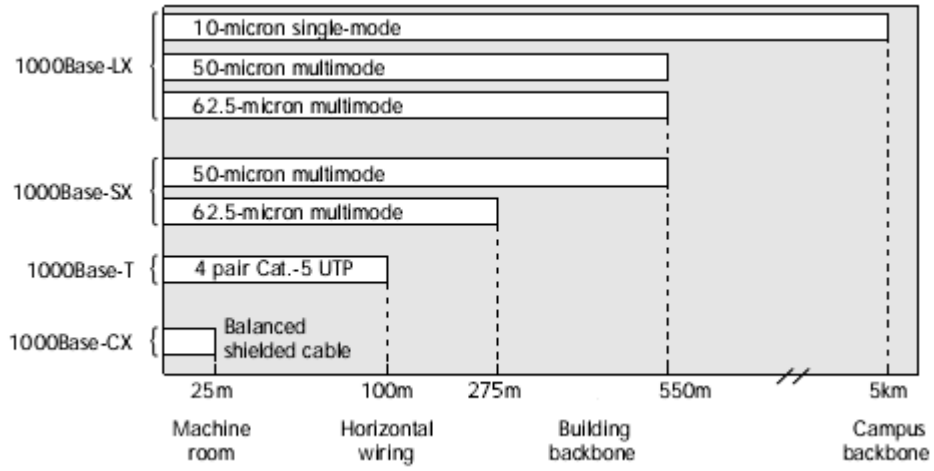


Figure 2.2.4 Gigabit Ethernet application environments and link distances [89]

The 8B/10B code provides excellent DC balance and also has excellent transition density, which is required to perform clock recovery.

AutoNegotiation

AutoNegotiation is used to ensure that the link characteristics are appropriately configured when links are initialized. Defined in Fast Ethernet to automatically select operational speeds between 10 and 100 Mbps, AutoNegotiation was adapted to Gigabit Ethernet mainly to select between duplex mode and the use of link-level flow control. On a 1000Base-X link, a special sequence of 8B/10B codes referred as a /C/ code group is used to exchange configuration information, which transfers bits of configuration information at a time. The AutoNegotiation process takes about 40 milliseconds after the cables are plugged in or the equipment is turned

on. Figure 2.2.4 charts the link distances for the transceiver types supported by Gigabit Ethernet: optical fiber, copper cabling, and UTP.

The 1 Gbps specification for IEEE 802.3 supports the following physical-layer media:

1000Base-LX: uses long-wavelength optic fibre and supports duplex links of upto 550 m of 62.5-µm or 50-µm multimode fiber or upto 5 km of 10-µm single-mode fiber. The wavelengths range from 1270 to 1355 nm. LX is used for longer distance, backbone connections.

1000Base-SX: uses short-wavelength optic fibre and supports duplex links of upto 275 m using 62.5-µm multimode or upto 550 m using 50-µm multimode fiber. The wavelengths range from 770 to 860 nm. SX is cheap and mainly used for short-distance desktop links.

1000Base-CX: uses specialized shielded twisted-pair cable of length less than 25 m. CX is used to connect devices located within a single room or equipment rack, using copper jumpers.

1000Base-T: uses four pairs of Category 5 unshielded twisted-pair copper wires to connect devices over a range of upto 100 m. The data is transmitted on each wire in both directions simultaneously. Data is encoded using 5-level pulse-amplitude modulation (PAM) at a signaling rate of 125 MBaud. AutoNegotiation is done by using patterns of simple link test pulses and facilitates interoperation with existing 10- and 100-Mbps Ethernet standards.

100 Mbps Fast Ethernet is upgraded to 1 Gbps by merging the two technologies, IEEE 802.3 Ethernet and ANSI X3T11 Fibre Channel, together as shown in Figure 2.2.5. Leveraging these two technologies has helped to use the existing high-speed physical interface technology of Fibre Channel and also maintain the IEEE 802.3 Ethernet frame format, backward compatibility for installed media. Full or half-duplex carrier sense multiple access collision detect (CSMA/CD) can be used optionally.

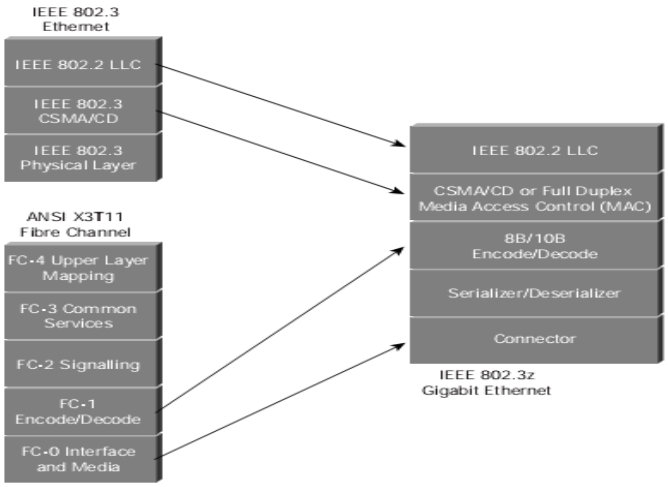


Figure 2.2.5 Gigabit Ethernet Protocol Stack [90]

2.3 Ethernet Media Access layer Implementation

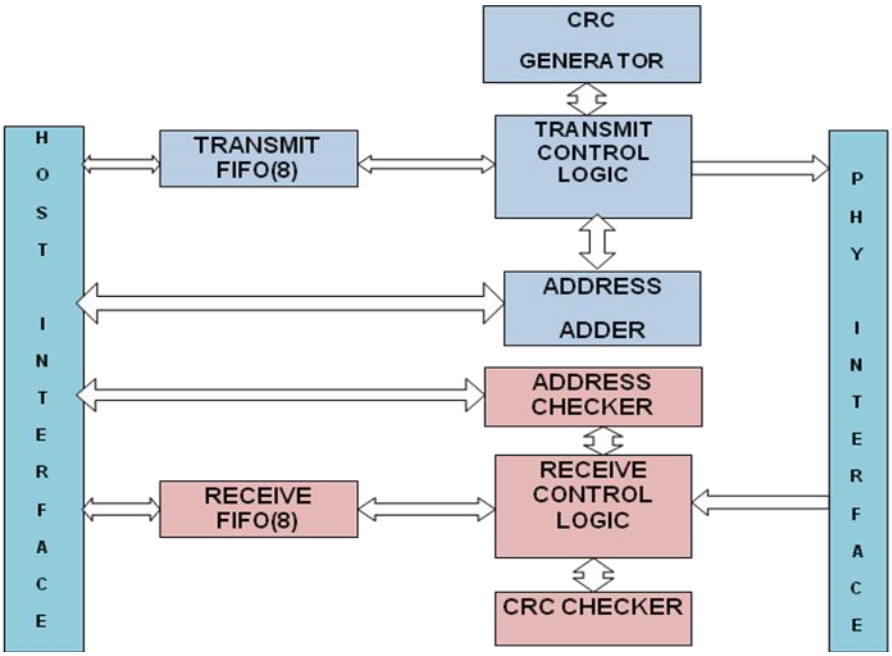


Figure 2.3.1 Proposed MAC Implementation Block Diagram

Figure 2.3.1 gives Block Diagram of the proposed Ethernet MAC architecture.

It comprises mainly of the Transmitter and the Receiver.

- For the transmitter there is a Transmitter FIFO (First In First Out) (8 bit, 1500 bytes) where the data received from host is stored before it is given to the transmitter control module.
- The receiver has a Receiver FIFO (8 bit, 1504 bytes) where it stores the data which it receives from the receiver control module.

From the host interface all the signals are generated which are needed by the MAC module and the respective FIFO buffers.

- The transmitter part contains the Transmitter control module which generates frames.
- The MAC address adder adds the destination address, the source address and the type/length field of the frame which is generated in the transmit control module.
- A CRC generator is included in the transmitter to generate the required CRC bits which is appended as the FCS (Frame Check Sequence).
- The receiver part contains the Receiver control module which decapsulates the frame.
- A CRC checker is used to check for errors in transmission.
- MAC address checker is used for MAC address checking. It also checks for multicast and broadcast packets.

2.4 Methodology of Implementation

To understand the working of the Ethernet MAC Core , we obtained the HDL source code for 10_100_1000 Mbps tri-mode Ethernet MAC IP core of opencores project [91] conforming to IEEE 802.3 specification. The said IP core was simulated and verified using tcl/tk user interface, Modelsim and cygwin platform.

Tcl/tk

Tool Command Language (Tcl) is a scripting language used for rapid prototyping scripted applications, Graphical User Interfaces (GUIs) and testing. Tcl is also used for CGI scripting. The combination of Tcl and the Tk GUI toolkit is referred to as Tcl/Tk.

Modelsim

ModelSim® is a HDL simulator which can be used to simulate electronic circuit behaviour. It can be used with Verilog as well as VHDL.

Cygwin

Cygwin is a Unix-like command-line interface, which can be used with Microsoft Windows. It provides integration of Windows-based applications, data, and other system resources with applications, software tools, and data of the Unix-like environment.The design was compiled using Altera's QUARTUS-II IDE (Integrated Development Environment)software tool.

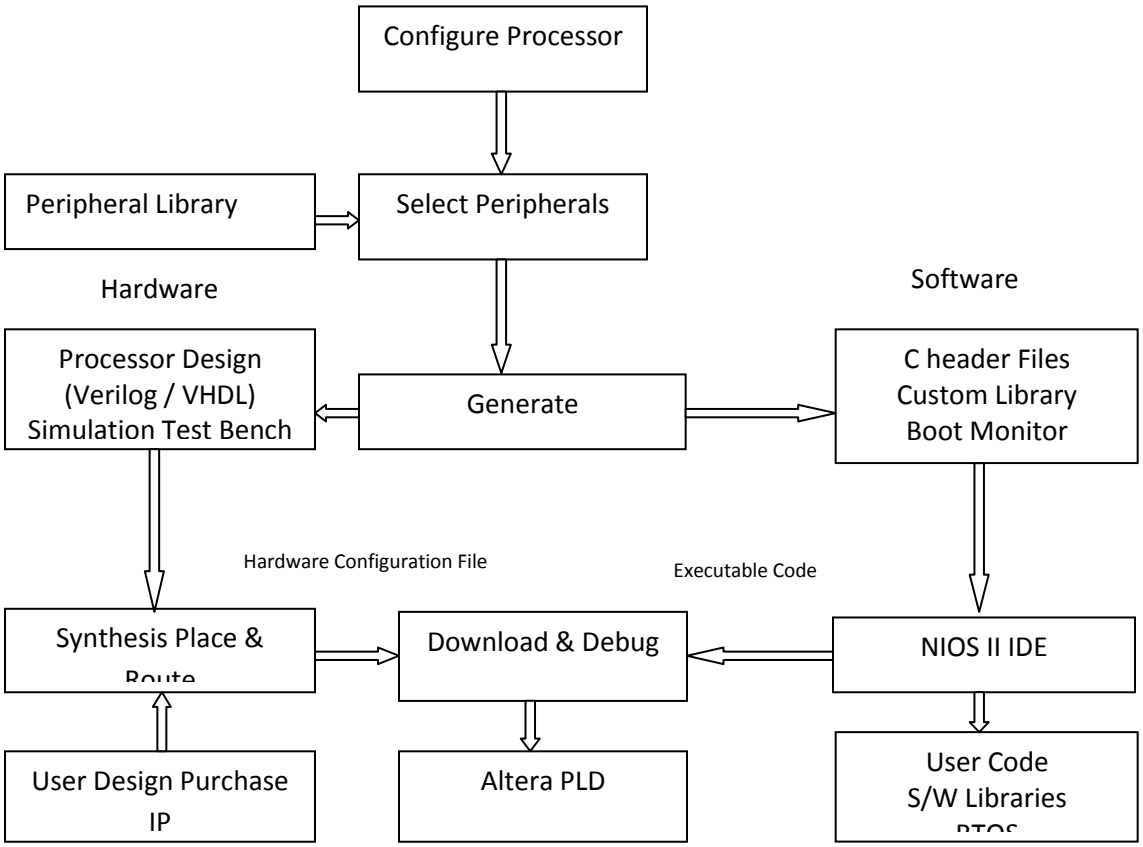


Figure 2.4.1 Various stages associated with QUARTUS-II IDE

Figure 2.4.1 gives the block diagram illustrating the sequence of compilation and the various stages associated with the IDE.

The Gigabit Ethernet protocol design was built using Altera's Triple Speed Ethernet (TSE) IP core. Altera's Quartus II software and System On Programmable Chip (SOPC) builder along with Nios II IDE was used to implement the design on Altera's EP2SGX90FF1508C3 device available on Stratix II GX PCI Express development board. The details of the implementation are described in Chapter 3.

2.5 Evaluation Techniques

The simulation and implementation of network interface architecture is a challenging issue. It requires accurate modeling of the host system architecture, behavior of the processor, memory interface, local peripheral interconnect, and also the design of the interface. All these mechanisms operate asynchronously and involve complex interaction of signals. Also the software such as operating system and device drivers need to be simulated. The Gigabit Ethernet protocol design using Altera's Triple Speed Ethernet IP Core helps to overcome these challenges.

The performance analysis of the implemented Gigabit Ethernet protocol design on Stratix II GX FPGA was carried out by using the test control API, which is provided by Altera Inc., wherein the test of frame transmission and reception can be initiated with different parameters such as speed, transmission mode, frame length, number of frames and also API was used to generate the reports of the test, such as the transmission time, number of frames sent, number of frames received, frames in error and throughput.

A testbed is developed for introducing optical attenuation using single mode optic fibre. Matlab Simulation models are developed for Error Detection and Error Correction Analysis of the Gigabit Ethernet frames. The frames generated by the Gigabit Ethernet protocol design implemented on Stratix II GX FPGA were captured in Wireshark in pcap file format and then interfaced with the Matlab simulation models to obtain the BER curves and the performance was evaluated.

CHAPTER 3

HARDWARE IMPLEMENTATION USING FPGA TECHNOLOGIES AND IP CORES

Gigabit Ethernet applications require line-rate processing for all frame sizes and hence require hardware implementation [92]. FPGAs consists of thousands of logic gates and hence it is possible to verify specific software functions on specific hardware, which reduces the design cycle and therefore the execution cycle time thus making the embedded system respond faster in real-time[93]. A reconfigurable NIC permits swift prototyping of new system architectures for network interfaces. The architectures can be verified in real environment, and prospective implementation bottlenecks can be identified [20]. Dynamically reconfigurable platform also reduces power consumption of the network device [94].FPGA based-platforms accomplish the performance requirements and provide extra flexibility compared to Application Specific Integrated Circuit (ASIC) implementations. Using an FPGA based custom design PCI platform, we have incorporated a 10/100/1000Mbps MAC design to build a low cost, high performance embedded network controller. The physical network interface is provided by a standard SFP module. The details of the hardware implementation are explained below.

3.1 Platform and Resources

The design uses a Stratix®II GX Peripheral Component Interconnect Express (PCIe) development board for the Hardware Implementation. This board has 16-integrated transceiver channels and support for high-speed, low-latency memory access via DDR2 SDRAM and QDRII memory interfaces and provides a fully-integrated solution for multi-channel, high-performance applications, while also using limited board space.

Following are the major components of the board:

Off-chip memory : DDR2 SDRAM and QDRII SRAM

FPGA configuration: MAX®II CPLD and 16-bit page mode flash memory, Joint Test Action Group (JTAG) interface

User and board-specific interfaces : Push-button switches, User Dual-in-line Package (DIP) switch, User LEDs (Light Emitting Diodes), Board-specific DIP switch, Board-specific LEDs

Power supply : PCIe motherboard or Laptop-style DC power supply via DC input jack

Communication ports: PCIe edge connector, High-speed Mezzanine cards (HMC), Gigabit Ethernet, Small Form-factor Pluggable (SFP) modules, JTAG header

Clocking circuitry: Three high-speed clock oscillators of 100 MHz, 155.52 MHz, 156.25 MHz and SMA connector for external clock input and output

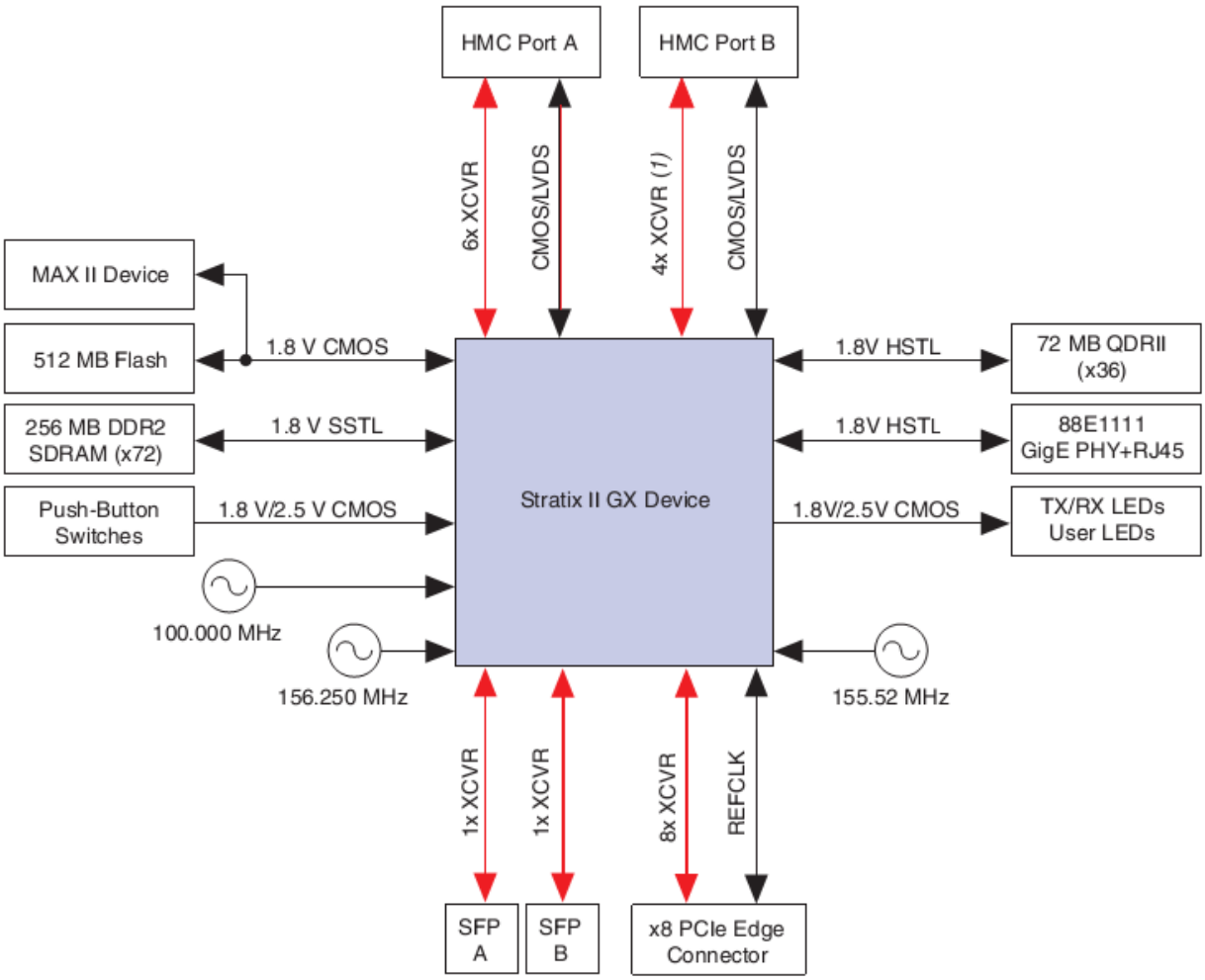


Figure 3.1.1 Stratix II GX PCI Express Development Board Block Diagram[95]

The board's default device, EP2SGX90FF1508C3 Stratix II GX , provides the following:

16 transceiver channels, 59 source-synchronous channels, 90,960 logic elements (LEs), 8 phase-locked loops (PLLs), 650 user I/O, 4,520,448 RAM bits and 192 18x18 multipliers

Figure 3.1.2 gives the photograph of Top View of the Stratix II GX PCIe Development Board.

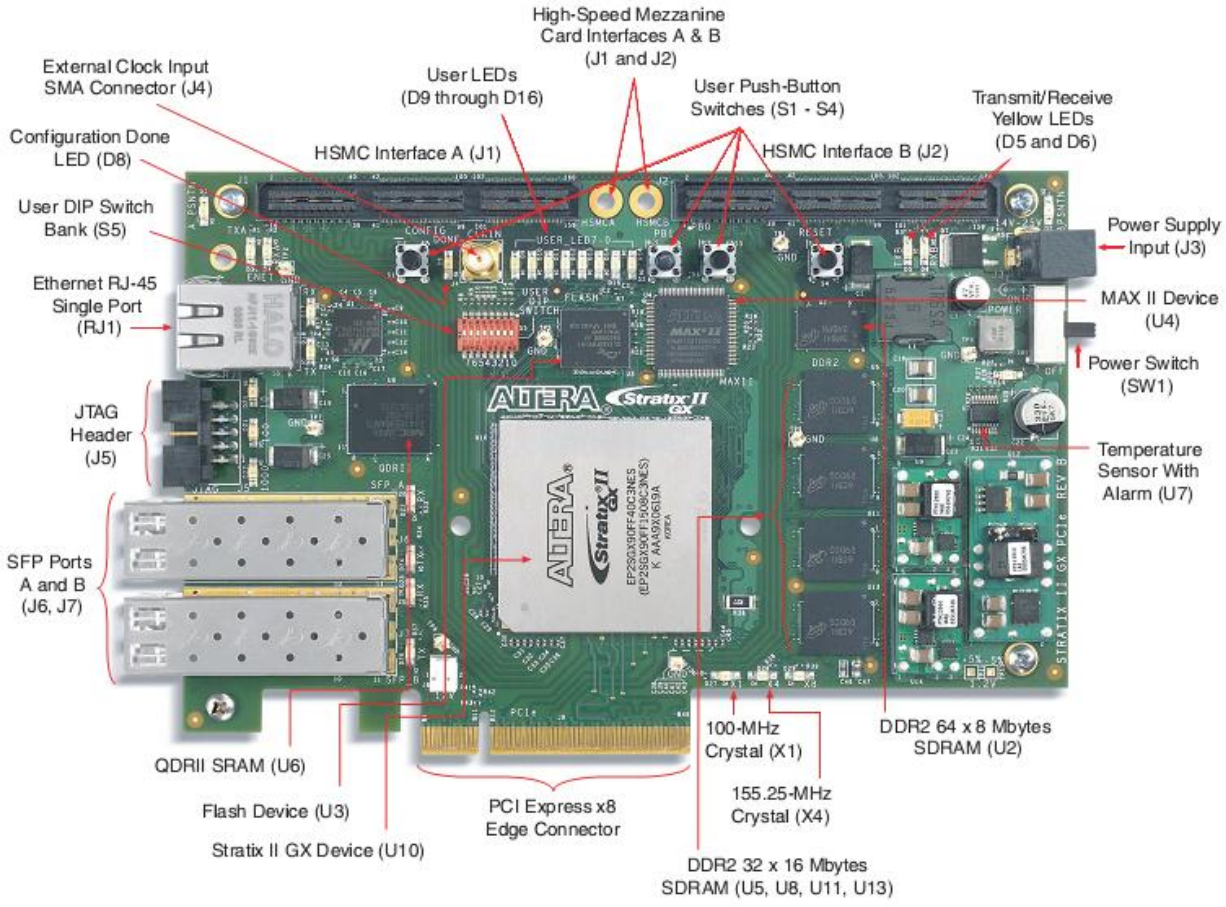


Figure 3.1.2 Photograph of Top View of the Stratix II GX PCIe Development Board[95]

3.2 System Organisation

The Gigabit Ethernet protocol design is built using the System On Programmable Chip (SOPC) builder, which is a tool used along with Quartus II software. The SOPC builder enables the user to create a system based on the Nios II processor, by selecting the desired functional units and specifying their parameters. Altera's Triple Speed Ethernet Design has been implemented and used as a platform for studying the performance of Gigabit Ethernet Standards 1000Base-LX, 1000Base-SX and 1000Base-T. Figure 3.2.1 shows a high-level block diagram of the TSE design [96].

The design includes two Altera TSE MegaCore functions , which implement the MAC, PCS and PMA layers and is downloaded on Altera's Stratix II GX PCI Express Development Kit. There are two SFP cages built onto the kit. This design interfaces the TSE MegaCore function [97] with a Copper or Optical Fibre SFP module via a 1.25 Gbps serial transceiver that enables all 10, 100, and 1000 Mbps Ethernet operations. The design sends stream of Ethernet packets to the TSE MegaCore function, which can be looped back using SFP modules with an Ethernet fibre optic cable, copper cable or a switch. The design operates in various modes with live traffic upto the maximum throughput rate and shows the error rate in the receiver. The Nios II processor is used as a control plane component for setting up and configuring the system components.

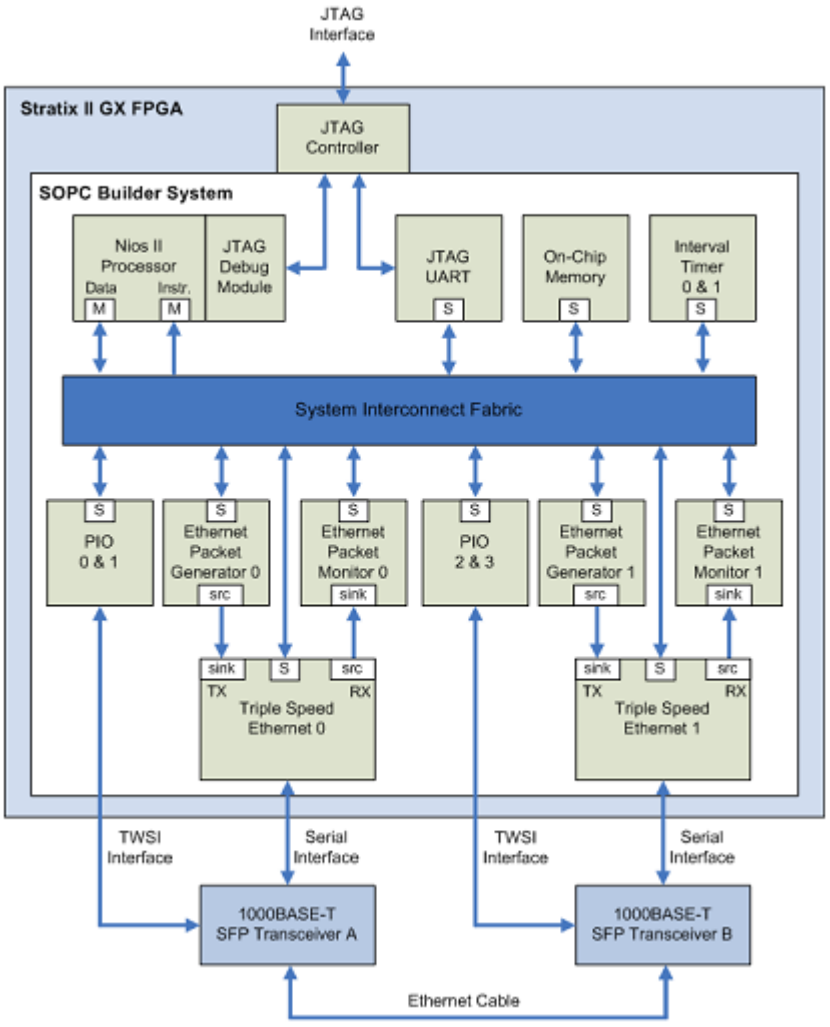


Figure 3.2.1 Block Diagram of Triple Speed Ethernet Design [96]

Components used in the design:

Nios II processor

The Nios II processor is used as a control plane for setting up and configuring the system components. The Ethernet packets are generated and monitored by the processor. The parameter settings for Nios II processor is shown in Figure 3.2.2.

On-chip memory

On-chip memory is of block size 256 Kbytes, which is used as system RAM for storage of software code.

