International Journal of Engineering Research and General Science Volume 3, Issue 6, November-December, 2015
ISSN 2091-2730

VLSI IMPLEMENTATION OF FULL DUPLEX, FSM BASED GIGABIT ETHERNET MAC

Aswin C[1], Aji Joy[2]
1.PG Scholar, Department of Electronics and communication Engineering, MACE. Kothamangalam, Kerala, India.
2. Asst.Professor, Department of Electronics and communication Engineering, MACE. Kothamangalam, Kerala, India.

Email: aswin.cms@gmail.com,+91-9495763459

Abstract— The communication network is analogous to the neural system of the human body for any embedded system. It carries data and control signals to various parts of the system. Hence, a high speed communication system is the need of the hour so that it does not become the bottleneck of the system's performance. Of all the available communication systems, Ethernet stands out as the most common network communication protocol. But due to advances in the technology, the primitive Ethernet could no longer match the processing speed of the new systems. Therefore, we need to improve the performance of the Ethernet. The evolution to Gigabit Ethernet improved the speed from few Megabits to one Gigabit and thus improved the performance of the whole embedded system. Increasing the bus width of the Ethernet is one way to increase the speed, but this comes with the overhead of encoders and decoders to be used and also increases the chance of causing interference. Another way is to use a clock with high speed, thus keeping the bus width low and avoiding the overheads and interference probabilities. This project selects a 125 MHz clock and 8 bit bus width for constructing a Gigabit Ethernet MAC on FPGA board. Finite state machine technique is used to construct the same. The project is coded using VHDL in Xilinx 14.1 and simulations are done in ISim simulator. The project was implemented on Spartan 3XC3S200 FPGA board.

Keywords—- Media Access Control(MAC), Finite State Machine(FSM), Start of Frame Delimiter(SFD), Frame Check Sequence(FCS), Cyclic Redundancy Check(CRC), Physical layer interface(PHY), Gigabit Media Independent Interface(GMII).

1. INTRODUCTION

The Ethernet is the most common communication network in the world constituting about 80 per cent of the total LAN networks. It started with the speed of the order of a few megabits per second. With the advent of fast Ethernet, this has increased up to 100 megabits per second. However, due to the various developments in the technology like in the case of processing speed, the 100 Mbps speed still fall short and bring down the performance of the whole system. So, we need to find ways to improve the speed of Ethernet network so that we need not replace it with another network, thereby saving the cost for the same. This is the reason for the research and development of even faster Ethernet like Gigabit Ethernet.

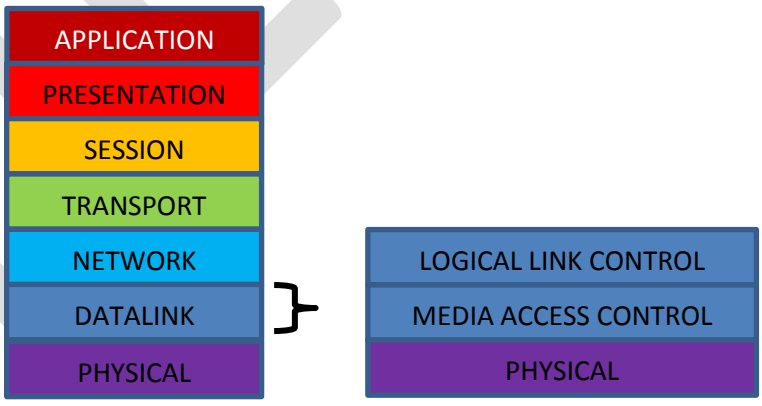

Fig 1: MAC layer represented in OSI model.

