Report Number: WUCSE-2002-15

2002-06-16

Field Programmable Port Extender (FPX) User Guide (Version 2.2)

John W. Lockwood

This manual summarizes how to insert the Field Programmable Port Extender (FPX) into the Washington University Gigabit Switch (WUGS), how to install the NCHARGE control software, how to initialize the system, and how to reprogram a user-defined module into the FPX over the network using the included web-based tools.

2002-15

Field Programmable Port Extender (FPX) User Guide (Version 2.2)

Authors: John W Lockwood

June 18, 2002

Abstract: The Field-programmable Port Extender (FPX) is a general-purpose, reprogrammable platform that performs data processing in Field Programmable Gate Array (FPGAs) hardware. The FPX extends operation of the Washington University Gigabit Switch (WUGS) by adding FPGA hardware at ingress and egress ports.

As with the Smart Port Card (SPC), data packets can be actively processed by user-defined, reprogrammable modules as they pass though the device. Unlike the SPC, however, the FPX uses reprogramamble hardware, not software, to process the packets. The hardware-based processing allows the FPX to achieve multi-Gigabit per second throughput, even when performing deep processing of the packet payload.

This manual summarizes how to insert FPX into the Washington University Gigabit Switch, how to install the NCHARGE control

Field Programmable Port Extender (FPX) User Guide: Version 2.2

John W. Lockwood [*]
and all students of the FPX project

Department of Computer Science
Applied Research Lab
Washington University
1 Brookings Drive, Box 1045
Saint Louis, MO 63130

Email: `lockwood@arl.wustl.edu`
Phone: (314) 935-4460

`http://www.arl.wustl.edu/arl/projects/fpx`

June 18, 2002

Abstract

This manual summarizes how to insert the Field Programmable Port Extender (FPX) into the Washington University Gigabit Switch (WUGS), how to install the NCHARGE control software, how to initialize the system, and how to reprogram a user-defined module into the FPX over the network using the included web-based tools.

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

Contents

List of Figures

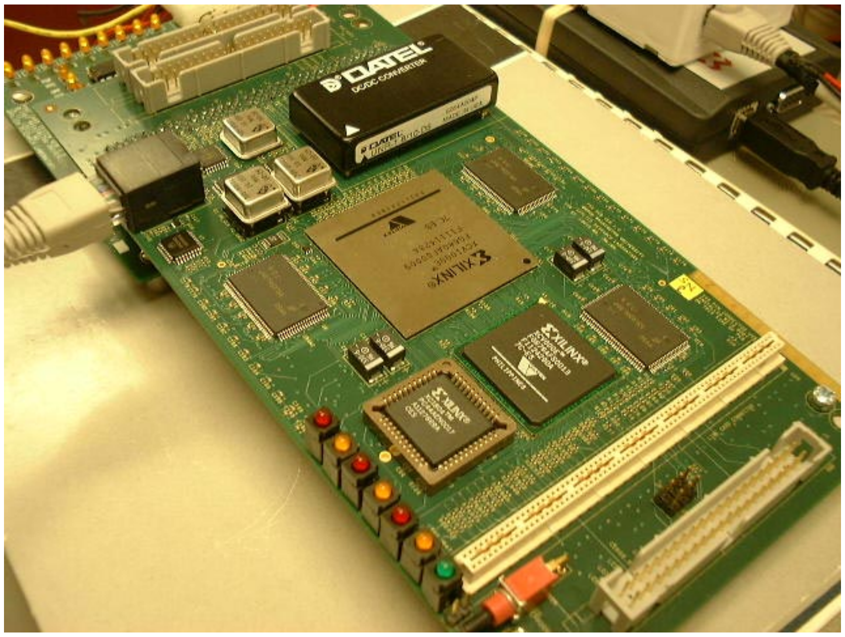


Figure 1: FPX Module

1 Contents of the package

The FPX module included in this package includes:

- An assembled FPX card with the Version 2.0 Printed Circuit Board
- A Xilinx XCV2000e-6 Field Programmable Gate Array (FPGA) that can be programmed by the user to perform high-speed networking functions.
- Two banks of Synchronous Dynamic Random Access Memory (SDRAM), each with a capacity of 64 MBytes.
- Three banks of Pipelined Zero Bus Turnaround (ZBT) Static RAM (SRAM), each with a capacity of 1MByte
- A programmable Read Only Memory (PROM) that programs the NID to forward cells to the RAD and to reconfigure the RAD via commands sent over the network.

A photograph of the FPX is shown in Figure 1.

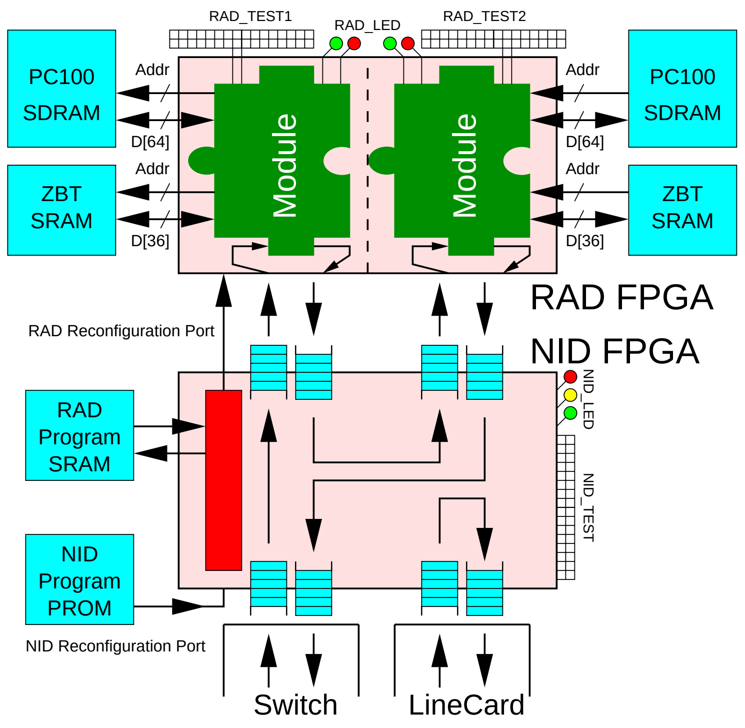


Figure 2: Logical Configuration of the FPX

2 Background

The Field-programmable Port Extender (FPX) is a general-purpose, reprogrammable platform that performs data processing in Field Programmable Gate Array (FPGAs) hardware. The FPX extends operation of the Washington University Gigabit Switch (WUGS) by adding FPGA hardware at ingress and egress ports [1]. As with the Smart Port Card (SPC), data packets can be actively processed by user-defined, reprogrammable modules as they pass though the device [2]. Unlike the SPC, however, the FPX uses reprogramamble hardware, not software, to process the packets [3]. The hardware-based processing allows the FPX to achieve multi-Gigabit per second throughput, even when performing deep processing of the packet payload.

The FPX has two high-capacity Xilinx Virtex FPGA that are called the Network Interface Device (NID) and the Reprogrammable Application Device (RAD). The NID is a Xilinx Virtex XCV600Efg676 FPGA that is programmed by the firmware on the FPX at startup. The RAD is a much larger FPGA that is available for user-defined functionality. The RAD on this second version (Version 2.0) of the FPX is an XCV2000Efg680 FPGA that provides even more logic resources than the original FPX (Version 1.0) that had the XCV-1000Efg680 FPGA.

3 Application Development on the FPX

User defined packet processing circuits are implemented as modules on on the RAD. 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. Arbitration logic within the RAD allows memory to be shared among multiple modules [4]. On the FPX, the memory interfaces support transfer of 36-bit wide data to and from off-chip, Zero-Bus-Turnaround Static RAM (ZBT SRAM) and the PC-100 Synchronous Dynamic RAM (SDRAM) interface supports transfer of 64-bit wide data [5]. Once an application has been implemented, it is programmed in to the RAD over the network.

Application-specific functionality is implemented on the RAD as modules. Hardware plugin modules

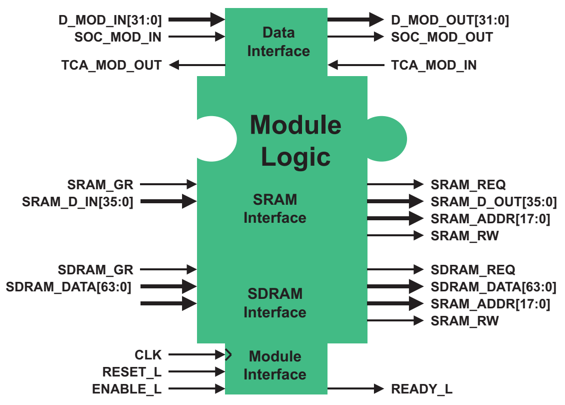


Figure 3: Modular Component of FPX

on the RAD consist of a region of FPGA gates, bounded by the well-defined interface to the network and external memories. New modules can be loaded into the system over the network at any time by run-time reconfiguration of the RAD FPGA [6]. Modules can occupy all of the RAD, half of the RAD, or a region of the RAD [7]. Dynamic reconfiguration allows the hardware on the FPX to evolve while new features are added [8]. The modular interface of an FPX component is shown in Figure 3.

3.1 Data Interfaces

The FPX reads and writes data from the network over a 32-bit Utopia interface. Data can be formatted directly as Asynchronous Transfer Mode (ATM) cells or in arbitrary protocols encapsulated over ATM.