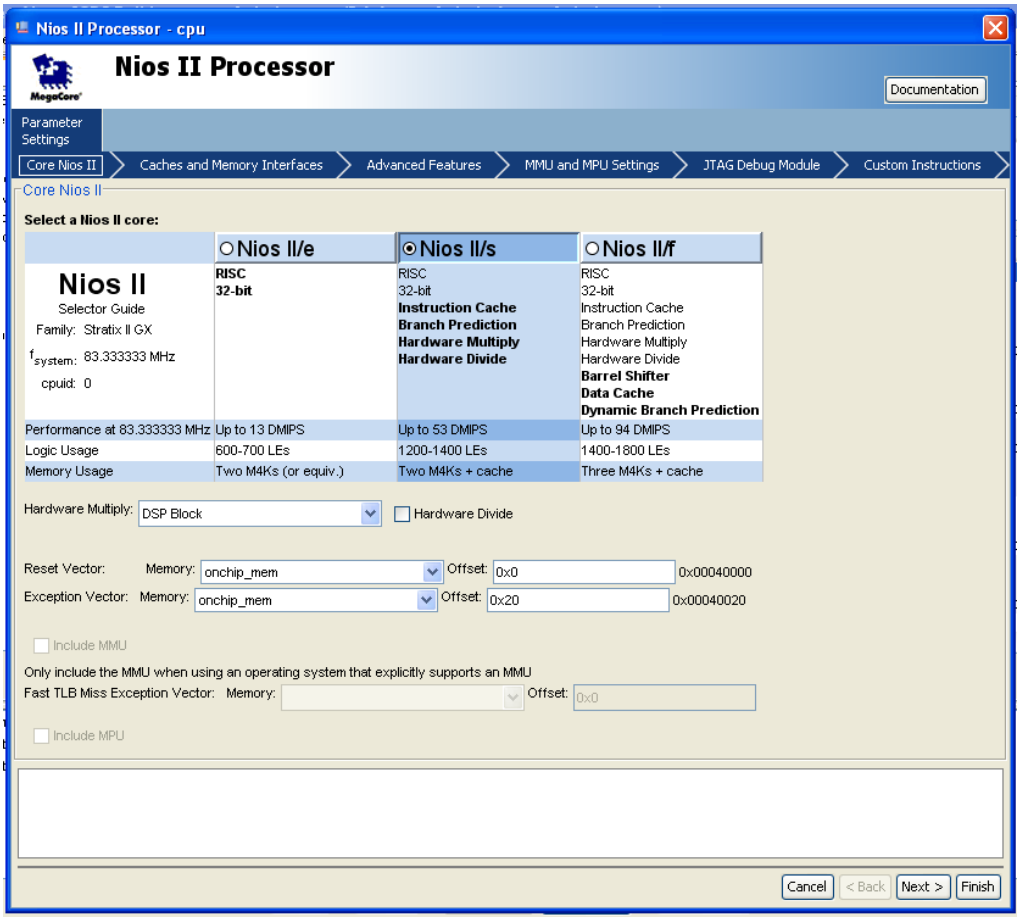


Figure 3.2.2 Parameter settings for Nios II processor

Parallel Input/Output

The parallel input/output (PIO) core provides easy I/O access to the 1000BASE-T Copper or Optical Fibre SFP module's PHY registers. A "bit banging" approach that conforms to the TWSI (Two Wire Serial Interface) protocol is used. There are four pios , out of which pio_0 and pio_2 are defined as output ports y with 1 bit width. pio_1 and pio_3 are configured as bidirectional ports with 1 bit width. They are set to synchronously capture data at rising edge of clock.

Similarly pio_1 and pio_3 inputs are hardwired in test bench. Reset vector for all the pios is set as 0x0.

JTAG UART

The JTAG UART core sends serial character streams linking a Nios II processor and the SOPC Builder system. The Nios II processor interfaces with the core by reading and writing control and data registers. A Write FIFO of 64 bytes buffer depth is defined for JTAG UART, which transfers data from Avalon to JTAG. Similarly a Read FIFO of buffer depth 64 bytes is used to transfer data from JTAG to Avalon. IRQ for both is set as 8.

Phase-locked loop

The phase-locked loop (PLL) core generates an 83.33.MHz PLL output from an input clock of 100 MHz crystal on the development board. Multiplication factor of 5 and division factor of 6 is used to generate a system clock of 83.33 MHz with 50% duty cycle. This PLL output clock of 83.33 MHZ is used as a system -wide clock source for the SOPC Builder system.

Triple Speed Ethernet MegaCore Function

The TSE Megacore function transmits the Ethernet packets from the Avalon Streaming (Avalon-ST) interface to a 1.25 Gbps serial transceiver interface which is built in the Stratix II GX device and receives packets from the opposite direction. The parameter setting for core configuration is set to 10/100/1000Mb Ethernet MAC with 1000BASE-X / SGMII(Serial GMII) PCS is shown in Figure 3.2.3. The MAC options of loopback logic and statistics counters are enabled. Receive and transmit FIFO are 32 bit width and 2048 bit depth. The PCS / SGMII options to enable SGMII bridge logic and export transceiver powerdown signal are selected.

Interval Timer

The interval timer core is a 32-bit timer of 1ms timeout period. This is used by the Nios II processor system to calculate the performance and throughput rate of the different Ethernet operations.

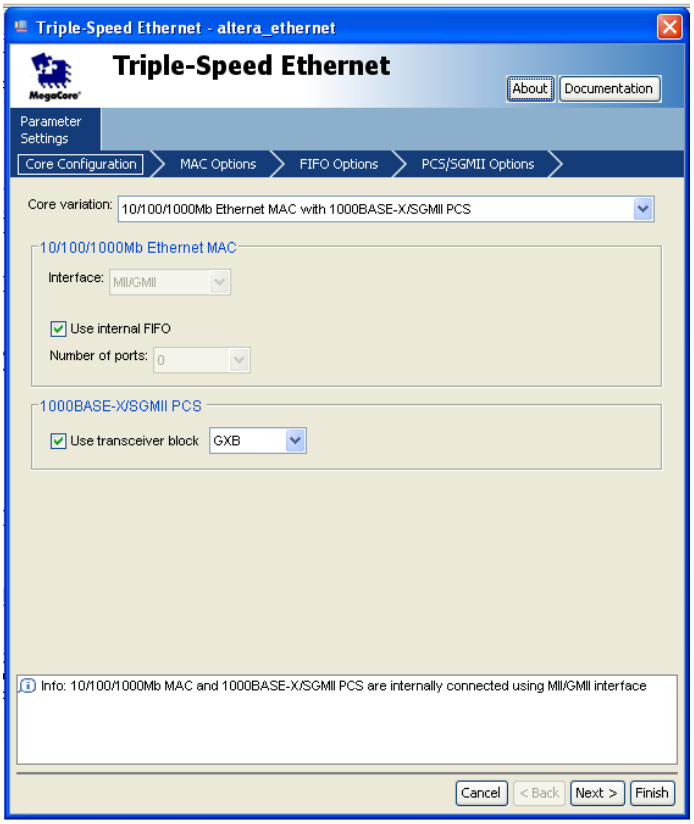


Figure 3.2.3 Parameter settings of Triple Speed Ethernet Megacore function

Ethernet Packet Generator

The Ethernet Packet Generator is created using component editor and generates a stream of Ethernet packets and drives the TSE Mega Core function Transmit FIFO interface. It uses Avalon-ST(Streaming) source interface for sending Ethernet packets and Avalon Memory-Mapped (Avalon-MM) slave interface for control purposes.

Ethernet Packet Monitor

The Ethernet Packet Monitor is created using component editor and receives a stream of Ethernet packets from the TSE MegaCore function Receive FIFO interface. It uses an Avalon-MM slave interface for control purposes and an Avalon-ST sink interface for the data path. The Ethernet Packet Monitor also verifies the accuracy of the received packet.

Table 3.2.1 Memory Map of the TSE design

Address	Description
0x00040000 – 0x0007FFFF	On-Chip Memory - RAM (256 KB)
0x00080800 – 0x00080FFF	CPU JTAG Debug Module
0x00081000 – 0x000813FF	Triple-Speed Ethernet 0
0x00081400 – 0x000817FF	Triple-Speed Ethernet 1
0x00081800 – 0x0008183F	Ethernet Packet Generator 0
0x00081840 – 0x0008187F	Ethernet Packet Generator 1
0x00081880 – 0x0008189F	Phased-Lock Loop
0x000818A0 – 0x000818BF	Ethernet Packet Monitor 0
0x000818C0 – 0x000818DF	Ethernet Packet Monitor 1
0x000818E0 – 0x000818FF	Interval Timer 0
0x00081900 – 0x0008191F	Interval Timer 1
0x00081920 – 0x0008192F	PIO 0
0x00081930 – 0x0008193F	PIO 1
0x00081940 – 0x0008194F	PIO 2
0x00081950 – 0x0008195F	PIO 3
0x00081960 – 0x00081967	JTAG UART

Table 3.2.1 gives the memory map of the TSE design and Figure 3.2.4 displays the list of all the components used in the SOPC builder.

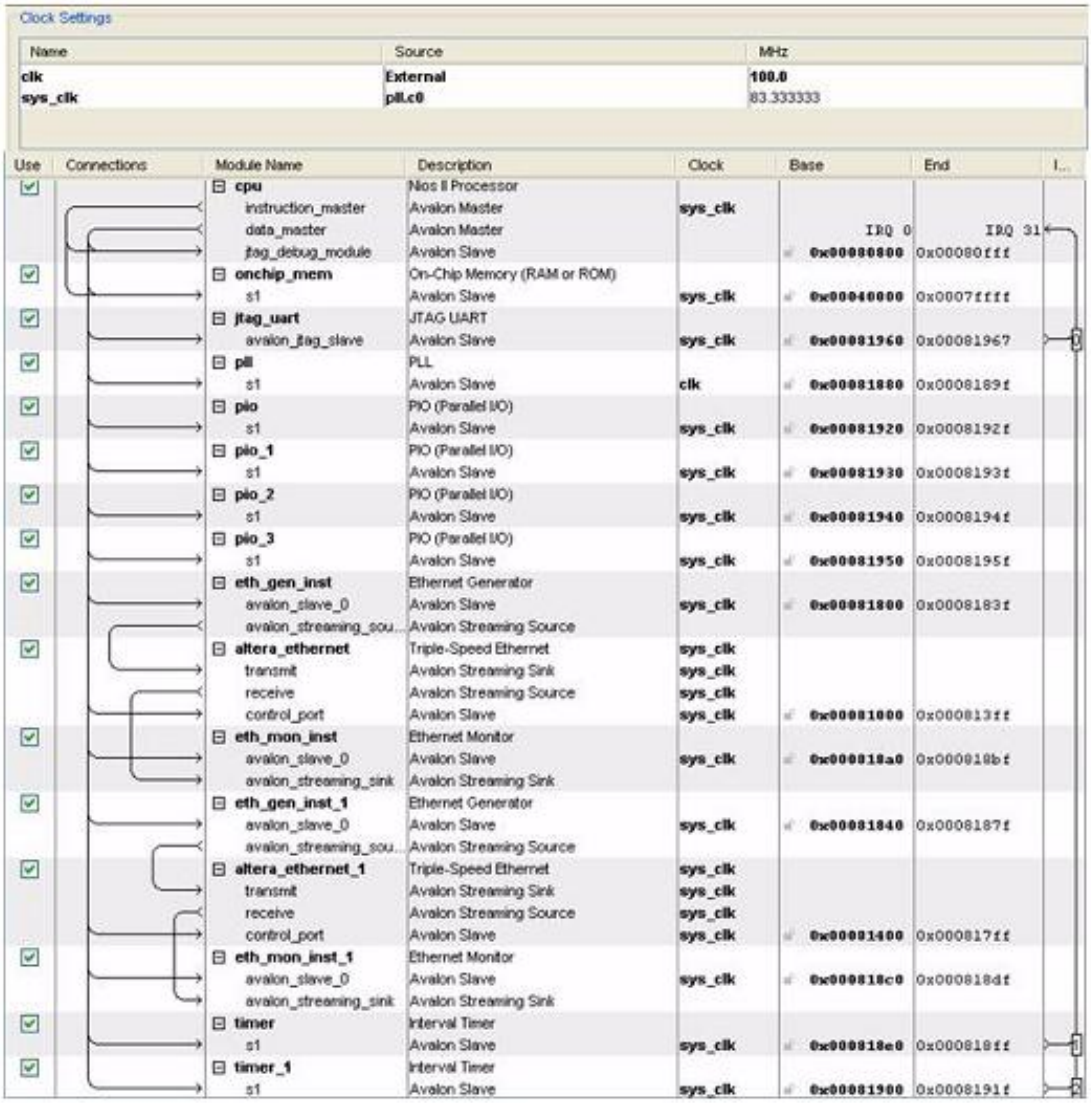


Figure 3.2.4 Components used in SOPC builder setup for the TSE design

3.3 FPGA Implementation

3.3.1 Programmable 10/100/1000Mbps Ethernet operation

Figure 3.3.1 illustrates the programmable 10/100/1000Mbps Ethernet operation. The 10/100/1000 Ethernet PHY devices implement a common interface that you connect to a 10/100-Mbps MAC through MII/RGMII (Reduced Gigabit Media Independent Interface) or to a gigabit MAC via GMII/RGMII [97]. On the receive path, the clock provided by the PHY device (2.5 MHz, 25 MHz or 125 MHz) is connected to the MAC clock, rx_clk. The PHY interface is connected to both the MII and GMII of the MAC function.

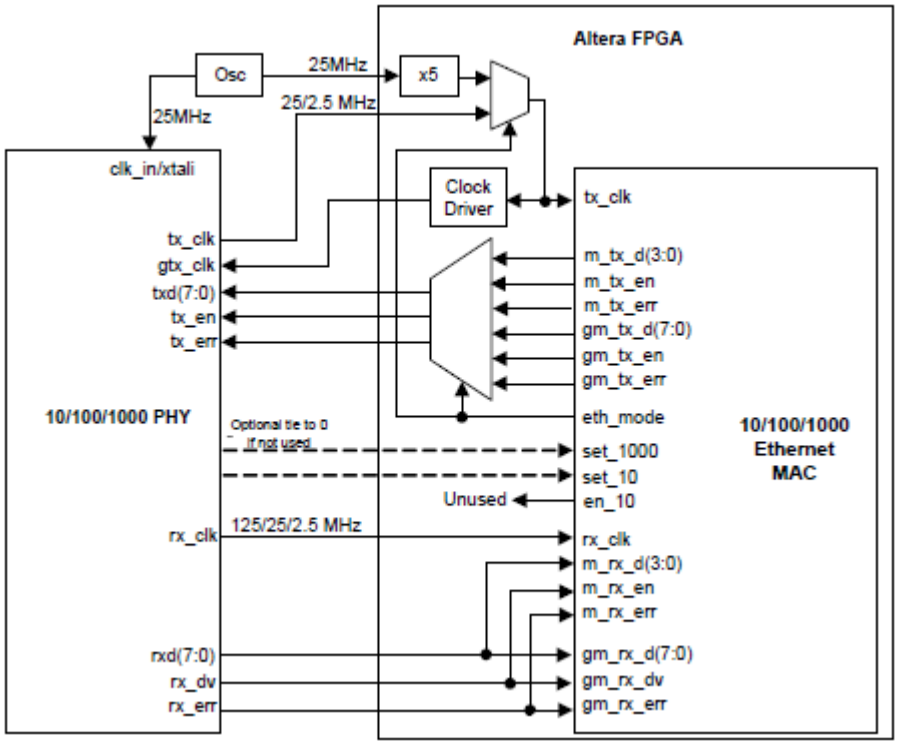


Figure 3.3.1 Programmable 10/100/1000Mbps Ethernet operation [97]

During the transmit path, standard programmable PHY devices which operate in 10/100 mode generate a 2.5 MHz or a 25 MHz clock. In gigabit mode, the PHY device requires a 125 MHz clock from the MAC function. Here, an external clock module is introduced to drive the 125 MHz clock to the MAC function and PHY devices, since the MAC function does not generate

a clock output. The clock generated by the MAC to the PHY can be tri-stated in 10/100 mode. During transmission, the MAC control signal eth_mode is used to select MII or GMII. The MAC function asserts the eth_mode signal when the MAC operates in gigabit mode, which drives the MAC GMII to the PHY interface. When the MAC function operates in 10/100 mode, the eth_mode signal is deasserted. In this mode, the MAC MII is driven to the PHY interface.

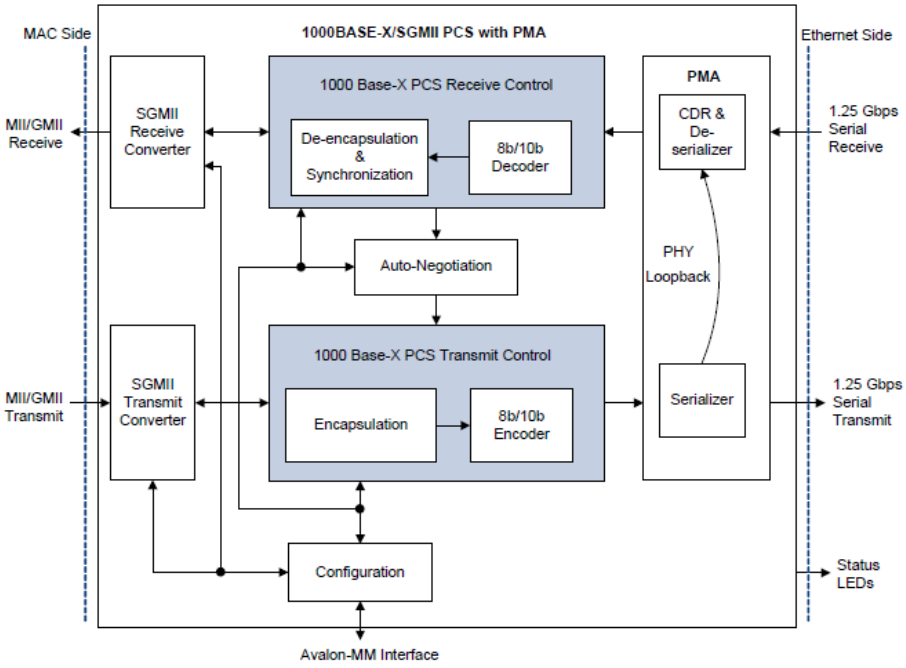


Figure 3.3.2 Block diagram of the PCS function with an embedded PMA [97]

The 1000BASE-X/SGMII PCS function is accessible with GMII (1000BASE-X/SGMII) or MII (SGMII). Figure 3.3.2 gives the Block Diagram of the PCS function with an embedded PMA. An on- or off-chip SERDES component is interfaced to the PCS function via the industry standard Ten-Bit Interface (TBI). The PCS function can be configured with an embedded PMA. This configuration complies with the IEEE 802.3 Standard 1000BASE-X PMA specification [96]. PMA interoperates with an external PMD device, which drives the external copper or optical fibre

network. The interconnect between Altera functions and PMD devices can be TBI or 1.25 Gbps serial.

3.3.2 Transmit Operation

The transmit operation includes frame encapsulation and encoding.

Frame Encapsulation

The PCS function replaces the first preamble byte in the MAC frame with the start of frame /S/ symbol. Then, the PCS function encodes the rest of the bytes in the MAC frame with standard 8B/10B encoded characters. After the last FCS byte, the PCS function inserts the end of frame sequence, /T/ /R/ /R/ or /T/ /R/, depending on the number of character transmitted. Between frames, the PCS function transmits /I/ symbols. If the PCS function receives a frame from the MAC function with an error (gm_tx_err asserted during frame transmission), the PCS function encodes the error by inserting a /V/ character.

The first byte of the preamble byte in the MAC frame is replaced with the start of frame /S/ symbol by the PCS function. The remaining bytes in the MAC frame is encoded with standard 8B/10B encoded characters. After the FCS byte, the end of frame sequence, /T/ /R/ /R/ or /T/ /R/, is inserted depending on the number of characters transmitted. The PCS function transmits /I/ symbols between frames. If an error is detected in the frame received , a /V/ character is inserted.

8b/10b Encoding

The 8B/10B encoder maps 8-bit words to 10-bit symbols.

3.3.3 Receive Operation

The receive operation includes comma detection, decoding, de-encapsulation, synchronization, and carrier sense.

Comma Detection

The comma detection function looks for the 10-bit encoded comma character, K28.1/K28.5/K28.7, in successive samples received from PMA devices. When the required code group is detected, the PCS function realigns the data stream on a valid 10-bit character boundary. A standard 8b/10b decoder is used to decode the aligned stream. The comma detection function resumes with the search for a valid comma character if the receive synchronization state machine loses the link synchronization.

8b/10b Decoding

The 8b/10b decoder does the disparity checking to ensure DC balancing and produces a decoded 8-bit stream of data, which is fed to the frame de-encapsulation function.

Frame De-encapsulation

The frame de-encapsulation state machine finds the start of frame when the /I/ /S/ sequence is received and the /S/ is replaced with a preamble byte (0x55). It decodes the frame bytes and transmits them to the MAC function. The /T/ /R/ /R/ or the /T/ /R/ sequence is decoded as the end of frame. A /V/ character is decoded as frame error and sent to the MAC function. Sequences other than /I/ /I/ (Idle) or /I/ /S/ (Start of Frame) are decoded as wrong carrier. When the de-encapsulation state machine detects invalid characters, during frame detection, it indicates an error to the MAC function.

Synchronization

The link synchronization constantly monitors the decoded data stream and determines if the underlying receive channel is ready for operation. The link synchronization state machine acquires link synchronization if the state machine receives three code groups with comma consecutively without error. When link synchronization is acquired, the link synchronization state machine counts the number of invalid characters received. The state machine increments an internal error counter for each invalid character received and incorrectly positioned comma character. The internal error counter is decremented when four consecutive valid characters are received. When the counter reaches 4, the link synchronization is lost. The PCS function

drives the led_link signal to 1 when link synchronization is acquired. This signal can be used as a common visual activity check using a board LED.

The link synchronization state machine continuously monitors the decoded data stream and acquires link synchronization if the state machine receives three code groups with comma consecutively without error. There is an internal error counter, which is incremented when invalid characters are received or there is an incorrectly positioned comma character. The link synchronization is lost, when the counter reaches a value of 4. When four consecutive valid characters are received, the internal error counter is decremented. The PCS function drives the led_link signal to 1, indicating that link synchronization is acquired and it can be used on a board LED.

Carrier Sense

The carrier sense state machine detects an activity after the link synchronization is acquired and the transmit and receive encapsulation or de-encapsulation state machines are not in the idle or error states. The mii_rx_crs and led_crs signals are driven to 1 when it detects an activity. The led_crs signal can be connected to a board LED, which can be used as an indicator.

3.3.4 Collision Detection

A collision occurs if non-idle frames are received and transmitted to the PHY simultaneously. Collisions can be detected only in SGMII and half-duplex mode. When a collision occurs, the mii_rx_col and led_col signals are driven to 1. The led_col signal can be used for visual check using a board LED.

3.3.5 SGMII Converter

When the SGMII is enabled, the SGMII converter is automatically configured with the capabilities advertised by the PHY. In 1000BASE-X mode, the PCS function always operates in

gigabit mode. The SGMII is a substitute interface to the GMII/MII that converts the parallel interface of the GMII into a serial format. It drastically reduces the I/O count.

Auto-Negotiation

Auto-negotiation is an optional function that can be started when link synchronization is acquired during system start up. During auto-negotiation, the PCS function advertises its device features and exchanges them with a link partner device.

Ten-bit Interface

In PCS variations with embedded PMA, the PCS function implements a TBI to an external SERDES. During transmission and reception, the least significant bit of the TBI should be serialized first.

PHY Loopback

Loopback can be enabled on the serial interface to test the PCS and embedded PMA functions in isolation of the PMD.

PHY Power-Down

Power-down is controlled by the POWERDOWN bit in the PCS control register. In this state, the PCS function is in reset and activities on the GMII transmit and the TBI receive interfaces are ignored. However, the management interface is active. In PCS variations with embedded PMA interfaced with GX transceivers, the power-down signal is connected to the power-down of the GX transceiver internally.

Reset

A hardware reset resets all the logic synchronized to the corresponding clock domains whereas a software reset only resets the PCS state machines, comma detection function, 8B10B encoder and decoder. Hardware reset is triggered by asserting the respective reset signals:

reg_clk, tx_clk, and rx_clk. To trigger a software reset, the RESET bit in the control register is set to 1.

3.3.6 Interfaces on the Stratix II GX development board

Gigabit Ethernet Interface

The board's Gigabit Ethernet interface is implemented with an RJ-45 jack and a 10/100/1000 base-T, auto-negotiating Ethernet physical device.

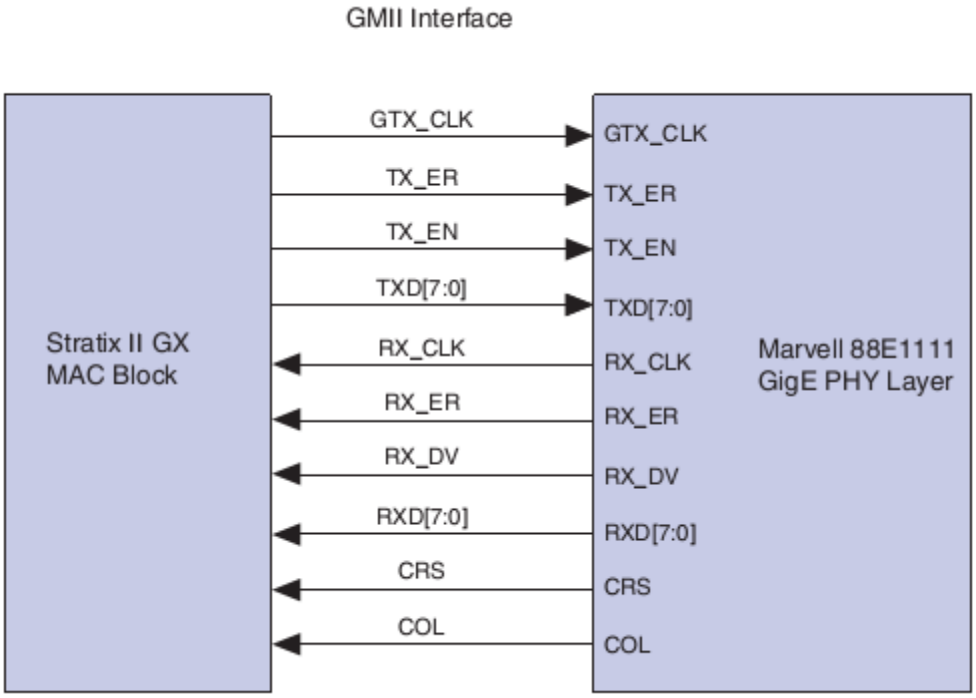


Figure 3.3.3 Marvell 88E1111 Gigabit Ethernet PHY Layer and GMII Interface to the FPGA [95]

The GMII interface shown in Figure 3.3.3 is single-data-rate (SDR) and source-synchronous in nature operating at 125 MHz, whereas, the reduced gigabit media independent interface (RGMII) uses half of the eight data pins, also operating at 125 MHz. The RGMII interface achieves 50% pin count reduction by using DDR flip flops. The Stratix II GX PCIe

development board can use either the GMII or RGMII interface. However, the GMII interface is preferred because of its simple timimg model.

SFP A and B Interfaces

There are two SFP standard cages SFP_A and SFP_B, connected to the Stratix II GX device's transceivers and project through the PCIe panel. These two interfaces are designed as per the SFP MSA specifications and support FDDI, Fiber Channel, ATM, and Gigabit Ethernet (both copper and optical). The SFP MSA requires signals only upto 5.0 Gb/s, but standard modules available today are usually 2.488 Gb/s synchronous optical net (SONET) mode or below.

