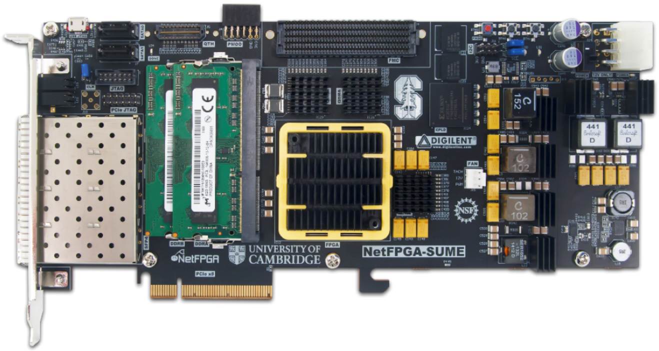
CS344 – Lecture 3
DEVELOPMENT TOOLS

P4 → NetFPGA

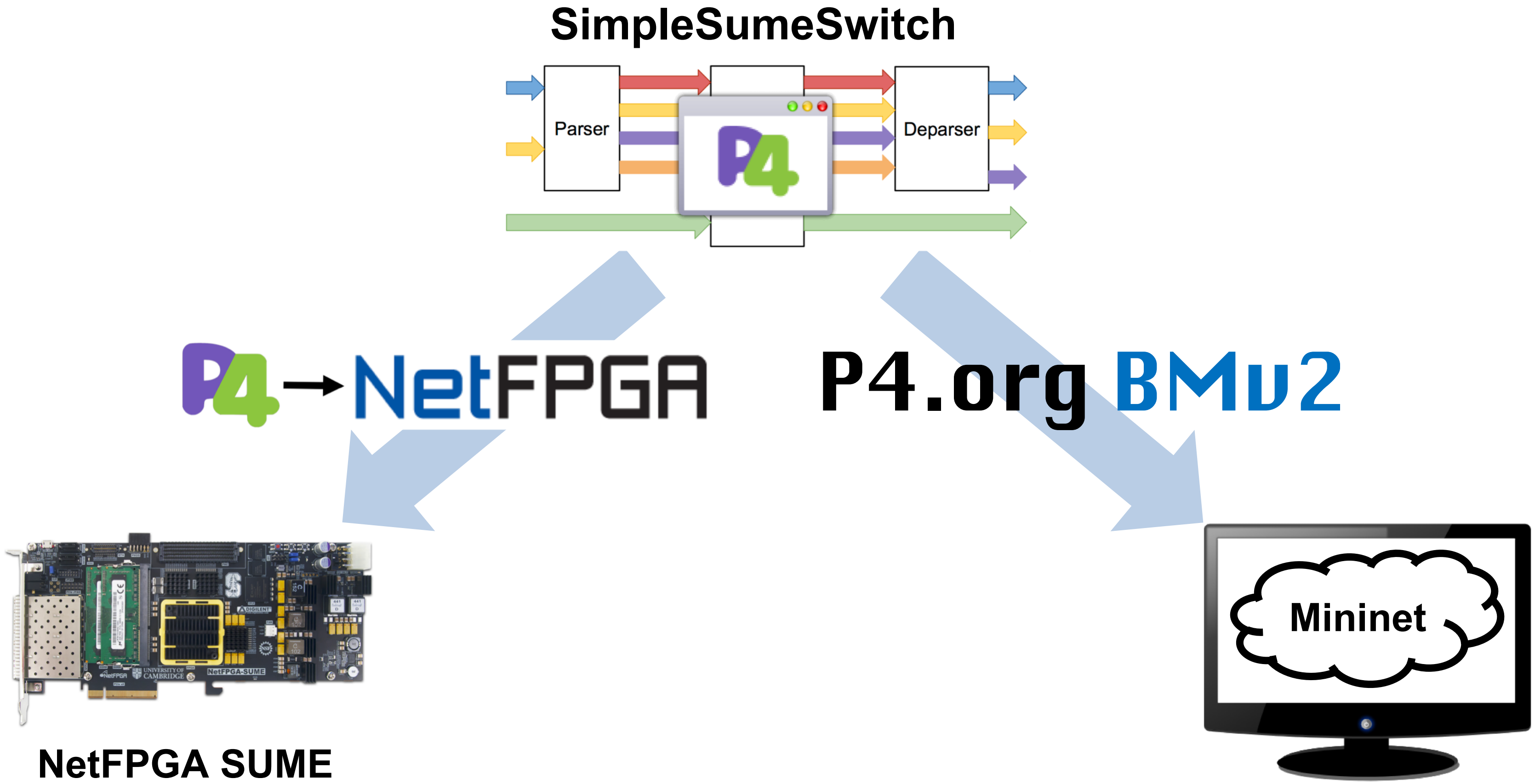


P4.org BMv2

Announcements

- **Office Hours:**
  - Tuesdays & Thursdays: 3 – 5pm, Gates 315
- **Lab Access**
- **Tutorial Exercises (Due Thursday – 4/11 11:59pm)**
- **No Lecture on Wednesday**
  - Tutorial exercises
  - Interoperability planning
  - Router project
- **Guest Lecture on Monday**

Development Tools



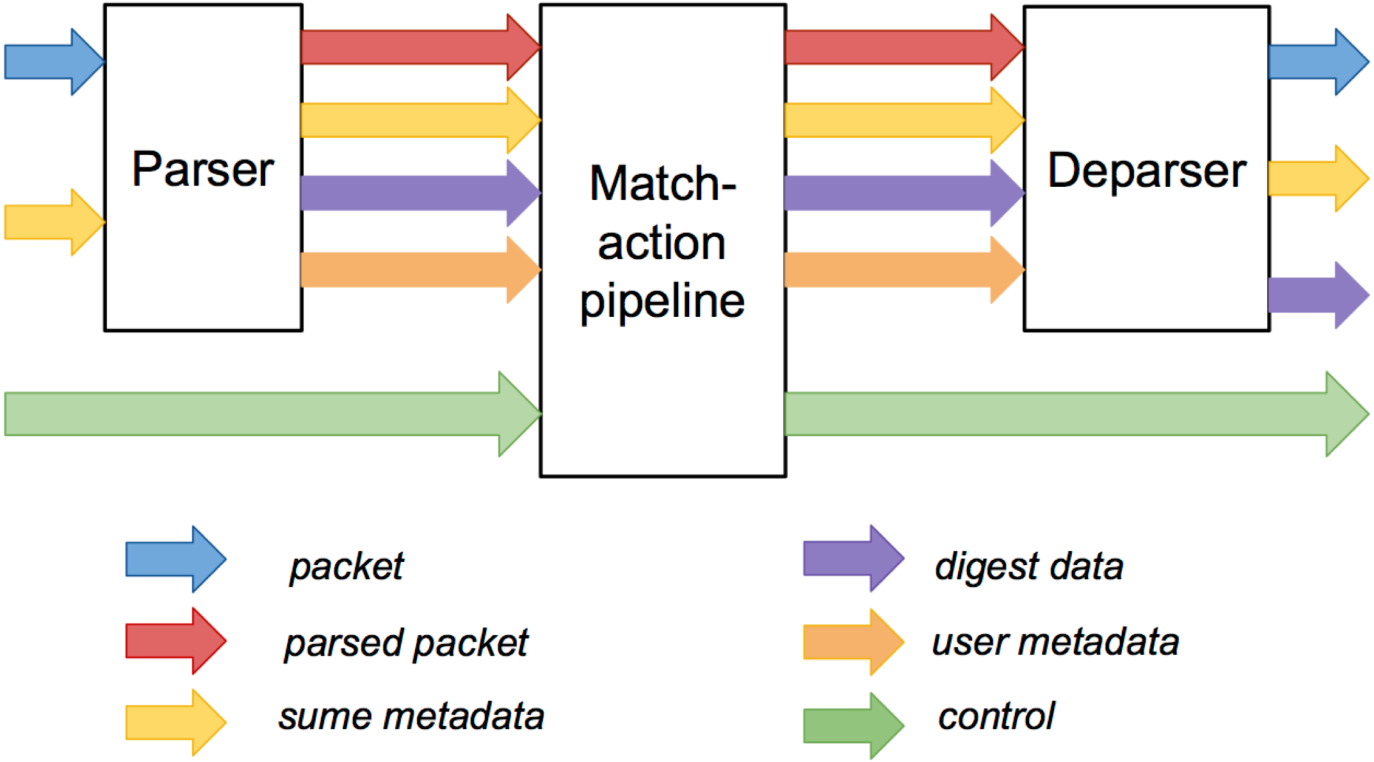
SimpleSumeSwitch

P4 → NetFPGA

P4.org BMv2

NetFPGA SUME

SimpleSumeSwitch Architecture Model for SUME Target



- P4 used to describe parser, match-action pipeline, and deparser

Standard Metadata in SimpleSumeSwitch Architecture

```
/* standard sume switch metadata */
struct sume_metadata_t {
    bit<16> dma_q_size;
    bit<16> nf3_q_size;
    bit<16> nf2_q_size;
    bit<16> nf1_q_size;
    bit<16> nf0_q_size;
    // send digest_data to CPU
    bit<8> send_dig_to_cpu;
    // ports are one-hot encoded
    bit<8> dst_port;
    bit<8> src_port;
    // pkt len is measured in bytes
    bit<16> pkt_len;
}
```

- **`src_port/dst_port` – one-hot encoded, easy to do multicast**
- **`*_q_size` – size of each output queue, measured in terms of 32-byte words, when packet starts being processed by the P4 program**

P4 Parsing

```
parser MyParser(packet_in packet,
                out headers hdr,
                out user_data_t user,
                out digest_data_t digest,
                inout sume_metadata_t smeta)
{

  state start {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.type) {
      0x800   : parse_ipv4;
      default : accept;
    }
  }

  state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
  }

}
```

- **Map packets into headers and metadata**
- **State machine**
- **Three predefined states:**
  - `start`
  - `accept`
  - `reject`
- **User defined states**
- **Loops are OK**

P4 Match-Action Processing

```
control MyIngress(inout headers hdr,
                  inout user_data_t user,
                  inout digest_data_t digest,
                  inout sume_metadata_t smeta) {

  action set_dst_port(bit<8> port) {
    smeta.dst_port = port;
  }

  table forward {
    key = { smeta.src_port : exact; }
    actions = { set_dst_port; }
    size = 1024;
    default_action = set_dst_port(0);
  }

  apply {
    if (hdr.ipv4.isValid()) {
      forward.apply();
    }
    else {
      smeta.dst_port = 0;
    }
  }
}
```

- **Declare tables and actions**
- **Similar to C functions**
- **No loops**
- **Functionality specified by code in apply statement**
- **Table entries populated by control-plane**

forward table entries:

Key	Action Name	Action Data
1	set_dst_port	2
2	set_dst_port	1

P4 Deparsing

```
control MyDeparser(packet_out packet,
                   in headers hdr,
                   in user_data_t user,
                   inout digest_data_t digest,
                   inout sume_metadata_t smeta) {

  apply {
    // insert valid headers into packet
    packet.emit(hdr.ethernet);
    packet.emit(hdr.ipv4);
  }
}
```

- **Assembles the headers back into a well-formed packet**
- **Header is only emitted if it valid**

