FPGA implementation of Real-time Ethernet communication using RMII Interface

Nima Moghaddami Khalilzad
nmi09001@student.mdh.se

Sheida Pourshakour
spr07001@student.mdh.se

School of Innovation, Design and Engineering
Malardalen University
Vasteras, Sweden

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 100 Mb/s FPGA-based Ethernet communications with timing analysis. The solution deals with "Reduced Media-Independent Interface" in its physical layer. Network protocol is implemented from physical to transport layer which is UDP. For getting used in real-time applications, timing analysis is done. Component based software engineering is used in the design and development process. In order to test the components inside FPGA two different approaches are utilized. Signal measurement in combination with introduced windows based application contributes much in testing and validation phase.

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. This paper is firs version of a complete version which will be published later.

In this paper, in order to solve the communication problem a UDP/IP core on 100Mb Ethernet is introduced. The solution uses advantages of CBSE (Component-based software engineering) in design and development phases. Using some testing techniques, validity of the solution is evaluated after implementation phase. Significant of solution is using reduced media-independent interface (RMII) in the physical layer of communication protocol. The presented solution in this paper is based on works that has been done for vision systems of underwater and football player robots in Malardalen University. These robots are utilizing FPGA-based stereo vision [10] which uses Ethernet to communicate with other parts of robot. Source files are available as open-source code on the web.

In the light of characteristics of FPGA applications, it is easy to calculate execution time for programs. Despite other common programs which OS (Operating System) runs them, the program in a FPGA executed as a hardware component independent from OS. Therefore presented solution can be used in real-time applications as a reliable communication component.

Some commercial IP (Intellectual Property) cores exist and can be used for Ethernet communication in FPGAs. Open-source IP cores are available for different communication speeds. Implementation of UDP/IP stack and performance measurements [3] and TCP/IP core [6] are presented in some previous works. Some designs and implementation on gigabyte Ethernet are done[12][15]. Because of our hardware limitation, we needed to interface RMII physical layer. Due to complexity, dependability and lack of documentation, making changes to open source IP cores is not easy at all. Thus, new implementation which takes advantages of previous works and software engineering principles is presented.

In this paper, section 2 provides basic theoretical information which is absolutely necessary for implementation of 100Mb Ethernet communication. Then section 3 describes the solution and the way it is implemented. Section 4 shows what hardware and software tools are used in the implementation and validation of solution. In section 5 validation of solution is discussed. Finally, conclusion is presented in section 6.

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

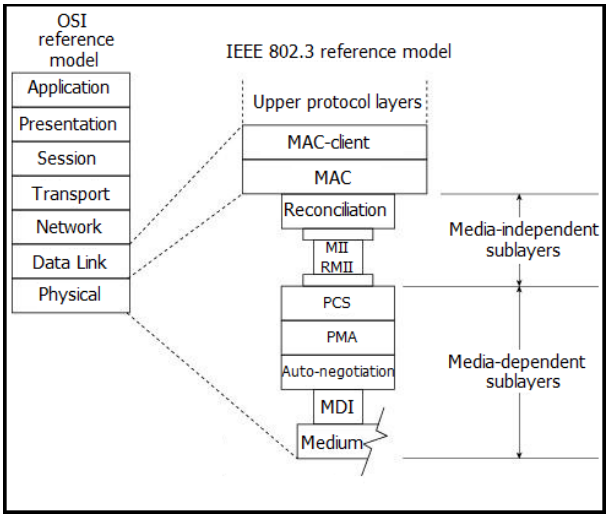

Figure 1: The OSI and generic Ethernet Physical Layer model[7].

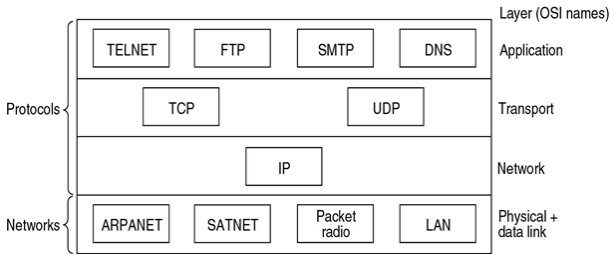

Figure 2: Top protocols and networks in the TCP/IP model initially, redrawn from[4].