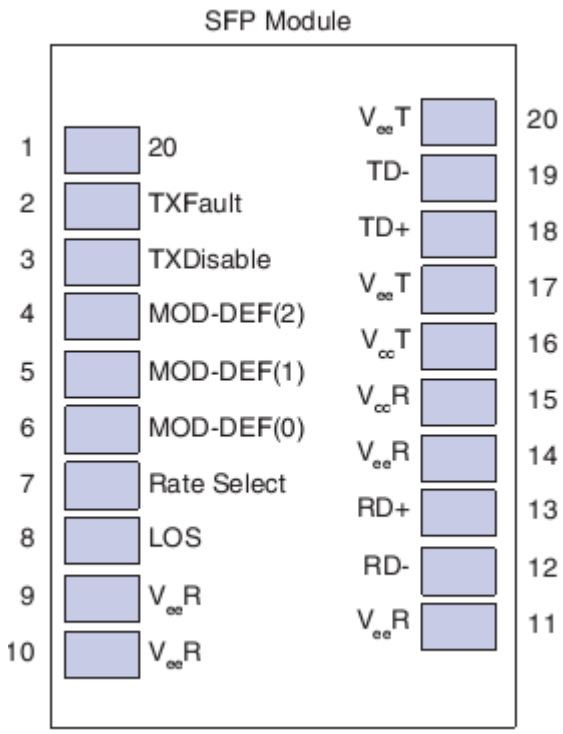


Figure 3.3.4 SFP Module Interface[95]

JTAG Interface

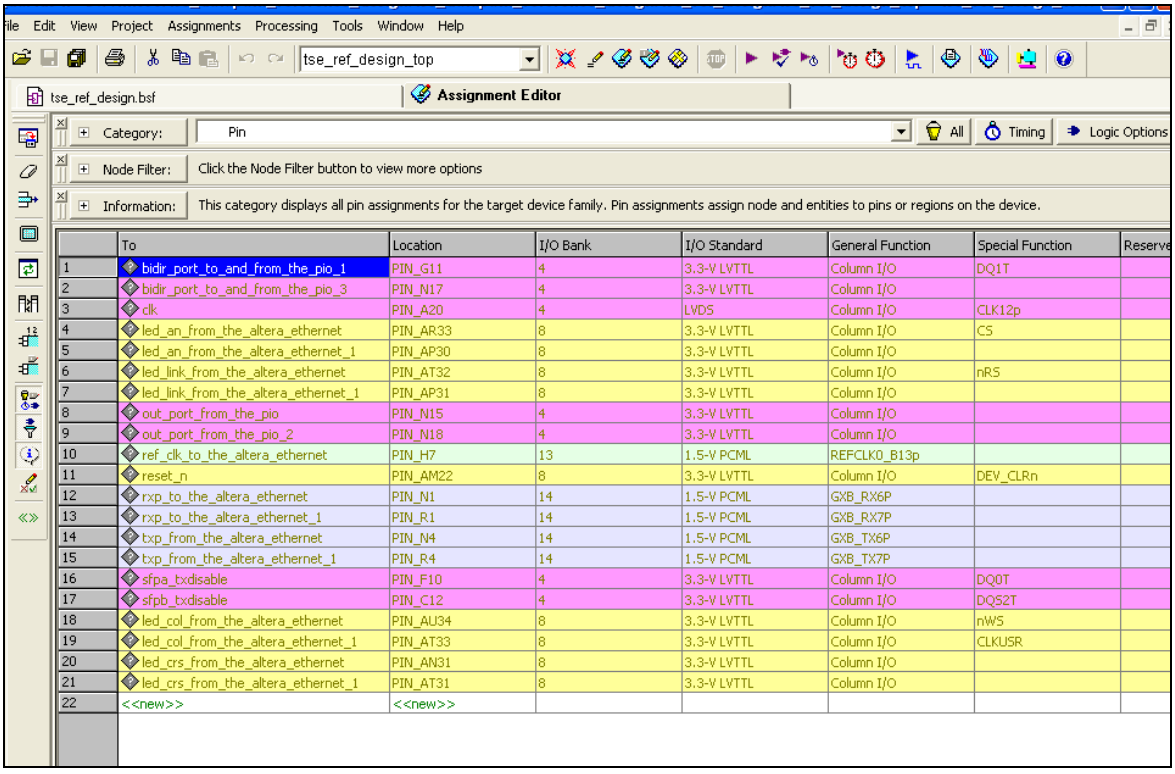
The board provides a right-angle, 10-pin JTAG header. The JTAG header is used for JTAG-based FPGA programming as well as communication to a standard computer using a USB-Blaster download cable. It achieves a speed of approximately 1 Mb/s using an SOPC Builder-based Nios II system in the FPGA.

3.3.7 Compilation Report

Figure 3.3.5 Block Symbol file of TSE design

The TSE design is built using the SOPC builder as explained in Section 3.2. The SOPC design is saved and compiled and then we go to Quartus II where the Block Symbol file as shown in Figure 3.3.5 is generated. The input, output and bidirectional pins are mapped to the

Stratix II GX FPGA pins as shown in Figure 3.3.6. Table 3.3.1, 3.3.2, 3.3.3 and 3.3.4 give the description of all the signals such as Clock and Reset, Triple-Speed Ethernet 0 and Triple-Speed Ethernet 1 signals, and SFP Interface signals. The pin assignments used in the design are also listed in the Tables.



	To	Location	I/O Bank	I/O Standard	General Function	Special Function	Reserve
1	bidir_port_to_and_from_the_pio_1	PIN_G11	4	3.3-V LVTTL	Column I/O	DQ1T	
2	bidir_port_to_and_from_the_pio_3	PIN_N17	4	3.3-V LVTTL	Column I/O		
3	clk	PIN_A20	4	LVDS	Column I/O	CLK12p	
4	led_an_from_the_altera_ethernet	PIN_AR33	8	3.3-V LVTTL	Column I/O	CS	
5	led_an_from_the_altera_ethernet_1	PIN_AP30	8	3.3-V LVTTL	Column I/O		
6	led_link_from_the_altera_ethernet	PIN_AT32	8	3.3-V LVTTL	Column I/O	nRS	
7	led_link_from_the_altera_ethernet_1	PIN_AP31	8	3.3-V LVTTL	Column I/O		
8	out_port_from_the_pio	PIN_N15	4	3.3-V LVTTL	Column I/O		
9	out_port_from_the_pio_2	PIN_N18	4	3.3-V LVTTL	Column I/O		
10	ref_clk_to_the_altera_ethernet	PIN_H7	13	1.5-V PCML	REFCLK0_B13p		
11	reset_n	PIN_AM22	8	3.3-V LVTTL	Column I/O	DEV_CLRn	
12	rxp_to_the_altera_ethernet	PIN_N1	14	1.5-V PCML	GXB_RX6P		
13	rxp_to_the_altera_ethernet_1	PIN_R1	14	1.5-V PCML	GXB_RX7P		
14	txp_from_the_altera_ethernet	PIN_N4	14	1.5-V PCML	GXB_TX6P		
15	txp_from_the_altera_ethernet_1	PIN_R4	14	1.5-V PCML	GXB_TX7P		
16	sfpa_txdisable	PIN_F10	4	3.3-V LVTTL	Column I/O	DQ0T	
17	sfpb_txdisable	PIN_C12	4	3.3-V LVTTL	Column I/O	DQS2T	
18	led_col_from_the_altera_ethernet	PIN_AU34	8	3.3-V LVTTL	Column I/O	nWS	
19	led_col_from_the_altera_ethernet_1	PIN_AT33	8	3.3-V LVTTL	Column I/O	CLKUSR	
20	led_crs_from_the_altera_ethernet	PIN_AN31	8	3.3-V LVTTL	Column I/O		
21	led_crs_from_the_altera_ethernet_1	PIN_AT31	8	3.3-V LVTTL	Column I/O		
22	<<new>>	<<new>>					

Figure 3.3.6 Pin Assignment for TSE design

Table 3.3.1 Clock and Reset Signals

Signal Name	Direction	Description	Pin Location
clk	In	Reference design clock. The clock is sourced from the 100 MHz Oscillator (X1)	A20
ref_clk_to_the_altera_ethernet	In	TSE transceiver reference clock. The clock is sourced from the 156.25 MHz Oscillator (X3).	H7

reset_n	In	Single reset for all logic, connected to the RESET (S4) push button.	AM22

Table 3.3.2 Triple-Speed Ethernet 0 Signals

Signal Name	Direction	Description	Pin Location
rxp_to_the_altera_ethernet	In	Serial Differential Receive Interface. It is connected to the sfpa_rx_p0 signal of SFP A (J6)	N1
txp_from_the_altera_ethernet	Out	Serial Differential Transmit Interface. It is connected to the sfpa_tx_p0 signal of SFP A (J6)	N4
led_an_from_the_altera_ethernet	Out	Auto-negotiation status. It is connected to USER_LED0 (D16)	AR33
led_link_from_the_altera_ethernet	Out	Link synchronization status. It is connected to USER_LED2 (D14)	AT32
led_col_from_the_altera_ethernet	Out	Collision detection of frame transmission. It is connected to USER_LED4 (D12)	AU34
led_crs_from_the_altera_ethernet	Out	Carrier sense on transmit and receive path. It is connected to USER_LED6 (D10).	AN31

Table 3.3.3 Triple-Speed Ethernet 1 Signals

Signal Name	Direction	Description	Pin Location
rxp_to_the_altera_ethernet_1	In	Serial Differential Receive Interface. It is connected to the sfpb_rx_p0 signal of SFP B (J7).	R1
txp_from_the_altera_ethernet_1	Out	Serial Differential Transmit Interface. It is connected to the sfpb_tx_p0 signal of SFP B (J7)	R4
led_an_from_the_altera_ethernet_1	Out	Auto-negotiation status. It is connected to USER_LED1 (D15)	AP30
led_link_from_the_altera_ethernet_1	Out	Link synchronization status. It is connected to USER_LED3 (D13)	AP31
led_col_from_the_altera_ethernet_1	Out	Collision detection of frame transmission. It is connected to USER_LED5 (D11).	AT33
led_crs_from_the_altera_ethernet_1	Out	Carrier sense on transmit and receive path. It is connected to USER_LED7 (D9)	AT31

Table 3.3.4 SFP Interface Signals

Signal Name	Direction	Description	Pin Location
out_port_from_the_pio	Out	SFP A Serial Clock Line. It is connected to the sfpa_mod1_scl signal of SFP A (J6).	N15
bidir_port_to_and_from_the_pio_1	In/Out	SFP A Serial Data Line. It is connected to the sfpa_mod2_sda signal of SFP A (J6)	G11
out_port_from_the_pio_2	Out	SFP B Serial Clock Line. It is connected to the sfpb_mod1_scl signal of SFP B (J7)	N18
bidir_port_to_and_from_the_pio_3	In/Out	SFP B Serial Data Line. It is connected to the sfpb_mod2_sda signal of SFP B (J7)	N17
sfpa_txdisable	Out	SFP A Transmitter Disable. It is connected to the sfpa_txdisable signal of SFP A (J6)	F10
sfpb_txdisable	Out	SFP B Transmitter Disable. It is connected to the sfpb_ txdisable signal of SFP B (J7).	C12

The Ethernet Packet Generator and Monitor Register map, config_setting Register Bit Descriptions, operation Register Bit Descriptions, Receive Control and Status Register Bit Descriptions are given in Appendix I. .

After the assignment of the pins, the design is compiled in Quartus II. Figure 3.3.7 gives a summary of the compilation report. The Resource utilization of the design is listed in Table 3.3.5.

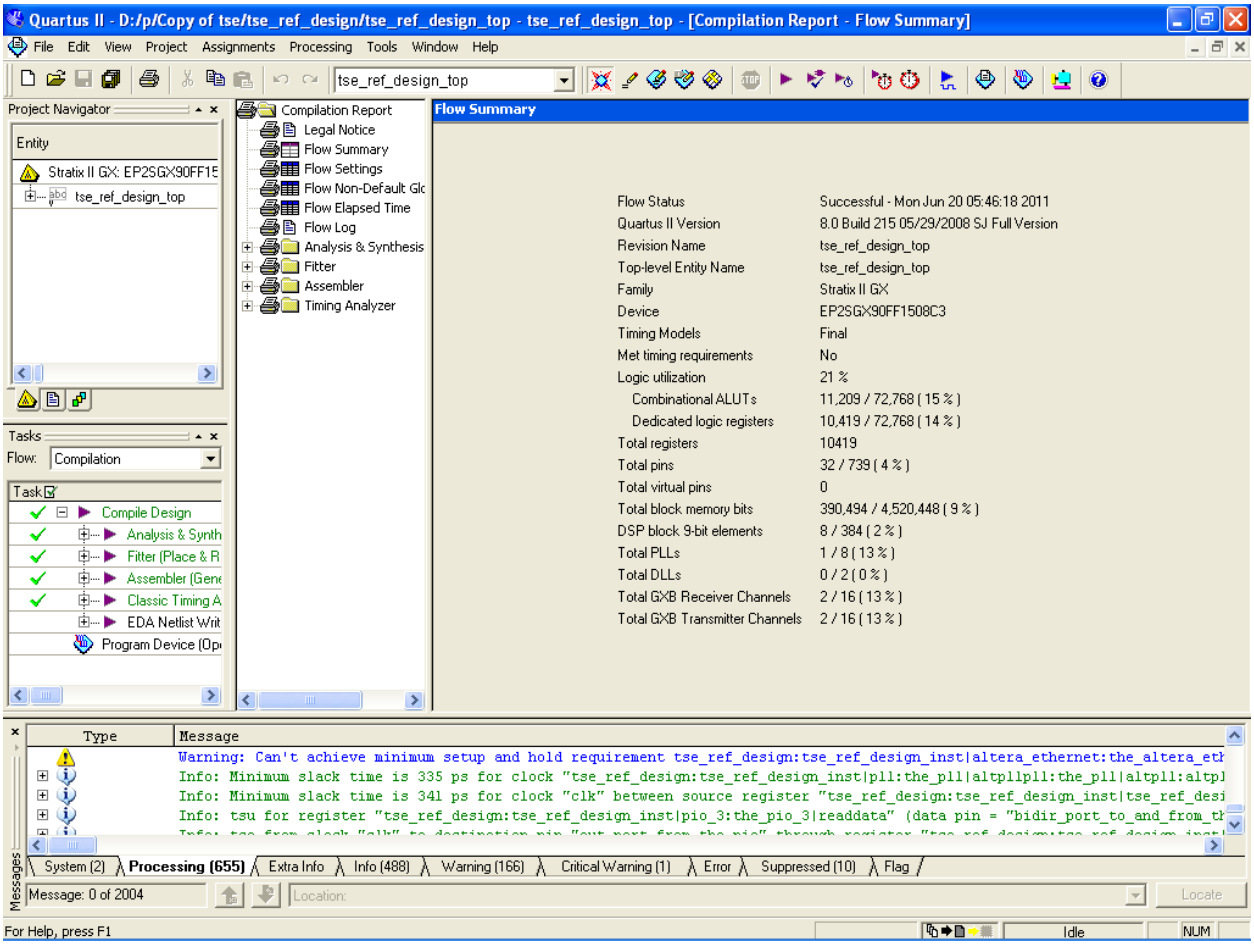


Figure 3.3.7 Compilation Report Summary

Table 3.3.5 Utilization of Resources of the TSE design

Parameter	**Value**	**Utilization (%)**
Logic	15281	21
Combinational ALUTs	11209	15
Dedicated logic registers	10419	14
Total pins	32	4
Total block memory bits	390494	9
DSP block 9-bit elements	8	2
Total PLLs	1	13
Total GXB Receiver Channels	2	13
Total GXB Transmitter Channels	2	13

3.4 Command Line Interface

Altera provides API code written in C language, which is compiled in the Nios II IDE along with the Hardware design. There are 4 menus, which you can use to parameterize and run tests on the hardware.

Following is the description of the different menus.

3.4.1 Test Menu

The Test menu is used to configure and run a specific test in the system .

Link Speed: It specifies the speed of the Ethernet link. It can be set to 10, 100, or 1000 Mbps.

Link Configuration: It specifies the transmission mode for the link. The mode can be set to either full or half duplex.

Packet Length: It specifies the length of the packets sent over the link. The length can be set to either a random or fixed size. The packet length can range from minimum 24 bytes to maximum 1518/9600 bytes for random/ fixed size.

Number of Packets: It specifies the number of packets to be send for the test. The number is a 32-bit, unsigned integer. The valid range is from 1 to 2^{32} packets.

Packet Data: It specifies the type of data sent in the payload of the packet. The type can be either increment or random.

Seed Value: It appears only when the Packet Data menu option is set to random. This is any 46-bit hexadecimal number

Test Control: It is used to start a test. After starting, the test runs until the number of packets sent is equal to the value specified in the Number of Packets parameter or until you manually interrupt the test.

3.4.2 TSE MAC Menu

The TSE MAC menu is used to configure the MAC address of both the sender and receiver TSE MACs using a 12 digit hexadecimal number.

3.4.3 Report Menu

The Report menu displays the status of the tests that are run on the system.

The Report menu provides the following information:

Number: An integer that corresponds to the number of the completed test.

Status: It indicates the status of the test, whether completed, interrupted, or error.

TSE Sender MAC: It displays the MAC address of the TSE MAC sender.

TSE Receiver MAC: It displays the MAC address of the TSE MAC receiver.

Link Speed: It displays the Ethernet link speed along with the transmission mode .

Packets to Send: It displays the number of packets to send, the packet length and the data type used. If the data type is random, the seed value is displayed.

Packets Sent: It displays the total number of packets actually sent during the test by the Ethernet Packet Generator. The throughput in packets per second is also displayed.

Packets Received: It displays the total number of packets received during the test by the Ethernet Packet Monitor. The number of valid packets and error packets are displayed.

Bytes Received: It displays the total number of bytes received by the Ethernet Packet Monitor. The byte throughput rate is displayed in terms of bits/second.

Report Control: It controls the selection of the report to be viewed. You can use next, previous controls or enter a report number to be viewed.

3.4.4 Console Menu

The Console menu provides a console-based interface to the system. It provides the following menu items:

Console: It controls the launching of the console application.

Console Operation: The console operates in a command-line mode, with commands terminated by pressing Enter.

Table 3.4.1 gives a list of the various console commands.

Table 3.4.1 List of Console commands and their purpose

Console Command	Purpose
speed	link speed
duplex	link duplex
plen	packet length
pnum	number of packets
pdata	packet data
seed	seed value
start	start test
macaddr_send	sender MAC Address
macaddr_recv	receiver MAC Address
report	displays report
reportn	queues report
oreg	read/write memory location with offset
reg	read/write memory location
test	list test parameters
map	display memory map
?	command information
x	exit console

3.5 Experimental Setup

The Testbed Setup for the Gigabit Ethernet protocol performance analysis is given below in Figure 3.5.1. The Gigabit Ethernet Testbed, designed using Altera's TSE Megacore function is downloaded onto the Stratix II GX FPGA using Quartus II 8.0 software and JTAG interface. The performance studies are done by varying parameters such as data rate, mode of transmission, frame length and number of frames using the command line interface. The reports such as transmission status, transmission time, throughput and number of frame errors can be obtained from the design. The loopback is obtained by using physical media such as Copper or optic fibre cable.

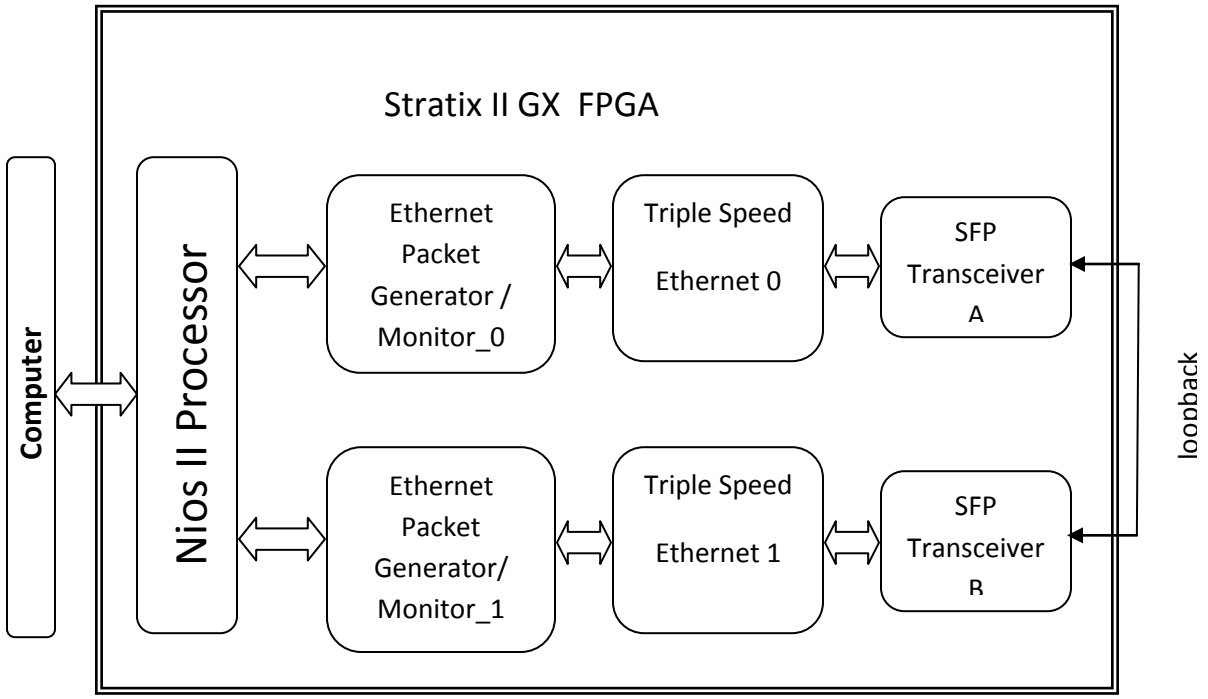


Figure 3.5.1 Testbed setup for Performance Analysis

Figure 3.5.2 shows the Experimental Setup for introduction of errors and Error detection. It consists of three main components.

1. Stratix II GX FPGA: where the design is downloaded

2. SFP Cages on the Stratix II GX FPGA: for interfacing different types of cables such as Cat5e, Cat6, Single-mode Fibre and Multi-mode Fibre using SFP Tranceivers

3. 3 axis Fibre Mounting and Positioning Stand: Platform for mounting the fibre and introducing optical attenuation

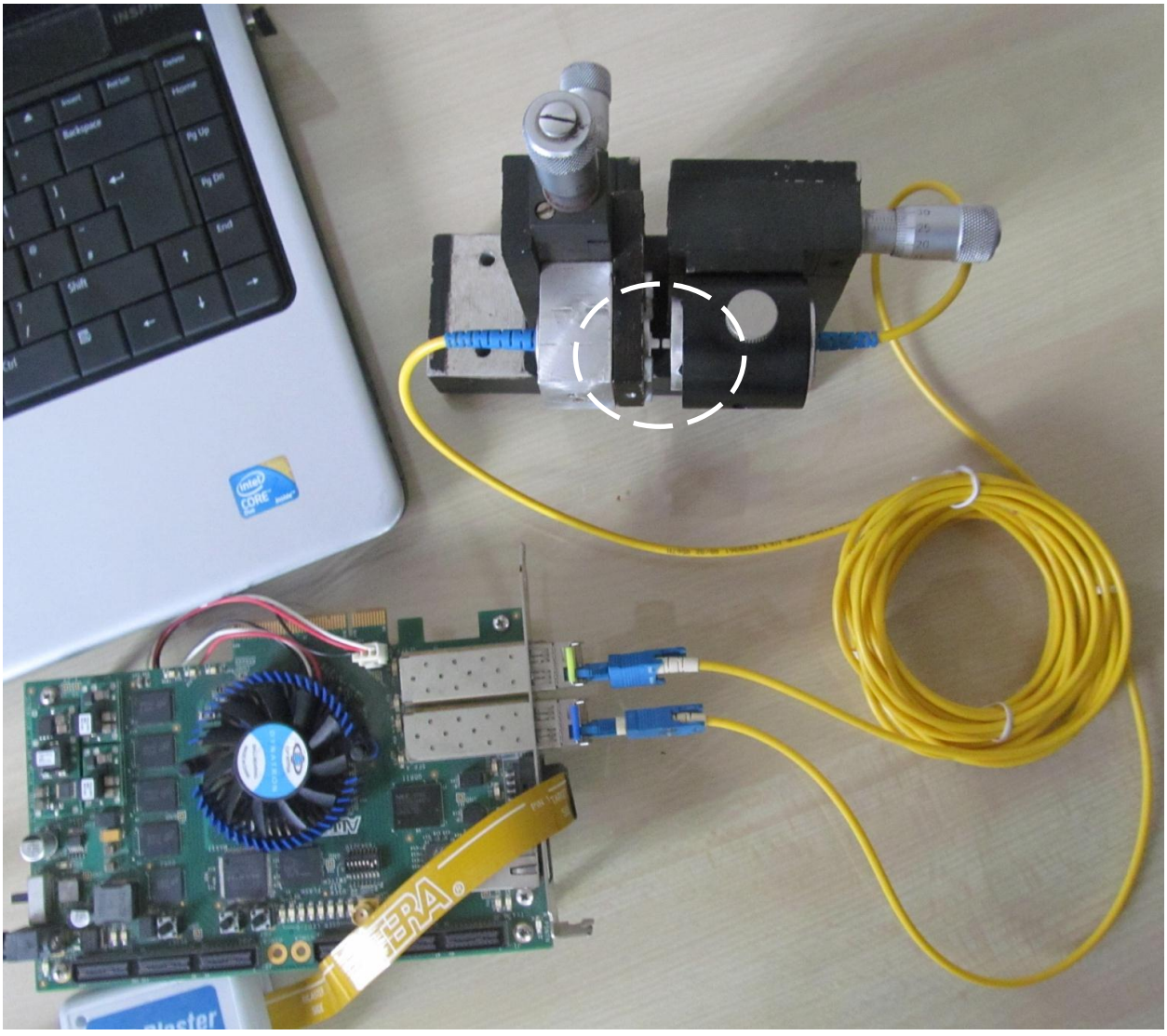


Figure 3.5.2 Setup for introducing errors in the Ethernet frames and Error Detection

This setup uses two LC to SC Single Mode Duplex optic fibre patch cables, each of 2 meters length. The SC connectorized ends of the cable are mounted and aligned together on the Fibre Mounting and positioning stand. A 5mW, 670nm wavelength laser source is used to launch optical power from the LC connectorized end of one of the cables. The power output from the LC end of the second cable is first optimized for maximum radiance by using the fine adjustment screws, which is fine tuned to achieve maximum power coupling.

Then the optical source is removed and the LC ends of the fibres are connected to the Stratix II GX FPGA board by using SFP Transceivers for bidirectional Single Mode Fibre and the cable is connected in loopback mode. The Ethernet frames of desired length and number are generated using the command line interface from the computer. Later errors were introduced in the fibre by lateral displacement using the adjustment screw along the x axis of the Mounting Stand. Figures 3.5.3 and 3.5.4 illustrate the packet loss introduced in the fibre with lateral displacement.