P4 Externs

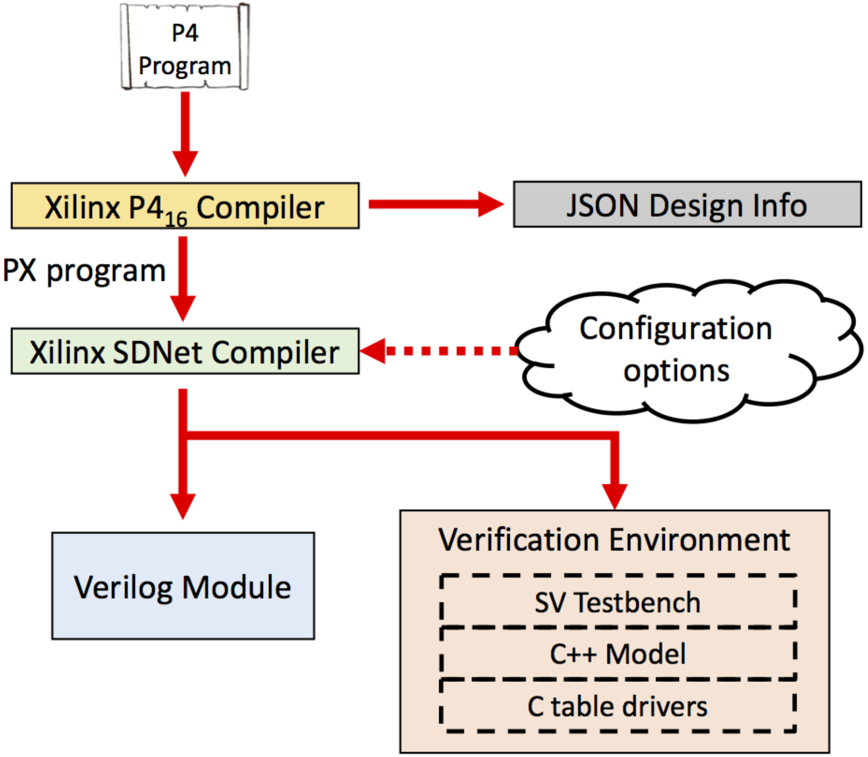
```
// special SUME hash function
extern void sume_hash(in bit<64> data, out bit<8> result);

control MyIngress(inout headers hdr,
                  inout user_data_t user,
                  inout digest_data_t digest,
                  inout sume_metadata_t smeta) {


  apply {
    bit<8> flowID;
    sume_hash(hdr.ip.src++hdr.ip.dst, flowID);
    ...
  }
}
```

- Black boxes for P4 programs
- Functionality is not described in P4
- Used to perform device/vendor specific functionality
- Can be stateless of stateful
- Can be accessed by the control-plane
- Set of supported externs is defined by architecture

Xilinx SDNet Compiler



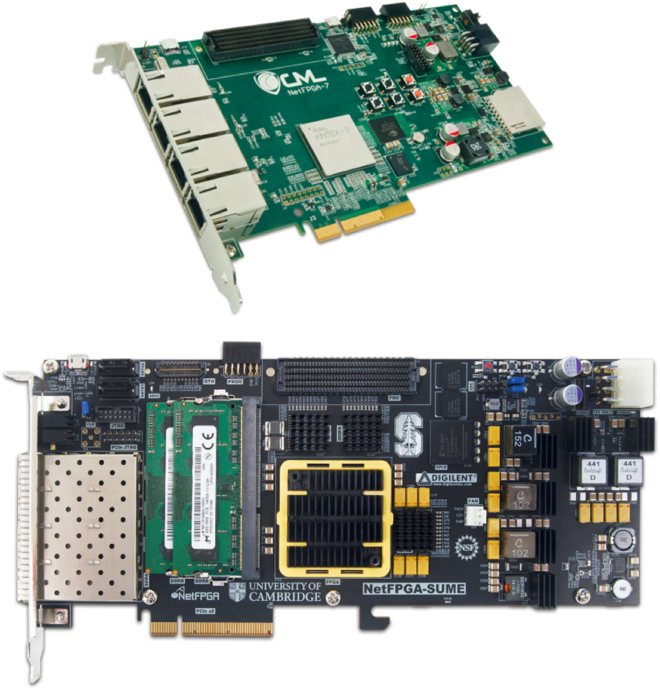
- P4 to PX frontend
- Produces:
  - JSON design info
  - HDL module
  - Verification environment
- Configuration:
  - Throughput (1 – 400 Gbps)
  - Latency
  - Resources

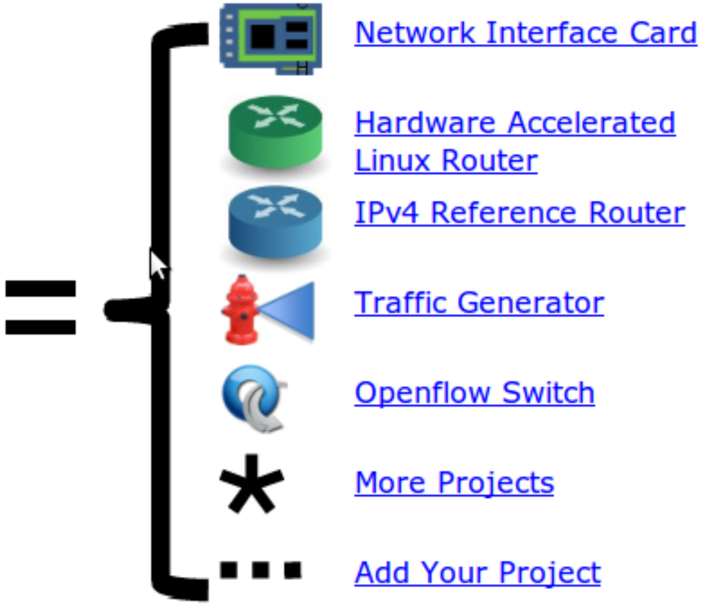
NetFPGA

Overview

NetFPGA = Networked FPGA

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

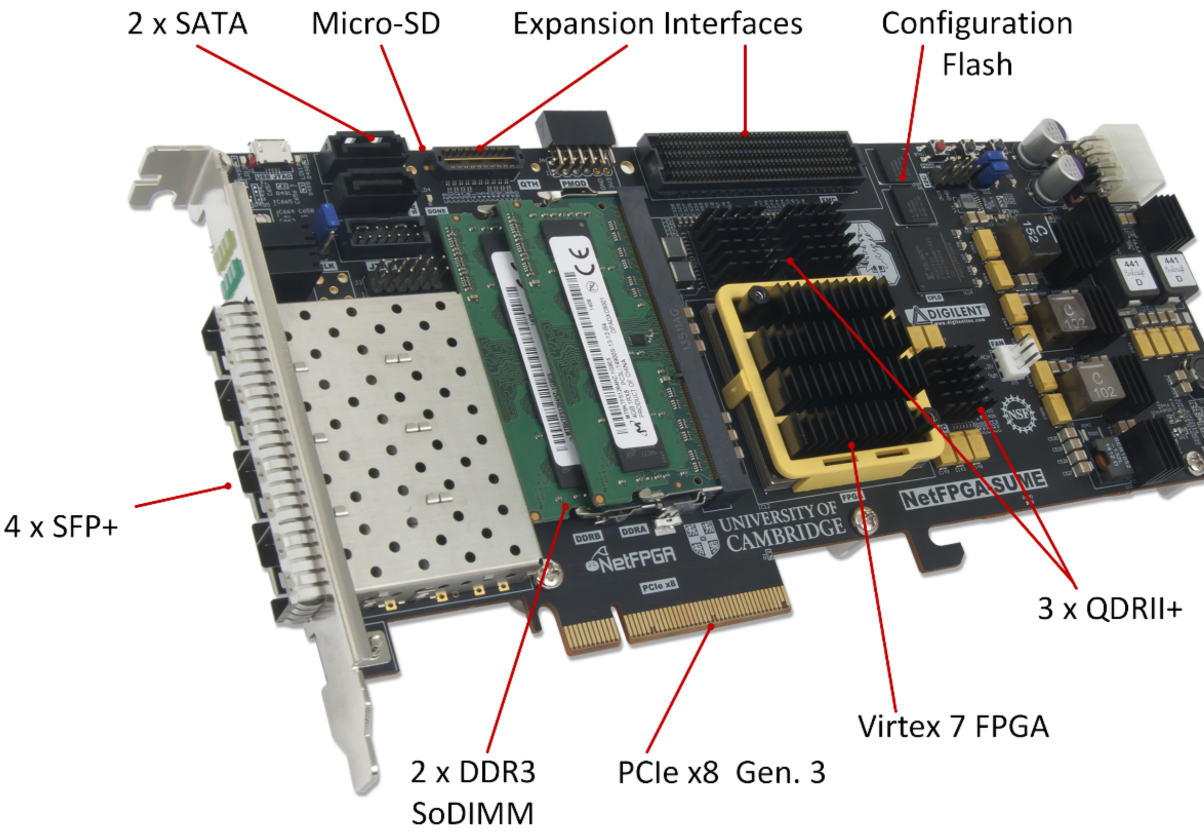




=

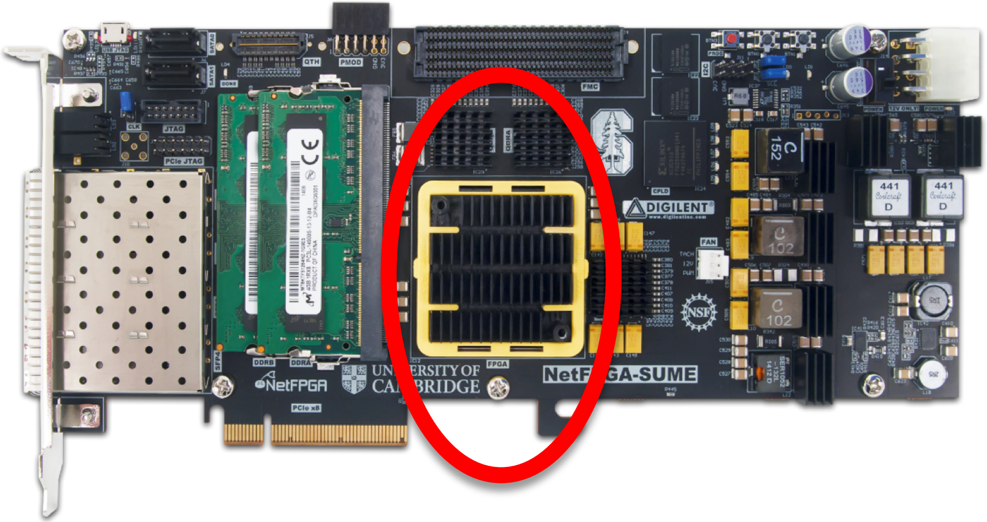
- Network Interface Card
- Hardware Accelerated Linux Router
- IPv4 Reference Router
- Traffic Generator
- Openflow Switch
- More Projects
- Add Your Project

