And, what about debugging for multi-FPGA systems?

Paul Chow

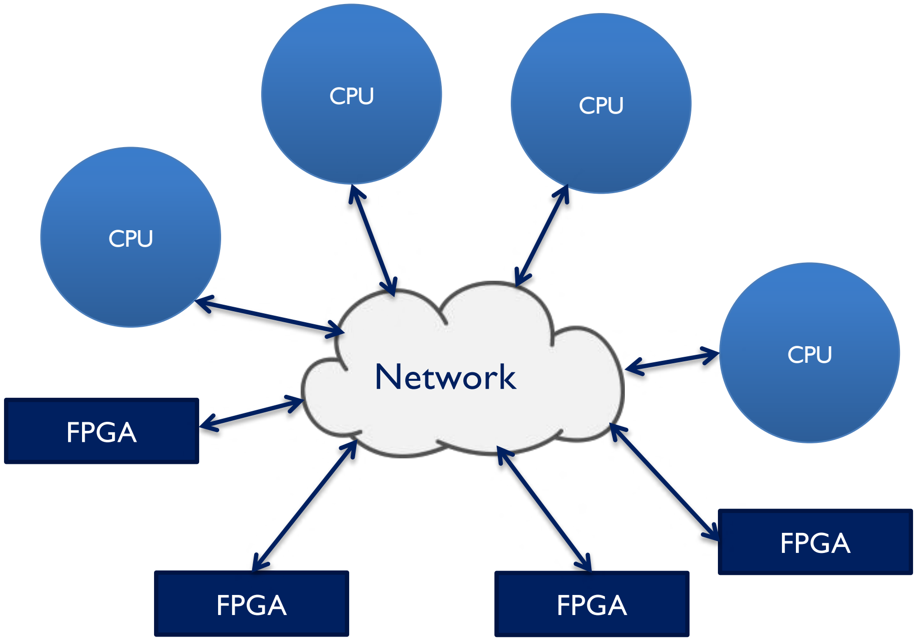
High-Performance Reconfigurable Computing Group
University of Toronto

Outline

- Context
- What we've been thinking
- The Debug Governor
- FFShark
- Pharos
- Simulation
- Final thoughts

CONTEXT

How do you debug something running on this?

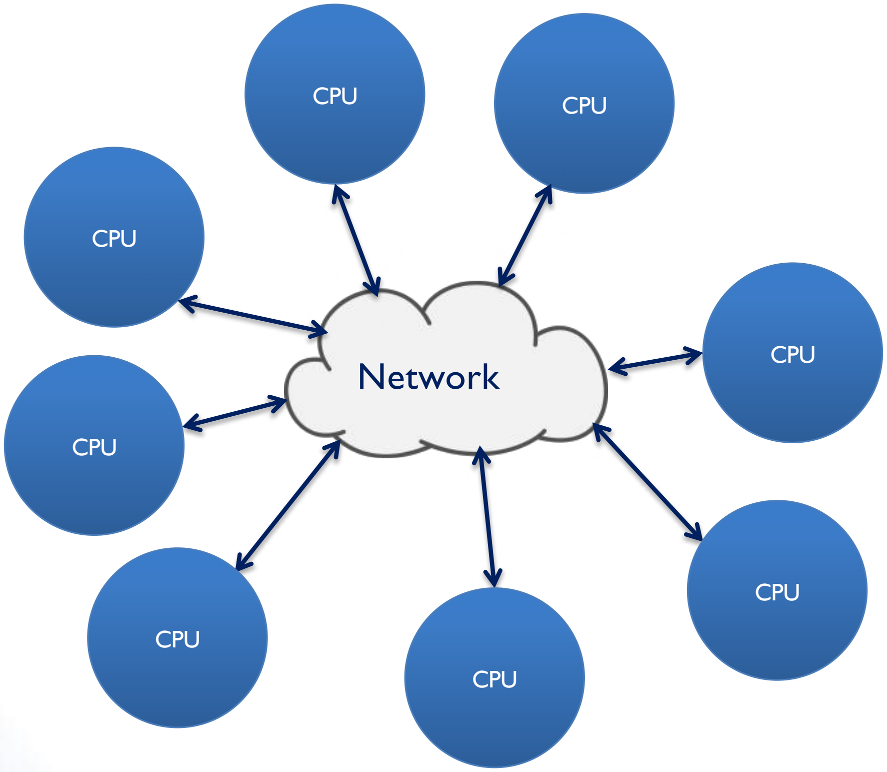


Network-connected heterogeneous platform

- Scalable and elastic

Examples include: Microsoft, cloudFPGA, UofT Galapagos

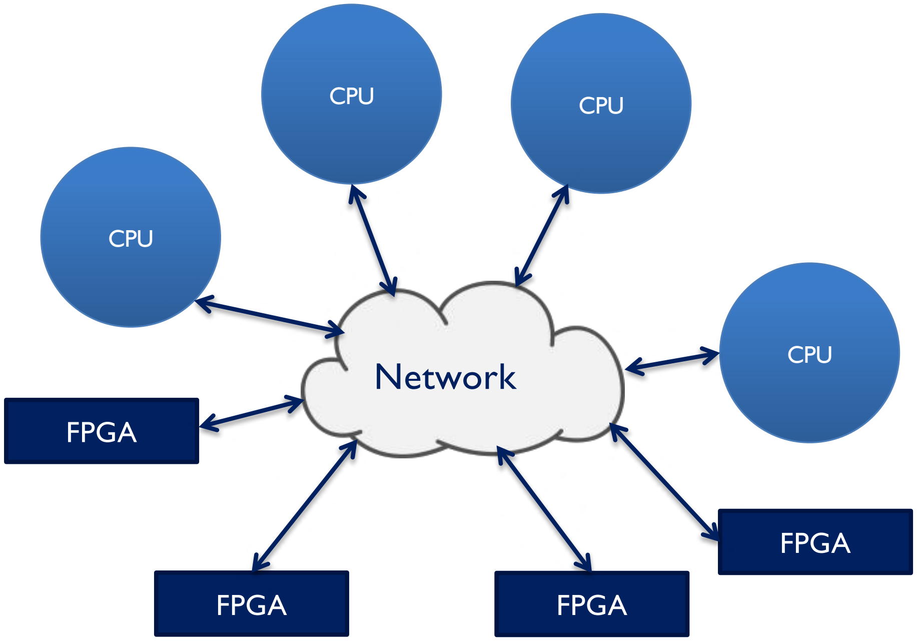
A bit (actually, a lot) harder than this



Typical multiprocessor platform

- Often MPI-based, many tools
- And it's software

Back to this



- There are no (standard) tools
- If anything exists, they are custom, one-off, and work on only one platform
- Catapult Flight Data Recorder records interesting state

We need to build an ecosystem of tools that can evolve and be used across multiple heterogeneous platforms

WHAT WE'VE BEEN THINKING

Ideas

- Compare software and hardware environments
- What does software do?
- High and low-level debugging
- Wireshark
- Logging
- Performance debugging

Distributed System Debugging

Software	Hardware
Highly observable	Difficult to observe
Logging "in series" with program code	Logging "in parallel" (zero runtime cost)
Underlying OS	Underlying "shell"
Breakpoints (maybe)	No breakpoints
Some consistent structure	No consistent structure

Software

- Many new techniques being developed and applied to large software systems
  - Dynamic invariant detection, model inference, declarative specifications
- Hardware doesn’t seem ready for that

HIGH AND LOW-LEVEL DEBUGGING

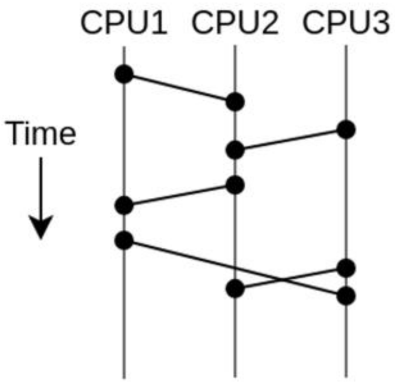
Definition

- Have multiple nodes
- Strategy
  - High-level – Which node exhibits the problem?
  - Low-level – Debug on that node

High-Level Debugging

- Log "API calls" instead of single instructions
- Goal: find location to apply low-level debugger

