Efficient High-Speed Ethernet for Real Time Simulation

Jowel Yusta, John Zenor, Kurtis Kredo II
McLeod Institute of Simulation Sciences &
Department of Electrical and Computer Engineering
California State University, Chico jzenor@csuchico.edu

Keywords: High-speed, real-time, Ethernet, FPGA.

Abstract

Recent research has focused on developing communication techniques for field-programmable gate arrays (FPGAs) to support real-time, distributed simulation with frame times of a few microseconds or less. It has been demonstrated that very high-speed, efficient, reliable FPGA-FPGA or FPGA-PC inter-model communication can be accomplished with point-to-point Giga-bit Ethernet using the standard UDP and Ethernet protocols. Predictable message timing and reliability required by real-time inter-model communication can be achieved, without introducing the significant overhead usually associated with Ethernet communication and software protocol stacks. This high-speed communication method has been applied to a simulation system with a high-speed simulation of an electric power system converter running in the FPGA communicating with a Virtual Test Bed (VTB) model in a PC, using the VTB for presentation of simulation results.

1. INTRODUCTION

Giga-bit Ethernet has many advantages when used as a communication link for real time simulation. Giga-bit Ethernet interfaces are very fast serial interfaces, with faster bit rates than most dedicated or special-purpose serial interfaces. Even faster Ethernet interfaces are on the horizon. The use of Ethernet interfaces with standard UDP/IP protocols has many advantages. Standard cabling is readily available, both copper wire and fiber. Ethernet traffic is easily monitored using standard PC's and UNIX systems with software programs such as Wireshark. High-speed Ethernet interfaces are commonly available and are supported by a wide variety of computers and network test equipment.

Ethernet is commonly used for TCP/IP communication over office networks and in the internet. In this environment it is not usually considered for high-speed special-purpose links as there is a high overhead associated with the implementation of the TCP/IP protocol stacks, and network contention with message re-transmission results in unpredictable delays and throughput. If the overhead of the IP stack can be limited, Ethernet becomes a more valid choice for real-time simulation communication links.

Ethernet point-to-point links offer some advantages over traditional networks. For one the latency is kept to a minimum, in modern networks the switches and routers introduce delays because packet buffering occurs in order to prevent collisions and handle flow control. For the point-to-point links usually considered for high-speed inter-computer links, the link is full duplex. This allows simultaneous transmit and receive traffic with no possible contention. Collisions and re-transmissions do not occur.

In a multi-system simulation package that used Ethernet for inter-system communication the first attempts to communicate between the FPGA and a PC used a software implementation of UDP/IP in a soft processor implemented inside the FPGA [1]. The implementation of the soft processor tied up a large percentage of the FPGA resources, and there was a significant overhead in the software implementation of the IP stack.

While monitoring the IP traffic over the link, it was observed that only a few IP message types were used, most of the traffic was composed of UDP messages communicating a fixed set of data between simulation models. Almost the only thing that changed between messages was the data itself. Only a small number of bytes in the message headers, such as the message number and header checksum changed from message to message. The entire message checksum also changed, but that was added by the FPGA Ethernet interface hardware. Address Resolution Protocol (ARP) and echo responses (pings) were also required, but only two simple message types had to be implemented. Using this knowledge a design could forgo the use of the soft processor and control the Ethernet directly. In order to achieve efficient, high-speed communication, "canned" message headers could be used, changing only a few bytes in the header and the actual data to be sent. The resulting packets followed applicable standards, and could be accepted by standard PC software or network monitors.

A PC itself is not able to keep up with the high-volume, high-speed traffic produced by a direct Ethernet control implementation. The overhead and time required for message transmission and reception in the FPGA was reduced to the microsecond range, and the soft processor could be eliminated from the FPGA, freeing considerable FPGA resources. The direct Ethernet implementation can be used for high speed FPGA-FPGA communication. The

Ethernet FPGA-PC link could be used for communicating data to low-rate models, user-interface information, or for communicating high-speed results to be plotted in the PC.

For FPGA-FPGA communication the Ethernet link may be used for high-speed communication from a simulation module in one FPGA to another FPGA. When directly connected the communication link can have higher efficiency and throughput than the FPGA-PC link, as messages headers, ARP and PING are not actually required by the FPGA hardware. Small message transfer could be accomplished in the sub-microsecond time frame using these "headerless" message types.

