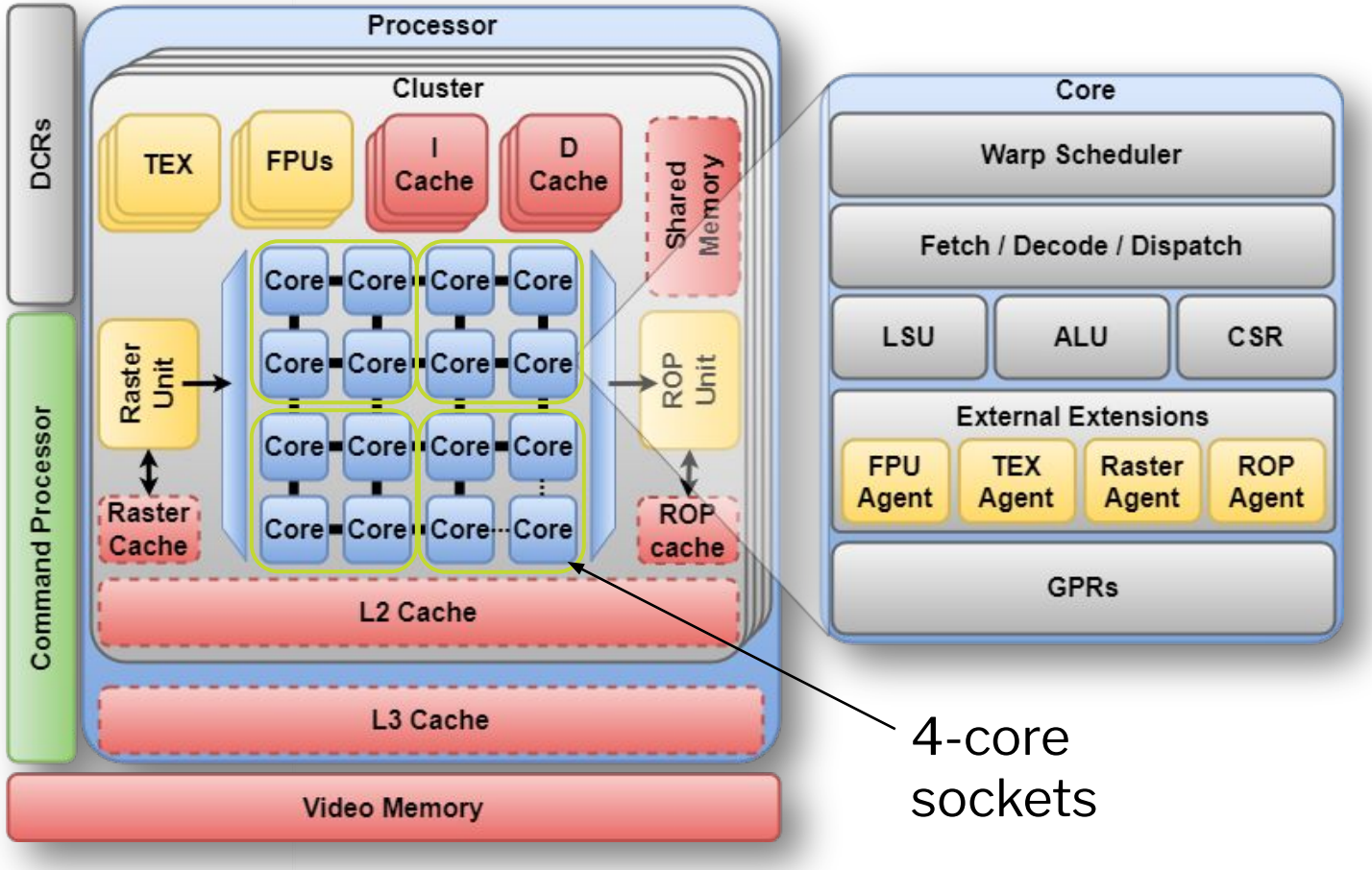
Speeding-Up RISC-V GPGPU Accelerator using High-Bandwith Memory on FPGA

