NetFPGA Programmable Networking for High-Speed Network Prototypes, Research and Teaching

Presented by:

Andrew W. Moore

(University of Cambridge)

CHANGE/OFELIA

Berlin, Germany

November 10th, 2011

http://NetFPGA.org

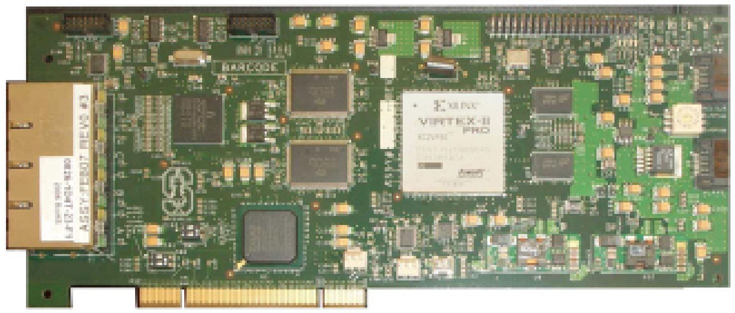
Tutorial Outline

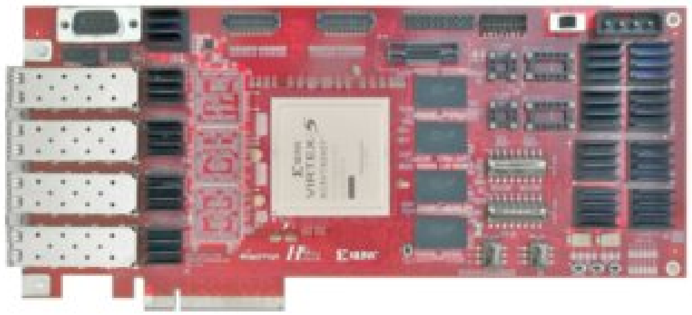
- Motivation
  - Introduction
  - The NetFPGA Platform
- Hardware Overview
  - NetFPGA 1G
  - NetFPGA 10G
- The Stanford Base Reference Router
  - Motivation: Basic IP review
  - Example 1: Reference Router running on the NetFPGA
  - Example 2: Understanding buffer size requirements using NetFPGA
- Community Contributions
  - Altera-DE4 NetFPGA Reference Router (UMassAmherst)
  - NetThreads (University of Toronto)
- Concluding Remarks

Section I: Motivation

NetFPGA = Networked FPGA

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





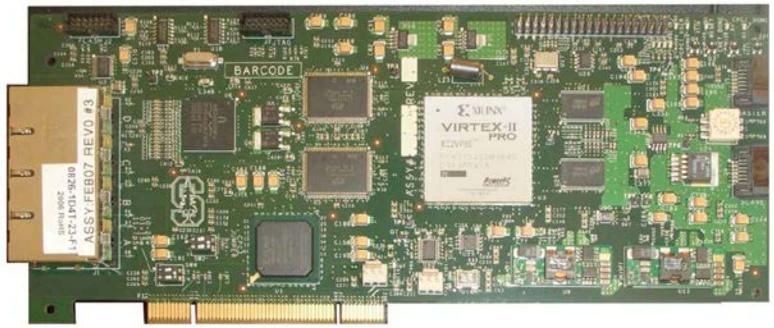
=

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

NetFPGA consists of…

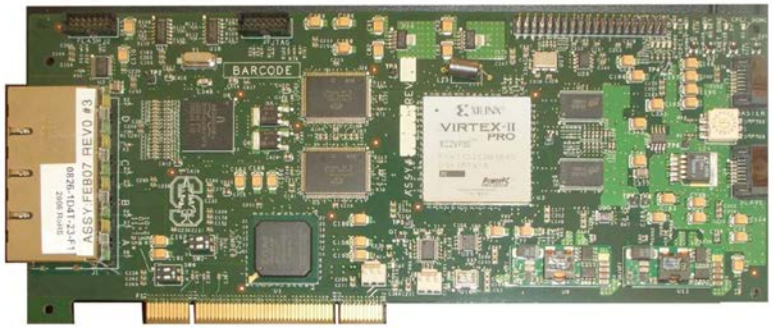
Four elements:

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


NetFPGA 1G Board

NetFPGA 10G Board

NetFPGA Board Comparison

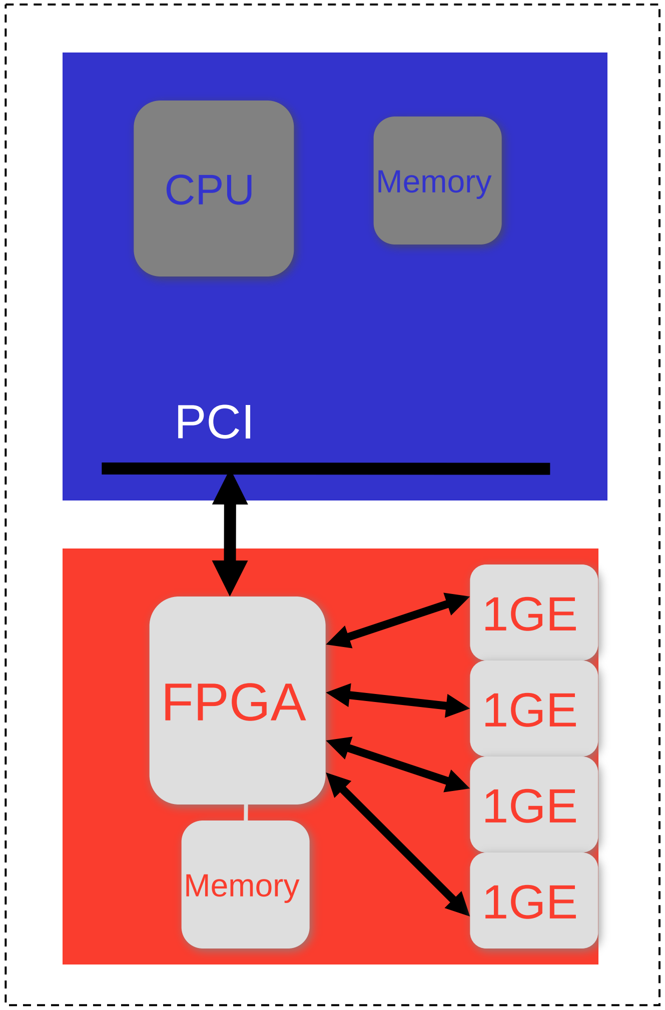


NetFPGA 1G	NetFPGA 10G
4 x 1Gbps Ethernet Ports	4 x 10Gbps SFP+
4.5 MB ZBT SRAM 64 MB DDR2 SDRAM	27 MB QDRII-SRAM 288 MB RLDRAM-II
PCI	PCI Express x8
Virtex II-Pro 50	Virtex 5 TX240T

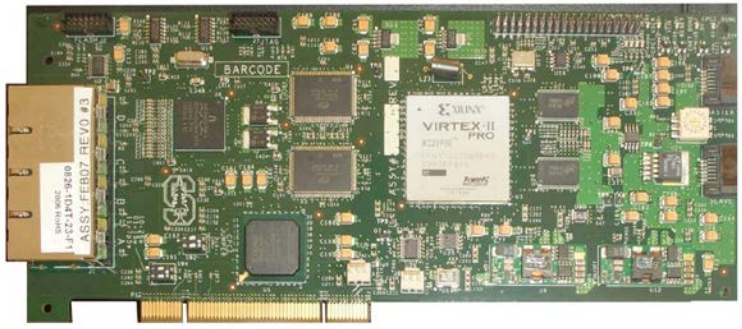
NetFPGA board

Networking Software running on a standard PC

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



PC with NetFPGA



NetFPGA Board

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)

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 proje

http://n

Community

Wiki

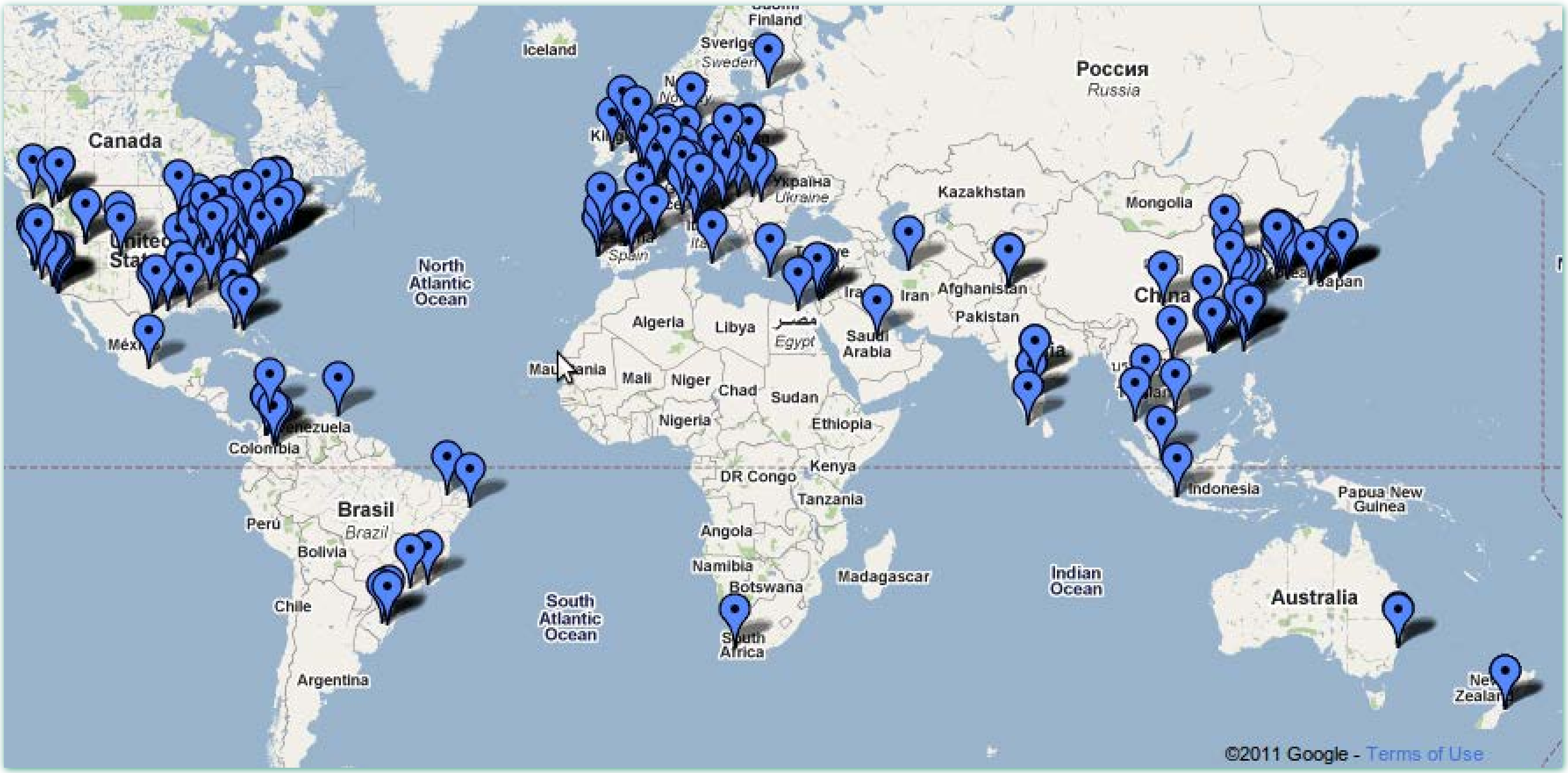
- Documentation
  - User's Guide
  - Developer's Guide
- Encourage users to contribute

Forums

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

International Community

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



NetFPGA's Defining Characteristics

- Line-Rate
  - Processes back-to-back packets
    - Without dropping packets
    - At full rate of Gigabit Ethernet Links
  - Operating on packet headers
    - For switching, routing, and firewall rules
  - And packet payloads
    - For content processing and intrusion prevention

- Open-source Hardware
  - Similar to open-source software
    - Full source code available
    - BSD-Style License
  - But harder, because
    - Hardware modules must meeting timing
    - Verilog & VHDL Components have more complex interfaces
    - Hardware designers need high confidence in specification of modules

Test-Driven Design

- Regression tests
  - Have repeatable results
  - Define the supported features
  - Provide clear expectation on functionality

- Example: Internet Router
  - Drops packets with bad IP checksum
  - Performs Longest Prefix Matching on destination address
  - Forwards IPv4 packets of length 64-1500 bytes
  - Generates ICMP message for packets with TTL <= 1
  - Defines how packets with IP options or non IPv4

… and dozens more …

Every feature is defined by a regression test

Who, How, Why

Who uses the NetFPGA?

- Teachers
- Students
- Researchers

How do they use the NetFPGA?

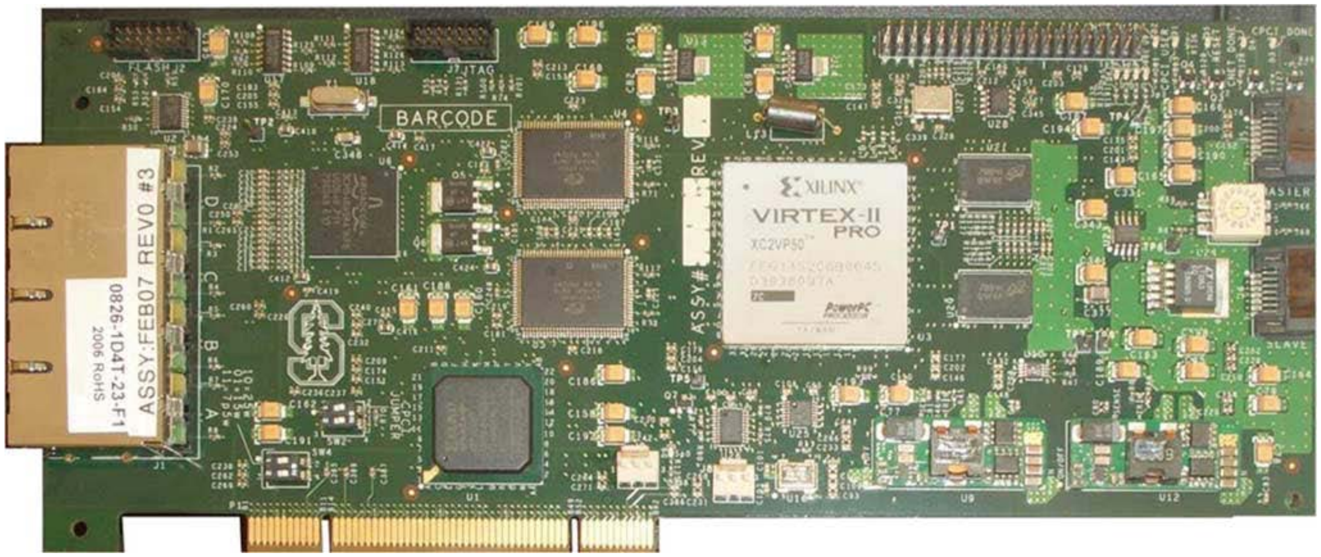
- To run the Router Kit
- To build modular reference designs
  - IPv4 router
  - 4-port NIC
  - Ethernet switch, …

Why do they use the NetFPGA?

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

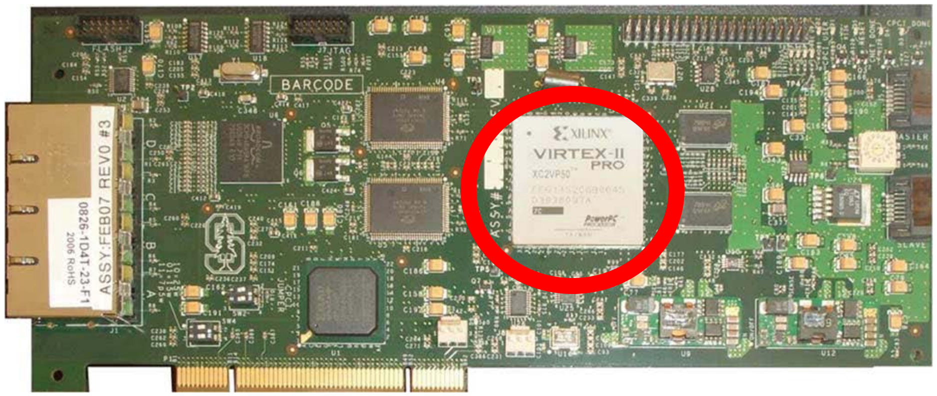
Section II: Hardware Overview

NetFPGA-1G



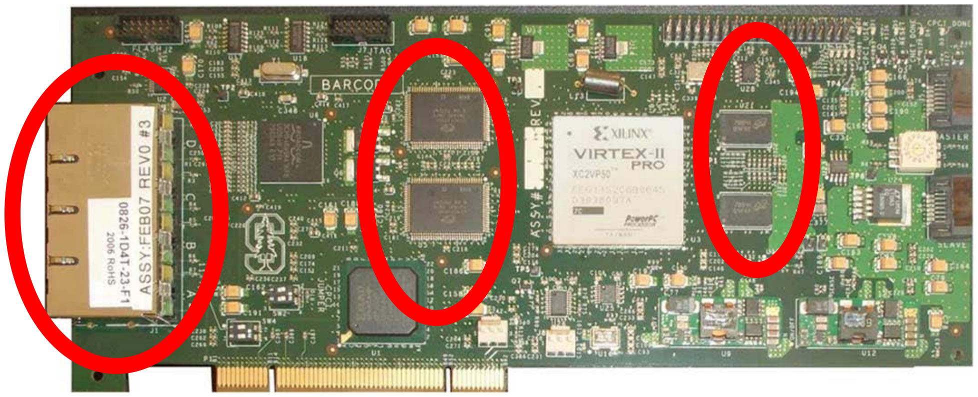
Xilinx Virtex II Pro 50

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



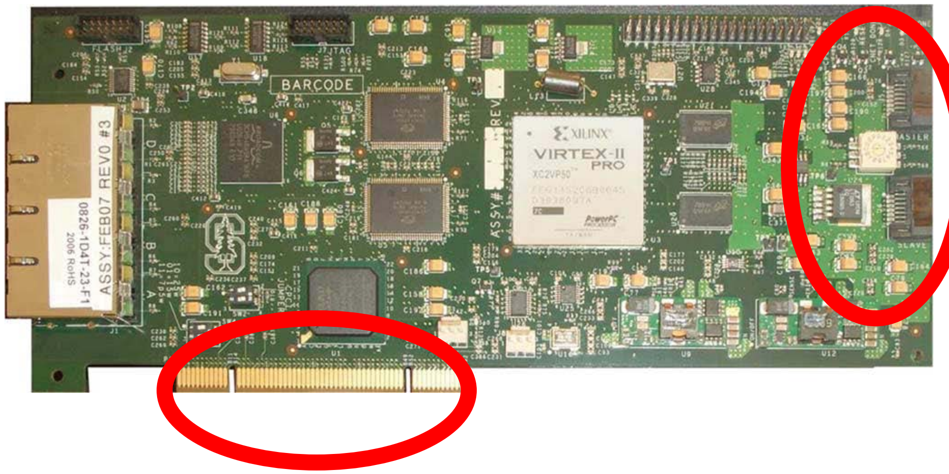
Network and Memory

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



Other IO

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



NetFPGA-10G

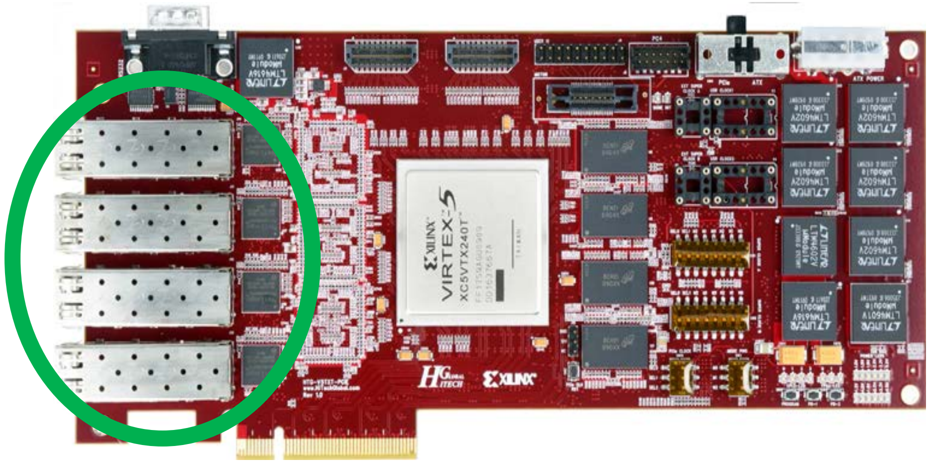
- A major upgrade
- State-of-the-art technology

