AN UDP/IP NETWORK STACK IN FPGA

Fernando Luís Herrmann, Guilherme Perin, Josué Paulo José de Freitas, Rafael Bertagnolli and João Baptista dos Santos Martins

Federal University of Santa Maria (UFSM) - Microelectronics Group (Gmicro)
Post-Graduate Program in Informatics (PPGI)
Av. Roraima n. 1000, Santa Maria, Rio Grande do Sul, Brazil
Email: (herrmann, guilhermeperin, josue.freitas, rafaelbertagnolli)@mail.ufsm.br, batista@inf.ufsm.br

ABSTRACT

This paper presents a proposal of a UDP/IP network stack in FPGA, which is the stack of the widely used in VoIP and Video-conference applications. This network node implements the Network, Transport and Link Layer from a traditional stack. This architecture is integrated and developed using Xilinx ISE tool and synthesized to a Spartan-3E FPGA. We show architecture details, timing and area results of a practical prototyping. Our proposed network node show a 1960 Mbps full-duplex throughput. Also, we compare our prototype and results with some other three works in terms of area (Xilinx slices) and speed (MHz). Our architecture obtained a intermediate solution in area and is the best implementation in terms of speed among the compared works.

1. INTRODUCTION

Nowadays, the great need of communication in society has collaborated to appear news forms of communication, that are more accessible and lower cost, for example Voice over IP (VoIP) or video conference. But, in a microprocessor of general purpose, this applications competes equally in processing time with other applications, causing a overload in the processing. In order to solve this problem, solutions implemented in dedicated hardware, ASICs or FPGAs become available. This solutions allow that part of the processing, instead of being realized by the microprocessor of general purpose, now can be executed by a dedicated hardware.

The applications, VoIP and video conference using audio and video streaming, respectively[8] [10]. This type of application uses the UDP/IP protocol for transmission data between its participants.

Some hardware UDP/IP stacks have already been realized. In [6], the author describes an analysis of FPGA-based UDP/IP stack parallelism for embedded Ethernet connectivity. In [12] an analysis of the TCP/IP sub-functions are made and the work describes the performance-critical functions that can be accelerated in FPGAs, how these sub-functions may be implemented and what speed-up gains that can be achieved. [11] Shown a RTP/UDP/IP protocol in Virtex FPGAs to accelerating VoIP applications. [1] Proposed 5 design guidelines and a corresponding architecture in TCP/IP Offload Engine (TOE). [7] Propose an implementation of UDP/IP protocol stack on FPGA and its performance evaluation.

The authors would like to thank CNPq and the FINEP/SEBRAE program for supporting the project VoIPWIFI and this work.

Our work presents a stack UDP/IP implementation and makes a comparison with some other existing works.

2. NETWORK STACK

The UDP/IP protocol is part of the Open Standards Interconnect (OSI) model. OSI is a theoretical model and is used to describe the behavior of a network and also to describe networking issues. Figure 1 shows, the OSI model consists of seven layers and the layers are named (starting from the highest layer): Application, Presentation, Session, Transport, Network, Link and Physical Layer. From a TCP/UDP/IP viewpoint the Session and Presentation Layers are often included in the Application Layer. The OSI layers are frequently referred in this paper, but it's not further explained. For a detailed description of the layers and protocols, see [10].

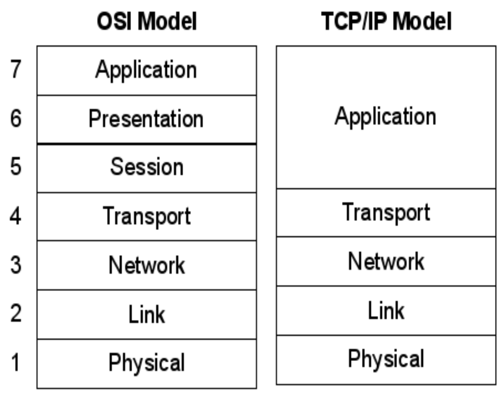

Figure 1. OSI and TCP/IP Model.