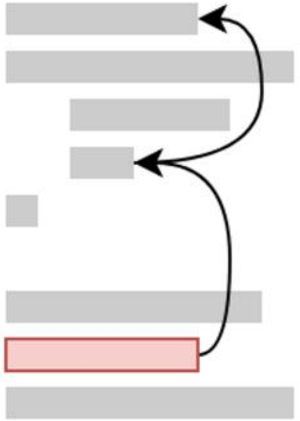
Visualizations



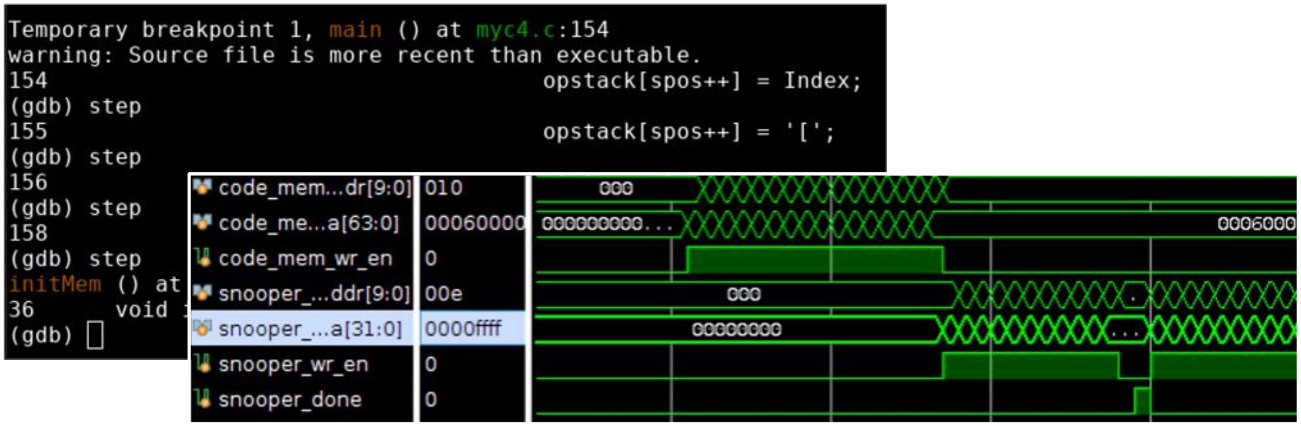
Performance Debugging

	# Calls	Avg. Lat.
fn1 ()	102	10 µs
fn2 ()	4	5 ms
fn3 ()	760	70 µs

Failure Reproduction



Low-Level Debugging



```
Temporary breakpoint 1, main () at myc4.c:154
warning: Source file is more recent than executable.
154                                     opstack[spos++] = Index;
(gdb) step
155                                     opstack[spos++] = '[';
(gdb) step
156
(gdb) step
158
(gdb) step
initMem () at
36        void
(gdb) 
```

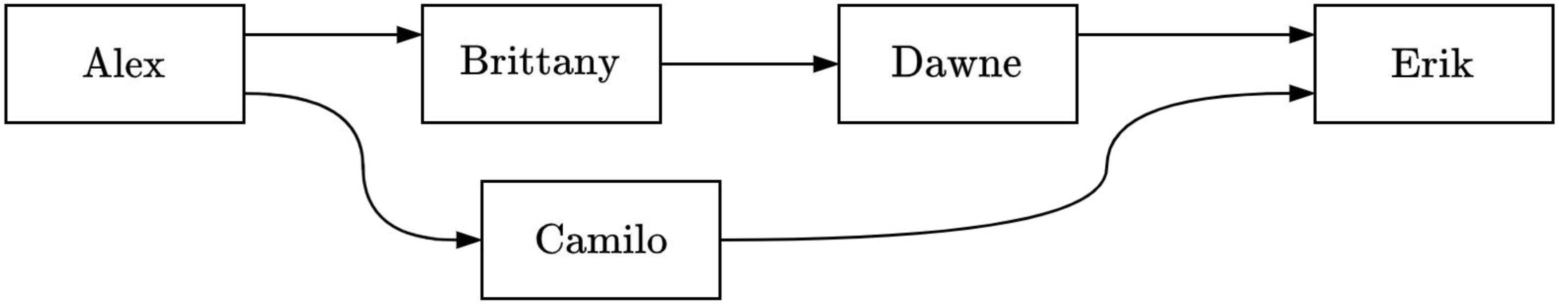
```
code_mem...dr[9:0]   010        000
code_me...a[63:0]    00060000   000000000...                0006000
code_mem_wr_en       0
snooper_...ddr[9:0]  00e                  000
snooper_...a[31:0]   0000ffff             00000000
snooper_wr_en        0
snooper_done         0
```

- Software single-stepping, watching state, etc.
- Hardware – SignalTap, ILA
- In a large project, "scrubbing a building with a toothbrush"

ASSUMPTIONS AND OBSERVATIONS

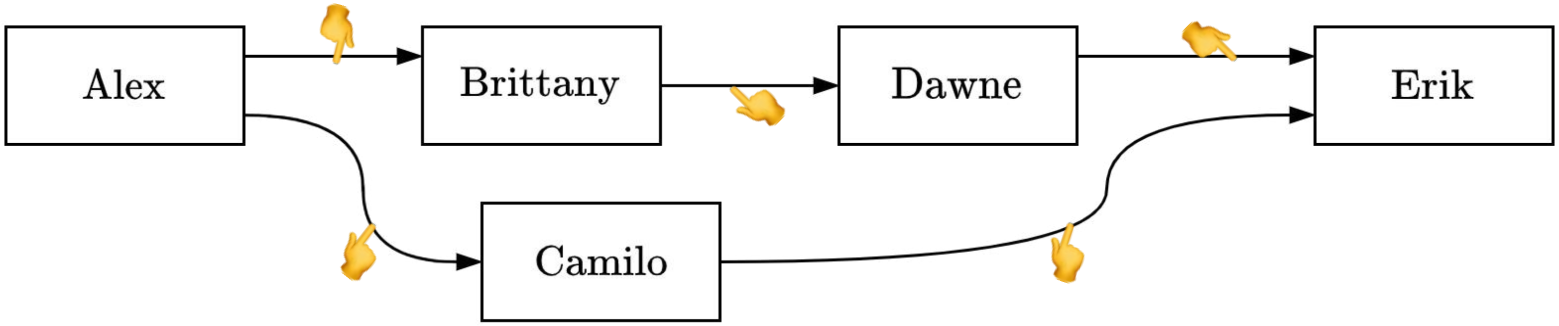
Observation

Many designs are dataflow graphs:

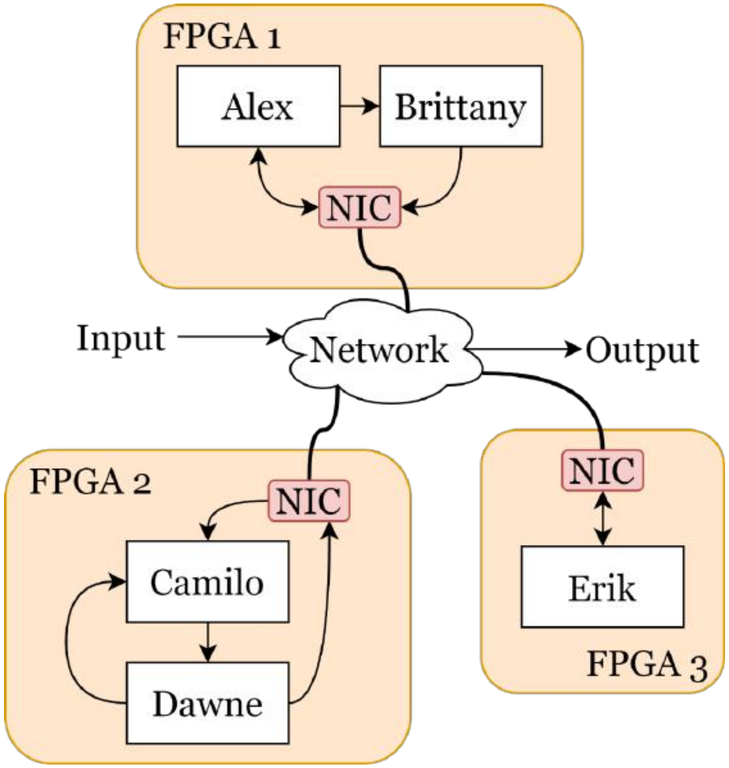


Observation

Many designs are dataflow graphs:

