

Instituto Tecnológico de Costa Rica



Escuela de Ingeniería Electrónica

**“Media Independent Interface (MII) implementation and data processing on
FPGA for Fast ETHERNET transmission over Plastic Optical Fiber”**

Politecnico di Torino

**Julio César Ramírez Molina
200109454**

Turin, February 2008

INSTITUTO TECNOLOGICO DE COSTA RICA
ESCUELA DE INGENIERIA ELECTRONICA
PROYECTO DE GRADUACIÓN
TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.


Miembros del Tribunal


Ing. Gabriela Ortiz León, M.Sc.

Profesor lector


Dr.-Ing. Paola Vega Castillo

Profesor lector


Ing. Anibal Coto Cortés, M.Sc.

Profesor asesor

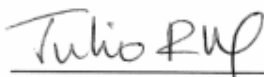
Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, Lunes 19 de mayo de 2008

Authenticity Declaration

Hereby I, Julio César Ramírez Molina, ID student card 200109454, declare that this project will be made entirely by me and applying my own ideas and knowledge to its development. I also commit myself to indicate the proper bibliographical references of the sources consulted.

Thus, I assume complete responsibility for the information included in the present document and used in the development of the project.



Julio César Ramírez Molina

Torino, March 28th ,2007

Abstract

This thesis focuses on the design and implementation of a Media Converter, which allows Ethernet Communication through 250 meters of Plastic Optical Fiber (POF). Three subsystems constitute this system.

First, a configurable MII Interface which provides Full Duplex Communication through the Ethernet at different data rates (10Mbps and 100Mbps). The User defines the data rate according to the capabilities of his Modulator.

The second subsystem is the Transmission Module. This module receives Ethernet data provided by the MII interface in order to encode it. Then it generates a Reed Solomon verifying sequence used by the receiver to correct the frame errors and to certificate the frame accuracy.

Finally, the third subsystem is a Testing Unit. It is used in the development and debugging process of the prototype. This module emulates a real data transmission using a Pseudo Random Bit sequence (PRBS).

The data received is stored and then downloaded by a block called Downloading Unit that executes Direct Memory Access (DMA) transfers between the memory of the Testing Unit and a computer. The transfers are made through the USB port of the PC.

Keywords: MII Interface; ETHERNET; Plastic Optical Fiber; POF; 100Mbps; 10Mbps; PRBS, Testing Unit; DP83848C; VHDL; Very High Speed Integrated Circuit Hardware Description Language; DMA; Direct Memory Access; USB; Universal Serial Bus; FPGA Virtex IV; Nallatech.

Table of Contents

1	INTRODUCTION	8
1.1	PROBLEM DEFINITION AND SOLUTION'S RELEVANCE	8
1.2	PROBLEM DEFINITION.....	9
1.2.1	<i>Generalities</i>	9
1.2.2	<i>Synthesis</i>	9
1.3	SELECTED SOLUTION	9
2	GOAL AND OBJECTIVES	11
2.1	GOAL.....	11
2.2	GENERAL OBJECTIVE	11
2.3	SPECIFIC OBJECTIVES	11
3	THEORETICAL BACKGROUND	12
3.1	SYSTEM DESCRIPTION.....	12
3.2	EXISTING LITERATURE	14
3.3	PHYSICS AND ELECTRONICS PRINCIPLES DESCRIPTION	15
4	METHODOLOGICAL PROCEDURES	19
4.1	PROBLEM DEFINITION.....	19
4.2	INVESTIGATION AND SOLUTION SYNTHESIS	19
4.3	SOLUTION IMPLEMENTATION	20
5	DESCRIPTION OF THE SELECTED SOLUTION	22
5.1	HARDWARE DESCRIPTION.....	22
5.1.1	<i>Hardware General Description</i>	22
5.1.2	<i>MII Initialization System</i>	24
5.1.2.1	Second Level Diagram.....	25
5.1.2.2	Third Level Diagram.....	26
5.1.3	<i>Transmission System</i>	30
5.1.3.1	First Level Diagram.....	30
5.1.3.2	Second Level Diagram.....	31
5.1.4	<i>Testing System of the Media Converter</i>	40
5.1.4.1	Traffic Generation Unit TGU	40
5.1.4.2	Data Compilation Unit DCU	41
5.1.4.3	Control Unit.....	44
5.1.4.4	Download Unit.....	45
5.1.5	<i>Reconstruction of the Preamble</i>	47
5.2	SOFTWARE DESCRIPTION	50
6	RESULTS ANALYSIS	52
7	CONCLUSIONS AND RECOMMENDATIONS	59
7.1	CONCLUSIONS	59
7.2	RECOMMENDATIONS.....	60
8	BIBLIOGRAPHY	61
9	APPENDIX	62
A.1	GLOSSARY, ABBREVIATIONS AND SYMBOLS.....	62
A.2	COMPANY INFORMATION	65
A.3	TESTING PROCEDURES	66
A.4	TESTING SET UP WITH THE POF AS LOOPBACK	70
A.5	C++ PROGRAMMING CODE OF THE SOFTWARE APPLICATION	71
10	ANNEX	77

ANNEX 1 PHOTOS OF THE CARDS AND DEVICES UTILIZED TO IMPLEMENT THE SYSTEM	77
ANNEX 2 DP83848 NATIONAL INSTRUMENTS CARD.....	79
ANNEX 3 NALLATECH PCI COMMUNICATION CORE APPLICATION NOTE.....	80
ANNEX 4 VIRTEX 4 FAMILY OVEVIEW.....	81
ANNEX 5 SECOND PART OF THE IEEE 802.3 PROTOCOL	82
ANNEX 6 XILINX CORE ASYNCHRONOUS FIFO v6.1.....	83
ANNEX 7 XILINX CORE REED SOLOMON ENCODER v5.0.....	84
ANNEX 8 SPECIAL WORDS (IFG, IDLE ,ASM) AND DC BALANCING	85

Table of Figures

Fig.1.1. Block Diagram of the Media Converter.....	9
Fig.3.1. Block Diagram of the Ethernet Interface.....	12
Fig.3.2. IEEE 802.3 standard relationship to the OSI/IEC for Open System Interconnection.....	15
Fig.5.1. BMCR writing and reading Protocols.....	17
Fig.5.1. General Diagram of the System	23
Fig.5.2. CRGU Block Diagram.....	24
Fig.5.3. CRGU Second Level Diagram.....	26
Fig.5.4. CRGU_MII Third Level Diagram.....	27
Fig.5.5. InitPHY Third Level Diagram	28
Fig.5.6. ControlGEN State Machine	29
Fig.5.7. First Level Diagram of the transmission system.....	30
Fig.5.8. BLOCK A block diagram.....	32
Fig.5.9. BLOCK B block diagram.....	34
Fig.5.10. BLOCK C block diagram.....	36
Fig.5.11. BLOCK D block diagram.....	38
Fig.5.12. System Second level diagram.	39
Fig.5.13. Traffic Generation Unit Diagram.....	41
Fig.5.14. DCU First Level Diagram.....	41
Fig.5.15. DCU Second Level Diagram.....	43
Fig.5.16. State Diagram of the Control Unit.....	44
Fig.5.17. PCI Communication Core block Diagram.....	45
Fig.5.18. State Diagram of the Control Unit.....	48
Fig.5.19. Block Diagram of the Preamble Generator.....	49
Fig.5.20. Flow Diagram implemented for the Software Application.....	51
Fig.6.1. PING summary in the Command Prompt	53
Fig.6.2. PING summary in the Command Prompt	53
Fig.6.3. PING summary in the Command Prompt	54
Fig.6.4. FPGA Utilization	56
Fig.6.5. Timing Summary of the Media Converter	57
Fig.A.1. Test 1 Configuration.....	66
Fig.A.2. Test 2 Configuration	67
Fig.A.3. Test 3 Configuration.....	68
Fig.A.4. Test 4 Configuration.....	69
Fig.A.5. Testing Setup with the POF as a LOOPBACK	70

List of Tables

Table 5.1. CRGU Pinout Configuration.....	25
Table 5.2. Truth Table of the ROM.....	28
Table 5.3. System Pinout Configuration	31
Table 5.4. Block A Pinout Configuration	33
Table 5.5. Block B Pinout Configuration	35
Table 5.6. Block C Pinout Configuration.....	37
Table 5.7. Block D Pinout Configuration.....	38
Table 5.8. DCU Pinout Configuration	42
Table 5.9. Core Pinout Configuration	46
Table 6.1 Results obtained from TEST1	52
Table 6.2 Results obtained from TEST2	52
Table 6.3. Error Positions and number of bits lost in a PRBS of 32767 bits long	54

1 Introduction

1.1 Problem Definition and Solution's Relevance

The goal of the project was to design and implement a Media Converter prototype fully compatible with the Ethernet standard and able to transmit at a data rate of 100Mbps over 250 meters of Plastic Optical Fiber (POF). It's important to notice that It was programmed in VHDL and loaded in an FPGA (Field Programmable Gate Array).

The project was developed in the PhotonLab, which is an experimental facility product of the cooperation and joint-venture of the Politecnico di Torino and "Istituto Superiore Mario Boella" (a private non-profit research center), both located in Turin, Italy.

This thesis is part of a big European project called "POF-ALL ("Paving the Optical Future with Affordable Lightning-fast Links"), which is sponsored by the European Research Sixth Framework Program (FP6), involving 11 partners and a total budget close to 3 Million Euros. POF-ALL develops a low-cost solution based in POF to increase the broadband access and to give advantage, independence, and better competitiveness to the European continent in the low-cost network access technologies.

1.2 Problem Definition

1.2.1 Generalities

The research group required a programmable Ethernet interface in order to give to the Media Converter a Full Duplex link with configurable data rates (10Mbps and 100Mbps) and autonegotiation capabilities (enable/disable). Also required the design and implementation of a hardware architecture to perform tests to the equalization algorithms designed by them.

1.2.2 Synthesis

The media converter requires a programmable ETHERNET link and the hardware architectures needed to test the equalization algorithms of the Media Converter.

1.3 Selected Solution

The design of the media converter involves the development of an optical link to and from the POF, an electrical link to and from the UTP Cat5 cable and a signal processing module in charge of the protocol conversion. Consequently, the system was divided in three main blocks, which are shown in figure 1.1.

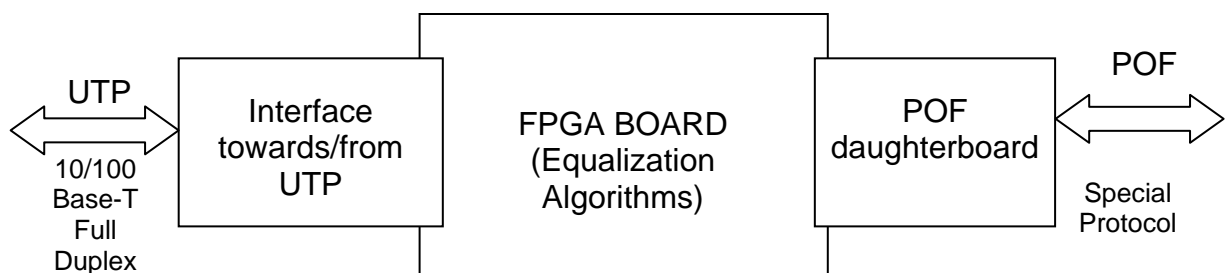


Fig.1.1. Block Diagram of the Media Converter

The first module on the left implements the MII interface. It was implemented with an special purpose card of National Instruments^[1] that is configured from the FPGA Board and that has all the circuitry and capabilities needed for the establishment of the Full Duplex link with data rates of 10Mbps and 100Mbps.

The center module in the diagram corresponds to the FPGA board and contains not only the necessary logic to configure the MII interface, but also the hardware architecture that execute the media conversion and the system field tests.

Finally, the last module on the right corresponds to a POF daughterboard designed and implemented by the PhotonLab researchers. It is an special card that converts the electrical signals into optical signals.

¹ The datasheet of the DP83848National Instruments Card is shown in the ANNEX 2

2 Goal and Objectives

2.1 Goal

- Design and implementation of a media converter prototype able to establish Fast Ethernet communication at 100Mbps over 250 meters of POF.

2.2 General Objective

- Provide the electronic structures to the media converter in order to establish Fast Ethernet communication.

2.3 Specific Objectives

- Implement the MII standard with the possibility of selecting between data rates of 10 Mbps and 100Mbps.
- Implement the hardware architecture that allows to perform field tests to the Media Converter.

3 Theoretical Background

3.1 System Description

Figure 3.1 shows the block diagram of the system without the modulation and demodulation blocks. The modules colored in green, brown and yellow are those developed for this thesis, those in orange were implemented by other members of the PhotonLab using the System Generator Tool of Matlab and finally the FEC (Forward Error Correction), modules on sky blue corresponds to Xilinx Cores. Also in this figure, the hierarchical configuration of the Transmission (highlighted in blue) and Reception (highlighted in red) blocks can also be observed in this figure.

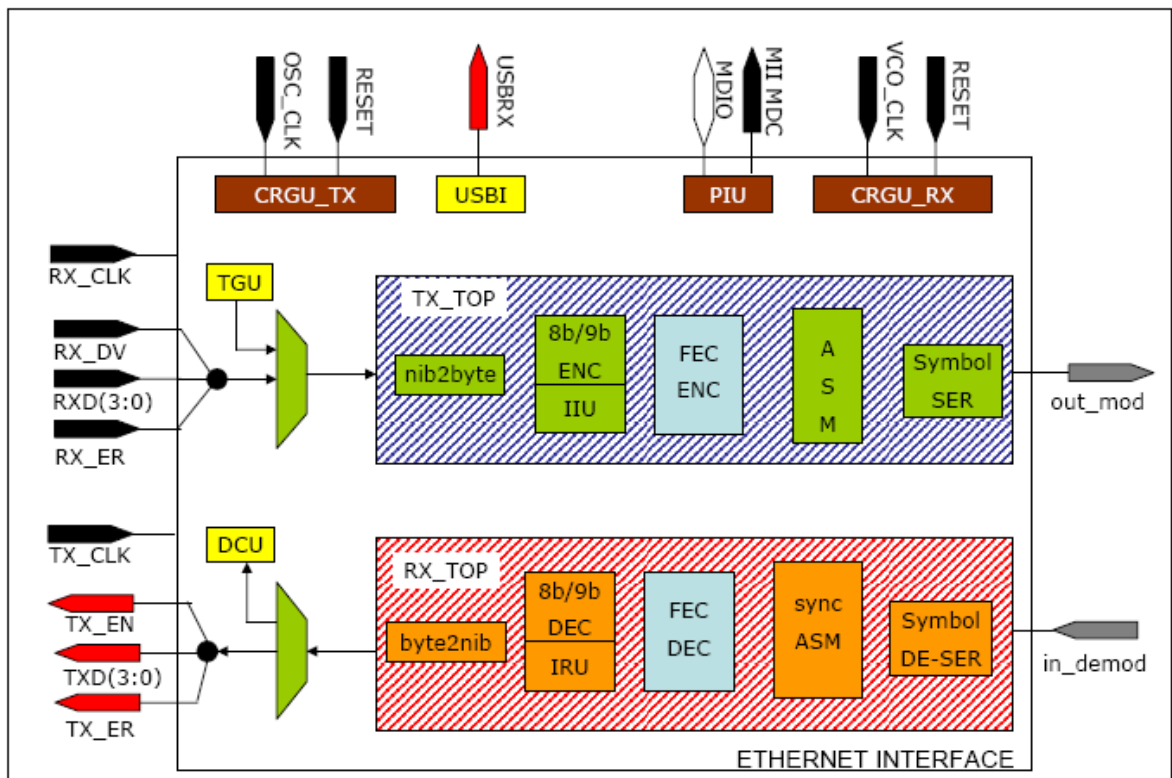


Fig.3.1. Block Diagram of the Ethernet Interface

The output pins of the system are colored in red whereas the inputs pins are black. The gray pins on the right side of the diagram are the connections to the modulation/demodulation blocks and finally the white pin on the top side of the diagram is a bidirectional line connected to the National Instruments card.

At this point the operation of some blocks shown in the fig 3.1 will be detailed. The brown modules are in charge of the DP83848 National Instruments Card configuration (Physical Initialization Unit, PIU) and generation of the internal clocks and resets(Clocks and Resets Generation Units, CRGU).

The green modules inside the Transmitter encode the information received in order to facilitate the data recovery process on the receiver side. The data stream is treated as follows: it's received as nibbles (4 bit words) by the first module (nib2byte) that concatenates two nibbles to form a byte word. Then the second module (8b/9b) encodes each byte as a 9 bit word, preparing it for the FEC module , where the data is encoded using the Reed Solomon Algorithm[3]. Once the data is encoded, the ASM (Attached Synchronizing Mark) is inserted for synchronization purposes and finally each 9 bit word is divided in 3 bit wide symbols (Symbol SER) and sent to the modulator.

At the end the yellow modules execute the system field tests, as may be observed in Figure. 3.1. The data stream can be received by the transmitter (TX_TOP) through a multiplexer from the TGU module (Traffic Generation Unit), which contains a PRBS sequence (Pseudo Random Bit Sequence). Then the receiver decodes the data and store them into the DCU module (Data Compilation Unit). At the end, the stored data are downloaded into the PC through the port called USBI, which is implemented with a core developed by the FPGA board manufacturer. Once in the PC the data are compared with the original PRBS sequence.

3.2 Existing Literature

The development of the project implied the study of the IEEE 802.3 standard in order to understand the operation of the ETHERNET protocol. Also a Xilinx Implementation Manual and a Nallatech Manual[6] were consulted. The first one includes information regarding the synchronizing and processing of data, also the operation of some Core Modules such as the DCMs for clock generation (Digital Clock Manager^[1], the Asynchronous FIFOs as interfacing blocks between clock domains^[2] and the Reed Solomon encoder/decoder used to guarantee the data integrity. The second manual explains the operation of the DMA interface provided by Nallatech used to download the results generated by the Testing Unit of the Media Converter (see USBI module in fig 3.1).

