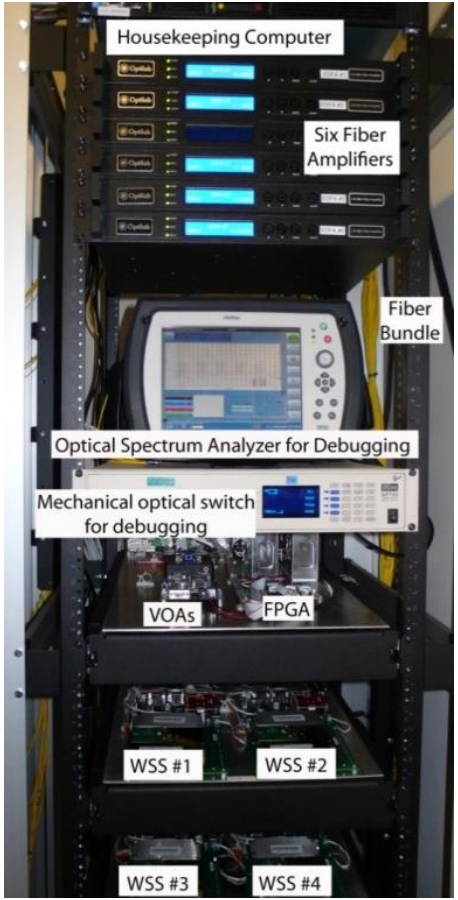
Corundum

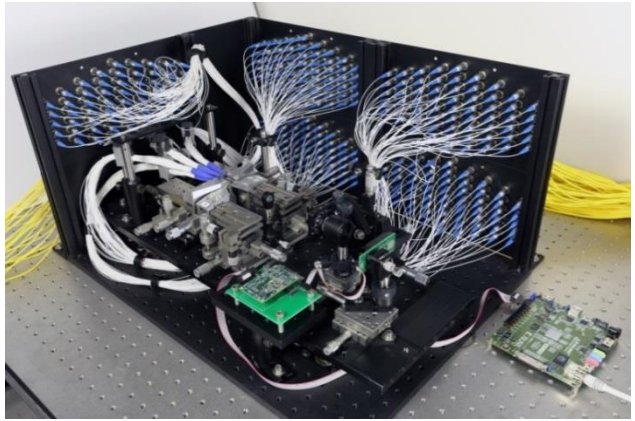
From a NIC to a Platform for In-Network Compute

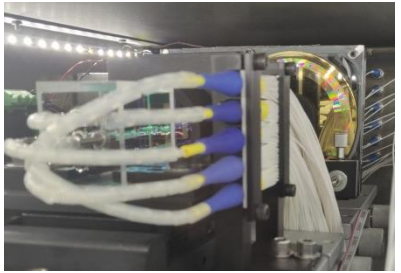
Alex Forencich
Ulrich Langenbach

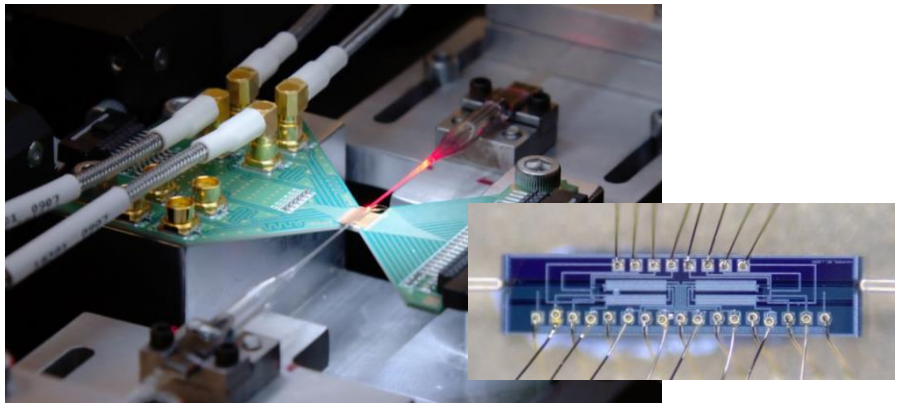
Circuit-Switching Research at UCSD

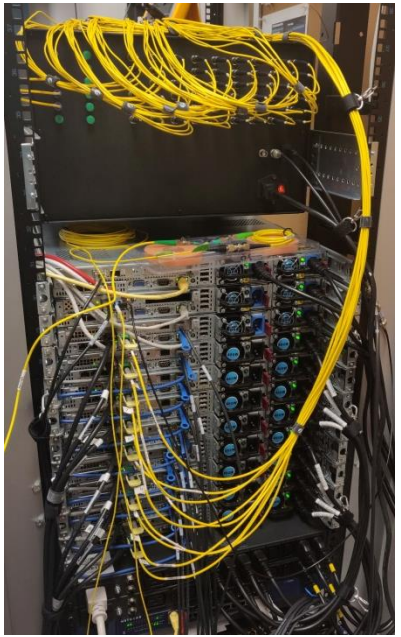
UC San Diego
JACOBS SCHOOL OF ENGINEERING



