NetFPGA Workshop
Day 1





Presented by:

Jad Naous (Stanford University)

Andrew W. Moore (Cambridge University)

Hosted by:
Manolis Katevenis
at
FORTH, Crete
September 15 - 16, 2010

http://NetFPGA.org





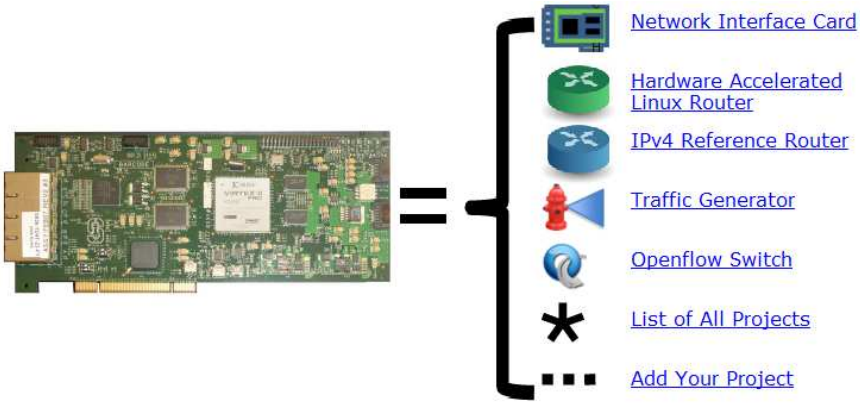
Tutorial Outline

- **Background**
  - Introduction
  - The NetFPGA Platform
- **The Stanford Base Reference Router**
  - Motivation: Basic IP review
  - Demo1: Reference Router running on the NetFPGA
- **The Enhanced Reference Router**
  - Motivation: Understanding buffer size requirements in a router
  - Demo 2: Observing and controlling the queue size
- **How does the NetFPGA work**
  - Utilities
  - Reference Designs
  - Inside the NetFPGA Hardware
- **The Life of a Packet Through the NetFPGA**
  - Hardware Datapath
  - Interface to software: Exceptions and Host I/O
- **Exercise: Drop Nth Packet**
- **Concluding Remarks**
  - Using NetFPGA for research and teaching

Section I: Motivation

What is the NetFPGA?

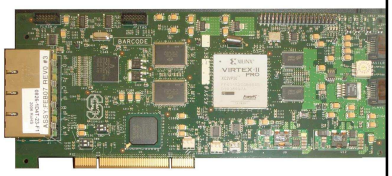
A line-rate, flexible, open networking platform for teaching and research



NetFPGA consists of…

Four elements:

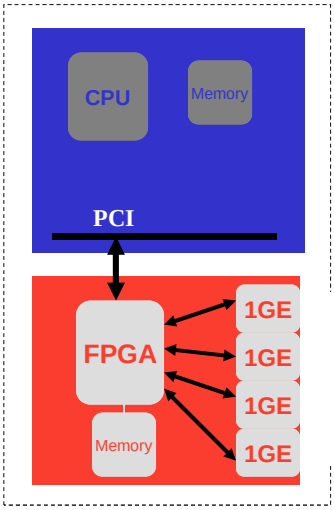
- NetFPGA board
- Tools + reference designs
- Contributed projects
- Community



NetFPGA Board

NetFPGA board

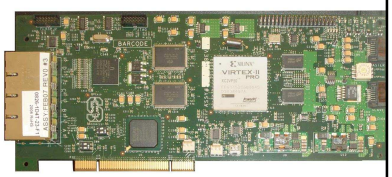
Networking Software running on a standard PC





PC with NetFPGA

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



NetFPGA Board

Tools + reference designs

Tools:

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

Reference designs:

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

Example 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 (slowly growing)**
- **Encourage users to contribute**

Forums

- **Support by users for users**
- **Active community – 10s to 100s of posts per week**

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

What you will learn

- **Overall picture of NetFPGA**
- **How reference designs work**
- **How you can work on a project**
  - NetFPGA Design Flow
  - Directory Structure, library modules and projects
  - How to utilize contributed projects
    - Interface/Registers
  - How to verify a design (Simulation and Regression Tests)
  - Things to do when you get stuck

AND… You can start your own projects!

Section II: Demo Basic Use

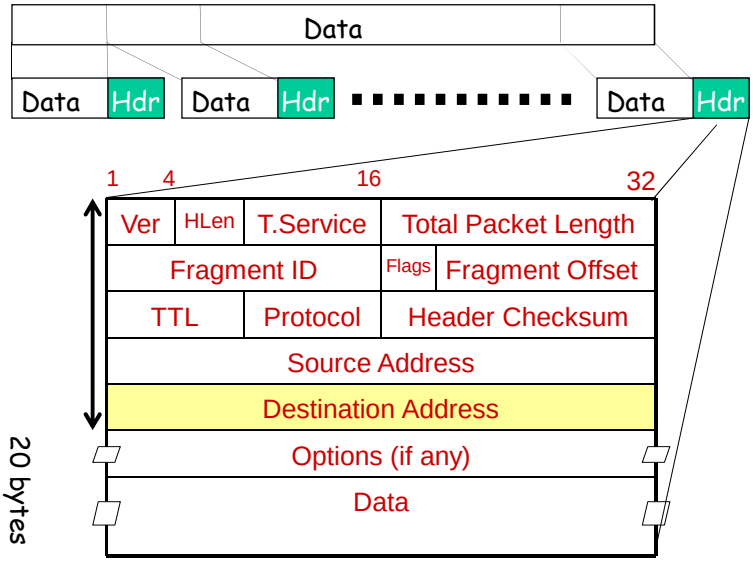
Basic Uses of NetFPGA

- **Recap Internet Protocol and Routing**
- **Demonstrate**
  - How you can use the NetFPGA as a router
  - See routing in action

What is IP?

- **IP (Internet Protocol)**
  - Protocol used for communicating data across packet-switched networks
  - Divides data into a number of packets (IP packet)
- **IP Packet**
  - Header (IP Header) including:
    - Source IP address
    - Destination IP address

IP Header

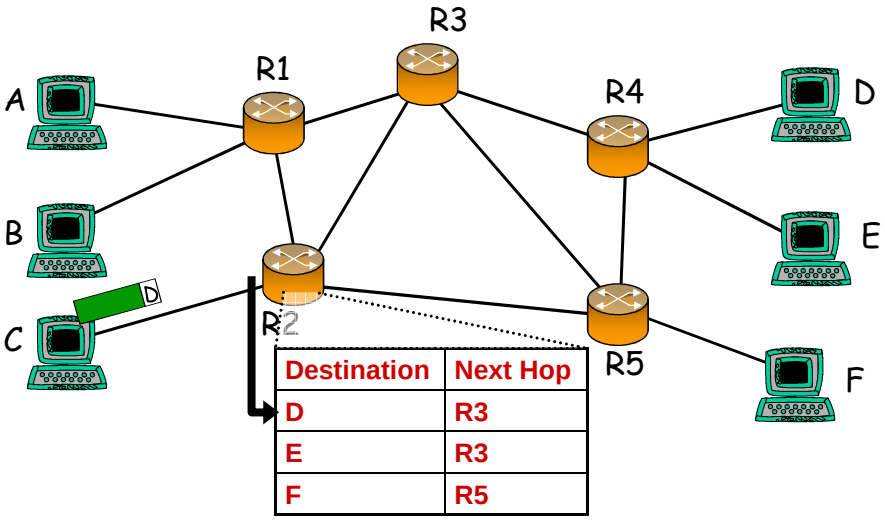


IP Address

- **Used to uniquely identify a device (such as a computer) from all other devices on a network**
  - Two parts
    - Identifier of a particular network on the Internet
    - Identifier of a particular device within a network

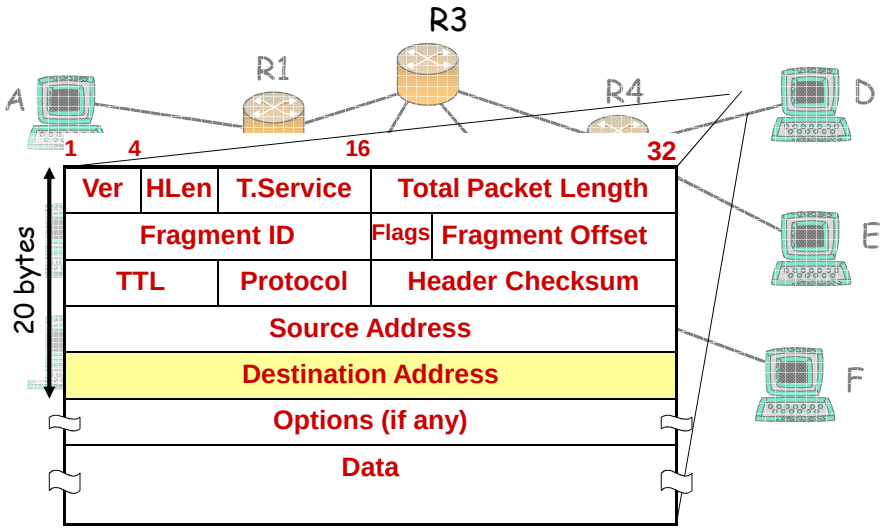
All packets, except the ones for the same network, first go to their gateway (router) and are transferred to the destination via routers.

