Novel RISC-V Implementations and Accelerators

Jeffrey Young with assistance from Hyesoon Kim and Blaise Tine

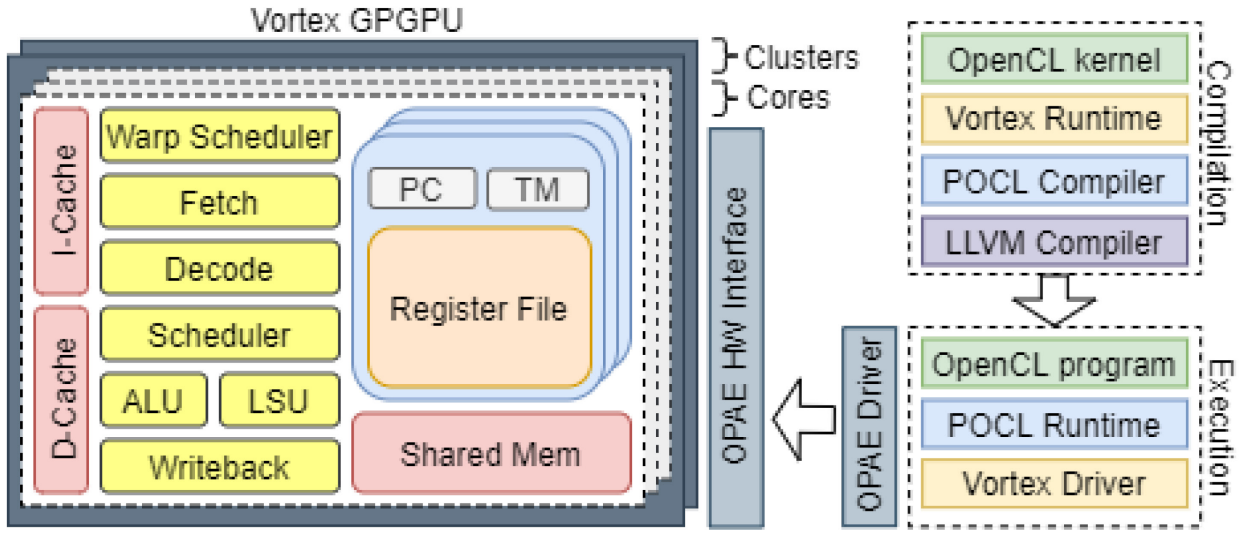
Goals for this session

- We want to explore the wider world of RISC-V accelerators to show what exists and what might be possible
  - Focus on one such complex implementation, Vortex, an open-source RISC-V GPGPU
  - Dive deeper into the world of more complex and open-source RISC-V processor implementations
  - Explore some possible implementations for post-Moore computing research

Vortex GPGPU

Vortex is a open-source SIMT-based GPU processor that

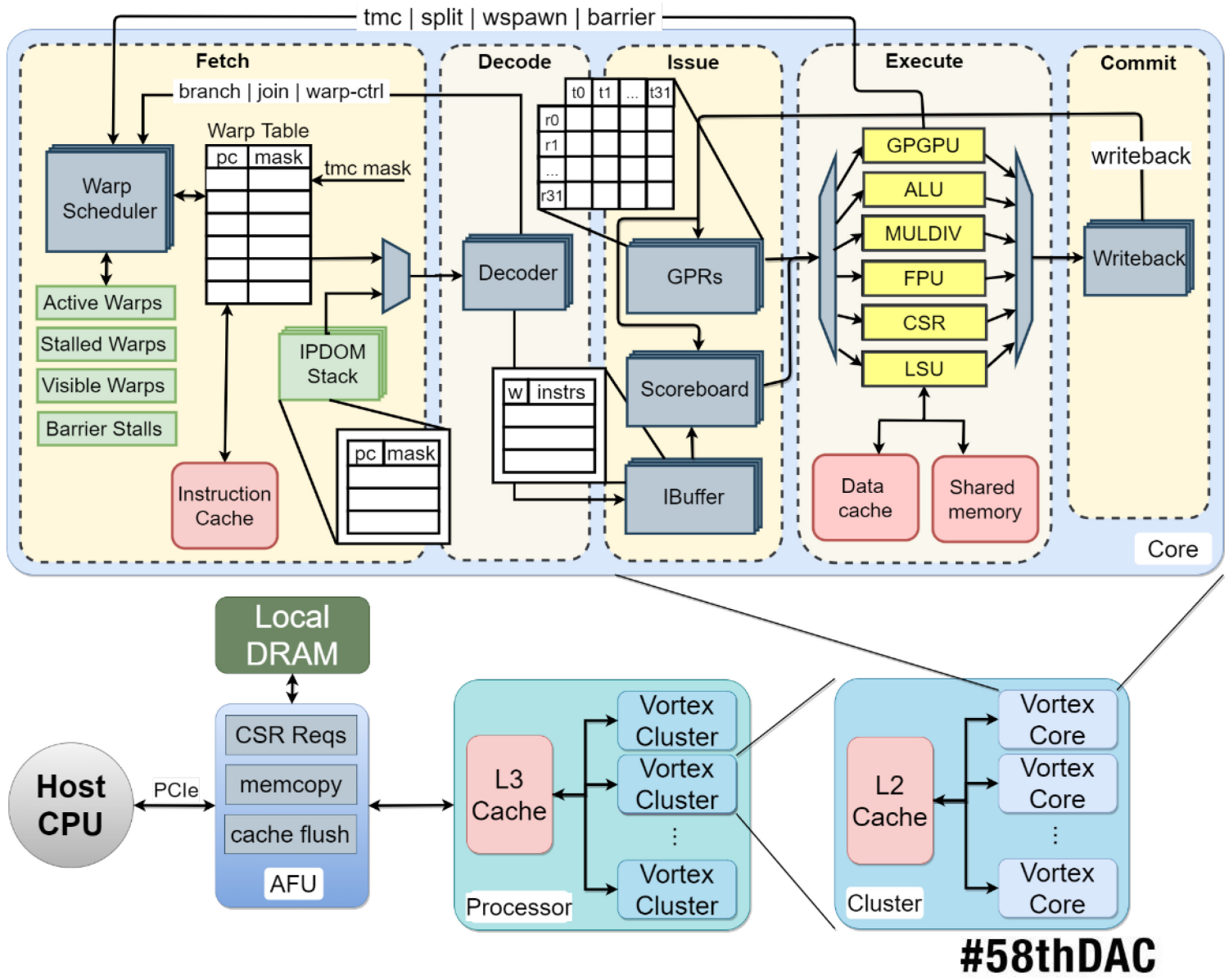
- Supports OpenCL program execution
- Has portable driver APIs
  - FPGA, ASE, RTLSim, SimX
- Uses an Open-source Toolchain
  - RISC-V: ISA support
  - POCL: OpenCL Compiler & runtime
  - OPAE: FPGA Driver API
  - Verilator: RTL simulation
  - Yosys: Synthesis



Vortex does not fully support the proposed RISC-V “V” extension but it enables SIMT compute using a minimal number of extensions to RV32IM (RISC-V 32 bit integer and multiply extensions)