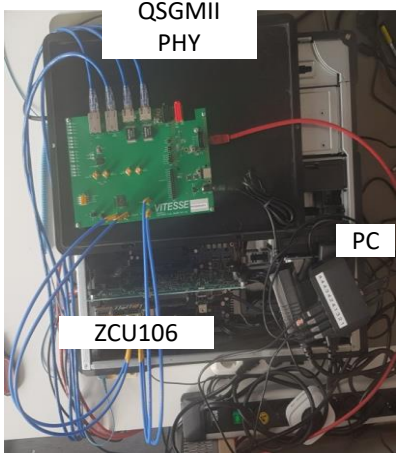


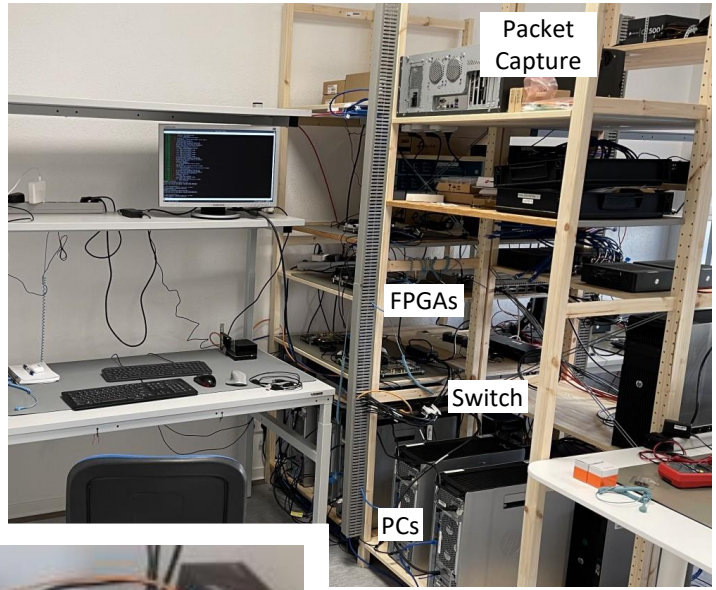


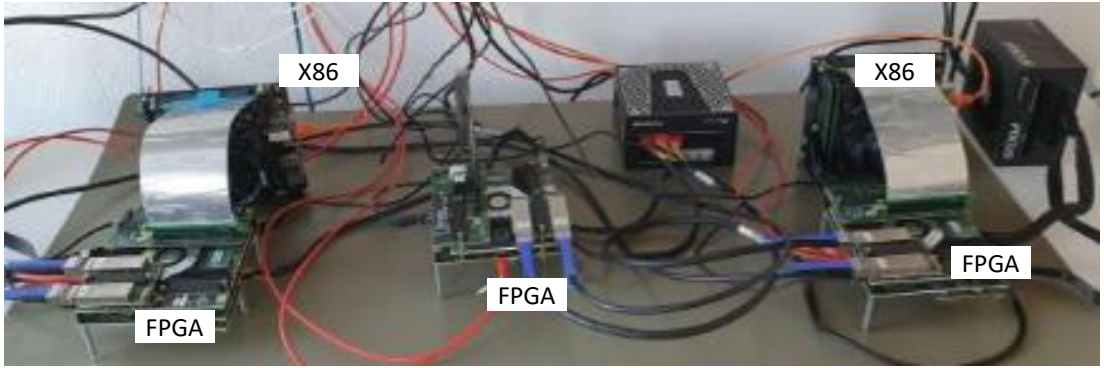




Networked Systems, IPs and Devices at MLE

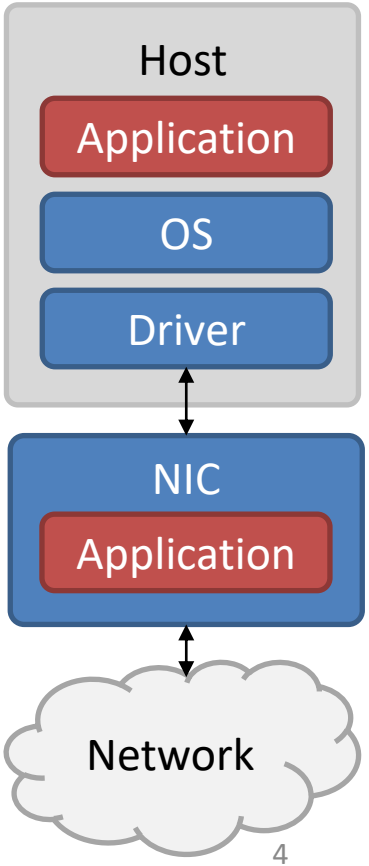






Introduction

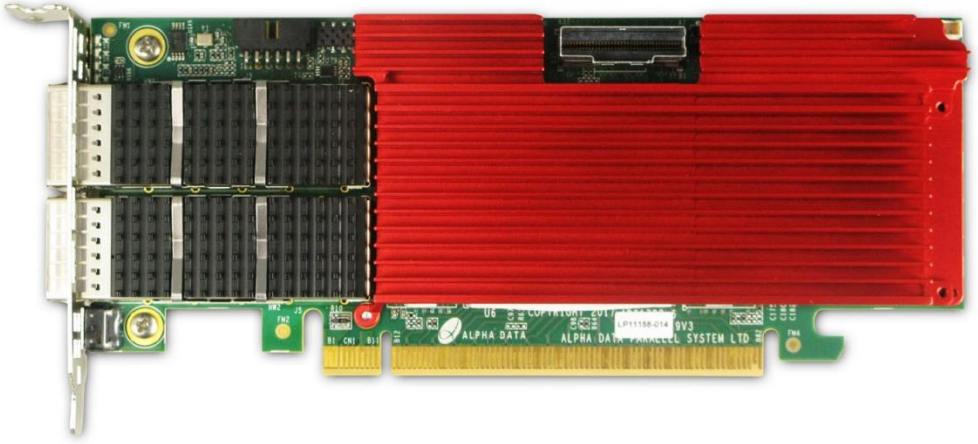
- Network Interface Controller (NIC) connects software to the network
- NIC functionality is evolving
  - Line rate increases
  - Offload networking functions from CPU to NIC
- More general: in-network compute
  - Offload compute to programmable NICs, switches, etc.
  - Not limited to network stack



