NetFPGA Informational Tutorial

Presented by:
Andrew W. Moore
(University of Cambridge)

Cambridge, UK
September 1st, 2011

http://NetFPGA.org

Welcome

Please organize into teams
2 or 3 People/computer

Printed Slides are available
Slides are also available online

The NetFPGA machines
Username: *root* Password: *on whiteboard*

NetFPGA homepage
http://NetFPGA.org

Tutorial Outline

- **Introduction**
  - Motivation
    - Network Review: Basics of an IP Router
  - Demo 1: Reference Router running on the NetFPGA
- **Exercise 1: Exploring the Reference Router**

10:30 – 11:00 Coffee/Tea break

  - Hardware : NetFPGA Platforms : 1G and 10G
  - Problem: Understanding buffer size requirements in a router
- **Exercise 2: Enhancing the Reference Router**

12:30 – 13:30 Lunch

  - Observing and controlling the queue size
  - NetFPGA Community
    - NetThreads
    - Altera DE4 port
  - NetFPGA in the Classroom
  - Problem: Exploring Controlled packet-loss
- **Exercise 3: Drop 1 in N Packets**

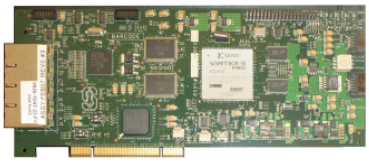
15:00 – 15:30 Coffee/Tea break

- **Concluding Remarks**
  - What next for you?
  - Group Discussion

Motivation

NetFPGA = Networked FPGA

A line-rate, flexible, open networking platform for teaching and research



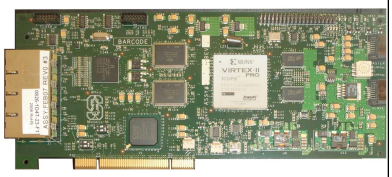
=

- Network Interface Card
- Hardware Accelerated Linux Router
- IPv4 Reference Router
- Traffic Generator
- Openflow Switch
- * More Projects
- ... Add Your Project

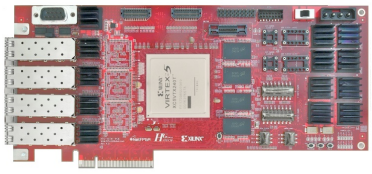
NetFPGA consists of…

Four elements:

- **NetFPGA board**
- **Tools + reference designs**
- **Contributed projects**
- **Community**

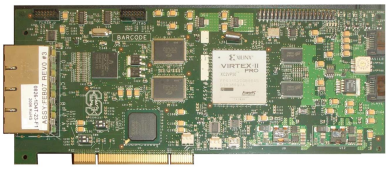


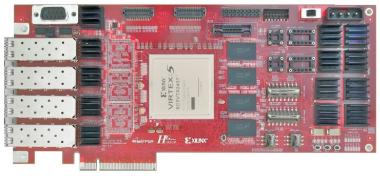
NetFPGA 1G Board



NetFPGA 10G Board

NetFPGA Board Comparison



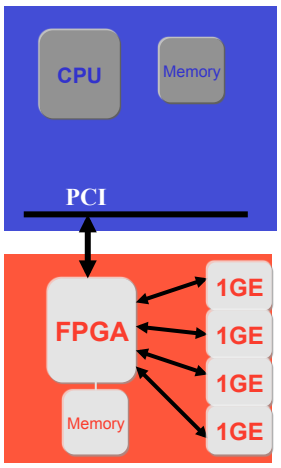


NetFPGA 1G	NetFPGA 10G
4 x 1Gbps Ethernet Ports	4 x 10Gbps SFP+
4.5 MB ZBT SRAM 64 MB DDR2 SDRAM	27 MB QDRII-SRAM 288 MB RLDRAM-II
PCI	PCI Express x8
Virtex II-Pro 50	Virtex 5 TX240T

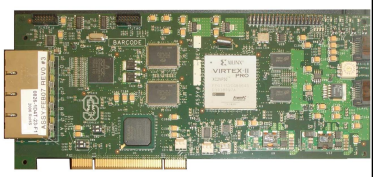
NetFPGA board

Networking Software running on a standard PC

A hardware accelerator built with Field Programmable Gate Array driving Gigabit network links



PC with NetFPGA

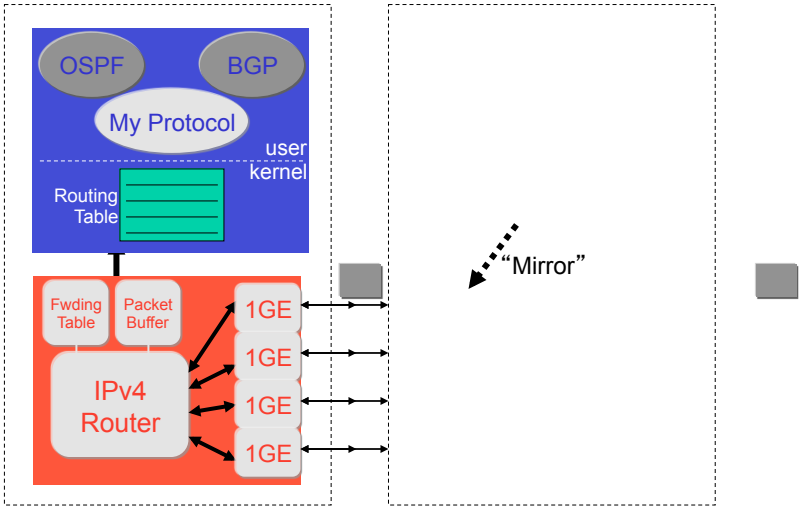


NetFPGA Board

Usage #1

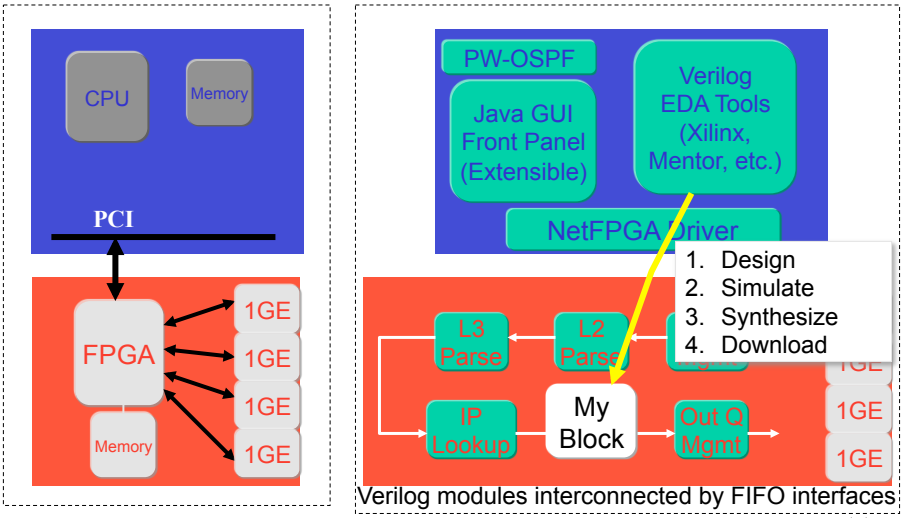
Running the Router Kit

User-space development, 4x1GE line-rate forwarding



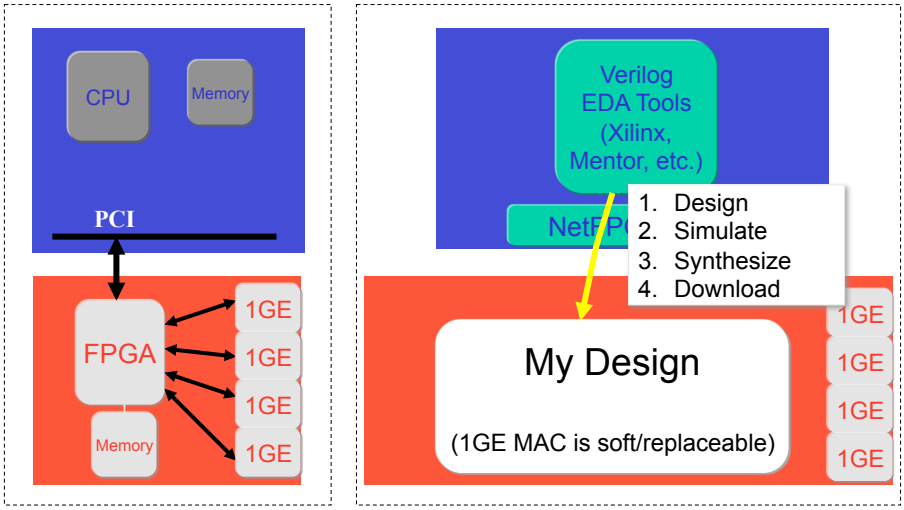
Usage #2

Enhancing Modular Reference Designs



Usage #3

Creating new systems



Tools + Reference Designs 1G

Tools:

- **Compile designs**
- **Verify designs**
- **Interact with hardware**

Reference designs:

- **Router (HW)**
- **Switch (HW)**
- **Network Interface Card (HW)**
- **Router Kit (SW)**
- **SCONE (SW)**

Contributed Projects

Project	Contributor
OpenFlow switch	Stanford University
Packet generator	Stanford University
NetFlow Probe	Brno University
NetThreads	University of Toronto
zFilter (Sp)router	Ericsson
Traffic Monitor	University of Catania
DFA	UMass Lowell

More projects:
http://netfpga.org/foswiki/NetFPGA/OneGig/ProjectTable

Community

Wiki

- **Documentation**
  - User's Guide
  - Developer's Guide
- **Encourage users to contribute**

Forums

- **Support by users for users**
- **Active community - 10s-100s of posts/week**

International Community

Over 1,000 users, using 1,900 cards at 150 universities in 32 countries

NetFPGA's Defining Characteristics

- **Line-Rate**
  - Processes back-to-back packets
    - Without dropping packets
    - At full rate of Gigabit Ethernet Links
  - Operating on packet headers
    - For switching, routing, and firewall rules
  - And packet payloads
    - For content processing and intrusion prevention
- **Open-source Hardware**
  - Similar to open-source software
    - Full source code available
    - BSD-Style License
  - But harder, because
    - Hardware modules must meeting timing
    - Verilog & VHDL Components have more complex interfaces
    - Hardware designers need high confidence in specification of modules

Test-Driven Design

- **Regression tests**
  - Have repeatable results
  - Define the supported features
  - Provide clear expectation on functionality
- ***Example:* Internet Router**
  - Drops packets with bad IP checksum
  - Performs Longest Prefix Matching on destination address
  - Forwards IPv4 packets of length 64-1500 bytes
  - Generates ICMP message for packets with TTL <= 1
  - Defines how packets with IP options or non IPv4

    … and dozens more …

    Every feature is defined by a regression test

Who, How, Why

Who uses the NetFPGA?
- Teachers
- Students
- Researchers

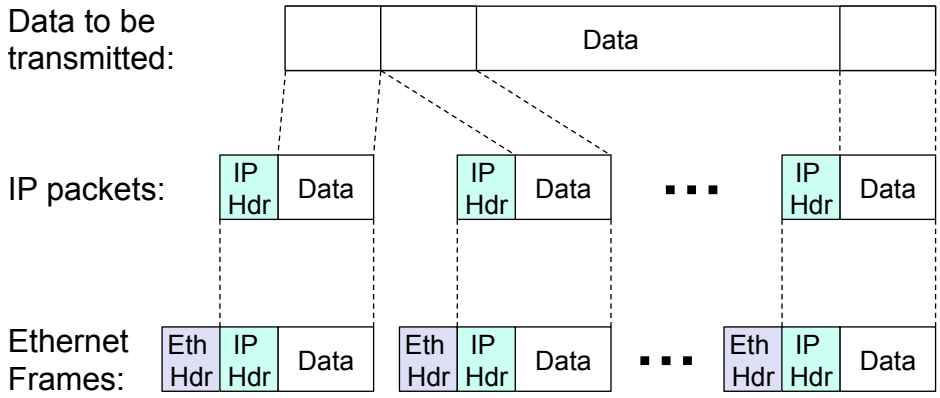
How do they use the NetFPGA?
- To run the Router Kit
- To build modular reference designs
  - IPv4 router
  - 4-port NIC
  - Ethernet switch, …

Why do they use the NetFPGA?
- To measure performance of Internet systems
- To prototype new networking systems

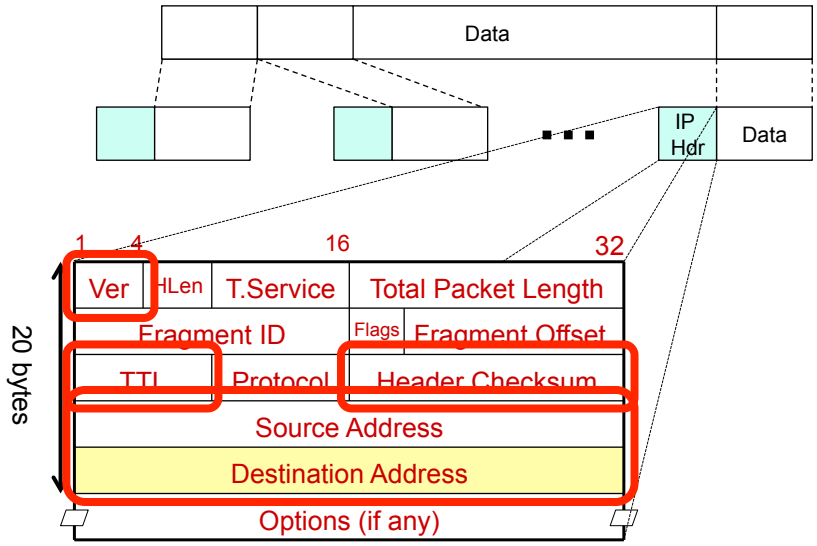
Lets have an example, but first….