2.1. Link Layers

As for serial links, the link layer provides data exchange between neighboring computers as well as data exchange between computers within a local network. For the Link Layer, the basic unit of data transfer is the data link packet frame. A data frame is composed of a header, payload, and trailer.

A frame carries the destination link address, source link address, and other control information in the header. The trailer usually contains the checksum of the transported data. By using the checksum, we can find out whether the payload has been damaged during transfer. The network-layer packet is usually included in the payload.

2.2. Network Layer

The Network Layer is responsible for establishing the route to be used between the originating and destination computers. End-to-end reliability across several physical links is more of a function of the Transport Layer.

The basic unit of transfer is a datagram that is encapsulated in a frame. The datagram is also composed of a header and data field. Trailers are not very common in network protocols. The datagram header, together with data (network-layer payload), creates the payload or data field of the frame.

The Network Layer is used to establish communications with computer systems that lie beyond the local LAN segment. It can do so because it has its own routing addressing architecture, which is separate and distinct from the Link Layer. Such protocols are known as routed or routable protocols, for example Internet Protocol (IP) [9].

The Internet Protocol (IP) is the most important protocol of the Network Layer. IP attempts to deliver messages to the destination, which is selected by a unique IP address.

Address Resolution Protocol (ARP) is a network-specific standard protocol. The address resolution protocol is responsible for converting the higher-level protocol addresses (IP addresses) to physical network addresses.

The Internet Control Message Protocol (ICMP) protocol is part of the Internet Protocol Suite. ICMP messages are typically generated in response to errors in IP datagrams or for diagnostic or routing purposes [4].

2.3. Transport Layer

A network layer facilitates the connection between two remote computers. As far as the transport layer is concerned, it acts as if there were no modems, repeaters, bridges, or routers along the way. The transport layer relies completely on the services of lower layers. It also expects that the connection between two computers has been established, and it can therefore fully dedicate its efforts to the cooperation between two distant computers. Generally, the transport layer is responsible for communication between two applications running on different computers. There can be several transport connections between two computers at any given time (for example, one for a virtual terminal and another for email). On the network layer, the transport packets are directed based on the address of the computer (or its network interface). On the transport layer, individual applications are addressed. Applications use unique addresses within one computer, so the transport address is usually composed of both the network and transport addresses [4].

In this case, the basic transmission unit is the segment that is composed of a header and payload. The transport packet is transmitted within the payload of the network packet.

The Transport Layer provides end-to-end reliability by having the destination host communicate with the source host. The idea here is that even though lower layers of protocols provide reliable checks at each transfer, the end-to-end layer double checks to make sure that no machine in the middle failed [2].

The Transmission Control Protocol (TCP) is the most used protocol of Transport Layer which gives a connection-oriented communication with reliable data delivery, duplicate data suppression and flow control. Another Transport Layer protocol is the User Datagram Protocol (UDP), which provides an unreliable and connectionless communication service. However, UDP is very effective when TCP is not suited for the application needs, e.g. for real-time applications like audio and video or in applications where low latency and low delay is preferred over reliable data delivery [5].

3. IMPLEMENTATION

The hardware UDP/IP stack core that's described in this paper is shown in figure 2. Figure 2(a) showed the architecture of a traditional TCP/IP stack. In figure 2(b), as the shadowed area, is showed the cores which are entirely implemented in the FPGA, in other words, the Transport Layer, Network Layer and Link Layer.

UDP is used for the Transport Layer. All TCP packets will be passed directly to the Application Layer, in other words, TCP packets will not be changed by the Transport Layer and this work must be done by the Application Layer.

For Network Layer the Internet Protocol version 4 (IPv4) is used, which gives a more area-effective design compared to the more recent IPv6 protocol.

The Layers, Transport, Network and Link in UDP/IP stack are designed using Verilog and VHSIC Hardware Description Language (VHDL).