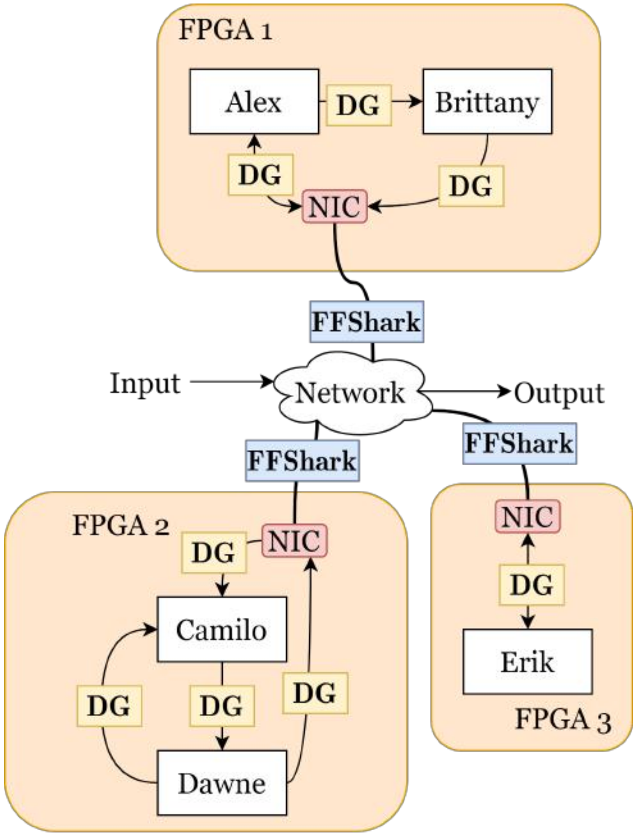


Multiple FPGAs



Our Debuggers

- Debug Governors
- FFShark

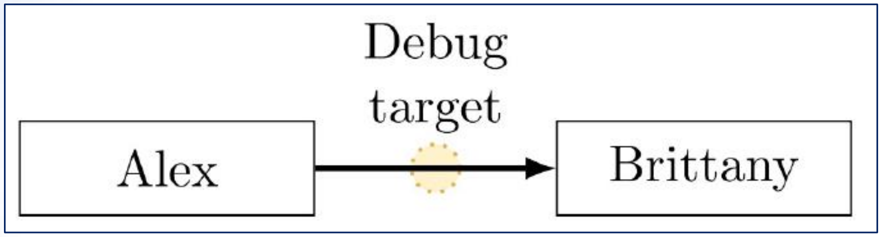


Marco Merlini, Isamu Poy, Paul Chow, FPGA 2021

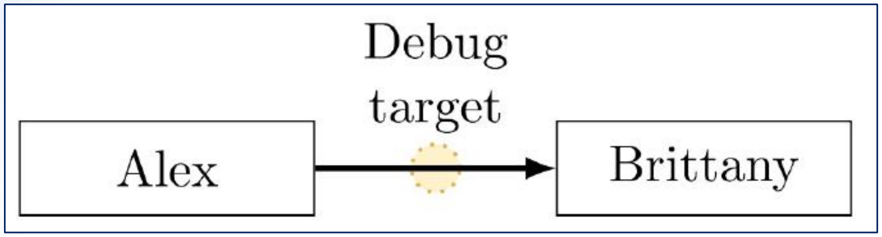
INTERACTIVE DEBUGGING AT IP BLOCK INTERFACES IN FPGAS

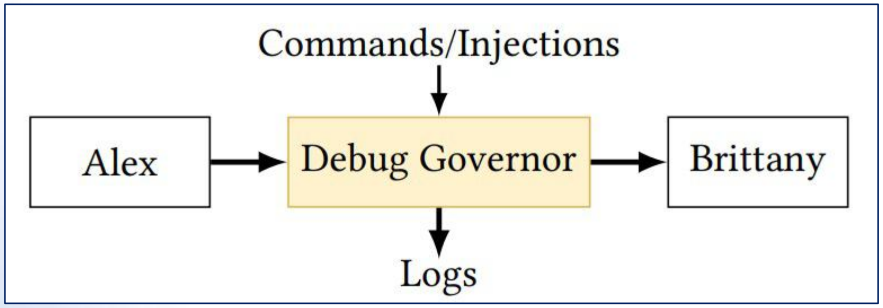
DEBUG GOVERNOR - OVERVIEW

Overview

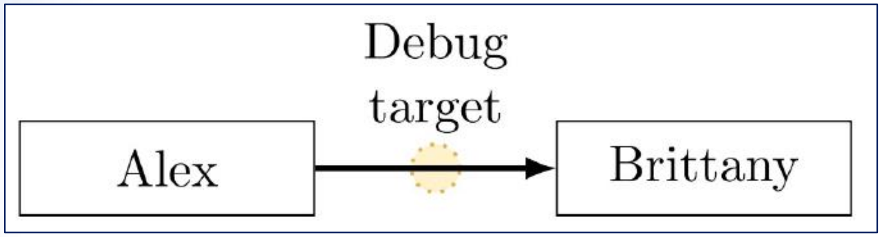


Overview

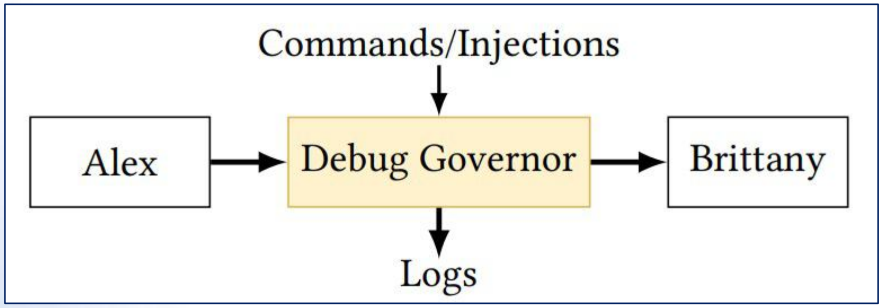

Debug
target
Alex
Brittany


Commands/Injections
Alex
Debug Governor
Brittany
Logs

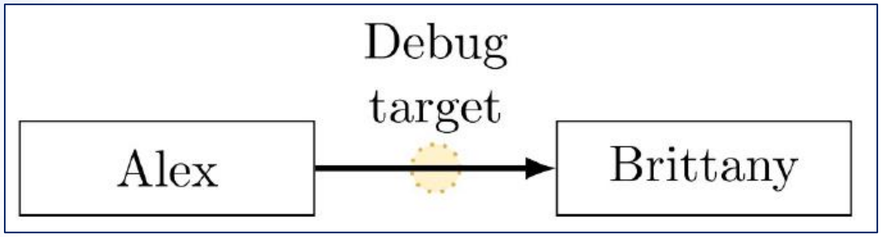
Overview



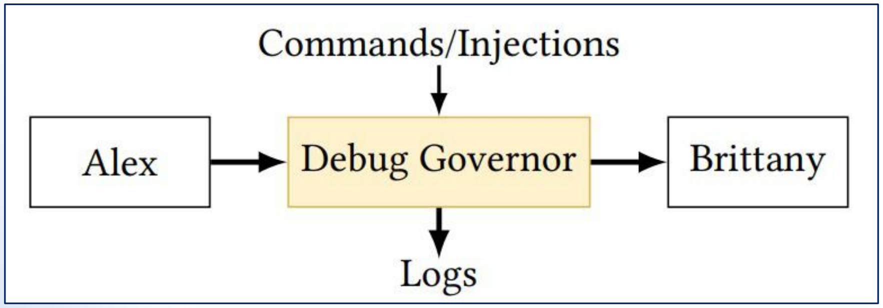
- Pause
  - Alex is prevented from sending data

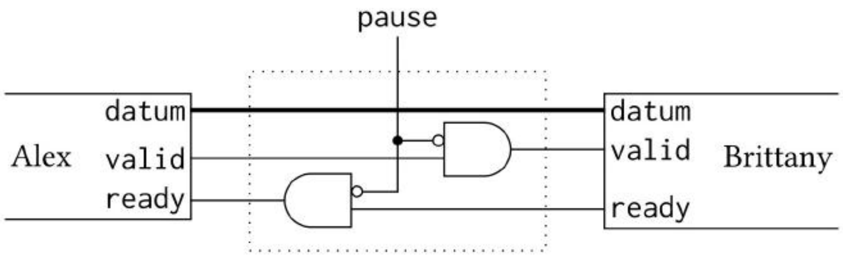


Overview

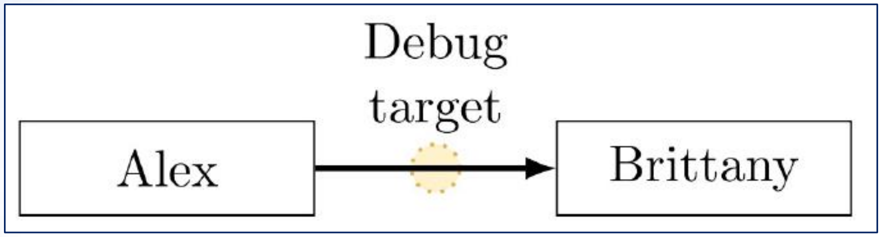


- Pause
  - Alex is prevented from sending data

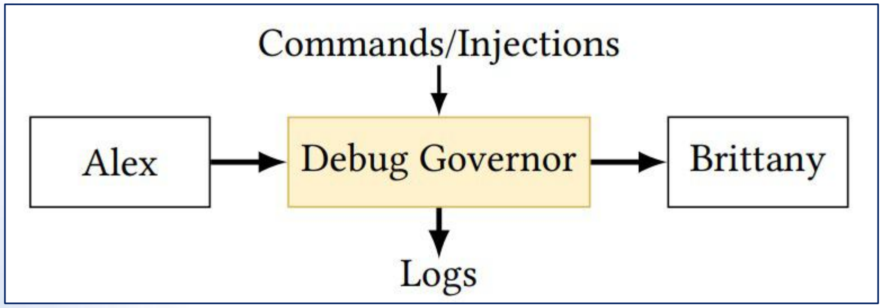




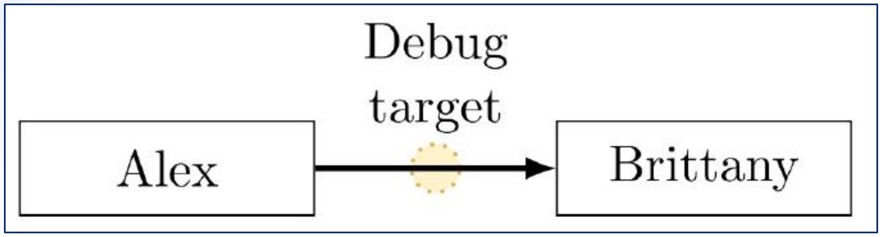
Overview

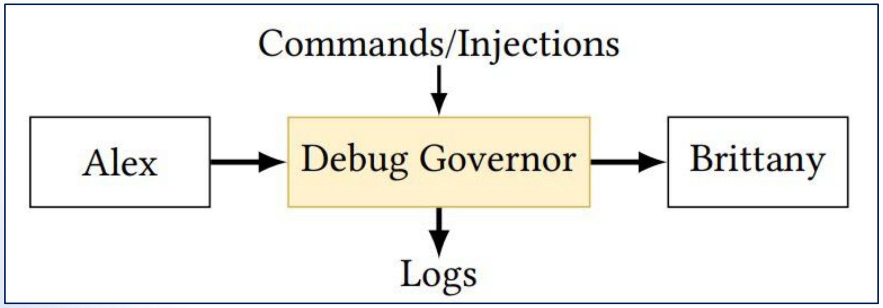


- Pause
- Log
  - Data from Alex are duplicated and sent to the developer.



