UCLA Samueli School of Engineering

Towards "True" GPU Performance Scaling for OpenGPU

Blaise Tine - UCLA
Hyesoon Kim - Georgia Tech

VORTEX

Abstract

General-Purpose Graphics Processing Units (GPGPUs) have gained significant attention for their high computational throughput and energy efficiency. Their ability to offer high-performance, general-purpose programmability makes them ideal for accelerating diverse applications in fields such as artificial intelligence, scientific computing, healthcare, financial modeling, and computer graphics.

The Vortex OpenGPU introduced the first open-source full-system GPGPU, extending the RISC-V base ISA to introduce a Single-Instruction-Multiple-Threads (SIMT) execution model. In this work, we extended the OpenGPU microarchitecture into a configurable superscalar pipeline, migrating it from a CPU-oriented microarchitecture - where the performance scaling was centered on increasing the number of GPU cores - to a GPU-oriented microarchitecture where the performance scaling is centered towards scaling the parallelism inside a single core. We evaluated the design on Intel FPGA, achieving a 29% performance increase with a 4-wide single-core 32 threads GPU versus an area-equivalent 16-core processor.

Motivations

Current OpenGPU scaling limitations

- Core scaling is inefficient beyond 4 warps & 4 threads
- Scoreboard critical path increases
- Register file overhead due to Block RAM replication
- Memory bandwidth bottleneck
- Single-port writeback bottleneck

Goals:

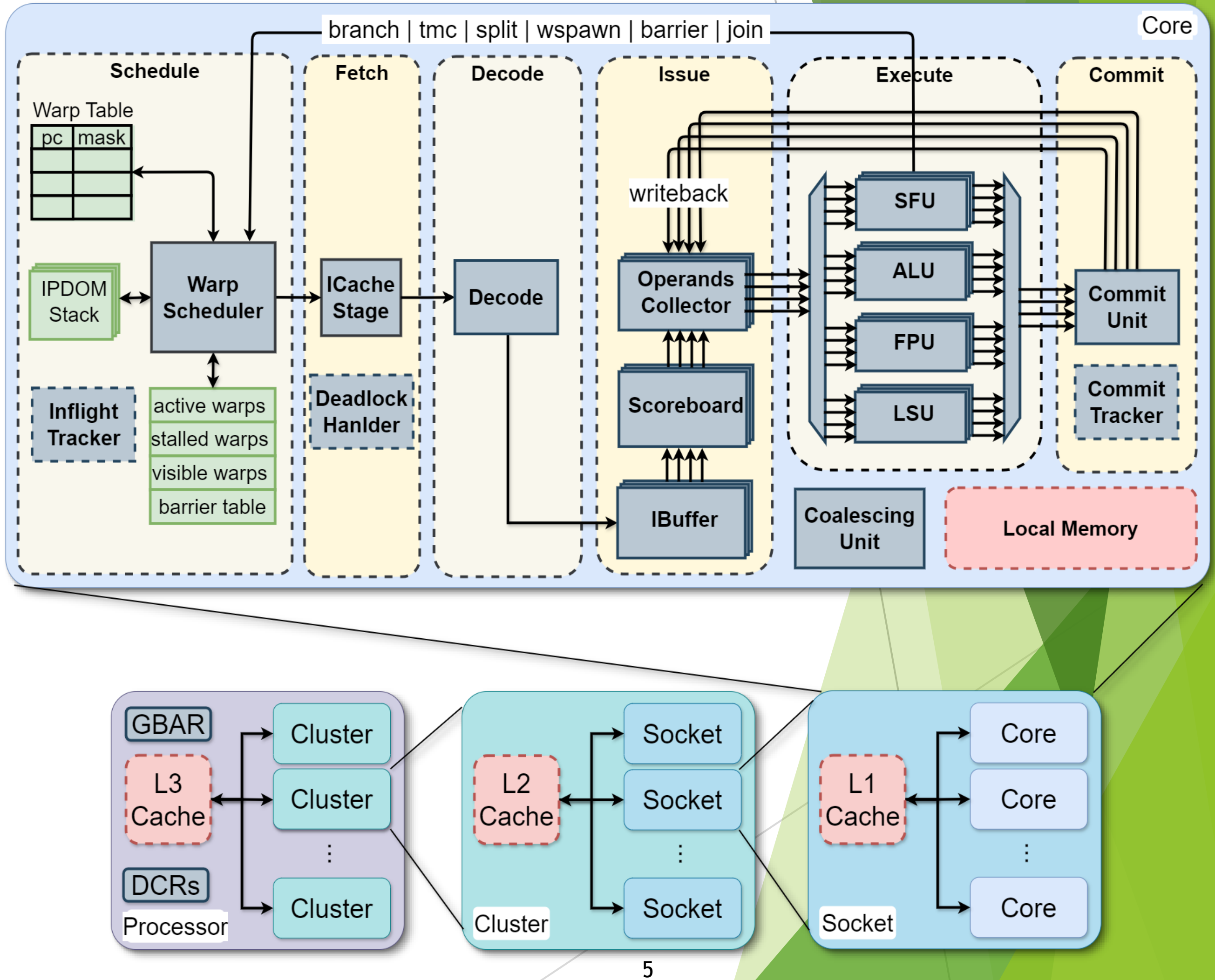
- Scaling the issue front-end throughput
- Scaling the compute pipeline throughput
- Scaling the memory bandwidth
- Enabling area-vs-performance h/w configuration

