

Evolvable Internet Hardware Platforms

John W. Lockwood*
Department of Computer Science
Applied Research Lab
Washington University
1 Brookings Drive, Campus Box 1045
Saint Louis, MO 63130

lockwood@arl.wustl.edu
(314) 935-4460
<http://www.arl.wustl.edu/arl/projects/fpx/>

Abstract

Network routing platforms and Internet firewalls of the next decade will be radically different than the platforms of today. They will contain modular components that can be dynamically reconfigured over the Internet. But, unlike the active networks that are in the research labs today, these new platforms will not suffer from the performance penalty of processing packets in software.

These platforms will implement routing, packet filtering, and queuing functions in reprogrammable hardware. The hardware of the system will evolve over time as packet processing algorithms and protocols progress. The granularity of the system will be configurable down to the level of the logic gates. These logic gates, and the interconnections between them, will be reconfigurable over the Internet. These routers will enable new services to be rapidly deployed over the Internet and operate at the full rate of the an Internet backbone link.

Through the development of the the Field Programmable Port Extender (FPX), a platform has been built that demonstrates how networking modules can be used for rapid prototype and deployment of networking hardware. The platform includes high-speed network interfaces, multiple banks of memory, and Field Programmable Gate Array (FPGA) logic. Applications have been developed for the FPX that include Internet packet routing, data queuing, and application-level data modification. The FPX is currently used as a component in an evolvable router.

1 Introduction

The Internet evolves as new protocols, features, and capabilities are added to the routers that implement the underlying network. New protocols, like IP version 6, allow far more devices to be individually addressed on the Internet. New features, like per-flow queuing, allow voice and video to be reliably delivered in real time over the Internet. New capabilities, like firewalls, enhance the security of the Internet. A major challenge relates to the deployment of these new services in hardware platforms that can evolve with the demands of the network.

1.1 Existing Network Hardware

Today's fastest routers perform their packet processing operations in custom Silicon or Application Specific Integrated Circuits (ASICs). These systems contain hundreds or thousands of parallel logic circuits and finite state machines that are optimized to route, filter, queue, and/or process Internet datagrams in hardware.

While ASICs and custom Silicon networking chips have high performance, they offer little or no reprogrammability. Through the research on *Active Networks*, mechanisms have been developed which enable Internet routers to dynamically load packet processing software over the Internet. This feature allows the functions implemented on the routers to dynamically evolve to handle new protocols and capabilities.

Several types of software-based routing platforms have been developed. Many platforms use standard microprocessors, like the Intel Pentium, AMD Athlon, or the Mo-

*This research is supported by NSF: ANI-0096052 and Xilinx Corp.