Network review

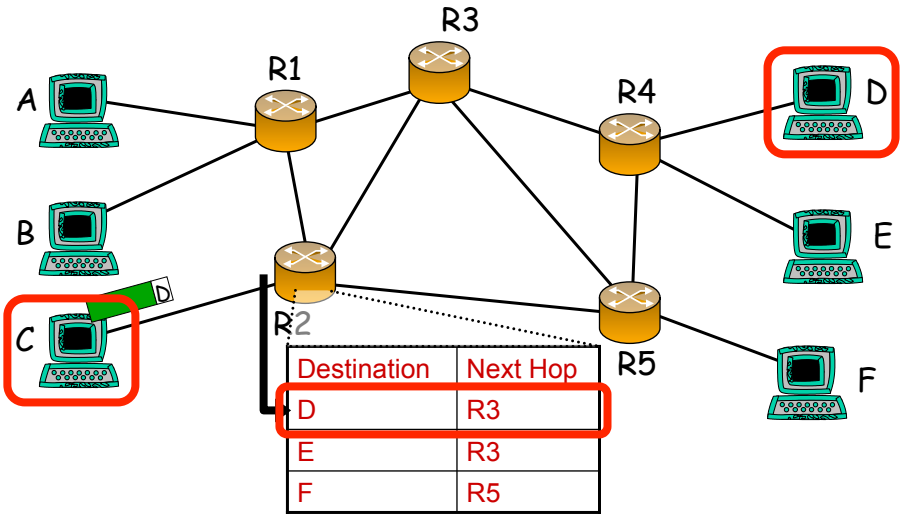
Internet Protocol (IP)



Internet Protocol (IP)

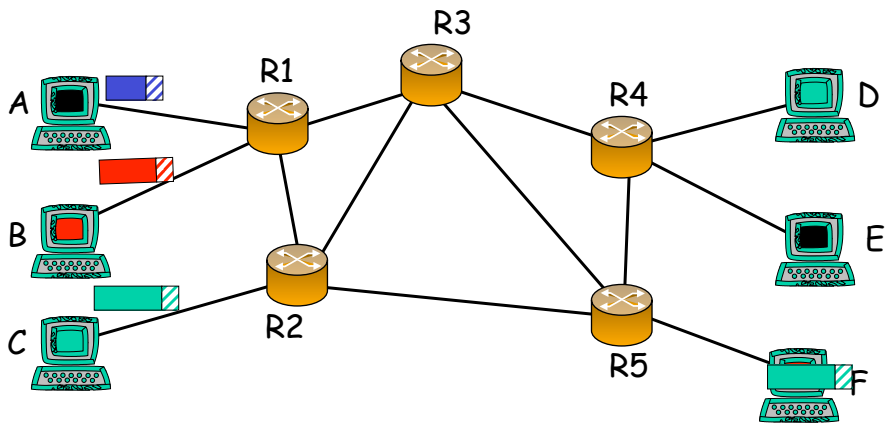


Basic operation of an IP router



Destination	Next Hop
D	R3
E	R3
F	R5

Basic operation of an IP router



Forwarding tables

IP address } 32 bits wide → ~ 4 billion unique address

Naïve approach:
One entry per address

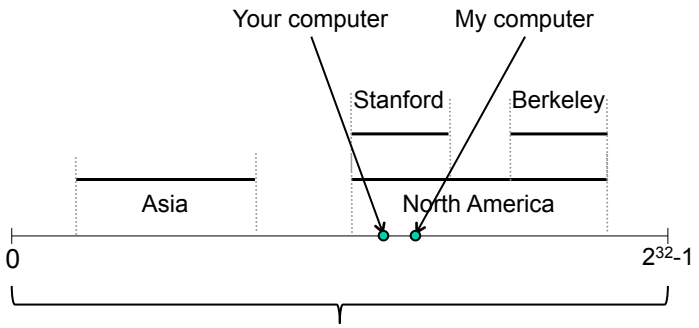
Entry	Destination	Port
1	0.0.0.0	1
2	0.0.0.1	2
⋮	⋮	⋮
2^{32}	255.255.255.255	12

} **~ 4 billion entries**

Improved approach:
Group entries to reduce table size

Entry	Destination	Port
1	0.0.0.0 – 127.255.255.255	1
2	128.0.0.1 – 128.255.255.255	2
⋮	⋮	⋮
50	248.0.0.0 – 255.255.255.255	12

IP addresses as a line



Entry	Destination	Port
1	Stanford	1
2	Berkeley	2
3	North America	3
4	Asia	4
5	Everywhere (default)	5

Longest Prefix Match (LPM)

Entry	Destination	Port
1	Stanford	1
2	Berkeley	2
3	North America	3
4	Asia	4
5	Everywhere (default)	5

Universities

Continents

Planet

Matching entries:

- Stanford — Most specific
- North America
- Everywhere

To: Stanford	Data

Longest Prefix Match (LPM)

Entry	Destination	Port
1	Stanford	1
2	Berkeley	2
3	North America	3
4	Asia	4
5	Everywhere (default)	5

Universities

Continents

Planet

Matching entries:

- North America — Most specific
- Everywhere

To: Canada	Data

Implementing Longest Prefix Match

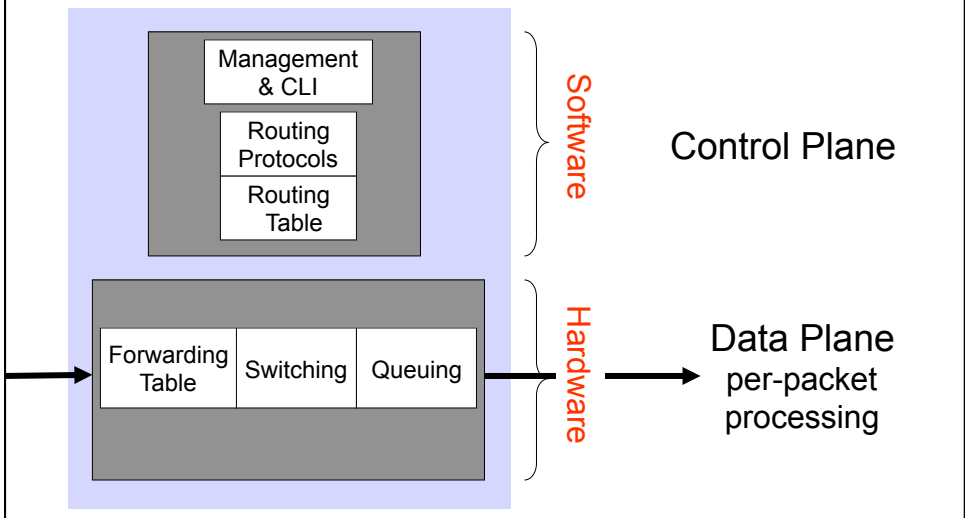
Entry	Destination	Port	
1	Stanford	1	Searching
2	Berkeley	2	
3	North America	3	
4	Asia	4	FOUND
5	Everywhere (default)	5	

Most specific

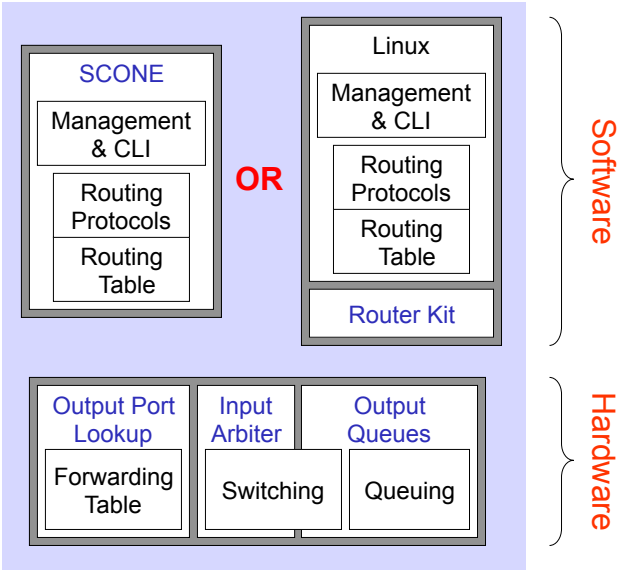
↓

Least specific

Basic components of an IP router



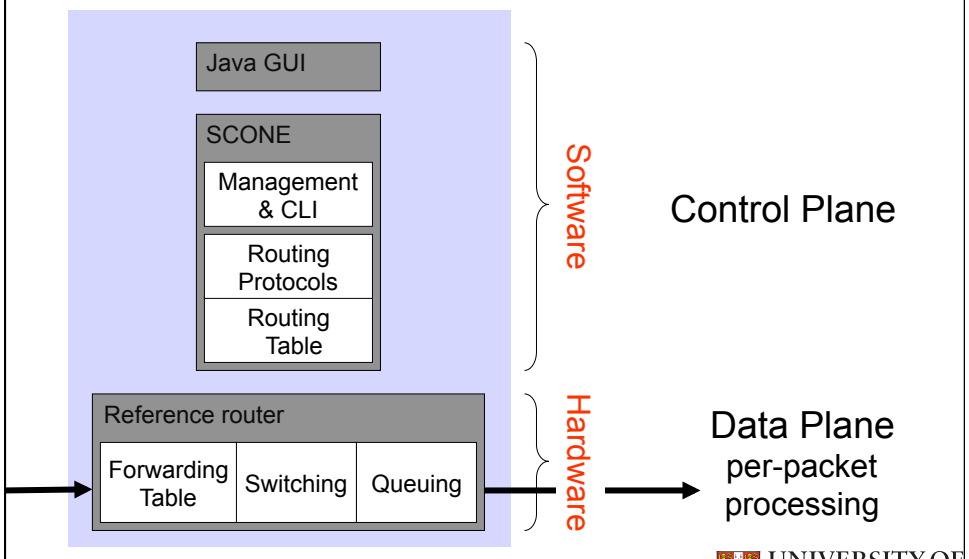
IP router components in NetFPGA



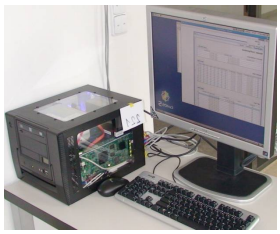
Example I

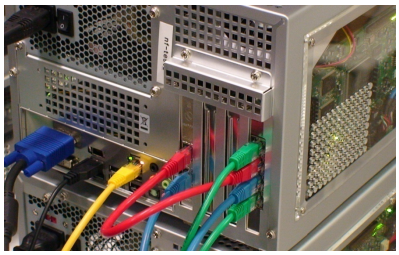
Reference Router running on the NetFPGA

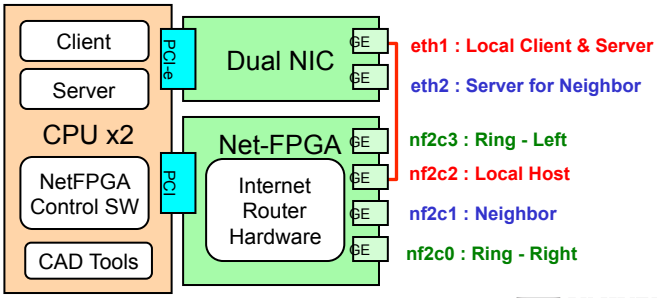
Operational IPv4 router



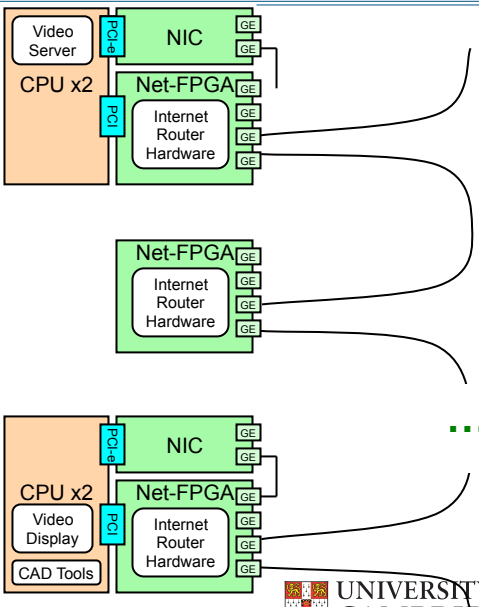
NetFPGA Lab Setup







NetFPGA Hardware Set for Demo #1



Server delivers streaming HD video through a chain of NetFPGA Routers

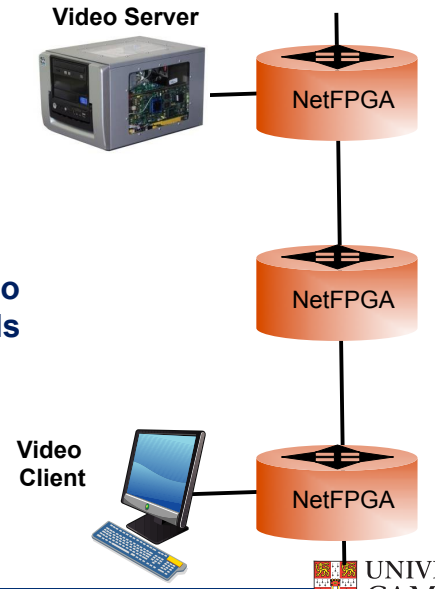
Demo 1

Setup for the Reference Router

Each NetFPGA card has four ports

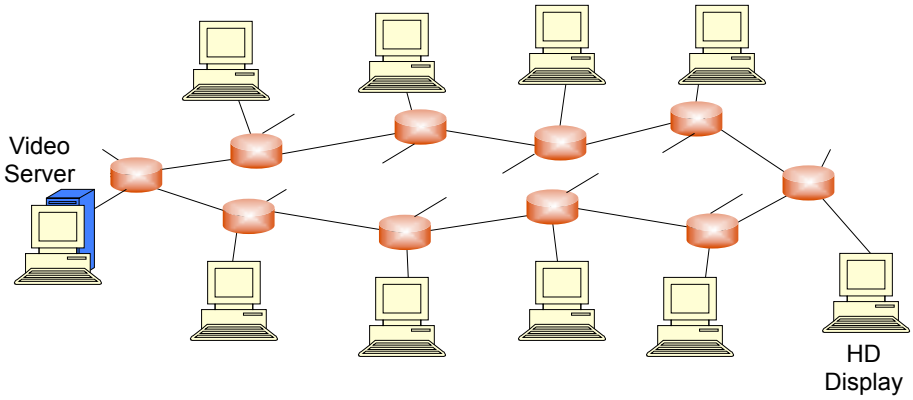
Port 2 connected to Client / Server

Ports 0 and 3 connected to adjacent NetFPGA cards



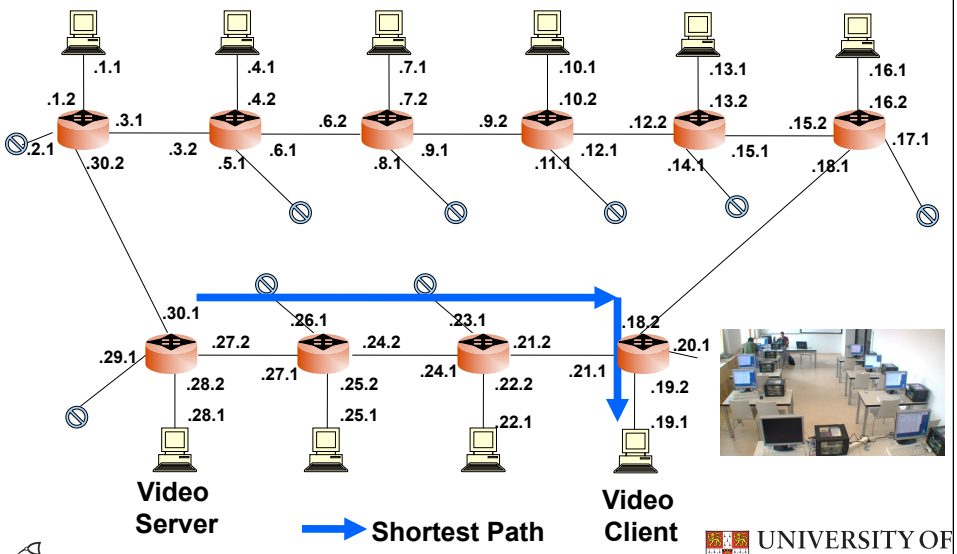
Demo 1

Topology of NetFPGA Routers



Demo 1

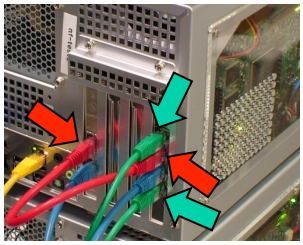
Subnet Configuration

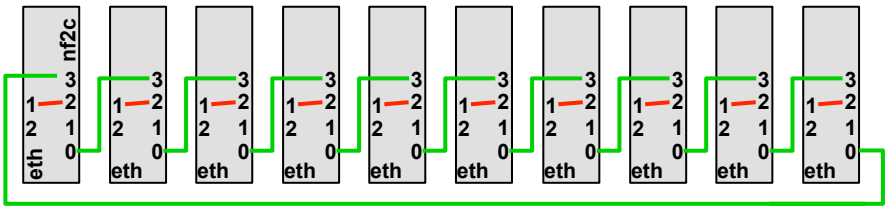


Demo 1

Cable Configuration for Demo 1

- **NetFPGA Gigabit Ethernet Interfaces**
  - nf2c3 : Left neighbor in network (green)
  - nf2c2 : Local host interface (red)
  - nf2c0 : Right neighbor in network (green)
- **Host Ethernet Interfaces**
  - eth1 : Local host interface (red)





