NetFPGA Summer Course

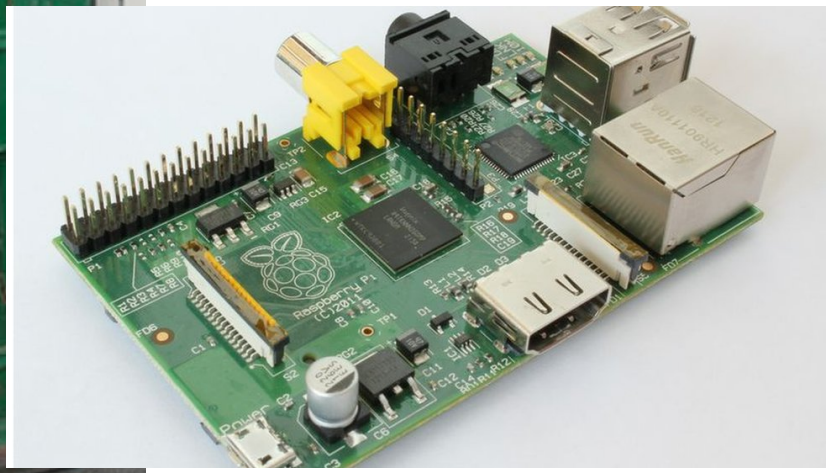
Presented by:

Andrew W Moore, Noa Zilberman, Gianni Antichi
Stephen Ibanez, Marcin Wojcik, Jong Hun Han,
Salvator Galea, Murali Ramanujam, Jingyun Zhang,
Yuta Tokusashi

University of Cambridge
July 24 – July 28, 2017

http://NetFPGA.org

Cambridge University Computer Lab



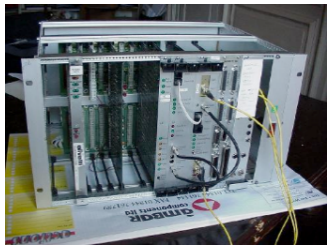
Cambridge? never heard of them

- But you may have heard of some of our more successful projects (some have changed name):

Autonomy alphamosaic VC ARM azuro TenisonEDA BBC MICRO Virata Level 5 networks ANSA Xen Source™

- And some of our not so successful projects:

ATM



Cambridge Backbone Ring
>1 Gb/s LAN/WAN in 1995

Sun's sunray

ATM - (we didn't want 48 byte payloads either – so very silly)

Course Outline

- **Combination of lectures and hands-on experience**
- **Day 1 –** Introduction and “High level” concepts.
  Goal: Be able to do “rapid prototyping”.
  Evening: Welcome dinner
- **Day 2 -** Complete high level usage aspects, Introduction to P4→NetFPGA and start projects.
- **Day 3 -** Projects development, Detailed Vivado aspects.
- **Day 4** – Continue projects development, detailed hardware aspects.
- **Day 5** – Applications of NetFPGA, Projects presentations

Tutors

Andrew Moore

- Introduction to NetFPGA

Gianni Antichi

- Timing & Debug
- Applications

Marcin Wojcik

- DMA

Noa Zilberman

- Architecture
- Design Flow
- Test Infrastructure
- IP Cores

Stephen Ibanez

- P4→NetFPGA

UNIVERSITY OF CAMBRIDGE

Tutors

Andrew Moore

- Introduction to NetFPGA

Gianni Antichi

- Timing & Debug
- Applications

Mystery Guest TBA

~~Marcin Wojcik~~

- DMA

Noa Zilberman

- Architecture
- Design Flow
- Test Infrastructure
- IP Cores

Stephen Ibanez

- P4→NetFPGA

UNIVERSITY OF CAMBRIDGE

Acknowledgments (I)

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

Andrew Moore, David Miller, Muhammad Shahbaz, Martin Zadnik, Matthew Grosvenor, Yury Audzevich, Neelakandan Manihatty-Bojan, Georgina Kalogeridou, Jong Hun Han, Noa Zilberman, Gianni Antichi, Charalampos Rotsos, Hwanju Kim, Marco Forconesi, Jinyun Zhang, Bjoern Zeeb, Robert Watson, Salvator Galea, Marcin Wojcik, Diana Andreea Popescu, Murali Ramanujam

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, Stephen Gabriel Ibanez

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, Lisa Donatini, Sergio Lopez-Buedo , Andreas Fiessler, Robert Soule, Pietro Bressana, Yuta Tokusashi

Steve Wang, Erik Cengar, Michael Alexander, Sam Bobrowicz, Garrett Aufdemberg, Patrick Kane, Tom Weldon

Patrick Lysaght, Kees Vissers, Michaela Blott, Shep Siegel, Cathal McCabe

Acknowledgements (II)


UNIVERSITY OF CAMBRIDGE

EPSRC
Pioneering research and skills

XILINX

Micron

The Leverhulme Trust

Isaac Newton Trust

Google

Atomic Rules

ALGO-LOGIC

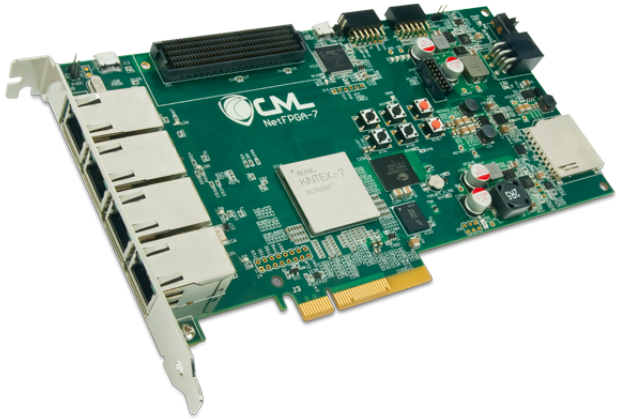
Day 1 Outline

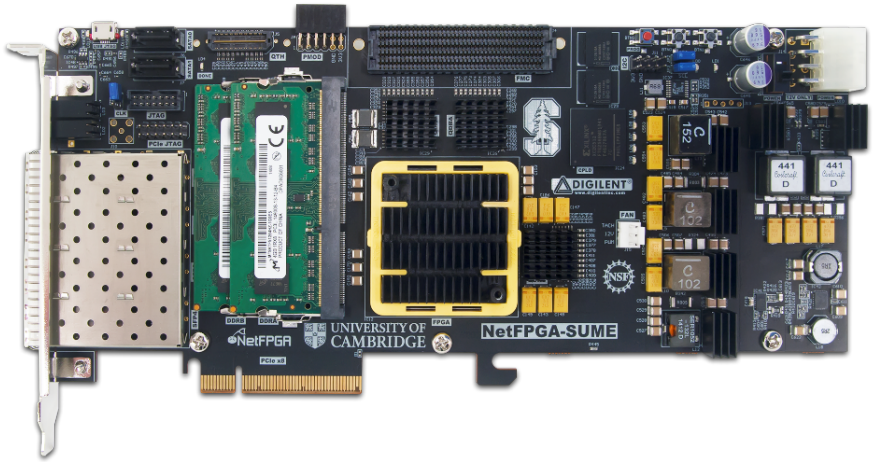
- **The NetFPGA platform**
  - Introduction
  - Overview of the NetFPGA Platform
- **NetFPGA SUME**
  - Hardware overview
- **Network Review**
  - Basic IP review
- **The Base Reference Switch**
  - Example I: Reference Switch running on the NetFPGA
- **The Life of a Packet Through the NetFPGA**
  - Hardware Datapath
  - Interface to software: Exceptions and Host I/O
- **Infrastructure**
  - Tree
  - Verification Infrastructure
- **Examples of Using NetFPGA**
- **Example Project: Crypto Switch**
  - Introduction to a Crypto Switch
  - What is an IP core?
  - Getting started with a new project.
  - Crypto FSM
- **Simulation and Debug**
  - Write and Run Simulations for Crypto Switch
- **Concluding Remarks**

Section I: The NetFPGA platform

NetFPGA = Networked FPGA

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



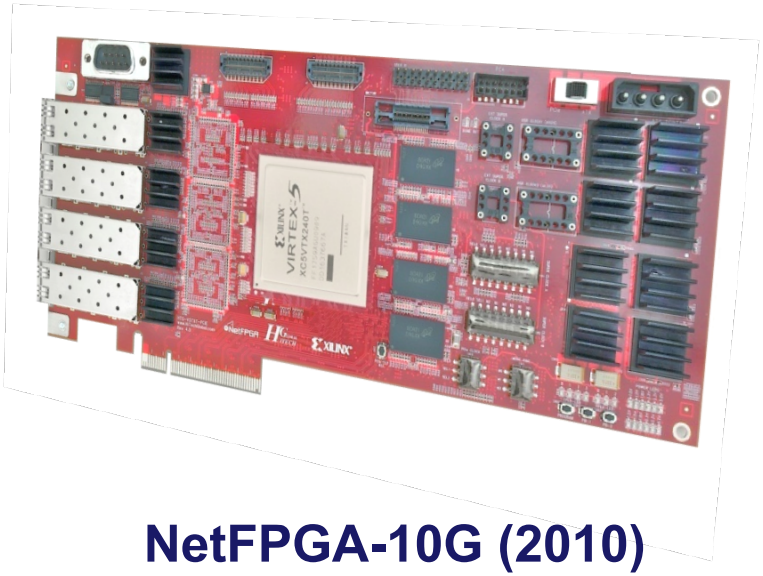


=

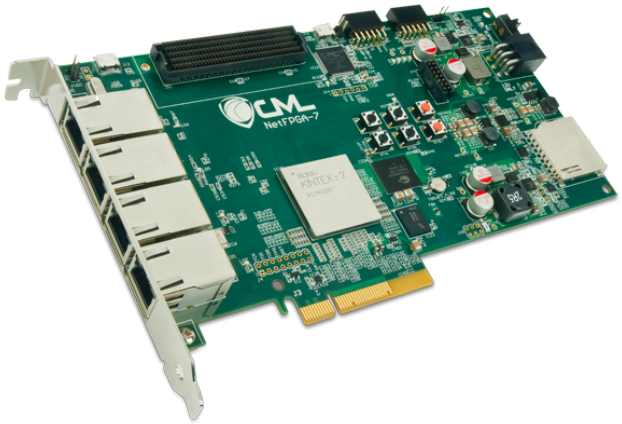
- Network Interface Card
- Hardware Accelerated Linux Router
- IPv4 Reference Router
- Traffic Generator
- Openflow Switch
- More Projects
- Add Your Project

NetFPGA Family of Boards

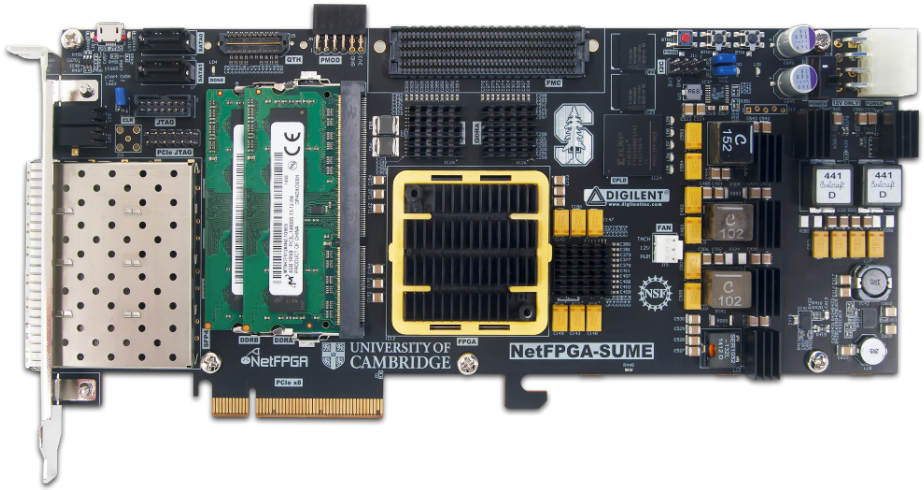
NetFPGA-1G (2006)



NetFPGA-10G (2010)



NetFPGA-1G-CML (2014)

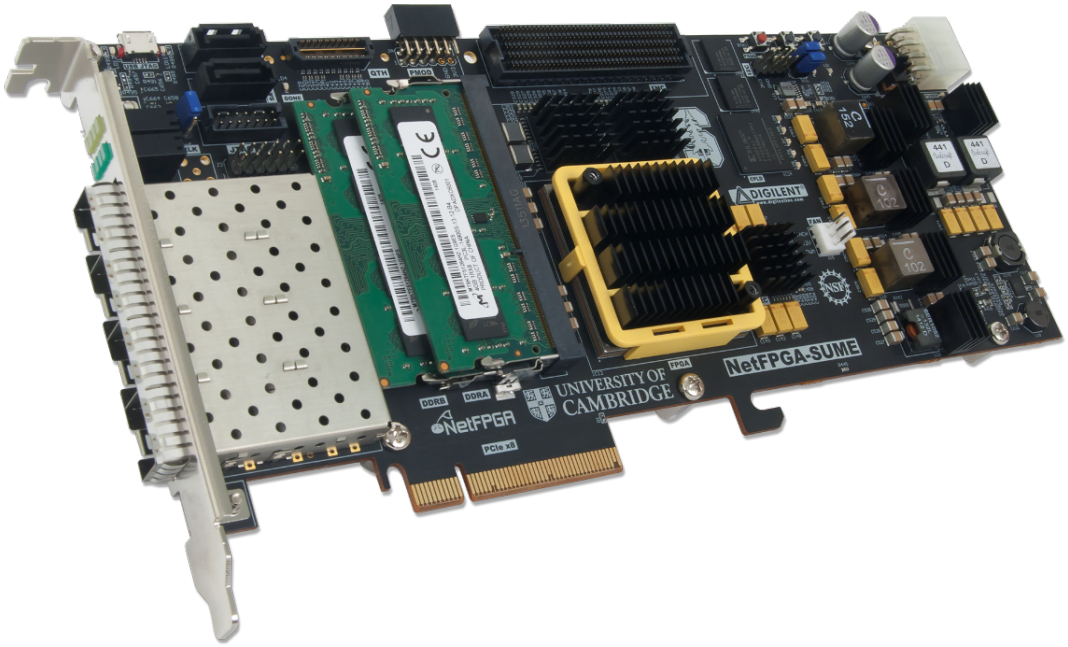


NetFPGA SUME (2014)

NetFPGA consists of…

Four elements:

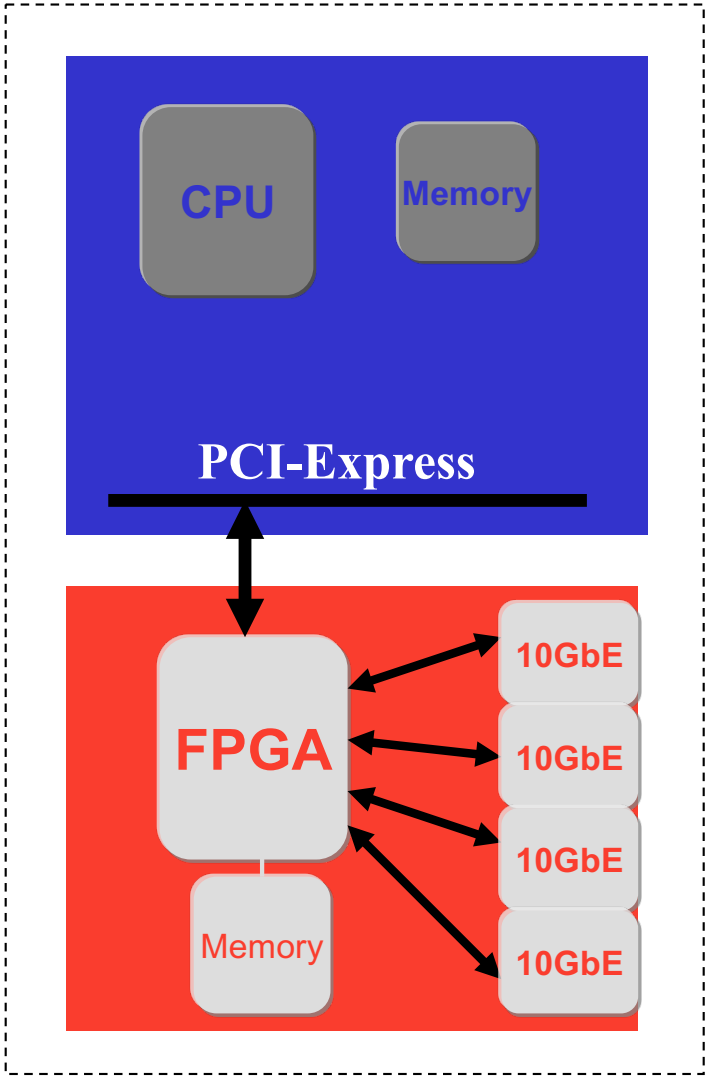
- NetFPGA board
- Tools + reference designs
- Contributed projects
- Community



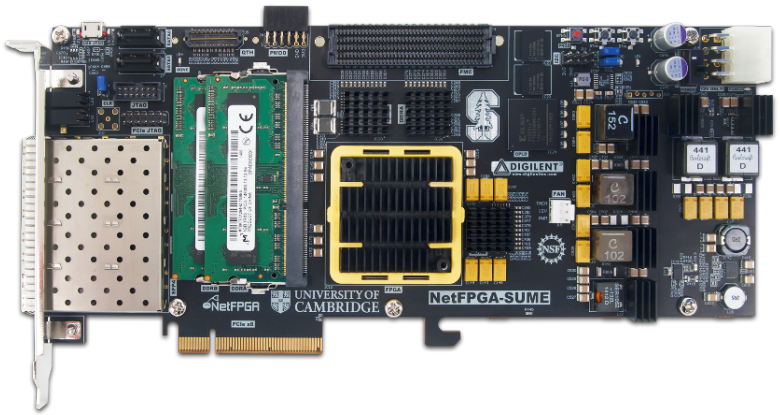
NetFPGA board

Networking Software running on a standard PC

A hardware accelerator built with Field Programmable Gate Array driving 1/10/ 100Gb/s network links



PC with NetFPGA



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)**

Community

Wiki

- **Documentation**
  - User's Guide *"so you just got your first NetFPGA"*
  - Developer's Guide *"so you want to build a …"*
- **Encourage users to contribute**

Forums

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

International Community

Over 1,200 users, using over 3500 cards at 200 universities in over 47 countries

NetFPGA SUME Community (*since Feb 2015*)

Over 600 users, using over 300 cards at
200 universities in 47 countries

NetFPGA's Defining Characteristics

- **Line-Rate**
  - Processes back-to-back packets
    - Without dropping packets
    - At full rate
  - 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 for SUME, LGPL 2.1 for 10G
  - But harder, because
    - Hardware modules must meet 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 to handle 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?

- Researchers
- Teachers
- Students

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

Summer Course Objectives

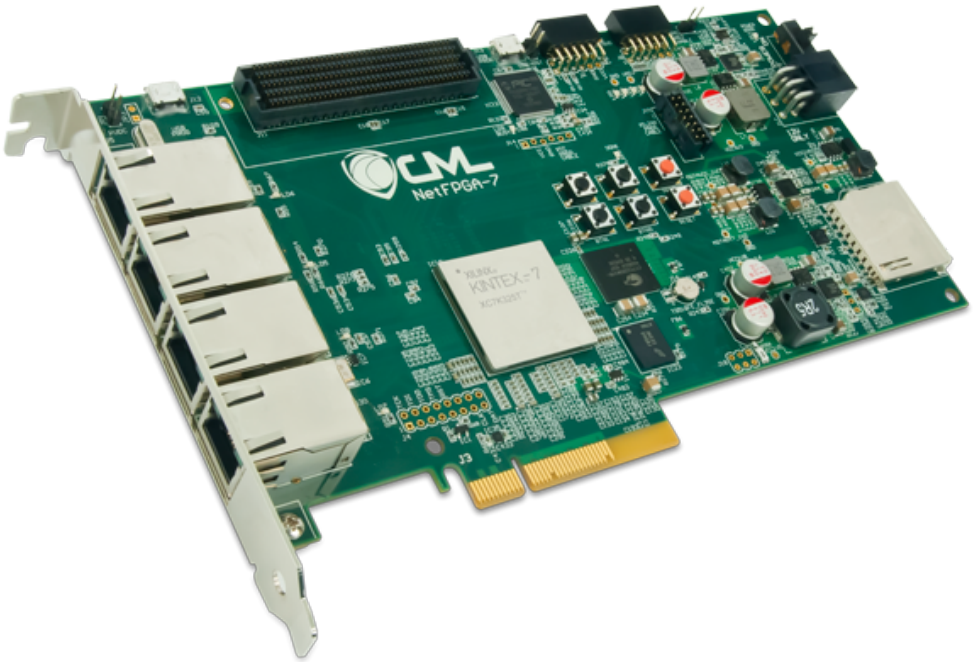
- **Overall picture of NetFPGA**
- **How reference designs work**
- **How you can work on a project**
  - NetFPGA Design Flow
  - Directory Structure, library modules and projects
  - How to utilize contributed projects
    - Interface/Registers
  - How to verify a design (Simulation and Hardware Tests)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

AND… You build your own projects!

