Quantitative Evaluation Framework of Machine Learning Processors (MLP) for High Energy Physics

August 9 and 10 - 11AM

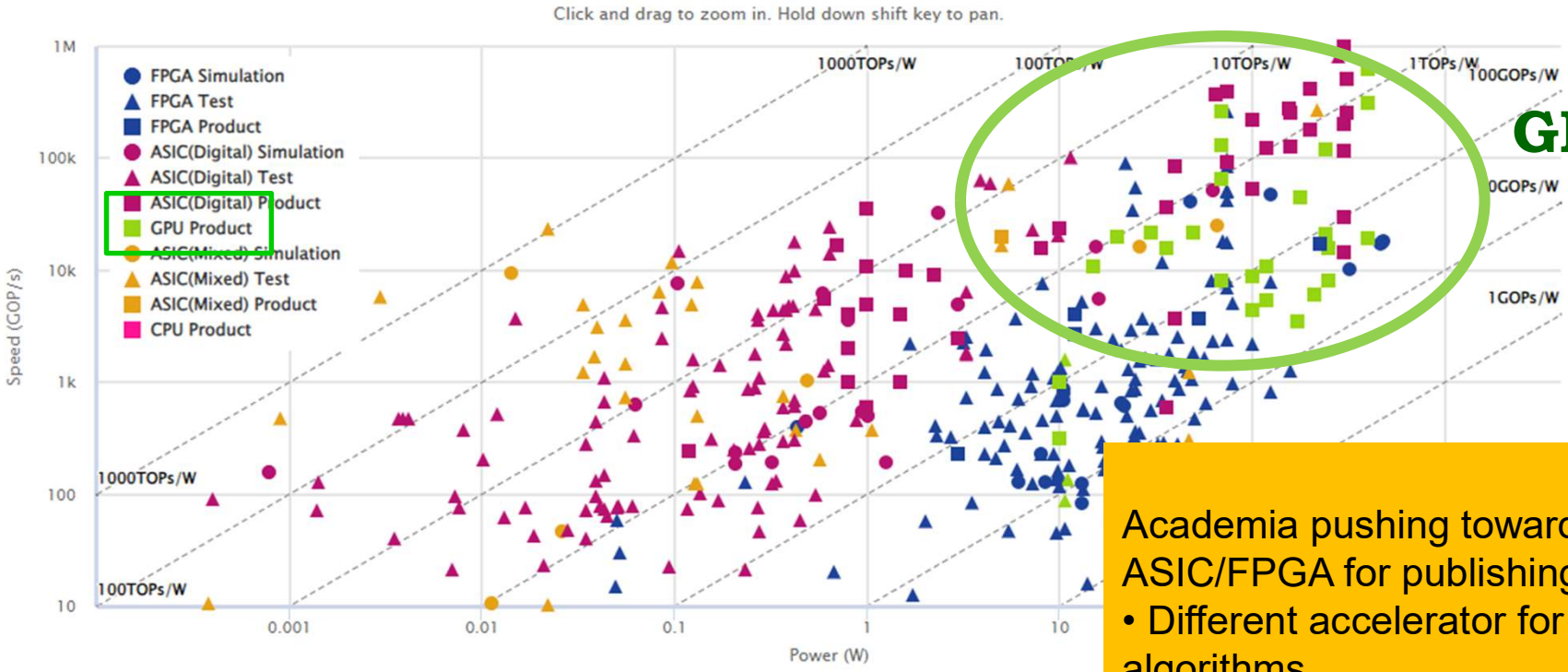
Youngsoo Kim, Bradley University

ykim@bradley.edu

Agenda

- Visiting Faculty Program Research Topics
  - **Quantitative Evaluation Framework of Graph Neural Networks Using Open Source Platforms**
  - Data type reduction/quantization for AI acceleration
  - eFPGA
- Background and motivation - Trends for Neural Network Acceleration
- The Vortex Platform
- HW-aware Optimal Mapping
- Validation
- Conclusions and Future Work