¹ For further Information regarding the DCM operation can be consulted the Virtex 4 Family Datasheet shown in the ANNEX 4

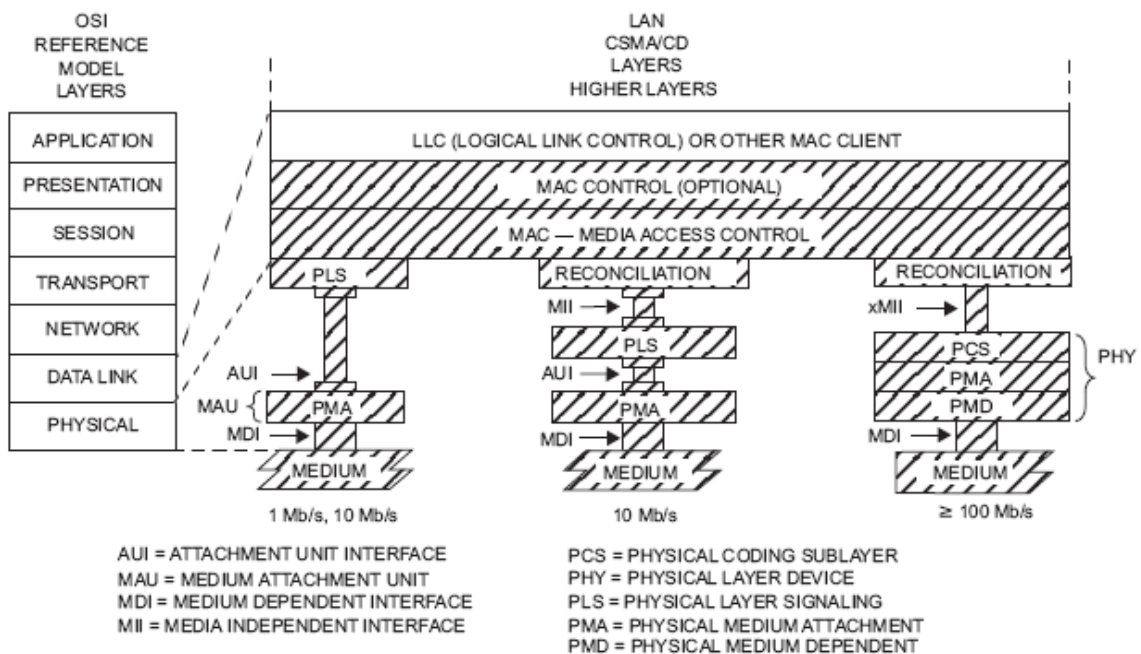
² The Asynchronous FIFO v6.1 datasheet is shown in the ANNEX 6

3.3 Physics and Electronics principles description

Ethernet Protocol

The Ethernet is a frame based computer networking technology for LANs (Local Area Networks). It acts as a packet-switching network where devices are connected and compete for access to the communication channel.

The IEEE 802.3 standard for Ethernet was defined in 1985 and intended to correspond with the lowest layers of the ISO/IEC Model for Open Systems Interconnection (OSI), see figure 3.2.



NOTE—In this figure, the xMII is used as a generic term for the Media Independent Interfaces for implementations of 100 Mb/s and above. For example: for 100 Mb/s implementations this interface is called MII; for 1 Gb/s implementations it is called GMII; for 10 Gb/s implementations it is called XGMII; etc.

Fig.3.2. IEEE 802.3 standard relationship to the OSI/IEC for Open System Interconnection^[1]

¹ Taken from the IEEE 802.3 standard shown in the ANNEX 5

The first part of this thesis is focused on the implementation of the Physical Layer (PHY) and the MII interface of the IEEE 802.3 standard, see figure 3.2. These layers interface the physical layer, where are received the Ethernet Frames from the serial line, and the Media Access Control (MAC) Layer, where the information is processed after being treated by the MII interface.

The Ethernet frames are transmitted following the scheme showed in table 3.1.

Table 3.1. Ethernet Frame configuration

Preamble	Start-of-Frame-Delimiter	MAC destination	MAC source	Length	Data	CRC32	Interframe gap IFG
7 octets of 10101010	1 octet of 10101011	6 octets	6 octets	2 octets	46-1500 octets	4 octets	12 octets minimum
64 octets		64-1518 octets					
72-1526 octets or more							

MII interface

The IEEE 802.3 standard defines the design and configuration of specific purpose systems that implements the MII interface. For this project, the DP83848 card of National Instruments^[1] was chosen.

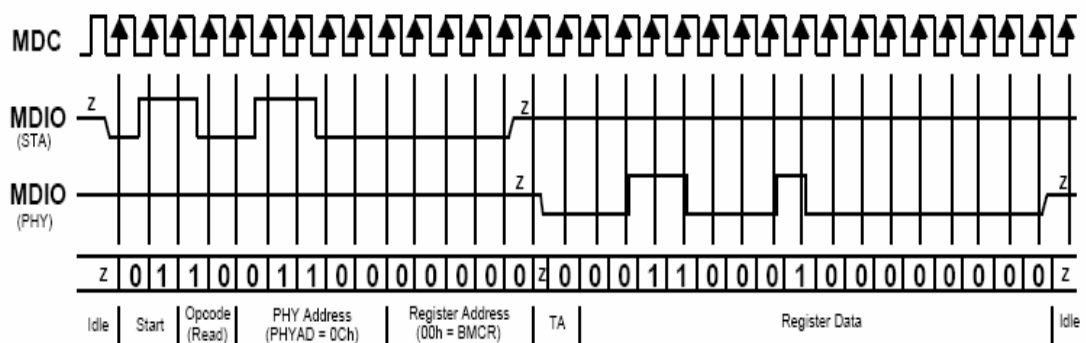
This card has a serial management interface that is accessible through two pins, the Management Data Clock (MDC) and the Management Data Input/Output (MDIO). The MDC has a maximum operation frequency of 25 MHz and doesn't have a minimum one. The MDIO is a bidirectional line that may be used to

¹ The datasheet of the DP83848 National Instruments Card is shown in the ANNEX 2.

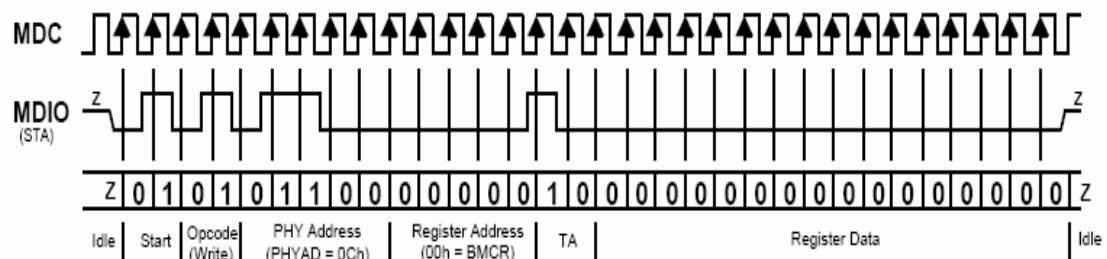
configure the MII interface through the writing of the Basic Mode Control Register (BMCR) and it's also used to obtained the link specifications through its reading.

Regarding the communication with the chip, is important to notice that the MDIO line requires an initialization sequence composed by 32 bit (logic ones) and that a minimum of one idle bit between transactions is required. Figure 5.1 shows the writing and reading protocol of the BMCR register established by the IEEE 802.3 standard.

MII Management Serial Protocol	<idle><start><op code><device addr><reg addr><turnaround><data><idle>
Read Operation	<idle><01><10><AAAAA><RRRRR><Z0><xxxx xxxx xxxx xxxx><idle>
Write Operation	<idle><01><01><AAAAA><RRRRR><10><xxxx xxxx xxxx xxxx><idle>



Typical MDC/MDIO Read Operation



Typical MDC/MDIO Write Operation

Fig.5.1. BMCR writing and reading Protocols^[1]

¹ Taken from the DP83848C datasheet shown in the ANNEX 2

Forward Error Correction (FEC)

Generalities and Formulation

This is a control error system for data transmission, in which redundant data are generated and concatenated at the end of the transmitted sequence. It allows, according with the algorithm implemented, the detection and correction of a limited number of errors that may be present on the receiver side. This method is usually applied in communication systems in which the retransmission is impossible.

The Reed Solomon is an error correcting algorithm commonly employed on the FEC encoding/decoding implementation. Basically, it over samples the data sequence that has to be transmitted and generates over-determined polynomial values that are sent. The over sampling gives to the receiver the capacity to recover the correct values of the erroneous data when decoding.

The mathematical formulation of this algorithm is as follows:

Given a finite field F (also referred as Galois Field) and polynomial ring $F[x]$, let n and k be chosen such that $1 \leq k \leq n \leq |F|$. Pick n distinct elements of F , denoted $\{x_1, x_2, \dots, x_n\}$. Then, the codebook \mathbf{C} is created from the tuples of values obtained by evaluating every polynomial (over F) of degree less than k at each x_i ; that is,

$$C = \left\{ (f(x_1), f(x_2), \dots, f(x_n)), f \in F[x], \deg(f) < k \right\}$$

\mathbf{C} is a $[n, k, n-k+1]$ code; in other words, it is a linear code of length n (over F) with dimension k and minimum distance $n-k+1$.

Hardware Implementation

The hardware implementation of the Reed Solomon Encoder and Decoder are explained by their respective datasheets^[1].

4 Methodological Procedures

4.1 Problem Definition

The problem definition was based on interviews and meetings with the POF-ALL research team.

4.2 Investigation and Solution Synthesis

The problem definition and the system design were developed base on interviews with the PhotonLab researchers. For the development and implementation of the system, the Xilinx and Nallatec^[2] manuals were consulted.

At the beginning, the system designed was simulated with Modelsim® edition 6.0d. and later on some field tests were performed using the Testing Unit provided to the Media Converter. The result was a hardware system that fulfills all the requirements specified.

¹ The Reed Solomon Encoder v5.0 Datasheet is shown in the ANNEX 7.

² Manufacturer of the FPGA Kit development board that has been used.

4.3 Solution Implementation

The procedure followed to implement the solution involves the following steps

First Stage: MII Implementation

Investigation:

This activity involves the familiarization with the IEEE/802.3/ETHERNET protocol, specifically with the MII interface. It also includes the learning of VHDL language, since this is the established language for the whole project.

Configuration of the MII interface:

Implement the MII configuration system in an FPGA using VHDL on Xilinx ISE Foundation 8.1.03i.

Simulation of the MII:

In this activity the behavioral and “post place and route” simulations with Modelsim® edition 6.0d. took place.

Testing of the MII:

Once the MII configuration system was simulated, it was tested for the different data rates required. The tests performed are detailed in Appendix A.3.

Second Stage: Test Unit and Transmitter Implementation

At this stage the field test architecture and the transmission module that treats the data received from the MII interface were design and implemented.

Investigation:

Familiarization with the Cores provided by Nallatech and Xilinx that were used in the implementation of the hardware architecture.

Programming:

The system was designed and implemented in a FPGA using VHDL on Xilinx ISE Foundation 8.1.03i.

Simulation:

In this activity the behavioral and “post place and route” simulations with Modelsim® Xilinx edition 6.0d. took place

Testing of the system:

In this activity different experiments to corroborate the correct operation of the implemented system were developed. The tests performed are detailed in Appendix A.3.

5 Description of the Selected Solution

5.1 Hardware Description

5.1.1 Hardware General Description

Figure 5.1 shows the general diagram of the hardware implementation of the selected solution. The details regarding the implementation of each module are given in the following sections. However, at this point the main generalities of the system are explained.

The MII initialization system is constituted by the CRGU (Clocks and Resets Generation Unit) module. It generates the clocks and the resets of the system. There is also a Preamble Generator in charge of the reconstruction of the preamble that precedes each ETHERNET communication. The modules called Traffic Generation Unit (TGU) and Data Compilation Unit (DCU) constitute the Internal Testing Unit based on the PRBS sequence. Figure 5.1 also shows the DMA USB interface which is used to download the information generated by the Internal Testing Unit into the PC. Finally, there is a MII interface implemented with the DP83849 card and also the modules that correspond to the Transmitter and the Receiver.

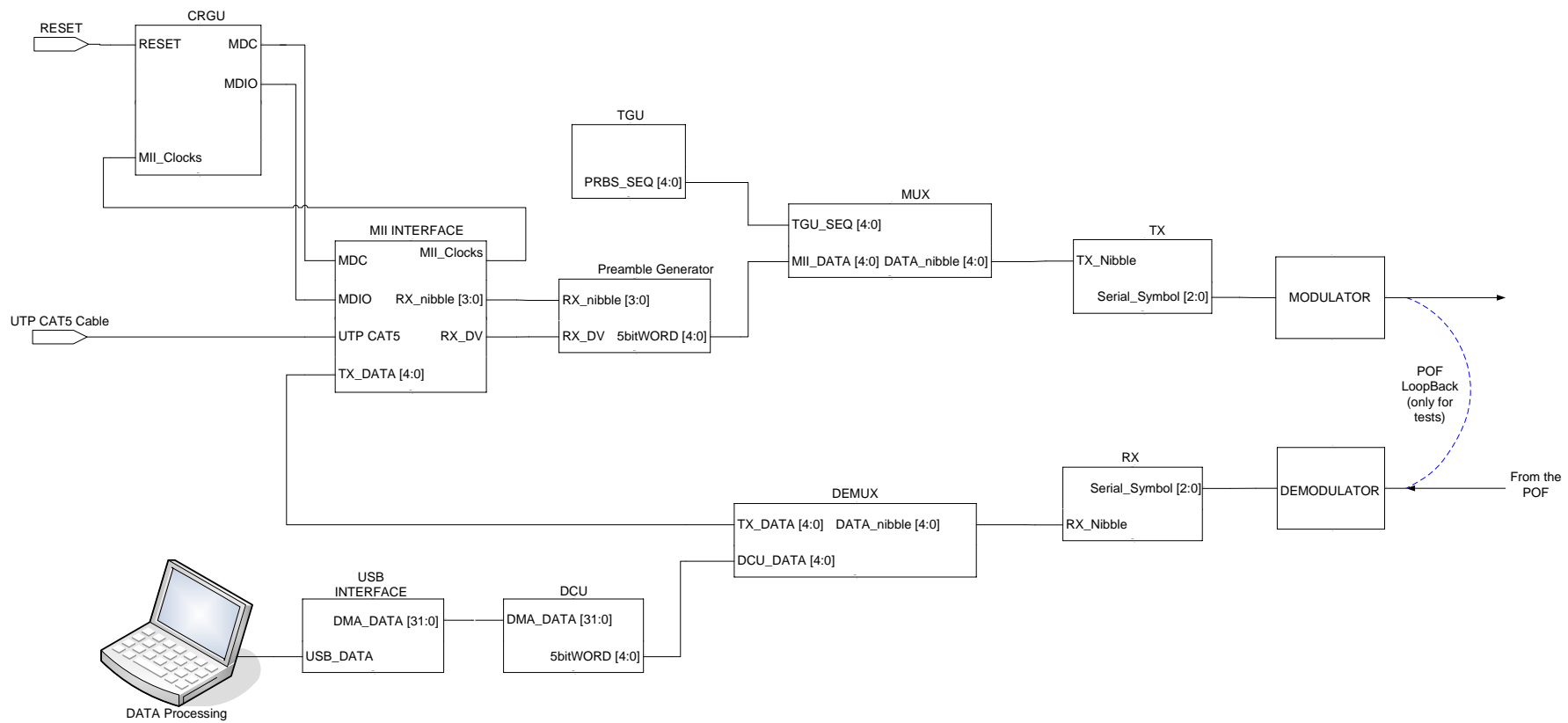


Fig.5.1. General Diagram of the System

5.1.2 MII Initialization System

The configuration system of the card that implements the MII interface is part of the module called Clocks and Resets Generation Unit (CRGU). This module is also in charge of the logic resets and of the system's clocks. Figure 5.2 shows its block diagram and the table 5.1 describes its port connections.

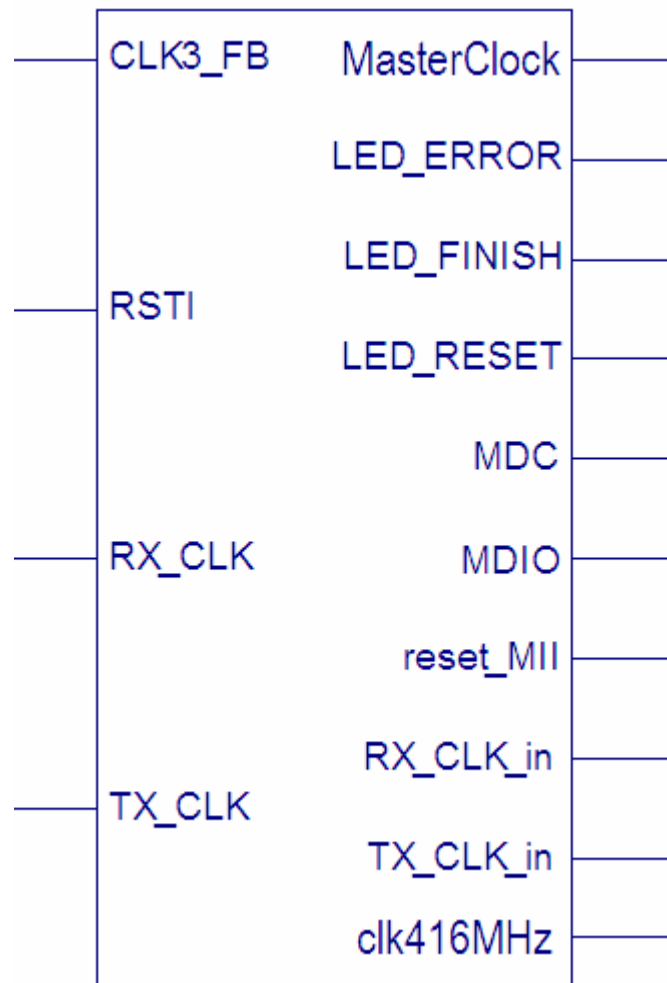


Fig.5.2. CRGU Block Diagram

Table 5.1. CRGU Pinout Configuration

Name of the Port	Direction	Description
CLK3_FB	Input	80 MHz clock provided by the crystal of the Board.
RSTI	Input	General Reset of the system
RX_CLK	Input	25/2.5MHz reception clock provided by the DP83848
TX_CLK	Input	25/2.5MHz transmission clock provided by the DP83848
MasterClock	Output	125MHz clock generated based on CLK3_FB
LED_ERROR	Output	Turns on a led when the initialization of the DP83848 fails.
LED_FINISH	Output	Lights a led when the initialization of the DP83848 succeeds.
LED_RESET	Output	Lights a led when the system is asserted the system reset
MDC	Output	25 MHz clock synchronous to the MDIO
MDIO	Output	Bi-directional signal used to configure the DP83848
Reset_MII	Output	Synchronous Reset deasserted when the MII clocks have been successfully regenerated
RX_CLK_in	Output	25/2.5 MHz clock regenerated based on RX_CLK
TX_CLK_in	Output	25/2.5 MHz clock regenerated based on TX_CLK
Clk416MHz	Output	41.6MHz clock generated based on CLK3_FB

5.1.2.1 Second Level Diagram

Figure 5.3 shows the second level diagram of the CRGU module. The initPHY block configures the DP83849 card. Once the MII interface is successfully configured this block asserts its FINISH output, allowing the operation of the CRGU_MII block and hence the generation of both the internal MII clocks and the internal reset.