Comparison

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

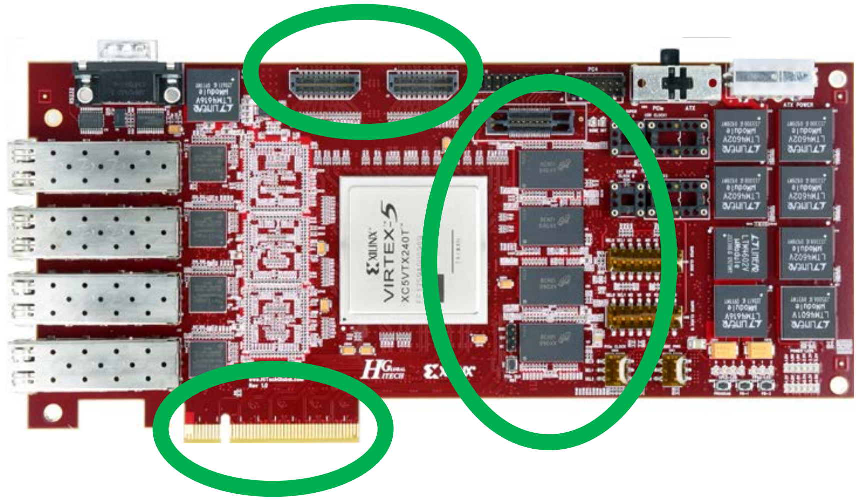
10 Gigabit Ethernet

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



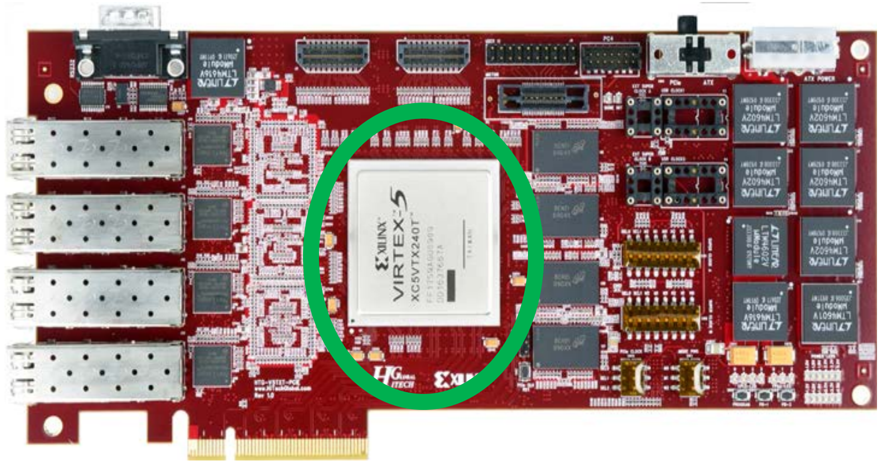
Others

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

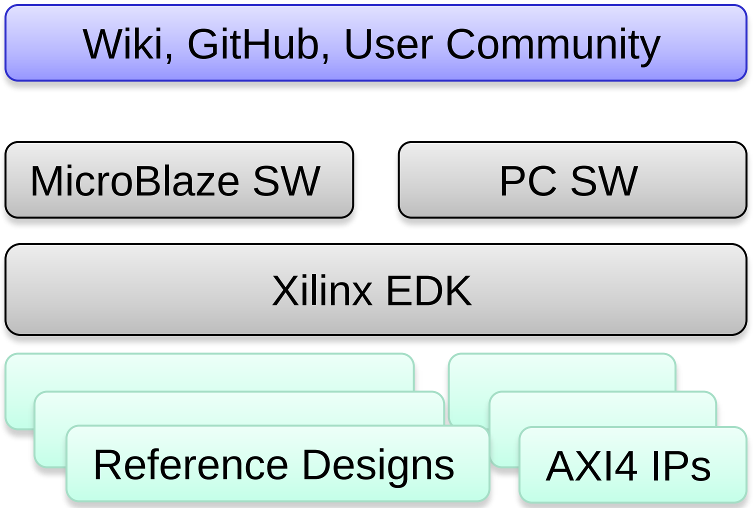


Xilinx Virtex 5 TX240T

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



Beyond Hardware



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

NetFPGA-1G Cube Systems

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

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

Rackmount NetFPGA-1G Servers

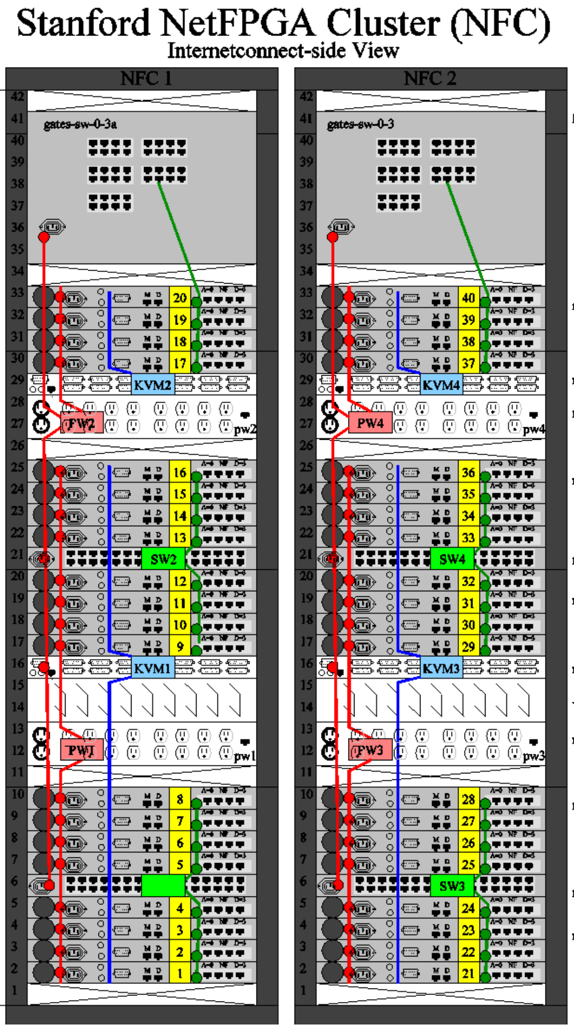
2U Server
(Dell 2950)

NetFPGA inserts in
PCI or PCI-X slot

1U Server
(Accent Technology Inc.)

Thanks: Brian Cashman for providing machine

Stanford NetFPGA-1G Cluster



Statistics

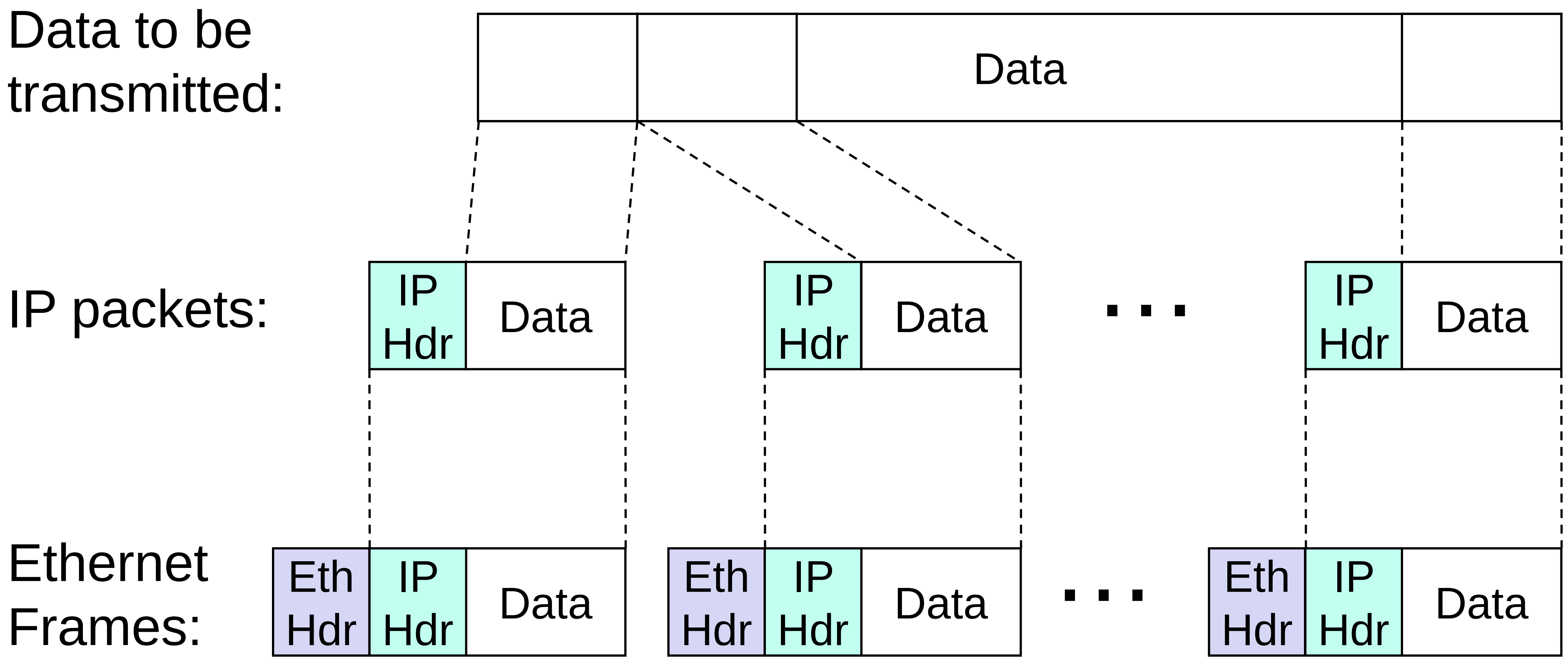
- Rack of 40
- 1U PCs with NetFPGAs

- Managed
- Power
- Console
- LANs

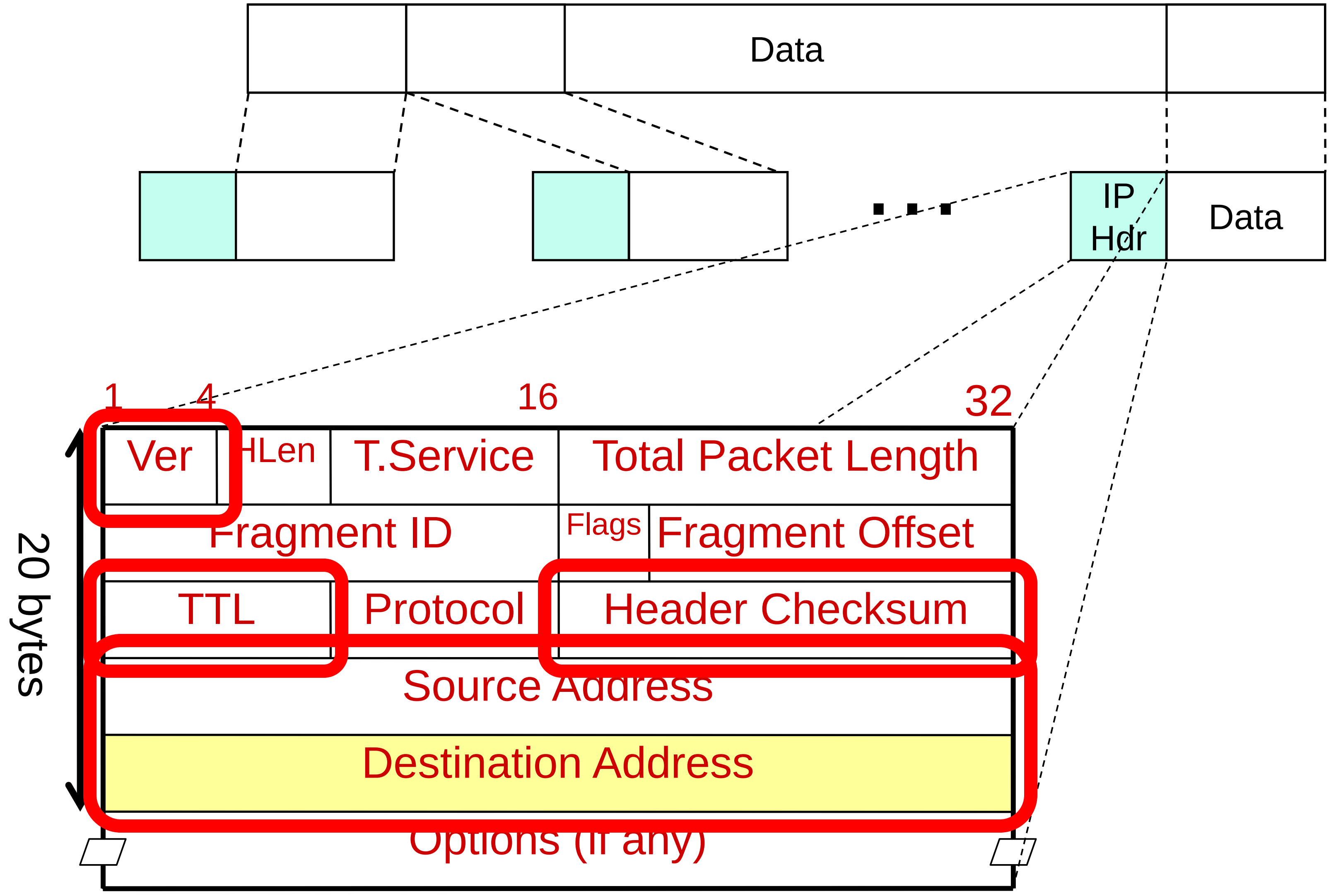
- Provides 4*40=160 Gbps of full line-rate processing bandwidth

Section III: Network review

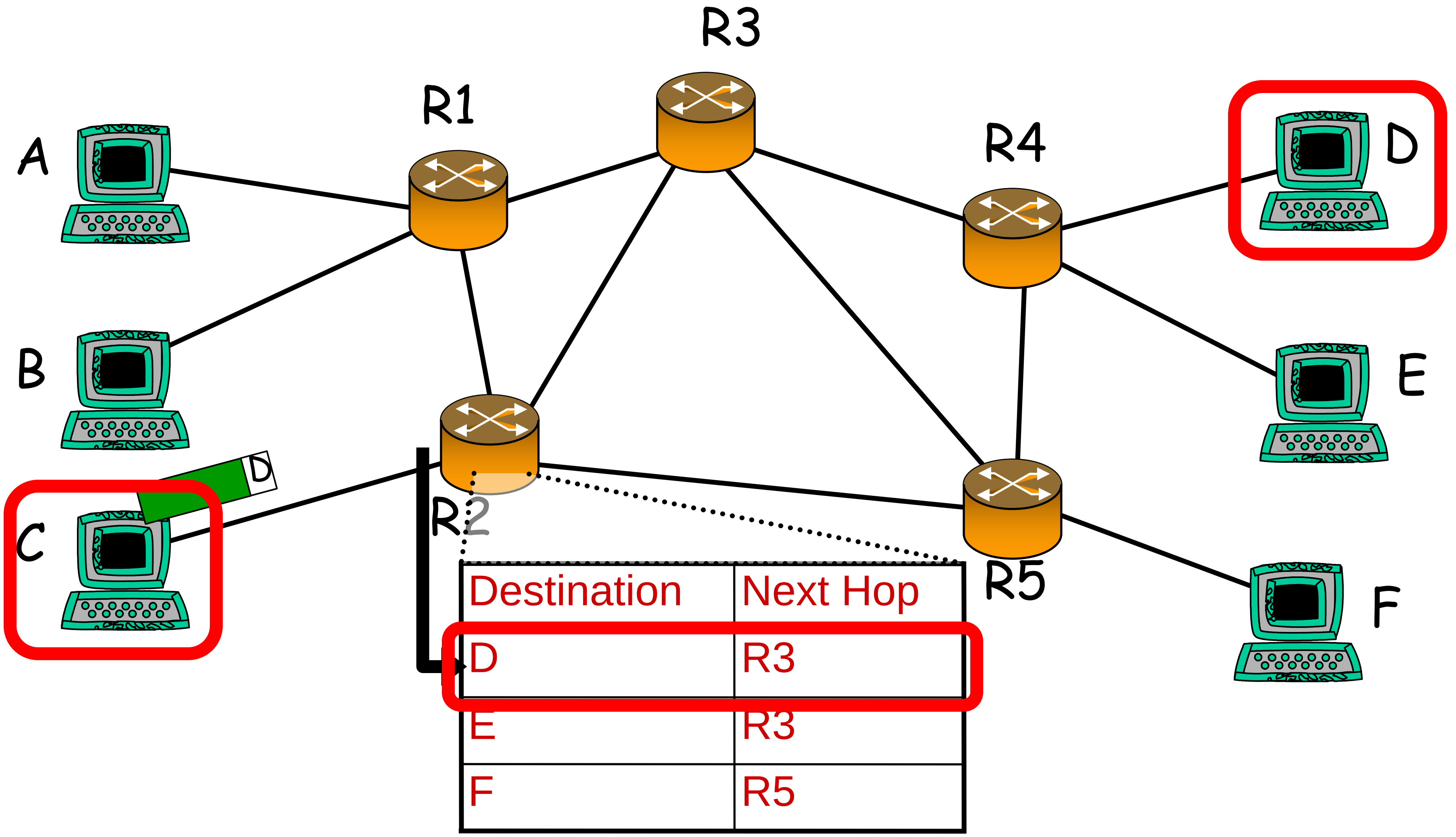
Internet Protocol (IP)



Internet Protocol (IP)

