Understanding Performance Gains of Accelerator-Rich Architectures

Zhenman Fang

Assistant Professor

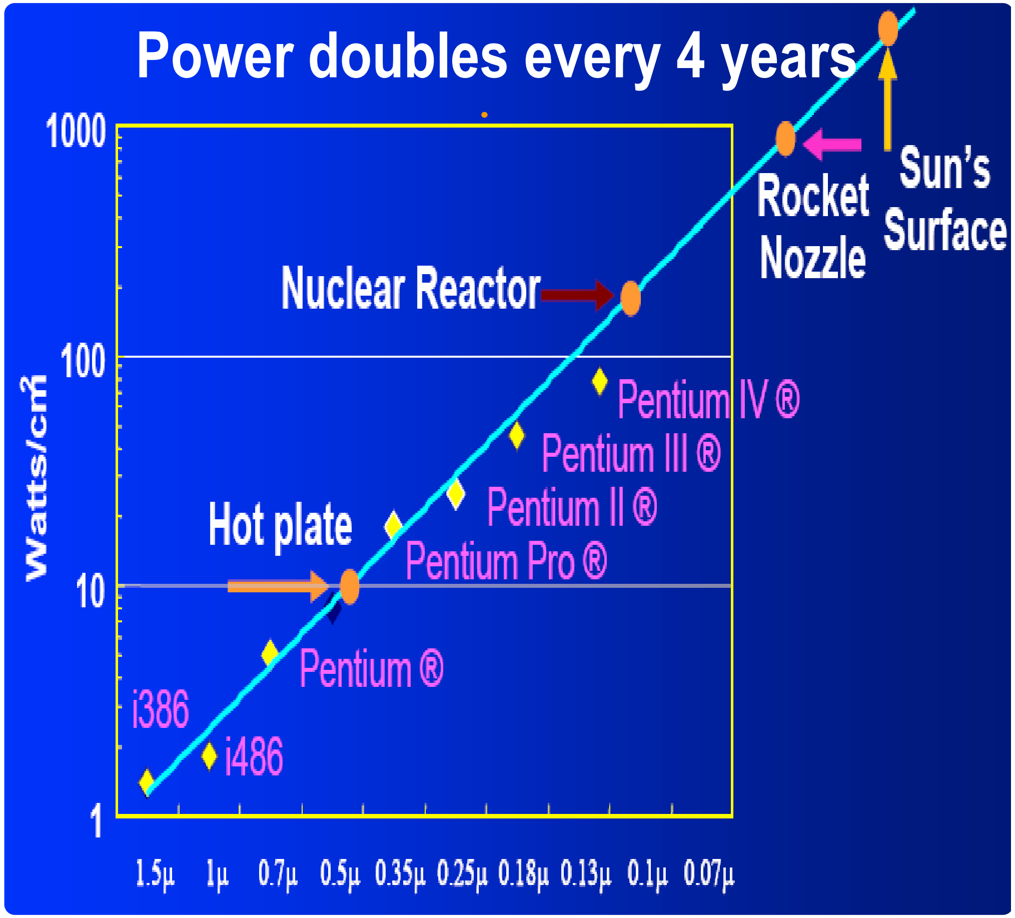
Computer Engineering, SFU

Email: zhenman@sfu.ca

http://www.sfu.ca/~zhenman

The Power Wall and Customized Accelerators

The famous power wall !



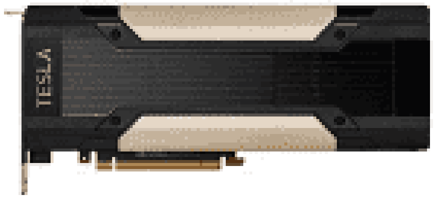
Source: Shekhar Borkar, Intel

Customized accelerators !