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 1 shows the O.S.I. model however, minor specifics may vary from version to version. For more detailed description of the layer and protocols, see [7].

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 2.

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

2.2.1 RMII Characteristics

Table 1: Pin Maps of Normal MII and Reduced MII[1]

Normal MII Mode	Reduced MII Mode
TXD [0:1]	TXD [0:1]
TXD [2:3]	NC
TXEN	TXEN
TXER/TXD [4]	NC
TXCLK	NC
RXD [0:1]	RXD [0:1]
RXD[2:3]	NC
RXEN	VCC
RXER/RXD[4]/RPTR/NODE	RPTR/NODE
RXDV	CRS DV
RXCLK	NC
COL	NC
CRS	NC
MDC	MDC
MDIO	MDIO
RESET	RESET
XT1 (25 MHz)	XT1 (Floating)
XT2 (25 MHz)	XT2 (REF CLK 50MHz)

RMII is one of the standard interfaces between MAC(Media Access Controller) and PHY (Physical Interface) which can support 100 Mbit/s Ethernet.

In order to run in RMII mode XT2 pin - which is reference clock - on PHY should be connected to 50MHz clock and in our board it is directly connected to 50MHz Oscillator so no need to control it from FPGA. In addition there are only two pins for transmitter and two pins for receiver instead of four in regular mode. It means that TXD[2:3] and RXD[2:3] should be disconnected. TXER (transmit coding error) is not getting used in RMII mode as well as COL pin which is for detecting Collision. One of the major differences between MII and RMII mode is that in RMII TXCLK and RXCLK pins are not used. All Transmission and receiving operations is synchronized using one reference clock which is 50MHz. Moreover TXEN pin is used to show when there is valid data on TXD. In addition, RXDV pin acts like CRS in RMII mode which indicates when there is valid data on RXD pins[1]. Table 2.2.1 shows how pins are mapped between MII and RMII.

2.2.2 RMII Reception Timing

In order to receive packets with RMII Interface, there are some points which should be taken into consideration. According to figure 3 CRS DV signal 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 DV signal waits for 7 bytes of preamble followed by one byte SFD.

2.2.3 RMII Transmission Timing

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

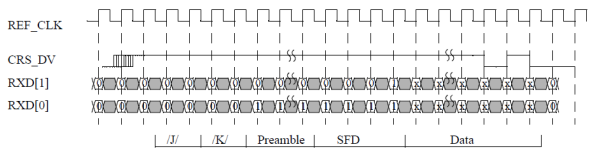

Figure 3: RMII Reception Timing for packets with no errors [13].

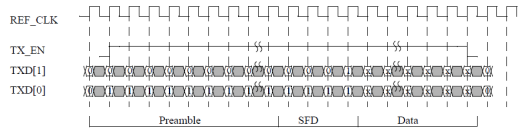

Figure 4: RMII Transmition Timing [13].

4.

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[5]. Figure 5 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)[8]. 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. CRC should be calculated from all bytes of data field which is being calculated in the implementation during sending time and it would be ready after transition of last byte. Despite other octets CRC field should be transmitted from MSB to LSB and after that should be 12 octets inter-frame gap before sending next packet. In CRC section, calculating CRC is explained. 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 which 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[5]:

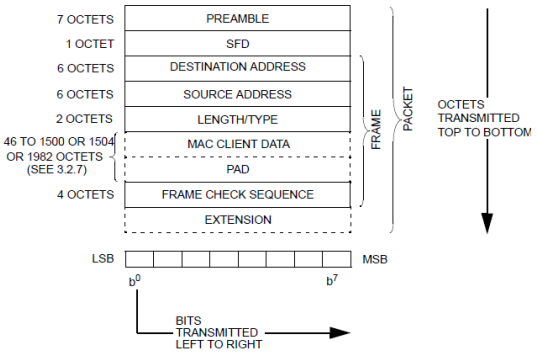

Figure 5: MAC Packet format [8].