The media access control (MAC) is the heart of the Ethernet network. It is a sub-layer of the data-link layer of the OSI model. It acts as a bridge or interface between the lower layer, i.e. the PHY and the upper sub-layer of the data-link layer, i.e. the logical link control (LLC) layer. Figure 1 gives a representation of MAC in the OSI model. The MAC layer provides a full-duplex communication

channel for these layers. It carries out various functions which will be discussed later. The data is handled as frames in the MAC. The next chapter, chapter 2, discusses about the basic ideologies used in developing gigabit Ethernet. It gives all the necessary base information on the features of the Ethernet. In the third chapter, the proposed design and its detailed working is discussed. In the fourth chapter, simulation results and implementation of the system is discussed in elaborate. Finally, chapter 5 gives the conclusions of the thesis with a brief discussion about the further development and future scope of the design. Chapter 6 gives the list of references used for the thesis.

2. BASIC IDEOLOGIES

The basic ideologies include the functions of the MAC, details about MAC addresses, gigabit Ethernet frame, CRC and GMII.

The functions of MAC include frame de-limiting and frame recognition, source and destination addressing, assurance of transparent data transfer in the Ethernet sub-layer, error protection by generation and checking of the FCS and finally, the control of accessing the physical transmission medium. It receives and transmits data in the form of frames. It checks the FCS of the received frames and appends FCS to the transmitted frames. It discards malformed or error frames. It prepends the preamble, SFD and padding while transmitting and removes the same while receiving.

MAC addresses are assigned so that each network interfaces can be identified uniquely in the physical network. These are assigned mostly by the manufacturer itself and will be stored in its hardware itself. If so, it is known as burned-in address. It is also known as Ethernet hardware address. It is expressed in groups of two hexadecimal numbers either separated by hyphens or colons or dots.

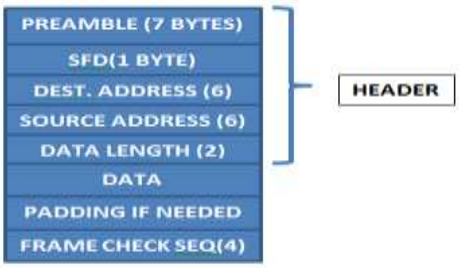

Fig 2: Gigabit Ethernet frame.

The data is handled in the form of packets known as frames in the MAC. As shown in the figure 2, the frame consists of 7 bytes of preamble, 1 byte of SFD, source address, destination address and data length. These constitute the header of the Ethernet frame. The addresses are of 6 bytes each. After the header, comes the data. The data field should be at least 46 bytes and should be at most of 1500 bytes. If the length is less than 46 bytes, padding is provided. The last field is the frame check sequence (FCS). It includes the cyclic redundancy check code for the data.

The CRC is used for error detection purposes. We use a 32 bit CRC polynomial for our purpose. The corresponding hex-code is given as 'C704DD7B'. The operation is EX-OR division. The whole frame minus the preamble and the SFD is considered for this. This data is divided by the CRC polynomial and the resulting remainder obtained will be the CRC code. This is then reversed and attached to the frame to avoid 'false negative' phenomenon. At the receiver end, dividing with the CRC polynomial yields zero remainder and ensures correct transmission. If the remainder is non-zero, the transmission is faulty.

Gigabit media-independent Interface (GMII) provides an interface between the MAC and the PHY. The various GMII transmitter and receiver signals include:

- GTXCLK – clock signal for gigabit TX signals (125 MHz)
- TXCLK – clock signal for 10/100 Mbit/s signals

- TXD[7..0] – data to be transmitted
- TXEN – transmitter enable
- TXER – transmitter error (used to corrupt a packet)
- RXCLK – received clock signal (recovered from incoming received data)
- RXD[7..0] – received data
- RXDV – signifies data received is valid
- RXER – signifies data received has errors

GMII defines speeds up to 1000 Mbit/s, implemented using a data interface clocked at 125 MHz with separate 8-bit data paths for reception and transmission. It can also operate on 10 or 100 Mbit/s as per the MII specification by selecting suitable clocks.