ASIC

e.g., Google TPU v3

GPU

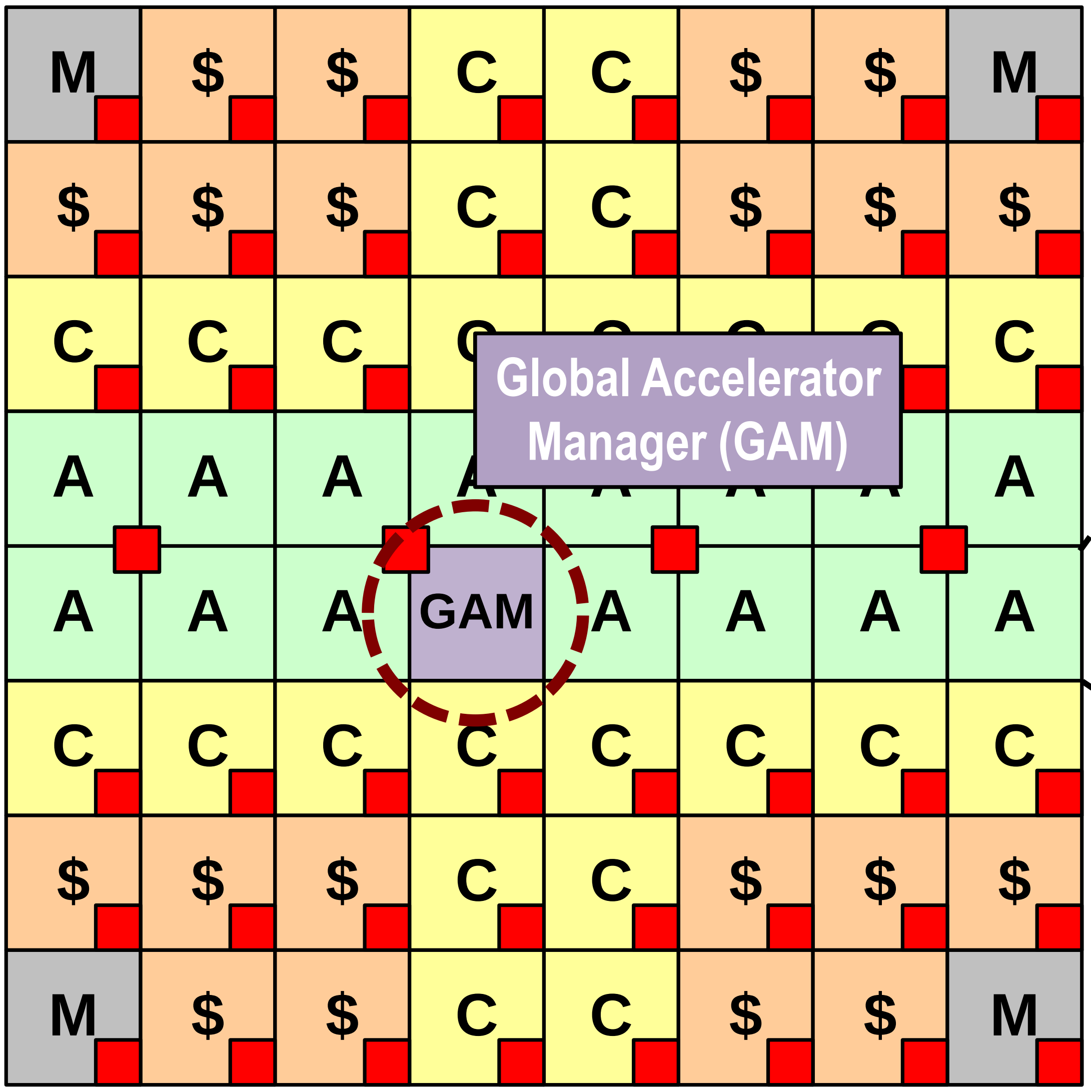


e.g., Nvidia Tesla GPUs

FPGA

e.g., Xilinx Alveo FPGAs

Trend of Accelerator-Rich Architectures (ARA)

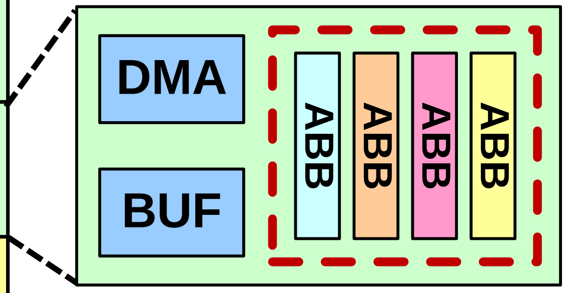


M	$	$	C	C	$	$	M
$	$	$	C	C	$	$	$
C	C	C	C	C	C	C	C
A	A	A	A	A	A	A	A
A	A	A	GAM	A	A	A	A
C	C	C	C	C	C	C	C
$	$	$	C	C	$	$	$
M	$	$	C	C	$	$	M