Review

NetFPGA as IPv4 router:
- **Reference hardware + SCONE software**
- **Routing protocol discovers topology**

Demo:
- **Ring topology**
- **Traffic flows over shortest path**
- **Broken link: automatically route around failure**

Demo 1

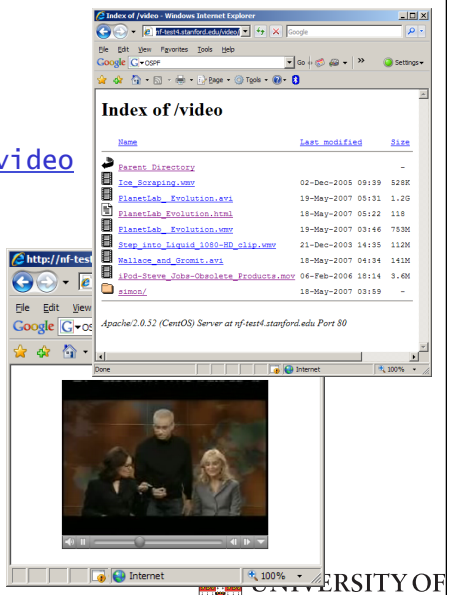
Working IP Router

- **Objectives**
  - Become familiar with Stanford Reference Router
  - Observe PW-OSPF re-routing traffic around a failure

Demo 1

Streaming Video through the NetFPGA

- **Video server**
  - Source files
    `/var/www/html/video`
  - Network URL :
    `http://192.168.Net.Host/video`
- **Video client**
  - Windows Media Player
  - Linux `mplayer`
- **Video traffic**
  - MPEG2 HDTV (35 Mbps)
  - MPEG2 TV (9 Mbps)
  - DVI (3 Mbps)
  - WMF (1.7 Mbps)



Demo 1 Physical Configuration

Key:

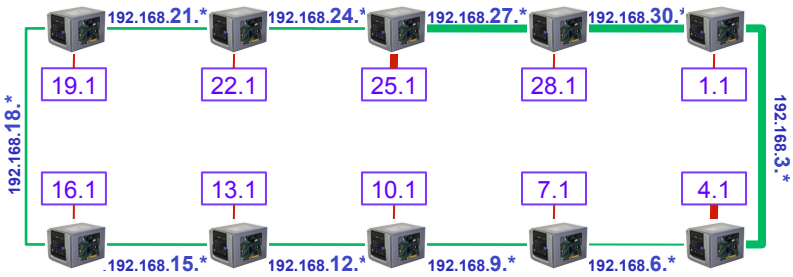
eth1 of Host PC
192.168.X.Y

NetFPGA Router #

E.G. To stream video from server 4.1, type:

```
cd ~/NF2/projects/tutorial_router/sw
./play 192.168.4.1
```

Any PC can stream traffic through multiple NetFPGA routers in the ring topology to any other PC



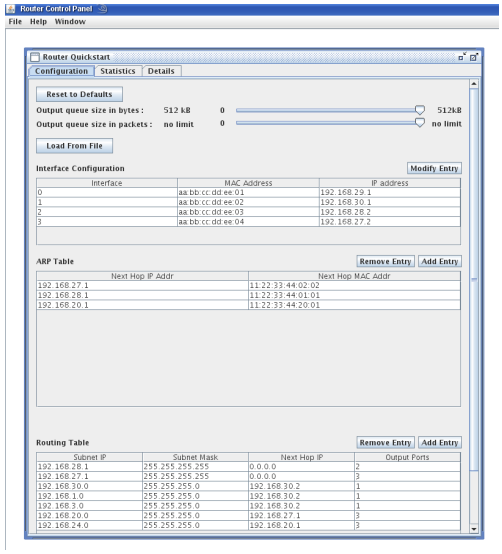
Demo 1

Step 1 – Observe the Routing Tables

The router is already configured and running on your machines

The routing table has converged to the routing decisions with minimum number of hops

Next, break a link …

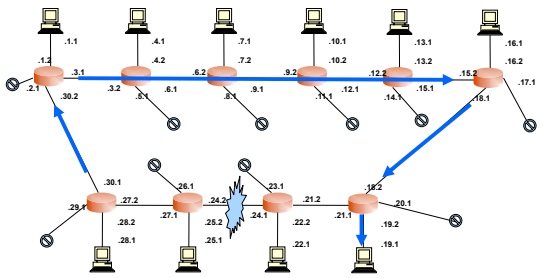


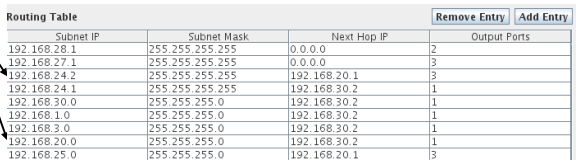
Demo 1

Step 2 - Dynamic Re-routing

Break the link between video server and video client

Routers re-route traffic around the broken link and video continues playing





Routing Table

Remove Entry Add Entry

Subnet IP	Subnet Mask	Next Hop IP	Output Ports
192.168.28.1	255.255.255.255	0.0.0.0	2
192.168.27.1	255.255.255.255	0.0.0.0	3
192.168.24.2	255.255.255.255	192.168.20.1	3
192.168.24.1	255.255.255.255	192.168.30.2	1
192.168.30.0	255.255.255.0	192.168.30.2	1
192.168.1.0	255.255.255.0	192.168.30.2	1
192.168.3.0	255.255.255.0	192.168.30.2	1
192.168.20.0	255.255.255.0	192.168.30.2	1
192.168.25.0	255.255.255.0	192.168.20.1	3

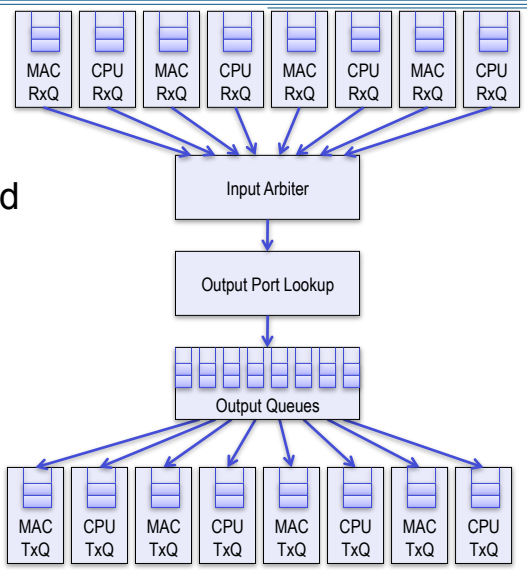
Exercise 1

Explore the Reference Router

Exercise 1

Reference Router Pipeline

- **Five stages**
  - Input
  - Input arbitration
  - Routing decision and packet modification
  - Output queuing
  - Output
- **Packet-based module interface**
- **Pluggable design**



Exercise 1

Exploring the Reference Router

Objectives:

- Run the software
- Explore router architecture

Execution

- Run the GUI
- Test connectivity and statistics with pings
- Explore pipeline in the details page
- Explore detailed statistics in the details page

Exercise 1

Step 2 - Run Reference Router

In a new terminal window

```
cd ~/NF2/projects/tutorial_router/sw
```

To use the router hardware, type:

```
./tut_router_gui.pl --use_bin \
 ../../../bitfiles/tutorial_router.bit
```

To stream video, run (in a new terminal)

```
cd ~/NF2/projects/tutorial_router/sw
./mp 192.168.X.Y
```

where ***X***.***Y*** = 25.1 or 19.1 or 7.1

Open `firefox` for the full list of host IP address

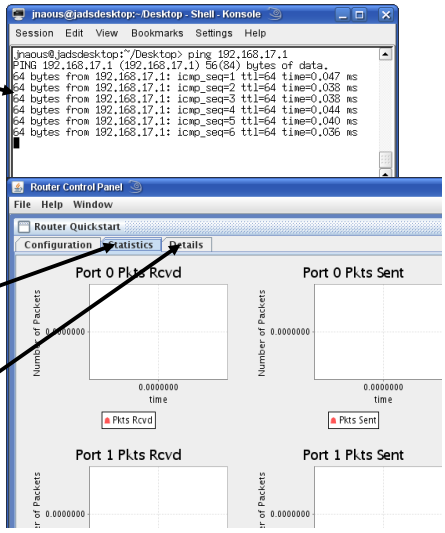
Exercise 1

Step 4 - Connectivity and Statistics

Ping any addresses 192.168.x.y where x is from 1-20 and y is 1 or 2

Open the statistics tab in the Quickstart window to see some statistics

Explore more statistics in modules under the details tab

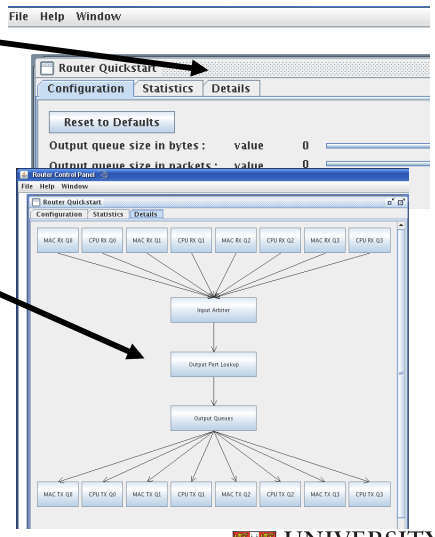


Exercise 1

Step 5 - Explore Router Architecture

Click the Details tab of the Quickstart window

This is the reference router pipeline – a canonical, simple-to-understand, modular router pipeline



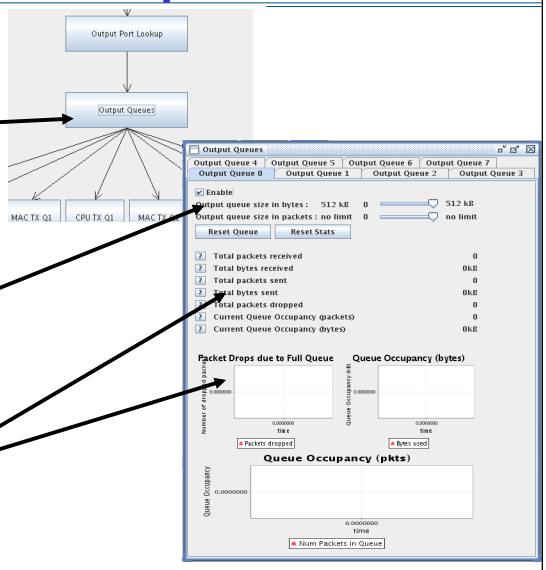
Exercise 1

Step 6 - Explore Output Queues

Click on the Output Queues module in the Details tab

The page gives configuration details

…and statistics

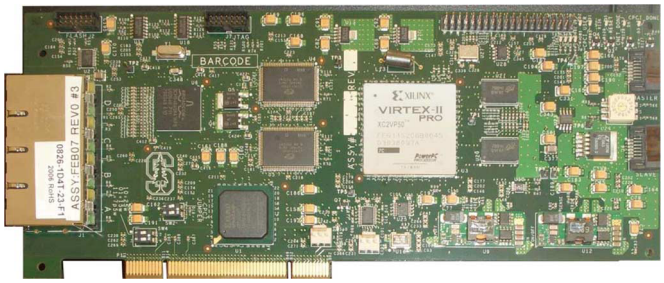


First Break

(examine the running router, watch *Wallace and Gromit*, or get tea/coffee)

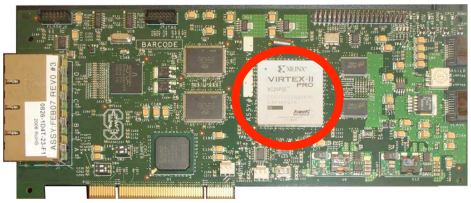
Hardware Overview

NetFPGA-1G



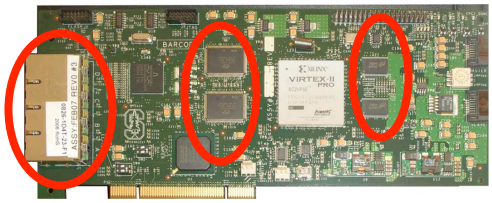
Xilinx Virtex II Pro 50

- **53,000 Logic Cells**
- **Block RAMs**
- **Embedded PowerPC**



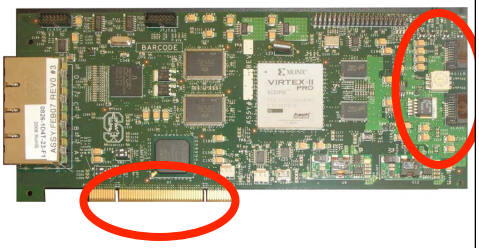
Network and Memory

- **Gigabit Ethernet**
  - 4 RJ45 Ports
  - Broadcom PHY
- **Memories**
  - 4.5MB Static RAM
  - 64MB DDR2 Dynamic RAM