FPGA based direct Ethernet links are high speed communication links that can be used as reliable data links for use in multi-FPGA simulations. When using standard packets the link can be routed by networking equipment and can be used for FPGA-PC communication. Non-standard packets can be used for directly connected FPGA-FPGA links and yield better results. The actual throughput and latency of packet generation is measured for both cases and presented in the following sections. When implemented in this method Ethernet is a high-speed link that can achieve the communication speeds required for use in real time simulation links.

2. BACKGROUND AND HISTORY

Ethernet communication using a FPGA can be implemented in one of two ways, by controlling the Ethernet chip directly or by using a processor to simplify communication. There are advantages and disadvantages of each method and both will be discussed in this section. Processor-based communication can be implemented by creating a processor out of logic gates within the FPGA (soft processor) or having a physical processor on die or on the board (hard processor). Designs that control the Ethernet chip directly (direct Ethernet) do not require a processor, but the user must design complex logic to adhere to Ethernet standards. Direct Ethernet lends itself well to real-time simulations where timing is critical, processor-based communication is beneficial when a complete Ethernet stack is required.

One major advantage of processor-based communication is that it provides a simple structure for communication, allowing the user to write to and read from Ethernet using code written ins C/C++. Processor-based designs usually offer support for a wide range of IP protocols giving the design flexibility. Although soft-processor communication is in most situations the easiest method, it has many drawbacks. Microprocessors are designed to do many tasks and do not provide the most efficient or fastest way of implementing Ethernet communication on FPGAs. The nature of microprocessors make them a poor choice for real-time systems where timing is critical, the processor is a multi-purpose device and is not designed around making timing guarantees. The speed of communication and timing guarantees will depend on the architecture of the microprocessor available for use. A drawback for designs built around a soft processor is that resources are used to implement the processor that could have been used to make a larger or more robust simulation design. Another drawback with regards to timing is that communication between the processor and the design must take place. Options available for interfacing between the user design and the processor include FIFOs, shared memory, and shared register all of which use resources and cost time.

Direct Ethernet communication offers many benefits when compared to processor based designs, but the design process is more difficult and time consuming. A major drawback of direct Ethernet communication is that the design complexity increases with support for larger communication protocol sets. Designs take FPGA resources for each protocol supported. One solution to this problem is implementing a smaller subset of protocols, keeping the complexity of the design and resources used to a minimum.

Designs based around direct Ethernet communication lend themselves well to real-time simulations in many areas. Timing can be guaranteed because the design is implemented in hardware where tasks can be run in parallel, and not affect the timing of the rest of the system. True parallel send and receive can be achieved by implementing transmit and receive sections in separate hardware. Maximum transmission rates can be obtained since the hardware is capable of transmitting packets at the maximum rate as allowed by the Ethernet standard. Another advantage of direct Ethernet is designs can write directly to the outputs negating the need for communication between the hardware design and the soft processor that take time and resources.

The design process for direct Ethernet communication is daunting by appearance with overwhelming amounts of documentation available. When the modules available from FPGA manufacturers are used as part of the design the task can be greatly simplified while still providing impressive speeds. Once the overall design process is understood the implementation is straight forward and does not require extensive knowledge of the inner workings of Ethernet. The following test platform was developed first as part of a graphical system generator based design (a Matlab toolbox that can be used in conjunction with the Xilinx software) and was later developed as a code based VHDL design[2].

3. TEST PLATFORM

The test platform is designed to find the minimum latency and maximum throughput of real time FPGA-based simulations based around direct Ethernet communication.

The design contains the smallest set of protocols necessary to communicate information between one system and another. Standard Ethernet protocols are used, giving the option for packet analysis on PCs and communication with PC-based simulation packages such as the Virtual Test Bed (VTB) software developed by the University of South Carolina [3]. The test design stems from a FPGA based multi-rate simulation with the FPGA running the fast and medium speed simulations and VTB running the slow simulations. Since the focus of this study is Ethernet communication, the simulation models were removed to find the best case communication scenario when using Ethernet as part of a real time simulation platform. The Ethernet design was modified to input and output UDP messages using a block RAM to store data without performing calculations between each send.