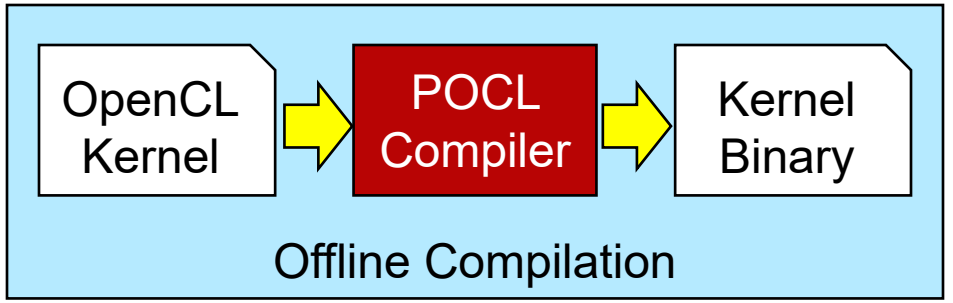
Vortex Microarchitecture

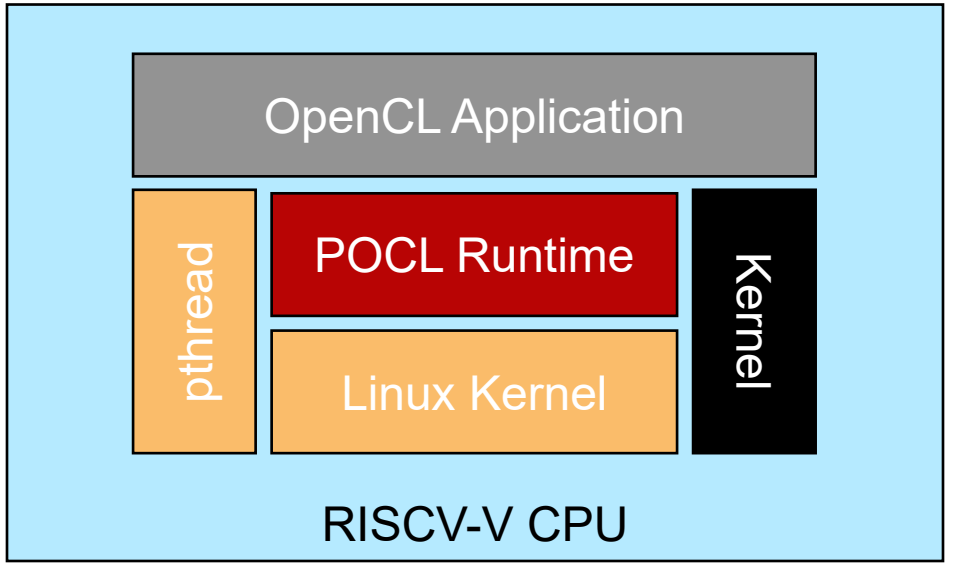
- Five-stage pipeline
  - FI | ID | Issue | EX | WB
- Clustering
  - Group multiple cores into clusters
    - Optionally share L2 cache
  - Group multiple clusters
    - Optionally share L3 cache
  - Configurable at build time
  - Default Configuration:
    - #Clusters: 1 #Cores: 4, #Warps: 4, #Threads: 4
- FPGA AFU Interface
  - Manage CPU-GPU communication
    - Query devices caps
    - Load kernel instructions
    - Load kernel resource buffers
    - Start kernel execution
    - Read destination buffers
  - Local Memory
    - GPU Access to local DRAM
  - Reserved I/O addresses
    - Redirect to host CPU
    - Console output



POCL Support for RISC-V CPUs

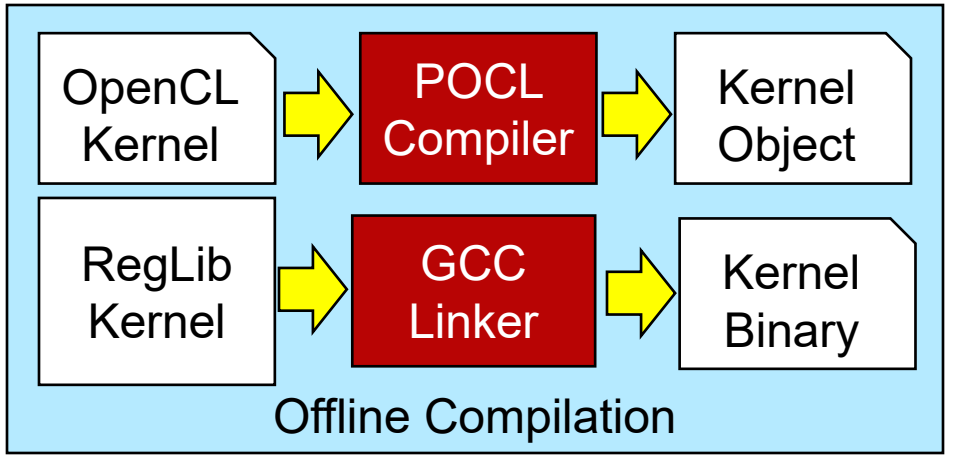
- There is currently no official OpenCL support for RISC-V
- Support for RISC-V can be added with limited changes
  - Clone the existing ARM backend implementation
  - Enable POCL cross-compilation for LLVM-RISC-V
  - Disable device-specific functions
  - Disable device-specific optimizations
- Requirements
  - Linux environment with pthread library
- Limitations
  - Offline kernel compilation only