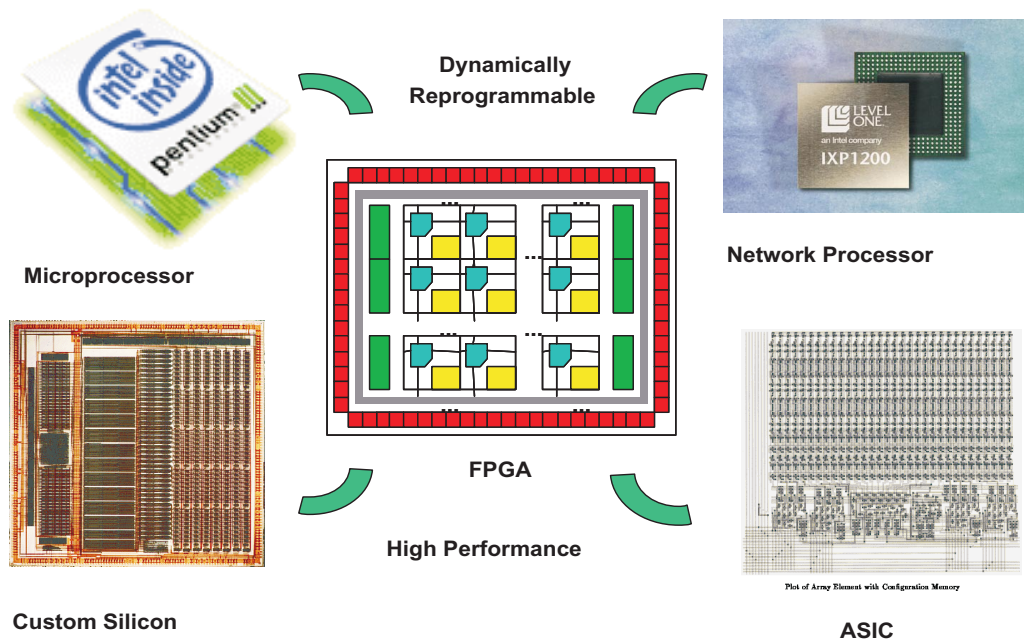


Figure 1. Technology Migration for Evolvable Hardware

torla/IBM PowerPC. Other platforms use network processors from vendors that include Agere, Intel, Motorola, MMC, and Vitesse. While these software-based systems have outstanding flexibility, their packet processing functionality is still limited due to the sequential nature of executing instructions in software.

1.2 Evolvable Network Hardware

Figure 1 shows technology options for implementing evolvable networking systems. Field Programmable Gate Arrays (FPGAs) are an exciting technology for evolvable networks. They share the performance advantage of ASICs and custom Silicon because they can implement parallel logic functions in hardware. They share the flexibility of the microprocessors and network processors in that they can be dynamically reconfigured [1].

Field Programmable Gate Arrays have proven to be an effective technology for implementation of networking hardware. In the development of the iPOINT testbed, a complete Asynchronous Transfer Mode (ATM) switch was built using FPGAs [2]. Further, a queuing module was implemented that provided per-flow queuing in FPGA hardware [3]. FPGAs have also proven to be effective for implementation of bit-intensive functions networking, such as Forward Error Correction (FEC) and boosting the performance of networking protocols [4]. The benefit of using

reprogrammable logic for networks is that complex algorithms for processing packets can evolve by reprogramming the actual hardware on the chip.

2 Field Programmable Port Extender (FPX)

Through support of the National Science Foundation and a grant from Xilinx, a platform called the Field Programmable Port Extender (FPX) has been developed. The FPX enables the rapid prototype and deployment of packet processing modules in reprogrammable hardware. The development of this system has provided insight into several design aspects of reprogrammable networking systems.

The FPX operates at the edge of a network router, processing packets as they pass between the backplane switch and the fiber-optic line cards. It uses a standard *Utopia* networking interface to transmit data on the network interfaces.

The FPX is used as a component in the Washington University Gigabit Switch (WUGS) [5]. Physically, the module is inserted between an optical line card and the WUGS gigabit switch backplane, as shown in Figure 2. The combined system is designed to implement a router which is both fully reconfigurable and capable of high performance [6].

The FPX implements all logic using two FPGA devices: the Network Interface Device (NID) and the Reprogrammable Application Device (RAD). The interconnec-

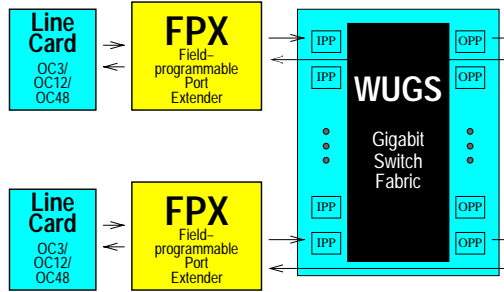


Figure 2. Evolvable Router using FPX

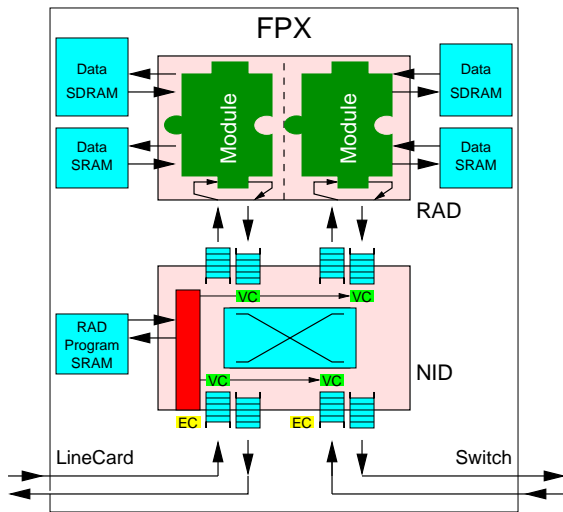


Figure 3. NID and RAD Configuration on FPX

tion of the RAD and NID to the network and memory components is shown in Figure 3.