The hardware UDP/IP stack implementation use the design structure which is showed in figure 3. This implementation is full-duplex because the Transmitter and the Receiver works simultaneous and independent. The subsections below provide a detailed description of the functionality of each block.

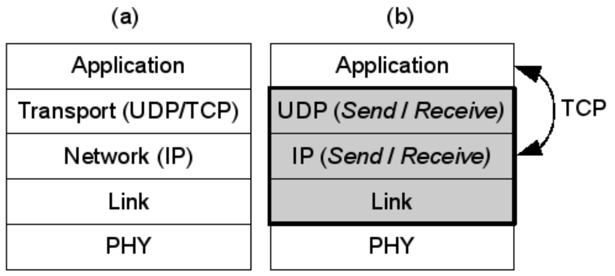

Figure 2. (a) Architecture of a traditional TCP/IP stack. (b) Architecture used by this work and which layers are implemented. Redraw from [8]

3.1. Control Transmitter/Receiver

These blocks provide the communication with the application layer. The Control Transmitter receives the packet from the application and stored it in the RAM Transmitter for sending forward to the block UDP Transmitter.

The Control Receiver writes the packet from UDP Receiver in RAM Receiver and sends it to the application layer.

3.2. UDP Transmitter/Receiver

These blocks represents the transport layer and manages UDP packets. If the packet a TCP, these two blocks just send the packet. The block UDP Transmitter encapsulate the packet with the UDP header and sends out to block IP Transmitter.

The block UDP Receiver check the packet and sends it to block Control Receiver without UDP header information.

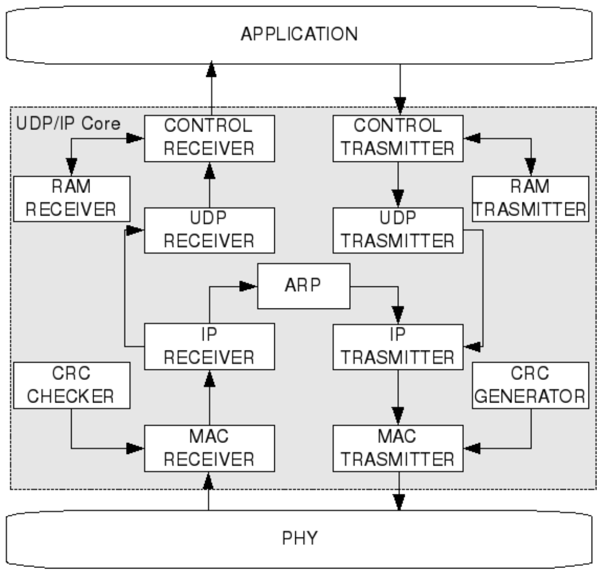

Figure 3. Block diagram of the UDP/IP core

3.3. IP Transmitter/Receiver

Represents the network layer and these blocks manages IPv4 packets. The block IP Transmitter calculates the checksum and encapsulate the packet with the IP header.

The block IP Receiver will be verified the checksum of the packet and the destination IP-address. Only IP-address that matches the core's IP-address and broadcast IP-addresses are accepted and send to block UPD Receiver or to ARP block if the packet a Address Resolution Protocol (ARP). If the packet check fails the packet will be rejected.

3.4. MAC Transmitter/Receiver

These blocks represents the link layer and manages the outcoming and incoming packets. The MAC Transmitter will be send the packet to the PHY. At the beginning, preamble is sent, where the last nibble is a start of frame delimiter. The MAC transmitter then puts out the transmit packet to the PHY data bus and sets control signals. Each byte is sent to the CRC generator, which progressively calculates the CRC. When the packet end is reached the calculated 32-bit CRC is sent.