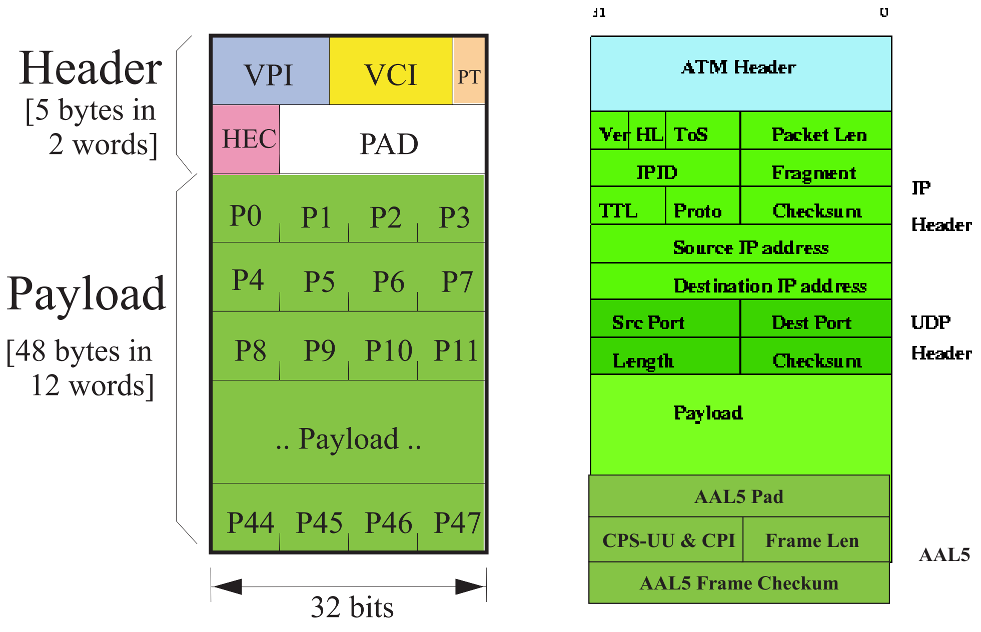


Figure 4: Cell and Packet Format

The cell and packet format for two typical communications is shown in Figure 4. As an ATM cell, the first word contains a Virtual Path Identifier (VPI), a Virtual Circuit Identitifier (VCI), a Payload Type (PT) field, a Header Error Check (HEC), a padding field (PAD), and the payload of the packet. As a User Datagram Protocol (UDP) packet, the FPX interface carries the data segmented into one or more cells along with the fields of the Internet Protocol (IP) header, the UDP header, the user Payload, and an ATM Adaptation Layer Five (AAL5) trailer.

3.2 Memory Interfaces

Standard interfaces have also been developed to provide access 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. Access to the external SRAM and SDRAM are arbitrated by infrastructure logic. The FPX provides arbitrated interfaces to SRAM and SDRAM. This feature allows multiple, on-chip modules to share access to a command SRAM or SDRAM [9].

3.3 SDRAM Controller

SDRAM is a suitable technology for packet buffering because it has high capacity at very low cost. The SDRAM memory controller on the FPX has been implemented to provide a simplied interface to SDRAM memory and allows multiple modules to to share memory. Multiple modules contend for the SDRAM access and the SDRAM controller arbitrates between the requests made by each module and provides access to the SDRAM. The system schedules requests from multiple modules to optimize the performance of the memory access [10].

3.4 Layered Protocol Wrappers

In order to simplify the development of Internet applications in hardware, a wrapper library has been implemented to abstract the processing of lower-level protocols. The Wrapper Library is a collection of VHDL components that process high-level Internet protocols directly in hardware. The hardware library has four components that are used together to transmit and receive: fixed-length cells, variable-length Adapation-Layer 5 (AAL5) frames, Internet Protocol (IP) packets, and User Datagram Procotol (UDP) frames. All header checksums and lengths fields are checked and/or computed within the wrapper library components [11] [12].

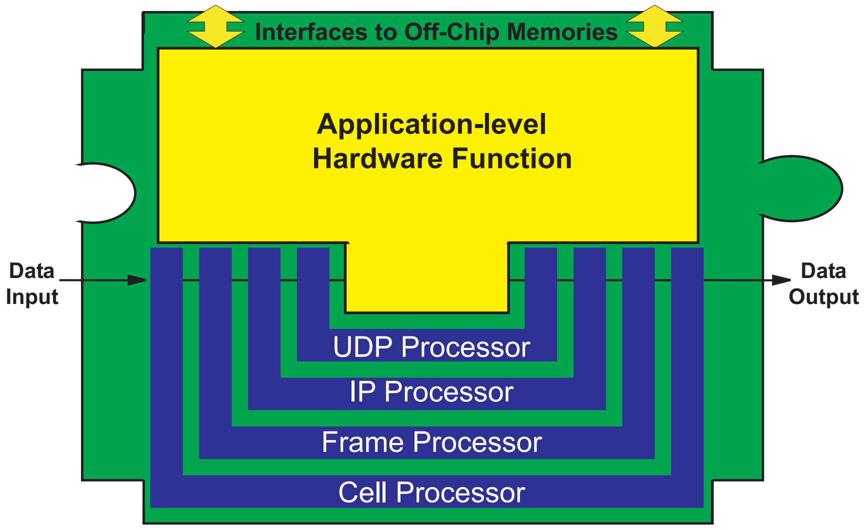
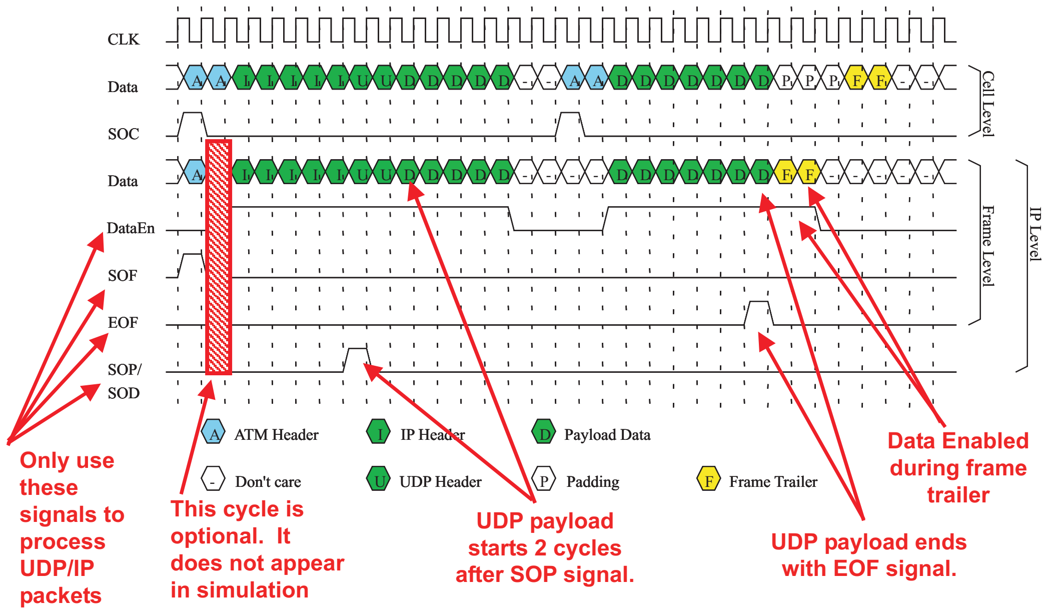


Figure 5: Module with Wrappers

A module that uses the wrappers is illustrated in Figure 5. Note that the application circuit need only process the user-level data. All of the lower-level protocol processing is handled by the hardware libraries.



CLK
Data
SOC
Data
DataEn
SOF
EOF
SOP/
SOD
Cell Level
Frame Level
IP Level
ATM Header
IP Header
Payload Data
Don't care
UDP Header
Padding
Frame Trailer
Only use these signals to process UDP/IP packets
This cycle is optional. It does not appear in simulation
UDP payload starts 2 cycles after SOP signal.
UDP payload ends with EOF signal.
Data Enabled during frame trailer

Figure 6: Typical Wrapper Waveforms

Typical waveforms for the wrappers are shown in Figure 6. At the cell level, the arrival of data is indicated by the Start of Cell (SOC) signal. At the frame level, the Start of Frame (SOF) signal goes high as a new frame arrives, the End of Frame (EOF) signal goes high on the last word of the payload, and the Data Enable signal remains high as each valid word is passed though the interface. At the Internet Protocol (IP) level, the Start of Packet (SOP) signal is active during the first word of the payload. For a User Datagram Protocol (UDP) message, the Start of Datagram (SOD) signal goes high during this same clock cycle.

3.5 Educational Program

There are several on-line resources that provide detailed information on the FPX.

3.5.1 The CS535 On-line Notes

The FPX has been used as the focus for a graduate-level research course called: *CS535: Acceleration of Algorithms in Hardware* [13]. Through the course of five machine problems, students learned how FPGA hardware could be used to implement packet processing functions in hardware. These example circuits can be downloaded as TAR files. All of the course is on-line as:

`http://www.arl.wustl.edu/~lockwood/class/cs535/index.html`

3.5.2 The Gigabit Kits Workshop

The FPX has been described during several talks of the Gigabit kits workshop. The talks remain on-line as:

`http://www.arl.wustl.edu/arl/projects/fpx/workshop_0101/agenda.html`
`http://www.arl.wustl.edu/arl/projects/fpx/workshop_0801/agenda.html`
`http://www.arl.wustl.edu/arl/projects/fpx/workshop_0102/agenda.html`
`http://www.arl.wustl.edu/arl/projects/fpx/workshop_0602/agenda.html`

3.5.3 Papers and Technical Reports

On-line versions of journal papers, conference papers, and technical reports can be downloaded as:

`http://www.arl.wustl.edu/arl/projects/fpx/references/`

4 Hardware Installation

To simplify the software configuration, A default configuration is of the FPX devices the Washington University Gigabit Switch (WUGS) is suggested for the first system installation. The default configuration is same as that used in the Washington University hardware laboratory during the January 2002 workshop. A photograph of a system is shown in Figure 7.

In this configuration, the primary FPX is attached to port 4. The switch controller is attached to port 3. Additional FPX devices are added to the ports as shown in the photo.