NetFPGA-SUME



Xilinx Virtex 7 690T

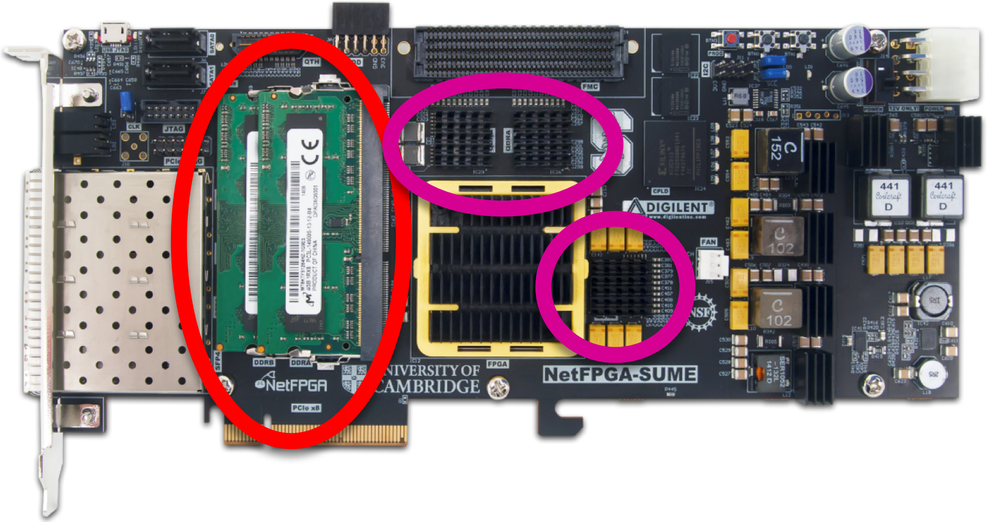
- Optimized for high-performance applications
- 690K Logic Cells
- 52Mb RAM
- 3 PCIe Gen. 3 Hard cores



Memory Interfaces

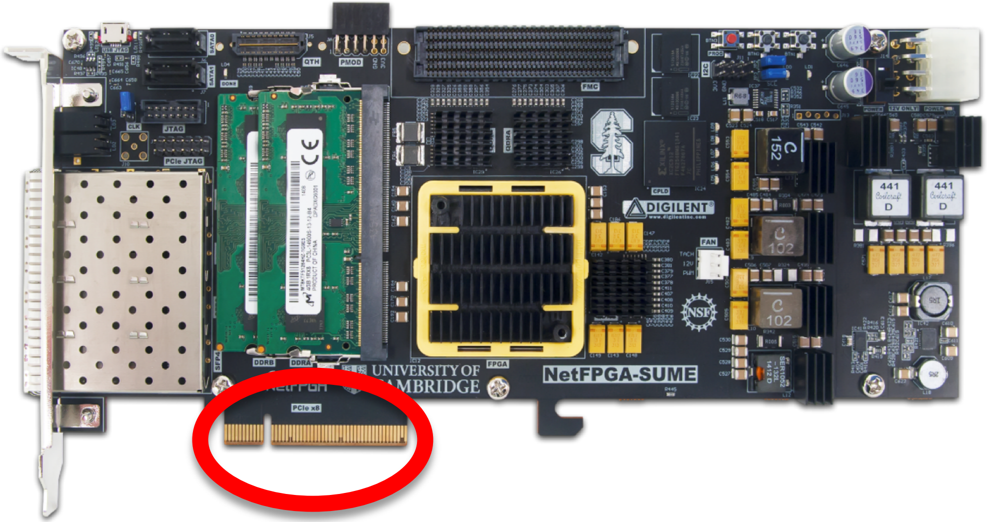
- DRAM:
  2 x DDR3 SoDIMM
  1866MT/s, 4GB

- SRAM:
  3 x 9MB QDRII+, 500MHz



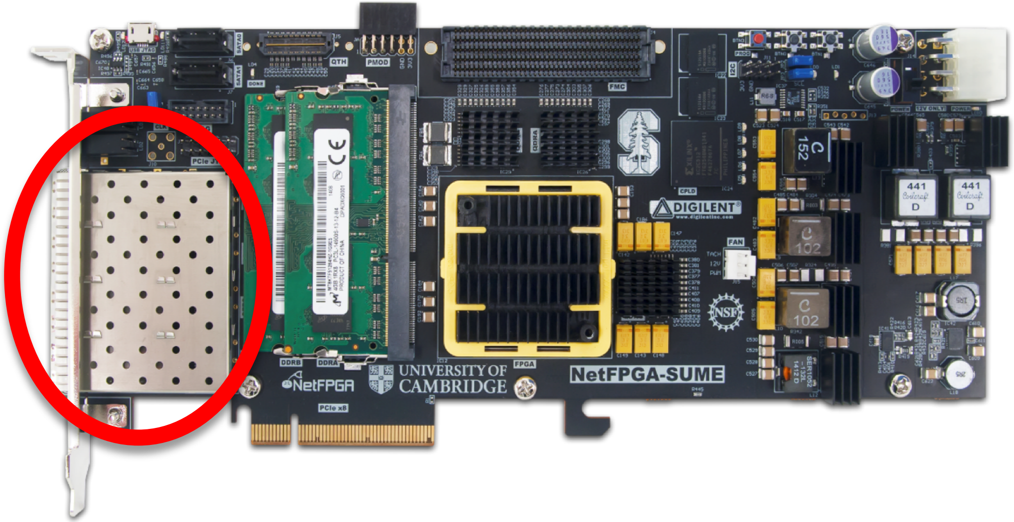
Host Interface

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



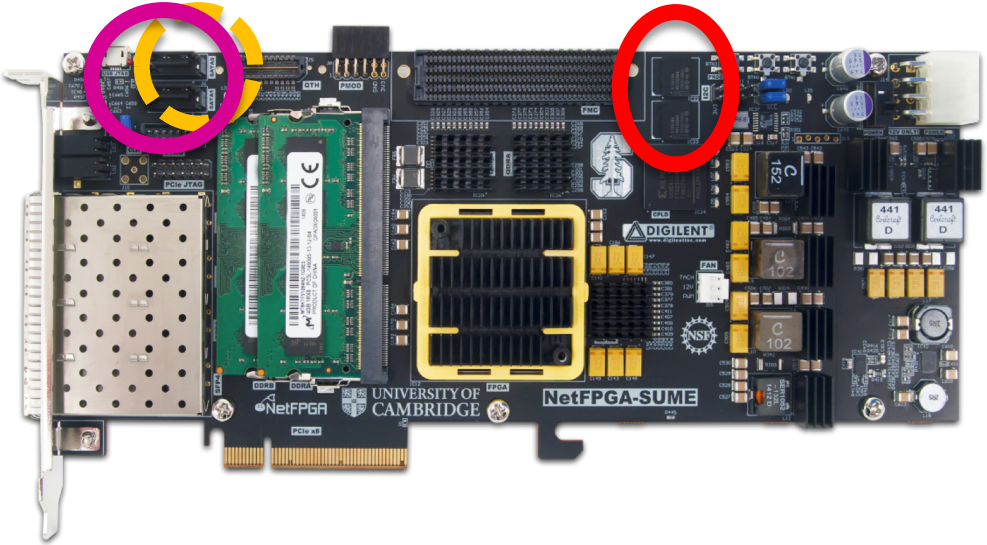
Front Panel Ports

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



Storage

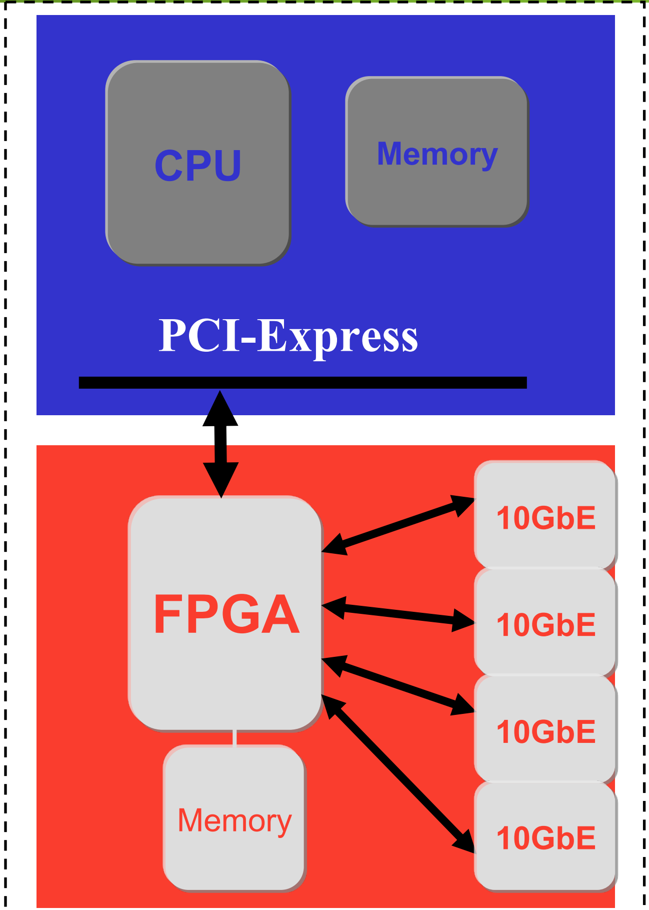
- 128MB FLASH
- 2 x SATA connectors
- Micro-SD slot
- Enable standalone operation