Generating a general purpose direct Ethernet implementation with dynamic values takes resources and is a complicated design process. By using UDP as the communication protocol, and not implementing the entire IP stack we keep the Ethernet design as simple as possible. The fact that most of the fields remain unchanged between subsequent UDP messages can be taken advantage of. By keeping all of the unchanging fields static and implementing a limited IP stack we keep the complexity to a minimum.

The design is developed around the ML506 development board and the Virtex 5 FPGA produced by Xilinx. The same techniques can be used for other FPGA-based gigabit Ethernet designs and will yield the same best-case results since we are able to communicate maximum rate as allowed by the Ethernet standard (line speed). The ML506 development board contains many peripherals and interfaces used for project development. This design makes use of the FPGA, the Ethernet PHY chip, output LEDs and input switches. The Ethernet design is based around the GMII interfacing standard, and uses the tri mode Ethernet MAC wrapper available from Xilinx to interface to the PHY chip as shown in Figure 1 [4]. The tri-mode Ethernet MAC wrapper simplifies the communication design and will be discussed in further detail below. The Alaska PHY chip provides the signal translation from the FPGA to the physical medium.

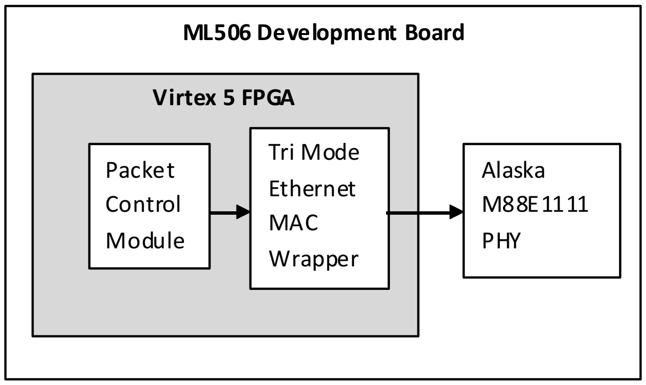

Figure 1. FPGA-to-PHY connection

The tri-mode Ethernet MAC simplifies communication while still giving the ability to transmit at line speed, giving the user simplified inputs and outputs. The project operates at gigabit speed with 4 twisted pairs transmitting 250 Mbps each resulting in gigabit communication [5]. The block wrapper provides FIFOs and control signals at the user level interfacing to the Ethernet MAC wrapper that handles the translation to the GMII standard, simplifying interfacing (Figure 2). Xilinx provides a similar MAC wrapper for the Virtex 6 and other FPGA series, giving the ability to port the current design with minimal effort. The Virtex 6 has greater resources available for simulation use but will not provide an increase in Ethernet speed since the Virtex 5 is able to communicate at line speed.

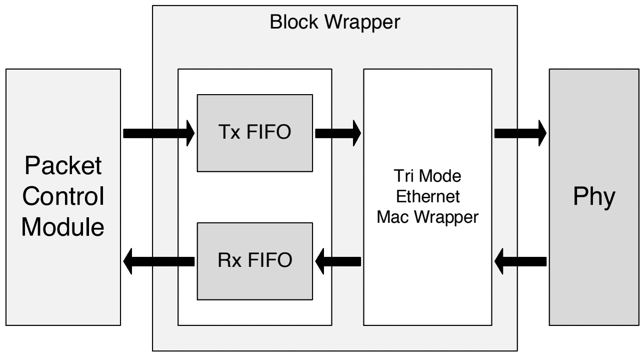

Figure 2. Block level wrapper and connections

The protocols supported by this implementation are limited to ARP, UDP and ICMP (ping). In this project ARP is used to establish a connection between the FPGA and the PC, and ping is a useful debugging tool. UDP is used for the actual data communication but does not provide some of the more robust features provided by other communication protocols such as Transfer Client Protocol (TCP). The advantage of UDP is that it requires less overhead and in this case less packet manipulation than most other protocols. For the purpose of real time simulations using Ethernet as a point-to-point link we do not require the features offered by other protocols and their implementation would take more resources and cost time when used.

