To appear: IEEE Transactions on Education, 2008

NetFPGA – An Open Platform for Teaching How to Build Gigabit-rate Network Switches and Routers

Glen Gibb, John W. Lockwood, Jad Naous, Paul Hartke, and Nick McKeown

Abstract—**The NetFPGA platform enables students and researchers to build high-performance networking systems using Field Programmable Gate Array (FPGA) hardware. A new version of the NetFPGA platform has been developed and is available for use by the academic community. The NetFPGA platform has modular interfaces that enable development of complex hardware designs by integration of simple building blocks. FPGA logic is used to implement the core data processing functions while software running on an attached host computer or embedded cores within the device implement control functions. Reference designs and component libraries have been developed for the CS344 course at Stanford University and an open-source Verilog reference design is available for download from the project website.**

Index Terms—Field programmable gate arrays, Internet, networks, protocols, routing, switches.

I. INTRODUCTION

High performance network switches and routers enabled the rapid growth of the Internet. Gigabit Ethernet switches are widely deployed to interconnect computers in Local Area Networks (LANs). Multi-Gigabit/second links are used to transport Internet Protocol (IP) packets across Wide Area Networks (WANs). At most universities, students only learn to build networking systems with software. Students who take a hands-on course in computer networking write software programs that send and receive packets through user-space sockets. Students who take advanced courses write software for the kernel that interfaces directly with a Linux operating system. While systems implemented with

software may be able to send and receive some of the packets to and from a Gigabit/second Ethernet line card, software alone is not suitable for switching, routing, and processing all of the traffic that that appears on high-speed networks.

Successful commercial network vendors use Application Specific Integrated Circuits (ASICs) and/or Field Programmable Gate Arrays (FPGAs) to accelerate the switching, routing, and processing of packet data. To be competitive, students need to understand how these hardware-accelerated systems operate. Using the NetFPGA platform, students can build and prototype their own hardware-accelerated networking systems. Using the NetFPGA, traffic on four Gigabit Ethernet links can be processed at the full Gigabit/second rate. By processing data with hardware, back-to-back packets can be scanned at Gigabit rates [1] [2].

For teachers of hardware or networking classes, a set of reusable infrastructure and courseware has been developed. By starting with this infrastructure, students need not build systems from scratch. Gateware and software provided with the NetFPGA enables the platform to receive and transmit packets, buffer data in memory, and communicate with the host processor. The courseware, consisting of web pages, homework assignments, machine problems, and presentation slides, provides instructors with the resources to teach students how to build an Internet router.

The NetFPGA platform fits into any standard desktop or rack-mount computer. The hardware augments a PC with an expansion card that has four ports of Gigabit Ethernet

and attaches to the Peripheral Communication Interconnect (PCI) bus. The FPGA on the platform directly handles all data-path switching, routing, and processing of Ethernet and Internet packets, leaving software to handle only control-path functions. A photograph of a PC with the NetFPGA installed is shown in Fig. 1.

Fig. 1: Photograph of a NetFPGA installed in a Desktop PC.

II. THE NETFPGA PLATFORM

At the center of the NetFPGA platform is a Xilinx FPGA device. Surrounding the FPGA are four memory devices—two Static RAMs (SRAMs) and two second-generation Double Date Rate (DDR2) SDRAM devices. On the left side of the platform, a quad-port physical-layer transceiver (PHY) is provided that enables the platform to send and receive packets over four standard twisted-pair Ethernet cables. On the right side of the board, two Serial ATA (SATA) connectors on the platform allow multiple NetFPGAs within a system to exchange data at high speeds without using the PCI bus. A photograph of the NetFPGA 2.1 platform is shown in Fig. 2.