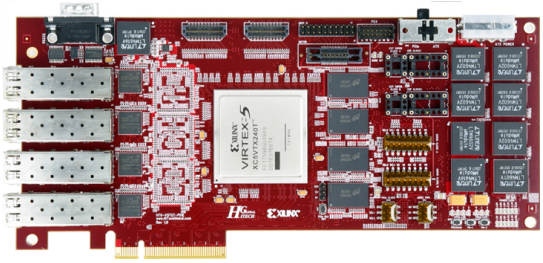
Other IO

- **PCI**
  - Memory Mapped Registers
  - DMA Packet Transferring
- **SATA**
  - Board to Board communication



NetFPGA-10G

- **A major upgrade**
- **State-of-the-art technology**

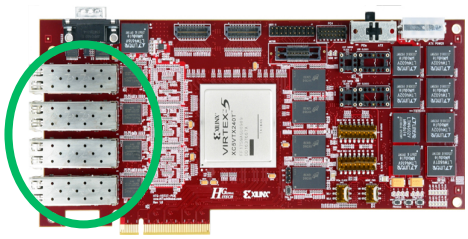


Comparison

NetFPGA 1G	NetFPGA 10G
4 x 1Gbps Ethernet Ports	4 x 10Gbps SFP+
4.5 MB ZBT SRAM 64 MB DDR2 SDRAM	27 MB QDRII-SRAM 288 MB RLDRAM-II
PCI	PCI Express x8
Virtex II-Pro 50	Virtex 5 TX240T

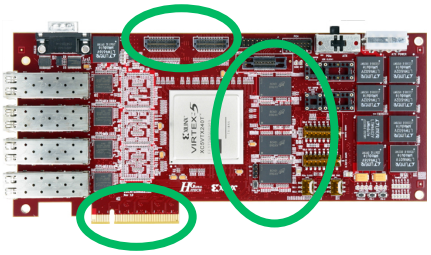
10 Gigabit Ethernet

- **4 SFP+ Cages**
- **AEL2005 PHY**
- **10G Support**
  - Direct Attach Copper
  - 10GBASE-R Optical Fiber
- **1G Support**
  - 1000BASE-T Copper
  - 1000BASE-X Optical Fiber



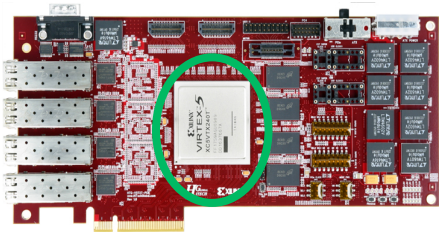
Others

- **QDRII-SRAM**
  - 27MB
  - Storing routing tables, counters and statistics
- **RLDRAM-II**
  - 288MB
  - Packet Buffering
- **PCI Express x8**
  - PC Interface
- **Expansion Slot**

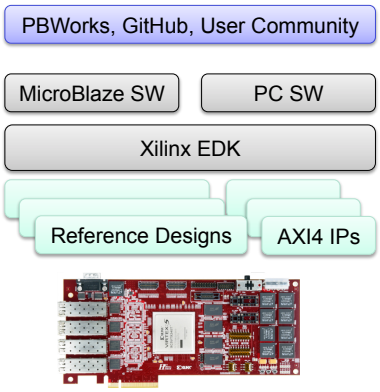


Xilinx Virtex 5 TX240T

- **Optimized for ultra high-bandwidth applications**
- **48 GTX Transceivers**
- **4 hard Tri-mode Ethernet MACs**
- **1 hard PCI Express Endpoint**



Beyond Hardware

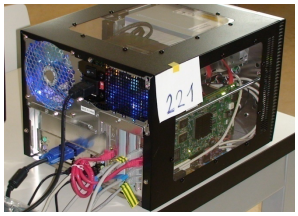


- NetFPGA-10G Board
- Xilinx EDK based IDE
- Reference designs with ARM AXI4
- Software (embedded and PC)
- Public Repository (GitHub)
- Public Wiki (PBWorks)

NetFPGA-1G Cube Systems

- PCs assembled from parts
  - Stanford University
  - Cambridge University
- Pre-built systems available
  - Accent Technology Inc.
- Details are in the Guide

  http://netfpga.org/static/guide.html



Rackmount NetFPGA-1G Servers

2U Server
(Dell 2950)

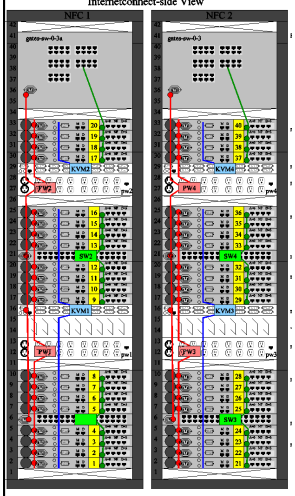
NetFPGA inserts in
PCI or PCI-X slot

1U Server
(Accent Technology Inc.)

Thanks: Brian Cashman for providing machine

Stanford NetFPGA-1G Cluster

Stanford NetFPGA Cluster (NFC)
Internetconnect-side View



Statistics

- Rack of 40
  - 1U PCs with NetFPGAs
- Managed
  - Power
  - Console
  - LANs
- Provides 4*40=160 Gbps of full line-rate processing bandwidth

Understanding Buffer Size Requirements in a Router

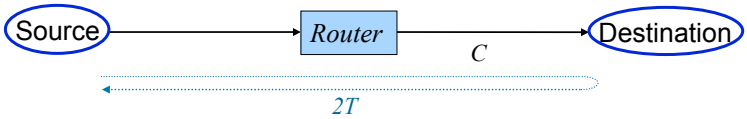
Buffer Requirements in a Router

Buffer size matters:
- Small queues reduce delay
- Large buffers are expensive

Theoretical tools predict requirements
- Queuing theory
- Large deviation theory
- Mean field theory

Yet, there is no direct answer
- Flows have a closed-loop nature
- Question arises on whether focus should be on equilibrium state or transient state

Rule-of-thumb



- **Universally applied rule-of-thumb:**
  - A router needs a buffer size: $B = 2T \times C$
  - 2T is the two-way propagation delay (or just 250ms)
  - C is capacity of bottleneck link
- **Context**
  - Mandated in backbone and edge routers
  - Appears in RFPs and IETF architectural guidelines
  - Already known by inventors of TCP
    - [Van Jacobson, 1988]
  - Has major consequences for router design

The Story So Far

# packets at 10Gb/s	1,000,000	10,000	20

$$2T \times C \xrightarrow{(1)} \frac{2T \times C}{\sqrt{n}} \xrightarrow{(2)} O(\log W)$$

(1) Assume: Large number of desynchronized flows; 100% utilization
(2) Assume: Large number of desynchronized flows; <100% utilization

Why 2TxC for a single TCP Flow?

Only W packets may be outstanding

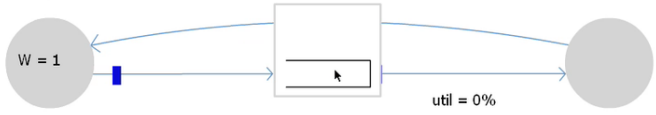
Rule for adjusting W
- If an ACK is received: W ← W+1/W
- If a packet is lost: W ← W/2

Why 2TxC for a single TCP Flow?

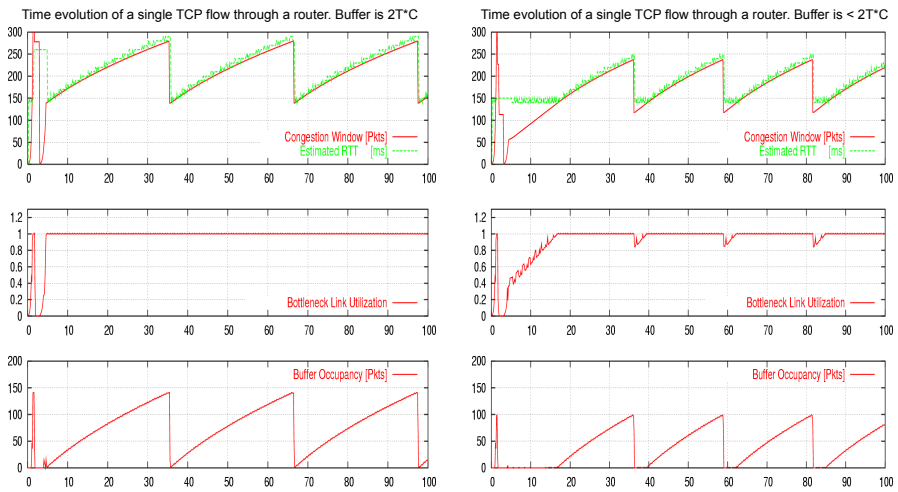
Continuous ARQ (TCP) adapting to congestion

Only W packets may be outstanding

Rule for adjusting W
- If an ACK is received: W ← W+1/W
- If a packet is lost: W ← W/2



Time Evolution of a Single TCP Flow



Using NetFPGA to explore buffer size

- **Need to reduce buffer size and measure occupancy**
- **Alas, not possible in commercial routers**
- **So, we will use the NetFPGA instead**

Objective:

– Use the NetFPGA to understand how large a buffer we need for a **single** TCP flow.

Exercise 2: Enhancing the Reference Router

Exercise 2

Enhance Your Router

Objectives

- Add new modules to datapath
- Synthesize and test router

Execution

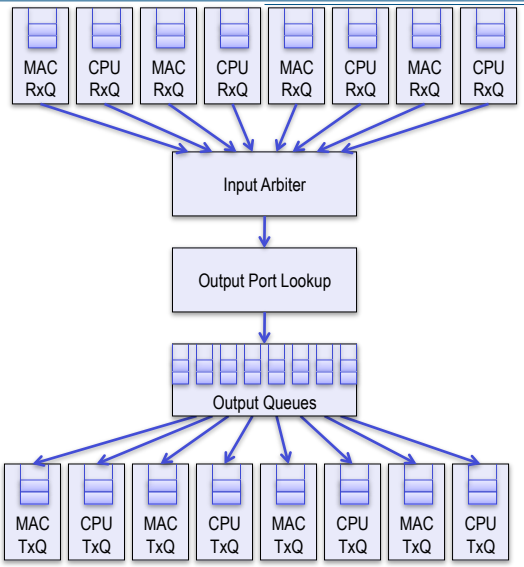
- Open user_datapath.v, uncomment delay/ rate/event capture modules
- Synthesize
- After synthesis, test the new system

Exercise 2

Reference Router Pipeline

We need to add two modules

1. **Event Capture** to capture output queue events (writes, reads, drops)
2. **Rate Limiter** to create a bottleneck

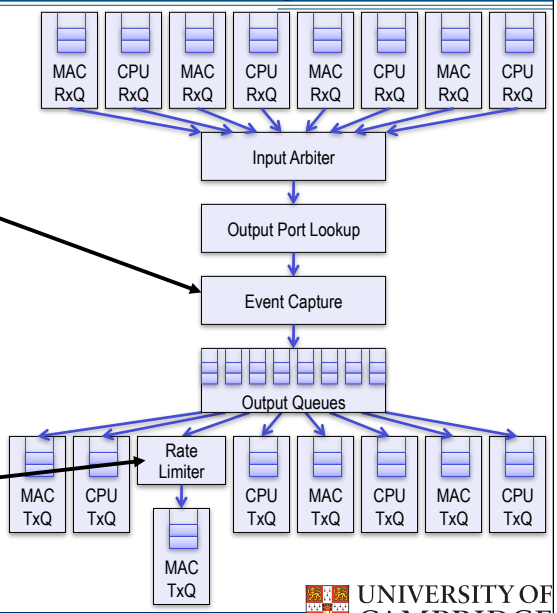


Exercise 2

Enhanced Router Pipeline

We need to add two modules

1. **Event Capture** to capture output queue events (writes, reads, drops)
2. **Rate Limiter** to create a bottleneck



An aside: emacs Tips

We will modify Verilog source code with emacs

- To undo a command, type
  - ctrl+shift+'-'
- To cancel a multi-keystroke command, type
  - ctrl+g
- To select lines,
  - hold shift and press the arrow keys
- To comment (remove from compilation) selected lines, type
  - ctrl+c+c
- To uncomment a commented block,
  - move the cursor inside the commented block
  - type ctrl+c+u
- To save, type
  - ctrl+x+s
- To search for a term, type
  - ctrl+s search_pattern

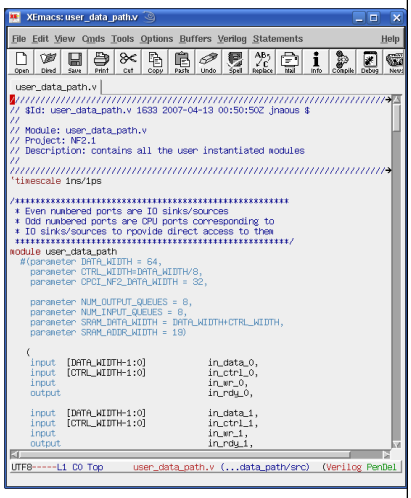
Exercise 2

Step 1 - Open the Source

We will modify the Verilog source code to add event capture and rate limiter modules

We will simply comment and uncomment existing code

Open terminal



Type
emacs NF2/projects/tutorial_router/src/user_data_path.v

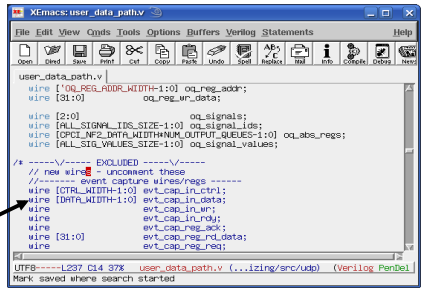
Exercise 2

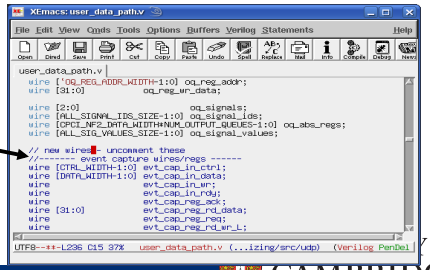
Step 2 - Add Wires

Now we need to add wires to connect the new modules

Search for “new wires” (ctrl +s new wires), then press Enter

Uncomment the wires (ctrl +c+u)