2.1 Reprogrammable Application Device (RAD)

The Reprogrammable Application Device (RAD) contains the modules that implement customized packet processing functions. Each module on the RAD connects to one Static RAM (SRAM) and to one, wide Synchronous Dynamic RAM (SDRAM). In total, the modules implemented on the RAD have full control over four independent banks of memory. The SRAM is typically used for applications that need to implement table lookup operations, while the SDRAM interface is intended for applications that need only transfer bursts of data and can tolerate a higher memory latency.

The RAD communicates with the NID using the same Utopia interface as the NID to the network interfaces. Data on this interface are segmented into a sequence of fixed-size cells that are formatted as IP over ATM. Each interface has

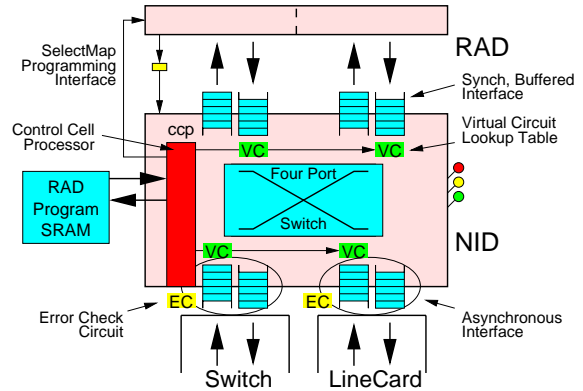


Figure 4. Network Interface Device Configuration

a small amount of buffering and implements flow control. A Start of Cell (SOC) signal is asserted to indicate the arrival of data, and the Transmit Cell Available (TCA) signal is asserted back towards the data source to indicate downstream congestion.

2.2 Network Interface Device (NID)

The Network Interface Device (NID) controls how packets are routed to and from modules. It also provides mechanisms to dynamically load hardware modules over the network and into the RAD. Modules can be dynamically loaded without affecting the processing of packets by the other modules in the system.

As shown in Figure 4, The NID has several components, all of which are implemented in FPGA hardware. It contains a four-port switch to transfer data between ports; *Virtual Circuit lookup tables* (VC) on each port in order to selectively route flows; a *Control Cell Processor* (CP), which is used to process control cells that are transmitted and received over the network; logic to reprogram the FPGA hardware on the RAD; and synchronous and asynchronous interfaces to the four network ports that surround the NID.

The NID routes flows among the two modules on the RAD, the network interface to the switch, and the network interface to the line card using a four-port switch. Each traffic flow that arrives on any incoming port can be forwarded to any destination port.

Each of the NID's four interfaces provide a small amount of buffering for short-term congestion. Buffers on the NID are implemented using on-chip memory. When packets contend for transmission to the same destination port, the NID performs arbitration. The design of the four-port