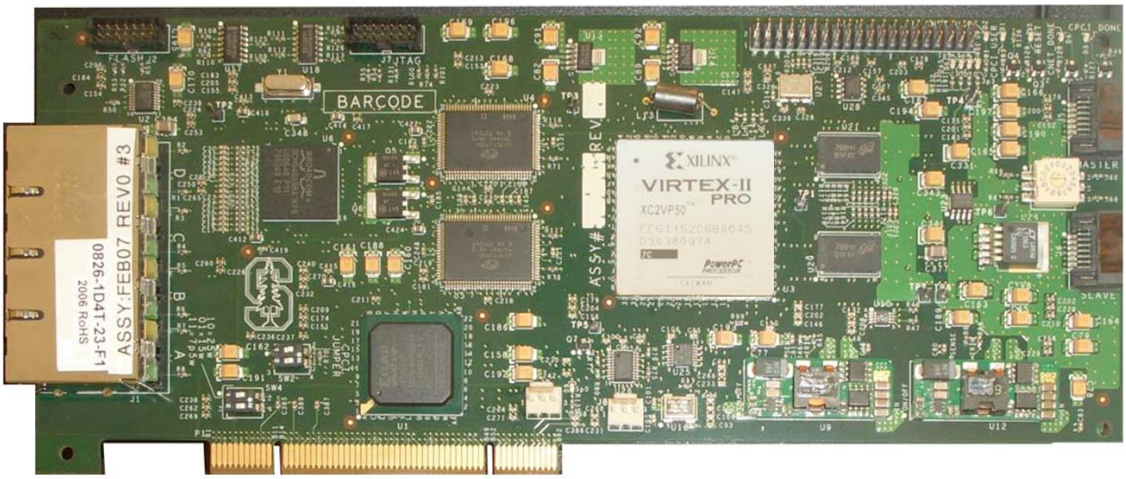

Fig. 2: Photograph of the NetFPGA Platform

A. Components of NetFPGA

A detailed block diagram that shows the major components of the NetFPGA platform is shown in Fig. 3. The NetFPGA 2.1 platform contains one large Xilinx Virtex2-Pro 50 FPGA, which is programmed with user-defined logic. The core clock on this device can be configured by the user to run at either 125 MHz or 62.5 MHz. The NetFPGA platform

also contains one small Xilinx Spartan II FPGA that implements the control logic for the PCI interface to the host processor.

Two external Cypress SRAMs are arranged in a configuration of 512k words by 36 bits (18 Mb / 4.5 MB total) and operate synchronously with the FPGA core logic at either 62.5 MHz or 125 MHz. One bank of external Micron DDR2 SDRAM is arranged in a configuration of 16M words by 32 bits (64 MB total). Using both edges of a separate 200 MHz clock, the memory has a bandwidth of 400 MWords/second (1,600 MB/s = 12,800 Mb/s).

The Broadcom quad-port gigabit physical-layer transceiver (PHY) sends packets over standard category 5, 5e, or 6 twisted-pair cables. The PHY interfaces with four Gigabit Ethernet Media Access Controllers (MACs) instantiated as soft cores within the FPGA. The NetFPGA also includes two multi-gigabit interfaces with Serial ATA (SATA) connectors that enable multiple NetFPGA boards in a system to exchange traffic directly without use of the PCI bus.