3. PROPOSED GIGABIT ETHERNET MAC DESIGN

This model supports full duplex operation only. The physical layer interface is done using 16 bit lines. Since full duplex, there are two separate 16 bit channels for the interface; one for transmission and other for reception. The data from upper layer is received serially and converted to 8 bit patterns. The internal buses of the MAC are of 8 bits. The 8 bit data is converted to serial bit stream to transmit out of the MAC to the upper layer. The figure 3 below represents a rough sketch of the proposed model. Data is handled as frames in the MAC sub layer.

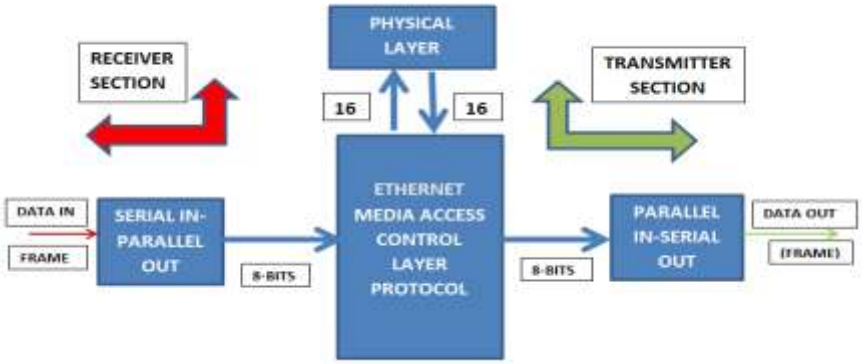

Fig 3: Proposed model of the Gigabit Ethernet Media Access Control.

As we have already discussed, this system is Finite State Machine (FSM) based. We manage the Ethernet frame and the data with the help of different FSMs. We use six different FSMs in our design. First, let us discuss the transmission of the stored data packet. The clock that we use is of 125 MHz, derived from the 30 MHz clock on board.

The process is as follows: This process is in the local clock domain. It gets data and puts it into a RAM. Once a packet worth of data has been stored, it is sent to the packet sending state machine. The data comes from the physical layer as 16 bit stream and gets to the RAM. This is to be converted into the frame format by appending the preamble, start of the frame delimiter and the frame check sequence. Then it is to be transmitted out serially from the MAC. A representation of the packet sending machine is given in the figure 4. This FSM has four states namely GET LENGTH, GET DATA, SEND PACKET and WAIT: NOT DONE. The first 16 bits represents the length of the data packet. Therefore, the first state has the function of getting the data length. Once it is attained, the control is transferred to the next state: to get the data. On the previous state, a counter was set with terminal count set to the value of the data length. On this state, upon getting each data, this counter, which was initially reset, will be incremented. Since we get 16 bits at a time, we increment the counter by two, since we are handling data byte-wise in MAC.

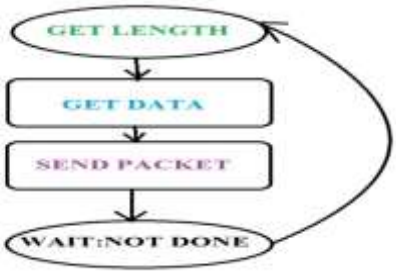

Fig 4: Packet sending machine FSM.

Upon getting the final data, the counter reaches the terminal value and the control is transferred to the next state, i.e. the SEND PACKET state. Here, the data is sent. Once sent, we wait for the synchronization with the receiver. Until we receive a 'done synchronization' signal from the receiver, the control stays in the next state, the WAIT: NOT DONE state. Once the synchronization signal is received, we go out of this state and goes back to the first state, where we get the length of the next data packet.

The next FSM is used to transmit the stored packet via the PHY. It does the job of creating the frame format for the data packet to be transmitted. This appends the preamble, SFD, creates the CRC code, and sends the CRC code with the data. The sketch of the FSM is given in the figure 5. In the first state, the machine waits for the arrival of the new data packet. Its arrival is denoted by a reception of a 'go' signal. Until this signal is received, the machine stays at the wait state. This is helpful in synchronizing the transmitter and receiver side. When this signal is received, it moves to the next state.