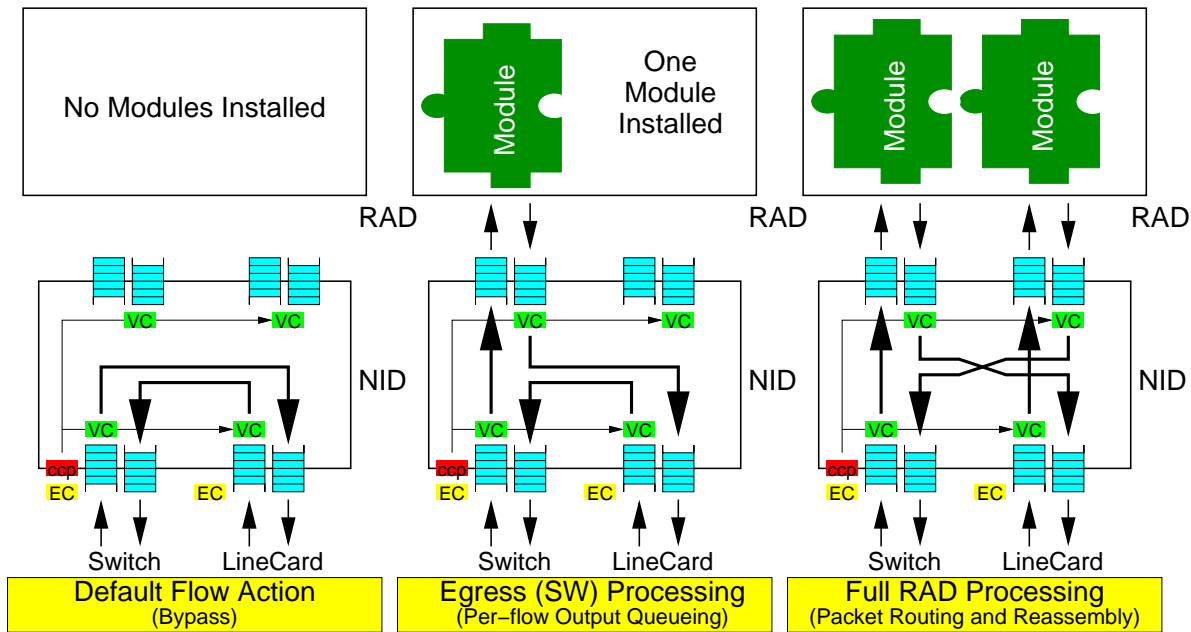


Figure 5. NID Switching Functionality

switch and scheduling algorithm used to arbitrate among flows is based on the design of the iPOINT switch [2] [7].

2.3 Network Module Switching

Application-specific functionality is implemented on the RAD as modules. Modules contain hardware and enforce a standard means of reading and writing the content of packets and interfacing with off-chip memory.

Hardware plugin modules on the RAD consist of a region of FPGA gates and internal memory, bounded by a well-defined interface to the network and external memory. Modules may occupy up to one half of an FPGA [8].

The NID reads a virtual circuit identifier to route traffic flows between modules. Examples that illustrate how the NID switches traffic to and from modules on the RAD are shown in Figure 5. By default, traffic flows are simply passed between the line card interface and the switch. To implement egress flow processing (i.e., to process packets as they exit the router), the NID routes a flow from the switch, to a RAD module, then out to the line card. To process packets on both the ingress path and egress path, the NID routes traffic to both modules on the RAD.

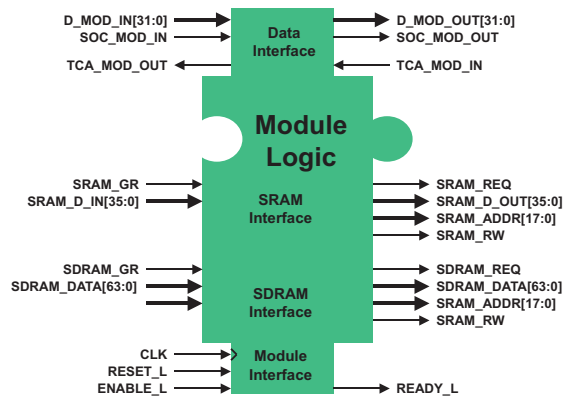


Figure 6. Modular Component of FPX

2.4 Modular Architecture

The modular entity of an FPX component is shown in Figure 6. Data arrives at and departs from a module using the D_MOD signals. A Start-of-Cell is indicated by the SOC_MOD signals. Congestion is indicated by the TCA signals.

The module provides two interfaces to off-chip memory. The SRAM interface supports transfer of 36-bit wide data to and from off-chip SRAM. The Synchronous Dynamic RAM (SDRAM) interface provides a 64-bit wide interface to off-chip memory. In the implementation of the IP lookup module, the off-chip SRAM is used to store the data struc-

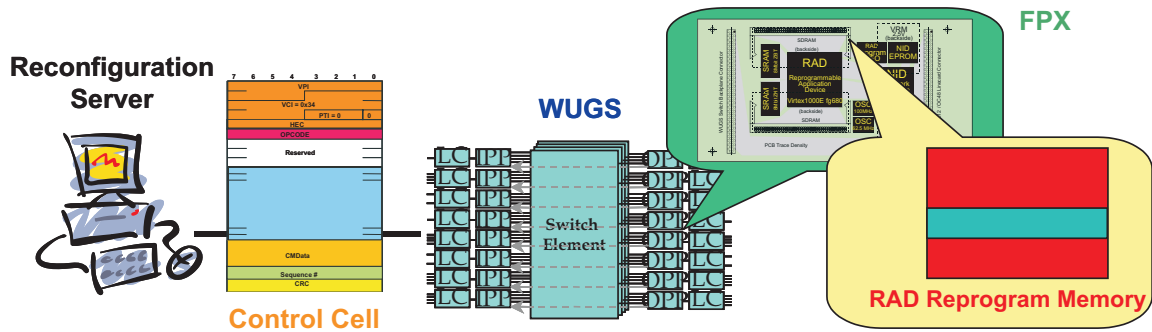


Figure 7. Dynamic Reconfiguration on FPX

tures of the fast IP Lookup algorithm [9]. Arbitration logic within the RAD allow memory devices to be shared among modules.