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

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.

2.3.2 Parallel Implementation of CRC-32

Two implementations of CRC-32 are available ; serial and parallel. The first one is not optimal because connection between MAC layer and the large number of IEEE 802.3 communications are established using a data bus more than four bits. Parallel implementation is recommended in order to save the time in the way that in each clock cycle it operates on a number of bits of data. In this implementation an algorithm is used which can generate the next state equations for CRC-32 registers. The Perl script uses this algorithm to generate the ready CRC component which is used in this solution[5].

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 6 [4].

The interface of sending the packets by IP is UDP which does not give delivery guarantees due to the lack of error

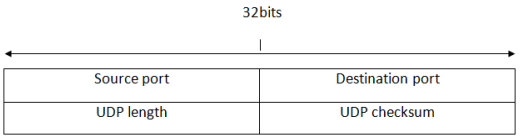

Figure 6: The UDP header [4]

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[4].

3. STRUCTURE OF CORE

Objective of this section is to provide detailed information about source code of presented 100MB Ethernet communication. CBSE (Component-based software engineering) approach is used in this project because variety of advantages are associated with that. A component should be feasible to use in different projects and different solutions (Reusability). Developing components by using different developers which speed up process would be easy when parts are independent from each other[9]. For example for 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 order to follow standards of CBSE they are developed discretely. In implementation of solution, VHDL programming language is mostly used. Except CRC generator component that is written in VERILOG, other parts of system are in VHDL.

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 7 shows all internal components in top layer. The figure indicates that identical components are used in different places (Reusability). ClockMaker component is a prime example of this characteristic which is used in four different places. ClockMaker and other components are explained in detail separately.

3.2 PHY Manager

Initiating Ethernet PHY is the first step in establishing communication. Based on desired communication type, inner PHY registers should be manipulated. PHYManager 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. MII or RMII is chosen using these I2C commands. I2C uses two lines (data and clock) for manipulating registers that are inside PHY.

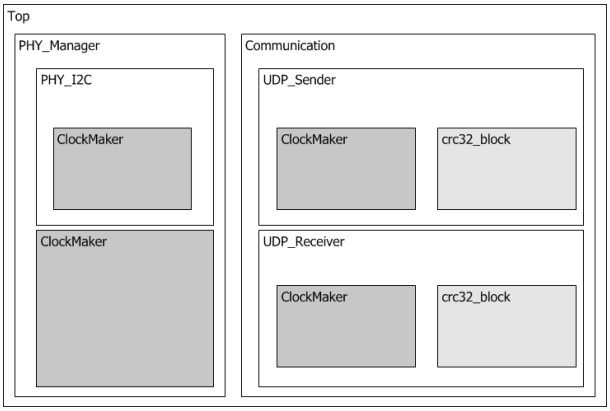

Figure 7: Top component and hierarchy of other components that are inside it.

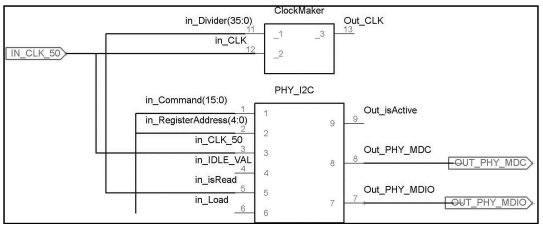

Figure 8: PHYManager Component consists of I2C and Clock Maker component.