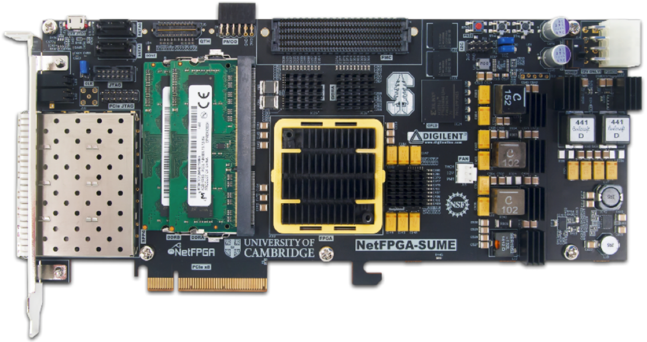
NetFPGA board

Networking Software running on a standard PC

A hardware accelerator built with FPGA driving 1/10/ 100Gb/s network links

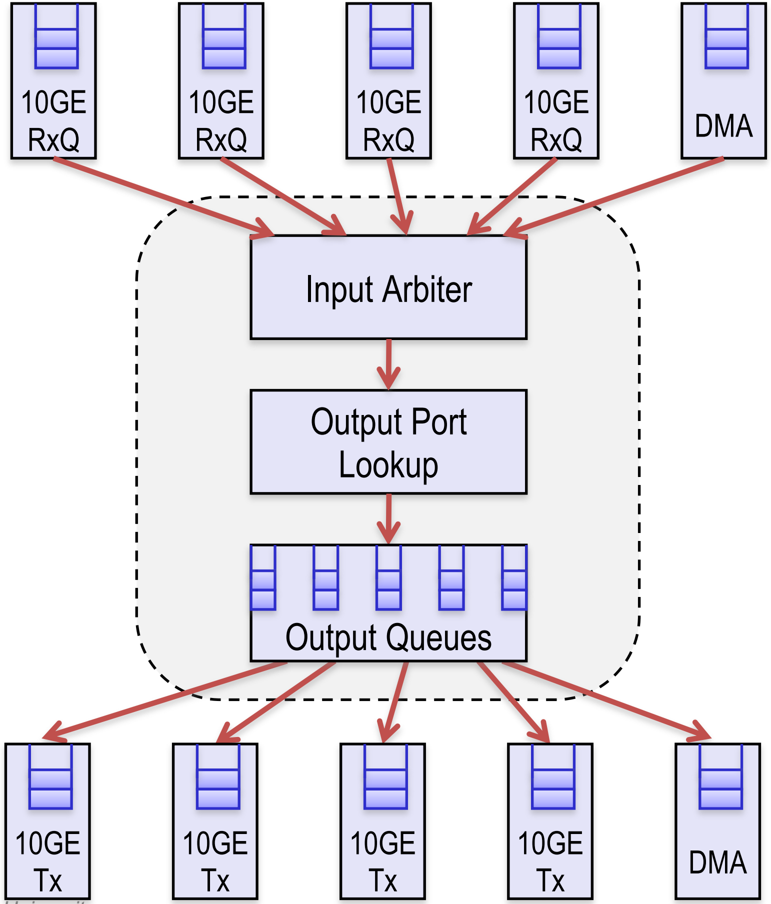


PC with NetFPGA

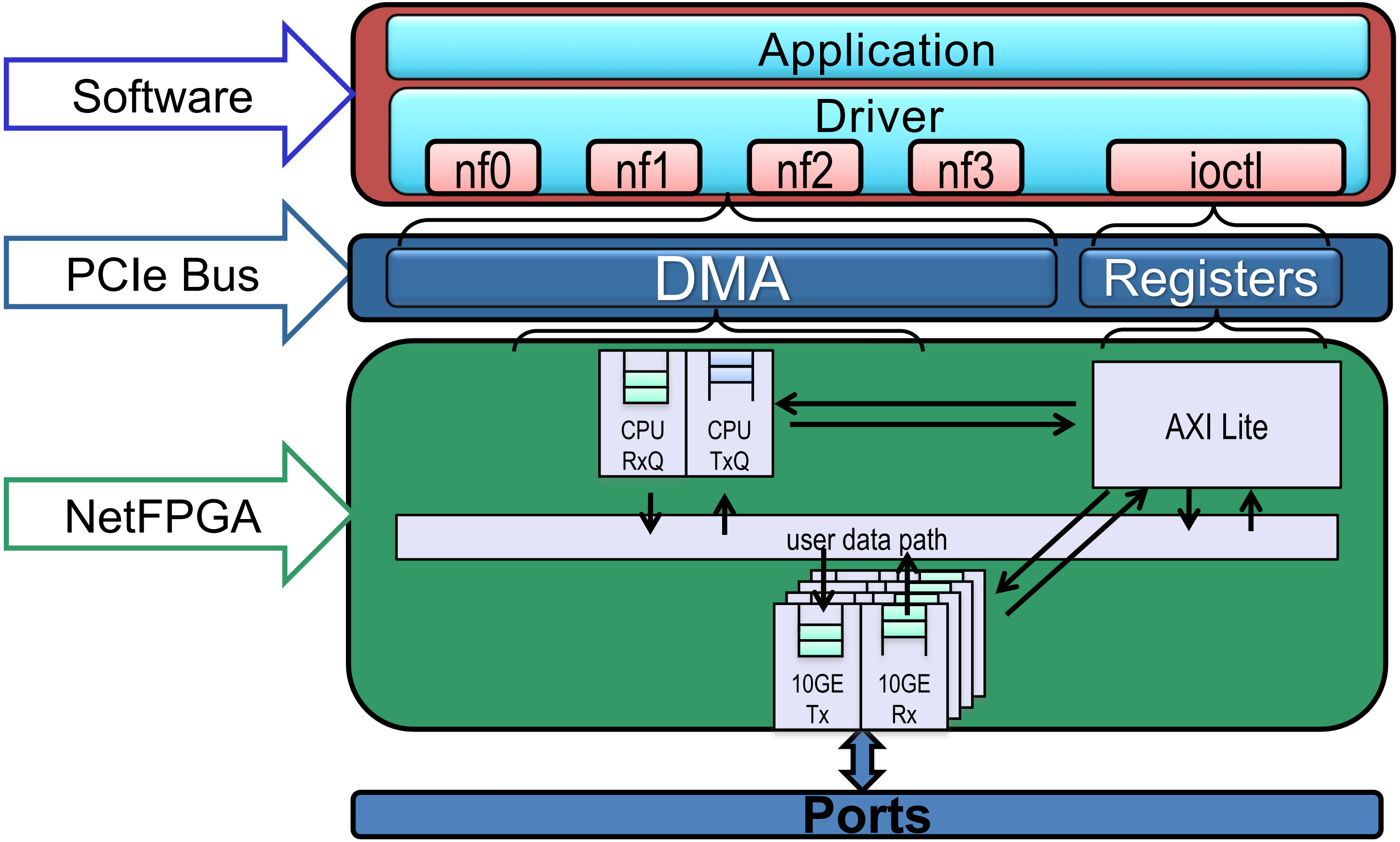


NetFPGA Reference Design

- Five stages
  - Input port
  - Input arbitration
  - Forwarding decision and packet modification
  - Output queuing
  - Output port
- 256-bit data bus
- 200 MHz

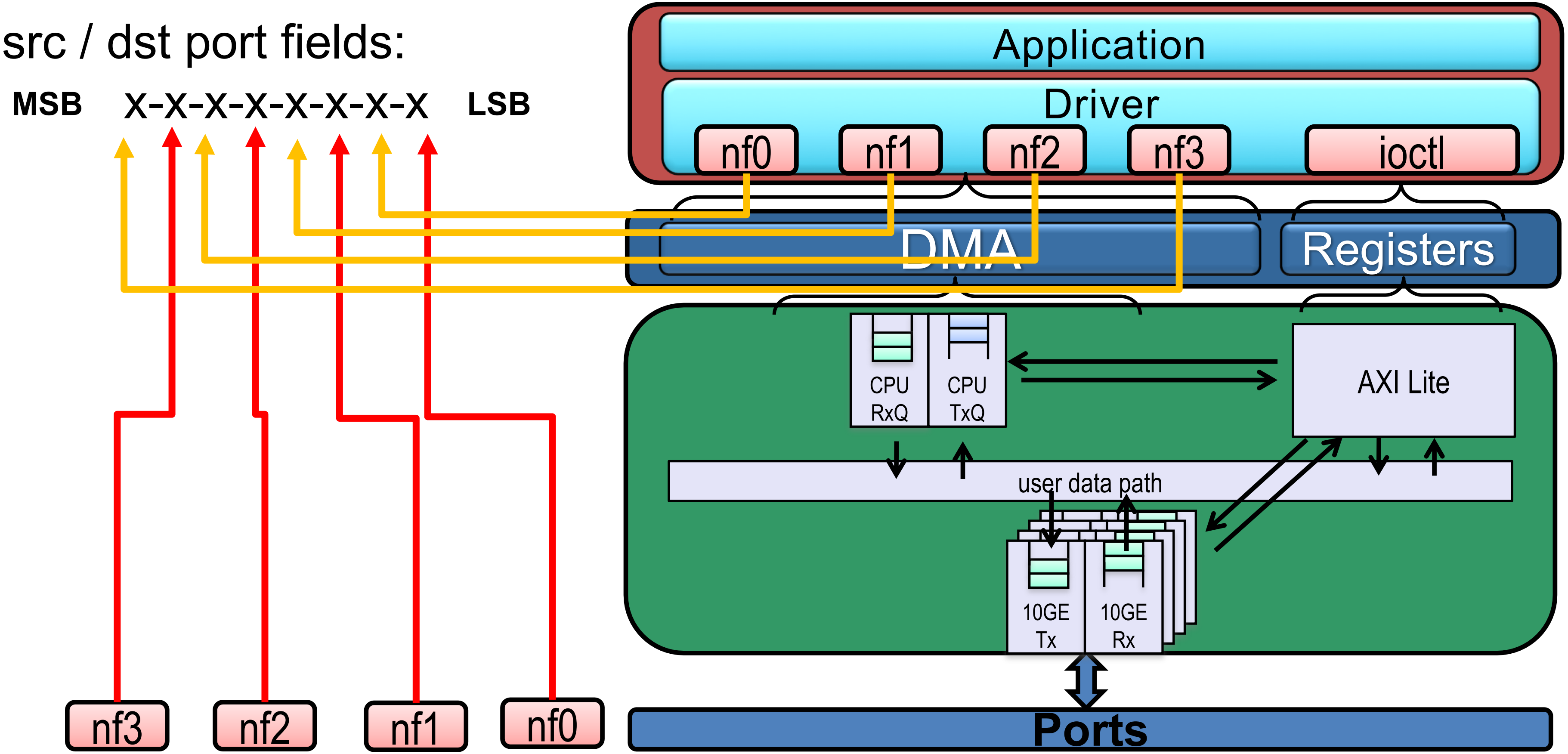


Full System Components



Interface Naming Conventions

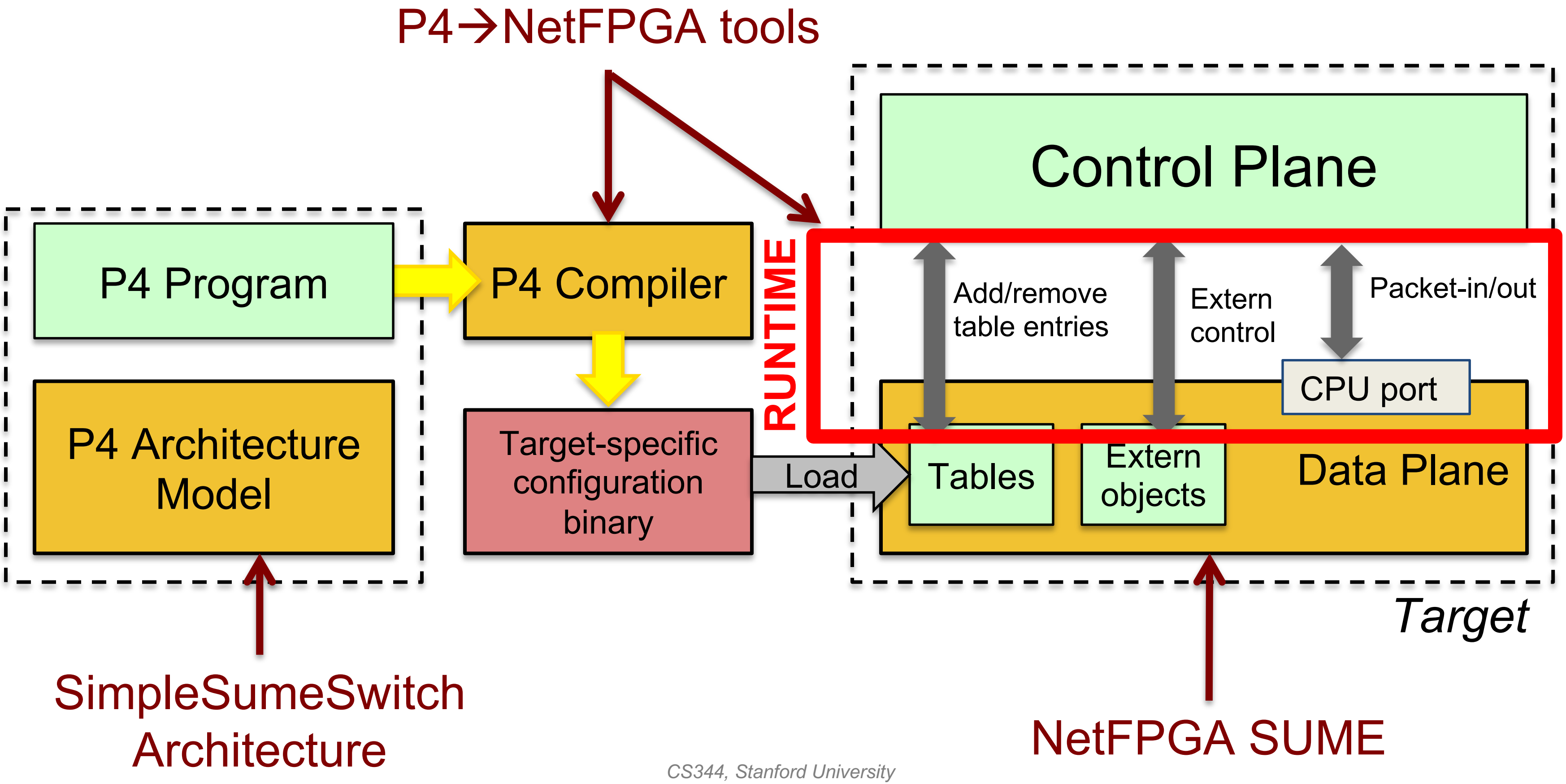
src / dst port fields:



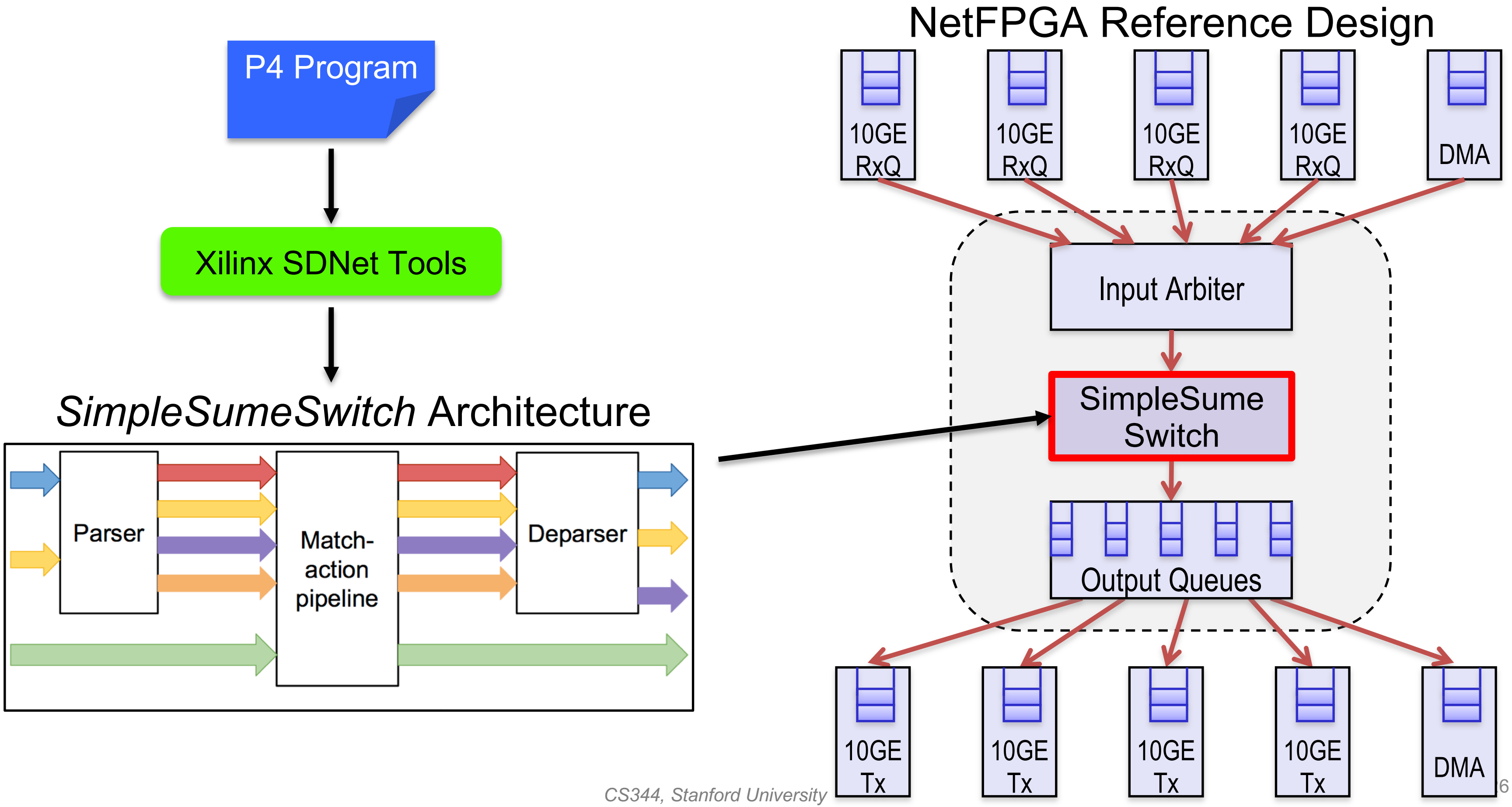
P4 → NetFPGA

Overview

General Process for Programming a P4 Target



P4→NetFPGA Compilation Overview



P4 Program

Xilinx SDNet Tools

SimpleSumeSwitch Architecture

Parser

Match-action pipeline

Deparser

NetFPGA Reference Design

10GE RxQ

10GE RxQ

10GE RxQ

10GE RxQ

DMA

Input Arbiter

SimpleSume Switch

Output Queues

10GE Tx

10GE Tx

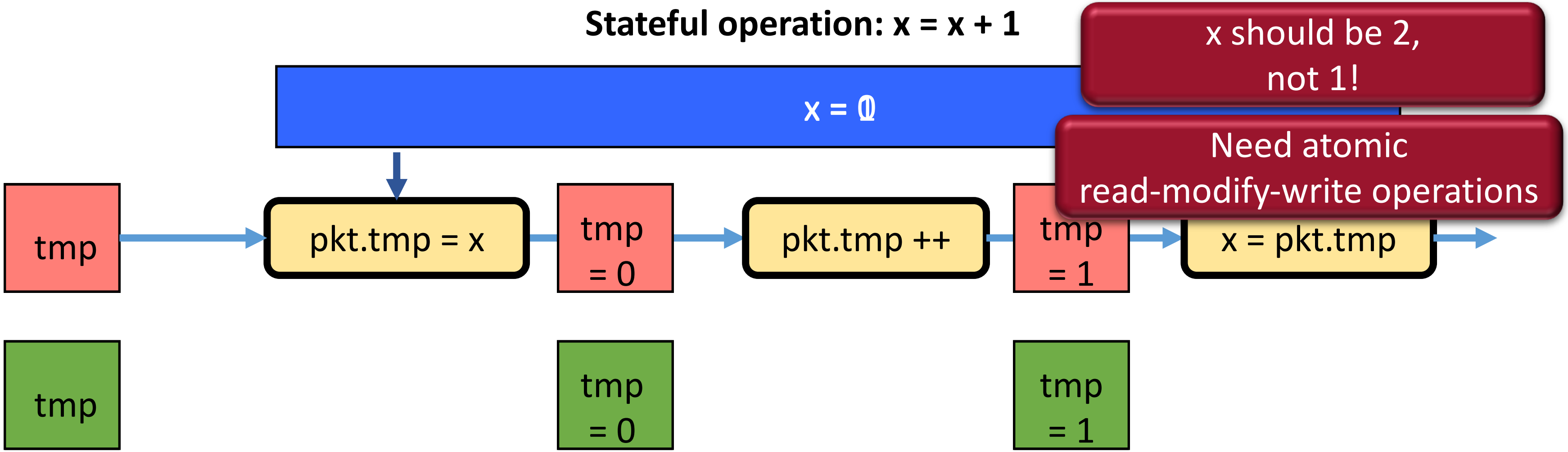
10GE Tx

10GE Tx

DMA

P4→NetFPGA Extern Function library

- HDL modules invoked within P4 programs
- Stateless – reinitialized for each packet
- Stateful – keep state between packets
  - Cannot pipeline stateful operations



Stateful operation: x = x + 1

x should be 2, not 1!

Need atomic read-modify-write operations

Slide Credit: Anirudh Sivaraman

P4→NetFPGA Extern Function library

- **Stateful Atoms[1]**

Atom	Description
R/W	Read or write state
RAW	Read, add to, or overwrite state
PRAW	Predicated version of RAW
ifElseRAW	Two RAWs, one each for when predicate is true or false
Sub	IfElseRAW with stateful subtraction capability

- **Stateless Externs**

Atom	Description
IP Checksum	Given an IP header, compute IP checksum
LRC	Longitudinal redundancy check, simple hash function
timestamp	Generate timestamp (granularity of 5 ns)

- **Add your own!**

[1] Sivaraman, et al. "Packet transactions" *ACM SIGCOMM Conference*, 2016.

Using Atom Externs in P4 – Resetting Counter

Packet processing pseudo code:

```
count[NUM_ENTRIES];

if (pkt.hdr.reset == 1):
    count[pkt.hdr.index] = 0
else:
    count[pkt.hdr.index]++
```

Using Atom Externs in P4 – Resetting Counter

- State can be accessed exactly *1 time*
- Using RAW atom here

```
#define REG_READ     0
#define REG_WRITE    1
#define REG_ADD      2
// count register
@Xilinx_MaxLatency(1)
@Xilinx_ControlWidth(3)
extern void count_reg_raw(
                in bit<3> index, in bit<32> newVal,
                in bit<32> incVal,in bit<8> opCode,
                out bit<32> result);

bit<16> index = pkt.hdr.index;
bit<32> newVal; bit<32> incVal; bit<8> opCode;
if(pkt.hdr.reset == 1) {
    newVal = 0;
    incVal = 0; // not used
    opCode = REG_WRITE;
} else {
    newVal = 0; // not used
    incVal = 1;
    opCode = REG_ADD;
}

bit<32> result; // the new value stored in count reg
count_reg_raw(index, newVal, incVal, opCode, result);
```

Instantiate extern

Set metadata for state access

Single state access!