The number of memory operations that can be used to process packets depend on the speed of the Line Card rate and the length of the packet. The faster the link, the fewer the number of memory operations. The smallest packet that would be processed by the WUGS switch/router have fifty-six bytes.

Memory on the FPX operates at 100 MHz. For line cards that operate at OC3 rates (155 Mbits/second), each memory provides $53 * 8 / 155M = 273$ operations per cell. All four parallel banks of memory can, therefore, perform a total of $273 * 4 = 1092$ memory operations within the time period of a cell slot. At OC12 (622 Mbits/second), the same hardware can implement 273 operations. At OC48 (2.4 Gbits/second), the FPX provides 68 memory operations per slot period. Of these operations, $56/8 = 7$ writes to the SDRAM memory are used to enqueue a cell, and $56/8 = 7$ reads from the SDRAM are used to dequeue a cell. All remaining memory operations can be used to implement the routing and buffer management functions.

2.5 Reprogrammability

In order to reprogram a RAD module, the NID implements a reliable protocol to fill the contents of the on-board RAM with configuration data that is sent over the network. As shown in Figure 7, a software process running on a reconfiguration server sends control cells to the FPX over the network. As each control cell arrives, the NID on the FPX writes the data into the RAD's program memory. Once the last cell has been correctly received, and the FPX holds an image of the reconfiguration bitstream that contains the new hardware circuit. At that time, another control cell can be sent to NID to initiate the reprogramming of RAD using the contents of the memory.

The FPX supports partial reprogramming the RAD by allowing configuration streams to contain commands that only program a portion of the logic on the RAD. Rather than issue a command to reinitialize the device, the NID just writes the frames of reconfiguration data to the RAD's reprogramming port. This feature enables the other module on the RAD to continue processing packets during the partial reconfiguration. Similar techniques have been implemented in other systems using software-based controllers [10] [11].

Standard VHDL synthesis tools are used to develop the FPX networking modules. In order to reprogram only a portion of the Virtex FPGA, a tool has been developed called *ParBit*. ParBit parses the content of the FPGA hardware configuration file and extracts a subset of the data that reprograms a region of the FPGA. It also recomputes checksums for the truncated bitstream.

3 Implementation of the FPX

The FPX platform was implemented on the 20 cm \times 10.5 cm printed circuit board shown in Figure 8. The two FP-GAs, SRAMs, and one high-speed network interface can be seen in the photograph. The other network interface and SDRAM is on the underside of the board.

FPGAs were chosen so that the system would have sufficient capacity to implement meaningful networking applications. The RAD is implemented with a Xilinx Virtex 1000E-fg680 FPGA. The NID is implemented with a Virtex 600E-fg676 FPGA. The NID FPGA contains sufficient logic to implement all existing functionality, while the larger FPGA for the RAD was oversized so that future functions could evolve into unused hardware resources.



Figure 8. FPX Module

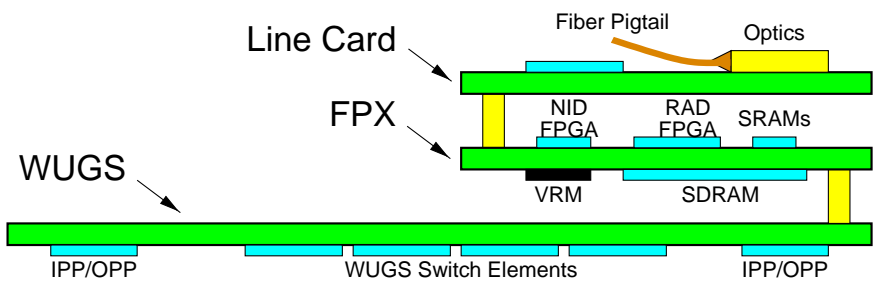


Figure 9. Configuration of switch backplane, FPX, and optical line card (side view)