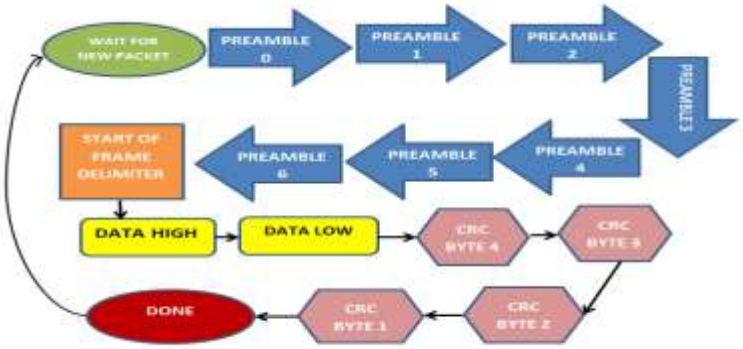

Fig 5: Transmission physical state FSM.

It constitutes 7 states of preamble. On each clock, a byte of preamble is added. This is done to synchronize the receiver clock. It appends alternating patterns of 1's and 0's and corresponding hexadecimal code is '55'. After the 7th byte is appended, it goes to the next state, where frame start delimiting is done. It disturbs the alternating sequence denoting the end of the preamble. After this, the frame includes the data part. The data include the length of the data, the destination and source address, the actual data and padding (if any). Since the packets are received 16 bits at a time, we need two states for the data; one for 8 MSB's and other for the 8 LSB's. Hence we have the states 'DATA HIGH' and 'DATA LOW'. Upon getting each byte of data, before transmitting it, the CRC is calculated. The starting CRC polynomial is (0 1 2 4 5 7 8 10 11 12 16 22 23 26 32) and new CRC is generated. Then with this, the same operation is done to the next byte of data. This is done to every byte of data to be transmitted including the addresses and length field too. The last 32 bit CRC that we get will be the one we will be sending along with the data to the receiver.

The next stages will be for appending the 32 bit CRC to the frame. This is done in 4 states because data is handled 1 byte at a time. Before sending, the CRC calculated is reversed and complemented to avoid the false negative situation discussed in section 3. At last, there is a done state where we generate an indicator to notify that the transmission of the present data is done. The control is then transferred back to the first state, where the transmitter waits for the new packet of data to be encapsulated to frame and transmitted. The data we converted to frame format, now is a sequence of byte patterns (8 bits) emerging from the MAC. Since we need to transmit the data over a serial communication channel, we need a parallel-to-serial converter for the purpose. This too, is implemented as an FSM. First, a data byte is received and is then transmitted one bit at time using 8 separate states. Wait states are also added for the synchronization purpose.

Now, let us discuss the other part of the full duplex communication: the reception. Like transmission, it is also done with the help of FSMs. The frames are received from the upper layer and the envelopes are removed in the MAC sub layer before giving it to the physical layer. During this process, the MAC does CRC checking and error frames or incorrectly received ones will be discarded, ensuring correctness of the data. The data in frame format received from upper layer is serial in nature. First we need to convert it in to an 8-bit format as the MAC we designed uses 8 bit internal buses. For this purpose, we need a serial to parallel converter, which we have implemented as an FSM. Each data is shifted and stored in a register and upon reaching 8; the register value is given out, ensuring the conversion complete. The data will be in full frame format and we have to do two main operations on it: one being the redundancy check and other being the removal of the envelope, before transferring it to the lower layer.

Let us now see how received data is handled in the MAC. We use FSM for this also. This process reads data out of the PHY and puts it into a buffer. There are many buffers on the receiver side to cope with data arriving at a high rate. If a very large packet is received, followed by many small packets, a large number of packets need to be stored.