Section II: Hardware Overview

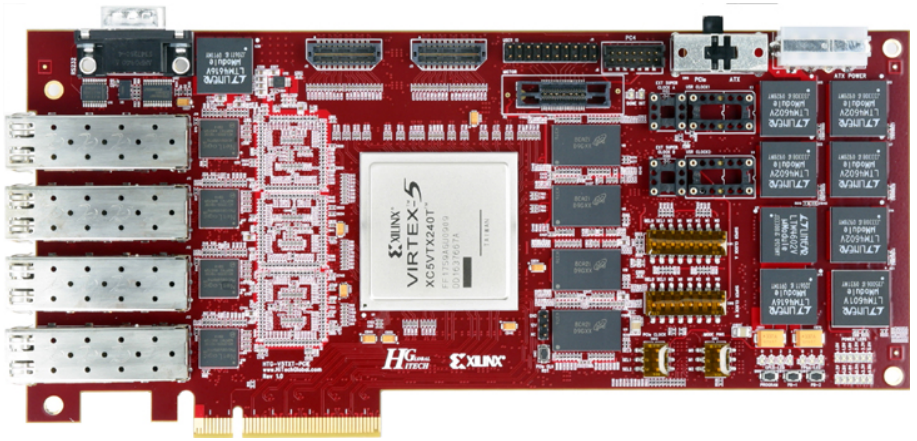
NetFPGA-1G-CML

- FPGA Xilinx Kintex7
- 4x 10/100/1000 Ports
- PCIe Gen.2 x4
- QDRII+-SRAM, 4.5MB
- DDR3, 512MB
- SD Card
- Expansion Slot



NetFPGA-10G

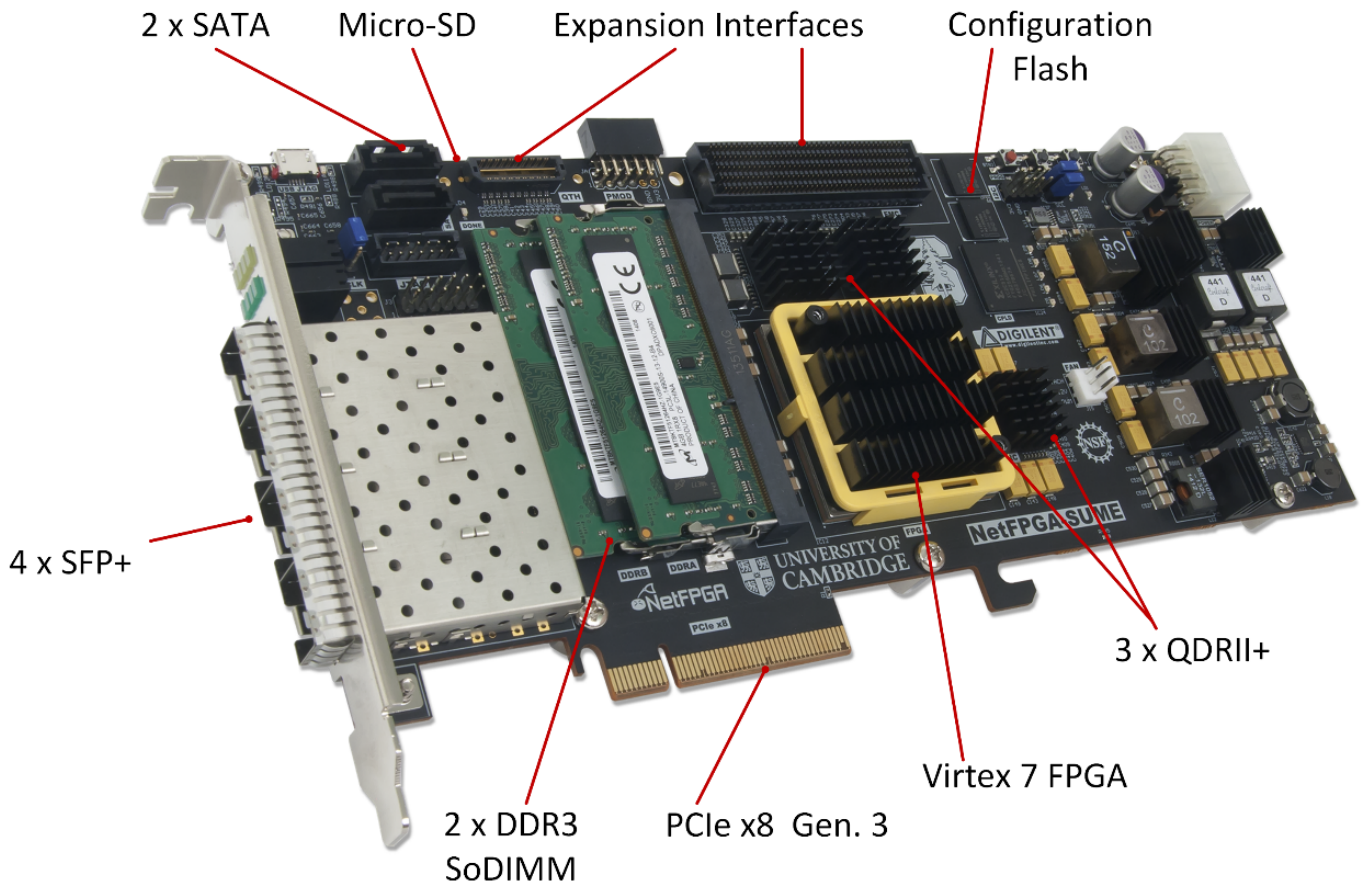
- **FPGA Xilinx Virtex5**
- **4 SFP+ Cages**
  - 10G Support
  - 1G Support
- **PCIe Gen.1 x8**
- **QDRII-SRAM, 27MB**
- **RLDRAM-II, 288MB**
- **Expansion Slot**



Time for a catch-up…

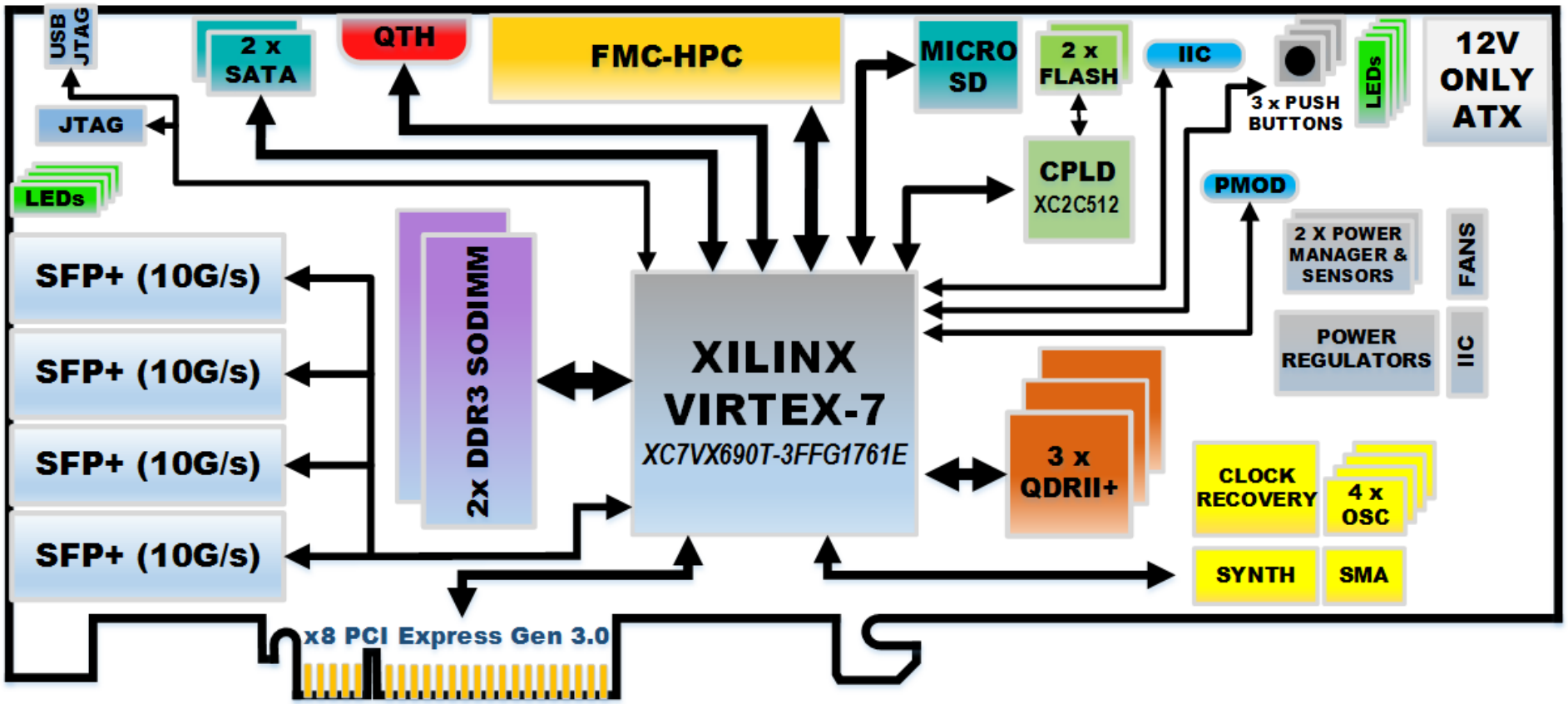
NetFPGA-SUME

- **A major upgrade over the NetFPGA-10G predecessor**



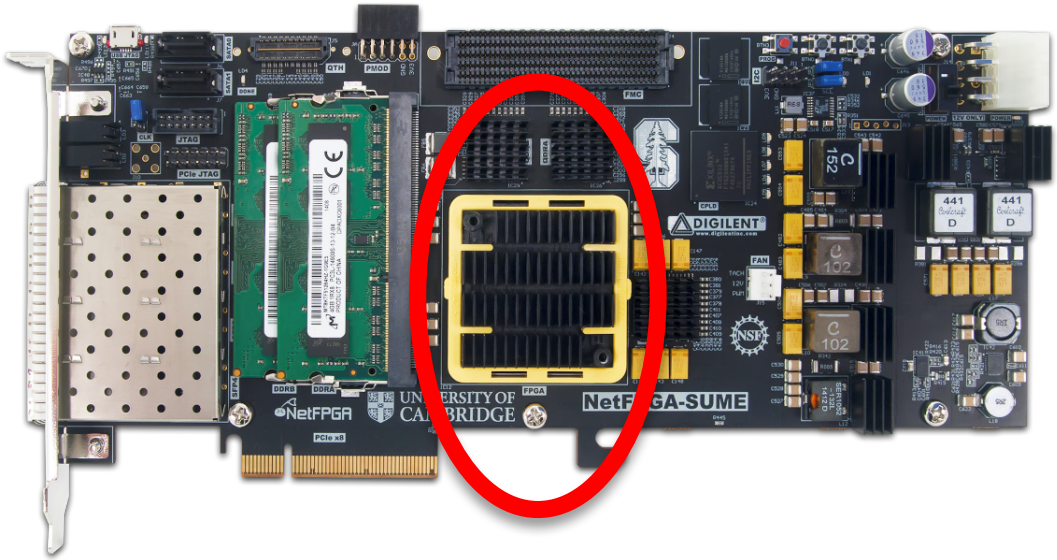
NetFPGA-SUME

- High Level Block Diagram



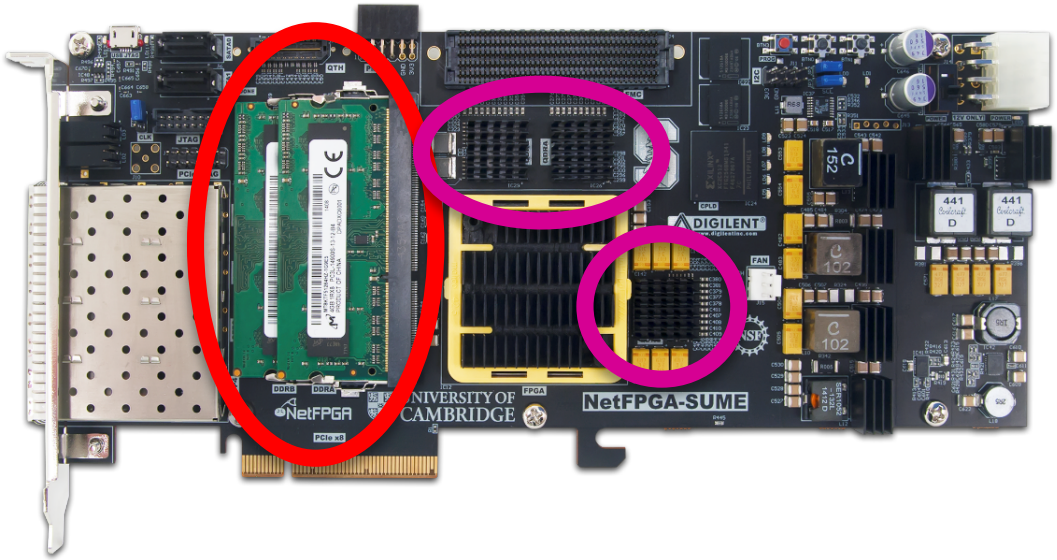
Xilinx Virtex 7 690T

- Optimized for high-performance applications
- 690K Logic Cells
- 52Mb RAM
- 3 PCIe Gen. 3 Hard cores



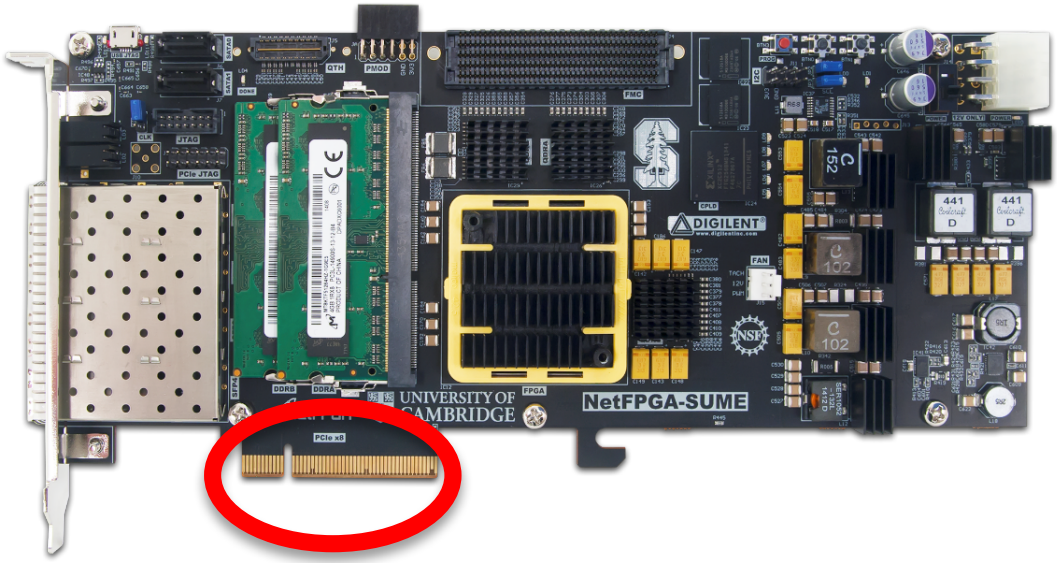
Memory Interfaces

- **DRAM:**
  2 x DDR3 SoDIMM
  1866MT/s, 4GB
- **SRAM:**
  3 x 9MB QDRII+,
  500MHz



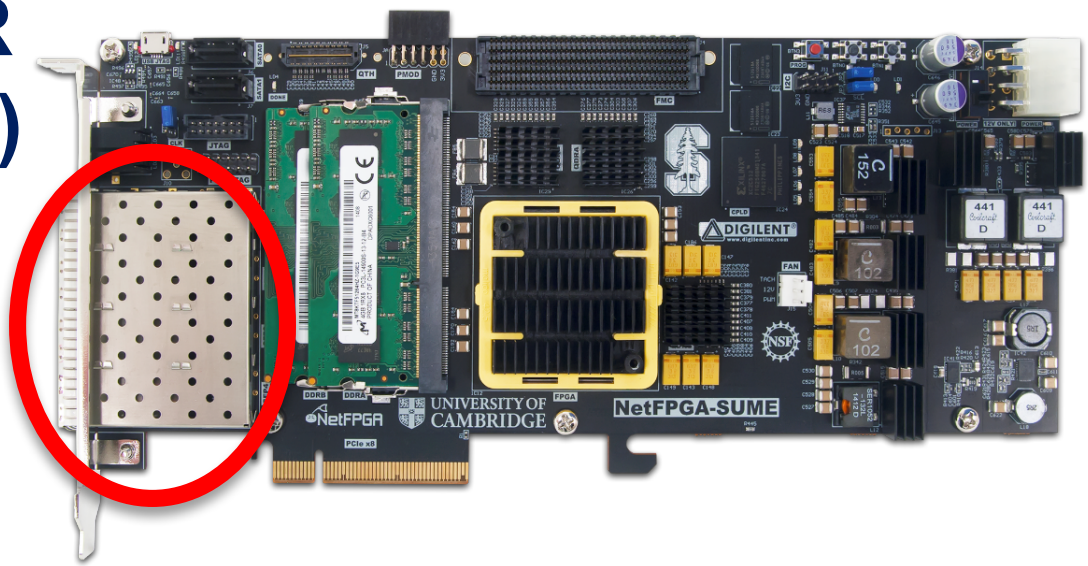
Host Interface

- PCIe Gen. 3
- x8 (only)
- Hardcore IP



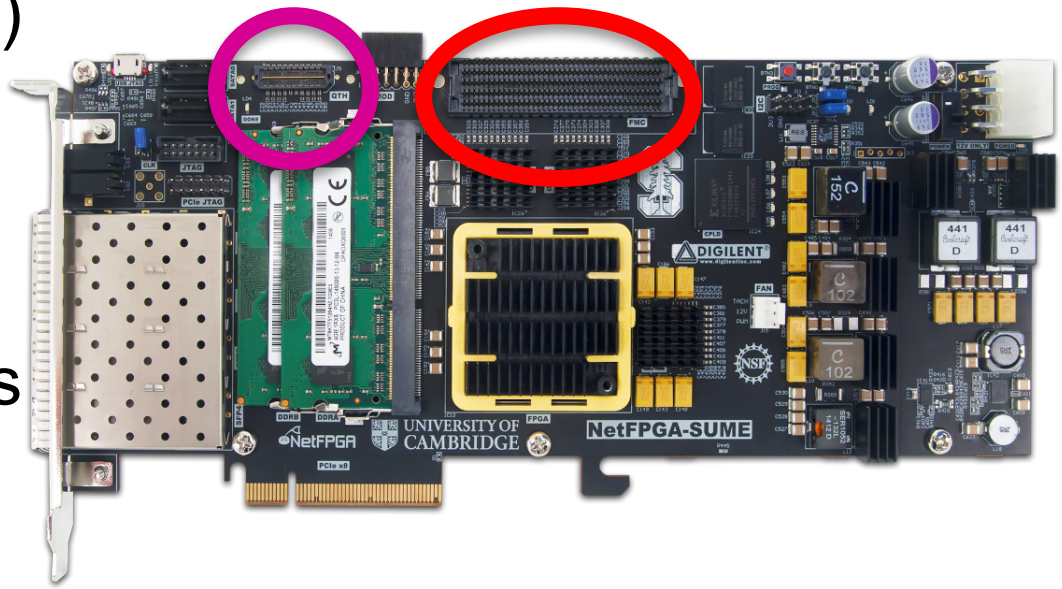
Front Panel Ports

- 4 SFP+ Cages
- Directly connected to the FPGA
- Supports 10GBase-R transceivers (default)
- Also Supports 1000Base-X transceivers and direct attach cables



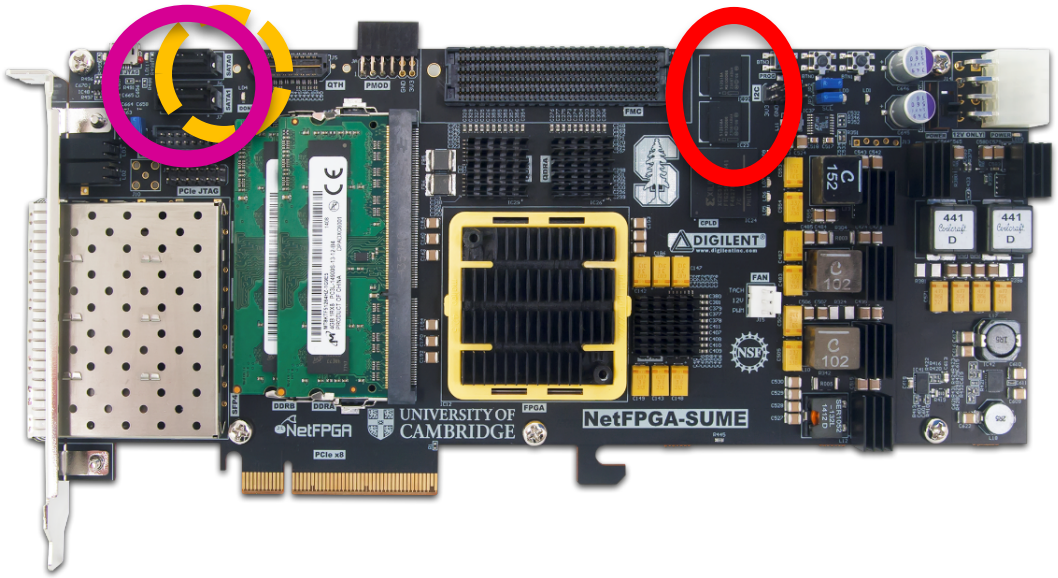
Expansion Interfaces

- **FMC HPC connector**
  - VITA-57 Standard
  - Supports Fabric Mezzanine Cards (FMC)
  - 10 x 12.5Gbps serial links
- **QTH-DP**
  - 8 x 12.5Gbps serial links