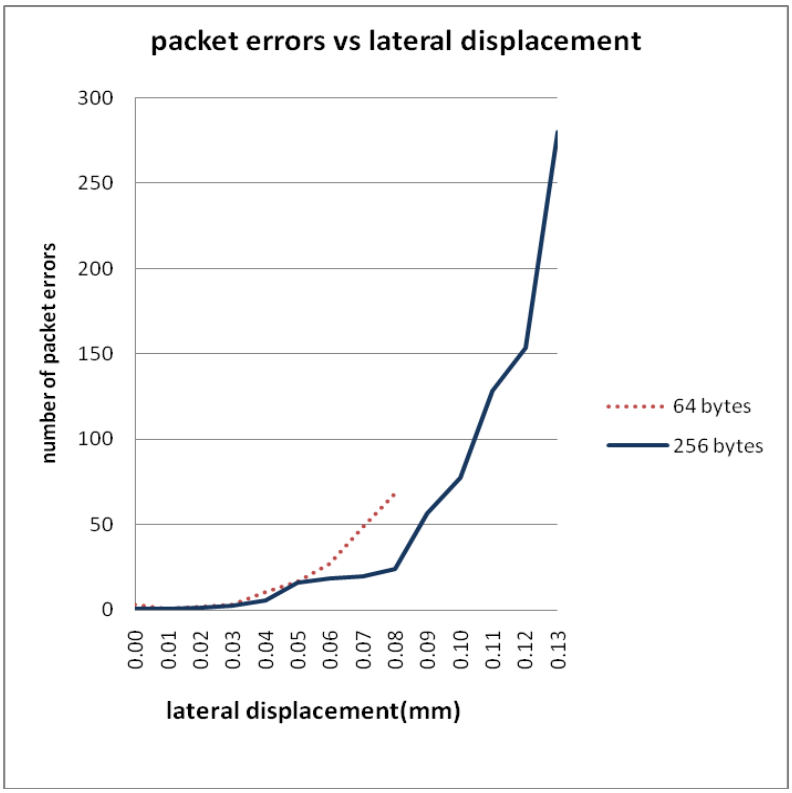


Figure 3.5.3 Number of packet errors vs lateral displacement for 1 million packets

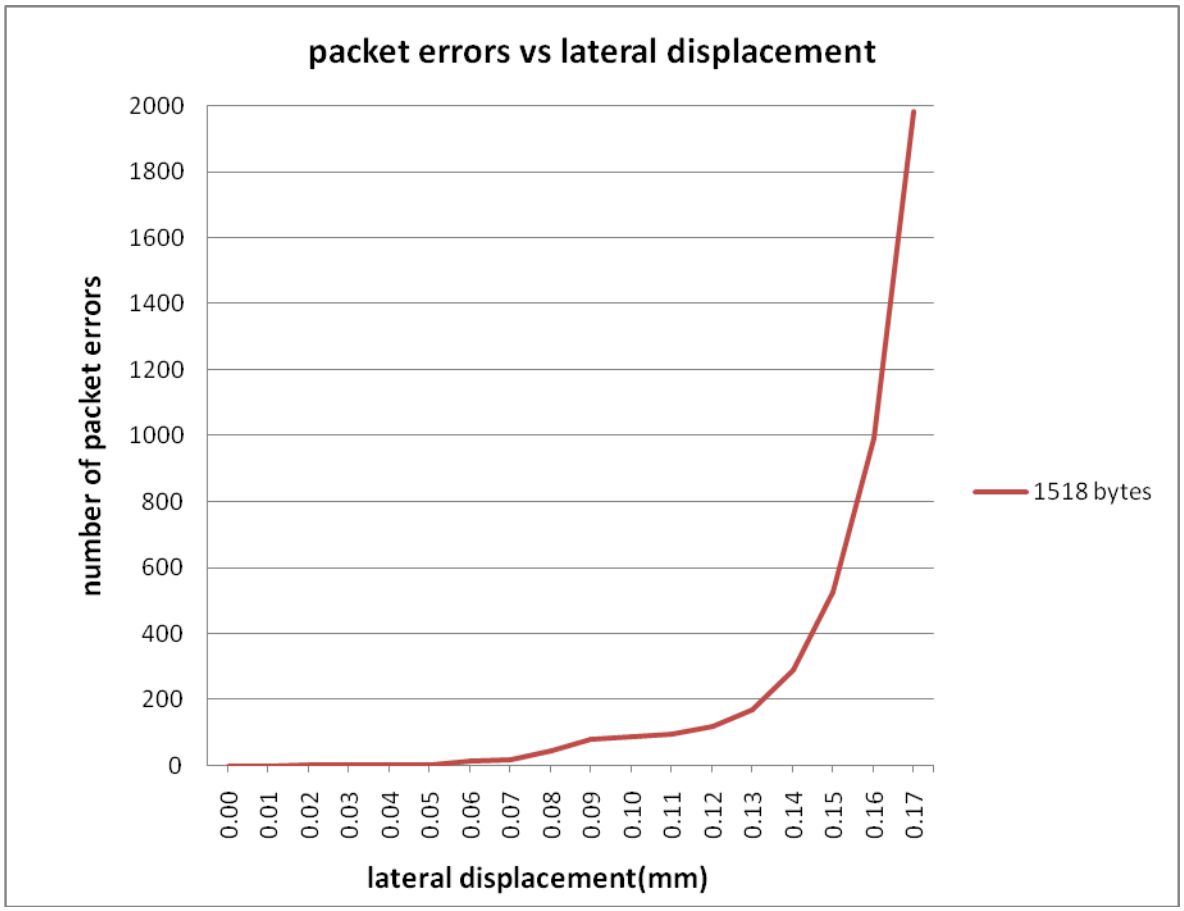


Figure 3.5.4 Number of packet errors vs lateral displacement for 1 million packets (1518 bytes)

This study of introduction of errors is carried out for Ethernet frames of three lengths namely 64 bytes, 256 bytes and 1518 bytes. The number of packets used for every test is 1000000.The test is repeated 10 times for each position and the average of the packet errors is taken. From Figure 3.5.3, it is seen that number of packet errors increases exponentially with increase in lateral displacement. For 64 bytes frame length, there is 100% packet loss at distance of 0.09 mm and the link is lost. Similarly for 256 bytes length frame, the number of packet errors is 280 at a distance of 0.13 mm and then link is lost at 0.14 mm. Figure 3.5.4 is plotted for a frame length of 1518 bytes and shows that the link can be sustained till a distance of 0.17 mm with a packet loss of 1982. Thus the developed platform is robust and gives repeatable and reliable results and can be used for experimental studies.

CHAPTER 4

MODELING AND PERFORMANCE ANALYSIS

In a digital transmission system, an error occurs when a bit is alters in between transmission and reception in a channel. A binary 1 is transmitted and a binary 0 is received or a binary 0 is transmitted and a binary 1 is received. There are two types of errors that can occur: single-bit errors and burst errors [98]. A single-bit error is an isolated error that alters one bit but does not affect adjacent bits. A burst error is an error in a contiguous sequence of B bits in which the first and last bits and any number of intermediate bits are received in error.

A single-bit error may occur in the presence of white noise, due to a slight random deterioration of the signal-to-noise ratio, which is sufficient to confuse the receiver's decision of a single bit. Burst errors can be caused by impulse noise or fading in a mobile wireless environment and are more common and more difficult to deal with. The effects of burst errors are larger at higher data rates.

4.1 Errors in a Digital Communication System

Bit errors occur in digital communication systems due to intrinsic or extrinsic factors [99]. Intrinsic errors are due to the components, design and implementation of a link. They are caused due to internal noise sources, poor electrical connections, and sometimes receiver sampling error. In optical links the errors occur mainly because of the physical components such as optical driver, optical receiver, connectors, optical fiber, etc. Errors are also caused due to optical attenuation and optical dispersion. One of the prevalent causes of random or noise-induced errors is the optical receiver. The extrinsic sources of error are caused by external sources and influences. These include power supply noise, electrostatic discharge, electromagnetic interference and connector vibrations.

4.1.1 Shannon Capacity Formula

Nyquist's formula $C = 2B\log_2 M$ where M is the number of discrete signal or voltage levels, indicates that, doubling the band-width doubles the data rate, all other factors being equal [98]. Higher the data rate, more are the errors. These concepts can be explained using a

formula developed by the mathematician Claude Shannon. For a given level of noise, a greater signal strength would improve the ability to receive data correctly in the presence of noise.

SNR is the most common performance measure characteristic of a digital communication system. Typically, this ratio is measured at the output of the receiver and is thus directly related to the data detection process [100]. Signal-to-noise ratio is a term for the ratio of the power in a signal to the power present in the noise:

Thus, SNR in decibels can be expressed as

$$SNR(dB) = 10 \cdot \log\left(\frac{P_s}{P_n}\right) = 20 \cdot \log\left(\frac{A_s}{A_n}\right) \quad (4.1)$$

where P_s is average signal power and P_n is average noise power, and A is root mean square (RMS) amplitude for signal and noise. SNRs are usually expressed in terms of the logarithmic decibel scale, since many signals have a very wide dynamic range. Thus the SNR is 10 times the base-10 logarithm of the power ratio and if the signal and noise is measured across the same impedance then SNR can be obtained by calculating 20 times the base-10 logarithm of the amplitude ratio.

Shannon's result is given by $C = B\log_2(1+SNR)$, where C is the capacity of the channel in bits per second and B is the bandwidth. Although Shannon formula represents the theoretical maximum that can be achieved, in reality, much lower rates are achieved because the formula assumes only white noise and other sources of noise such as impulse noise, attenuation distortion or delay distortion is not accounted. Even in an ideal white noise environment, the present technology has not been able to achieve Shannon capacity due to encoding issues, such as coding length and complexity. The Shannon's formula provides a yardstick for the measurement of performance of practical communication schemes.

From the formula, it can be observed that the data rate could be increased by increasing either signal strength or bandwidth. However, as the signal strength increases, the effects of nonlinearities in the system also increases leading to an increase in intermodulation noise. Similarly, since the noise is assumed to be white, the wider the bandwidth, the more noise is admitted in the system. Thus, as B increases, SNR decreases.

The parameter E_b/N_0 is the ratio of signal energy per bit to noise power density per Hertz. It is a standard quality measure for digital communication system performance. The ratio E_b/N_0 is important because the bit error rate for digital data is a function of this ratio.

4.1.2 Bit Error Rate

BER is the most fundamental measure of system performance. It is defined as the number of bits received in error, divided by the total number of bits received [100]. It is usually expressed as 10^{-x}. For example, a transmission system might have a BER of 10^{-6}, meaning that on an average, 1 out of every of 1000000 bits transmitted exhibits an error. The BER performance is affected by factors such as signal-to-noise ratio and distortion. However, the quality of the link is ultimately defined by the ability to receive error-free information. The BER indicates of how often a packet or other data unit needs to be retransmitted because of an error.

4.1.3 Packet Error Rate

The PER is the number of incorrectly received data packets divided by the total number of received packets. A packet is declared incorrect even if one bit is erroneous.

4.1.4 Binary Symmetric Channel

The BSC is a binary channel, which can transmit only one of two symbols ether 0 or 1 [101]. It is one of the simplest noisy channels to analyze. The probability for having an error is denoted as p and the probability for no errors is 1 - p. It is known as symmetric because the probability of error is independent of the transmitted symbol. Figure 4.1.1 gives the graphical representation of BSC, where X is the input variable and Y is the output variable.

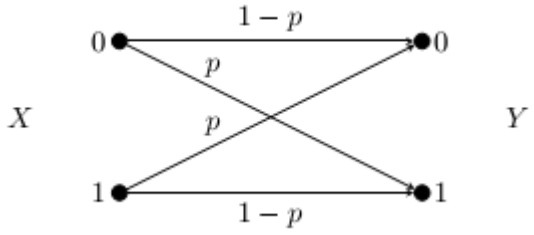


Figure 4.1.1 Graphical representation of BSC

4.1.5 Additive White Gaussian Noise Channel

AWGN is the commonly used and generally accepted model for thermal noise in communication channels [100]. It can be represented as in Figure 4.1.2 where s(t) is transmitted signal and n(t) is background noise and r(t) = s(t) + n(t), is the received signal.

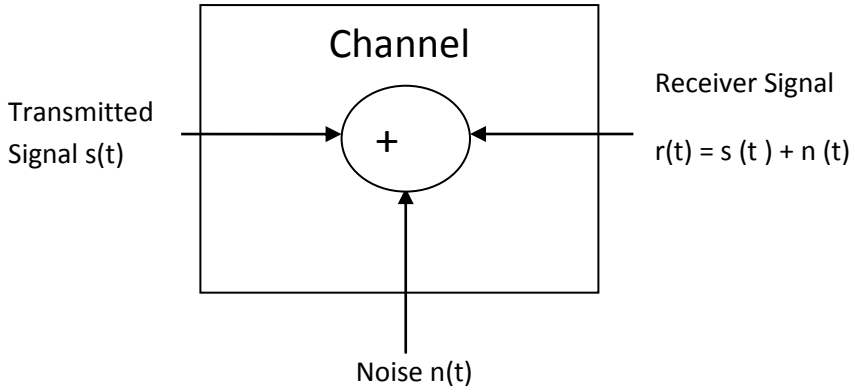


Figure 4.1.2 AWGN channel model

4.1.6 BER of BPSK (Binary Phase Shift Keying)

Phase-shift keying (PSK) is a technique of digital communication in which the phase of a transmitted signal is varied to convey information. There are several ways to accomplish PSK. BPSK is the simplest form of PSK. It uses two phases which are separated by 180° and hence also termed as 2-PSK [102]. This modulation is the most robust of all the PSKs since it requires serious distortion to make the demodulator reach an incorrect decision.

The BER of BPSK in AWGN can be calculated as:

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \text{ or } P_b = 1/2 erfc\left(\sqrt{\frac{E_b}{N_o}}\right) \quad (4.2)$$

4.2 Error Detection using CRC

CRCs have a long history of use for error detection in computing [103]. Error correction codes provide a means to detect and correct errors introduced by a transmission channel. Two main categories of code exist: block codes and convolutional codes. Both the codes introduce redundancy by adding parity symbols to the message data. Cyclic codes are an important class of linear (n,k) block codes 'C' which are said to be cyclic if for each of its code vectors c=(C_0, C_1, C_2, …….. C_{n-1}) in 'C' the i^{th} right cyclic shift c'= (C_{n-i}, C_{n-i+1}, … C_0 ,C_1, C_{n-i-1}) is a code vector also in 'C'. CRC codes are a subset of cyclic codes which are also a subset of linear block codes. CRC provides error control coding and protection to data by introducing some redundancy in the data in a controlled manner. It is a very effective way of detecting transmission errors during transmissions in various types of networks and it is also used in storage applications. Common CRC polynomials are able to detect following types of errors: i. All single bit error, ii. All double bit errors, iii. All odd number of errors having sufficient constraint length, iv. Any burst error for which the burst length is less than the polynomial length, v. Most large burst errors [104].

CRC uses binary alphabets, 0 and 1. Arithmetic is based on GF(2) i.e. modulo-2 addition (logical XOR) and multiplication (logical AND). The coding scheme uses systematic codes. Suppose m(x) is the message polynomial, c(x) the code word polynomial and g(x) the generator polynomial. We have c(x)= m(x)g(x) which is also written using the systematic form as c(x)=

$m(x)x^{n-k} + r(x)$, where r(x) is the remainder of the division of $m(x)x^{n-k}$ by g(x) and r(x) represents the CRC bits. The transmitted message c(x) contains k-information bits followed by 'n-k' CRC bits, for example $c(x) = m_{k-1}x^{n-1}+\ldots\ldots+ m_0x^{n-k}+ r_{n-k-1}x^{n-k-1}+\ldots\ldots+ r_0$. So encoding is straightforward: multiply m(x) by x^{n-k} , that is , append 'n-k' bits to the message, calculate the CRC bits by dividing $m(x)x^{n-k}$ by g(x), and append the resulting 'n-k' CRC bits to the message. The same algorithm can be used for decoding. If c'(x) is the received message, then no error or undetectable errors have occurred if c'(x) is multiple of g(x) , which is equivalent to determining that if $c'(x)x^{n-k}$ is a multiple of g(x), that is, if the remainder of the division from $c'(x)x^{n-k}$ by g(x) is 0[14]**.** An error is declared to have occurred if there is any discrepancy between these two sets of parity bits and thus indicates the presence of transmission errors in the received data.

However, as with all digital signature schemes, there is a small, but finite, probability that a data corruption that inverts a sufficient number of bits in just the right pattern will occur and lead to an undetectable error [105]. The minimum number of bit inversions required to achieve such undetected errors, known as Hamming Distance (HD) is a central issue in the design of CRC polynomials. However, there may be transmission errors residing in the received frame that are not detected by this procedure. Whenever channel noise affects both the information block ***i*** and the block *r* of parity bits in such a way that the received frame $\hat{c}=[\acute{r}, \hat{\imath}]$ satisfies $\acute{r}(x) = (x^p . \hat{\imath} (x))$ mod g(x), the errors cannot be detected by parity checking. Authors have proposed solution for the said problem using double CRC in the payload of the Ethernet frame [106].

There are four major CRC implementation solutions. Many researchers have done comparative studies over CRC implementation [107]. CRC implementation can use either hardware or software solutions. In the traditional hardware implementation, a simple shift register circuit performs the computations by handling the data one bit at a time and parallel implementation by handling data in one word(n-bit) at a time [108,109,37,41]. Software implementations of CRC encoding/ decoding do not resort to dedicated hardware requirements; their applicability, however, is limited to lower encoding rates. In traditional hardware implementations, a LFSR performs the computations by handling the data one bit at a time. With the increasing demands of high transmission rate requirement in various

applications, the need of a simple and systematic method to rapidly calculate the CRC has resulted. The most frequently used CRC digest computation algorithm in iSCSI and Ethernet, the Simultaneous Multiply and Divide (SMD), is derived from the long division algorithm. Sample implementations are provided of both algorithms in [110,111].

4.2.1 Software Solution

The CRC algorithm can always be implemented as a software algorithm on a standard CPU, with all the flexibility of reprogramming. Also, the software solution will be much cheaper, since a CPU is present in most communication network terminals. However, the computation speed will be lower, since no general purpose CPU can achieve the same throughput as dedicated hardware. Kounavis et al. implemented novel lookup table based algorithm based on 'slicing-by-x' algorithm which doubles the performance of existing software-based, table driven CRC implementations [112]. Software implementation, which requires many steps of the polynomial division, is typically slower than other codes. Nguyen et al. implemented fast CRCs as well as an effective technique to implement them without using lookup table, to eliminate or to greatly reduce many steps of the polynomial division [113].

The CRC algorithm is straightforward to implement in software. However, it requires considerable CPU bandwidth to implement the basic requirements, such as shift, bit test and XOR. Moreover, CRC calculation is an iterative process and additional software overhead for data transfer instructions puts enormous burden on the MIPS requirement of a microcontroller. The CRC engine in 'dsPIC33E/PIC24E' devices calculates the CRC checksum without CPU intervention and it is much faster than the software implementation. The CRC engine consumes only half of an instruction cycle per bit for its calculation as the frequency of the CRC shift clock is twice that of the dsPIC33E/PIC24E instruction clock cycle. For example, the CRC hardware engine takes only 64 instruction cycles to calculate a CRC checksum on a message that is 128 bits (16x8) long. If the same calculation is implemented in software, it will consume more than a thousand instruction cycles, even for an optimized piece of code.

4.2.2 Traditional Hardware Solution

LFSR with serial data feed has been used since the sixties to implement the CRC algorithm. Like all hardware implementations, this method simply performs a division and then the remainder which is the resulting CRC checksum, is stored in the registers after each clock cycle. The registers can then be read with the help of enabling signals. The main advantages are simplicity and low power dissipation. This method gives much higher throughput than the Software solution however, the speed requirements of today's network nodes cannot be fulfilled. Since fixed logic is used there is no possibility of reconfiguring the architecture or changing the generator polynomial[114].

4.2.3 Parallel Solution

Parallelism improves the computational speed in CRC generating hardware. The speed is increased by a factor between 4 and 6 when using a parallelism of 8 bits. By using fixed logic, implemented as parallelized hardware, CRC generation can achieve wire speed and therefore it is the pre-dominant method used in computer networks. Like any other combinatorial circuit, parallel CRC hardware could be synthesized with only two levels of gates. Thus, this implies a huge number of gates. Also, the minimization of the number of gates is an NP-hard optimization problem. Therefore, for realization of complex circuits, heuristics is used or one seeks customized solutions. Campobella et al. derive a recursive formula that can be used to deduce the parallel CRC circuits as in modern synthesis tools [41].

Also, Albertengo and his team designed the parallel CRC encoders based on digital system theory and z-transforms, which allows designers to derive the logic equations of the parallel encoder circuit for any generator polynomial [31].By inspecting the operations of the single-input LFSR, Pei[32], derived the state transition equation for the parallel CRC circuit by merging a number of the shift and modulo-2 (XOR) operations together within a single clock cycle [115]. Also some attempts of fast CRC computation, which is based on the observation that only a small portion residing in the beginning of an Ethernet frame is changed during the hops for updating the relevant source and destination address over the network. Here, the fast CRC update only calculates the changed portion of a frame and performs a single step update

afterwards to obtain the new CRC code. This method dramatically improves the throughput of CRC recalculation which can support a throughput of about 56 Gbps[116]. In above designs if the CRC polynomial is changed or a new protocol is added, new changed hardware must be installed in the network terminal. The lack of flexibility makes this architectures non suitable for use in a protocol processor. Hence, very often new hardware schemes which compute the transition and control matrix of a parallel cyclic redundancy checksum are proposed. This opens possibilities for parallel high-speed cyclic redundancy checksum circuits that reconfigure very rapidly to new polynomials [117].

4.2.4 Configurable Hardware

Configurable hardware are implemented using Look-Up-Tables (LUT). This implementation can be modified by using a larger or smaller LUT. If the size of the LUT is reduced the hardware-cost in terms of power consumption and area will be reduced but at the same time the Combinational Network will be increased so the effect will be cancelled. The optimal solution has not been derived. Another, novel implementation method is the Radix-16 Configurable CRC Unit. Toal and his team proposed architecture using field reprogrammable chips so that it is fully flexible in terms of the polynomial deployed and the input port width. They synthesized circuit and mapped it to 130-nm UMC standard cell ASIC technology which is capable of supporting line speeds of 5 Gb/s [118].

4.3 Simulation Model for Error Detection using CRC

When the errors are introduced in Ethernet data streams using the experimental setup, the report of the number of packet errors generated can be obtained from the Report menu of the TSE design, however the user application is not able to catch the Ethernet frame which is in error, since the frame is discarded. Also the CRC field of the Ethernet frame is not read. Hence it is difficult to implement direct methods of monitoring the packet loss for the performance study of the Ethernet for the particular CRC polynomial. We have adopted indirect methods of quantifying the performance of the protocol. We have developed a Simulation Model in Matlab for BER performance evaluation, as illustrated in Figure 4.3.1.

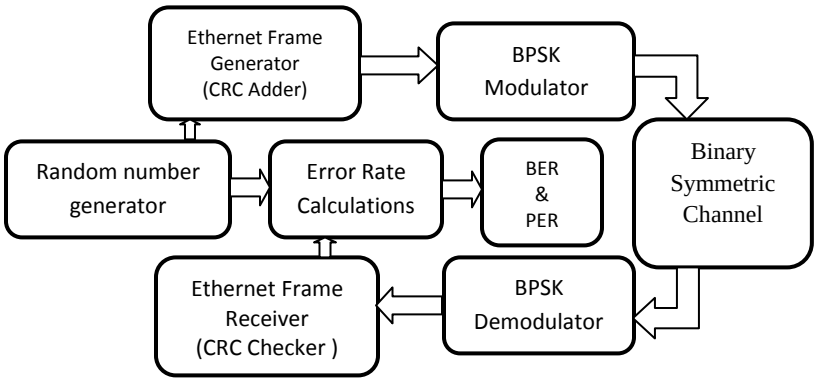


Figure 4.3.1 Matlab Simulation model for BER performance for BSC channel

This model can be used as a platform for Error Detection Analysis using CRC algorithms. A random number generator is used to generate a sequence of Ethernet frames of length varying from 64 bytes to 9600 bytes. The frames includes 4 bytes FCS, which is computed using Matlab code.

The transmitter modulates the frame bits using Binary Phase Shift Keying (BPSK), which is sent through a BSC. The amount of noise in the channel can be controlled. The receiver receives the noisy signal, demodulates it and produces a sequence of recovered bits. The FCS is recomputed at the receiver to find whether there was an error in the transmitted packet. Finally, we compare the received bits to the transmitted bits to get the BER and PER [119]. The model is user friendly and has a capability to vary the CRC polynomial, size of frames and number of frames. The platform can be used to test the performance of CRC error detection algorithms using different CRC polynomials. The Matlab code for this model is given in Appendix II. Also the study of BER performance for BPSK modulation in AWGN channel is performed.

4.4 Error Correction using LDPC codes

The data communication rates continue to increase due to bandwidth intensive applications such as video transport, data over IP, wireless base station, and medical applications[99]. Also increasing data rates entail more tight latency requirements, specifically

for full-duplex communication that involves streaming data like voice. Lower latency means less scope for error correction, which in turn puts more stringent requirements on the number of acceptable bit errors in the transmission system. LDPC codes have a performance close to Shannon limit coding gains when iteratively decoded [120]. They are used in several digital communication systems such as digital video broadcasting MIMO-WLAN (802.11n), (DVB-S2), WMAN (802.16e), mobile broadband wireless access (MBW A) (802.20) [121] and have a very good error correcting performance over a range of channels.

The main advantage of LDPC codes is that their performance is very close to the capacity for a lot of different channels and linear time complex algorithms are used for decoding. They are suited for implementations that make substantial use of parallelism. LDPC codes were first introduced by Gallager in his Ph.D. thesis in 1960. However, due to the computational effort in the implemention of coder and encoder and the introduction of Reed-Solomon codes, they were mostly ignored until 1990s. LDPC codes were independently rediscovered by MacKay [122,123] and by Luby et al. [124]. Since this time, much work has been carried out on the design of good codes and on the analysis of their performance [125,126,127]. Turbo codes and LDPC codes provide coding gains close to Shannon's limit [128,129,126]. LDPC codes however outperform turbo codes in terms of coding gain for large SNR [130,126] Also LDPC codes require less computational complexity and less number of iterations compared to turbo codes. Thus it results in higher throughput and lower power dissipation. Error correction algorithms are frequently implemented in hardware for fast processing to meet the real-time needs of communication systems [76].

4.4.1 Representation of LDPC Codes

LDPC can be represented in two ways. The first is using matrices and second is graphical representation [131].

Matrix Representation

Equation (4.3) defines a parity check matrix with dimension 'n × m' for a (8, 4) code.

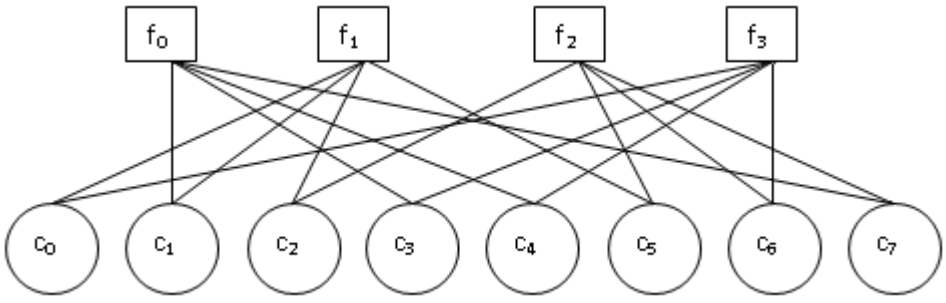
$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (4.3)$$

The number of 1's in each row is denoted by 'w_r' and 'w_c' represents the number of 1's in each for the column. The matrix can be called as low-density if the two conditions 'w_c << n' and 'w_r << m' are satisfied. Hence, in reality, the parity matrix should be very large.

Graphical Representation

Tanner introduced an effective graphical representation for LDPC codes. These graphs provide a complete representation of the code and they also help to describe the decoding algorithm. Tanner graphs are bipartite graphs. The nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The two types of nodes are called variable nodes (v-nodes) and check nodes (c-nodes). Figure 4.4.1 is an example for such a Tanner graph and represents the same code as the matrix in equation (4.3). The graph consists of 'm' check nodes (the number of parity bits) and 'n' variable nodes (the number of bits in a codeword). Check node 'f_i' is connected to variable node 'c_j' if the element 'h_{ij}' of 'H' is a 1.

c_nodes



v_nodes

Figure 4.4.1 Tanner graph corresponding to the parity check matrix in equation (4.3)

A LDPC code is called regular if 'w_c' is constant for every column and '$w_r = w_c \cdot (n/m)$' is also constant for every row. The example matrix given in equation (4.3) is regular with '$w_c = 2$' and '$w_r = 4$'. Graphical representation shows the regularity of this code. The number of incoming edges is same for every 'v-node' and also for all the 'c-nodes'. If the numbers of 1's in each row or column are not constant, the code is called an irregular LDPC code. Conditions for the decoding algorithm to be optimal is graph remains cycle free and all the information in updating the decoded bits or messages is unrelated and originating from the distinct received bits[132]. However, once a cycle is encountered the algorithm becomes suboptimal. Thus, the performance of the decoder is improved by maximizing the cycle lengths in the course of the design of the parity check matrix 'H' subject to the constraints of the block size and required column/row weight profiles[133].

4.4.2 LDPC decoding algorithms

The LDPC codes can be decoded by a local message-passing algorithm on the graph, the Sum Product Algorithm (SPA).

Sum-Product Algorithm – Probability Domain

The parity check matrix has a dimension of (N-K) x N and consists of N-K "Check nodes and N "Bit nodes" representing parity check equations and code bits, respectively. Decoding is performed iteratively [134]. In each iteration, every Bit node passes a message to the Check nodes that are connected to it. In the next half iteration, each Check node sends a message to the Bit nodes. This message is a function of all the extrinsic information that it has received from the Bit nodes in the last part. Then, the codeword is checked for validity. It performs the iterations until it finds the valid code word or reaches the maximum number of the iterations. The following steps should be performed for all the 'is' and 'js' for which the element in the parity check matrix is a one (h_{ij}=1).