Adding Custom Extern Functions

1. Implement Verilog extern module in

   `$SUME_SDNET/templates/externs/`

2. Add entry to `$SUME_SDNET/bin/extern_data.py`

- No need to modify any existing code
- AXI Lite control interface

P4→NetFPGA Simulations

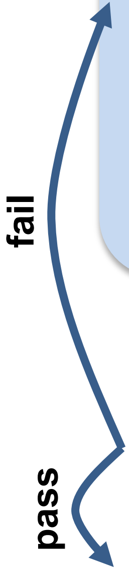
- Python Scapy based script to generate test packets and metadata
- Two stages of simulations:
  - Testbench produced by SDNet compiler
  - Full NetFPGA pipeline simulation

API & Interactive CLI Tool Generation

- Both Python API and C API
  - Manipulate tables and externs in P4 pipeline
  - Used to implement control-plane
- CLI tool
  - Useful debugging feature
  - Query various compile-time information
  - Interact directly with tables and externs at run time

P4→NetFPGA Workflow

1. Write P4 program
2. Write externs
3. Write python gen_testdata.py script
4. Compile to Verilog / generate API & CLI tools
5. Run simulations
6. Build bitstream
7. Check implementation results
8. Test the hardware



All of your effort will go here

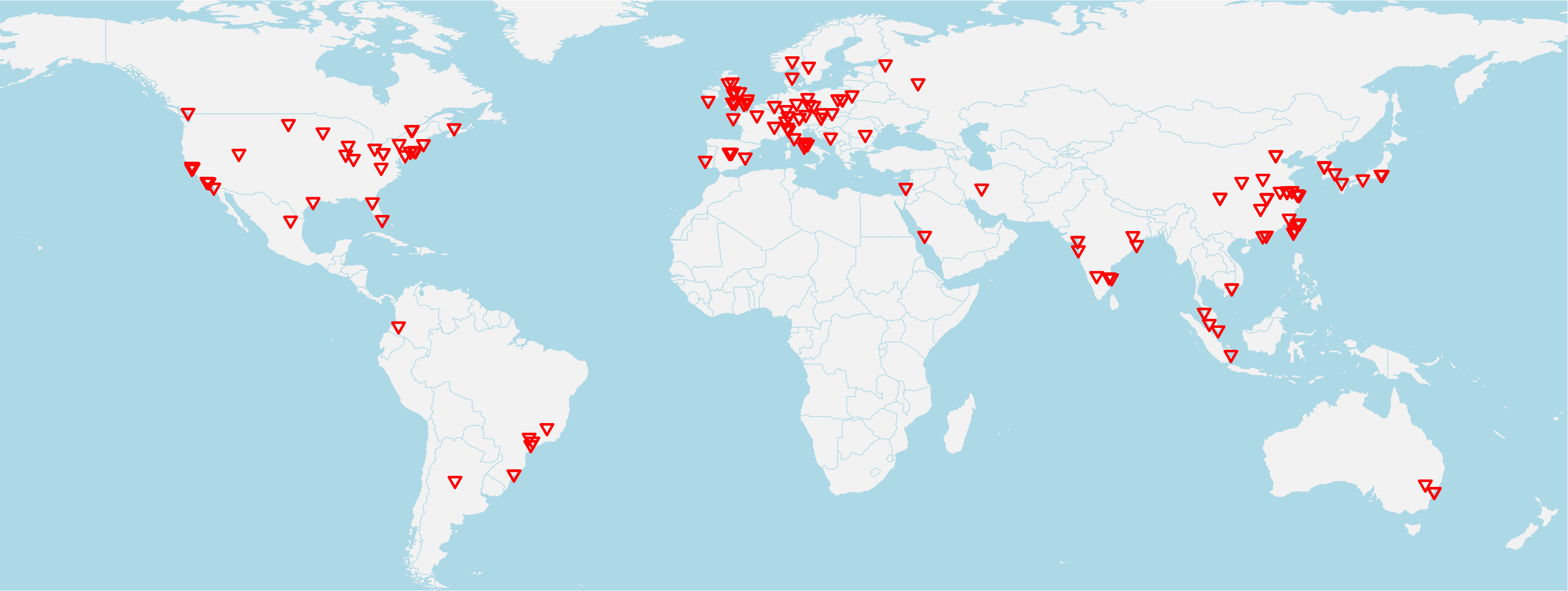
P4→NetFPGA Online Tutorials[1]

- Step-by-step guide
- Solutions available
- Three assignments:
  - Switch as calculator
  - TCP flow monitor
  - In-band Network Telemetry (INT)

[1] https://github.com/NetFPGA/P4-NetFPGA-public/wiki/Tutorial-Assignments

P4→NetFPGA Community

- 150 different institutions
- Mailing list: cl-netfpga-sume-beta@lists.cam.ac.uk



Debugging P4 Programs

- **SDNet HDL simulation**
- **SDNet C++ simulation**
  - Verbose packet processing info
  - Output PCAP file
- **Full SUME HDL simulation**

Directory Structure of $SUME_FOLDER

```
P4-NetFPGA-live/
 |
 |- contrib-projects/
 |   |- sume-sdnet-switch/ → the main directory for P4 dev
 |
 |- lib/ → contains all of the SUME IP cores
 |
 |- tools/ → various NetFPGA scripts for test infra.
 |
 |- Makefile → builds all of the SUME IP cores
```

Directory Structure of $SUME_SDNET

```
sume-sdnet-switch/
 |
 |- bin/ → scripts used to automate workflow
 |
 |- templates/ → templates for externs, wrapper module,
 |               CLI tools, new projects
 |
 |- projects/ → all of the P4 project directories
 |   |- switch_calc/
```

Directory Structure of $P4_PROJECT_DIR

```
$P4_PROJECT_DIR/
 |
 |- src/ → P4 source files and commands.txt
 |
 |- testdata/ → scripts to generate testdata used for
 |                verifying functionality of P4 program
 |
 |- simple_sume_switch/ → main SUME project directory,
 |                top level HDL files and SUME sim scripts
 |
 |- sw/ → populated with API files and CLI tools and any
 |          user software for the project
 |
 |- nf_sume_sdnet_ip/ → SDNet output directory
```

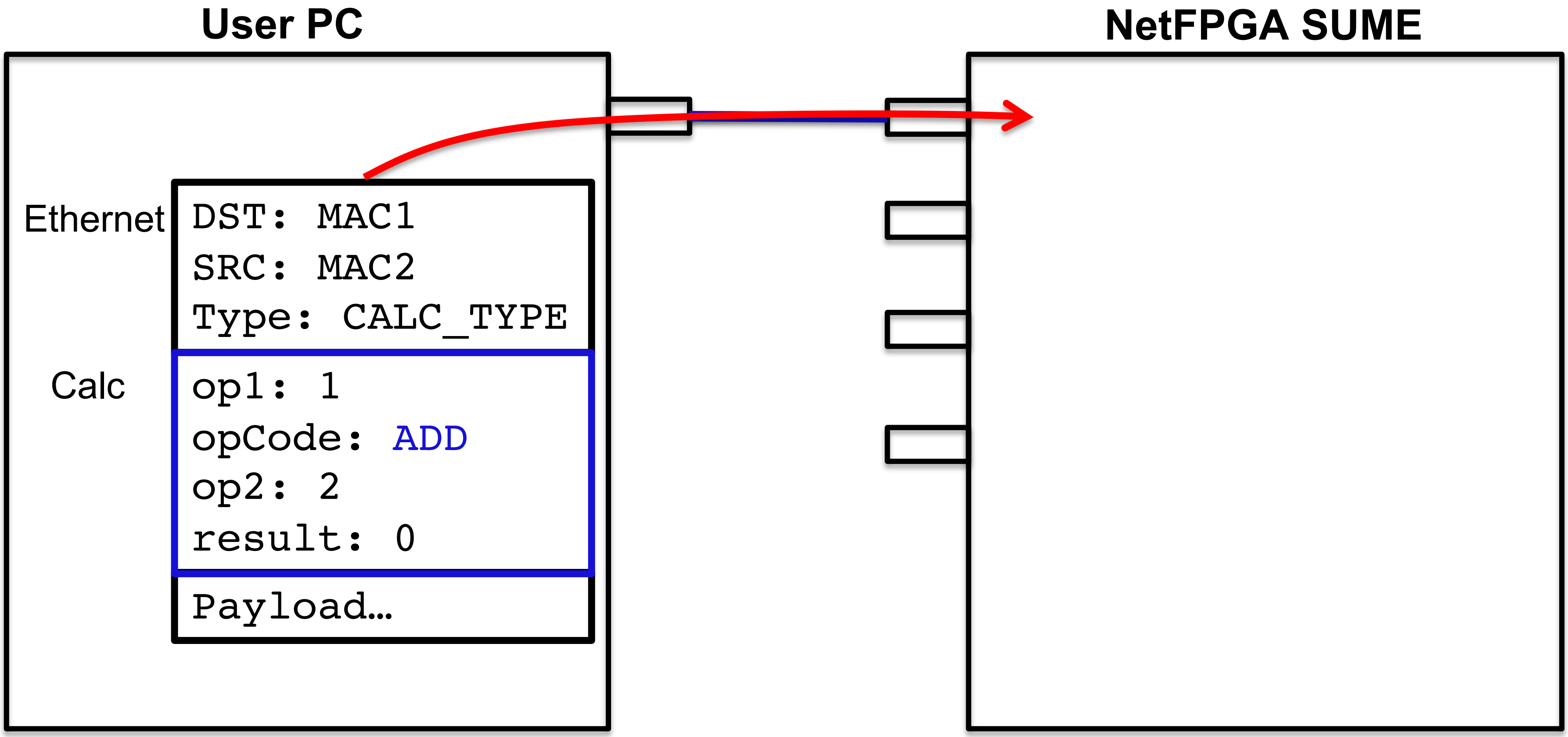
Assignment 1: Switch as a Calculator

Switch as a Calculator

- **Supported Operations**
  - ADD – add two operands
  - SUBTRACT – subtract two operands
  - ADD_REG – add operand to current value in the register
  - SET_REG – overwrite the current value in the register
  - LOOKUP – Lookup the given key in the table

```
header Calc_h {
    bit<32> op1;
    bit<8> opCode;
    bit<32> op2;
    bit<32> result;
}
```