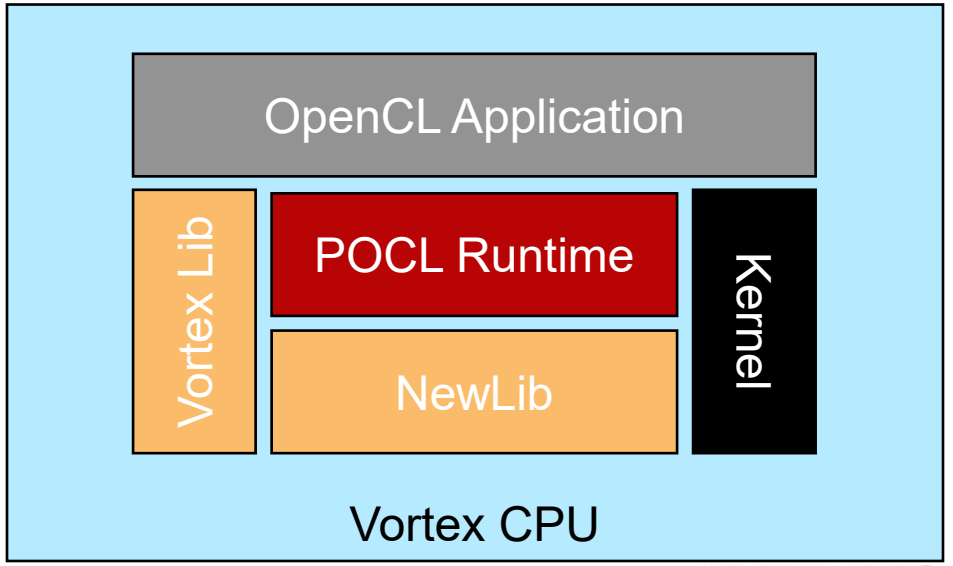




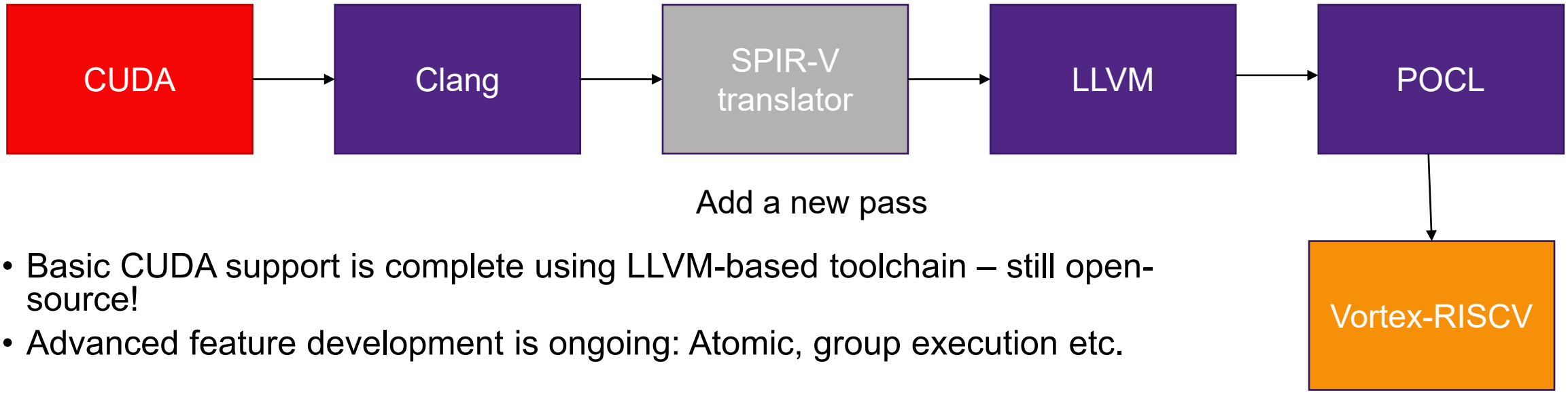
POCL Support for Vortex

- NewLib BSP
  - A barebone std C++ libraries without OS dependency
  - Implement NewLib stub functions for Vortex emulator
    - Memory allocator, File IO and console output
- POCL Runtime
  - Remove linux & pthread dependencies to use NewLib
  - Modify kernel compilation to use load from static library
  - Modify kernel invocation to use Vortex *spawn* instr.
  - Modify linker to generate runtime as static library
- POCL Compiler
  - Enable cross-compilation for RISC-V CPU targets
  - Implement registration API for static kernels (RegLib)
  - Cross-compile kernel to static library





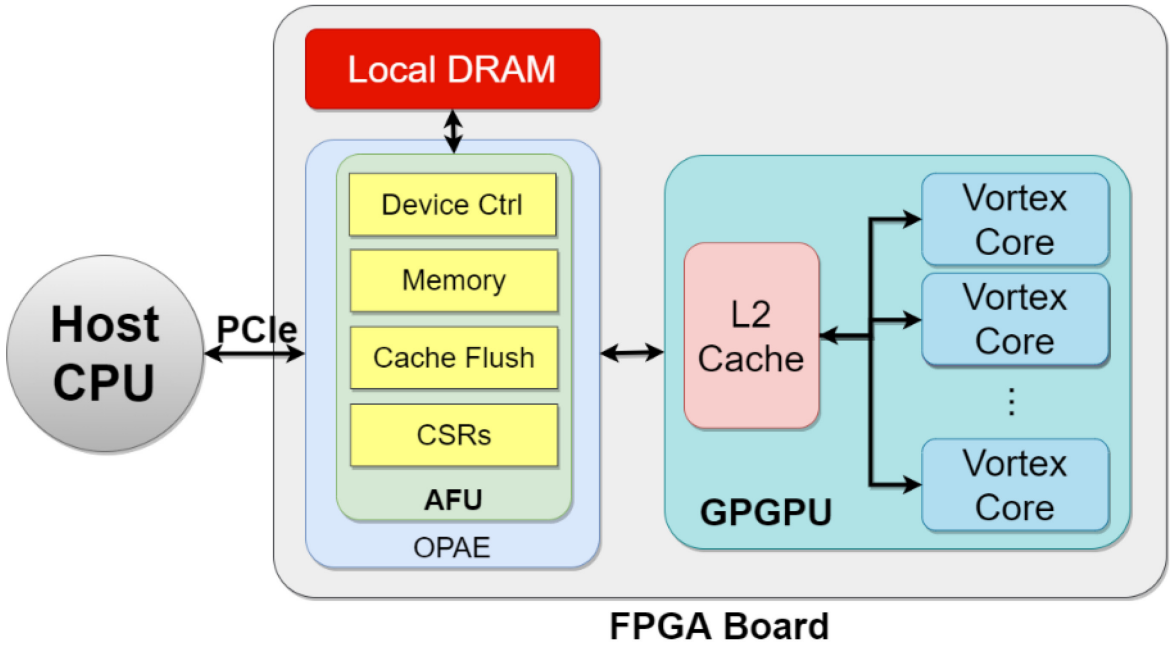
CUDA to RISC-V translation Pass



- Basic CUDA support is complete using LLVM-based toolchain – still open-source!
- Advanced feature development is ongoing: Atomic, group execution etc.

FPGA Driver Architecture for Vortex

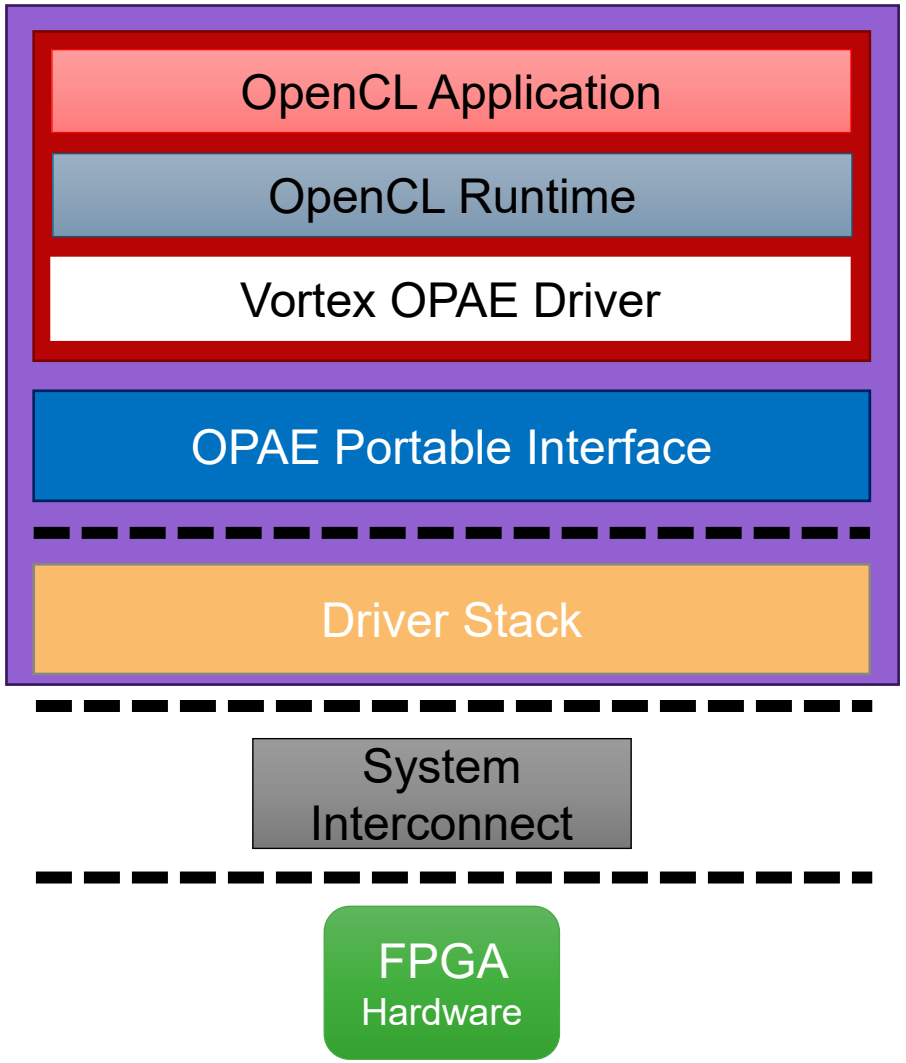
- OPAE[1] Hardware Interface
  - Open Programmable Accelerator Engine
    - Device Independent
      - Altera[1], Xilinx[2]
    - Customizable
      - AFU: Accelerator Functional Unit
  - CPU-FPGA Services
    - CPU requests handling
    - Use MMIO interface
  - Local Memory Services
    - Memory read/write
    - Simplified interface



