NetFPGA: A Tool for Network Research and Education

Greg Watson, Nick McKeown and Martin Casado
Department of Electrical Engineering
Stanford University
Stanford, CA 94305-9030
{gwatson, nickm, casado}@stanford.edu

***Abstract*—NetFPGA is a platform that allows students to build real networking hardware, using industry-standard design tools (e.g. Verilog), then deploy and debug their hardware in an operational network. In the canonical classroom design exercise, a student builds an Ethernet switch, or an Internet router and makes it interoperate with other students' solutions. NetFPGA-v1 has been used in classes at Stanford for several years, and is just being replaced by NetFPGA-v2, which has four Gigabit Ethernet interfaces. NetFPGA-v2 is designed for teaching and research; it is open, easy to use, and simple enough to give away for free.**

I. Introduction

The NetFPGA project is motivated by a need for tools to teach computer network systems at the undergraduate and graduate level. Specifically we observed that students only gain practical experience with networking at layers three and above (routing protocols, transport protocols, etc.) Students' exposure to the Link and Physical layers is limited to in-class teaching – they don't get to build network systems. There are many reasons for this, some of the obvious ones being:

- the inherent complexity of building custom hardware systems,
- the practical reality that interesting network systems need multiple devices before they become interesting, and this has inherent cost.
- the observation that digital design classes focus more on microprocessors than networks,

We believe that a custom platform will address the cost and complexity arguments. By developing interesting network-based project classes that are easy to use we believe we can get network systems design adopted more widely in digital design classes. We set out to build a platform that students could use to build interesting and realistic network systems (NICs, switches, routers) within a 10-week class. Our success with the tool has encouraged us and others to expand its use into research.

NetFPGA has gone through two major design versions. This paper starts with the first version and explains how it was used in one particular class at Stanford. The limitations of this first version are noted, and lead into the description of the second version which we are just starting to use. We conclude with the current status and a mention of some other groups who are using NetFPGA for their research.

This work was funded by the NSF, under grant 02-082 and grant EIA-0305729

Martin Casado is funded by the DHS Scholarship and Fellowship Program.

II. NetFPGA Version 1

The first version of NetFPGA was started in late 2001 and the first prototypes were used in a post-graduate level project class at Stanford in 2003. Our original vision was for a rack of NetFPGA-v1 boards located somewhere in Stanford. Students would not have physical access to the boards, but rather they would download and debug their designs remotely. We chose to provide only remote access because the long term goal was for Stanford to host the boards and make them available to other institutions, which would preclude physical access.

A. The NetFPGA-v1 system

The NetFPGA-v1 board itself measures six inches by nine inches and contains three Altera EP20K400 APEX devices, an eight-port Ethernet controller, three 1MByte SRAMs and ancillary logic. There is no on-board CPU. One of the three FPGAs, called the Control FPGA (CFPGA) is pre-programmed and connects the Ethernet controller to the two User FPGAs (UFPGAs). All communication with the board is via the Ethernet ports - the only other physical connectors are for power and reset.

Students develop their designs using Synopsys tools for synthesis and simulation, and the Altera Quartus tool for place-and-route. We provide some scripts and libraries to simplify the development and debug process.

When their design has been simulated and synthesized, students upload the FGPA configuration file to a NetFPGA-v1 board via a simple web interface. The students then send packets to their board, and capture packets received from their board, in order to verify the hardware functionality. Ultimately the students interconnect their board to the campus network and their design then carries regular internet traffic. We used a locally developed tool, called the Virtual Network System [1], to map the NetFPGA-v1 ports into the campus internet.

In most FPGA-based projects we have seen, including those at Stanford, there is an emphasis on hands-on debugging of the hardware using logic analyzers. For NetFPGA-v1 we consider this a last resort – students are strongly encouraged to simulate first and debug last. The hands-off nature of the NetFPGA-v1

system means that it is not practical to use external logic analyzers (though this would be feasible, albeit expensive, with network-based analyzers). We considered using the FPGA vendor's analysis tool, but that required local CPUs with serial ports attached to each board. Instead we developed our own simple logic analzyer which can be downloaded to the NetFPGA-v1 boards as part of the system. Students then set trigger conditions via a simple GUI-based application, capture the data on the board itself, and display the signals in a conventional waveform viewer.