Step 0: Initialize 'q_{ji}' by:

$$q_{ji}(0) = 1 - p_i = P_r(x_i = +1 \mid y_i) = \frac{1}{1 + e^{-2y_i/\sigma^2}} \tag{4.4}$$

$$q_{ji}(1) = p_i = P_r(x_i = -1 \mid y_i) = \frac{1}{1 + e^{2y_i/\sigma^2}} \tag{4.5}$$

This step initialized the values of $Q(x)_{ij}$ which are set to a priori estimates of the received symbols 'y_i' obtained after hard-decision decoding of the received decoded vector. Here 'σ' is standard deviation of the noise over the AWGN channel.

Step 1: Horizontal stepping on 'r_{ji}' by:

$$r_{ji}(0) = 1/2 + 1/2 \prod_{i' \varepsilon R_{j \setminus i}} \left(1 - 2q_{ji'}\ (1)\right) \tag{4.6}$$

$$r_{ji}(1) = 1 - r_{ji}(0) \tag{4.7}$$

Here, the coefficients '$r_{ji}(x)$' are the values which are estimates that go from parity check node to the symbols nodes, in the bipartite graph. The probability associated with each combinations are added to calculate the estimates of '$r_{ji}(x)$'.

Step 2: Vertical stepping on 'q_{ji}':

$$q_{ji}(0) = K_{ji}(1 - p_i) \prod_{j' \varepsilon C_i} r_{j'i}(0) \tag{4.8}$$

$$q_{ji}(1) = K_{ji} p_i \prod_{j' \varepsilon C_{i \setminus j}} r_{j'i}(1) \tag{4.9}$$

where the constants K_{ji} are chosen to ensure that $q_{ji}(0) + q_{ji}(1) = 1$. Here the normalizing constants 'K_{ji}' satisfy the normalizing condition $\sum_x Q(x)_{ij}=1$. The $(1-p_i)$ and (p_i) are the coefficients corresponding to the received hard-decision vector.

Step 3: For all the i's compute:

$$Q_i(0) = K_i(1-p_i)\prod_{j\varepsilon C_i} r_{ji}(0) \tag{4.10}$$

$$Q_i(1) = K_i p_i \prod_{j\varepsilon C_i} r_{ji}(1) \tag{4.11}$$

where the constants 'K_i' are chosen to ensure that $Q_i(0) + Q_i(1) = 1$.

Step 4: For every row index i:

$$\hat{c}_i = \begin{cases} 1 \; if Q_i(1) > Q_i(0) \\ 0 \; else \end{cases} \tag{4.12}$$

If $\hat{c}H^T=0$, or if maximum number of iteration is reached then stop, else continue iterations from Step 1.

Sum-Product Algorithm- Log Domain

SPA used in decoding of LDPC codes requires a large number of multiplications of probabilities which makes the algorithm numerically unstable, specially for very long codes. It may be noted that the best performance is obtained for very long codes. This long block length, combined with the need for iterative decoding, introduces latency which is unacceptable in many applications. Thus, a log-domain version of the algorithm is preferred. We define the following log likelihood ratios as part of the decoding algorithm:

$$Lc_i = \log\left(\left(P_r(x_i = +1 \mid y_i)\right)/\left(P_r(x_i = -1 \mid y_i)\right)\right); \tag{4.13}$$

$$Lr_{ji} = \log(r_{ji}(0)/r_{ji}(1)); \tag{4.14}$$

$$Lq_{ji} = \log(q_{ji}(0)/q_{ji}(1)); \tag{4.15}$$

$$LQ_i = \log(Q_i(0)/Q_i(1)). \tag{4.16}$$

This algorithm iterates over columns and rows of parity check matrix H, and operates on nonzero entries by performing the following steps:

Step 0: Initialize Lq_{ji} by: $Lq_{ji} = Lc_i = 2y_i/\sigma^2$; which initialized the values of coefficients $Q(x)_{ij}$ over logarithmic scale for the received symbols 'y_i' obtained after hard-decision decoder of the received decoded vector having 'σ' as standard deviation of the noise over the channel.

Step 1: Evaluate ***Lr$_{ji}$*** by: $Lr_{ji} = \left(\prod_{i'\varepsilon R_{j\backslash i}} \alpha_{ji'}\right).\phi\left(\sum_{i'\varepsilon R_{j\backslash i}} \phi(\beta_{ji'})\right)$; (4.17)

where, $\alpha_{ji} = sign(Lq_{ji})$, $\beta_{ji} = \left\|Lq_{ji}\right\|$, $\phi(x) = -\log(\tanh(x/2)) = \log\left(e^x + 1/e^x - 1\right)$.

Step 2: $Lq_{ji} = Lc_i + \sum_{j'\varepsilon C_i \backslash j} Lr_{j'i}$. Step 3: $LQ_i = Lc_i + \sum_{j\varepsilon C_i} Lr_{ji}$. (4.18)

Step 4: For every row index i: $\hat{c}_i = \begin{Bmatrix} 1 & if\ LQ_i < 0 \\ 0 & else \end{Bmatrix}$. (4.19)

If $\hat{c}H^T$=0, or if maximum number of iteration is reached then stop, else continue iterations from Step 1.

Min-Sum Algorithm

Consider the update equation for 'Lr_{ji}' in the Sum-Product algorithm:

$$Lr_{ji} = \left(\prod_{i'\varepsilon R_{j\backslash i}} \alpha_{ji'}\right).\phi\left(\sum_{i'\varepsilon R_{j\backslash i}} \phi(\beta_{ji'})\right) \quad (4.20)$$

The 'ϕ (x)' is a function which is decreasing for the values of x > 0. It is intuitive that the term corresponding to the smallest β_{ji} in the above summation dominates, so that the second term in above equation : $\phi\left(\sum_{i'\varepsilon R_{j\backslash i}} \phi(\beta_{ji'})\right) = \phi(\phi(\min_{i'} \beta_{ji'})) = \min_{i'} \beta_{ji'}$. The second equality follows from ϕ (ϕ (x)) = x. Thus the Min-Sum algorithm is the same as SPA in which Step(1) is replaced by this equation: Step 1': $Lr_{ji} = (\prod_{i'\varepsilon R_{j\backslash i}} \alpha_{ji'}).(\min_{i'\varepsilon R_{j\backslash i}} \beta_{ji'})$. Because of the approximation in this equation, there is a degradation in the performance of Min-Sum compared to SPA.

Modified Min-Sum Algorithm

Scaling the soft information during the decoding using min-sum algorithm, results in better performance. Scaling slows down the convergence of iterative decoding and reduces the overestimation error compared to SPA. In this algorithm, it is enough to change the Step 2 in Min-Sum Algorithm with

$$[Step2':]Lq_{ji} = (Lc_i + \sum_{j' \varepsilon C_i \setminus j} Lr_{j'i}) * \gamma \quad (4.21)$$

in which γ is the scaling factor. Min-Sum algorithm which is an approximation of the Sum-Product, suffers from some performance loss because of the approximations. On the other hand, Modified Min-Sum shows even better performance than SPA in some SNR ranges. Modified Min-Sum algorithm replaces the costly function evaluations with addition and shift operations. Although Modified Min-Sum has a few more additions than other algorithms, it is still preferred since nonlinear function evaluations are omitted. Also Modified Min-Sum algorithm requires less memory, hence better implemented in the hardware [134].

The direct implementation of the decoding algorithm for binary codes using the probability domain has several drawbacks as compared log-SPA, based on log-likelihood ratios (LLR). The direct implementation is more sensitive to quantization effects and requires more quantization levels than when using LLRs [135,136]. The SPA requires message multiplications, whereas the log-SPA implementation uses message additions. The latter is more efficient in fixed point implementations, as fixed point multiplications can take up many clock cycles compared to additions. Additions in the SPA are replaced by the Jacobi logarithm [137] which can be approximated by a very simple function, resulting in the max-log-SPA. Finally, the log-SPA implementation requires no normalization step [138]. A log-domain approach for non-binary turbo codes was investigated in [139]. In this contribution the log-SPA decoding scheme is derived for general non-binary LDPC codes based on LLRs. This algorithm is mathematically equivalent to the original decoding algorithm [140].

4.5 Simulation Model for Error Correction using LDPC codes

Performance of a communication system can be characterized by the BER expected from the channel. BER value is the probability of occurrence of bit-error at the receiver, and for reliable communication, the BER values should be very low. Forward Error Correction (FEC) schemes using error-control codes are widely used to reduce the BER values. Figure 4.5.1 shows a MATLAB model developed to study the BER performance of Gigabit Ethernet protocol using LDPC codes.

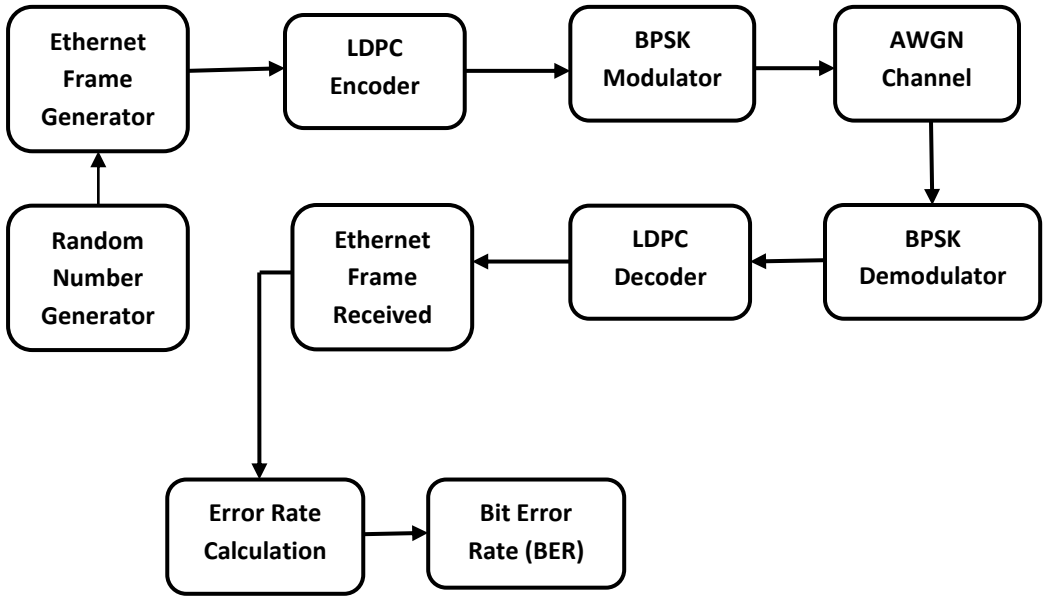


Figure 4.5.1 Matlab Simulation model for Error Correction Analysis

This model can be used for Error Correction Analysis of Gigabit Ethernet protocol. The Ethernet Frame is generated using Matlab code and is encoded using a generator matrix. The BPSK modulator maps the input binary signals, to an analog signal for transmission. The channel is the medium through which information is transmitted from the transmitter to the receiver and can be a copper, fibre optic or wireless channel. The channel is modelled as an AWGN channel. The LDPC decoder is implemented at the receiver. The decoding process uses standard functions in MATLAB for SPA-logdomain and SPA-Min-Sum algorithm, that loops through passing messages back and forth along the Tanner graph until maximum number of passes have

occurred. The BER performance is studied by measuring the BER of the received decoded message [141,142].

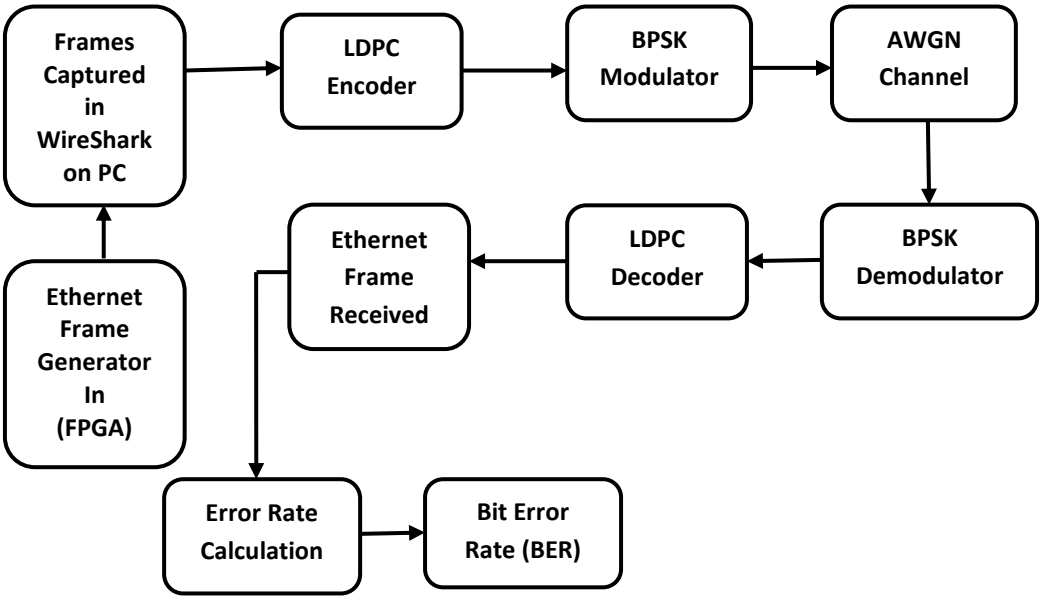


Figure 4.5.2 Matlab Simulation model for Error Correction interfaced with Gigabit Ethernet protocol design

Figure 4.5.2 illustrates the interfacing of the Matlab Simulation model with Gigabit Ethernet protocol design implemented on FPGA. Ethernet frame is generated using Altera's TSE IP core generator and these frames are captured using WireShark and given to the Matlab simulation model of error correction and the BER performance is plotted.

CHAPTER 5

DISCUSSIONS AND CONCLUSIONS

5.1 Performance Analysis of Gigabit Ethernet protocol design

The performance of the implemented Gigabit Ethernet protocol design was studied for Gigabit Ethernet standards 1000Base-LX, 1000Base-SX and 1000Base-T using various physical media and corresponding SFP Transceivers as mentioned in Table 5.1.1. Figure 5.1.1 gives the photograph of the different SFP modules used for the study.

Table 5.1.1 List of SFP Transceivers used for the different Gigabit Ethernet standards

SFP Standard	**Fiber / Cable Type**
LX	1310 nm 10 km single-mode fiber (SMF)
GOGS	1310 nm 20km SMF
BX	1550 nm 1310 nm 565m Bi-Directional SMF
SX	850 nm 550 m multi-mode fiber (MMF)
GOIP	1000BaseT Module Cat5e, Cat6 Copper Cables

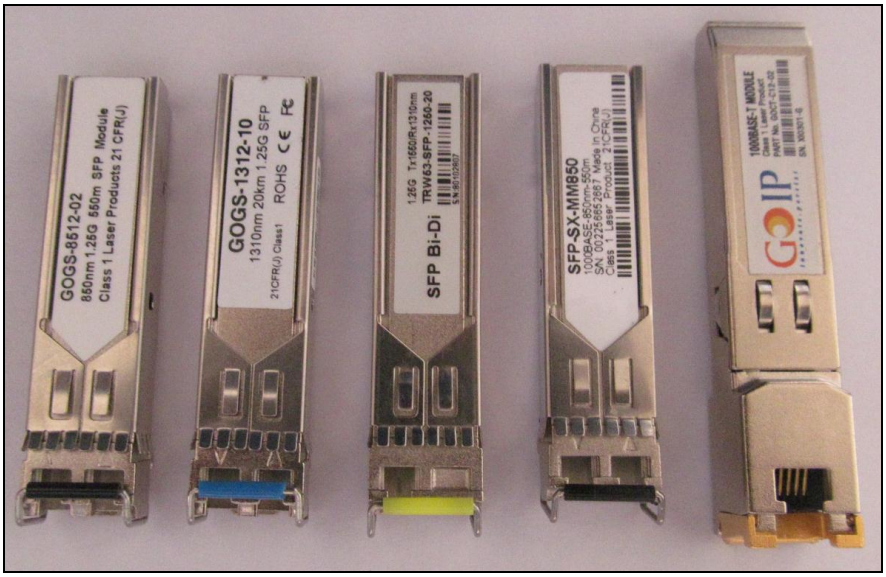


Figure 5.1.1 Top view photograph of the different SFP modules

The performance studies of Gigabit Ethernet protocol design on the Altera FPGA were done by using the Testbed setup for Performance Analysis explained in Section 3.5 for the different physical media. The studies were carried out for frame lengths 64 bytes to 9600 bytes. This measurement was taken for fixed number of packets 10^5 and 10^7.

The line rate and throughput obtained for different frame lengths for 1000Base-LX is listed in Table 5.1.2. From the table, it is seen that the line rate is found to be 76.19% for minimum 64 bytes frame length and approaches 100% for 9600 bytes frame length. Similarly, the throughput is lowest for 64 bytes frame size and approaches 1Gbps for 9600 bytes frame length. Throughput is also determined in packets per second (pps).

Table 5.1.2 Line Rate and Throughput of different frame lengths

Frame length (bytes)	**Line Rate (%)**	**Throughput (bits/sec)**	**Throughput (pps)**
64	76.19	7.62E+08	1488106
128	86.48	8.65E+08	844607
256	92.75	9.28E+08	452901
512	96.24	9.62E+08	234964
1024	98.08	9.81E+08	119733
1518	98.7	9.87E+08	81274
2048	99.03	9.90E+08	60445
4096	99.51	9.95E+08	30369
8192	99.75	9.98E+08	15221
9600	99.79	9.98E+08	12993

The study was repeated for 1000Base-X and 1000Base-T and the results were plotted in Figure 5.1.2 and 5.1.3. Both these figures indicate that for a particular frame length, the throughput and line rate remain almost the same for 1000Base-LX, 1000Base-SX as well as 1000Base-T standard.

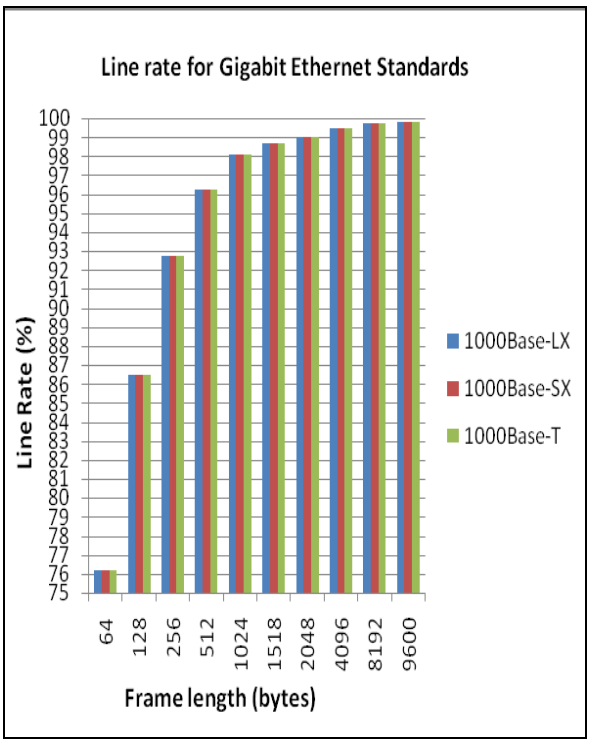


Figure 5.1.2 Line rate vs Frame length

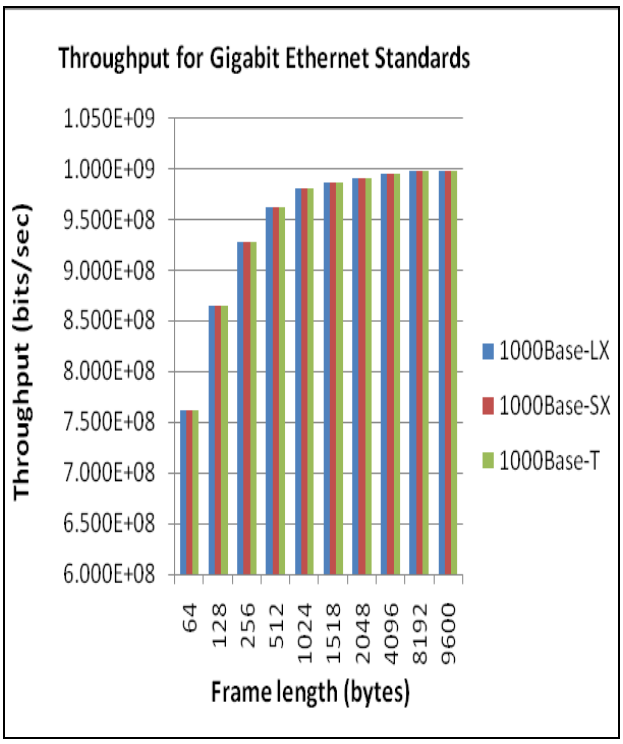


Figure 5.1.3 Throughput vs Frame length

Table 5.1.3. Throughput of the network obtained by Duan and Han[1]

Length (byte)	Line rate (%)	throughout (pps)
64	7	20825
128	12	20268
256	18	16303
512	44	20670
1024	87	20825
1280	100	19224
1518	100	16250

The results of the studies can be compared with [1], wherein Gigabit Ethernet protocol is implemented using Broadcom's BCM5421S chip and Xilinx FPGA, which is summarized in Table 5.1.3. Comparing Table 5.1.2 and 5.1.3, it is seen that in[1] for 64 bytes length frame, the line rate is only 7 % and the throughput in pps is 20825, which is very low compared to our implemented design, wherein the throughput is 1488106 pps for 64 bytes frame length. From Table 5.1.3, for a frame length of 1024 and 1518, the line rate is 87% and 100% respectively, whereas, in our design, it is 98% and 98.7% respectively. Studies have also been performed to find the variation of line rate and throughput for constant length of data and similarly, the studies on effect of increasing frame length on the total transmission time was carried out. The results of the tests for 1000Base-SX standard are plotted. In Figures 5.1.4 and 5.1.5, N is the total number of bytes of data, which is kept constant at 1.28E+07 and the number of frames transmitted is altered, depending on the frame length. The tests are repeated for N=1.28E+08 and N=1.28E+09. From this tests, we can say that, the values of line rate and throughput remain the same, for a particular frame length for any value of data bytes.

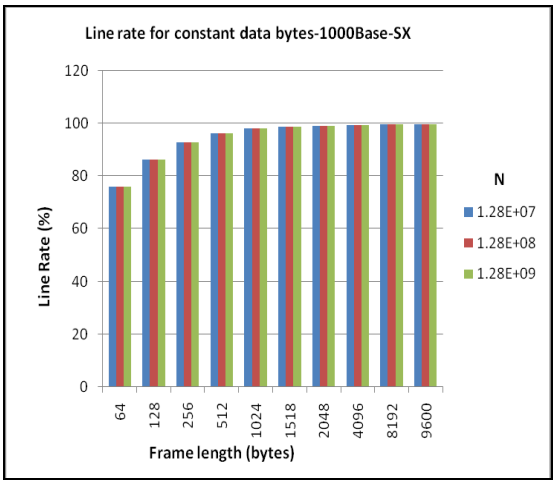


Figure 5.1.4 Line Rate vs Frame length

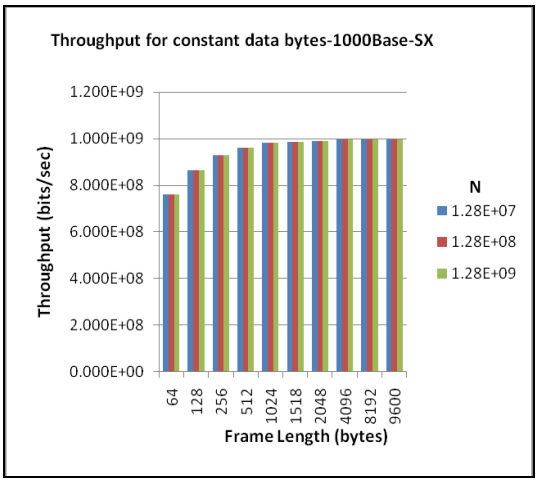
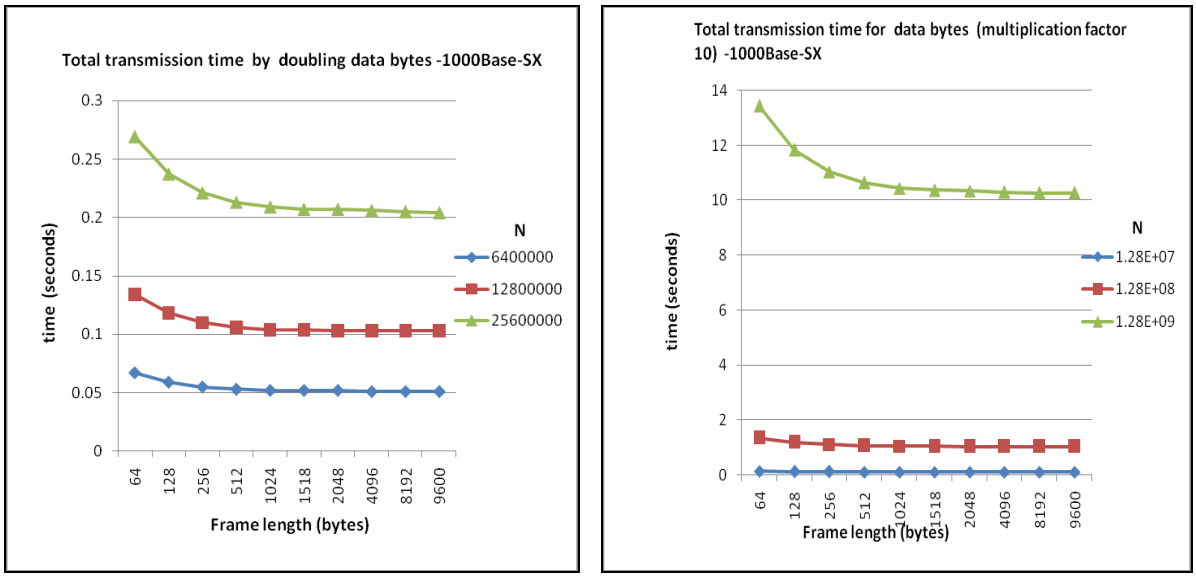


Figure 5.1.5 Throughput vs Frame length



(a) (b)

Figure 5.1.6 Transmission time vs Frame length a) doubling data bytes b) multiplying N data bytes with a factor of 10

From Figure 5.1.6, for a fixed N in bytes, as the frame length increases, the total transmission time is found to be decreasing and remains constant from 1024 bytes frame length onwards. Also, as N is doubled, the total transmission time approximately doubles. Similar result is obtained in Figure 5.1.6 b) multiplying N data bytes with a factor of 10.

5.2 Simulation Results for Error Detection

Simulation studies have been carried out for wordwise IEEE-802 CRC polynomial for Gigabit Ethernet using the model given in Section 3.4. Figure 5.2.1 and 5.2.2 show the BER and PER performance respectively of Gigabit Ethernet protocol model for a BSC. Both the results are plotted for a frame length of 64 bytes to 1518 bytes for 100 frames. The bit error rate probability of BSC is varied from 0.001 to 0.00001. From Figure 5.2.1, it can be seen that the BER decreases exponentially with decrease in Error probability. Similarly, Figure 5.2.2 shows that PER approaches zero for .00001 error probability for all frame lengths.

Table 5.2.1 summarizes the list of CRC codes for which the performance could be studied over the model.