Basic Operation of an IP Router

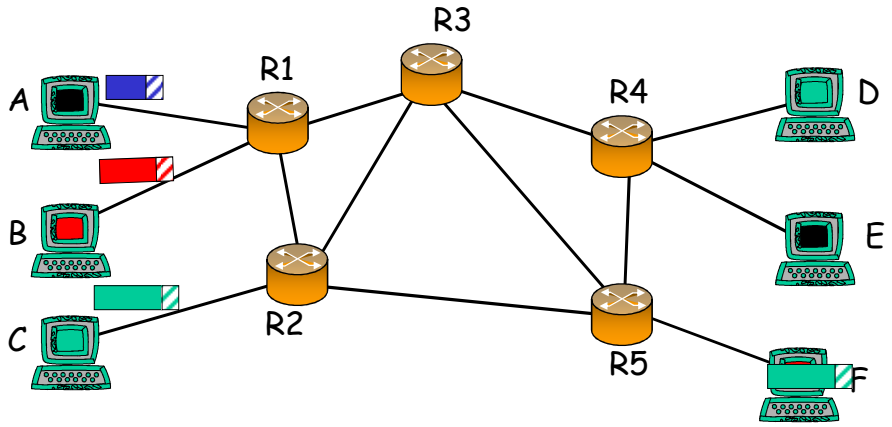


Destination	Next Hop
D	R3
E	R3
F	R5

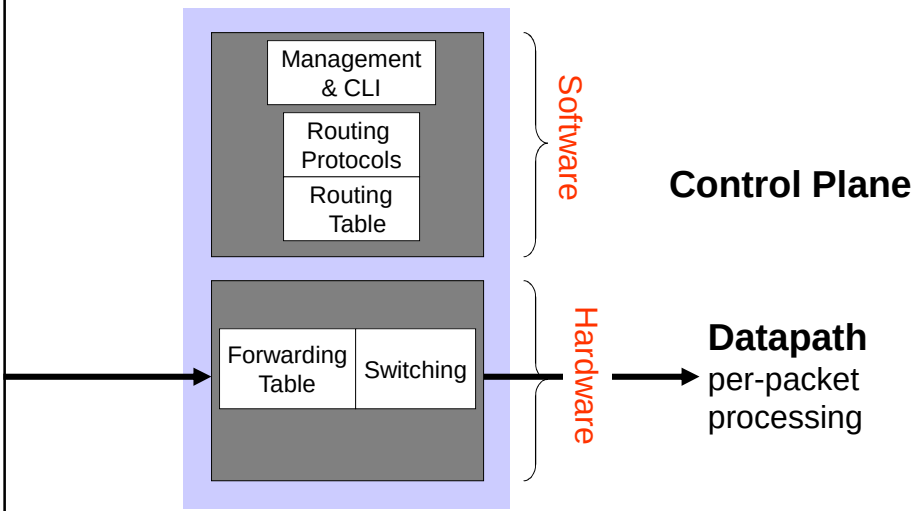
What does a router do?



What does a router do?



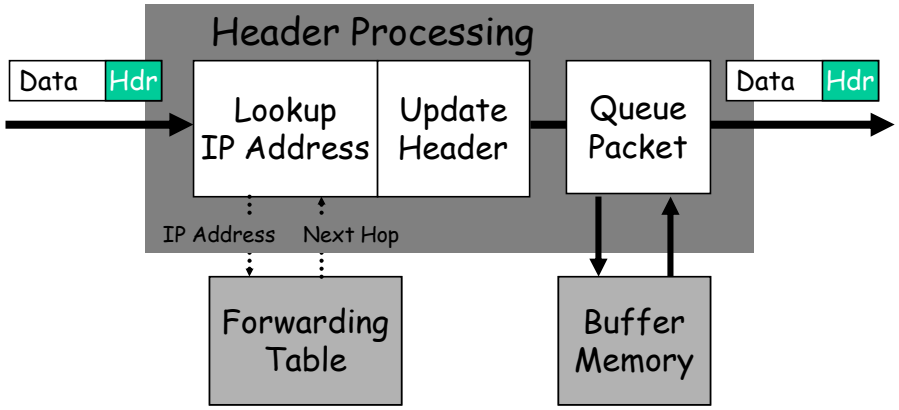
Basic Components of an IP Router



Per-packet processing in an IP Router

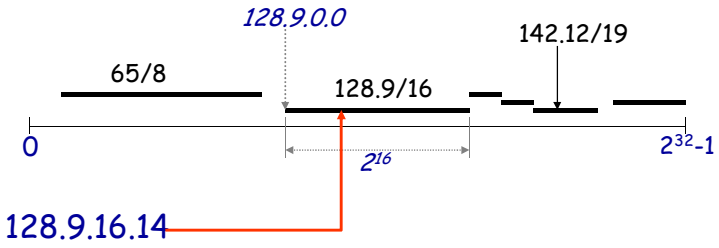
1. Accept packet arriving on an incoming link.
2. Lookup packet destination address in the forwarding table to identify outgoing port(s).
3. Manipulate IP header: e.g., decrement TTL, update header checksum.
5. Buffer packet in the output queue.
6. Transmit packet onto outgoing link.

Generic Datapath Architecture

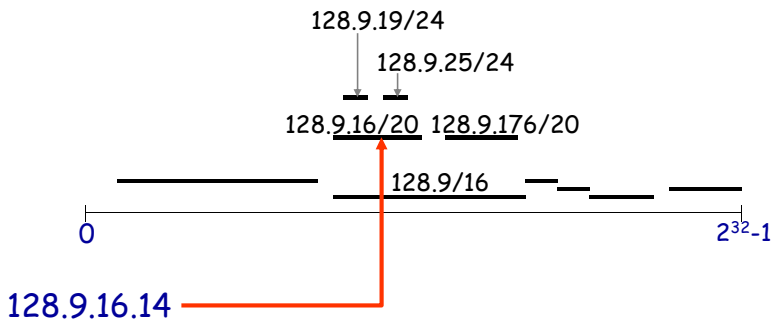


CIDR and Longest Prefix Matches

- The IP address space is broken into line segments.
- Each line segment is described by a ***prefix***.
- A prefix is of the form x/y where x indicates the prefix of all addresses in the line segment, and y indicates the length of the segment.
- e.g. The prefix 128.9/16 represents the line segment containing addresses in the range: 128.9.0.0 … 128.9.255.255.



Classless Interdomain Routing (CIDR)

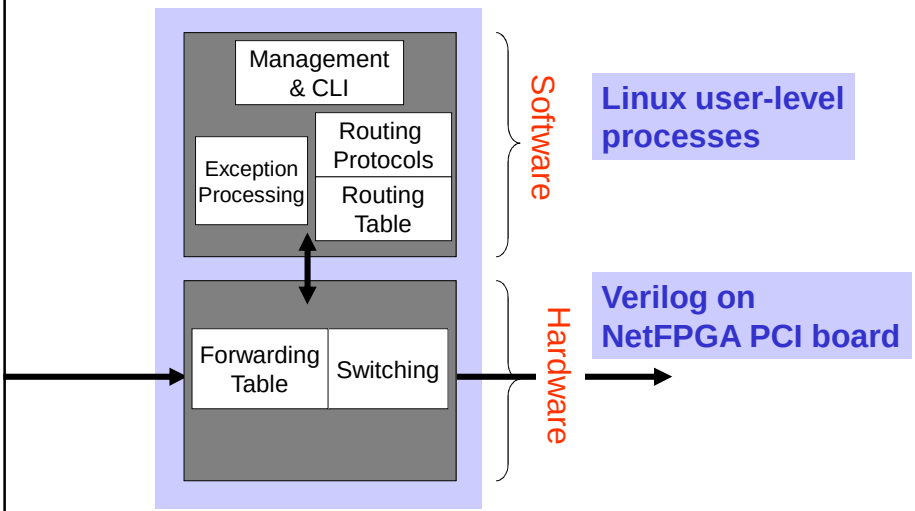


Most specific route = "longest matching prefix"

Techniques for LPM in hardware

- **Linear search**
  - Slow
- **Direct lookup**
  - Currently requires too much memory
  - Updating a prefix leads to many changes
- **Tries**
  - Deterministic lookup time
  - Easily pipelined but require multiple memories/references
- **TCAM (Ternary CAM)**
  - Simple and widely used but have lower density than RAM and need more power
  - Gradually being replaced by algorithmic methods