A CPU is an essential aspect of most modern digital systems. Although NetFPGA-v1 has no CPU, students write software to control their board and this software can run on any computer connected to the internet, thanks to the VNS system. The software can access hardware registers using special Ethernet packets which are decoded by the CFPGA and translated into register read/write transactions for the user's design. This mechanism enables students to build complex hardware/software systems.

B. Build your own IP Router

NetFPGA-v1 has been used for two years within the CS344 class at Stanford [2]. In that class students design, build, and test their own IP router. The students must also demonstrate interoperability with other students' routers, and finally they add some enhanced functionality of their own choosing.

One of the goals of teaching the design of network systems (as with almost any digital system these days) is to expose the students to the hardware-software interface. While the functionality of an IP router is very clear (RFC 1812), the students must decide whether to implement each feature in hardware or software, based on issues such as performance and complexity.

The students design a "classical" IP router - a fast path is provided by the FPGA, while a slow path is implemented in software. The fast path decodes incoming packets. If the packet is a well-formed IPv4 packet then its destination IP address is looked up in a local routing table. If the next hop is found then the MAC address of that next hop is looked up in an ARP cache. If found then the packet is forwarded (TTL decremented and new checksum calculated). If, at any stage, these tests or look-ups fail, then the packet is forwarded to the slow path – the student's control software which is running on an external computer. The software handles all ARP packets, as well as running an OSPF-like routing protocol. The software also must populate and manage the routing table and ARP cache.

Once their IP router is working then students add new functionality of their choice such as IPSEC encryption, a web server, NAT server, etc. One enterprising group used their router to demonstrate man-in-the-middle attacks on ssh connections.

C. Limitations of NetFPGA Version 1

Having used the first version of the board for several years, we identified a number of aspects that we wanted to change.

- Board Format. The PCB is an awkward format, requiring an obscure rack that is difficult to obtain and requires self-assembly. In addition, a custom backplane is needed. Clearly, an easier to use, standard format was desirable.
- Slow. The original board provides eight 10Mb/s Ethernet ports. This is a perfectly adequate rate for classes but is insufficient for many research projects.
- CPU. The lack of an on-board CPU has not prevented students from developing sophisticated software to control their boards. Indeed the flexibility of being able to run their control software on any campus computer has been very helpful thanks to the familiar development and debug flow. However, the poor performance does constrain what the software can achieve. Consequently the option of an on-board CPU would be desirable.
- Development platform. NetFPGA-v1 uses Linux and Solaris platforms for development. We would like to expand this platform support to include Microsoft Windows given its widespread adoption.

These have all been addressed in the new version of the board.

III. NETFPGA VERSION 2

In the summer of 2004 we started to address the issues listed above, and in September 2005 we received the first prototypes of the new boards.

NetFPGA Version 2 (NetFPGA-v2), is a full-length 32-bit, 33MHz PCI board shown in figure 1. It has a small Xilinx Spartan device which is programmed at power-on and provides the PCI interface. It also has a Xilinx V2P30 device which will have the user designs. The V2P30 has two 512Kx36 SRAMs. The entire system is clocked at 62.5 Mhz. The V2P30 also connects to a Marvell quad 10/100/1000 Ethernet PHY using standard GMII interfaces. Finally, two of the high-speed RocketIO serial interfaces on the V2P30 are brought out onto SATA connectors to support inter-board connections.

The main change was the move to the PCI format. We realized that version one's custom rack and backplane was a serious impediment to the adoption of NetFPGA-v1. One of our new goals was ease-of-use and so NetFPGA-v2 uses the standard PCI interface. This had two benefits: first, the format is very familiar to our intended users, and second, the greater bandwidth provides new opportunities for hardware/software systems. We briefly considered using PCI-Express which is achieving rapid market success, but we believe that PCI is so well established that it will be available for many years to come.

The PCI bus supplies power and reset, as did the connector in version one. However we also exploit the speed of PCI to program the Virtex device. At power-on the Spartan device configures itself from a flash eprom and identifies itself to the PCI bus. Software on the host PC can then configure the Virtex through the PCI interface - configuration takes about one second. A user can reconfigure the Virtex device without needing to reboot the computer.