The figure 6 shows the FSM used for the handling of the received byte sequenced frame. The machine remains in a wait state until a 'data valid' signal is received. When it gets this signal and when the received byte is '55' (hexadecimal value), the machine moves to the next state, that is the preamble state. It remains in this state until a variation in the bit pattern, i.e. a 'D5' is received. After the delimiter is received, the next received ones will be the data we need. It may include the destination and source addresses, the length of the data and the data itself. Thus, control goes to the next state, the 'data high' state. Because we need to convert the byte pattern to 16 bit pattern to transfer out to the physical layer, the first received byte is handled by the DATA HIGH state and the second received byte by the 'DATA LOW' state and is combined to form 16 bit pattern. CRC checking of the received bytes is done on both the states. Then the control goes to the state: 'END OF FRAME'. If any abnormalities like mismatch while CRC calculation or number of bytes received is lesser than 64 or greater than 1518, which are all errors in the reception, the frame is discarded and control is transferred back to the first state: 'WAIT START'.

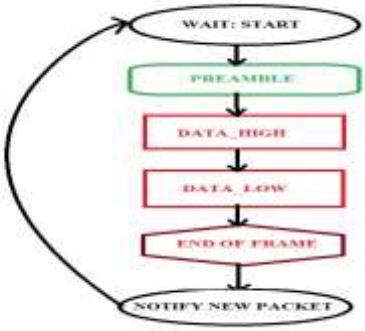

Fig 6: Receiver physical state FSM.

If none of these problems are observed, then control is transferred to the state, which is the 'NOTIFY NEW PACKET' which is used to notify that reception was successful and the machine is ready to accept new packets. All the buffers and registers used for the reception purpose are reset to initial default values in this state. The buffers include receiver start address buffer, packet length buffer, receiver write buffer and CRC register. Once all the sufficient steps are taken to start accepting new packet, the control is transferred to the first state, 'WAIT START' from this state.

Next we discuss about the packet receiving machine. The figure 7 denotes the FSM for the same. The first state is the wait to initialize state, where the receiver read buffer is reset to zero. When a synchronization signal, which indicates that the receive buffers are ready, the control moves to the next state, i.e. the 'WAIT: NEW PACKET'. In this stage, we wait for the arrival of a new packet. When new packet arrives, first the length is sent. It is attained in the next state 'SEND LENGTH'. Then, the following states include the prefetching of the addresses. Once that is done, the data is sent until the read address matches the end address. At this point of time, the control is given back to the first state.

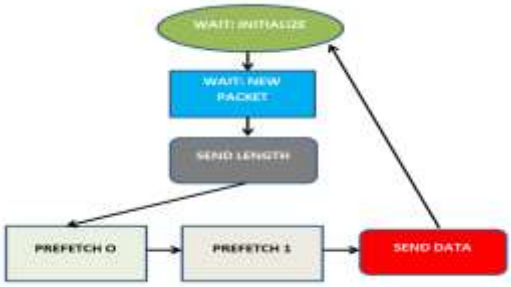

Fig 7: Packet receiving FSM.

4. SIMULATION RESULTS

The programming was done using VHDL on Xilinx ISE design suite version 14.1. The simulations are done in ISim simulator. The project was implemented on Spartan 3 xc3s200-5tq144 board.

First, let us see the simulation result of the transmitter section. Here, a continuous 16 bit pattern of '001b' (represented in hexadecimal form) is given as data packet to be transmitted. The MAC encapsulates the data with preamble, SFD and FCS and converted into frame format to be transmitted out to the upper layer.