M Memory Controller

$ Cache Banks

C Core



Dedicated/Composable Accelerators

ABB: Accelerator Building Block

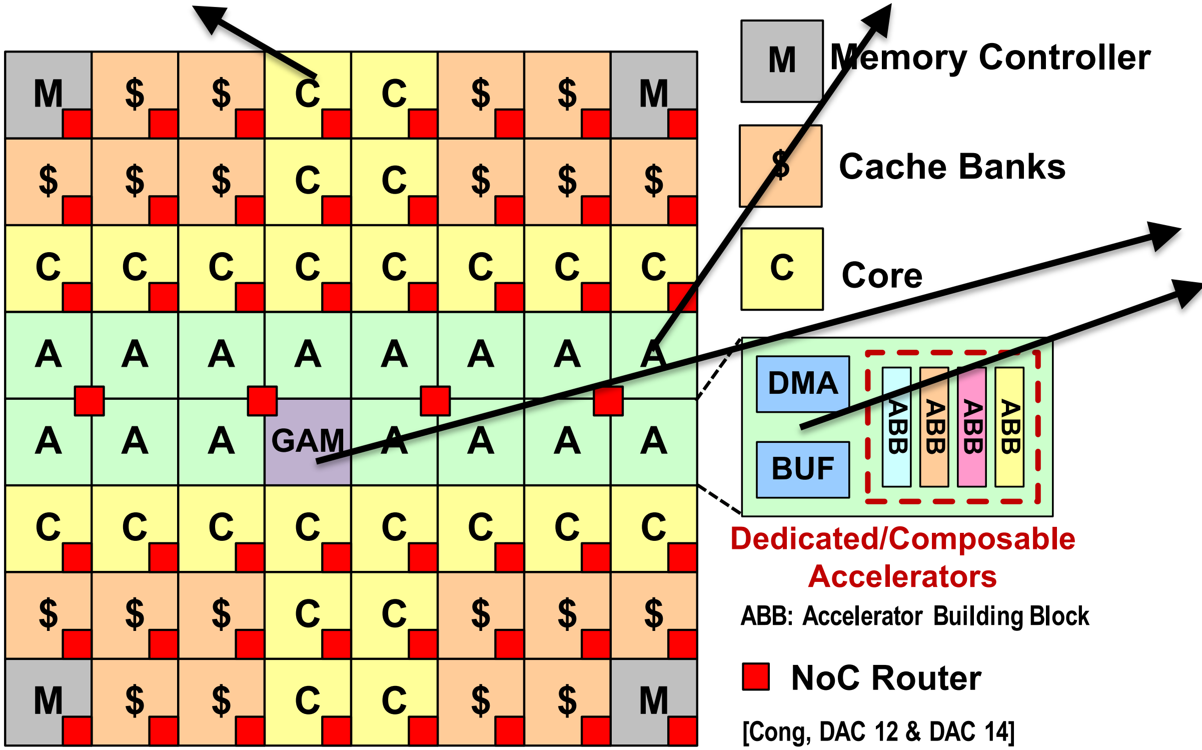
🟥 NoC Router

[UCLA, DAC 12 & DAC 14]

PARADE: Platform for ARA Design & Exploration

Extended gem5 for X86 CPU & the rest of system

Auto-generated accelerators based on HLS (C->RTL->timing)



M	$	$	C	C	$	$	M
$	$	$	C	C	$	$	$
C	C	C	C	C	C	C	C
A	A	A	A	A	A	A	A
A	A	A	GAM	A	A	A	A
C	C	C	C	C	C	C	C
$	$	$	C	C	$	$	$
M	$	$	C	C	$	$	M

[Cong, DAC 12 & DAC 14]