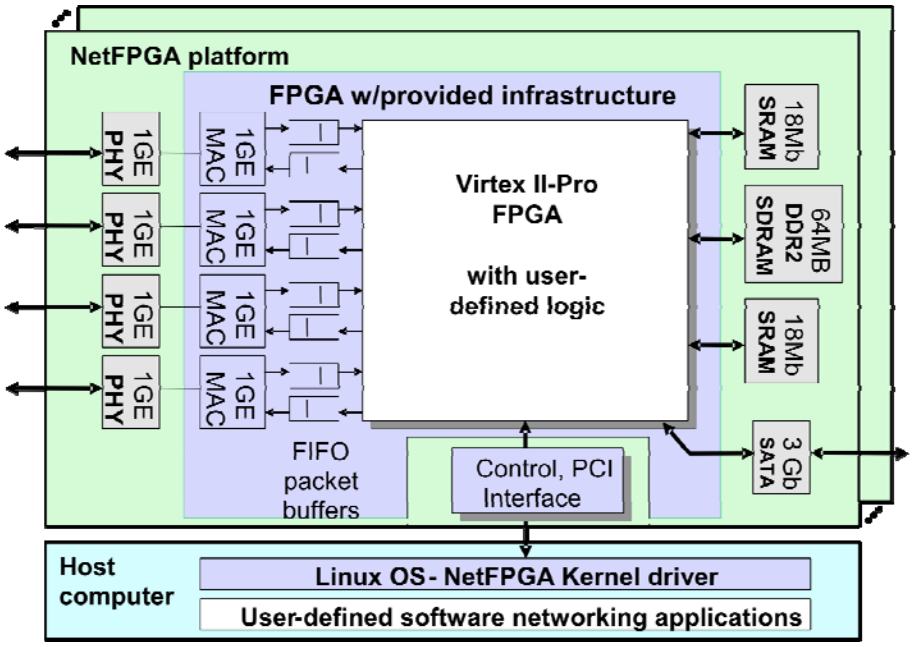

Fig. 3: Block Diagram of the NetFPGA

B. Computer Aided Design Tool Flow

A number of Computer Aided Design (CAD) tools are required to simulate and implement designs for the NetFPGA. Designs are simulated using Mentor Graphics ModelSim to verify correct logical operation. Verilog source code is synthesized using either Xilinx ISE or Synplicity's Symplify Pro. Logic is mapped, placed and routed using Xilinx ISE backend tools. Finally the resultant bitfile is programmed into the FPGA using a command-line program that runs on the host PC. Debugging of hardware circuits can be performed via the JTAG port using Synplicity Identify or Xilinx ChipScope on-chip logic analyzers.

C. Interface to Software

A Linux kernel device driver allows user space software on the host PC to send packets and receive packets via the Ethernet ports on NetFPGA board. The driver also allows application programs to read or write the memory-mapped registers that correspond to a

design's on-chip registers, SRAM, and DRAM. In total, 128 MBytes of host memory address space is reserved for each NetFPGA board installed in a host computer.

The NetFPGA device driver is a Loadable Kernel Module (LKM) that should be compatible with any Linux 2.6 kernel. Loadable kernel modules allow drivers to be installed in the kernel without requiring a full kernel recompilation. The driver provides four network interfaces named nf2c0 .. nf2c3 to userspace.

User-space programs send and receive network packets to the NetFPGA via the Linux networking stack using a standard socket interface. Programs initiate communication with the NetFPGA using standard `socket()`and `bind()` functions and then use the `connect()` or `listen()` functions to open a channel of communication with the NetFPGA. Programs then use standard functions such as `read(), write(), send()` and `recv()` to send and receive data to/from the NetFPGA.

DMA is used to transfer complete Ethernet frames between the host and NetFPGA—these DMA transfers are initiated by the driver but performed by logic in the NetFPGA. This allows the host to perform concurrent operations while the I/O is in progress.

User-space programs can perform memory-mapped reads and writes via Input/Output Control commands (`ioctl`). An ioctl command performs a single word transfer to or from the NetFPGA. These calls provide a convenient method to access registers and memory on the board.

D. Availability of Hardware

NetFPGAs are available to the community directly from a third-party vendor who manufactures the hardware. This company, Digilent Incorporated, has a long history of making low-cost evaluation boards through their partnerships with the Xilinx University Program (XUP).

III. STANFORD'S CS344 ROUTER DESIGN COURSE

Stanford University's CS344 course is a graduate-level, project-based course that teaches Ph.D., masters, and advanced undergraduate students how to build an Internet router in just eight weeks [3]. Students who take the class learn how to develop both the hardware and software components for a high-speed router. Each team consists of one student to work on hardware and one to two students to write router control software.

Using the NetFPGA, student teams build a fully functional, Gigabit/second Ethernet switch and router. Together, teams interconnect multiple NetFPGA platforms in a network that routes packets over several Gigabit Ethernet links. The topology of one such network implemented with five NetFPGA routers is shown in Fig. 4. Forwarding tables are implemented in hardware, and use values populated by the routing software to forward packets along the shortest path between machines.