Switch as a Calculator

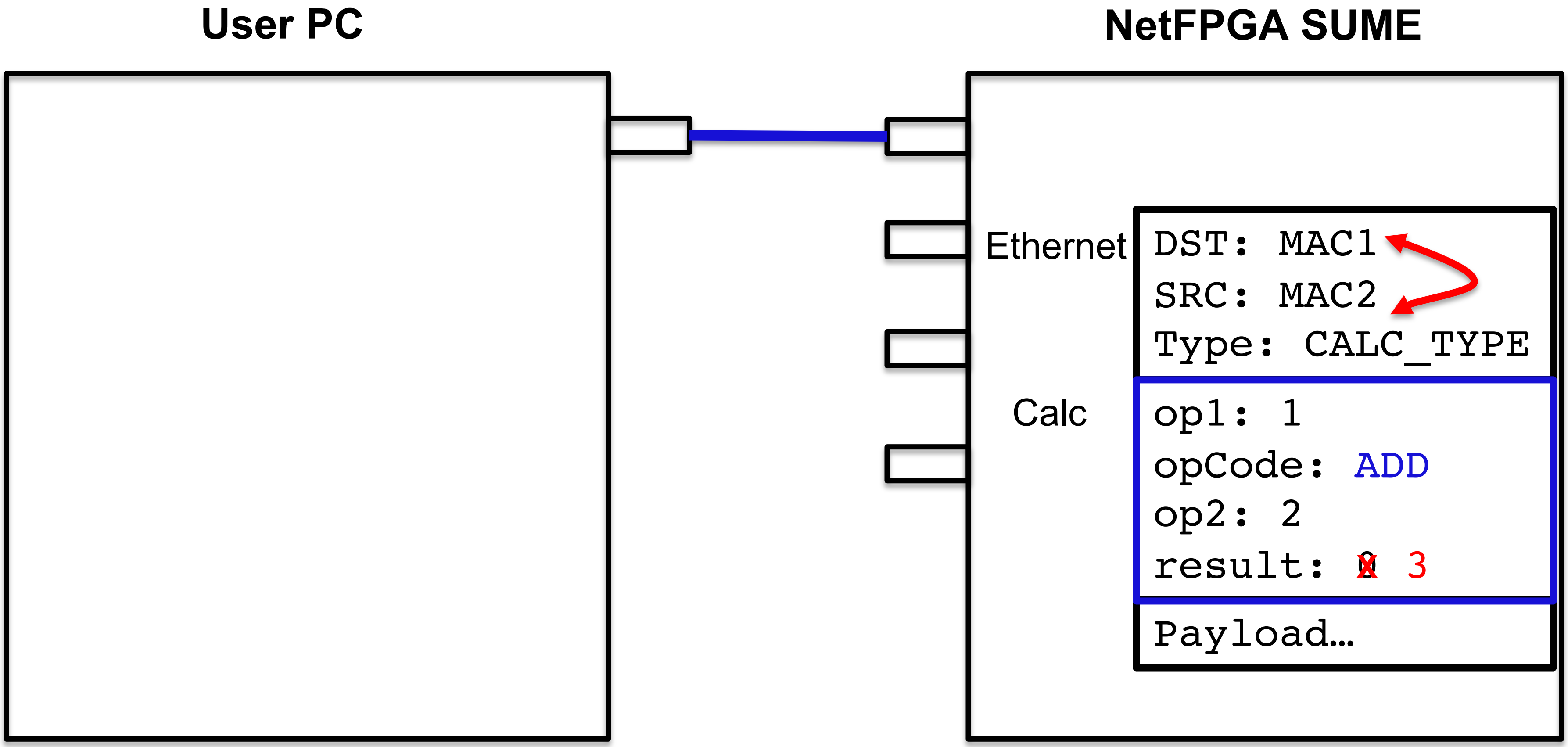


DST: MAC1
SRC: MAC2
Type: CALC_TYPE

op1: 1
opCode: ADD
op2: 2
result: 0

Payload…

Switch as a Calculator

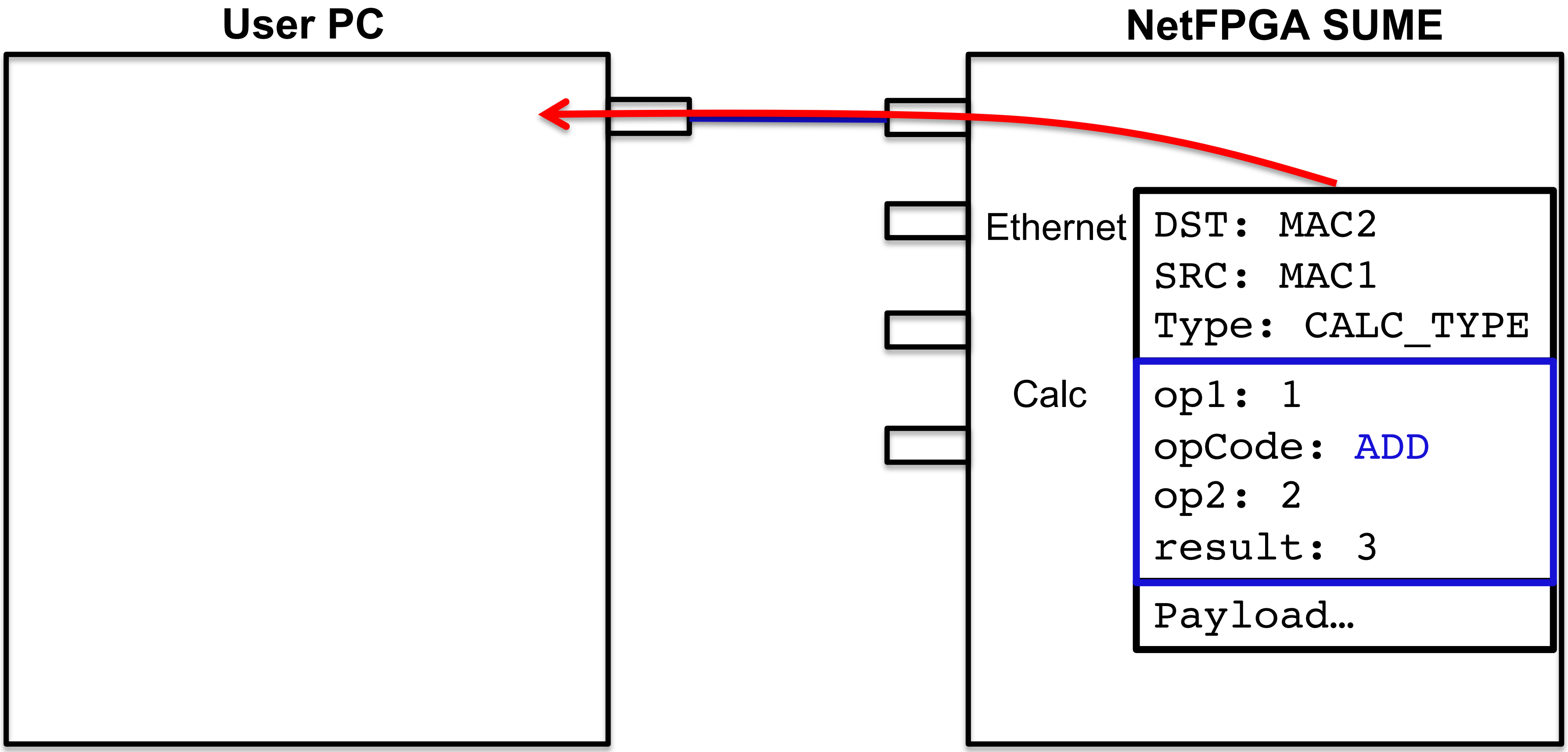


DST: MAC1
SRC: MAC2
Type: CALC_TYPE

op1: 1
opCode: ADD
op2: 2
result: ~~0~~ 3

Payload…

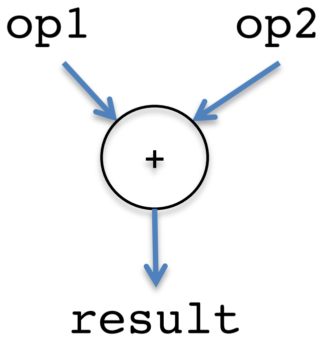
Switch as a Calculator



DST: MAC2
SRC: MAC1
Type: CALC_TYPE
op1: 1
opCode: ADD
op2: 2
result: 3
Payload…

Switch Calc Operations

ADD

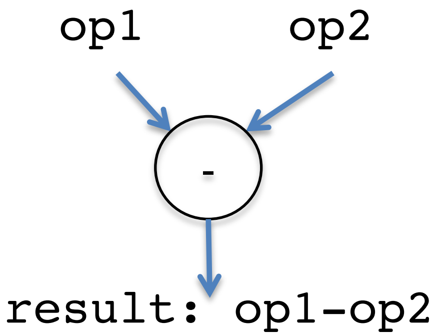


op1 op2

+

result

SUB

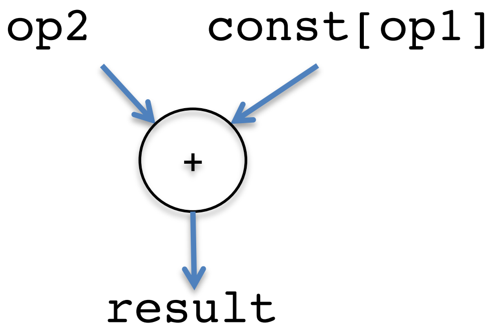


op1 op2

-

result: op1-op2

ADD_REG

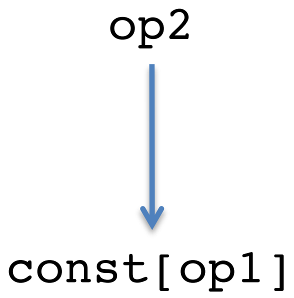


op2 const[op1]

+

result

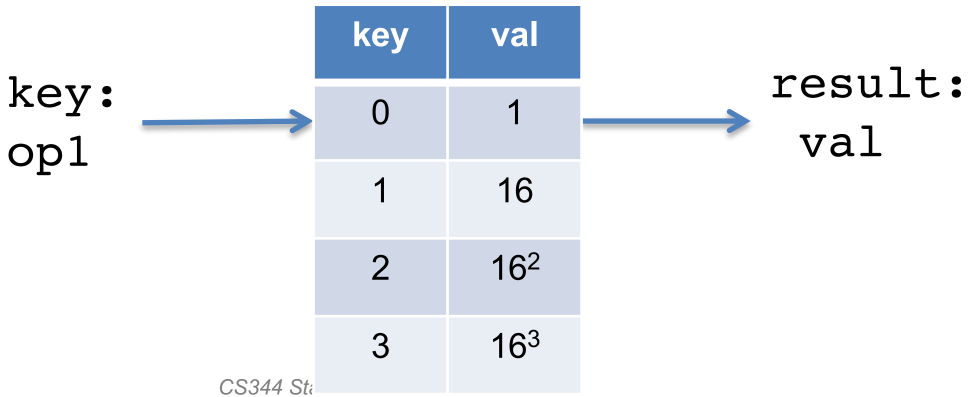
SET_REG



op2

const[op1]