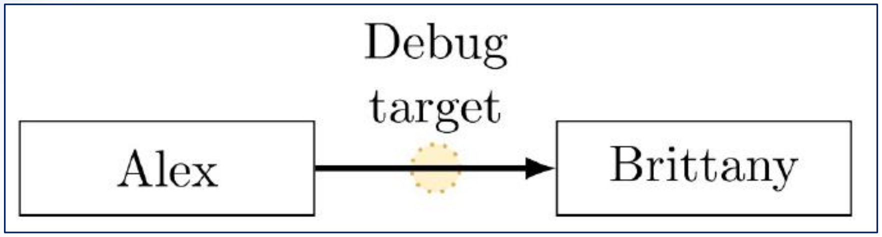
Overview

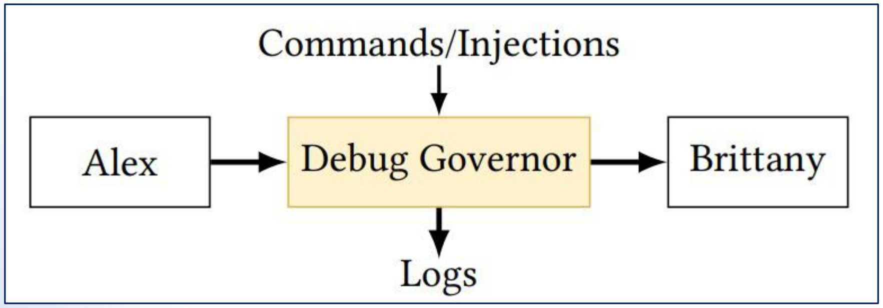




- Pause
- Log
- Drop
  - Alex can send data, but they are not sent to Brittany

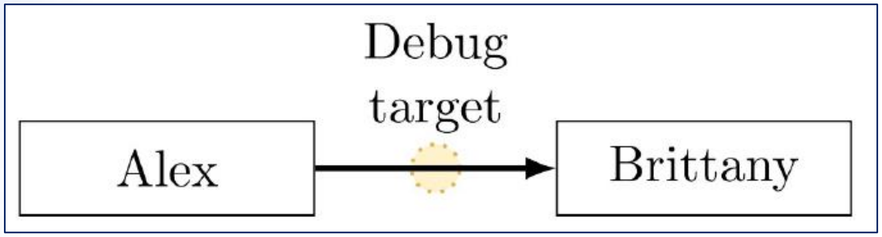
Overview

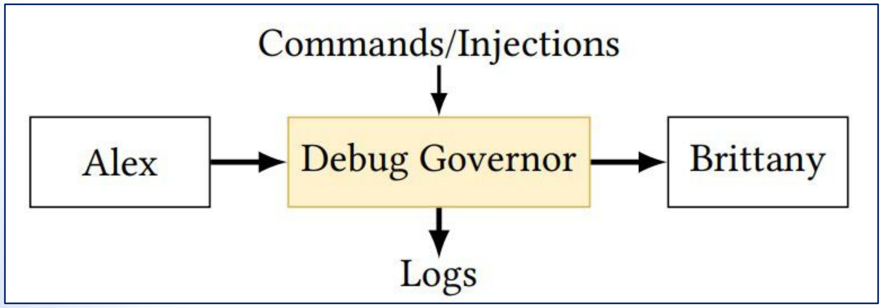




- Pause
- Log
- Drop
- Inject
  - Developer can send data to Brittany.