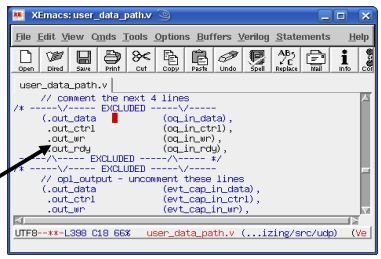
Exercise 2

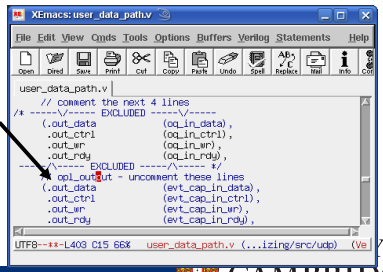
Step 3a - Connect Event Capture

Search for opl_output (ctrl+s opl_output), then press Enter

Comment the four lines above (up, shift + up + up + up + up, ctrl+c+c)

Uncomment the block below to connect the outputs (ctrl+s opl_out, ctrl+c+u)





Exercise 2

Step 3b - Connect the Output Queue Registers

Search for opl_output (ctrl+s opl_output, Enter)

Comment the 6 lines (select the six lines by using shift+arrow keys, then type ctrl+c+c)

Uncomment the commented block by scrolling down into the block and typing ctrl+c +u

```
emacs: user_data_path.v
File Edit View Cmds Tools Options Buffers Verilog Statements Help
user_data_path.v
    // --- Interface to the input arbiter
    .in_data                (op_lut_in_data),
    .in_ctrl                (op_lut_in_ctrl),
    .in_wr                  (op_lut_in_wr),
    .in_rdy                 (op_lut_in_rdy),

    // --- Register interface
    .reg_req_in             (op_lut_in_reg_req),
    .reg_ack_in             (op_lut_in_reg_ack),
    .reg_rd_wr_L_in         (op_lut_in_reg_rd_wr_L),
    .reg_addr_in            (op_lut_in_reg_addr),
    .reg_data_in            (op_lut_in_reg_data),
    .reg_src_in             (op_lut_in_reg_src),

   // comment the next 6 lines
    .reg_req_out            (oq_in_reg_req),
    .reg_ack_out            (oq_in_reg_ack),
    .reg_rd_wr_L_out        (oq_in_reg_rd_wr_L),
    .reg_addr_out           (oq_in_reg_addr),
    .reg_data_out           (oq_in_reg_data),
    .reg_src_out            (oq_in_reg_src),

   // opl_output - uncomment these lines
/* -----\/----- EXCLUDED -----\/-----
    .reg_req_out            (evt_cap_in_reg_req),
    .reg_ack_out            (evt_cap_in_reg_ack),
    .reg_rd_wr_L_out        (evt_cap_in_reg_rd_wr_L),
    .reg_addr_out           (evt_cap_in_reg_addr),
    .reg_data_out           (evt_cap_in_reg_data),
    .reg_src_out            (evt_cap_in_reg_src),
 -----/\----- EXCLUDED -----/\----- */

    // --- Misc
    .clk                    (clk),
    .reset                  (reset));

  // uncomment the module here
u-----XEmacs: user_data_path.v      (Verilog Font)----50%--
Not over a window.
```


Exercise 2

Step 4 - Add the Event Capture Module

Search for evt_capture_top (ctrl+s evt_capture_top), then press Enter

Uncomment the block (ctrl +c+u)

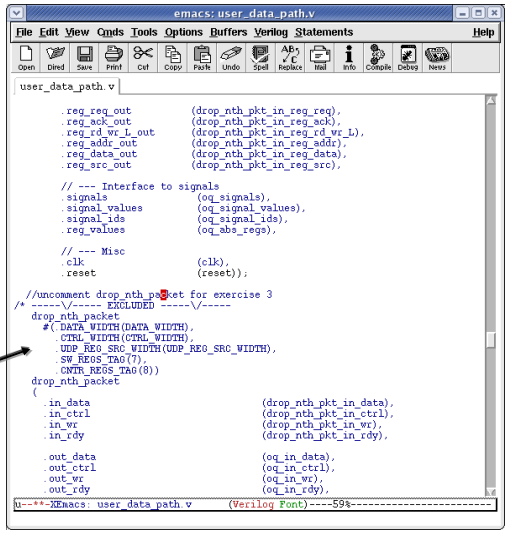
```
XEmacs: user_data_path.v
File Edit View Cmds Tools Options Buffers Verilog Statements Help
user_data_path.v
    // --- Misc
    .clk                    (clk),
    .reset                  (reset));

  // uncomment the module here
  evt_capture_top
    #(.DATA_WIDTH(DATA_WIDTH),
      .CTRL_WIDTH(CTRL_WIDTH),
      .NUM_ABS_REG_PAIRS(NUM_OUTPUT_QUEUES/2),
      .NUM_MONITORED_SIGS(3),
      .SIG_VALUE_SIZE(SIG_VALUE_SIZE),
      .SIGNAL_ID_SIZE(SIGNAL_ID_SIZE),
      .OP_LUT_STAGE_NUM(OP_LUT_STAGE_NUM),
      .CPCI_NF2_DATA_WIDTH(CPCI_NF2_DATA_WIDTH))
    evt_capture_top
      (// --- Interface to next module
       .out_data                (oq_in_data),
       .out_ctrl                (oq_in_ctrl),
       .out_wr                  (oq_in_wr),
       .out_rdy                 (oq_in_rdy),

       // --- Interface to previous module
       .in_data                 (evt_cap_in_data),
       .in_ctrl                 (evt_cap_in_ctrl),
       .in_wr                   (evt_cap_in_wr),
       .in_rdy                  (evt_cap_in_rdy),

       // --- Register interface
       .evt_cap_reg_req         (evt_cap_reg_req),
       .evt_cap_reg_rd_wr_L     (evt_cap_reg_rd_wr_L),
       .evt_cap_reg_addr        (evt_cap_reg_addr),
       .evt_cap_reg_wr_data     (evt_cap_reg_wr_data),
       .evt_cap_reg_rd_data     (evt_cap_reg_rd_data),
       .evt_cap_reg_ack         (evt_cap_reg_ack),

       // --- Interface to signals
       .signals                 (oq_signals),
       .signal_values           (oq_signal_values),
       .signal_ids              (oq_signal_ids),
       .reg_values              (oq_abs_regs),

       // --- Misc
       .clk                     (clk),
       .reset                   (reset));

    output_queues
      #(.DATA_WIDTH(DATA_WIDTH),
UTF8--**-L428 C18 71%   user_data_path.v (...izing/src/udp)  (Ve
C-c C-u
```

Exercise 2

Step 5 - Add the Drop Nth Module

Search for drop_nth_packet (ctrl+s drop_nth_packet), then press Enter

Uncomment the block (ctrl +c+u)



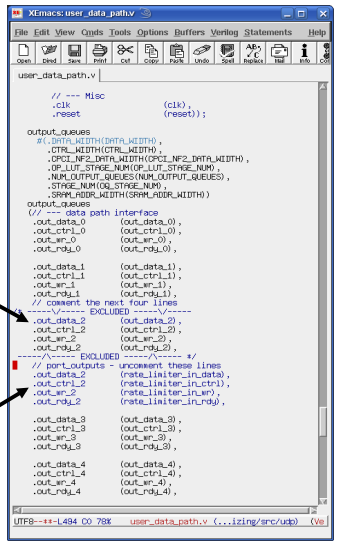
Exercise 2

Step 6 - Connect the Output Queue to the Rate Limiter

Search for port_outputs (ctrl +s port_outputs), then press (Enter)

Comment the 4 lines above (select the four lines by using shift+arrow keys), then type (ctrl+c+c)

Uncomment the commented block by scrolling down into the block and typing ctrl+c +u



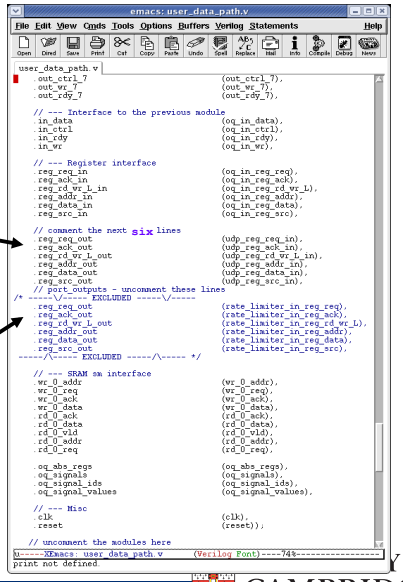
Exercise 2

Step 7 - Connect the Registers

Search for port_outputs (ctrl +s port_outputs), then press (Enter)

Comment the 6 lines (select the six lines by using shift+arrow keys), then type (ctrl+c+c)

Uncomment the commented block by scrolling down into the block and typing (ctrl+c +u)



Exercise 2

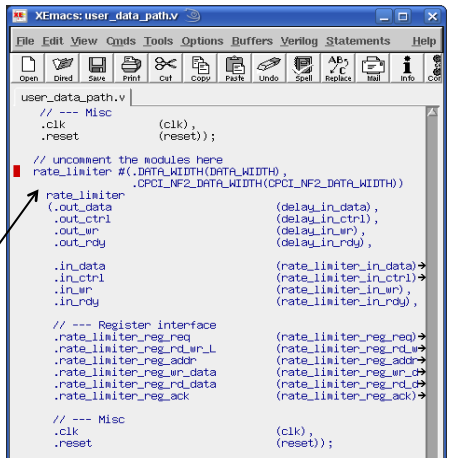
Step 8 - Add Rate Limiter

Scroll down until you reach the next "excluded" block

Uncomment the block containing the rate limiter instantiations.

Scroll into the block, type (ctrl+c+u)

Save (ctrl+x+s)



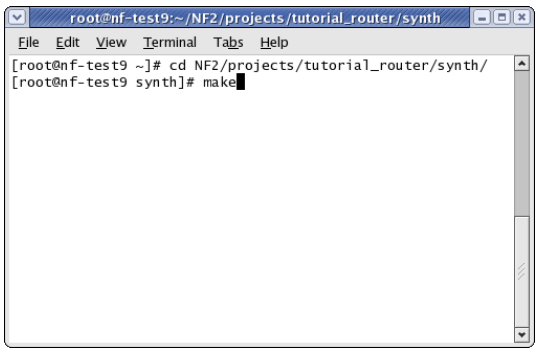
Exercise 2

Step 9 - Build the Hardware

Start terminal, cd to “NF2/projects/tutorial_router/synth”

Run “make clean”

Start synthesis with “make”



Second Break

(while hardware compiles)

Lunch is downstairs

Exercise 2

Observing and Controlling the Queue Size
With *YOUR* enhanced router!

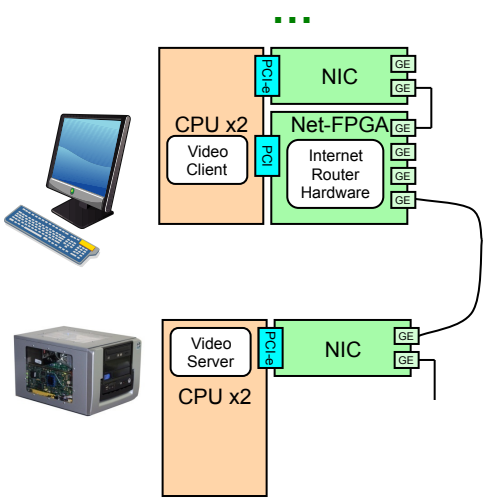
Using NetFPGA to explore buffer size

- **Need to reduce buffer size and measure occupancy**
- **Alas, not possible in commercial routers**
- **So, we will use the NetFPGA instead**

Objective:

– Use the NetFPGA to understand how large a buffer we need for a **single** TCP flow.

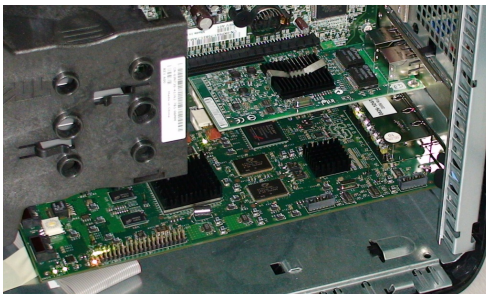
NetFPGA Hardware Set for Exercise #2

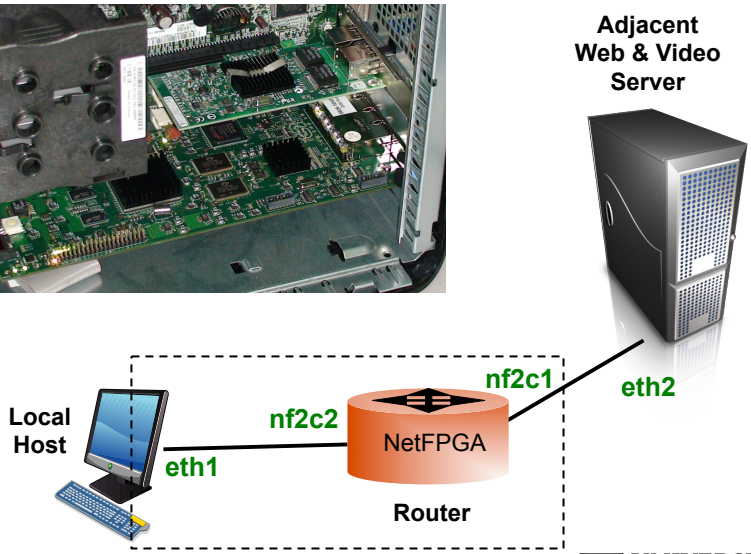


Server delivers streaming HD video to adjacent client

Exercise 2

Setup for the Exercise 2

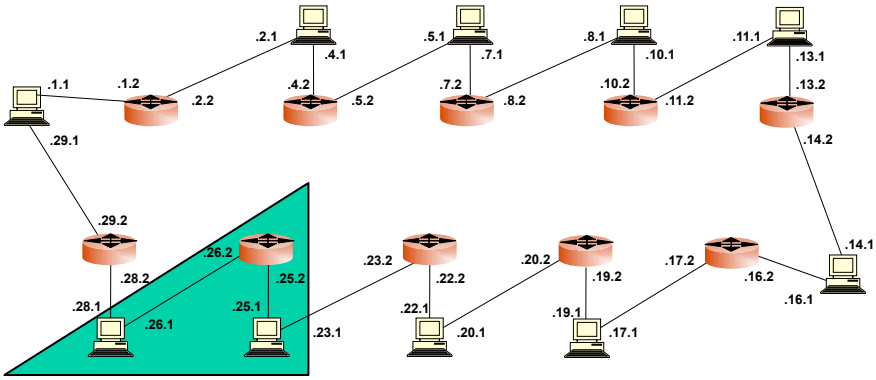




Exercise 2

Interfaces and Subnets

- eth1 connects your host to your NetFPGA Router
- nf2c2 routes to nf2c1 (your adjacent server)
- eth2 serves web and video traffic to your neighbor
- nf2c0 & nf2c3 (the network ring) are unused

