

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>

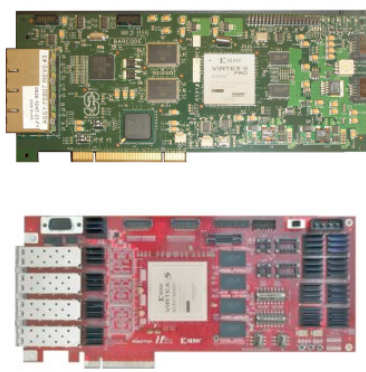
Tutorial Outline








- **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](#)


Cambridge – September 1st, 2011

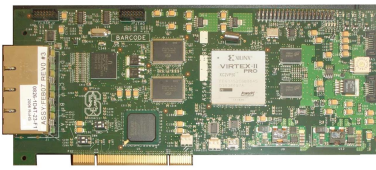
5


UNIVERSITY OF CAMBRIDGE


NetFPGA consists of...

Four elements:


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




NetFPGA 1G Board



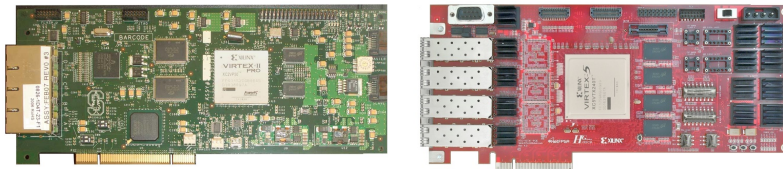
NetFPGA 10G Board


Cambridge – September 1st, 2011

6


UNIVERSITY OF CAMBRIDGE

NetFPGA Board Comparison

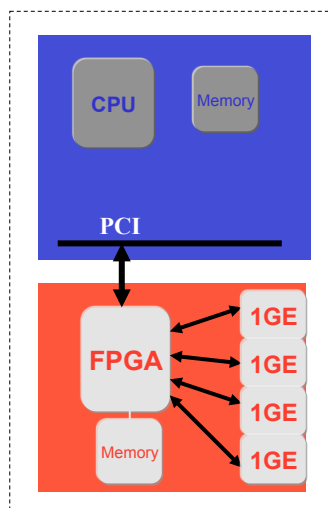


| NetFPGA 1G | NetFPGA 10G |
|-------------------------------------|--------------------------------------|
| 4 x 1Gbps Ethernet Ports | 4 x 10Gbps SFP+ |
| 4.5 MB ZBT SRAM 64 MB DDR2 SDRAM | 27 MB QDRII-SRAM 288 MB RLDRAM-II |
| PCI | PCI Express x8 |
| Virtex II-Pro 50 | Virtex 5 TX240T |

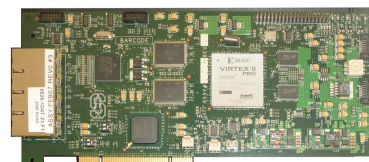
NetFPGA board

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

The diagram illustrates the Router Kit architecture. On the left, a blue box represents the user-space development environment, containing OSPF, BGP, and My Protocol running in the user kernel. Below this is a Routing Table. On the right, a red box represents the hardware IPv4 Router, which includes a Forwarding Table, Packet Buffer, and four 1GE ports. A dashed box on the right is labeled "Mirror" with a dashed arrow pointing towards it. The University of Cambridge logo and name are visible at the bottom right.

UNIVERSITY OF CAMBRIDGE

Cambridge – September 1st, 2011 9

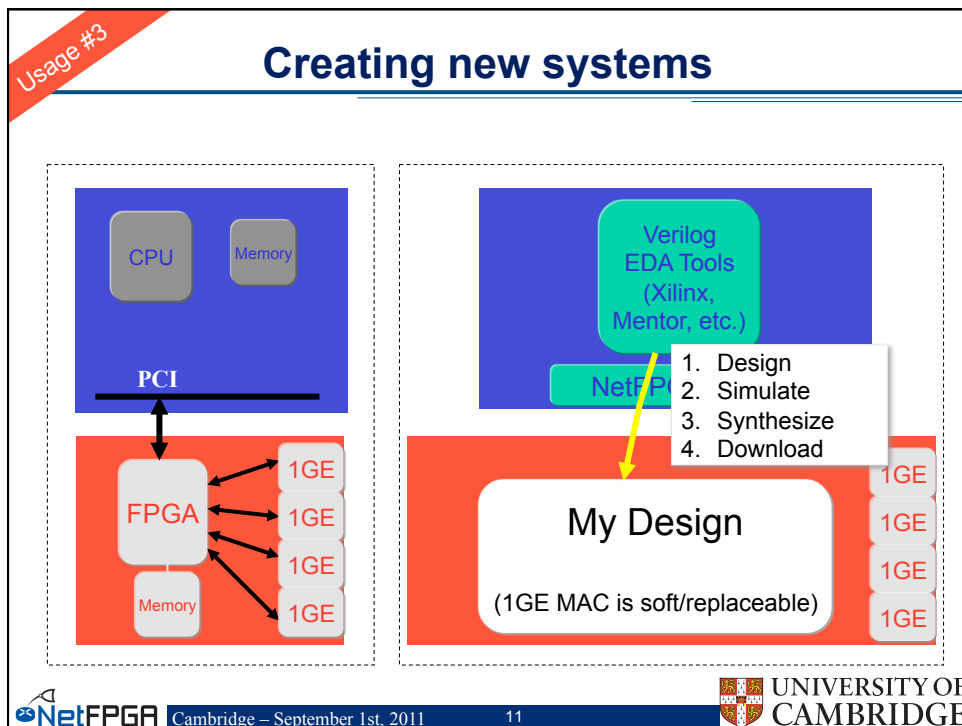
Usage #2

Enhancing Modular Reference Designs

The diagram shows the modular reference design architecture. On the left, a blue box contains CPU and Memory connected to an FPGA via a PCI interface. The FPGA is connected to four 1GE ports. On the right, a blue box contains software components: PW-OSPF, Java GUI Front Panel (Extensible), Verilog EDA Tools (Xilinx, Mentor, etc.), and a NetFPGA Driver. Below this, a red box shows the hardware flow: L3 Parse, L2 Parse, My Block, and Out Q Mgmt, all interconnected by FIFO interfaces. A yellow arrow points from the Verilog EDA Tools box to the L2 Parse block. A list of steps is shown: 1. Design, 2. Simulate, 3. Synthesize, 4. Download. The University of Cambridge logo and name are visible at the bottom right.

UNIVERSITY OF CAMBRIDGE

Cambridge – September 1st, 2011 10



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)

UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 12

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>



Cambridge – September 1st, 2011

13



UNIVERSITY OF
CAMBRIDGE

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**



Cambridge – September 1st, 2011

14



UNIVERSITY OF
CAMBRIDGE

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, ...

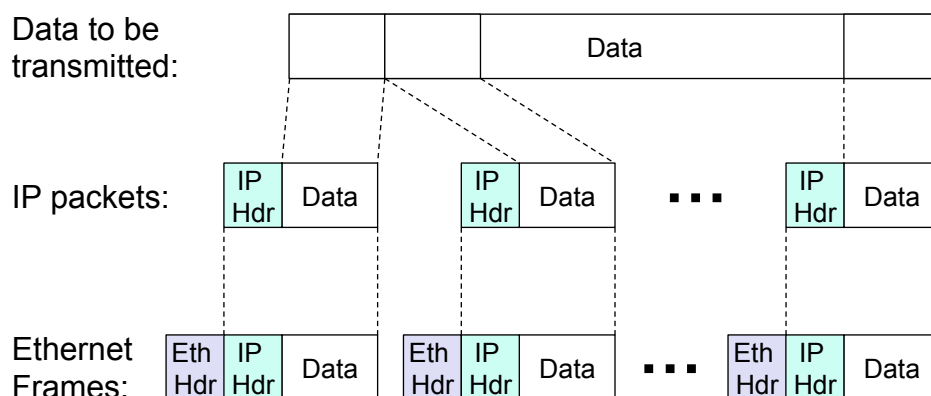
Why do they use the NetFPGA?

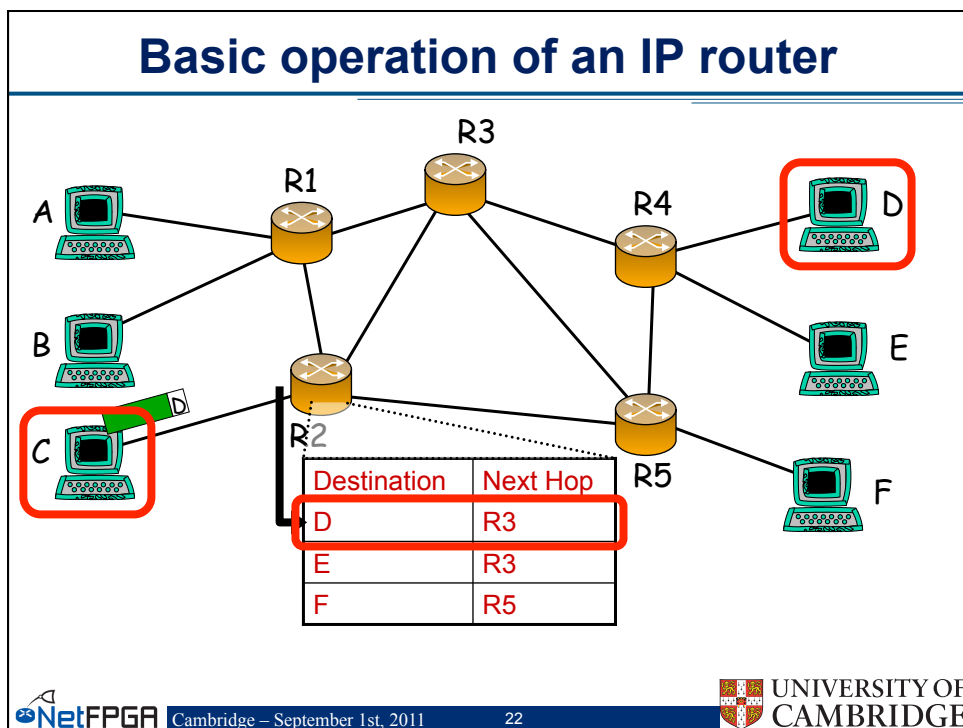
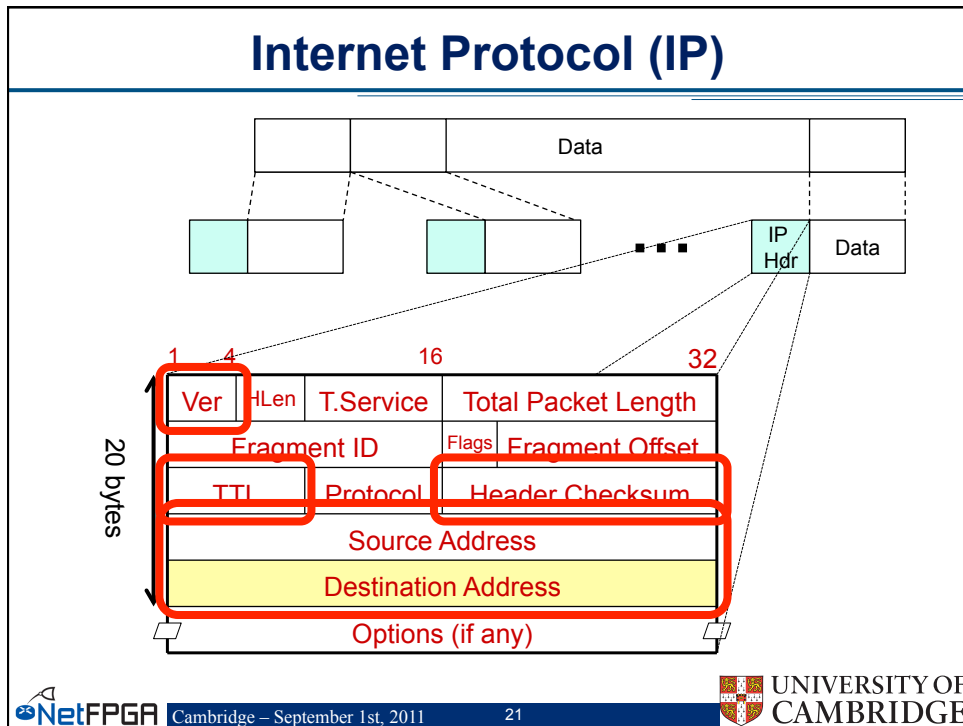
- To measure performance of Internet systems
- To prototype new networking systems