Overview

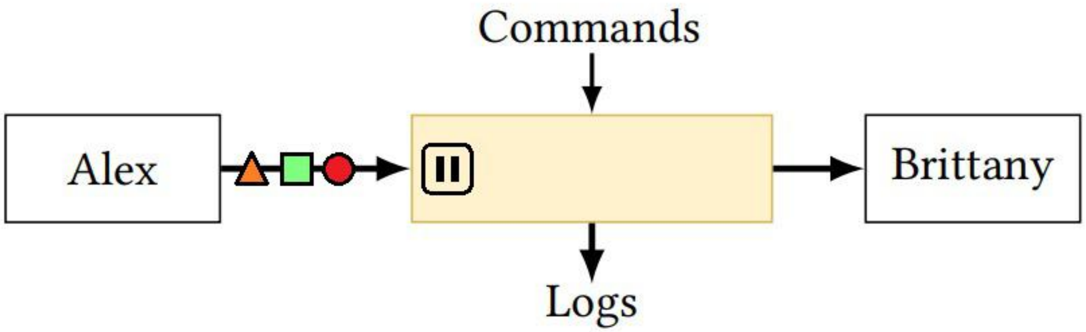




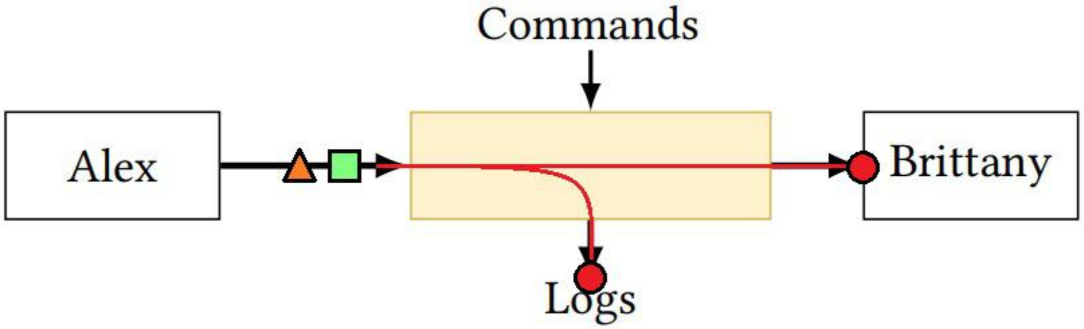
- Pause
- Log
- Drop
- Inject

+ Any subset thereof

Single-Stepping

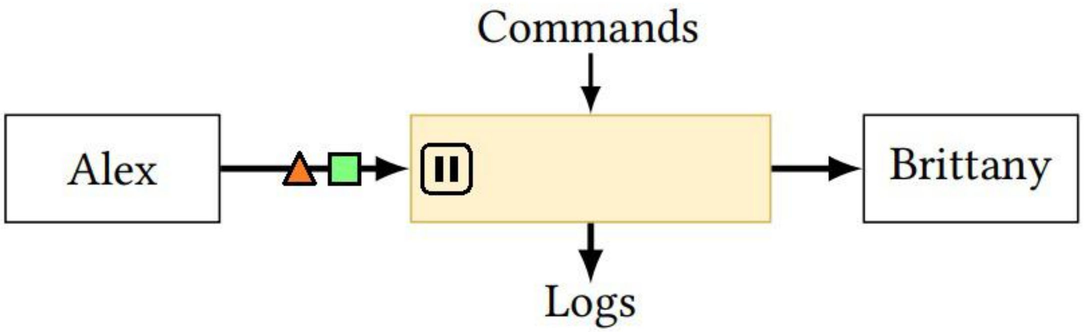


Single-Stepping



Debug Governor

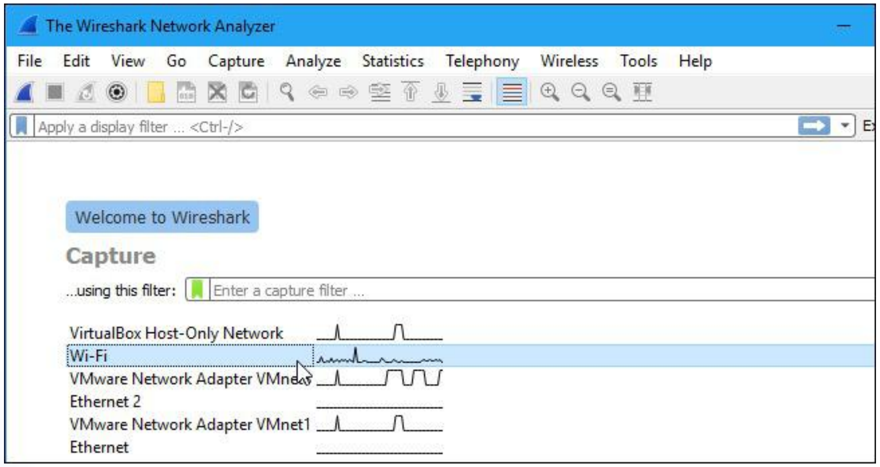
Single-Stepping



Juan Camilo Vega, Marco Merlini and Paul Chow, FCCM 20

FFSHARK: A 100G FPGA IMPLEMENTATION OF BPF FILTERING FOR WIRESHARK

Network Debugging with Wireshark

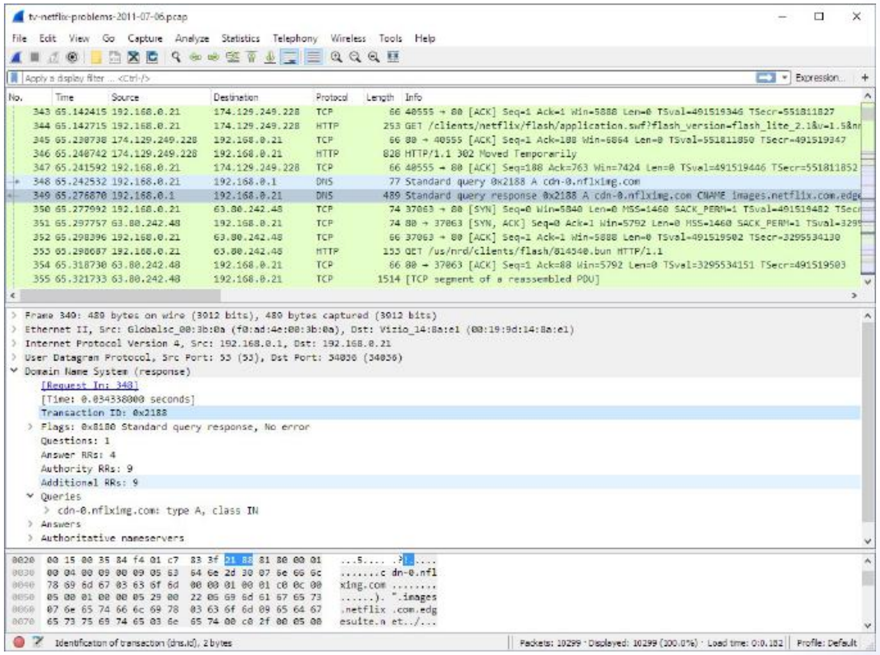

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

Welcome to Wireshark

Capture

...using this filter: Enter a capture filter ...

- VirtualBox Host-Only Network
- Wi-Fi
- VMware Network Adapter VMnet8
- Ethernet 2
- VMware Network Adapter VMnet1
- Ethernet


tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
No.   Time      Source          Destination     Protocol Length Info
  343 65.142415 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
  344 65.142715 192.168.0.21    174.129.249.228 HTTP      253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
  345 65.230738 174.129.249.228 192.168.0.21    TCP        66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
  346 65.240742 174.129.249.228 192.168.0.21    HTTP      828 HTTP/1.1 302 Moved Temporarily
  347 65.241592 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
  348 65.242532 192.168.0.21    192.168.0.1     DNS        77 Standard query 0x2188 A cdn-0.nflximg.com
  349 65.276870 192.168.0.1     192.168.0.21    DNS       489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
  350 65.277992 192.168.0.21    63.80.242.48    TCP        74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
  351 65.297757 63.80.242.48    192.168.0.21    TCP        74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
  352 65.298396 192.168.0.21    63.80.242.48    TCP        66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
  353 65.298687 192.168.0.21    63.80.242.48    HTTP      153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
  354 65.318730 63.80.242.48    192.168.0.21    TCP        66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
  355 65.321733 63.80.242.48    192.168.0.21    TCP      1514 [TCP segment of a reassembled PDU]
```

```
> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
  > Answers
  > Authoritative nameservers
```