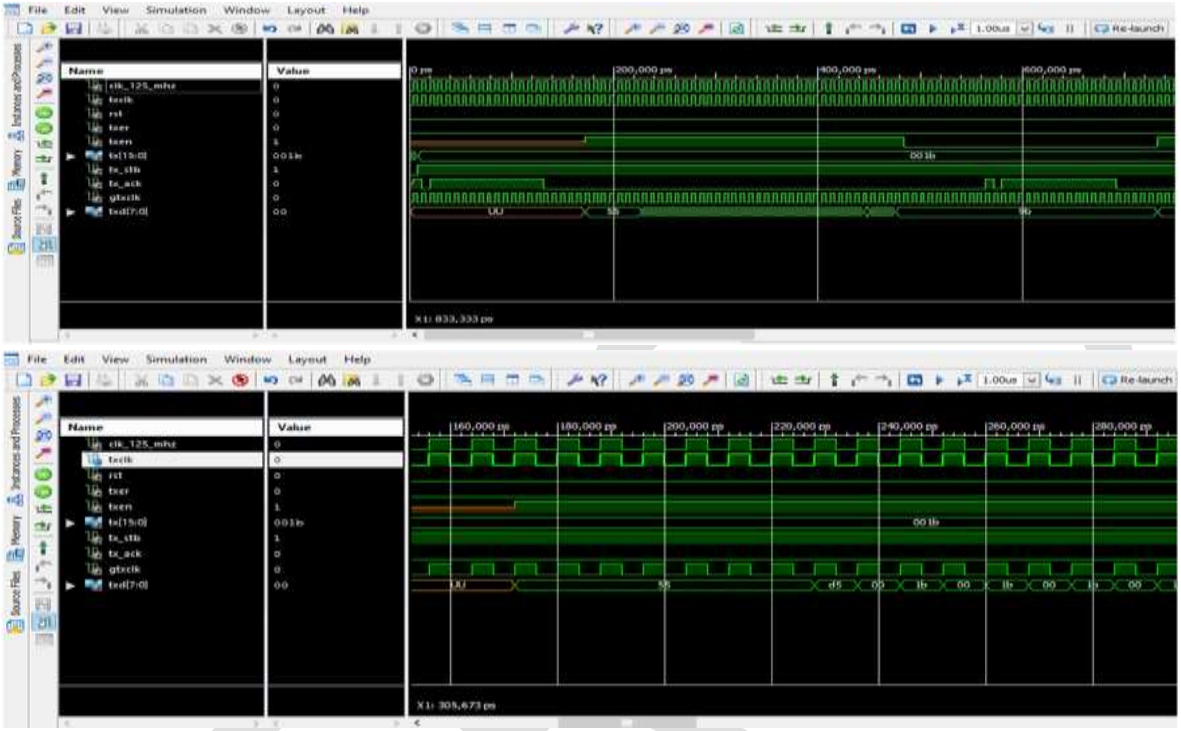

Fig 8: Simulation result of the transmitter section of MAC.

Next is the simulation result of the 8 bit parallel to serial converter FSM. A sample data of 01010101 is given to the machine and as you can see, the output tx is a serial one and each bit is transmitted one-by-one.

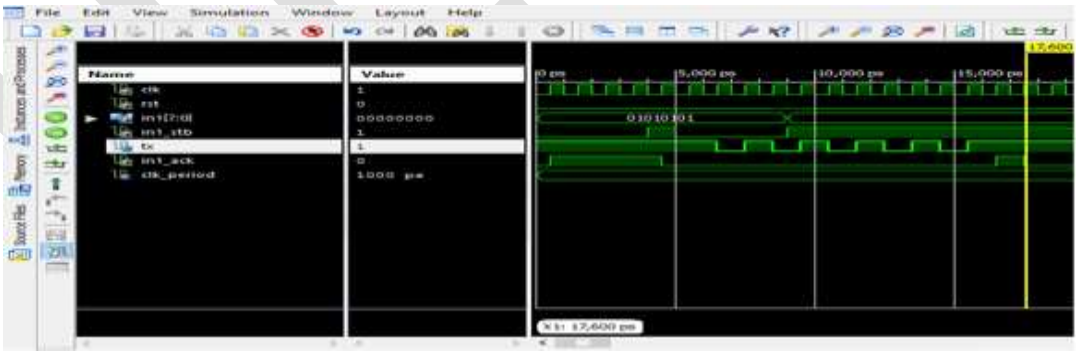

Fig 9: Simulation of the parallel to serial converter.

