NetFPGA Hardware Architecture

Jeffrey Shafer

Some slides adapted from Stanford NetFPGA tutorials
NetFPGA

http://netfpga.org
NetFPGA Components

- **Virtex-II Pro 50 FPGA**
  - 53,136 logic cells
  - 4,176 Kbit block RAM
  - 2 x PowerPC cores (not used)

- **Gigabit Ethernet ports** - 4

- **Memory**
  - SRAM – 4.5MB (2 parallel banks - 18Mbit / 2.25 MByte)
  - DDR2 DRAM – 64 MBytes

- **PCI Bus Interface**
  - 32 bits / 33 MHz / 1066 Mbit/s bandwidth
  - Spartan FPGA used as PCI controller

- **Multi-gigabit I/O (2 SATA ports)**
  - Allows multiple NetFPGAs within a PC to be chained together

- **More Details?**
  - Schematic: [http://netfpga.org/NetFPGA_PCB_r2.pdf](http://netfpga.org/NetFPGA_PCB_r2.pdf)

How does the NetFPGA board become a router?
Reference Designs

- NetFPGA design team has provided
  - Reference NIC – 4 port Ethernet NIC
  - Reference Router – 4 port IP router

- What factors influenced these designs?
  - Simple architecture (for use in education)
  - Fits on the FPGA and meets timing constraints

- These designs do not necessarily model a commercial NIC or router
  - We’ll talk about real devices later this semester…

- We’ll give you the NIC design, and you will turn it into a switch and IP router
Router Tasks (Per Packet)

- Receive packet on incoming port
  - Read destination

- Where is packet going?
  - Lookup destination in forwarding table
  - Determine next hop address and outgoing port
    - If not found, use ICMP to handle error
  - Lookup link-layer address of next hop (e.g. Ethernet MAC)
    - If not found, use ARP to resolve address

- Manipulate IP header
  - Decrement TTL and update header checksum

- Manipulate link-layer header
  - Modify link-layer source and destination address, update CRC

- Buffer packet in the output queue
- Transmit packet onto outgoing link

This is only a partial list of router tasks
(Real routers, and your project, do more!)
Generic Datapath Architecture

Header Processing

Lookup Dst IP Address
Lookup Dst IP Address
Update Header
Update Header
Queue Packet
Queue Packet

Forwarding Table
Forwarding Table
Buffer Memory
Buffer Memory

Data Hdr
Data Hdr

IP Address Next Hop
NetFPGA IP Router

Network Systems Architecture
NetFPGA Block Diagram

NetFPGA PCI Board

Virtex-II Pro 50 FPGA

Custom Router Pipeline

Composed of:
- Verilog source code
- Xilinx Cores

18Mb SRAM
1GE PHY
1GE MAC
1GE MAC
1GE MAC
1GE MAC

FIFO packet buffers

Spartan FPGA PCI Interface

18Mb DDR2 SDRAM
64MB SDRAM

3 Gb SATA

Board to Board Interconnect

Four Gigabit Ethernet Interfaces

Host Computer

Linux OS - NetFPGA Kernel driver

User-defined software networking applications

Network Systems Architecture
Router Pipeline

- Five stages
  - Input
  - Input Arbitration
  - Routing Decision and packet modification
  - Output Queuing
  - Output

- Packet-based module interface
- Pluggable design
Inter-module Communication

Module \(i\) → Module \(i+1\)

- Data (64 bits)
- Ctrl (8 bits)
- wr
- rdy
### Inter-module Communication

- **Headers are appended to packet**

<table>
<thead>
<tr>
<th>Ctrl Word (8 bits)</th>
<th>Data Word (64 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Module Hdr</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>y</td>
<td>Last Module Hdr</td>
</tr>
<tr>
<td>0</td>
<td>Start of Ethernet Header</td>
</tr>
<tr>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>Start of IP Header</td>
</tr>
<tr>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>0x10</td>
<td>Last word of packet</td>
</tr>
</tbody>
</table>

- Router control data such as packet length, input port, output port, ...
- Packet being routed (including its headers)
  - Ctrl x00 = Packet
  - Ctrl x10 = End of packet
Exploring the Pipeline...