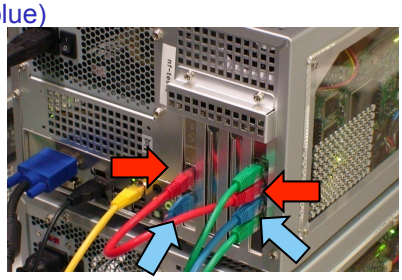


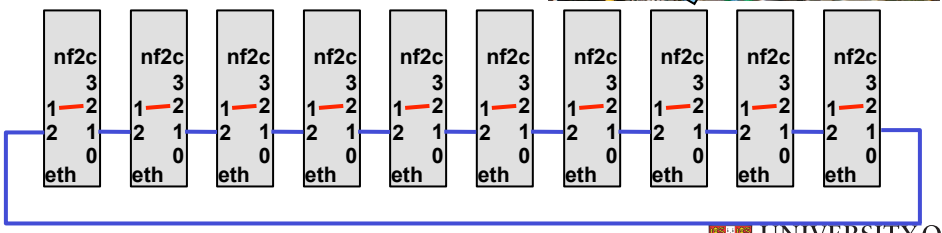
This configuration allows you to modify and test your router without affecting others

Exercise 2

Cable Configuration for Exercise 2

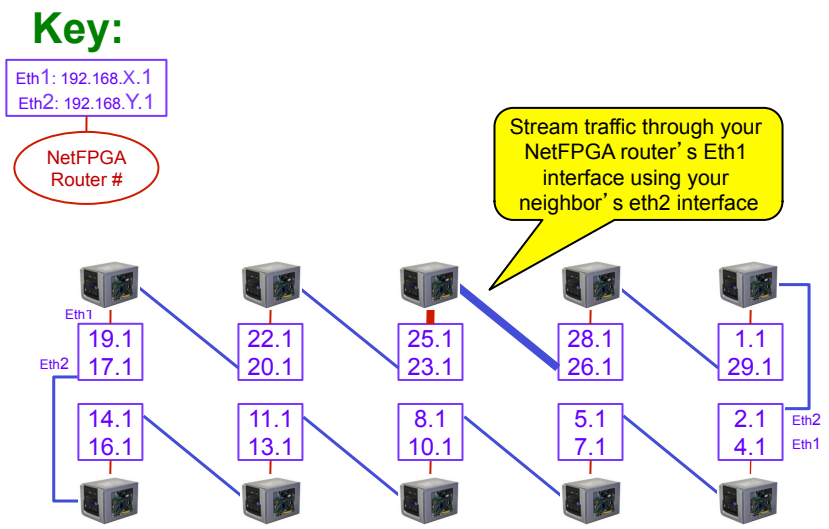
- **NetFPGA Gigabit Ethernet Interfaces**
  - nf2c2 : Local host interface (red)
  - nf2c1 : Router for adjacent server (blue)
- **Host Ethernet Interfaces**
  - eth1 : Local host interface (red)
  - eth2 : Server for neighbor (blue)





Exercise 2

Exercise 2 Configuration



Exercise 2

Enhanced Router

Objectives

- Observe router with new modules
- New modules: rate limiting, event capture

Execution

- Run event capture router
- Look at routing tables
- Explore details pane
- Start tcp transfer, look at queue occupancy
- Change rate, look at queue occupancy

Exercise 2

Step 1 - Run Your Enhanced Router

Start terminal and cd to

`NF2/projects/`

`tutorial_router/sw/`

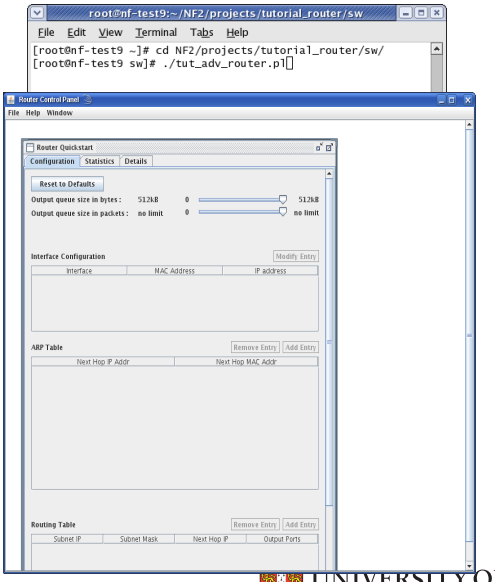
NB: **…_ADV_…**

Type

./tut_adv_router_gui.pl -use_bin\

../../../bitfiles/tutorial_router.bit

A familiar GUI should start

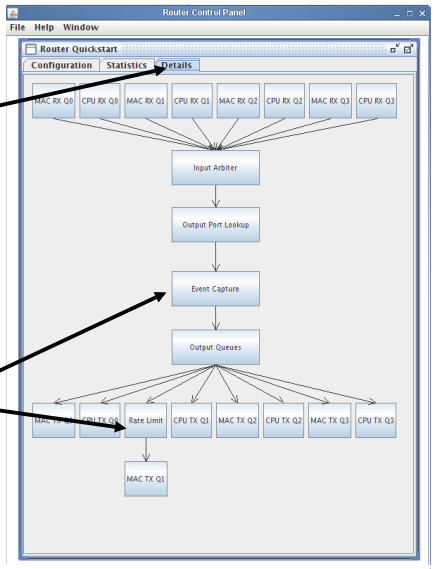


Exercise 2

Step 2 - Explore Your Enhanced Router

Click on the Details tab

A similar pipeline to the one seen previously shown with some additions

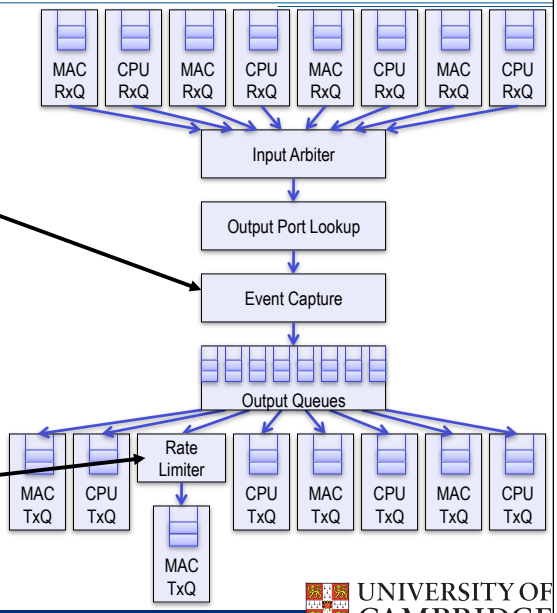


Exercise 2

Enhanced Router Pipeline

Two modules added

1. **Event Capture** to capture output queue events (writes, reads, drops)
2. **Rate Limiter** to create a bottleneck

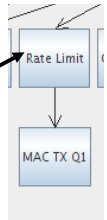


Exercise 2

Step 3 - Decrease the Link Rate

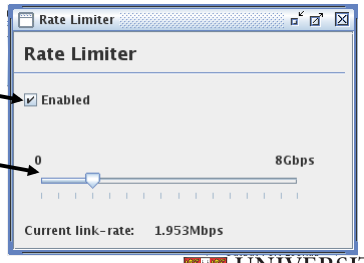
To create bottleneck and show the TCP "sawtooth," link-rate is decreased.

In the Details tab, click the "Rate Limit" module



Check Enabled

Set link rate to 1.953Mbps



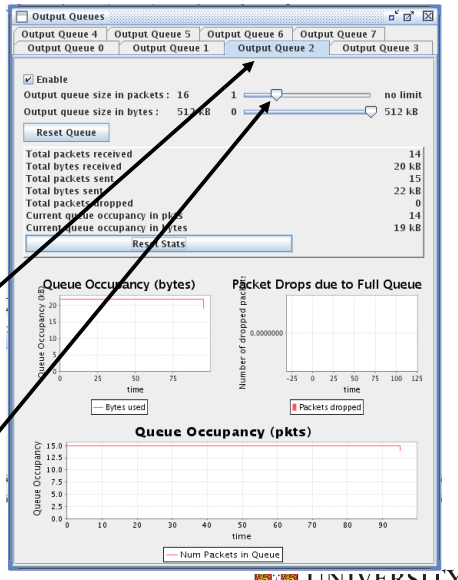
Exercise 2

Step 4 – Decrease Queue Size

Go back to the Details panel and click on "Output Queues"

Select the "Output Queue 2" tab

Change the output queue size in packets slider to 16

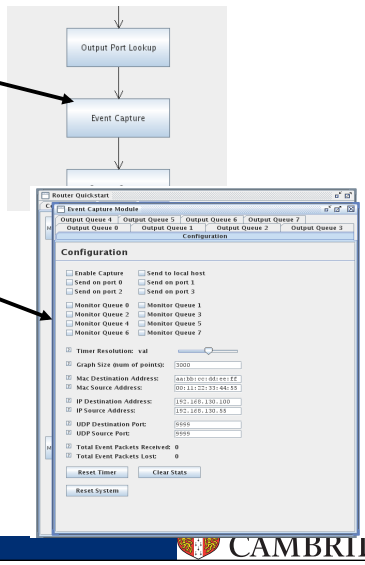


Exercise 2

Step 5 - Start Event Capture

Click on the Event Capture module under the Details tab

This should start the configuration page



Exercise 2

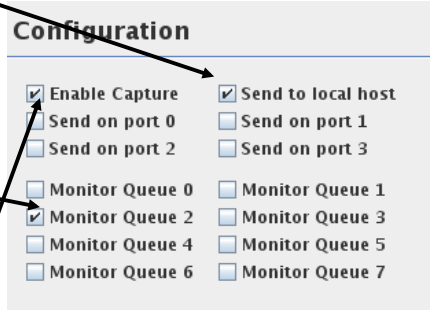
Step 6 - Configure Event Capture

Please do these in the ORDER below…

Check **Send to local host** to receive events on the local host

Check **Monitor Queue 2** to monitor output queue of MAC port1

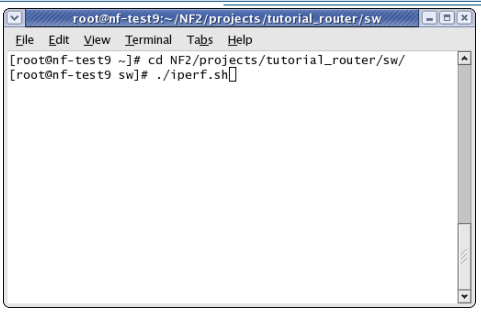
Check **Enable Capture** to start event capture



Exercise 2

Step 7 - Start TCP Transfer

We will use *iperf* to run a large TCP transfer and look at queue evolution



Start a new terminal and cd to "NF2/projects/tutorial_router/sw"

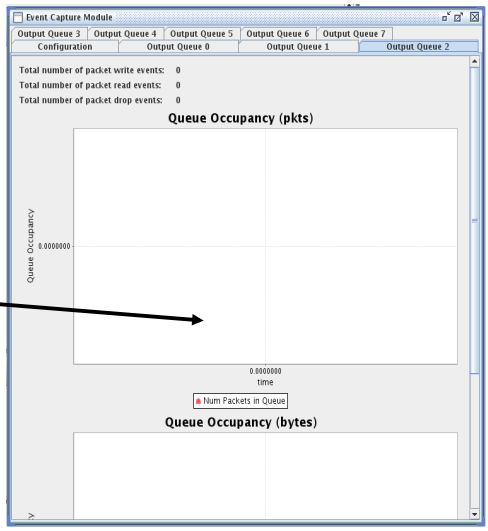
Type "./iperf.sh"

Exercise 2

Step 8 - Look at Event Capture Results

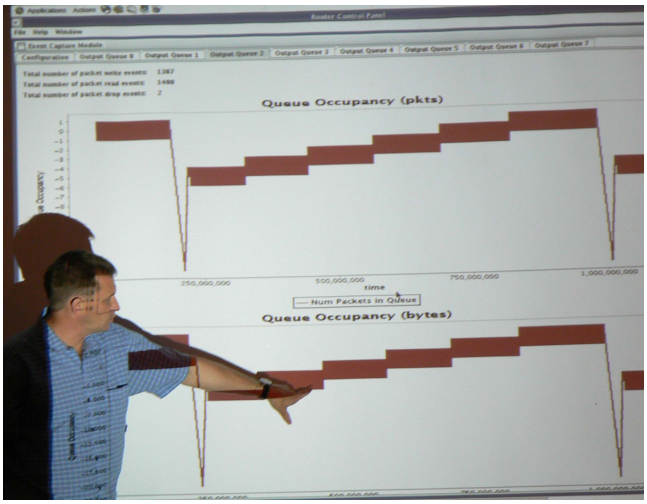
Click on the Event Capture module under the Details tab.

The sawtooth pattern should now be visible.