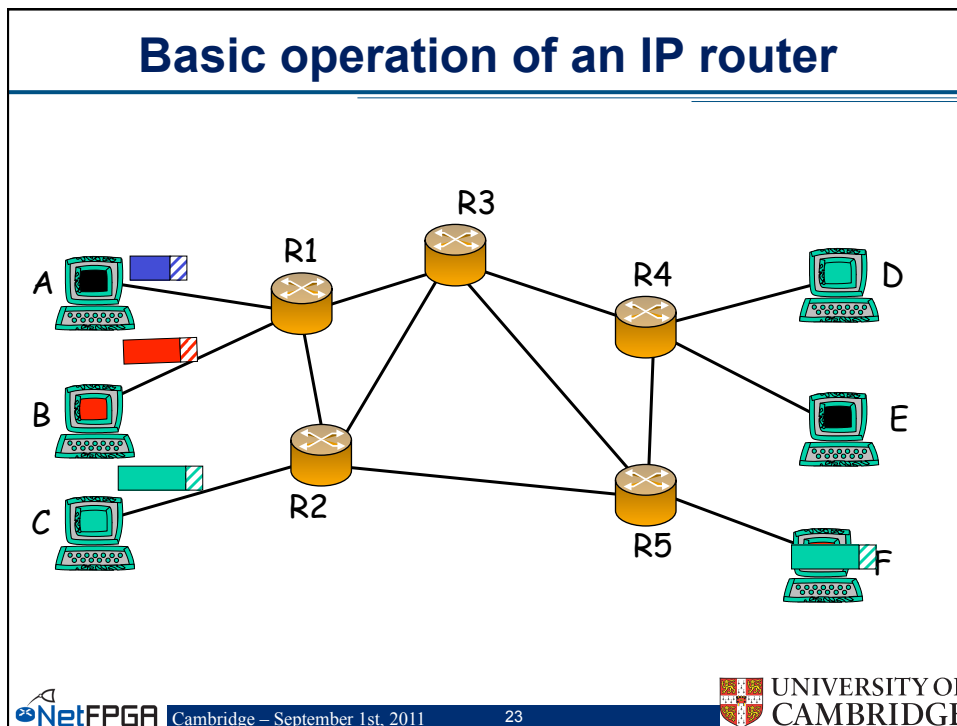
Lets have an example, but first....

Network review

Internet Protocol (IP)







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 |

 Cambridge – September 1st, 2011 24  UNIVERSITY OF CAMBRIDGE

IP addresses as a line

All IP addresses

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

Cambridge – September 1st, 2011 25 UNIVERSITY OF CAMBRIDGE

Longest Prefix Match (LPM)

| Entry | Destination | Port | |
|-------|----------------------|------|--------------|
| 1 | Stanford | 1 | Universities |
| 2 | Berkeley | 2 | |
| 3 | North America | 3 | Continents |
| 4 | Asia | 4 | |
| 5 | Everywhere (default) | 5 | Planet |

Matching entries:

- Stanford Most specific
- North America
- Everywhere

To: Stanford Data

Cambridge – September 1st, 2011 26 UNIVERSITY OF CAMBRIDGE



Longest Prefix Match (LPM)

| Entry | Destination | Port | |
|-------|----------------------|------|--------------|
| 1 | Stanford | 1 | Universities |
| 2 | Berkeley | 2 | |
| 3 | North America | 3 | Continents |
| 4 | Asia | 4 | |
| 5 | Everywhere (default) | 5 | Planet |

Matching entries:
• North America Most specific
 • Everywhere

To:
Canada

Data


Cambridge – September 1st, 2011
27

UNIVERSITY OF CAMBRIDGE



Implementing Longest Prefix Match

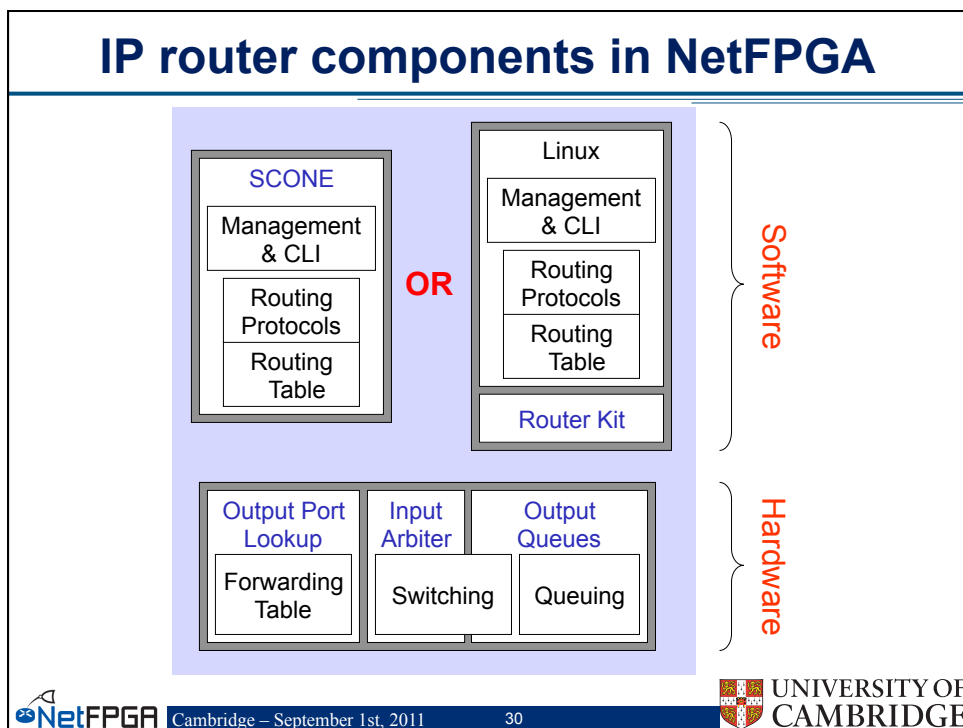
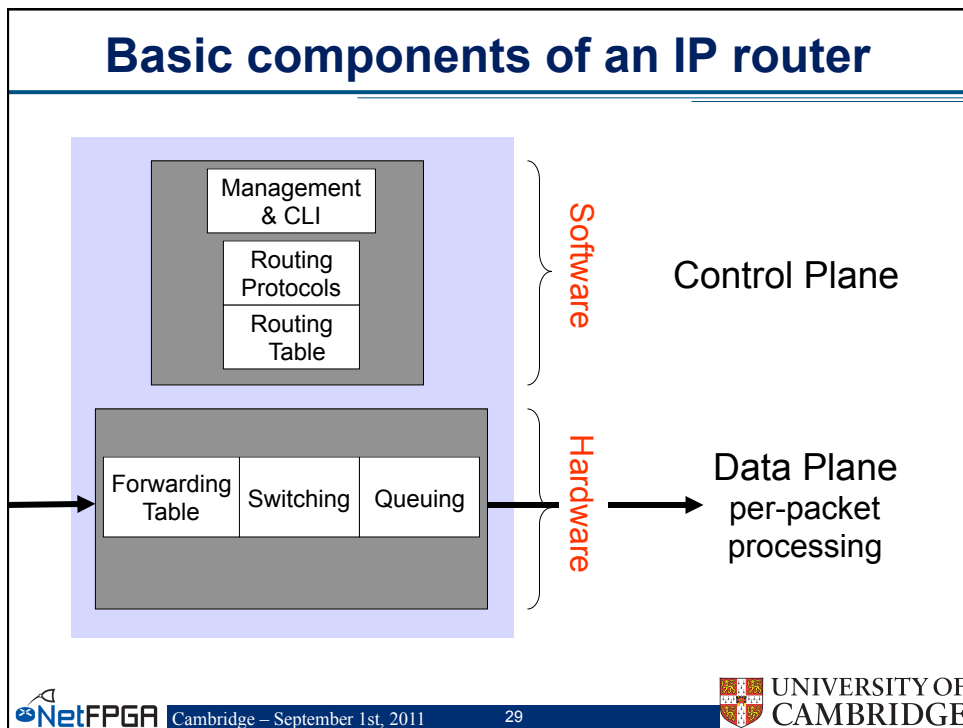
| Entry | Destination | Port | |
|-------|----------------------|------|-----------|
| 1 | Stanford | 1 | Searching |
| 2 | Berkeley | 2 | |
| 3 | North America | 3 | |
| 4 | Asia | 4 | FOUND |
| 5 | Everywhere (default) | 5 | |