[1] https://01.org/opae
[2] https://github.com/RSPwFPGAs/opae-xilinx

FPGA Driver Architecture (3)

- Vortex Driver Stack
  - Vortex OPAE Driver
    - Use OPAE Interface to access FPGA
      - Device control
      - Local DRAM access
      - Device cache flush
      - Device CSRs access
    - OpenCL runtime HAL abstraction
    - OpenCL application top layer client
  - OPAE Driver Interface
    - Implement OS Driver support
    - Communication with FPGA hardware
      - via system interconnect



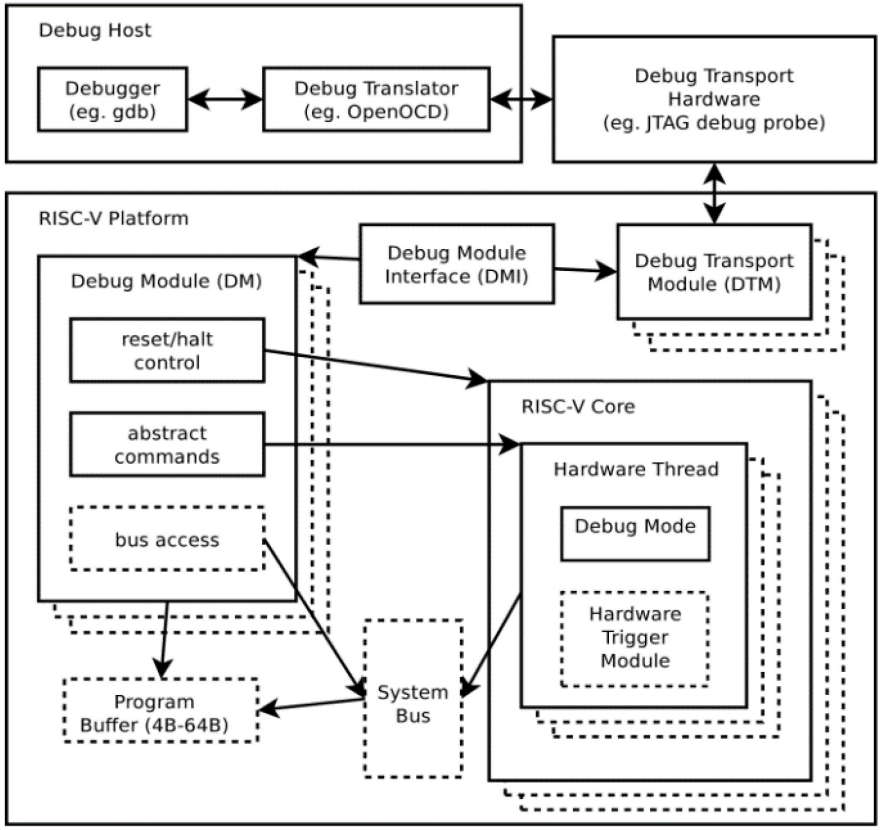
Preliminary Evaluation using Arria 10 Programmable Accelerator

- Experimental Setup
  - Intel PAC Arria 10 GX FPGA with 2x4GB DDR @42133 MT/s
  - Intel Xeon E5-1650 3.5 Ghz CPU
- Vortex Configuration
  - Processor: 4 cores / 4 warps / 4 threads, 16 cores/4warps/4threads
  - Core: I$: 2KB, D$: 4KB, SM: 2KB
- Vortex Synthesis
  - ALMs: 50%, RAM: 40%, REG: 22%, DSP: 17%
- Vortex Performance
  - Clock frequency: 206 – 270 Mhz
  - Peak Memory Bandwidth: 16 GB/s
  - Core Static Power: 2.2 W

Debugging and Testing Support for Vortex

- Support for Hardware Performance Counters
  - Various hardware performance counters are added
    - Current : # of instructions, # cycles
    - On-going: # pipeline bubbles # cache behavior etc.
  - Store the results into CSR
  - Read through CSR register value to access HPC

- (Ongoing) Supporting Debugging features
  - Enable openOCD[1] and GDB debugging for cycle-level simulator and FPGA
  - Similar efforts with RISC-V external debug support
  - Necessary features
    - Selecting harts (RISC-V execution context
    - Halt and resume
    - Single stepping



RISC-V debug system overview [2]

[1] OpenOCD : http://openocd.org/

[2] https://riscv.org/wp-content/uploads/2019/03/riscv-debug-release.pdf

gem5 RISC-V Extensions for Vortex

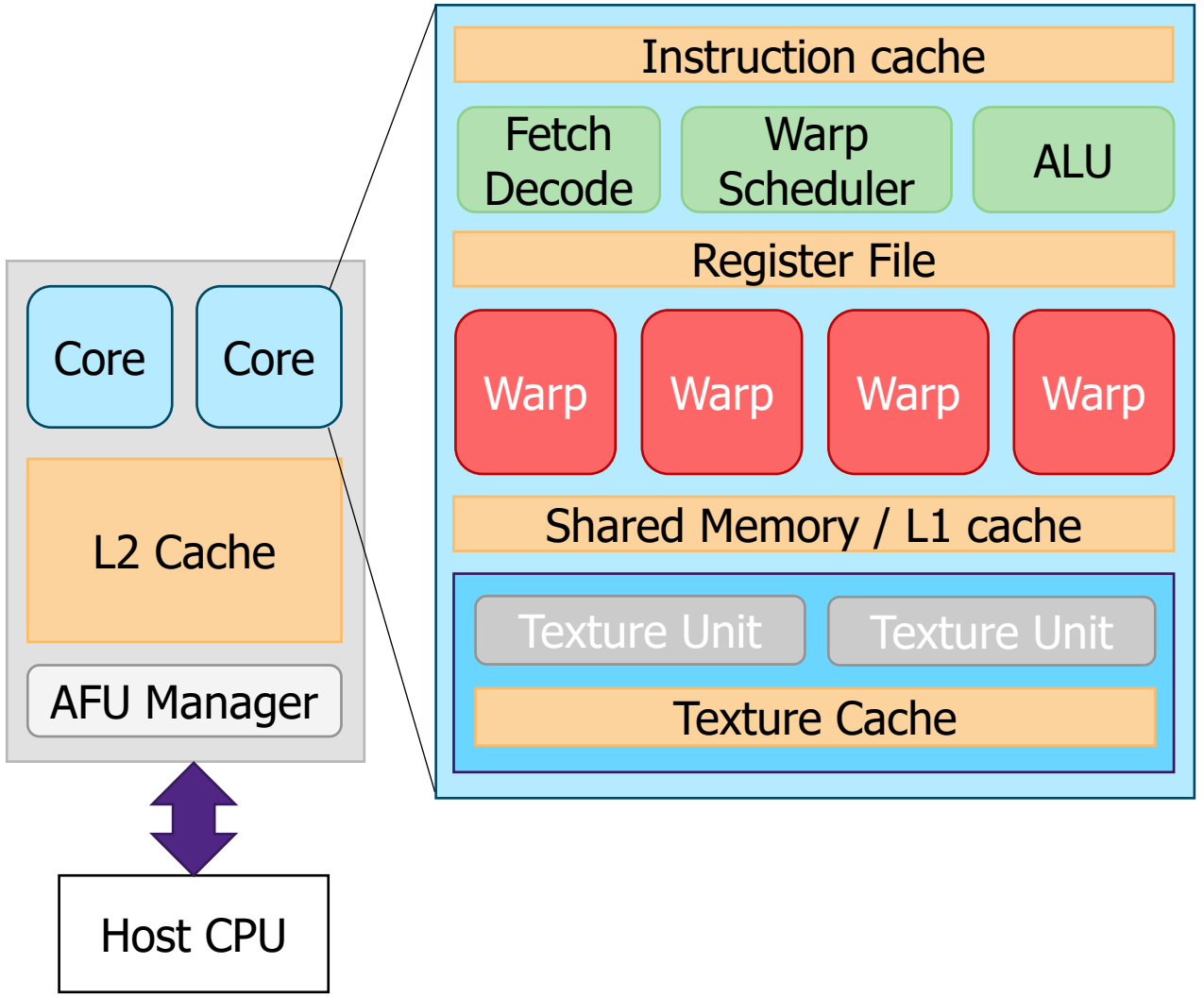
- Allow for simulation of the Vortex SIMT design with RISC-V ISA support in gem5
- Based on Minor CPU Model
- A warp is modelled using ”SimpleThread”
- Initialize with 32 simple threads to model 32 warps. All the warps will be pointing to the same workload

```
for (unsigned i = 0; i < numThreads; i++) {
    if (FullSystem) {
        thread = new SimpleThread(this, i, p->system,
                                  p->itb, p->dtb, p->isa[0]);
        thread->threadMask[0] = true;
    } else {
        thread = new SimpleThread(this, i, p->system, p->workload[0],
                                  p->itb, p->dtb, p->isa[0]);

        thread->threadMask[0] = true;
    }
    threadInfo.push_back(new SimpleExecContext(this, thread));
    threadInfo[i]->warpNumber = i;
    ThreadContext *tc = thread->getTC();
    threadContexts.push_back(tc);
}
```