Blaise Tine, Hyesoon Kim

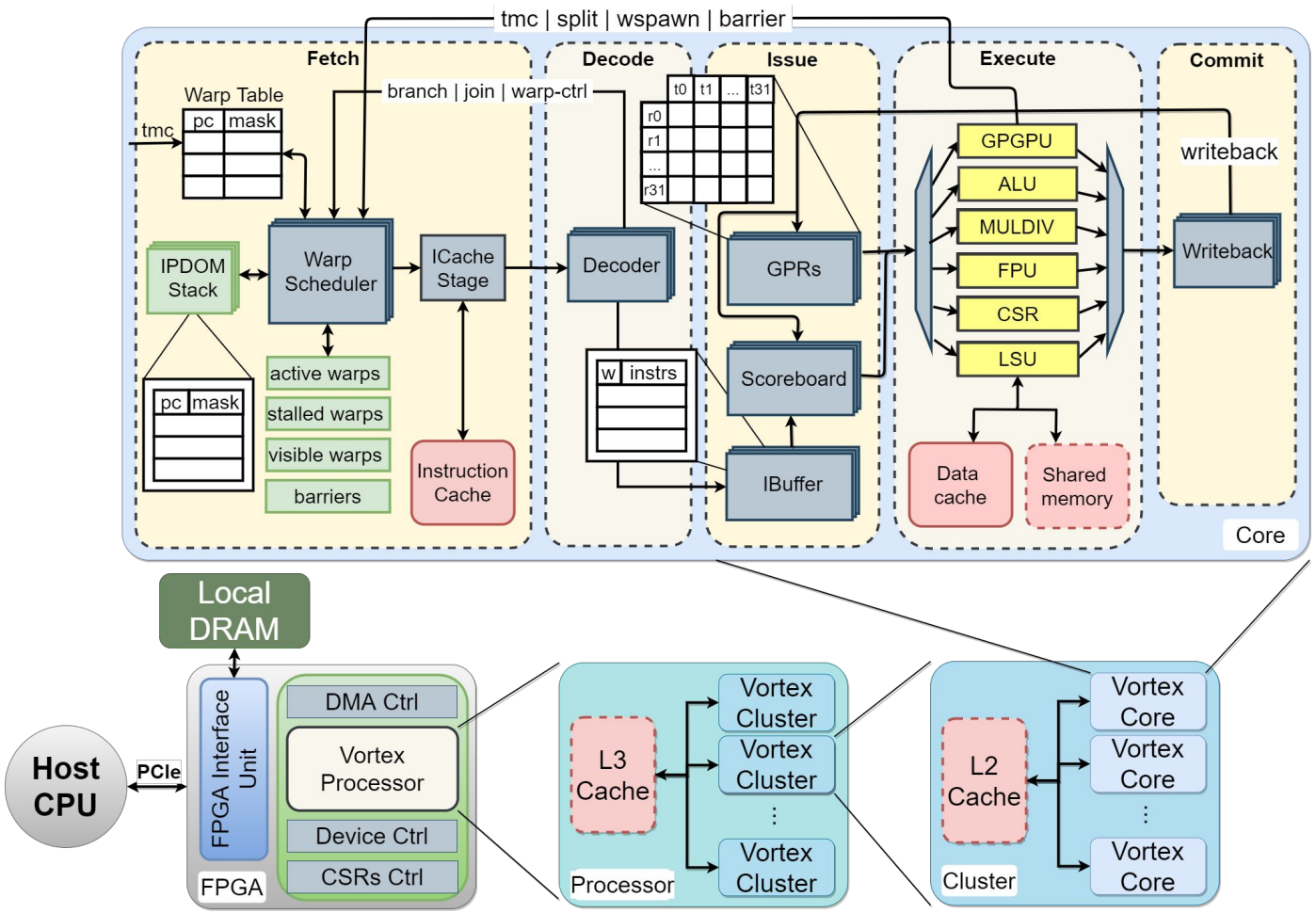
Background: Vortex Open-Source GPU

https://vortex.cc.gatech.edu



Background: Vortex Open-Source GPU

- Five-stage RISC-V Pipeline
  - IF | ID | Issue | EX | Commit
  - In-order issue / OoO commit
  - Single-precision FPU
  - Wavefront Scheduler
  - Register File Expansion
  - Instruction Buffer
  - High-bandwidth Caches
  - Shared Memory
  - Cores Clustering