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

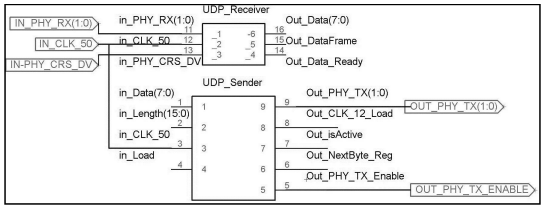

Figure 9: UDPSender and UDPreceiver components inside communication layer

```
WHEN    MAC_Preamble_3  =>
    Current_Byte
    <= Value_Preamble(Counter_Mac_Preamble downto Counter_Mac_Preamble - 7);
    IF Counter_Mac_Preamble > 8 THEN
        Counter_Mac_Preamble    <= Counter_Mac_Preamble - 8;
    ELSE
        UDP_NextState           <= MAC_Des_4;
        CRCGenerator_In_Enabled <= '1';
        CRCGen_In_NewFram       <= '1';
    END IF;
```

Figure 10: Preamble state in UDPSender state machine

UDPSender and UDPReceiver components which are shown in figure 9. TXD and TXEN output of UDPSender are directly connected to output of communication layer as well as RXD and CRS DV inputs which are directly connected to UDPReceiver component.

3.4 UDPSender

This component uses three synchronized processes to make a packet which is sent out by using two 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. Figure 10 shows preamble state that is inside the state machine which generates appropriate MAC preamble. As it is clear in the figure after sending preamble, state changes to MAC destination address. This component starts working after a pulse load input is high. 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 UDPSender makes CRC and attaches four bytes of CRC to the end of the packet. While UDPSender sends the packet, the output "Is Active" is high to show other components that sender is not ready to receive new order for sending packet.

3.5 UDPReceiver

Getting help of two processes, 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 CRS 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. Independent 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.

3.6 Timing Analysis

In real-time applications timing, analysis plays a vital role. Since programs inside FPGA run like a hardware component and there is no operating system, calculation of execution time is much easier than other types of programs that are executed in operating system. Here execution time for sending a UDP packet by using presented solution is calculated. Sending package operation starts in the first state of state machine and ends in the last state of that. Therefore if the number of clock cycles between start and end of state machine is calculated, the execution time of sending operation can be calculated by using the following simple equation:

$$C = \sum_{i=1}^{15} C_i$$

$$\text{where } C_i = K_i * \frac{1}{12.5*10^6}$$

C represents total execution time, C_i is execution time and K_i is number of clock cycles for state i. Since there are 15 states in state machine, the summation is from 1 to 15. Execution time for each state is a number of clock cycles multiplied by clock frequency of state machine process. Clock frequency for process is considered to be 12.5, consequently one single byte can be sent in each state transmission. Based on length of the data field execution time could differ. Since data field sending state is state number 13 ($i = 13$) and considering n as a number of bytes in data field, then:

$$K_{13} = n$$

$$\text{and } C_{13} = n * 80 * 10^{-9}$$

Other states need constant number of clock cycles for their execution, therefore they have constant execution time. For instance, in preamble state which is shown in figure 10 $i = 3$ and $k_3 = 8$. Hence execution time is:

$$C_3 = 8 * 80 * 10^{-9}$$

$$C_3 = 640ns$$

Basically, since presented design does not impose any extra delay to packet transmission, packet sending time is equal to minimum time needed in physical layer. Indeed, there is one cycle delay between sending request from Communication Component and start of sending in UDPSender Component which is absolutely unavoidable.

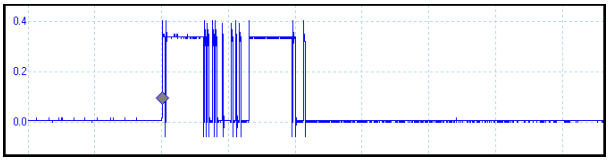

Figure 11: MDIO (Data line of I2C command) signal captured by Picoscope.

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 3A DSP (Digital signal processing). This FPGA, in addition to the features of SPARTAN 3A family, is using 90 nm process technology which is consecutive to more bandwidth. Moreover, in comparison to 3A family, this family of Xilinx FPGA's are using an additional block RAM. This RAM has some additional output registers which help it to perform faster [14].

The PHY, which is used in the board, is DM9161 from Davicom Company. By using this PHY ,100BASETX 100BASE-FX communication is possible which fulfills project requirements. Using Media Independent Interface (MII), this device can connect to MAC layer. In order to reduce the number of pins, it is also possible to use Reduced Media Independent Interface (RMII). In this project RMII was the option which has been set [1].