Trends for Neural Network Acceleration



GPU

Academia pushing towards ASIC/FPGA for publishing

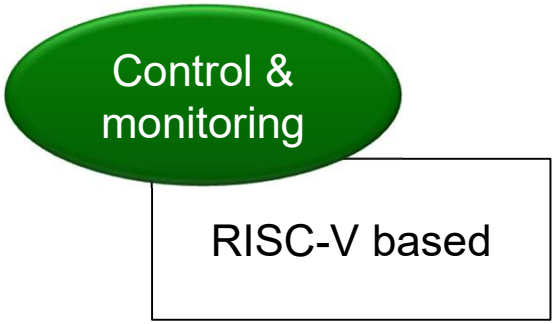
- Different accelerator for different algorithms

Question:

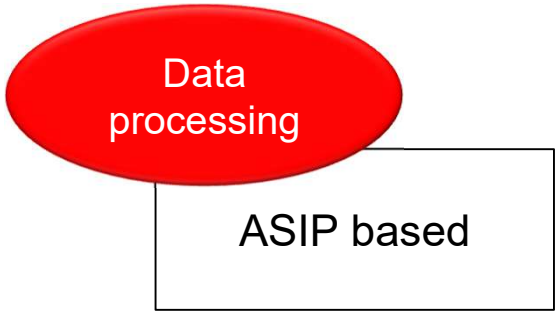
Is this **future proof, customizable** ?

[Ref: https://nicsefc.ee.tsinghua.edu.cn/network.html

Two complementary approaches: Microcontroller, ASIP



- Standardized solution
- Standard open-source RISC-V ISA
- Fully radiation tolerant
- SoC Ecosystem and IP blocks for the community
  - RadTol SoC IP clocks, interconnect



- Application Specific Instruction set Processor
- Tool base - ASIP Designer by Synopsis
- Custom optimized ISA and microarchitecture

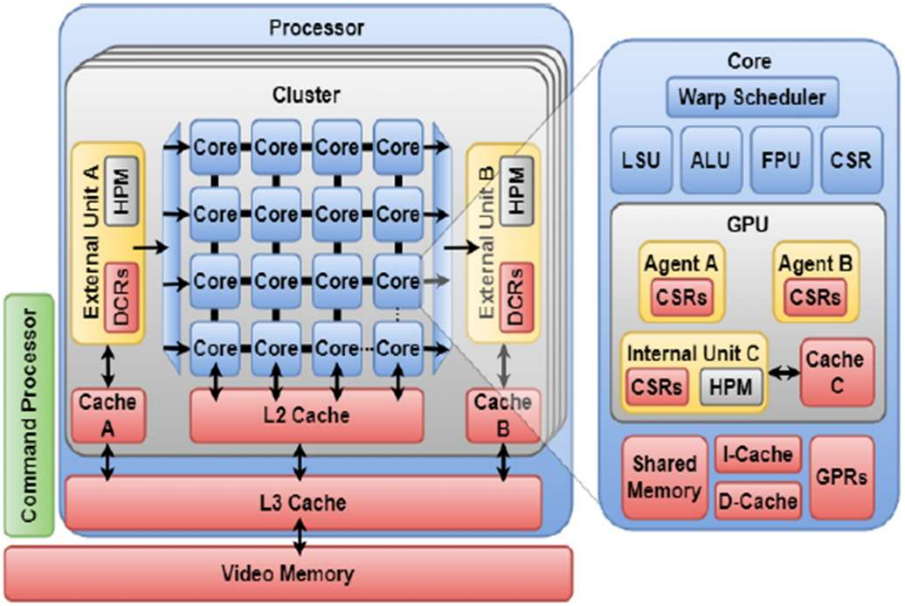
Open Source GPU

- GPU/ASIPs are alternative to application specific designs (ASIC or FPGA) in ML acceleration context
  - But need further specialization to improve efficiency
  - GPUs are complex architectures
  - Most GPU solutions are commercial, based on proprietary design, ISA, and SW stack

- But open-source, academic alternative was proposed in ASPLOS 2022
  - Vortex GPGPU from Georgia Tech.