Queue Occupancy Charts

Observe the TCP/IP sawtooth – observe the BUFFER occupancy



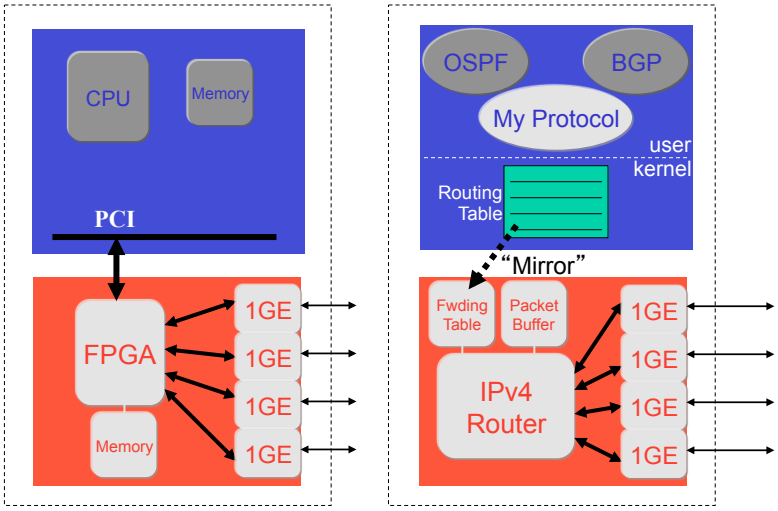
Leave the control windows open

NetFPGA is a Community

Usage #1

Running the Router Kit

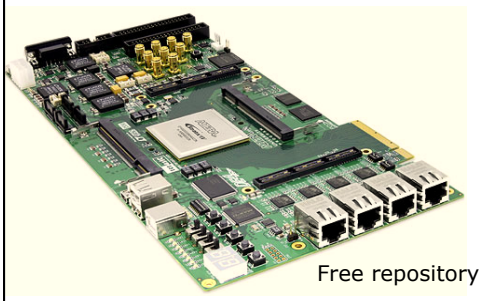
User-space development, 4x1GE line-rate forwarding

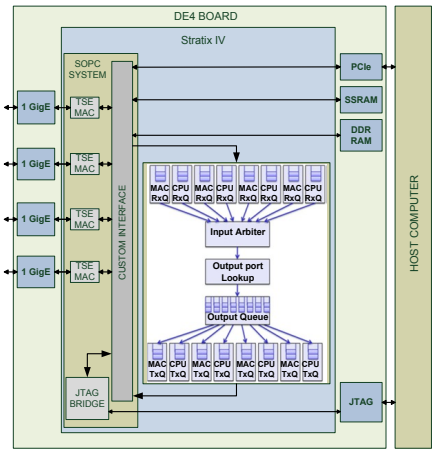


Altera-DE4 NetFPGA Reference Router

UMassAmherst

- Migration of NetFPGA infrastructure to DE4 Stratix IV – 4X logic vs. Virtex 2
- PCI Express Gen2 – 5.0Gbps/lane data
- External DDR2 RAM – 8-Gbyte capacity.
- Status: Functional – basic router performance matches current NetFPGA
- Lots of logic for additional functions
- Russ Tessier (tessier@ecs.umass.edu)

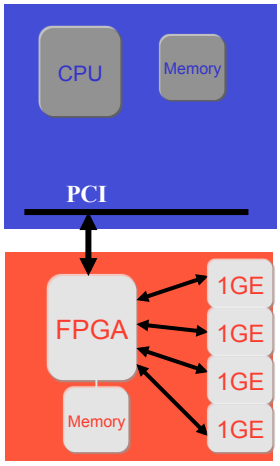


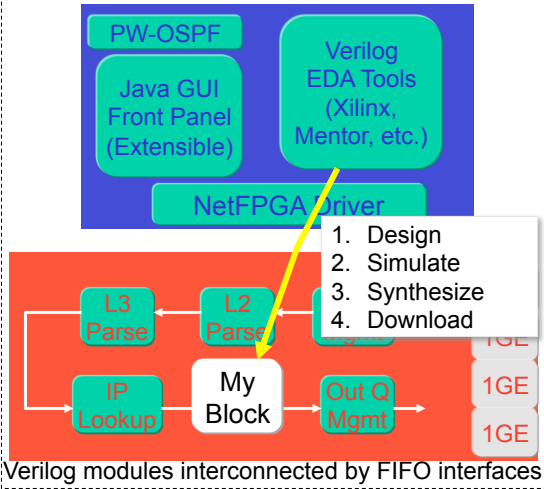


Free repository available from UMass in September 2011

Usage #2

Enhancing Modular Reference Designs

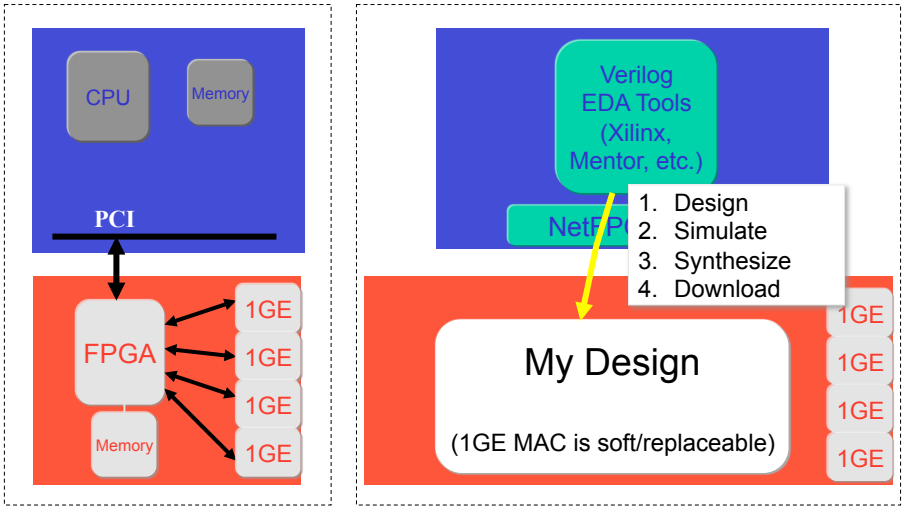




Verilog modules interconnected by FIFO interfaces

Usage #3

Creating new systems



NetThreads, NetThreads-RE, NetTM

U. of Toronto

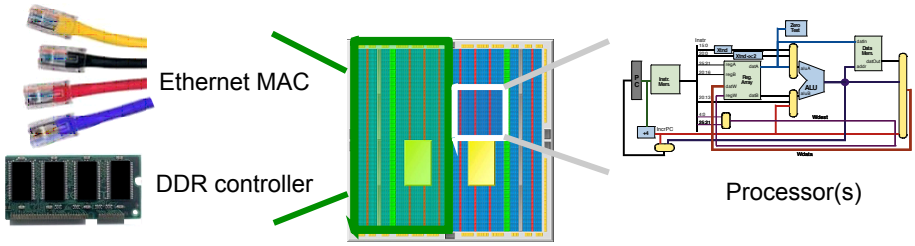
Martin Labrecque
Gregory Steffan

ECE Dept.

Geoff Salmon
Monia Ghobadi
Yashar Ganjali

CS Dept.

Soft Processors in FPGAs



- Soft processors: processors in the FPGA fabric
- User uploads program to soft processor
- Easier to program software than hardware in the FPGA
- Could be customized at the instruction level

Process packets in software!

Fast enough?

Performance In Packet Processing

- **The application defines the requirements**

Scientific instruments
(< 100 Mbps/link)

Home networking
(~100 Mbps/link)

Edge routing
(≥ 1 Gbps/link)

Are soft processors fast enough?

Realistic Goals

- **1 gigabit stream**
- **2 processors running at 125 MHz**
- **Cycle budget for back-to-back packets:**
  - –152 cycles for minimally-sized 64B packets;
  - –3060 cycles for maximally-sized 1518B packets

Soft processors can perform non-trivial processing at 1gigE!

Latency to answer a ping request:

Quad Xeon server → 48.9 us +/-17.5 us

NetThreads in NetFPGA 1G → 5.1 us +/- 0.04us

NetThread projects provide:

- **Efficient multithreaded design**
  - –Parallel threads deliver performance
- **System Features**
  - –System is easy to program in C
  - –Time to results is very short

We hope to see many projects

We also need help:

- e.g. software that could be ported: operating system, lwIP

NetThread followup Questions?

Ask: Martin Labrecque
martinL@eecg.utoronto.ca

- **NetThreads, NetThreads-RE & NetTM available with supporting software at:**
  http://www.netfpga.org/foswiki/bin/view/NetFPGA/OneGig/NetThreads
  http://www.netfpga.org/foswiki/bin/view/NetFPGA/OneGig/NetThreadsRE
  http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/NetTM

Using the NetFPGA in the Classroom

NetFPGA in the Classroom

- Stanford University
  - EE109 "Build an Ethernet Switch"
    - Undergraduate course for all EE students
    - http://www.stanford.edu/class/ee109/
  - CS344 "Building an Internet Router" (since '05)
    - Quarter-long course targeted at graduates
    - http://cs344.stanford.edu
- Rice University
  - Network Systems Architecture (since '08)
    - http://comp519.cs.rice.edu/
- Cambridge University
  - Build an Internet Router (since '09)
    - Quarter-long course targeted at graduates
    - http://www.cl.cam.ac.uk/teaching/0910/P33/
- University of Wisconsin
  - CS838 "Rethinking the Internet Architecture"
    - http://pages.cs.wisc.edu/~akella/CS838/F09/

See: http://netfpga.org/teachers.html

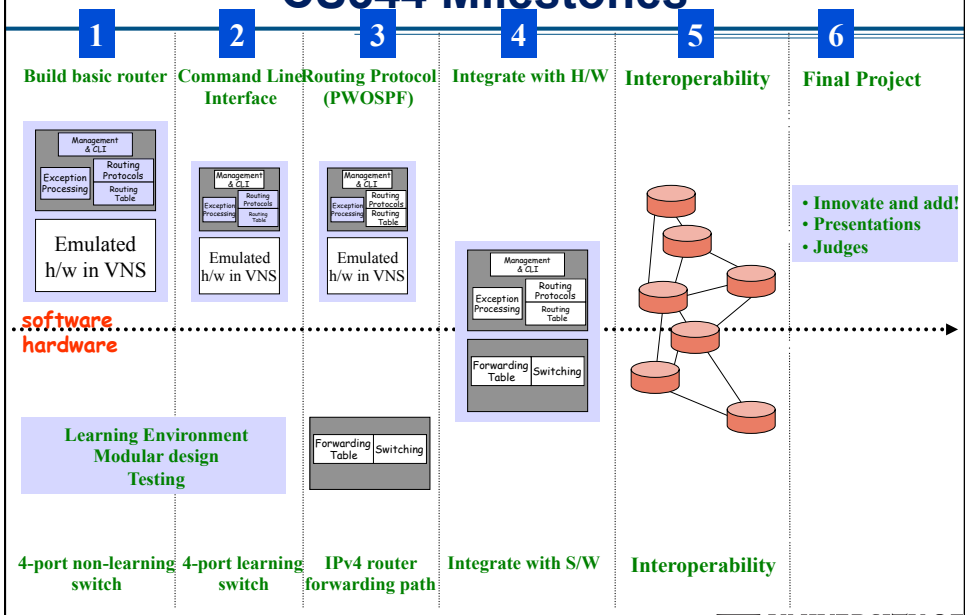
Components of NetFPGA Course

- **Documentation**
  - System Design
  - Implementation Plan
- **Deliverables**
  - Hardware Circuits
  - System Software
  - Milestones
- **Testing**
  - Proof of Correctness
  - Integrated Testing
  - Interoperabilty
- **Post Mortem**
  - Lessons Learned

NetFPGA in the Classroom

- **Stanford CS344: "Build an Internet Router"**
  - Courseware available on-line
  - Students work in teams of three
    - 1-2 software
    - 1-2 hardware
  - Design and implement router in 8 weeks
  - Write software for CLI and PW-OSPF
  - Show interoperability with other groups
  - Add new features in remaining two weeks
    - Firewall, NAT, DRR, Packet capture, Data generator, …

CS344 Milestones



Typical NetFPGA Course Plan

Week	Software	Hardware	Deliver
1	Verify Software Tools	Verify CAD Tools	Write Design Document
2	Build Software Router	Build Non-Learning Switch	Run Software Router
3	Cmd. Line Interface	Build Learning Switch	Run Basic Switch
4	Router Protocols	Output Queues	Run Learning Switch
5	Implement Protocol	Forwarding Path	Interface SW & HW
6	Control Hardware	Hardware Registers	HW/SW Test
7	Interoperate Software & Hardware		Router Submission
8	Plan New Advanced Feature		Project Design Plan
9	Show new Advanced Feature		Demonstration

Presentations

Stanford CS344

http://cs344.stanford.edu

Cambridge P33

http://www.cl.cam.ac.uk/teaching/0910/P33/

From our classroom to yours…

Exercise 3: Controlled packet-loss

Beyond just observation,
using NetFPGA for an experiment

Exercise 3

Controlled packet-loss

- Packet networks have loss; evaluating loss we use modeling, simulation, emulation, real-world experiments
- NetFPGA can implement a controlled, packet loss mechanism with none of the disadvantages of emulation…
- Exercise 3: Drop 1 in N Packets….

Exercise 3

Drop 1 in N Packets

Objectives

- Add counter and FSM to the code
- Synthesize and test router

Execution

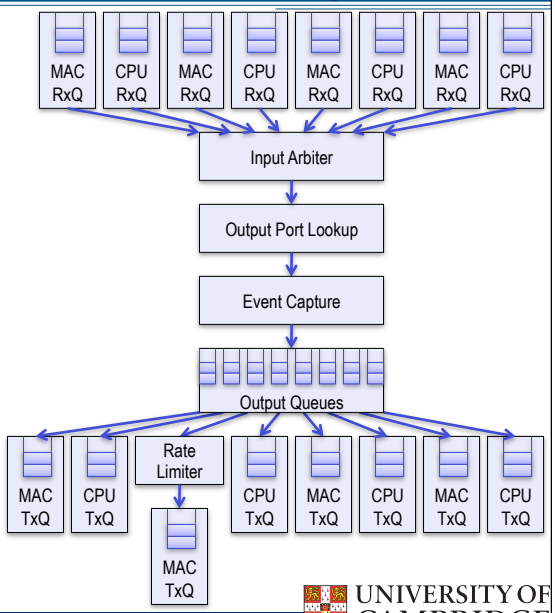
- Open drop_nth_packet.v
- Insert counter code
- Synthesize
- After synthesis, test the new system.