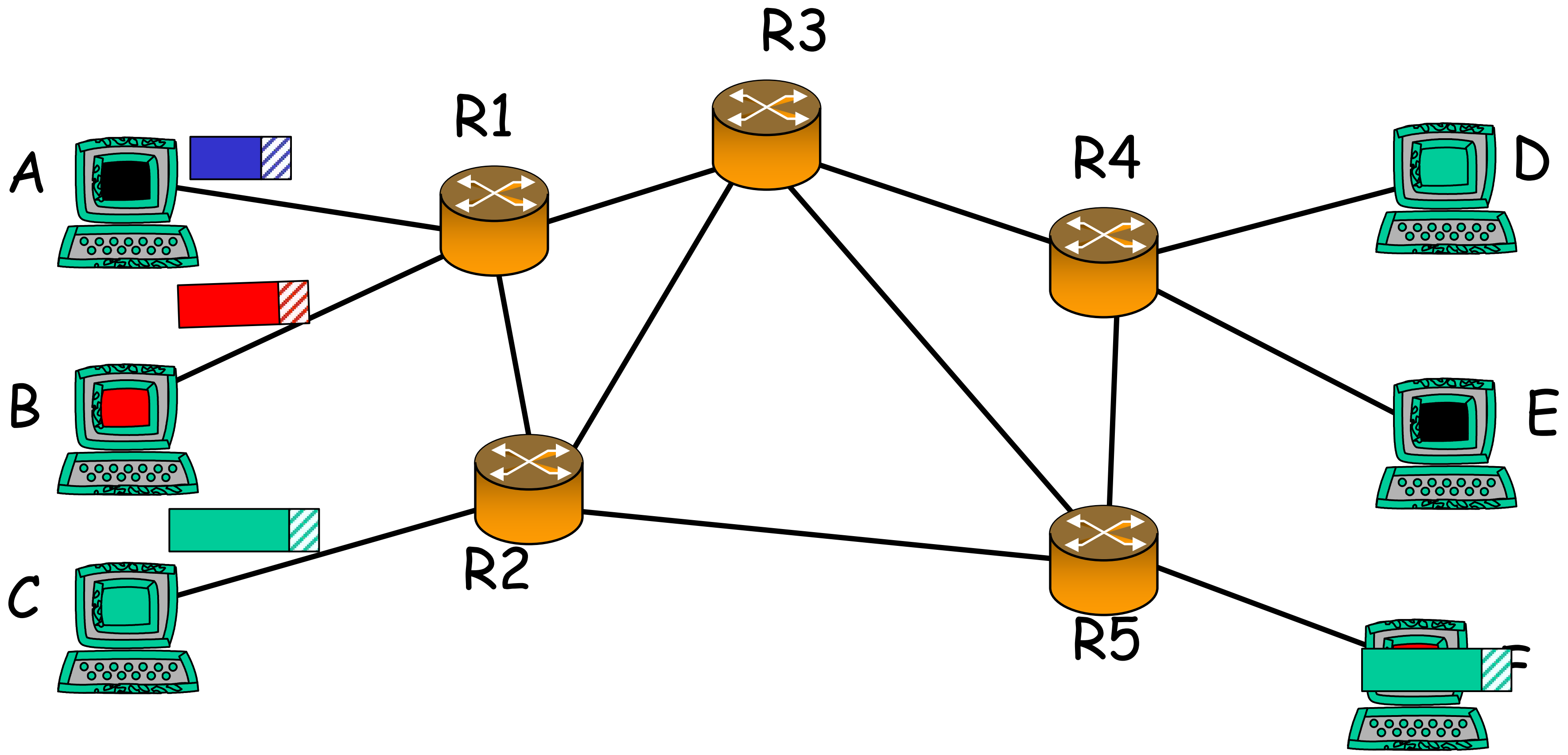


Basic operation of an IP router



Destination	Next Hop
D	R3
E	R3
F	R5

Basic operation of an IP router



Forwarding tables

$\boxed{\text{IP address}}\}$ 32 bits wide → ~ 4 billion unique address

Naïve approach:
One entry per address

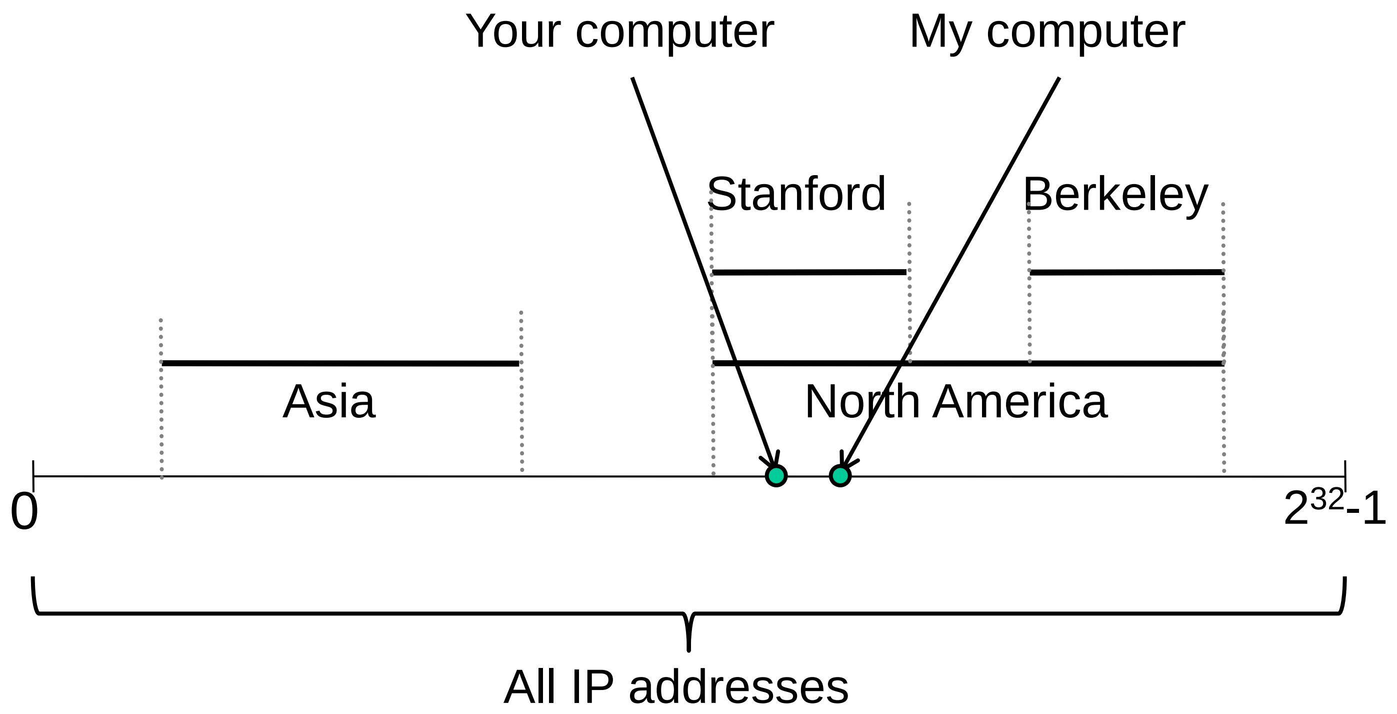
Entry	Destination	Port
1	0.0.0.0	1
2	0.0.0.1	2
⋮	⋮	⋮
2^{32}	255.255.255.255	12

} ~ 4 billion entries

Improved approach:
Group entries to reduce table size

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

IP addresses as a line



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

Longest Prefix Match (LPM)

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

Universities

Continents

Planet

Matching entries:

- Stanford Most specific
- North America
- Everywhere

To: Stanford	Data

Longest Prefix Match (LPM)

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

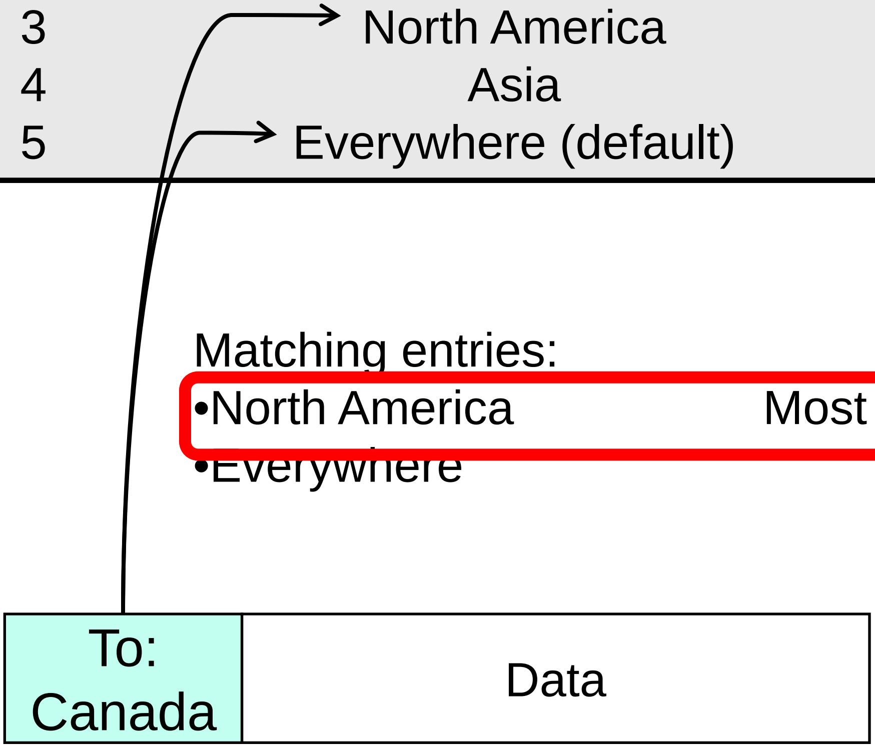
Universities

Continents

Planet

Matching entries:

- North America — Most specific
- Everywhere



Implementing Longest Prefix Match

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

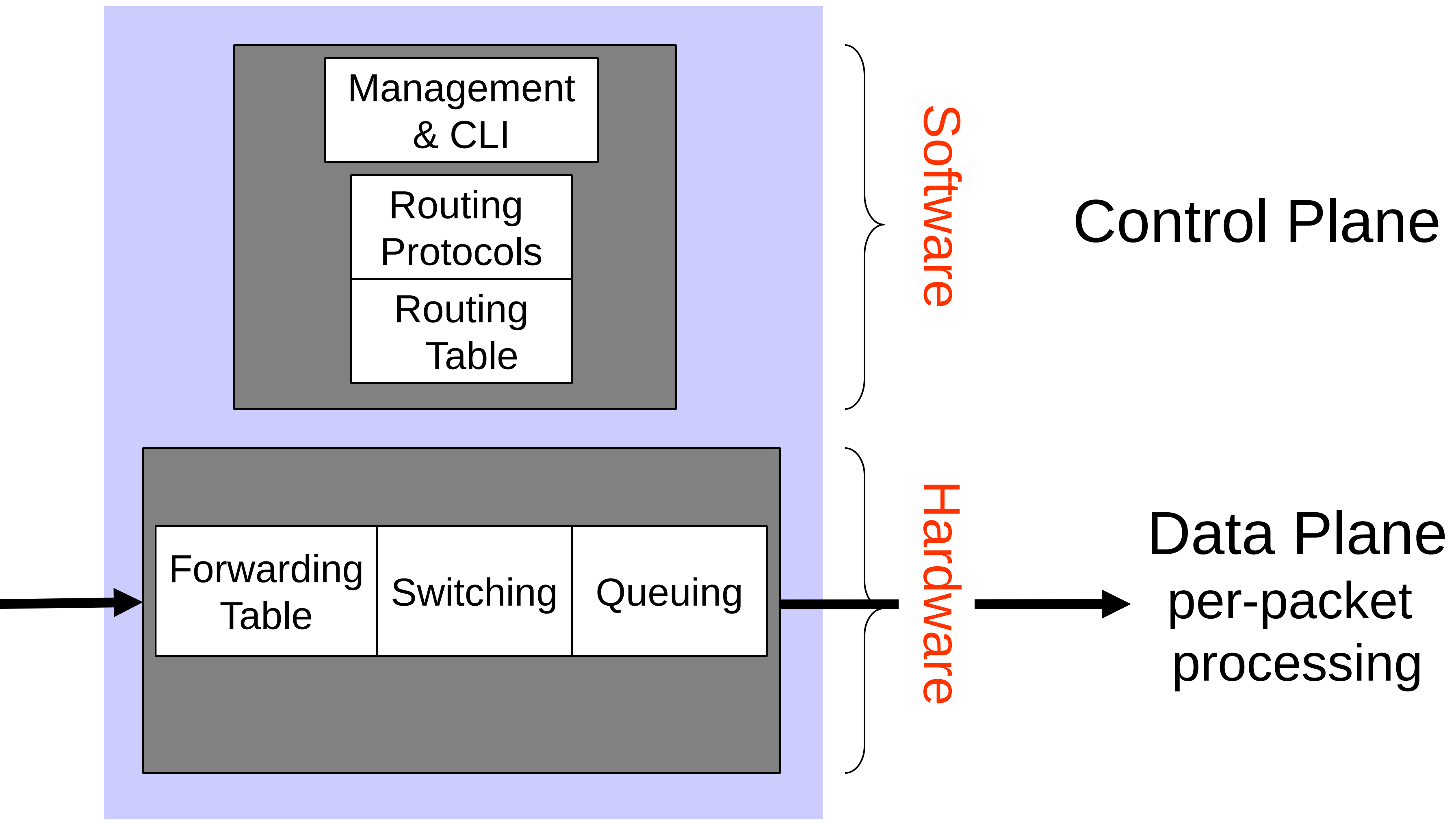
Searching

FOUND

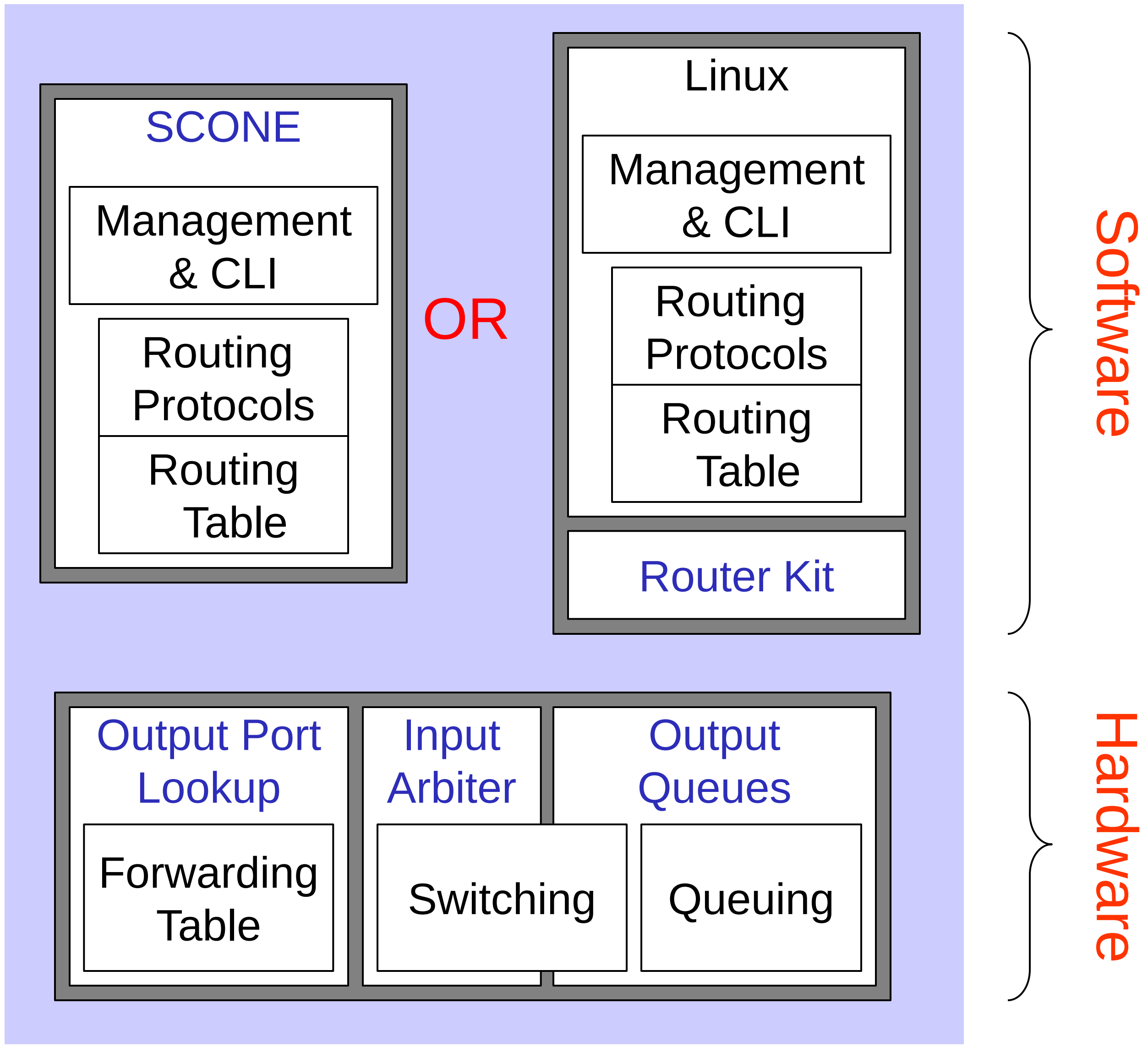
Most specific

Least specific

Basic components of an IP router

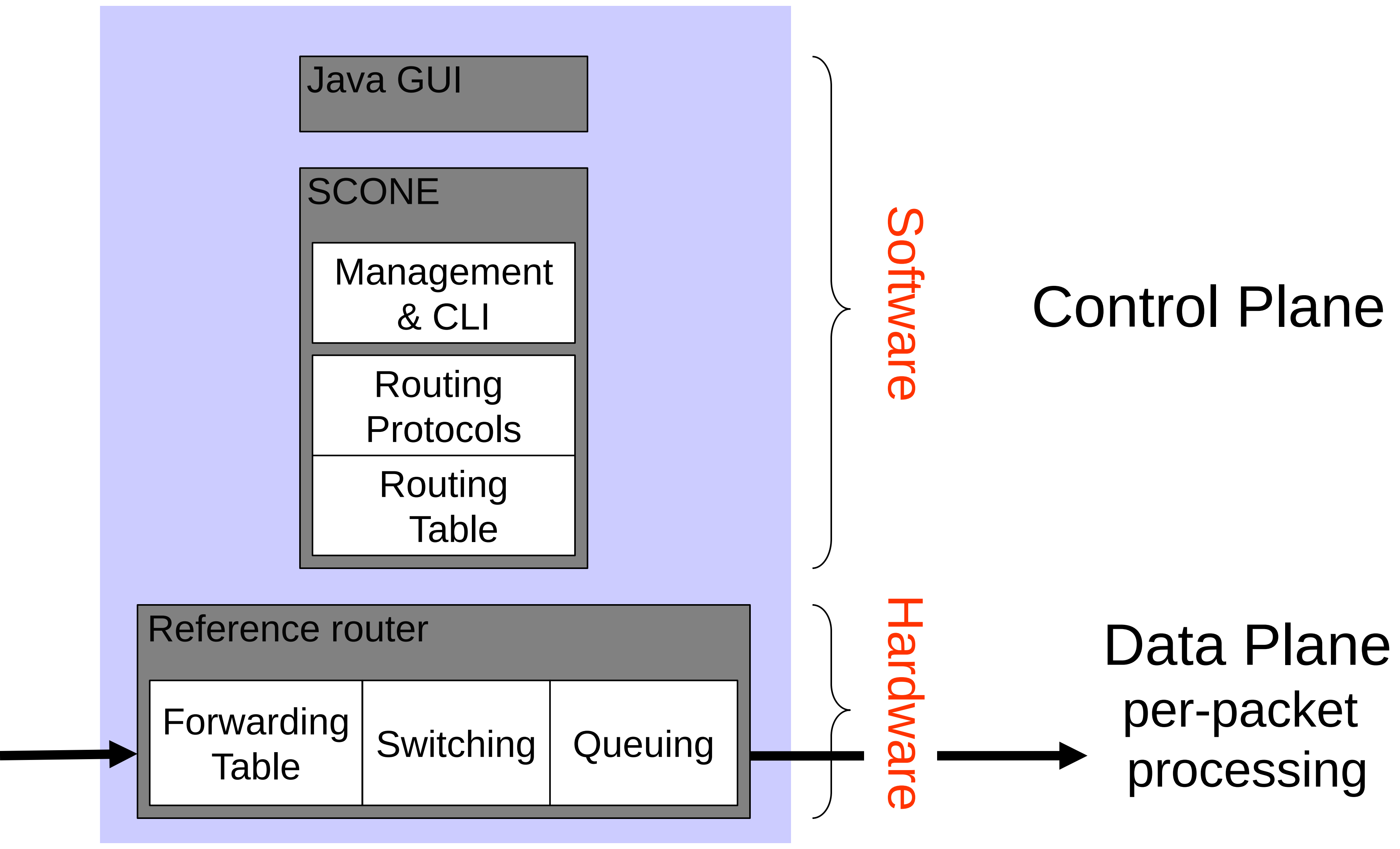


IP router components in NetFPGA

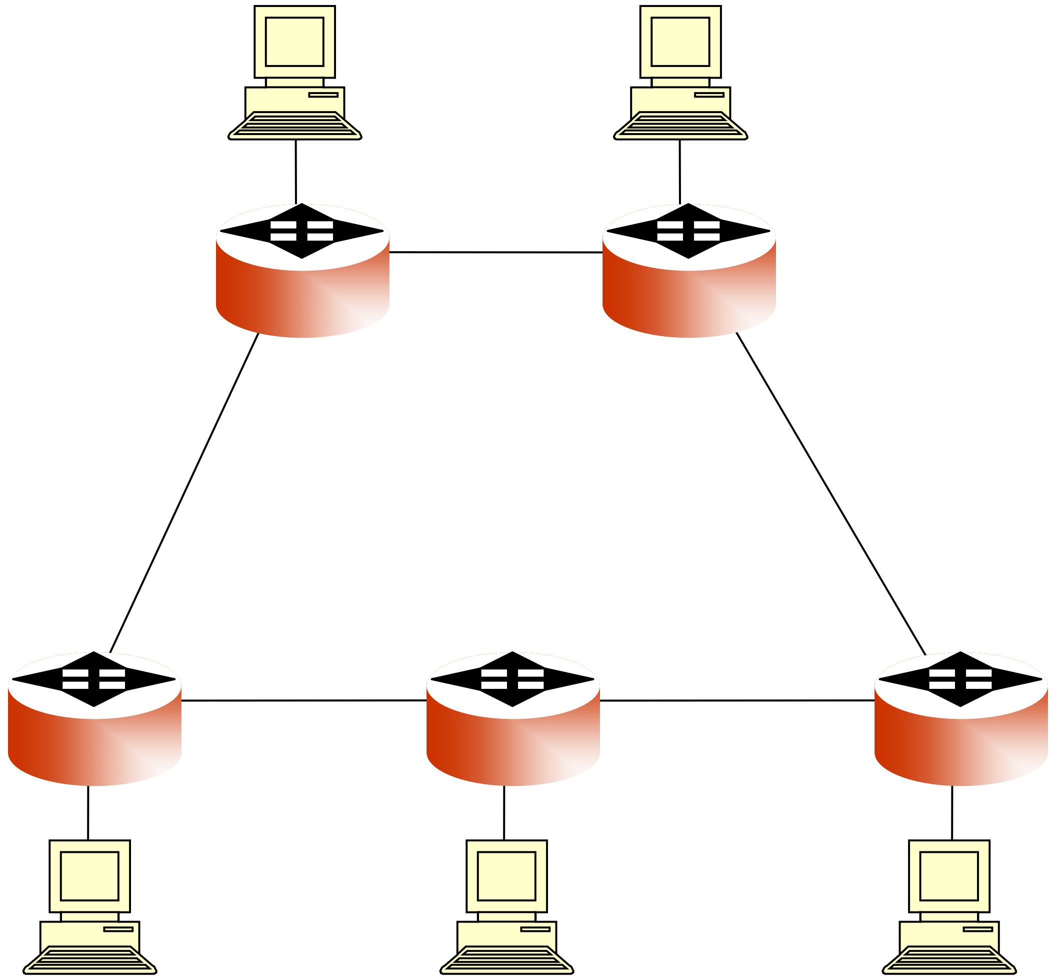


Section IV: Example I

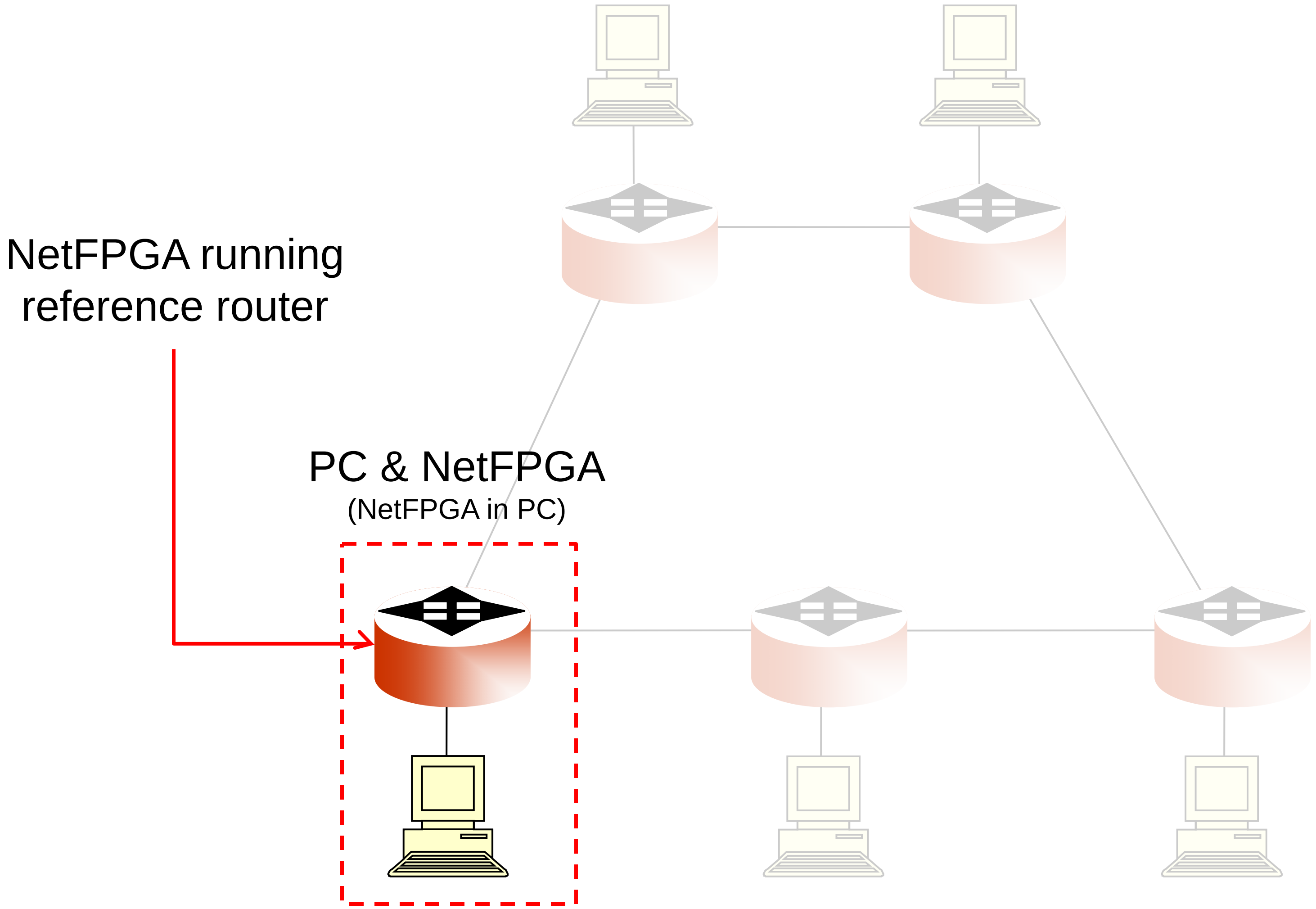
Operational IPv4 router



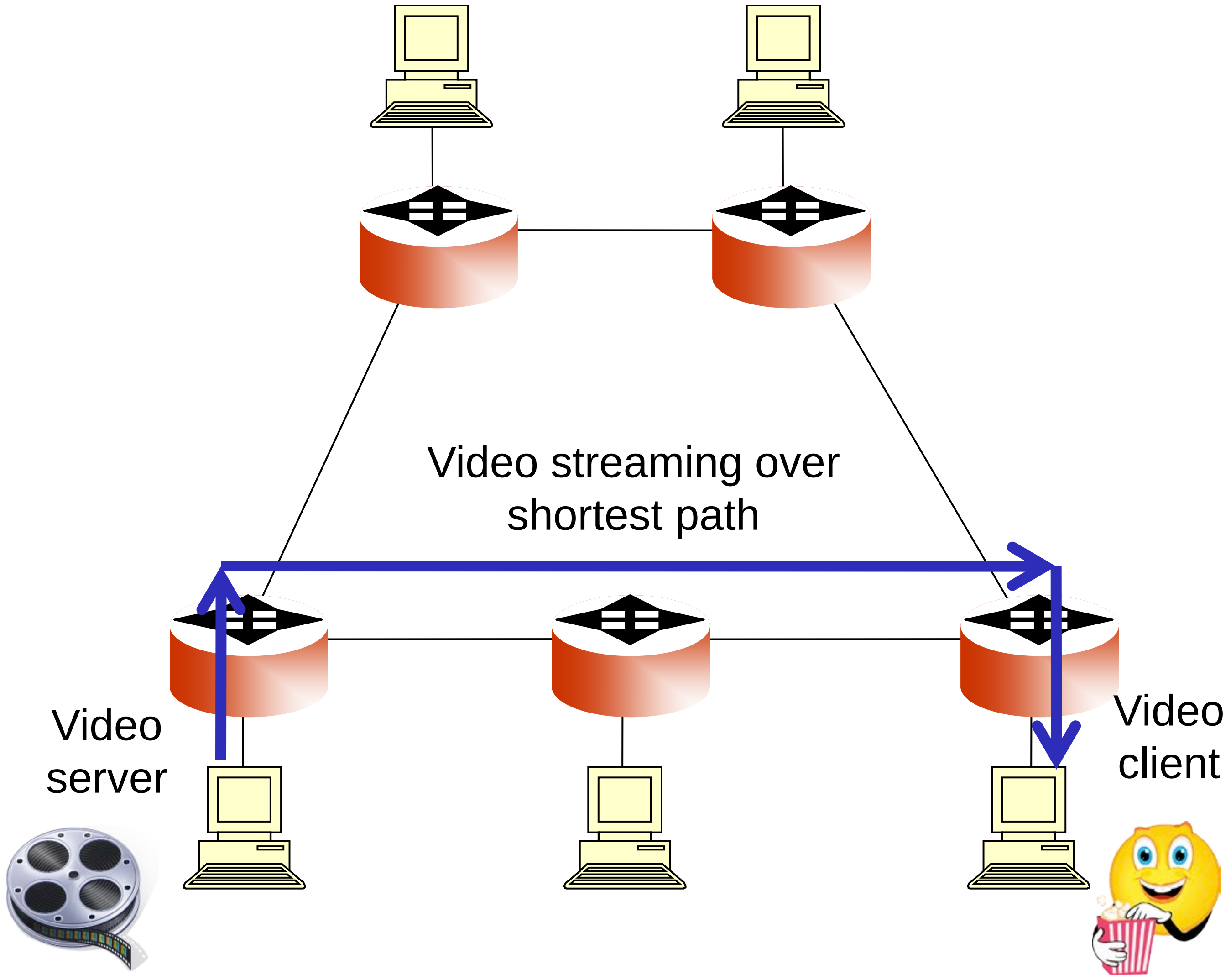
Streaming video



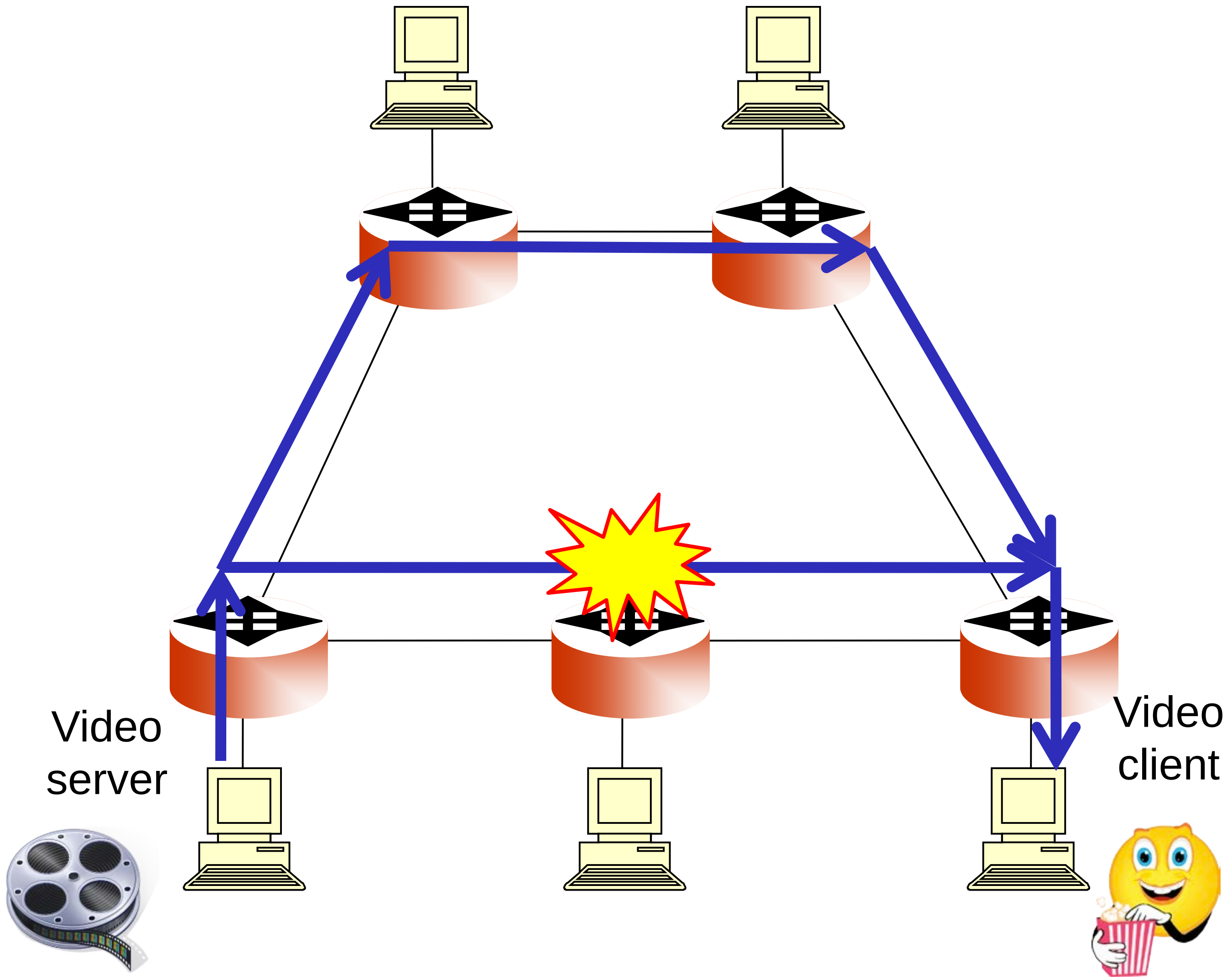
Streaming video



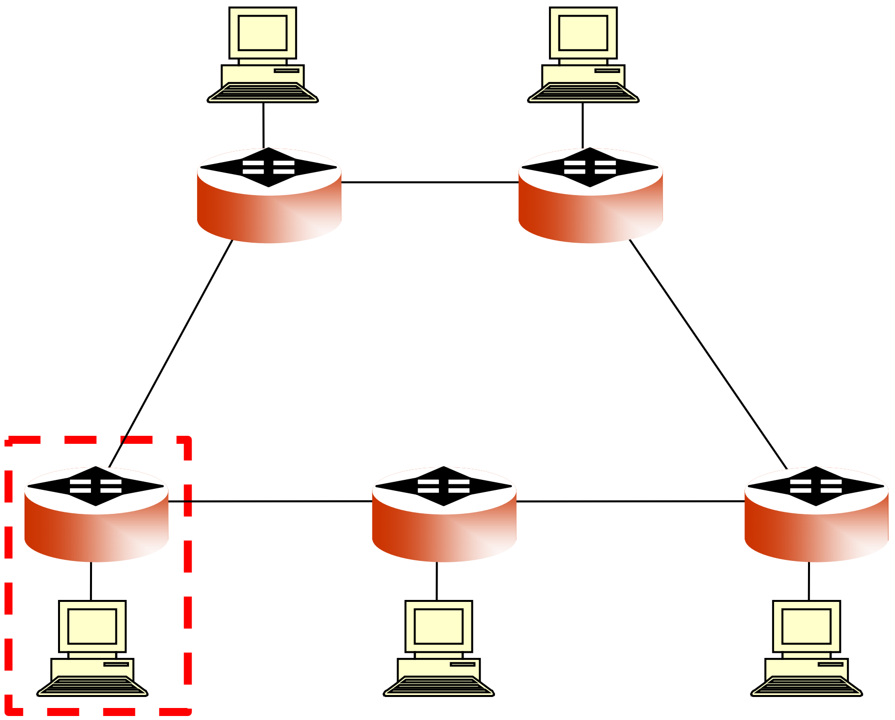
Streaming video



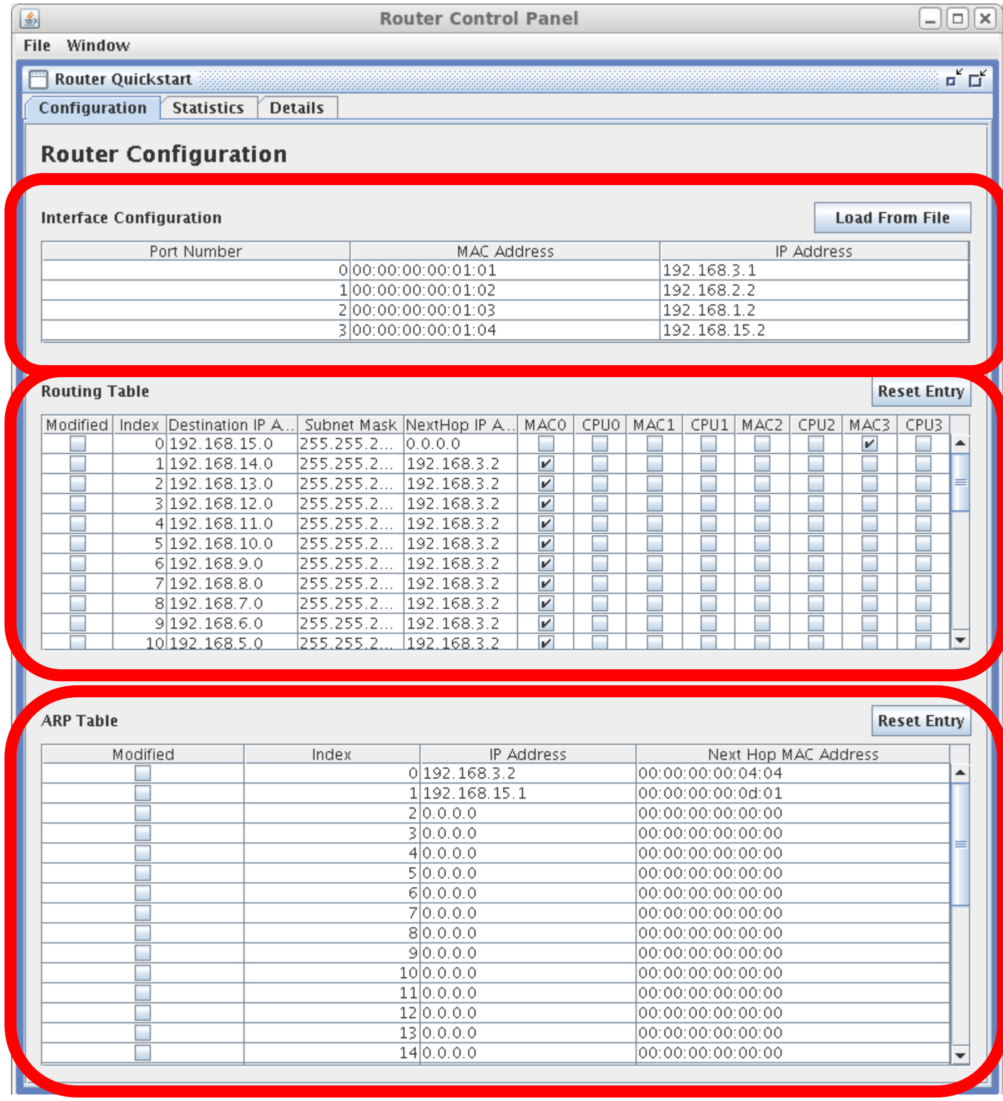
Streaming video



Observing the routing tables



Columns:
- Subnet address
- Subnet mask
- Next hop IP
- Output ports



Router Configuration

Interface Configuration

Port Number	MAC Address	IP Address
0	00:00:00:00:01:01	192.168.3.1
1	00:00:00:00:01:02	192.168.2.2
2	00:00:00:00:01:03	192.168.1.2
3	00:00:00:00:01:04	192.168.15.2

Routing Table

Modified	Index	Destination IP A...	Subnet Mask	NextHop IP A...	MAC0	CPU0	MAC1	CPU1	MAC2	CPU2	MAC3	CPU3
☐	0	192.168.15.0	255.255.2...	0.0.0.0	☐	☐	☐	☐	☐	☐	✓	☐
☐	1	192.168.14.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	2	192.168.13.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	3	192.168.12.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	4	192.168.11.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	5	192.168.10.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	6	192.168.9.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	7	192.168.8.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	8	192.168.7.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	9	192.168.6.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐
☐	10	192.168.5.0	255.255.2...	192.168.3.2	✓	☐	☐	☐	☐	☐	☐	☐

ARP Table

Modified	Index	IP Address	Next Hop MAC Address
☐	0	192.168.3.2	00:00:00:00:04:04
☐	1	192.168.15.1	00:00:00:00:0d:01
☐	2	0.0.0.0	00:00:00:00:00:00
☐	3	0.0.0.0	00:00:00:00:00:00
☐	4	0.0.0.0	00:00:00:00:00:00
☐	5	0.0.0.0	00:00:00:00:00:00
☐	6	0.0.0.0	00:00:00:00:00:00
☐	7	0.0.0.0	00:00:00:00:00:00
☐	8	0.0.0.0	00:00:00:00:00:00
☐	9	0.0.0.0	00:00:00:00:00:00
☐	10	0.0.0.0	00:00:00:00:00:00
☐	11	0.0.0.0	00:00:00:00:00:00
☐	12	0.0.0.0	00:00:00:00:00:00
☐	13	0.0.0.0	00:00:00:00:00:00
☐	14	0.0.0.0	00:00:00:00:00:00

Example 1

http://www.youtube.com/watch?v=xU5DM5Hzqes

Review Exercise 1

NetFPGA as IPv4 router:

- Reference hardware + SCONE software
- Routing protocol discovers topology

Example 1:

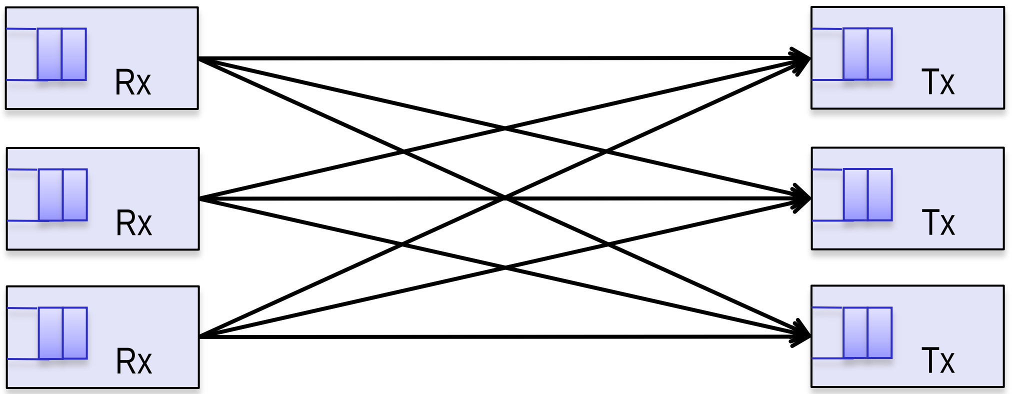
- Ring topology
- Traffic flows over shortest path
- Broken link: automatically route around failure

Section IV: Example II

Buffers in Routers

- Internal Contention
- Congestion
- Pipelining

Buffers in Routers



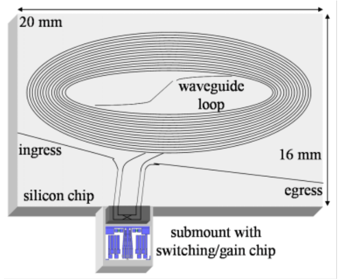
Buffers in Routers

- So how large should the buffers be?