Figure 10. Photo of FPX mounted in WUGS Switch

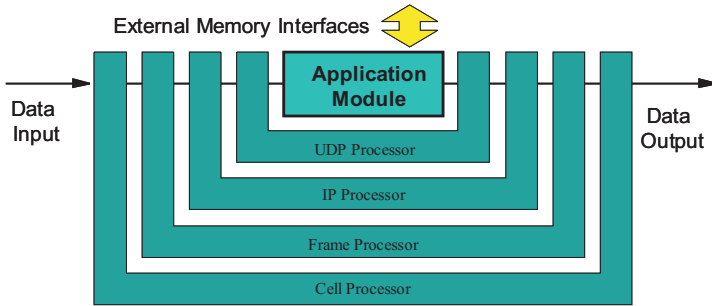


Figure 11. Packet Processing Wrapper

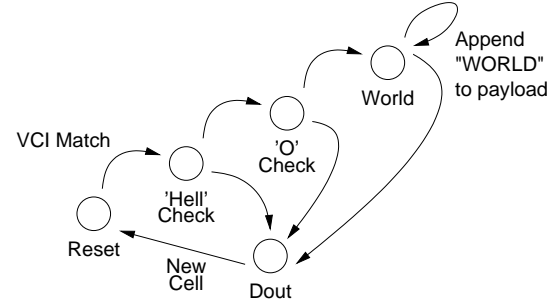


Figure 12. State machine for the Hello World circuit

3.1 Networking Interfaces

Networking interfaces on the FPX were optimized to enable the simultaneous arrival and departure of data cells at SONET OC48 rates. This is the equivalent bandwidth of multiple channels of Gigabit Ethernet.

Physically, the networking interfaces on the FPX were implemented in a way that allows the card to be stacked between the line card and backplane of a switch fabric. Figure 9 shows the configuration of an existing gigabit switch augmented with the evolvable hardware of the FPX.

3.2 Complete System

A photograph of the FPX module is shown in Figure 8. SRAM components on the FPX are mounted above and below the RAD. SDRAM memories are mounted on the back of the FPX in sockets. Reconfiguration data for the FPGAs are stored both in non-volatile Flash memory for the NID and SRAM memory for the RAD.

Figure 10 shows an FPX mounted in one port of the Washington University Gigabit Switch (WUGS) [5]. By inserting FPX modules at each port of the switch, parallel FPX units can be used to simultaneously process packets on all ports of the network switch.

4 Applications

Hardware components on the FPX are built as generic modules on the FPX. The Reprogrammable Application Device is used to implement the user-defined logic. Thus far, Modules have been developed for the FPX that implement IP packet routing [9], packet buffering, and application-level packet content modification. Additional hardware modules can be easily implemented on the FPX. The architecture of the FPX makes it well suited to imple-

ment applications like per-flow queuing [3] and flow control algorithms [12] in hardware.

4.1 Internet Packet Wrappers

Data can be transported over a network in a variety of formats. The FPX process data formatted in cells or packets. Cells that are processed by the FPX are assumed to be in the format of Asynchronous Transfer Mode. Internet packets are assumed to be in the format of an encapsulated frame carried in Adaptation Layer Five [13]. To simplify the development of applications at various levels of the protocol stack, a set of wrappers have been developed. A wrapper fits within an FPX module and provides the module inside with an higher-level interface to the incoming and outgoing data. As shown in Figure 11, wrappers have been developed to process cells, frames, IP packets, and UDP datagrams.

4.2 Content Matching Application

An application called “Hello, World” illustrates how easily an FPX module can be implemented to perform application-level content modification functionality. This module searches cells for a payload starting with the string “HELLO”. If and only if a match was found, the circuit concatenates the payload of the cell with the string “WORLD.”

The FPX implements the “Hello World” circuit very efficiently as a finite state machine on the RAD. A state transition diagram of the circuit is shown in Figure 12. The circuit was synthesized to FPGA logic for a module using Mentor Graphic’s Exemplar tool and Xilinx’s back-end placement and routing tools. The resulting circuit operates at: 119 MHz. The 8.4ns critical path in this circuit is well within the 10ns period provided by the RAD’s clock. Since the circuit can handle back-to-back cells; this circuit achieves the maximum packet processing rate of (100

MHz)/(14 Clocks/Cell)=7.1 Million packets per second.

The utilization of the FPGA to implement this circuit was 49 out of 12,288 slices on the FPGA. Thus, all of the logic to perform the search and replacement operation for this module was implemented in less than one-half of one percent of the chip resources.

A full report on the implementation of the Hello World module is available on-line [14]. The full VHDL source code and I/O pin mappings on the RAD can be downloaded from the project website [15].