Now, let us see the simulation result of the receiver section. We use a sample frame with dummy destination and source addresses and a finite length and sample data for the simulation purpose. As you can see, the preamble and the CRC being discarded by the receiving machine. We obtain the output in the sequence of 16 bits. The input RXDV determines whether the received data is valid or not. This will be provided by the transmitter. So, the data received only when this signal is active is considered as the needed one and the rest is discarded.

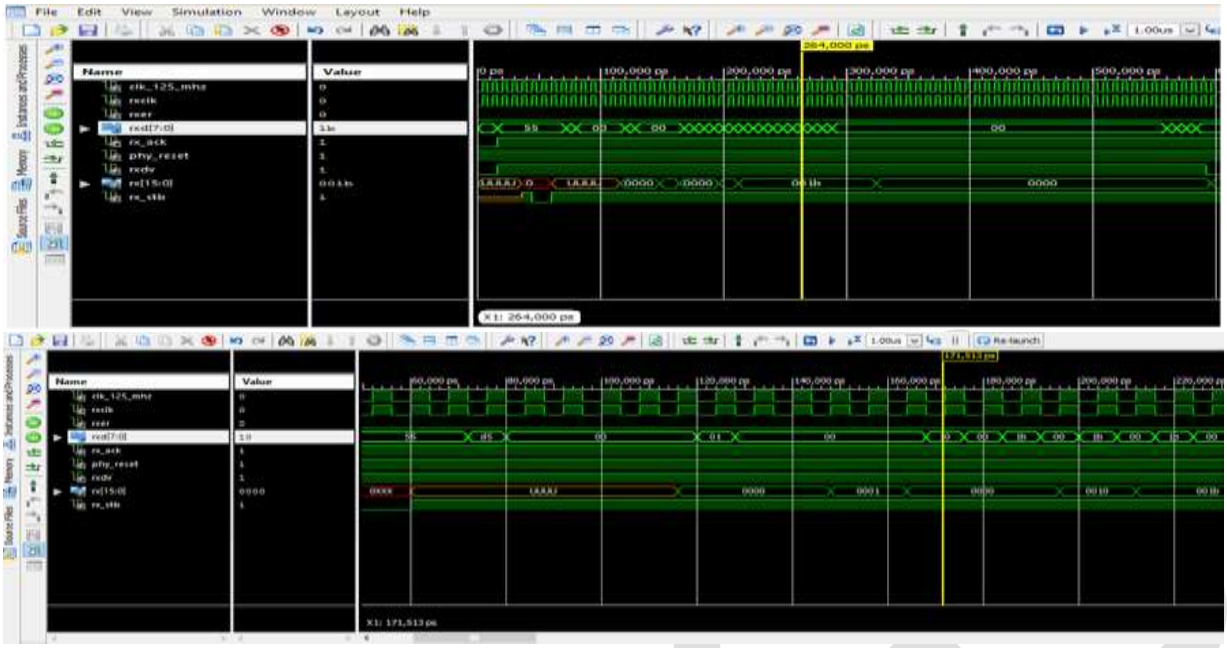

Fig 10: Simulation result of the receiver section of the MAC.

The serial data coming from the upper layer need to be converted to 8 bit parallel data and the simulation result of the machine is shown in figure 11.

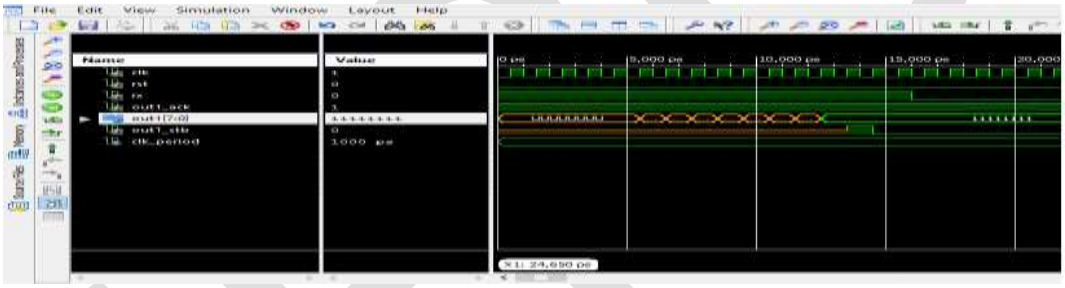

Fig 11: Simulation result of serial to parallel converter.