The test platform design is based around a packet control module that generates and receives Ethernet packets (Figure 3). The control logic is implemented as a VHDL based state machine, the block RAM is implemented from hard logic on the FPGA, and the timing measurements are done in a separate process to keep it running in parallel to the rest of the design. The packet control module responds to ARP and ping requests, sends ARP requests, and transmits and receives UDP messages. When UDP messages are received, the data field is placed in a block RAM for use by the simulator package. Transmitted UDP messages originate from a RAM block and are transmitted directly into the Ethernet wrapper's FIFO. The format of the UDP messages is shown in Figure 4; the control logic changes the data, ID and checksum fields between successive UDP message transmissions. The ID field within the IP header is incremented between sends and the checksum is recalculated as part of the designs standard UDP transmission sequence. Only 3 header fields need to be changed between subsequent sends and since we do not have the overhead of the whole stack the design is kept relatively simple. The UDP protocol includes an optional checksum field that is not used in this design and is held at 0. Implementing the checksum calculation would complicate the sending logic since the checksum includes the data field as part of its calculation.

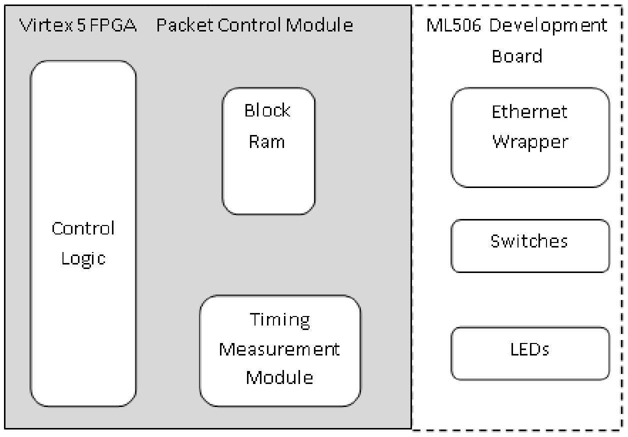

Figure 3. Project overview

Destination MAC 6 Bytes	Source MAC 6 Bytes	Type 2 Bytes	
Header Length 1 Byte	Service 1 Byte	Total Length 2 Bytes	ID 2 Bytes
Flags 1 Byte	Fragment 2 Bytes	TTL 1 Byte	Protocol 1 Byte
Header Checksum 2 bytes	Source IP 4 Bytes	Destination IP 4 Bytes	
Source Port 2 Bytes	Dest Port 2 Bytes	Length 2 Bytes	UDP Checksum 2 Bytes
Data – Variable Length			

Figure 4. Structure of the Ethernet frame

The timing measurements module shown in Figure 3 is implemented in parallel to the transmissions and measures the maximum amount of time it takes for packet sends. The total transmission time for sending multiple packets is also measured within the module. The timing measurements are implemented as a clock counting routine that is active while transmitting. The project runs on a 125 MHz clock and time based measurements can be calculated from this clock. For example if the packet send takes 100 clock cycles, we multiply that by the 8 nS clock period giving us an 800nS packet send.

This test module was derived from a larger simulation package that uses the packet generating module to send and receive data. The simulator reads in initial values and other parameters from an UDP initialization message, from there it performs simulation calculations and transmits the results back to the PC. The project can be run in single rate, where the simulation must receive values after each time step or in multi-rate where the simulator will perform multiple step calculations before requiring communication to occur. In either case the results from the FPGA-based simulation are sent to a PC running VTB which performs a simulation step and returns new inputs and outputs to the FPGA using UDP communication.

4. MEASUREMENTS

Ethernet communication is often considered a slow medium in regards to real-time simulations where frame times in the microseconds are common. In this section we will test the maximum throughput and lowest latency that can be achieved with FPGA-based simulations using Ethernet communication. Results will be presented for communication links for FPGA-to-FPGA based simulations and for FPGA-to-PC based simulations.

4.1. Standard UDP messages

FPGA-to-PC communications require the use of a communication protocol, in this case UDP. The results will tell us the maximum rate a user can communicate with UDP for use in a FPGA-based simulator using standard packets. If the PC is not able to sustain the transfer rates in the test it will drop packets once the PC's buffer is full. The test varies the length of the data field within a UDP message and transmits multiple packets filling the receive buffer on the PC. The format of the UDP message is shown in Figure 4; the header bytes remain constant except the ID, length, and checksum fields between tests.