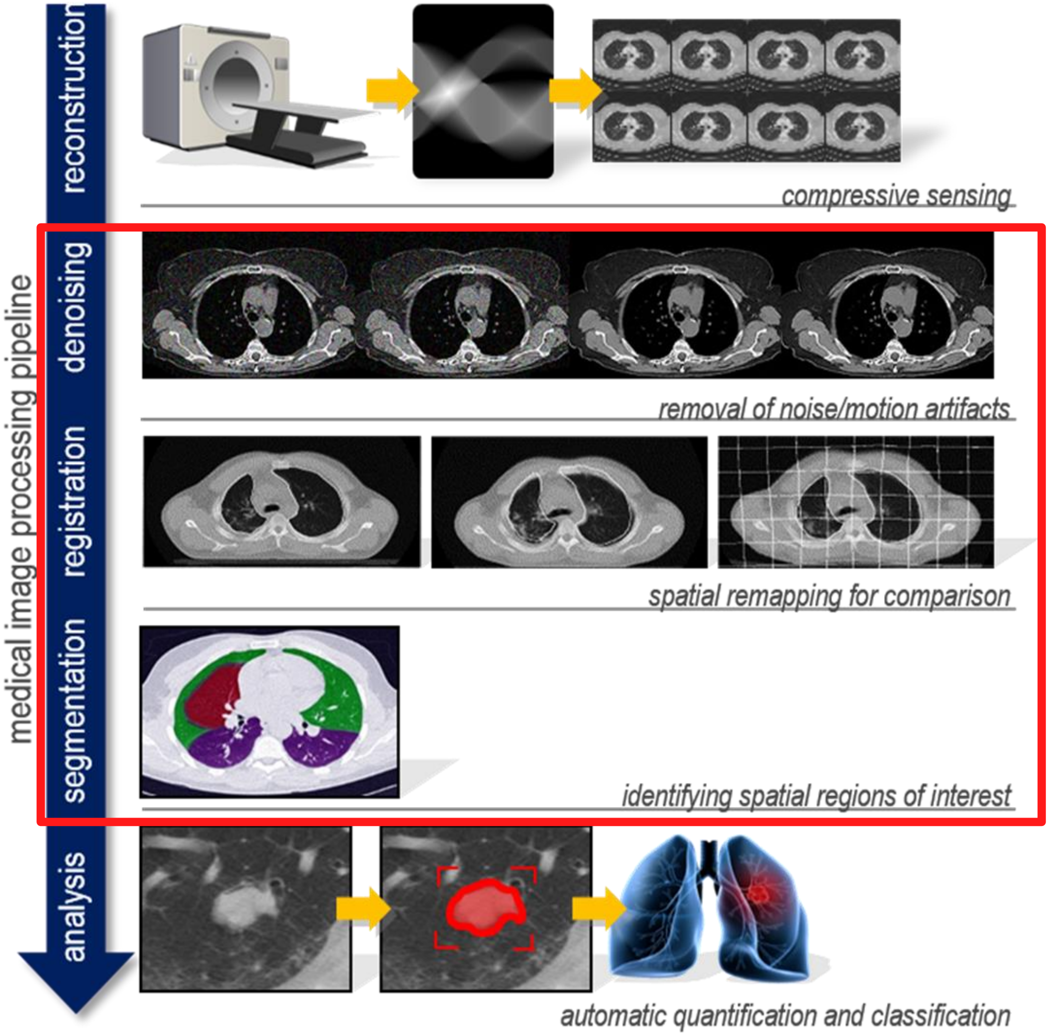
Added BUF, DMA, GAM & TLB model

Paper at [ICCAD'15], Tutorials at [ISCA'15 & MICRO'16]