Buffer size matters

– End-to-end delay
  - Transmission, propagation, and queueing delay
  - The only variable part is queueing delay

– Router architecture
  - Board space, power consumption, and cost
  - On chip buffers: higher density, higher
  - Optical buffers: all-optical routers



Buffer Sizing Story

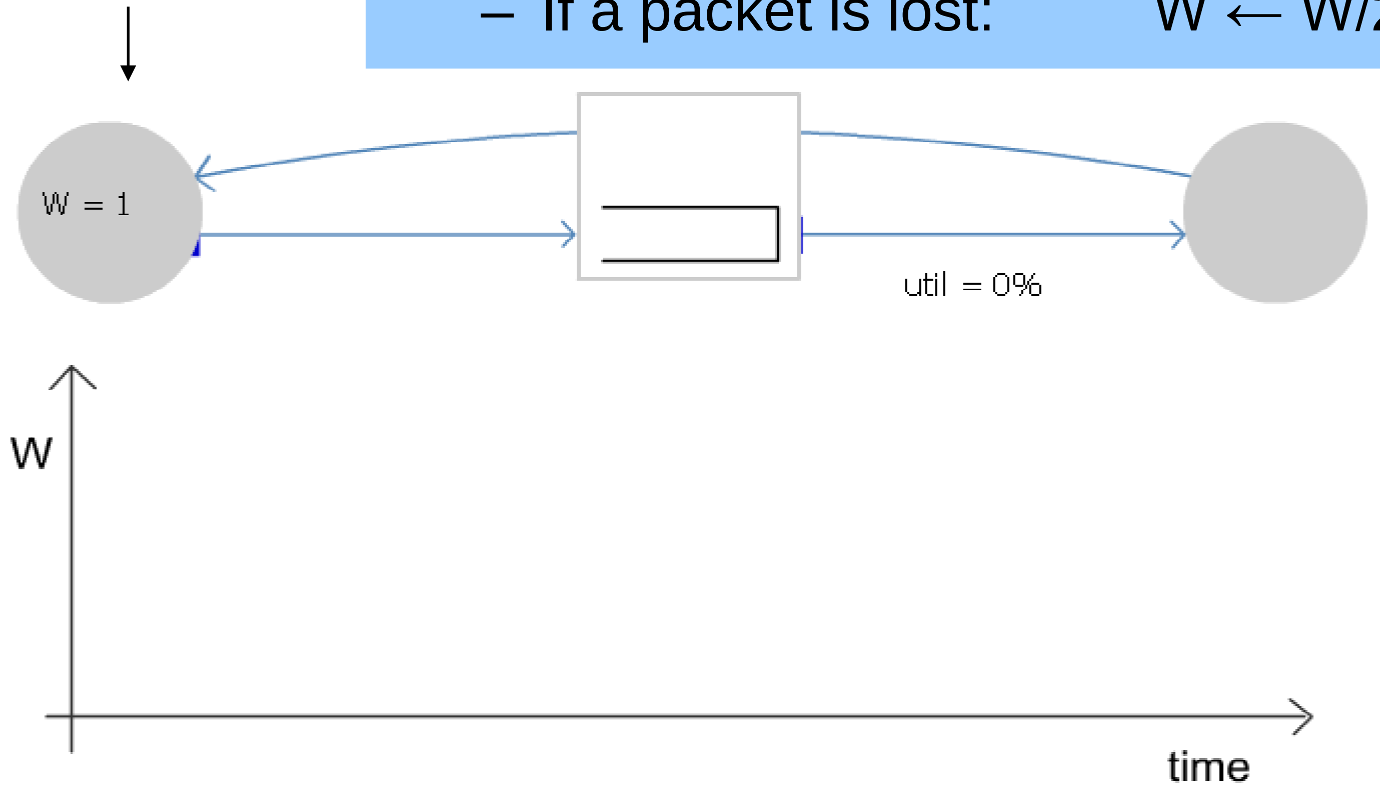
	Rule-of-thumb	Small Buffers	Tiny Buffers
	$2T \times C$	$\frac{2T \times C}{\sqrt{n}}$	$O(\log W)$
# of packets	1,000,000	10,000	20 - 50
Intuition	TCP Sawtooth	Sawtooth Smoothing	Non-bursty Arrivals
Assume	Single TCP Flow, 100% Utilization	Many Flows, 100% Utilization	Paced TCP, 85-90% Utilization
Evidence	Simulation, Emulation	Simulations, Test-bed and Real Network Experiments	Simulations, Test-bed Experiments

Why 2TxC for a single TCP Flow?

Rule for adjusting W

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

Only W packets may be outstanding

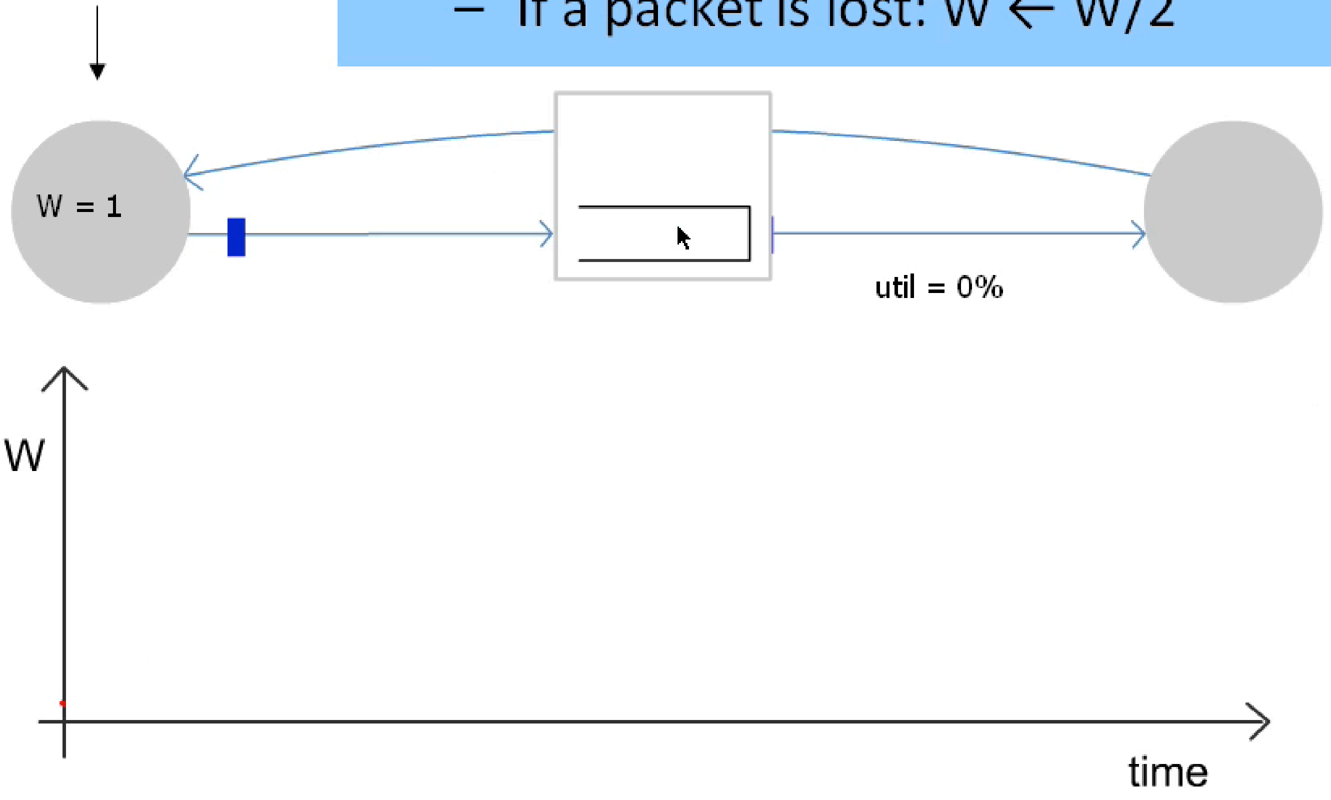


Continuous ARQ (TCP) adapting to congestion

Rule for adjusting W

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

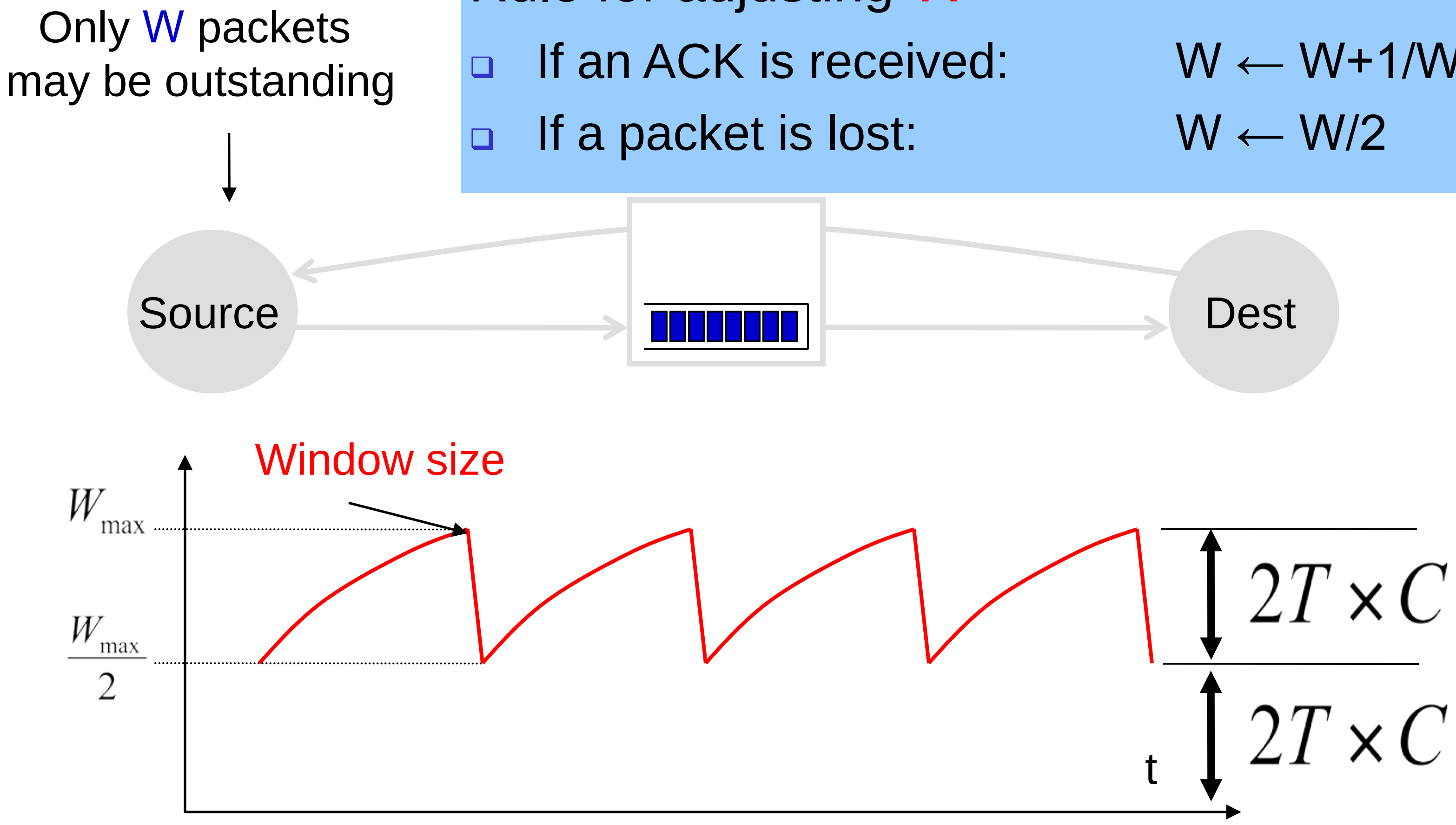
Only W packets may be outstanding



Rule-of-thumb – Intuition

Rule for adjusting W

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



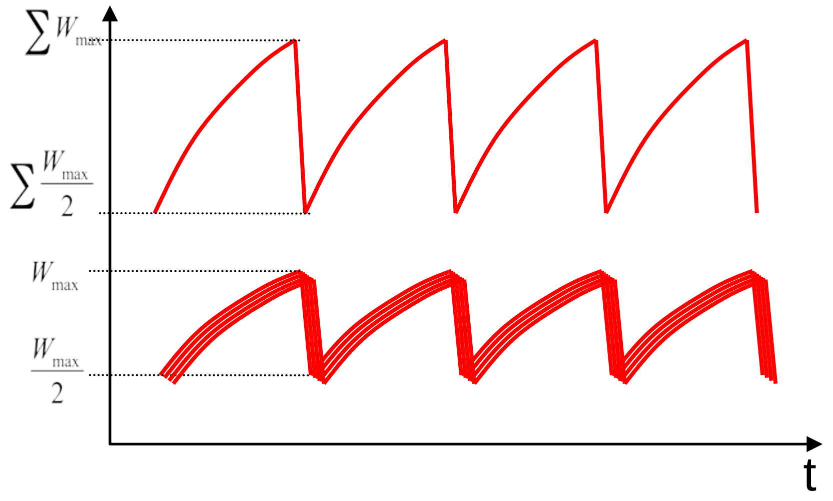
Small Buffers – Intuition

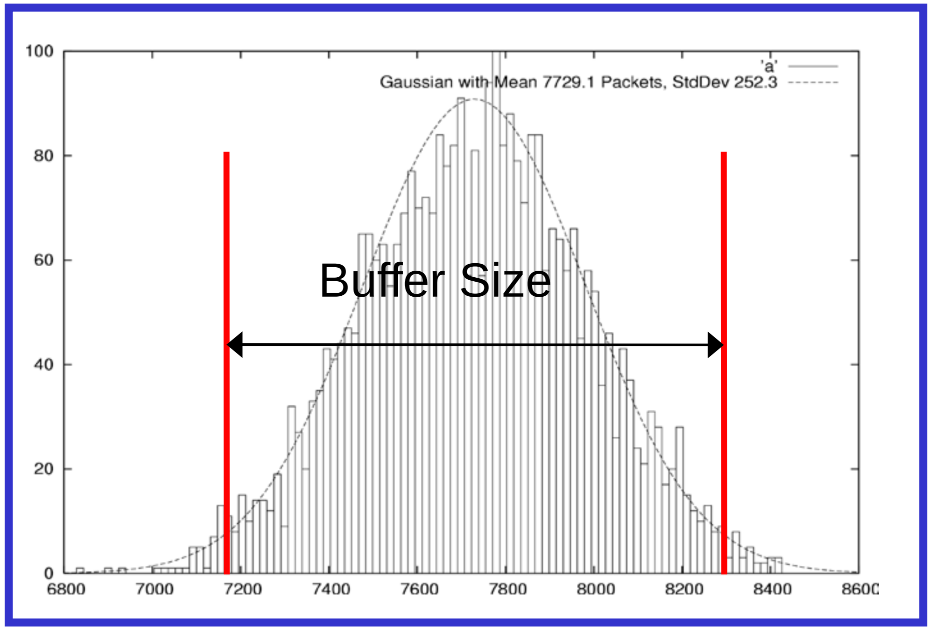
Synchronized Flows

- Aggregate window has same dynamics
- Therefore buffer occupancy has same dynamics
- Rule-of-thumb still holds.

Many TCP Flows

- Independent, desynchronized
- Central limit theorem says the aggregate becomes Gaussian
- Variance (buffer size) decreases as N increases

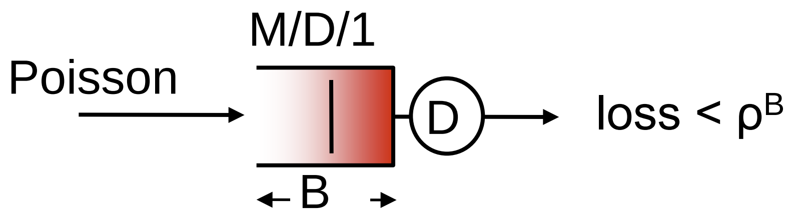




Tiny Buffers – Intuition

Poisson Traffic

- Theory. For Poisson arrivals tiny buffers are enough.



- Example: ρ = 80%, B = 20 pkts ➔ loss < 1%
- Loss independent of link rate, RTT, number of flows, etc.
- Question. Can we make traffic look like Poisson when it arrives to the core routers?

Smooth Traffic

- Assumptions:
  - Minimum distance between consecutive packets of the same flow;
  - Desynchronized flows
  - Random and independent start times for flows
- Under these assumptions traffic is be smooth-enough.
- In practice:
  - Slow access links
  - TCP Pacing

Buffer Sizing Experiments are Difficult

Problem

- Convincing network operators not easy
- Packet drops are scary
- Varying traffic (shape, load, ...) extremely difficult
- Tiny buffers: no guarantees on assumptions
  - i.e. slow access or pacing

Using NetFPGA to explore buffer size

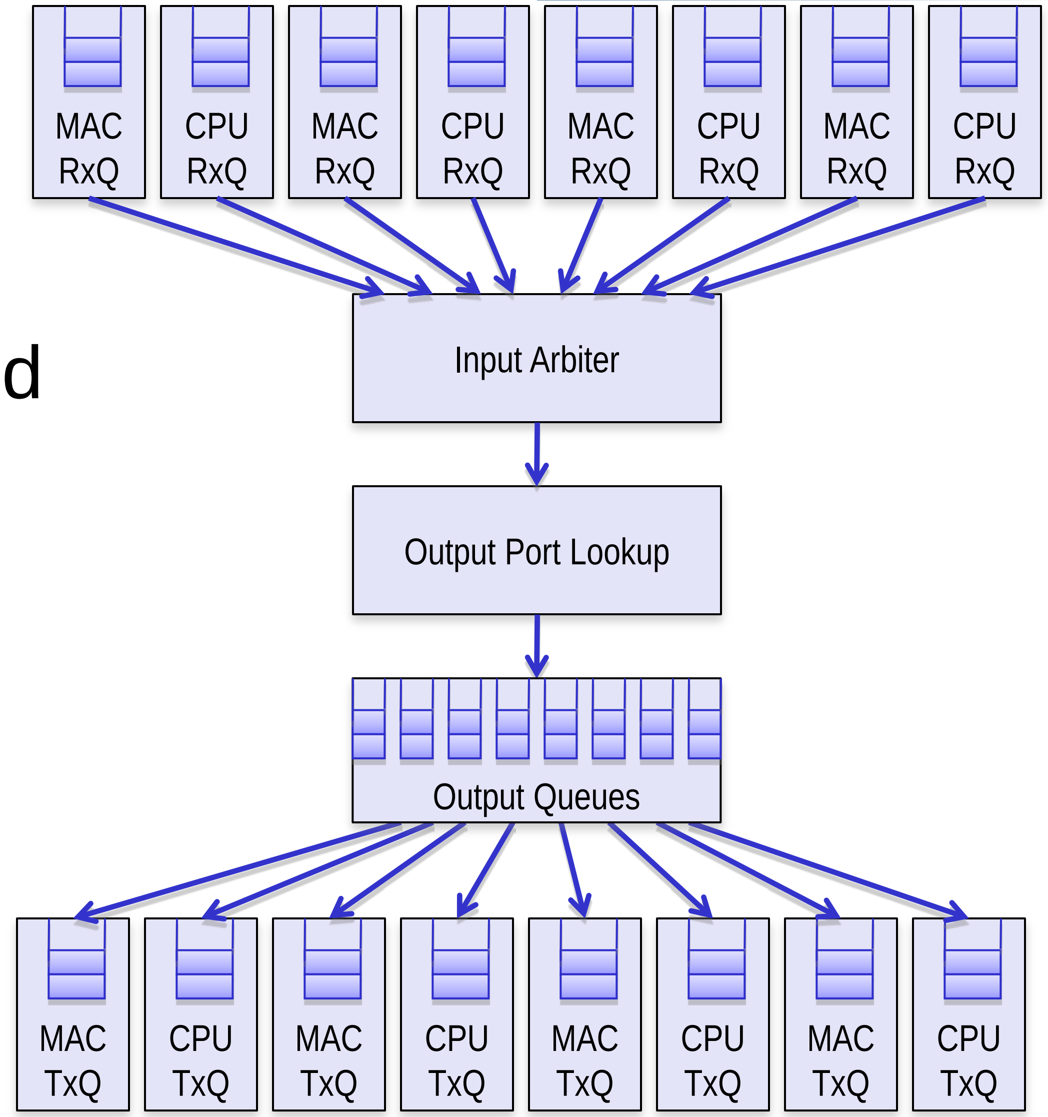
- Need to reduce buffer size and measure occupancy
- Alas, not possible in commercial routers
- So, we will use the NetFPGA instead

Objective:

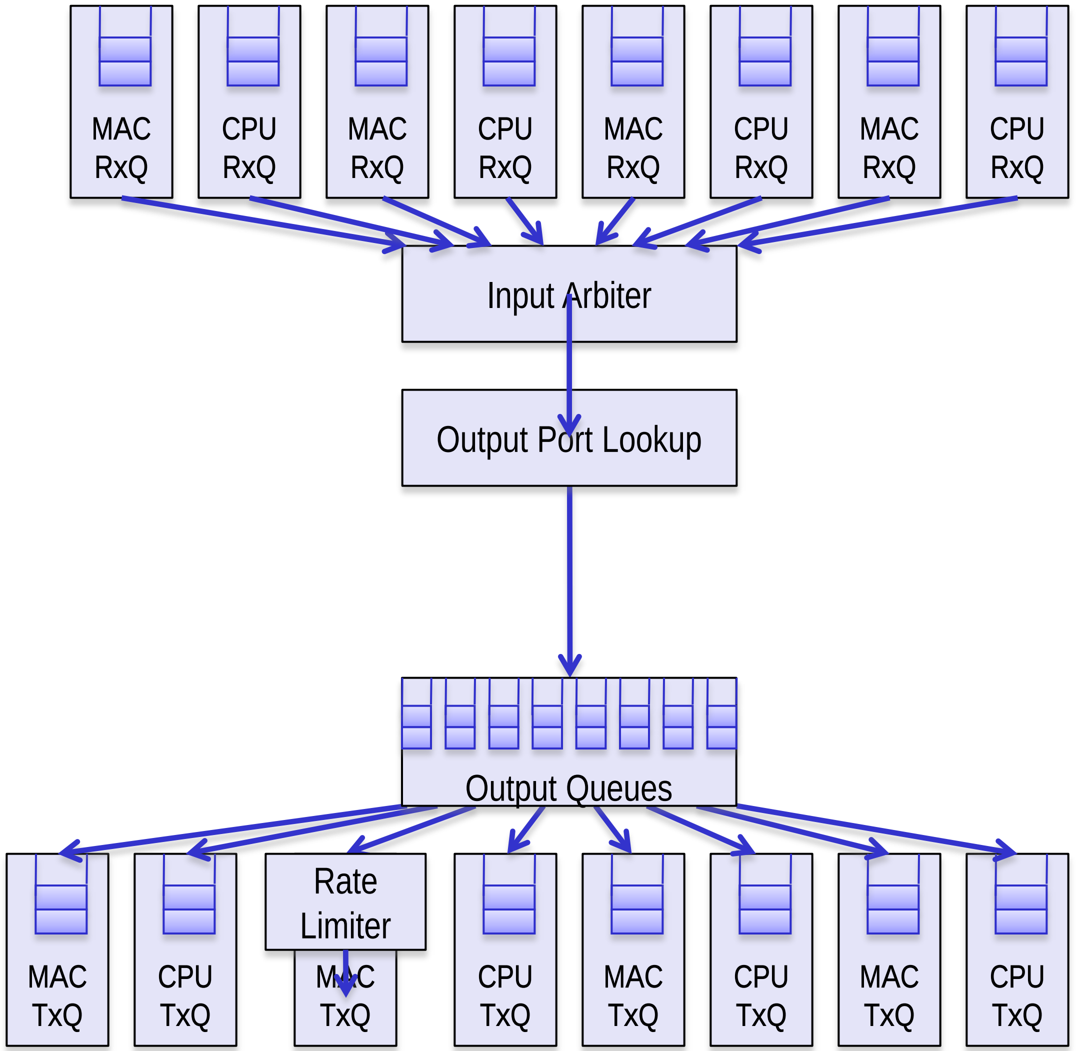
  – Use the NetFPGA to understand how large a buffer we need for a single TCP flow.

Reference Router Pipeline

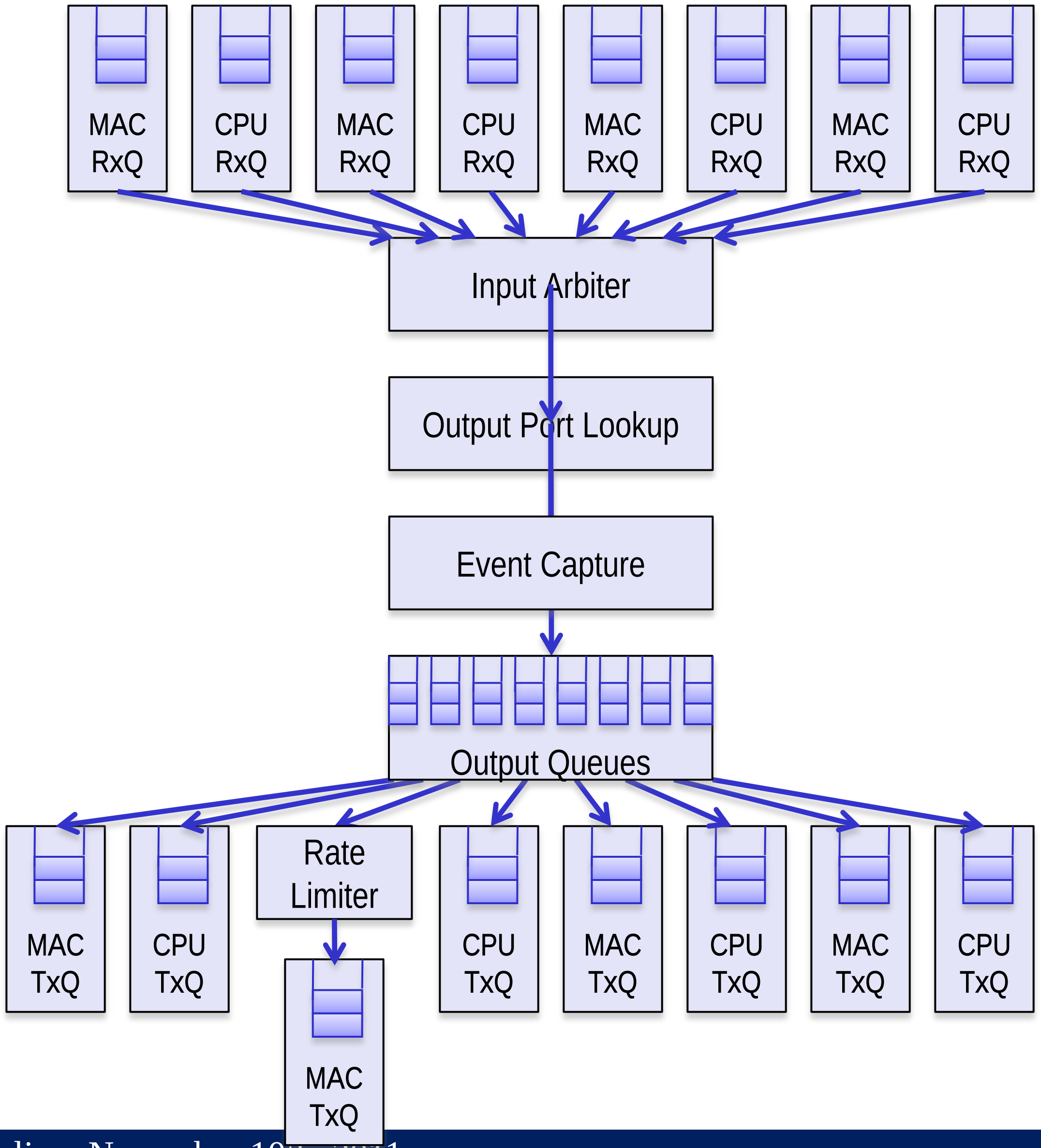
- Five stages
  - Input interfaces
  - Input arbitration
  - Routing decision and packet modification
  - Output queuing
  - Output interfaces
- Packet-based module interface
- Pluggable design



Extending the Reference Pipeline



Extending the Reference Pipeline

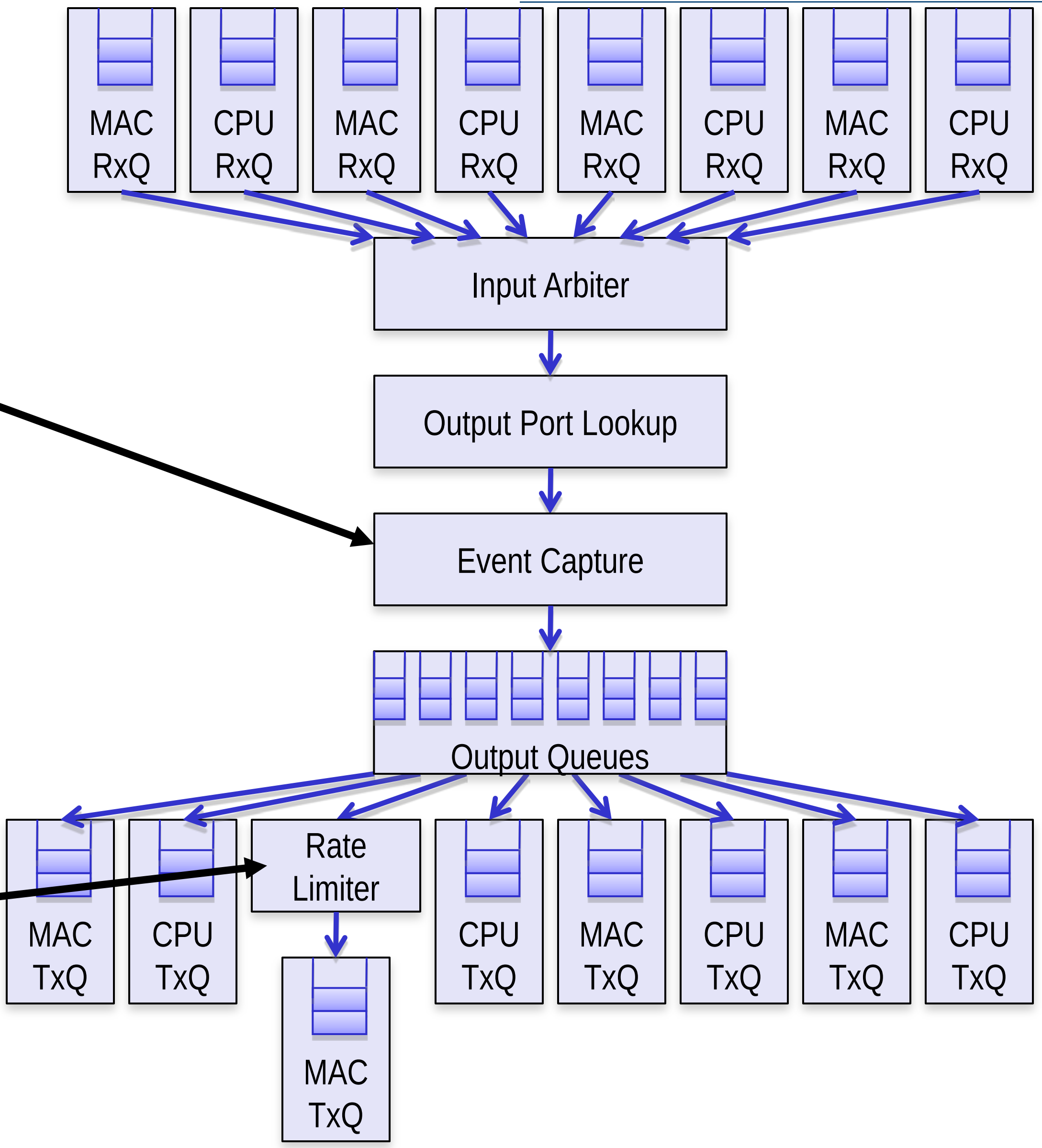


Enhanced Router Pipeline

Two modules added

1. Event Capture to capture output queue events (writes, reads, drops)

2. Rate Limiter to create a bottleneck



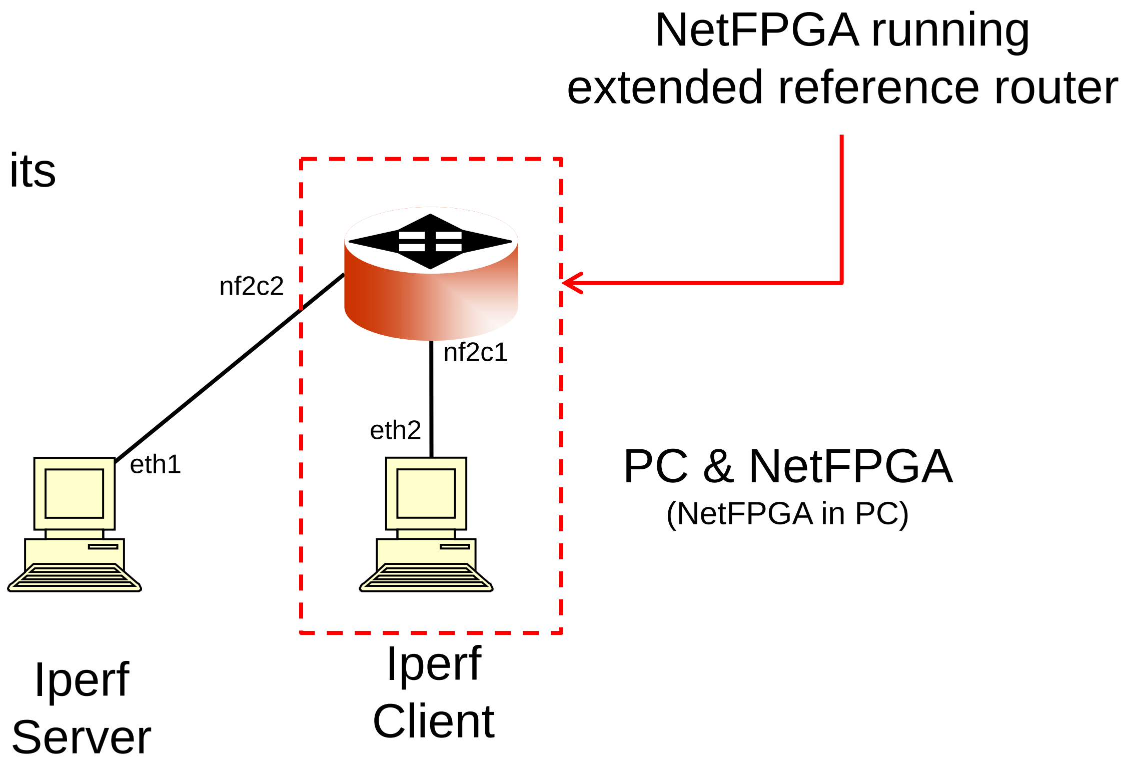
Topology for Exercise 2

Recall:

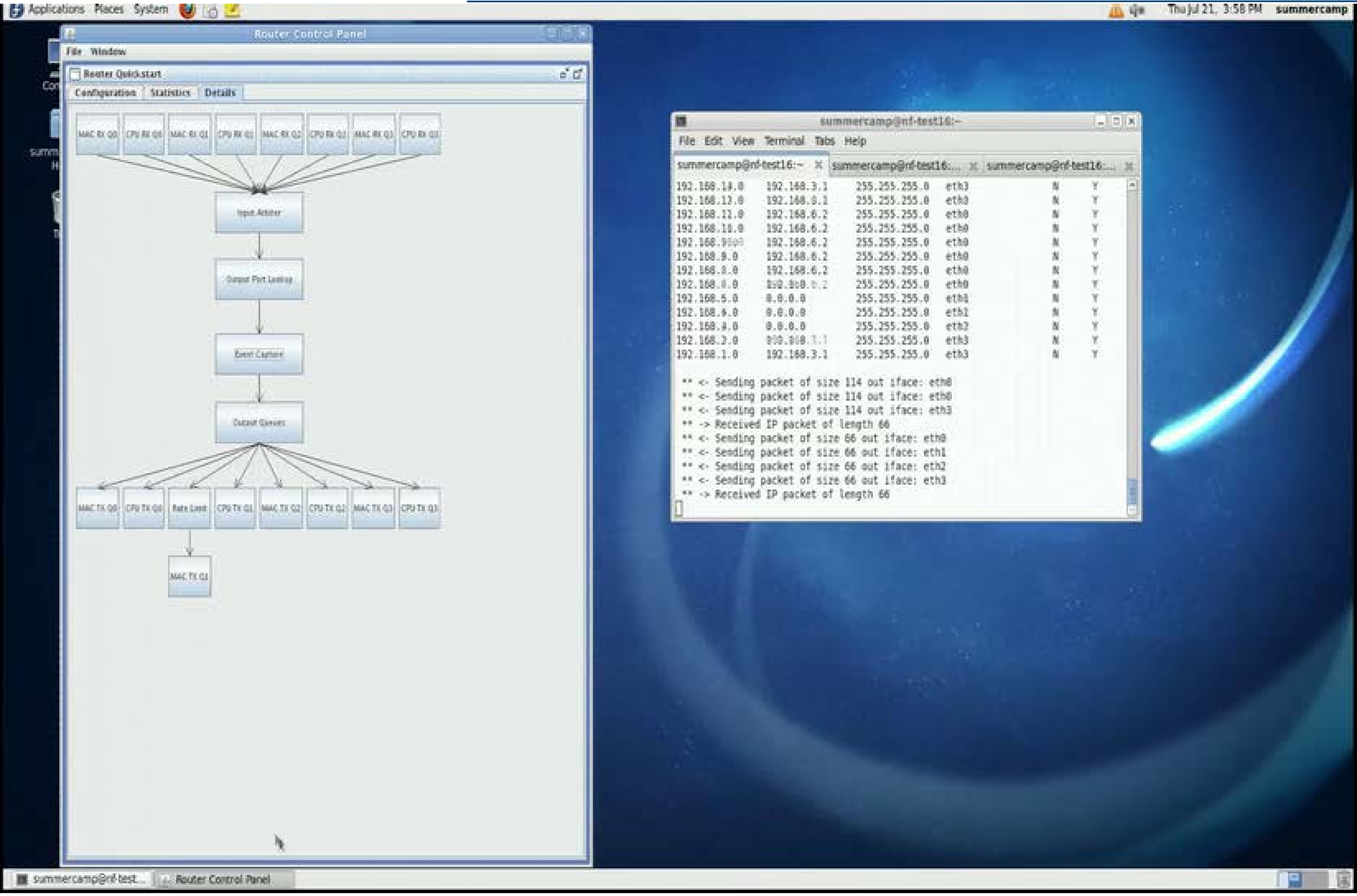
NetFPGA host PC is life-support:
power & control

So:

The host PC may physically route its traffic through the local NetFPGA



Example 2



```
192.168.14.0    192.168.3.1     255.255.255.0   eth3                N     Y
192.168.13.0    192.168.0.1     255.255.255.0   eth0                N     Y
192.168.12.0    192.168.6.2     255.255.255.0   eth0                N     Y
192.168.10.0    192.168.6.2     255.255.255.0   eth0                N     Y
192.168.[illegible]    192.168.6.2     255.255.255.0   eth0                N     Y
192.168.9.0     192.168.6.2     255.255.255.0   eth0                N     Y
192.168.8.0     192.168.6.2     255.255.255.0   eth0                N     Y
192.168.[illegible].0     [illegible]     255.255.255.0   eth0                N     Y
192.168.5.0     0.0.0.0         255.255.255.0   eth1                N     Y
192.168.6.0     0.0.0.0         255.255.255.0   eth1                N     Y
192.168.4.0     0.0.0.0         255.255.255.0   eth2                N     Y
192.168.2.0     [illegible]     255.255.255.0   eth3                N     Y
192.168.1.0     192.168.3.1     255.255.255.0   eth3                N     Y

 ** <- Sending packet of size 114 out iface: eth0
 ** <- Sending packet of size 114 out iface: eth0
 ** <- Sending packet of size 114 out iface: eth3
 ** -> Received IP packet of length 66
 ** <- Sending packet of size 66 out iface: eth0
 ** <- Sending packet of size 66 out iface: eth1
 ** <- Sending packet of size 66 out iface: eth2
 ** <- Sending packet of size 66 out iface: eth3
 ** -> Received IP packet of length 66
```

Review

NetFPGA as flexible platform:

•Reference hardware + SCONE software

•new modules: event capture and rate-limiting

Example 2:

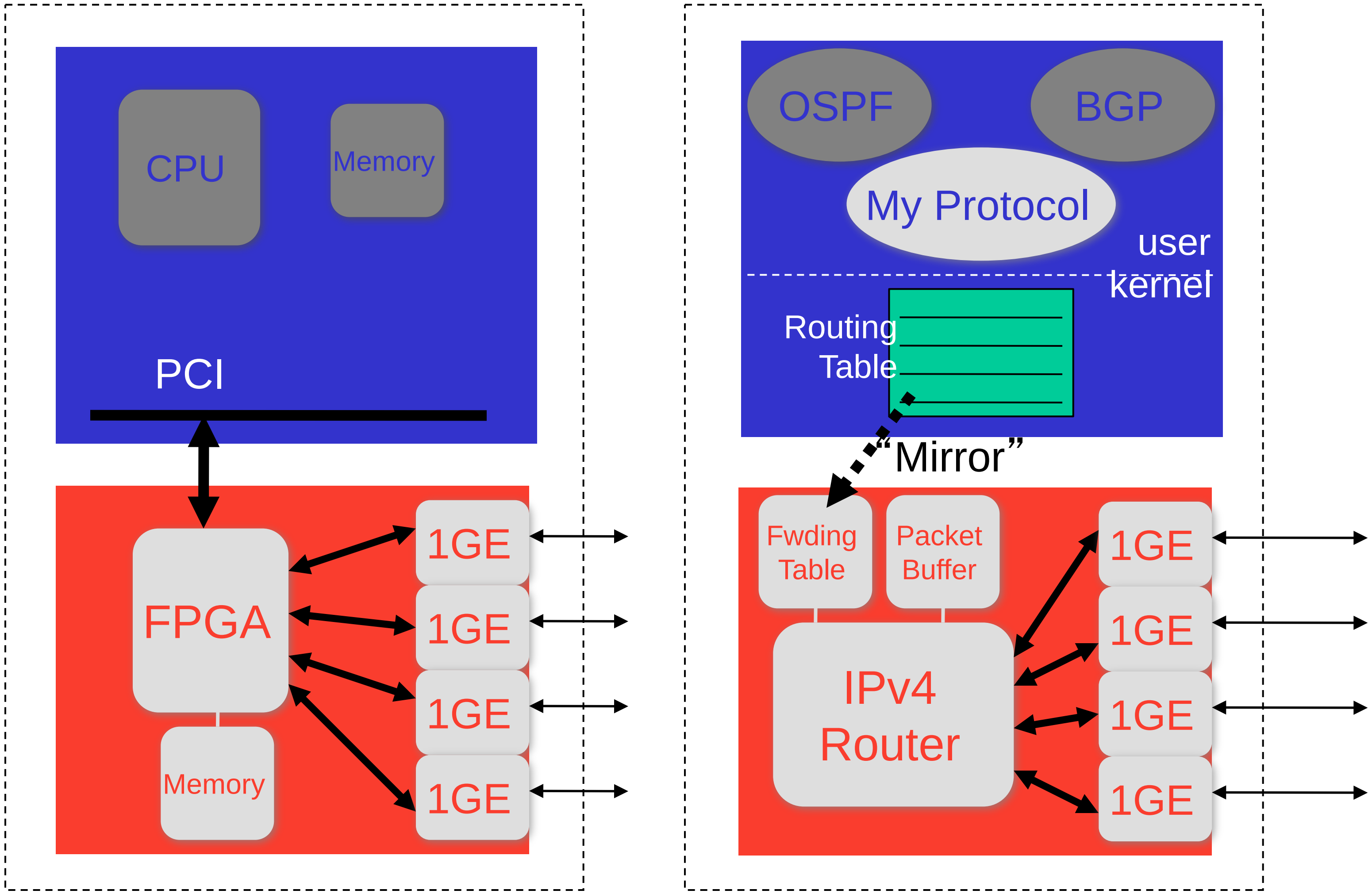
Client ⇔ Router ⇔ Server topology

- Observed router with new modules
- Started tcp transfer, look at queue occupancy
- Observed queue change in response to TCP ARQ

Section V: Community Contributions

Running the Router Kit

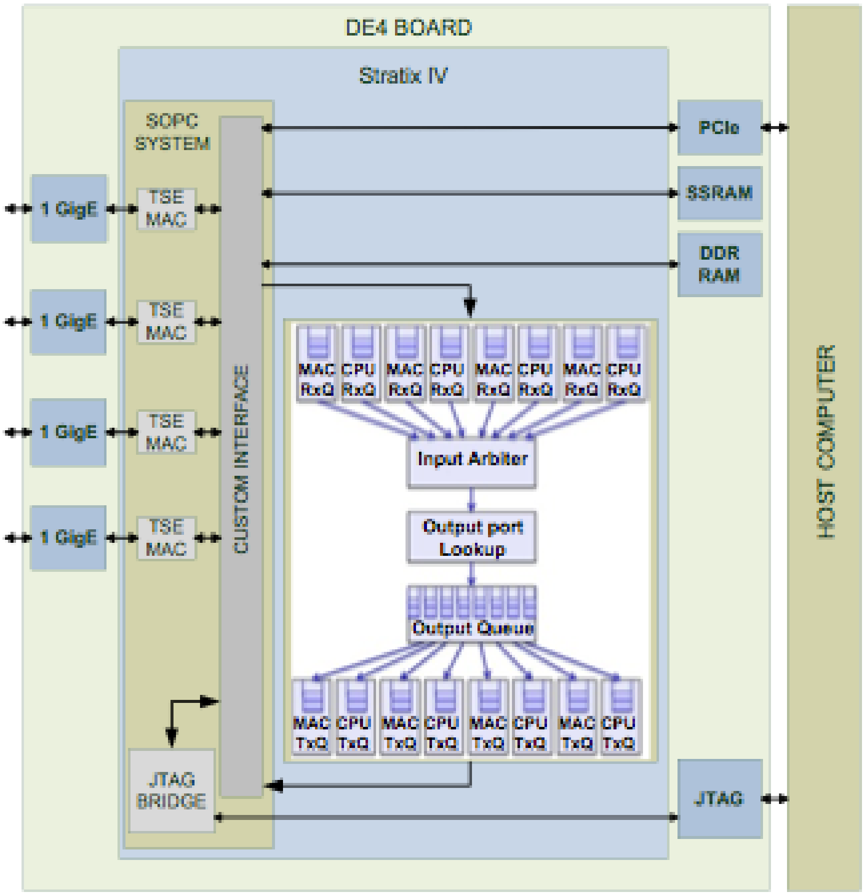
User-space development, 4x1GE line-rate forwarding

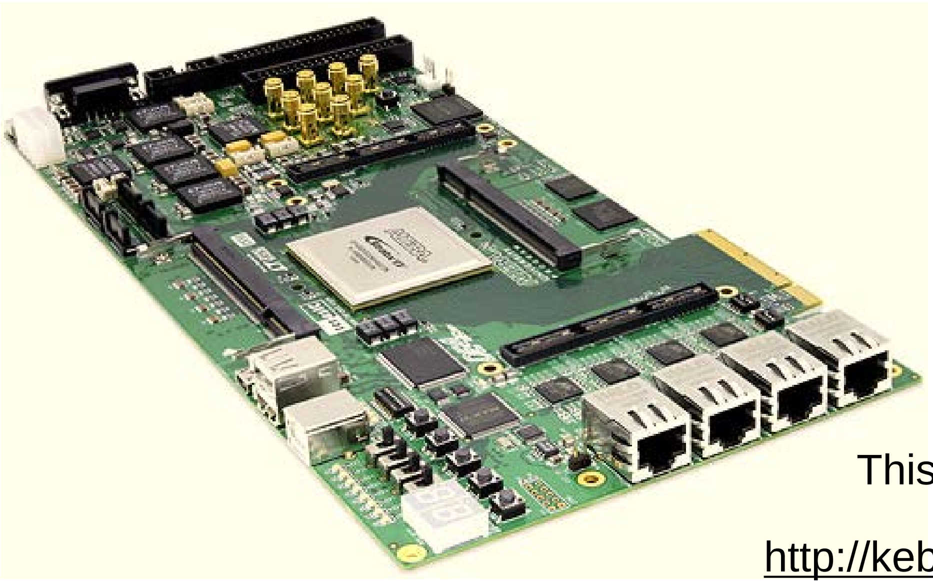


Altera-DE4 NetFPGA Reference Router

UMassAmherst

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



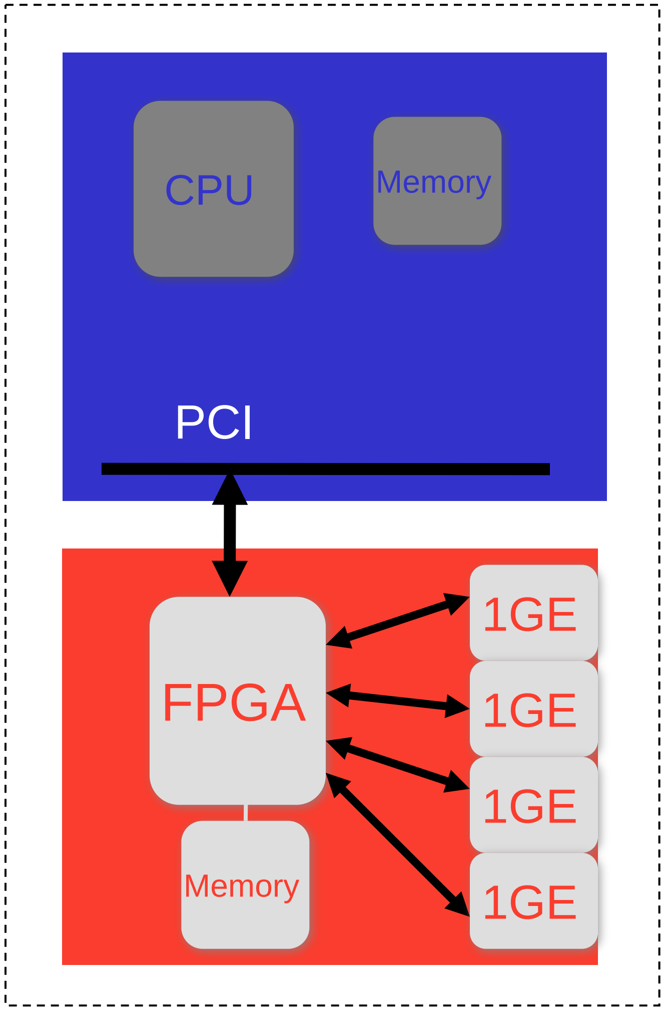


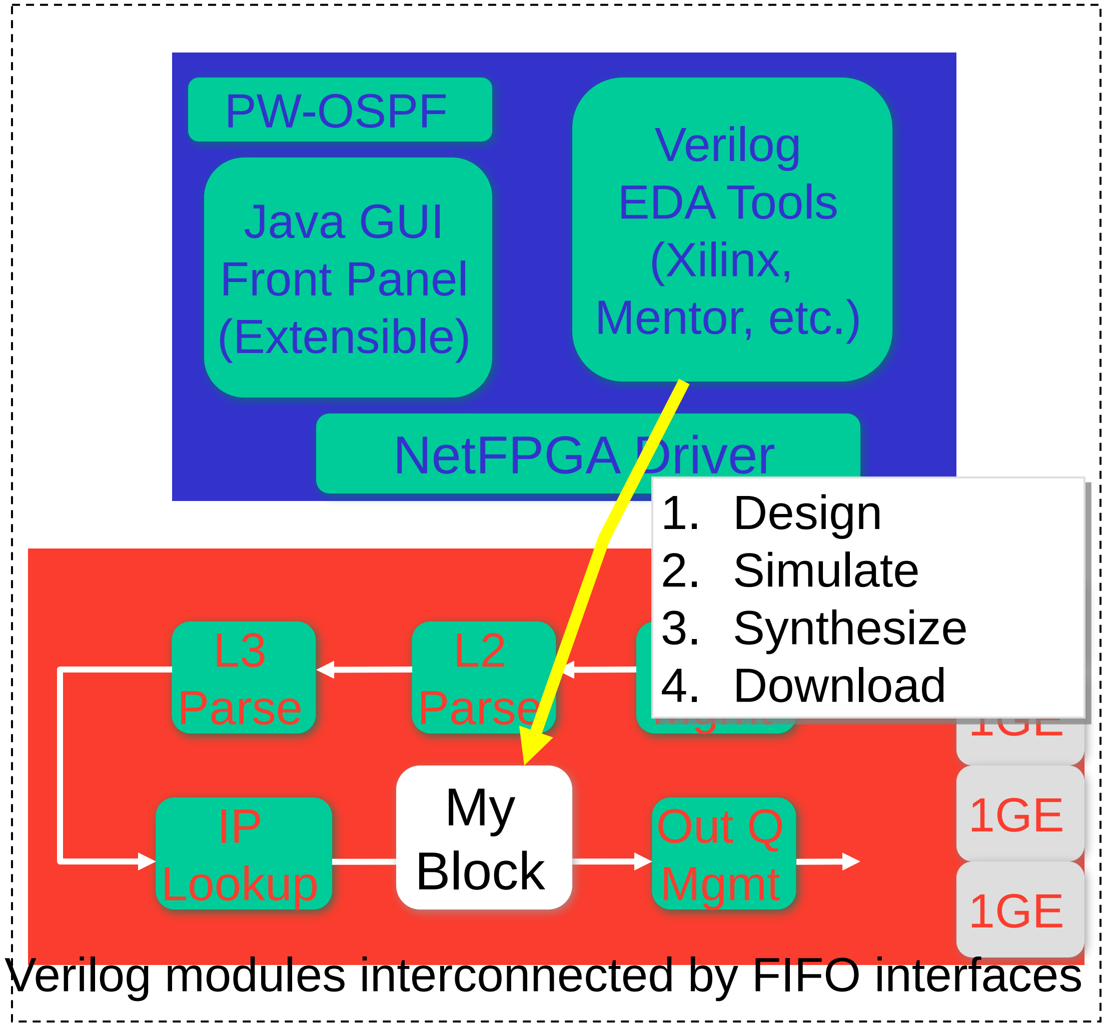
This provides a template for all NetFPGA 1G projects

http://keb302.ecs.umass.edu/de4web/DE4_NetFPGA/

Usage #2

Enhancing Modular Reference Designs



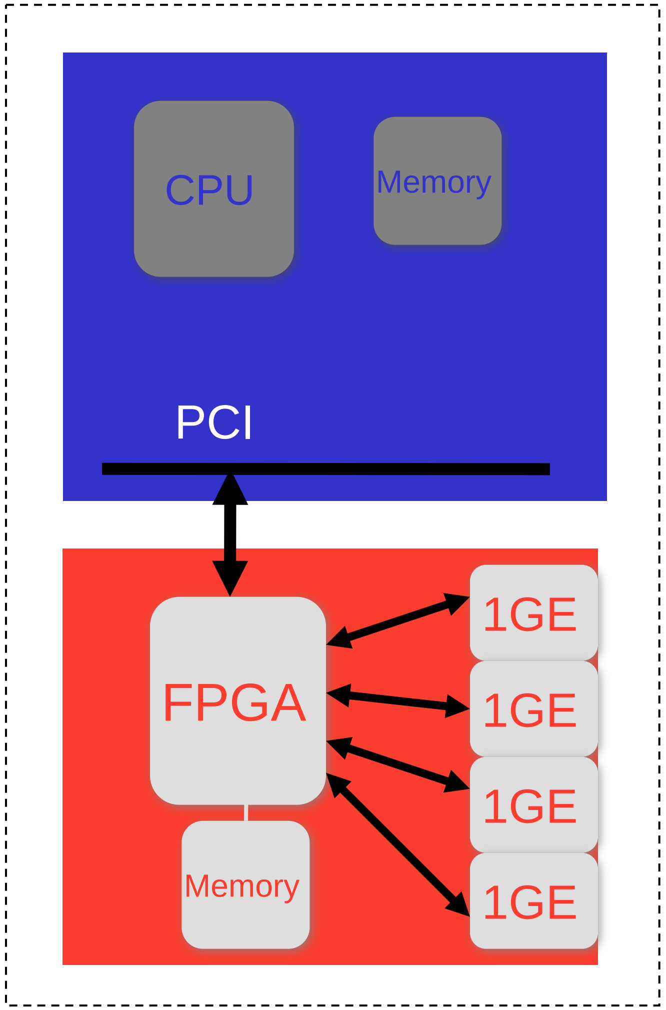

PW-OSPF
Java GUI Front Panel (Extensible)
Verilog EDA Tools (Xilinx, Mentor, etc.)
NetFPGA Driver
L3 Parse
L2 Parse
IP Lookup
My Block
Out Q Mgmt
1GE
1GE
1GE

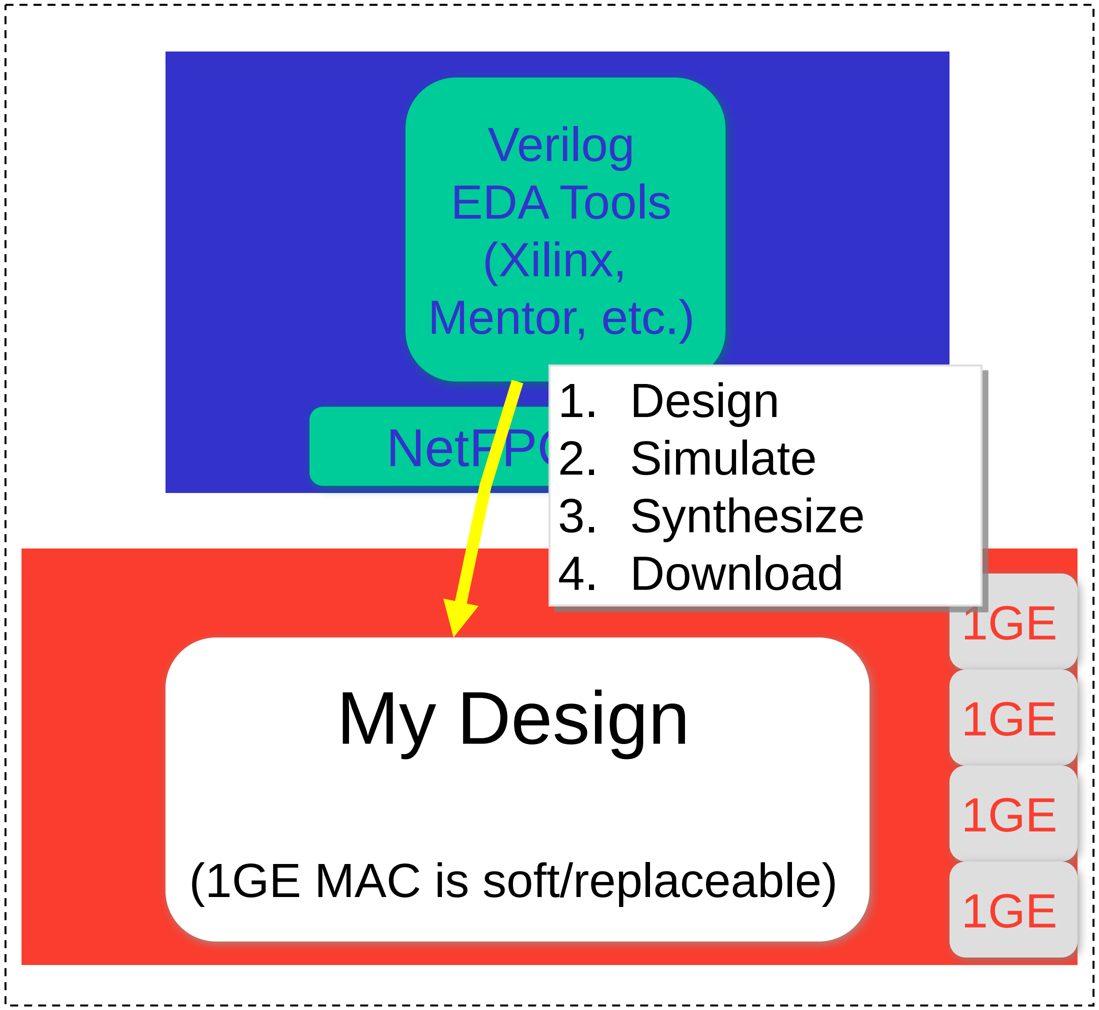
1. Design
2. Simulate
3. Synthesize
4. Download

Verilog modules interconnected by FIFO interfaces

Usage #3

Creating new systems





Verilog EDA Tools (Xilinx, Mentor, etc.)
NetFPG
My Design
(1GE MAC is soft/replaceable)
1GE
1GE
1GE
1GE

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

NetThreads, NetThreads-RE, NetTM

U. of Toronto

Martin Labrecque
Gregory Steffan

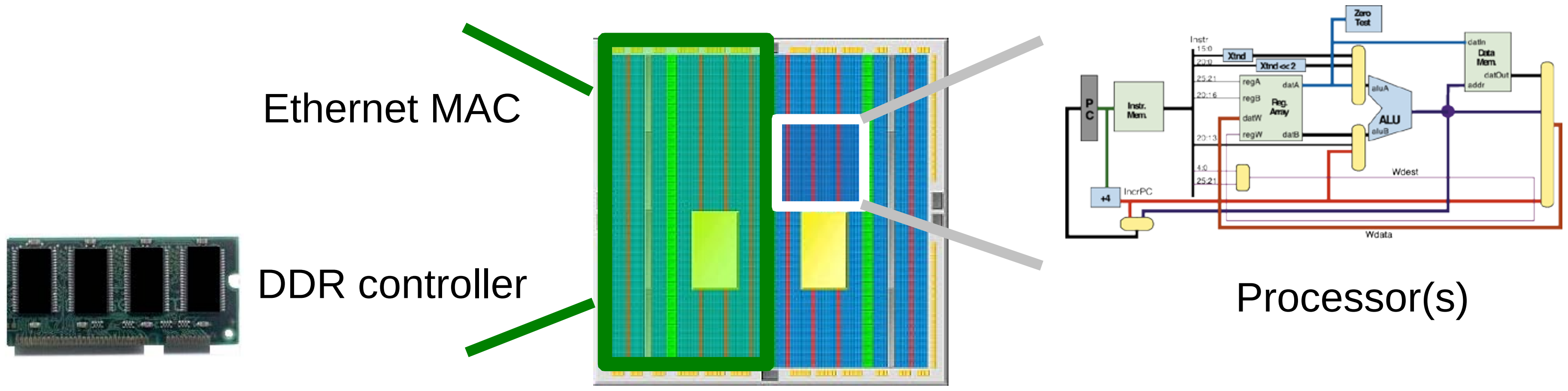
ECE Dept.

