International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 8 2377 – 2383

FPGA Implementation of Low Power Serial to High Speed Data Networks

Padmaneela Nallani
M. Tech, Embedded systems, BVRIT
E.C.E Dept, Hyderabad, India

Mr. T. Vasudeva Reddy
M. Tech (PhD), Associate Professor, BVRIT
E.C.E Dept, Hyderabad, India

***Abstract*—** FPGA based solutions become more common in embedded systems these days. These systems need to communicate with external world. Considering high-speed and popularity of Ethernet communication, developing a reliable real-time Ethernet component inside FPGA is of special value. To that end, we present a new solution for FPGA Gigabit Ethernet communications with timing analysis. The solution deals with "Gigabit Media-Independent Interface" in its physical layer. Network protocol is implemented from physical to transport layer which is UDP. In this Project using LAN connection various data will be captured by FPGA and will be sent on a serial line. Read the data from UART Receive and transform the data in it. On FPGA logic is implemented to read data from serial port. Write data in to memory location and transmitted data out put The FPGA module takes data from serial port and sends to PC in Ethernet form. In PC application will be developed to read data from Ethernet.

***Keywords*—** *Gigabit Ethernet; UDP IP Protocol; EMAC*

*****

1. INTRODUCTION

FPGA (Field Programmable Gate Array) based systems are playing an increasingly important role in embedded systems. Ever since FPGA has vastly used in embedded systems, communication between FPGA and other parts of system was turned to be an important necessity. Depending on amounts of data which should be transferred, different types of connections can be used. Ethernet communication provides enough bandwidth for most of the high demanded applications.

The Gigabit Ethernet technology is an extension of the 10/100-Mbps Ethernet standard. Gigabit Ethernet provides a raw data bandwidth of 1000 Mbps. Gigabit Ethernet includes both full and half-duplex operating modes. A Gigabit Ethernet is imperative for two reasons: faster systems and faster backbones. With the development of Ethernet systems and the growing capacity of modern silicon technology, embedded communication networks are playing an increasingly important role in embedded and safety critical systems. Advances in VLSI technology have also pushed integration to the point where it is now possible to design and implement a microprocessor and network controller on a single chip, known as System-on-Chip (SoC). In a network of embedded systems, each system can communicate with the other systems in the network, sharing information and sending and responding to requests as needed. Embedded devices need to be designed to solve specific problems. It is a challenge to find the right balance between power and cost. This becomes even more complicated when adding network capability to a device. The advent of Field Programmable Gate Arrays (FPGAs) with thousands of logic gates has made it possible to verify specific software functions on specific hardware. This reduces the design cycle and hence the execution cycles time to make the embedded system respond faster in real-time. A reconfigurable NIC (Network Interface Card) allows rapid prototyping of new system architectures for network interfaces. The architectures can be verified in real environment, and potential implementation bottlenecks can be identified. Dynamically reconfigurable platform will also reduce power consumption of the network device. FPGA based platforms fulfill the performance requirements and provide extra flexibility in comparison to ASIC implementations. Using an FPGA based custom design PCI platform, we have incorporated a 10/100/1000Mbps MAC design to build a low cost, high performance embedded network controller. Figure 1 gives an overview of the IEEE 802.3 Ethernet Standards

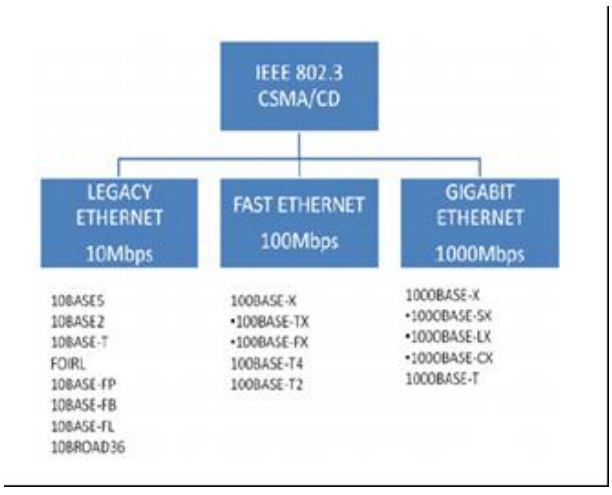

Figure 1.IEEE 802.3 Ethernet Standards overview

2. ETHERNET COMMUNICATION

This section covers OSI (Open System Interconnection) and Protocols of the layers which are discussed in this paper.