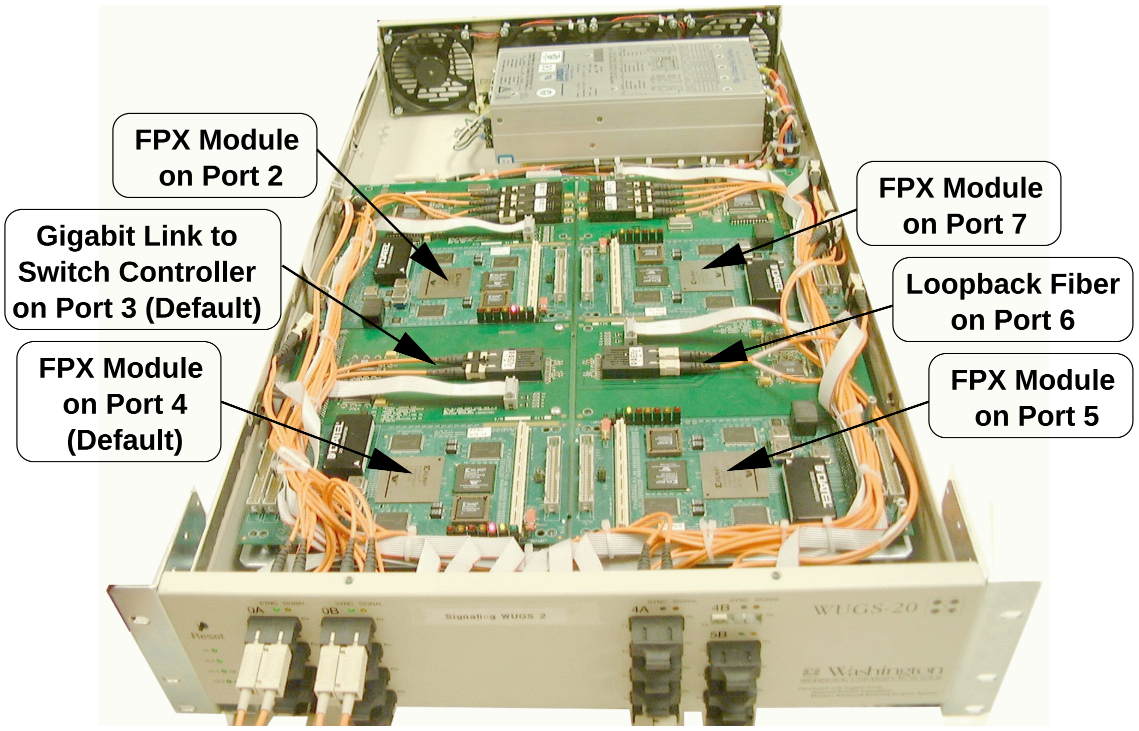


Figure 7: Default configuration

4.1 Cautions and Warnings

In order to install the FPX in the WUGS, the cover to the switch must opened and line cards removed. Before taking any action, it is critical to note that:

- Both the FPX and the WUGS are electronic devices that are highly sensitive to electrostatic shock. Before physically handling any such device, be sure to discharge any static electricity by touching a well-grounded, metal fixture and/or attaching a grounding strap to your wrist.
- The WUGS switch contains high-voltage circuits and moving fans. Use caution when working on or near the system.
- Be sure that the power to the system is OFF before working with the switch. The line cards are NOT intended to be installed or removed while the power is on.

4.2 Existing Line Card Configuration

By default, the Washington University Gigabit Switch (WUGS) is shipped fully populated with eight Line Cards. Each line card (LC) in the WUGS is mounted above one of the eight ports in the switch, as shown in Figure 8.

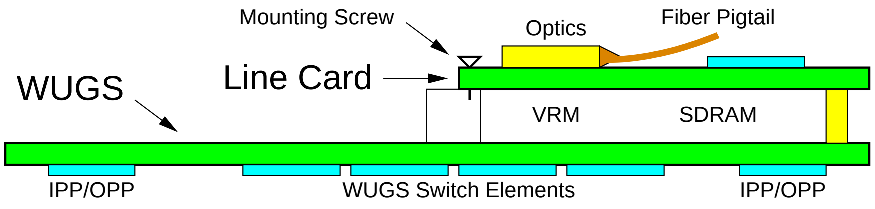


Figure 8: Original configuration of WUGS backplane with line card (side view)

4.3 FPX Installation

Once the precautions to protect the hardware and yourself have been taken, the FPX can be installed into the switch. As shown in Figure 7, the default location for a switch with a single FPX is port 4.

The FPX includes mounting posts to securely support the card in the switch and mount screws to ensure that the card remains in place. The resulting configuration is shown in Figure 9.

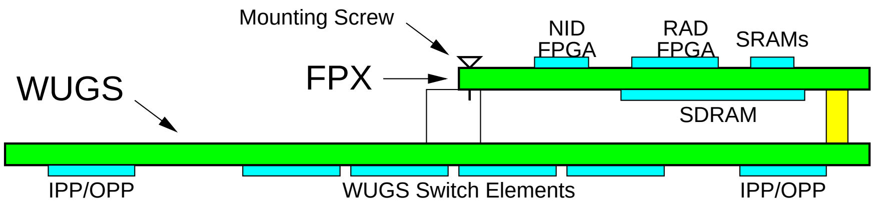


Figure 9: New configuration with FPX

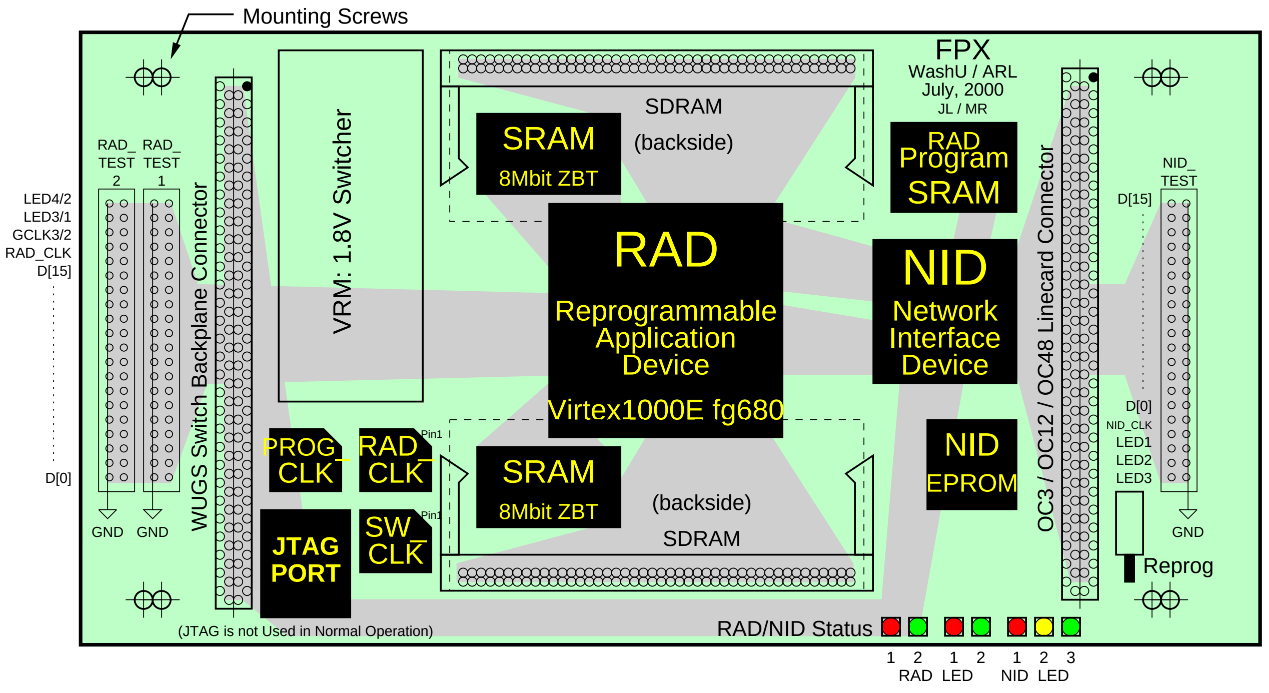


Figure 10: FPX Printed Circuit Board

5 Hardware Components

The FPX includes several hardware components for monitoring the status of the system and controlling the operation of the FPX. The FPX itself is implemented as a 12-layer printed circuit board with the dimensions 20 cm $\times$ 10.5 cm, as shown in Figure 10.

5.1 LED Status Display

Light Emitting Diodes (LEDs) on the FPX can be observed to determine the status of the system. The FPX has a total of seven LEDs that are both visible on the surface of the card and electronically accessible via the *test pin* connectors.

5.1.1 NID LEDs

Three LEDs indicate the status of the NID. By default, the NID is programmed to blink the NID LEDs to indicate The status of system clocks.

- The blink rate of NID1 LED (red) is proportional to `RAD_CLK`
- The blink rate of NID2 LED (yellow) is proportional to `SW_CLK`
- The blink rate of NID3 LED (green) is proportional to the line-card clock.

Note that the red and yellow LEDs should always blink when the system is powered, but that the green LED will only light if a line card is placed atop the FPX.

5.1.2 RAD LEDs

Four LEDS indicate the status of the RAD. The infrastructure logic on the RAD is, by default, programmed to blink the LEDs on the RAD to indicate that a module has been programmed.

- LED1 (red) and LED2 (yellow) indicate the status of the left half of the RAD.
- LED3 (red) and LED4 (yellow) indicate the status of the right half of the RAD

If LEDs on the RAD do not blink, it is likely that the RAD is not programmed.

5.2 Test Pins

The FPX has a three I/O connectors which can be used to interconnect the FPGA with external devices. Each connector has provides a 16-bit data bus, a clock output, an external clock input (for the RAD connectors), and a ground connector.

- All I/O signals attached to the FPX test connectors must conform to the 3.3V LVTTL standard. If 5.0V signals are applied to the test pins, the FPX can be damaged.
- Note that the orientation of the connector is critical. Every other pin on the connector is connected to ground. If a cable is attached in the wrong orientation, the FPX can be damaged.
- The LED signals operate by pulling the emitter of a LED to ground. External LEDs must be powered by an external 3.3V power source referenced to the FPX ground plane. Current limiting series resistors must be used to ensure that current though the LED never exceeds 10mA.