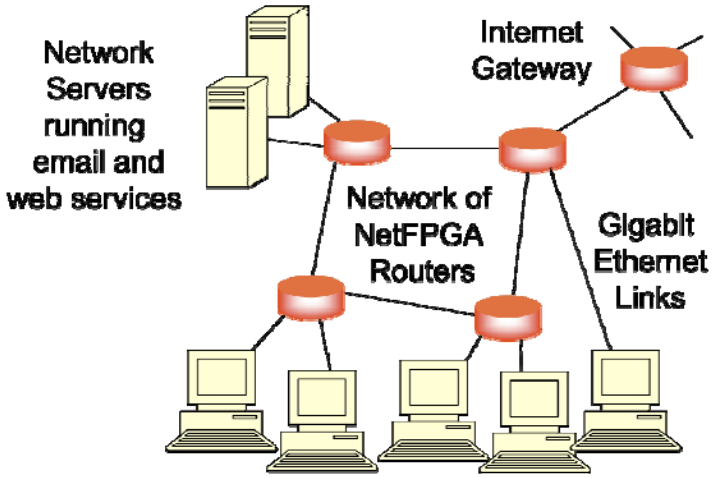

Fig. 4: Sample Topology of a Network with Five NetFPGA Routers

A. Student Background

Students undertaking the hardware design task are expected to enter the class with basic hardware design experience. The assumption is made that these students already understand the basic concepts of synchronous logic and hardware pipelining. Most computer engineering students entering CS344 have already coded at least a small project using Verilog.

Students undertaking the software design task are expected to have taken a networking class. They are assumed to have already been exposed to the concepts of network stacks and socket programming. Students should also be proficient in programming in C.

Hardware students are also encouraged to have taken a networking course prior to enrolling in this course, so as to have a basic understanding of addressing and routing.

Students without this background are expected to do background reading during the first few weeks of the course.

B. Course structure

This course is conducted as a quarter-long (10 week) course at Stanford University. Students taking the course typically take two other classes during the quarter, or are working on a part-time research project. Students are expected to allocate 10 to 20 hours per week to complete the class assignments. The course is structured with periodic assignments that have regular deliverables. A short lecture is given before each assignment to describe the objectives and milestones. Teams meet with the teaching staff on a weekly basis. The teaching staff provides guidance to each team and monitors progress to ensure that students don't fall behind on the assignments.

At the start of the course, students are provided with the NetFPGA hardware platform, Verilog HDL code that implements an Ethernet repeater, and skeleton software code that provides a framework for implementing the router's control path. The hardware and software members of each team work independently during the early stages of the course except for meetings where they discuss the interfaces between hardware and software.

In the first assignment, the hardware members of each team extend the design of the given two-port Ethernet repeater to implement a four-port learning Ethernet switch. This assignment provides the hardware students with an opportunity to familiarize themselves

with the CAD tools, the NetFPGA platform, and the format of fields within an Ethernet packet.

In the next assignment, hardware students replace the switching logic with routing logic that forwards packets using the Internet Protocol (IP) address in the header. The students compare the destination IP address in the packet to entries stored in the software-generated routing table to find a longest matching prefix. Based on which entry matches, they next look up the next-hop using the Address Resolution Protocol (ARP) and update the destination address field in the Ethernet header.

While some team members built hardware, the rest wrote software to implement the control path. Students use the Virtual Network System (VNS) in order to test control programs independently of the hardware [4]. The control path software includes a routing algorithm based upon OSPF, a command-line interface, and the logic to route packets in software (necessary when the hardware forwards packets to software, such as in the case of an ARP table miss). Also included in the implementation is support for ARP and ICMP protocols.

At the mid-point of the course, the hardware and software members integrate their designs. The hardware and software members test their designs together to ensure that the hardware and software components properly communicate. Teams test the interoperability of their routers with one another by interconnecting their routers and verifying that PW-OSPF properly exchanges routes between the machines.

In the final few weeks of the course, student groups each implement an advanced feature for their router. Students choose their project and discuss their plans with the instructor. Examples of features implemented by students during the Spring 2007 quarter include network address translation (NAT) module, a multicast router, access control lists (ACLs), and DES encryption.

C. Hardware design

A core feature of the NetFPGA platform is a modular design that is easy to understand and modify. The class and reference material encourage students to carry that modularity through to their resultant designs.