2.1. OSI MODEL

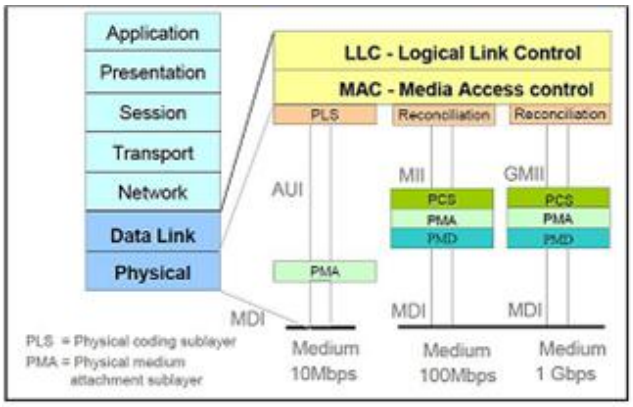

Figure 2: The OSI and generic Ethernet Physical Layer model

In order to describe the behavior of a network, the OSI model is used that has seven layers which are Application, Presentation, Session, Transport, Network, Data link and Physical layer. Figure.2 shows the O.S.I. model however, minor specifications may vary from version to version.

2.2. PHYSICAL LAYER

The Physical layer indicates how signals can be transmitted on a network, gives interfaces for a network and defines the different types of physical aspects. Relation between protocols which are implemented in the FPGA and also protocol hierarchy is shown in figure 3.

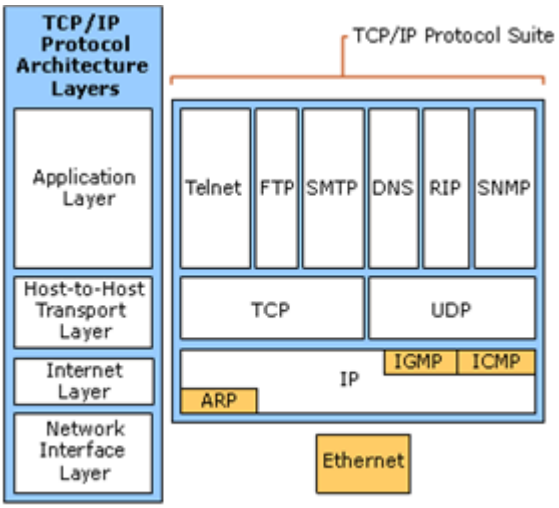

Figure 3: Top protocols and networks in the TCP/IP model

In order to have an Ethernet connection physical layer uses PHY device. For connecting PHY device to the FPGA, GMII interface is used.

2.2.1. GMII Characteristics

The GMII is intended to be an alternative to the IEEE802.3u MII and RMII. The principle objective is to increase speed of communication. In Gigabit operation, the clocks will operate at 125MHz, and for 10/100 operation, the clocks will operate at 2.5MHz or 25MHz respectively

The Gigabit Media Independent Interface [GMII] is the interface between the Media Access Controller [MAC] layer and Physical Layer and is divided into three sub layers. Those are PCS, PMA and PMD.

i **Physical Coding Sub layer [PCS]** is the GMII sub-layer responsible for the interface to the Reconciliation layer. The PCS layer uses 8B/10B encoding and performs auto negotiation.