An IP Router on NetFPGA



Demo 1

Reference Router running on the NetFPGA

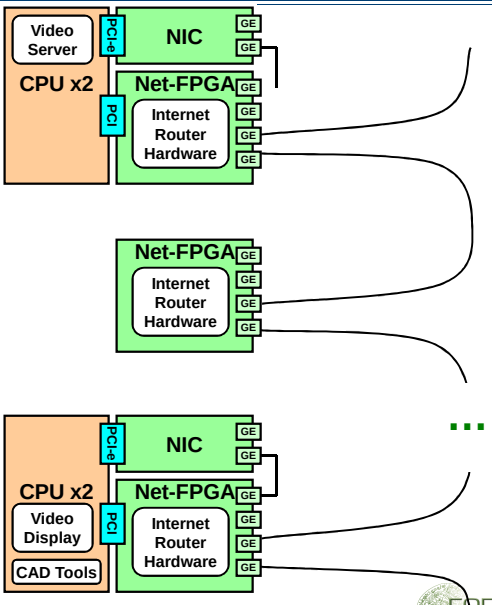
Demo 1

Hardware Setup for Demo #1

Server delivers streaming HD video through a chain of NetFPGA Routers

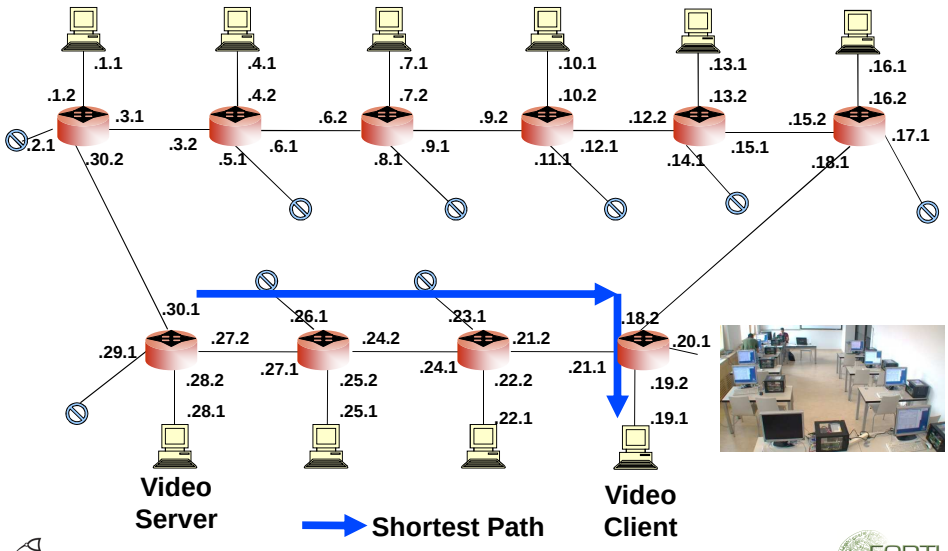






Demo 1

Topology



Demo 1

Working IP Router

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

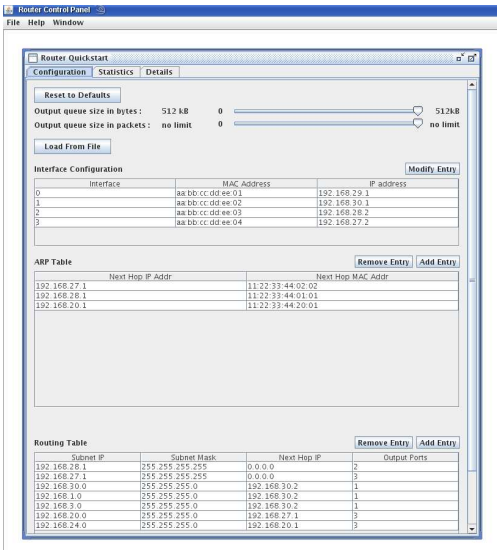
Demo 1

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

Key:

eth1 of Host PC
192.168.X.Y

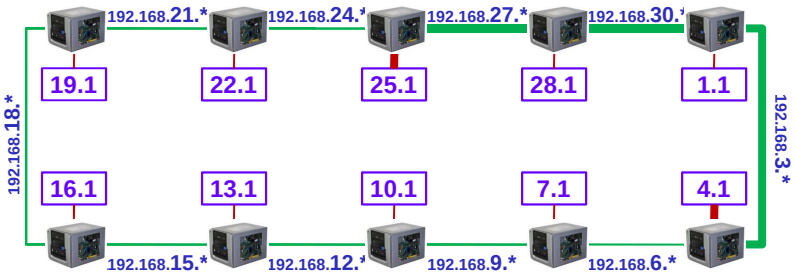
NetFPGA Router #

Example:

To stream mplayer video from server 4.1, type:

```
./mp 192.168.4.1
```

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

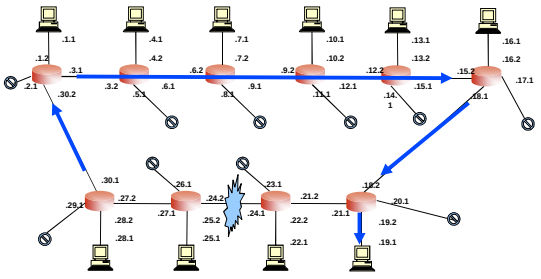


Demo 1

Step 3 - 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

Section III: Demo Advanced Use

Advanced Uses of NetFPGA

- **Introduction on TCP and Buffer Sizes**
- **Demonstrate**
  - NetFPGA used for real time measurement
  - See TCP Saw tooth in real time

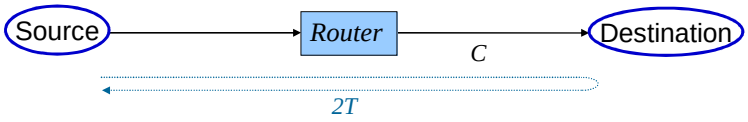
Buffer Requirements in a Router

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

Exploring Buffer Sizes

- Need to reduce buffer size and measure occupancy
- 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.