5.2.1 NID-TEST

The test bus on the NID has the following signals:

- Ground
- Data (15 downto 0)
- `NID_CLK` (output)
- LED1 (output pulled to ground to light LED)
- LED2 (output pulled to ground to light LED)
- LED3 (output pulled to ground to light LED)

5.2.2 RAD-TEST

The test bus on the NID has the following signals:

- `RAD_TEST1`
  - Ground
  - Data (15 downto 0)
  - LED1 (output pulled to ground to light LED)
  - LED2 (output pulled to ground to light LED)
  - GCLK2 (Input to RAD)
- `RAD_TEST2`
  - Ground
  - Data (15 downto 0)
  - LED3 (output pulled to ground to light LED)
  - LED4 (output pulled to ground to light LED)
  - GCLK2 (Input to RAD)

5.3 Clocking

The user logic on the RAD operates on a clock called`RAD_CLK`. The oscillator which controls this clock is socketed on the FPX. In the current distribution of the systems, the clock operates at a frequency of 62.5 MHz (corresponding to clock period of 16ns). Logic on the RAD should be synthesized to operate as fast or faster than `RAD_CLK` or faster. Most FPX modules developed at Washington University operate at 100 MHz.

5.3.1 Changing the clock rate

The clock on the FPX can be changed. The four-pin oscillator labeled as `U2: RAD_CLK` can be replaced with a compatible oscillator.

- Note that pin 1 of the oscillator is indicated on the board via an angled notch in the square of the silkscreen pattern on the PCB. Be sure that the dot on the oscillator lines up with this notch! Proper orientation of the device is critical. The FPX can be damaged if components are placed into the system In the wrong orientation!

- Note that the oscillator must be a 3.3V device. The voltage of the oscillator is critical. The FPX can be damaged if 5.0V components are placed into the system.

5.3.2 Other clocks

Components of the FPX operate on several clock domains. The NID clocks with the switch at a frequency specified by by a clock which is socketed and labeled as `SW_CLK`. By default, it operates at 62.5 MHz. The `RAD_CLK` is socketed, and labeled: `RAD_CLK` 100 MHz

5.3.3 User-defined clocks

The Test pin connector allows external clocks called `GCLK2` and `GCLK3` to feed logic on the RAD. The FPGA contains internal global clock distribution circuits that feed these signals to flops. External clocks must conform to 3.3V LVTTL specification detailed in the Xilinx VirtexE manual.

5.3.4 Clock multipliers

The Xilinx VirtexE FPGA has DLLs that allow clocks to operate at certain multiples of a base frequency. Consult the Xilinx manual for further information about DLLs.

6 Installation of the NCHARGE Control Software

A suite of tools called NCHARGE (Networked Configurable Hardware Administrator for Reconfiguration and Governing via End-systems) has been developed to simplify the co-design of hardware and software components within an FPGA-based Internet router or firewall. NCHARGE provides a standard Application Programming Interface (API) between software and reprogrammable hardware modules. Using this API, multiple software processes can communicate to one or more hardware modules using standard TCP/IP sockets. NCHARGE also provides a Web-Based User Interface to simplify the configuration and control of an entire network switch that contains several software and hardware modules [14].

6.1 Obtaining the FPX Software

- The FPX software can be downloaded from the WashU Gigabits Kit program's CVS repository.
  - Set up the CVS remote shell to use secure shell
    ```
    setenv CVS_RSH ssh
    setenv CVSROOT :ext:cvs.arl.wustl.edu:/usr/cvsroot
    ```
  - Check out the files from the remote CVS machine
    ```
    cvs checkout FPX_ROOT/NCHARGE
    ```

6.2 Directories in the CVS Repository

When you check out files from the CVS repository, you should obtain:

- `NCHARGE/*`
- `NCHARGE/FPX_SOFTWARE/*`
- `NCHARGE/FPX_SOFTWARE/cgi-bin/*`
- `NCHARGE/FPX_SOFTWARE/htdocs/*`
- `NCHARGE/FPX_SOFTWARE/htdocs/documents/*`
- `NCHARGE/FPX_SOFTWARE/etc/*`
- `NCHARGE/FPX_SOFTWARE/conf/*`

6.3 Required Components

In order to run the Control software (web-based scripts and NCHARGE program), the following are assumed to be installed and configured correctly:

- NetBSD on Linux: Running the WashU APIC kernel
- Perl: Located in `/usr/pkg/bin/perl`
- Jammer: Located in `/usr/SPC/bin/Jammer`
- GBNSC: Located in `/usr/SPC/bin/GBNSC`
- `apic0`: connected to port 3 of the switch acting as the switch controller
- Apache Web Server: located in `/usr/local/apache` we use Apache 1.3.14 (httpd.conf is included for reference)

6.4 Description of the Included Files

The `fpx_software` includes the following major components:

- `cgi-bin/`: Contains all the scripts and executable programs to control and manage the FPX.
- `htdocs/`: Contains the HTML text for the FPX web page
- `conf/`: Contains configuration for the Apache web server. You may copy our httpd.conf file to your own apache/conf folder, as you wish, or use it as a reference
- `etc/` Contains htpd and htgroups.dat files, copy to etc/ if you don't want to store these files under /etc make sure to edit the following files to the correct path: `cgi-bin/check_ports.cgi` `/usr/local/apache/conf/httpd.conf`

6.5 Installation

After the directories (above) have been extracted:

- Copy the cgi-bin and htdocs folders to your Apache directory (/usr/local/apache) or you can create a symbolic link inside the apache folder for cgi-bin and htdocs.
- To set up the correct binaries for NetBSD or Linux, run the

  ```
  link_bins.pl
  ```

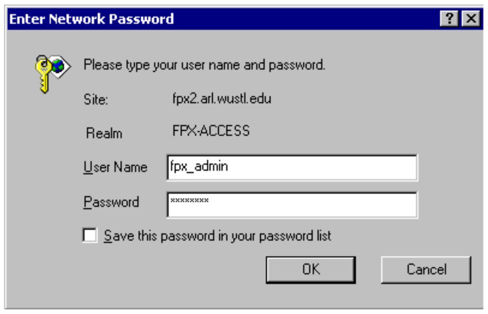
- Configure your httpd.conf to handle cgi-bin (see our httpd.conf file)
- Set up a password protected scheme to control access to your FPX.

Full documentation and details regarding the installation, testing, and operation of the NCHARGE software are included in the README file of the NCHARGE software.

7 FPX Operation

Once the software and hardware are in place, it is simple to use the FPX to process packets. The NCHARGE software provides a graphical user interface to the FPX that can be used to initialize the system, program RAD modules, generate test data.

The FPX is most easily controlled using the Web-based Graphical User Interface (GUI). From any web browser, open a browser window to a FPX controller. At this point a login and password are required, as shown in Figure 11. By default an administrative account with the user name fpx_admin has been created. The password is gigakits. Refer to the readme file included in the distribution for help on creating additional accounts. After logging in the screen should appear exactly like Figure 12.


Enter Network Password

Please type your user name and password.

Site: fpx2.arl.wustl.edu

Realm FPX-ACCESS

User Name fpx_admin

Password ********

Save this password in your password list

Figure 11: NCHARGE Login Prompt for FPX Homepage

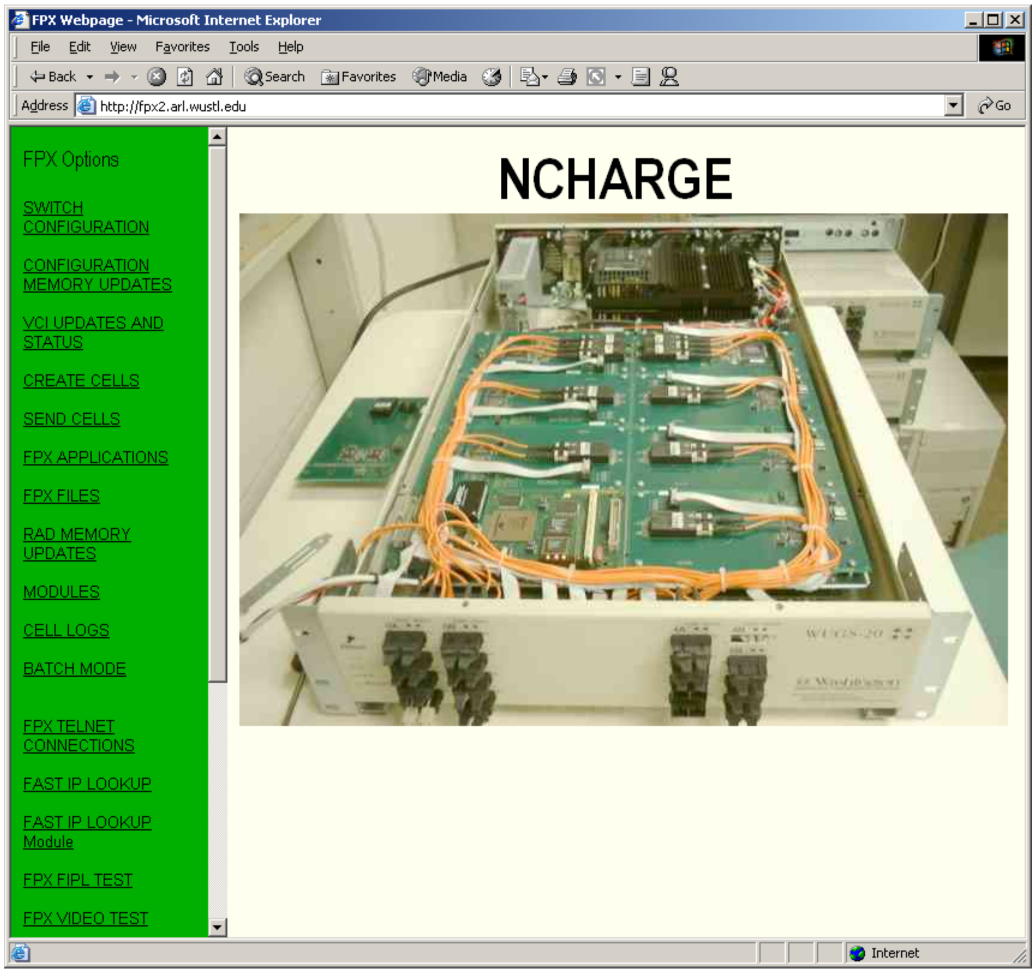
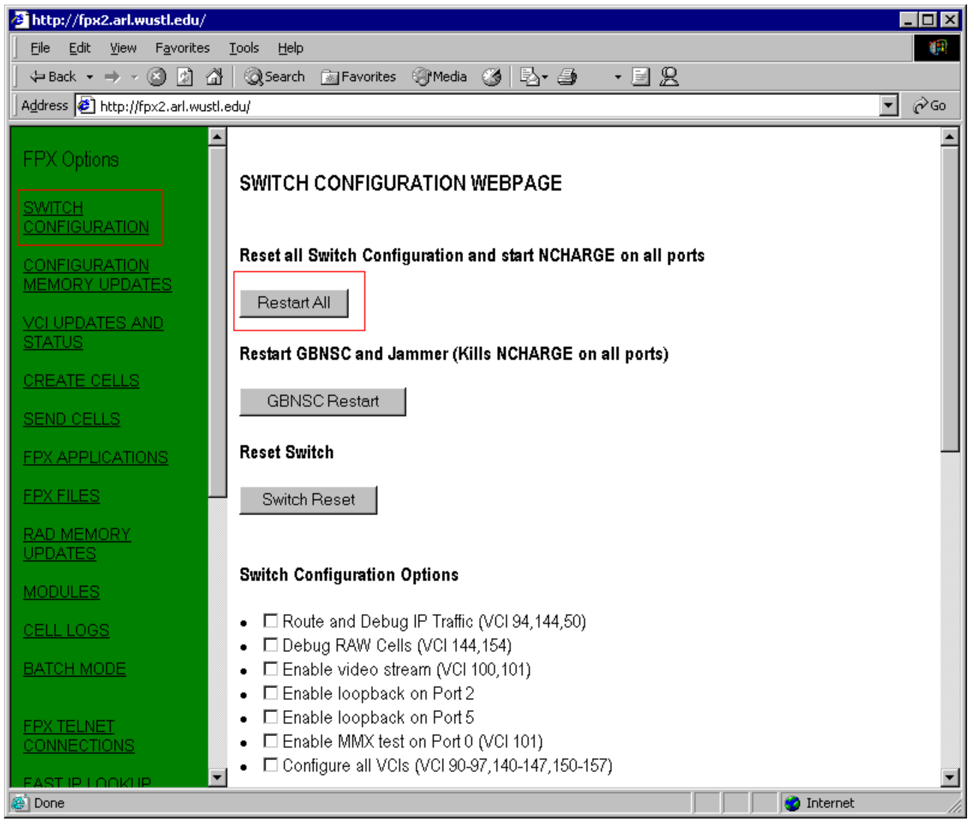