Vortex GPU

- The Vortex GPU High-level architecture



RTL level implementation of GPU
- Based on the RISC-V ISA
- Scalable in cores, warps and threads
- Uses open-source software stack
- Support for OpenCL

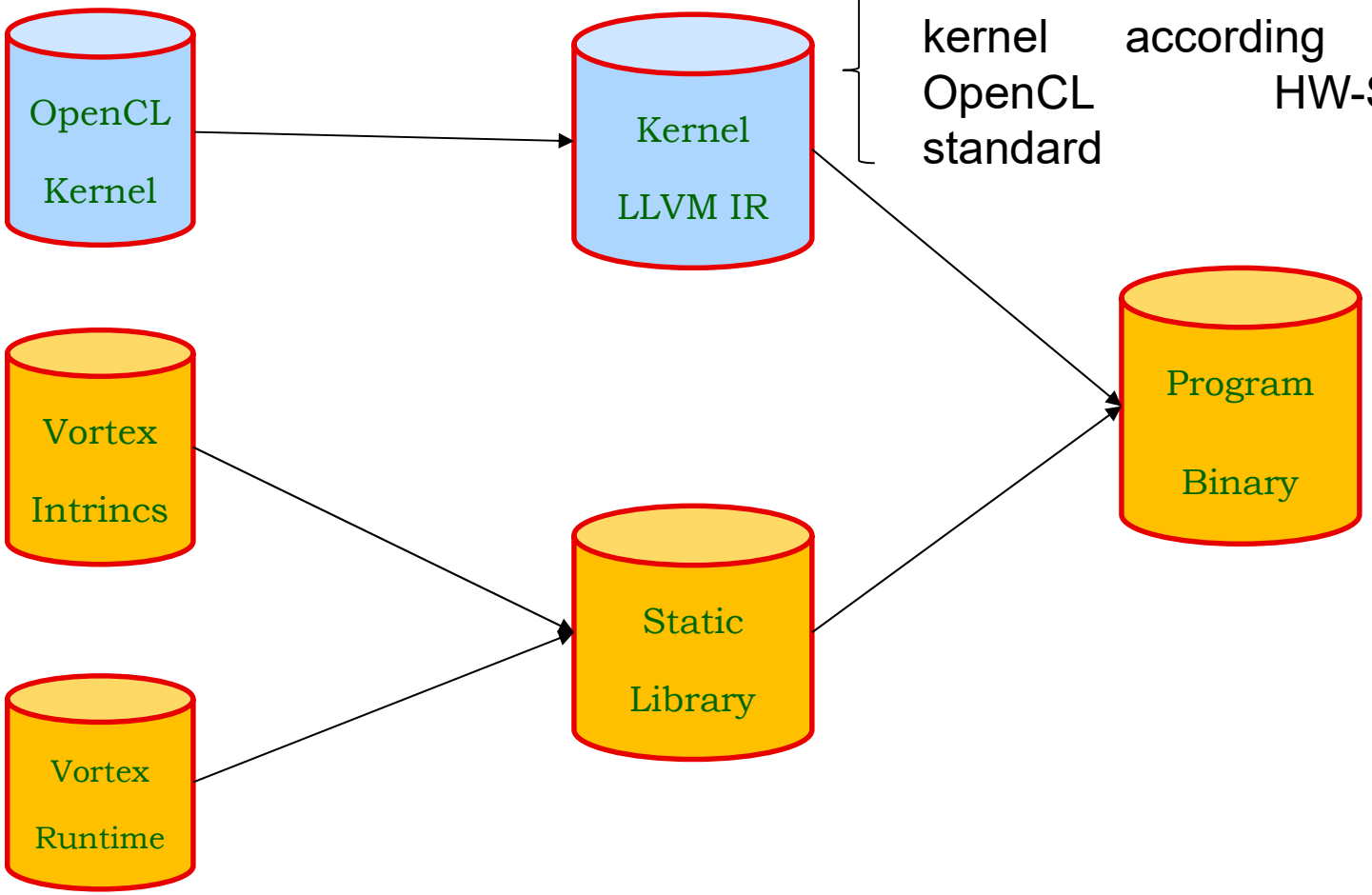
Why Vortex?
- Closed HW-SW loop + HW validation
- Open-source SW stack and customizable and extendable