The starter code is structured around the following set of components: the user data path, the Ethernet I/O blocks (encapsulating the MAC and clock domain crossing logic), and the host interface logic (enabling communication and packet transfer between the host and NetFPGA). Students add logic to perform routing or switching to the user data path.

The user data path is implemented as a pipeline, using multiple modules. An arbiter reads packets from the Ethernet or from the host interface then feeds them through the pipeline. Because packets are multiplexed together, no crossbar is needed within the data path. Data is de-multiplexed as packets are written to output queues. Peak bandwidth through the user data path is 8 Gb/s in the FPGA fabric. A block diagram of the data path is shown in Fig. 5.

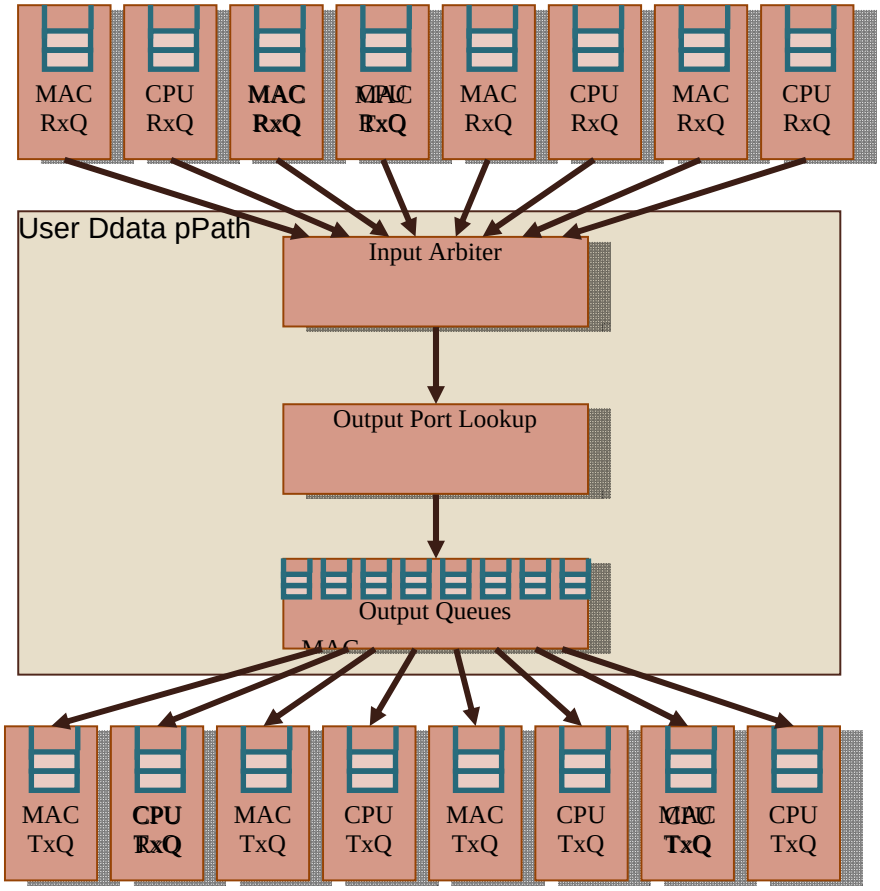

Fig. 5: NetFPGA's Router Data Path

The main components implemented by the students within the user data path were the input arbiter and an output port lookup module. The output port lookup module was built as a set of multiple sub-modules that holds the ARP and routing tables, performs LPM on the routing table, performs next-hop lookups on the ARP table, verifies the IP header, and updates the necessary fields in the packet. The output queues and the modules, except for the user data path, were provided to the students as part of the starter code.

D. Hardware testing

Students first test their designs in simulation. The NetFPGA test environment uses Perl scripts to instruct the simulator to inject packets into the PHY, perform DMA transfers,

read packets from the PHY, monitor a DMA transfer, and read/write registers through the PCI interface. Cycle-accurate simulation allows basic design errors to be identified and corrected prior to synthesis. Once the hardware design appeared to be functional, students synthesized their circuits using the Xilinx ISE CAD tools and downloaded the resulting bitfile into the Virtex FPGA on the NetPGA. Simulation worked well for testing small volumes of traffic, but was found to be far too slow for testing large volumes of traffic.