Figure 12: NCHARGE Step 0: Login to FPX Homepage

7.1 System Initialization

From the FPX homepage, a single button can be used to reset the GBNSC switch software, reset the switch hardware, and initialize the NCHARGE control applications. The button to press is shown in Figure 13.



SWITCH CONFIGURATION WEBPAGE

Reset all Switch Configuration and start NCHARGE on all ports

Restart GBNSC and Jammer (Kills NCHARGE on all ports)

Reset Switch

Switch Configuration Options

- ☐ Route and Debug IP Traffic (VCI 94,144,50)
- ☐ Debug RAW Cells (VCI 144,154)
- ☐ Enable video stream (VCI 100,101)
- ☐ Enable loopback on Port 2
- ☐ Enable loopback on Port 5
- ☐ Enable MMX test on Port 0 (VCI 101)
- ☐ Configure all VCIs (VCI 90-97,140-147,150-157)

Figure 13: NCHARGE Step 1: One-button Switch Reset

7.2 Module Reprogramming

A key feature of the FPX relates to the mechanism by which it can load new modular functions over the network. The FPX supports partial reprogramming of the RAD by allowing configuration streams to program only a portion of the logic on the RAD. Rather than issue a command to reload the entire device, the NID writes only selected reconfiguration frames to the RAD's *SelectMap* reprogramming port. This feature enables other module(s) on the RAD to continue processing packets during the partial reconfiguration. The method of programming the Virtex FPGA is similar to [15] [16] in that only a portion of the device is reconfigured. The method differs, however, in that the FPX can reprogram modules on the RAD using a hardware-based controller rather than a microprocessor.

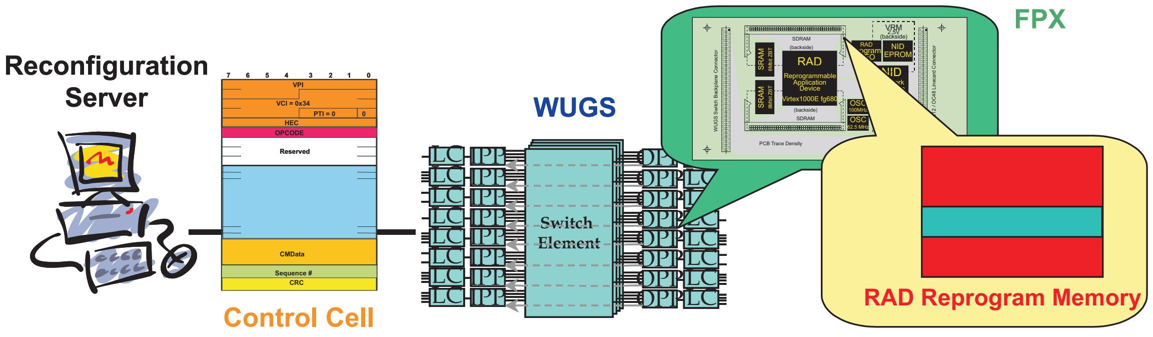
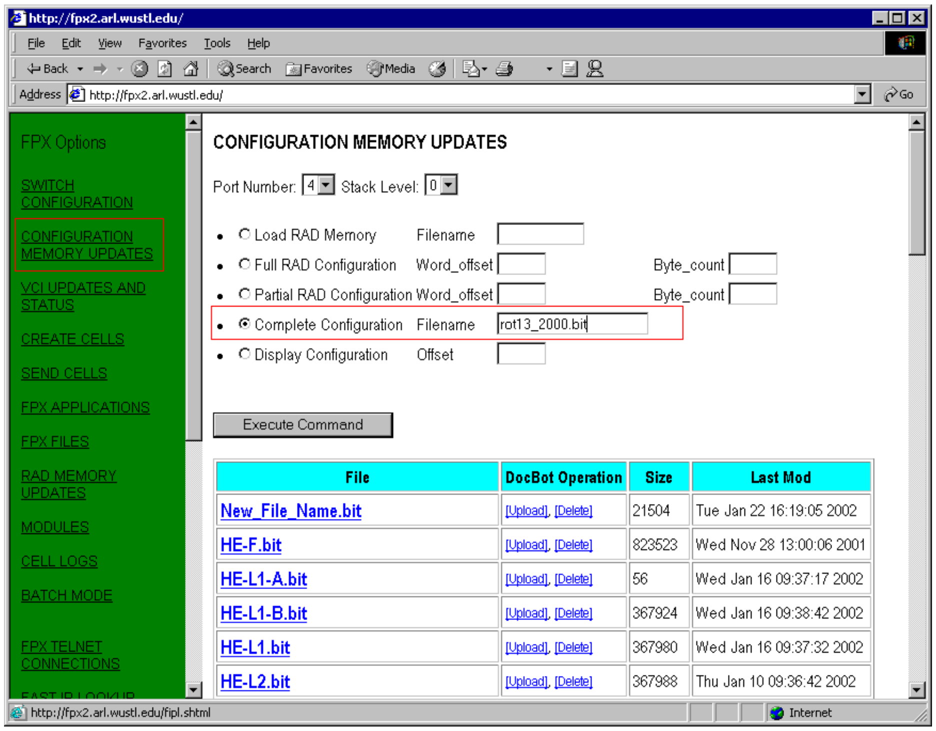


Figure 14: Dynamic Reconfiguration on FPX

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 14, a reconfiguration server on the network sends reconfiguration data to the FPX as control cells over the network. As control cells arrive, the NID on the FPX writes the data into the *RAD reprogram memory*. Once the last cell has arrived, the FPX holds an image of the reconfiguration bytestream that contains the new hardware circuit. Once the network is ready to load the module, a final control cell is sent to NID to initiate the reprogramming of RAD using the contents of the reprogram memory.

The RAD can be reprogrammed from the NCHARGE menu. To program the ROT13 encryption module in the FPX, submit a page as shown in Figure 15.



CONFIGURATION MEMORY UPDATES

Port Number: 4 Stack Level: 0

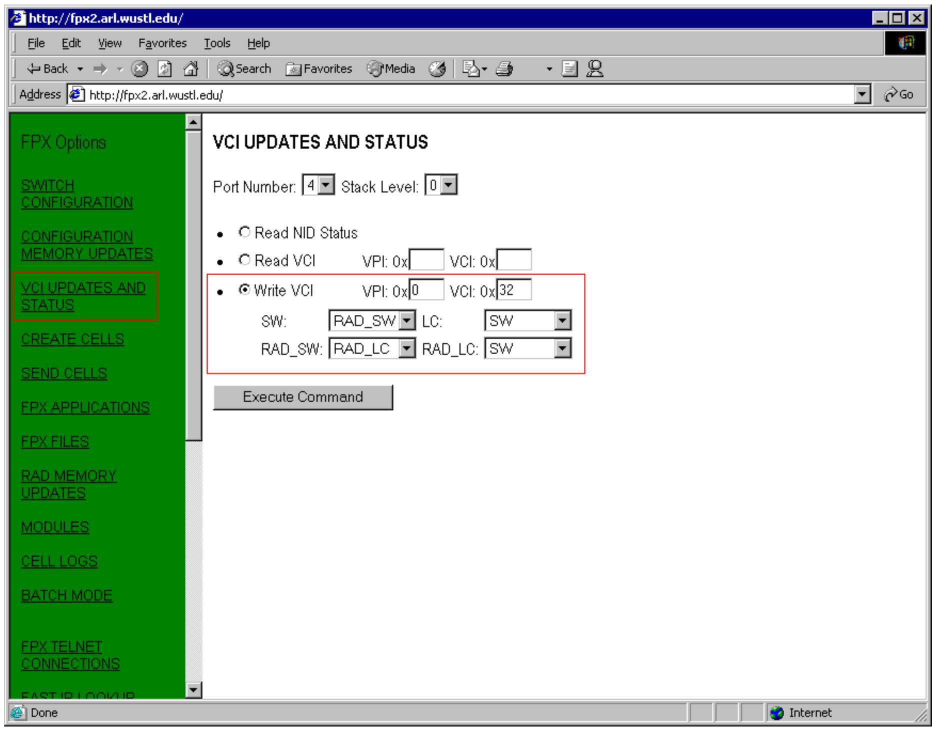
- ○ Load RAD Memory Filename
- ○ Full RAD Configuration Word_offset Byte_count
- ○ Partial RAD Configuration Word_offset Byte_count
- [✓ ● Complete Configuration] Filename rot13_2000.bit
- ○ Display Configuration Offset