Most specific

↓

Least specific

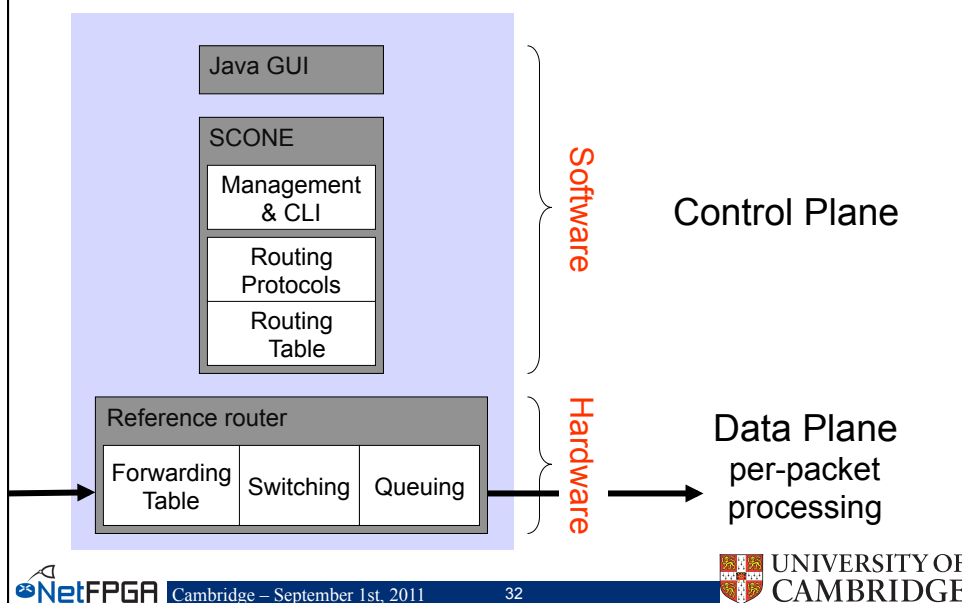

Cambridge – September 1st, 2011
28

UNIVERSITY OF CAMBRIDGE



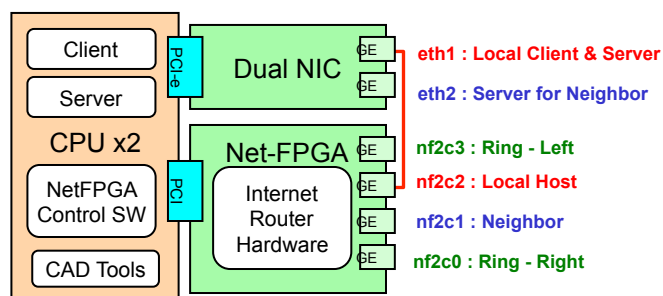
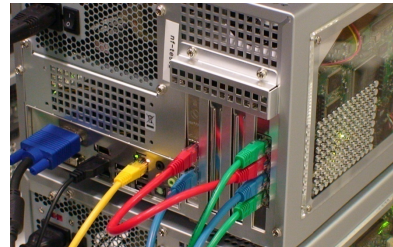
Example I

Reference Router running on the NetFPGA

Operational IPv4 router

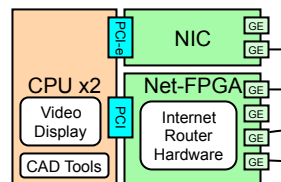
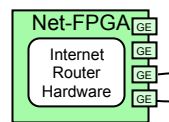
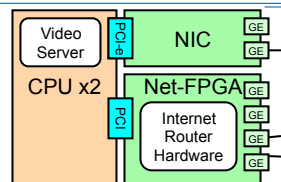


NetFPGA Lab Setup



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

UNIVERSITY OF CAMBRIDGE

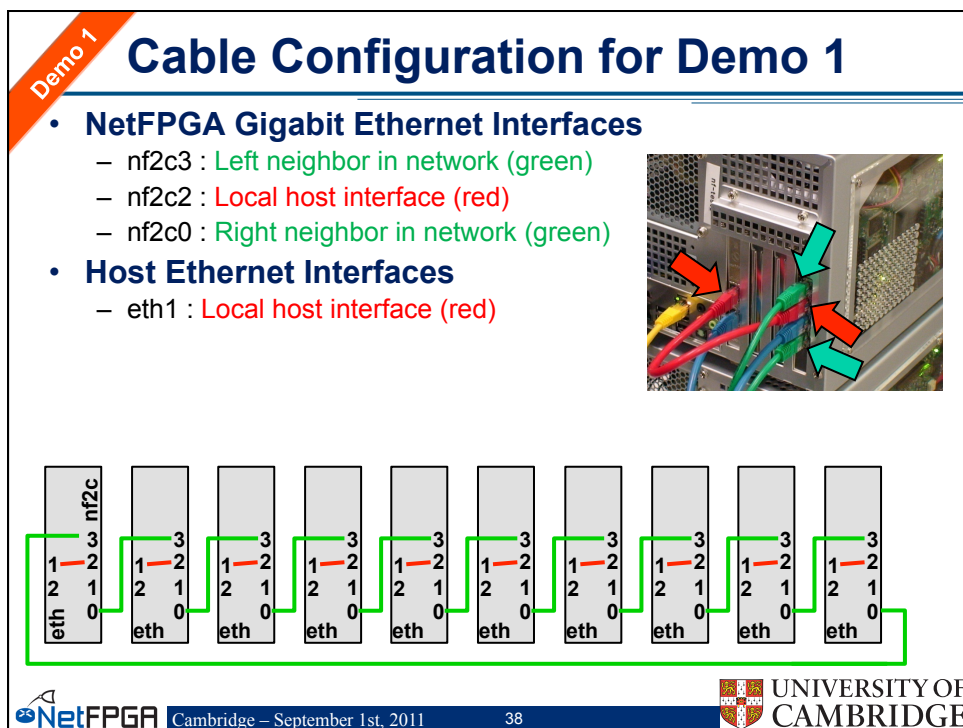
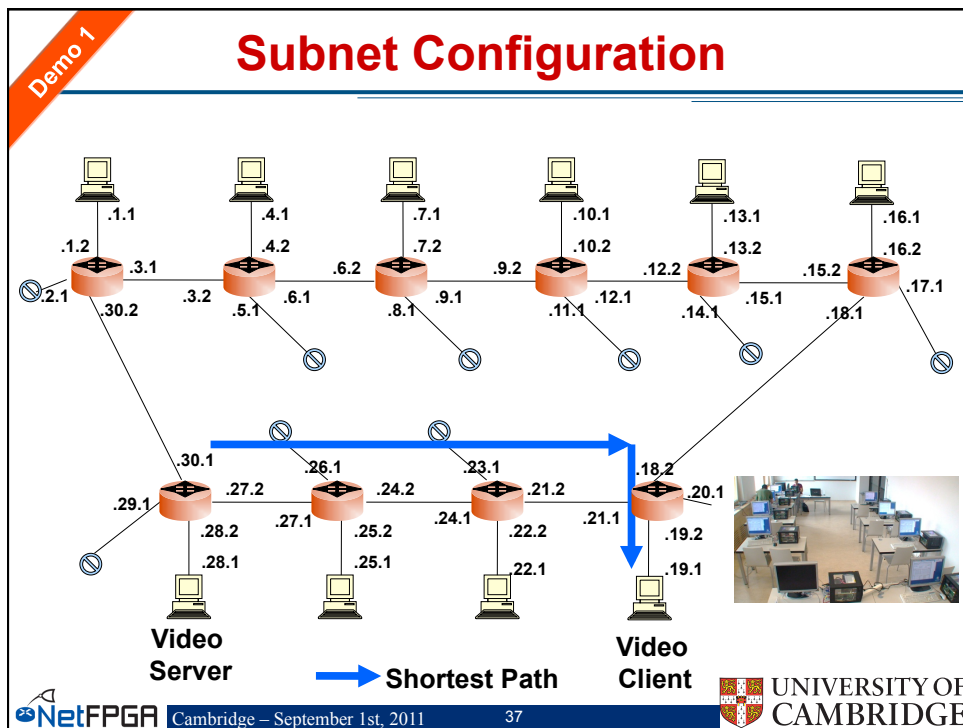
NetFPGA Cambridge – September 1st, 2011 35

Demo 1

Topology of NetFPGA Routers

UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 36



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

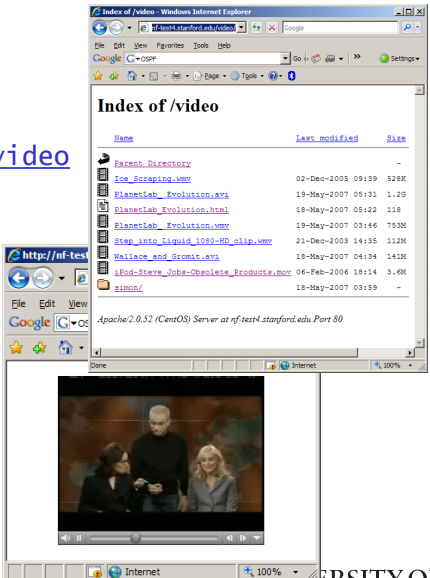
Working IP Router

• 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 mp1ayer
- **Video traffic**
 - MPEG2 HDTV (35 Mbps)
 - MPEG2 TV (9 Mbps)
 - DVI (3 Mbps)
 - WMF (1.7 Mbps)



UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 41

Demo 1 Physical Configuration

Key:

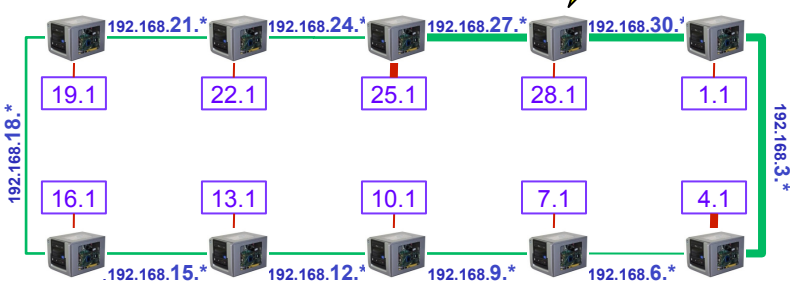
eth1 of Host PC
192.168.X.Y

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



UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 42

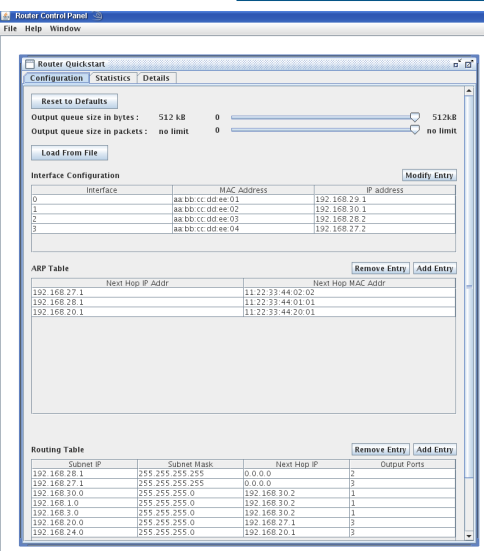
Demo 1

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 ...