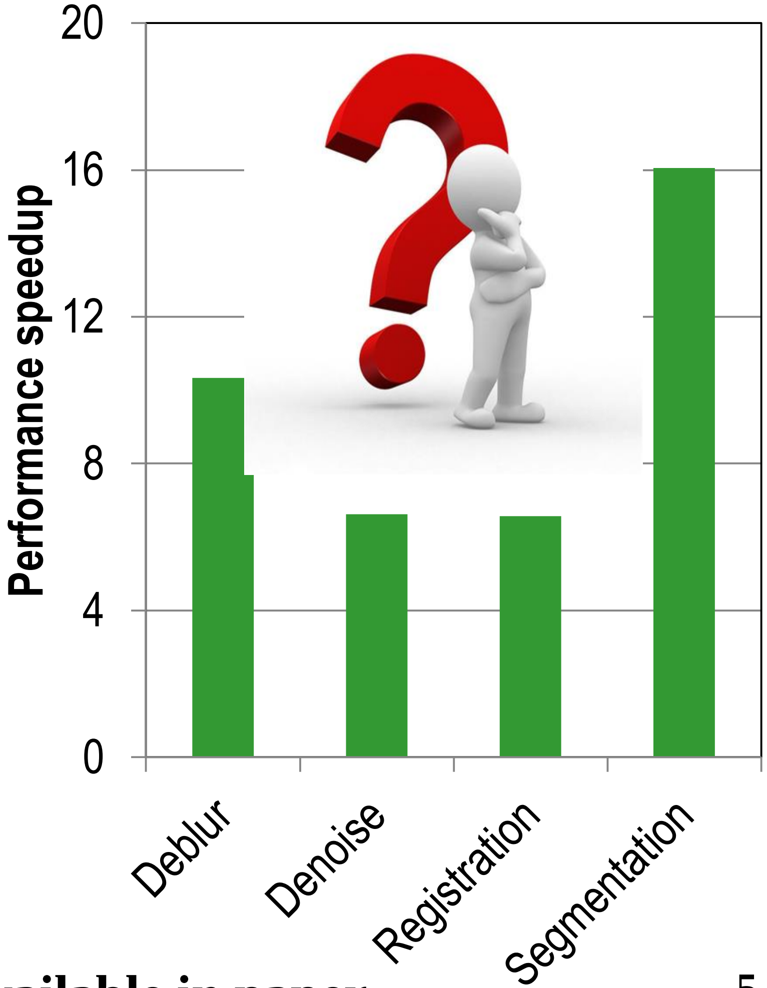
PARADE open source link: http://vast.cs.ucla.edu/software/parade-ara-simulator

Example Acceleration Results.. and Insights?

Low-dose CT screening for lung cancer


medical image processing pipeline
reconstruction
compressive sensing
denoising
removal of noise/motion artifacts
registration
spatial remapping for comparison
segmentation
identifying spatial regions of interest
analysis
automatic quantification and classification