The average packet send times are measured in clock cycles for varying data lengths and presented in Figure 5 with the data field ranging from 4 bytes to 1024 bytes. The data length to average packet send time is linear as expected for larger packet sizes. Measurements of average send time for smaller data lengths is shown in Figure 6, small increases in length do not correlate to large differences in send times because the headers are 42 bytes in length and take up most of the transmit time.

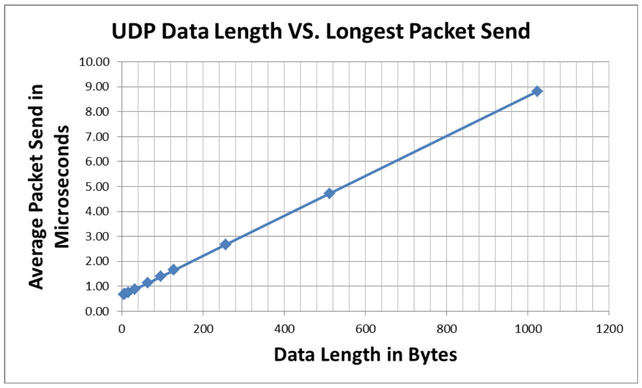

Figure 5. Data send length versus send time

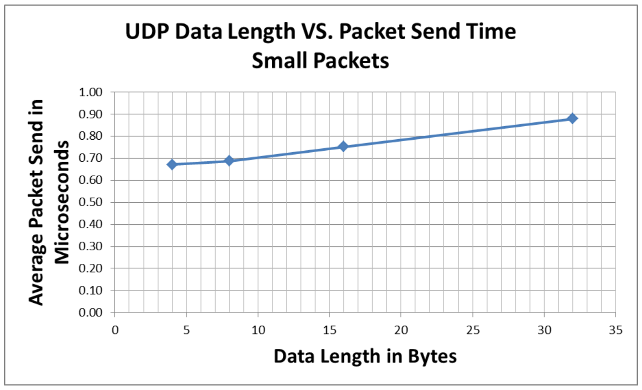

Figure 6. Data send length versus time for small packets

When performing real time simulations we need to know the transmission time that can be guaranteed, this measurement is taken by measuring the longest packet send that occurs within the FPGA. The longest packet send times take more than average because the Ethernet standard requires time flow control request and minimum inter packet wait time between packets [6]. A clock counting module measures the maximum transmission (or longest packet send) in clock cycles for the UDP message transmissions. The longest packet send column in Table 1 shows the guaranteed transmission time for each data length. The guaranteed send times vary depending on the amount of information that needs to be transferred. For a simulation that requires 96 Bytes to be transferred (12 double length floating point numbers) we would need a minimum transmission time of 1.39 micro seconds. Note that this is only the transmission time, for use in a real time system the data would have to be received and processed on the receiving end.

Table 1. FPGA packet send times for UDP messages

Number of UDP messages sent out	Data Field Length	Total Transmit time in clock periods	Total Transmit time (mS)	Average Time/Packet (µS)	Longest Packet send in Clock Periods	Longest Packet Send (µS)
40000	4	3354343	26.83	0.67	93	0.75
40000	8	3434609	27.48	0.69	93	0.75
40000	16	3754791	30.04	0.75	93	0.75
40000	32	4395095	35.16	0.88	109	0.88
40000	64	5675465	45.40	1.14	141	1.14
40000	96	6955703	55.65	1.39	173	1.39
80000	128	16475887	131.81	1.65	205	1.65
80000	256	26716358	213.73	2.67	333	2.67
80000	512	47197003	377.58	4.72	589	4.72
80000	1024	88157801	705.26	8.82	1101	8.82

For the test results in Table 1 the FPGA was connected directly to a PC with a packet monitoring tool running, the number of received packets and time data was tabulated and is shown in Table 2. The table demonstrates the kind of packet loss we can expect from a PC. The computer used to tabulate the result is a quad core Intel based PC running 32 bit windows XP and represents a typical PC used for simulations. To achieve the least latency, the board is connected directly to the PC; switches can introduce transit delays which we are trying to avoid in this test [7]. All measurements in the table are taken from the packet monitoring software Wireshark running on the PC. The total transmit time is calculated from the difference between the time stamp between the first and last received packet, the average time data is calculated based on (total transmit time / total number of packets sent). Once the receive buffers are full on the receiving PC the packets will be dropped as is shown in the number of UDP messages received field of the table.