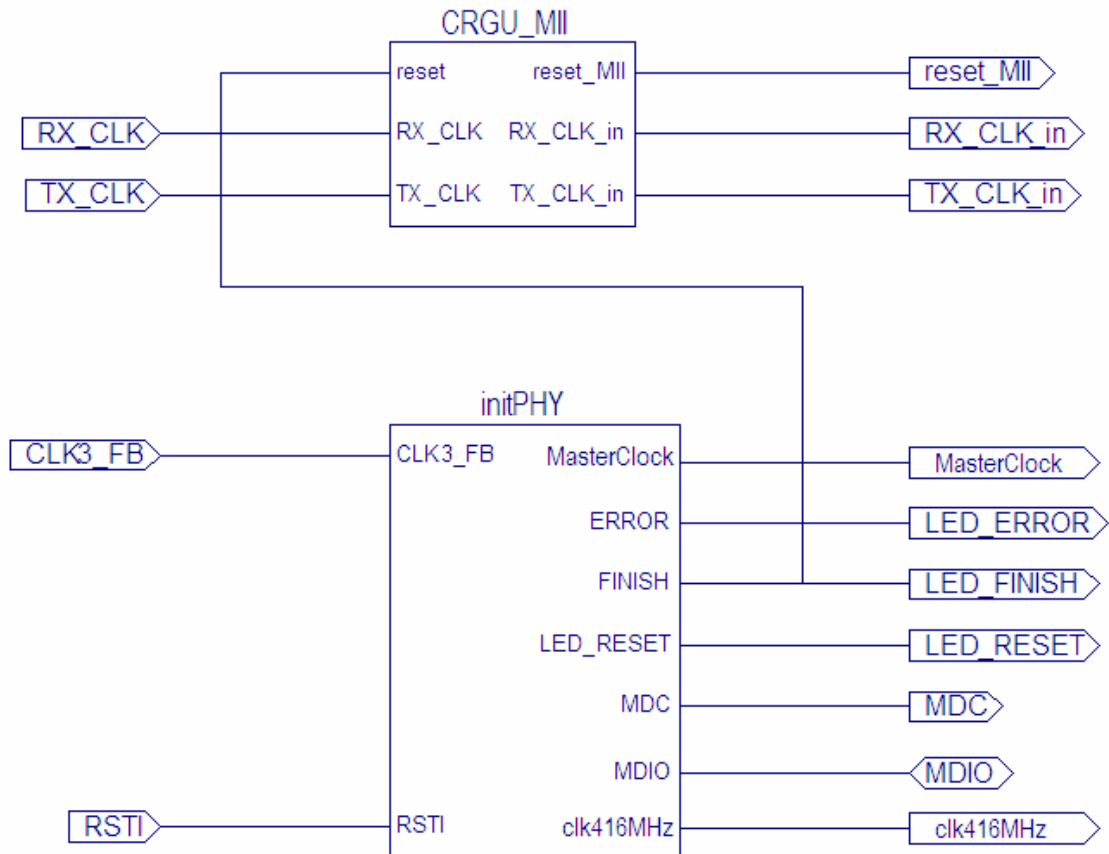


Fig.5.3. CRGU Second Level Diagram

5.1.2.2 Third Level Diagram

Module CRGU_MII

The figure 5.4 shows the hardware implementation of the CRGU_MII. The two bigger blocks (DCM_TX and DCM_RX) correspond to the Xilinx's Core named Digital Clock Manager (DCM)¹, which basically allows the clock synthesis, phasing and synchronization based on its input clock signal. The reset logic is generated by the OR function of the LOCKED outputs of the DCMs that are asserted high when the generated clocks are stable.

¹ For further information, regarding the DCM operation, can be consulted the Virtex 4 Family Datasheet shown in the ANNEX 4

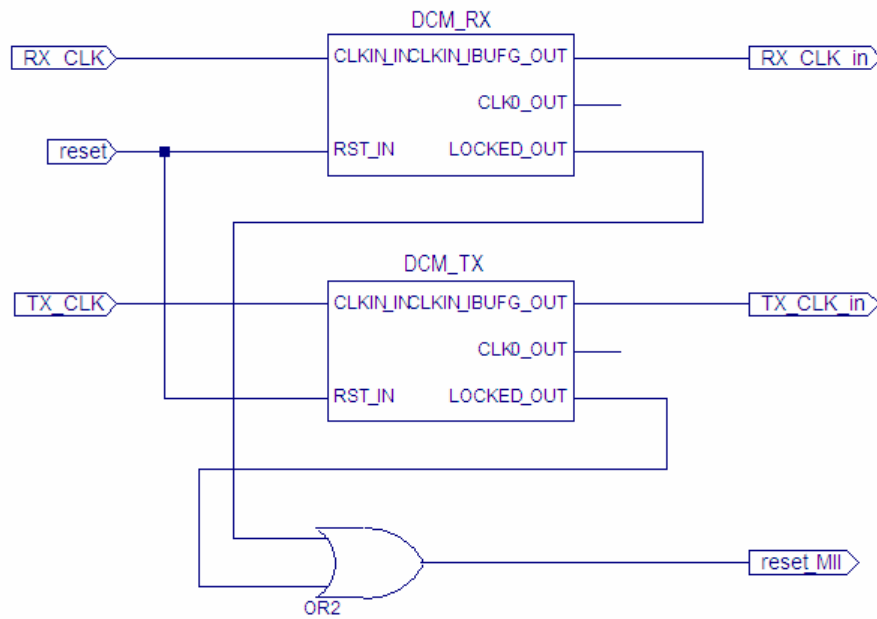


Fig.5.4. CRGU_MII Third Level Diagram

Module initPHY

Figure 5.5 shows the third level diagram of this module. It shows the minimum system architecture developed to configure the DP83848 National Instruments Card. The module controlGEN constitutes the CPU of the system, the ROM memory (whose truth table is shown in the table 5.2) stores the programming words and finally the module called specialPORT executes the communication with the card through the bidirectional serial line (MDIO).

It's important to mention that initPHY has an input called speed, employed by the user to define the data rate of the Ethernet Link. However, it isn't included in the diagrams because the research team required its fixation in order to configure the link for a data rate of 100Mbps.

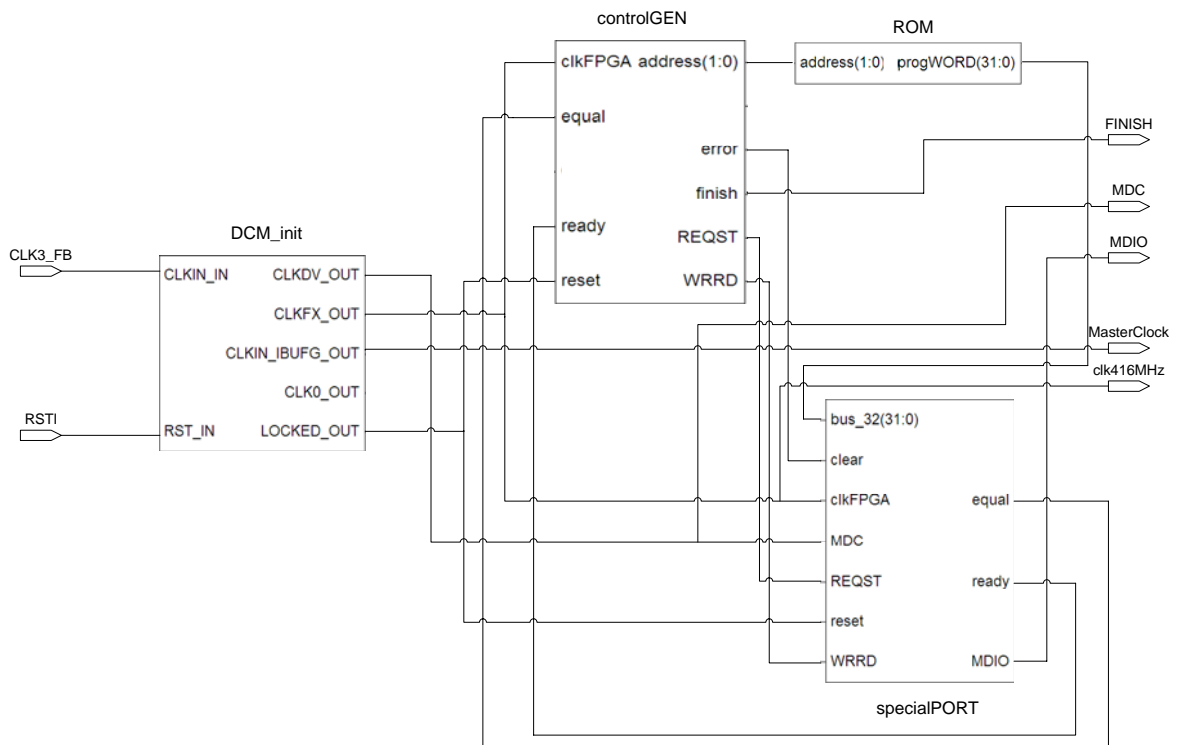


Fig.5.5. InitPHY Third Level Diagram

Table 5.2. Truth Table of the ROM

Address (2:0)	progWORD (31:0)
000	MDIO initializing word
001	100Mbps programming word
010	100Mbps verifying word
011	Empty Position
100	MDIO initializing word
101	10Mbps programming word
110	10Mbps verifying word
111	Empty Position

Figure 5.6 shows the state machine that controls the configuration of the MII interface. It's important to notice that when the card is not successfully initialized the system retries the programming process until it succeeds.

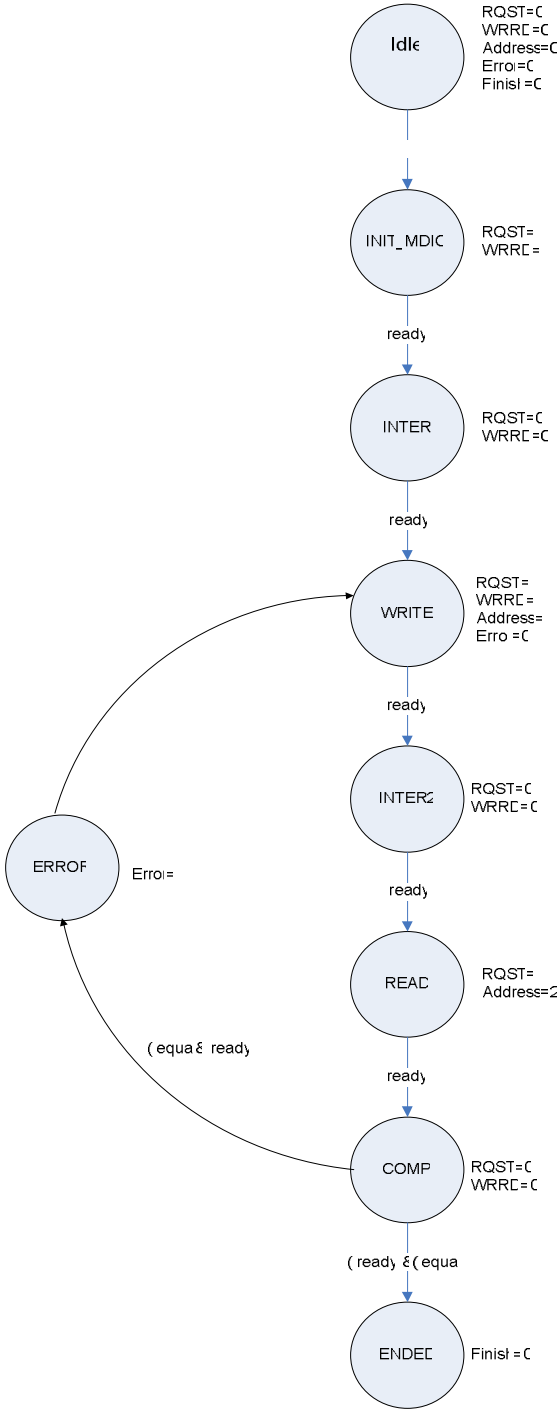


Fig.5.6. ControlGEN State Machine

5.1.3 Transmission System

The transmission block is part of a system that treats the data provided by the MII interface in order to send them through the Plastic Optical Fiber (POF). The next sections explain the design that handles the data up to the point at which is sent to the pulse modulator as a 8-PAM (Pulse Amplitud Modulation) modulated symbol. Figure 5.7 shows the first level diagram of the implemented module and table 5.1 describes its port connections.

5.1.3.1 First Level Diagram

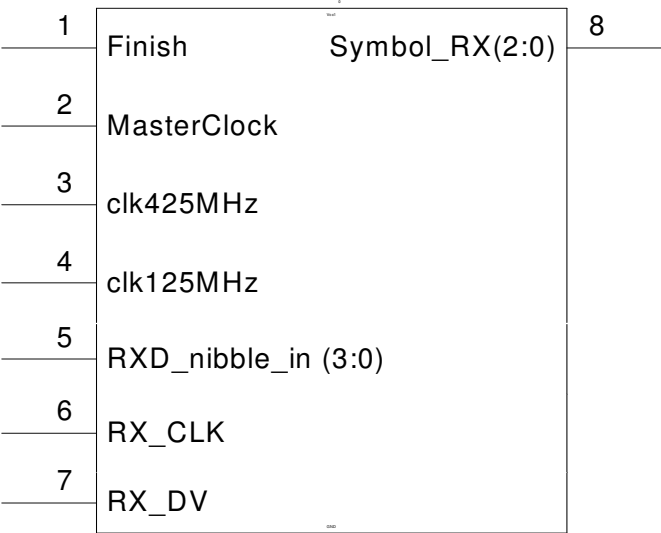


Fig.5.7. First Level Diagram of the transmission system

Table 5.3. System Pinout Configuration

Name of the Port	Direction	Description
Finish	Input	This signal informs that the MII initialization is complete and constitutes the system reset.
MasterClock	Input	125 MHz clock provided by an external module
clk125MHz	Input	12.5 MHz Output clock
clk425MHz	Input	41.6 MHz Output clock
RXD_nibble_in (3:0)	Input	Receive Data: nibble wide received data provided by the MII interface (Synchronous to RX_CLK).
RX_CLK	Input	25 MHz clock provided by the MII interface
RX_DV	Input	Receive Data Valid: Asserted high to indicate that valid data are present on RXD_nibble_in (3:0)
symbol_RX (2:0)	Output	Data Output: synchronous to clk425MHz

5.1.3.2 Second Level Diagram

The transmission block is composed by five fundamental sub-blocks, that from now on, will be referred as A,B,C,D and E. The location of the synchronization interfaces was the criterion applied to define the functional blocks in the logic chain. This means that each block is separated from the next one by a synchronization interface, which is implemented with a special circuit involving asynchronous FIFO. The next sections explain the main characteristics of these Blocks and the connections between them.

Block A: word and Inter Frame Gap codification.

Clock Generalities

Block A works with two different clock domains. At the input, the data received (RX_nibble_in (3:0)) are synchronized with RX_CLK (25 MHz), which is provided by the MII interface. At its output the transmitted data (DATA_ETH_RX, RE_ETH_RX) are synchronized to MasterClock_in (125MHz), which is provided by the field programmable gate array (FPGA) (see fig. 5.8). These different clock domains imply the implementation of a synchronization circuit implemented with an Asynchronous FIFO that helps to avoid the lost of data.

Operation

The module concatenates two samples of RX_nibble_in (3:0) in order to form a byte. Then, it adds RX_DV and forms a 9 bit data word that is assigned to DATA_ETH_RX (8:0). This output will be valid data when RX_DV is asserted high, or on the contrary, an Inter Frame Gap (IFG). The encoding of the IFG is necessary because the system must generate it on the other side of the POF in order to fulfill the ETHERNET protocol. There are two IFG's encoding words to foresee the effect of the encoded data on the communication line due to the DC Balancing^[1]. The table 5.4 describes its port connections.

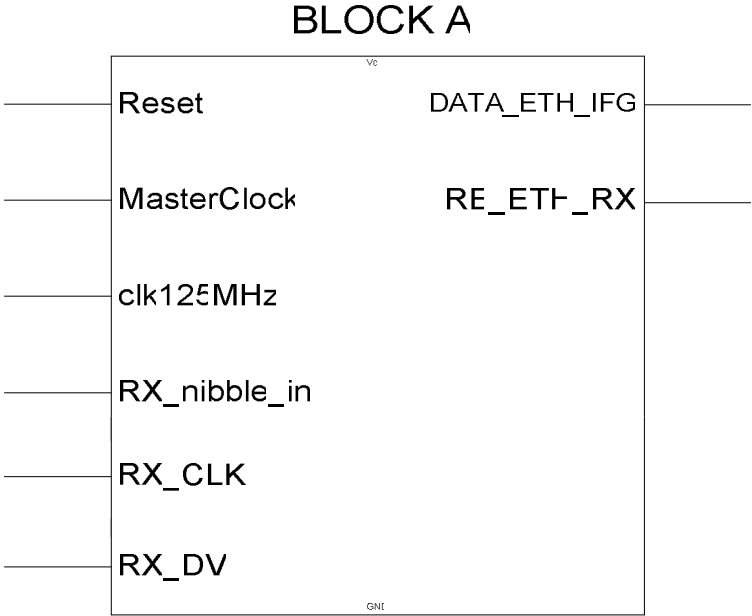


Fig.5.8. BLOCK A block diagram.

¹ For further information regarding DC balancing can be consulted the ANNEX 8

Table 5.4. Block A Pinout Configuration

Name of the Port	Direction	Description
Reset	Input	Active High reset
MasterClock	Input	125 MHz clock provided by the FPGA
clk125MHz	Input	A 12.5MHz clock provided by the Block E.
RXD_nibble_in (3:0)	Input	Receive Data: nibble wide receive data provided by the MII interface (Synchronous to RX_CLK).
RX_CLK	Input	25 MHz clock provided by the Block E which treats the MII interface clock input.
RX_DV	Input	Receive Data Valid: Asserted high to indicate that valid data are present on RXD_nibble_in (3:0)
DATA_ETH_RX	Output	Data Output: nine bit wide sent data, synchronous to MasterClock
RE_ETH_RX	Output	Read Enable: Write Enable signal of the next module of the logic chain.

Block B: IDLE insertion**Clock Generalities**

This block is simpler than the previous one, because it just works with the MasterClock at 125 MHz (see fig.5.9).