```
0020  00 15 00 35 84 f4 01 c7  83 3f 21 88 81 80 00 01   ...5.... .?!.....
0030  00 04 00 09 00 09 05 63  64 6e 2d 30 07 6e 66 6c   .......c dn-0.nfl
0040  78 69 6d 67 03 63 6f 6d  00 00 01 00 01 c0 0c 00   ximg.com ........
0050  05 00 01 00 00 05 29 00  22 06 69 6d 61 67 65 73   ......). ".images
0060  07 6e 65 74 66 6c 69 78  03 63 6f 6d 09 65 64 67   .netflix .com.edg
0070  65 73 75 69 74 65 03 6e  65 74 00 c0 2f 00 05 00   esuite.n et../...
```

Identification of transaction (dns.id), 2 bytes

Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182 | Profile: Default

Debugging 10G and 100G networks?

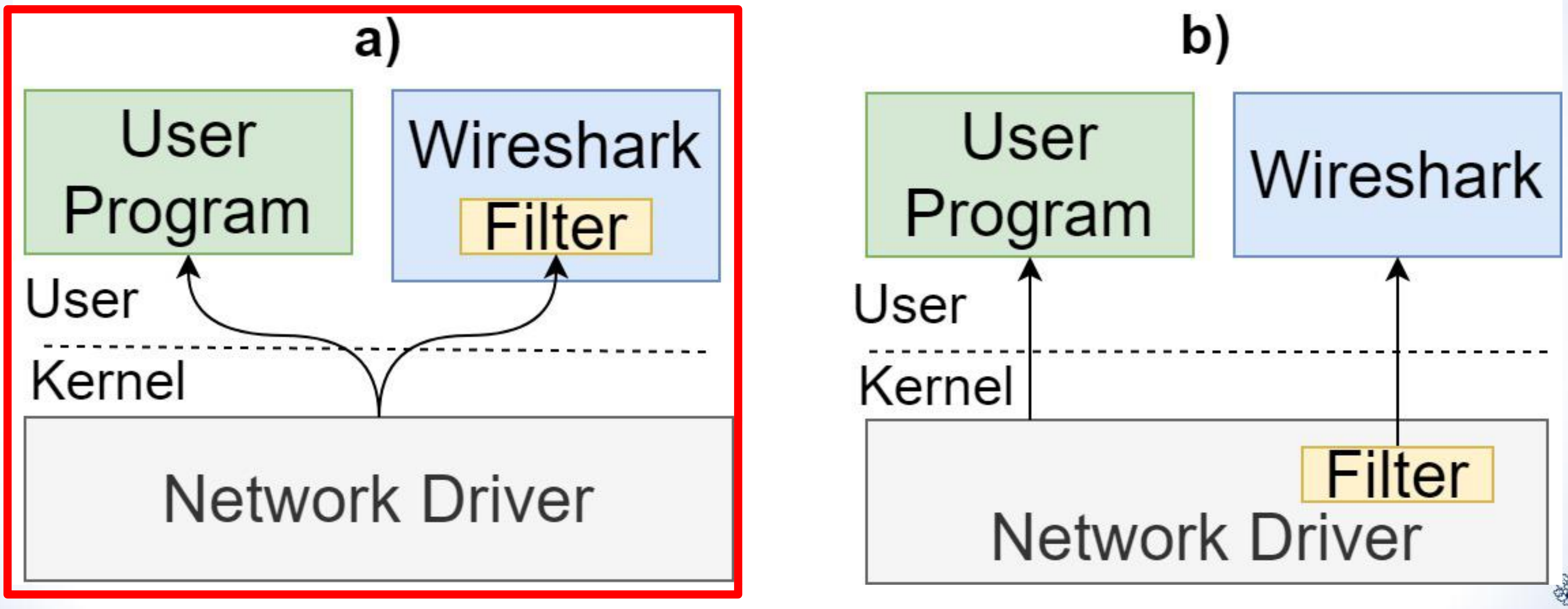
On a CPU, Wireshark is difficult at 10G, and *extremely* difficult at 100G

Example: Intel i9 processor

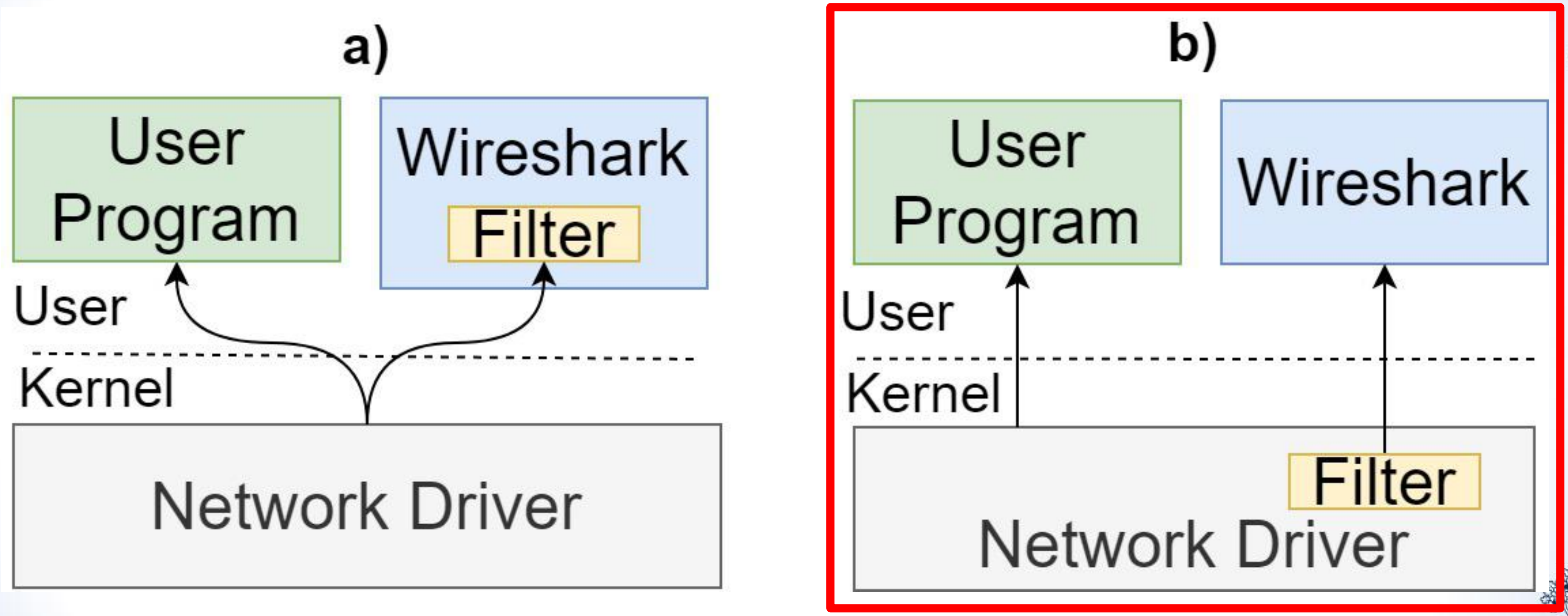
- Sixteen PCIe 3.0 lanes (8 Gbps each)
- 5 GHz
- 64-bit

Must transfer, filter, and copy accepted packets in 3 cycles/word!

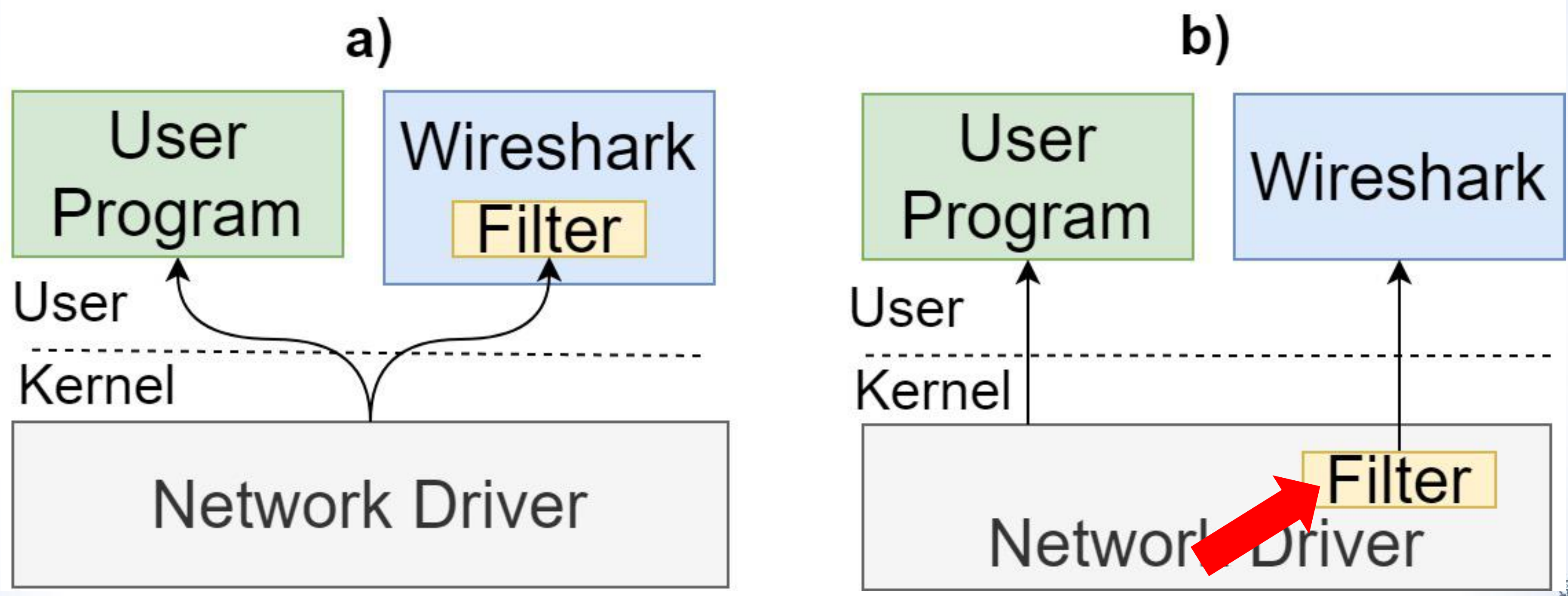
Improving Packet Capture Performance



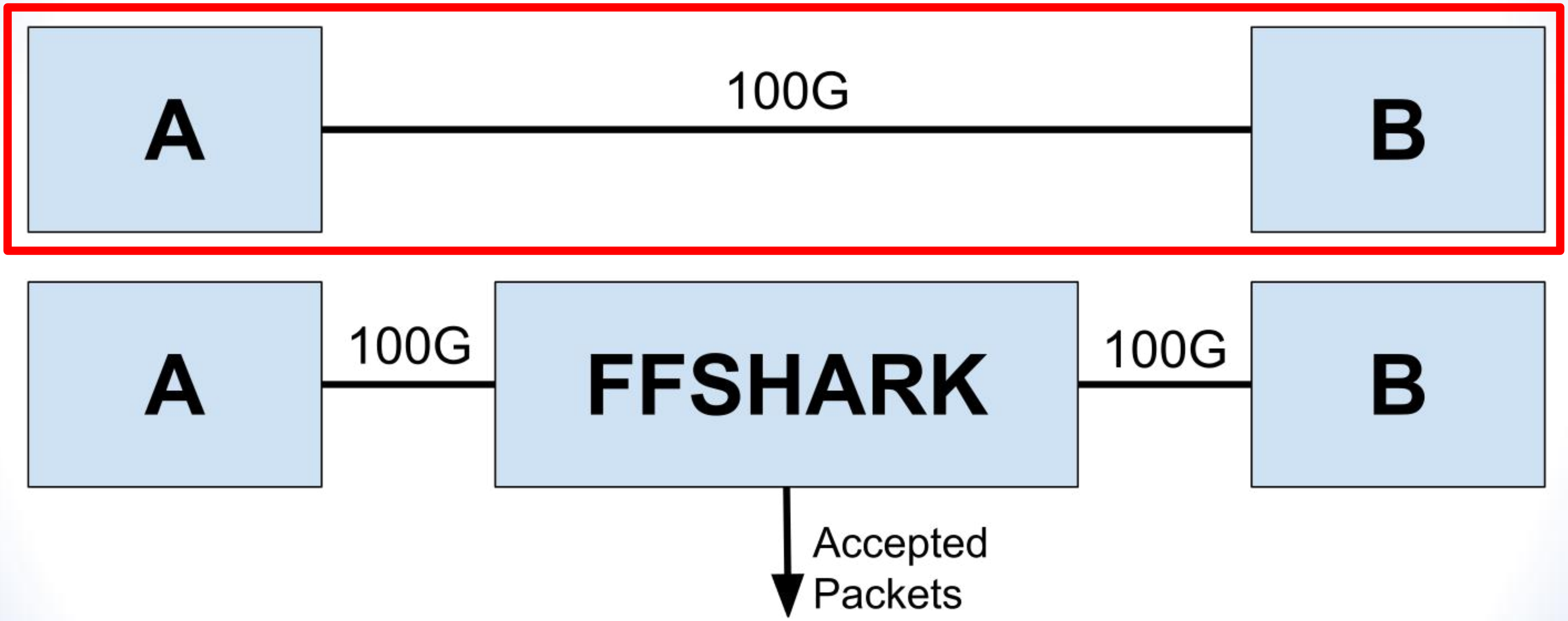
Improving Packet Capture Performance



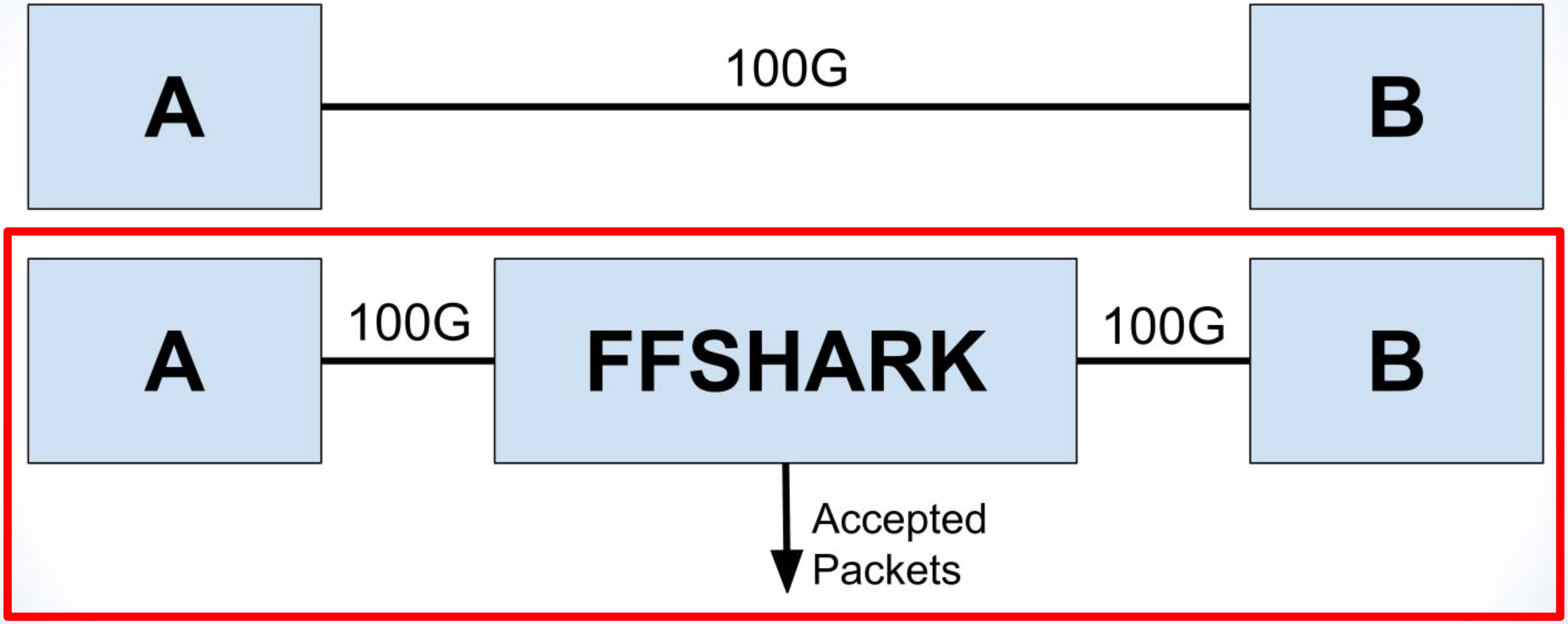
Improving Packet Capture Performance