Background: Vortex Open-Source GPU

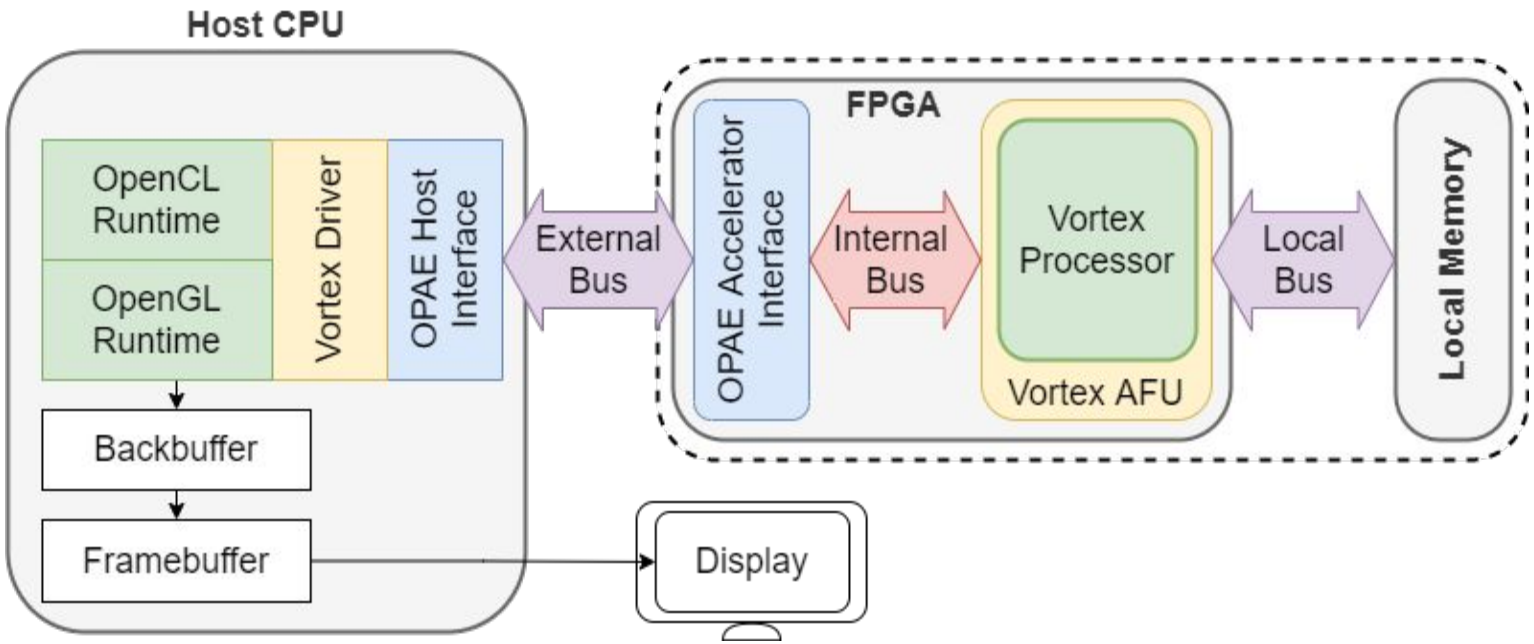
- Altera FPGA Synthesis
  - Single core (4 wavefronts x 4 threads)
  - Up to 32 cores: 512 total threads
  - Caches: (16KB I$, 16KB D$)
  - Shared Memory: 16KB
  - Intel PAC Arria 10 GX FPGA with 2x4GB DDR
  - Clock frequency: 200-234 Mhz
  - Peak Memory Bandwidth: 16 GB/s

Motivation: Porting to Xilinx HBM

- High-Bandwidth Memory Attributes
  - Large capacity memory storage
  - 32 Multi-banked parallel transfer
- Existing Vortex GPU Memory System
  - L1, L2, L3 Multi-banked Cache hierarchy
  - 4 Multi-banked DDR4 memory
- Target FPGAs
  - Xilinx Alveo Cards