Students were given access to PCs that allowed them to test their designs in real hardware at full Gigabit line rates. Each PC had both a NetFPGA card and a quad-port gigabit Network Interface Controller (NIC) installed. Initially, each port of the NIC was connected to a port on the NetFPGA. Students ran utilities such as ping, traceroute, iperf, and Wireshark to inject and monitor real traffic to and from the NetFPGA.

At a larger volume of traffic, students found problems with their hardware that they had not observed in simulation. These problems were generally caused by corner cases not covered in simulation. To debug the hardware, signals on the FPGA were analyzed using Xilinx Chipscope, an on-chip logic analyzer. Chipscope allowed the students to monitor and capture the sequence of events that caused a packet to be dropped or corrupted.

Later in the course, students interconnected different machines to created network topologies with multiple routers. In these networks, students performed experiments to ensure that multi-hop routes were created correctly. The students unplugged Ethernet

ports to verify that the routers responded properly to link failures and were able to re-route packets along the next-best path. Multiple groups tested their routers together to verify that the routers were interoperable.

E. Student feedback

Students were asked to provide informal feedback at the conclusion of the class. Common feedback received indicated that the class was interesting, personally satisfying and quite challenging. Students also indicated that the workload was relatively high. One student wrote that *"Overall it was awesome. I came in without knowing much about networks or routers, and I left with a wealth of information. It must sound weird but I truly liked everything since I have never seen this stuff in such detail beforehand."* Another student wrote: *"It was one of the most satisfying, interesting, and time consuming courses I've taken at Stanford."*

The two most common issues reported by the hardware students were (1) problems meeting timing and (2) problems identifying elusive bugs. Work has been undertaken to reduce the resource requirements of the starter code to allow more room for student code, thus making it easier for the place and route tools to meet timing. The timing issues can also be significantly eased by targeting to the 62.5 MHz clock. Additional steps have also been undertaken to further simplify the process of writing simulation tests to aid in identifying more bugs before testing in hardware.

IV. NETFPGA INFRASTRUCTURE

The NetFPGA is an open platform that can be used at other universities to teach courses like CS344. The components that implement the functionality to switch and route packets can be reused to teach courses like CS344 at other institutions.

Most NetFPGA designs include both hardware components and software components. For example, with the four-port router, all of the forwarding functions are implemented as hardware components using gateware developed in Verilog, while the software components that implement and control the PW-OSPF is written in C, C++, and Java.

Multiple circuits have been developed for the NetFPGA that include: a four-port Network Interface Card (NIC), a four-port learning Ethernet switch, and a four-port IP Version 4 (IPv4) router. The router not only forwards packets, but also negotiates routes using a simplified version of the Open Shortest Path First (OSPF) protocol called Pee Wee OSPF (PW-OSPF).

NetFPGA infrastructure was designed to be capable of handling packets at Gigabit/second line rates. The modules provide a high degree of visibility into the operation of the device through use of memory-mapped registers that can be accessed by the CPU on the host. These registers are used to configure the mode of the hardware, set the sizes of buffers, and monitor the progress of packets through the pipeline.

A. The Reference Router Gateware

The reference hardware design gateware consists of a modular pipeline with multiple components. Components are modular and can be reused in other projects that build upon the function of the base router. By starting with a working reference design, the barrier to entry for implementing complex network designs is lowered for students and researchers, who are mainly interested in adding a feature rather than rebuilding a complete network system from scratch. The modular design allows the components to be easily shared thus facilitating the development of a community-supported project.

B. Student Design Projects

Once the students completed assignments to implement the Ethernet switch and Internet Router, they were given the opportunity to implement their own advanced feature that built upon the reference router during the last two weeks of the course. The NetFPGA allows packets to be created, modified, or dropped. Packets can be sent to and from the Ethernet ports and the host processor. Packets can also be forwarded between any of the four physical ports on the platform.