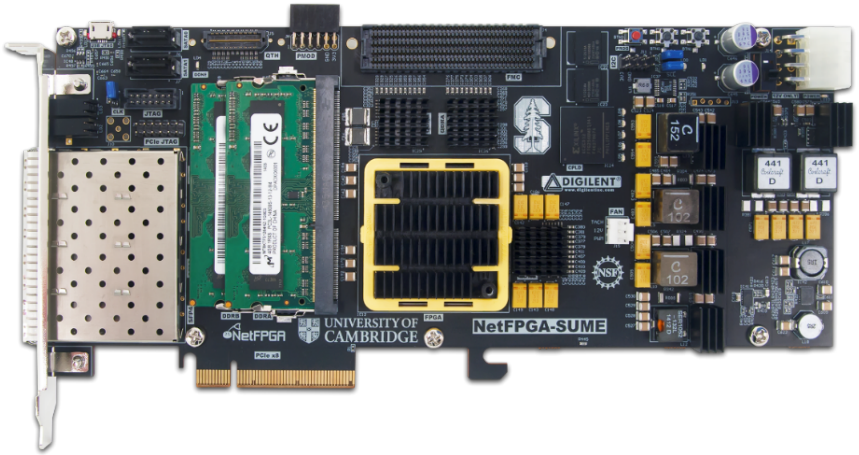


Storage

- 128MB FLASH
- 2 x SATA connectors
- Micro-SD slot
- Enable standalone operation

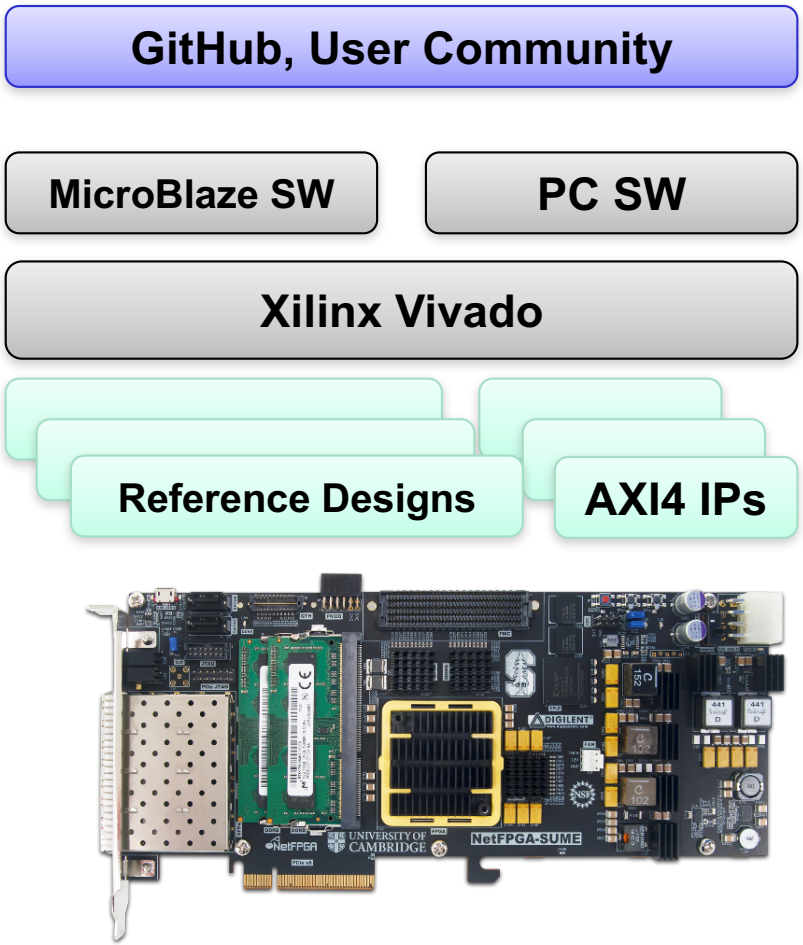


NetFPGA Board Comparison



NetFPGA SUME	NetFPGA 10G
Virtex 7 690T -3	Virtex 5 TX240T
8 GB DDR3 SoDIMM 1800MT/s	288 MB RLDRAM-II 800MT/s
27 MB QDRII+ SRAM, 500MHz	27 MB QDRII-SRAM, 300MHz
x8 PCI Express Gen. 3	x8 PCI Express Gen. 1
4 x 10Gbps Ethernet Ports	4 x 10Gbps Ethernet Ports
18 x 13.1Gb/s additional serial links	20 x 6.25Gb/s additional serial links

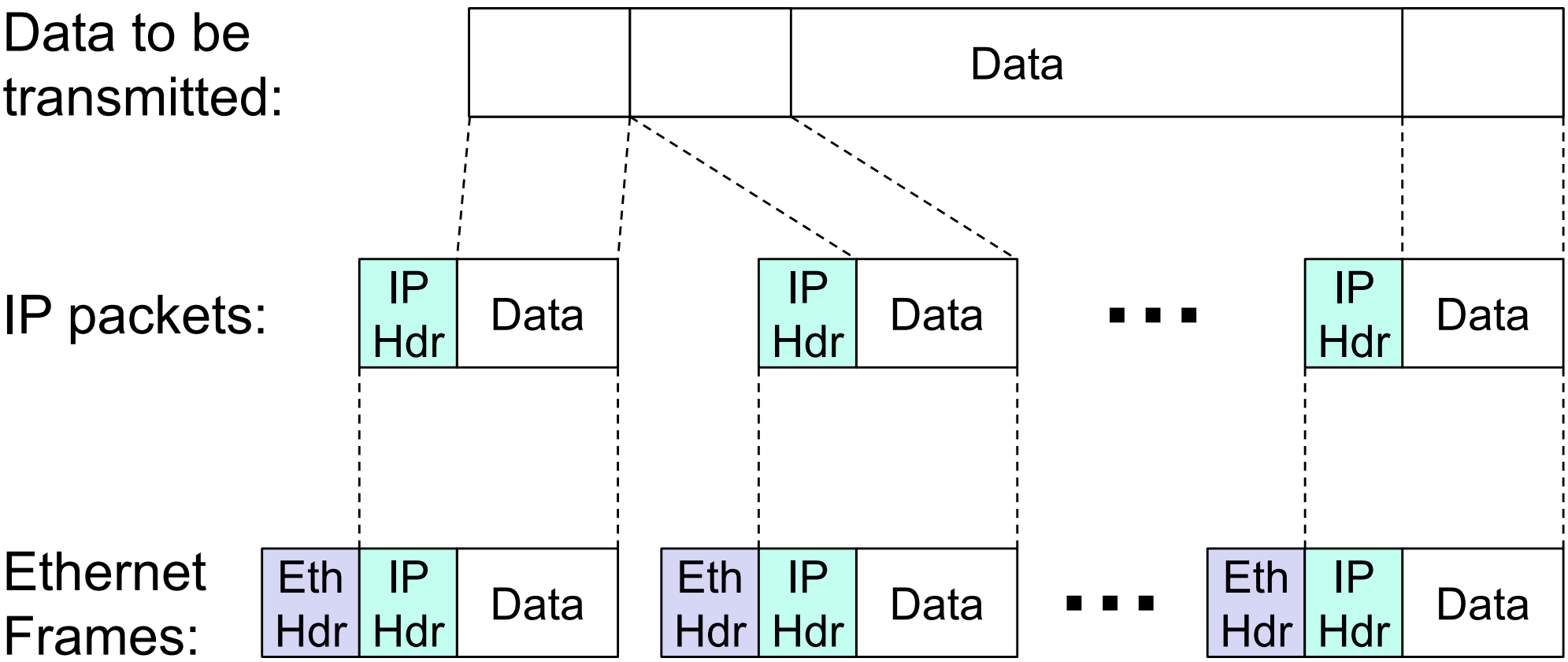
Beyond Hardware



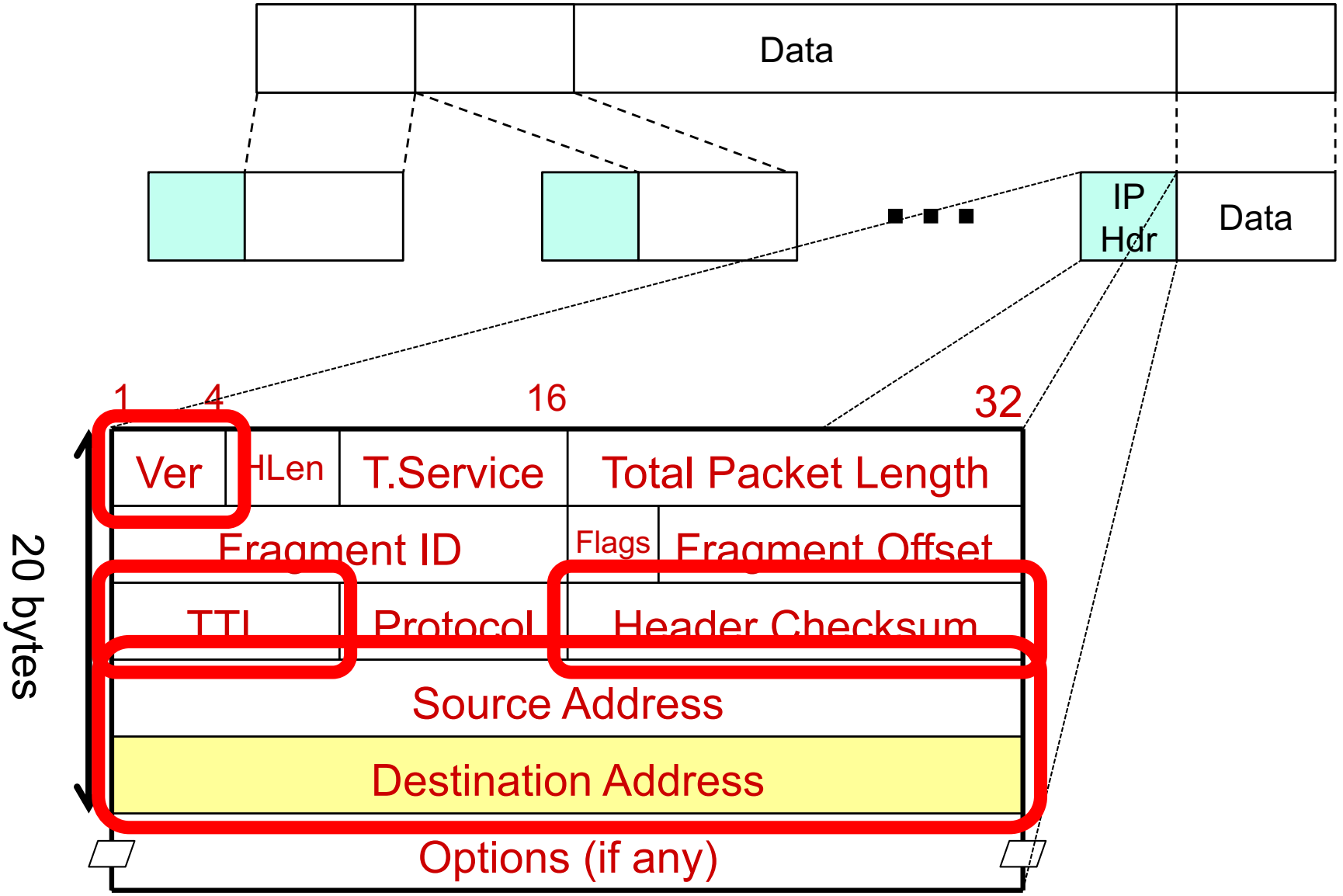
- NetFPGA Board
- Xilinx Vivado based IDE
- Reference designs using AXI4
- Software (embedded and PC)
- Public Repository
- Public Wiki

Section II: Network review

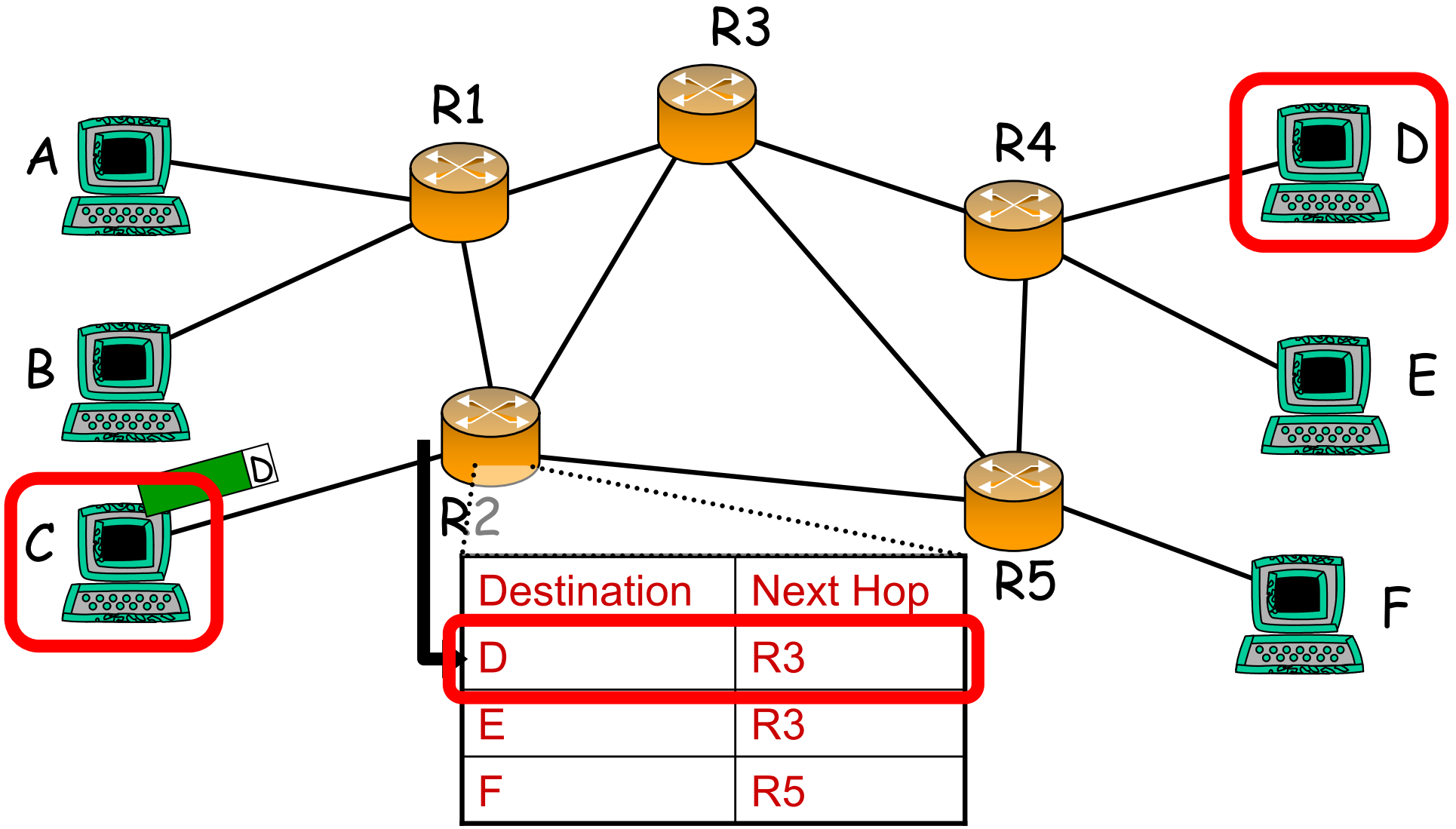
Internet Protocol (IP)



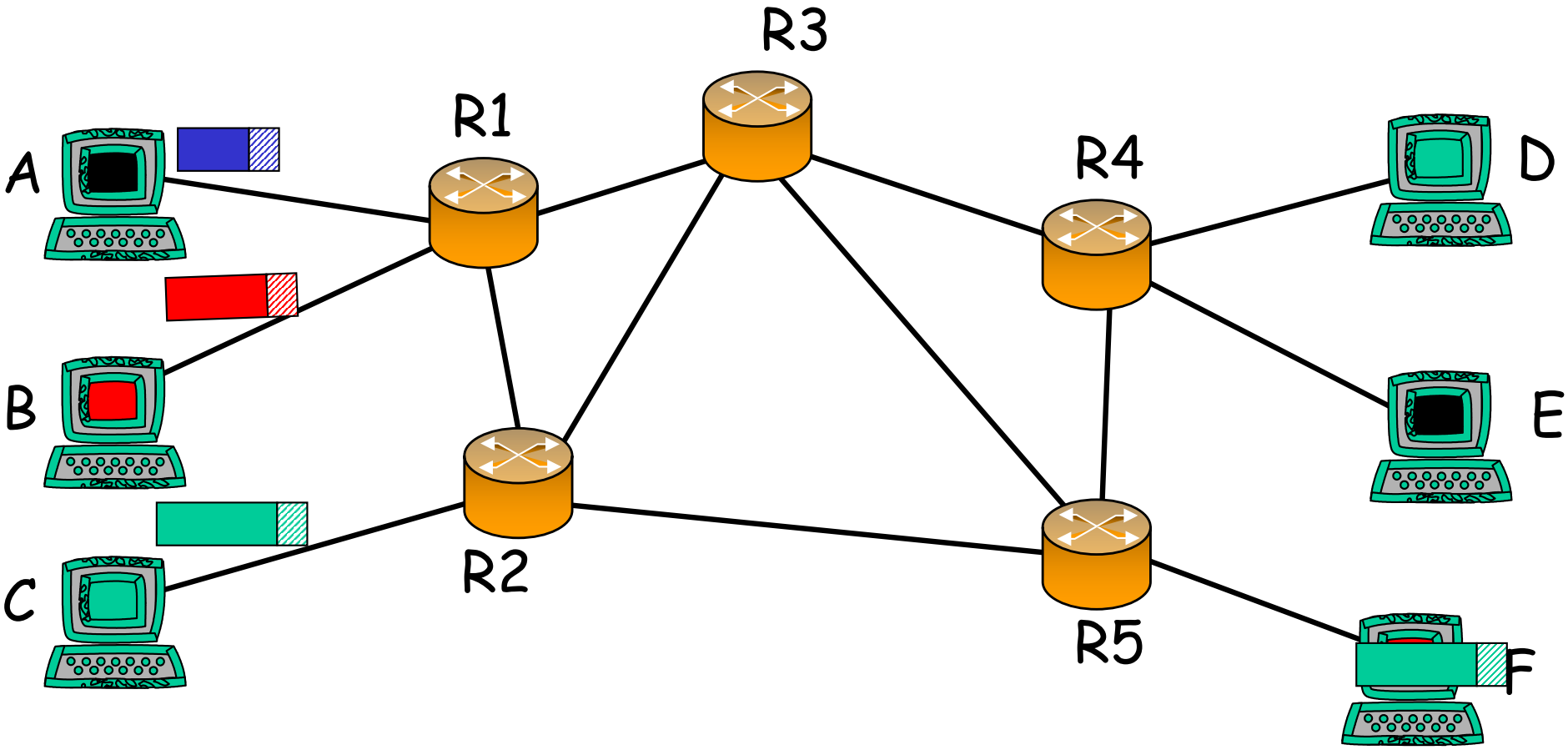
Internet Protocol (IP)



Basic operation of an IP router



Basic operation of an IP router



Forwarding tables

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

Naïve approach:

One entry per address

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

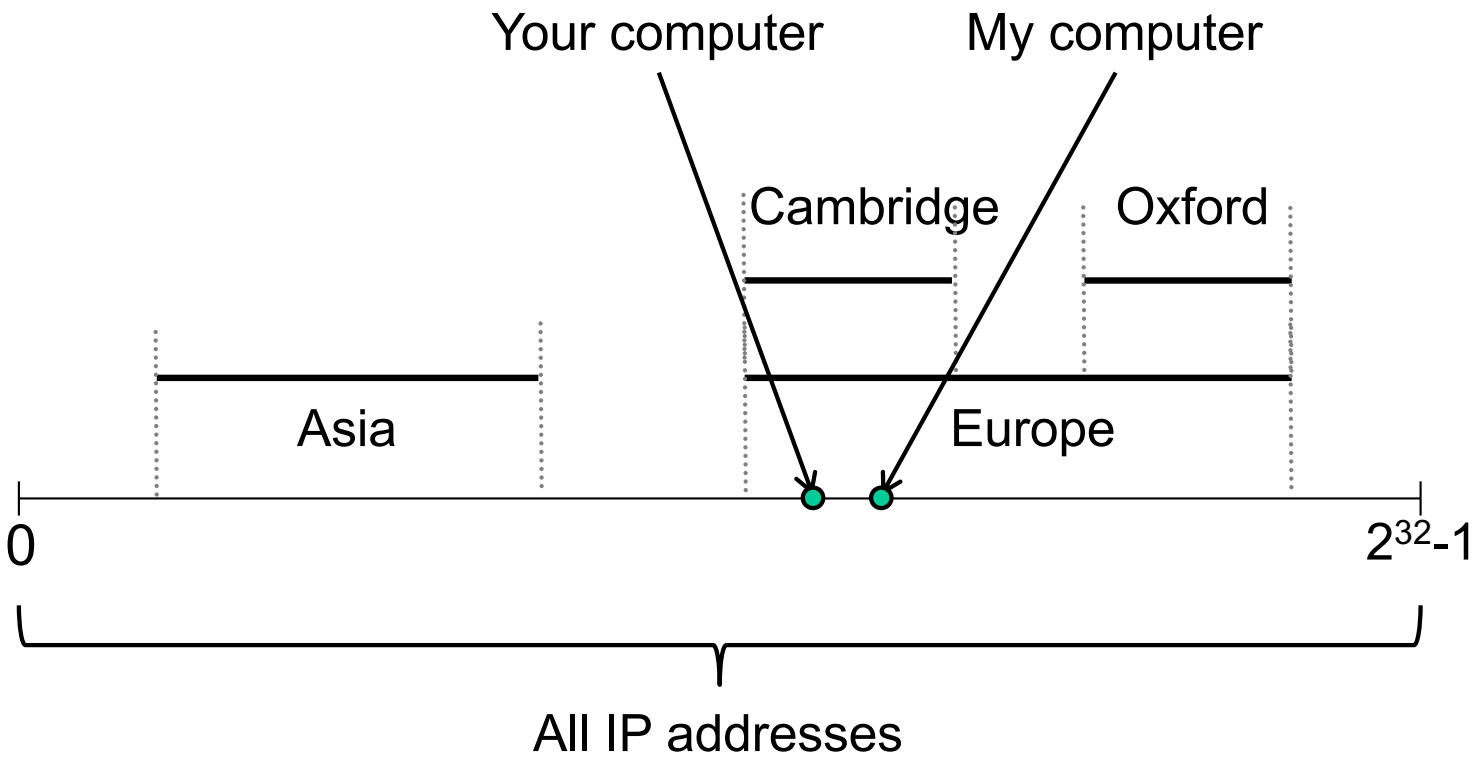
} **~ 4 billion entries**

Improved approach:

Group entries to reduce table size

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

IP addresses as a line



Entry	Destination	Port
1	Cambridge	1
2	Oxford	2
3	Europe	3
4	Asia	4
5	Everywhere (default)	5

Longest Prefix Match (LPM)

Entry	Destination	Port
1	Cambridge	1
2	Oxford	2
3	Europe	3
4	Asia	4
5	Everywhere (default)	5

Universities

Continents

Planet

Matching entries:

- Cambridge — Most specific
- Europe
- Everywhere

To: Cambridge	Data

Longest Prefix Match (LPM)

Entry	Destination	Port
1	Cambridge	1
2	Oxford	2
3	Europe	3
4	Asia	4
5	Everywhere (default)	5

Universities

Continents

Planet

Matching entries:

- Europe — Most specific
- Everywhere

To: Germany	Data

Implementing Longest Prefix Match

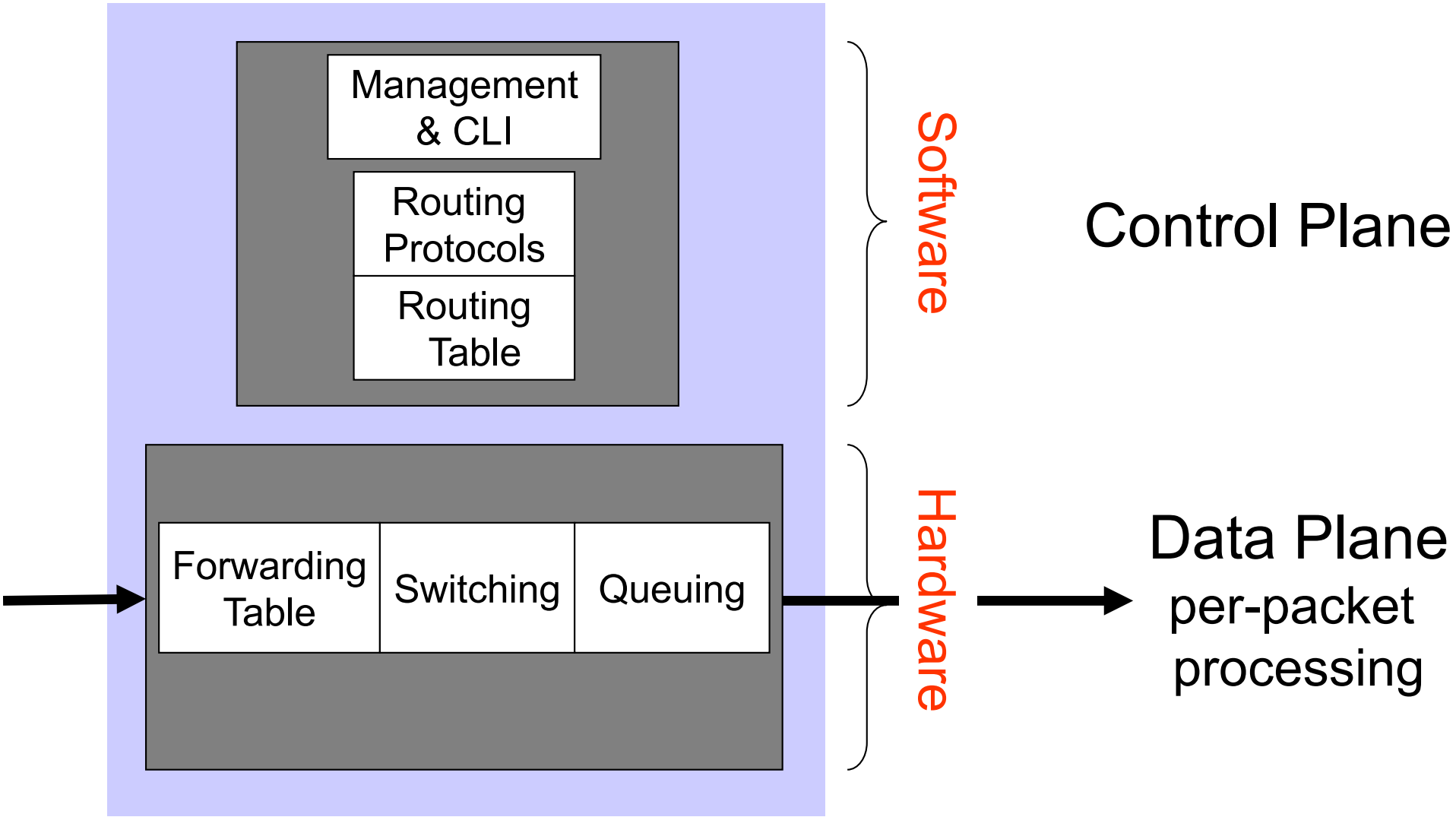
Entry	Destination	Port	
1	Cambridge	1	Searching
2	Oxford	2	
3	Europe	3	
4	Asia	4	FOUND
5	Everywhere (default)	5	