File	DocBot Operation	Size	Last Mod
New_File_Name.bit	[Upload], [Delete]	21504	Tue Jan 22 16:19:05 2002
HE-F.bit	[Upload], [Delete]	823523	Wed Nov 28 13:00:06 2001
HE-L1-A.bit	[Upload], [Delete]	56	Wed Jan 16 09:37:17 2002
HE-L1-B.bit	[Upload], [Delete]	367924	Wed Jan 16 09:38:42 2002
HE-L1.bit	[Upload], [Delete]	367980	Wed Jan 16 09:37:32 2002
HE-L2.bit	[Upload], [Delete]	367988	Thu Jan 10 09:36:42 2002

Figure 15: NCHARGE Step 2: Program bitfile for ROT13 FPX Module

7.3 Configuring NID Routing

The NID allows individual traffic flows to be routed between the interfaces of the switch, the line card, and modules on the RAD. A few typical routings are shown in Figure 17. Once the module is loaded on the FPX, the NID needs to be made aware of which cells to send to the RAD to be processed by the newly installed module. This is done through writing to a VC table on the NID. For our ROT13 module, we are interested in encrypting traffic that comes in on VC 0x32. For all of these cells we need to route them through our FPX module loaded on the RAD. From the VCI UPDATES AND STATUS frame, a write command is issued to VCI 0x32. This command takes cells from the switch, passes them to the RAD_SW interface of the RAD to be processed. From the RAD_SW interface the cells are sent to the RAD_LC interface of the RAD. After leaving the RAD_LC the cells are sent back down to the switch. Cells on a different virtual circuits, by default, are not sent to the RAD to be processed. As shown in figure Figure 16 a NID VCI entry has been setup to send cells on VCI 0x32 through both interfaces of the RAD.



VCI UPDATES AND STATUS

Port Number: 4 Stack Level: 0

- Read NID Status
- Read VCI VPI: 0x VCI: 0x
- [✓ Write VCI] VPI: 0x0 VCI: 0x32
  SW: RAD_SW LC: SW
  RAD_SW: RAD_LC RAD_LC: SW

Figure 16: NCHARGE Step 3: Configure NID Routing

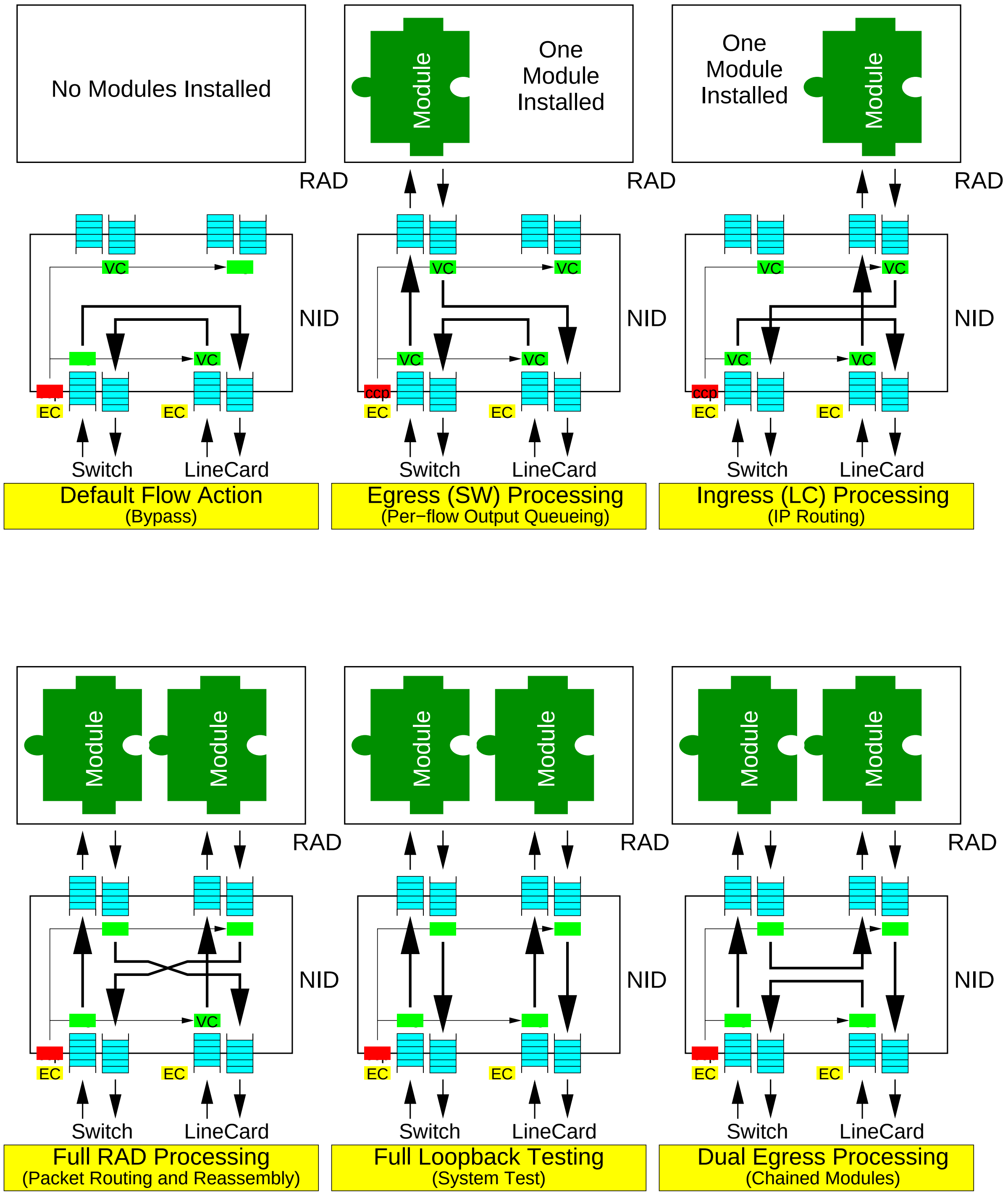


Figure 17: NID Switching Example

7.4 Sending and Receiving Test Data

The NCHARGE suite allows several types of test traffic to be generated on the switch controller. These options can be selected from the *Create Cell* menu, as shown in Figure 18.

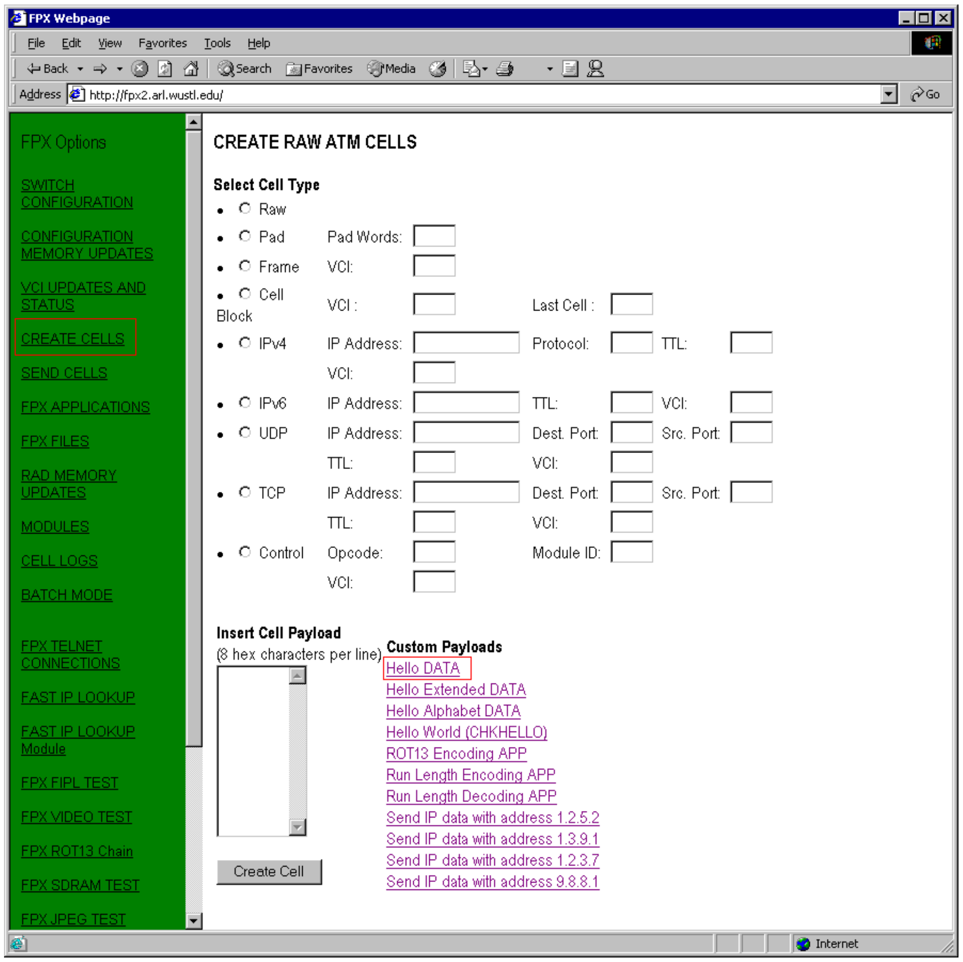
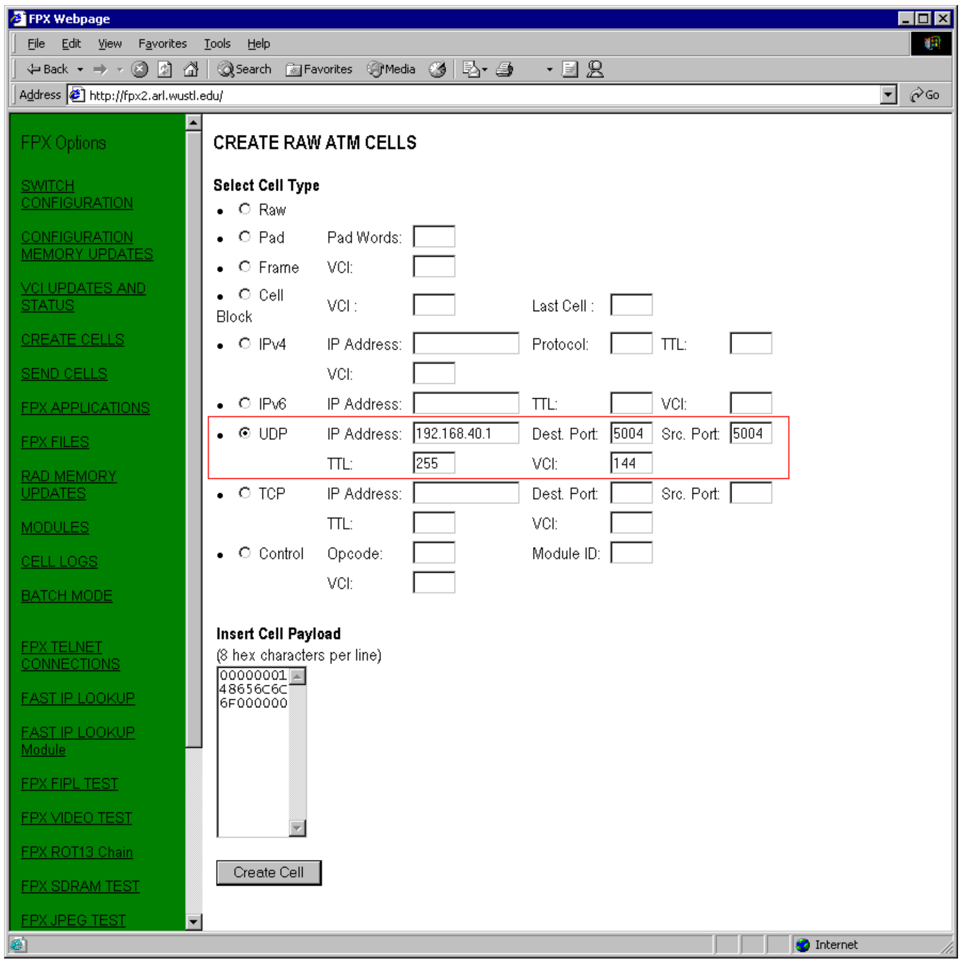