Operation

The system works with the premise that there must always be data flow in the logic chain, but this is impossible considering that the data from the MII are sampled with a 25 MHz clock whereas the data in the chain are transmitted with a 125 MHz. Then is necessary to include special data words in order to avoid the absence of real data. These words are referred as IDLE's. Again, there are two of them because of the DC balancing consideration. The module inserts these words when the SER_FLAG signal is asserted high by a logic that detects the moment at which there are less than five data words in the last FIFO of the chain (FIFO_AUX), located in the Block D. Table 5.5 describes its port connections.

The insertion of these words also generates the necessity of a synchronization and storage circuitry, since is necessary to store them while all the signals needed for their transmission are generated, and also in order to accumulate them to maintain the data flow.

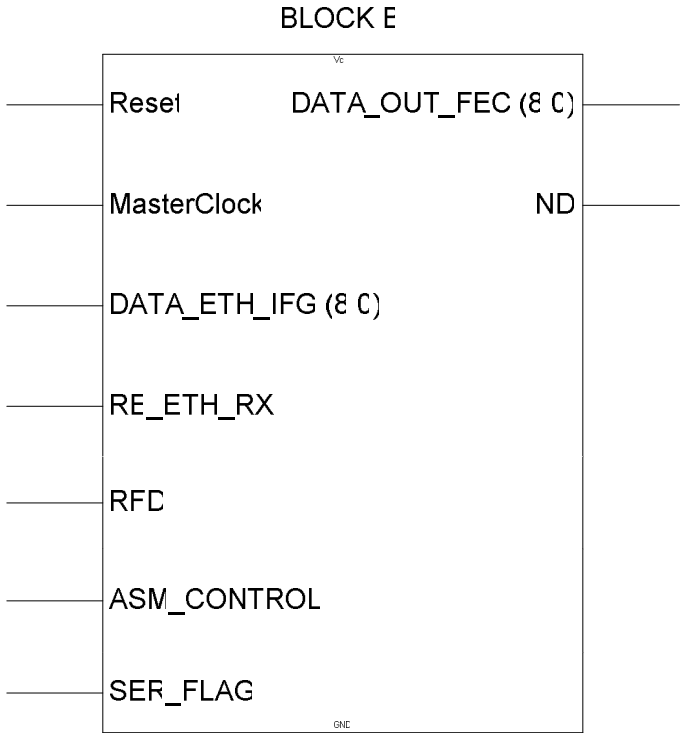


Fig.5.9. BLOCK B block diagram.

Table 5.5. Block B Pinout Configuration

Name of the Port	Direction	Description
Reset	Input	Active High reset
MasterClock	Input	125 MHz clock provided by the FPGA
RX_CLK	Input	25 MHz clock provided by the Block E which treats the MII interface clock input.
DATA_ETH_RX (8:0)	Input	Data Input: nine bit wide received data provided by the Block B and synchronous to MasterClock
RE_ETH_RX	Input	Read Enable: Asserted high to indicate that data valid is present on DATA_ETH_RX (8:0)
ASM_CONTROL	Input	Feedback signal from Block C that is asserted high when the ASM_machine ¹ is receiving data.
RFD	Input	Feedback signal from the Block C that is asserted high when the FEC ² is ready to accept new data.
SER_FLAG	Input	Feedback signal from the Block D that prevents the interruption of the data flow.
DATA_OUT_FEC (8:0)	Output	Data Output: nine bit wide sent data, synchronous to MasterClock
ND	Output	New Data: Asserted high to indicate that valid data are present at the inputs of the FEC.

Note (1): The ASM_machine (Attached Synchronizing Mark) is a module that inserts four data words at the beginning of each transmission in order to synchronize its decoding on the other side of the POF, this function is part of the next block and is explained in the corresponding section.

Note (2): The FEC is the acronym of Forward Error Correction which generates a verification sequence in order to certificate the veracity of the information when arrives to its destination, this function is part of the next block and is explained in the corresponding section.

Block C: ASM insertion and FEC generation.

Clock Generalities

This block works with the MasterClock (125 MHz).

Operation

This module implements the Forward Error Correction (FEC). The sequence is constituted by 479 information words (9 bits codification) and 32 verification words, generated applying the Reed Solomon algorithm^[1]. Each sequence is preceded by four special words called ASM, which are needed to synchronize the data acquisition in the module located on the other side of the POF. In figure 5.10 the block diagram of the block C is shown and table 5.6 describes its port connections.

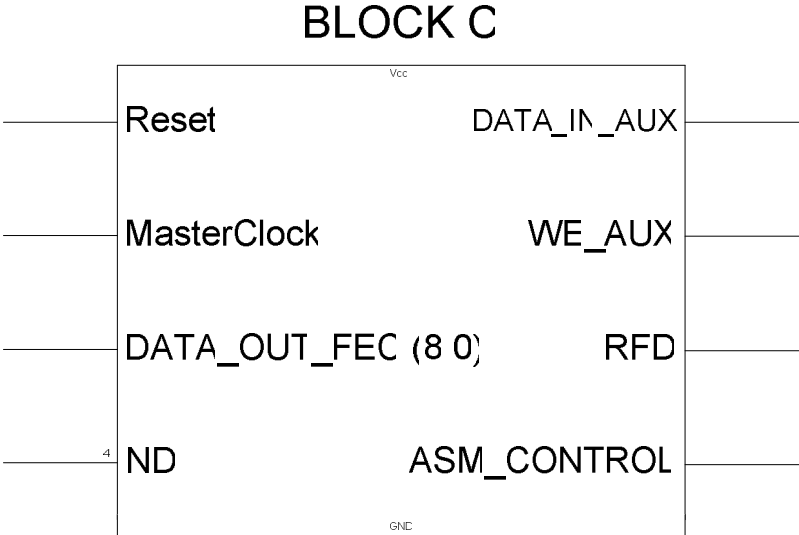


Fig.5.10. BLOCK C block diagram.

¹ The datasheet of the block that executes the Reed Solomon algorithm is shown in the ANNEX 7

Table 5.6. Block C Pinout Configuration

Name of the Port	Direction	Description
Reset	Input	Active High reset
MasterClock	Input	125 MHz clock provided by the FPGA
DATA_OUT_FEC (8:0)	Input	Data Input: nine bit wide received data provided by the Block B and synchronous to MasterClock
ND	Input	New Data: Asserted high to indicate that data valid is present on the inputs of the FEC.
DATA_IN_AUX (8:0)	Output	Data Output: nine bit wide sent data, synchronous to MasterClock
WE_AUX	Output	Write Enable: asserted high to indicate to the Block D that valid data are present on DATA_IN_AUX (8:0)
RFD	Output	Feedback signal from the Block C that is asserted high when the FEC is ready to accept new data.
ASM_CONTROL	Output	Feedback signal from Block C that is asserted high when the ASM_machine is receiving data.

Block D: Data serialization**Clock Generalities**

This module works with two clocks, the first one is the MasterClock (125MHz), used to sample the data present at its inputs and the second one is the clk425MHz used to serialize the data words into symbols constituted by three bits.

Operation

This module samples its input DATA_IN_AUX (8:0) when WE_AUX is asserted high by the preceding module. This operation is realized in synchrony with MasterClock. Once received, the data word is divided into three symbols, constituted as well by three bits of data. This serialization is made using clk425MHz (41.6MHz). In the end these symbols are sent to the 8-PAM modulator to be treated before sending them to the transmission module through the POF. In figure 5.11 is shown the block diagram of this block and the table 5.7 describes its port connections.

BLOCK D

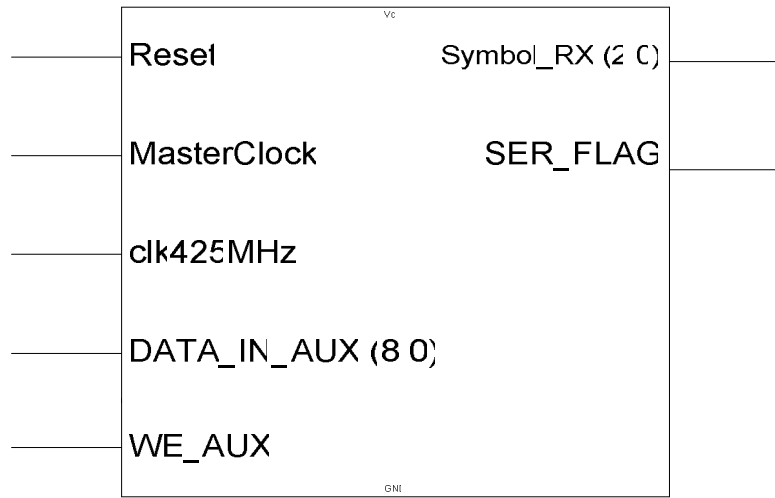


Fig.5.11. BLOCK D block diagram.

Table 5.7. Block D Pinout Configuration

Name of the Port	Direction	Description
Reset	Input	Active High reset
MasterClock	Input	125 MHz clock provided by the FPGA
Clk425MHz	Input	Serialization Clock: a 41.6 MHz clock generated inside the FPGA by the Block E in order to serilize the data words sampled in DATA_IN_AUX.
DATA_IN_AUX (8:0)	Input	Data Input: nine bit wide received data provided by block C and synchronous to MasterClock.
WE_AUX	Input	Write Enable: asserted high to indicate to the Block D that valid data is present on DATA_IN_AUX (8:0)
Symbol_RX (3:0)	Output	Data Output: three bit wide output data sent to the 8-PAM modulator and synchronous to clk425MHz.
SER_FLAG	Output	Feedback signal sent to the block B in order to avoid the interruption of the data flow.

In the figure 5.12 is shown the second level diagram of the system.

System Second Level Diagram

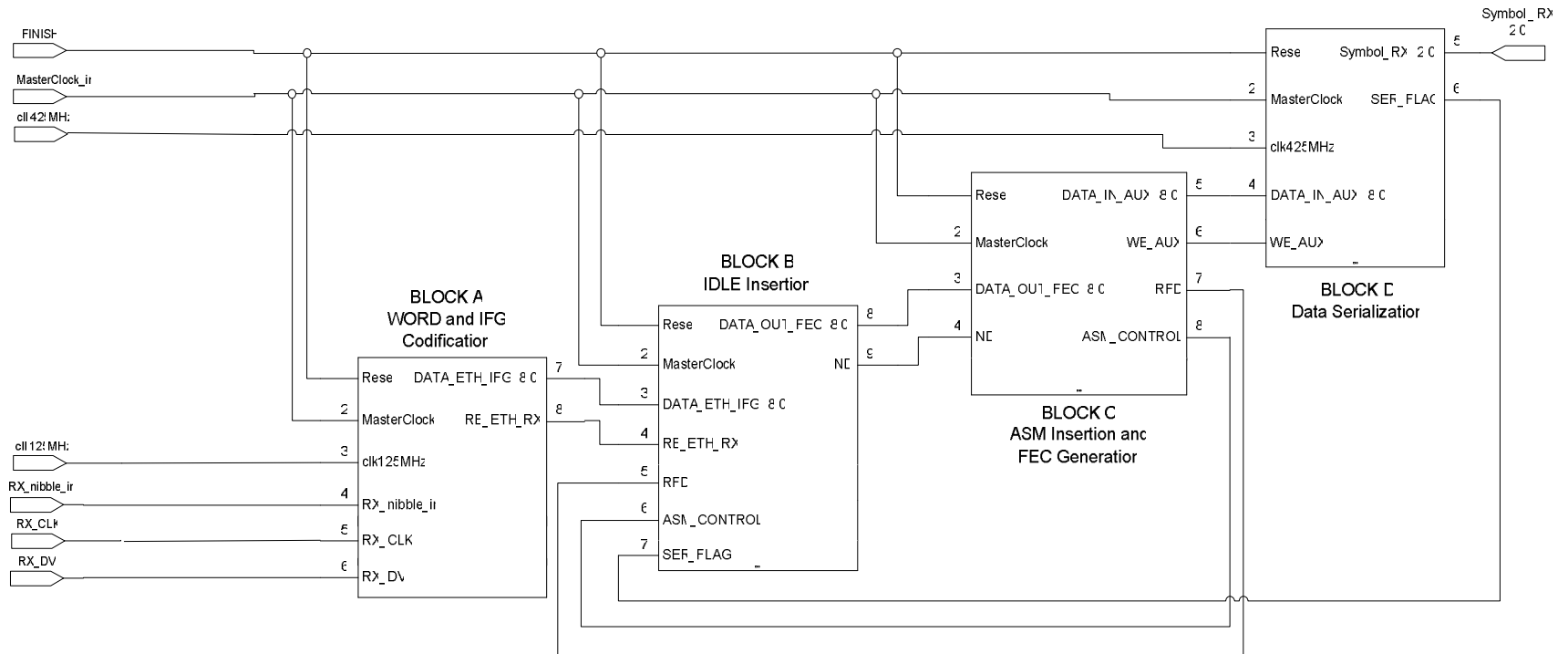


Fig.5.12. System Second level diagram.

5.1.4 Testing System of the Media Converter

This system is used to execute the field tests of the Media Converter constituting a fundamental part of its debugging process. The test procedure is as follows:

- 1) Disable the MII interface as a data source.
- 2) Isolate the modules that will be tested configuring an internal loop connection in an appropriate point.
- 3) Transmit a Pseudo Random Binary Sequence (PRBS) through the prototype.
- 4) Download the data sequence received into a computer.

The testing system is composed by a Traffic Generation Unit (TGU), a Data Compilation Unit (DCU), a downloading interface (see figure 3.1), and a State Machine that controls the data transfer between the modules mentioned and the computer. These blocks will be explained in the next sections.

5.1.4.1 Traffic Generation Unit TGU

This module generates the PRBS that is transmitted through the Media Converter. Due to the intrinsic complexity of the implementation of a PRBS generator, the sequence was created with Matlab and uploaded into a ROM inside the PFGA using a Xilinx Core Tool. Nevertheless, this solution implied the use of a large ROM which had an obvious impact on the system efficiency. So at this point, was decided to generate a PRBS composed by 6554 words 5 bits wide and transmit it continuously until a larger DCU's memory is filled. Figure 5.13 shows the hardware configuration of the TGU. It's important to mention that the counter's pin called Count Enable (CE) is employed by the Control Unit to enable/disable the PRBS emulation.

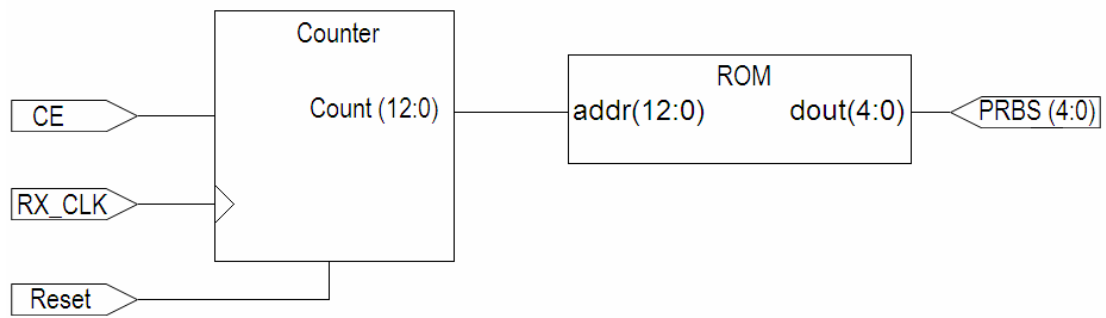


Fig.5.13. Traffic Generation Unit Diagram

5.1.4.2 Data Compilation Unit DCU

DCU First Level Diagram

The information is stored by the DCU once it passes through the media converter. The FIFO_OUT width was defined according to the Downloading Unit requirements, which is based on a core provided by Nallatec. Figure 5.14 shows the DCU first level diagram and the table 5.8 describes its port connections.

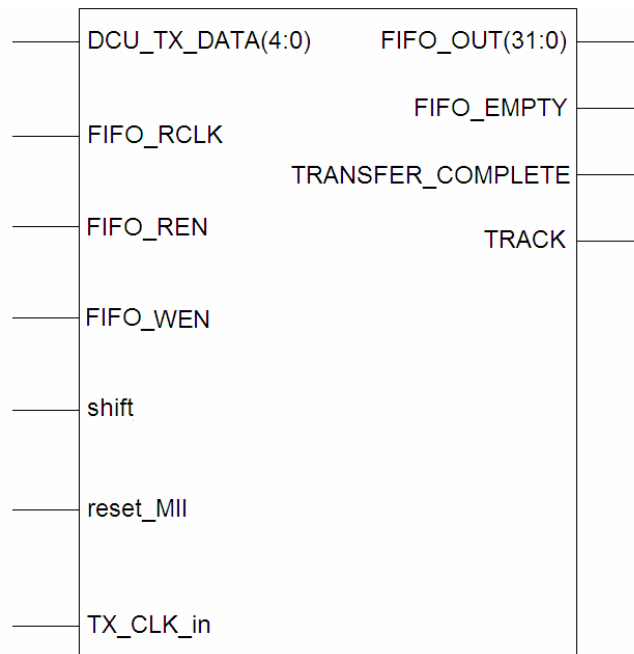


Fig.5.14. DCU First Level Diagram

Table 5.8. DCU Pinout Configuration

Name of the Port	Direction	Description
Reset_MII	Input	Reset provided by the CRGU
TX_CLK_in	Input	25MHz clock generated based on TX_CLK
FIFO_RCLK	Input	40MHz clock provided by the Downloading Interface
FIFO_REN	Input	Read Enable: connected to the downloading interface.
DCU_TX_DATA	Input	Data Input: 32 bits wide word synchronous to TX_CLK_in
FIFO_WEN	Input	Write Enable: Asserted High by the Control Unit to write a 32 bit word into the Internal FIFO.
Shift	Input	Asserted High by the Control Unit to process a new PRBS symbol.
FIFO_OUT (31:0)	Output	Data Output: 32 bits wide word synchronous to FIFO_RCLK
TRANSFER_COMPLETE	Output	Asserted High to Indicate to the Downloading Unit that the DCU's FIFO is full and ready to transfer the data collected.
FIFO_EMPTY	Output	Asserted High to indicate to the Downloading Unit that the DCU's FIFO is empty.
TRACK	Output	Asserted High to indicate to the Control Unit that the sequence that precedes a new transmitting process has been detected.
FIFO_FULL	Output	Asserted High to indicate to the Control Unit and to the downloading interface that the DCU's FIFO is full.

Second Level Diagram

This unit is composed by a concatenation module and a 16Kbyte FIFO. Figure 5.15 shows its hardware configuration. The next sections will explain the operation of each module.