ii **Physical Medium Attachment [PMA]** is the GMII sub-layer responsible for providing a medium-independent for the PCS to support serial bit-oriented physical media. This layer serializes code groups for transmission and deserilizes bits received from the medium into code groups Encoding is 8B/10B. It performs PMA framing, octet synchronization/detection, and x^7+x^6+1 scrambling / descrambling

iii **Physical Medium Dependent [PMD]** is the GMII sub-layer responsible for mapping the physical medium to the PCS. The Medium Dependent Interface [MDI] is the physical layer interface, and is part of the PMD

2.2.2. GMII Reception TIMING

In order to receive packets with GMII Interface, there are some points which should be taken into consideration. According to figure.4 RX_DV signal is High means that following bits are valid data, but after that preamble can start after undefined number of clock cycles. This problem is solved using one state machine which after CRS signal waits for 7 bytes of preamble followed by one byte SFD. While operating in full duplex mode, the behavior of CRS is not specified by standard, hence it is ignored by the MAC.

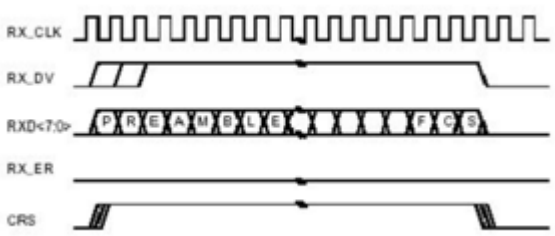

Figure 4: GMII Reception timing for packet with no error

2.2.3. GMII Transmission Timing

In transmission an important point is to change TXEN on rising edge and TXD on falling edge which is clear in figure5

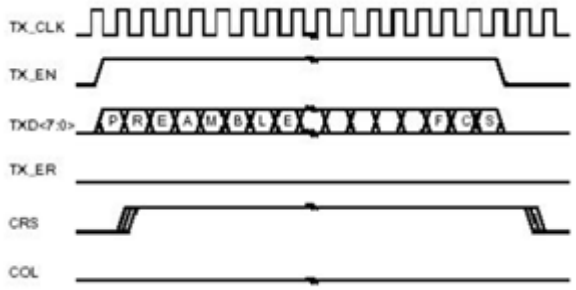

Figure 5: GMII Transmission timing for packet with no error

2.3. Data link layer (MAC)

One of two sub layers of the Data Link layer is the Media Access Control Layer (MAC). The MAC sub layer uses Carrier Sense Multiple Access protocol which has Collision Detection ability (CSMA/CD) to ensure sent signals from different stations over the same channel do not collide. Considering that IEEE802.3 which is one of the world's most used protocols is of the CSMA/CD type, importance of CSMA/CD class becomes more obvious. The MAC layer is responsible for delivering data packets over a shared channel. All MAC fields should be sent by sequence and their sizes are fixed except data field which could vary from 46 to 1500 bytes.

Figure 6 shows all MAC fields. The important point in the implementation is that bits are sent from LSB (Least Significant Bit) to MSB (Most Significant bit). If the size of data is less than minimum size, then some bytes of zero should be added to this field to reach to 46 byte.

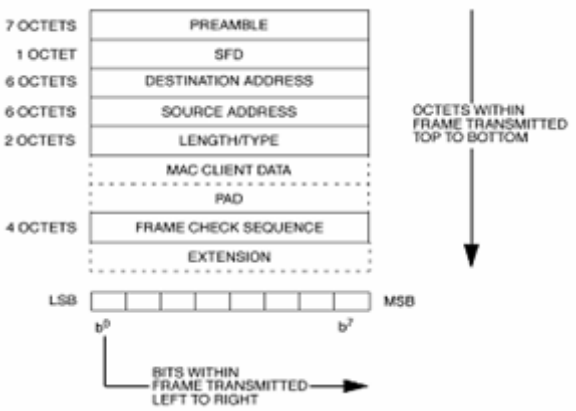

Figure 6: MAC Packet format.