Geoff Salmon
Monia Ghobadi
Yashar Ganjali

CS Dept.

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

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

NetThreads

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

Section VI: What to do next?

To get started with your project

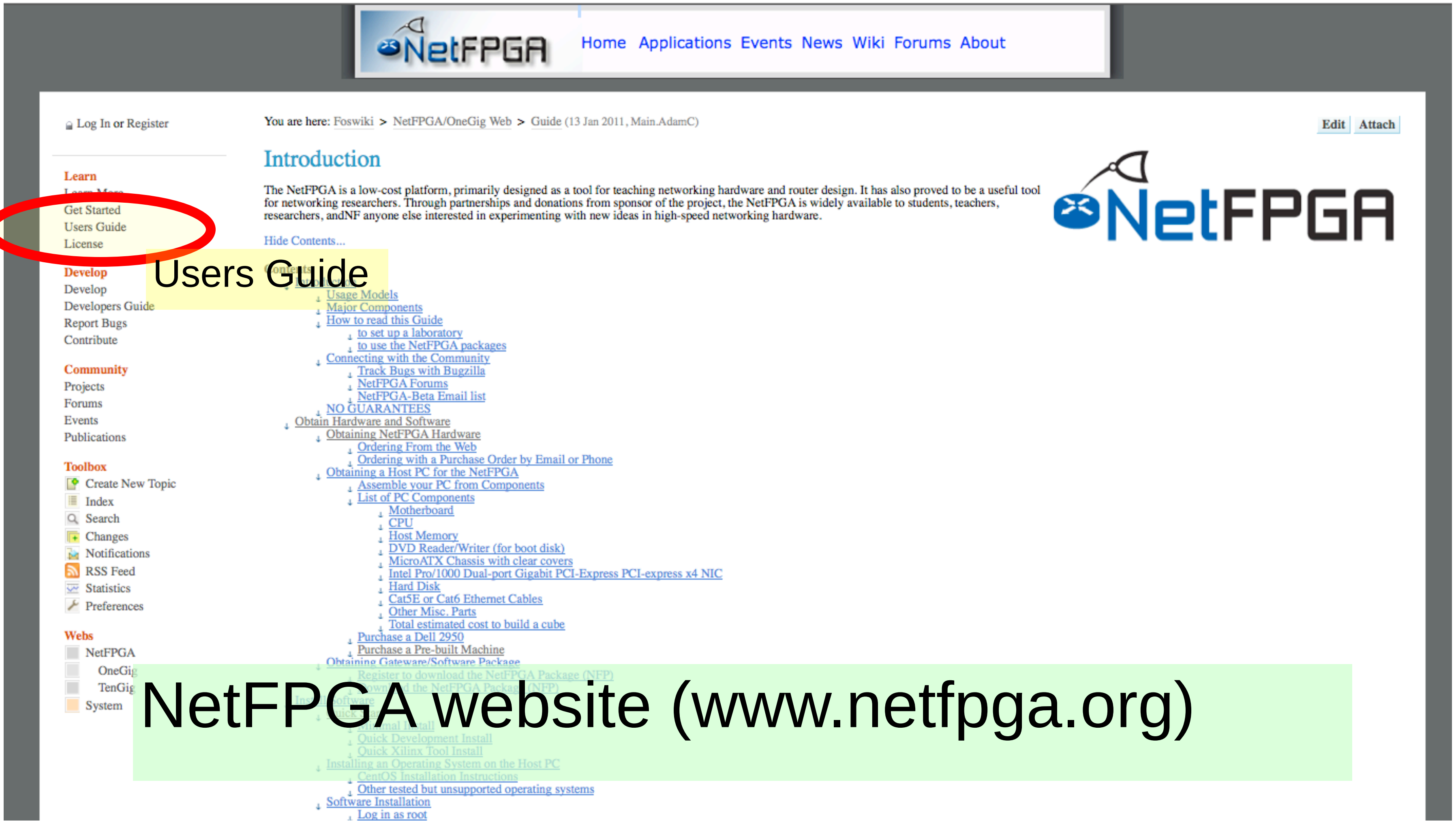
1. Get familiar with hardware description language

2. Prepare for your project

   a) Learn NetFPGA by yourself

   b) Get a hands-on tutorial

Learn by yourself



Home Applications Events News Wiki Forums About

Log In or Register

Learn
Learn More
Get Started
Users Guide
License

Develop
Develop
Developers Guide
Report Bugs
Contribute

Community
Projects
Forums
Events
Publications

Toolbox
Create New Topic
Index
Search
Changes
Notifications
RSS Feed
Statistics
Preferences

Webs
NetFPGA
OneGig
TenGig
System

You are here: Foswiki > NetFPGA/OneGig Web > Guide (13 Jan 2011, Main.AdamC)

Edit Attach

Introduction

The NetFPGA is a low-cost platform, primarily designed as a tool for teaching networking hardware and router design. It has also proved to be a useful tool for networking researchers. Through partnerships and donations from sponsor of the project, the NetFPGA is widely available to students, teachers, researchers, andNF anyone else interested in experimenting with new ideas in high-speed networking hardware.

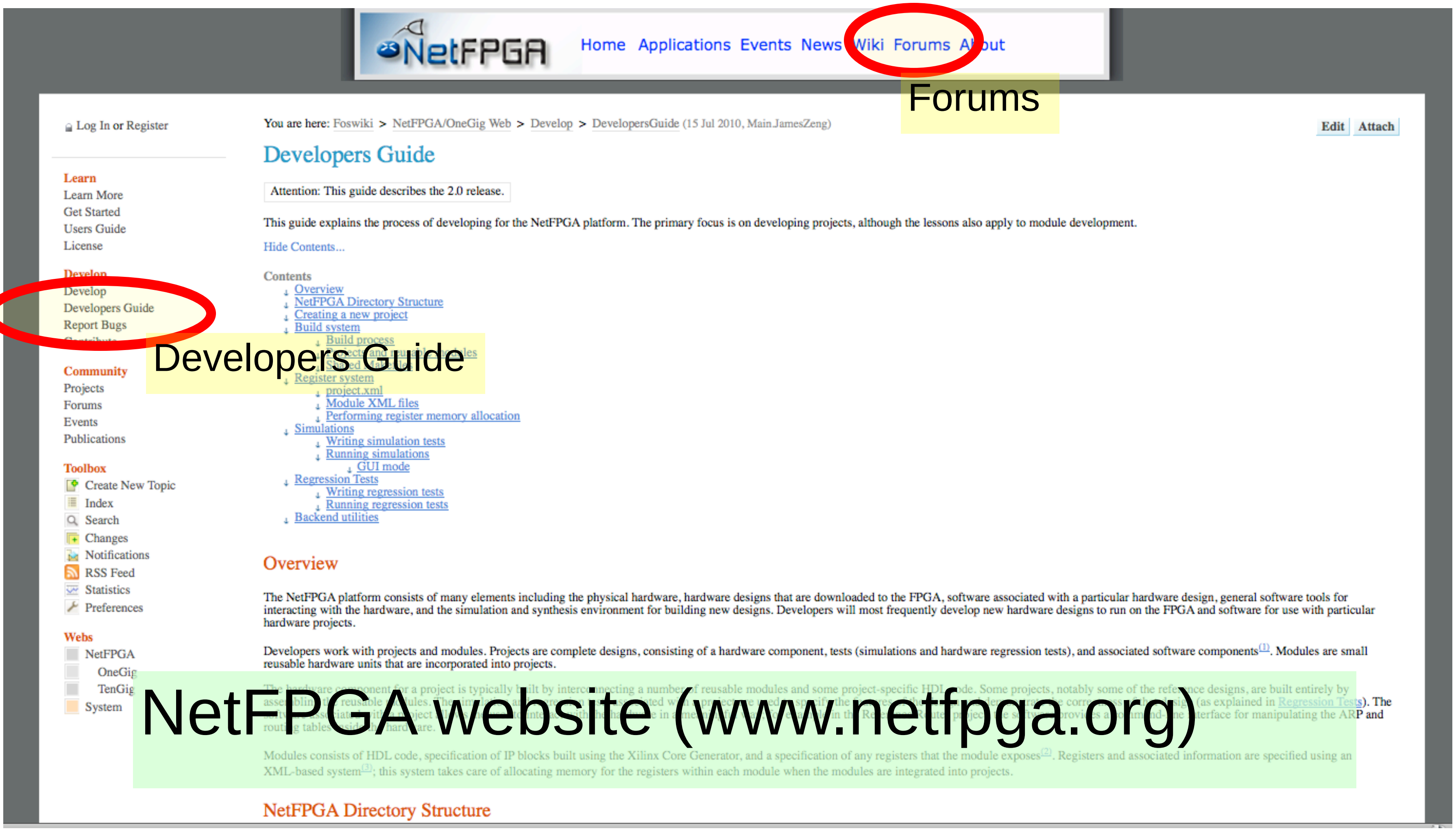
Hide Contents...

Contents

Users Guide

NetFPGA website (www.netfpga.org)

Learn by yourself



Forums

Developers Guide

You are here: Foswiki > NetFPGA/OneGig Web > Develop > DevelopersGuide (15 Jul 2010, Main.JamesZeng)

Developers Guide

Attention: This guide describes the 2.0 release.

This guide explains the process of developing for the NetFPGA platform. The primary focus is on developing projects, although the lessons also apply to module development.

Hide Contents...

Contents

Overview

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.

Developers work with projects and modules. Projects are complete designs, consisting of a hardware component, tests (simulations and hardware regression tests), and associated software components(1). Modules are small reusable hardware units that are incorporated into projects.

The hardware component for a project is typically built by interconnecting a number of reusable modules and some project-specific HDL code. Some projects, notably some of the reference designs, are built entirely by assembling [illegible] reusable modules. The simulation and regression [illegible] associated with [illegible] project [illegible] specify the [illegible] the design (as explained in Regression Tests). The software associated with a project [illegible] interact with the hardware in [illegible] in the Reference Router project the software provides a command-line interface for manipulating the ARP and routing tables inside the hardware.

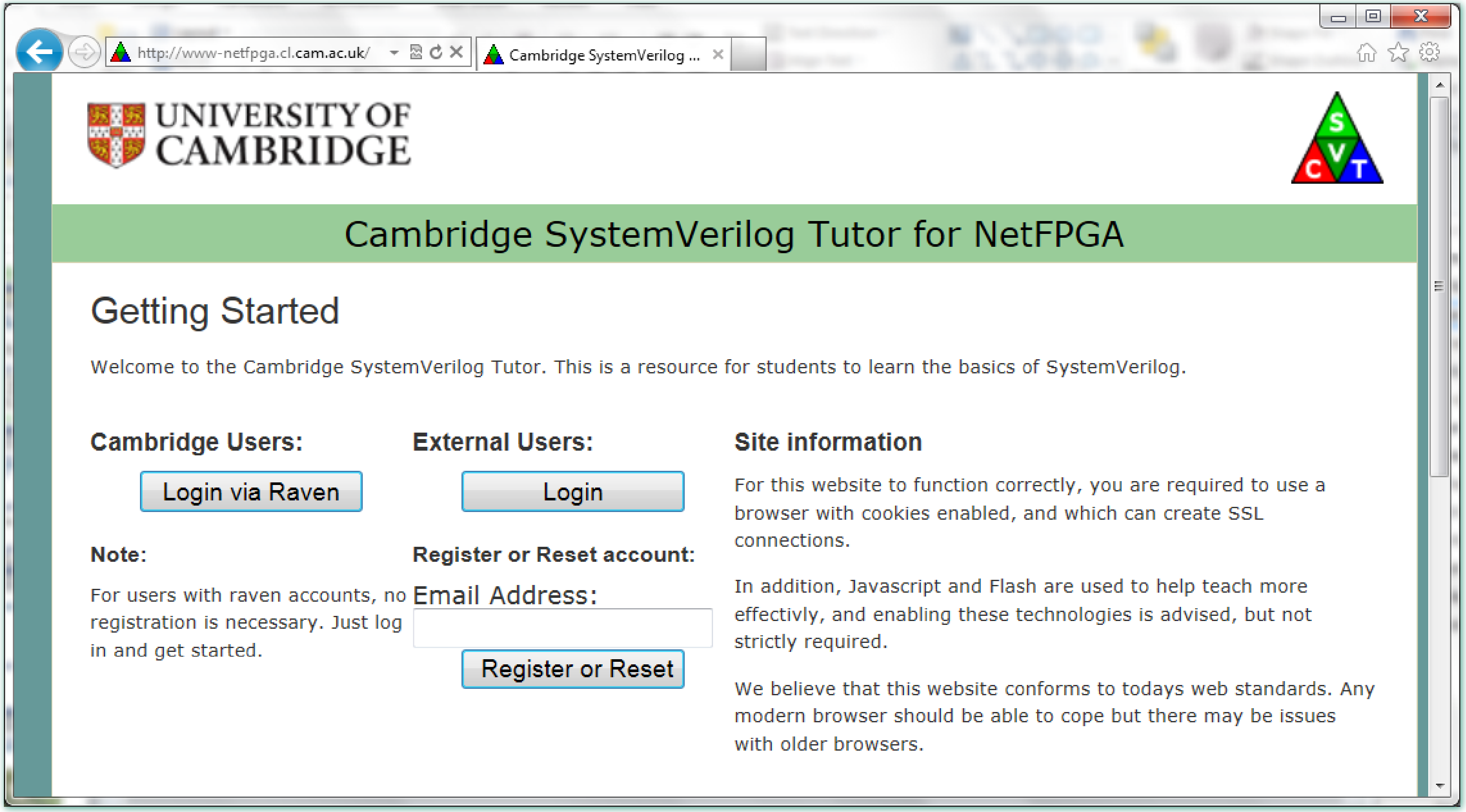
Modules consists of HDL code, specification of IP blocks built using the Xilinx Core Generator, and a specification of any registers that the module exposes(2). Registers and associated information are specified using an XML-based system(3); this system takes care of allocating memory for the registers within each module when the modules are integrated into projects.

NetFPGA Directory Structure

NetFPGA website (www.netfpga.org)

Learn by Yourself

Online tutor – coming soon!



Cambridge SystemVerilog Tutor for NetFPGA

Getting Started

Welcome to the Cambridge SystemVerilog Tutor. This is a resource for students to learn the basics of SystemVerilog.

Cambridge Users:

Login via Raven

Note:

For users with raven accounts, no registration is necessary. Just log in and get started.

External Users:

Login

Register or Reset account:

Email Address:

Register or Reset

Site information

For this website to function correctly, you are required to use a browser with cookies enabled, and which can create SSL connections.

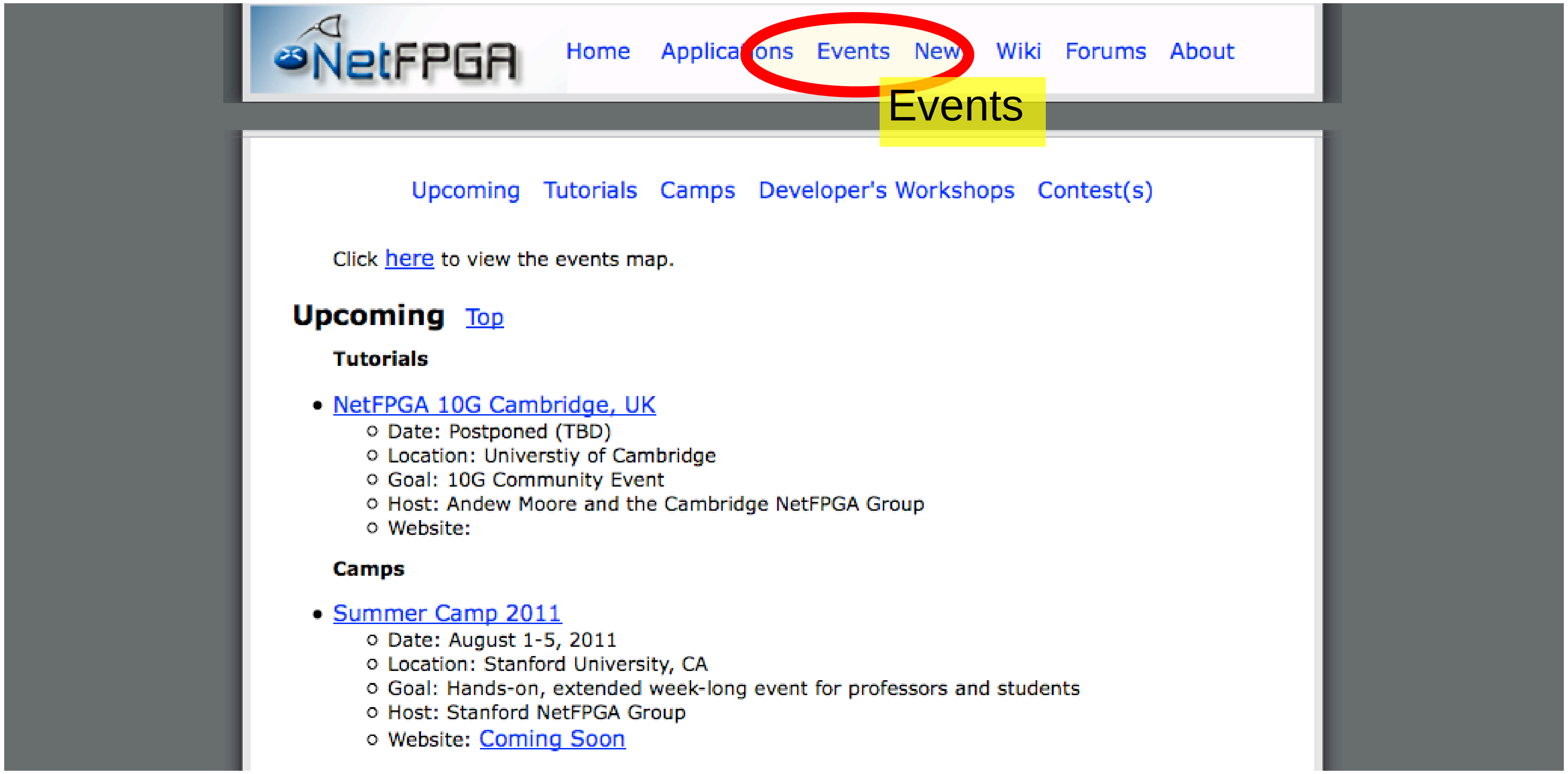
In addition, Javascript and Flash are used to help teach more effectivly, and enabling these technologies is advised, but not strictly required.

We believe that this website conforms to todays web standards. Any modern browser should be able to cope but there may be issues with older browsers.

Support for NetFPGA enhancements provided by redgate software

Get a hands-on tutorial

Get a hands-on tutorial



Click here to view the events map.

Upcoming Top

Tutorials

- NetFPGA 10G Cambridge, UK
  - Date: Postponed (TBD)
  - Location: Universtiy of Cambridge
  - Goal: 10G Community Event
  - Host: Andew Moore and the Cambridge NetFPGA Group
  - Website:

Camps

- Summer Camp 2011
  - Date: August 1-5, 2011
  - Location: Stanford University, CA
  - Goal: Hands-on, extended week-long event for professors and students
  - Host: Stanford NetFPGA Group
  - Website: Coming Soon

NetFPGA website (www.netfpga.org)

Section VII: Conclusion

Conclusions

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

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, Shahbaz Muhammad, David Miller, Martin Zadnik

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)

XILINX®

<u>Other NetFPGA Tutorial Presented At</u>:

HOT INTERCONNECTS

UNIVERSITY OF CAMBRIDGE

NICTA

SIGMETRICS

acm

VIA VERITAS VITA

UNIVERSITY of GLASGOW

CESNET

UNIVERSITAS MASARYKIANA BRUNENSIS

THE UNIVERSITY OF NEW SOUTH WALES

SIGCOMM 2008 SEATTLE

FIT

Indian Institute of Science
Bangalore, India

1909 BEIJING JIAOTONG UNIVERSITY

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

Thanks to our Sponsors:

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

<u>Disclaimer:</u> 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.