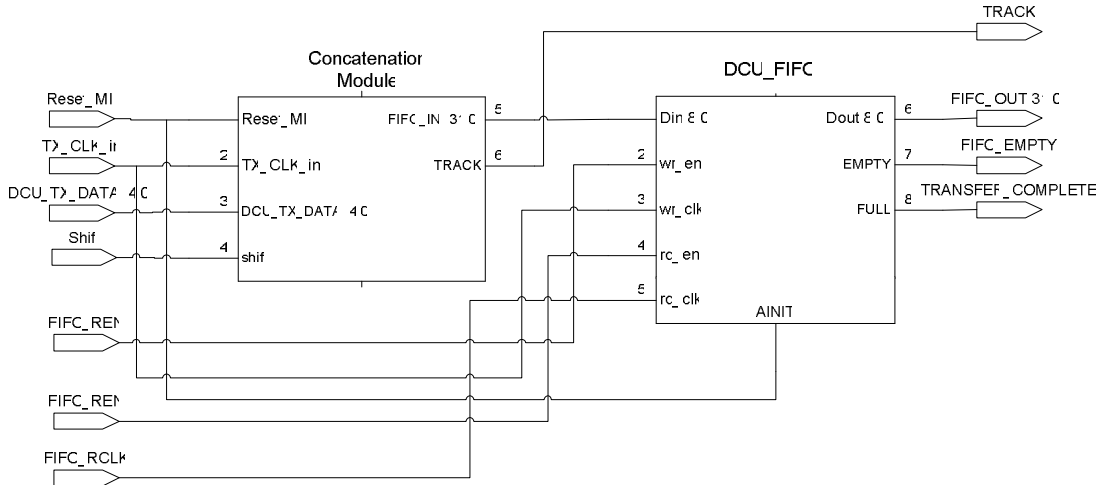


Fig.5.15. DCU Second Level Diagram

Concatenation Module

This module takes the 5 bit PRBS symbols and concatenates them in order to form a 32 bit word that is stored into DCU_FIFO. This process is performed while “shift” corresponds to a logic one. Moreover, it recognizes the synchronization sequence at the beginning of each transmission, communicating it to the Control Unit through its pin labeled as TRACK.

DCU_FIFO

This is a 16Kbytes memory block that stores the data words received from the Media Converter. Its full flag (labeled as TRANSFER_COMPLETE) indicates to the Control Unit that the PRBS emulation must be stopped and its empty flag (labeled as FIFC_EMPTY) indicates to the Downloading Unit that the transfer to the computer is done.

5.1.4.3 Control Unit

Figure 5.16 shows the states diagram of the Control Unit.

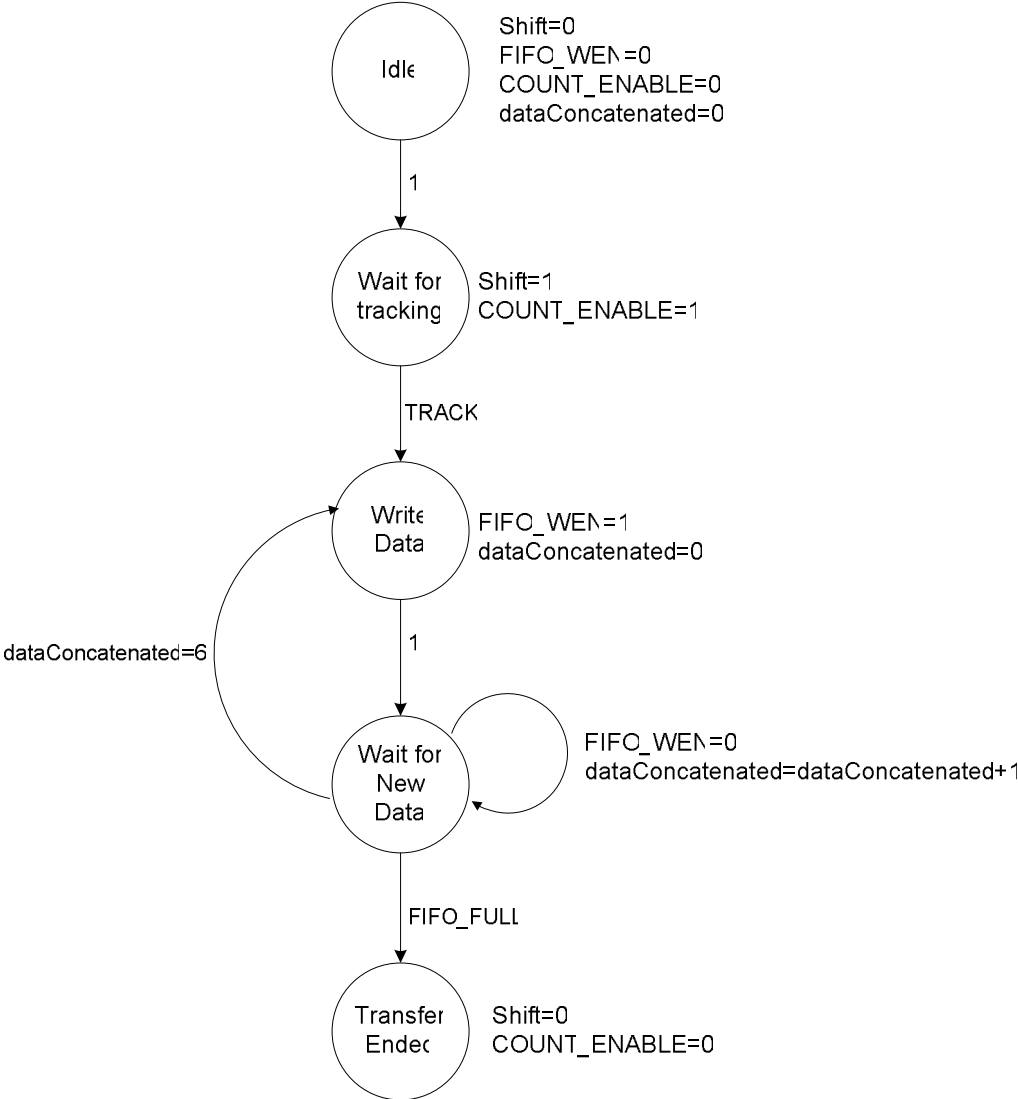


Fig.5.16. State Diagram of the Control Unit

5.1.4.4 Download Unit

This module has been implemented using the PCI communications Core developed by Nallatech. It can establish communication between the computer and the FPGA by means of two different interfaces, the first one is a Memory Map interface and the second one is a DMA interface, it also can be configured by software to use the USB port instead of the PCI bus, which facilitates the board connection to the computer.

For this project the communication through the DMA interface has been configured and established using the USB as communication port with the PC. Figure 5.17 shows the block diagram of the core and the table 5.9 describes its pinout configuration.

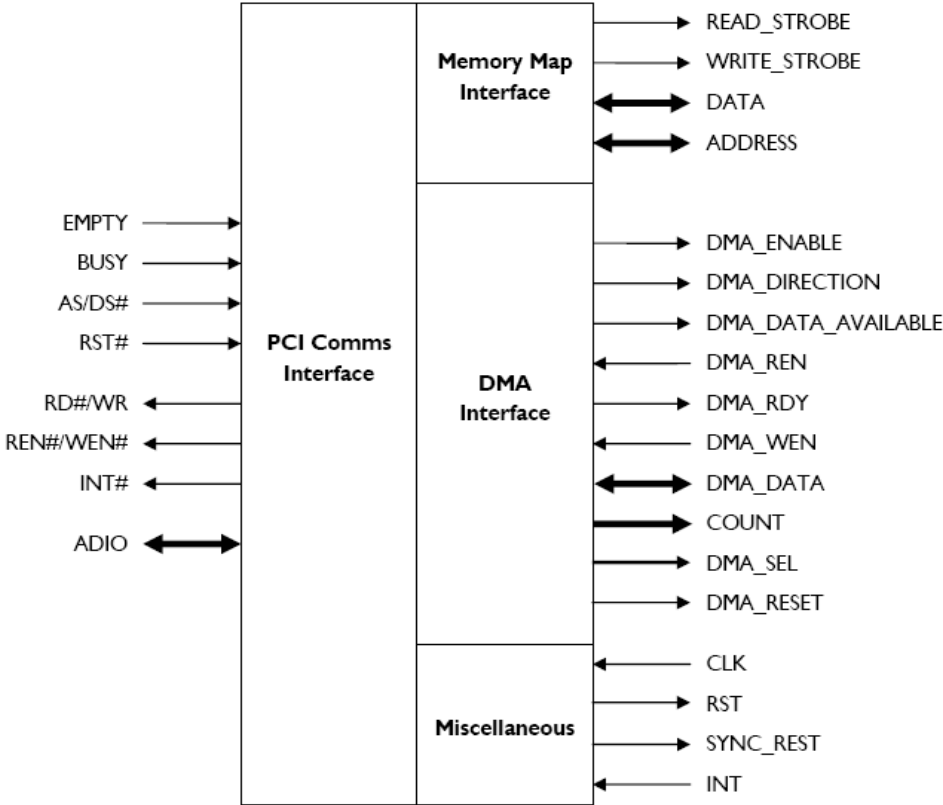


Fig.5.17. PCI Communication Core block Diagram^[1]

¹ Taken from PCI communication Core Datasheet Included in the Annex 3

Table 5.9. Core Pinout Configuration^[1]

Signal	Description	Width	Description
EMPTY	INPUT	1	Indicates that PCI FPGA has data to transfer
BUSY	INPUT	1	Indicates that PCI FPGA can receive data
AS/DS#	INPUT	1	Determines if data from PCI FPGA is address or data (LOW indicates Data)
RST#	INPUT	1	Reset signal from PCI FPGA (Active LOW)
RD#/WR	OUTPUT	1	Read/Write Select to PCI FPGA (LOW indicates Read)
REN#/WEN#	OUTPUT	1	Read/Write Enable to PCI FPGA (Active LOW)
INT#	OUTPUT	1	Interrupt signal to PCI FPGA (Active LOW)
ADIO	INPUT/OUTPUT	32	Data to/from PCI FPGA
READ_STROBE	OUTPUT	1	Read strobe for User Design
WRITE_STROBE	OUTPUT	1	Write strobe for User Design
DATA	INPUT/OUTPUT	32	Data for memory map interface
ADDRESS	INPUT/OUTPUT	31	Address for memory map interface
DMA_ENABLE	OUTPUT	1	Indicates that DMA interface is enabled
DMA_DIRECTION	OUTPUT	1	Indicates direction of DMA interface (LOW indicates data from core to user design, HIGH data from user to core)
DMA_DATA_AVAILABLE	OUTPUT	1	Indicates the DMA interface has data for user design
DMA_REN	INPUT	1	Read enable to read data from DMA interface
DMA_RDY	OUTPUT	1	Indicates the DMA interface is ready to receive data from user design
DMA_WEN	INPUT	1	Write enable to write data to the DMA interface
DMA_DATA	INPUT/OUTPUT	32	Data for DMA interface
COUNT	OUTPUT	32	DMA count. Counts down from number of words to transfer down to zero
DMA_SEL	OUTPUT	4	Selects one of sixteen DMA channels
DMA_RESET	OUTPUT	1	Can be used to reset any buffers used for DMA data transfer
CLK	INPUT	1	Interface clock (typically 40MHz)
RST	OUTPUT	1	Reset for user design
SYNC_RESET	OUTPUT	1	Synchronous reset for user design
INT	INPUT	1	Interrupt signal from user design

¹ Taken from PCI communication Core Datasheet Included in the Annex 3

DMA Interface Configuration

The PCI Communication Interface has two internal register that controls the DMA interface: the DMA Control/Status Register (Address 0x00000000) and the DMA count Register (Address 0x00000001). These registers must be initialized before a transfer begins and must be cleared after it has finished. The initialization and configuration of this interface is achieved by software, using the C++ programs and libraries developed by Nallatech. Further information regarding the DMA hardware configuration can be consulted on the datasheet included in Annex 3.

5.1.5 Reconstruction of the Preamble

Once the modules explained in the previous sections were programmed, took place first attempts to assemble the system. The connection of the first blocks revealed the existence of a problem regarding the reception of the information from the MII interface.

A loopback was connected to the DP83849 national card and several PING transmitting sequences were executed in order to determine what the problem was. In the end it was determined that during clock synchronizing the card discarded 1 to 10 bits of the beginning of the ETHERNET frame. The lost bit corresponded to the Preamble of the frame, hence it was necessary to implement an interface in charge of its reconstruction.

The system implemented generates a new preamble for each ETHERNET sequence received. It has a control unit that responds to the Start of Frame Delimiter (SFD) interrupting the sequence and sending 15 preamble nibbles (1010). The States Machine that executes this operation is shown in figure 5.18.

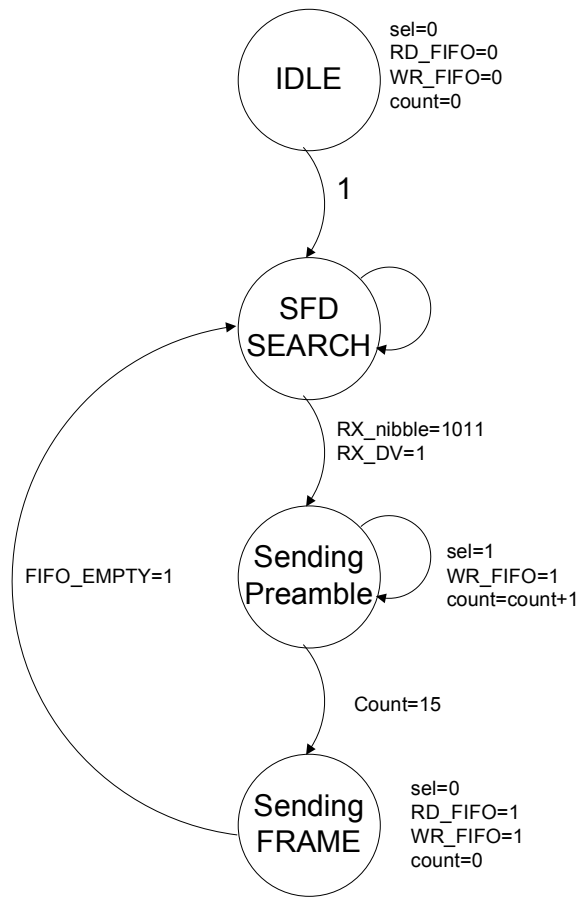


Fig.5.18. State Diagram of the Control Unit

Figure 5.19 shows the hardware implementation of this system. It's important to notice the presence of the FIFO memory, which is used to store the data while the preamble is sent. It has a capacity of 32 words of 5 bits each (5 bits per word).

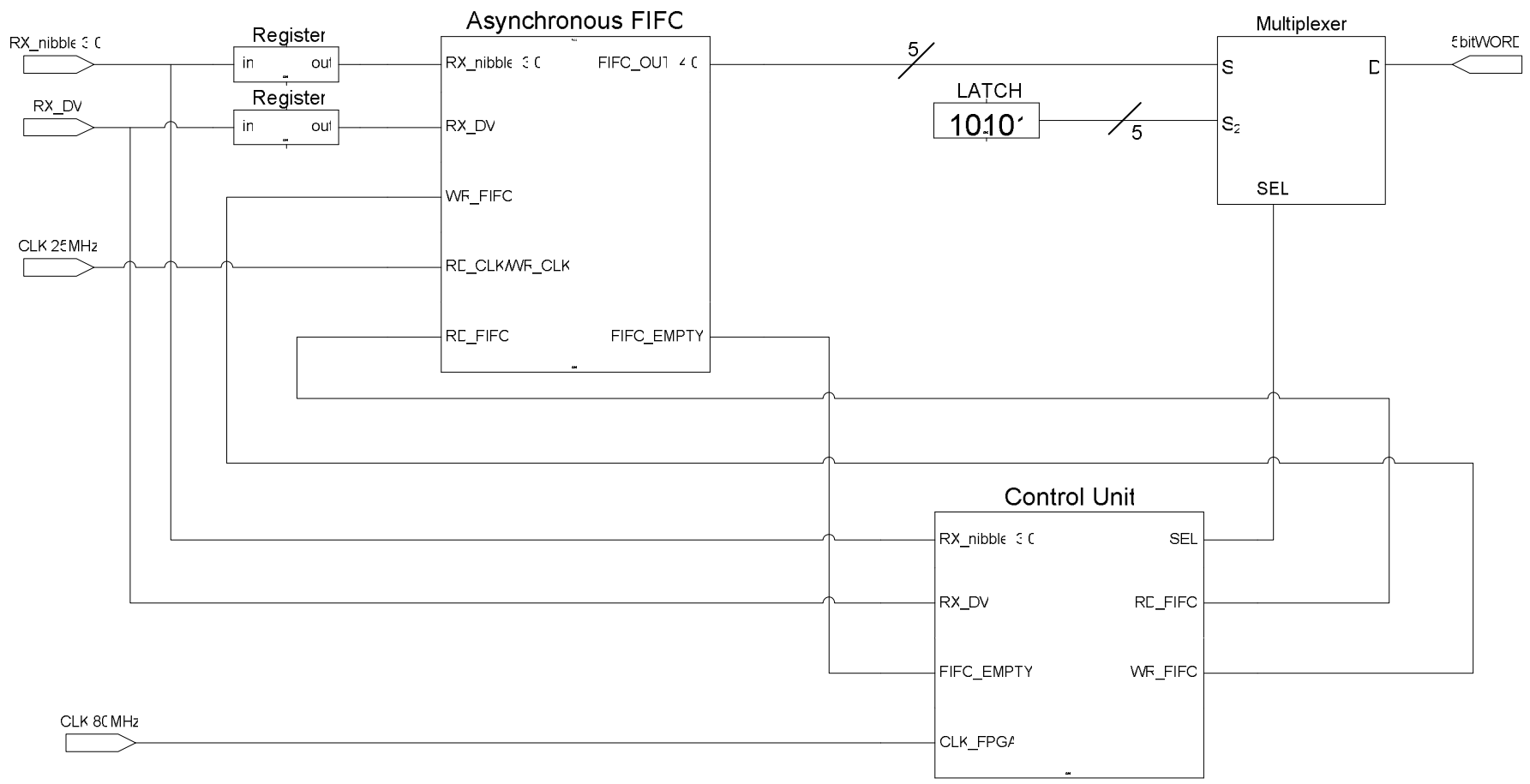


Fig.5.19. Block Diagram of the Preamble Generator

5.2 Software Description

A C++ application programs the FPGA and the DMA interface, then it controls the data transfer and finally it generates the .txt file required by Matlab to compare the received sequence with the transmitted sequence in order to verify the proper operation of the media converter.

The libraries developed by Nallatech were used^[1] both to program the board and to execute the data transfer. The program code can be consulted in the appendix A.5, the flow diagram showed in the figure 5.20 explains the sequence executed by the software in order to download the information.