Network Systems Architecture
MAC Rx Queue (port 0)

<table>
<thead>
<tr>
<th>Eth Hdr:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst MAC = MAC(0),</td>
</tr>
<tr>
<td>Ethertype = IP</td>
</tr>
<tr>
<td>IP Hdr:</td>
</tr>
<tr>
<td>IP Dst: 192.168.2.3,</td>
</tr>
<tr>
<td>TTL: 64, Csum:0x3ab4</td>
</tr>
<tr>
<td>Data</td>
</tr>
</tbody>
</table>
### MAC Rx Queue (port 0)

<table>
<thead>
<tr>
<th>0xff</th>
<th>Pkt length, input port = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Eth Hdr: Dst MAC = MAC(0), Ethertype = IP</td>
</tr>
<tr>
<td>0</td>
<td>IP Hdr: IP Dst: 192.168.2.3, TTL: 64, Csum:0x3ab4</td>
</tr>
<tr>
<td>0</td>
<td>Data</td>
</tr>
</tbody>
</table>

Network Systems Architecture
Input Arbiter

Rx
Q 7
Pkt

...

Rx
Q 1
Pkt

Rx
Q 0
Pkt
Output Port Lookup
Output Port Lookup

1- Check input port consistent with Dst MAC

2- Check TTL, checksum

3- Lookup next hop IP & output port (LPM)

4- Lookup next hop MAC address (ARP)

5- Update header with output port(s)

6- Modify MAC Dst and Src addresses

7- Decrement TTL and update checksum

<table>
<thead>
<tr>
<th>0xff</th>
<th>Pkt length, Input port = 0, Output port = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EthHdr: Dst MAC = nextHop, Src MAC = port 4, Ethertype = IP</td>
</tr>
<tr>
<td>0</td>
<td>IP Hdr: IP Dst: 192.168.2.3, TTL: 63, Csum: 0x3ac2</td>
</tr>
<tr>
<td>0</td>
<td>Data</td>
</tr>
</tbody>
</table>
Output Queues

Network Systems Architecture
MAC Tx Queue
## MAC Tx Queue

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x04</td>
<td>output port = 4</td>
</tr>
<tr>
<td>0xff</td>
<td>Pkt length, input port = 0</td>
</tr>
<tr>
<td>0</td>
<td>EthHdr: Dst MAC = nextHop, Src MAC = port 4, Ethertype = IP</td>
</tr>
<tr>
<td>0</td>
<td>IP Hdr: IP Dst: 192.168.2.3, TTL: 63, Csum:0x3ac2</td>
</tr>
<tr>
<td>0</td>
<td>Data</td>
</tr>
</tbody>
</table>
Any Questions About Pipeline?
Course Projects

- You have to build the Output Port Lookup module shown in the preceding slides
  - The initial version provided is a simple NIC
    - Directly connects input port A to output port A
    - No intelligence!
- You will build Output Port Lookup modules to accomplish
  - Ethernet Hub
  - Learning Ethernet Switch
  - IP Router
What is Each Group Provided?

- User Space
- Linux Kernel
- Monitoring Software
- Control Software
- PCI
- NetFPGA Router
- Hardware
Per-Group Network Topology

Private Network (switch) - 192.168.0.xxx
Purpose: Management and Configuration

NetFPGA System
(number 1, address 201)

Host PC:
1 NIC

Router PCI Card:
4 Ports

RiceNet 2

comp519.cs.rice.edu
192.168.0.1
192.168.0.201

SSH Only

Server1
192.168.0.2

10.143.201.2
10.143.201.65

Server2
192.168.0.3

10.143.201.66

Server3
192.168.0.4

10.143.201.130

Cisco Router
(to the world!)
10.130.20.132

10.143.201.194

Public Networks (Accessible from RiceNet) - 10.143.xxx.yyy
Purpose: Data Streaming

Network Systems Architecture
Next Three Fridays (16th, 23rd, 30th)

- Meet in Lab (Abercrombie A123)

- Goals
  - Learn about NetFPGA library and basic reference NIC design
  - Write Verilog
    - Turn the NIC into an Ethernet Hub
  - Build hardware bitfile
  - Simulate design
  - Test design on real hardware
Assignment

- Before Friday’s tutorial:
  - Go to Tutorials page at [http://comp519.cs.rice.edu/](http://comp519.cs.rice.edu/)
  - Follow the link to “Hardware Initial Setup” (under the Hardware Tutorial section)
  - Complete all tasks (5 minutes)
    - Setting environment variables for tools