Table 2. PC behavior: receiving high volume UDP messages

Number of UDP messages sent out	Data Field Length	Number of UDP messages received	Total Transmit time (mS)	Average time (uS)
40000	4	19765	26.92	0.67
40000	8	19828	27.24	0.68
40000	16	21181	29.87	0.75
40000	32	23300	35.19	0.88
40000	64	27809	45.39	1.13
40000	96	30916	55.41	1.39
40000	128	34753	65.92	1.65
80000	128	66792	131.90	1.65
40000	256	40000	106.90	2.67
80000	256	79118	213.75	2.67
80000	512	57569	377.67	4.72
80000	1024	35912	697.89	8.72

The rates measured in this section give us the absolute maximum rates of transmission for standard UDP messages. When the module is used as part of a simulation package where two way communication is required, the send time between step calculations for each packet will be significantly higher. For each packet sent with this wrapper based design, the user fills a FIFO and once the entire packet is in the FIFO the physical transmission of the packet begins, meaning the total sending time is actually the time it takes to fill the FIFO + the physical transmission time

4.2. Non-Standard Packets

Using Ethernet communication for point-to-point communication between multiple FPGA boards would not require use of a standard protocol such as UDP or even Ethernet. This section will measure the transmit time required to send data over Ethernet without the headers and overhead of the UDP protocol. This FPGA-to-FPGA communication requires a direct connection between the boards. Network hardware such as switches and routers would not route the packets correctly since the Ethernet headers are no longer used.

The system was redesigned to send the only the data field from the UDP message shown in Figure 4. The headers are not needed for direct communication and therefore recalculation of the header fields is not required. Figures 7 and 8 show the data length vs. average packet transmit results for this data only test case and for the UDP message case.

.

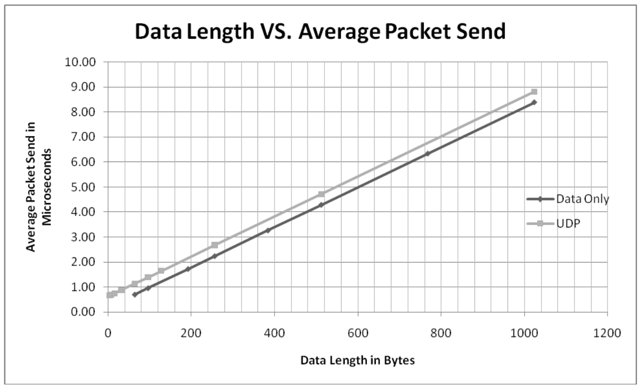

Figure 7. Data only versus UDP average send time

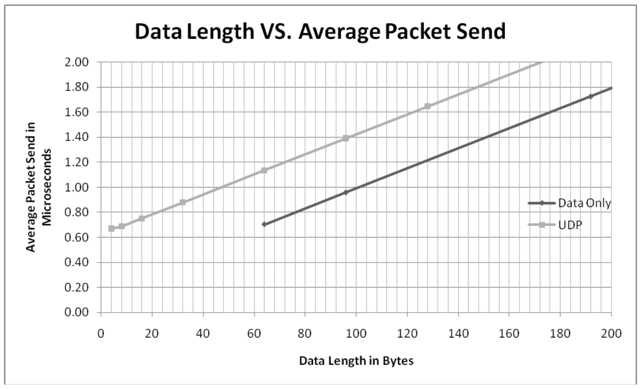

Figure 8. Data only versus UDP average send time zoomed

The minimum packet size for Ethernet is 64 bytes. Attempting to send smaller packets will result in trailing zeros to meet length requirements, so smaller packet sizes are not tested. In this case we see the graph is linear, with average transmit times ranging from 0.7 to 8.8 micro seconds. Table 3 shows the longest send times for packets of various lengths. The longest packet send field is the send time we can guarantee while conforming to the Ethernet standard.