5. TEST AND DEBUG

In order to test the presented solution, two steps have been taken. First, output signals of FPGA are measured using Picoscop. Second, communication between FPGA and computer is tested by using a windows application.

5.1 Picoscop

In program development process for FPGAs, there are two important methods for debugging. First, a program should be simulated using simulation tools and output signals should be verified by requirements of program. This step is done using simulation tool which is embedded in Xilinx ISE tool. The second step is to make exactly the same signals inside FPGA. In order to make sure that program is executing in a way which is supposed, output signals are measured using Picoscope. Signals can be observed in a monitor because of the connection between Picoscop, which is an Oscilloscope, and the computer. Figure 11 shows captured output signal of FPGA which is one line of I2C command for initiating Ethernet PHY. This signal measurement is done for all single components individually. After integrating components, again output signals are observed and verified.

Successfully simulated programs might not work on device because synthesizer tries to optimize design and it can delete some connections which affect result. Hence a good knowledge of synthesizer is of importance in FPGA program development.

5.2 Windows Application

A convenient way for testing solution is connecting FPGA to a PC and trying to communicate with FPGA using a windows based application. In order to be able to send and receive packages with any MAC addresses, manipulation in network adapter of windows is necessary. A C#.Net application is developed which can send and receive custom packets. Using a graphical user interface, it is possible to assign value to all fields of different network layers. Basic idea of this application is from presented work at [11]. Figure 12 shows sending operation tab and indicates how packet field can be set and sent.

6. SUMMARY AND CONCLUSIONS

In this work, we introduce a new solution for Ethernet communication in FPGAs. The solution uses RMII interface in physical layer. Although RMII interface uses less pins for communicating with Ethernet PHY, it has no other advantages and when there is no hardware limitation, it may not worth it to develop a new component for interfacing RMII inside FPGA because there are already made IP cores for MII. For real-time applications FPGA is a reliable option since timing analysis can be done easily. The program is converted to hardware blocks which do not require operating system to be run and do not accept interrupts. Actually FPGA is as flexible as software and as reliable as hardware. Some problems such as lack of knowledge about synthesizer and hardware connection problems make it so difficult to develop components in FPGA. Picoscope helps very much in debugging. Using this devise, signals can be stored in the PC memory and can be analyzed by developers and also compared with simulation results. This method comes in handy when program works in simulation but not in real devise. In safety critical real-time application, using Real-time Transport Protocol (RTP) for communication is an excellent option. Since RTP is used in transferring video and image, it could be useful in vision systems which are developed in a FPGA. Presented solution provides developers with fertile ground in developing the RTP or other higher protocols.

7. REFERENCES

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

[2] Micrel KSZ8873MML datasheet 3-port 10/100 integrated switch with phy and frame buffer. page 1, May 2005.

[3] Andreas Lofgren, Lucas Lodesten, Stefan Sjoholm, Hans Hansson. An analysis of FPGA-based UDP/IP stack parallelism for embedded Ethernet connectivity. *NORCHIP Conference*, November 2005.

[4] Andrew S. Tanenbaum. Computer Networks. *Prentice Hall. 4:0-13-066102-3*, March 2003.

[5] Chris Borrelli. Ieee 802.3 cyclic redundancy check. pages 1 – 2, march 2001.