The screenshot shows the Router Control Panel with the following data:

| Interface | MAC Address | IP Address |
|-----------|-------------------|--------------|
| 0 | aa:bb:cc:dd:ee:01 | 192.168.29.1 |
| 1 | aa:bb:cc:dd:ee:02 | 192.168.30.1 |
| 2 | aa:bb:cc:dd:ee:03 | 192.168.28.2 |
| 3 | aa:bb:cc:dd:ee:04 | 192.168.27.2 |

| Next Hop IP Addr | Next Hop MAC Addr |
|------------------|-------------------|
| 192.168.27.1 | 11:22:33:44:02:02 |
| 192.168.30.1 | 11:22:33:44:02:01 |
| 192.168.20.1 | 11:22:33:44:20:01 |

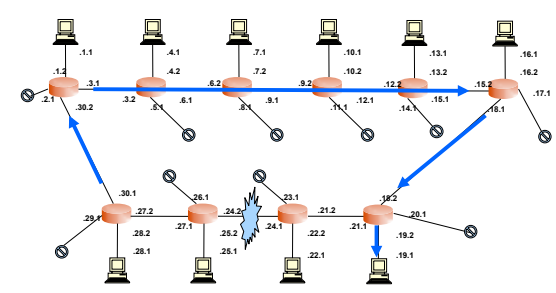
| Subnet IP | Subnet Mask | Next Hop IP | Output Ports |
|--------------|-----------------|--------------|--------------|
| 192.168.28.1 | 255.255.255.255 | 0.0.0.0 | 2 |
| 192.168.27.1 | 255.255.255.255 | 0.0.0.0 | 3 |
| 192.168.30.0 | 255.255.255.0 | 192.168.30.2 | 1 |
| 192.168.1.0 | 255.255.255.0 | 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 | 192.168.27.1 | 3 |
| 192.168.24.0 | 255.255.255.0 | 192.168.20.1 | 3 |

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



The diagram shows a network topology with several routers and clients. A blue link between two routers is broken, indicated by a lightning bolt. The routing table below shows the updated paths:

| 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 |

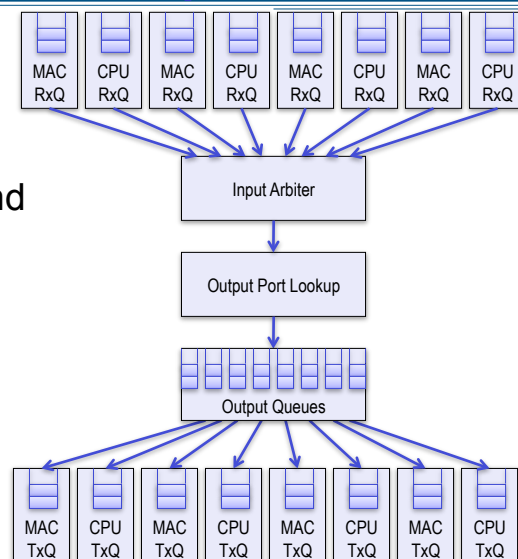
Exercise 1

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

 Cambridge – September 1st, 2011 47  UNIVERSITY OF CAMBRIDGE

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

 Cambridge – September 1st, 2011 49  UNIVERSITY OF CAMBRIDGE

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

The screenshot shows a terminal window with the following output:

```

jnaous@jadsdesktop:~/Desktop$ ping 192.168.17.1
PING 192.168.17.1 (192.168.17.1) 56(84) bytes of data:
64 bytes from 192.168.17.1: icmp_seq=1 ttl=64 time=0.047 ms
64 bytes from 192.168.17.1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 192.168.17.1: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 192.168.17.1: icmp_seq=4 ttl=64 time=0.044 ms
64 bytes from 192.168.17.1: icmp_seq=5 ttl=64 time=0.040 ms
64 bytes from 192.168.17.1: icmp_seq=6 ttl=64 time=0.036 ms
  
```

The Router Control Panel window shows the 'Statistics' tab with four graphs: Port 0 Pkts Rcvd, Port 0 Pkts Sent, Port 1 Pkts Rcvd, and Port 1 Pkts Sent. The 'Details' tab is also visible.

NetFPGA Cambridge – September 1st, 2011 50 **UNIVERSITY OF CAMBRIDGE**

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

The screenshot shows the 'Details' tab of the Router Quickstart window. It displays a diagram of the router pipeline with the following components:

- Input Addreser
- Output Port Lookup
- Output Queues

The diagram also shows MAC and CPU registers for both RX and TX sides.

NetFPGA Cambridge – September 1st, 2011 51 **UNIVERSITY OF CAMBRIDGE**

Exercise 1

Step 6 - Explore Output Queues

Click on the Output Queues module in the Details tab

The page gives configuration details

...and statistics

NetFPGA Cambridge – September 1st, 2011 52 **UNIVERSITY OF CAMBRIDGE**

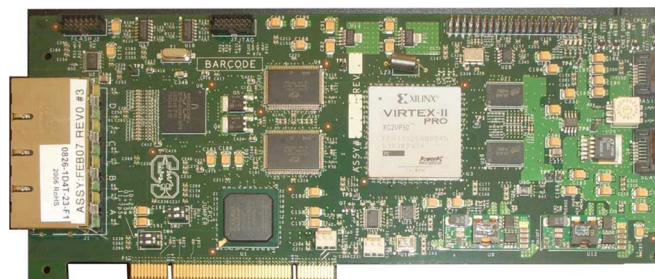
First Break

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

NetFPGA Cambridge – September 1st, 2011 53 **UNIVERSITY OF CAMBRIDGE**

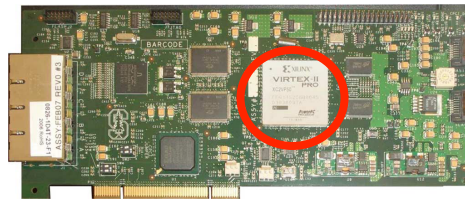
Hardware Overview

NetFPGA-1G



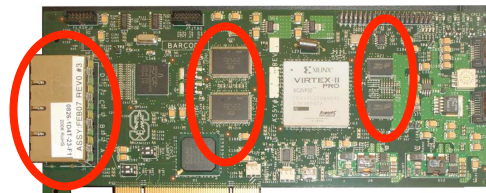
Xilinx Virtex II Pro 50

- 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



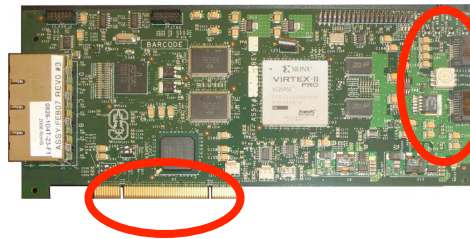
Other IO

•PCI

- Memory Mapped Registers
- DMA Packet Transferring

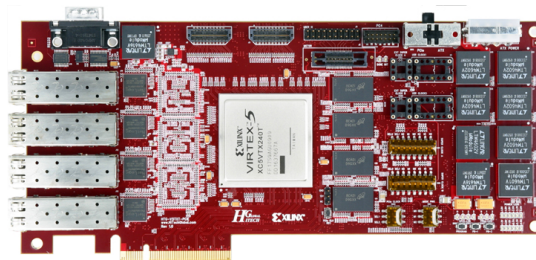
•SATA

- Board to Board communication



NetFPGA-10G

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

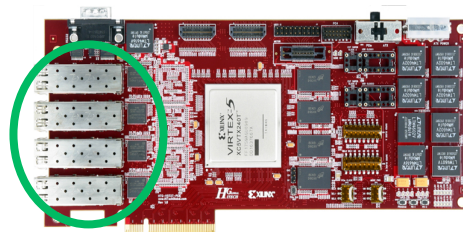


Comparison

| 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 |

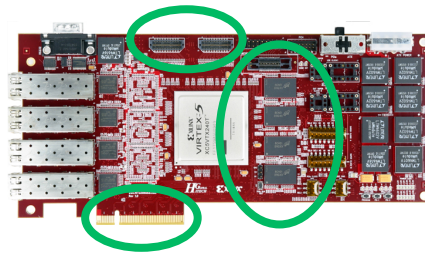
10 Gigabit Ethernet

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



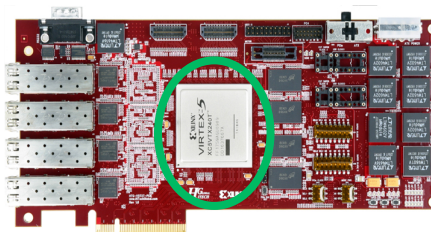
Others

- **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

PBWorks, GitHub, User Community

MicroBlaze SW

PC SW

Xilinx EDK

Reference Designs

AXI4 IPs



- 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

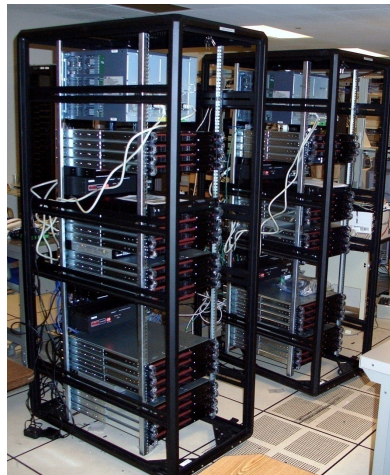
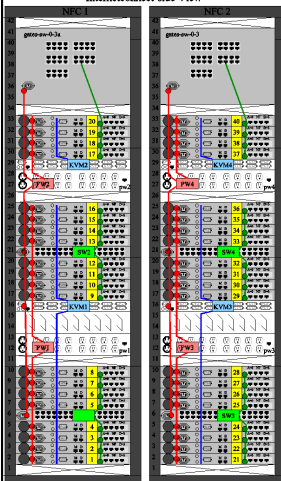


1U Server
(Accent Technology Inc.)

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

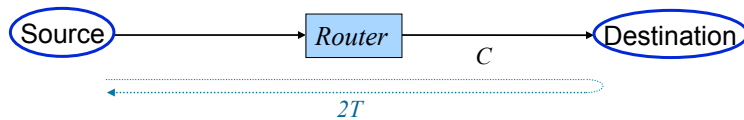
Theoretical tools predict requirements

- 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: $W \leftarrow W+1/W$
- If a packet is lost: $W \leftarrow W/2$

Cambridge – September 1st, 2011
72
UNIVERSITY OF CAMBRIDGE

Why 2TxC for a single TCP Flow?

Continuous ARQ (TCP) adapting to congestion

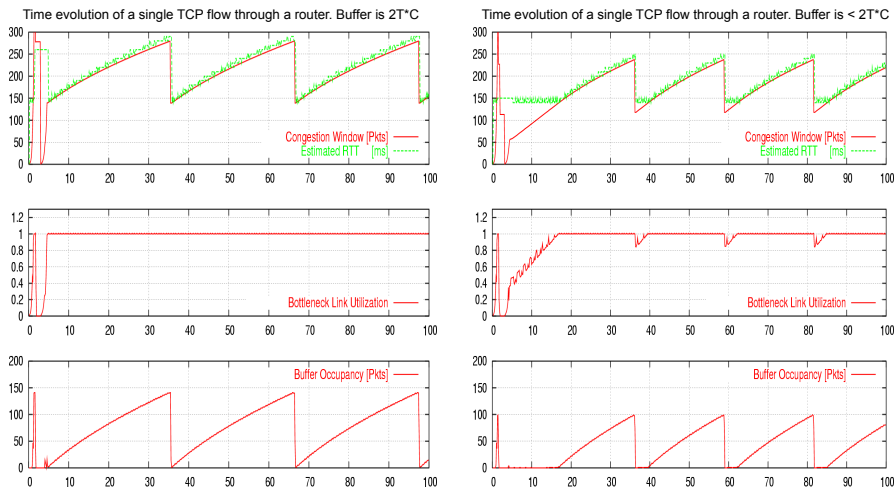
Only W packets
may be outstanding

Rule for adjusting W

- If an ACK is received: $W \leftarrow W+1/W$
- If a packet is lost: $W \leftarrow W/2$

Cambridge – September 1st, 2011
73
UNIVERSITY OF CAMBRIDGE

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

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

The diagram illustrates the Reference Router Pipeline. At the top, there are eight input queues, each containing a MAC RxQ and a CPU RxQ. Arrows from these queues point to a central 'Input Arbiter' block. Below the arbiter is an 'Output Port Lookup' block. This leads to a row of eight 'Output Queues'. From each of these queues, arrows point to a corresponding output queue at the bottom, each containing a MAC TxQ and a CPU TxQ.