RISC-V ISA extension for GPU control
wspawn – wavefront generation
split/join – control flow divergence/reconvergence
bar – wavefront barrier

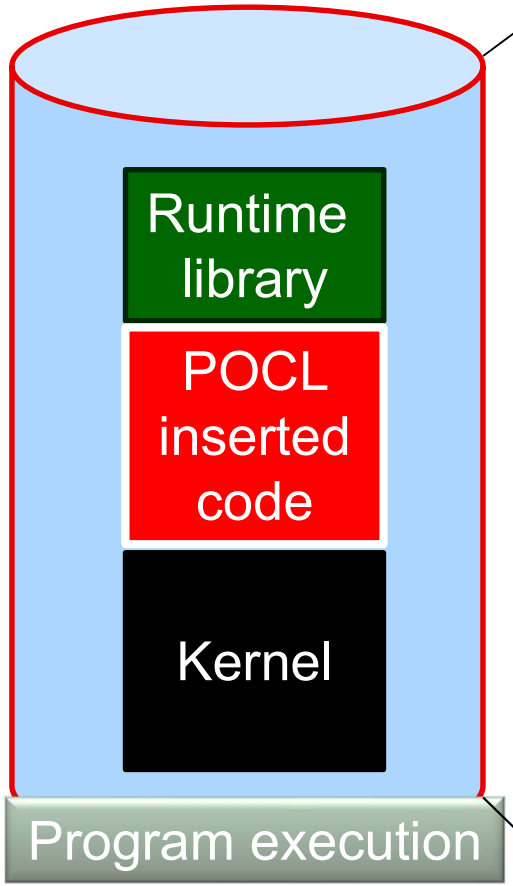
[Ref: B. Tine, et al, "Implementing Hardware Extensions for Multicore RISC-V GPUs", CARRV (2022)]

OpenCL Compilation Flow

Inputs to compiler: Hand-code vecadd, sgemm, gcn, etc.



Workload distribution on Vortex



1) Spatial workloads distribution – workload is distributed e.g. vector add C = A + B for 16 elements, each thread will add two elements

Example: 1 core, 2 warps, 4 threads

2) Temporal unrolling – Determines the elements executed by every spawned warp This depends on a runtime parameter passed by the host global kernel call (e.g. local work size)

Example – Execution changing local work load sizes for vecadd 16 elements

lws	wspawn	tmask
1	4	1111
2	4	0011
4	2	0001
8	2	0001

Under-utilization!

Problem

- The lws parameter impacts the execution
- "Wrong" values lead to suboptimal HW utilization (slower exec, more instr. issues)
- How to determine optimal lws dynamically for different kernels?