Exercise 3

Our Enhanced Router Pipeline

One module added

1. **Drop Nth Packet**
   to drop every Nth packet from the reference router pipeline

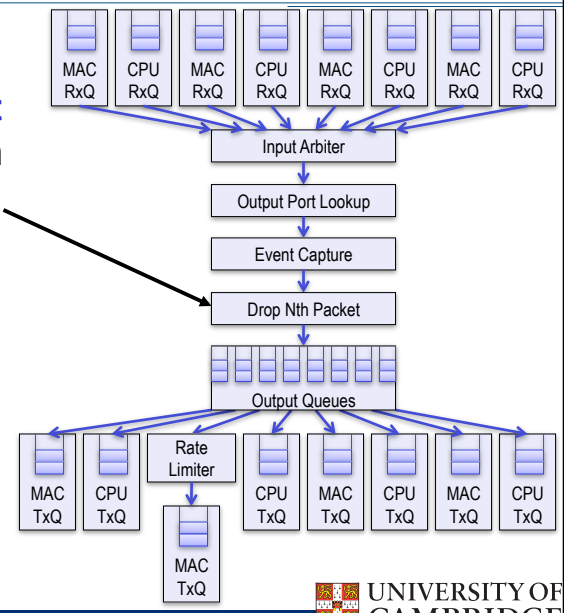


Exercise 3

New Even-More Enhanced Router Pipeline

One module added

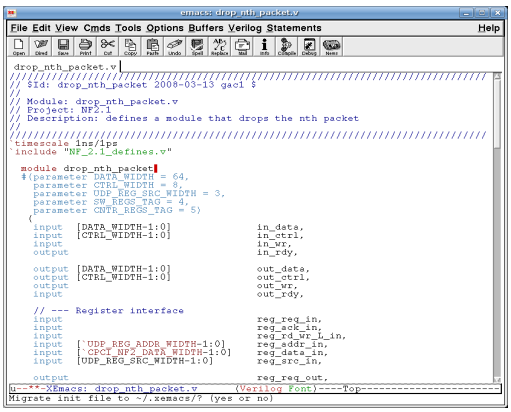
1. **Drop Nth Packet**
   to drop every Nth packet from the reference router pipeline



Exercise 3

Step 1 - Open the Source

We will modify the Verilog source code to add a counter to the drop_nth_packet module



Open terminal

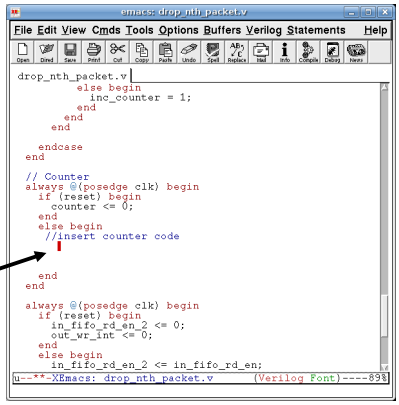
Type "`emacs NF2/projects/tutorial_router/src/drop_nth_packet.v`

Exercise 3

Step 2 - Add Counter to Module

Add counter using the following signals:

- `counter`
  - 16 bit output signal that you should increment on each packet pulse
- `rst_counter`
  - reset signal (a pulse input)
- `inc_counter`
  - increment (a pulse input)



Search for insert counter

(ctrl+s insert counter, Enter)

Insert counter and save

(ctrl+x+s)

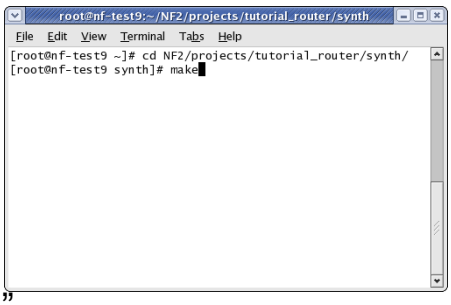
Exercise 3

Step 3 - Build the Hardware

Start terminal, cd to "NF2/projects/tutorial_router/synth"

Run "make clean"

Start synthesis with "make"



Third Break

(while hardware compiles)

Exercise 3

Step 4 – Test your Router

You can watch the number of received and sent packets to watch the module drop every Nth packet. Ping a local machine (i.e. 192.168.7.1) and watch for missing pings

To run your router:

1- Enter the directory by typing:
cd NF2/projects/tutorial_router/sw

2- Run the router by typing:
./tut_adv_router_gui.pl --use_bin ../../../bitfiles/tutorial_router.bit

To set the value of N (which packet to drop)
type regwrite 0x2000704 N (Open a new terminal first)
– replace N with a number (such as 100)

To enable packet dropping, type:
regwrite 0x2000700 0x1

To disable packet dropping, type:
regwrite 0x2000700 0x0

Exercise 3

Step 5a – Measurements (network)

- **Explore loss across network**
- **Ping to neighbour's server (and other servers)**
  - Set a loss of 1 in 100 then, similar to Exercise 1
  - Ping 192.168.x.2 (where x is your immediate neighbour's server)
    - What is the loss? 1 in 100?
  - Now, ping any addresses 192.168.x.y where x is from 1-20 and y is 1 or 2
  - Can you compute the loss-rate of a neighbour's router?
  - Apart from ping packets, what other packets might be lost?

    (routing activities, control packets, ARP, …..)

Exercise 3

Step 5b – Measurements (transport)

- **Determine iperf TCP throughput to neighbour's server for each of several values of N**
  - Similar to Exercise 2, Step 7
  - TCP throughput with:
    - Drop circuit disabled
      - TCP Throughput = _____ Mbps
    - Drop one in N = 10,000 packets
      - TCP Throughput = _____ Mbps
    - Drop one in N = 1,000 packets
      - TCP Throughput = _____ Mbps
    - Drop one in N = 100 packets
      - TCP Throughput = _____ Mbps
- **Explain why TCPs throughput is so low given that only a tiny fraction of packets are lost**

Exercise 3

Step 5c – Measurements (subjective)

- **Consider video throughput to a neighbour's server for each of several values of N**

  To stream video, run (in a new terminal)

  ```
  cd ~/NF2/projects/tutorial_router/sw
  ./mp 192.168.X.Y
  ```

  where ***X.Y*** = 25.1 or 19.1 or 7.1

  Open `firefox` **for the full list of host IP address**

  – Similar to Exercise 1, Step 2
  – Subjective video quality…
    - Drop circuit disabled
      – Video Quality = Excellent / Good / Fair / Poor / Bad
    - Drop one in N = 10,000 packets
      – Video Quality = Excellent / Good / Fair / Poor / Bad
    - Drop one in N = 1,000 packets
      – Video Quality = Excellent / Good / Fair / Poor / Bad
    - Drop one in N = 100 packets
      – Video Quality = Excellent / Good / Fair / Poor / Bad

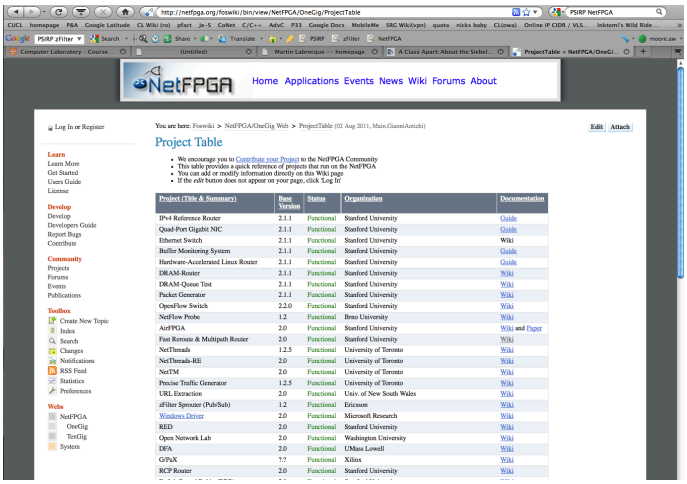
Conclusion

Conclusions

- **NetFPGA Provides**
  - Open-source, hardware-accelerated Packet Processing
  - Modular interfaces arranged in reference pipeline
  - Extensible platform for packet processing
- **NetFPGA Reference Code Provides**
  - Large library of core packet processing functions
  - Scripts and GUIs for simulation and system operation
  - Set of Projects for download from repository
- **The NetFPGA Base Code**
  - Well defined functionality defined by regression tests
  - Function of the projects documented in the Wiki Guide

What to do next?

Explore existing projects:



Networked FPGAs in Research

1. **Managed flow-table switch**
   - http://OpenFlowSwitch.org/
2. **Buffer Sizing**
   - Reduce buffer size & measure buffer occupancy
3. **RCP: Congestion Control**
   - New module for parsing and overwriting new packet
   - New software to calculate explicit rates
4. **Deep Packet Inspection (FPX)**
   - TCP/IP Flow Reconstruction
   - Regular Expression Matching
   - Bloom Filters
5. **Packet Monitoring (ICSI)**
   - Network Shunt
6. **Precise Time Protocol (PTP)**
   - Synchronization among Routers

To get started with your project

Prepare for your project

a) Learn NetFPGA by yourself

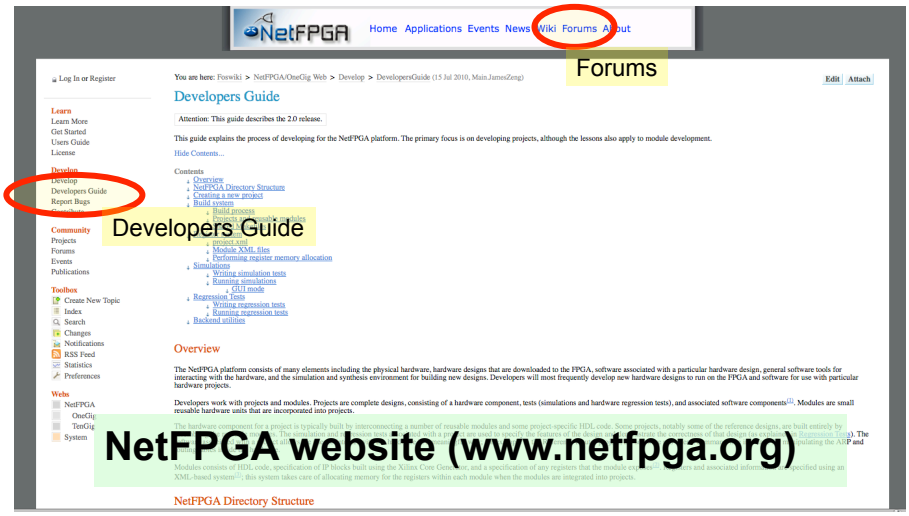
b) *Encourage others to*
Complete a hands-on tutorial *too*

c) Consider attending (hosting)
a *summer* school – doesn't have to be summer!

EU/OFELIA-CHANGE NetFPGA Introduction
November 10, 2011 09:00-11:30
http://changeofelia.info.ucl.ac.be/

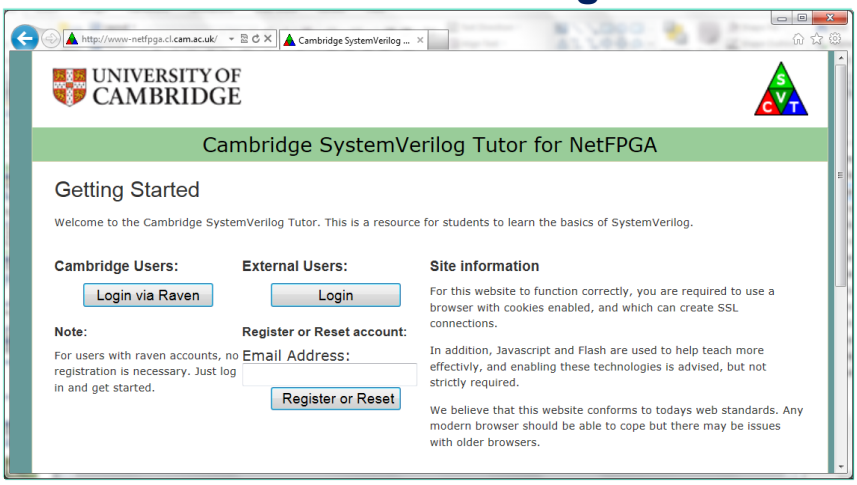
Learn by Yourself

Learn by Yourself



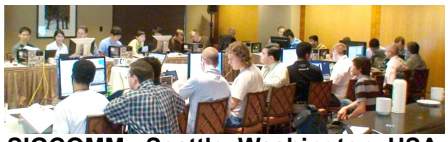
Learn by Yourself

Online tutor – coming soon!



Support for NetFPGA enhancements provided by redgate software

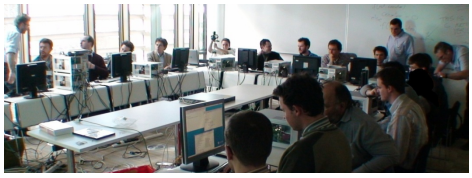
Photos from NetFPGA Tutorials



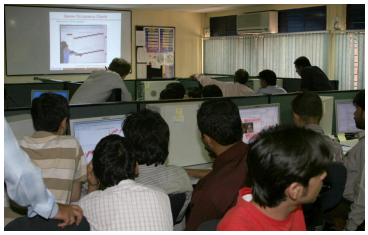
SIGCOMM - Seattle, Washington, USA

Beijing, China

SIGMETRICS - San Diego, California, USA



EuroSys - Glasgow, Scotland, U.K.



Bangalore, India

http://netfpga.org/pastevents.php and http://netfpga.org/upcomingevents.php

NetFPGA 2-Day workshop in Cambridge

Want a tutorial/workshop at your institution?

talk to Andrew

- 20 attendees (full house)
- accommodation for non-locals
- 30% commercial attendees

Thoughts for (Prospective) Contributors

- **Build Modular components**
  - Describe shared registers (as per 2.0 release)
  - Consider how modules would be used in larger systems
- **Define functionality clearly**
  - Through regression tests
  - With repeatable results
- **Disseminate projects**
  - Post open-source code
  - Document projects on Web, Wiki, and Blog
- **Expand the community of developers**
  - Answer questions in the Discussion Forum
  - Collaborate with your peers to build new applications

Acknowledgments

NetFPGA Team at Stanford University (Past and Present):

Nick McKeown, Glen Gibb, Jad Naous, David Erickson,
G. Adam Covington, John W. Lockwood, Jianying Luo, Brandon Heller, Paul Hartke, Neda Beheshti, Sara Bolouki, James Zeng,
Jonathan Ellithorpe, Sachidanandan Sambandan, Eric Lo

NetFPGA Team at University of Cambridge (Past and Present):

Andrew Moore, David Miller, Muhammad Shahbaz, Martin Zadnik
For help with todays tutorial
Yury Audzevich, Ed Cree, Andriy Gordiychuk, James Snee

All Community members (including but not limited to):

Paul Rodman, Kumar Sanghvi, Wojciech A. Koszek,
Yahsar Ganjali, Martin Labrecque, Jeff Shafer,
Eric Keller , Tatsuya Yabe, Bilal Anwer,
Yashar Ganjali, Martin Labrecque

Kees Vissers, Michaela Blott, Shep Siegel

Special thanks to our Partners:

Ram Subramanian, Patrick Lysaght, Veena Kumar, Paul Hartke, Anna Acevedo
Xilinx University Program (XUP)

Other NetFPGA Tutorials Presented At:

See: http://NetFPGA.org/tutorials/

Thanks to our Sponsors:

- **Support for the NetFPGA project has been provided by the following companies and institutions**

Disclaimer: Any opinions, findings, conclusions, or recommendations expressed in these materials do not necessarily reflect the views of the National Science Foundation or of any other sponsors supporting this project.

Group Discussion

- **Your plans for using the NetFPGA**
  - Teaching
  - Research
  - Other
- **Resources needed for your class**
  - Source code
  - Courseware
  - Examples
- **Your plans to contribute**
  - Expertise
  - Capabilities
  - Collaboration Opportunities

Feedback

- **We thrive on feedback – please fill in the survey now…..**

http://www.netfpga.org/php/tutorial_survey.php

Thanks!

NetFPGA website (www.netfpga.org)