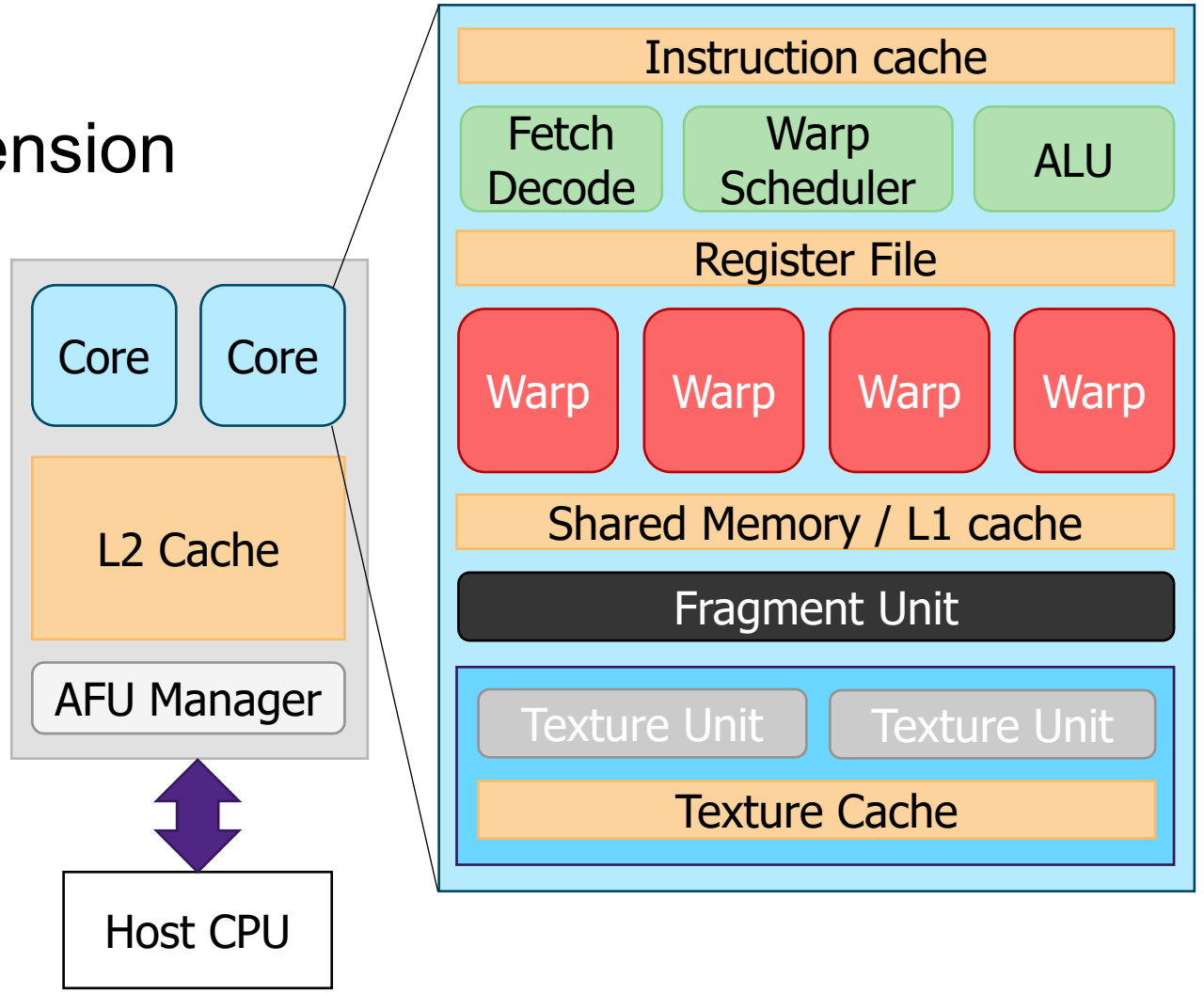
Vortex GPU Architecture, V1

- V1: “Fixed-Function” GPU Extension
  - OpenGL-ES 1.1 API
  - No support for shaders
    - Fixed-Function Pipeline
    - Implemented in Software
      - Geometry processing
      - Pixel processing
      - Fragment processing
  - Tiled-Based Rendering
    - Software implementation
  - Texture Acceleration
    - 2 Texture Units per core
    - A shared Texture cache
  - Frame buffer support