One accelerator processing element (PE) vs. one X86 core

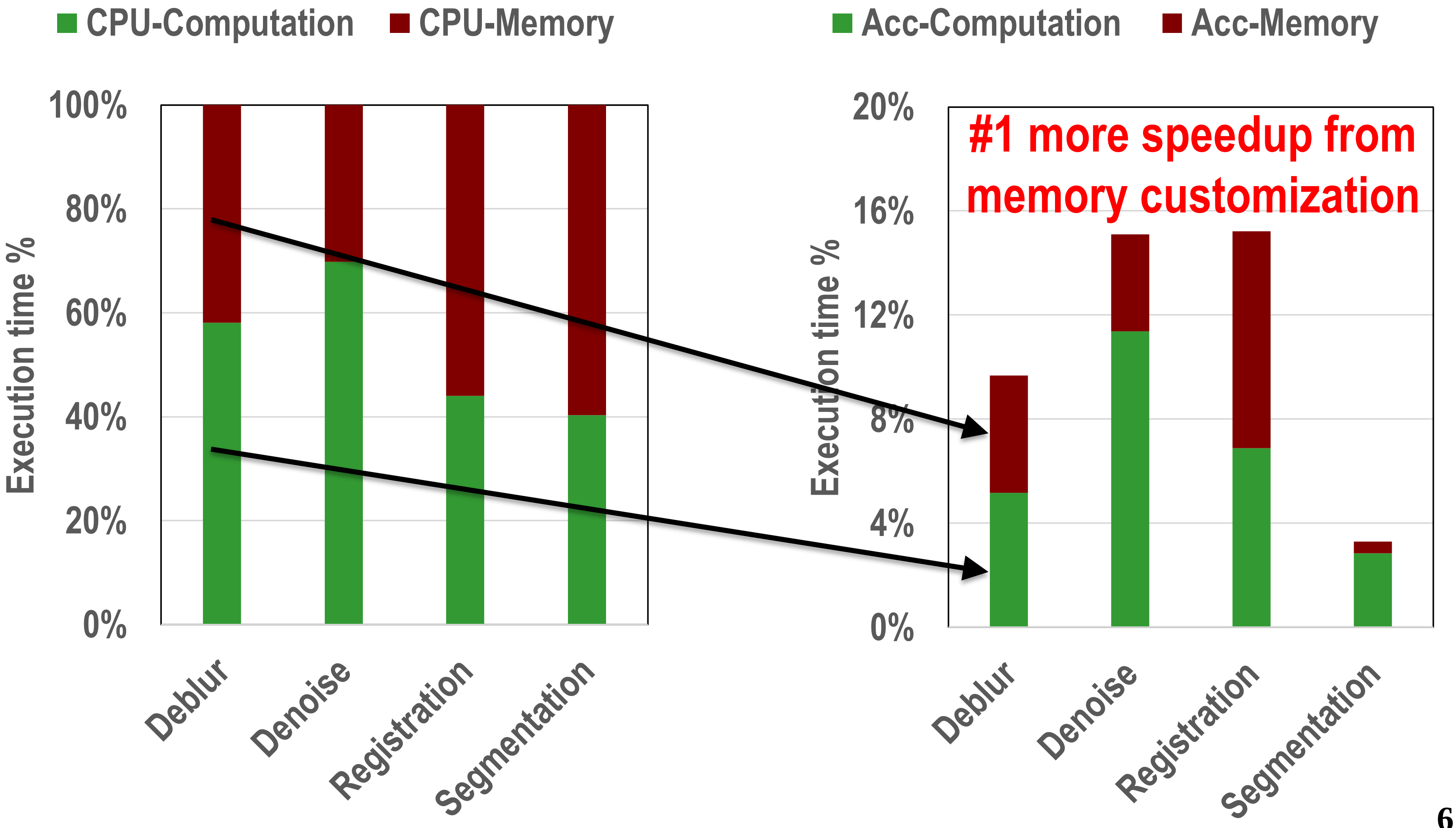


More benchmarks and results available in paper.

Gains from Both Computation and Memory

CPU performance [optimized]