A portable platform for FPGA synthesis

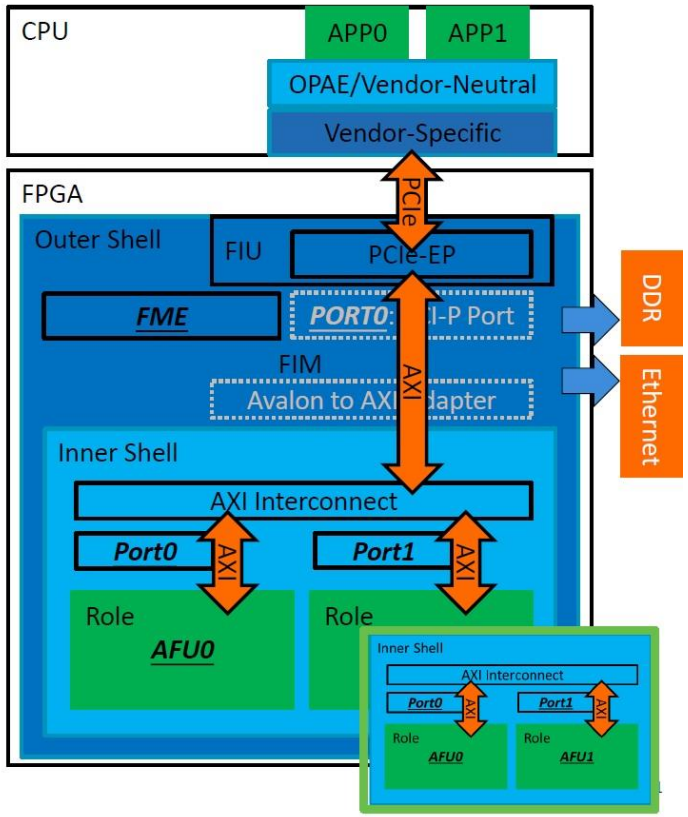
- Vortex OPAE driver stack
  - OPAE SW Interface
    - Runtime API
    - Kernel Driver
  - OPAE HW Interface
    - PCIe Controller
    - Memory Controller
    - Host-Device Interface