5. CONCLUSION

Gigabit Ethernet MAC was implemented on the FPGA using VHDL with a clock of 125 MHz and 8-bit bus width. As there is only 8 bits handling at a time, the need of encoding of the data was negotiated in this project. The whole system is constructed on the basis of finite state machine concept and ensures efficient transmission and reception of data between physical layer and the upper data link layer. Every module was constructed independently and was later combined to form the whole system. Effective CRC generation and checking was ensured. The whole system was simulated successfully and implemented on the Spartan 3 FPGA board. It was observed that the data was transmitted and received at a speed of 1 Gbps with the new MAC. Selecting higher clock rate further increases the speed of the system. Also, selecting higher data width on the internal components can also increase speed of the Ethernet but this comes with the overhead of encoders and decoders needed in the MAC.

REFERENCES:

[1] R. M. Metcalfe. "Computer/Network Interface Design: Lessons from Arpanet and Ethernet ", IEEE Journal on Selected Areas in Communications SAC-11(2), pp.173-180,(1999).

[2] D. J. Aldous "Ultimate Instability of Exponential Back-Off Protocol for Acknowledgement-Based Transmission Control of Random Access Communication Channels", IEEE Transactions on Information Theory IT-33(2), pp. 219-223,(1997).

[3] G. T. Almes and E. D. Lazowska, "The Behaviour of Ethernet- Like Computer Communication Networks", Proc. 7th Symposium on Operating Systems Principles, pp.66-81, (2004).
[4] D. R. Boggs, J. C. Mogul, and C. A. Kent "Measured Capacity of an Ethernet: Myths and Reality", ACM SIGCOMM '02 Symposium on Communications Architectures & Protocols, pp.222- 234 , (2002).
[5] G. A. Cunningham and J. S. Meditch 'Distributed Retransmission Controls for Slotted, Nonpersistent and Virtual Time CSMA', IEEE Transactions on Communications COM-36(6), pp.685-691(2000).
[6] G. Fayolle, E. Gelenbe, and J. Labetoulle "Stability and Optimal Control of the Packet Switching Broadcast Channel", Journal of the ACM 24(3), pp.375-386 , (1999).
[7] J. Goodman, A. G. Greenberg, N. Madras, and P. March ''Stability of Binary Exponential Backof ", Journal of the ACM 35(3), pp.579-602, (2001).
[8] A. Leon-Garcia."Probability and Random Processes for Electrical Engineering",Addison-Wesley,Reading, (1989).
[9] Qiu Dong-li, Tang Lin-bo, Zhao Bao-jun, Sun Xing, "Design and Implementation of Gigabit Ethernet Based on SOPC", International Conference on Control Engineering and Communication Technology (ICCECT), vol. pp. 274–277, (2012).
[10] D. Thomas and K. S. Mohanachandra Panicker, "VLSI implementation of gigabit Ethernet with data compression and decompression", IET-UK International Conference on Information and Communication Technology in Electrical Sciences (ICTES 2007), pp. 826 – 831, (December 2007).
[11] G. Ciaccio, M. Ehlert and B. Schnor, "Exploiting Gigabit Ethernet capacity for cluster applications", 27th Annual IEEE Conference on Local Computer Networks, Proceedings. LCN 2002, pp. 669–678, (2002).
[12] J. Mache, "An assessment of Gigabit Ethernet as cluster interconnect," 1st IEEE Computer Society International Workshop on Cluster Computing, pp. 36-42, (1999).