Why 2TxC for a single TCP Flow

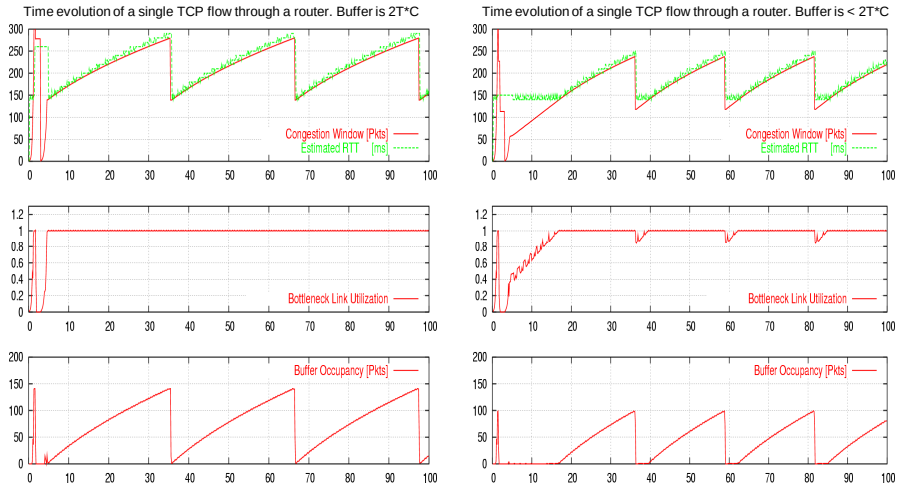
Only W packets may be outstanding



Rule for adjusting W

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

Time Evolution of a Single TCP Flow

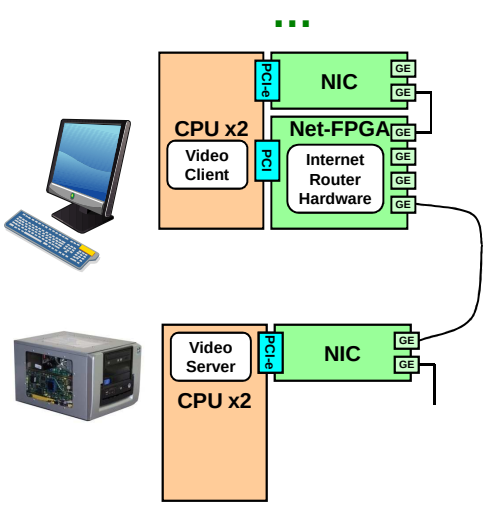


Demo 2

Buffer Sizing Experiments using the NetFPGA Router

Demo 2

Hardware Setup for Demo #2

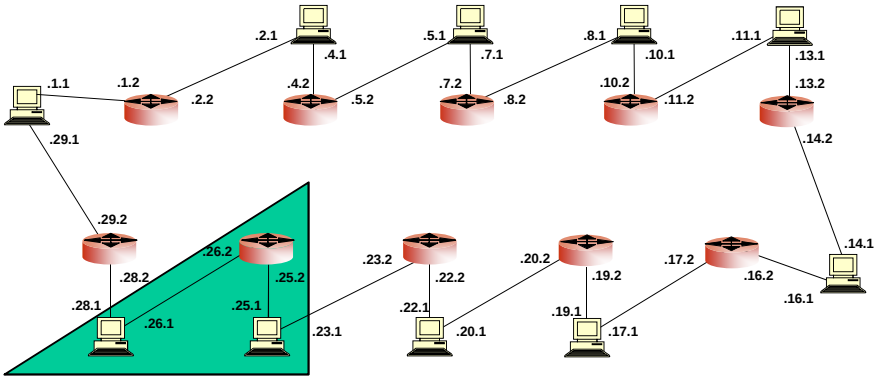


Server delivers streaming HD video to adjacent client

Demo 2

Topology

- 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

Demo 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

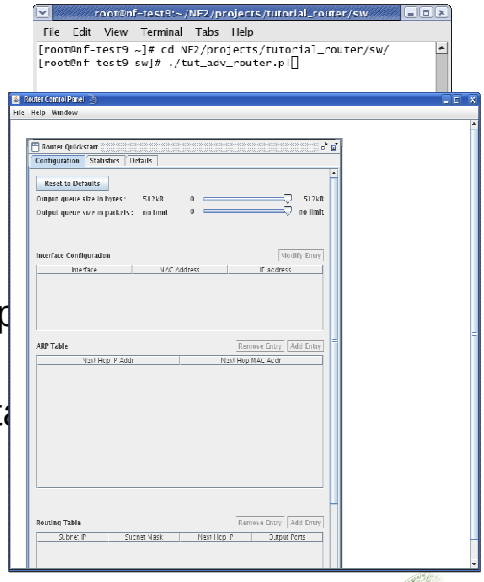
Demo 2

Step 1 - Run Pre-made Enhanced Router

Start terminal and cd to "netfpga/projects/ tutorial_router/sw/"

Type "./tut_adv_router_gui.p

A familiar GUI should st

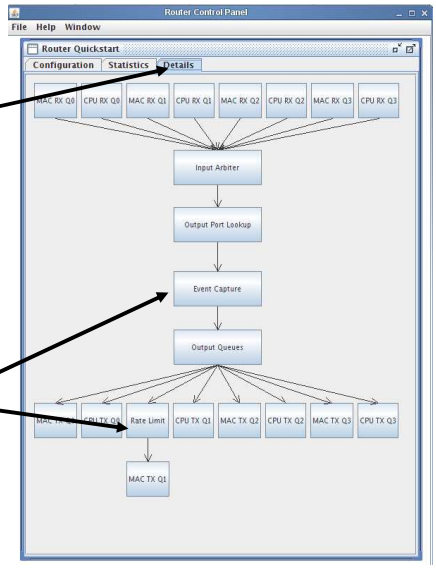


Demo 2

Step 2 - Explore Enhanced Router

Click on the Details tab

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

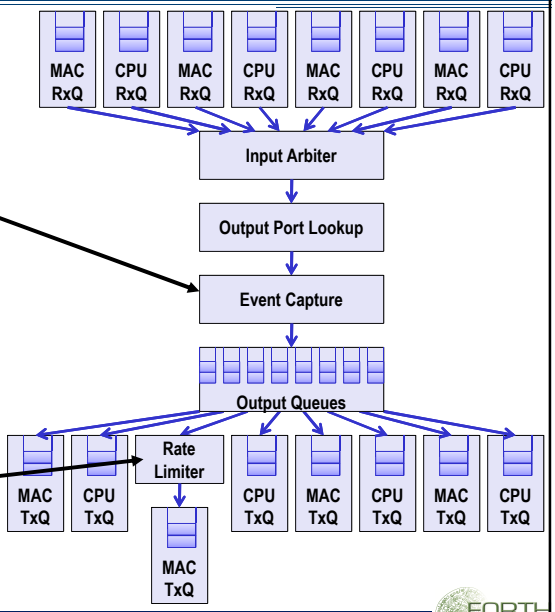


Demo 2

Enhanced Router Pipeline

Two modules added

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

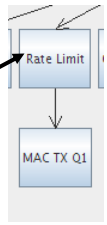


Demo 2

Step 3 - Decrease the Link Rate

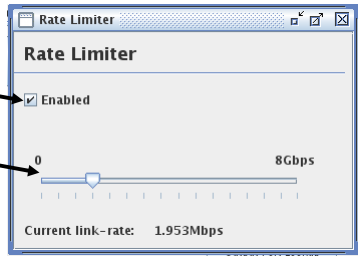
To create bottleneck and show the TCP “sawtooth,” link-rate is decreased.

In the Details tab, click the “Rate Limit” module



Check Enabled

Set link rate to 1.953Mbps



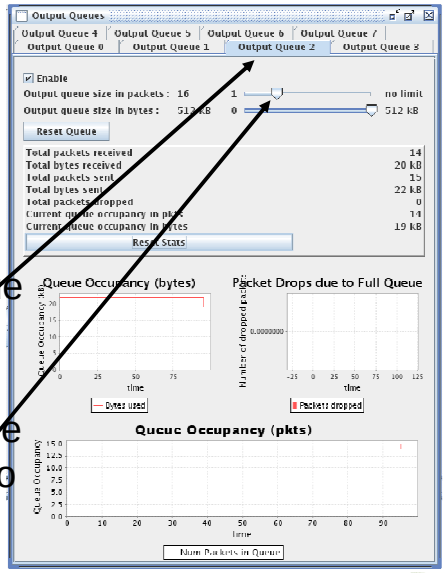
Demo 2

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

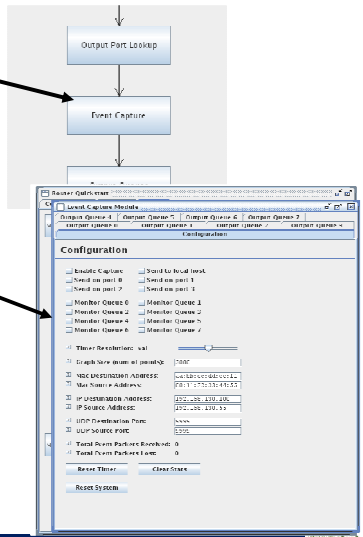


Demo 2

Step 5 - Start Event Capture

Click on the Event Capture module under the Details tab

This should start the configuration page



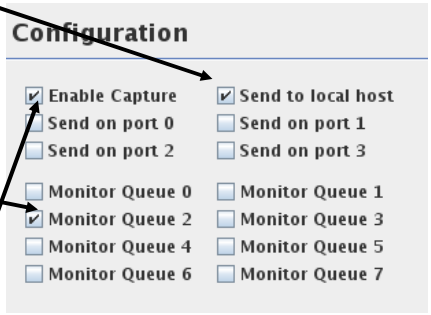
Demo 2

Step 6 - Configure Event Capture

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

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

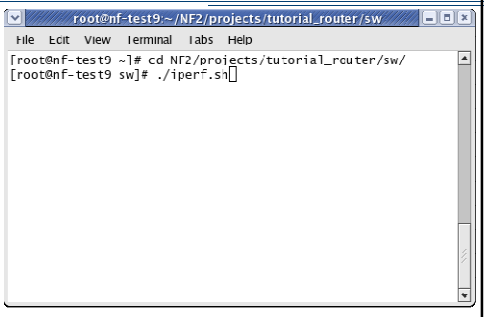
Check **Enable Capture** to start event capture



Demo 2

Step 7 - Start TCP Transfer

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



Start a terminal and cd to "netfpga/projects/tutorial_router/sw"

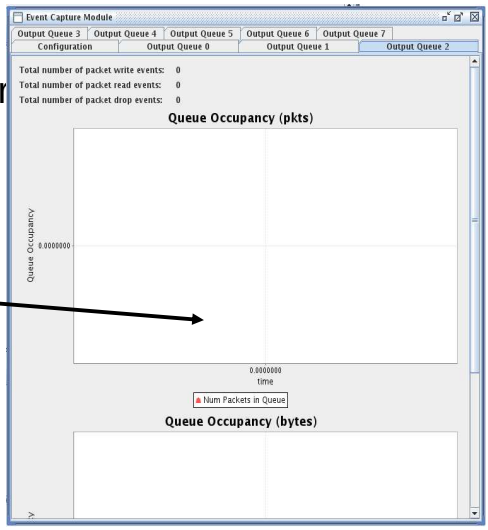
Type "./iperf.sh"

Demo 2

Step 8 - Look at Event Capture Results

Click on the Event Capture module under the Details tab.