Intel OPA:
https://opae.github.io

Platform-independent Xilinx solutions

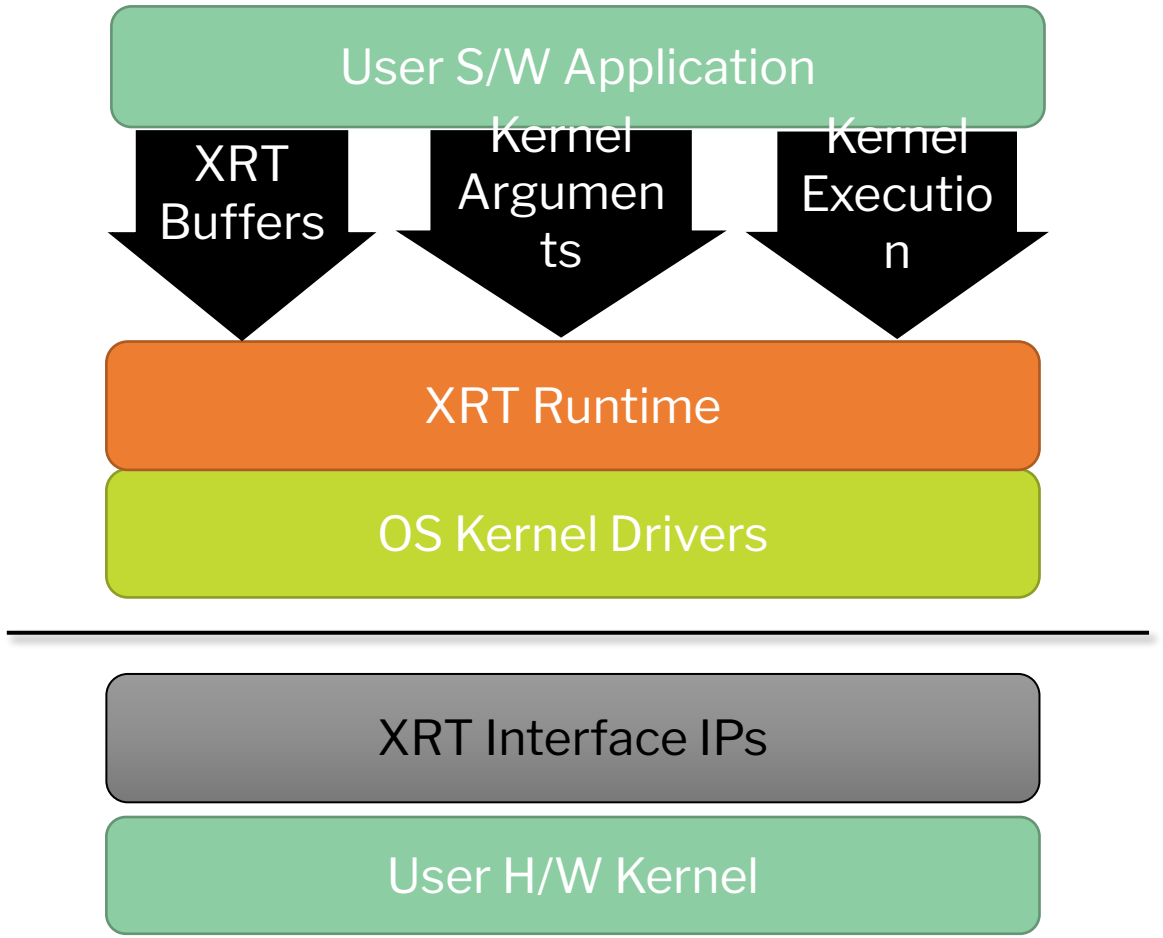
- OPAE-Xilinx
  - Avalon to AXI adapter overhead
  - Old devices, DDR only
- TAPASCO
  - Kernel-based accelerators
  - https://github.com/esa-tu-darmstadt/tapasco
- Xilinx XRT
  - Kernel-based accelerators
  - https://xilinx.github.io/XRT