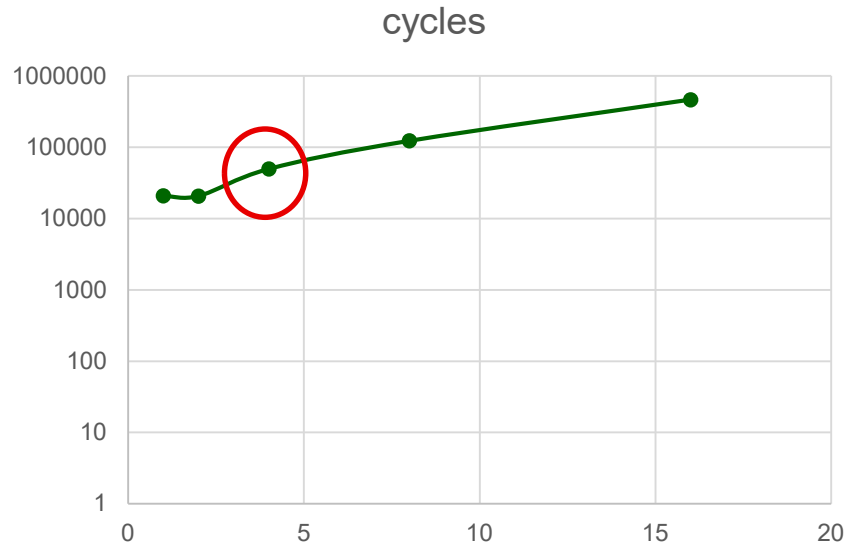
Optimal HW-aware Mapping

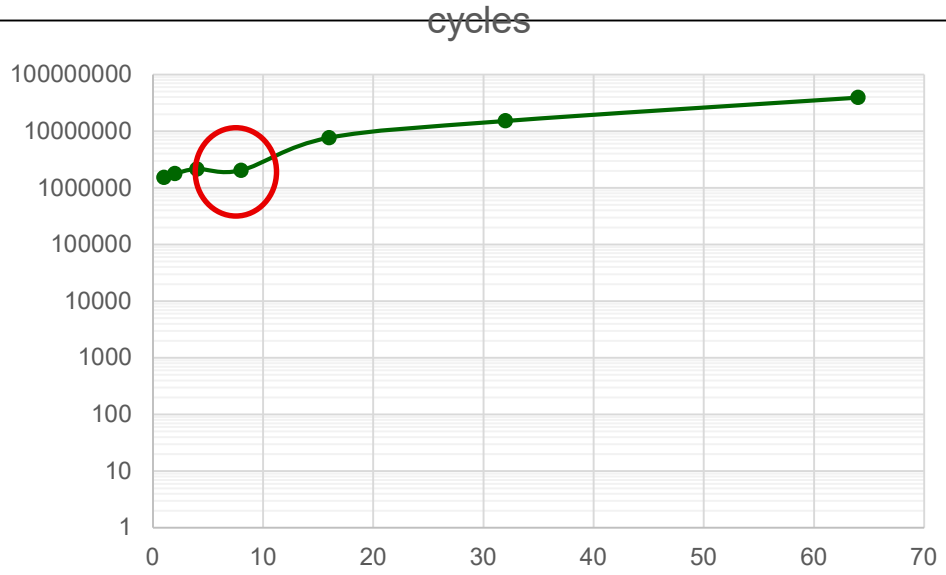
- This problem can be formulated as an ILP problem. The value of cost function can be expressed by the following formula:
- minimize $\sum_i \ aws \ - weights * (cores * warp * threads)$
- $lws\_opt = \frac{aws}{cores * warp * threads}$