¹ The libraries are included in the datasheet of the PCI Communication Core shown in ANNEX 3

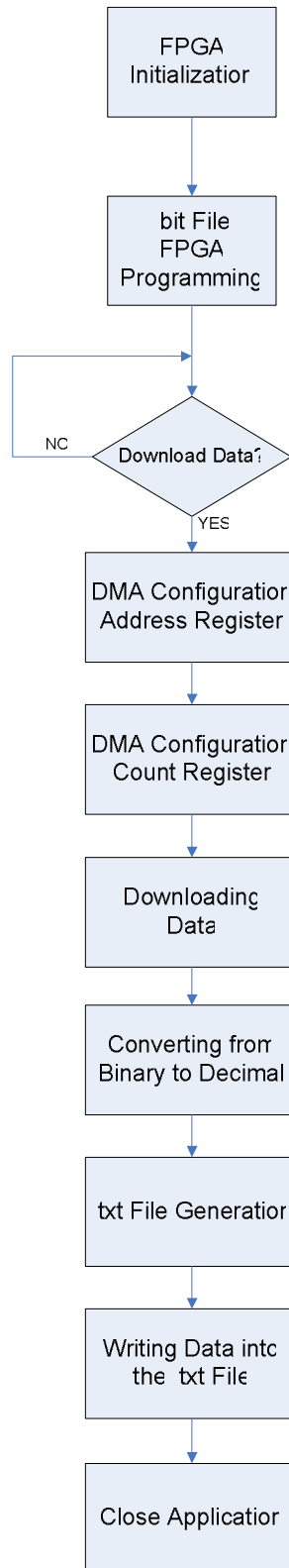


Fig.5.20. Flow Diagram implemented for the Software Application

6 Results Analysis

MII Initialization System

The table 6.1 shows the results obtained when the TEST 1 (see procedure in Appendix A.4) was performed.

Table 6.1 Results obtained from TEST1

Tested Action	Result Obtained	
	First Attempt	Second Attempt
The system continuously executed the initialization sequence of the DP83848	✓	✓
The system initialized properly the DP83848	✗	✓

Table 6.1 shows that the reprogramming sequence executed by the MII initialization System worked as expected. The BMCR register was programmed successfully every single time. However, the reading process presented some problems on the first test, because the system retried the initialization process despite the fact of being properly configured. At this point a Logic Analyzer was used to monitor the MDIO signal finding that the given time between transfers didn't fulfill the minimum value specified by the DPC83848C datasheet. Thus, a module to generate a delay was included; further tests showed that the MII Interface is initialized without any problem.

The table 6.2 shows the results obtained when the TEST 2 (see procedure in Appendix A.4) was performed.

Table 6.2 Results obtained from TEST2

Tested Action	Result Obtained
	First Attempt
The system initialized properly the DP83848 at a 10Mbps	✓
The system initialized properly the DP83848 at a 100Mbps	✓

Table 6.2 demonstrate that the configuration given to the DPC83948 card allows the establishment of the data link speed both for 100Mbps and for 10Mbps.

Figure 6.1 shows the summary of the first PING that was executed in the TEST 3 (see procedure in the Appendix A.4).

```
C:\> Command Prompt
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Ping statistics for 130.192.85.73:
    Packets: Sent = 2979, Received = 2945, Lost = 34 (1% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\>
```

Fig.6.1. PING summary in the Command Prompt

Figure 6.1 shows that when the TEST 3 was performed, 1% of the PING packages were lost. Hence, a Logic Analyzer was used to determine what the problem was. At the end it was discovered that, during clock synchronizing, the card discarded 1 to 10 bits of the preamble. This problem was solved with the implementation of the Preamble Reconstruction Block; the results obtained are shown in the figures 6.2 and 6.3, which show a 0% of PING packages lost.

```
C:\> Command Prompt
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Reply from 130.192.85.73: bytes=32 time<10ms TTL=128
Ping statistics for 130.192.85.73:
    Packets: Sent = 1064, Received = 1064, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\>
```

Fig.6.2. PING summary in the Command Prompt

6.3 shows the position and the quantity of bits lost, concluding that 0.122% of the information is lost through the downloading process. The Nallatech interface was composed of an Asynchronous FIFO with a 16x32 bits capacity and a Control Unit. The internal FIFO was replaced with an Asynchronous FIFO generated with the Xilinx Core generator. The test was repeated several times with FIFOs of 32, 64, 128 and 1024 words capacity. However, the errors were present at the same positions, consequently it was concluded that the problem wasn't provoked by an overflow error in the FIFO.

At this point it was considered to modify the Control Unit, but this module was provided by Nallatech as a "Black Box", making impossible to change its logic design and hardware implementation. The programming of a new interface wasn't possible either, because the USB interface works with the C++ Nallatech Libraries and the protocol followed for the port management and initialization isn't specified by the provider.

The impossibility to correct the data lost in the downloading process implied the evaluation of other solutions. At the beginning the use of the same downloading interface jointly with a module that would locate the number of errors and its positions in the sequence was considered. The downloading process wouldn't be a problem because the first bit lost is located at position 870, which means that 27x32bit words could be received in the computer without the presence of errors. In other words, this solution would allow the location of 32 bit errors with their respective positions in the sequence. Nevertheless, the incapability to isolate more than 32 errors and the inherit complexity of the module needed to locate and generate the report didn't accomplished the simplicity and small space consumption required for the implementation of the internal Testing Unit. Thus, it was discarded.

Afterward, the serial communication through the RS232 port was considered. Nevertheless, its hardware implementation implied the establishing of the interface inside the FPGA, also the physical construction of the RS232 port and finally the

design and implementation of the analog circuits needed to convert the TTL levels voltage into RS232 levels voltage. Moreover, regarding the software, it was necessary to program an application to receive and generate the errors report as a .txt file. These facts and the RS232 obsolescence lead to the conclusion that this wasn't either the best answer to the problem.

At last, considering the simplicity, hardware size, effort and time implementation, it was decided to use the same downloading interface, but considering the bits lost (see table 6.3) in the information analysis. The test configuration may be seen in the figure A.5 in the Appendix A.5.. Hence, it was recommended to program a MatLab application to remove the bits lost from the original PRBS in order to compare it later with the downloaded PRBS, with the advantage that this solution doesn't limit the number of findable error bits and doesn't need further hardware resources from the FPGA.

FPGA Utilization

Figure 6.4 shows the FPGA Virtex IV utilization percentages.

```

Device utilization summary:
-----

Selected Device : 4vsx35ff668-10

Number of Slices:                756 out of 15360    4%
Number of Slice Flip Flops:      706 out of 30720    2%
Number of 4 input LUTs:          1108 out of 30720    3%
    Number used as logic: 964
    Number used as RAMs: 144
Number of bonded IOBs:           108 out of 448      24%
Number of FIFO16/RAMB16s:        12 out of 192      6%
    Number used as RAMB16s:      12
Number of GCLKs:                  9 out of 32      28%
Number of DCM_ADVs:              4 out of 8        50%

```

Fig.6.4. FPGA Utilization

This summary shows the minimum requirements needed to implement the Media Converter, which are far from the maximum capacity of the FPGA Virtex IV.

```
Timing Summary:
-----
Speed Grade: -10

Minimum period: 5.840ns (Maximum Frequency: 171.223MHz)
Minimum input arrival time before clock: 5.484ns
Maximum output required time after clock: 6.465ns
Maximum combinational path delay: 6.539ns

Timing Detail:
-----
All values displayed in nanoseconds (ns)
```

Fig.6.5. Timing Summary of the Media Converter

Figure 6.5 shows that the maximum clock frequency of the Media Converter is 171.223MHz. This means that the system could add more control words or even more logic levels without affect the ETHERNET data rate. This figure also shows that the Media Converter has an input constraint of 5.484ns and an output constraint of 6.465ns. This information is useful for the interaction of the Media Converter with other applications and/or systems.

7 Conclusions and Recommendations

7.1 Conclusions

- The Media Converter is capable to establish Fast Ethernet Communication both at 100Mbps and at 10Mbps.
- The Media Converter counts with two possible test methodologies and their respective hardware, the first one uses the Fast Ethernet Link as a data source, collecting the results with the network monitoring Tools embedded on Windows. The second one consists on the internal emulation, transmitting, receiving and downloading of a PRBS that is verified with Matlab.
- The Media Converter uses a 4% of the total slices of the FPGA Virtex IV.
- The Media Converter minimum FPGA requirements are: 706 Flip Flops, 1108 LUTs (Look Up Tables) with 4 input (964 used as logic and 144 used as RAM, random access memory), 108 bonded IOBs (Input/Output Blocks), 12 FIFO16/RAMB16s, 9 GCLKs (Gated Clocks) and finally 4 DCMs.
- The Media Converter has maximum combinational path delay of 6.539ns, and a Maximum Clock frequency of 171.223MHz.
- The Media Converter has an input constraint of 5.484ns and an output constraint of 6.465ns.

7.2 Recommendations

- The main recommendation is to use both testing methods in the Media Converter depuration process, because the Internal Testing Unit allows to debug the system, whereas the establishment of Fast Ethernet Communication validates the implementation of the system as a commercial product compatible with a worldwide standard.

- Request to Nallatech a debugged version of its PCI communications Core. Specify the presence of errors and their positions when the interface is configured to execute DMA transfers through the USB port of the Nallatech card. Also inform that it was concluded that the error is located in the Control Unit of the Core. Finally, explain that error location was deduced based on the lack of satisfactory results when hardware modifications were made to the other modules inside the Core.

- Acquire specialized network analysis software in order to generate statistics to validate the system. This software should have access to a set of capabilities and analysis variables such as:
 1. An accurate calculation of the data lost percentage.
 2. The number of resending processes and the time needed to send the data.
 3. Operation time with error free and consequently the error frequency to determine its possible periodicity.
 4. A debugging tool capable of emulate Ethernet information from an algorithm or file given by the user. This feature could allow the transmission of a more complex PRBS directly to the MII interface and later to the system, which could be an alternative to the Internal Testing Unit.

8 Bibliography

- [1] CrossLink, "The Aerospace Corporation Magazine of advances in aerospace technology" [online]. *Includes an article that explains the FEC.*
< <http://www.aero.org/publications/crosslink/winter2002/04.html>>
- [2] WIKIPEDIA, "The Free Encyclopedia" [online]. *Optical Fibers.*
< http://en.wikipedia.org/wiki/Optical_fiber>.
- [3] Mouse Home Page, "The Radio Network Processor" [online]. *Includes a section regarding the Reed Solomon Algorithm with several articles and related documents.*
< <http://www.radionetworkprocessor.com/reed-solomon.html>>.
- [4] POF ALL Association, Internet Site [online]. *Plastical Optical Fiber.*
< http://www.pofac.de/pofac/en/what_are_pof/step_index.php>.
- [5] Daum,W.;Krauser. J.; Zamzow, P.; Ziemann,O.**Polymer Optical Fibers for Data Communication** . Germany, Springer-Verlag Berlin Heidelberg, 2002.
- [6] Nallatech. **XtremeDSP Development Kit-IV User Guide**.Nallatech Limited, 2005.
- [7] Spurgeon, Charles E..**Ethernet: The Definitive Guide** . First Edition, O'Reilly Editorial, USA, 2000.

9 Appendix

A.1 Glossary, abbreviations and Symbols

A

ASM: Attached Synchronizing Mark

B

BMCR: Basic Mode Control Register

C

CRGU: Clock and Reset Generation Unit

CPU: Central Processing Unit

D

DCM: Digital Clock Management

DCU: Data Compilation Unit

DMA: Direct Memory Access

F

FEC: Forward Error Correction

FIFO: First In First Out

FPGA: Field Programmable Logic Array

G

GCLK: Gated Clock

H

HDL: Hardware Description Language

I

IEEE: Institute of Electrical and Electronics Engineers

IFG: Inter Frame Gap

IP: Internet Protocol

IOB: Input/Output Block

L

LAN: Local Area Network

LUT: Look Up Table

M

MAC: Medium Access Controller
MDC: Management Data Clock
MDIO: Management Data Input/Output
MII: Media Independent Interface

O

OSI: Open System Interconnection

P

PAM: Pulse Amplitud Modulation
PCI: Peripheral Component Interconnect
PHY: Physical Layer of the OSI Reference Model
POF: Plastic Optical Fiber
POF-ALL: Paving the Optical Future with Affordable Lightning-fast Links
PRBS: Pseudo Random Bit Sequence

R

RAM: Random Access Memory
ROM: Read Only Memory

S

SFD: Start Frame Delimiter

T

TGU: Traffic Generation Unit

U

USB: Universal Serial Bus

V

VHDL: Very High Speed Integrated Circuit Hardware Description Language

A.2 Company Information

PhotonLab is a 250 square meters laboratory constituted by different work areas, such as the optical system and network experiment area, an electronics lab, a dark room, a filtered air system in a class 1000 clean room and a special design that isolates it from external vibration. Furthermore, this facility is equipped with test and measurement equipment, several BER tester (12.5 Gbit/s and 40Gbit/s), 50 GHz oscilloscopes, optical spectrum analyzers, electrical and optical network analyzers up to 40 GHz, numerous optical and electro-optical components and access to 240 Km of optical fiber deployed in the metropolitan area of Turin.

A.3 Testing Procedures

This section explains the field tests performed to verify the proper operation of the systems implemented. The FPGA was programmed using the software provided by Nallatech.

TEST1

The figure A.1 shows the setup of the test performed in order to verify the programming system that initializes the MII interface.

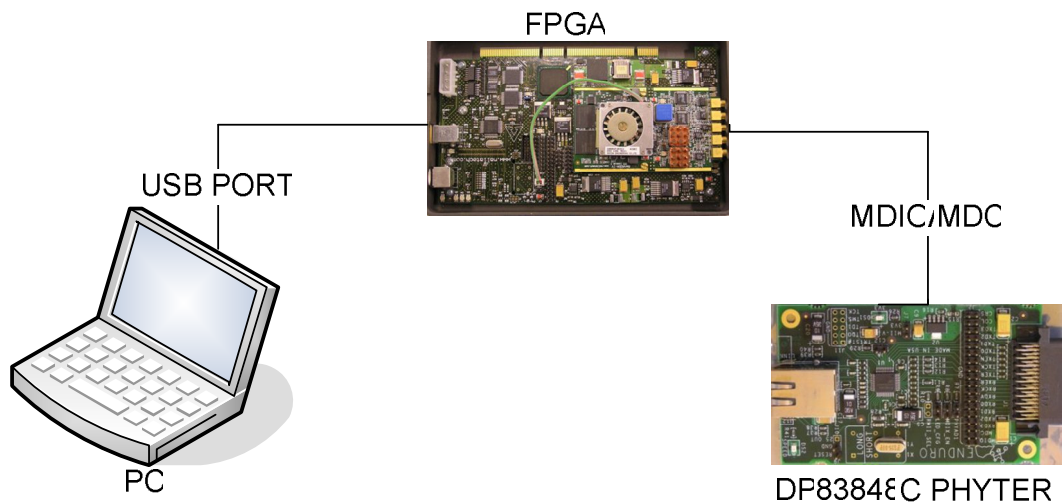


Fig.A.1. Test 1 Configuration

Test procedure:

- 1) Connect the Reset jumper of the DP83848C.
- 2) Start the programming sequence with the FPGA.
- 3) Verify that the programming system execute continuously the initialization sequence.
- 4) Remove the Reset jumper of the DP83848C.
- 5) Verify the proper initialization of the DP83848C.

TEST 2

The figure A.2 shows the set up of the test performed to the MII interface in order to verify its capacity to establish the link specifications.

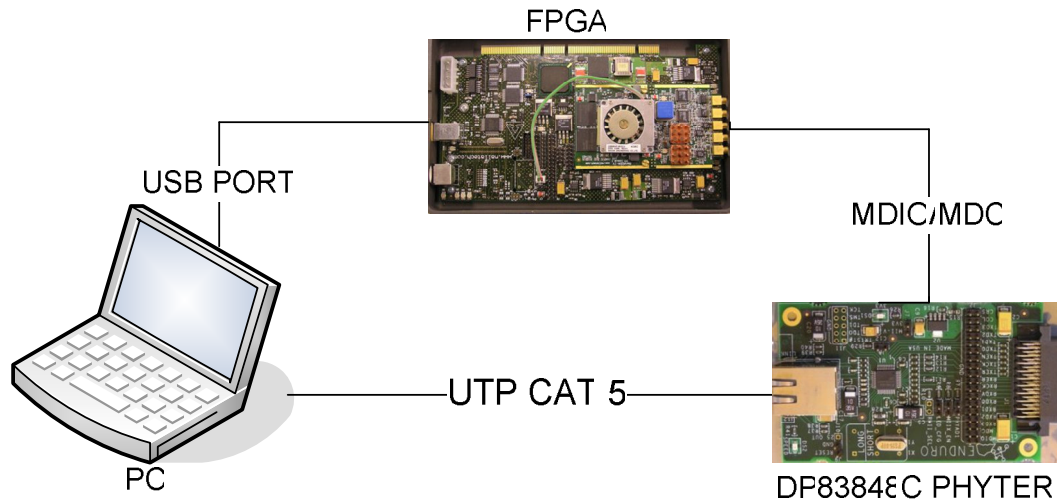


Fig.A.2. Test 2 Configuration

Test Procedure:

- 1) Configure the computer link at 100 Mbps and with the auto-negotiation capability.
- 2) Program the DP83848C at 10Mbps without the auto-negotiation capability.
- 3) Check with the Windows Network Tools the reconfiguration of the computer link due to the auto-negotiation process.

This test was performed configuring the PC link at 10Mbps and the card at 100Mbps.

TEST 3

The figure A.2 shows the set up of the test performed in order to verify the reliability of the data transmission and receiving through and from the MII interface. It was performed to corroborate the proper operation of the Internal Testing System in the FPGA.