OPAE-Xilinx:
https://github.com/RSPwFPGAs/opae-xilinx

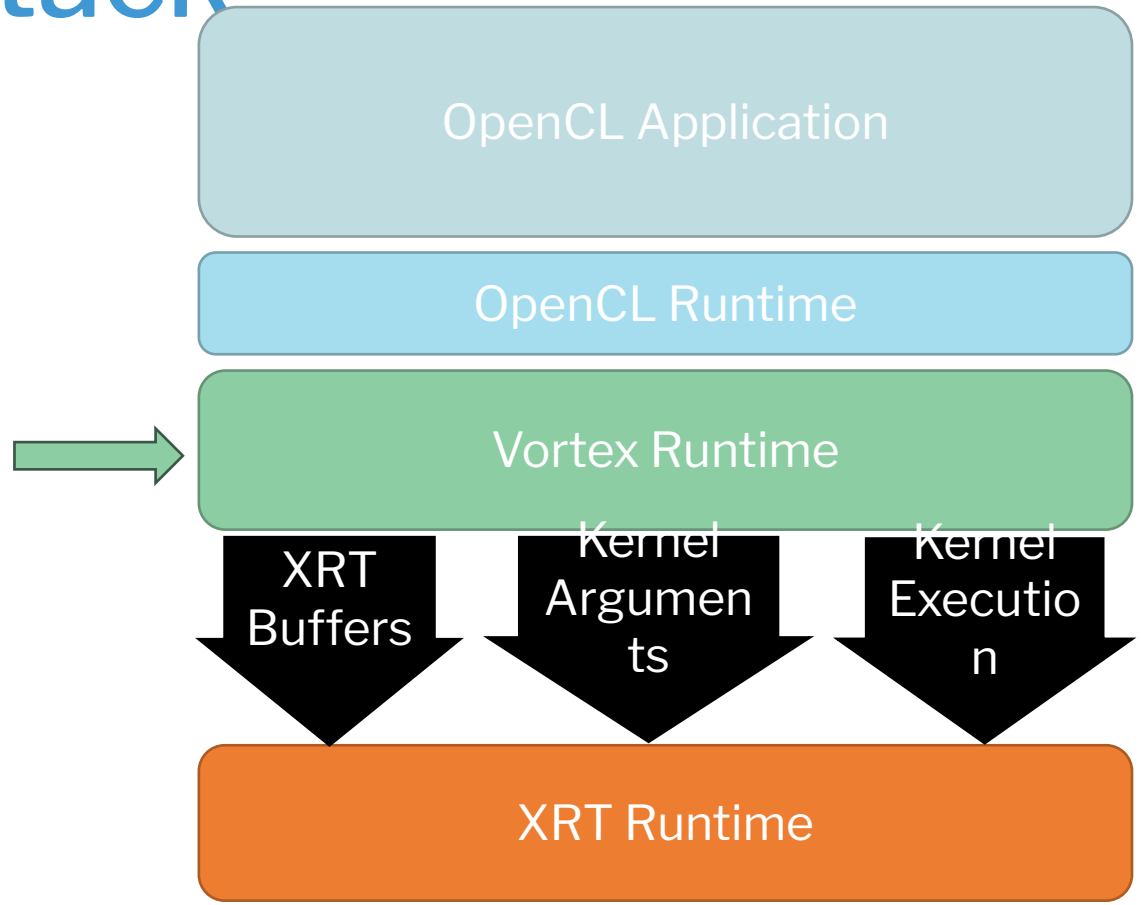
Xilinx XRT Framework

- S/W Abstraction
  - Runtime S/W interface
    - XRT Buffers
    - Kernel Arguments
    - Kernel Execution
  - Hide: PCIe and other I/O drivers
- H/W abstraction
  - Runtime H/W interface
    - AXI memory bus
    - Kernel controller (start, busy, etc…)
  - Hide: PCIe, Memory controller