Challenges

- Scaling GPU core's warps and threads challenges
  - Increase resource contention to shared resources
    - Register file, shared memory, caches
  - Per-core synchronization overhead
  - Branch divergence overhead
  - Per-core memory bandwidth increase
  - Fan-out management overhead
  - Synthesis routing complexing

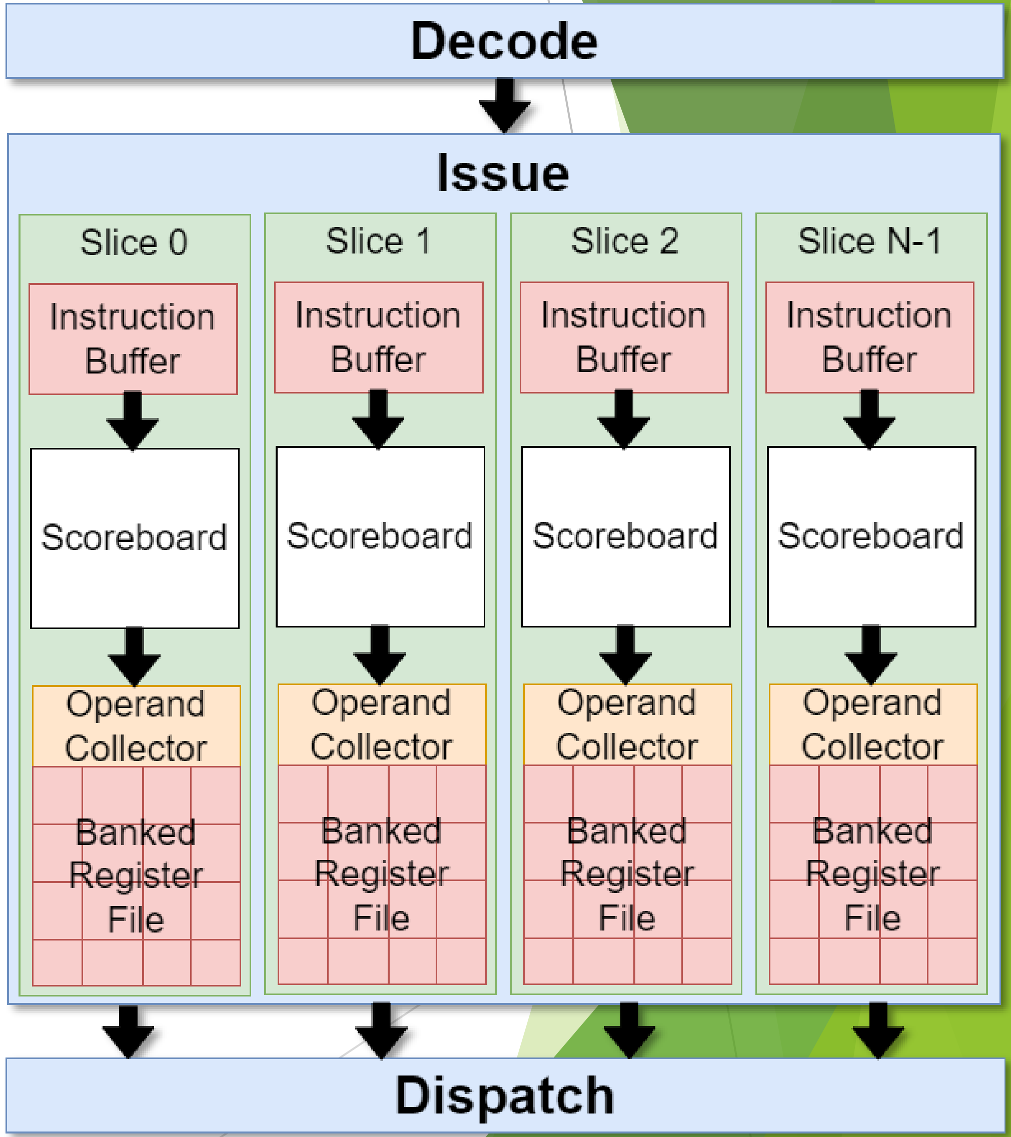
OpenGPU Microarchitecture Extension

- Issue