Corundum

UC San Diego
JACOBS SCHOOL OF ENGINEERING

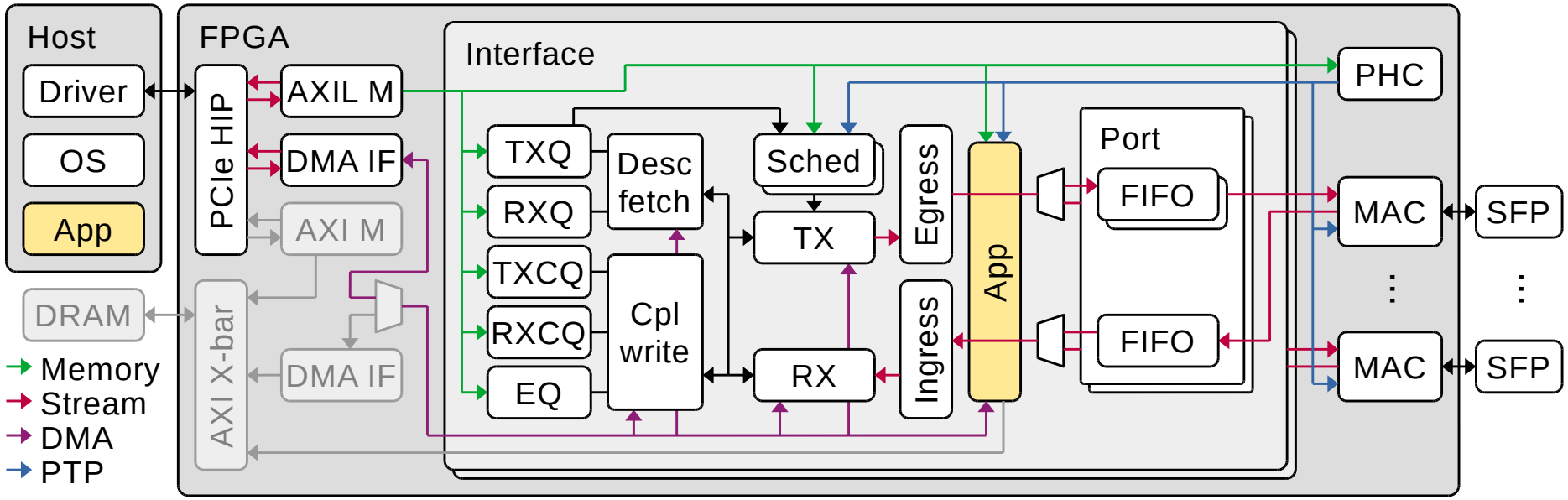
- Open-source, FPGA-based NIC and platform for in-network compute
  - A high performance “reference” NIC
  - Extensible: application block for implementation of custom features
  - Key applications: hardware prototyping of experimental networks/protocols, custom compute offload



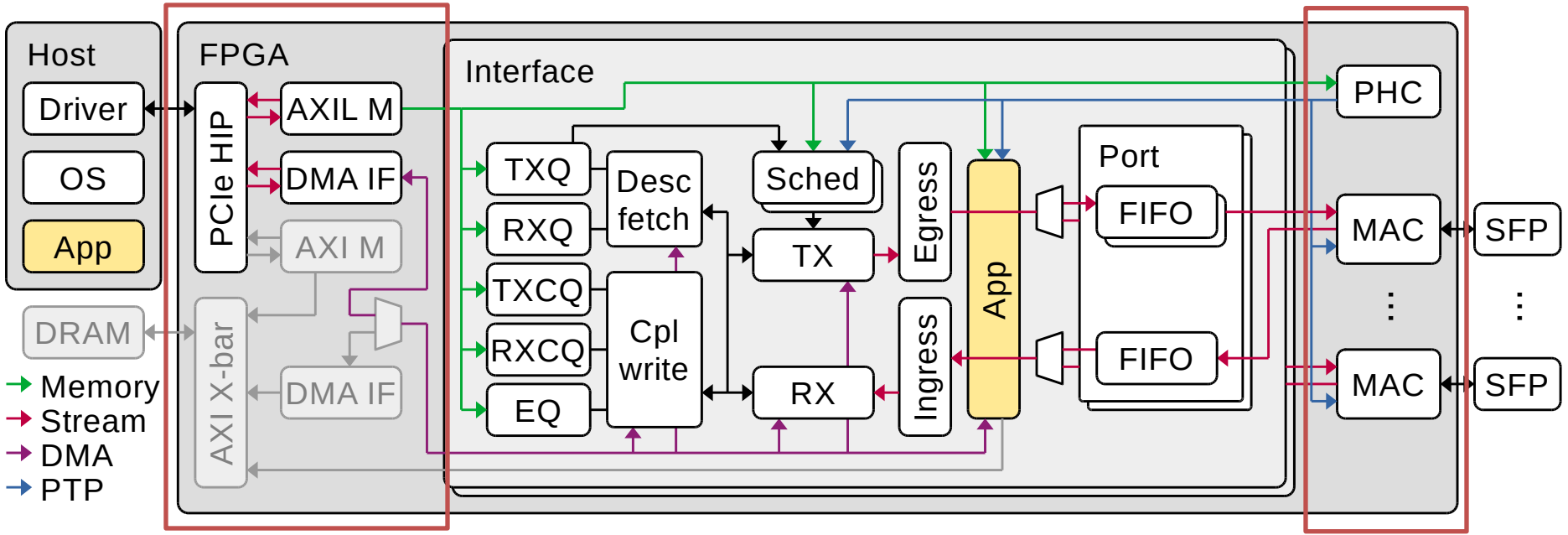
High-Level Features of Corundum

- Open-source, high-performance, FPGA-based NIC
  - PCIe gen 3 x16, multiple 10G/25G/100G Ethernet ports
  - Fully custom, high-performance DMA engine; Linux driver
- Application block for custom logic
  - Access to network traffic, DMA engine, on-card RAM, PTP time
- Fine-grained traffic control
  - 10,000+ hardware queues, customizable schedulers
- PTP timestamping and time synchronization
- Management features (FW update, etc.)
- Wide device support (Xilinx and Intel)
- Source code: https://github.com/corundum/corundum