Cambridge – September 1st, 2011
78
UNIVERSITY OF CAMBRIDGE

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

The diagram illustrates the Enhanced Router Pipeline. It follows the same structure as the reference pipeline but with two additions. An 'Event Capture' block is inserted between the 'Output Port Lookup' and the 'Output Queues'. A 'Rate Limiter' block is placed between the 'Output Queues' and the output queues. Arrows from the 'Event Capture' and 'Rate Limiter' blocks point to their respective descriptions in the text on the left.

Cambridge – September 1st, 2011
79
UNIVERSITY OF CAMBRIDGE

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)

```

user_data_path.v
wire [0:REG_ADDR_WIDTH-1:0] oq_reg_addr;
wire [31:0] oq_reg_wn_data;

wire [2:0] oq_signals;
wire [ALL_SIGNAL_IDS_SIZE-1:0] oq_signal_ids;
wire [CTRL_NPCS_DATA_WIDTH*NUM_OUTPUT_QUEUES-1:0] oq_obs_regs;
wire [ALL_SIG_VALUES_SIZE-1:0] oq_signal_values;

// ----- EXCLUDED -----
// new wires - uncomment these
// ----- event capture wires/regs -----
wire [CTRL_WIDTH-1:0] evt_cap_in_ctrl;
wire [DATA_WIDTH-1:0] evt_cap_in_data;
wire evt_cap_in_wr;
wire evt_cap_in_rd;
wire evt_cap_reg_ack;
wire [31:0] evt_cap_reg_rd_data;
wire evt_cap_reg_req;

// -----
UTFB--L297 C14 37% user_data_path.v (...izing/src/udp) (Verilog Panel)
Mark saved where search started
    
```

```

user_data_path.v
wire [0:REG_ADDR_WIDTH-1:0] oq_reg_addr;
wire [31:0] oq_reg_wn_data;

wire [2:0] oq_signals;
wire [ALL_SIGNAL_IDS_SIZE-1:0] oq_signal_ids;
wire [CTRL_NPCS_DATA_WIDTH*NUM_OUTPUT_QUEUES-1:0] oq_obs_regs;
wire [ALL_SIG_VALUES_SIZE-1:0] oq_signal_values;

// new wires - uncomment these
// ----- event capture wires/regs -----
// ----- EXCLUDED -----
wire [CTRL_WIDTH-1:0] evt_cap_in_ctrl;
wire [DATA_WIDTH-1:0] evt_cap_in_data;
wire evt_cap_in_wr;
wire evt_cap_in_rd;
wire evt_cap_reg_ack;
wire [31:0] evt_cap_reg_rd_data;
wire evt_cap_reg_req;

// -----
UTFB--L298 C15 37% user_data_path.v (...izing/src/udp) (Verilog Panel)
    
```

OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 82

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)

```

user_data_path.v
// comment the next 4 lines
// ----- EXCLUDED -----
(.out_data (oq_in_data),
.out_ctrl (oq_in_ctrl),
.out_wn (oq_in_wr),
.out_rd (oq_in_rd),
// ----- EXCLUDED -----
// opl_output - uncomment these lines
(out_data (evt_cap_in_data),
.out_ctrl (evt_cap_in_ctrl),
.out_wn (evt_cap_in_wr),
.out_rd (evt_cap_in_rd),
// ----- EXCLUDED -----
// -----
UTFB--L390 C18 66% user_data_path.v (...izing/src/udp) (Verilog Panel)
    
```

```

user_data_path.v
// comment the next 4 lines
// ----- EXCLUDED -----
(.out_data (oq_in_data),
.out_ctrl (oq_in_ctrl),
.out_wn (oq_in_wr),
.out_rd (oq_in_rd),
// ----- EXCLUDED -----
// opl_output - uncomment these lines
(out_data (evt_cap_in_data),
.out_ctrl (evt_cap_in_ctrl),
.out_wn (evt_cap_in_wr),
.out_rd (evt_cap_in_rd),
// ----- EXCLUDED -----
// -----
UTFB--L403 C15 66% user_data_path.v (...izing/src/udp) (Verilog Panel)
    
```

OF CAMBRIDGE

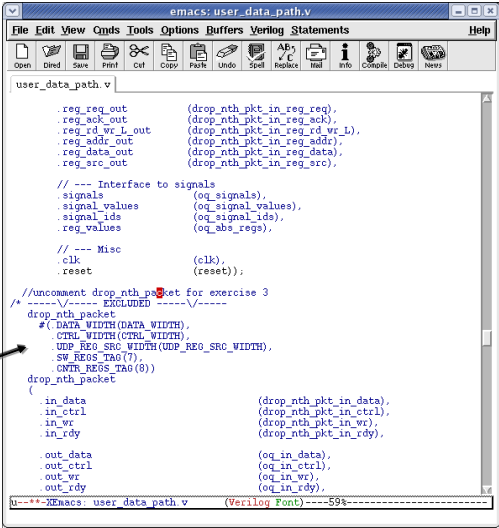
NetFPGA Cambridge – September 1st, 2011 83

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)



```

user_data_path.v
...
drop_nth_packet
  #(DATA_WIDTH(DATA_WIDTH),
    CTRL_WIDTH(CTRL_WIDTH),
    UDP_REG_SRC_WIDTH(UDP_REG_SRC_WIDTH),
    SW_REGS_TAO(7),
    CNTR_REGS_TAO(8))
  drop_nth_packet
  (
    .in_data          (drop_nth_pkt_in_data),
    .in_ctrl         (drop_nth_pkt_in_ctrl),
    .in_wr           (drop_nth_pkt_in_wr),
    .in_rdy         (drop_nth_pkt_in_rdy),
    .out_data        (eq_in_data),
    .out_ctrl        (eq_in_ctrl),
    .out_wr          (eq_in_wr),
    .out_rdy         (eq_in_rdy)
  )
...

```

UNIVERSITY OF CAMBRIDGE

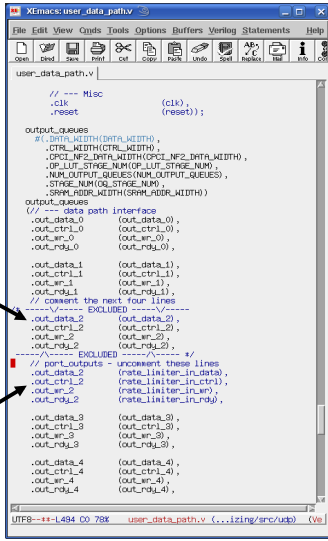
Exercise 2

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



```

user_data_path.v
...
output_queues
  #(DATA_WIDTH(DATA_WIDTH),
    CTRL_WIDTH(CTRL_WIDTH),
    OP1_NF2_DATA_WIDTH(OP1_NF2_DATA_WIDTH),
    OP1_LUT_STAGE_NUM(OP1_LUT_STAGE_NUM),
    NUM_OUTPUT_QUEUES(NUM_OUTPUT_QUEES),
    STAGE_NUM(O2_STAGE_NUM),
    SWP_CTRL_WIDTH(SWP_CTRL_WIDTH))
output_queues
  (
    .out_data_0      (out_data_0),
    .out_ctrl_0      (out_ctrl_0),
    .out_wr_0        (out_wr_0),
    .out_rdy_0       (out_rdy_0),
    .out_data_1      (out_data_1),
    .out_ctrl_1      (out_ctrl_1),
    .out_wr_1        (out_wr_1),
    .out_rdy_1       (out_rdy_1),
    .out_data_2      (out_data_2),
    .out_ctrl_2      (out_ctrl_2),
    .out_wr_2        (out_wr_2),
    .out_rdy_2       (out_rdy_2),
    .out_data_3      (out_data_3),
    .out_ctrl_3      (out_ctrl_3),
    .out_wr_3        (out_wr_3),
    .out_rdy_3       (out_rdy_3),
    .out_data_4      (out_data_4),
    .out_ctrl_4      (out_ctrl_4),
    .out_wr_4        (out_wr_4),
    .out_rdy_4       (out_rdy_4)
  )
...

```

UNIVERSITY OF CAMBRIDGE

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)

```

user_data_path v
(out_ctrl_7),
(out_wr_7),
(out_rdy_7),

// --- Interface to the previous module
.in_data (eq_in_data),
.in_ctrl (eq_in_ctrl),
.in_rdy (eq_in_rdy),
.in_wr (eq_in_wr),

// --- Register interface
.reg_req_in (eq_in_reg_req),
.reg_ack_in (eq_in_reg_ack),
.reg_rd_wr_l_in (eq_in_reg_rd_wr_l),
.reg_addr_in (eq_in_reg_addr),
.reg_data_in (eq_in_reg_data),
.reg_src_in (eq_in_reg_src),

// comment the next six lines
.reg_req_out (udp_req_eq_in),
.reg_ack_out (udp_req_ack_in),
.reg_rd_wr_l_out (udp_req_rd_wr_l_in),
.reg_addr_out (udp_req_addr_in),
.reg_data_out (udp_req_data_in),
.reg_src_out (udp_req_src_in),

// port_outputs - uncomment these lines
//----- EXCLUDED -----\/-----
.reg_req_out (rate_limiter_in_req),
.reg_ack_out (rate_limiter_in_ack),
.reg_rd_wr_l_out (rate_limiter_in_rd_wr_l),
.reg_addr_out (rate_limiter_in_addr),
.reg_data_out (rate_limiter_in_data),
.reg_src_out (rate_limiter_in_src),

//----- EXCLUDED -----\/----- */

// --- SRAM as interface
.wr_0_addr (wr_0_addr),
.wr_0_req (wr_0_req),
.wr_0_ack (wr_0_ack),
.wr_0_data (wr_0_data),
.rd_0_ack (rd_0_ack),
.rd_0_data (rd_0_data),
.rd_0_ctrl (rd_0_ctrl),
.rd_0_addr (rd_0_addr),
.rd_0_req (rd_0_req),

.sq_abs_reqs (eq_abs_reqs),
.sq_signals (eq_signals),
.sq_signal_ids (eq_signal_ids),
.sq_signal_values (eq_signal_values),

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

// uncomment the modules here
//----- Xilinx: user_data_path.v (Verilog Font)-----748-----
print not defined
    
```

UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 88

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)

```

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

// uncomment the modules here
rate_limiter #(DATA_WIDTH(DATA_WIDTH),
               .CPCL_INF2_DATA_WIDTH(CPCL_INF2_DATA_WIDTH))
(out_data (delay_in_data),
 out_ctrl (delay_in_ctrl),
 out_wr (delay_in_wr),
 out_rdy (delay_in_rdy),

.in_data (rate_limiter_in_data),
.in_ctrl (rate_limiter_in_ctrl),
.in_wr (rate_limiter_in_wr),
.in_rdy (rate_limiter_in_rdy),

// --- Register interface
.rate_limiter_reg_req (rate_limiter_reg_req),
.rate_limiter_reg_rd_wr_l (rate_limiter_reg_rd_wr_l),
.rate_limiter_reg_addr (rate_limiter_reg_addr),
.rate_limiter_reg_data (rate_limiter_reg_data),
.rate_limiter_reg_ack (rate_limiter_reg_ack),

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

UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 89

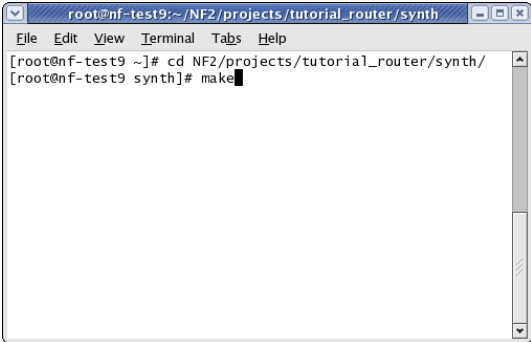
Exercise 2

Step 9 - Build the Hardware



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

Run “make clean”

Start synthesis with “make”