Accelerator performance



#1 more speedup from memory customization

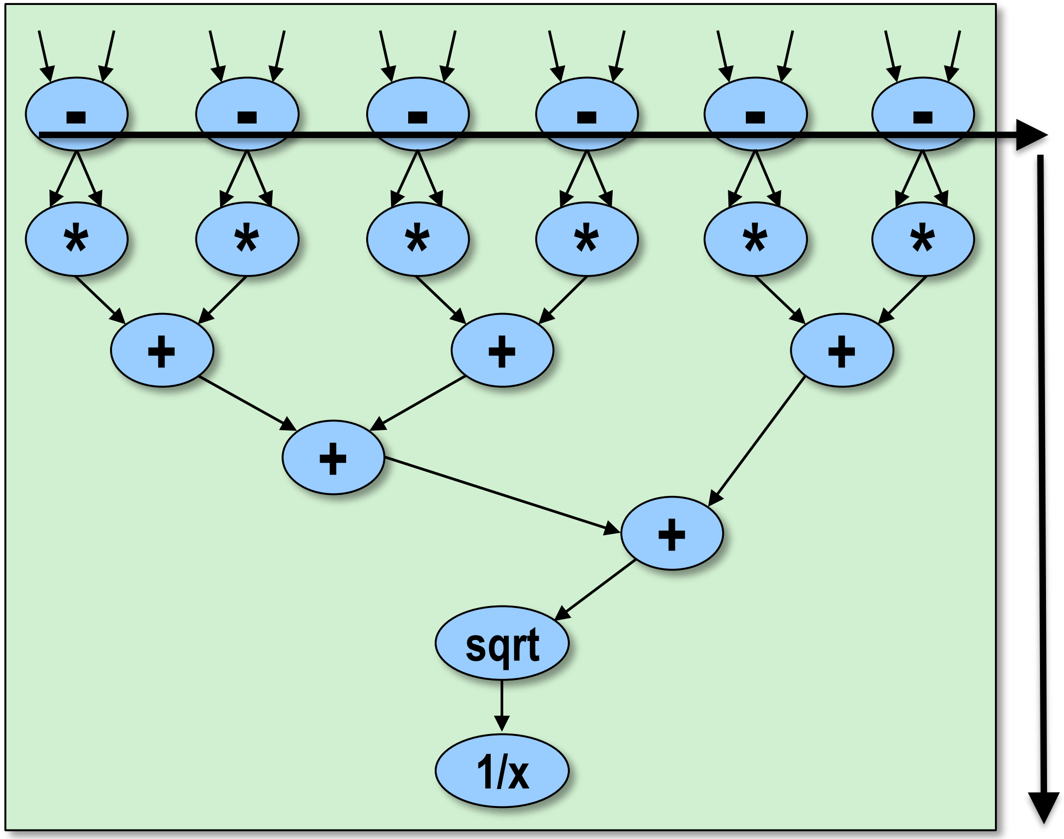
Gains from Computation Customization

6.1x speedup

⇧

#2 customized accelerator pipeline

Denoise core computation: $$1/\sqrt{\sum_{i=0}^{5}(x_c - x_i)^2}$$



a) fine-grained parallelism:
more flexible than SIMD

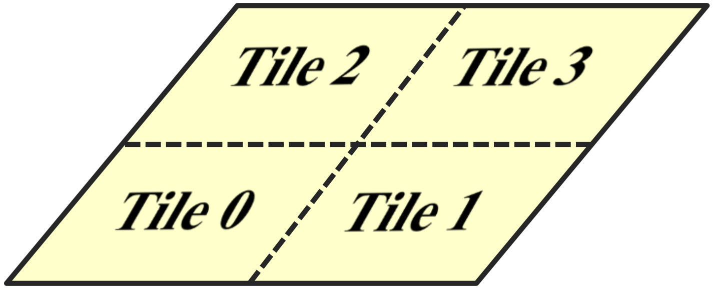
b) customized pipeline:
no instruction overhead