Figure 18: NCHARGE Step 4: Create a test cell



CREATE RAW ATM CELLS

Select Cell Type

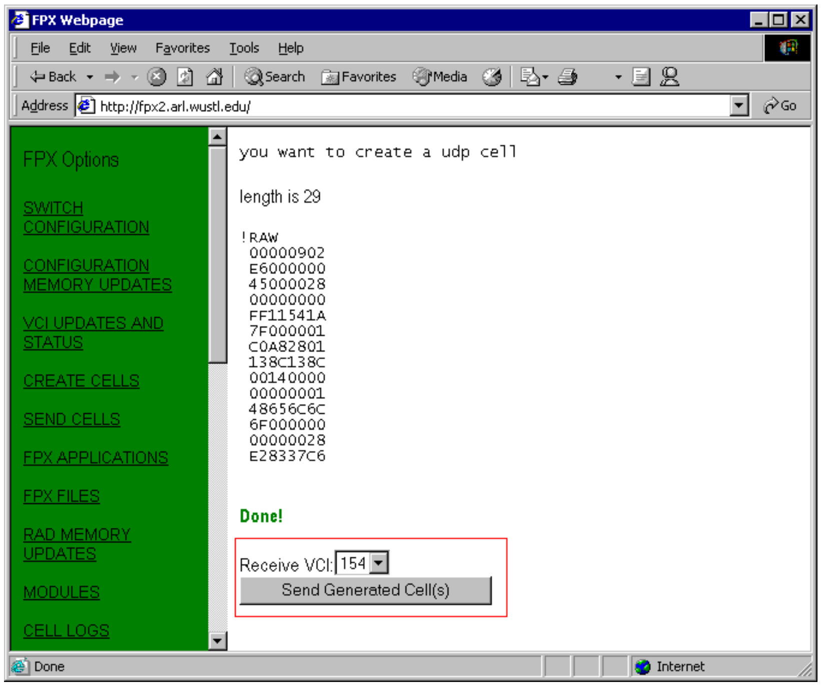
- Raw
- Pad — Pad Words:
- Frame — VCI:
- Cell Block — VCI : — Last Cell :
- IPv4 — IP Address: — Protocol: — TTL: — VCI:
- IPv6 — IP Address: — TTL: — VCI:
- [✓ UDP] — IP Address: 192.168.40.1 — Dest. Port: 5004 — Src. Port: 5004 — TTL: 255 — VCI: 144
- TCP — IP Address: — Dest. Port: — Src. Port: — TTL: — VCI:
- Control — Opcode: — Module ID: — VCI:

Insert Cell Payload
(8 hex characters per line)

```
00000001
48656C6C
6F000000
```

Figure 19: NCHARGE Step 5: Choose the pre-formatted data in the Hello cell

To simplify the process of creating test data, a few pre-formatted cells have been created. One such cell is shown in Figure 19. This cell contains a UDP packet that has been encapsulated into a cell using classical IP-over-ATM. The cell generation software automatically computes the UDP, IP, AAL5, and cell-level checksums. The resulting cell is shown in Figure 19.



```
you want to create a udp cell
```

length is 29

```
!RAW
 00000902
 E6000000
 45000028
 00000000
 FF11541A
 7F000001
 C0A82801
 138C138C
 00140000
 00000001
 48656C6C
 6F000000
 00000028
 E28337C6
```

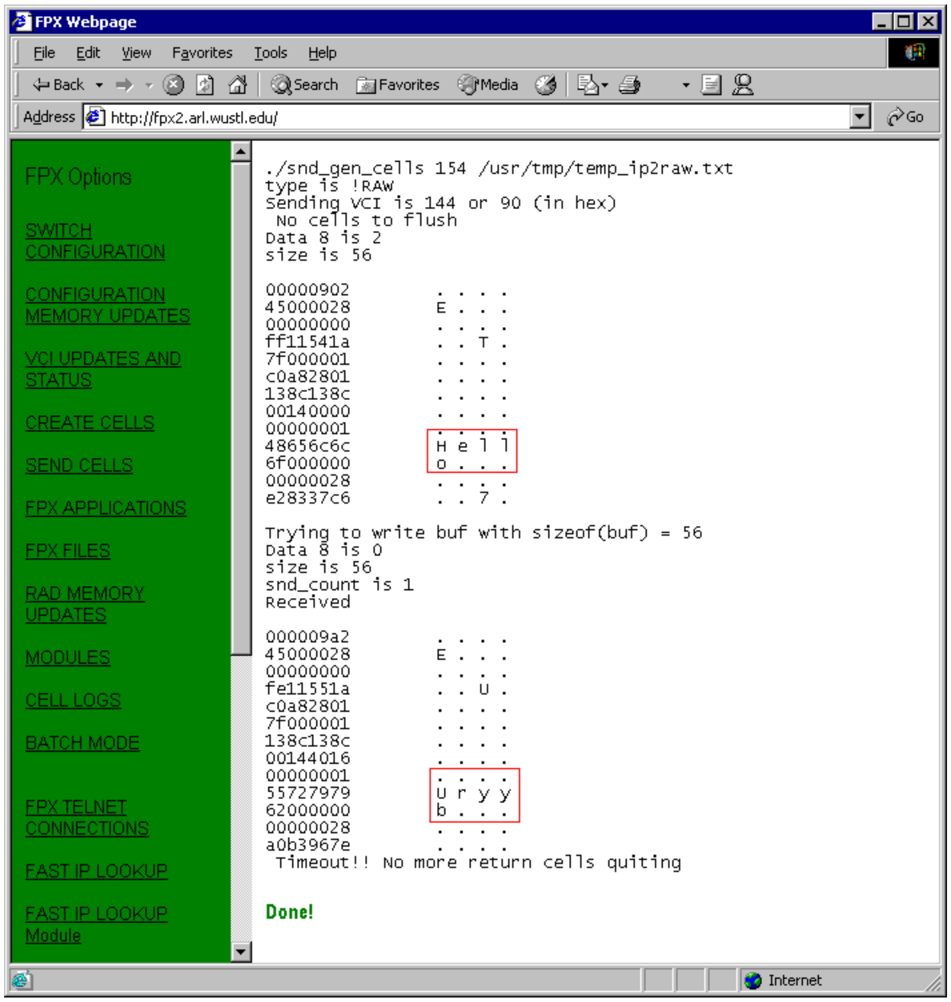
Done!

Receive VCI: 154

Figure 20: NCHARGE Step 6: Send the test cell to the FPX

7.5 Virtual Circuits on the Switch

NCHARGE can also be used to create bidirectional and unidirectional virtual circuits. These circuits can be created from the *Switch Configuration* menu.



```
./snd_gen_cells 154 /usr/tmp/temp_ip2raw.txt
type is !RAW
Sending VCI is 144 or 90 (in hex)
 No cells to flush
Data 8 is 2
size is 56

00000902        . . . .
45000028        E . . .
00000000        . . . .
ff11541a        . . T .
7f000001        . . . .
c0a82801        . . . .
138c138c        . . . .
00140000        . . . .
00000001        . . . .
48656c6c        H e l l
6f000000        o . . .
00000028        . . . .
e28337c6        . . 7 .

Trying to write buf with sizeof(buf) = 56
Data 8 is 0
size is 56
snd_count is 1
Received

000009a2        . . . .
45000028        E . . .
00000000        . . . .
fe11551a        . . U .
c0a82801        . . . .
7f000001        . . . .
138c138c        . . . .
00144016        . . . .
00000001        . . . .
55727979        U r y y
62000000        b . . .
00000028        . . . .
a0b3967e        . . . .
 Timeout!! No more return cells quiting

Done!
```

Figure 21: NCHARGE Step 7: Receive the response from the FPX

8 Applications

Over a dozen applications have been implemented on the FPX. Several applications are included within the distribution to illustrate the function of the FPX.