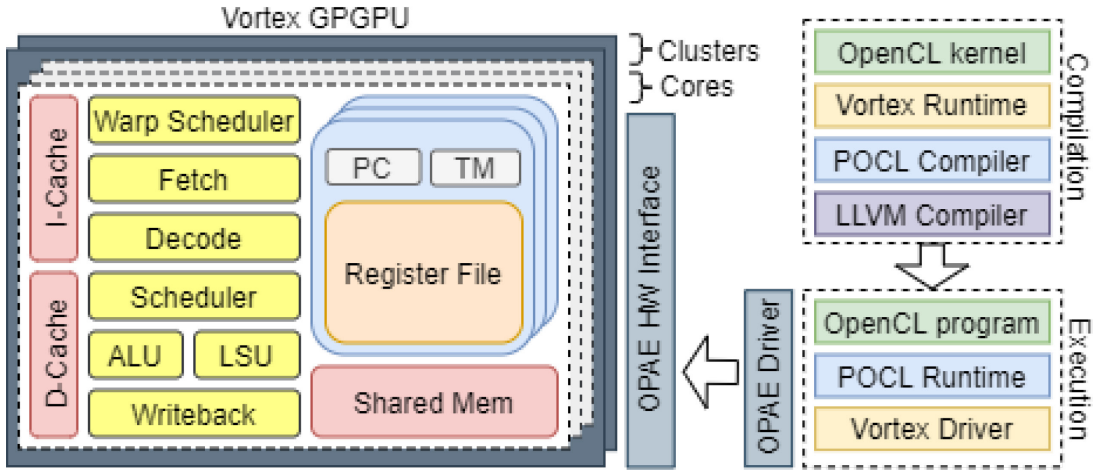
Vortex GPU Architecture, V2

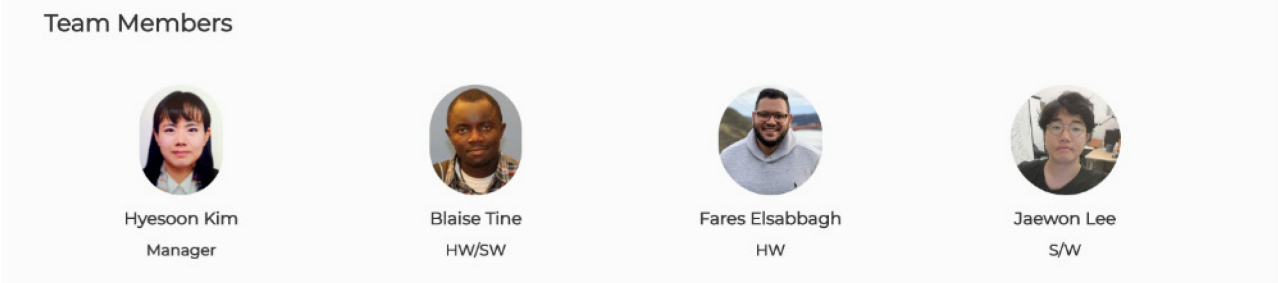
- V2: “Programmable” GPU Extension
  - Vulkan API
  - Support for shaders
  - SPIR-V shader support
    - LunarG: GLSL => SPIR-V
    - POCL: SPIR-V => RISC-V
  - Hardware Expansion
    - Fragment processing
    - Tiled Buffer (shared memory)



Learn More About Using Vortex For Your Own Work!

- vortex.cc.gatech.edu
- https://github.com/vortexgpgpu/
  - Includes POCL with newlib and Vortex support
- Check out the recent MICRO tutorial materials at https://github.com/vortexgpgpu/vortex_tutorials





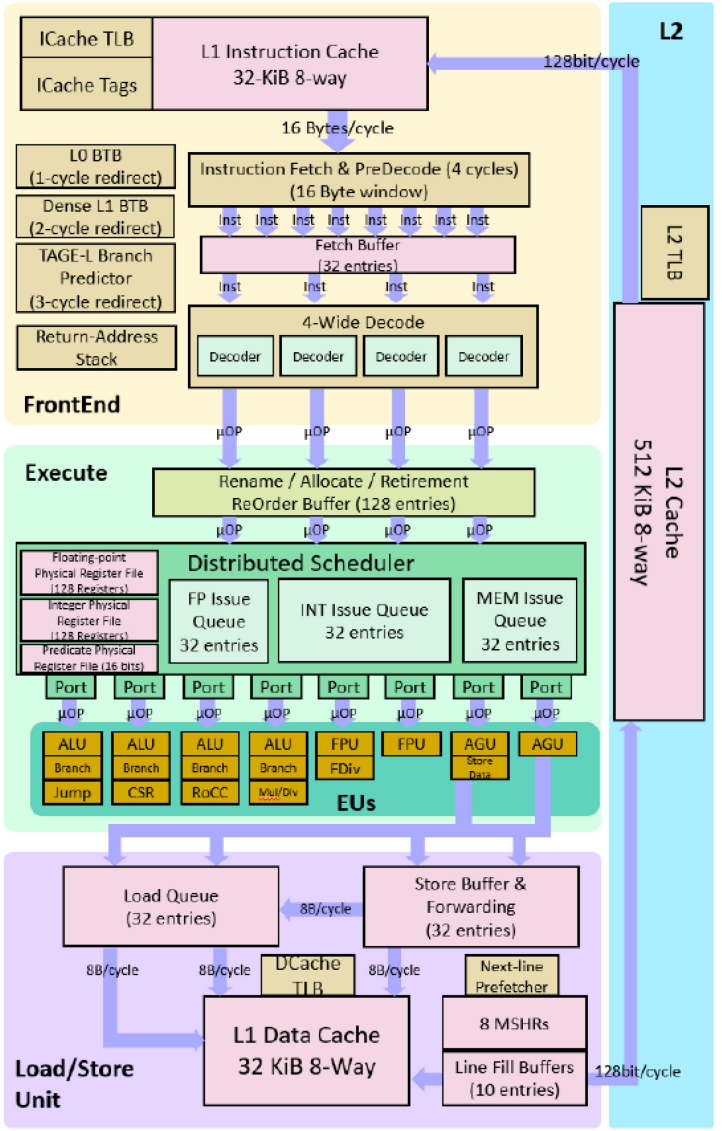
State-of-the-Art for RISC-V Processors and Accelerators

- Processor – Boom v3
- Gemmini – systolic array GEMM implementation
- Hwacha – research processor for RISC-V
- PULP Variants – PULPino, xPulpNN, PsPIN, NeuroStream
- Many others that we don’t have time to explore in depth!

State-of-the-Art – BOOM v3

Arguably the most advanced open-source RISC-V CPU, SonicBOOM [1] integrates new features a reworked load-store unit, Tagged geometric (TAGE) branch predictor, dual-ported L1 cache and improved prefetching.

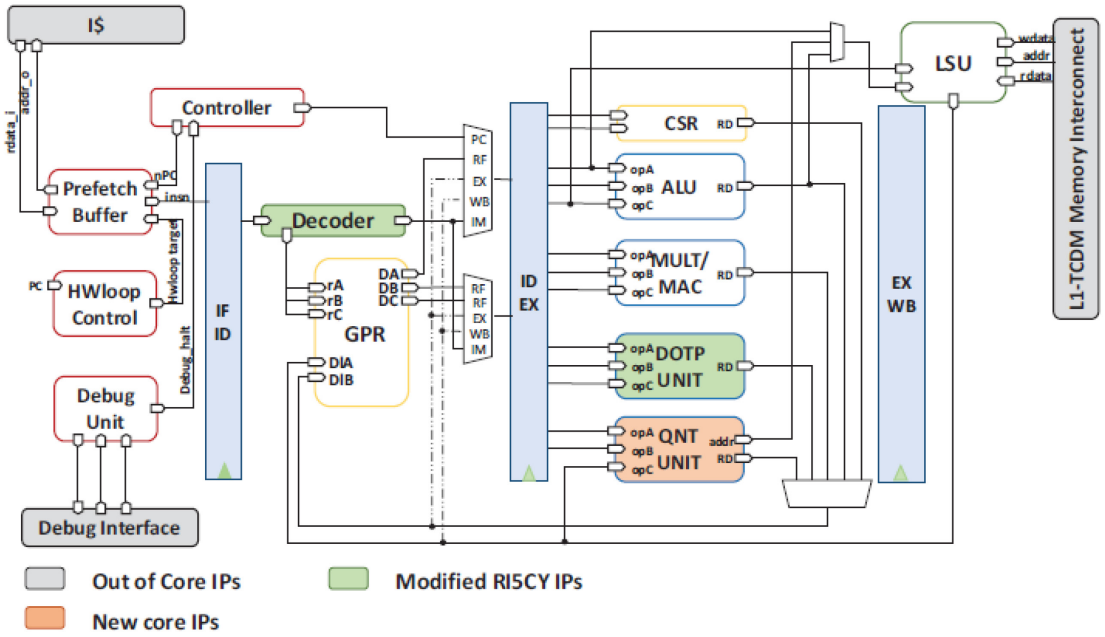
These improvements lead to CoreMark performance similar to an AWS Graviton processor and 2x improvements on many measured benchmarks [2].