MAC protocol has been used in order to have raw packet communication and to verify solution then it has been improved to UDP communication.

2.3.1. CRC:

CRC (Cyclic Redundancy Check) is a popular and reliable technique of error detection in data communication systems. This can detect a large number of errors. This method is based on polynomial arithmetic. CRC is used in order to distinguish damaged frames from correct ones. CRC-32 is one of common CRC standards used for Ethernet which defines polynomial function G(x) as a common generator polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0.$$

X represents bit value and superscript of x shows the position of bit in a bit stream. CRC should be calculated by both MAC sender and receiver and in a case that they do not have equal values receiver should discard the packet.

2.4. Network Layer

IP (Internet Protocol) is an important Protocol at this layer because it introduces a way to deliver messages from source to destination. Both source and destination use one fixed address. This protocol should be called by host-to-host protocols. For example in this project UDP module calls this protocol to take UDP packet as data and transmit datagram.

2.5. Transport Layer

Transport Layer has two protocols; the Transmission Control Protocol (TCP), which provides a communication with reliable data delivery, and the User Datagram Protocol (UDP) which gives an unreliable communication. UDP is such a simple transport protocol that allows applications to send datagram and handle translation between ports and sockets. UDP has twofold ports: source and destination; Also its segments consist of 8-byte header which is shown in figure 7

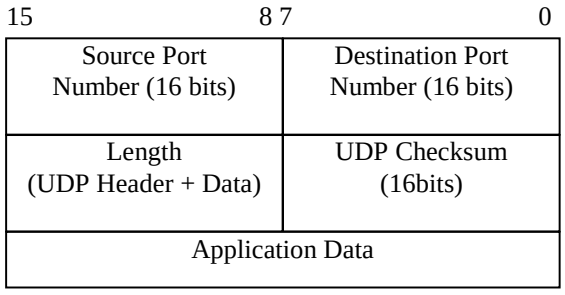

Figure 7: The UDP header

The interface of sending the packets by IP is UDP which

does not give delivery guarantees due to the lack of error handling. Moreover, in some applications which TCP is not suited, UDP can be used. For instance, in applications which low latency seems more significant compared with reliable data delivery, UDP is a better choice.

2.6. Application layer

The application layer is the OSI layer closest to the end user, which means both the OSI application layer and the user interact directly with the software application. Some examples of application layer are implementations also include On OSI stack: FTAM File Transfer and Access Management Protocol, X.400 Mail and Common Management Information Protocol (CMIP). On TCP/IP stack: Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Simple Network Management Protocol (SNMP)

3. STRUCTURE OF CORE

Objective of this section is to provide detailed information about source code of presented 1000MB Ethernet communication. A component should be feasible to use in different projects and different solutions (Re-usability). Developing components by using different developers which speed up process would be easy when parts are independent from each other. For example in presented architecture UDP communication there are two different components which are UDP sender and UDP receiver. Therefore in a project that just sending operation is needed, it is completely possible to only use UDP sender component individually. Although in our case both components are used, In implementation of solution, VERILOG HDL programming language is used.

3.1. TOP COMPONENT

In the FPGA applications there is a top layer which is composed of other components and plays the role of a container and organizer for other components. In presented design, the top component is responsible to control all other components, this component provides all necessary inputs to other components and also handles output signals of them. Figure 8 shows all internal components in top layer. The figure indicates that identical components are used in different places (Re-usability). Clock Maker component is a prime example of this characteristic which is used in four different places. Clock Maker and other components are explained in detail separately.

3.2. SIMPLE_GEMAC_WRAPPER

Initiating Ethernet PHY is the first step in establishing communication. Based on desired communication type, inner PHY registers should be manipulated. **Simple_gemac_wrapper** component is responsible for initiating Ethernet PHY. It sends I2C (Inter-Integrated Circuit) commands to PHY and prepares it for specific speed and duplex mode. GMII is chosen using these I2C commands. I2C uses two lines (data and clock) for manipulating registers that are inside PHY.