Most specific

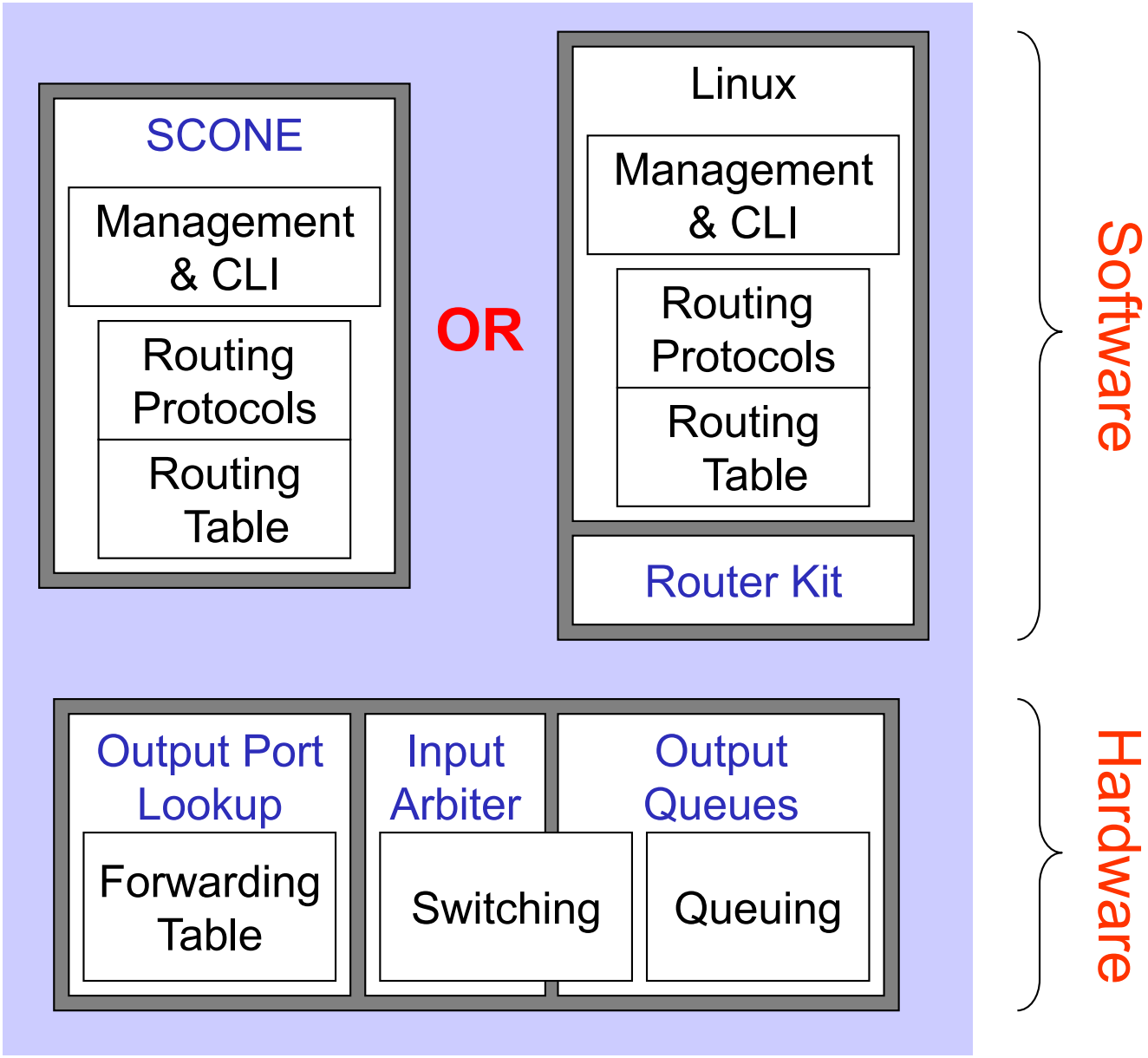
↓

Least specific

Basic components of an IP router

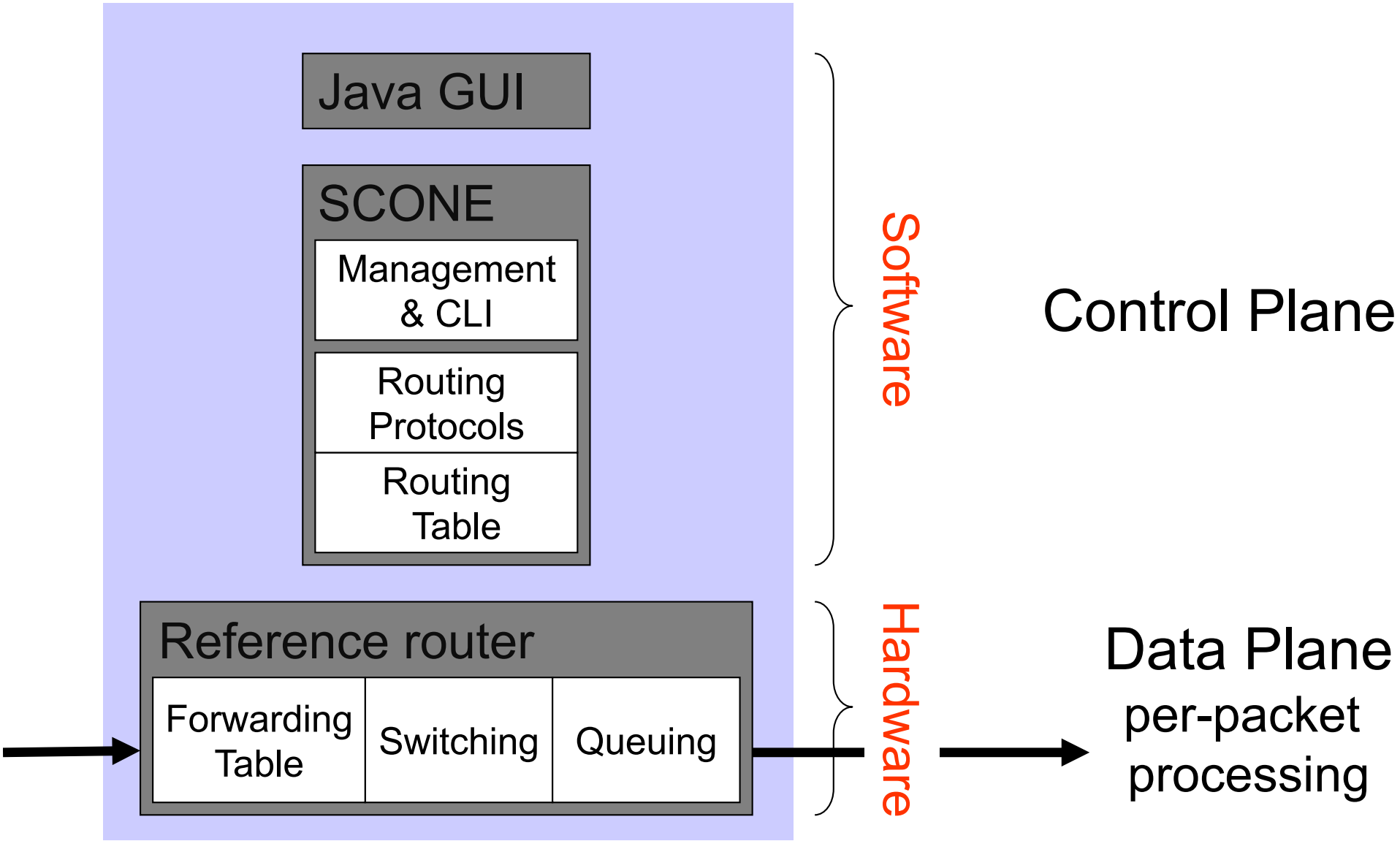


IP router components in NetFPGA

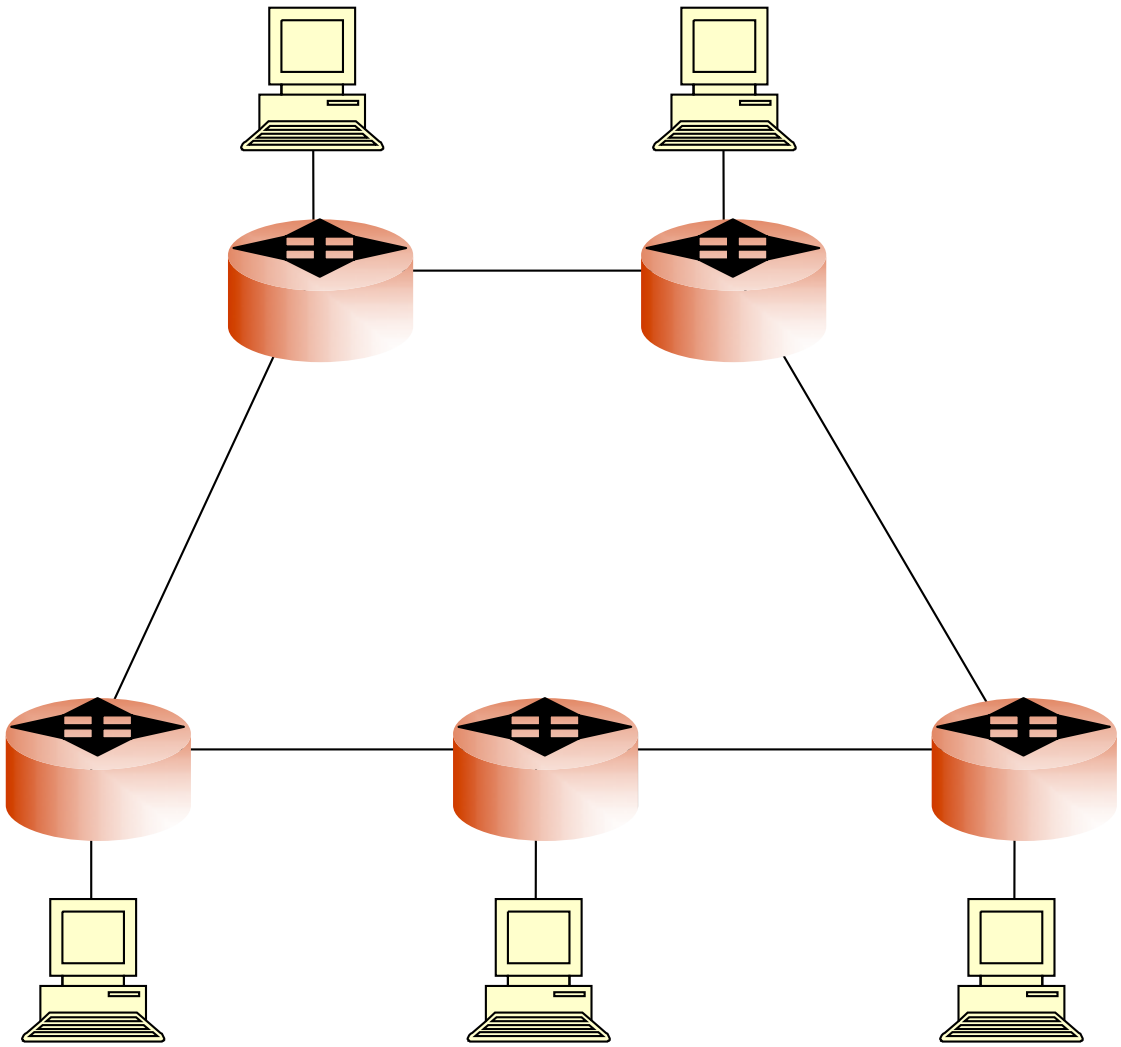


Section III: Example I

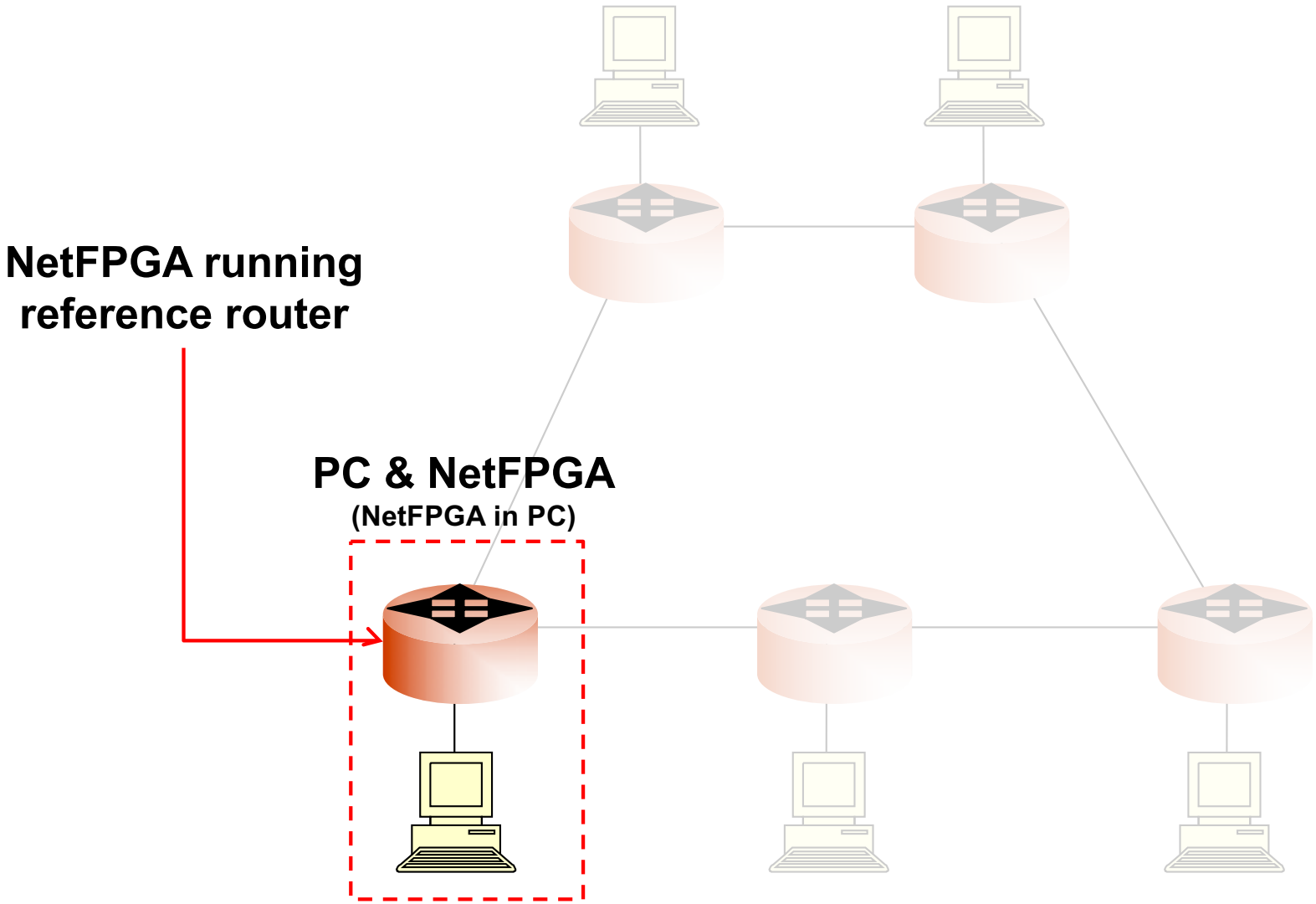
Operational IPv4 router



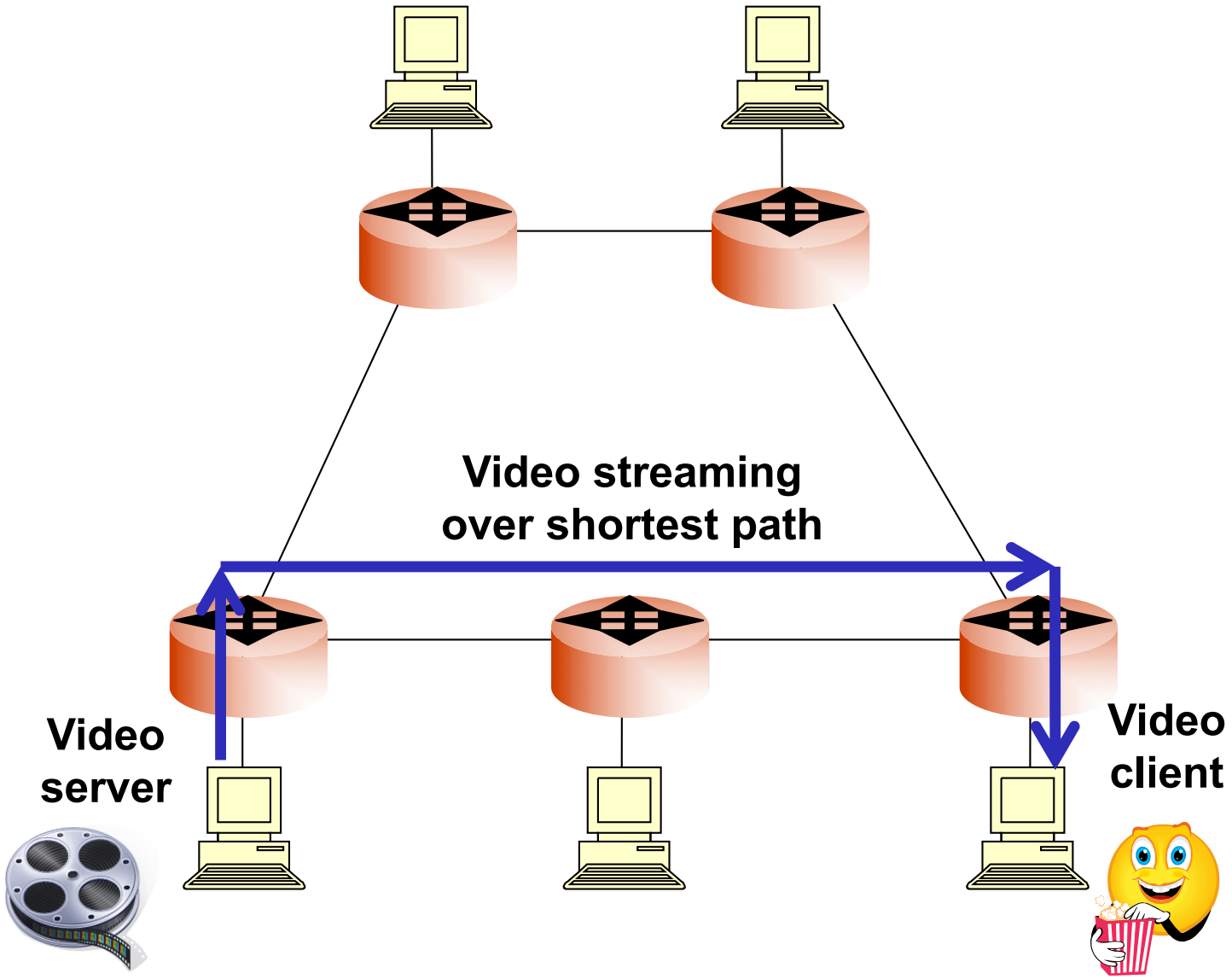
Streaming video



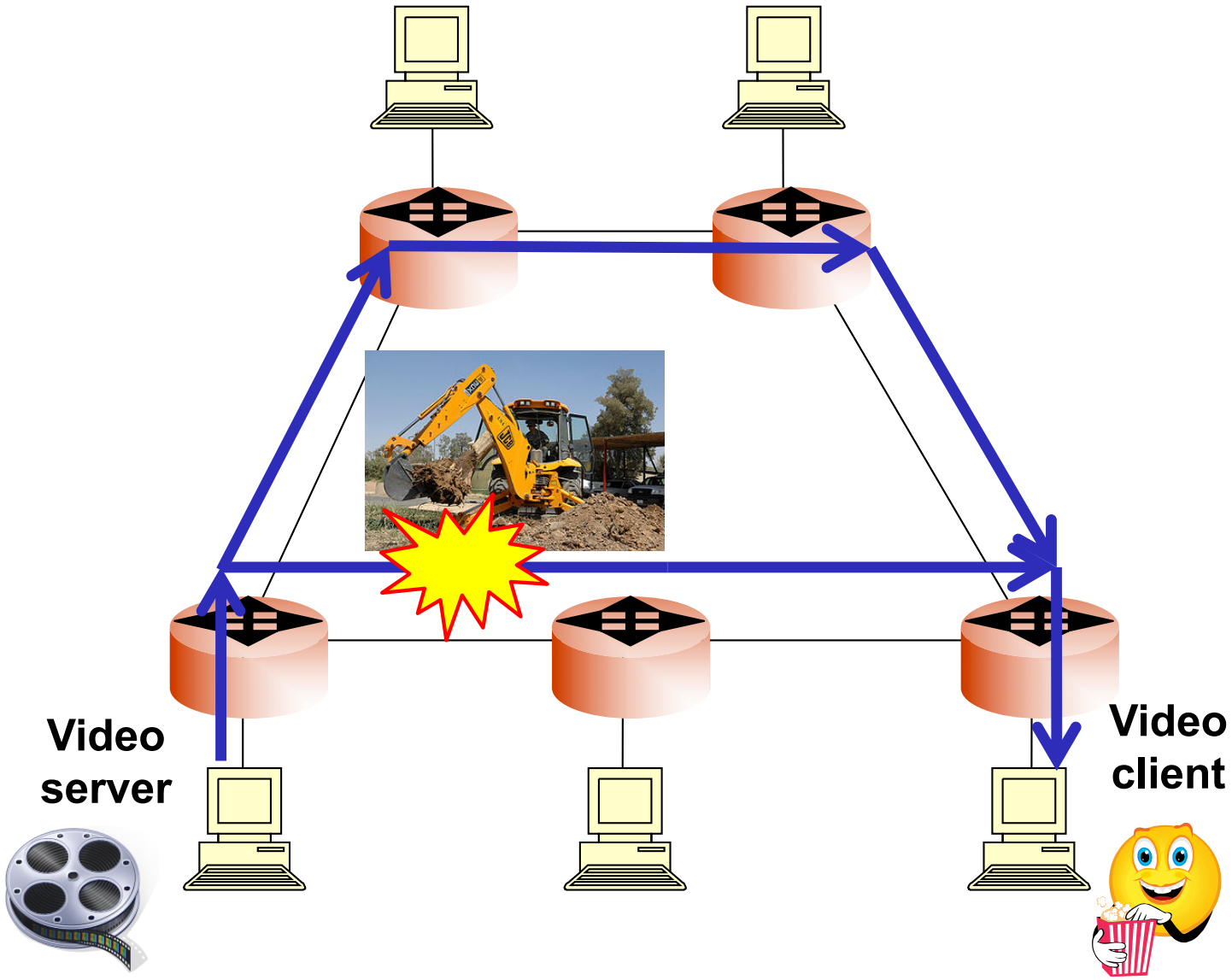
Streaming video



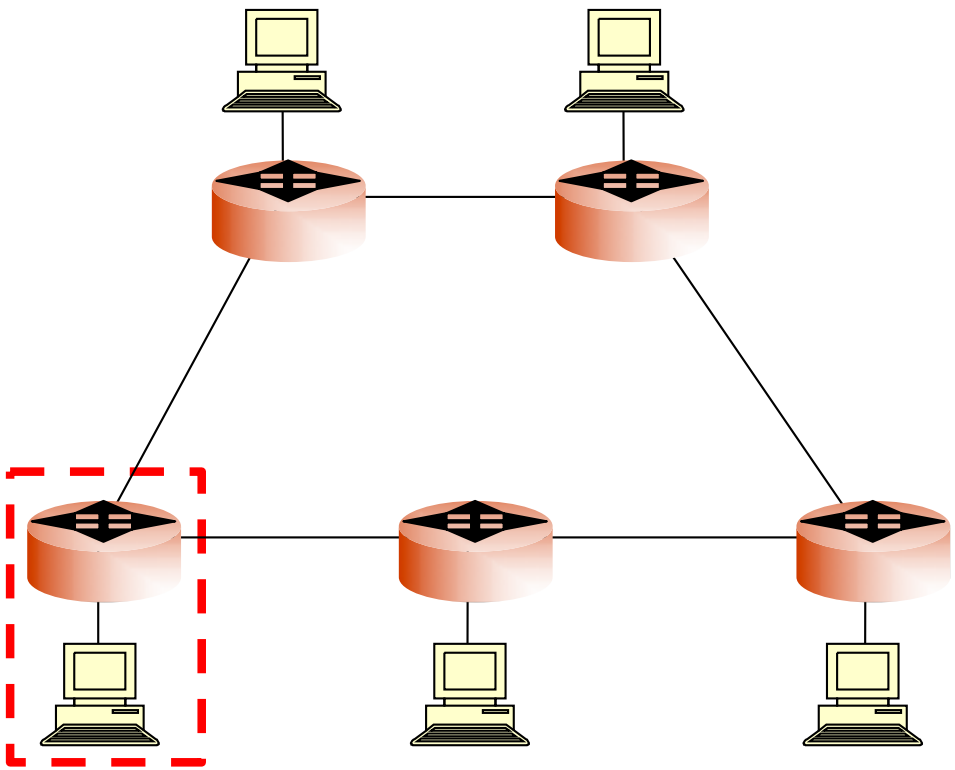
Streaming video



Streaming video



Observing the routing tables



Router Control Panel

File Window

Router Quickstart

Configuration | Statistics | Details

Router Configuration

Interface Configuration — Load From File

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 — Reset Entry

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	☑	☐	☐	☐	☐	☐	☐	☐

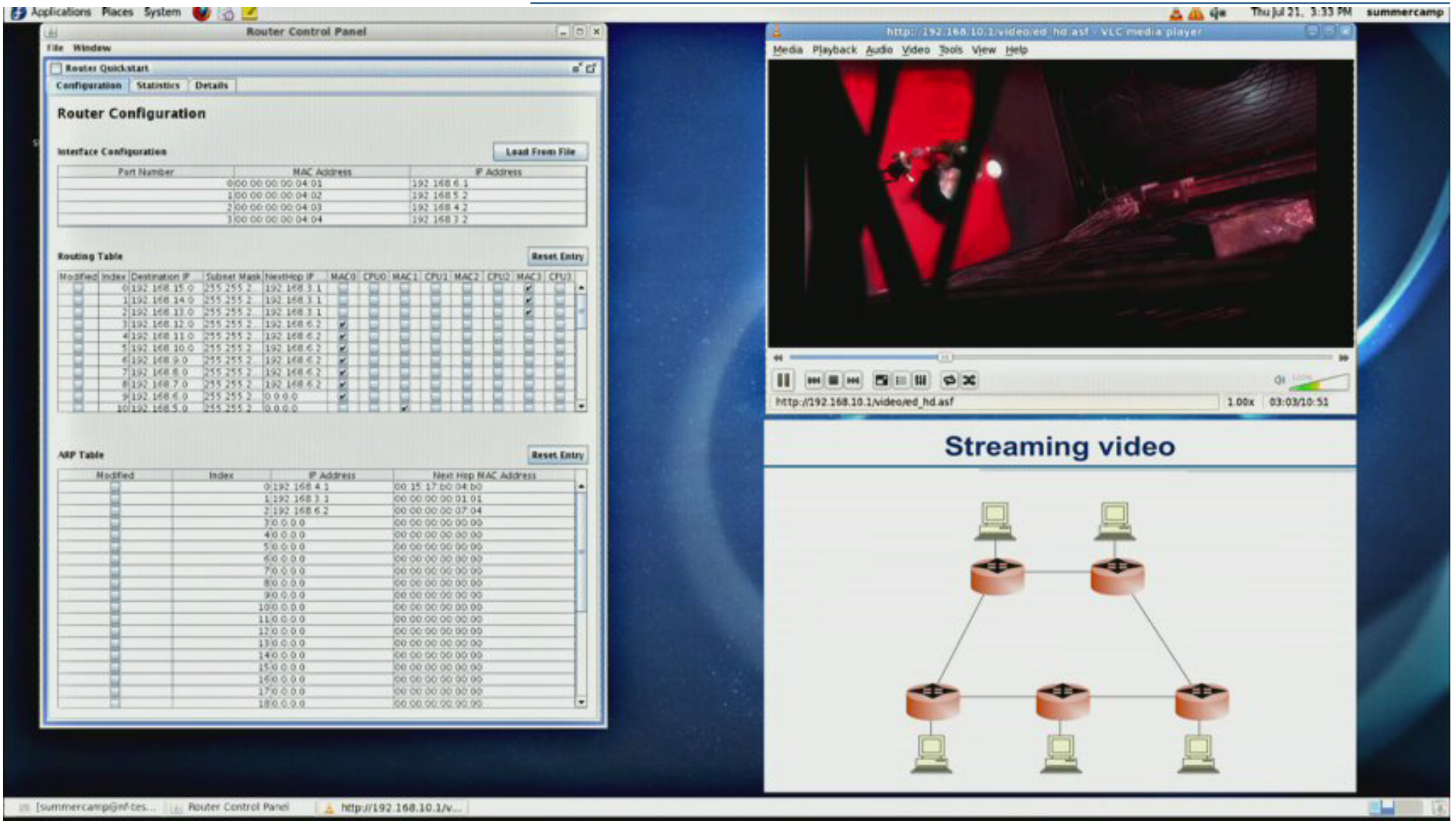
ARP Table — Reset Entry

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

Columns:

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

Example 1


Applications Places System
Thu Jul 21, 3:33 PM summercamp
Router Control Panel
File Window
Router Quickstart
Configuration Statistics Details
Router Configuration
Interface Configuration
Load From File
Part Number
MAC Address
IP Address
0 00:00:00:00:04:01 192.168.6.1
1 00:00:00:00:04:02 192.168.5.2
2 00:00:00:00:04:03 192.168.4.2
3 00:00:00:00:04:04 192.168.3.2
Routing Table
Reset Entry
Modified Index Destination IP Subnet Mask NextHop IP MAC0 CPU0 MAC1 CPU1 MAC2 CPU2 MAC3 CPU3
0 192.168.15.0 255.255.2... 192.168.3.1
1 192.168.14.0 255.255.2... 192.168.3.1
2 192.168.13.0 255.255.2... 192.168.3.1
3 192.168.12.0 255.255.2... 192.168.6.2
4 192.168.11.0 255.255.2... 192.168.6.2
5 192.168.10.0 255.255.2... 192.168.6.2
6 192.168.9.0 255.255.2... 192.168.6.2
7 192.168.8.0 255.255.2... 192.168.6.2
8 192.168.7.0 255.255.2... 192.168.6.2
9 192.168.6.0 255.255.2... 0.0.0.0
10 192.168.5.0 255.255.2... 0.0.0.0
ARP Table
Reset Entry
Modified Index IP Address Next Hop MAC Address
0 192.168.4.1 00:15:17:b0:04:b0
1 192.168.3.1 00:00:00:00:01:01
2 192.168.6.2 00:00:00:00:07:04
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
15 0.0.0.0 00:00:00:00:00:00
16 0.0.0.0 00:00:00:00:00:00
17 0.0.0.0 00:00:00:00:00:00
18 0.0.0.0 00:00:00:00:00:00
http://192.168.10.1/video/ed_hd.asf - VLC media player
Media Playback Audio Video Tools View Help
http://192.168.10.1/video/ed_hd.asf
1.00x
03:03/10:51
Streaming video
[summercamp@nf-tes...
Router Control Panel
http://192.168.10.1/v...

Integrated Circuit Technology And Field Programmable Gate Arrays (FPGAs)

Integrated Circuit Technology

Full-custom Design

- Complementary Metal Oxide Semiconductor (CMOS)

Semi-custom ASIC Design

- Gate array
- Standard cell

Programmable Logic Device

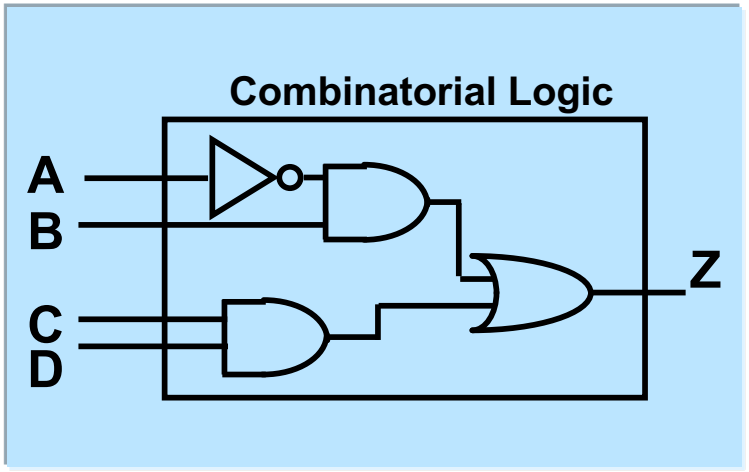
- Programmable Array Logic
- Field Programmable Gate Arrays

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

An older 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

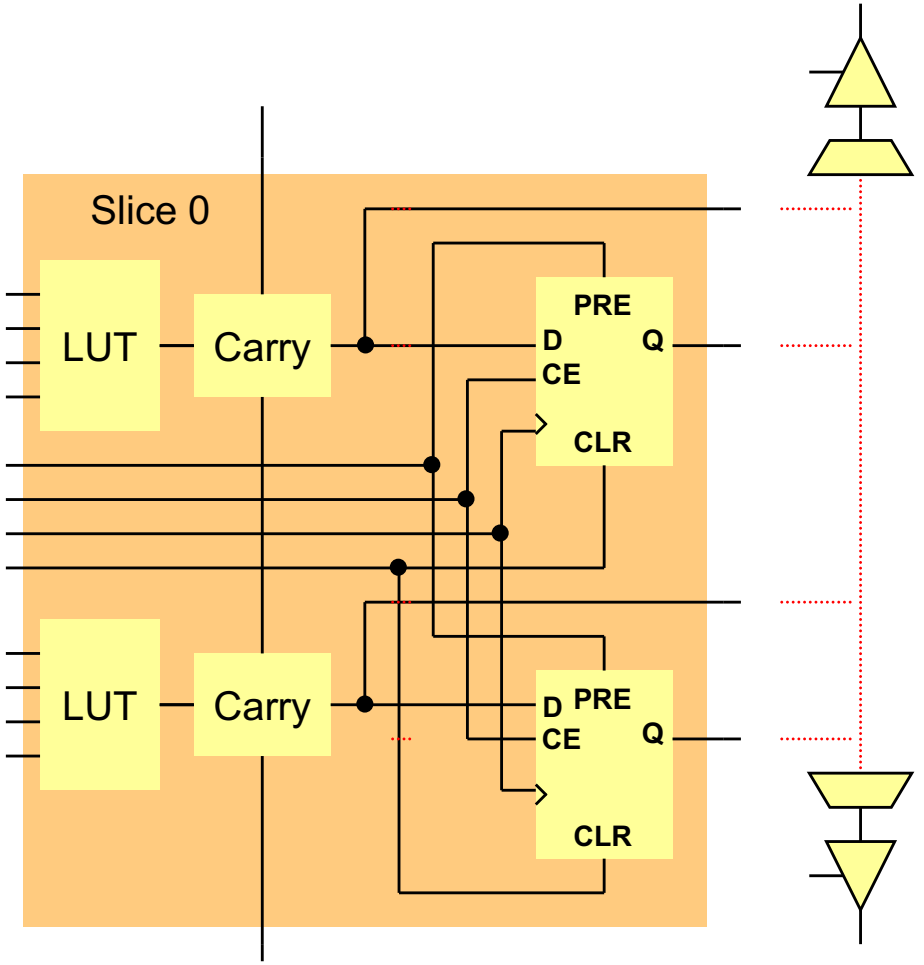
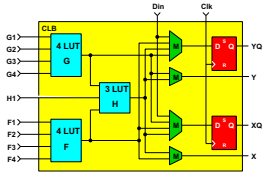


Diagram From: Xilinx, Inc.