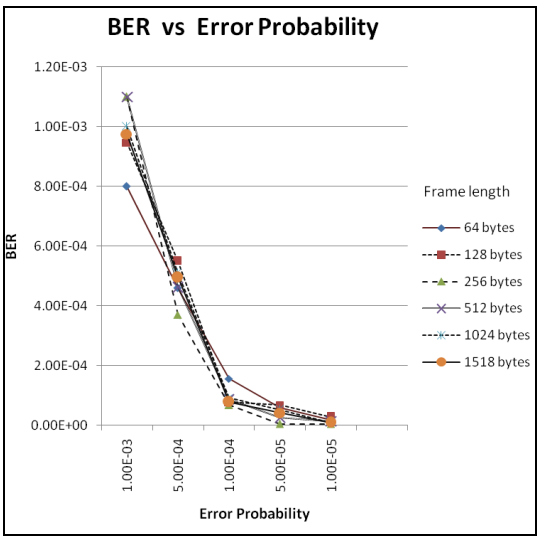


Figure 5.2.1 BER Performance for BSC

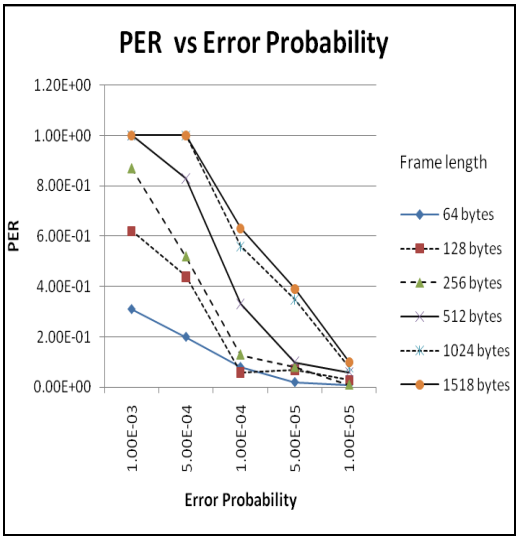


Figure 5.2.2 PER Performance for BSC

Table 5.2.1 List of CRC codes and their generator polynomials

Sr. No.	CRC Code	Generator Polynomial g(x)
1	CRC-32/8	1F1922815
2	MP-CRC-32/8.1	1404098E2
3	MP-CRC-32/8.2	10884C512
4	CRC-32/6	1F6ACFB13
5	CRC-32/5.1	1A833982B
6	CRC-32/5.2	1572D7285
7	CRC-32/4	11EDC6F41
8	**IEEE-802**	**104C11DB7**
9	CRC-40/GSM	0004820009
10	CRC-64	42F0E1EBA9EA3693

We also studied the BER performance for BPSK modulation in AWGN channel. The frame length is varied from 64 bytes to 9600 bytes. The number of frames was kept constant at 100.

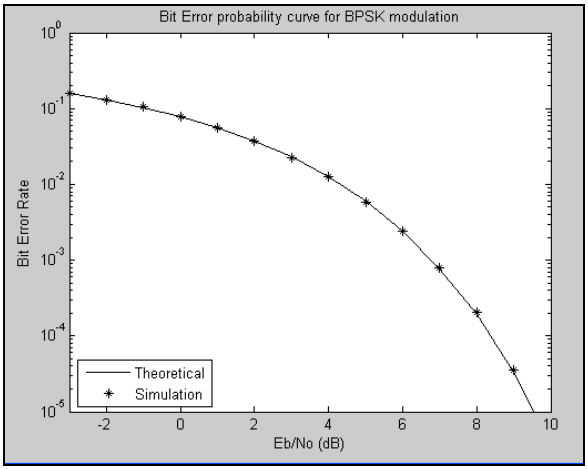


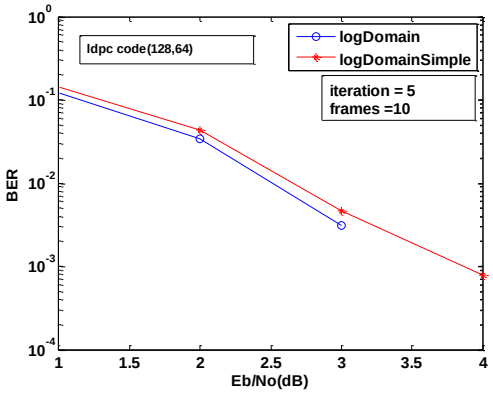
Figure 5.2.3 BER vs Signal to Noise Ratio (Eb/No) in dB

Figure 5.2.3 gives the Bit Error Rate Probability curve for frames of different lengths plotted for Signal to Noise ratio (Eb/No) varying from -3dB to 10 dB. It is found that the theoretical value of Bit Error Rate Probability given by the equation $P_b = 1/2 erfc(\sqrt{(E_b / N_0)})$ and simulated BER values for all the frame lengths i.e. 64, 128, 256, 512, 1024, 1518, 2048, 4096 and 9600 bytes are exactly the same.

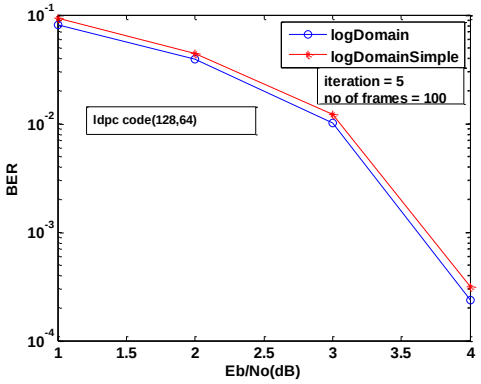
5.3 Results of Performance Studies of Error Correction using LDPC codes

The LDPC decoding algorithms, Sum Product Algorithm (SPA)-Logdomain and SPA-Min Sum Algorithm-logdomainSimple is chosen for Simulation studies, since they are easy for hardware implementation. The study is performed using rate 1/2 regular LDPC codes and SNR from 1dB to 6dB. Since LDPC codes require powerful computations, a Xeon processor is used for running the algorithms. The studies were performed using the Matlab model described in Section 4.5. Initially, the studies were peformed for block length 64 bits. The number of frames was

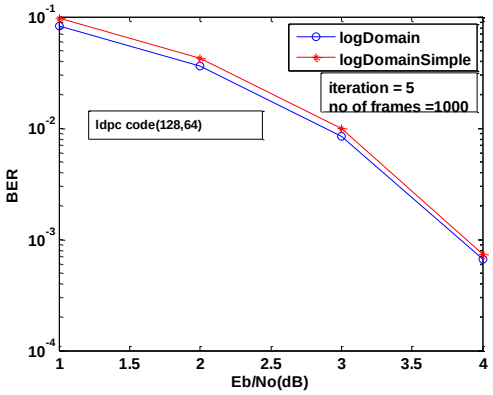
increased from 10 to 10^5 .The first set of results was plotted for 5 iterations as shown in Figure 5.3.1 a) to e).The second set of results in Figure 5.3.2 a) to e) was plotted for 10 iterations. From the Figures, it is revealed that SPA-Logdomain Algorithm has better performance than SPA-Min Sum Algorithm. Also, increasing the number of iterations improves the decoding performance. Comparing Figure 5.3.2 b), it is seen that BER approaches zero above SNR =3dB. Similarly, increasing number of frames gives more realistic results. Similar results were obtained for frames=100, 1000, 10000 & 10000. Hence it was decided to carry further simulations with optimum number of frames, 100 so that computation time can be reduced.



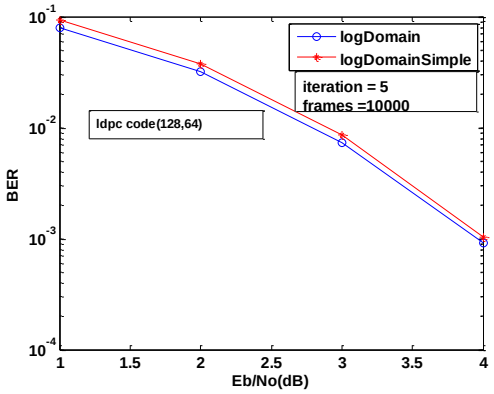
(a)



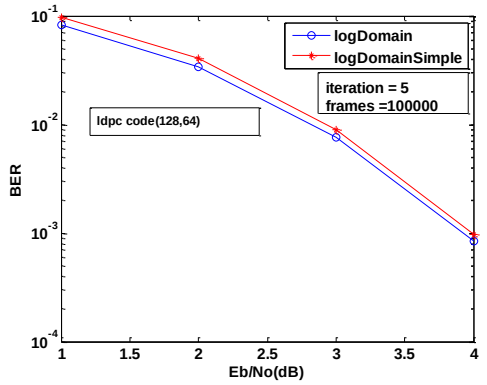
(b)



(c)



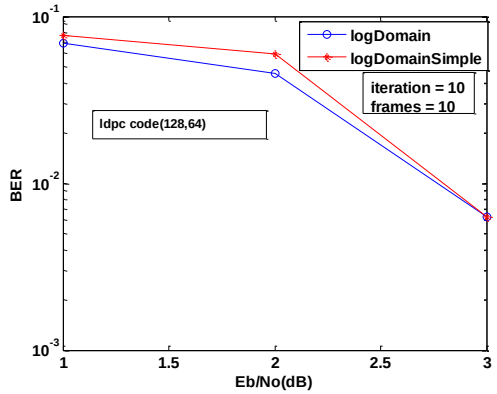
(d)



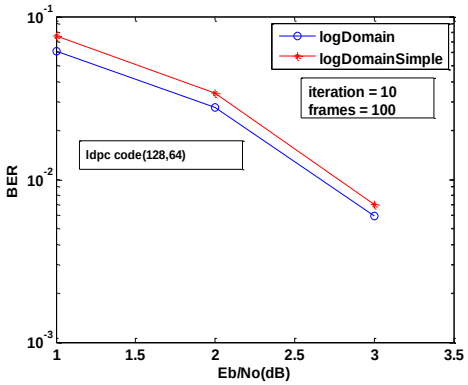
(e)

Figure 5.3.1 a) to e) Showing BER vs SNR for iteration= 5 for various number of frames

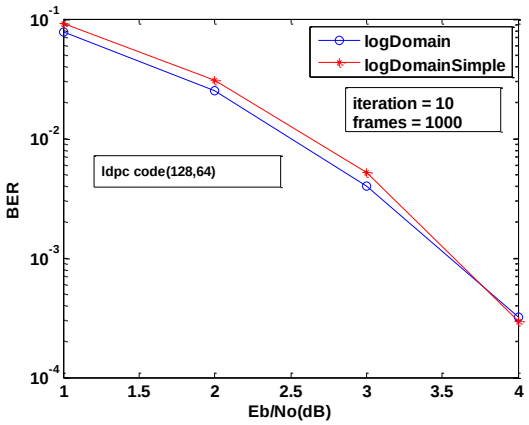
Note : It may be noted that BER of lower values cannot be obtained on the graph practically in Matlab, because the block length is only 64-bits, and hence, the curves in 5.3.1 can be implicitly extrapolated.



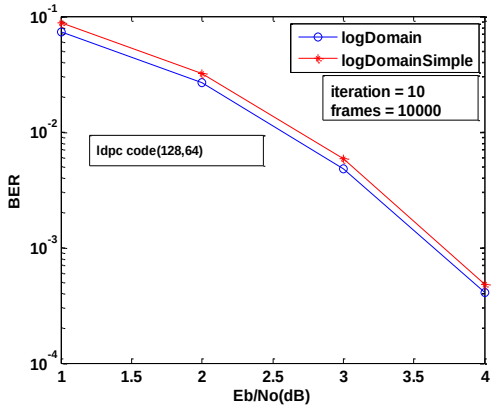
(a)



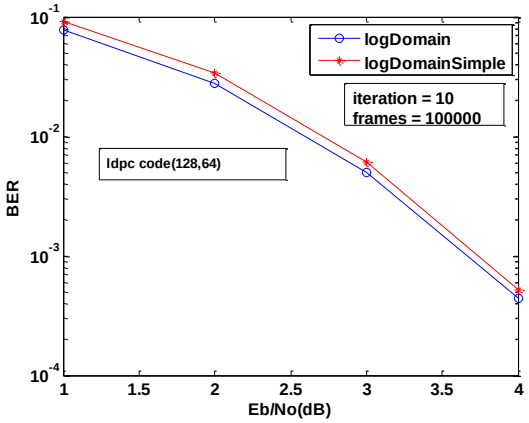
(b)



(c)

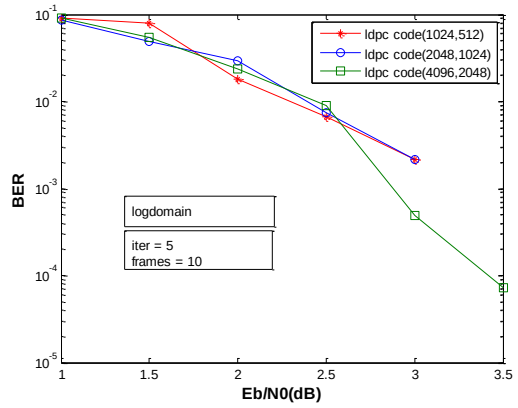


(d)

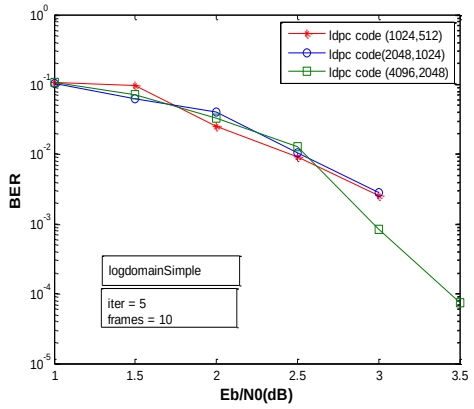


(e)

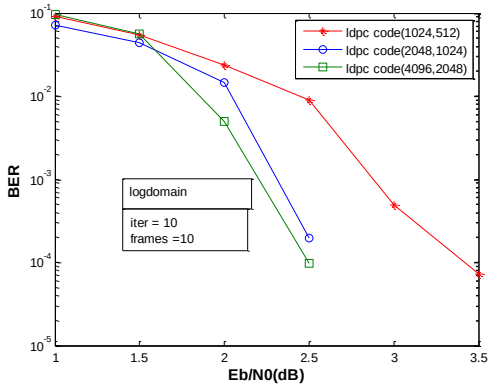
Figure 5.3.2 a) to e) Showing BER vs SNR for iteration= 10 for various number of frames



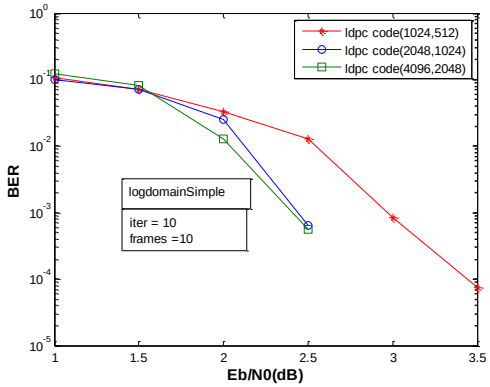
(a)



(b)



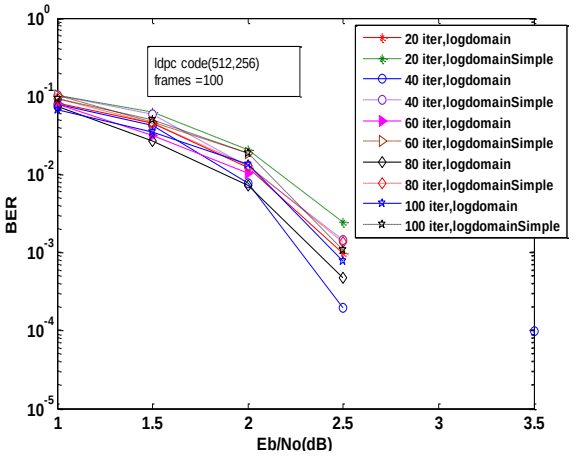
(c)



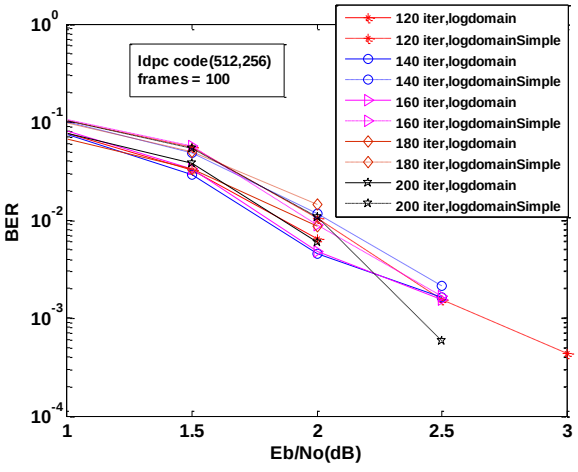
(d)

Figure 5.3.3 a) to d) Showing BER vs SNR for iteration= 5 and 10 for different block lengths and SPA log domain & SPA-Min Sum algorithms

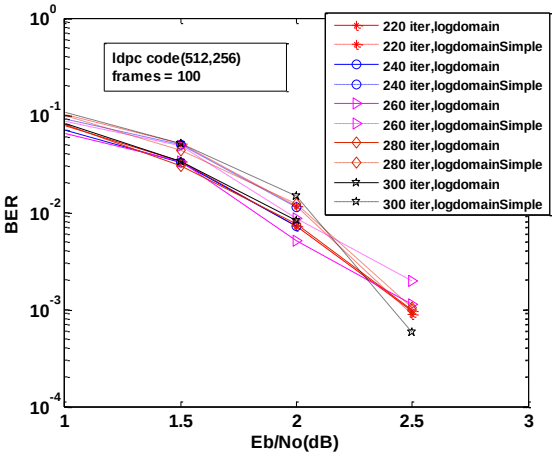
Figure 5.3.3 a) to d) gives the BER performance for SPA-Logdomain and SPA-Min Sum Algorithm-logdomainSimple for block length 512,1024 and 2048 bits corresponding to 64 bytes,128 bytes and 256 bytes frames respectively. Increasing the number of iterations gives better performance.



(a)

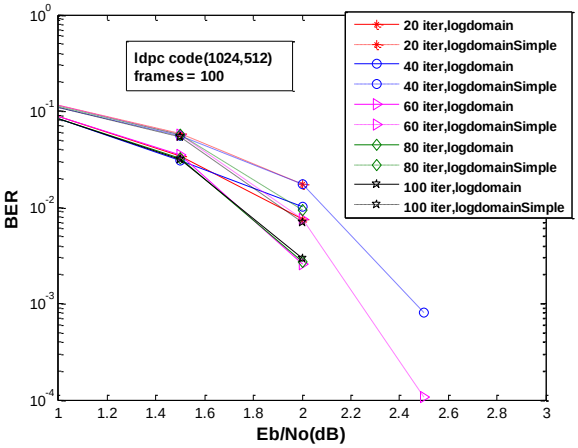


(b)

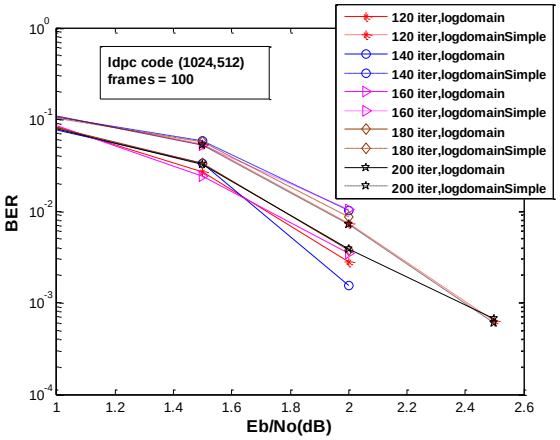


(c)

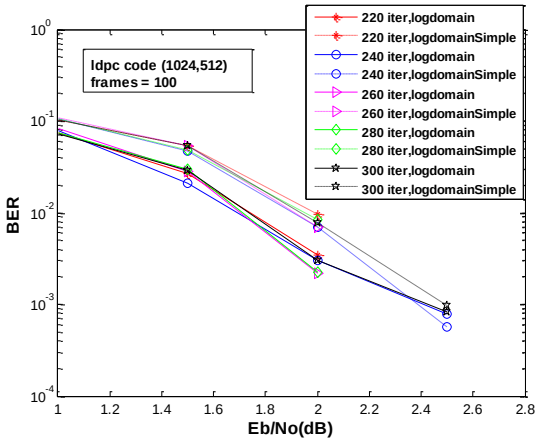
Figure 5.3.4 a) to c) BER vs SNR for iteration= 220 to 300 and block length 256 bits



(a)



(b)

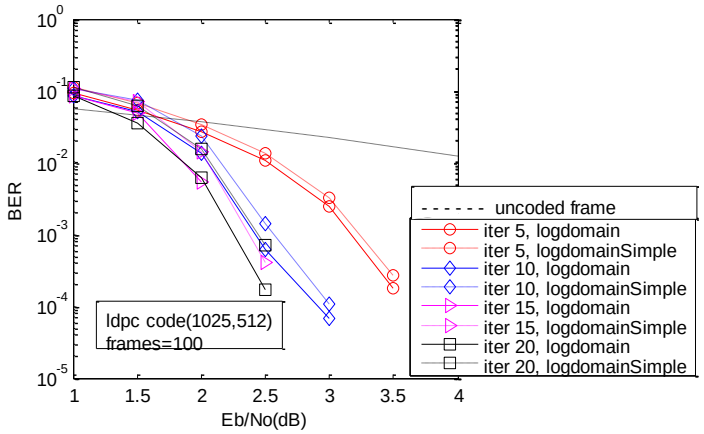


(c)

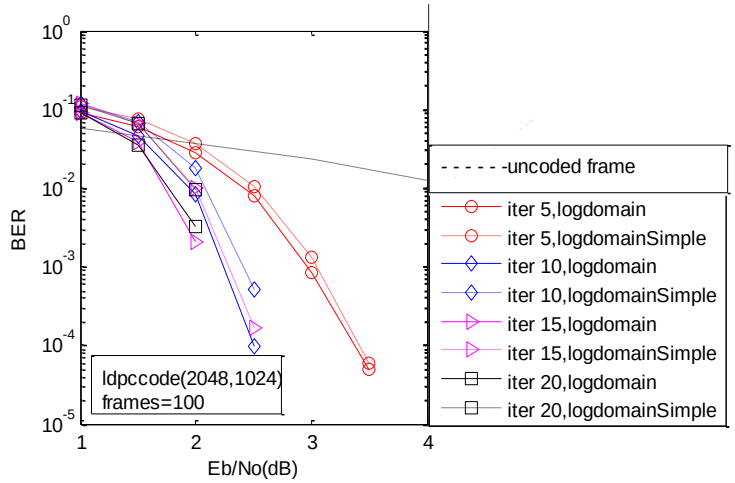
Figure 5.3.5 a) to c) Showing BER vs SNR for iteration= 20 to 300

Elaborate studies were performed to study the BER performance for 256 bits block length, with iterations 20 to 300. Comparing the results in Figures 5.3.4 and 5.3.5, it is seen that, for 256 as well 512 bits, increasing the number of iterations does not improve the performance proportionately. Hence the block length needs to be increased.

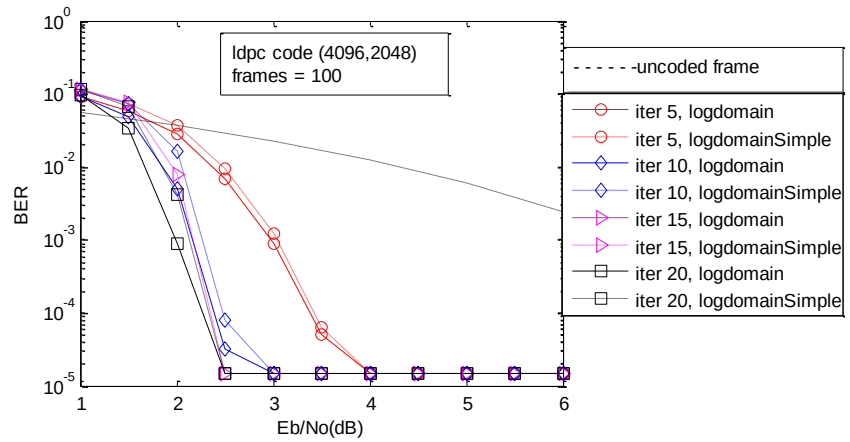
Next, the performance studies of LDPC codes was carried out for Ethernet Frames of block length 512,1024 and 2048 bits corresponding to 64 ,128 and 256 bytes frame lengths respectively. Figures 5.3.6 a) to 5.3.6 c) are plotted for number of iterations from 5 to 20 and Figure 5.3.6 d) is plotted for number of iterations 25 to 50.



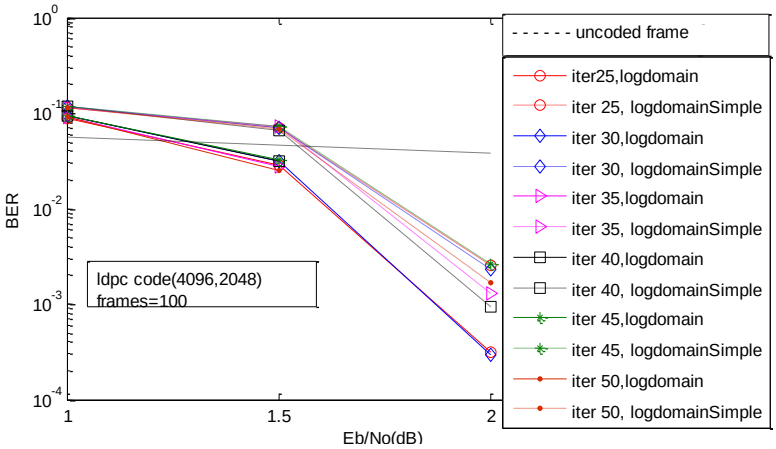
(a)



b)



c)

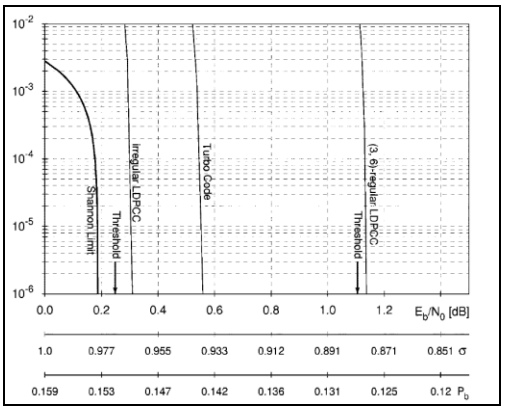


d)

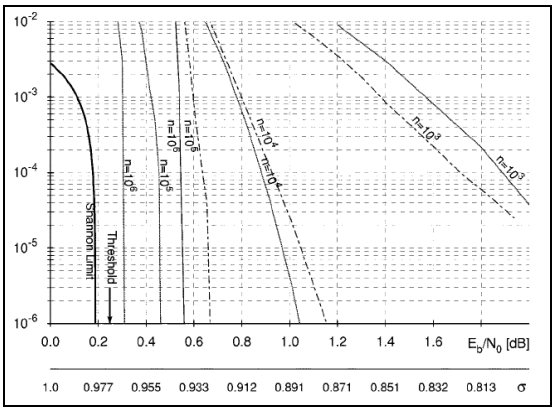
Figure 5.3.6 a) to d) BER vs SNR for different block lengths

Results, as illustrated in Figure 5.3.6 a) and 5.3.6 b) indicate that the BER performance improves with increase in block length and optimum error correction can be achieved by setting the number of iterations between 10 to 20 for a SNR above 2dB. However from Figure 5.3.6c) and Figure 5.3.6d), it is seen that more errors are introduced for larger frame lengths and hence the number of iterations need to be increased for better Error Correction performance.

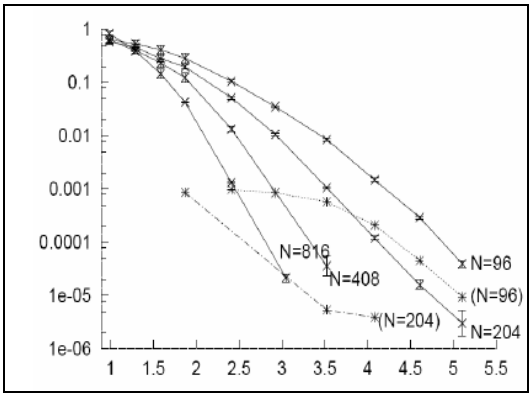
Figures 5.3.7 a) , b) and c) give the results of previous research of LDPC codes as illustrated below.