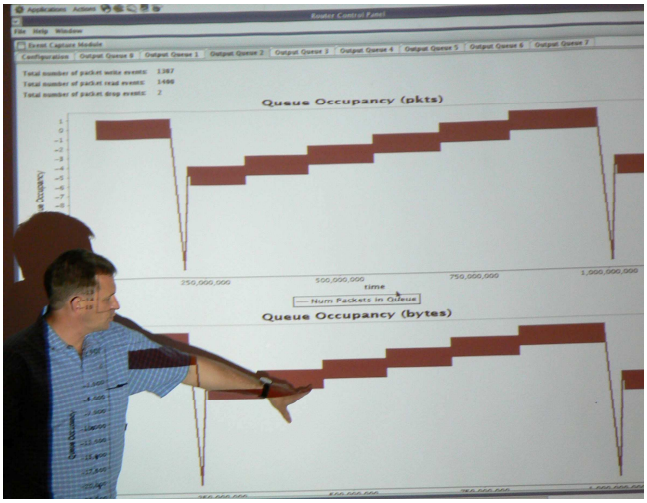
The sawtooth pattern should now be visible.



Demo 2

Queue Occupancy Charts

Observe the TCP/IP sawtooth



Leave the control windows open

Section IV: How does the NetFPGA Work

Integrated Circuit Technology

Full-custom Design
- Complementary Metal Oxide Semiconductor (CMOS)

Semi-custom ASIC Design
- Gate array
- Standard cell

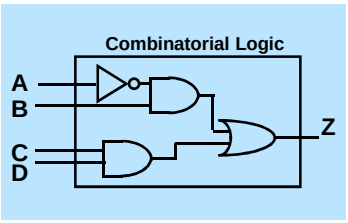
Programmable Logic Device
- Programmable Array Logic
- Field Programmable Gate Arrays

Processors

Look-Up Tables

Combinatorial logic is stored in Look-Up Tables (LUTs)

- Also called Function Generators (FGs)
- Capacity is limited only by number of inputs, not complexity
- Delay through the LUT is constant



A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Diagram From: Xilinx, Inc

Xilinx CLB Structure

Each slice has four outputs

- Two registered outputs, two non-registered outputs
- Two BUFTs associated with each CLB, accessible by all 16 CLB outputs

Carry logic run vertically

- Signals run upward
- Two independent carry chains per CLB

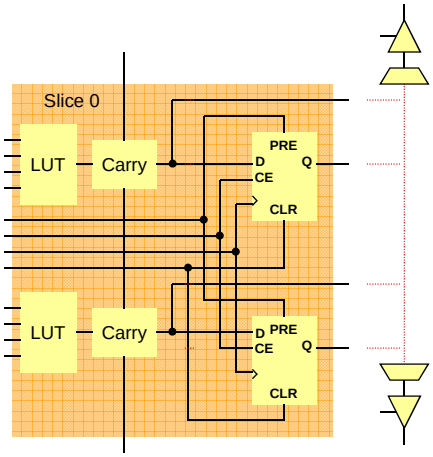
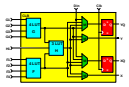


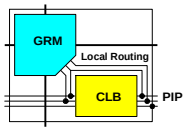
Diagram From: Xilinx, Inc.

Field Programmable Gate Arrays

CLB
- Primitive element of FPGA

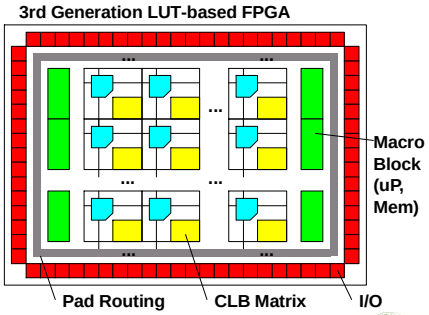


Routing Module
- Global routing
- Local interconnect



Macro Blocks
- Block Memories
- Microprocessor

I/O Block



NetFPGA Package

- **Utilities**
  - Simulation
  - Synthesis
  - Registers
- **Verilog Libraries (shared modules)**
- **Projects (reference and contributed)**

Simulation and Synthesis

- **Simulation (nf_run_test.pl)**
  - Allows simulation from command line or GUI
  - Uses backend libraries (Perl and Python) to create packets for simulation
- **Synthesis (make)**
  - In the projects synth directory
  - Automatically includes Xilinx Coregen components from shared libraries
  - Includes all Xilinx Coregen components form a projects synth directory (.xco)

Shared Verilog Libraries (modules)

- **Located at netfpga/lib/verilog**
- **Specify shared libraries in project.xml**
  - Any project can use any module
- **Local modules in a project's src dir over rides a shared library**
  - If arp_reply is found both in shared library and project's src directory only the project's src directory version is used

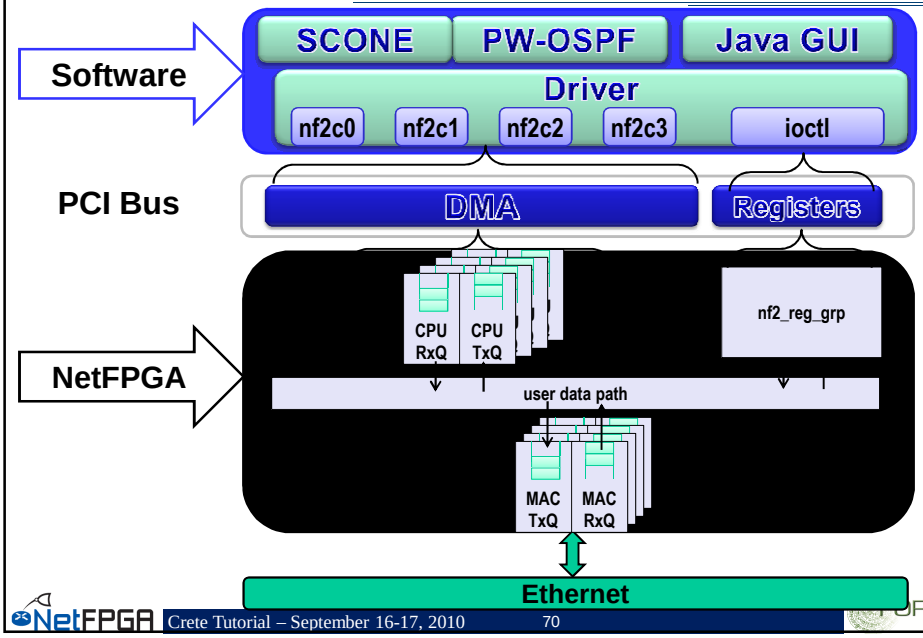
Register System

- **Project XML (project.xml)**
  - Found in project/include directory
  - Specifies shared libraries and location of registers in pipeline
- **Each module with registers has an XML file**
  - Specifies the register names and widths
- **Register files automatically created using nf_register_gen.pl**
  - Perl header files
  - C header files
  - Verilog file defining registers

Reference Projects

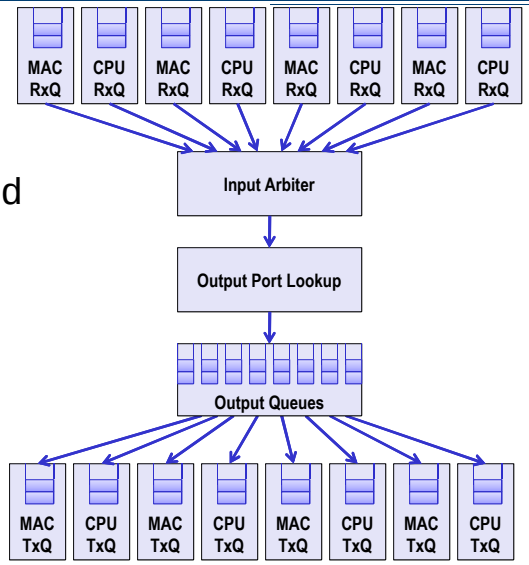
- **Easily extend and add modules**
- **Currently**
  - Reference NIC
  - Reference Router
  - Reference Switch
  - Router KIT
  - Router Buffer Sizing

Full System Components



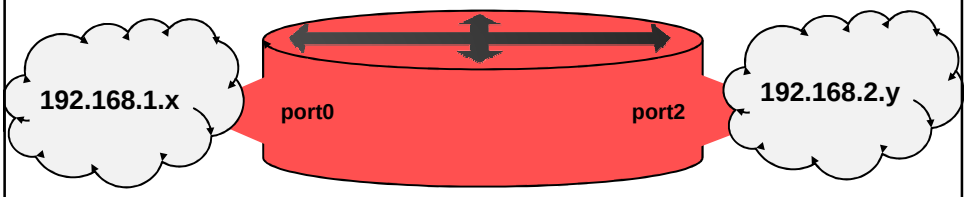
Reference Router Pipeline

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



Section V: Life of a Packet Through Hardware

Life of a Packet through the Hardware



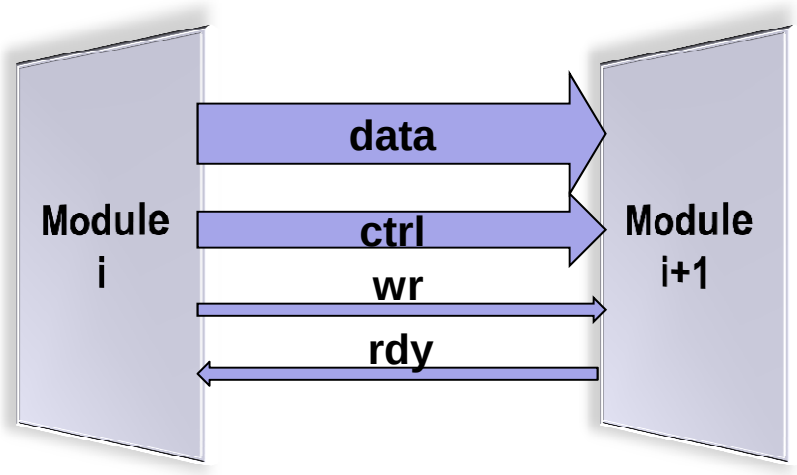
Inter-Module Communication

Using "Module Headers":