  - Wide issue width
  - Scalable scoreboard
  - Operands collector
- Execute
  - Configurable FUs arrays
- Commit
  - Multi-ported writeback
- Memory
  - Coalescing unit



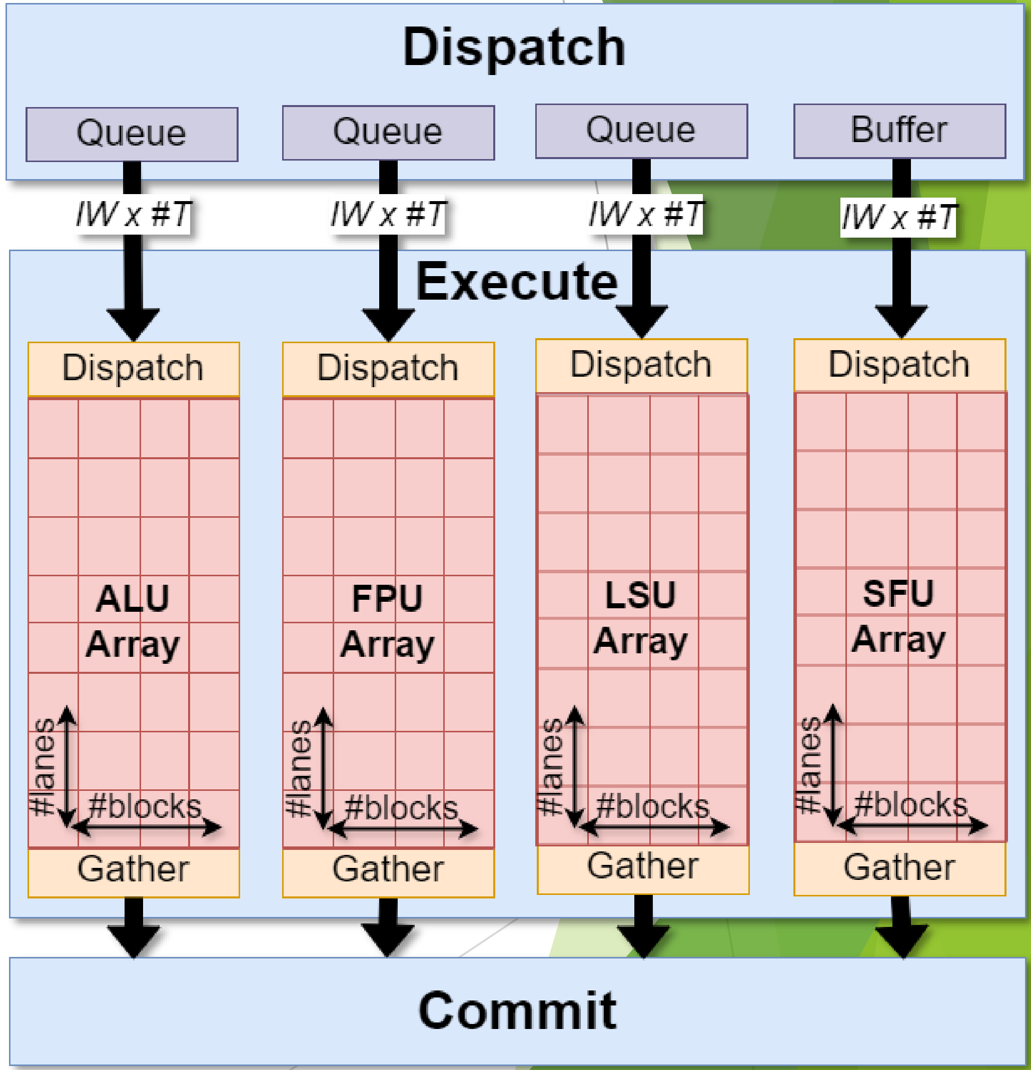
Scalable Issue Front-End

- Parallel Issue Partitions
  - Wide Issue width: **IW**
  - Instruction Buffer
  - Scoreboard
  - Operand Collector
- Operand Collector
  - Non-blocking Pipeline
  - Multi-Banked Register File