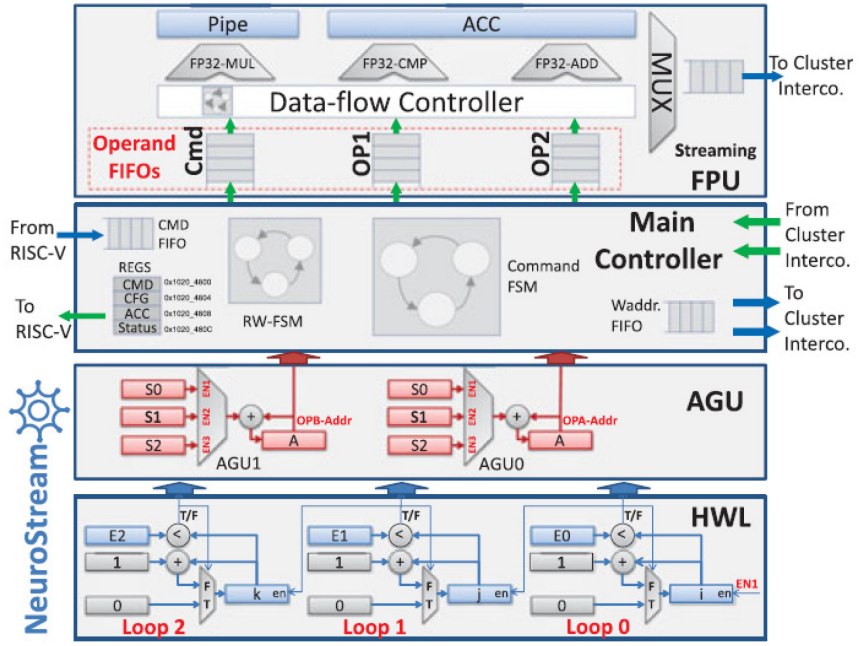


[1] Jerry Zhao, *et al.*, SonicBOOM: The 3rd Generation Berkeley Out-of-Order Machine, CARRV 2020

[2] Alexander Dörflinger, *et al.*, A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations, Computing Frontiers, 2021

State-of-the-Art – xPulpNN, NeuroStream



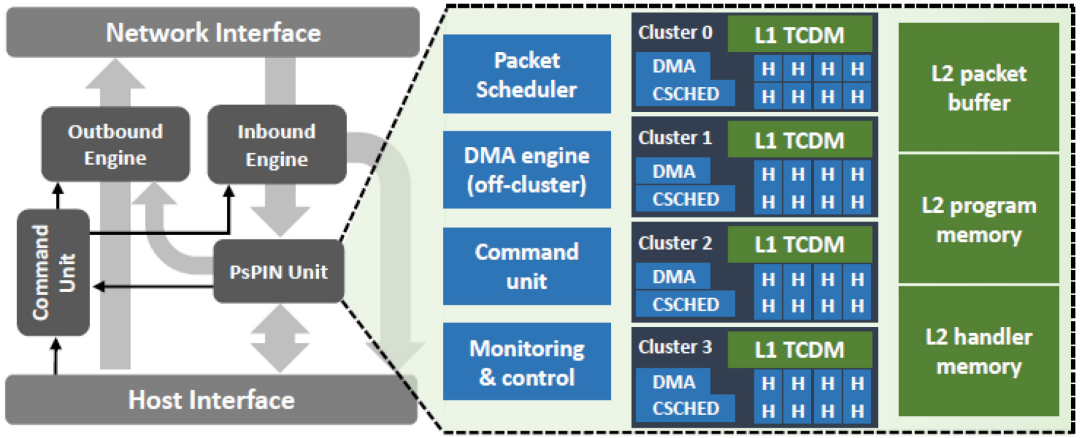


xPulpNN [1] extends the RI5CY pipeline with ISA, tooling, and RTL support for quantized neural networks while NeuroStream [2] integrates clusters of RISC-V CPU and FPU coprocessor modules into a smart memory cube fabric.

[1] Angelo Garofalo, *et al.*, *XpulpNN: Accelerating Quantized Neural Networks on RISC-V Processors Through ISA Extensions*, DATE 2020

[2] Erfan Azarkhish, *et al.*, *Neurostream: Scalable and Energy Efficient Deep Learning with Smart Memory Cubes*, TPDS 2018

State-of-the-Art – PsPIN



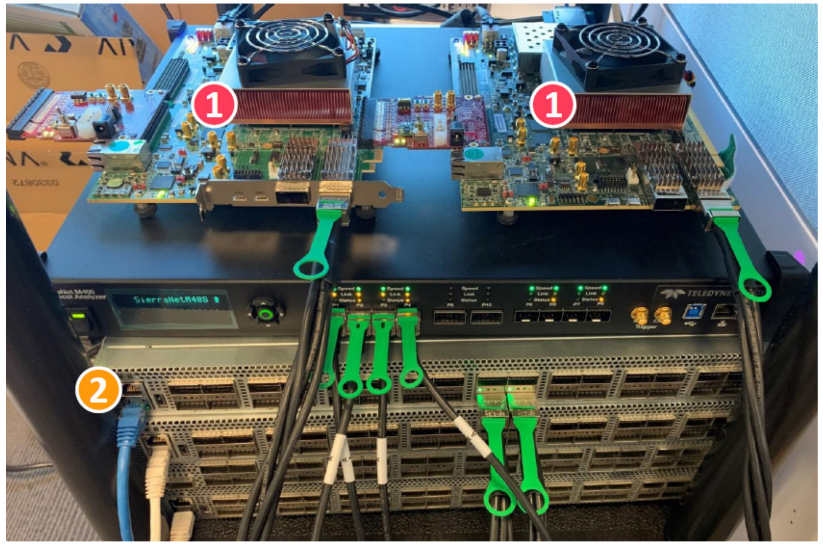
PsPIN [1] combines elements from the PULP ecosystem and the sPIN [2] smart networking project by implementing packet handlers in the NIC using Handler Processing Units (HPUs) implemented with light-weight RISC-V cores and and shared scratchpad memories. This design is meant to be low-latency and parallel to handle upcoming 400 Gbps networking environments.

Related efforts like PANIC [3] have looked at implementations and evaluations of RISC-V RV32 cores as streaming co-processors in the NIC.

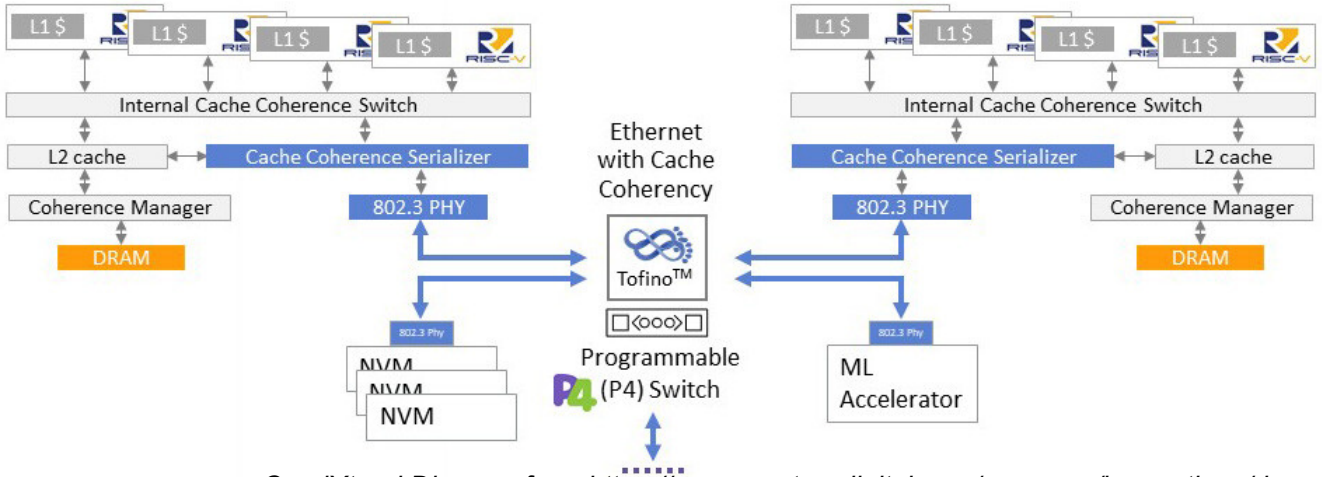
[1] Salvatore Di Girolamo, *et al.*, *PsPIN: A high-performance low-power architecture for flexible in-network compute*, https://arxiv.org/abs/2010.03536, 2020
[2] Torsten Hoefler, *et al.*, *sPIN: High-performance streaming Processing In the Network*, SC 2017
[3] Jiaxin Lin, *et al.*, *PANIC: A High-Performance Programmable NIC for Multi-tenant Networks*, OSDI 2020

State-of-the-Art – Industry Open-source Designs

In addition to closed-source CPU implementations by SiFive, Andes, Alibaba and others there are other interesting open-source implementations currently coming to market like Western Digital's **SweRV** [1] for disk drive controllers that integrates TileLink coherence protocols for its OmniXtend protocol.