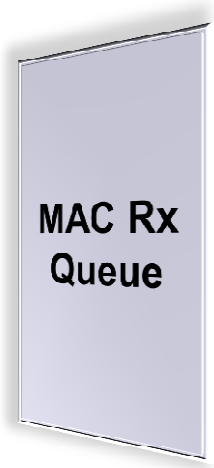
Ctrl Word (8 bits)	Data Word (64 bits)
x	Module Hdr
…	…
y	Last Module Hdr
0	Eth Hdr
0	IP Hdr
0	…
0x10	Last word of packet

Contain information such as packet length, input port, output port, …

Inter-Module Communication



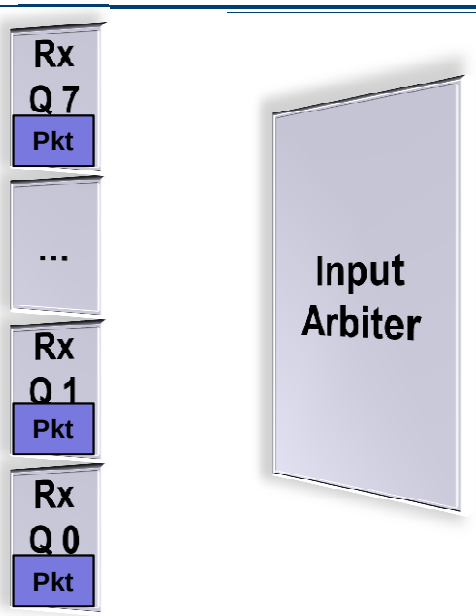
MAC Rx Queue



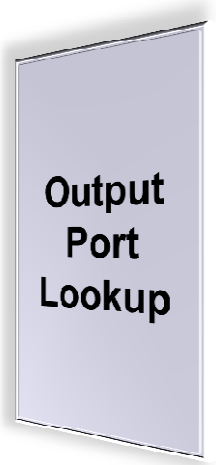
Rx Queue

0xff	Pkt length, input port = 0
0	Eth Hdr: Dst MAC = port 0, Ethertype = IP
0	IP Hdr: IP Dst: 192.168.2.3, TTL: 64, Csum:0x3ab4
0	Data

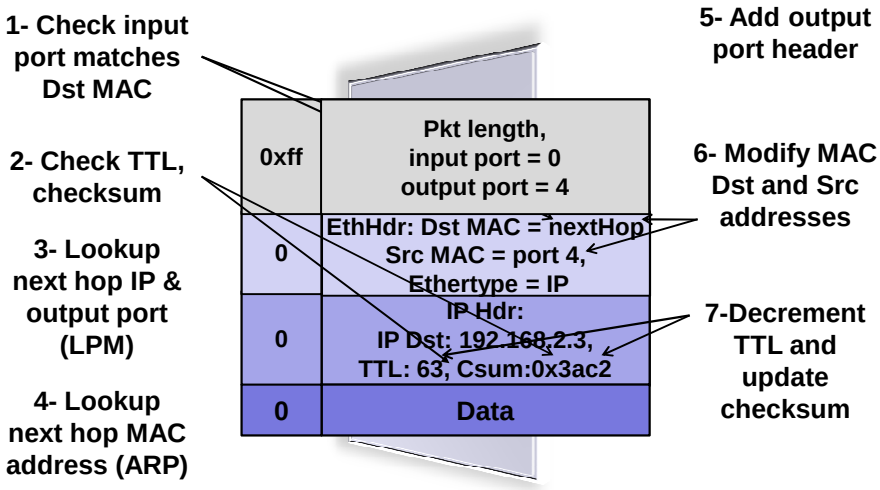
Input Arbiter



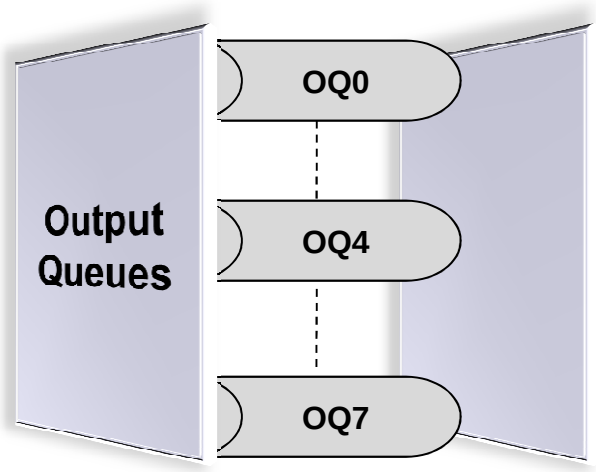
Output Port Lookup



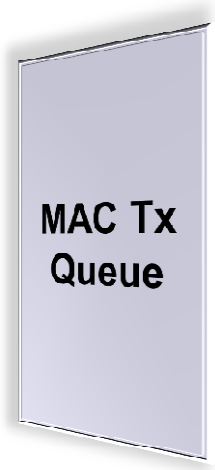
Output Port Lookup



Output Queues



MAC Tx Queue



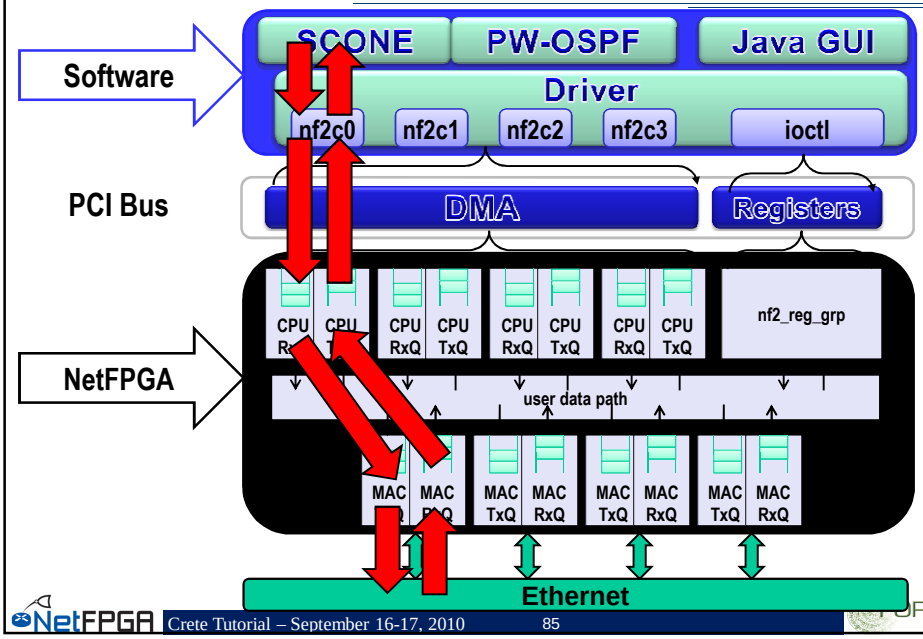
MAC Tx Queue

0xff	Pkt length, input port = 0 output port = 4
0	EthHdr: Dst MAC = nextHop Src MAC = port 4, Ethertype = IP
0	IP Hdr: IP Dst: 192.168.2.3, TTL: 63, Csum:0x3ac2
0	Data

Exception Packet

- Example: TTL = 0 or TTL = 1
- Packet has to be sent to the CPU which will generate an ICMP packet as a response
- Difference starts at the Output Port lookup stage

Exception Packet Path

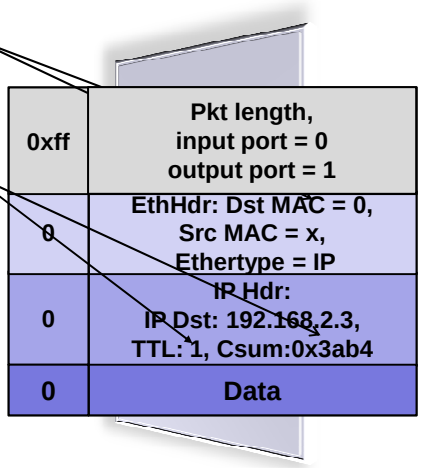


Output Port Lookup

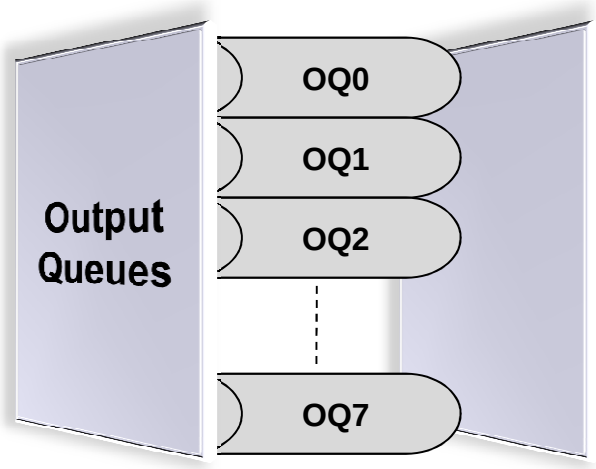
1- Check input port matches Dst MAC

2- Check TTL, checksum – EXCEPTION!