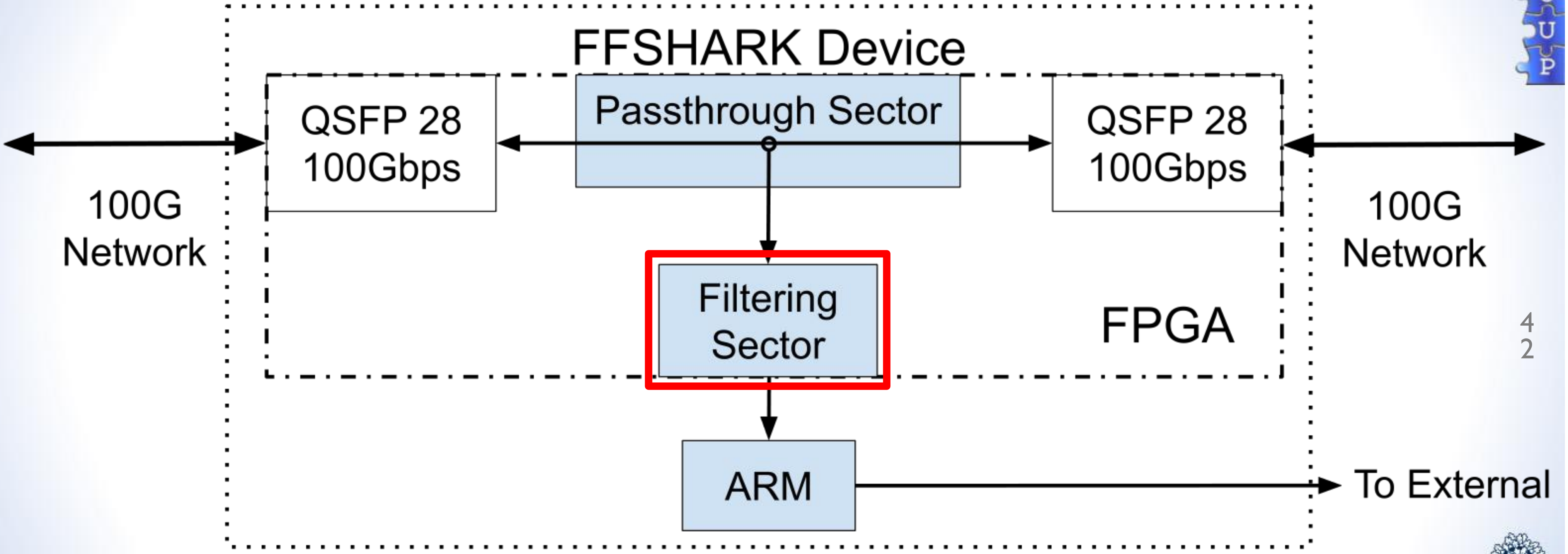
FFShark



FFShark



Architecture of the FFShark System

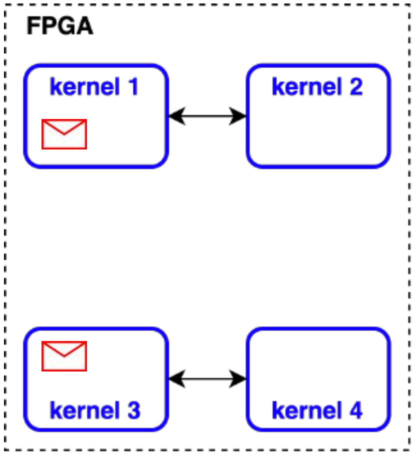


Arzhang Rafii and Paul Chow, FPL 2021, Thursday Session 4A

PHAROS: A MULTI-FPGA PERFORMANCE MONITOR

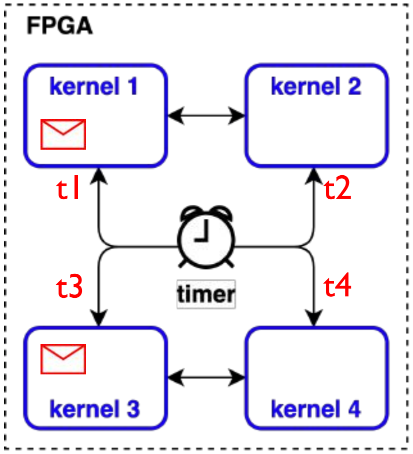
Motivation

- Monitoring events in a single FPGA
  - Which event happened first?
  - How long did each event take?



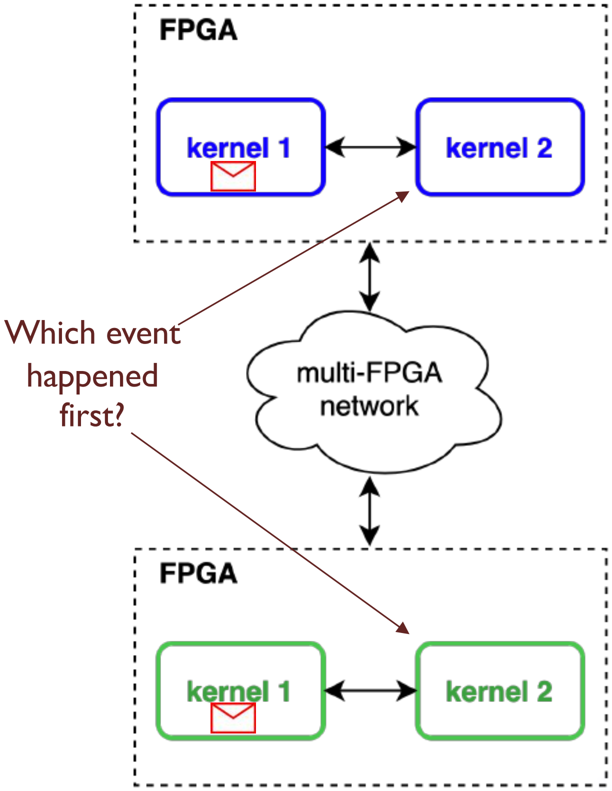
Motivation

- Monitoring events in a single FPGA
  - Which event happened first?
  - How long did each event take?
- Use a single timer to generate timestamps



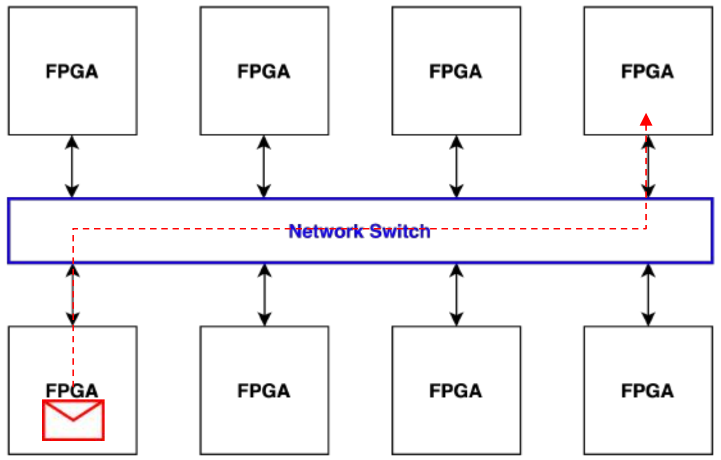
Motivation

- Monitoring events in a single FPGA
  - Use a single timer to generate timestamps
- Monitoring events in a multi-FPGA system
  - How to order events in the right sequence?



Motivation

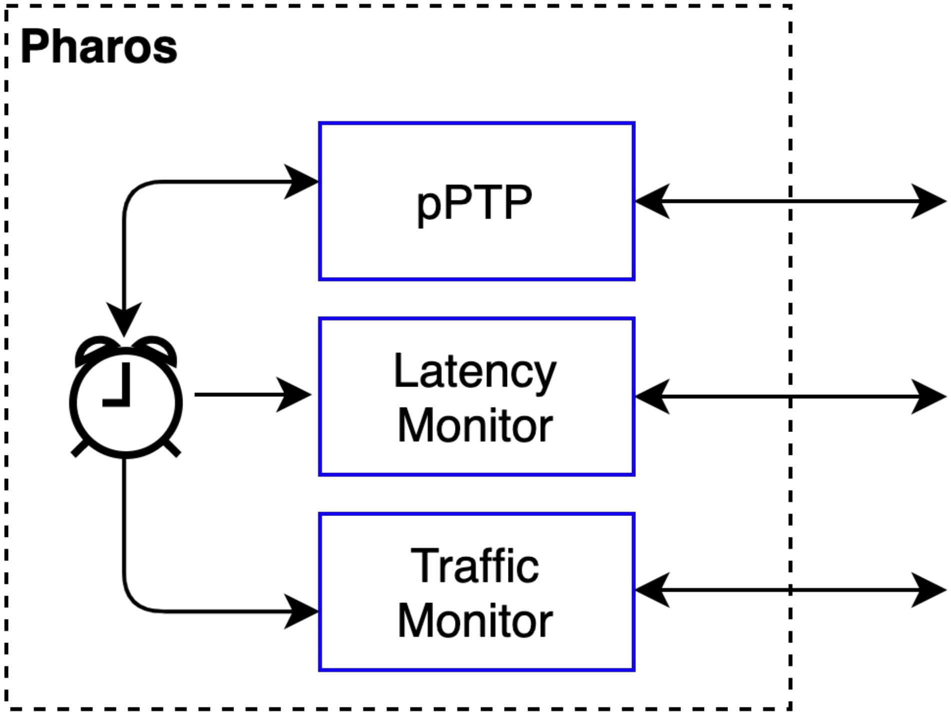
- Monitoring events in a single FPGA
  - Use the same timer to generate timestamps
- Monitoring events in a multi-FPGA system
  - How to order events in the right sequence?
  - How to find point-to-point latency?



The Pharos Performance Monitor

- Uses the idea of "global time"