Field Programmable Gate Arrays

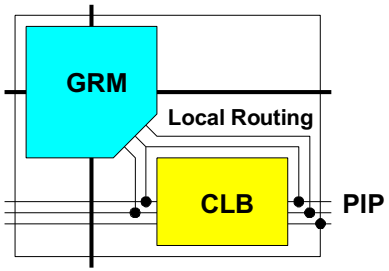
An Older CLB

- Primitive element of FPGA



Routing Module

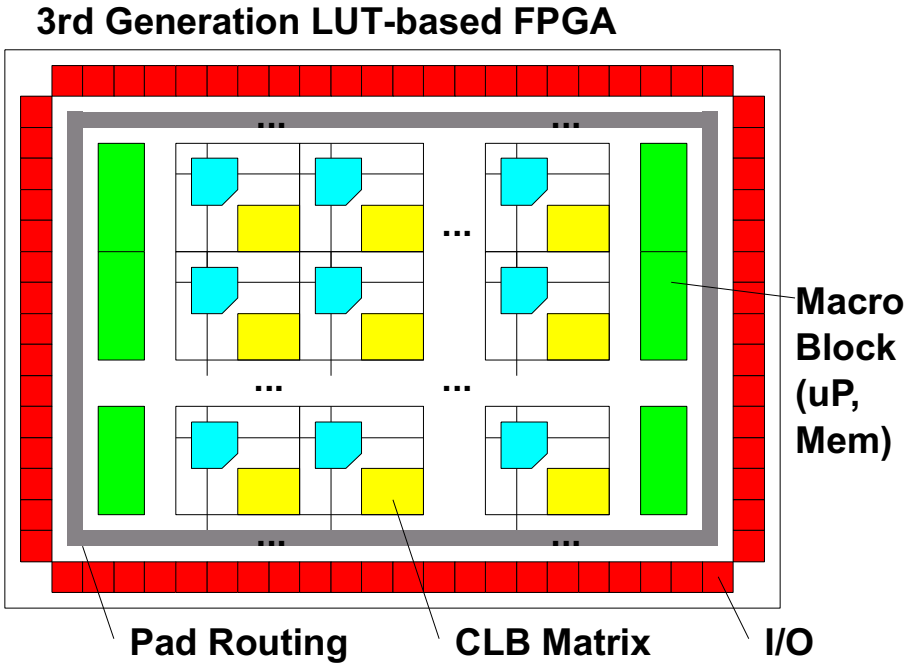
- Global routing
- Local interconnect



Macro Blocks

- Block Memories
- Microprocessor

I/O Blocks



Xilinx Virtex-7

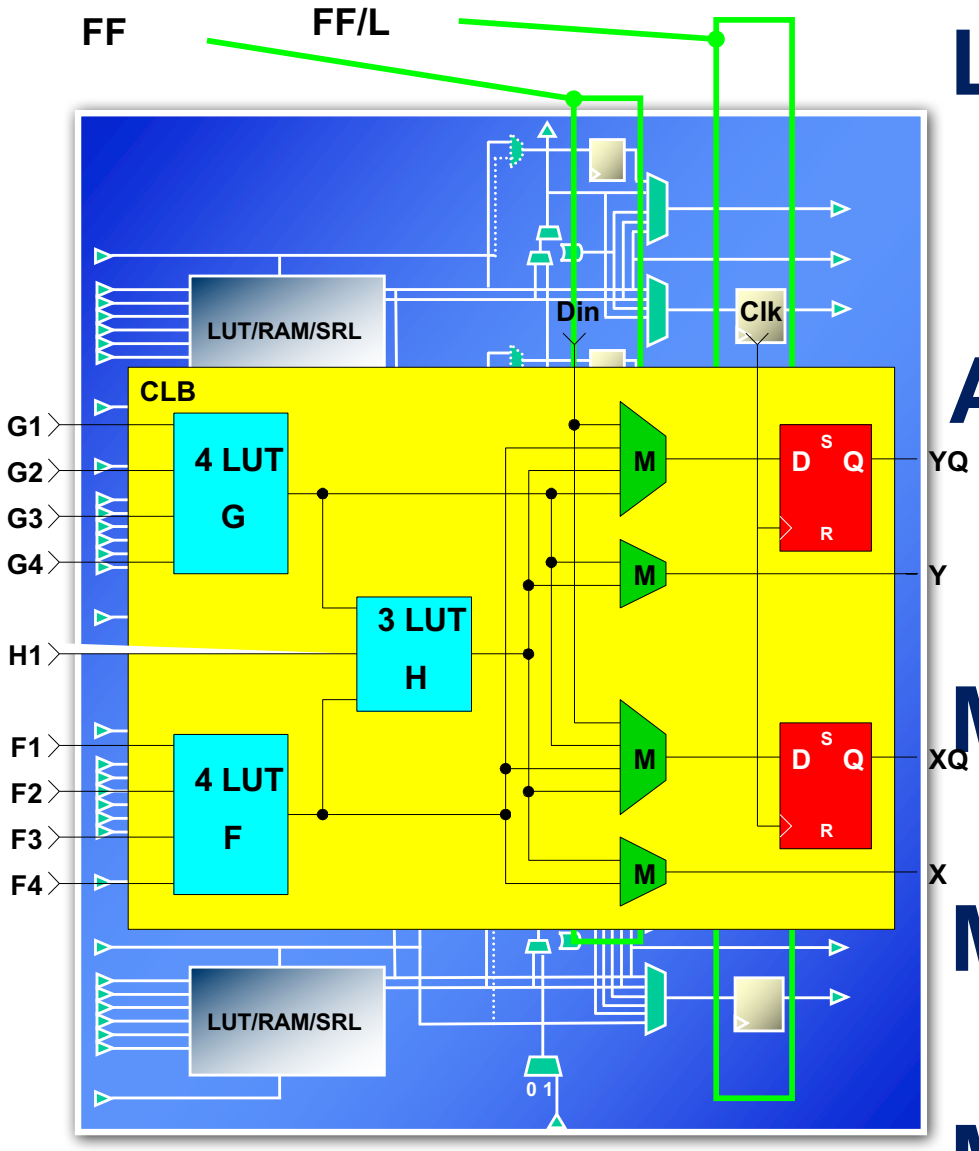


Diagram From: Xilinx, Inc.

Larger

CLB Slice Structure

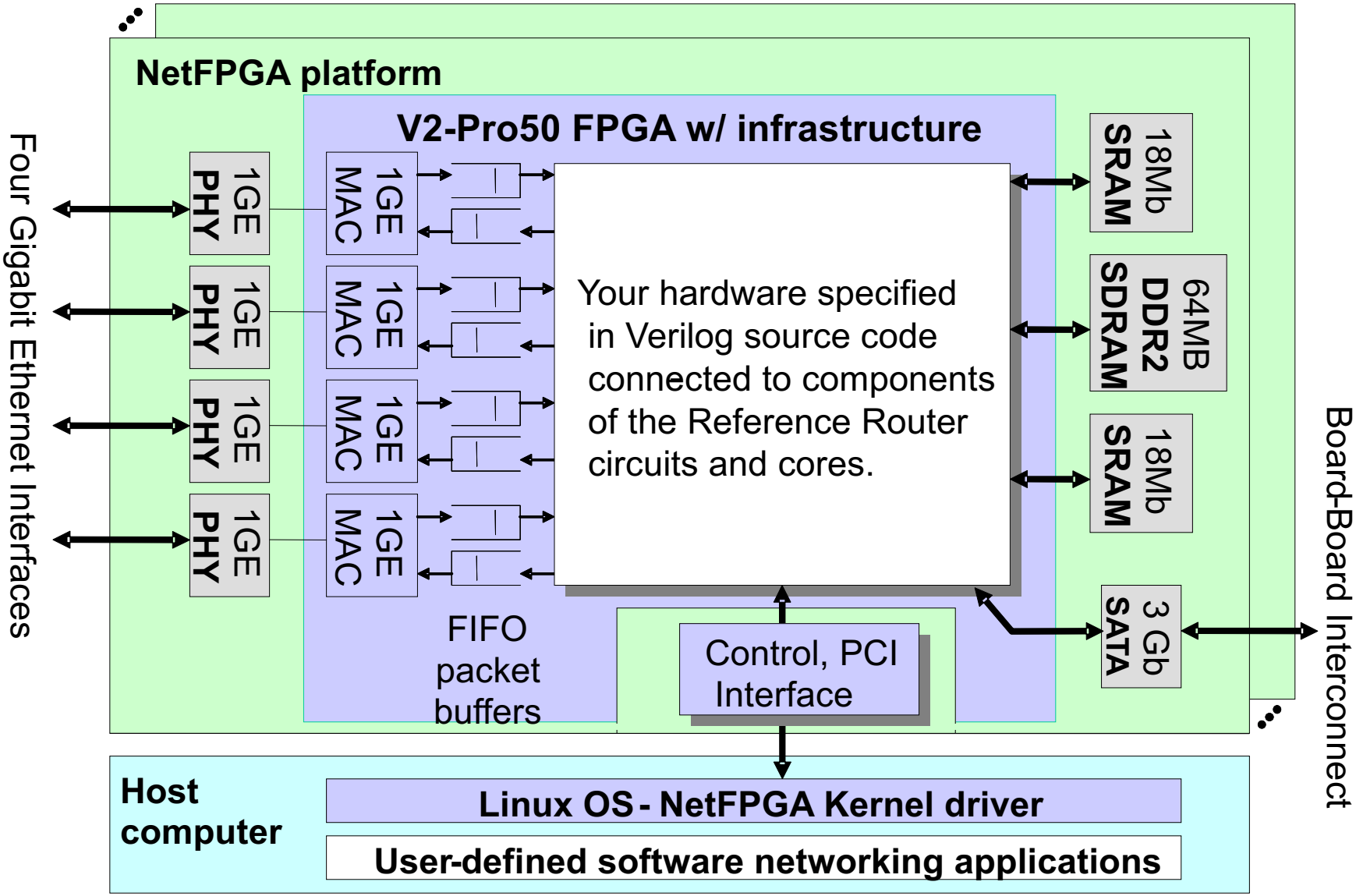
Along with increases or additions of

More Macro blocks

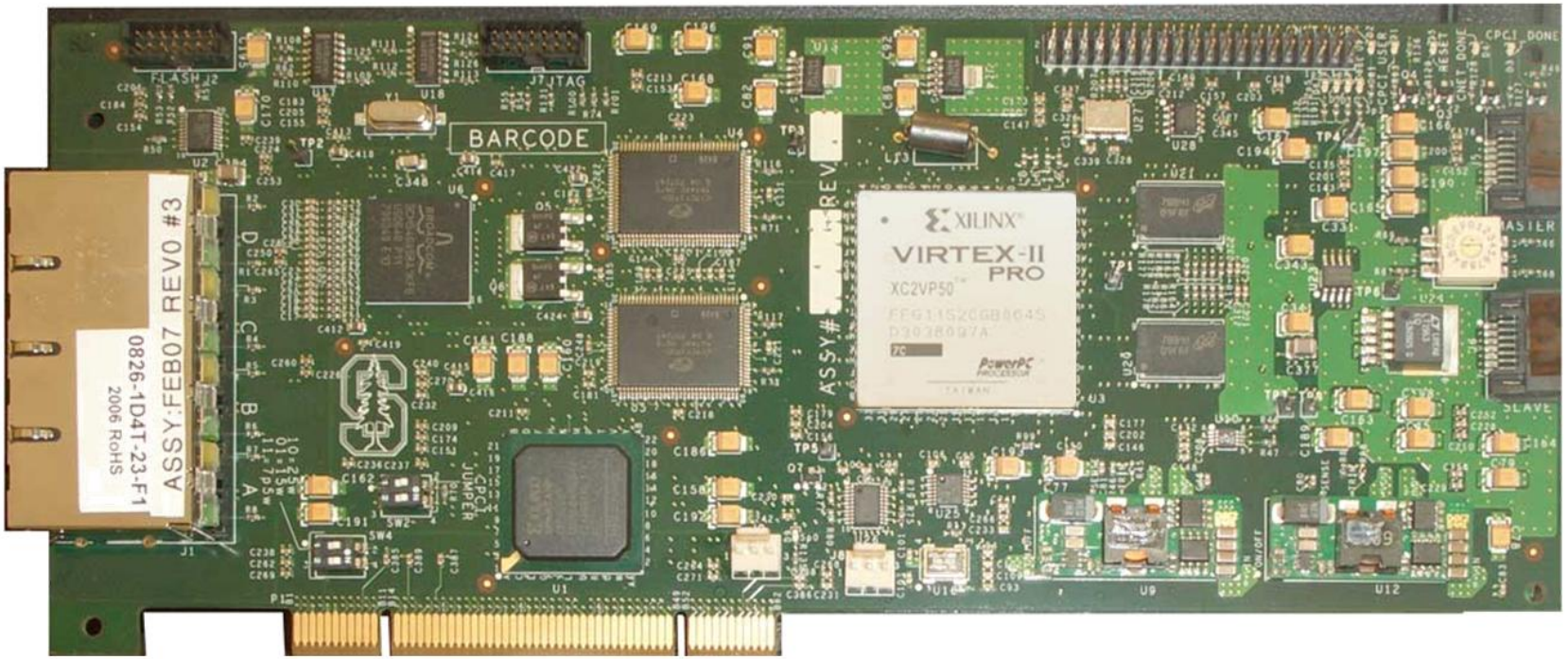
More Routing resources

More I/O resource

NetFPGA Block Diagram



Details of the NetFPGA



- **Fits into standard PCI slot**
  - Standard Bus: 32 bits, 33 MHz
- **Provides interfaces for processing network packets**
  - 4 Gigabit Ethernet Ports
- **Allows hardware-accelerated processing**
  - Implemented with Field Programmable Gate Array (FPGA) Logic

Introduction to the Verilog Hardware Description Language

Hardware Description Languages

- **Concurrent**
  - By default, Verilog statements evaluated concurrently
- **Express *fine grain* parallelism**
  - Allows *gate-level* parallelism
- **Provides Precise Description**
  - Eliminates ambiguity about operation
- **Synthesizable**
  - Generates hardware from description

Verilog Data Types

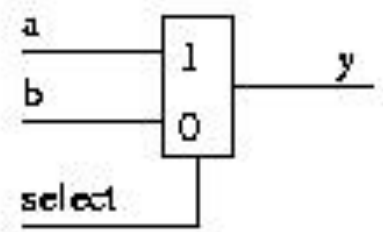
```
reg [7:0] A;  // 8-bit register, MSB to LSB
              // (Preferred bit order for NetFPGA)
reg [0:15] B;  // 16-bit register, LSB to MSB

B = {A[7:0],A[0:7]}; // Assignment of bits

reg [31:0] Mem [0:1023]; // 1K Word Memory

integer Count;   // simple signed 32-bit integer
integer K[1:64]; // an array of 64 integers
time Start, Stop; // Two 64-bit time variables
```

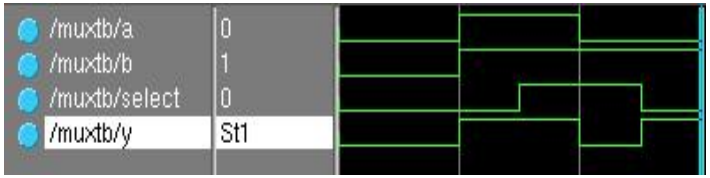
From: CSCI 320 Computer Architecture
Handbook on Verilog HDL, by Dr. Daniel C. Hyde :
https://www.ge.infn.it/~pratolo/verilog/Handboook_Bucknell.pdf

Signal Multiplexers

Two input multiplexer (using if / else)

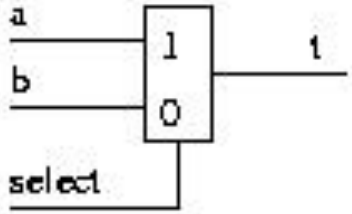
```
reg y;
always @*
      if (select)
         y = a;
      else
         y = b;
```

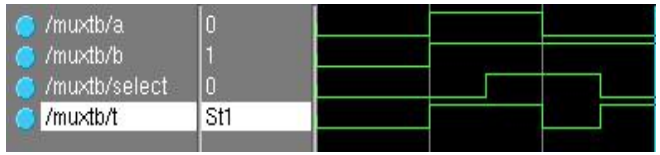




Two input multiplexer (using ternary operator ?:)

```
wire t = (select ? a : b);
```



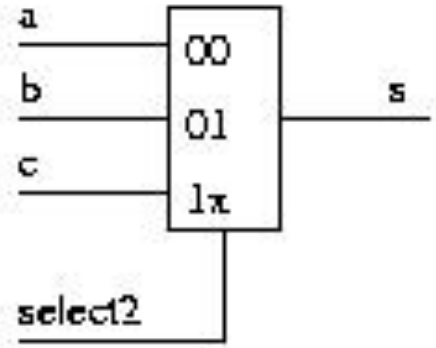


From: http://eesun.free.fr/DOC/VERILOG/synvlg.html
(now defunct but *an alternative is* here http://www.asic-world.com/verilog/)

Larger Multiplexers

Three input multiplexer

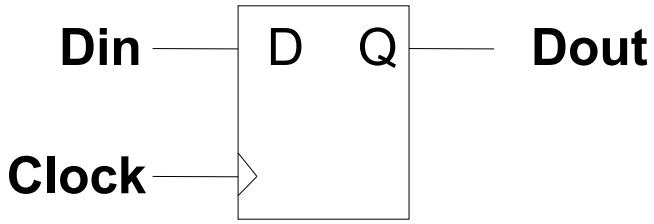
```
reg s;
always @*
   begin
   case (select2)
      2'b00: s = a;
      2'b01: s = b;
      default: s = c;
   endcase
   end
```



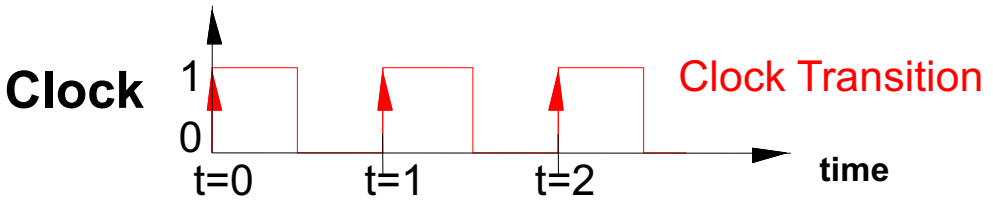
From: http://eesun.free.fr/DOC/VERILOG/synvlg.html
(now defunct but *an alternative is* here http://www.asic-world.com/verilog/)