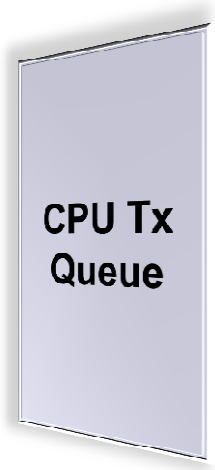
3- Add output port module



Output Queues



CPU Tx Queue



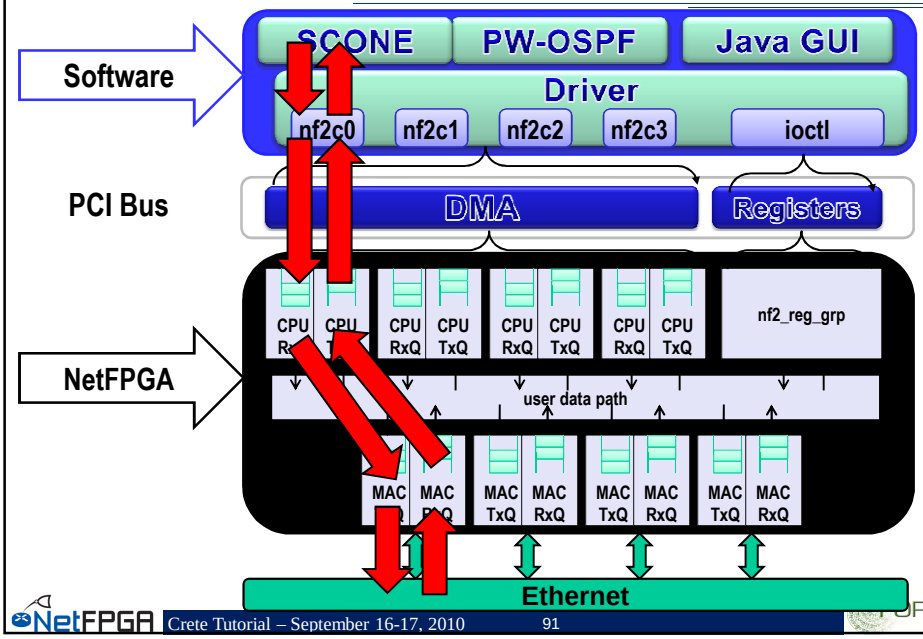
CPU Tx Queue

0xff	Pkt length, input port = 0 output port = 1
0	EthHdr: Dst MAC = 0, Src MAC = x, Ethertype = IP
0	IP Hdr: IP Dst: 192.168.2.3, TTL: 1, Csum:0x3ab4
0	Data

ICMP Packet

- For the ICMP packet, the packet arrives at the CPU Rx Queue from the PCI Bus
- It follows the same path as a packet from the MAC until it reaches the Output Port Lookup
- The OPL module sees the packet is from the CPU Rx Queue 1 and sets the output port directly to 0
- The packet then continues on the same path as the non-exception packet to the Output Queues and then MAC Tx queue 0

ICMP Packet Path



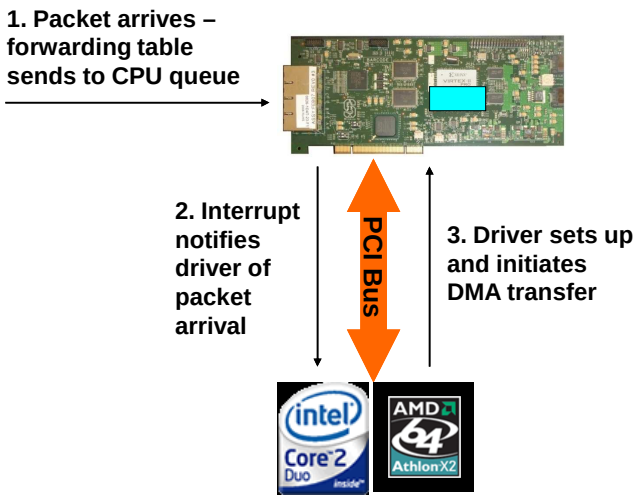
NetFPGA-Host Interaction

- **Linux driver interfaces with hardware**
  - Packet interface via standard Linux network stack
  - Register reads/writes via ioctl system call with wrapper functions:
    - readReg(nf2device *dev, int address, unsigned *rd_data);
    - writeReg(nf2device *dev, int address, unsigned *wr_data);

    eg:
    readReg(&nf2, OQ_NUM_PKTS_STORED_0, &val);

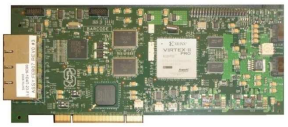
NetFPGA-Host Interaction

NetFPGA to host packet transfer



NetFPGA-Host Interaction

NetFPGA to host packet transfer (cont.)



4. NetFPGA transfers packet via DMA

PCI Bus

5. Interrupt signals completion of DMA



6. Driver passes packet to network stack

NetFPGA-Host Interaction

Host to NetFPGA packet transfers



2. Driver sets up and initiates DMA transfer

PCI Bus

3. Interrupt signals completion of DMA

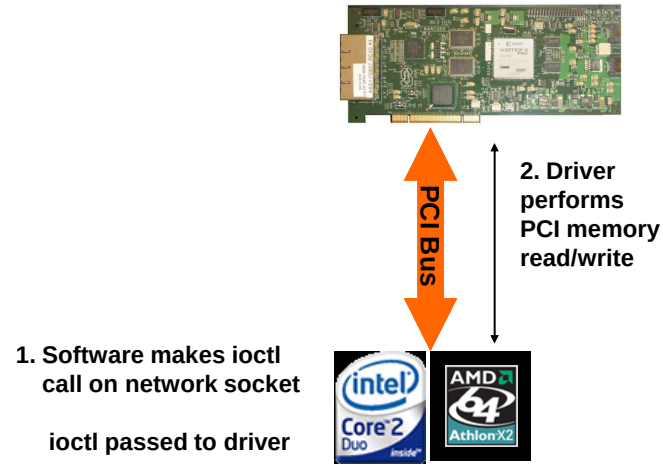
1. Software sends packet via network sockets

Packet delivered to driver



NetFPGA-Host Interaction

Register access



NetFPGA-Host Interaction

- **Packet transfers shown using DMA interface**
- **Alternative: use programmed IO to transfer packets via register reads/writes**
  - slower but eliminates the need to deal with network sockets

Section VI: Exercise

Exercise 1

Drop 1 in N Packets

Objectives

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

Execution

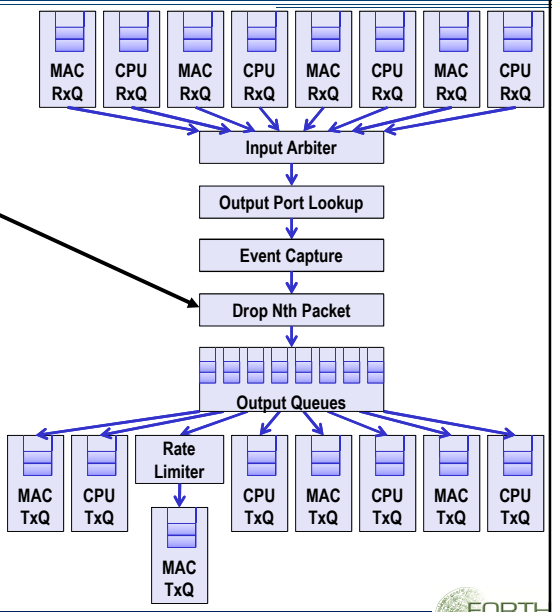
- Open drop_nth_packet.v
- Insert counter code
- Synthesize
- After synthesis, test the new system.