(a)



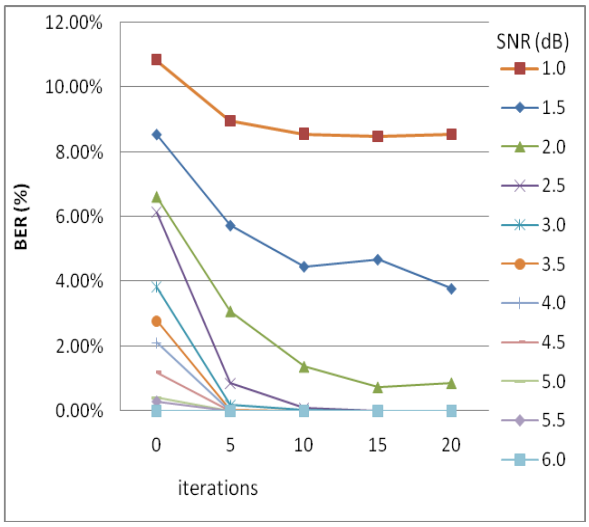
(b)



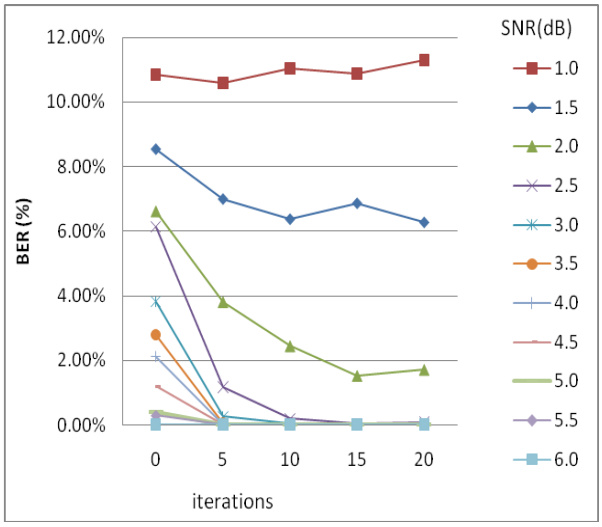
(c)

Figure 5.3.7 a) Comparison of LDPC codes $n=10^6$, R=1/2, b) Comparison of LDPC with Turbo codes for $N=10^3,10^4,10^5,10^6$ [125], c) Mackay's Results [143]

The results obtained in our study can be compared to the results obtained in above graphs. From Figure 5.3.7 b), for $n=10^3$ a BER of 10^{-4} can be achieved for a SNR of 1.65dB. From the result in Figure 5.3.6 b) for n=1024, a BER of 10^{-4} can be achieved for an SNR of 2.5dB, which is consistent with the result obtained in Figure 5.3.7c) for N=816.

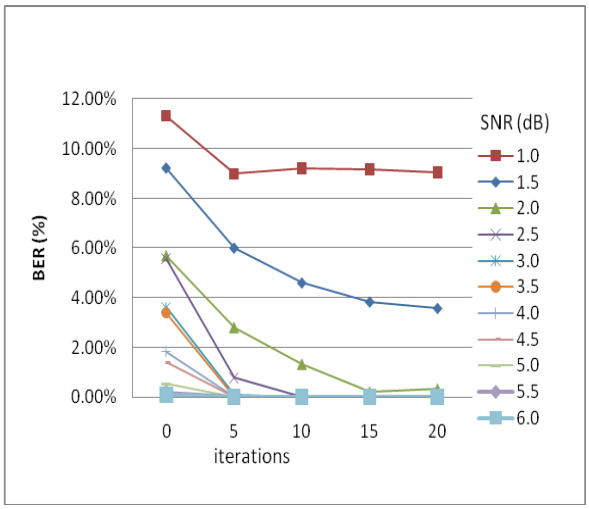


(a)

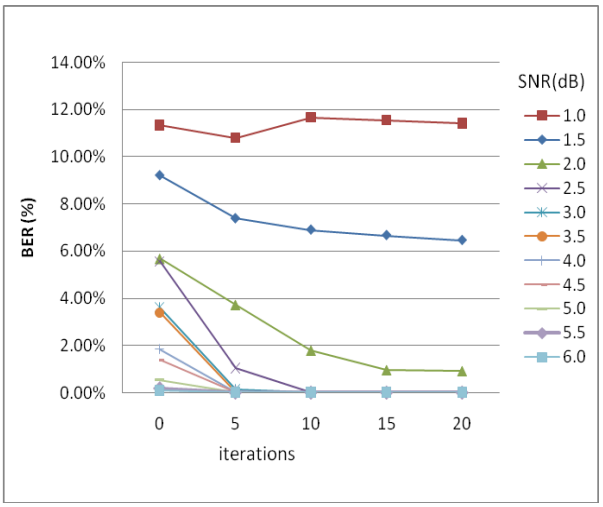


(b)

Figure 5.3.8 a) Showing Error correction performance of LDPC codes for block length =512 bits, SPA-logdomain algorithm b) Showing Error correction performance of LDPC codes for block length =512 bits and SPA-Min-Sum algorithm – logdomainSimple. The figures are plotted for number of iterations 0, 5 , 10, 15, 20 and SNR =1.0dB to 6.0dB.



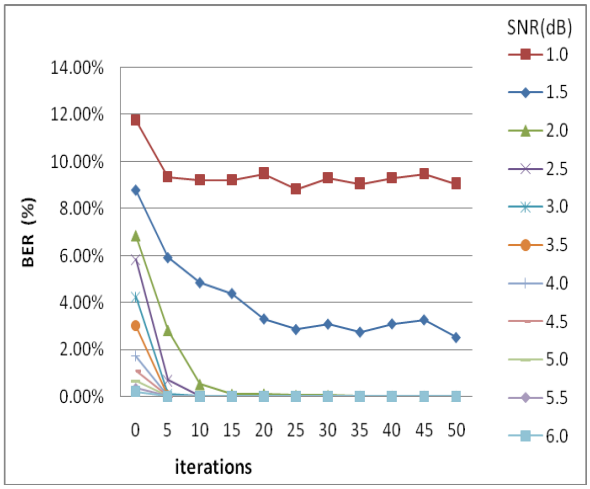
(c)



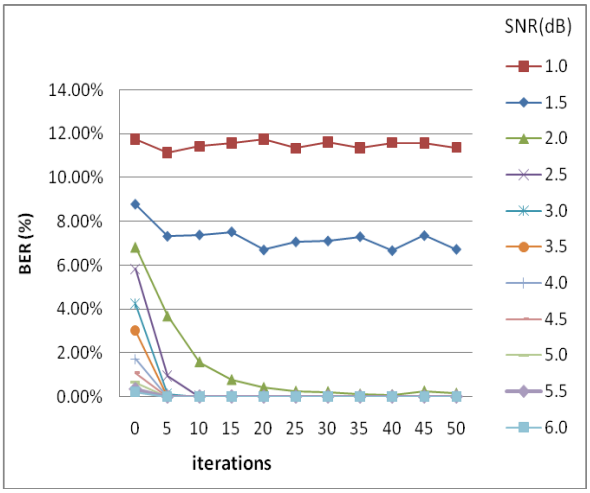
(d)

Figure 5.3.8 c) Showing Error correction performance of LDPC codes for block length =1024 bits, SPA-logdomain algorithm d) Showing Error correction performance of LDPC codes for block length =1024 bits and SPA-Min-Sum algorithm – logdomainSimple. The figures are plotted for number of iterations 0, 5 , 10, 15, 20 and SNR =1.0dB to 6.0dB.

From Figure 5.3.8 a), it can be seen that the BER percentage is about 11% at 1dB and 0 iterations of the decoding algorithm and is reduced to less than 9% at 20 iterations. The BER percentage reduces with increase in the SNR. For value of SNR greater than 2.5dB, the BER percentage approaches zero for number of iterations=10 to 20. Similar trend is found in Figures 5.3.8 b) , c) and d). Figure 5.3.8 c) and d) indicate that, at 1dB SNR, there is no considerable reduction in BER percentage, although the number of iterations are increased. Comparing Figures 5.3.8 a), b), c) and d), it can be seen that, for block length=512 and 1024 bits, SPA-logdomain algorithm has better error correction performance than SPA-Min-Sum algorithm – logdomainSimple.



(e)



(f)

Figure 5.3.8 e) Showing Error correction performance of LDPC codes for block length =2048 bits, SPA-logdomain algorithm f) Showing Error correction performance of LDPC codes for block length =2048 bits and SPA-Min-Sum algorithm – logdomainSimple. The figures are plotted for number of iterations 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50 and SNR =1.0dB to 6.0dB.

From Figure 5.3.8 e) and f), it is seen that the error correction performance considerably improves with increase in block length. Similarly, the performance of SPA-Min-Sum algorithm – logdomainSimple approaches that of SPA-logdomain algorithm for SNR greater than 2.5dB.

5.4 Conclusion

The thesis describes the designing and implementation of Gigabit Ethernet protocol testbed using FPGA platform and its Performance Analysis. The system is designed on Altera's Stratix II GX PCI Express Development Kit using Altera's TSE Megacore function. The logic utilization of the design is found to be 21% and the design is robust. The performance of the implemented design is studied for frame sizes from minimum 64 bytes to 9600 bytes. The line rate is found to be 76.19% for minimum 64 bytes frame length and approaches 100% for 9600 bytes. The throughput of Gigabit Ethernet protocol was found to be 7.62E+08 bps for 64 bytes frame and approaches 1Gbps for 9600 frame length, which is tested for different physical media. Thus, we can conclude that the throughput of short length frames is smaller because of the overhead of short frames. The thesis also addresses the complexities involved in the implementation of the said design.

Also a Simulation model in Matlab was developed for Error Detection Analysis using IEEE-802 CRC polynomial and Error Correction Analysis using LDPC codes. The Gigabit Ethernet testbed using Altera FPGA is interfaced with Matlab Simulation model, which was used to study the BER performance in a noisy Gigabit Ethernet channel. A testbed is developed for introducing optical attenuation for single mode optic fiber channel and the packet loss was studied. The results for CRC32 Error Detection algorithm in a BSC for Gigabit Ethernet frame lengths from 64 bytes to 1518 bytes is presented. The error correction performance using rate 1/2 regular LDPC codes in an AWGN channel is studied elaborately for block lengths of 256 and 512 bits for decoding iterations of 20 to 300 and SNRs 1dB to 6dB. The results for BER performance using LDPC codes is presented for Gigabit Ethernet frame lengths of 64, 128 and 256 bytes respectively. Simulation studies of LDPC codes required a lot of computing power as well as time for increased block lengths and increased number of iterations. Also, the study needs to be repeated for different data sets to observe whether the same performance is repeated.

5.5 Scope for future work

The Gigabit Ethernet design implemented on FPGA can be further extended to include 10Gbps standard. The simulation models developed in Matlab can be used to enhance the understanding of error detection and error correction algorithms. The FPGA design, interfaced with the Matlab model can be used to perform analysis on the actual Ethernet frames generated in a network. With the increasing speed and bandwidth requirement and the use of wireless and optical communication systems, present day computer networks have been found to be increasingly error-prone, which has necessitated the study of error correction algorithms.

Presently, there is an increasing research being carried out in the area of LDPC codes, with the help of embedded system platforms, where the computation speed is greatly improved compared to software simulation platforms. Hence, there is lot of potential to develop LDPC algorithms, implement them and study their performance for different networking protocols. The developed system can also be extended to a larger computer network and the network performance issues such as throughput, latency, flow control, congestion control and packet loss in a real network environment can be studied.

Appendix I

Ethernet Packet Generator Register Map

Address Offset	Name	Description
0x00	number_packet	32-bit Number of Packets Register.
0x04	config_setting	16-bit Configuration Setting Register.
0x08	rand_seed0	32-bit Lower Random Seed Register. Used to preload bits 0 to 31 of the PRBS generator when config_setting[15] is set to 1.
0x0C	rand_seed1	16-bit Upper Random Seed Register. Used to preload bits 32 to 47 of the PRBS generator when config_setting[15] is set to 1. Bits 16 to 31 are reserved.
0x10	source_addr0	32-bit Lower Source MAC Address Register. Used to define bits 0 to 31 of the source MAC address field of the Ethernet packet.
0x14	source_addr1	16-bit Upper Source MAC Address Register.
0x18	destination_addr0	32-bit Lower Destination MAC Address Register.
0x1C	destination_addr1	16-bit Upper Destination MAC Address Register.
0x20	operation	3-bit Operation Register
0x24	packet_tx_count	32-bit Packet Transmit Count Register.

config_setting Register Bit Descriptions

Bit(s)	Bit Name	Description
0	length_sel	Packet Length Type Select. 0: Fixed 1: Random
14:1	pkt_length	Fixed Packet Length. Programmable packet length ranges from 24–9600. This field is only valid when the Packet Length Type Select bit is set to 0.
15	pattern_sel	Data Type Select. 0: Incremental 1: Random

31:16	reserved	Reserved bits. Reads return 0.

operation Register Bit Descriptions

Bit(s)	Bit Name	Description
0	start	Start Operation. 1: Start the Ethernet Packet Generation
1	stop	Stop Operation. 1: Stop the Ethernet Packet Generation
2	tx_done	Transmit Done Status. set to 1
31:3	reserved	Reserved bits. Reads return 0

Ethernet Packet Monitor Register Map

Address Offset	Name	Description
0x00	number_packet	32-bit Number of Packets Register.
0x04	packet_rx_ok	32-bit Packet Received OK Register.
0x08	packet_rx_error	32-bit Packet Received with Error Register.
0x0C	byte_rx_count0	32-bit Lower Byte Received Count Register.
0x10	byte_rx_count1	32-bit Upper Byte Received Count Register.
0x14	cycle_rx_count0	32-bit Lower Cycle Receive Count Register.
0x18	cycle_rx_count1	32-bit Upper Cycle Receive Count Register.
0x1C	receive_ctrl_status	10-bit Receive Control and Status Register.

Receive Control and Status Register Bit Descriptions

Bit(s)	Bit Name	Description
0	start	Start Operation. 1 to start the Packet reception.

1	stop	Stop Operation. 1 to stop the Ethernet Packet Monitor from receiving further packets.
2	rx_done	Receive Done Status.
3	crcbad	CRC Error Packet. set to 1 if error in packet.
9:4	rx_err	Receive Error Status. The rx_err [5:0] signals from the receive FIFO interface.
31:10	reserved	Reserved bits. Reads return 0.

Appendix II

Matlab code for CRC32 Error Detection Analysis in BSC channel

```
clc
%  totfrm=input('Input number of frames :');
totfrm=100;
errcnt=0;
 for i=1:totfrm %for1
 len=60;                                  % set the frame length
 msg=randint(len,8);
 [crc_out]=computeCrc(len,msg)     % compute CRC32 for particular Frame length
 End          %end1
 %Receiver end
fid = fopen('frame1.txt');
%A = fscanf(fid, '%d', [8 64])
y=((totfrm)*(len+4))/4;
[A,count] = fscanf(fid, '%d', [32 y]);
A
A=A'                %     16r 32c;
fclose(fid);
fid = fopen('frame1.txt');
x=(len+4)*totfrm;
T = fscanf(fid, '%d', [8 x]);
T=T';
R=bsc(T,.001);                        % introduce noise in binary symmetric channel
 [errno]=checkCrc(len,R,A,totfrm)
```

```
fclose(fid);


[num1, rat1] = biterr(R,T)       % calculate bit error rate “rat1” in the received frame
per=errno/totfrm                      % calculate packet error rate
fid = fopen('result2.txt','a+');
fprintf(fid,'%d %d %d %f %f',len,errno,num1,rat1,per);  % print result in the file
fclose(fid);
delete('frame1.txt');
Function to compute CRC32 to append at the end of Ethernet Frame
function [crc_out] = computeCrc(len,msg)
fid=fopen('frame1.txt','a+');      % read the frame to be transmitted
for i=1:len%for2 change 60 to 4
   i;
   msg(i,:);
   fprintf(fid,'%d ',msg(i,:));
   for k=1:8
                                rmsg(i,k) = msg(i,9- k) ;
   end
      rmsg(i,:);
    end                 %end2
 crc_reg=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
next_crc=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
 for i=1:len  %change 60 to 4
 d=rmsg(i,:);
    next_crc(1) = xor(xor(xor(d(7),d(1)),crc_reg(25)),crc_reg(31));
```

```
   next_crc(2) =
xor(xor(xor(xor(xor(xor(xor(d(8),d(7)),d(2)),d(1)),crc_reg(25)),crc_reg(26)),crc_reg(31)),crc_reg(32));

   next_crc(3)= xor(xor(xor(xor(xor(xor(xor(xor(d(8),d(7)),d(3)),d(2)),d(1)),crc_reg(25)),
crc_reg(26)),crc_reg(27)),crc_reg(31)),crc_reg(32));

   next_crc(4) = xor(xor(xor(xor(xor(xor(xor(d(8),d(4)),d(3)),d(2)),crc_reg(26)),crc_reg(27)),
crc_reg(28)),crc_reg(32));

   next_crc(5) = xor(xor(xor(xor(xor(xor(xor(xor(d(7),d(5)),d(4)),d(3)),d(1)),crc_reg(25)),

crc_reg(27)), crc_reg(28)),crc_reg(29)),crc_reg(31));

   next_crc(6) = xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(d(8),d(7)),d(6)),d(5)),d(4)),d(2)),d(1)),

crc_reg(25)),crc_reg(26)),crc_reg(28)),crc_reg(29)),crc_reg(30)),crc_reg(31)),crc_reg(32));

   next_crc(7) = xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(d(8),d(7)),d(6)),d(5)),d(3)),d(2)),crc_reg(26)),

crc_reg(27)),crc_reg(29)),crc_reg(30)),crc_reg(31)),crc_reg(32));

   next_crc(8) = xor(xor(xor(xor(xor(xor(xor(xor(d(8),d(6)),d(4)),d(3)),d(1)),crc_reg(25)),

crc_reg(27)),crc_reg(28)),crc_reg(30)),crc_reg(32));

   next_crc(9) = xor(xor(xor(xor(xor(xor(xor(d(5),d(4)),d(2)),d(1)),crc_reg(1)),crc_reg(25)),

crc_reg(26)),crc_reg(28)),crc_reg(29));

   next_crc(10) = xor(xor(xor(xor(xor(xor(xor(d(6),d(5)),d(3)),d(2)),crc_reg(2)),crc_reg(26)),

crc_reg(27)),crc_reg(29)),crc_reg(30));

   next_crc(11) = xor(xor(xor(xor(xor(xor(xor(d(6),d(4)),d(3)),d(1)),crc_reg(3)),crc_reg(25)),

crc_reg(27)),crc_reg(28)),crc_reg(30));

   next_crc(12) = xor(xor(xor(xor(xor(xor(xor(d(5),d(4)),d(2)),d(1)),crc_reg(4)),crc_reg(25)),

crc_reg(26)),crc_reg(28)),crc_reg(29));

   next_crc(13) = xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(d(7),d(6)),d(5)),d(3)),d(2)),

d(1)),crc_reg(5)),crc_reg(25)),crc_reg(26)),crc_reg(27)),crc_reg(29)),crc_reg(30)),crc_reg(31));

   next_crc(14) = xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(d(8),d(7)),d(6)),d(4)),d(3)),d(2)),

crc_reg(6)),crc_reg(26)),crc_reg(27)),crc_reg(28)),c90rc_reg(30)),crc_reg(31)),crc_reg(32));
```

```
    next_crc(15) = xor(xor(xor(xor(xor(xor(xor(xor(xor(d(8),d(7)),d(5)),d(4)),d(3)),crc_reg(7)),
crc_reg(27)),crc_reg(28)),crc_reg(29)),crc_reg(31)),crc_reg(32));
    next_crc(16) = xor(xor(xor(xor(xor(xor(xor(xor(d(8),d(6)),d(5)),d(4)),crc_reg(8)),crc_reg(28)),
crc_reg(29)),crc_reg(30)),crc_reg(32));
    next_crc(17) = xor(xor(xor(xor(xor(xor(d(6),d(5)),d(1)),crc_reg(9)),crc_reg(25)),
crc_reg(29)),crc_reg(30));
    next_crc(18) = xor(xor(xor(xor(xor(xor(d(7),d(6)),d(2)),crc_reg(10)),crc_reg(26)),
crc_reg(30)),crc_reg(31));
   next_crc(19) =
xor(xor(xor(xor(xor(d(8),d(7)),d(3)),crc_reg(11)),crc_reg(27)),crc_reg(31)),crc_reg(32));
    next_crc(20) = xor(xor(xor(xor(d(8),d(4)),crc_reg(12)),crc_reg(28)),crc_reg(32));
    next_crc(21) = xor(xor(d(5),crc_reg(13)),crc_reg(29));
    next_crc(22) = xor(xor(d(6),crc_reg(14)),crc_reg(30));
    next_crc(23) = xor(xor(d(1),crc_reg(15)),crc_reg(25));
    next_crc(24) =
xor(xor(xor(xor(xor(xor(d(7),d(2)),d(1)),crc_reg(16)),crc_reg(25)),crc_reg(26)),crc_reg(31));
    next_crc(25) =
xor(xor(xor(xor(xor(xor(d(8),d(3)),d(2)),crc_reg(17)),crc_reg(26)),crc_reg(27)),crc_reg(32));
    next_crc(26) = xor(xor(xor(xor(d(4),d(3)),crc_reg(18)),crc_reg(27)),crc_reg(28));
    next_crc(27) = xor(xor(xor(xor(xor(xor(xor(d(7),d(5)),d(4)),d(1)),crc_reg(19)),crc_reg(25)),
crc_reg(28)),crc_reg(29)),crc_reg(31));
    next_crc(28) = xor(xor(xor(xor(xor(xor(xor(d(8),d(6)),d(5)),d(2)),crc_reg(20)),crc_reg(26)),
crc_reg(29)),crc_reg(30)),crc_reg(32));
    next_crc(29) = xor(xor(xor(xor(xor(xor(d(7),d(6)),d(3)),crc_reg(21)),crc_reg(27)),crc_reg(30)),
crc_reg(31));
    next_crc(30) = xor(xor(xor(xor(xor(xor(d(8),d(7)),d(4)),crc_reg(22)),crc_reg(28)),
```

```
crc_reg(31)),crc_reg(32));
   next_crc(31) = xor(xor(xor(xor(d(8),d(5)),crc_reg(23)),crc_reg(29)),crc_reg(32));
   next_crc(32) = xor(xor(d(6),crc_reg(24)),crc_reg(30));
   crc_reg=next_crc;
 end
 for j=1:32 % invert
                                %crc_out(j) =~crc_reg(33-j);
         crc_out(j) =~crc_reg(j);
 end
 fprintf(fid,'%d ',crc_out);
 fclose(fid);
 Matlab Function to check  CRC32 of Received Ethernet Frame
function [errno] = checkCrc(len,R,A,totfrm)
errno=0;
d=0;
j=1;
% for j=1:x %for x ch 60 to 4
%    j
for i=1:totfrm
   i;
   for l=j:len+j-1 %for len
          R(l,:);
      for k=1:8
                                rR(l,k) = R(l,9- k) ;
   end
```

```
      rR(l,:);
      end     %endl
 crc_regM=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
next_crcM=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
 for l=j:len+j-1% ch 60 to 4
 dM=rR(l,:);
    next_crcM(1) = xor(xor(xor(dM(7),dM(1)),crc_regM(25)),crc_regM(31));
   next_crcM(2) = xor(xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(2)),dM(1)),crc_regM(25)),
crc_regM(26)),crc_regM(31)),crc_regM(32));
   next_crcM(3) = xor(xor(xor(xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(3)),dM(2)),dM(1)),
crc_regM(25)),crc_regM(26)),crc_regM(27)),crc_regM(31)),crc_regM(32));
   next_crcM(4) = xor(xor(xor(xor(xor(xor(xor(dM(8),dM(4)),dM(3)),dM(2)),crc_regM(26)),
crc_regM(27)),crc_regM(28)),crc_regM(32));
   next_crcM(5) = xor(xor(xor(xor(xor(xor(xor(xor(dM(7),dM(5)),dM(4)),dM(3)),dM(1)),
crc_regM(25)),crc_regM(27)),crc_regM(28)),crc_regM(29)),crc_regM(31));
   next_crcM(6) = xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(6)),dM(5)),
dM(4)),dM(2)),dM(1)),crc_regM(25)),crc_regM(26)),crc_regM(28)),crc_regM(29)),crc_regM(30)),crc_reg
M(31)),crc_regM(32));
   next_crcM(7) = xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(6)),dM(5)),dM(3)),
dM(2)),crc_regM(26)),crc_regM(27)),crc_regM(29)),crc_regM(30)),crc_regM(31)),crc_regM(32));
   next_crcM(8) = xor(xor(xor(xor(xor(xor(xor(xor(xor(dM(8),dM(6)),dM(4)),dM(3)),dM(1)),
crc_regM(25)),crc_regM(27)),crc_regM(28)),crc_regM(30)),crc_regM(32));
   next_crcM(9) = xor(xor(xor(xor(xor(xor(xor(xor(dM(5),dM(4)),dM(2)),dM(1)),crc_regM(1)),
crc_regM(25)),crc_regM(26)),crc_regM(28)),crc_regM(29));
   next_crcM(10) = xor(xor(xor(xor(xor(xor(xor(xor(dM(6),dM(5)),dM(3)),dM(2)),crc_regM(2)),
crc_regM(26)),crc_regM(27)),crc_regM(29)),crc_regM(30));
```

```
    next_crcM(11) = xor(xor(xor(xor(xor(xor(xor(dM(6),dM(4)),dM(3)),dM(1)),crc_regM(3)),
crc_regM(25)),crc_regM(27)),crc_regM(28)),crc_regM(30));
    next_crcM(12) = xor(xor(xor(xor(xor(xor(xor(dM(5),dM(4)),dM(2)),dM(1)),crc_regM(4)),
crc_regM(25)),crc_regM(26)),crc_regM(28)),crc_regM(29));
    next_crcM(13) = xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(dM(7),dM(6)),dM(5)),dM(3)),
dM(2)),dM(1)),crc_regM(5)),crc_regM(25)),crc_regM(26)),crc_regM(27)),crc_regM(29)),crc_regM(30)),c
rc_regM(31));
    next_crcM(14) =
xor(xor(xor(xor(xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(6)),dM(4)),dM(3)),dM(2)),
crc_regM(6)),crc_regM(26)),crc_regM(27)),crc_regM(28)),crc_regM(30)),crc_regM(31)),crc_regM(32));
    next_crcM(15) = xor(xor(xor(xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(5)),dM(4)),dM(3)),
crc_regM(7)),crc_regM(27)),crc_regM(28)),crc_regM(29)),crc_regM(31)),crc_regM(32));
    next_crcM(16) = xor(xor(xor(xor(xor(xor(xor(dM(8),dM(6)),dM(5)),dM(4)),crc_regM(8)),
crc_regM(28)),crc_regM(29)),crc_regM(30)),crc_regM(32));
    next_crcM(17) =
xor(xor(xor(xor(xor(dM(6),dM(5)),dM(1)),crc_regM(9)),crc_regM(25)),crc_regM(29)),crc_regM(30));
    next_crcM(18) = xor(xor(xor(xor(xor(xor(dM(7),dM(6)),dM(2)),crc_regM(10)),
crc_regM(26)),crc_regM(30)),crc_regM(31));
    next_crcM(19) = xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(3)),crc_regM(11)),crc_regM(27)),
crc_regM(31)),crc_regM(32));
    next_crcM(20) = xor(xor(xor(xor(dM(8),dM(4)),crc_regM(12)),crc_regM(28)),crc_regM(32));
    next_crcM(21) = xor(xor(dM(5),crc_regM(13)),crc_regM(29));
    next_crcM(22) = xor(xor(dM(6),crc_regM(14)),crc_regM(30));
    next_crcM(23) = xor(xor(dM(1),crc_regM(15)),crc_regM(25));
    next_crcM(24) = xor(xor(xor(xor(xor(xor(dM(7),dM(2)),dM(1)),crc_regM(16)),crc_regM(25)),
crc_regM(26)),crc_regM(31));
```

```
   next_crcM(25) = xor(xor(xor(xor(xor(xor(dM(8),dM(3)),dM(2)),crc_regM(17)),crc_regM(26)),
crc_regM(27)),crc_regM(32));
   next_crcM(26) = xor(xor(xor(xor(dM(4),dM(3)),crc_regM(18)),crc_regM(27)),crc_regM(28));
   next_crcM(27) = xor(xor(xor(xor(xor(xor(xor(dM(7),dM(5)),dM(4)),dM(1)),crc_regM(19)),
crc_regM(25)),crc_regM(28)),crc_regM(29)),crc_regM(31));
   next_crcM(28) = xor(xor(xor(xor(xor(xor(xor(dM(8),dM(6)),dM(5)),dM(2)),crc_regM(20)),
crc_regM(26)),crc_regM(29)),crc_regM(30)),crc_regM(32));
   next_crcM(29) = xor(xor(xor(xor(xor(xor(dM(7),dM(6)),dM(3)),crc_regM(21)),crc_regM(27)),
crc_regM(30)),crc_regM(31));
   next_crcM(30) = xor(xor(xor(xor(xor(xor(dM(8),dM(7)),dM(4)),crc_regM(22)),crc_regM(28)),
crc_regM(31)),crc_regM(32));
   next_crcM(31) = xor(xor(xor(xor(dM(8),dM(5)),crc_regM(23)),crc_regM(29)),crc_regM(32));
   next_crcM(32) = xor(xor(dM(6),crc_regM(24)),crc_regM(30));
   crc_regM=next_crcM;
end %endl
 for m=1:32 % invert
                                %crc_out(j) =~crc_regM(33-j);
        crc_outM(m) =~crc_regM(m);
 end
       d=d+(len/4)+1;
        if isequal(A(d,:),crc_outM)
        e='0';          crc_outM;
        else
        errno=errno+1;     end
    j=j+len+4;   end    %totfrm loop
```

REFERENCES

1. Meixia Duan, Hongling Han, Research and Implementation of Gigabit Ethernet Full Line Rate, Cross Strait Quad-Regional Radio Science and Wireless Technology Conference, 2011, pp. 736-738.
2. Gigabit Ethernet Technology and Solutions, Copyright © 2001, Intel Corporation.
3. Gerd Keiser, Local Area Network, Tata McGraw-Hill , Second Edition, 2002.
4. http://support.microsoft.com/kb/103884
5. Behrouz A. Forouzan, Data Communication and Networking, Tata McGraw-Hill , 4th Edition, 2008.
6. http://www.ieee802.org/dots.shtml
7. Fundamentals of Ethernet: 10 Megabit Ethernet to 10 Gigabit Ethernet , JDSU, (www.jdsu.com/test).
8. Robert Metcalfe, David Boggs, Ethernet: Distributed packet switching for local computer networks, Association for Computing Machinery, Vol.19, No.5, July 1976.
9. http://timeline.ethernethistory.com.
10. Phil Edholm, Paul Littlewood, Next-generation Ethernet:The key to infrastructure transition, Nortel Technical Journal, Issue 4, pp. 7-24.
11. http://archiv.cesnet.cz/doc/techzpravy/2010/100ge-study/#phys-layer
12. Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Std 802.3™ 2008.
13. William Stallings, Gigabit Ethernet, The Internet Protocol Journal – Vol. 2, No. 3,September 1999, pp. 21-25.
14. AnySpeed Ethernet MAC Core Product Brief Version 1.0, August 2005, 1, MorethanIP GmbH, Muenchner Str. 199, 85757 Karlsfeld, Germany.
15. Alan F. Benner, Petar K. Pepeljugoski, Renato J. Recio, A Roadmap To 100g Ethernet At The Enterprise Data Center, IBM Corp, IEEE Applications & Practice, November 2007, pp. 10-17.
16. Neil Mcallister, Networking Industry To Collaborate On Terabit Ethernet, 20 Aug 2012, (http://www.theregister.co.uk/data_centre/hpc/).
17. Wayne Rash, Terabit Ethernet Is in Your Future, March 26, 2010, (http://www.eweek.com/networking/terabit-ethernet-is-in-your-future/#sthash.lM6Sr5Y4. dpuf).
18. Oleh et al., Terabit Ethernet: Ambitious or Reality For High Speed Network At Future, 2010, (http://www.ptmk.ump.edu.my › Artikel ICT › Rangkaian).
19. Finkler, S. Sidhu, D., Performance analysis of IEEE 802.3z Gigabit Ethernet Standard, Global Telecommunications Conference, 1999, GLOBECOM '99 , Vol. 2, pp. 1302-1306.
20. Tinoosh Mohsenin , Thesis: Design and Evaluation of FPGA-Based Gigabit-Ethernet/PCI Network Interface Card, Rice University, 2004.
21. A. Chaubal, Thesis: Design and Implementation of an FPGA-based Partially Reconfigurable Network Controller, Bradley Department of Electrical and Computer Engineering Blacksburg, Virginia, 2004.
22. M. Ciobotaru, et al., Versatile FPGA-based Hardware Platform for Gigabit Ethernet Applications, 6th Annual Postgraduate Symposium, Liverpool, UK, June 2005.