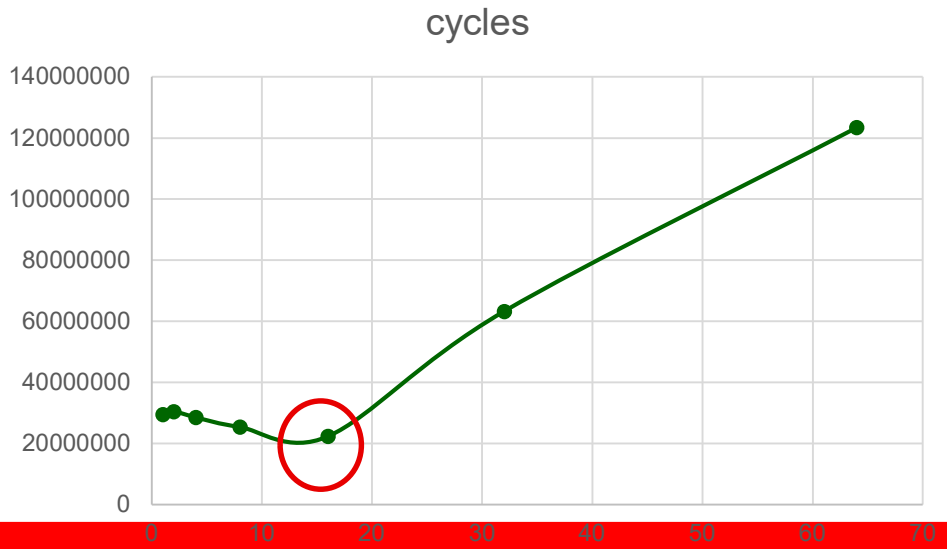
SW

HW

Execution latency of sgemm across different local work load sizes





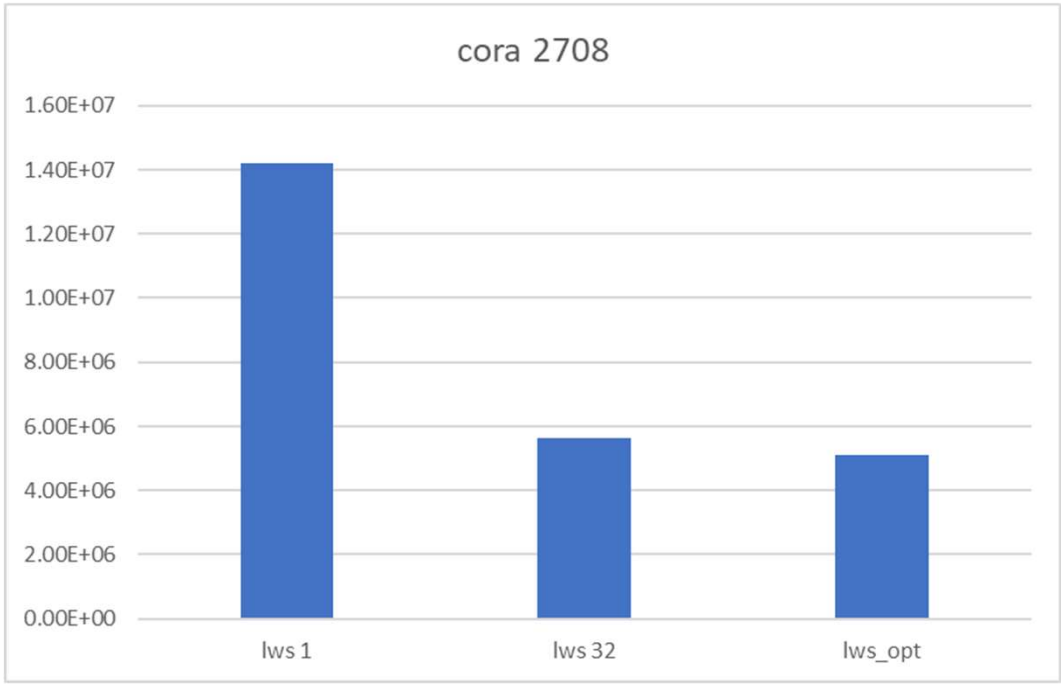


Our solutions

hw conf:
2 cores, 4 warps, 32 threads
aws: 32, 128, 256 …

Validation methodology

- Generate/develop 8-32 different Vortex architecture configurations
- Evaluate our mapping on GNN/GCN layers
  - hidden feature size
  - single aggregation, or full layer
- Compare execution time and latency results with different mapping
  - Energy Delay Product (EDP) if possible