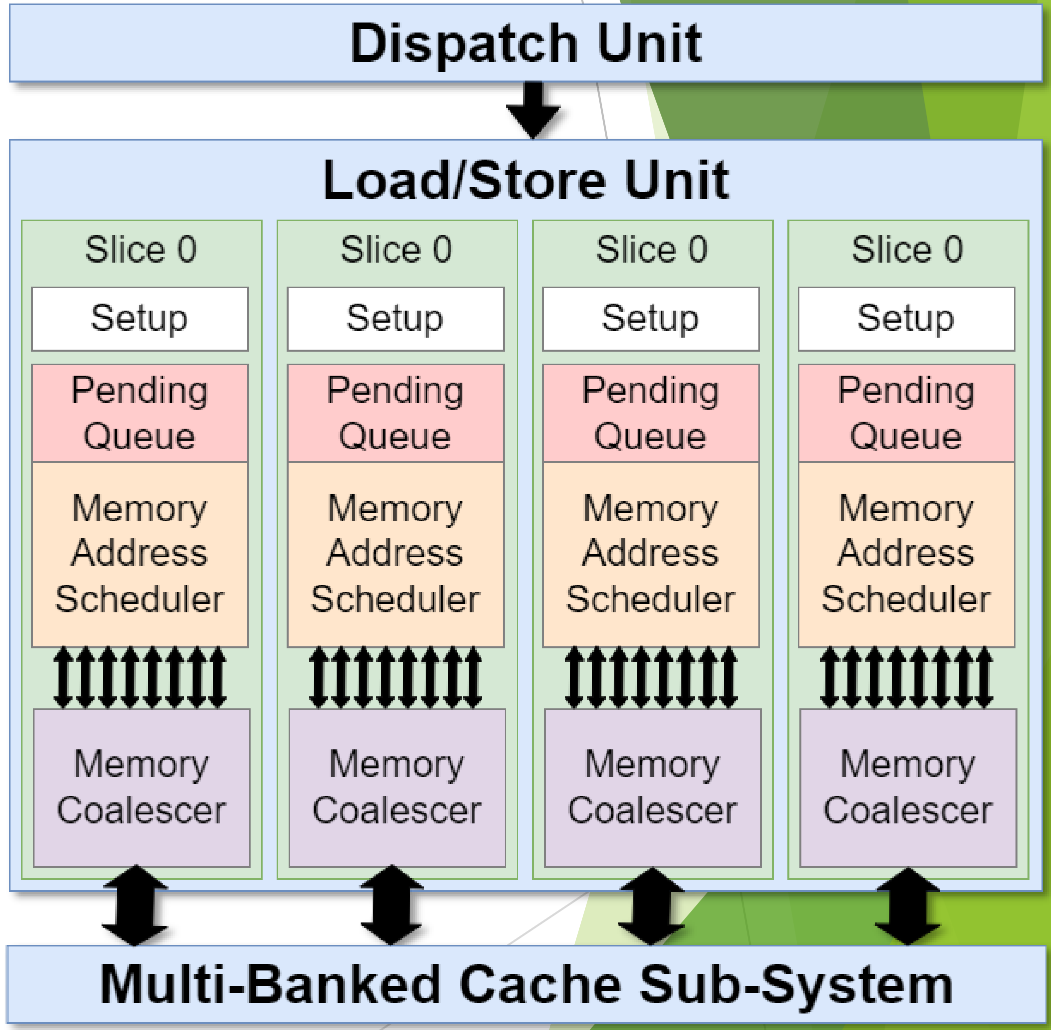
Configurable Compute Throughput

- Systolic Compute Array
  - Dispatch Units
  - Functional Units
  - Gather Units
- Wide Issue Throughput
  - **IW** instructions/cycle
  - **IW** * **NT** threads
- Per-FU Configuration
  - **NB** blocks (NB ≤ IW)
  - **NL** lanes (NL ≤ NT)
  - Throughput: **NB** * **NL** ops/cycle