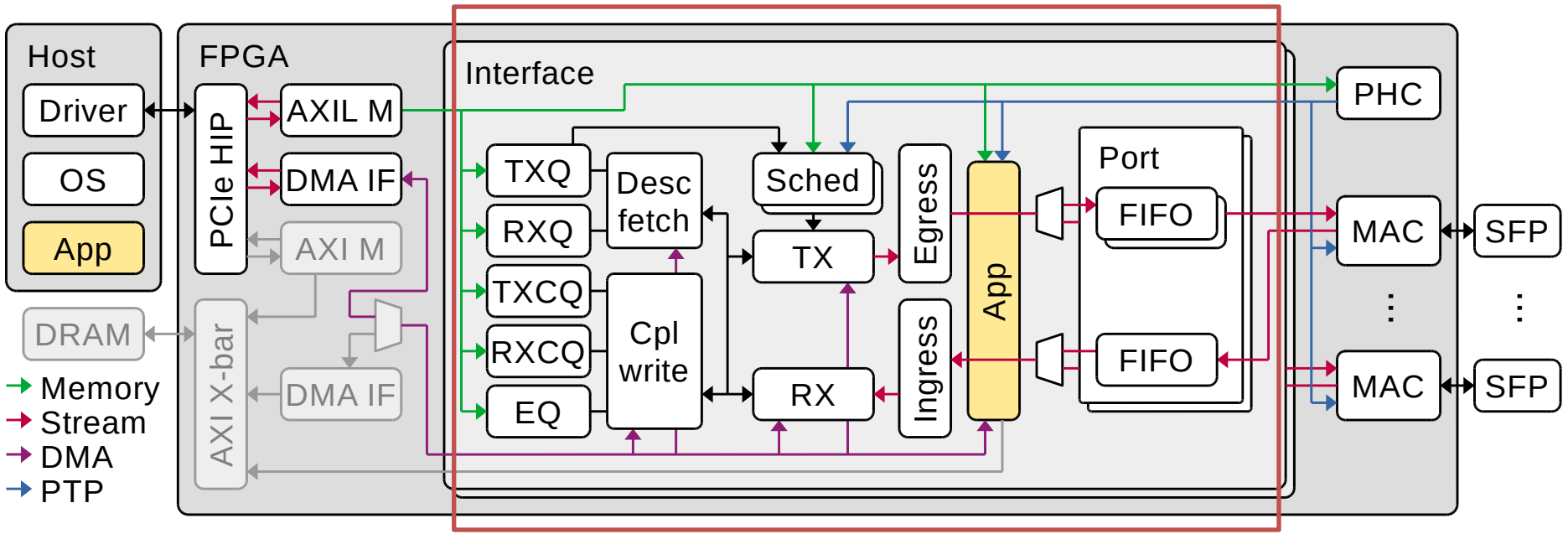
Corundum Block Diagram



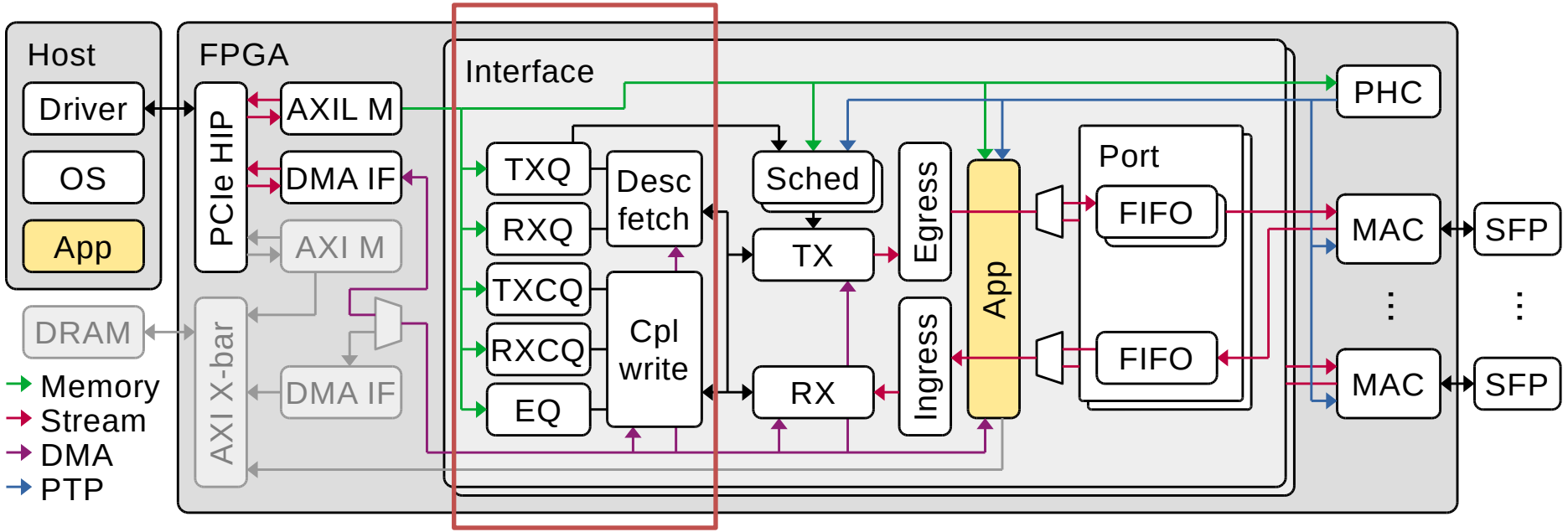
Corundum Block Diagram



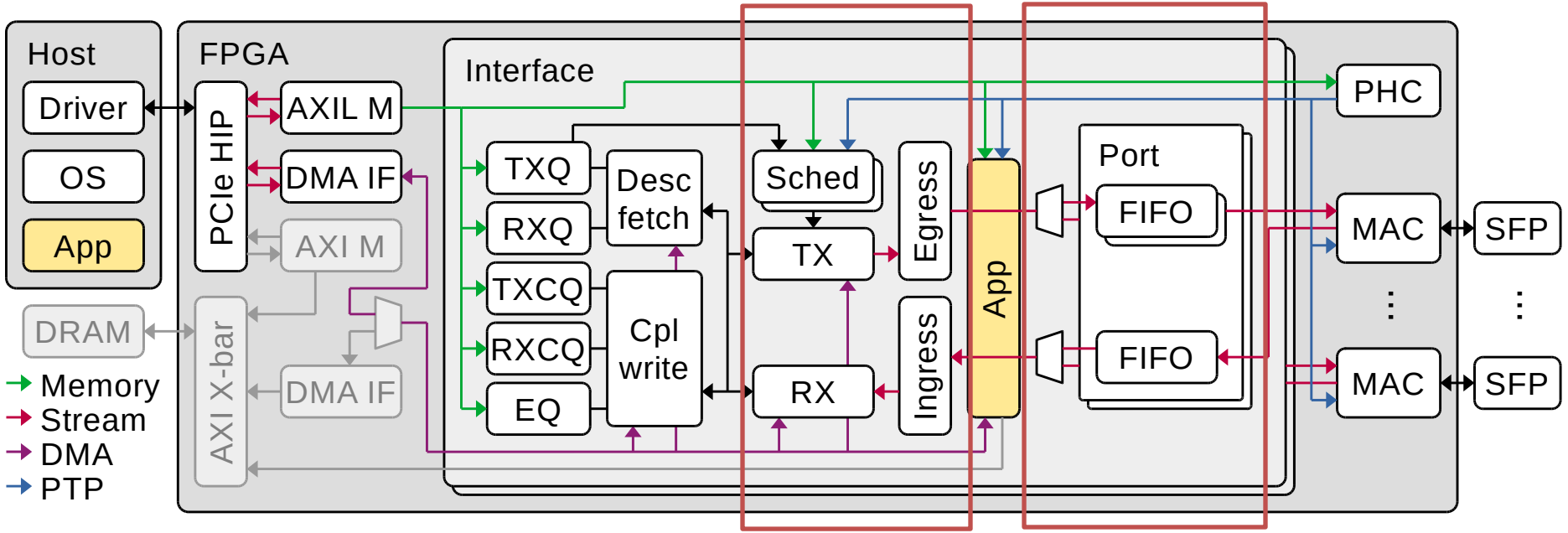
Corundum Block Diagram



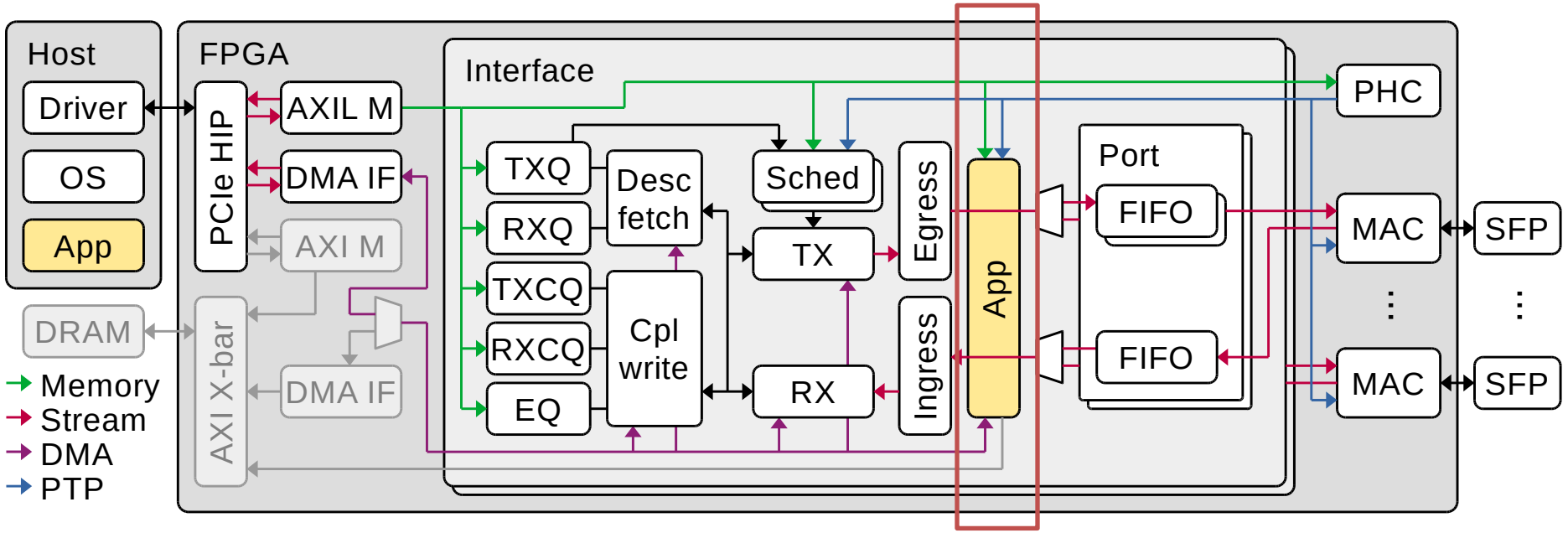
Corundum Block Diagram



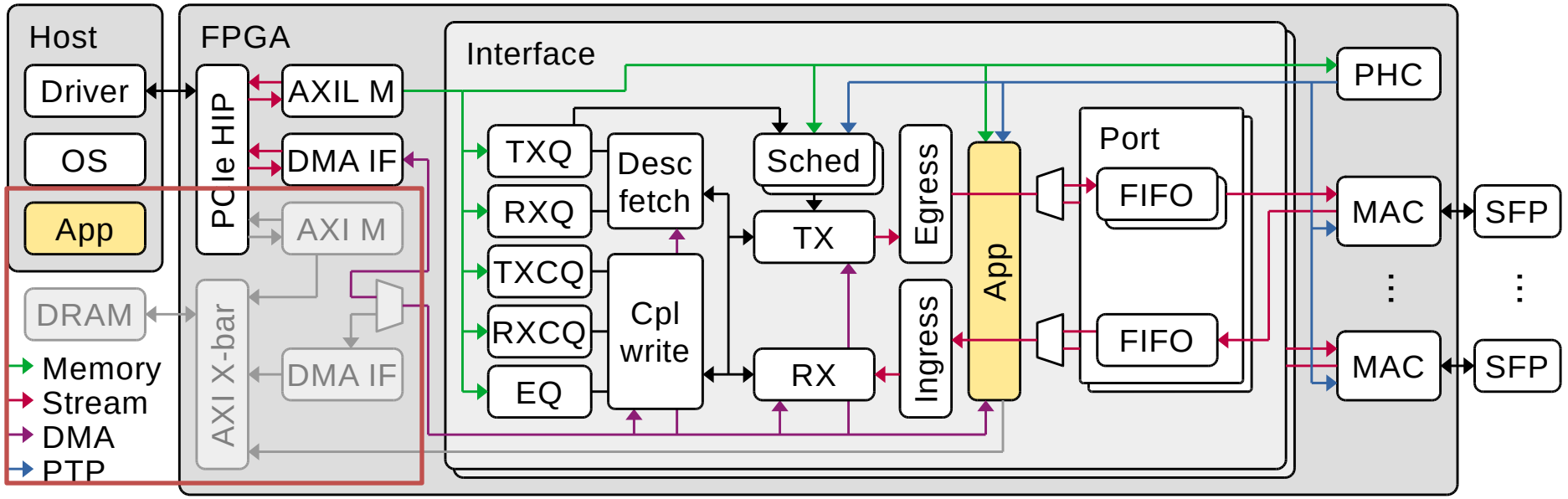