Table 3. PC FPGA packet send times for data messages

Number of data messages sent out	Data Field Length	Total Transmit time in clock periods	Total Transmit time (mS)	Average Time/Packet (µS)	Longest Packet send in Clock Periods	Longest Packet Send (µS)
80000	64	7034664	56.28	0.70	91	0.74
80000	96	9595229	76.76	0.96	123	0.99
80000	192	17275926	138.21	1.73	219	1.76
80000	256	22396194	179.17	2.24	283	2.27
80000	384	32636576	261.09	3.26	411	3.30
80000	512	42876903	343.02	4.29	539	4.32
80000	768	63357446	506.86	6.34	795	6.37
80000	1024	83837747	670.70	8.38	1051	8.42

When comparing the results from the UDP messages versus nonstandard messages for larger packets, there is not a significant amount of overhead from the header fields. When comparing the smaller packets we see a significant decrease in send times. For instance at 64 bytes of data the average data send time is 50% less than with standard UDP messages. Table 4 tabulates the results for easy comparison.

Table 4. PC FPGA message send times for data packets

Data Field Length	Non Standard Packet Longest Send (uS)	UDP Packet Longest Send (uS)	Percent Difference
64	0.736	1.136	54.3
96	0.992	1.392	40.3
256	2.272	2.672	17.6
512	4.32	4.72	9.3
1024	8.416	8.816	4.8

5. CONCLUSIONS

The use of Giga-bit Ethernet in simulation systems for FPGA-to-FPGA communication has been demonstrated with communication at microsecond rates. This system demonstrated that Ethernet communication can be fast, efficient, predictable and reliable for point-to-point communication links using the UDP/IP standard protocols. Ethernet communication has the advantage of using standard, readily available hardware and can be monitored and tested using PCs with freely available network monitoring software. The difficulty of implementing direct Ethernet communication on an FPGA can be kept to a minimum with UDP based communication where only 3 fields in the header need to be changed between packet sends. Further speed improvements can be made for FPGA-to-FPGA communication using non-standard messages and results in more efficient transmissions, but can only be used for directly connected links because of the lack of the Ethernet header.

Acknowledgements

The authors wish to thank the Office of Naval Research for the funding that supports this research. They would also like to thank Wilfred Gonzales, a former masters student, who began the work with direct Ethernet communication. They also recognize the valuable contributions by Dr. Roger Dougal and the VTB team at the University of South Carolina.

.

References

[1] Word, D., J.J. Zenor, and R. Powelson, "Using FPGAs for Ultra-High-Speed Real-Time Simulation" *Proceedings of Conference on Grand Challenges in Modeling and Simulation*, Edinburgh, Scotland, June 16-19, 2008.

[2] Crosbie, R. E., "Low-Cost, High-Speed, Real-Time Simulation for Electric Ship Power Systems", *IEEE Electric Ship Technologies Symposium,* Philadelphia, PA July 25-27, 2005.

[3] Dougal, R. A., Liu, S. Gao, L., and Blackwelder, M. *Virtual Test Bed for Advanced Power Sources,* J. of Power Sources, Vol. 110, No. 2, pp. 285-294, 09/02.

[4] Xilinx Inc. Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC Wrapper v1.5, Getting Started Guide, September 2008.

[5] Cisco Press, *Internetworking Technologies Handbook, Third Edition*, Cisco Press, Indianapolis, IN, pp. 106-107

[6] Xilinx Inc. Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC User Guide, UG194 (v1.9). October 2009. pp. 55.

[7] Cunningham, D. G., Lane, W. G., *Gigabit Ethernet Networking*, Macmillan Technical Publishing, Indianapolis, IN, pp. 128.

Biographies

Jowel Yusta received his BS in Electircal Engineering at Chico State University, CA in 2008. He is currently a part time instructor at Chico State University where he teaches circuits and applied mathematics courses while finishing his Masters in Computer and Electrical Engineering.

John Zenor received a PhD in Geophysical Engineering from the University of Missouri at Rolla, Mo. in 1968, MS in Computer Science, 1966, B.S. in Applied Mathematics in 1963. He spent twenty years at the Naval Air Warfare Center, China Lake, CA in the areas of real-time simulation, software engineering, and tool development and twenty-five years at California State University, Chico, CA where he is currently Professor Emeritus in the Department of Electrical and Computer Engineering specializing in real-time computing and computer networking.

Kurtis Kredo II received a PhD in Electrical and Computer Engineering from the University of California at Davis in 2010. In 2004, he received a MS in Electrical Engineering and a BS in Computer Engineering from the California Polytechnic State University, San Luis Obispo. He is currently an Assistant Professor at California State University, Chico specializing in wireless networking.