- Measures unidirectional point-to-point latency
- Collects traffic data
- Logs events in a multi-FPGA system
- Independent from lower-level communications

The Pharos Performance Monitor



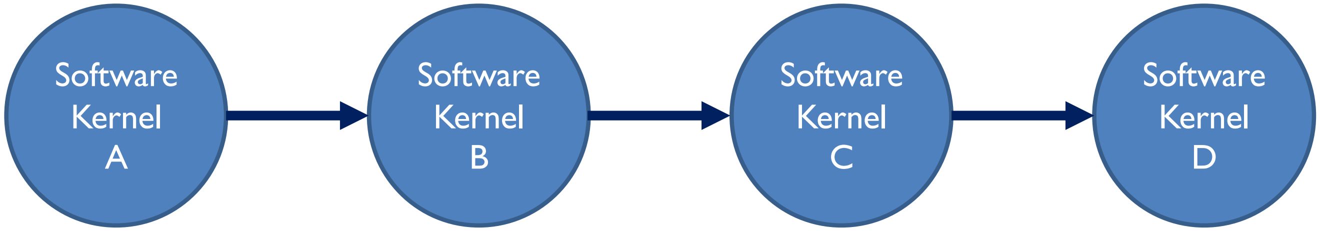
BUT WHAT WORKS BEST SO FAR?

Simulation!

If you don’t simulate, it won’t work

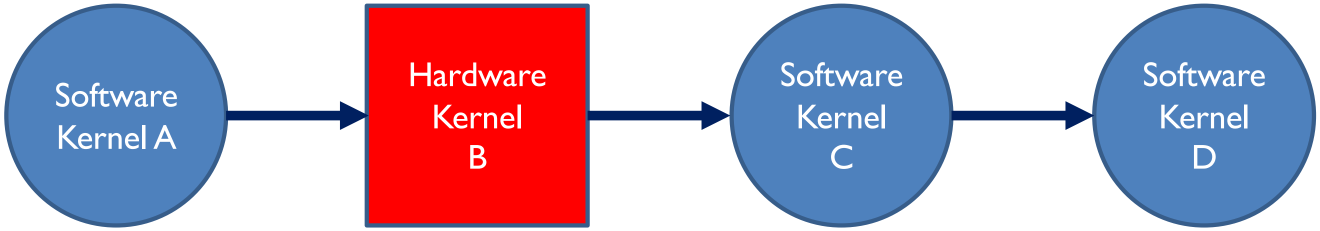
If you do simulate, it might work

Co-Simulation using Galapagos

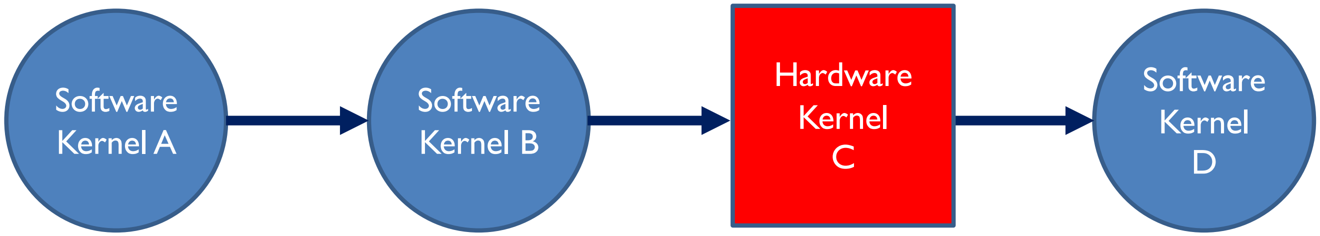


Start with everything running in software

Incremental Hardware Implementation

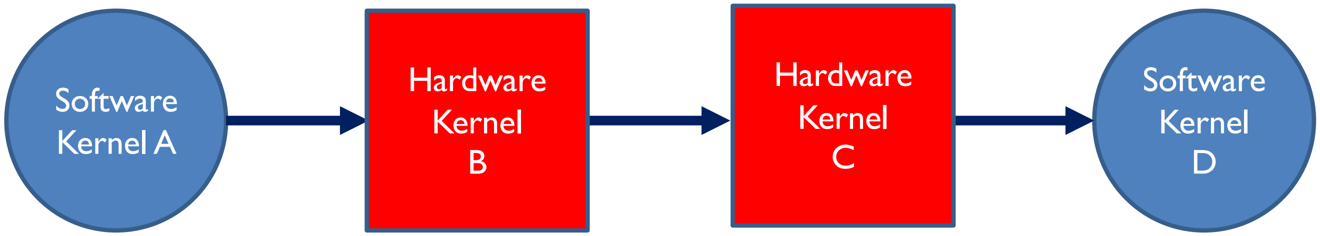


Test each kernel in hardware individually



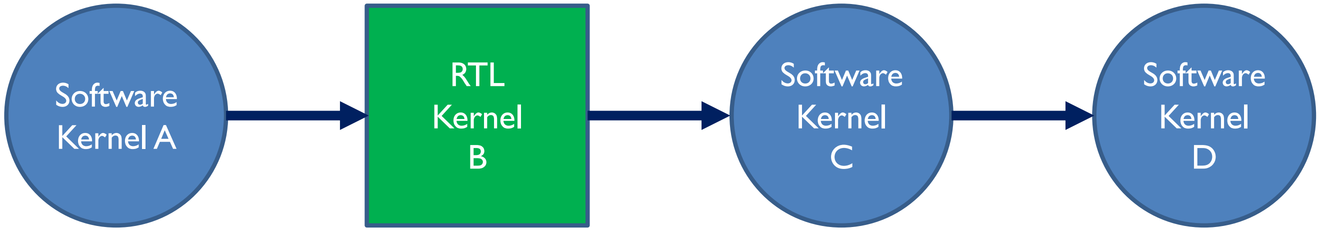
Test each kernel in hardware individually

Full System Integration



Then put it all together

Co-simulation with RTL Model



If you really need to…

FINAL THOUGHTS

- We need a collection of tools to help us debug
  - Have described some possibilities we've considered
  - Catapult Flight Data Recorder
- Should be usable by everyone, i.e., not one-off
  - Build on the work of others, don't re-invent
- How and what to standardize?
- We need to think more about debugging!!

THANKS FOR LISTENING