LOOKUP



key: op1

key	val
0	1
1	16
2	16^2
3	16^3

result: val

P4.org BMv2 Mininet Emulation

Virtual Machine

- **Dependencies:**
  - Bmv2
  - p4c
  - Mininet

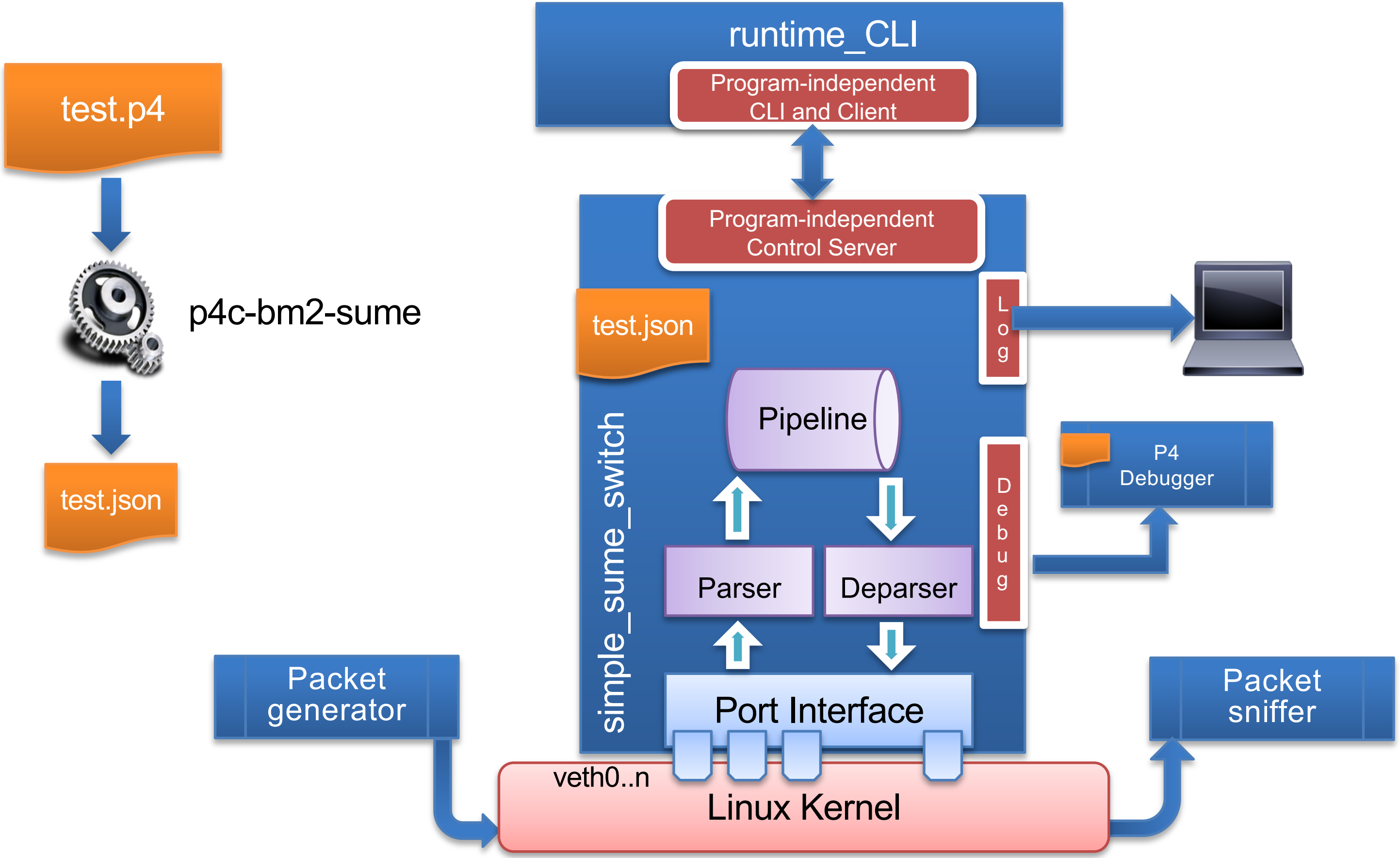
- **Getting Set Up:**
  - Install Vagrant and VirtualBox
  - `$ git clone -b si/skt/SimpleSumeSwitch https://github.com/CS344-Stanford/tutorials.git`
  - `$ cd tutorials/vm`
  - `$ vagrant up`

Behavioral Model v2

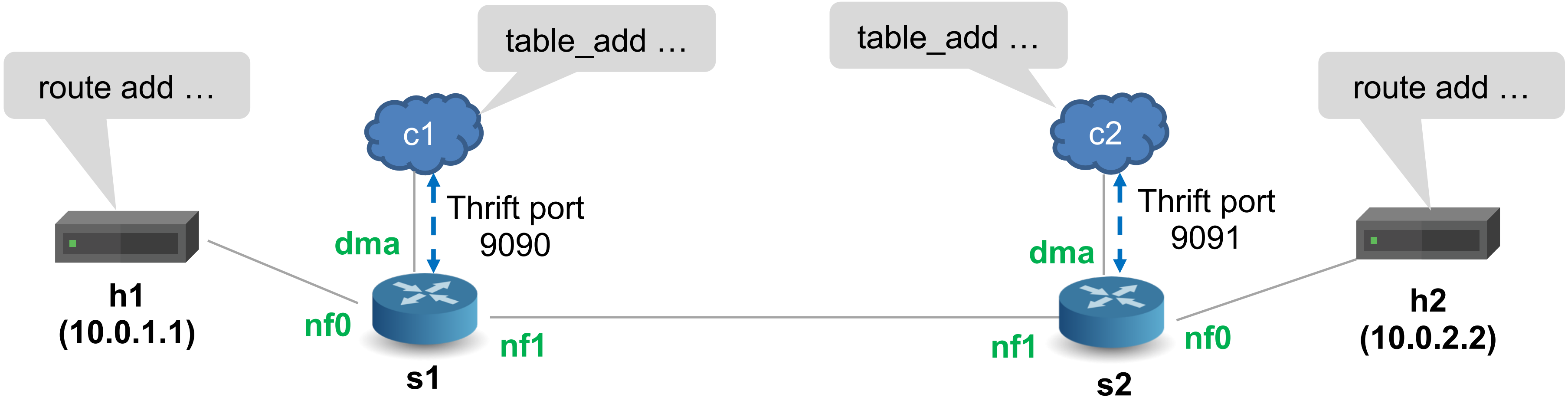
- **C++ packet processing library for P4 programs**

	Mininet **(Network Emulator)**	**NS3** **(Network Simulator)**
Supported Architectures	• V1Model • SimpleSumeSwitch	• P4QueueDisc

Behavioral Model v2



Mininet Topology



```
{
    "hosts": {
        "h1": {"ip": "10.0.1.1/24", "mac": "08:00:00:00:01:01",
               "commands":["route add default gw 10.0.1.10 dev eth0"]},
        "h2": {"ip": ”10.0.2.2/24", "mac": "08:00:00:00:02:02",
               "commands":["route add default gw 10.0.2.20 dev eth0"]}
    },
    "switches": { "s1": {"cli_input": "s1-commands.txt"},
                  "s2": {"cli_input": "s2-commands.txt"} },
    "links": [ ["h1", "s1-nf0"], ["s1-nf1", "s2-nf1"], ["h2", "s2-nf0"] ]
}
```

Working with Tables in runtime_CLI

```
RuntimeCmd: show_tables
m_filter                       [meta.meter_tag(exact, 32)]
m_table                        [ethernet.srcAddr(ternary, 48)]

RuntimeCmd: table_info m_table
m_table                        [ethernet.srcAddr(ternary, 48)]
********************************************************************************
_nop
[]m_action                       [meter_idx(32)]

RuntimeCmd: table_dump m_table
m_table:
0: aaaaaaaaaaaa &&& ffffffffffff => m_action - 0,
SUCCESS

RuntimeCmd: table_add m_table m_action 01:00:00:00:00:00&&&01:00:00:00:00:00 => 1 0
Adding entry to ternary match table m_table
match key:           TERNARY-01:00:00:00:00:00 &&& 01:00:00:00:00:00
action:              m_action
runtime data:        00:00:00:05
SUCCESS
entry has been added with handle 1
```

Value and mask for ternary matching. No spaces around "&&&"

Entry priority

"=>" separates the key from the action data

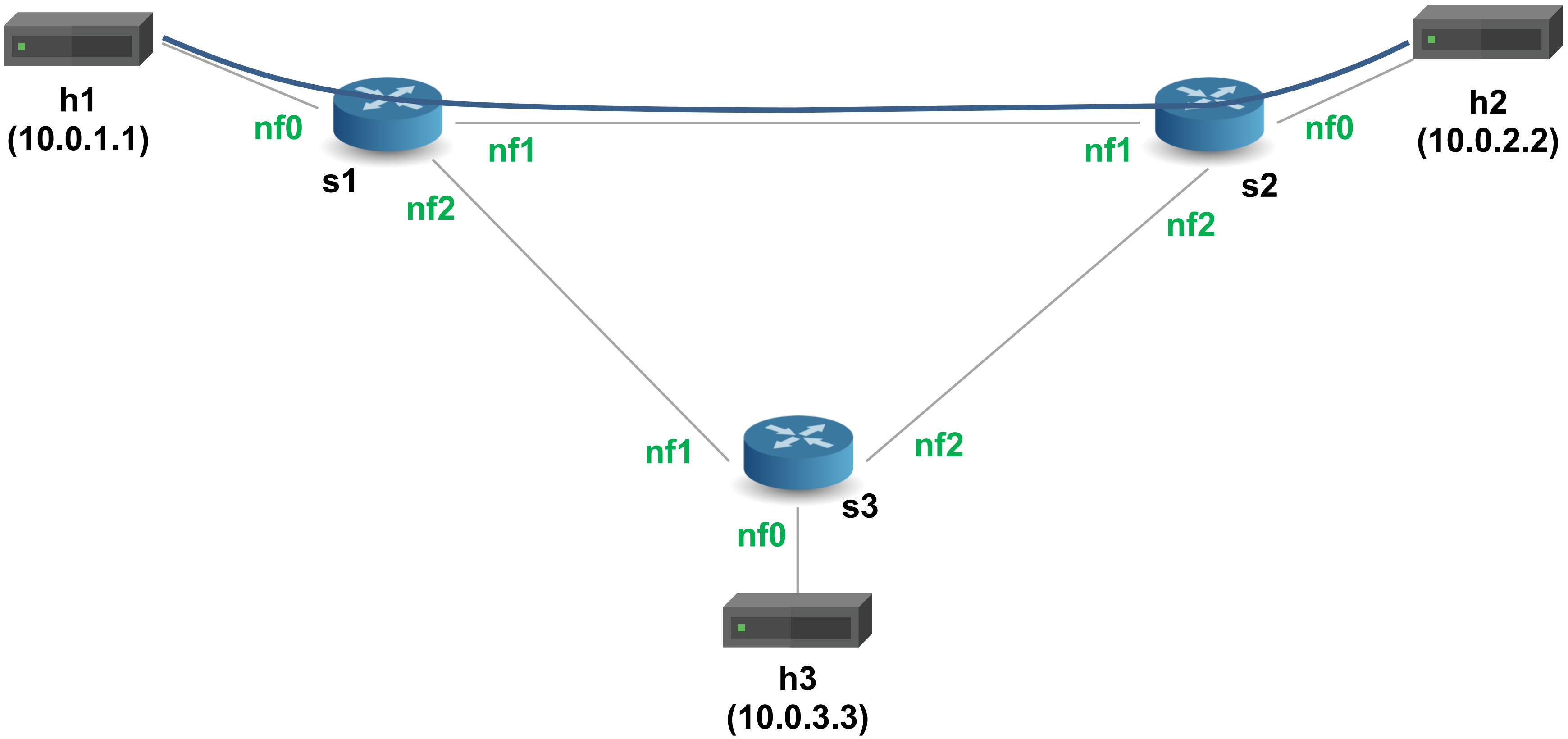
SimpleSumeSwitch Support in bmv2

- No extern support
- No data-plane broadcasting
- No `digest_data` support
- Different CLI commands from P4→NetFPGA

Basic Exercise

- **Basic Router Functionality:**
  - Parse Ethernet and IPv4 headers
  - Find destination in IPv4 routing table
  - Update source / destination MAC addresses
  - Decrement time-to-live (TTL) field
  - Update IPv4 checksum
  - Set egress port
  - Deparse header back into packet
- **Starter code in `basic.p4`**
- **Static control-plane**

Basic Exercise



FIN