The MAC Receiver will check for a new packet (preamble from Ethernet PHY). Once a new packet is detected will be sent to the CRC checker, which progressively calculates the checksum. When the end of frame is signaled from the Ethernet PHY the CRC check will be completed and the destination MAC-address will be verified. Only MAC-addresses that matches the core's MAC-address and broadcast MAC-addresses are accepted. If the packet check fails the packet will be rejected.

3.5. ARP

This block provide ARP functionality and manages ARP packets. Allowing the core to request the MAC address from other nodes when only the IP address of its neighbors is known. The core will also respond to ARP requests from neighboring nodes.

3.6. RAM Transmitter/Receiver

These blocks just temporarily stores the packets.

3.7. CRC Checker/Generator

These blocks are identical and progressively calculate the Cyclic Redundancy Check (CRC). It uses the CRC32 polynomial for Ethernet. The polynomial is shown below:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X^{1} + 1$$

A 32-bit CRC provides error detection in the case where line errors (or transmission collisions in Ethernet) result in corruption of the MAC frame. Any frame with an invalid CRC is discarded by the MAC Receiver without further processing. The MAC protocol does not provide any indication that a frame has been discarded due to an invalid CRC. The link layer CRC therefore protects the frame from corruption while being transmitted over the physical medium.

4. EVALUATION AND RESULTS

The UDP/IP Core was developed under Xilinx ISE 10.1 and passed through procedures of "Synthesis" and "Place and Route" for the Xilinx Spartan 3E, XC3S500e-4FG320 FPGA. Three simulation sub procedures were performed: "Functional", "Post Synthesis" and "Post Place and Route". Input values were supplied by macro files and test benches. Test benches also provided an overall report which includes signal comparison and signal sequence comparison, incoming TCP and UDP packets.

Table I shows the resource utilization and the speed results of the UDP/IP stack in the Spartan 3E, XC3S500e-4FG320 FPGA. The fastest blocks for this core are CRC Checker and CRC Generator due the small number of include logic. But, blocks with more complexity and bigger sizer tend to lower speeds, as example, blocks IP Transmitter and Control Transmitter.

The maximum core speed is the same of the block IP Transmitter, 122.76 MHz, because this block has the lowest speed. Considering this information and that this design can send and receive 8 bits per cycle we can estimate the maximum throughput roughly about 980 Mbps which means a 1960 Mbps full duplex throughput.

We compare our work in terms of area (Xilinx Slices) and speed (MHz) with other implementations and we get the results that can be visualize in the figure 4. The chosen implementations were Löfgren "Medium" and Löfgren "Advanced"of [6] and Dollas in [3]. [6] Cites three implementations, which are Minimum, Medium and Advanced, but only Medium and Advanced were selected, because the Minimum doesn't have the ARP protocol implemented. [3] Presents a complete TCP/IP stack implementation, therefore only related data to UDP/IP implementation were observed.

In occupied area terms, the left part of figure 4, our implementation has some more slices than Löfgren "Medium", but when comparing with Löfgren "Medium" and Dollas our occupied area is smaller. The speed of the architecture is presented at the right side of the figure 4 and is possible see that our work presented a better performance.

Table I. Resource utilization and subsystem speed

Subsystem	Slices	% FPGA	Speed (MHz)
Control Transmitter	215	2.31	128.20
Control Receiver	202	2.17	190.13
RAM Transmitter	54	0.58	218.20
RAM Receiver	54	0.58	218.20
UDP Transmitter	110	1.18	196.10
UDP Receiver	87	0.93	181.60
IP Transmitter	123	1.32	122.76
IP Receiver	115	1.23	275.94
ARP	85	0.91	130.40
MAC Transmitter	101	1.08	153.30
MAC Receiver	95	1.02	177.72
CRC Generator	40	0.43	295.64
CRC Checker	40	0.43	295.64
Total	**1321**	**14.19**	**122.76**