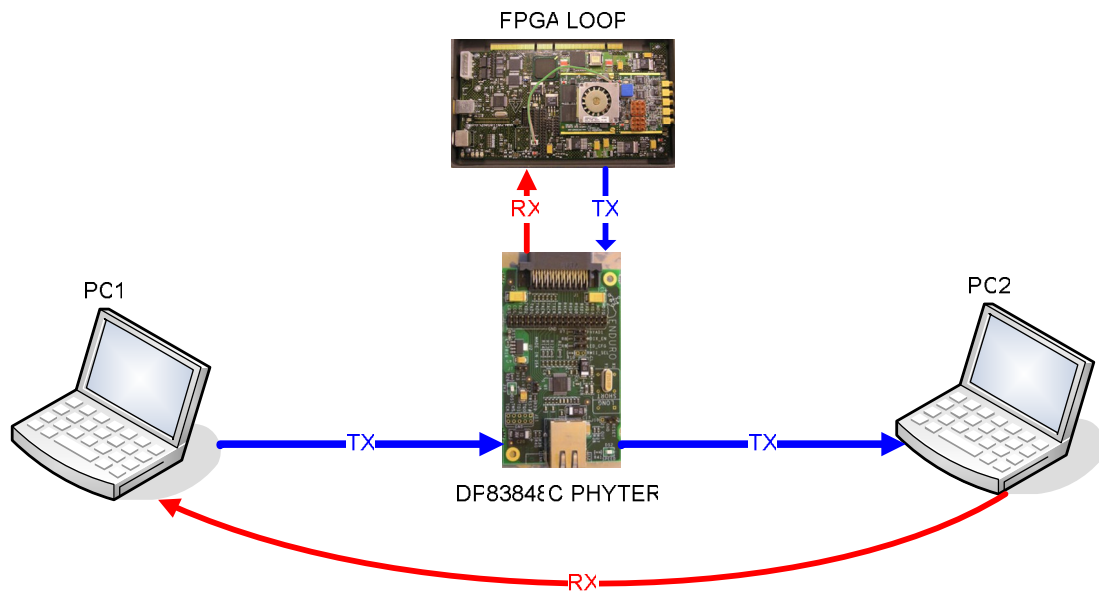


Fig.A.3. Test 3 Configuration

Test procedure:

- 1) Program the DP83848C at 10Mbps without the auto-negotiation capability.
- 2) Execute a PING from the PC1.
- 3) Wait the answer from PC2.

TEST4

The figure A.4 shows the set up of the test performed in order to verify the testing system implemented inside of the FPGA .

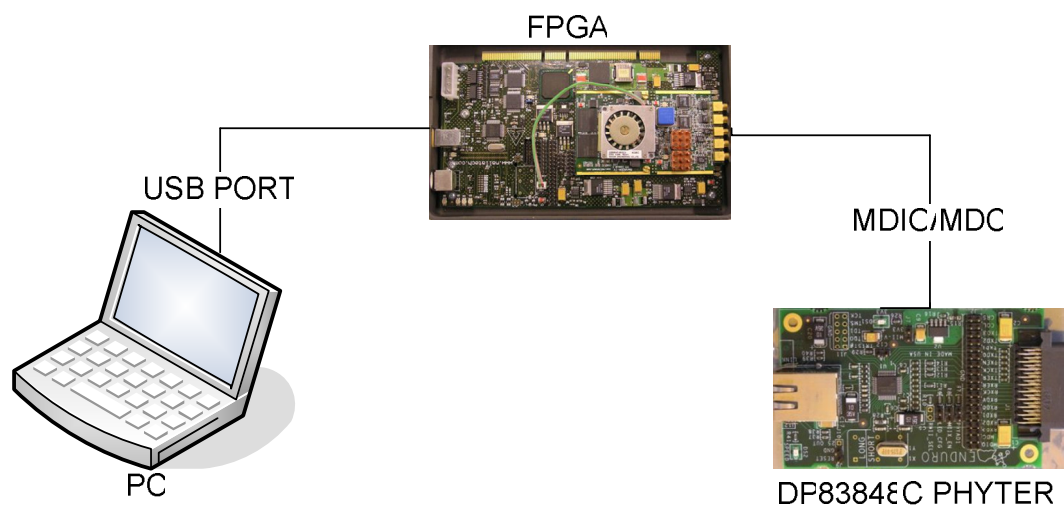


Fig.A.4. Test 4 Configuration

Test procedure:

- 1) Disable the MII Interface as a data source.
- 2) Configure the DP83848C.
- 3) Start the testing transmission inside the FPGA.
- 4) Download the PRBS sequence into the PC.
- 5) Compare the downloaded sequence with the original one.

A.4 Testing Set up with the POF as loopback

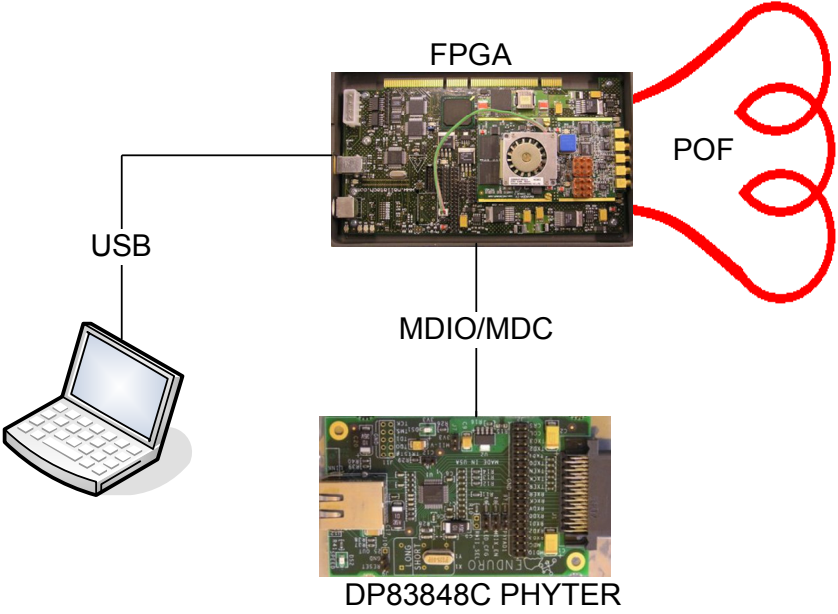


Fig.A.5. Testing Setup with the POF as a LOOPBACK

A.5 C++ Programming Code of the Software Application

The following lines show the libraries inclusion, the DWORD global variables declaration, the programming files assignment and finally the local variables declaration and its initialization takes place.

```
#include "dimesdl.h"
#include "vidime.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

DWORD ModuleNumber=0;

DWORD PrimaryFPGADeviceNum=1;

DWORD SecondaryFPGADeviceNum=0;

char Filename1[]="mediaconverter.bit";
char Filename2[]="osc_clk80DAC80.bit";

int main(int argc, char* argv[])
{
    // FPGA Variables Declaration
    LOCATE_HANDLE hLocate = NULL;
    DWORD ErrorNum, NumOfCards, LoopCtr;
    char ErrorString[1000];
    DIME_HANDLE hCard1;
    double ActualFrequency;
    VIDIME_HANDLE viHandle;

    DWORD addressControl= 0;// DMA Control/Status Register Adress
    DWORD addressCount= 1;// DMA Count Register Adress
    DWORD regCONTROL= 3; // Initialization Value of the Control Register
    DWORD regCOUNT= 256; // Initalization Value of the Count Register
    DWORD dmaRST =64; // Value to Reset the DMA through the Control Reg
    DWORD transfer_completed= 0;// Value to finish the Transfer

    DWORD Result= 0;
    DWORD j;
    DWORD i;
    DWORD n;
    DWORD m;
    DWORD k;
    DWORD h;
    DWORD t;
    double d;
    double toPRINT;
    DWORD ReadData[256]; // Array where are loaded the data transfer
    DWORD DATA [4096];// Number of 32 bit data words to download
    BOOL binDATA [32];// 32 bit word to be converted to double
    BOOL printDATA[5];// Data ready to be printed

    FILE *fp;
    char file_name[40];// File Generated
```


At this point the communication with the FPGA is established, the Card specifications are obtained and it's opened in order to initiate the programming process.

```
//Call the function to locate all Nallatech cards on the PCI interface.
if( (hLocate = DIME_LocateCard(dlUSB,mbtALL,NULL,dldrDEFAULT,dlDEFAULT)) == NULL)
{//Error hLocate NULL
    //Print the error then terminate the program
    DIME_GetError(NULL,&ErrorNum,ErrorString);
    printf("Error Number %d\n", ErrorNum);
    printf("%s\n",ErrorString);
    exit(1);
}

//Determine how many Nallatech cards have been found.
NumOfCards = DIME_LocateStatus(hLocate,0,dlNUMCARDS);
printf("%d Nallatech card(s) found.\n", NumOfCards);

//Get the details for each card detected.
for (LoopCntr=1; LoopCntr<=NumOfCards; LoopCntr++)
{
    printf("Details of card number %d, of %d:\n",LoopCntr,NumOfCards);

    printf("\tThe card driver for this card is a %s.\n"
    ,(char*)DIME_LocateStatusPtr(hLocate,LoopCntr,dlDESCRIPTION));
    printf("\tThe cards motherboard type is %d.\n"
    ,DIME_LocateStatus(hLocate,LoopCntr,dlMBTYPE));
}

//At this stage we now have all the information we need to open a card up.

//Open up the first card found.
hCard1 = DIME_OpenCard(hLocate,1,dccOPEN_NO_OSCILLATOR_SETUP);
if (hCard1 == NULL) //check to see if the open worked.
{
    printf("Card Number One failed to open.\n");
    DIME_CloseLocate(hLocate);
    printf ("\nPress return to terminate the application.\n");
    getchar();
    return(1);
}
```

Once the card is opened, the program enables the resets, sets the clocks, configures the FPGAs^[1] and executes a general reset.

```
//Enable the resets
// Reset the circuit and clear the PCI FIFOs for the card
DIME_CardResetControl(hCard1,drONBOARDFPGA, drENABLE,0);
DIME_CardResetControl(hCard1,drSYSTEM, drENABLE,0);
DIME_CardResetControl(hCard1,drINTERFACE, drTOGGLE,0);

//Set the clocks
DIME_SetOscillatorFrequency(hCard1,1,80.000,&ActualFrequency);
printf("ClkA: Actual frequency is %f.\n",ActualFrequency);

DIME_SetOscillatorFrequency(hCard1,2,40.000,&ActualFrequency);
printf("ClkB: Actual frequency is %f.\n",ActualFrequency);

DIME_SetOscillatorFrequency(hCard1,3,60.000,&ActualFrequency);
printf("ClkC: Actual frequency is %f.\n",ActualFrequency);

//Configure the FPGAs
//Now configure the modules primary FPGA with the bif file.
DIME_ConfigDevice(hCard1,Filename2,ModuleNumber,SecondaryFPGADeviceNum,0,0);
//Now configure the modules primary FPGA with the bif file.
DIME_ConfigDevice(hCard1,Filename1,ModuleNumber,PrimaryFPGADeviceNum,0,0);

//Disable the resets
DIME_CardResetControl(hCard1,drONBOARDFPGA, drDISABLE,0);
DIME_CardResetControl(hCard1,drSYSTEM, drDISABLE,0);

Sleep(500);
//Enable the resets
DIME_CardResetControl(hCard1,drONBOARDFPGA, drENABLE,0);
DIME_CardResetControl(hCard1,drSYSTEM, drENABLE,0);
DIME_CardResetControl(hCard1,drINTERFACE, drTOGGLE,0);
Sleep(500);
//Disable the resets
DIME_CardResetControl(hCard1,drONBOARDFPGA, drDISABLE,0);
DIME_CardResetControl(hCard1,drSYSTEM, drDISABLE,0);
Sleep(500);
//Give it a couple second for the embedded proc to sort out
//the DIMETalk network i.e. reset control etc.

printf("\nCard succesfully programmed!\nPress return to continue\n");
getchar();
```

¹ The nallatech board has two FPGAs, a Virtex-II Pro and a Virtex IV.-

Subsequently the DMA configuration and the downloading process takes place.

```
// Transfer Process
//Open the handle to vidime
viHandle = viDIME_Open(hCard1, 0);

for(j=0;j<16;j++){
//Register Initialization
viDIME_WriteRegister (viHandle,addressCount,regCOUNT,5000);
viDIME_WriteRegister (viHandle,addressControl,regCONTROL,5000);
//Data transfer to the PC
Result = viDIME_DMARead(viHandle,ReadData,regCOUNT,0, NULL, NULL, 5000);
//Ending Transfer
viDIME_WriteRegister (viHandle,addressControl,transfer_completed,5000);
viDIME_WriteRegister (viHandle,addressControl,dmaRST,5000);
Sleep(5);

if (Result != 0)
{
    printf("Reading Process Failed!.\n");
    break;
}
else{
    printf("Data %d tranfered succesfully!.\n",j);
    }
    for(i=0;i<256;i++)
    {
        DATA[j*256+i]=ReadData [i];
    }
}
} // end for

// Transfer Completed
```

At this point is opened the .txt file, and begins the writing process, which implies the data type conversion in order to print it.

```
//File Name assignment
strcpy(file_name,"pofall.txt");
fp=fopen(file_name,"a");

// File writing
for(j=0;j<4096;j++)
{
```

This is the method that converts the data downloaded from DWORD to binary, returns the binary data in the printDATA array.

```

//*****BEGIN*****
// dword2bin
    DWORD q, r, s;
    for(t=0;t<32;t++)
    {
        binDATA[t]=FALSE;
    }
    q= DATA[j];
    s=0;
    if (q != 0)
    {
        while (q != 1)
        {
            r = q % 2;
            if (r == 0)
            {
                q = q / 2;
                binDATA[s] = FALSE;
            }
            else
            {
                q = (q - 1) / 2;
                binDATA[s] = TRUE;
            }
            s++;
        }
        binDATA[s] = TRUE;
    }
}
//*****END*****
//
for(n=6;n>0;n--)
{
    for (m=5;m>0;m--){
        k= n*8-6+m;

        printDATA [m-1]= binDATA[k];

    }// cierre for m
}

```

The next method converts the data contained in printDATA from binary to double allowing its printing in the .txt file.

```
//*****BEGIN*****  
// bin2double  
  
    d = 0;  
  
    for (h = 0; h < 5; h++)  
    {  
        if (printDATA[h] == 1)  
        {  
            d = d + pow(2, h);  
        }  
    }  
  
|  
//*****END*****  
  
    fprintf(fp, "%f\n", toPRINT);  
  
        } //closing for n  
    } // closing for j
```

Finally , the file and the card are closed.

```
//File Closed  
fclose (fp);  
//Close the card  
viDIME_Close(viHandle);  
  
DIME_CloseCard(hCard1); //Closes down the card.  
  
//Finally the locate is closed  
DIME_CloseLocate(hLocate);  
  
printf ("\nPress return to terminate the application.\n");  
getchar();  
return 0;
```

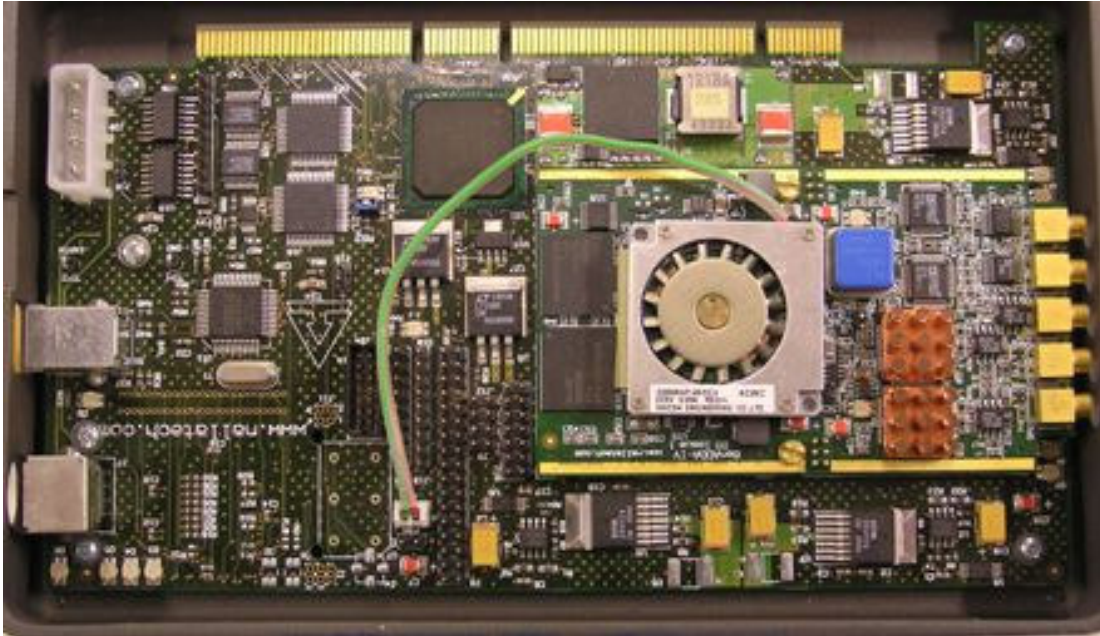
10 Annex

ANNEX 1 Photos of the cards and devices utilized to implement the system.

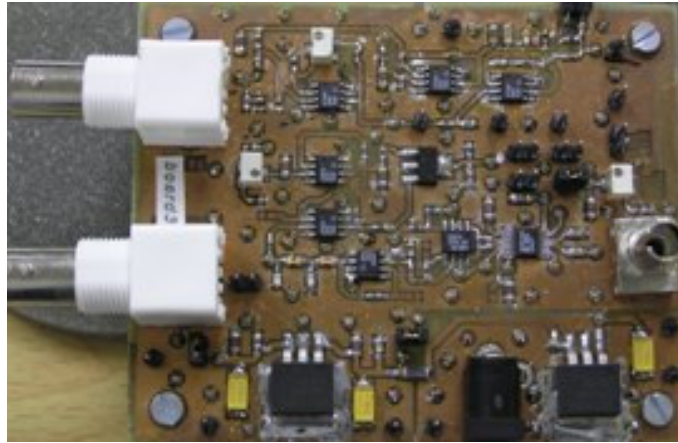
DP83848 National Instruments Card



FPGA Virtex-4 Nallatec Board



POF daughterboard



ANNEX 2 DP83848 National Instruments Card



February 2007

DP83848C PHYTER® - Commercial Temperature Single Port 10/100 Mb/s Ethernet Physical Layer Transceiver

General Description

The DP83848C is a robust fully featured 10/100 single port Physical Layer device offering low power consumption, including several intelligent power down states. These low power modes increase overall product reliability due to decreased power dissipation. Supporting multiple intelligent power modes allows the application to use the absolute minimum amount of power needed for operation.

The DP83848C includes a 25MHz clock out. This means that the application can be designed with a minimum of external parts, which in turn results in the lowest possible total cost of the solution.

The DP83848C easily interfaces to twisted pair media via an external transformer. Both MII and RMII are supported ensuring ease and flexibility of design.

The DP83848C features integrated sublayers to support both 10BASE-T and 100BASE-TX Ethernet protocols, which ensures compatibility and interoperability with all other standards based Ethernet solutions.

The DP83848C is offered in a small form factor (48 pin LQFP) so that a minimum of board space is needed.