High-Bandwidth Memory Access

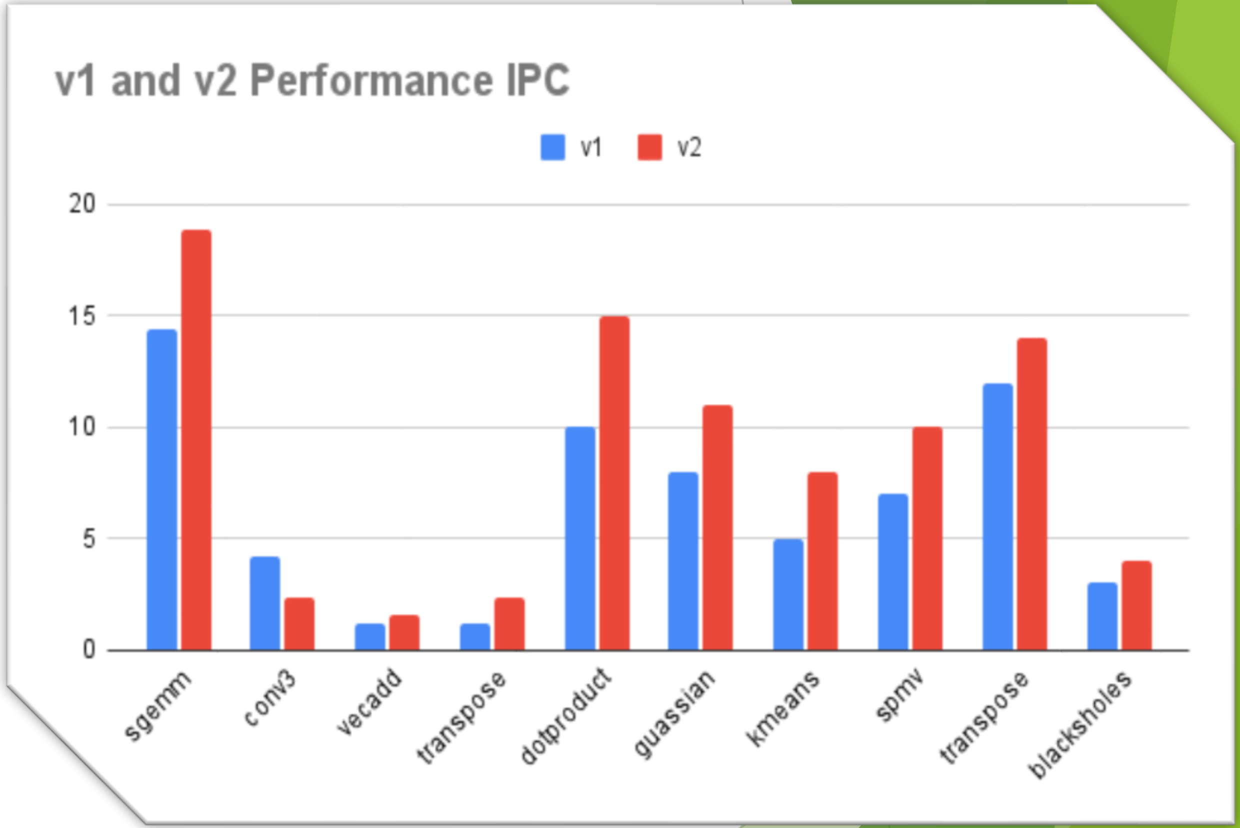
- Parallel LSU Partitions
  - Per-Block Slices
  - Pending Queue
  - Address Scheduler
  - Memory Coalescer
- Coalescing Unit
  - **#lanes** addresses
  - cacheline coalescing
  - Multi-ported output



8

Evaluation

- Evaluation on Stratix 10 FPGA
  - V1: 16 cores (4x4 threads)
    31% LUTs, 200 MHz
  - V2: 1 core (32x32 threads)
    35% LUTs, 185 MHz
- Synthesis challenge
  - Routing pressure
- Performance scaling
  - Average speed up of 29%
  - LSU bottleneck



Summary

- Scaling single-core GPU performance required
  - Re-architecting GPU frontend and backend
    - Scoreboard, Register File, compute pipeline
  - Extending per-core memory bandwidth
  - Addressing synthesis challenges
- Best synthesis @ 16x16 threads

- Website: https://vortex.cc.gatech.edu
- Website: https://github.com/vortexgpgpu/vortex

Thank you!