Synchronous Storage Elements

- **Values change at times governed by clock**

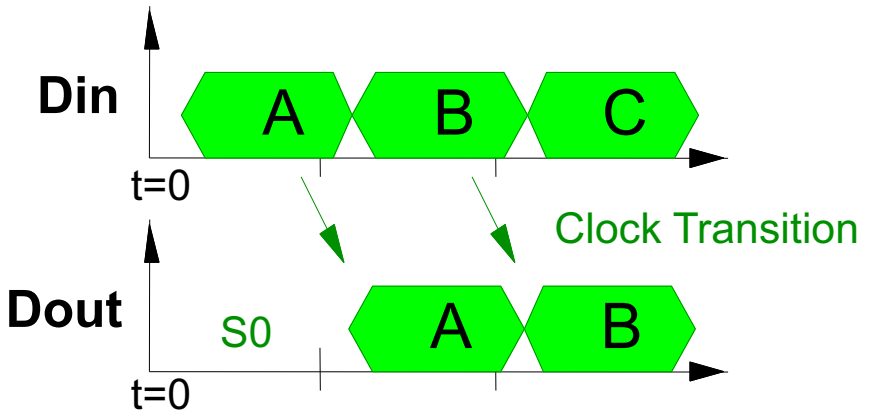


– Clock
  - Input to circuit

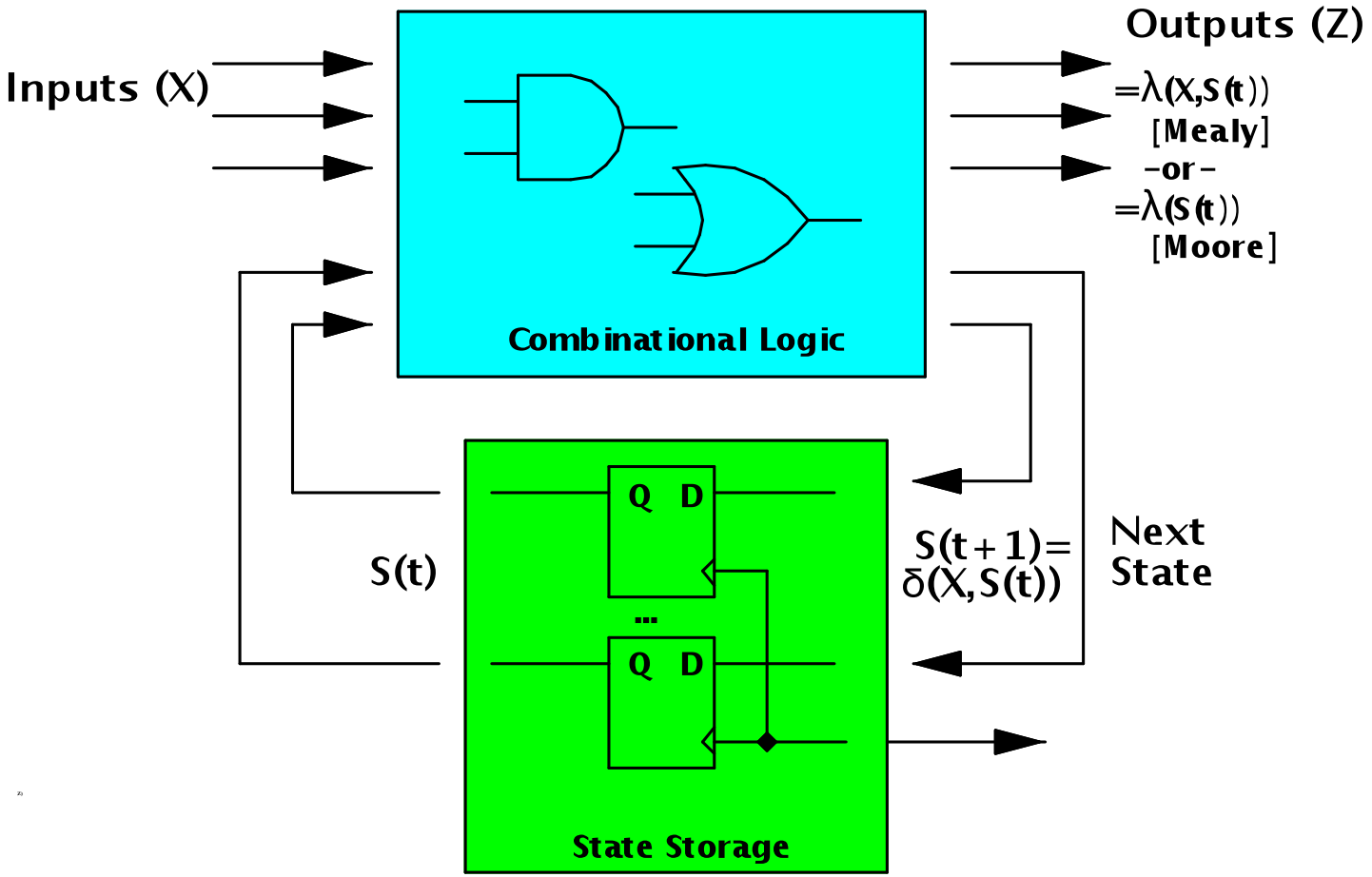


– Clock Event
  - Example: Rising edge

– Flip/Flop
  - Transfers value from D_{in} to D_{out} on clock event



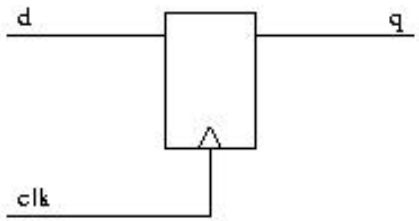
Finite State Machines

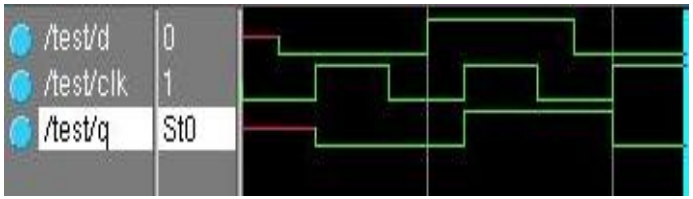


Synthesizable Verilog: Delay Flip/Flops

D-type flip flop

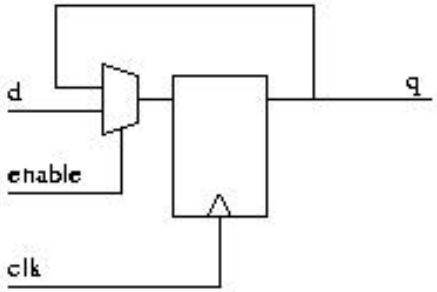
```
reg q;
always @ (posedge clk)
     q <= d;
```

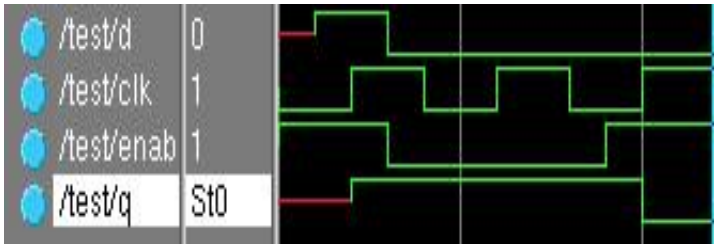




D type flip flop with *data enable*

```
reg q;
always @ (posedge clk)
     if (enable)
       q <= d;
```





From: http://eesun.free.fr/DOC/VERILOG/synvlg.html
(now defunct but *an alternative is* here http://www.asic-world.com/verilog/)

Review

NetFPGA as IPv4 router:

- **Reference hardware + SCONE software**
- **Routing protocol discovers topology**

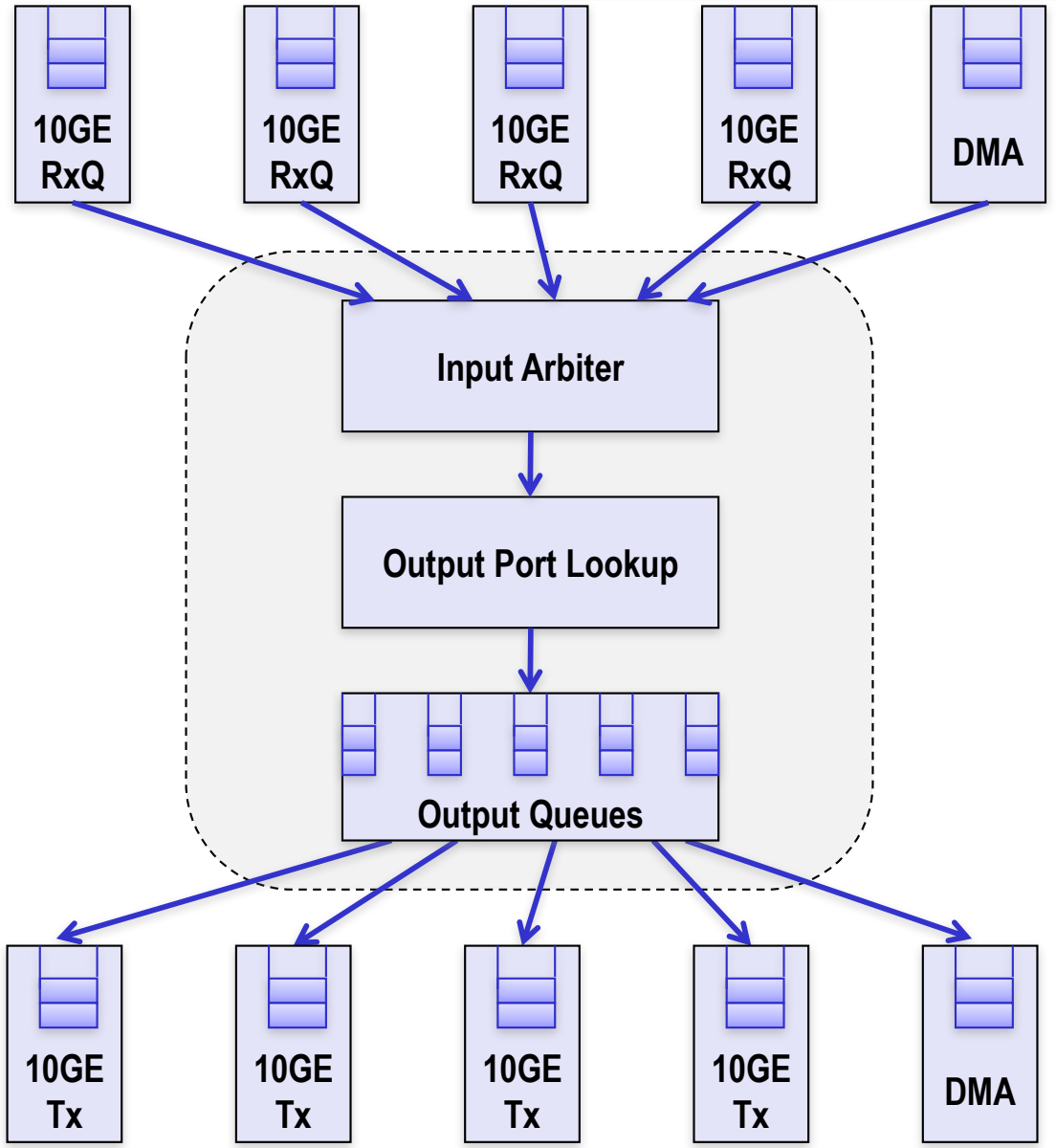
Video:

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

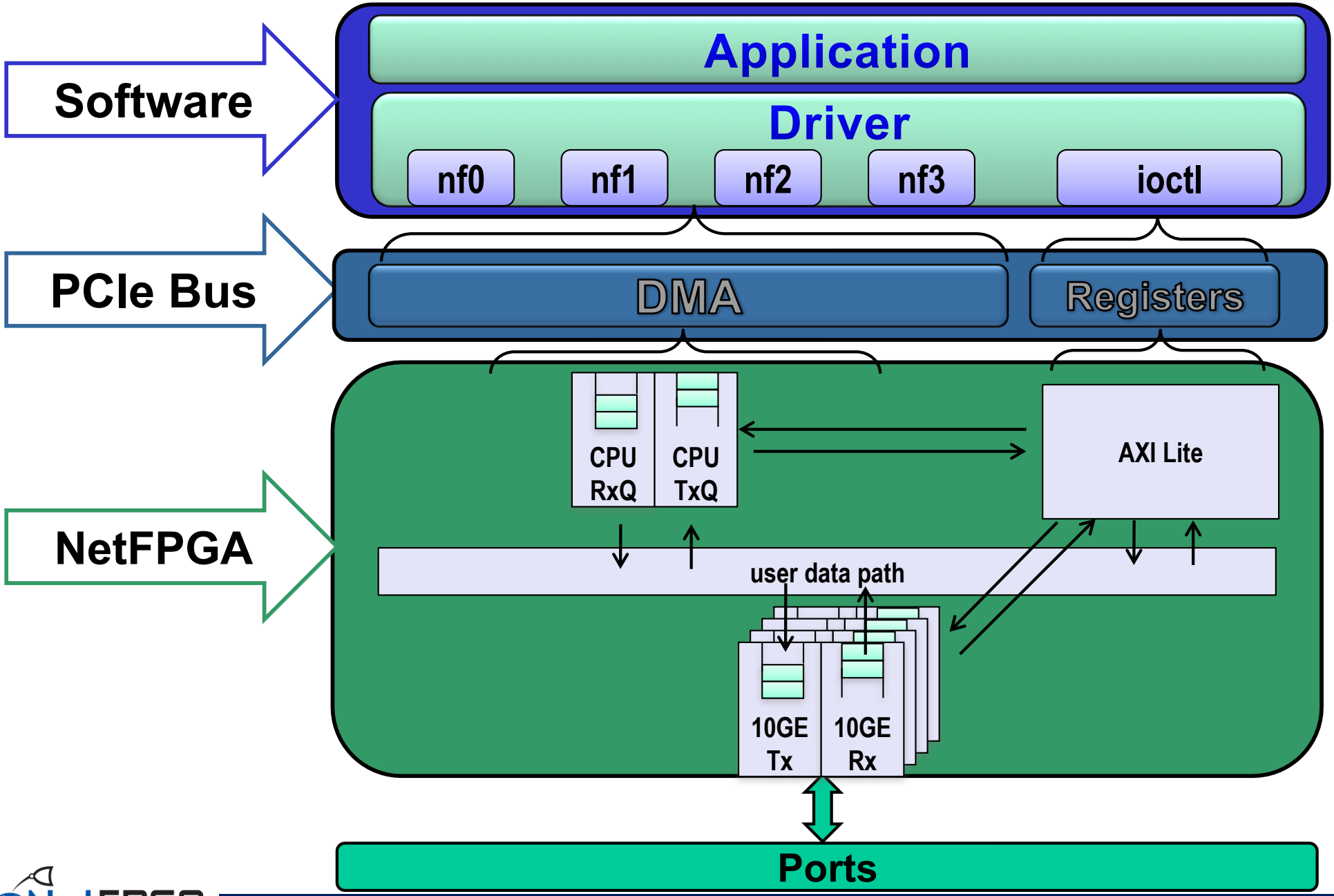
Section III: Life of a Packet

Reference Switch Pipeline

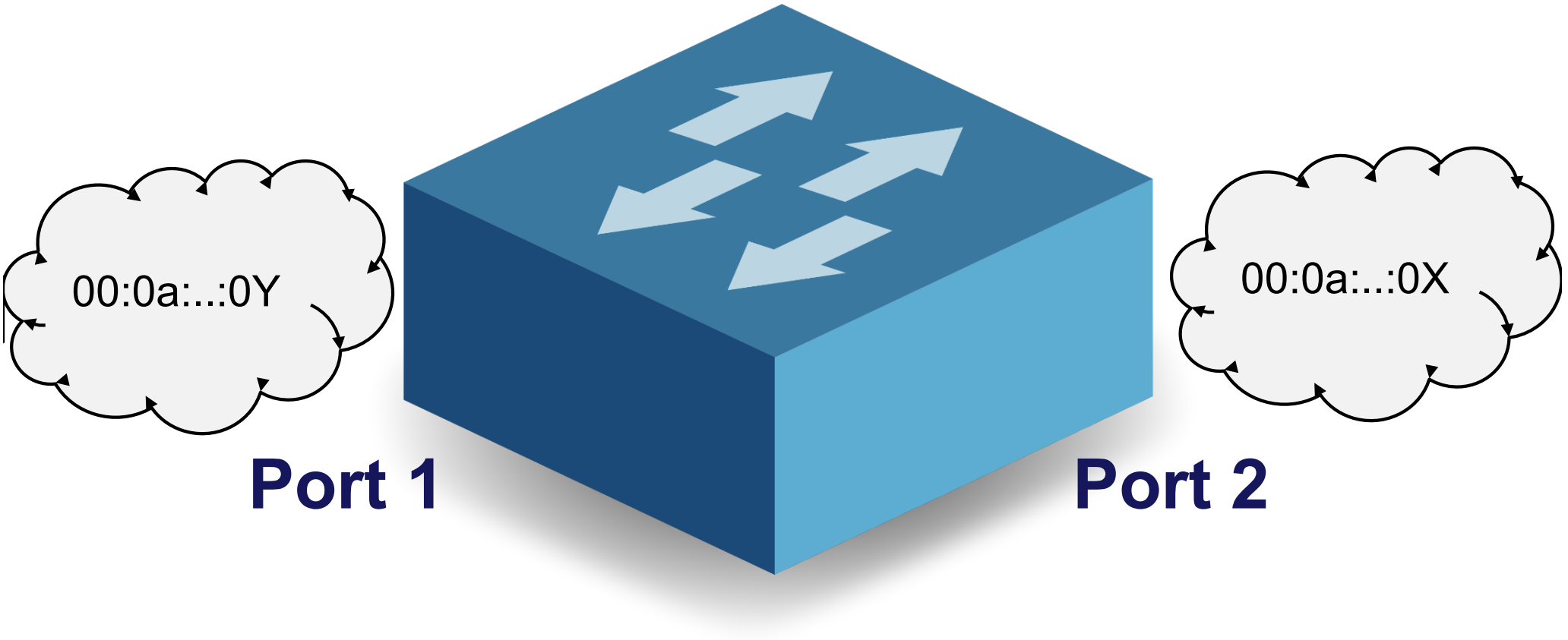
- **Five stages**
  - Input port
  - Input arbitration
  - Forwarding decision and packet modification
  - Output queuing
  - Output port
- **Packet-based module interface**
- **Pluggable design**