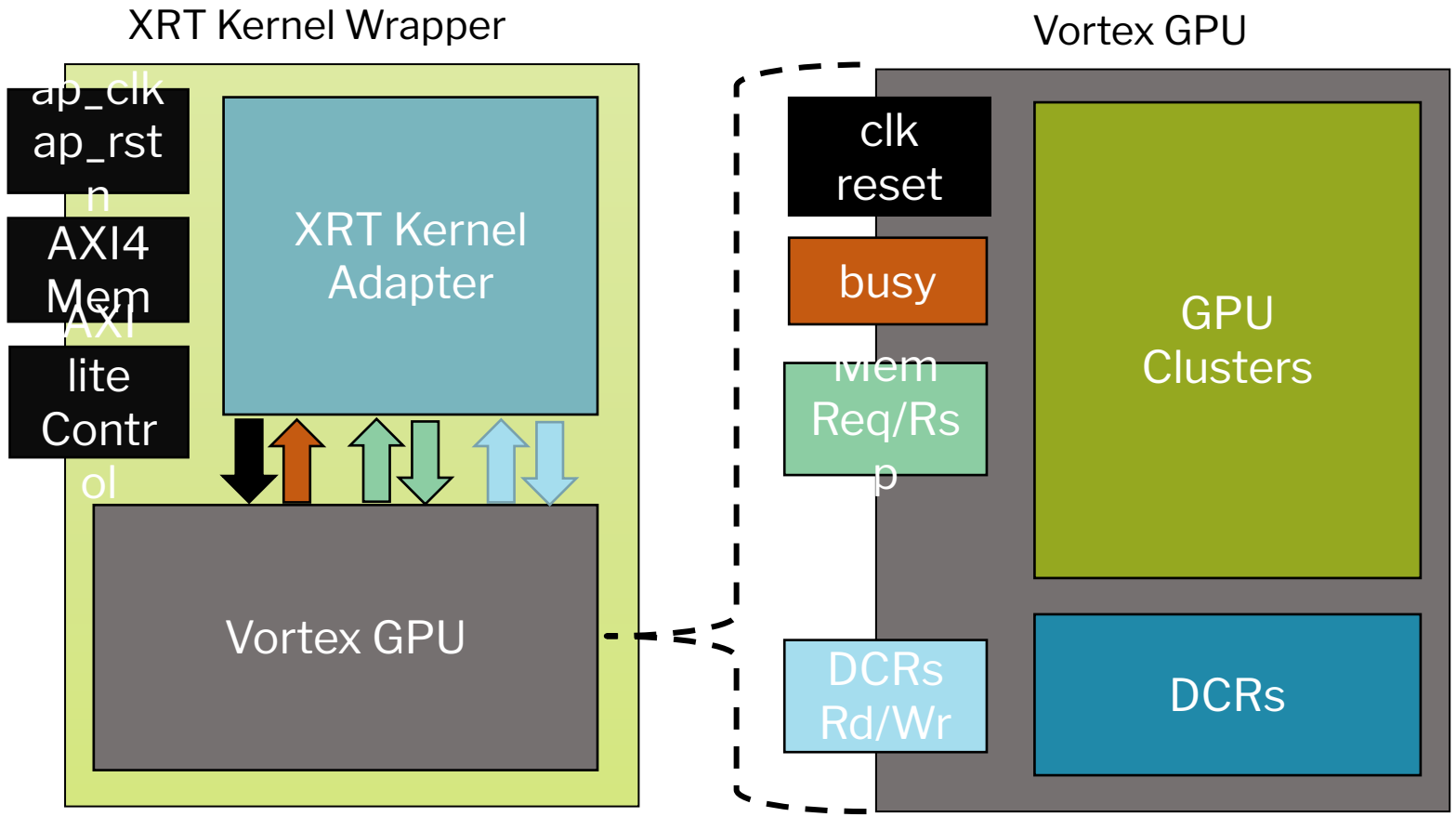
Vortex XRT Runtime Stack

- XRT Buffers
  - Dynamic Buffer allocation
  - Buffer reuse and recycling
  - Handle GPU program transfer
  - Handle GPU memory transfer
- Kernel Arguments
  - GPU device query
  - GPU device configuration
- Kernel Execution
  - GPU device execution
  - GPU status query



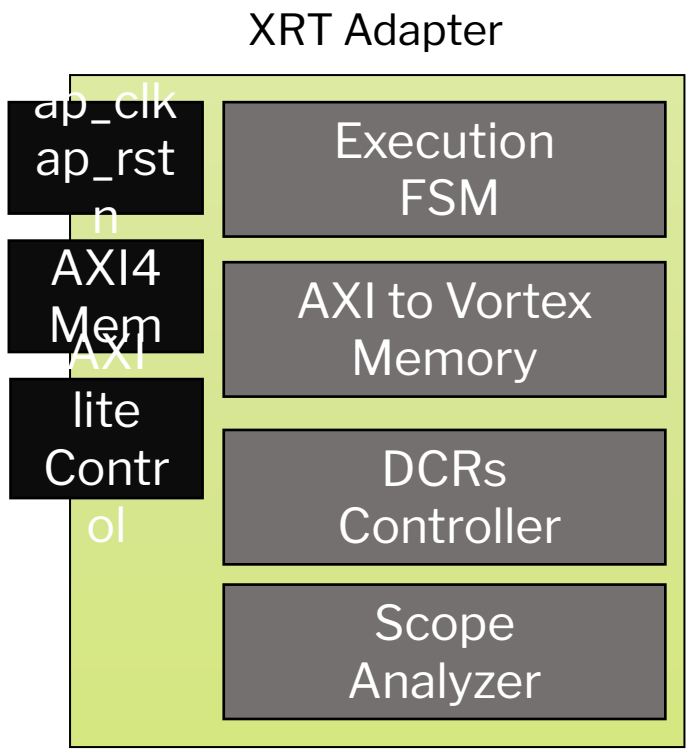
Vortex XRT Hardware Stack

- AXI4 Memory Bus
  - Multi-channels
  - HBM interface
- AXI Lite Control Bus
  - Kernel Arguments
  - Kernel Execution
- XRT Kernel Adapter
  - Vortex DCRs
  - Vortex Memory
  - Vortex Execution