A. Design Environment

The design environment for NetFPGA-v2 remains simple and easy to use. We still support Synopsys VCS (under Linux) for verilog simulations, but now there is also the option of Mentor's ModelSim (under Linux or XP). Many other simulators would also likely work, but have not been tested. Synthesis and

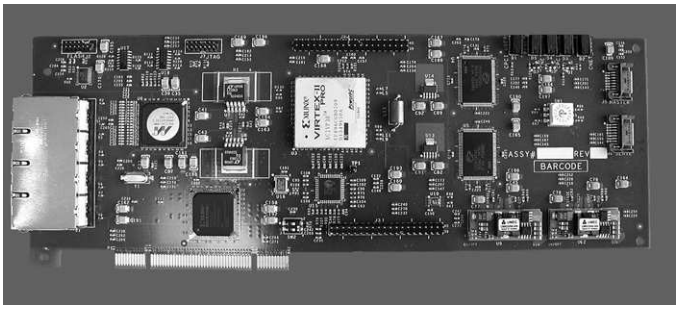

Fig. 1. The NetFPGA Version 2 Board.

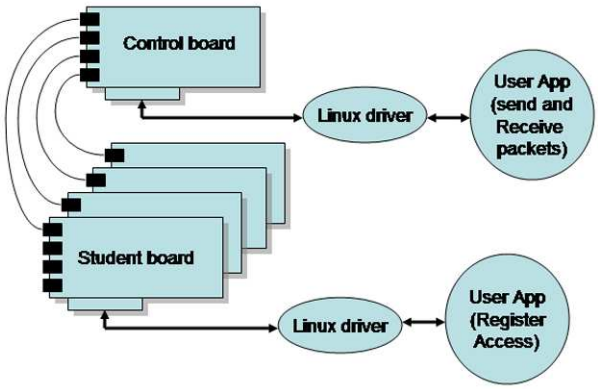

Fig. 2. Classroom configuration

place-and-route tasks are performed with the Xilinx ISE tool (Linux or XP).

As before, we want to simplify the development environment wherever it is appropriate to do so. Consequently, we supply students with a Perl module that makes it easy to specify ingress and egress packets in a system simulation. What makes this interesting is that the students can use the same test script to test their hardware. Not only does this reflect real-world verification methodology, but this encourages students to develop good verification suites, knowing that the same test can be used on the hardware. Our tools use the pcap format [3] for storing packet information, so that packets can be examined using widely available utilities such as tcpdump [4] and ethereal [5].

With NetFPGA-v2 it is practical for students to have hands-on access to the board, if desired. Our lab configuration uses several Linux servers each with five NetFPGA-v2 boards. One of the five boards acts as a 4-port Ethernet NIC, which we call the Control board. The remaining four boards are allocated one per student team. One Ethernet port on each student board is then connected directly to the Control board, as shown in figure 2. This means that the PC and the five NetFPGA-v2 boards form a complete network development environment: students can program their board and send and receive packets to/from their board without the need for any external infrastucture.

B. Using CPUs in the Virtex device

We are exploring how to use CPUs within the Virtex devices, either the two on-chip PowerPCs or the soft-core MicroBlaze processors. The Xilinx Platform Studio software provides a simple and powerful way to incorporate CPUs and their associated software tool chain such as gcc and libc. We are currently developing the board description files needed by Platform Studio so that the NetFPGA-v2 board can be used in this flow. This will enable students to have their router software, for example, run on the NetFPGA-v2 board directly - which means that their learning experience is more relevant to the way that routers are develped in industry. Although NetFPGA-v2 has only 4MB of RAM, we think that the uClinux operating system [6] will run on the board with a networking stack. A student is currently investigating how feasible (and useful) this might be.

C. Status

We currently have assembled 18 NetFPGA-v2 boards. We also have a bus-master DMA driver (Linux 2.6.13+ kernels) for the 4-port Ethernet Control board described earlier. This driver presents the NetFPGA-v2 board as four standard Ethernet interfaces. A simpler version of this driver can be used to enable student software to control their board via ioctl commands that perform register accesses. The ioctl mechanism is substantially faster than the packet-based register accesses of Version 1.