```
root@nf-test9:~/NF2/projects/tutorial_router/synth
File Edit View Terminal Tabs Help
[root@nf-test9 ~]# cd NF2/projects/tutorial_router/synth/
[root@nf-test9 synth]# make
```

 Cambridge – September 1st, 2011 90  UNIVERSITY OF CAMBRIDGE

Second Break

(while hardware compiles)

Lunch is downstairs

 Cambridge – September 1st, 2011 91  UNIVERSITY OF CAMBRIDGE

Exercise 2

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

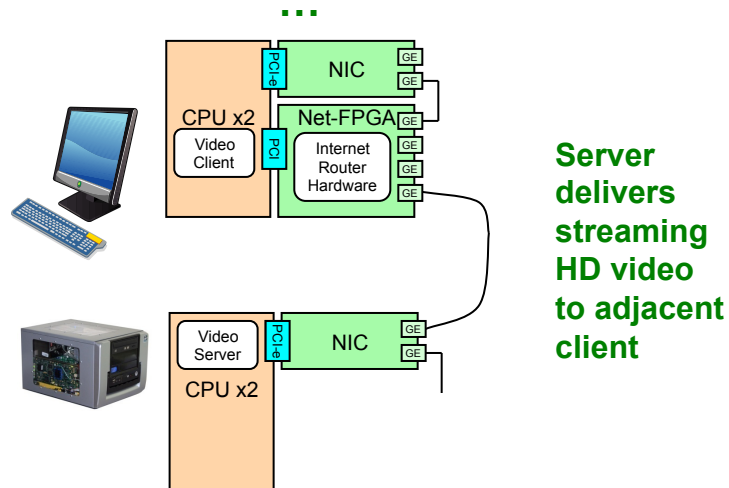
Using NetFPGA to explore buffer size

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

Objective:

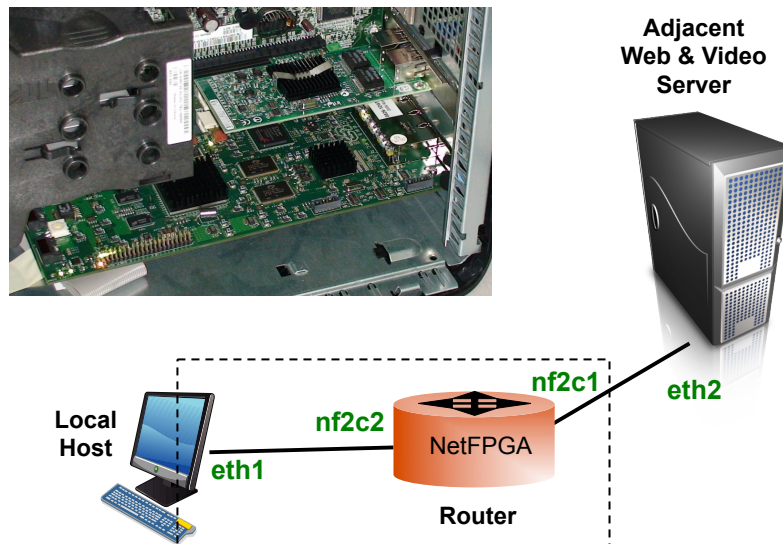
- Use the NetFPGA to understand how large a buffer we need for a **single** TCP flow.

NetFPGA Hardware Set for Exercise #2



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

Cambridge – September 1st, 2011
96
UNIVERSITY OF CAMBRIDGE

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)

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

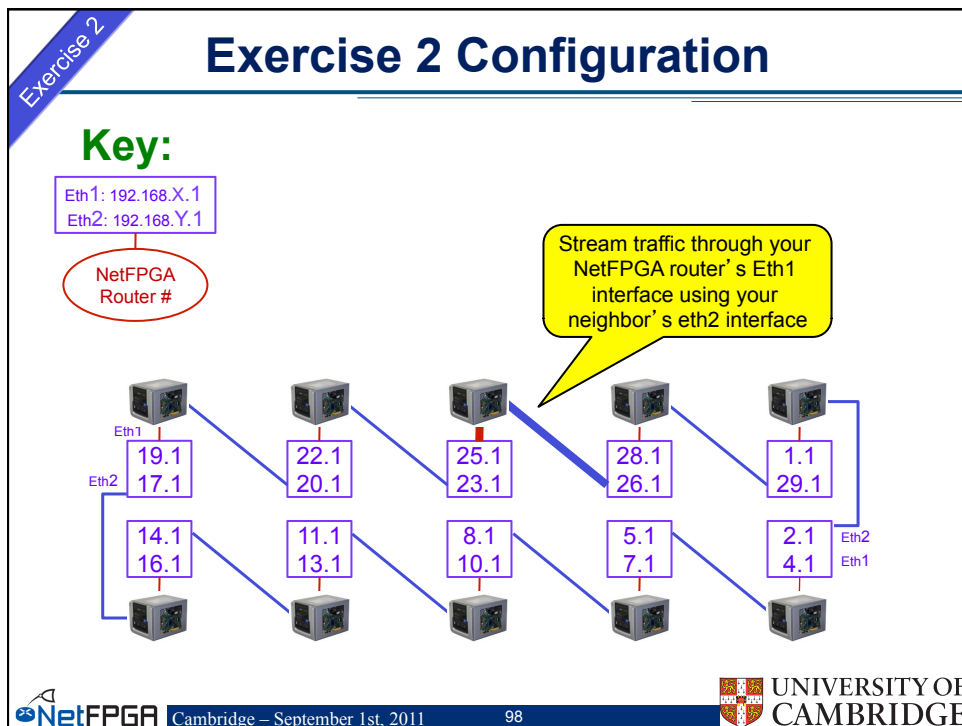
| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

| | |
|------|---|
| nf2c | 3 |
| 1-2 | 3 |
| 2-1 | 0 |
| eth | 0 |

Cambridge – September 1st, 2011
97
UNIVERSITY OF CAMBRIDGE



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

NetFPGA Cambridge – September 1st, 2011 99 UNIVERSITY OF CAMBRIDGE

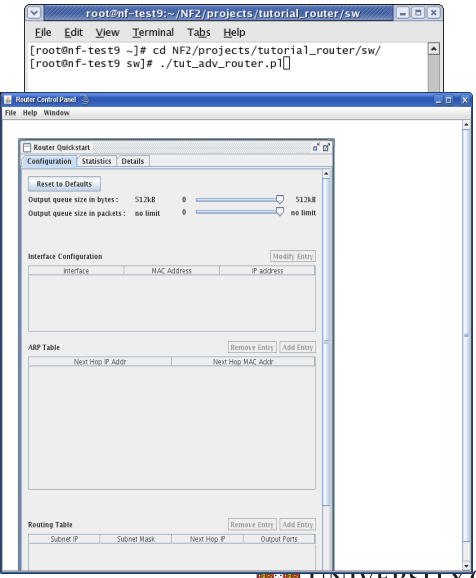
Exercise 2



Step 1 - Run Your Enhanced Router

Start terminal and cd to
 NF2/projects/
 tutorial_router/sw/

Type ↙ NB: ..._ADV_...
 ./tut_adv_router_gui.pl -use_bin\
 ../../bitfiles/tutorial_router.bit

A familiar GUI should start



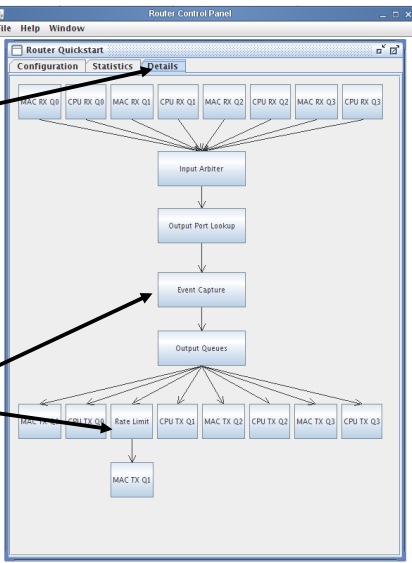
 Cambridge – September 1st, 2011 100  UNIVERSITY OF CAMBRIDGE



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



 Cambridge – September 1st, 2011 101  UNIVERSITY OF CAMBRIDGE

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

NetFPGA Cambridge – September 1st, 2011 102 UNIVERSITY OF CAMBRIDGE

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

NetFPGA Cambridge – September 1st, 2011 103 UNIVERSITY OF CAMBRIDGE

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

UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 104

Exercise 2

Step 5 - Start Event Capture

Click on the Event Capture module under the Details tab

This should start the configuration page

UNIVERSITY OF CAMBRIDGE

NetFPGA Cambridge – September 1st, 2011 105

Exercise 2

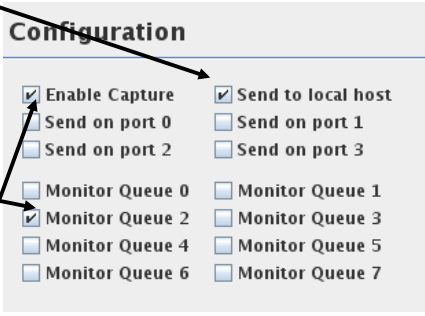
Step 6 - Configure Event Capture

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

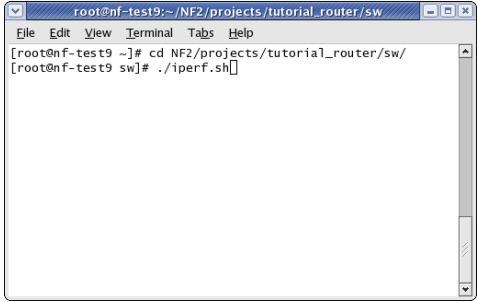


NetFPGA Cambridge – September 1st, 2011 106 UNIVERSITY OF CAMBRIDGE

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”

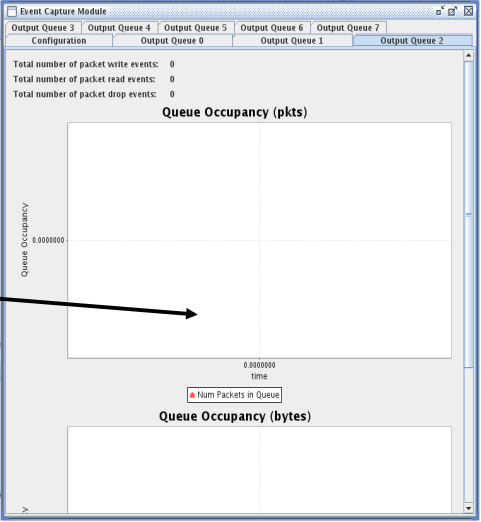
NetFPGA Cambridge – September 1st, 2011 107 UNIVERSITY OF CAMBRIDGE

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.



Event Capture Module

Output Queue 3 | Output Queue 4 | Output Queue 5 | Output Queue 6 | Output Queue 7
 Configuration | Output Queue 0 | Output Queue 1 | Output Queue 2

Total number of packet write events: 0
 Total number of packet read events: 0
 Total number of packet drop events: 0

Queue Occupancy (pkts)