23. A. Bianco et al., Boosting the performance of PC-based softwares routers with FPGA-enhanced line cards, Politecnico di Torino, 10129 Torino.
24. J. Shafer, S. Rixner, RiceNIC: A reconfigurable net-work interface for experimental research and education, Proc. of the Workshop on Experimental Computer Science (ExpCS'07), June 2007.
25. Zou et al., Co-Design For An SoC Embedded Network Controller, Journal of Zhejiang University, Science A, ISSN 1009-3095 (Print), 2006, pp. 591-596.
26. Dilip Thomas, K.S. Mohanachandra Panicker, VLSI Implementation Of Gigabit Ethernet With Data Compression And Decompression, IET-UK International Conference on Information and Communication Technology in Electrical Sciences (ICTES 2007), M.G.R. University, Chennai, Tamil Nadu, India, December 2007, pp.826-826.
27. Dave Bailey, Richard Hughes-Jones, Marc Kelly,Using FPGAs to Generate Gigabit Ethernet Data Transfers and Studies of the Network Performance of DAQ Protocols, The University of Manchester, UK. , (http://www.docstoc.com/docs/38636314/Using-FPGAs-to-Generate-Gigabit).
28. G. A. Covington, G. Gibb, J. Lockwood, N. McKeown, A packet generator on the netfpga platform, Proc. In The 17th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, April 2009.
29. Nicholas Tsakiris, Greg Knowles, Gigabit IP Core for Embedded Systems, International Journal Of Circuits, Systems And Signal Processing, Vol.2, No.2, 2008 p. 347.
30. T. V. Ramabadran, S. S. Gaitonde, A Tutorial on CRC Computations, IEEE Micro, Vol.8, No. 4, August 1988, pp. 62-75.
31. G. Albertango, R. Sisto, Parallel CRC Generation, IEEE Micro, Vol. 10, No. 5, October 1990, pp. 63-71.
32. T. B. Pei, C. Zukowski, High-speed parallel CRC circuits in VLSI, IEEE Transactions on Communications, Vol. 40, No. 4, Apr. 1992, pp. 653 - 657.
33. G. Castagnoli, S. Brauer, M. Herrmann, Optimization of cyclic redundancy-check codes with 24 and 32 parity bits, IEEE Transactions on Communications, Vol. 41, No.6, June 1993, pp. 883 -892.
34. Sait S.M., Tanvir, M.S.K., VLSI Layout Generation Of A Programmable CRC Chip, IEEE Transactions on Consumer Electronics, Vol.39, No.4, November 1993, pp. 911 – 916.
35. U. Nordqvist, T. Henriksson, D. Liu, CRC Generation for Protocol Processing, Norchip, 2000, Turku, Finland, pp. 288-293.
36. A. Perez, Byte-wise CRC Calculations, IEEE Micro, Vol. 3, No. 3, June 1983, pp. 40-50.
37. Ming-Der Shieh, Ming-HwaSheu, Chung-Ho Chen, Hsin-Fu Lo, A Systematic Approach for Parallel CRC Computations, Journal Of Information Science And Engineering, Vol. 17, 2001, pp. 445-461.
38. R. Nair, G. Ryan, F. Farzaneh, A Symbol Based Algorithm for Implementation of Cyclic Redundancy Check (CRC), In Proc. of VHDL International Users' Forum, 1997, pp. 82 -87.
39. Tridib Chakravarty, Performance of Cyclic Redundancy Codes for Embedded Networks, MS Report, Dept. of Elect. & Comp. Engg., Carnegie Melon University, December 2001.
40. Koopman, P., 32-bit cyclic redundancy codes for Internet applications, Intl. Conf. Dependable Systems and Networks (DSN), Washington DC, July 2002, pp.459-468.
41. Giuseppe Campobello, Giuseppe Patane, Marco Russo, Parallel CRC Realization, IEEE Transactions On Computers, Vol. 52, No. 10, October 2003.
42. Philip Koopman, TridibChakravarty, Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks, The International Conference on Dependable Systems and Networks, DSN-2004, (http://users.ece.cmu.edu/~koopman/roses/dsn04/ koopman04_crc_poly_embedded.pdf).

43. Baicheva, T., S. Dodunekov, P. Kazakov, On The Cyclic Redundancy-Check Codes With 8-Bit Redundancy, Computer Communications, Vol. 21, 1998, pp. 1030-1033.
44. Chao Cheng, K.K. Parhi, High-Speed Parallel CRC Implementation Based On Unfolding, Pipelining and Retiming, IEEE Transactions On Circuits And Systems, Vol. 53, No. 10, Oct. 2006, pp. 1017 – 1021.
45. M. Walma, Pipelined Cyclic Redundancy Check (CRC) calculation, ICCCN 2007, In Proc. of 16th International Conference on Computer Communications and Networks, Honolulu, HI, USA, 2007, pp. 365 - 370.
46. Kazakov, P., Fast calculation of the number of minimum-weight words of CRC codes, IEEE Trans. Information Theory, Vol.47, No.3, March 2001, pp. 1190-1195.
47. H. Michael Ji., Earl Killian, Fast Parallel CRC Algorithm and Implementation on a Configurable Processor, ICC 2002, IEEE International Conference on Communications, Vol.3, 2002, pp. 1813 – 1817.
48. Tomas Henriksson, Dake Liu , Implementation of Fast CRC Calculation, Proceedings of the ASP-DAC 2003, Asia and South Pacific, Jan. 2003, pp. 563 – 564.
49. Tomas Henriksson, In-Line CRC Calculation and Scheduling for 10 Gigabit Ethernet Transmission, (http://www.da.isy.liu.se/pubs/tomhe/SSoCC2002_crc.pdf).
50. Ulf Nordqvist, Tomas Henrikson, Dake Liu, Configurable CRC Generator, Department of Electrical Engineering, Linköpings University Linköpings, Sweden, (http://www.da.isy.liu.se/ pubs/ulfnor/ulfnor-ddecs2002.pdf).

51. M. Braun, J. Freidich, T. Grun, J. Lembert, Parallel CRC computation in FPGAs, In Proc. of Workshop on Field Programmable Logic Application, 1996, pp. 156-165.
52. Yan Sun and Min Sue Kim, A table based algorithm for pipelined CRC calculation, In Proc. of Communications (ICC), 2010 IEEE International Conference, Cape Town, South Africa, 2010, pp. 1 - 5.
53. D. J. C. MacKay, R. M. Neal, Near Shannon limit performance of low density parity check codes, Electronics Letters, vol. 32, pp. 1645–1646, Aug. 1996. Reprinted Electronics Letters, Vol 33, No. 6, 13th March 1997, pp. 457–458
54. M. Davey , David J.C. Mackay, Low Density Parity Check Codes over GF(q), Cavendish Laboratory, Cambridge, UK, June 1998, (http://citeseerx.ist.psu.edu/viewdoc).
55. C. J. Howland, A. J. Blanksby, Parallel decoding architectures for low density parity check codes, In Proc. IEEE ISCAS, Vol. 4, May 2001, pp. 742–745.
56. R. Gallager, Low-density parity-check codes, IEEE Trans. Inf. Theory, Vol. IT-8, No. 1, Jan. 1962, pp. 21–28.
57. D. MacKay, R. Neal, Good codes based on very sparse matrices., IMA Conf. Cryptography and Coding, 1995, (http://www.cs.toronto.edu/~mackay/mnc4s.pdf).
58. D. MacKay, Good error correcting codes based on very sparse matrices, IEEE Trans. Information Theory, Vol. 47, No. 5, 2001, pp. 399-431.
59. B. Levine, R. Reed Taylor , Herman Schmit, Implementation of Near Shannon Limit Error-Correcting Codes Using Reconfigurable Hardware , In Proc. of IEEE Symposium on Field-Prog. Cust. Comput. Mach 2000, pp. 217-226.
60. T. J. Richardson, R. L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, IEEE Trans. Inf. Theory, Vol. 47, No. 2, Feb. 2001, pp. 599–618.
61. Tong Zhang, Keshab Parhi, Joint Code and decoder design for implementation-oriented (3,k) – regular LDPC codes, Published in Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on, Vol. 2, Nov. 2001, pp.1232-1236.

62. Yeo et al., High Throughput Low-Density Parity-Check Decoder Architectures, In Proc. of the IEEE GLOBECOM 2001, IEEE Press, pp. 3019 -3024.
63. Hisashi Futaki, Tomoaki Ohtsuki, Performance of Low-Density Parity-Check (LDPC) Coded OFDM Systems, Published in Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th, Vol.1, 2001, pp. 82-86.
64. Hisashi Futaki, Tomoaki Ohtsuki, Low-Density Parity-Check (LDPC) Coded OFDM Systems with M-PSK, Published in Vehicular Technology Conference, 2002, VTC Spring 2002, IEEE 55th, Vol.2, 2002, pp. 1035-1039.
65. Blanksby , Howland, A 690-mW 1-Gb/s 1024-b, Rate-1/2 Low-Density Parity-Check Code Decoder , IEEE J. Solid State Circuits, Vol. 37, No. 3, 2002, pp. 404–412.
66. Ki-Moon Lee , Hayder Radha, The Design of the Maximum-Likelihood Decoding Algorithm of LDPC Codes over BEC Dept. of Math, Michigan State University, (citeseerx.ist.psu.edu/viewdoc).
67. David Burshtein, Gadi Miller, An Efficient Maximum-Likelihood Decoding of LDPC Codes Over the Binary Erasure Channel, IEEE Trans. Inform. Theory, Vol.50, 2004, pp. 2837-2844.
68. H. Xiao, A. H. Banihashemi, Improved Progressive-Edge-Growth (PEG) Construction of Irregular LDPC Codes, IEEE Comm. Letters, Vol. 8, No. 12, December 2004, pp. 715-717.
69. Yang Sun, Marjan Karkooti, Joseph R. Cavallaro, High Throughput, Parallel, Scalable Ldpc Encoder/Decoder Architecture For Ofdm Systems, Published in Design, Applications, Integration and Software, IEEE Dallas/CAS Workshop on, October 2006, pp. 39-42.
70. K. Shuaib et al., Performance evaluation of IEEE 802.15.4: Experimental and Simulation Results, Journal of Communications, Vol. 2, No. 4, June 2007, pp. 29–37.
71. C. P. Fewer, M. F. Flanagan, A. D. Fagan, A versatile variable rate LDPC codec architecture, IEEE Transaction on Circuits Systems-I, Vol. 54, No.10, October 2007, pp. 2240–2251.
72. V.S. Ganepola, R.A. Carrasco , I. J. Wassell, Le Goff, Performance study of Non-binary LDPC Codes over GF(q), Sch. of Electr., Univ. of Newcastle, Newcastle, (http://www.cl.cam.ac.uk/research/dtg/publications/public/ic231/40862.pdf).
73. Hua Xiao, Amir H. Banihashemi, Estimation of Bit and Frame Error Rates of Finite-Length Low-Density Parity-Check Codes on Binary Symmetric Channels, IEEE Trans. in Communications, Vol. 55 , No. 12, Dec. 2007, pp. 2234 – 2239.
74. Hua et al., Estimation of Bit and Frame Error Rates of Finite-Length Low-Density Parity-Check Codes on Binary Symmetric Channels, June 2009, (citeseerx.ist.psu.edu/viewdoc).
75. Zhou Zhong et al., Wang Modified Min-sum Decoding Algorithm for LDPC Codes Based on Classified Correction, Published in Proceedings of ChinaCom Conference, Aug. 2008, pp. 932 – 936.
76. Syed Aziz, Mahfuzul, Minh Duc Pham, Duc, Implementation Of Low Density Parity Check Decoders Using A New High Level Design Methodology, Journal Of Computers, Vol. 5, No. 1, January 2010, pp. 81-90.
77. Md. Murad Hossain et al., Modified Log Domain Decoding Algorithm for LDPC Codes over GF (q), Journal of Selected Areas in Telecommunications (JSAT), June 2011, pp. 30-36.
78. Yeo et al., Architectures and Implementations of Low-Density Parity Check Decoding Algorithms , IEEE International Midwest Symposium on Circuits and Systems,Vol.3, August 2002, pp. 437-440.
79. Zhang et al., Design of LDPC Decoders for Low Error Rate Performance , Published in IEEE Transactions on Communications, Vol.57 , No. 11, Nov. 2009, pp. 3258 – 3268.
80. Yang et al., 428-Gb/s single-channel coherent optical OFDM transmission over 960-km SSMF with constellation expansion and LDPC coding, Optics Express, Vol. 18, No. 16, August 2010, pp. 16883-16889.

81. Zhang et al., Evaluation of four-dimensional nonbinary LDPC-coded modulation for next-generation long-haul optical transport networks, Optics Express, Vol. 20, No. 8, April 2012, pp. 9296-9301.
82. Tao Jin-jing et al., Application of LDPC codes in atmospheric optical communication with coherent detection, Optoelectronics letters, Vol. 9, No. 2, March 2013, pp. 132-134.
83. Arun Kumar, Rajendar Bahl, An Architecture for High data rate Very Low Frequency communication, Defence Science Journal, 2013, Vol. 63, No.1, pp.25-33.
84. Jianguo Yuan et al., A new construction method of LDPC codes for optical transmission systems, Frontiers of optoelectronics, Vol. 5, No.3, Sept. 2012, pp. 311-316.
85. Yuan Jianguo et al., A novel construction algorithm of the LDPC code for high-speed long-haul optical transmission systems, Optic International J. for Light & Electron Optics, Vol.124, No.18, September 2013, pp. 3181-3186.
86. L. B. James, A. W. Moore, M. Glick, Structured Errors in Optical Gigabit Ethernet, Passive and Active Measurement Workshop (PAM 2004), Antibes Juan-les-Pins, France, Published in book Passive and Active Network Measurement by Springer-Verlag Berlin Heidelberg, pp. 195-204.
87. L. B. James et al., Packet error rate and bit error rate non-deterministic relationship in optical network applications, Univ of Cambridge, UK, 2005, (www.cl. cam.ac.uk/~awm22/publications/james2005packet.pdf).
88. A. Xiang et al., Design and verification of an FPGA-based bit error rate tester , Proceedings in TIPP 2011, Published by Elsevier, 2012, pp. 1875-3892 doi: 10.1016/j.phpro.2012.02.492.
89. Frazier, Johnson, Gigabit Ethernet: From 100 to 1,000 Mbps, IEEE Internet Computing, Jan. 1999, (http://www.gigabit-ethernet.org).
90. Introduction to Gigabit Ethernet,Technology brief, Copyright © 2000 Cisco Systems, Inc.
91. http://opencores.org/project,ethernet_tri_mode.
92. M. Ciobotaru, et al., Versatile FPGA-based Hardware Platform for Gigabit Ethernet Applications, 6th Annual Postgraduate Symposium, Liverpool, UK, June 2005, (cern.ch/ciobota/papers/2005_getb_liverpool.pdf).
93. Zou et al., Co-design for an SoC embedded network controller, Journal of Zhejiang University SCIENCE A, 2006, pp. 591-596.
94. C. Kachris, et al., Design and performance evaluation of an adaptive FPGA for network applications, Microelectron. J , 2008, doi:10.1016/j.mejo.2008.05.01.
95. Stratix II GX Device Handbook, Volume 2 © 2007 Altera Corporation SIIGX5V2-4.3.
96. Triple Speed Ethernet Data Path Reference Design AN-483-June 2009 ver. 1.1 Altera Corporation.
97. Triple-Speed Ethernet MegaCore Function User Guide © December 2010 Altera Corporation
98. Stallings, William, Data and computer communications, Upper Saddle River, N.J. : Pearson/Prentice Hall, 8th Edition, 2007.
99. Palani Subbaiah, Bit- Error Rate for High Speed Serial Data Communication, Data-communications Division, Cypress Semiconductor, November 2008, (www.pdfgeni.me/pdf/b6a80fe2e0)
100. ATM Shafiul Alam, Effect of Additive White Gaussian Noise (AWGN) on the Transmitted Data, Department of Electrical, Computer and Communications Engineering, London South Bank University , Dec 2008, (atmshafiulalam.webs.com/ IDC-Ex-3-additive_white_Gaussian_noise.pdf).
101. http://en.wikipedia.org/wiki/Binary_symmetric_channel

102.http:// www.dsplog.com/2007/08/05/bit-error-probability-for-bpsk-modulation/
103.Peterson, W. & E. Weldon, Error-Correcting Codes, Second Edition, MIT Press, 1972.
104.Ulf Nordqvist, Thesis: Protocol Processing in Network Terminals, Department of Electrical Engineering , Linkopings University, SE-581 83 Linkoping , Sweden 2004.
105.Koopman, 32-bit cyclic redundancy codes for Internet applications, Intl. Conf. Dependable Systems and Networks (DSN), 2002, pp.459-468.
106.V. R. Gad, R. S. Gad, G. M. Naik, Implementation of Gigabit Ethernet using Double CRC-32 technique, Symposium on VLSI & Embedded System, Goa University & VSI , Goa Chapter, Feb 2010.
107.U. Nordqvist, T. Henriksson, D. Liu, CRC Generation for Protocol Processing, Norchip 2000, Turku, Finland, pp. 288-293.
108.Ming-Der Shieh, Ming-Hwa Sheu, Chung-Ho Chen, Hsin-Fu Lo , A Systematic Approach for Parallel CRC Computations, Journal Of Information Science And Engineering, Vol. 17, 2001 pp. 445-461.
109.Giuseppe Campobello, Giuseppe Patane,Marco Russo , Parallel CRC Realization, IEEE Transactions On Computers, Vol. 52, No. 10, 2003, pp. 245-256.
110.Luben, Cavannay, The iSCSI CRC32C Digest and the Simultaneous Multiply and Divide Algorithm, USA ,Jan. 2002, (www.research.ibm.com/haifa/satran/ips/Vince-Luben-crc32c-01.pdf) .
111.Luben, Cavannay, The iSCSI CRC32C Digest and the Simultaneous Multiply and Divide Algorithm, USA , Jan. 2002, (www.research.ibm.com/haifa/satran/ips/Vince-Luben-crc32c-01.pdf).
112.Kounavis, M.E., Berry, F.L., Novel Table Lookup-Based Algorithms for High-Performance CRC Generation, IEEE Transactions on Computers, Vol. 57, No. 11, 2008, pp. 1550-1560.
113.Gam D. Nguyen , Fast CRCs, IEEE Transactions On Computers, Vol. 58, No. 10, Oct. 2009, pp. 1321-1331.
114.Ulf Nordqvist, Thesis: Protocol Processing in Network Terminals, Department of Electrical Engineering , Linkopings University, SE-581 83 Linkoping , Sweden 2004.
115.Ji. H. Michael, Killian E., Fast parallel CRC algorithm and implementation on a configurable processor, IEEE Int. Conf. on Communications, ICC 2002, Vol. 3, 2002, pp. 1813-1817.
116.Weidong Lu, Stephan Wong, A Fast CRC Update Implementation, The Netherlands International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 3, No.1, October 2013, pp. 113-120.
117.Grymel, Furber, A Novel Programmable Parallel CRC Circuit, IEEE Transactions on VLSI Systems , Vol. 19, No.10, 2011, pp. 1898-1902.
118.Toal et al., Design and Implementation of a Field Programmable CRC Circuit Architecture, IEEE Transactions on VLSI Systems, Vol.17, No. 8, 2009, pp. 1142-1147.
119. James E. Gilley, Bit-Error-Rate Simulation Using Matlab, Transcrypt International, Inc., 2003, (wr.lib.tsinghua.edu.cn/sites/default/files/1195180204498.pdf).
120.Ivan B. Djordjevic, Bane Vasic, LDPC-coded OFDM in fiber-optics communication systems, J. Of Optical Networking, Vol. 7, No. 3, March 2008, pp. 217-226.
121.Chin-Kuang Lian, A partially parallel LDPC decoder with reduced memory for long code length, Dept. of Elect. Engg., National Central University, , Taiwan, (www.researchgate.net).
122.D. MacKay, R. Neal, Good codes based on very sparse matrices., IMA Conf. Cryptography and Coding, 1995,(http://www.cs.toronto.edu/~mackay/mnc4s.pdf).
123.D. MacKay, Good error correcting codes based on very sparse matrices, IEEE Trans. Information Theory, Vol. 47, No. 5, 2001, pp. 399-431.
124.N. Alon ,M. Luby, A linear time erasure-resilient code with nearly optimal recovery, IEEE Trans. Inf. Theory, Vol. 42, No. 6, Nov. 1996, pp. 1732–1736.

125.T. J. Richarson, R. L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, IEEE Trans. Inf. Theory, Vol. 47, No. 2, Feb. 2001, pp. 599–618.
126.T. J. Richarson, M. A. Shokrollahi, R. L. Urbanke, Design of capacity-approaching irregular low-density parity-check codes, IEEE Trans. Inf. Theory, Vol. 47, No. 2, Feb. 2001, pp. 619–637.
127.S. Chung, G. Forney, T. Richardson, R. Urbanke, On the design of low-density parity-check codes within 0.0045 db of the Shannon limit, IEEE Commun. Lett., Vol. 5, No. 2, Feb. 2001, pp. 58–60.
128.Lin ,Costello, Error Control Coding: Fundamentals and Applications, New Jersey: Prentice Hall, 2nd Edition, 2004.
129.B. Reiffen, Sequential Decoding for Discrete Input Memoryless Channels, IRE Trans. Inf. Theory, Vol. 8, No. 3, April 1962, pp. 208-220.
130.A. J. Blanksby , C. J. Howland, A 690- mW 1-Gb/s 1024-b, rate-1/2 low-density parity check code decoder, IEEE J. Solid State Circuits, Vol. 37, No. 3, 2002, pp. 404–412.
131.Bernhard M. J. Leiner, LDPC codes - a Brief Tutorial, April 2005 (www.bernh.net/media/download/papers/ldpc.pdf).
132.Eckford, Kschischang, Pasupathy, Analysis of Low-density Parity-check Codes for the Gilbert-Elliott Channel, IEEE Trans. Inf. Theory, Vol. 51, No. 11, Nov. 2005, pp. 3872–3889.
133.Vijay Nagarajan, Stefan Laendner, Olgica Milenkovic, High-throughput VLSI Implementations of Iterative Decoders and Related Code Construction Problems, Journal of VLSI Signal Processing, Vol. 49, 2007, pp. 185–206.
134.Marjan Karkooti, Thesis: Semi-Parallel Architectures for Real-time LDPC encoding, Houston, Texas, 2004.
135.L. Ping, W.K. Leung, Decoding low density parity check codes with finite quantization bits, IEEE Comm. Letters, Vol. 4, No.2, Feb. 2000, pp. 62–64.
136.H. Wymeersch, H. Steendam and M. Moeneclaey, Computational complexity and quantization effects of decoding algorithms of LDPC codes over GF(q), In Proc. ICASSP, Montreal, Canada, May 2004, pp. 772-776.
137.X. Hu, E. Eleftheriou, D.-M. Arnold , A. Dholakia, Efficient implementations of the sum-product algorithm for decoding LDPC codes, In Proc. IEEE Globecom, San Antonio, TX, Vol. 2, November 2001, pp. 1036-1036E.
138.M. Davey , David J.C. Mackay, Low Density Parity Check Codes over GF(q), Cavendish Laboratory, Cambridge, UK, June 1998, (http://citeseerx.ist.psu.edu/viewdoc).
139. J. Berkmann, On turbo decoding of nonbinary codes, IEEE Comm. Letters, Vol. 2, No. 4, April 1998, pp. 94–96.
140.H Wymeersch, H Steendam, M Moeneclaey, Log-domain decoding of LDPC codes over GF(q), , In Proc. IEEE Intern. Conf. on Commun., Vol. 2 , No.2, June 2004, pp. 772-776.
141.Dan Dechene, Kevin Peets, Thesis: Simulated Performance of Low –Density Parity-Check Codes: A Matlab implementation, Lakehead University, Faculty of Engineering, 2006.
142.www.mathworks.in/matlabcentral/.../8977-ldpc-code-simulation.
143.W. E. Ryan, An Introduction to LDPC Codes, in CRC Handbook for Coding and Signal Processing for Recording Systems (B. Vasic, ed.) , CRC Press, 2004.
144.10 Gigabit Ethernet Technology Overview and Applications for Enterprise Data Centers, Blade network technologies, ©2009 BLADE Network Technologies, Inc., 2009.