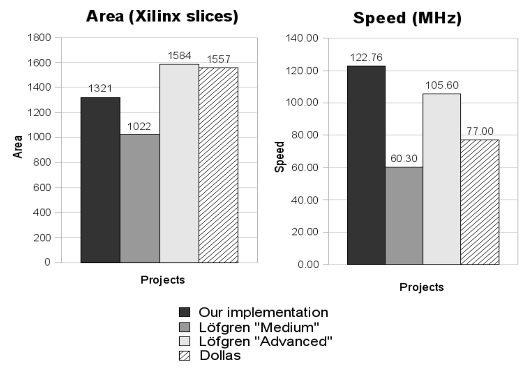

Figure 4. Projects comparison in area and speed

Our implementation presents 2 times more speed performance that Löfgren "Medium" and occupies just 300 more slices. Depending of the application purposes, is accepted that a more few elements in area terms and a better speed performance is advantageous for transmission and reception data.

5. CONCLUSIONS

This paper shows a UDP/IP network stack in FPGA. We present a core that is successfully implemented and verified in Xilinx Spartan 3E. The hardware UDP/IP network stack is using around 14% (1,321 Xilinx slices) of Xilinx Spartan 3E XC3S500e-4FG320 FPGA and its maximum frequency operation is 122.76MHz. Considering the maximum frequency and that this design can send and receive 8 bits per cycle we can estimate the maximum throughput roughly about 980 Mbps which means a 1960 Mbps full duplex throughput.

Also, we present a comparison with other three works, in terms of area (Xilinx slices) and speed (MHz). In occupied area, our implementation is like an intermediate solution. But, in speed our work obtained the best results among the compared works. Our implementation presents 2 times more speed performance that Löfgren "Medium" and occupies just 300 more slices. Depending of the application purposes, is accepted that a more few elements in area terms and a better speed performance is advantageous for transmission and reception data.

6. REFERENCES

[1] S.-M. Chung, C.-Y. Li, H.-H. Lee, J.-H. Li, Y.-C. Tsai, and C.-C. Chen. *Design and implementation of the high speed TCP/IP offload engine*. In Proceedings of the 7th International Symposium on Communications and Information Technologies, pages 574–579, 2007.

[2] D. E. Comer. *Internetworking with TCP/IP*. Volume 1. Prentice Hall, 5th edition, July 2005.

[3] A. Dollas, I. Ermis, I. Koidis, I. Zisis, and C. Kachris. *An open TCP/IP core for reconfigurable logic*. In 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05), pages 297–298, 2005.

[4] L. Dostálek and A. Kabelová. *Understanding TCP/IP: A Clear And Comprehensive Guide*. Packt Publishing, April 2006.

[5] P.-K. Lam and S. Liew. *UDP-Liter: an improved UDP protocol for real-time multimedia applications over wireless links*. In 1st International Symposium on Wireless Communication Systems, pages 314–318, 2004.

[6] A. Löfgren, L. Lodesten, S. Sjoholm, and H. Hansson. *An analysis of FPGA-based UDP/IP stack parallelism for embedded Ethernet connectivity*. In Proceedings of NORCHIP Conference, Oulu, Finland, pages 94–97, 2005.

[7] K. Morita and K. Abe. *Implementation of UDP/IP protocol stack on FPGA and its performance evaluation*. In Proceedings of IPSJ General Conference Special5, pages 157–158, 2001.

[8] A. Rodriguez, J. Gatrell, and R. Peschke. *TCP/IP Tutorial and Technical Overview*. Prentice Hall, 7th edition, Upper Saddle River, NJ, USA, 2001.

[9] K. S. Siyan and T. Parker. *TCP/IP Unleashed*. Sams Publishing, 3rd edition, August 2002.

[10] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 4th edition, August 2002.

[11] A. Tavoularis, M. G. Manousos, D. Economou, and G. Lykakis. *Accelerating VOIP applications using Virtex FPGAs*. In FPGA and Structured ASIC Journal, 2004.

[12] L. Weidong. *Designing TCP/IP Functions In FPGAs*. Master's thesis, Code Number CE-MS-2003-09, 2003.