Queue Occupancy

0.0000000

0.0000000

time


Num Packets in Queue

Queue Occupancy (bytes)

NetFPGA Cambridge – September 1st, 2011 108 UNIVERSITY OF CAMBRIDGE

Queue Occupancy Charts

Observe the TCP/IP sawtooth – observe the BUFFER occupancy



Queue Occupancy (pkts)

Queue Occupancy

time

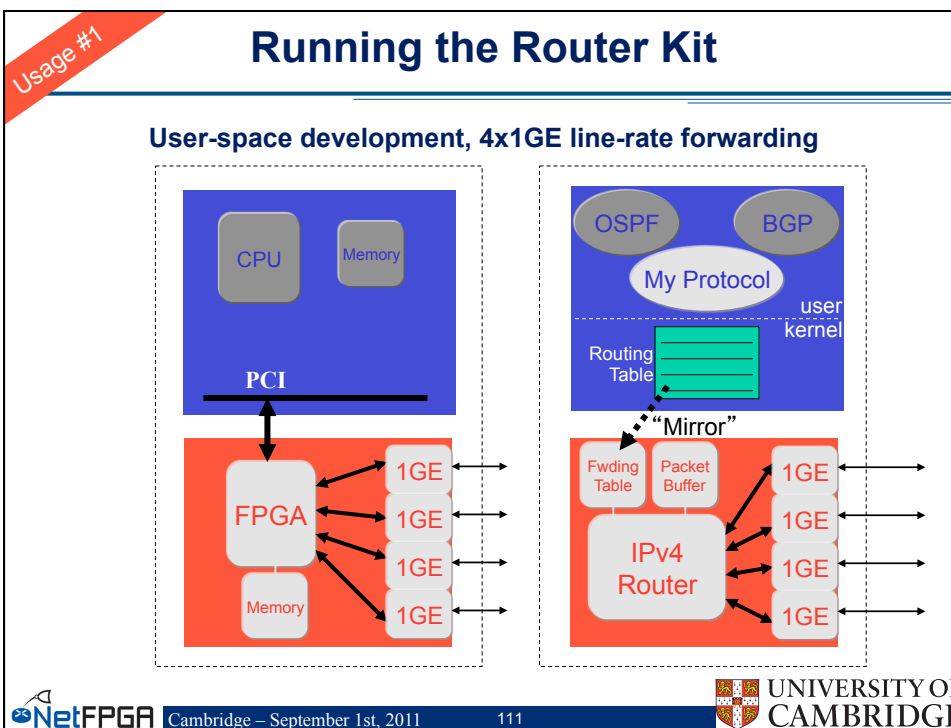
Num Packets in Queue

Queue Occupancy (bytes)

Leave the control windows open

NetFPGA Cambridge – September 1st, 2011 109 UNIVERSITY OF CAMBRIDGE

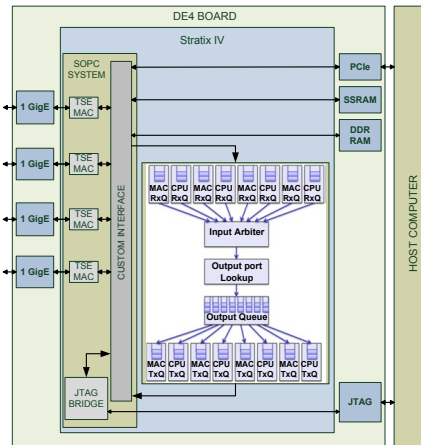
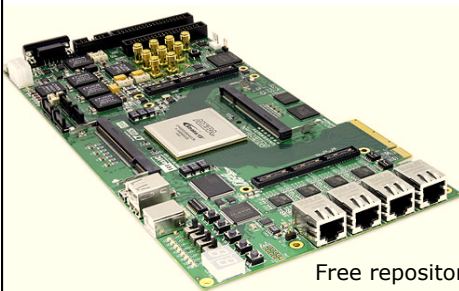
NetFPGA is a Community



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



Cambridge – September 1st, 2011

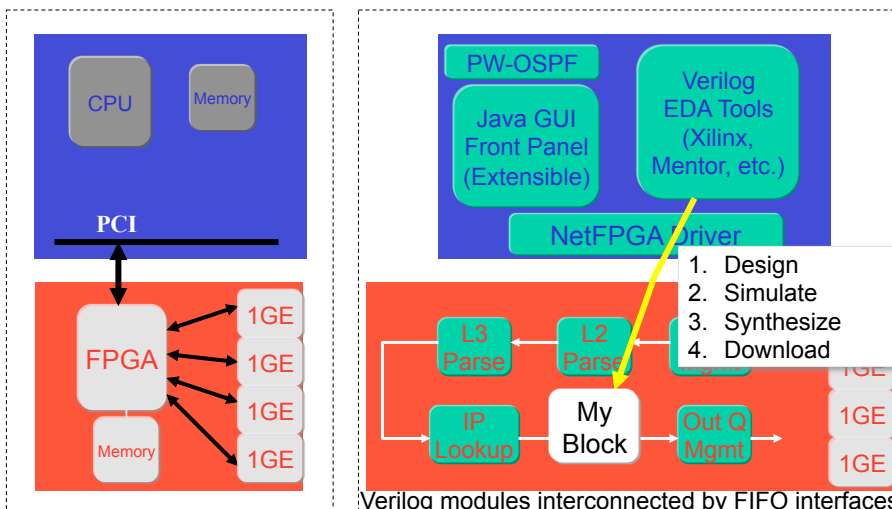
112



UNIVERSITY OF CAMBRIDGE

Usage #2

Enhancing Modular Reference Designs



Verilog modules interconnected by FIFO interfaces



Cambridge – September 1st, 2011

113



UNIVERSITY OF CAMBRIDGE


Usage #3

Creating new systems

1. Design
2. Simulate
3. Synthesize
4. Download

Cambridge – September 1st, 2011 114 UNIVERSITY OF CAMBRIDGE

NetThreads, NetThreads-RE, NetTM



U. of Toronto

Martin Labrecque
Gregory Steffan

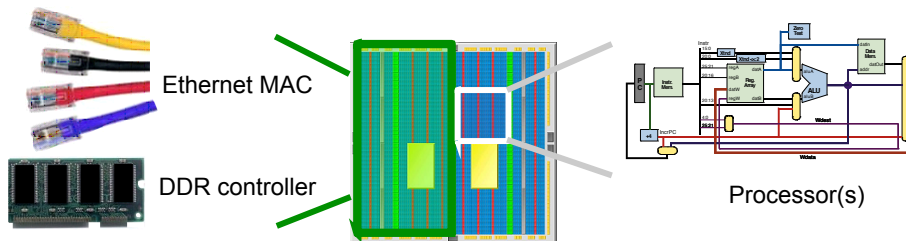
ECE Dept.

Geoff Salmon
Monia Ghobadi
Yashar Ganjali

CS Dept.

Cambridge – September 1st, 2011 115 UNIVERSITY OF CAMBRIDGE

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

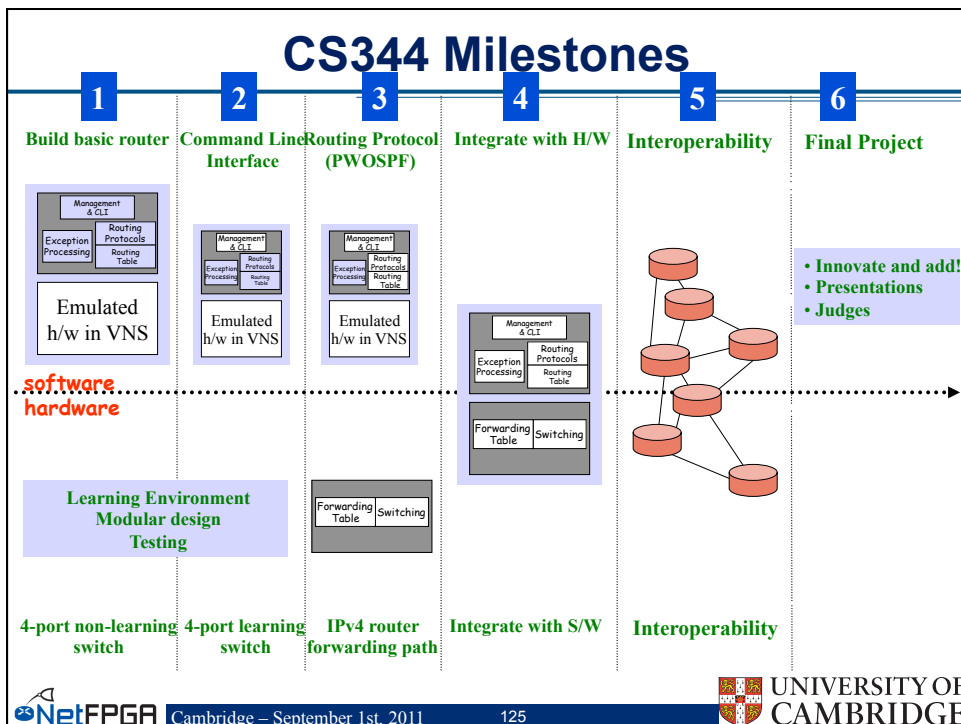
- **Stanford University**
 - EE109 "Build an Ethernet Switch"
 - Undergraduate course for all EE students
 - <http://www.stanford.edu/class/ee109/>
 - CS344 "Building an Internet Router" (since '05)
 - Quarter-long course targeted at graduates
 - <http://cs344.stanford.edu>
 - **Rice University**
 - 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/~akelia/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
 - Interoperability
- **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, ...



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 |



Cambridge – September 1st, 2011

126

UNIVERSITY OF
CAMBRIDGE

Presentations



Stanford CS344

<http://cs344.stanford.edu>


Cambridge P33

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


Cambridge – September 1st, 2011

127

UNIVERSITY OF
CAMBRIDGE

From our classroom to yours...

Exercise 3: Controlled packet-loss

Beyond just observation,
using NetFPGA for an experiment



Cambridge – September 1st, 2011

128

UNIVERSITY OF
CAMBRIDGE

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....



Cambridge – September 1st, 2011

129

UNIVERSITY OF
CAMBRIDGE

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.

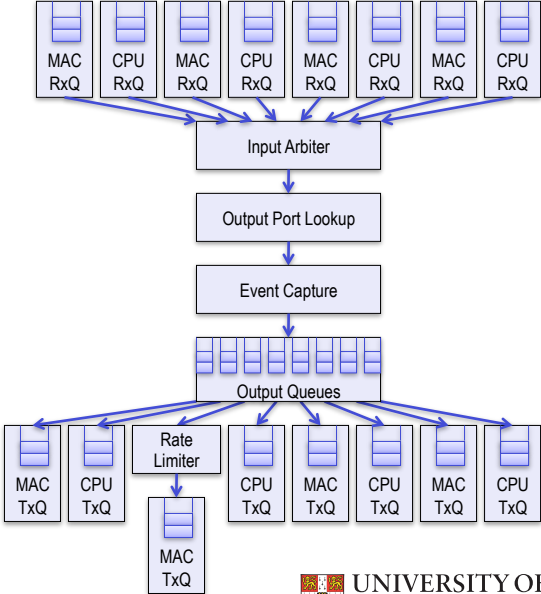
 Cambridge – September 1st, 2011
130
 UNIVERSITY OF CAMBRIDGE



Exercise 3

Our Enhanced Router Pipeline

One module added

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



 Cambridge – September 1st, 2011
131
 UNIVERSITY OF CAMBRIDGE

Exercise 3

New Even-More Enhanced Router Pipeline

One module added

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

UNIVERSITY OF CAMBRIDGE

Cambridge – September 1st, 2011 132

Exercise 3

Step 1 - Open the Source

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