Exercise 1

New Reference Router Pipeline

One module added

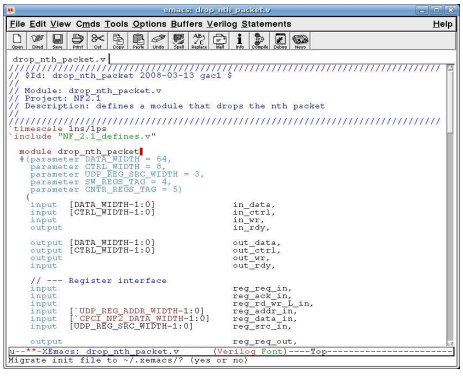
1. **Drop Nth Packet**
   to drop every Nth packet from the reference router pipeline



Exercise 1

Step 1 - Open the Source

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



Open terminal

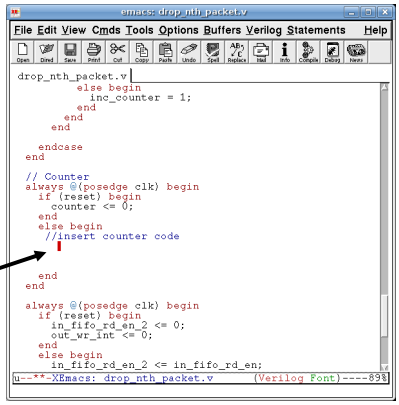
Type “xemacs netfpga/projects/tutorial_router/src/drop_nth_packet.v

Exercise 1

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 1

Step 3 - Build the Hardware

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

Run "**make clean**"

Start synthesis with "make"



Exercise 1

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 netfpga/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

– 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 1

Step 5 – Measurements

- **Determine iperf TCP throughput to neighbor's server for each of several values of N**
  - Similar to Demo 2, Step 8
    - `cd netfpga/projects/tutorial_router/sw`
    - `./iperf.sh`
  - Ping 192.168.x.2 (where x is your neighbor's server)
  - TCP throughput with:
    - Drop circuit disabled
      - TCP Throughput = _____ Mbps
    - Drop one in N = 1,000 packets
      - TCP Throughput = _____ Mbps
    - Drop one in N = 100 packets
      - TCP Throughput = _____ Mbps
    - Drop one in N = 10 packets
      - TCP Throughput = _____ Mbps
- **Explain why TCPs throughput is so low given that only a tiny fraction of packets are lost**

Section VII: Concluding Remarks

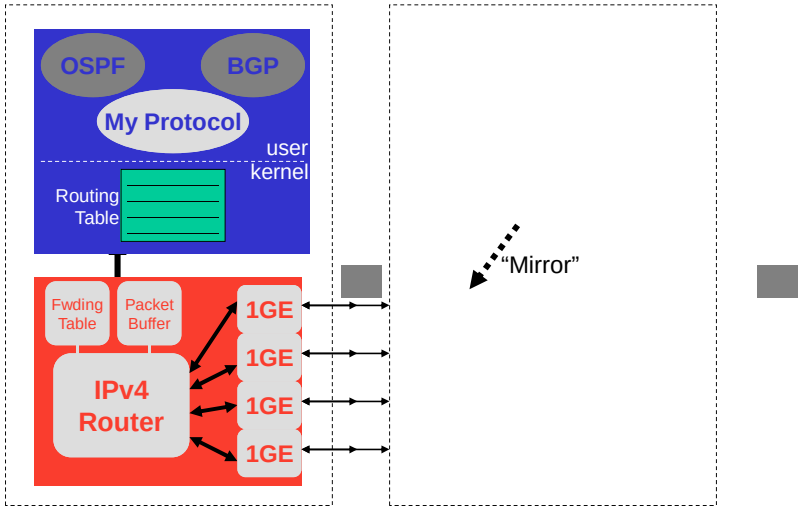
NetFPGAs are used:

- **To run laboratory courses on network routing**
  - Professors teach courses (Stanford, Cambridge, Rice, ...)
- **To teach students how to build real Internet routers**
  - Train students to build routers (Cisco, Juniper, Huawei, ..)
- **To research how new features in the network**
  - Build network services for data centers (Google, UCSD..)
- **To prototype systems with live traffic**
  - That measure buffers (while maintaining throughput, ..)
- **To help hardware vendors understand device requirements**
  - Use of hardware (Xilinx, Micron, Cypress, Broadcom, ..)

Usage #1

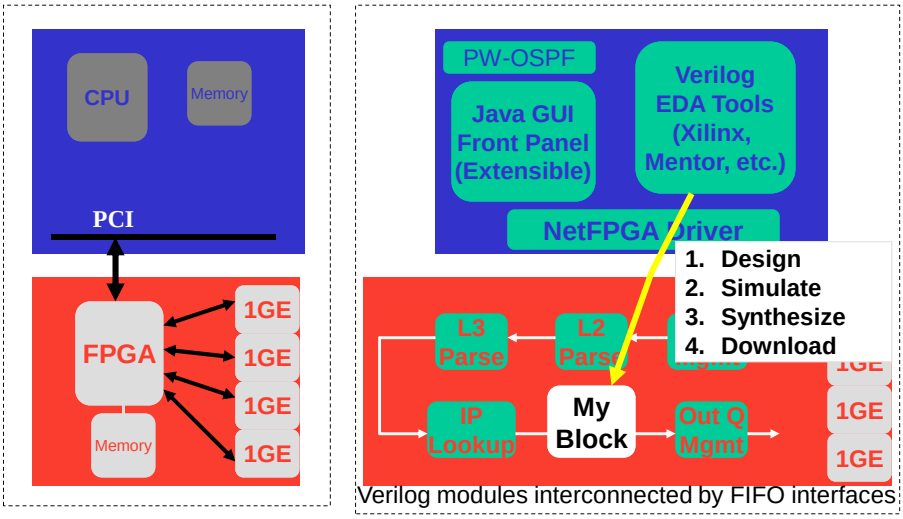
Running the Router Kit

User-space development, 4x1GE line-rate forwarding



Usage #2

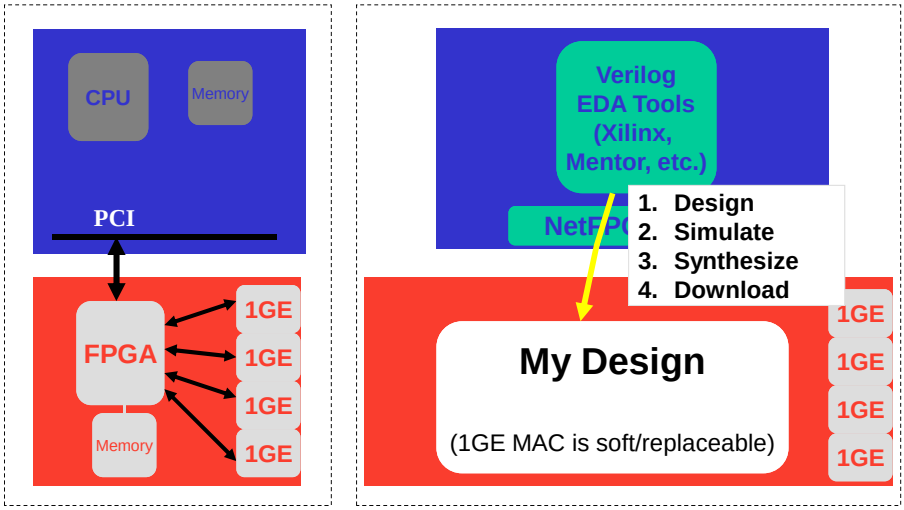
Enhancing Modular Reference Designs



Verilog modules interconnected by FIFO interfaces

Usage #3

Creating new systems



NetFPGA Platform

Major Components

- Interfaces
  - 4 Gigabit Ethernet Ports
  - PCI Host Interface
- Memories
  - 36Mbits Static RAM
  - 512Mbits DDR2 Dynamic RAM
- FPGA Resources
  - Block RAMs
  - Configurable Logic Block (CLBs)
  - Memory Mapped Registers

NetFPGA 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 Servers



2U Server (Dell 2950)



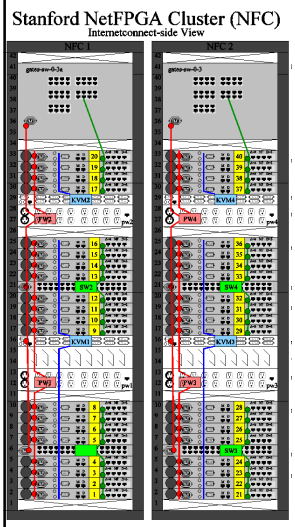
NetFPGA inserts in PCI or PCI-X slot



1U Server (Accent Technology Inc.)

Thanks: Brian Cashman for providing machine

Stanford NetFPGA Cluster





Statistics

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

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

Special thanks to our Partners:

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

Past NetFPGA Tutorial 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.*