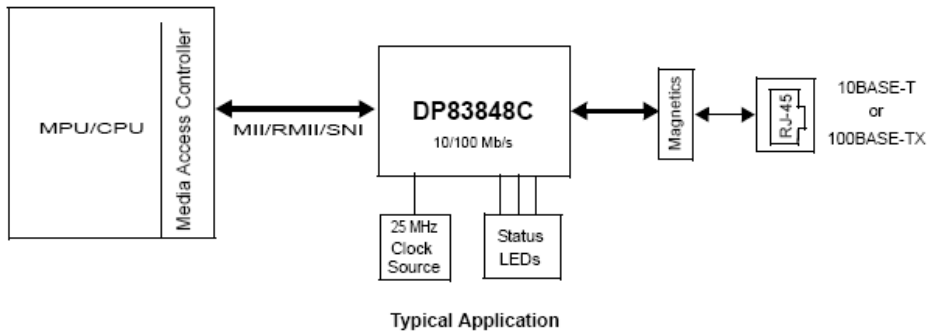
Applications

- High End Peripheral Devices
- Industrial Controls and Factory Automation
- General Embedded Applications

Features

- Low-power 3.3V, 0.18µm CMOS technology
- Low power consumption < 270mW Typical
- 3.3V MAC Interface
- Auto-MDIX for 10/100 Mb/s
- Energy Detection Mode
- 25 MHz clock out
- SNI Interface (configurable)
- RMII Rev. 1.2 Interface (configurable)
- MII Serial Management Interface (MDC and MDIO)
- IEEE 802.3u MII
- IEEE 802.3u Auto-Negotiation and Parallel Detection
- IEEE 802.3u ENDEC, 10BASE-T transceivers and filters
- IEEE 802.3u PCS, 100BASE-TX transceivers and filters
- Integrated ANSI X3.263 compliant TP-PMD physical sub-layer with adaptive equalization and Baseline Wander compensation
- Error-free Operation up to 137 meters
- Programmable LED support Link, 10 /100 Mb/s Mode, Activity, and Collision Detect
- Single register access for complete PHY status
- 10/100 Mb/s packet BIST (Built in Self Test)
- 48-pin LQFP package (7mm) x (7mm)

System Diagram



PHYTER® is a registered trademark of National Semiconductor.

© 2007 National Semiconductor Corporation

www.national.com

DP83848C PHYTER® — Commercial Temperature Single Port 10/100 Mb/s Ethernet Physical Layer Transceiver

ANNEX 3 Nallatech PCI Communication Core Application Note

	APPLICATION NOTE	
	PCI Communications Core	
Sep 2004	www.nallatech.com	NT302-0000 – Issue 6

1. Features

- FPGA IP Core for insertion into user application designs
- Optimised for use with Xilinx® Virtex®, Virtex-E, Virtex-II, Spartan-II and Virtex-II Pro FPGAs
- Core controls data/address/controls of DIME PCI communications interface
- Operates at PCI Comms frequency (typically 40MHz)
- Core provided in VHDL format.
- Includes example design
- Provided testbench to allow user to evaluate the core

2. Core Overview

2.1 Overview

The PCI Communications Core was developed to provide Nallatech board users with an easy interface to communicate with the PCI interface FPGA commonly used on Nallatech carrier cards (Ballynuey2, Benera, Bennuey, etc...).

The core provides the user with a simple interface that comes in two parts:

- Memory Map Interface (for register read/writes, RAM read/writes)
- DMA Interface (for burst data read/writes)

These different blocks are discussed in Section3.

Note: This datasheet will not discuss the interface between the PCI FPGA and the User Application FPGA. For this please read the appropriate carrier card User Guide.

ANNEX 4 Virtex 4 Family Overview



Virtex-4 Family Overview

DS112 (v2.0) January 23, 2007

Preliminary Product Specification

General Description

Combining Advanced Silicon Modular Block (ASMBL™) architecture with a wide variety of flexible features, the Virtex™-4 Family from Xilinx greatly enhances programmable logic design capabilities, making it a powerful alternative to ASIC technology. Virtex-4 FPGAs comprise three platform families—LX, FX, and SX—offering multiple feature choices and combinations to address all complex applications. The wide array of Virtex-4 hard-IP core blocks includes the PowerPC™ processors (with a new APU interface), tri-mode Ethernet MACs, 622 Mb/s to 6.5 Gb/s serial transceivers, dedicated DSP slices, high-speed clock management circuitry, and source-synchronous interface blocks. The basic Virtex-4 building blocks are enhancements of those found in the popular Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X product families, so previous-generation designs are upward compatible. Virtex-4 devices are produced on a state-of-the-art 90-nm copper process using 300-mm (12-inch) wafer technology.

Summary of Virtex-4 Family Features

- Three Families — LX/SX/FX
 - Virtex-4 LX: High-performance logic applications solution
 - Virtex-4 SX: High-performance solution for digital signal processing (DSP) applications
 - Virtex-4 FX: High-performance, full-featured solution for embedded platform applications
- Xesium™ Clock Technology
 - Digital clock manager (DCM) blocks
 - Additional phase-matched clock dividers (PMCD)
 - Differential global clocks
- XtremeDSP™ Slice
 - 18 x 18, two's complement, signed Multiplier
 - Optional pipeline stages
 - Built-in Accumulator (48-bit) and Adder/Subtractor
- Smart RAM Memory Hierarchy
 - Distributed RAM
 - Dual-port 18-Kbit RAM blocks
 - Optional pipeline stages
 - Optional programmable FIFO logic automatically remaps RAM signals as FIFO signals
 - High-speed memory interface supports DDR and DDR-2 SDRAM, QDR-II, and RLDRAM-II.
- SelectIO™ Technology
 - 1.5V to 3.3V I/O operation
 - Built-in ChipSync™ source-synchronous technology
 - Digitally controlled impedance (DCI) active termination
 - Fine grained I/O banking (configuration in one bank)
- Flexible Logic Resources
- Secure Chip AES Bitstream Encryption
- 90-nm Copper CMOS Process
- 1.2V Core Voltage
- Flip-Chip Packaging including Pb-Free Package Choices
- RocketIO™ 622 Mb/s to 6.5 Gb/s Multi-Gigabit Transceiver (MGT) [FX only]
- IBM PowerPC RISC Processor Core [FX only]
 - PowerPC 405 (PPC405) Core
 - Auxiliary Processor Unit Interface (User Coprocessor)
- Multiple Tri-Mode Ethernet MACs [FX only]

Table 1: Virtex-4 FPGA Family Members

Device	Configurable Logic Blocks (CLBs) ⁽¹⁾				XtremeDSP Slices ⁽²⁾	Block RAM		DCMs	PMCDs	PowerPC Processor Blocks	Ethernet MACs	RocketIO Transceiver Blocks	Total I/O Banks	Max User I/O
	Array ⁽³⁾ Row x Col	Logic Cells	Slices	Max Distributed RAM (Kb)		18 Kb Blocks	Max Block RAM (Kb)							
XC4VLX15	64 x 24	13,824	6,144	96	32	48	864	4	0	N/A	N/A	N/A	9	320
XC4VLX25	96 x 28	24,192	10,752	168	48	72	1,296	8	4	N/A	N/A	N/A	11	448
XC4VLX40	128 x 36	41,472	18,432	288	64	96	1,728	8	4	N/A	N/A	N/A	13	640
XC4VLX60	128 x 52	59,904	26,624	416	64	160	2,880	8	4	N/A	N/A	N/A	13	640
XC4VLX80	160 x 56	80,640	35,840	560	80	200	3,600	12	8	N/A	N/A	N/A	15	768
XC4VLX100	192 x 64	110,592	49,152	768	96	240	4,320	12	8	N/A	N/A	N/A	17	960
XC4VLX160	192 x 88	152,064	67,584	1056	96	288	5,184	12	8	N/A	N/A	N/A	17	960
XC4VLX200	192 x 116	200,448	89,088	1392	96	336	6,048	12	8	N/A	N/A	N/A	17	960

© 2004–2007 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at <http://www.xilinx.com/legal.htm>. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners. All specifications are subject to change without notice.

DS112 (v2.0) January 23, 2007
Preliminary Product Specification

www.xilinx.com

1

ANNEX 5 Second Part of the IEEE 802.3 Protocol

**Information technology—
Telecommunications and information exchange between systems—
Local and metropolitan area networks—Specific requirements—**

Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications

SECTION TWO: This section includes Clauses 21 through 33 and Annexes 22A through 32A.

21. Introduction to 100 Mb/s baseband networks, type 100BASE-T

21.1 Overview

100BASE-T couples the IEEE 802.3[®] CSMA/CD MAC with a family of 100 Mb/s Physical Layers. While the MAC can be readily scaled to higher performance levels, new Physical Layer standards are required for 100 Mb/s operation.

The relationships between 100BASE-T, the existing IEEE 802.3[®] (CSMA/CD MAC), and the ISO/IEC Open System Interconnection (OSI) reference model is shown in Figure 21-1.

100BASE-T uses the existing IEEE 802.3[®] MAC layer interface, connected through a Media-Independent Interface layer to a Physical Layer entity (PHY) sublayer such as 100BASE-T4, 100BASE-TX, or 100BASE-FX.

100BASE-T extends the IEEE 802.3[®] MAC to 100 Mb/s. The bit rate is faster, bit times are shorter, packet transmission times are reduced, and cable delay budgets are smaller—all in proportion to the change in bandwidth. This means that the ratio of packet duration to network propagation delay for 100BASE-T is the same as for 10BASE-T.

21.1.1 Reconciliation Sublayer (RS) and Media Independent Interface (MII)

The Media Independent Interface (Clause 22) provides an interconnection between the Media Access Control (MAC) sublayer and Physical Layer entities (PHY) and between PHY Layer and Station Management (STA) entities. This MII is capable of supporting both 10 Mb/s and 100 Mb/s data rates through four bit wide (nibble wide) transmit and receive paths. The Reconciliation sublayer provides a mapping between the signals provided at the MII and the MAC/PLS service definition.

21.1.2 Physical Layer signaling systems

The following portion of this standard specifies a family of Physical Layer implementations. 100BASE-T4 (Clause 23) uses four pairs of ISO/IEC 11801 Category 3, 4, or 5 balanced cabling. 100BASE-TX (Clauses

ANNEX 6 Xilinx Core Asynchronous FIFO v6.1



Asynchronous FIFO v6.1

DS232 November 11, 2004

Product Specification

Introduction

The Asynchronous FIFO is a First-In-First-Out memory queue with control logic that performs management of the read and write pointers, generation of status flags, and optional handshake signals for interfacing with the user logic.

About This Revision

Version 6.1 is the final release of the Asynchronous FIFO core. For new designs, Xilinx suggests you use the FIFO Generator Logiccore, which includes expanded support for applications requiring independent (asynchronous) or common (synchronous) read/write clock domains. See [FIFO Generator](#) for detailed information.

Features

- Drop-in module for Virtex™, Virtex-E, Virtex-II™, Virtex-II Pro™, Virtex-4™, Spartan-II™, Spartan-III, and Spartan-3™ FPGAs
- Supports data widths up to 256 bits
- Supports memory depths of up to 65,535 locations
- Memory may be implemented in either SelectRAM+ or Distributed RAM
- Fully synchronous and independent clock domains for the read and write ports
- Supports FULL and EMPTY status flags
- Optional ALMOST_FULL and ALMOST_EMPTY status flags
- Invalid read or write requests are rejected without affecting the FIFO state
- Four optional handshake signals (WR_ACK, WR_ERR, RD_ACK, RD_ERR) provide feedback (acknowledgment or rejection) in response to write and read requests in the prior clock cycle
- Optional count vector(s) provide visibility into number of data words currently in the FIFO, synchronized to either clock domain

- Uses relationally placed macro (RPM) mapping and placement technology for maximum and predictable performance
- Incorporates Xilinx Smart-IP™ technology for utmost parameterization and optimum implementation
- To be used with v6.3i or later of the Xilinx CORE Generator™ system

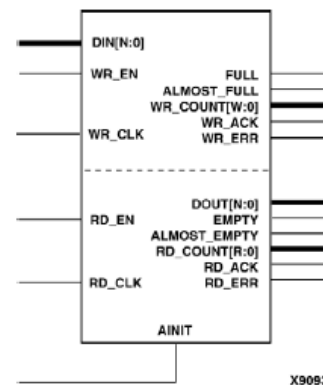


Figure 1: Core Schematic Symbol

© 2004 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.
NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

DS232 November 11, 2004
Product Specification

www.xilinx.com

1

ANNEX 7 Xilinx Core Reed Solomon Encoder v5.0



Reed-Solomon Encoder v5.0

DS251 April 28, 2005

Product Specification

Features

- Available for all Virtex™, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-4, Spartan™-II, Spartan-IIe, Spartan-3, and Spartan-3E FPGA family members
- Implements many different Reed-Solomon coding standards, including all ITU-T J.83 and CCSDS codes
- Automatically configured by user-entered parameters
- Efficiently handles multiple channels
- Fully synchronous design using a single clock
- Supports continuous output data with no gap between code blocks
- Symbol width from 3 to 12 bits
- Code block length variable up to 4095 symbols with up to 256 check symbols
- Block length can be changed in real time
- The number of check symbols can be changed in real time
- Supports shortened codes
- Supports any primitive field polynomial for a given symbol width
- User-configured generator polynomial
- Easy to use interface with handshaking signals
- Incorporates Xilinx Smart-IP™ technology for maximum performance
- To be used with Xilinx CORE Generator™ system v7.1i or later

Applications

Forward Error Correction (FEC) using Reed-Solomon algorithm. Used in systems where data are transmitted and subject to errors before reception. For example: communications systems, disk drives, etc.

Pinout

Port names for the core module are shown in Figure 1 and described in Table 1.

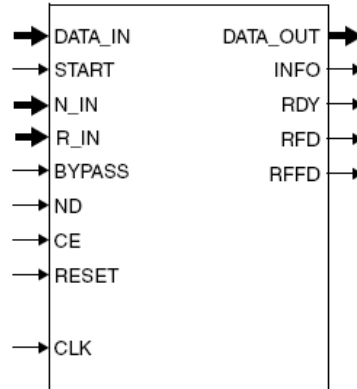


Figure 1: Core Schematic Symbol

Table 1: Core Signal Pinout

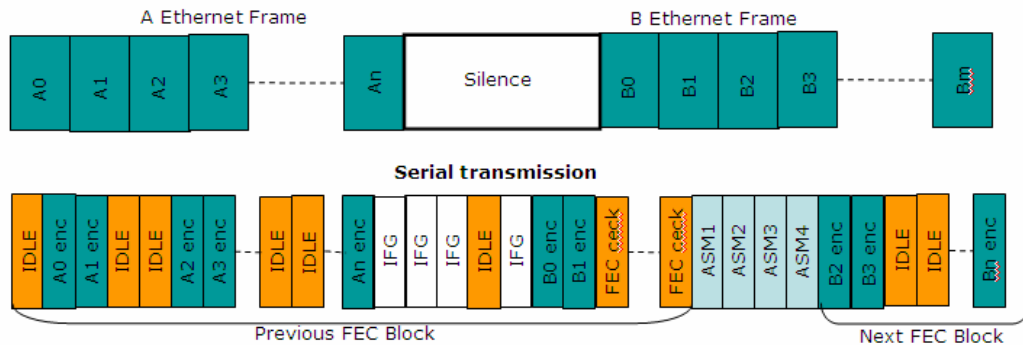
Signal	Direction	Description
DATA_IN	Input	Data Input: The data to be encoded
N_IN	Input	Block Size (N) Input: Used when block size is variable
R_IN	Input	Check Symbols (R) Input: Used when number of check symbols is variable
RESET	Input	Reset: Active high asynchronous clear
CE	Input	Clock Enable: Core is frozen when low
ND	Input	New Data: Input signals are sampled when high

© 2005 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners. Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

ANNEX 8 Special Words (IFG, IDLE ,ASM) and DC Balancing

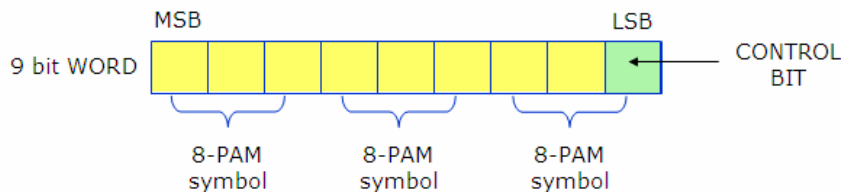
Ethernet interfacing: IFG, IDLE and ASM 19

- The IFG insertion unit adds control 9 bit words (**IFG**) to the frames during Ethernet silence phases. This helps to counteract the possible frequency mismatch between the sending 10/100BaseT device and the receiving one
- Since the chosen Baud Rate is greater than the required minimum one, control 9 bit words (**IDLE**) are inserted to ensure a continuous transmission at the serializer output
- For a correct synchronization of the FEC block, four control 9 bit words (**ASM**) are inserted in the serial data transmission.



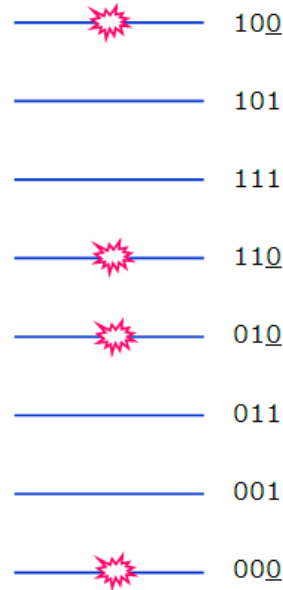
Ethernet interfacing: 8b/9b encoder and decoder 20

- The 8b/9b encoder goals are:
 - Avoid of any "systematic" creation of a DC component
 - The resulting code is "balanced" under the assumption of "random" data
 - Provide easily detectable control words (IDLE, IFG, ASM)
- The encoded "WORD" is made by 9 bits, and thus 3 8-PAM symbols
- The Least Significant bit is a special bit, which is:
 - "1" for control words
 - "0" for data



- The 8b/9b decoder goals:
 - Detect and delete control words to reconstruct the transmitted Ethernet frames

- The fixed "0" bit at the end of a DATA word does not give a "systematic" DC component



- In fact, after GRAY mapping, the four possible 8-PAM symbols carrying the fixed "0" bit are "balanced"

Ethernet interfacing: DC-balancing for Control words 22

- The control WORD have a "1" at the end
- Unfortunately, with 3 Gray mapped symbols it is impossible to select a combination that gives exactly zero average in terms of analog levels
- Since control words may appear in long consecutive sequences (for instance for IFG) we must avoid this "systematic unbalance"
- We need an alternating mechanism between a given word and its "opposite"