Figure 8 shows interior components of simple_gemac_wrapper.clk_125_tx produces proper clock for management process. I2C Component needs command, register address, clock and load input. This component writes input command and input register address on MDIO line synchronized with the clock.

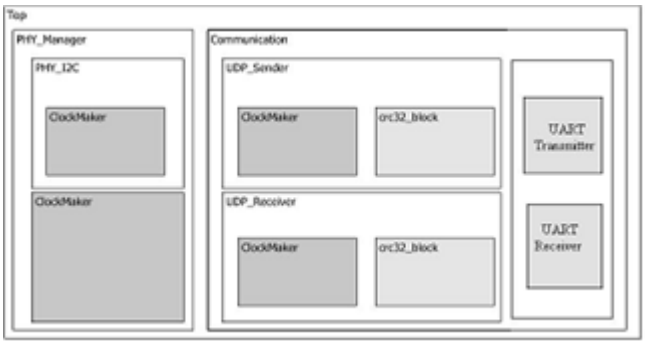

Figure 8: Top component and hierarchy of other components that are inside it.

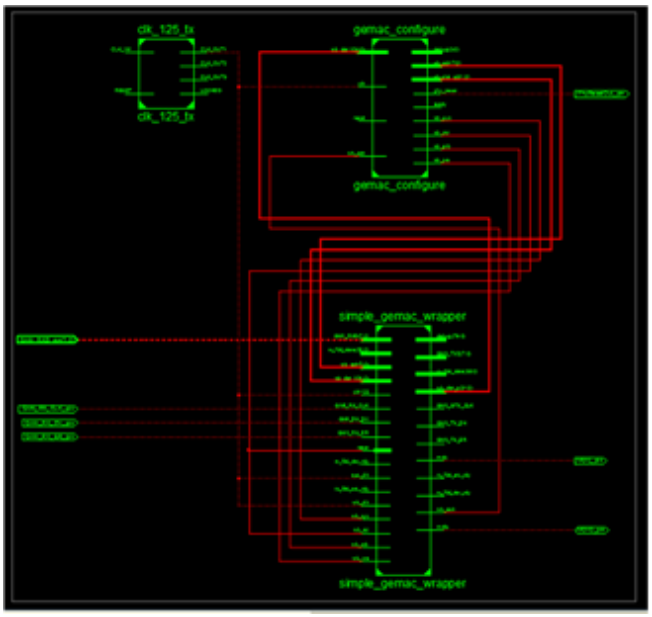

Figure 9: simple_gemac_wrapper Component consists of I2C and Clock Maker component.

When wb_ack input is high, it means that command is read and after turn around, I2C should release the line and let PHY write the response, then I2C reads the response. After input values become ready, load input should be high for at least one cycle of clock and then it should go low to start making commands and writing on MDIO line. Output MDC and MDIO of I2C component are directly connected to the top layer. MDIO can be input when command is read, so it should be declared as an inOut signal.

3.3. COMMUNICATION

Communication component takes care of all actions related to Ethernet communication. Top layer can ask this component to send UDP Packets which are vary in size. In order to keep design as simple as possible, packets do not have dynamic size and they can be in only two types: small

and large packets. Top layer provides communication component with data and asks it to send the data. Sending entire data to communication component is not necessary. Top component just have to send data faster than the speed of UDP sender component. Moreover, this component delivers received data to the top layer. The output of component could be connected to a memory or other components. This layer of system for handling Ethernet communication uses packet_sender and packet_receiver components which are shown in figure 10 TXD and TXEN output of packet_sender are directly connected to output of communication layer as well as RXD and RX_DV inputs which are directly connected to packet_receiver component.

Figure 10: Ethernet communication Component consists packet sender and Receiver.