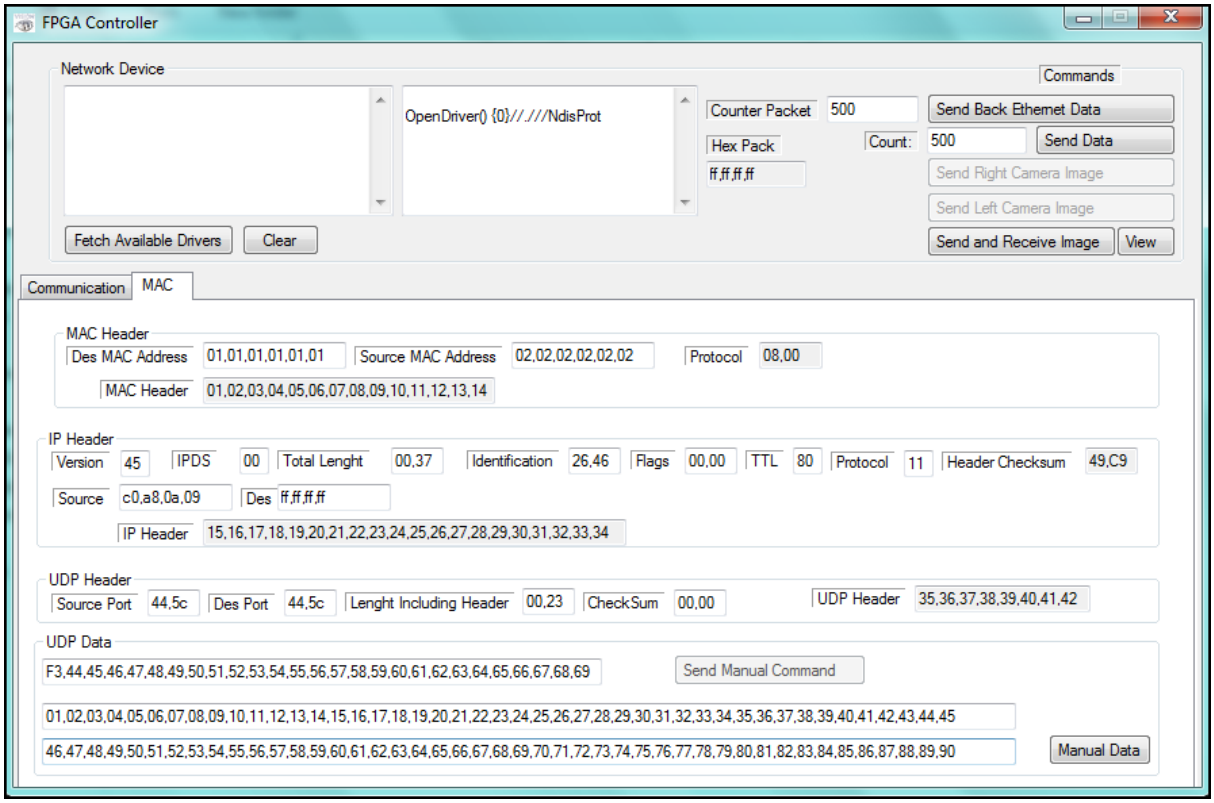

Figure 12: Windows application screen shot.

[6] Christophoros Kachris Design and Implementation of a TCP/IP core for reconfigurable logic. *Technical University of Crete Electronic and Computer Engineering Department*, July 2001.

[7] Cisco Systems. Internetworking technology handbook, 2010.

[8] IEEE. Ieee std 802.3-2008. page 49, 2008.

[9] Ivica Crnkovic. Component-based software engineering - new challenges in software development. Proceedings of the 25th International Conference on Information Technology Interfaces. June 2003.

[10] L. A. Jorgen Lidholm, Fredrik Ekstrand. Two camera system for robot applications; navigation. *13th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 1 – 2, march 2008.

[11] Miahrugger. Raw ethernet packet sending. The Code Project. October 2003.

[12] Nikolaos Alachiotis, Simon A. Berger, Alexandros Stamatakis. Efficient PC-FPGA Communication over Gigabit Ethernet. *10th IEEE International Conference on Computer and Information Technology*, 2010.

[13] RMII Consortium. Rmii specification. page 1, March 1998.

[14] Spartan-3a dsp fpga family: Xilinx datasheet. page 3, march 2009.

[15] Tomohisa Uchida, Member, IEEE. Hardware-based TCP processor for Gigabit Ethernet. *IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 55, NO. 3,*, JUNE 2008.