During the Spring 2007 quarter of CS344, four advanced features from the 2007 class were network address translation (NAT), time synchronization, multicast and packet capture. All four features were implemented within a two-week period.

The first team implemented NAT to allow multiple hosts to share a single IP address. The NAT device translated packet headers between a publicly-accessible IP address and a private address space for hosts behind the NAT device. The team added an extra look-up to identify the IP address/port translation to apply. Packets without a match in the hardware table were forwarded to software for further processing. Frequently-used translations were stored in a small table in hardware, while a much larger table of infrequently-used translations was stored in software.

The second team implemented a simplification of the IEEE 1588 protocol. This protocol provides a mechanism for precision time synchronization, on the order of tens of nanoseconds, of devices over Ethernet. Implementation required deterministic time stamping of packets at arrival/departure. No elastic buffers could be used between the wire and the point of time stamping. The team successfully demonstrated synchronization between multiple routers by showing sub-microsecond time synchronization.

The third team implemented a multicast circuit that allowed a single packet to be copied to multiple destinations using hardware. The team processed join and leave messages to add and remove the host from a multicast group table in hardware.

The fourth team built a circuit to capture packets transiting through the router and forward those packets to the host. The implementation compiled Berkeley Packet Filters (BPF) into a form suitable for the hardware. The hardware contained a set of circuits that matched parts of a packet simultaneously.

C. Presentation of Results of Advance Features

A photo of the CS344 final project presentations on June 5, 2007 is shown in Fig. 6. Students presented their projects to the class and to a panel of judges from industry, who were invited to evaluate the results of the student projects.

Fig. 6: Photo from the CS344 Final Presentations in Spring 2007

V. RELATED WORK

Other platforms have been used to implement some of the functionality that is provided by the NetFPGA platform. Similar to the NetFPGA, the RiceNIC utilized a Virtex2-Pro FPGA to implement the function of a network interface card [5]. Modules on the Field Programmable Port Extender (FPX) platform have also been used in the classroom to implement Internet routing functions [6] [7].

Several research projects have used the NetFPGA platform to implement customized packet switches with advanced features. With Ethane, the NetFPGA was used to provide fine-grained control on how packets were switched in a local area network [8]. The

NetFPGA has also been integrated into the DETER testbed where it was used to implement intrusion prevention functions [9].

ACKNOWLEDGEMENTS

The authors wish to acknowledge the contributions of Greg Watson, Jim Weaver, and Ramanan Raghuraman who contributed to earlier versions of the NetFPGA platform. Jianying Luo contributed to the DMA.

This research is supported by gifts from Xilinx, Cisco, Huawei, Agilent, Micron, Cypress, Broadcom, Digilent, and Synplicity as well as National Science Foundation grants: NSF EIA-0305729 and CNS-0551703. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or other sponsors of this project.

REFERENCES

[1] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman and J. Luo, "An Open Platform for Gigabit-rate Network Switching and Routing," in *Proc. Int'l Conf. Microelectronic System Education (MSE'2007)*, San Diego, CA, Jun. 2007, pp. 160-161.

[2] G. Watson, N. McKeown and M. Casado, "NetFPGA: A tool for network research and education," presented at the Workshop on Architecture Research using FPGA Platforms (WARFP), Austin, TX, Feb. 2006.

[3] "Build an Internet Router", Stanford University Computer Science 344, May 2007. On-line as: http://cs344.stanford.edu

[4] M. Casado and N. McKeown, "The Virtual Network System", in *Proc. 36th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, MO, 2005, pp. 76-80.

[5] J. Shafer and S. Rixner, "A Reconfigurable and Programmable Gigabit Ethernet Network Interface Card", Rice University, Department of Electrical and Computer Engineering, Houston, TX, Technical Report: TREE0611, December, 2006.

[6] J. W. Lockwood, "Platform and Methodology for Teaching Design of Hardware Modules in Internet Routers and Firewalls," in *Proc. Int'l Conf. Microelectronic System Education (MSE)*, Las Vegas, NV, July 2001, pp. 56-57.