3.3.1. PACKET_SENDER

This component uses three synchronized processes to make a packet which is sent out by using eight lines of data. One process is responsible for TXEN output, the other sends bits out and the last one is a state machine. Different layers of network protocols are built using this state machine. The state machine which generates appropriate MAC preamble after sending preamble, state changes to MAC destination address. It sends all header data such as MAC header, IP header and UDP header and in the need of sending data field sends out a request and the communication layer provides a byte of data for each request. In the meantime packet_sender makes CRC and attaches four bytes of CRC to the end of the packet.

3.3.2. PACKET_RECEIVER

The component is able to receive UDP packets. First process watches CRS input for catching packets and the other one is a state machine which receives all fields in different layers. Whenever RX_DV is high, it starts monitoring input data and if it is in correct format (all header fields such as preamble, SFD, MAC header and UDP header are correct), it receives packet and sends out data field of packet to communication layer. In-dependent from size, data field is sent to upper layer byte by byte. Communication layer takes care of this data and could save it in memory or start processing and analyzing data. While different fields of packet are getting received in state machine, CRC is getting generated. Therefore when CRC field is getting received, it can be compared with calculated CRC. So with no delay, packets are validated. Finally when packet is received, a bit is send to upper layer which indicates whether CRC field is correct or not.

3.4. UART BLOCK

The UART is a serial interface with a frame format of start bit of active low ‘0’at beginning of frame and 8 bit of information with a stop bit of active high‘1’ signal at the end. The operation of UART is controlled by Clock signal which is fed from external crystal.

3.4.1. UART FUNCTIONS:

In addition to the basic job of converting data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media, and to regulate the flow of data in the event that the remote device is not prepared to accept more data.

3.4.2. UART RECEIVER

This module is mainly responsible for data reception and the Conversion of data. It mainly contains data registers and receivers. When the valid stop bit is detected, data are sent into the register. When the size equals to the setting value, data in the register are transferred into the I/O buffer. Receive controller mainly includes a counter and a 8 bits shift-register. It is controlled by the state machines.

3.4.3. UART TRANSMITTER

This module is used to send bus data received from Ethernet. It consists of transmitter and data registers. When the command is executed, transfer will send data in serial form until the send counter is 0, which means over. At last, the flag bit is set. The transmitter is a 8-bit shift register. As long as data register is not empty, shift register would constantly read data, add start and stop bits and send them in asynchronous frame format. It is also controlled by state machines. When the send clock is high, if the data register is full, the data will sent in order. The coordinated communication of multiple serial ports: in order to allow ports communicate effectively, there is a need to administer them.

As we know, the FPGA processing speed is far greater than peripheral transmit data rate, so once I/O buffer receives data from any serial port, the data can be read by Ethernet interface chip directly. So long as the serial data start address and end address doesn’t conflict. When the send clock is high, if the data register is full, the data will sent in order. As we know, the FPGA processing speed is far greater than peripheral transmit data rate, so once I/O buffer

receives data from serial port, the data can be read by Ethernet interface chip directly. So long as the serial data UART receiver handles reception of data from RS232 port. Main functions of receiver block are to convert the serial data to parallel data, and check the correctness of data from parity and store the received data. UART receiver state machine. The receiver is in IDLE state by default. When the serial data pin goes low, indicating the start bit, the state machine enters DATA0 state. The data is received, one bit at a time from LSB to MSB in states DATA0 to DATA7.

If parity is enabled, the state machine checks the parity bit received against the parity obtained from received data. If the data received is fine, the (RxD_data_ready) bit is set to '1' and the receiver goes back to IDLE state again.

4. EXPERIMENT CONSTRAINTS

The solution is synthesized and transferred into a FPGA by Xilinx ISE which is from Xilinx Company. A custom board is used for testing purpose. Specifications of the board components are described.

The FPGA which is used in this project is Xilinx SPARTAN 6 XC6SLX45 CSG324C.