```

module drop_nth_packet
// Src: drop_nth_packet 2009-03-15 gael s
// Project: NF2.1
// Description: defines a module that drops the nth packet
//-----
timescale 1ns/1ps
include "NF_2.1_defines.v"

module drop_nth_packet
#parameter DATA_WIDTH = 64,
#parameter CTRL_WIDTH = 8,
#parameter UDP_REG_SRC_WIDTH = 3,
#parameter SW_REGS_TAG = 4,
#parameter CNTR_REGS_TAG = 5)
input [DATA_WIDTH-1:0] in_data,
input [CTRL_WIDTH-1:0] in_ctrl,
input in_wr,
input in_rdy,
output [DATA_WIDTH-1:0] out_data,
output [CTRL_WIDTH-1:0] out_ctrl,
output out_wr,
output out_rdy,
// --- Register interface
input reg_req_in,
input reg_ack_in,
input [UDP_REG_ADDR_WIDTH-1:0] reg_rd_wr_in,
input [CPCI_NP2_DATA_WIDTH-1:0] reg_addr_in,
input [UDP_REG_SRC_WIDTH-1:0] reg_data_in,
output reg_req_out,
output reg_ack_out
endmodule
    
```

Open terminal
Type "emacs NF2/projects/tutorial_router/src/drop_nth_packet.v"

UNIVERSITY OF CAMBRIDGE

Cambridge – September 1st, 2011 133

Exercise 3

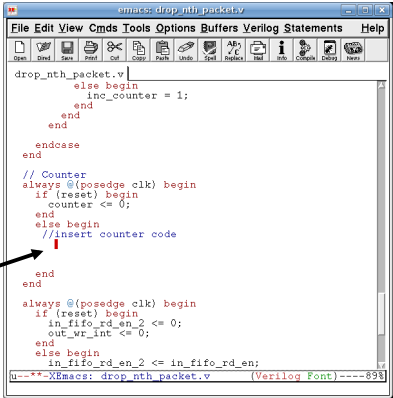
Step 2 - Add Counter to Module

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)



```

drop_nth_packet.v |
else begin
  inc_counter = 1;
end
endcase
end
// Counter
always @(posedge clk) begin
  if (reset) begin
    counter <= 0;
  end
  else begin
    //insert counter code
  end
end
always @(posedge clk) begin
  if (reset) begin
    in_fifo_rd_en_2 <= 0;
    out_wr_int <= 0;
  end
  else begin
    in_fifo_rd_en_2 <= in_fifo_rd_en;
  end
end

```

NetFPGA Cambridge – September 1st, 2011 134 UNIVERSITY OF CAMBRIDGE

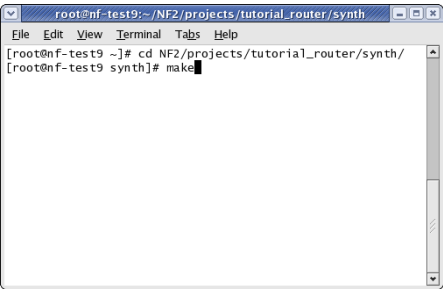
Exercise 3

Step 3 - Build the Hardware

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

Run “make clean”

Start synthesis with “make”



```

root@nf-test9:~/NF2/projects/tutorial_router/synth
[root@nf-test9 ~]# cd NF2/projects/tutorial_router/synth/
[root@nf-test9 synth]# make

```

NetFPGA Cambridge – September 1st, 2011 135 UNIVERSITY OF CAMBRIDGE

Third Break

(while hardware compiles)

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,)

 Cambridge – September 1st, 2011
 138
 UNIVERSITY OF CAMBRIDGE

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**

 Cambridge – September 1st, 2011
 139
 UNIVERSITY OF CAMBRIDGE

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

 Cambridge – September 1st, 2011 140  UNIVERSITY OF CAMBRIDGE

Conclusion

 Cambridge – September 1st, 2011 141  UNIVERSITY OF CAMBRIDGE

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:

| Project (Title & Summary) | Base System | Status | Organization | Documentation |
|-----------------------------------|-------------|------------|--------------------------|----------------|
| IP4 Reference Router | 2.1.1 | Functional | Stanford University | Guide |
| Quad-Pet Gigabit NIC | 2.1.1 | Functional | Stanford University | Guide |
| Ethernet Switch | 2.1.1 | Functional | Stanford University | Wiki |
| Buffer Monitoring System | 2.1.1 | Functional | Stanford University | Guide |
| Hardware-Accelerated Linux Router | 2.1.1 | Functional | Stanford University | Guide |
| DRAM Router | 2.1.1 | Functional | Stanford University | Wiki |
| DRAM Queue Test | 2.1.1 | Functional | Stanford University | Wiki |
| Packet Generator | 2.1.1 | Functional | Stanford University | Wiki |
| OpenFlow Switch | 2.0.0 | Functional | Stanford University | Wiki |
| NetFlow Probe | 1.2 | Functional | Brown University | Wiki |
| NetFPGA | 2.0 | Functional | Stanford University | Wiki and Paper |
| Fast Return & Multipath Router | 2.0 | Functional | Stanford University | Wiki |
| NetThreat | 1.2.5 | Functional | University of Toronto | Wiki |
| NetThreat-RE | 2.0 | Functional | University of Toronto | Wiki |
| NetDM | 2.0 | Functional | University of Toronto | Wiki |
| Precise Traffic Generator | 1.2.5 | Functional | University of Toronto | Wiki |
| URL Extraction | 2.0 | Functional | Univ. of New South Wales | Wiki |
| iFiber Sponsor (Pub-Sub) | 1.2 | Functional | Ericsson | Wiki |
| Wireless-Driver | 2.0 | Functional | Microsoft Research | Wiki |
| RED | 2.0 | Functional | Stanford University | Wiki |
| Open Network Lab | 2.0 | Functional | Washington University | Wiki |
| IRI | 2.0 | Functional | UMass Lowell | Wiki |
| GPX | 1.7 | Functional | Xilinx | Wiki |
| RCP Router | 2.0 | Functional | Stanford University | Wiki |

Networked FPGAs in Research

- Managed flow-table switch**
 - <http://OpenFlowSwitch.org/>
- Buffer Sizing**
 - Reduce buffer size & measure buffer occupancy
- RCP: Congestion Control**
 - New module for parsing and overwriting new packet
 - New software to calculate explicit rates
- Deep Packet Inspection (FPX)**
 - TCP/IP Flow Reconstruction
 - Regular Expression Matching
 - Bloom Filters
- Packet Monitoring (ICSI)**
 - Network Shunt
- 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/>



Cambridge – September 1st, 2011

146



UNIVERSITY OF
CAMBRIDGE

Learn by Yourself

The screenshot shows the NetFPGA website interface. The navigation menu on the left includes 'Learn', 'Develop', 'Community', 'Tools', and 'Web'. The 'Learn' section is circled in red. The main content area displays the 'Introduction' and 'Users Guide' sections. The 'Users Guide' section is highlighted in yellow and contains the text: 'Users Guide - for those that have just got their first NetFPGA board'. Below this, the 'NetFPGA website (www.netfpga.org)' is highlighted in green. The website header includes the NetFPGA logo and navigation links: Home, Applications, Events, News, Wiki, Forums, About.



Cambridge – September 1st, 2011

147



UNIVERSITY OF
CAMBRIDGE

Learn by Yourself

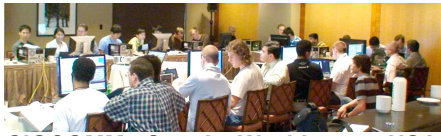
The screenshot shows the NetFPGA website interface. At the top, there is a navigation bar with links for Home, Applications, Events, News, Wiki, Forums, and About. The 'Forums' link is circled in red. Below the navigation bar, the page title is 'Forums'. The main content area is titled 'Developers Guide' and contains an attention notice about the 2.0 release, a table of contents, and an overview section. The overview section states: 'The NetFPGA platform consists of many elements including the physical hardware, hardware designs that are downloaded to the FPGA, software associated with a particular hardware design, general software tools for interacting with the hardware, and the simulation and synthesis environment for building new designs. Developers will most frequently develop new hardware designs to run on the FPGA and software for use with particular hardware projects.' A sidebar on the left contains various navigation links such as 'Learn', 'Development', 'Community', 'Tools', and 'Web'. The 'Developers Guide' title is highlighted in yellow. At the bottom of the screenshot, the text 'NetFPGA website (www.netfpga.org)' is displayed in a green box.

Learn by Yourself

Online tutor – coming soon!

The screenshot shows a web browser window displaying the 'Cambridge SystemVerilog Tutor for NetFPGA' page. The page features the University of Cambridge logo and a green header with the title. Below the header, there is a 'Getting Started' section with a welcome message. The page is divided into three columns: 'Cambridge Users' with a 'Login via Raven' button, 'External Users' with a 'Login' button, and 'Site information' with text about browser requirements and JavaScript/Flash usage. There is also a 'Note' section for users with raven accounts and a 'Register or Reset account' section with an 'Email Address' input field and a 'Register or Reset' button. The browser's address bar shows the URL 'http://www-netfpga.cl.cam.ac.uk/'.

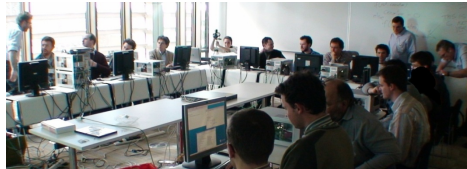
Photos from NetFPGA Tutorials



SIGCOMM - Seattle, Washington, USA



SIGMETRICS - San Diego, California, USA



EuroSys - Glasgow, Scotland, U.K.



Beijing, China



Bangalore, India

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



Cambridge – September 1st, 2011

150



UNIVERSITY OF
CAMBRIDGE

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



Cambridge – September 1st, 2011

151



UNIVERSITY OF
CAMBRIDGE

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 today's 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:



UNIVERSITY OF
CAMBRIDGE



SIGMETRICS



CESNET

NICTA



THE UNIVERSITY OF
NEW SOUTH WALES

UNIVERSITY
of
GLASGOW



Indian Institute of Science
Bangalore, India



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



Cambridge – September 1st, 2011

154



UNIVERSITY OF
CAMBRIDGE

Thanks to our Sponsors:

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



Agilent Technologies



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.



Cambridge – September 1st, 2011

155



UNIVERSITY OF
CAMBRIDGE

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)