OmniXtend Testbed from[2]



OmniXtend Diagram from https://www.westerndigital.com/company/innovations/risc-v

[1] https://github.com/chipsalliance/Cores-SweRV

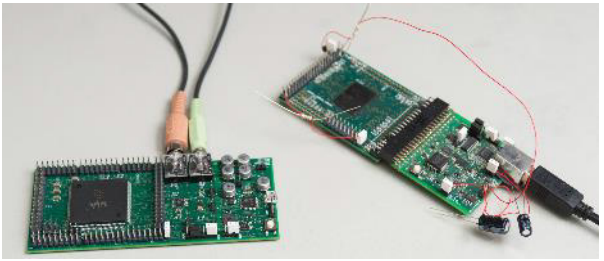
[2] Marjan Radi, et al., *An Open and Coherent Memory Centric Architecture Enabled by RISC-V*, 2nd RISC-V Summit, 2019

RISC-V for post-Moore Computing

Post-Moore computing refers to the period that is *post-Moore's Law*; that is when we can no longer get **performance**, **power**, and **cost scaling** using smaller transistor technology nodes.

It is speculated that this era will see an explosion of *application-specific custom accelerators* in categories like:

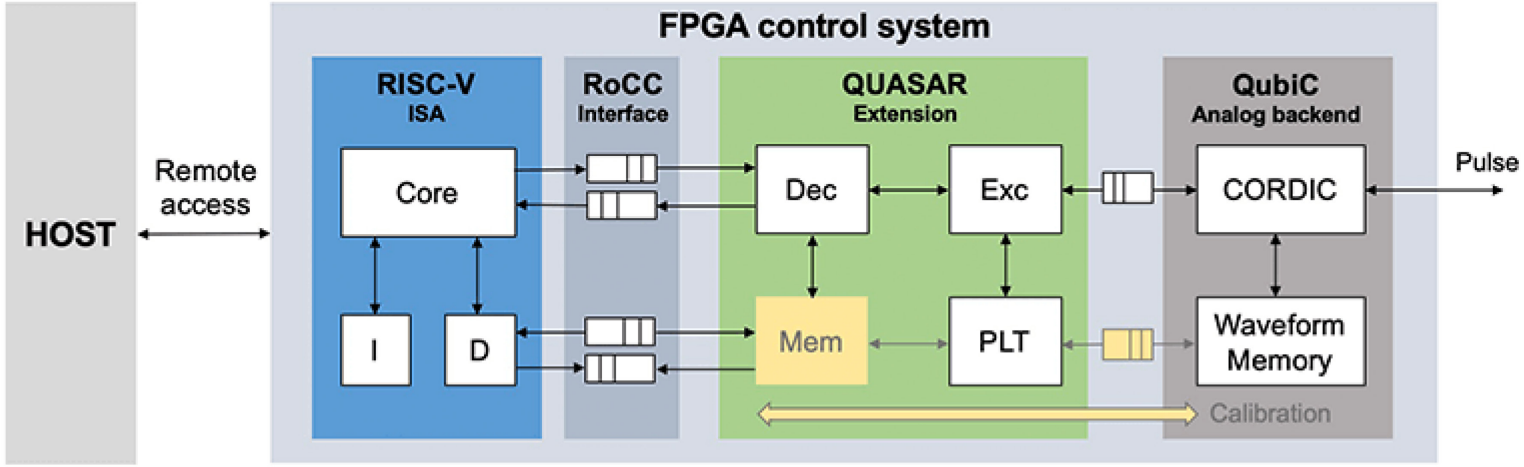
- Neuromorphic or brain-inspired computing
- Quantum computing
- Highly reconfigurable hardware and interconnects
- Reversible and thermodynamic computing
- Approximate Computing



FPAA Neuromorphic Platform from Dr. Jennifer Hasler, Georgia Tech

We will likely first see RISC-V ISA show up in control processors, coprocessors for SNN devices (e.g. Intel Loihi), or as simple processing elements implemented in new logic families to implement reversible processors.

RISC-V for post-Moore Computing – Quantum Control



Recent work from Lawrence Berkeley National Lab (LBL) [1] looked at the design of a quantum ISA to enable classical control processors to effectively encode quantum circuits into pulse-level commands.

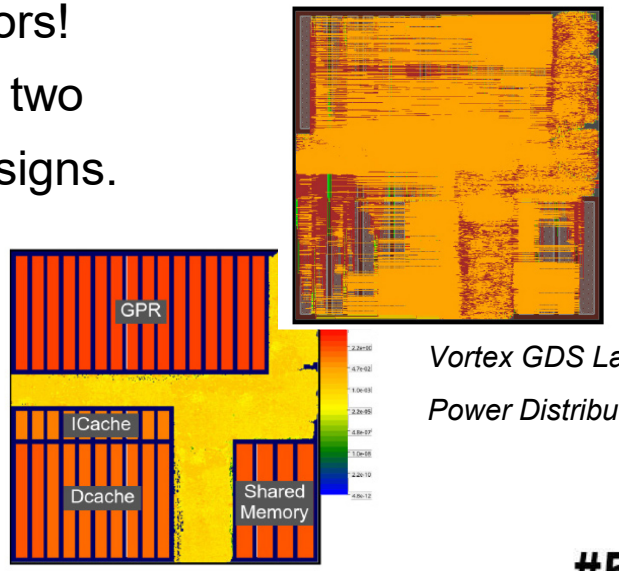
- The authors propose QUASAR, a scalar extension to RISC-V RV32 ISA and qV, a double-indexed vector extension and compare to RV32 and eQASM implementations for synthetic and realistic circuits.
- Both QUASAR and qV show effectiveness for encoding quantum circuits with some trade-offs in the complexity of the RISC-V hardware support for the control processor

[1] Anastasiia Butko, et al., *Understanding Quantum Control Processor Capabilities and Limitations through Circuit Characterization*, 2020 International Conference on Rebooting Computing (ICRC)

Wrap-up

- Most current work is still focused on high-performance RISC-V CPU implementations
- However, there are emerging solutions for novel RISC-V ISA accelerators including GPGPUs, neuromorphic (ML) accelerators, storage and network controllers, and control processors for post-Moore systems
  - This heterogeneity will lead to requirements for an improved ecosystem of tools, simulators, and benchmarks that can be used across different kinds of accelerators!
  - System Architect and Vortex's OpenCL compilation workflow are two examples of open-source tools to help address novel RISC-V designs.

CoreGen repo: https://github.com/opensocsysarch/CoreGen
Tutorial resources: https://github.com/gt-crnch-rg/riscv-tutorial-dac-21



Vortex GDS Layout and Power Distribution