Difference between Xilinx families:

	Spartan-3, Virtex-4	Virtex-5, Virtex-6, Spartan-6
No. of slices per 1 CLB slice	4	2
No. of LUTs per each slice	2	4
No. of LUT inputs	4	6
Max. single-port memory size per LUT	16x1	64x1
Max. shift register size per LUT	16 bits	32 bits

Figure 11: Comparison of different Xilinx Families

Between 25 and 50% of all slices can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers (SRL32) or as two SRL16s. From table 2, we can summarize that there are only 2 slices per CLB slice in Spartan-6 each contains 4 6-inputs LUTs while 4 slices per CLB slice are exist in Spartan-3 families each contains 2 4-inputs LUTs and from table 5 the results show that Spartan-6 requires a much smaller number of LUTs for the design than Spartan-3A which lead to the use of less silicon area in Spartan-6, built upon 45 nm triple-oxide technology while Spartan-3 is built upon 90 nm triple-oxide technology. Spartan-6 consumes less power than Spartan-3A, Virtex-4, and Virtex-5 and is higher in performance.

The PHY, which is used in the board, is Alaska 88e1111 PHY from Marvell Company. By using this PHY 1000BASE-T 1000BASE-X, 100BASE-TX and 10BASE-T communication is possible which fulfills project requirements. Using Gigabit Media Independent Interface (GMII), this device can connect to MAC layer.

5. FLOW CHARTS

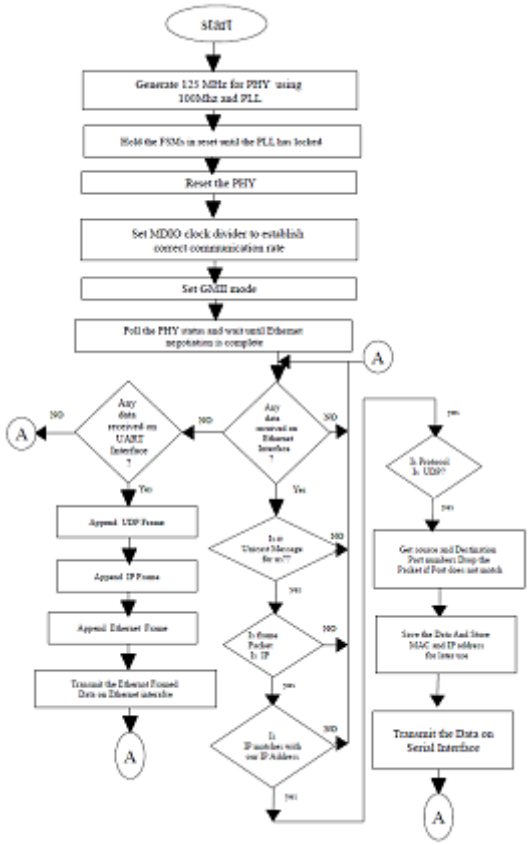

Figure 12: Flow chart of complete system.

6. TEST AND DEBUG

In order to test the presented solution, communication between FPGA and computer is tested by using a windows application.

A convenient way for testing solution is connecting FPGA to a PC and trying to communicate with FPGA using windows based application. In order to be able to send and receive packets with any IP addresses, manipulation in network adapter of windows is necessary. Dock light application which can send and receive custom packets is used. Wire shark can also be used to analyze the Packets

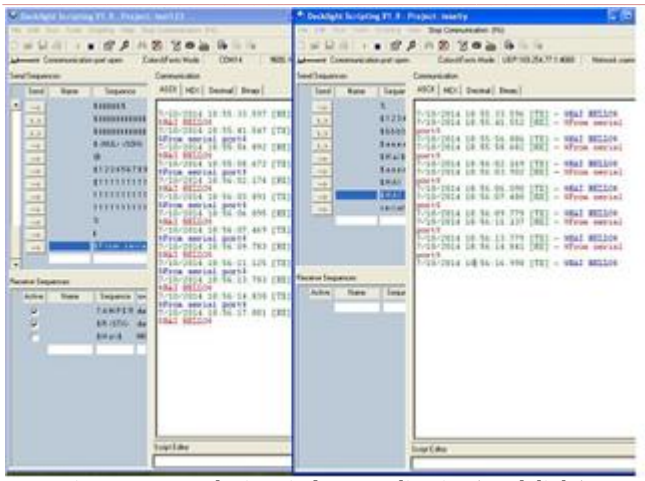

Figure 13: Results in Windows application(Dock light).