Corundum Block Diagram



Corundum Block Diagram



Corundum Block Diagram

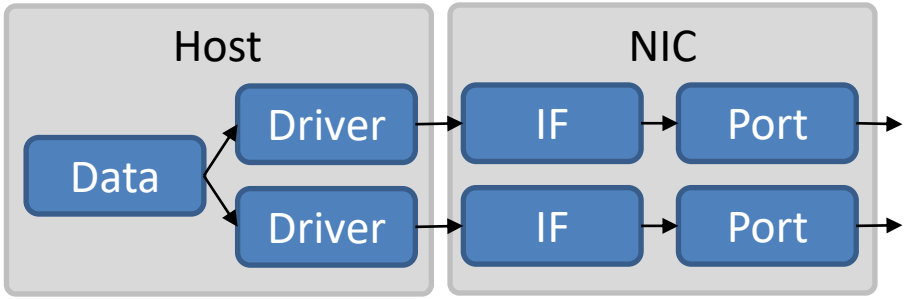


Fine-grained traffic control

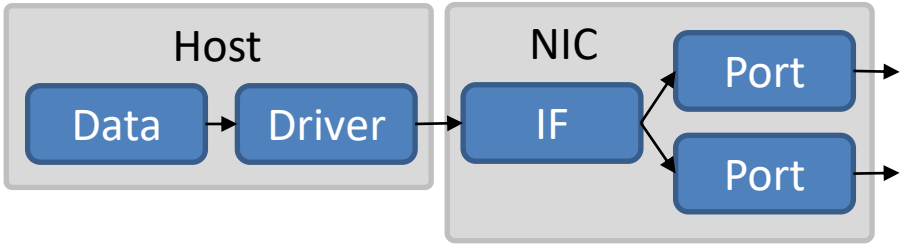
- 10,000+ transmit queues
  - Each queue is an independent channel between SW and HW
  - Classify in SW, control in HW
  - Fine-grained, per-flow or per-destination control
  - 128 bits/queue -> 4096 queues in 2 URAM on US+
- Transmit scheduler
  - Determines which queue to transmit from
  - Default scheduler is round robin
  - Can be used to implement traffic shaping, rate limiting, etc.