8.1 Hello, World

An introductory application that demonstrations data processing on the FPX with cells is called *Hello, World*. This application scans for packets on a particular flow (VCI=5) looking for the string “Hello”. When the string is found, it appends the word “World” to the cell [17].

8.2 The ROT13 Encryption Processor

The ROT13 module encrypts or decrypts the string stored in the UDP payload using the ROT13 algorithm. The ROT13 algorithm ”rotates” each character in the string by 13 characters. ’A’ maps to ’N’; ’M’ maps to ’Z’; ’a’ maps to ’n’; and ’m’ maps to ’z’, and so on. Since it rotates each character by 13 and there are 26 characters in the English alphabet, applying the string with the ROT13 algorithm once encrypts the data and doing it again decrypts the data. For example, ”Hello” maps to ”Uryyb”, and ”Uryyb” maps to ”Hello”.

8.3 The KCPSM

The FPX KCPSM Module is a small, active, and reconfigurable processing node for the Field-programmable Port Extender (FPX) using the Layered Protocol Wrappers and the KCPSM. It allows the program memory of the KCPSM to be dynamically reprogrammed over the network through the use of UDP Datagrams; therefore, the function of the processing node can be changed dynamically on a packet by packet basis. It is primarily designed for the FPX, but it can be changed to use in any FPGA-based design[18].

8.4 Fast Internet Protocol Lookup (FIPL)

In the implementation of the *Fast Internet Protocol Lookup (FIPL)*, which is a FPX module that computes the longest matching prefix for Internet addresses, the off-chip SRAM stores the data structures that contain an Internet routing table [5] [19]. Arbitration logic within the RAD allows memory to be shared among multiple modules [4].

8.5 Packet Buffering in SDRAM for Rate Control

There are many ways in which packets can be buffered. The simpliest of all is First-In-First-Out (FIFO). In machine problem four of CS535, a circuit was built that uses the SDRAM to implement a FIFO=based packet buffer for the FPX. The module will accept cells at the input, store them into the FIFO memory, then transmit them to the output in the same order in which they arrived.

In machine problem five, the circuit was augmented with a control cell processor. This circuit allowed an external software process to set the maximum queue length and the pacing rate of the traffic on the outgoing link. It also allowed the size of queue to be monitored by external control cells.

8.6 Other FPX Applications

At the end of the CS535 course, students implemented a number of projects using the FPX. The topics of these projects included:

- Deficit Round Robin (DRR) Scheduler
- Fair Queuing Module
- An Interface to an IDE disk drive
- Circuit to allow IPv6 tunneling over IPv4 networks
- Motion JPEG decoder
- Network Traffic Generator
- Optic Flow Processor
- Audio processor
- Segmentation and Reassembly Circuit
- Tree-based packet filter
- Video Scaling in Reprogrammable Hardware

Details of these projects remain on-line as:
`http://www.arl.wustl.edu/˜lockwood/class/cs535/project/index.html`

9 Revisions

- Version 2.1:
  - NID PROM updated.
  - NCHARGE software opcodes updated to work with new NID PROM.
  - Note that Version 2.0 PROMS will not program with Version 2.1 Software.

10 Technology License for Research and Education Purposes

Washington University (”University”) in St. Louis has developed and is the owner of the copyright for Field-programmable Port Extender (FPX) (”Technology”) which includes switch and network interface hardware, associated software, documentation and detailed specifications, including source code, VHDL specifications of integrated circuits and printed circuit board schematics and which is distributed as part of an open research platform to universities and research organizations.

University grants to select not-for-profit educational institution (”Licensee”) which agree to these terms the right to use its Technology solely for Licensee’s internal nonprofit research and academic purposes. This License is nontransferrable, royalty free, and non-exclusive.

Licensee agrees that it will not use Technology as part of any commercial product. Licensee agrees that it will not rewrite any part of Technology in an alternate form to circumvent the requirement for obtaining a license from University for uses other than internal research and education.

Licensee may reproduce Technology only for educational, research, backup or archival purposes. Licensee agrees not to remove or destroy notices identifying material as the copyrighted or patented property of University. Licensee further agrees to include such notices in all copies it makes of the material.

UNIVERSITY GIVES NO WARRANTIES AND MAKES NO REPRESENTATIONS, EITHER EXPRESS OR IMPLIED, FOR THE DESIGNS AND/OR DOCUMENTATION PROVIDED UNDER THIS AGREEMENT, INCLUDING, WITHOUT LIMITATION, WARRANTY OF MERCHANTABILITY AND/OR WARRANTY OF FITNESS FOR A PARTICULAR USE AND/OR WARRANTY OF FREEDOM OF INFRINGEMENT FROM THIRD PARTY PROPRIETARY RIGHTS.

Licensee understands that Technology is a research tool for which no warranties as to capabilities or accuracy are made, and Licensee accepts Technology ”as is.” University makes no warranties that Technology is free from any harmful code.

Licensee assumes the entire risk as to the performance of Technology and/or associated materials, and to the performance and validity of information generated using Technology. Licensee agrees that University shall not be held liable for any direct, indirect, consequential or incidental damages with respect to any claim by Licensee or any third party on account of or arising from this Agreement or the use of Technology and/or associated materials.

Licensee agrees to accept Technology on an ”as is” basis. Accordingly, University is not required to load Technology onto Licensee’s computers, to test for proper operation, to perform any debugging or other corrections, or otherwise maintain Technology at any time. University is not required to provide any updates to Technology if subsequent versions are developed, or to provide any assistance in understanding or using these materials.

Title to Technology (including copyright), including the original and any copies of Technology which are made hereunder, in whole or in part, shall remain at all times with University. Licensee may not sublicense, sell, assign, or otherwise transfer or allow the transfer of Technology or any portion thereof. Licensee may not reproduce Technology or any portion thereof, except as permitted in Article 2.

References

[1] J. S. Turner, T. Chaney, A. Fingerhut, and M. Flucke, “Design of a Gigabit ATM Switch,” in *INFO-COM’97*, Mar. 1997.

[2] S. Choi, J. Dehart, R. Keller, J. W. Lockwood, J. Turner, and T. Wolf, “Design of a flexible open platform for high performance active networks,” in *Allerton Conference*, (Champaign, IL), 1999.

[3] J. W. Lockwood, “An open platform for development of network processing modules in reprogrammable hardware,” in *IEC DesignCon’01*, (Santa Clara, CA), pp. WB–19, Jan. 2001.

[4] D. E. Taylor, J. W. Lockwood, and N. Naufel, “Generalized RAD Module Interface Specification of the Field-programmable Port eXtender (FPX),” tech. rep., WUCS-01-15, Washington University, Department of Computer Science, July 2001.

[5] J. W. Lockwood, J. S. Turner, and D. E. Taylor, “Field programmable port extender (FPX) for distributed routing and queuing,” in *ACM International Symposium on Field Programmable Gate Arrays (FPGA’2000)*, (Monterey, CA, USA), pp. 137–144, Feb. 2000.

[6] 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, USA), pp. 87–93, Feb. 2001.

[7] E. L. Horta, J. W. Lockwood, D. E. Taylor, and D. Parlour, “Dynamic hardware plugins in an FPGA with partial run-time reconfiguration,” in *Design Automation Conference (DAC)*, (New Orleans, LA), June 2002.

[8] J. W. Lockwood, “Evolvable Internet hardware platforms,” in *The Third NASA/DoD Workshop on Evolvable Hardware (EH’2001)*, (Long Beach, CA), pp. 271–279, July 2001.

[9] D. E. Taylor, J. W. Lockwood, and N. Naufel, “RAD Module Infrastructure of the Field-programmable Port eXtender (FPX),” tech. rep., WUCS-01-16, Washington University, Department of Computer Science, July 2001.

[10] S. Dharmapurikar and J. Lockwood, “Synthesizable design of a multi-module memory controller,” tech. rep., WUCS-01-26, Washington University, Department of Computer Science, Oct. 2001.

[11] F. Braun, J. W. Lockwood, and M. Waldvogel, “Layered protocol wrappers for Internet packet processing in reconfigurable hardware,” in *Proceedings of the Symposium on High Performance Interconnects (HotI-9)*, (Stanford University, CA, USA), p. 4.3, Aug. 2001.

[12] F. Braun, J. W. Lockwood, and M. Waldvogel, “Layered protocol wrappers for Internet packet processing in reconfigurable hardware,” Tech. Rep. WU-CS-01-10, Washington University in Saint Louis, Department of Computer Science, June 2001.

[13] J. W. Lockwood, “Platform and methodology for teaching design of hardware modules in internet routers and firewalls,” in *International Conference on Microelectronic Systems Education (MSE’2001)*, (Las Vegas, NV), June 2001.

[14] D. E. T. Todd Sproull, John W. Lockwood, “Control and configuration software for a reconfigurable networking hardware platform,” in *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, (Napa, CA), Apr. 2002.

[15] W. Westfeldt, “Internet reconfigurable logic for creating web-enabled devices.” Xilinx Xcell, Q1 1999.

[16] S. Kelem, “Virtex configuration architecture advanced user’s guide.” Xilinx XAPP151, Sept. 1999.

[17] J. W. Lockwood and D. Lim, “Hello World: A simple application for the field programmable port extender (FPX),” tech. rep., WUCS-00-12, Washington University in Saint Louis, Department of Computer Science, July 11, 2000.

[18] H. Fu and J. W. Lockwood, “The FPX KCPSM module: An embedded, reconfigurable processing module for the field programmable port extender (FPX),” Tech. Rep. wucs-01-14, Washington University in Saint Louis, July 2001.

[19] D. E. Taylor, J. W. Lockwood, T. S. Sproull, J. S. Turner, and D. B. Parlour, “Scalable IP lookup for programmable routers,” in *IEEE Infocom 2002*, (New York NY), June 2002.