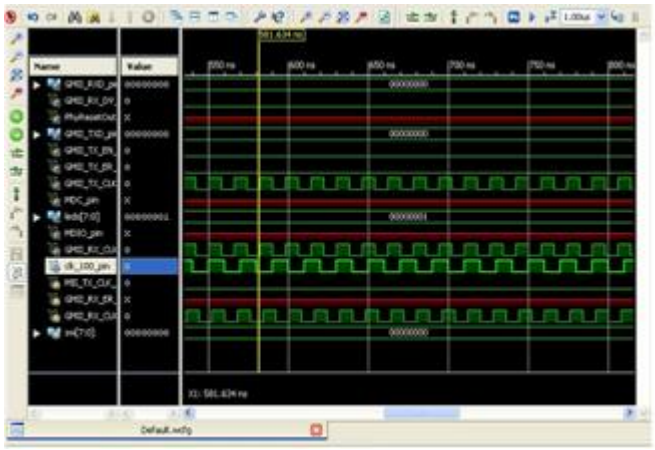

Figure 14: Simulated Results

7. SUMMARY AND CONCLUSIONS

In this work, we introduce a new solution for Ethernet communication in FPGAs. The solution uses GMII interface in physical layer. The communication between monitor computer and port devices, improve the efficiency of CPU, and ensure the processing of system in real time. FPGA's flexible programming features also allow further upgrade for system. Low cost, as the multicard solution can come in single FPGA card with small modules around it communicating local area no of systems transferring the data on system to anther system Low power in comparison with multicard solution using Xilinx ISE 12.4 , Lot of scope for adding additional functionality on FPGA

8. REFERENCES

[1] Davicom datasheet 10/100 mbps fast Ethernet physical layer tx/fx single chip transceiver. pages 1 { 41,September 2008.

[2] A. L¨ofgren, L. Lodesten, S. Sj ¨oholm, and H. Hansson,"An analysis of FPGA-based UDP/IP stack parallelism forv embedded Ethernet connectivity," in *Proceedings of the 23rd IEEE NORCHIP Conference*, November 2008, pp. 94–97.

[3] Ethernet h10032.www1.hp.com/ctg/Manual/bpe50027.pdf

[4] Shouqian Yu, Lili Yi, Weihai Chen, Zhaojin Wen,"Implementation of a Multi-channel UART Controller Based on FIFO Technique and FPGA," industrial Electronics and Applications, Harbin, China, May2007, pp. 2633- 2638. |

[5] N. Alachiotis, S. A. Berger, and A. Stamatakis, "Efficient PC-FPGA Communication over Gigabit Ethernet," in *CIT*, 2010,pp. 1727–1734.

[6] "User datagram protocol," RFC 768 (Standard), Internet Engineering Task Force, August 1980.

[7] Tinoosh Mohsenin , (2004) Rice University, "Design and Evaluation of FPGA-Based Gigabit-Ethernet/PCI Network Interface Card Thesis"

[8] Yafang Wang, Cheng Zhang, Yanli Hou,Boning Hu, (2010) "Implementation of Gigabit Ethernet Network based on SOPC", Asia Pacific Conference on Wearable Computing Systems, pp.341-343.

[9] hearlink.tripod.com/CandCDB/GMII_REPORT.pdf

[10] C. Kachris, et al., (2008) "Design and performance evaluation of an adaptive FPGA for network applications", Microelectron. J (2008) , doi: 10.1016/ j.mejo. 2008.05.011 .