Vortex XRT Adapter

- Execution FSM
  - Manages start/busy
- AXI to Vortex Mem
  - AXI adaptor
  - Multi-bank support
- DCR controller
  - Configuration
  - Query
- Scope Analyzer
  - Custom core



Evaluation

- Target FPGA Xilinx U50
  - Intel Xeon E5-1650 CPU
  - Xilinx Alveo U50
  - 37 Gb/s BW
  - Rodinia benchmark
- GPU Specs
  - Up to 8 cores
  - 150-300 Mhz
  - 19.2 GB/s BW
  - 4 wavefronts per core
  - 4 threads per wavefront
  - Caches: 48KB

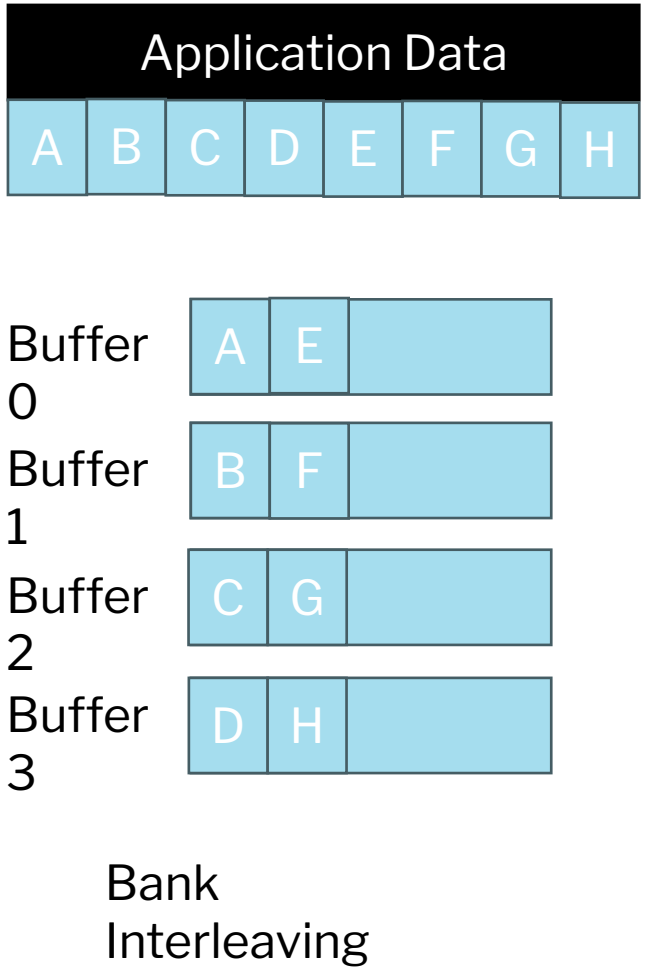
Evaluation

- Synthesis
  - No adapter overhead
  - XRT runtime overhead
- Compared to OPAE
  - Intel Arria 10 FPGA
  - 0.6% adapter LUTs
  - 12% OPAE Ips LUTs

	Lut	Registers	BRAMs	DSPs
Vortex	16%	7%	10%	2%
XRT Adapter	0.2%	0.05%	0%	0%
XRT IPs	8%	4%	7%	0.02%
Total	24%	11.05%	17%	2.02%

XRT Limitations

- Buffer Interface
  - Per-bank allocations
  - Bank-interleaving complexity
  - Buffer interface S/W overheard
  - Need lower-level API
- Kernel Arguments
  - 32-bit registers, 2 transfers for DCRs
- Xilinx SLR constraint
  - Large designs crosses SLR boundaries
  - SLR I/O overhead limits scaling
  - Manual partitioning adds complexity to S/W



Conclusions

- HW/SW XRT extension to support HBM
  - New XRT extension for custom processors
  - Showcase implementation for Vortex GPU
  - Open-source SW and HW implementation
- Research Impact
  - Enabling large memory-bound GPU workloads
  - Exploring HBM-centric GPU memory architectures

Website: https://vortex.cc.gatech.edu

Github: https://github.com/vortexgpgpu/vortex

THANK YOU!