In the Winter quarter (Jan-Mar '06) we plan to use the boards in an undergraduate digital design class in which students will design and build a four port learning Ethernet switch. This is another step on our path towards having NetFPGA gain wider acceptance among the academic community (starting with our own!)

Much work remains, particularly in the area of software. We need some good diagnostic programs to help analyze configuration and hardware problems. We would like to port our logic analyzer to the new system. Our current drivers work only under a specific O/S release and so we would like to test them on a wider variety of Linux distributions.

IV. Research Applications

Our initial vision for NetFPGA was as a teaching tool in the classroom. However, the arrival of NetFPGA-v2 with its ease-of-use, higher performance, and greater capacity has started to interest researchers. Consequently we are now exploring how the system can be enhanced to make it more attractive to network researchers.

NetFPGA-v1 has already demonstrated its use as a platform for building sophisticated network systems. NetFPGA-v2 builds on this and, thanks to its greater performance and capacity, makes an excellent candidate for some of the "clean-slate" internet research that is being proposed such as the NSF GENI program [7]. In this section we describe two nascent research efforts using NetFPGA-v2.

A. RCP

The Rate Control Protocol [8] is a new protocol developed at Stanford to provide TCP sources with the optimal rate at which they should transmit. We are implementing RCP-capable routers using NetFPGA-v2 in order to understand the implementation costs of such a protocol. Without NetFPGA-v2 we would be limited to a software-only router running on a Linux box.

B. Intrusion Detection at ICSI

Nicholas Weaver, Jose Gonzalez, and Vern Paxson are currently implementing shunting, a technique to enable very high speed (1 Gbps and higher) layer 7 intrusion detection and prevention system, based on the Bro IDS [9] by providing a controllable bridge which the IDS can use to manage the traffic: allowing it to examine traffic of interest while passing "known good" traffic (such as encrypted SSH sessions) without loading the IDS. The Shunt itself will behave just like a dual-port Ethernet card, except that the Shunt can process data directly, either forwarding it onward, dropping known-malicious sources, or passing packets to the host IDS for further analysis.

The NetFPGA-v2 board is being used as the hardware prototype. They have started with the control-board configuration, and are actively extending it to make it easier for research use, including multiple output FIFOs on each MAC, additional memory interfaces to the SRAM for user logic, and a shim interface as a placeholder for user logic, which will actually implement the shunting logic. Any packets the shim can't process are passed through to the host using the existing control-board infrastructure.

V. Conclusion

NetFPGA has been in development and use at Stanford over the past four years. In that time it has been used to teach several advanced network design classes. We have recently redesigned the system to make it easier to use and more powerful, and we believe that NetFPGA-v2 is an excellent tool for research as well as the classroom. Further details are available at [10].

VI. Acknowledgements

The authors gratefully acknowledge the contributions made to this project by the NSF and also by many companies, in particular Altera, Cypress, Digilent, Synopsys and Xilinx. We are also extremely thankful to all those who have put many days of work into the system and who are listed at the websites.

References

[1] Martin Casado, Nick McKeown. The Virtual Network System *ACM SIGCSE Bulletin*, Volume 37, Pages 76 - 80, 2005
[2] Martin Casado, Gregory Watson, Nick McKeown. Teaching Networking Hardware *ACM ITiCSE*, 2005
[3] libpcap packet capture library. http://www.tcpdump.org.
[4] tcpdump network sniffer. http://www.tcpdump.org.
[5] The ethereal network analyzer. http://www.ethereal.com. *IEC DesignCon 2001*, Santa Clara, CA, Jan. 2001, Paper WB-19.
[6] The uClinux operating system. http://www.uclinux.org/
[7] The NSF GENI Program http://www.nsf.gov/cise/geni/
[8] N. Dukkipati, M. Kobayashi, R. Zhang-Shen and N. McKeown, "Processor Sharing Flows in the Internet" *Thirteenth International Workshop on Quality of Service (IWQoS), Passau, Germany, June 2005.* http://yuba.stanford.edu/ nanditad/RCP-IWQoS.pdf
[9] The Bro Intrusion Detection System http://bro-ids.org/
[10] The NetFPGA2 web site http://klamath.stanford.edu/nf2/