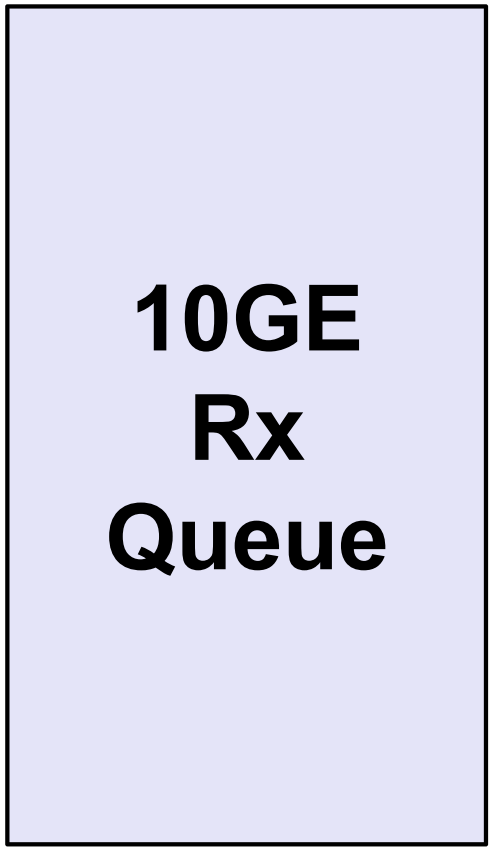
Full System Components



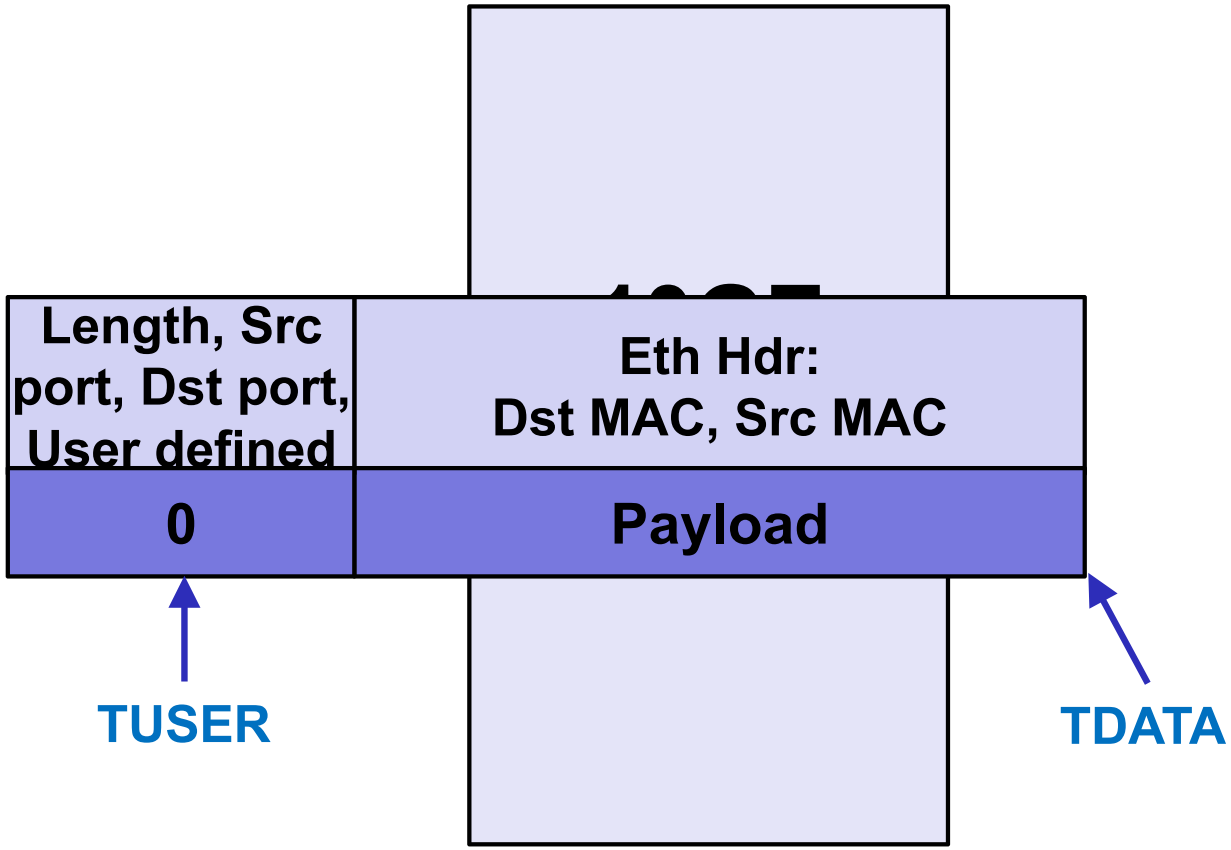
Life of a Packet through the Hardware



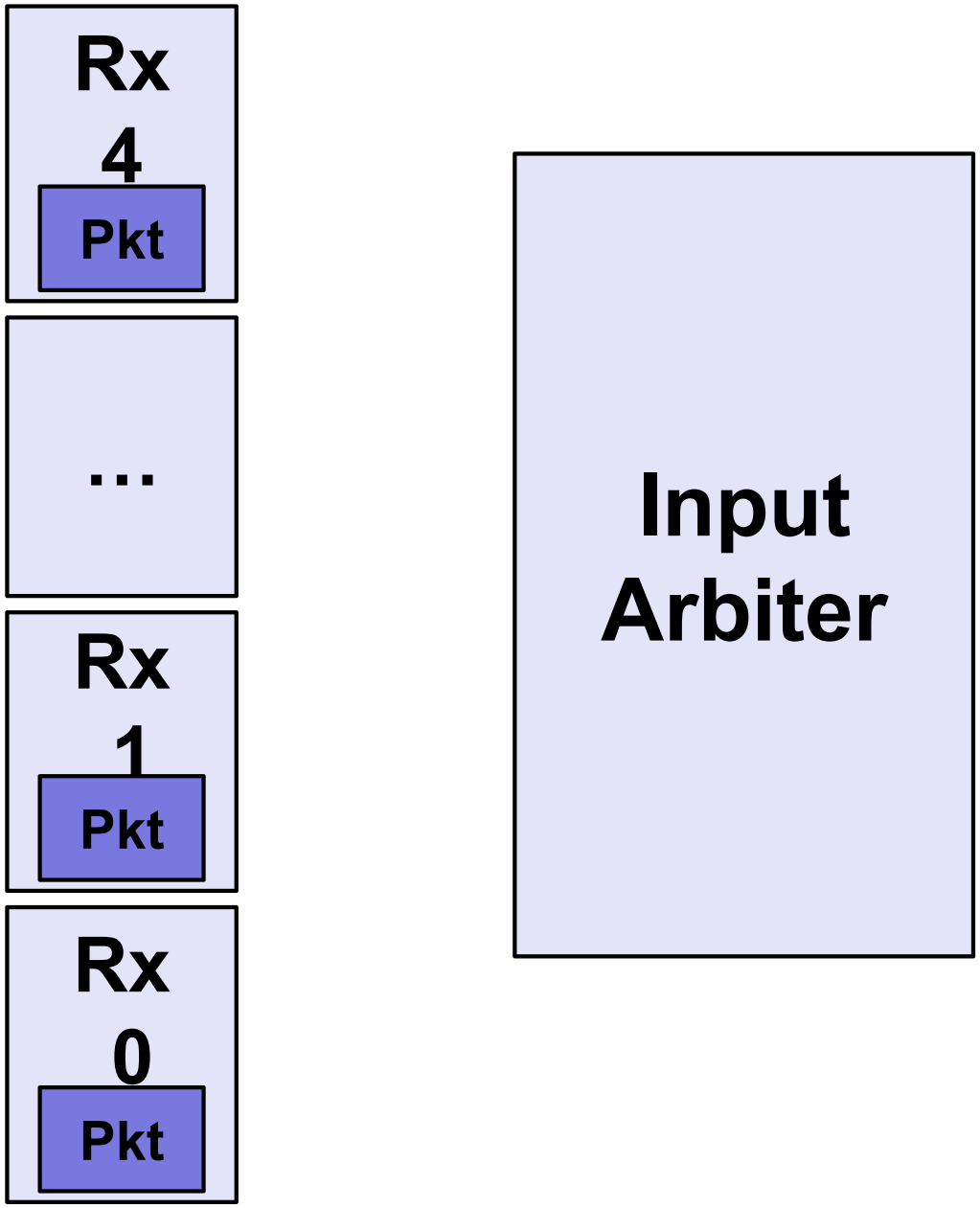
10GE Rx Queue



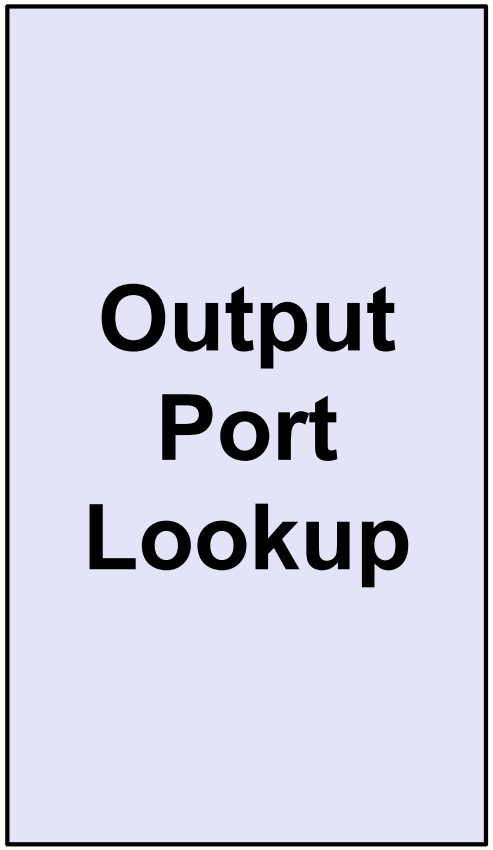
10GE Rx Queue



Input Arbiter



Output Port Lookup



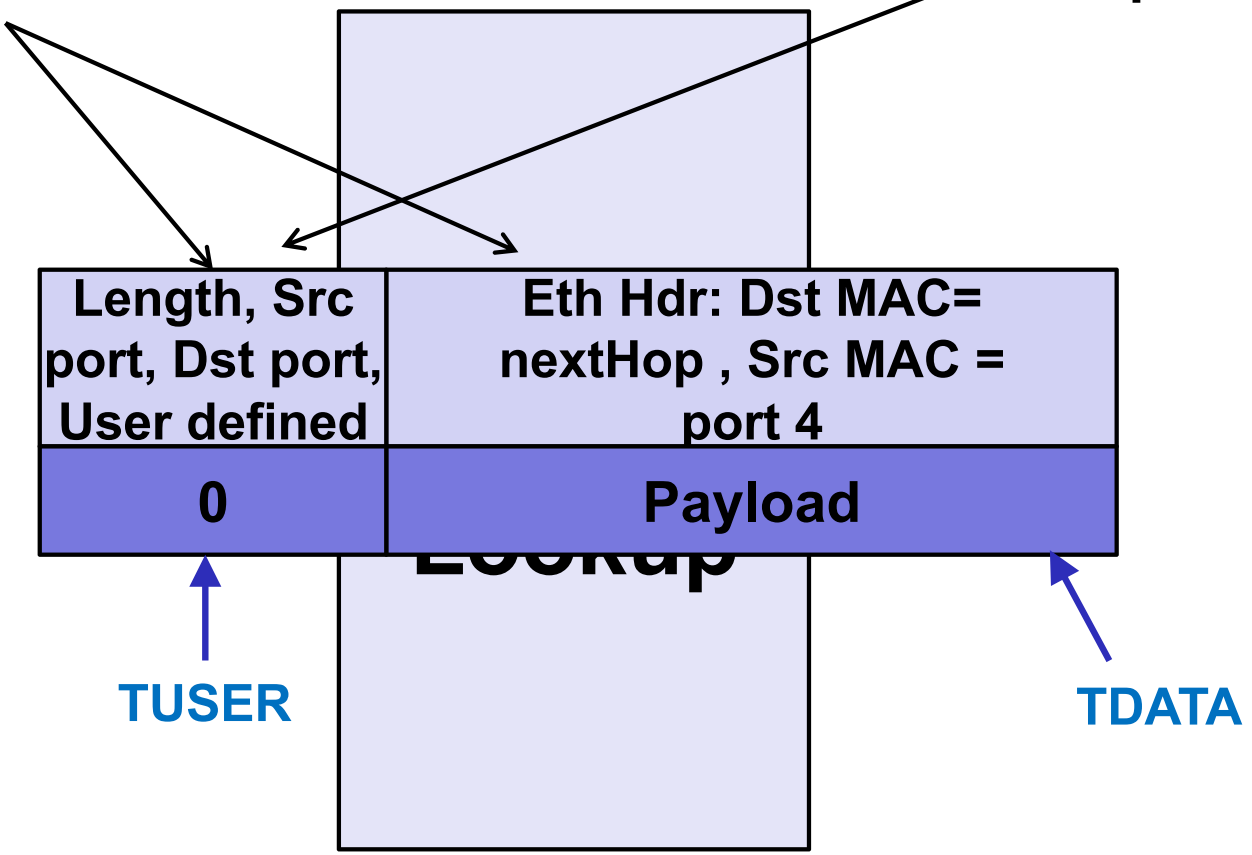
Output Port Lookup

1- Parse header: Src MAC, Dst MAC, Src port

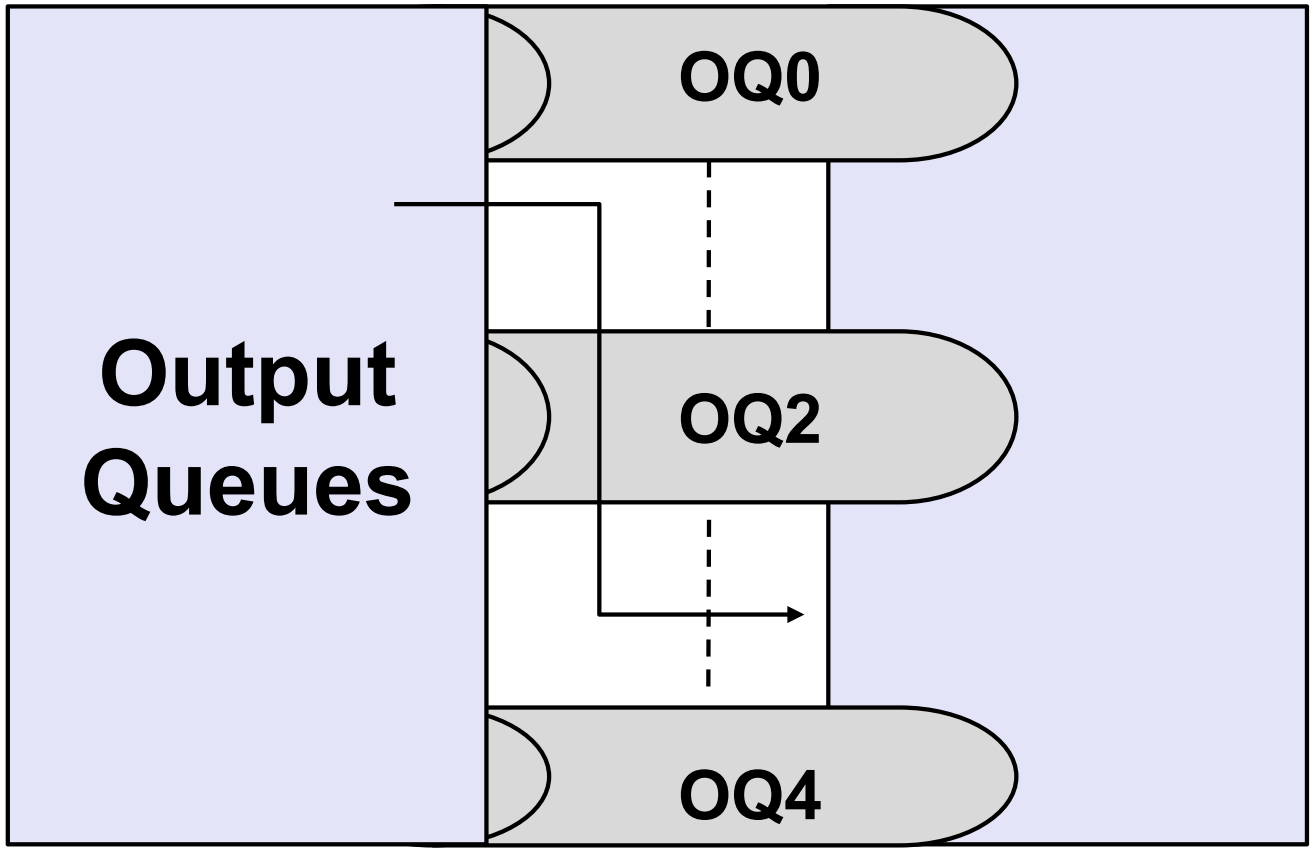
2 - Lookup next hop MAC& output port

3- Learn Src MAC & Src port

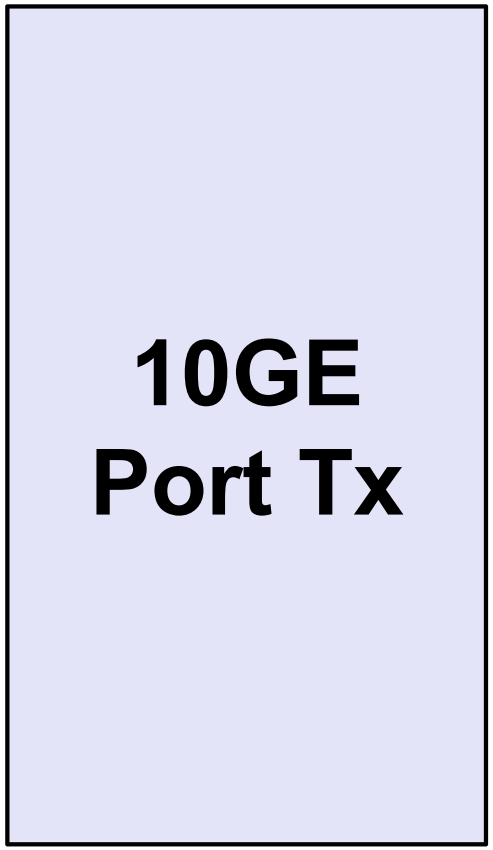
4- Update output port in TUSER



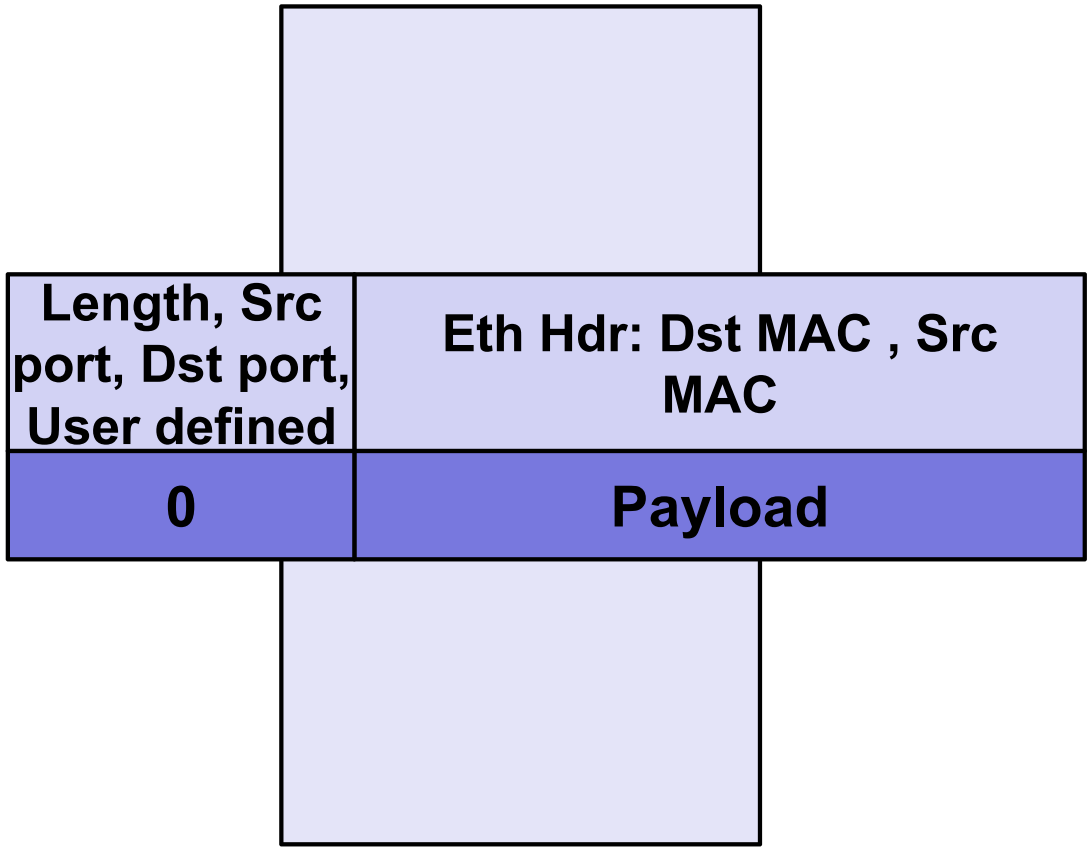
Output Queues



10GE Port Tx



MAC Tx Queue


Length, Src port, Dst port, User defined
Eth Hdr: Dst MAC , Src MAC
0
Payload

NetFPGA-Host Interaction

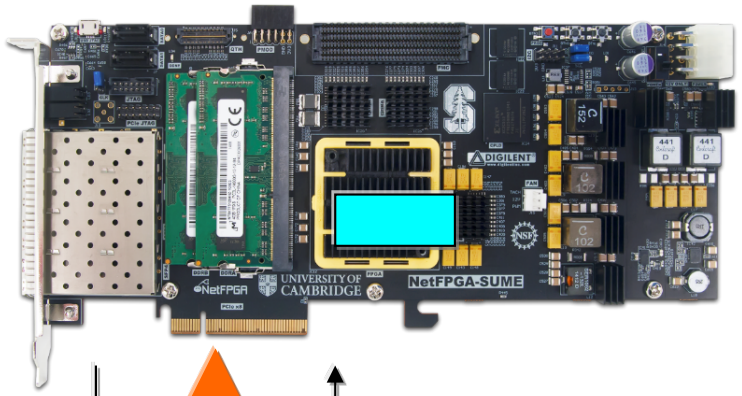
- **Linux driver interfaces with hardware**
  - Packet interface via standard Linux network stack
  - Register reads/writes via ioctl system call with wrapper functions:
    - rwaxi(int address, unsigned *data);

    eg:
    rwaxi(**0x7d4000000**, &val);

NetFPGA-Host Interaction

NetFPGA to host packet transfer

1. Packet arrives – forwarding table sends to DMA queue



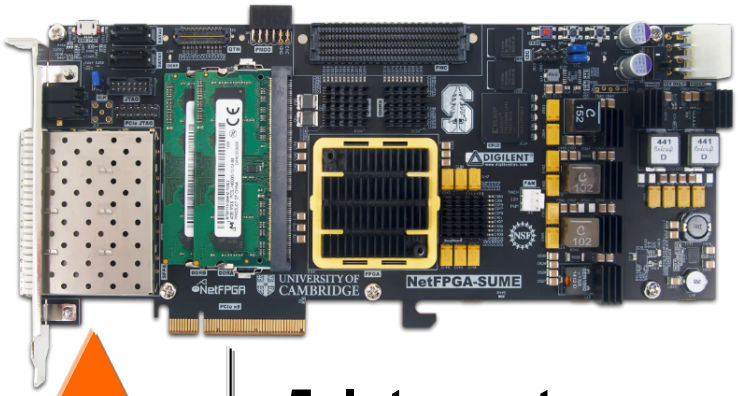
2. Interrupt notifies driver of packet arrival

PCIe Bus

3. Driver sets up and initiates DMA transfer

NetFPGA-Host Interaction

NetFPGA to host packet transfer (cont.)



4. NetFPGA transfers packet via DMA

PCIe 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

PCIe Bus

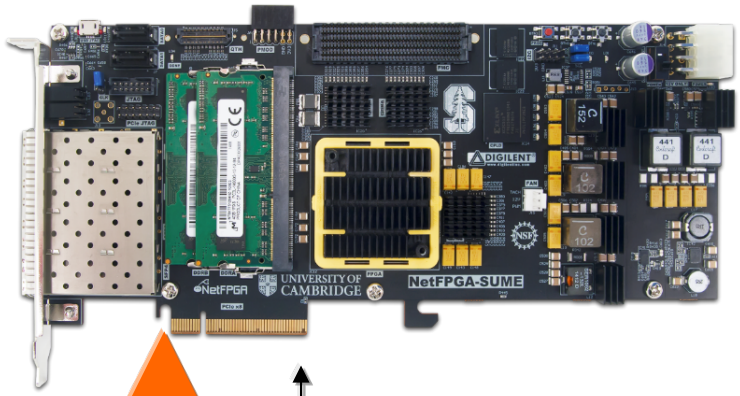
3. Interrupt signals completion of DMA

1. Software sends packet via network sockets

Packet delivered to driver

NetFPGA-Host Interaction

Register access



PCIe Bus

2. Driver performs PCIe memory read/write

1. Software makes ioctl call on network socket

ioctl passed to driver