Ports and Interfaces

- Hardware support for multiple uplinks
- Multiple physical ports appear as single OS-level interface
- Ports have separate schedulers
- Migrate or stripe flows across ports by changing scheduler settings



Traditional NIC: assignment in software

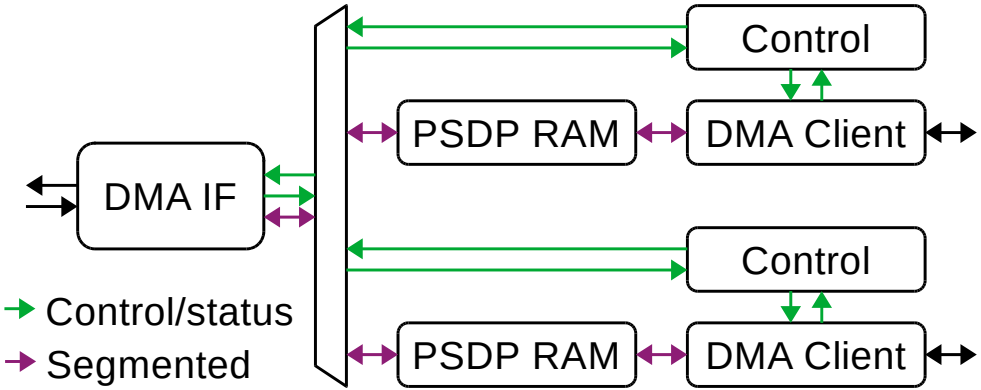


Corundum NIC: assignment in hardware

Modular DMA engine

UC San Diego
JACOBS SCHOOL OF ENGINEERING

- DMA engine split between interface and client modules
  - Interface connects to host – PCIe, AXI, etc.
  - Client modules form internal ports – AXI stream, memory-mapped AXI
- Clients connected to interface with dual port RAMs
- Support both servers (PCIe) and SoCs (AXI) with same core logic



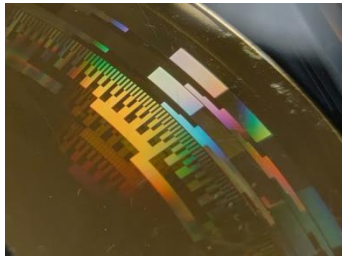
Applications

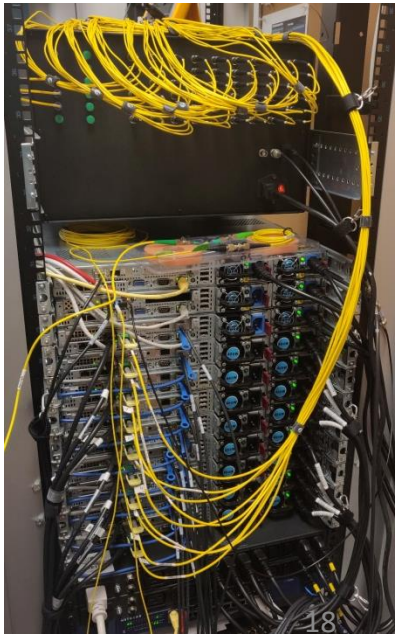
- Offload application-specific processing
- Datapath for novel transmit schedulers
- Instrument Corundum for performance measurements
- Direct transceiver access permits physical-layer measurements and development of new wire protocols
- Use core logic as a packet DMA engine in a larger system
- Discuss two applications:
  - TDMA for microsecond circuit switching
  - PHY layer BER measurement for link characterization

Application: TDMA

UC San Diego
JACOBS SCHOOL OF ENGINEERING

- Scheduler can control queues based on PTP time
  - Enables sub-microsecond-resolution TDMA
- TDMA off
  - 94 Gbps
- 200 us period, 50% duty cycle TDMA schedule
  - Guard time 2 us
- Using TDMA schedule, can run iperf through pinwheel switch
  - Initial test with old FW: 3 Gbps on 10 Gbps link with low packet loss

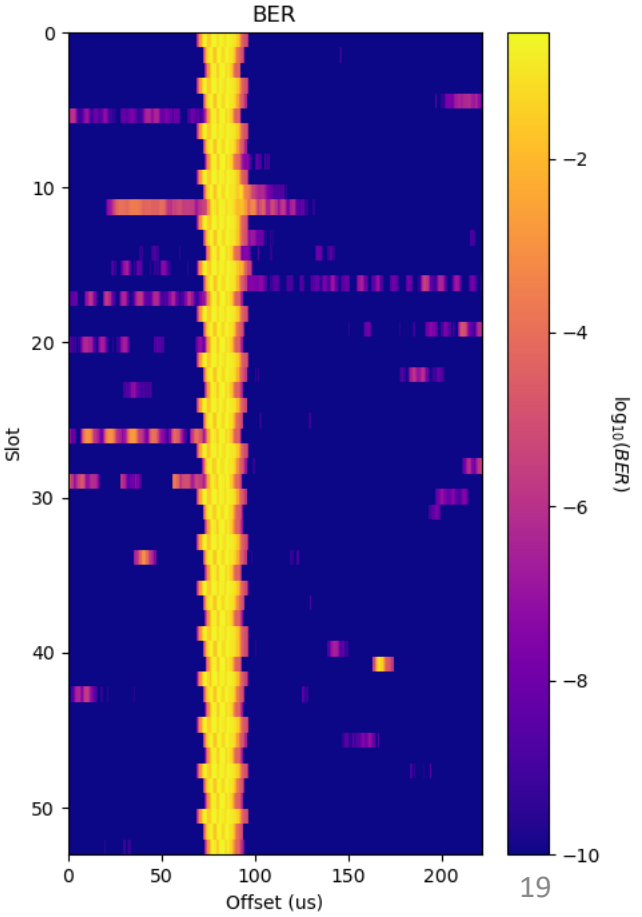




Application: link-level characterization

UC San Diego
JACOBS SCHOOL OF ENGINEERING

- *In situ* link-level measurements
- BER measurement capability integrated into NIC
- Measure link-level performance from vantage point of every NIC in datacenter
- Supports time-domain BER
  - Synchronized over network via PTP
  - Measure every path through switch
- Heat map represents signal at one receiver through pinwheel switch