5 Conclusions

Evolvable Internet hardware platforms will enable future routers to perpetually improve their packet processing functions and rapidly deploy new features and services. By implementing packet processing functions in reprogrammable logic, routers and firewalls can be implemented that have both flexibility and performance.

A prototype platform called the Field Programmable Port Extender (FPX) was built to experiment with evolvable Internet hardware. Core functionality on the FPX is implemented on an FPGA called the Networking Interface Device. It forwards traffic between modules; it reliably transports byte-streams over a network; and it implements the logic to dynamically reconfigure other FPGA logic in the system.

Application-specific functionality is implemented on the Reprogrammable Application Device. The RAD has modular interface to interface with application-specific hardware. Our experience has shown that there are several benefits to implementing features as modular components. First, the modular interface provides a well-defined interface for receiving and transmitting data between the network interfaces and off-chip memories. Second, it allows multiple modules to be easily integrated together. Lastly, due to the reprogrammable nature of the FPGA, it allows all functions of the router or firewall to be dynamically reprogrammed so that the system can evolve to provide enhanced features and functionality.

6 Acknowledgments

The development of the FPX has been supported by NSF: ANI-0096052 and Xilinx Corp. Contributors to this work include Jon S. Turner, David E. Taylor, Florian Braun, Edson Horta, and David Lim of Washington University and Dave Parlour of Xilinx.

References

- [1] S. Hauck, "The roles of FPGAs in reprogrammable systems," *Proceedings of the IEEE*, vol. 86, pp. 615–638, Apr. 1998.
- [2] "Illinois Pulsar-based Optical Interconnect (iPOINT) Homepage." <http://ipoint.vlsi.uiuc.edu>, Sept. 1999.
- [3] H. Duan, J. W. Lockwood, S. M. Kang, and J. Will, "High-performance OC-12/OC-48 queue design prototype for input-buffered ATM switches," in *INFOCOM'97*, (Kobe, Japan), pp. 20–28, Apr. 1997.
- [4] W. Marcus, I. Hadzic, A. McAuley, and J. Smith, "Protocol boosters: Applying programmability to network infrastructures," *IEEE Communications Magazine*, vol. 36, no. 10, pp. 79–83, 1998.
- [5] J. S. Turner, T. Chaney, A. Fingerhut, and M. Flucke, "Design of a Gigabit ATM switch," in *INFOCOM'97*, 1997.
- [6] S. Choi, J. Dehart, R. Keller, J. Lockwood, J. Turner, and T. Wolf, "Design of a flexible open platform for high performance active networks," in *Allerton Conference*, (Champaign, IL), 1999.
- [7] J. W. Lockwood, H. Duan, J. J. Morikuni, S. M. Kang, S. Akkineni, and R. H. Campbell, "Scalable optoelectronic ATM networks: The iPOINT fully functional testbed," *IEEE Journal of Lightwave Technology*, pp. 1093–1103, June 1995.
- [8] J. W. Lockwood, N. Naufel, J. S. Turner, and D. E. Taylor, "Reprogrammable network packet processing on the field programmable port extender (fpx)," in *ACM International Symposium on Field Programmable Gate Arrays (FPGA'2001)*, (Monterey, CA), pp. 87–93, Feb. 2001.
- [9] J. W. Lockwood, J. S. Turner, and D. E. Taylor, "Field programmable port extender (FPX) for distributed routing and queuing," in *FPGA'2000*, (Monterey, CA), pp. 137–144, Feb. 2000.
- [10] W. Westfeldt, "Internet reconfigurable logic for creating web-enabled devices." Xilinx Xcell, Q1 1999.
- [11] S. Kelem, "Virtex configuration architecture advanced user's guide." Xilinx XAPP151, Sept. 1999.
- [12] M. Bossardt, J. W. Lockwood, S. M. Kang, and S.-Y. Park, "Available bit rate architecture and simulation for an input-buffered and per-vc queued ATM switch," in *GLOBECOM'98*, (Sydney, Australia), pp. 1817–1822, Nov. 1998.

- [13] M. Laubach and J. Halpern, "Classical IP and ARP over ATM." RFC 2225, Apr. 1998.
- [14] J. Lockwood and D. Lim, "Hello World: A simple application for the field programmable port extender (FPX)," tech. rep., WUCS-00-12, Washington University, Department of Computer Science, July 11, 2000.
- [15] "Field Programmable Port Extender Homepage." <http://www.arl.wustl.edu/projects/fpx/>, Aug. 2000.