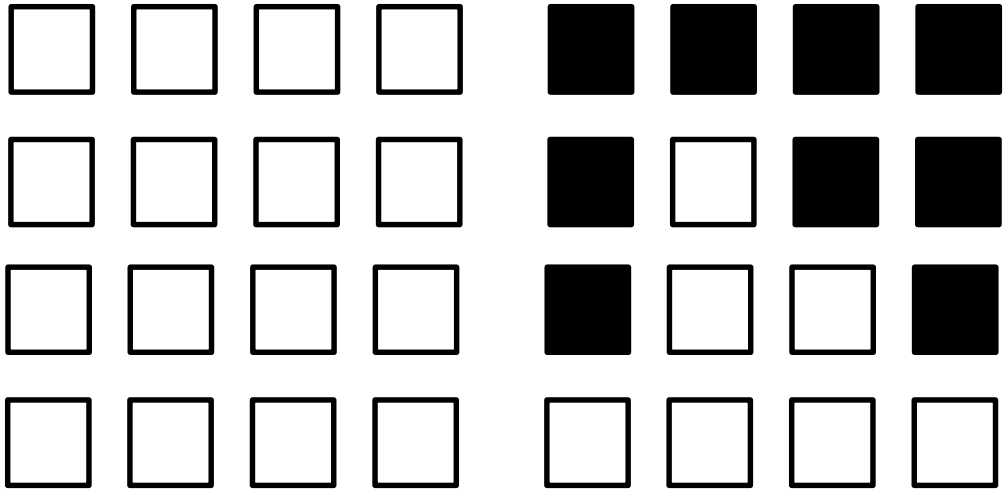
Execution latency for GCN with different mappings

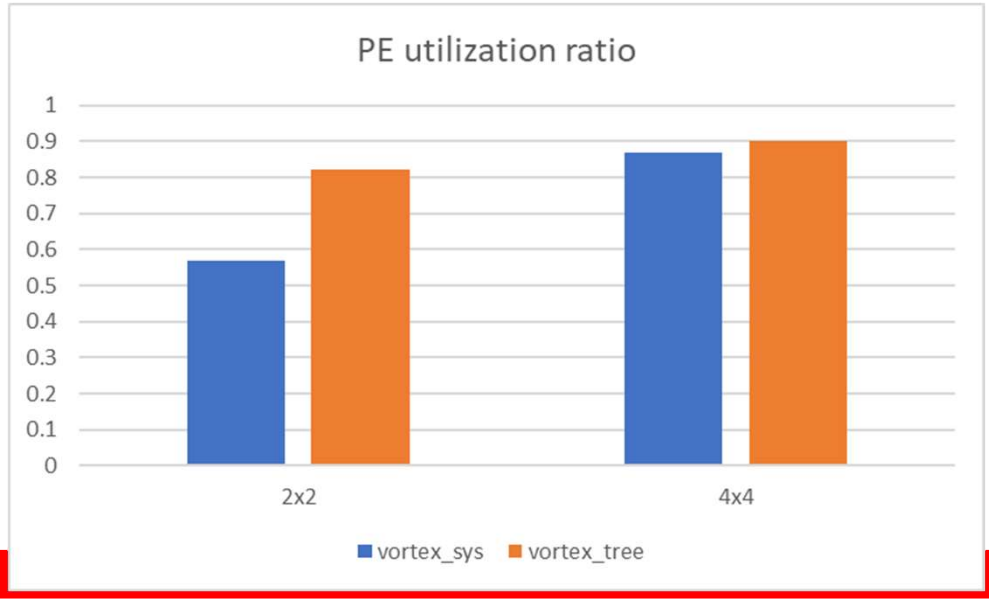
Validation methodology

- GCN/GNN benchmarks from an MIT benchmarking tool in C++. They implement kernel calls in plain C++, vector extensions and CUDA

- My contribution mainly ported the calls in **OpenCL**

- The repo does not include formatted datasets (for space reasons), I had to generate them





Conclusions

- ML acceleration research optimizes architectures for specific models
- A different approach is bridging the gap between open-source GPUs and ASIC, and ASIP

- Investigated limitations of the Vortex GPGPU platform
- Proposed an optimal, HW-aware mapping (dynamic at runtime) that ensures an efficient execution, minimizing cycle latency
- Validated on several configurations of GCN layers

Future Work

- There is no characterizing effort of GCN/GNNs on GPUs
  - Write a workloads characterization paper
- Extend this work to the ESP platform with risc-v and set up as a loosely coupled accelerator
  - risc-v + custom ALUs on ESP for an ASIC/RTL implementation

- PI will develop a funding proposal for high performance and radiation hardened configurable fault tolerant processor platforms by mobile customized FPGA platforms
  - Naval Sea Systems Command (NSWC) Crane Division