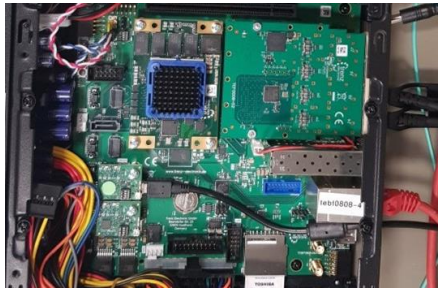
Open Source

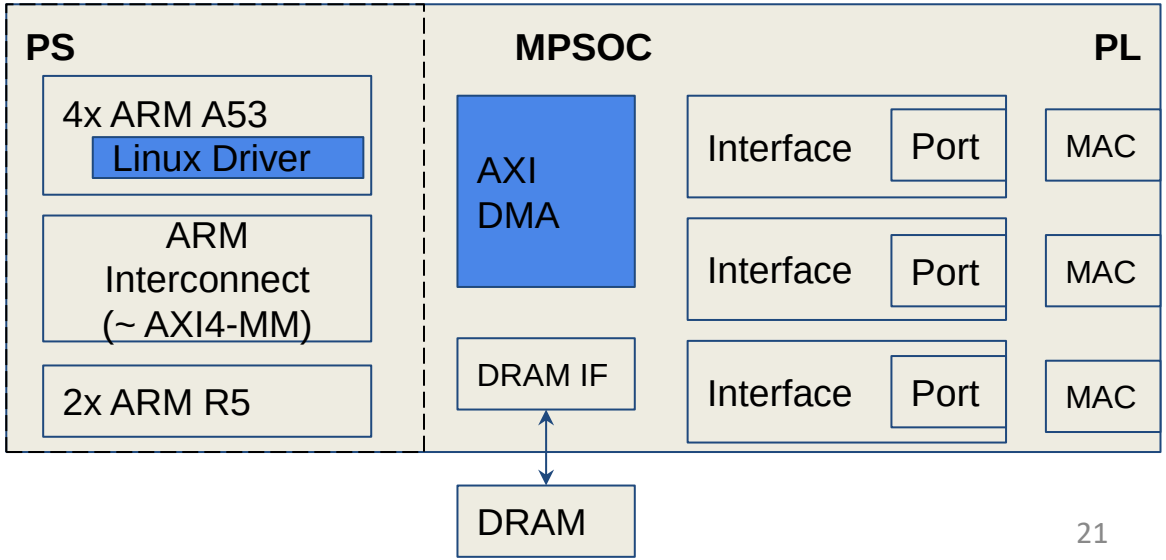
UC San Diego
JACOBS SCHOOL OF ENGINEERING

- Corundum core logic 100% open-source Verilog
  - Including 10G/25G MACs, PCIe DMA, AXI, AXI stream, etc.
  - Device hard IP used for interfacing – PCIe, 100G MAC, serdes
- Simulation uses open-source tools
  - Cocotb + Icarus Verilog
  - Eventually will use cocotb + Verilator (blocked on Verilator bugs)
  - Wrote cocotb extensions for AXI, Ethernet, and PCI express
- Tox + pytest for CI
- Makefiles for build automation

Short term work / contributions by MLE

1. Xilinx Zynq UltraScale+ MPSoC Integration
   a. Run Linux on the PS ARM connected to the NIC
   b. Support some additional platforms, e.g. Fidus Sidewinder-100 and Trenz TE0808 based





Short term work / contributions by MLE

1. Xilinx Zynq UltraScale+ MPSoC Integration (WIP)
2. Intel Device Support
   a. Kicked-off by the initial intel support for Stratix 10 MX
   b. Optimize RTL to efficiently infer the intended RAM structures
   c. Utilize a price-efficient Stratix 10 GX FPGA

Short term work / contributions by MLE

1. Xilinx Zynq UltraScale+ MPSoC Integration (WIP)
2. Intel Device Support (WIP)
3. 1 GbE interface support
    a. SGMII reference design for ZCU106 (PR pending rebase)

Short term work / contributions by MLE

1. Xilinx Zynq UltraScale+ MPSoC Integration (WIP)
2. Intel Device Support (WIP)
3. 1 GbE interface support (WIP)
4. DPDK Driver
   to provide one DPDK device per mqnic interface

Short term work / contributions by MLE

1. Xilinx Zynq UltraScale+ MPSoC Integration (WIP)
2. Intel Device Support (WIP)
3. 1 GbE interface support (WIP)
4. DPDK Driver (WIP)
5. Add a Traffic Shaper (customization)
    a. Enforce per queue (queue group) bandwidth assignments
    b. Based on the bucket shaping algorithm
    c. Implements a scheduler controller to complement other available schedulers, e.g. the current upstream round robin scheduler

Long-term goals

UC San Diego
JACOBS SCHOOL OF ENGINEERING

- Core features
  - RDMA support in core datapath
  - Variable-length descriptor support
  - Unified DMA address space (on-card DRAM/HBM)
- Device and board support
  - Improve Intel device support
  - SoC support
  - Move board-dependent code from driver to soft core
  - Simplify porting process
- Software
  - Improved interface to application logic
  - DPDK driver

Thank you

UC San Diego
JACOBS SCHOOL OF ENGINEERING

Corundum source code available on GitHub:

https://github.com/corundum/corundum