[7] J. W. Lockwood, C. Neely, C. Zuver and D. Lim, "Automated Tools to Implement and Test Internet Systems in Reconfigurable Hardware," SIGCOMM Computer Communications Review (CCR), vol. 33, no. 3, pp. 103-110, July 2003, pp 103-110.

[8] J. Luo, J. Pettit, M. Casado, N. McKeown and J. Lockwood, "Prototyping Fast, Simple, Secure Switches for Ethane," in *Proc. Hot Interconnects 15*, Stanford, CA, August 2007, pp 73-82.

[9] N. Weaver and V. Paxson, "Stress-Testing a Gbps Intrusion Prevention Device on DETER", Meeting Presentation, Jun. 2006 [Online]. Available: http://www.isi.edu/deter/

BIOGRAPHIES

Glen Gibb received the M.S. degree from Stanford University in 2005 and he is currently pursuing a PhD candidate in the department of Electrical Engineering. His research interests include switch and router design and implementation, FPGA architectures and digital logic implementation. He has been an IEEE member since 2004.

John W Lockwood (PhD/MS/BS UIUC 95/93/91) leads the Alpha and Beta release program and organizes the worldwide NetFPGA tutorial program and is currently a Consulting Associate Professor in the department of Electrical Engineering at Stanford University. Prior to joining Stanford, Lockwood worked at Washington University and St. Louis and was granted tenure in the Department of Computer Science in 2006. At Washington University, Lockwood led the Reconfigurable Network Group (RNG) and develop the Field programmable Port Extender (FPX) to enable rapid prototype of extensible network modules in Field Programmable Gate Array (FPGA) hardware. John Lockwood has served as the principal investigator on grants from the National Science Foundation, Xilinx, Altera, Nortel Networks, Rockwell Collins, and Boeing. He has worked in industry for AT&T Bell Laboratories, IBM, Science Applications International Corporation (SAIC), and the National Center for Supercomputing Applications (NCSA) and served as co-founded of Global Velocity, a networking company focused on high-speed data security. He is a member of ACM, Tau Beta Pi, Eta Kappa Nu and has been a member of IEEE since 1988. Lockwood's research interests include prototype of reconfigurable hardware, implementation of systems that provide Internet security, and content processing technologies to enable development of new types of networks.

Jad Naous received his B.Eng. degree in Computer Engineering from McGill University in 2005, and his M.S.E.E. degree from Stanford University in 2007. He is currently pursuing a PhD in Electrical Engineering at Stanford University. He has previously worked as a Graduate Intern for Sun Microsystems Labs in 2006, where he worked on the next generation switch project. In 2007, he joined Agilent Technologies Labs as a Graduate Intern where he helped implement special devices for the IEEE1588 Precision Time Protocol using NetFPGA.

Paul Hartke is a graduate student at Stanford University and has served as a Teaching Fellow Instructor for EE183: Advanced Logic Design Laboratory and EE108A: Digital Systems I. Paul currently works at Xilinx on networking solutions based on the Virtex II Pro FPGA family and had previously worked as an intern for Archway Digital Solutions, Abrizio, and Hewlett-Packard Laboratories.

Nick McKeown (PhD/MS UC Berkeley '95/'92; B.E Univ. of Leeds, '86) is a Professor of Electrical Engineering and Computer Science, and Faculty Director of the Clean Slate Program at Stanford University. From 1986-1989 he worked for Hewlett-Packard Labs in Bristol, England. In 1995, he helped architect Cisco's GSR 12000 router. In 1997 Nick co-founded Abrizio Inc. (acquired by PMC-Sierra), where he was CTO. He was co-founder and CEO of Nemo Systems, which is now part of Cisco Systems. Nick McKeown is the STMicroelectronics Faculty Scholar, the Robert Noyce Faculty Fellow, a Fellow of the Powell Foundation and the Alfred P. Sloan Foundation, and recipient of a CAREER award from the National Science Foundation. In 2000, he received the IEEE Rice Award for the best paper in communications theory. Nick is a Fellow of the Royal Academy of Engineering (UK), the IEEE and the ACM. In 2005, he was awarded the British Computer Society Lovelace Medal. Nick's research interests include the architecture of the future Internet, tools and platforms for networking teaching and research.