~~load Xc~~
~~load Xi~~
sub Xc – Xi
~~store result~~

CPU execution
↓
Acc execution

c) coarse-grained parallelism:
by duplicate this pipeline

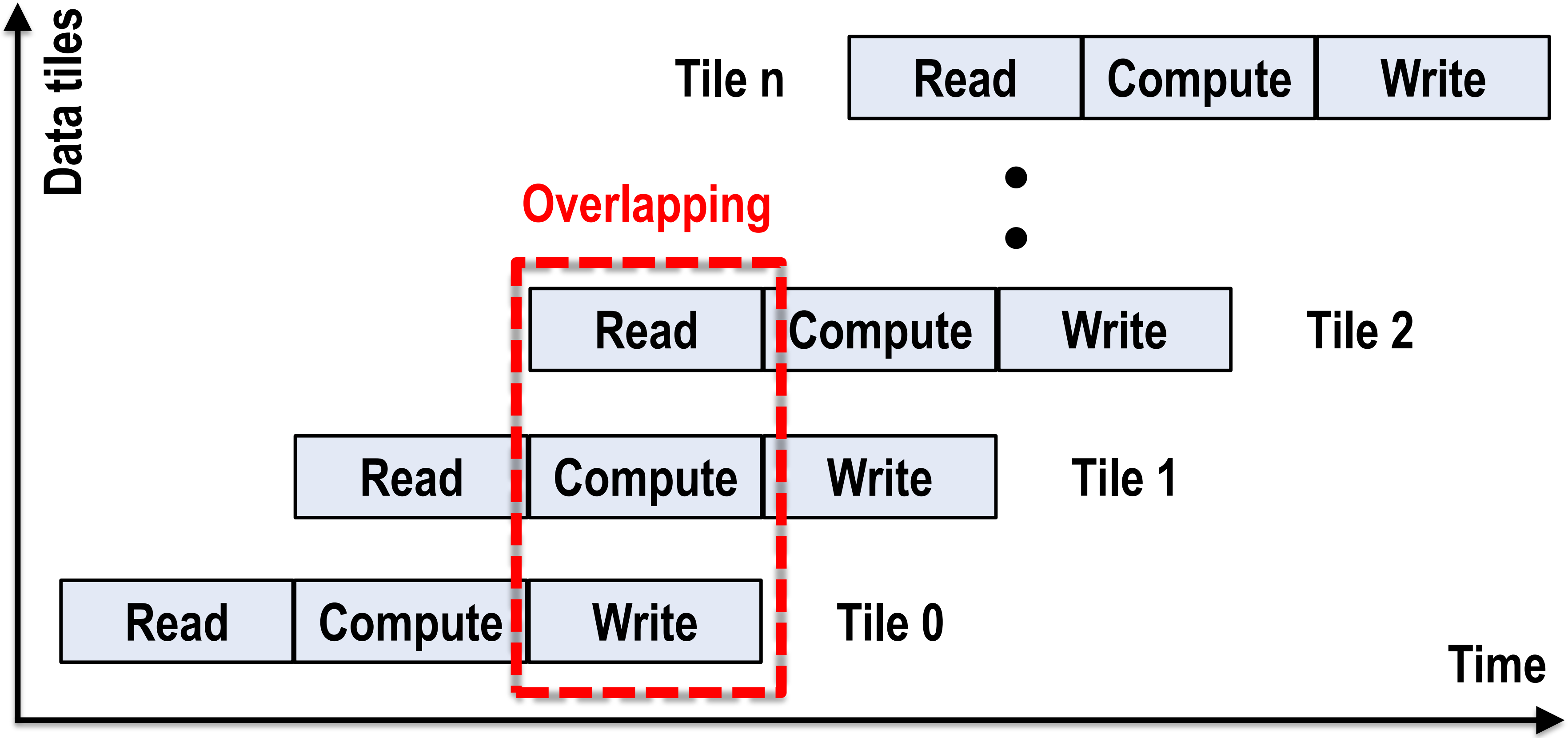
Memory Customization


Tile 2
Tile 3
Tile 0
Tile 1

#3 Memory access reduction

#4 Memory-level parallelism improvement

Aggregate all data accesses in a tile to a short read/write period to overlap access latency


Data tiles

Tile n: Read | Compute | Write

Overlapping

Tile 2: Read | Compute | Write

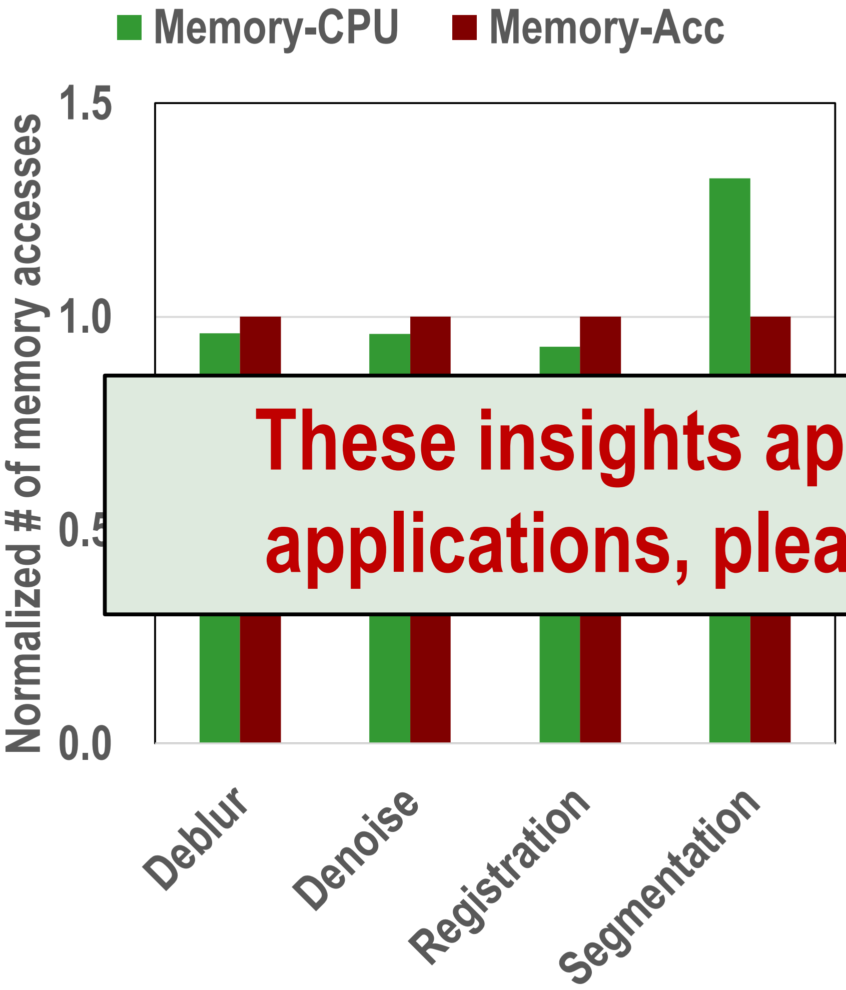
Tile 1: Read | Compute | Write

Tile 0: Read | Compute | Write

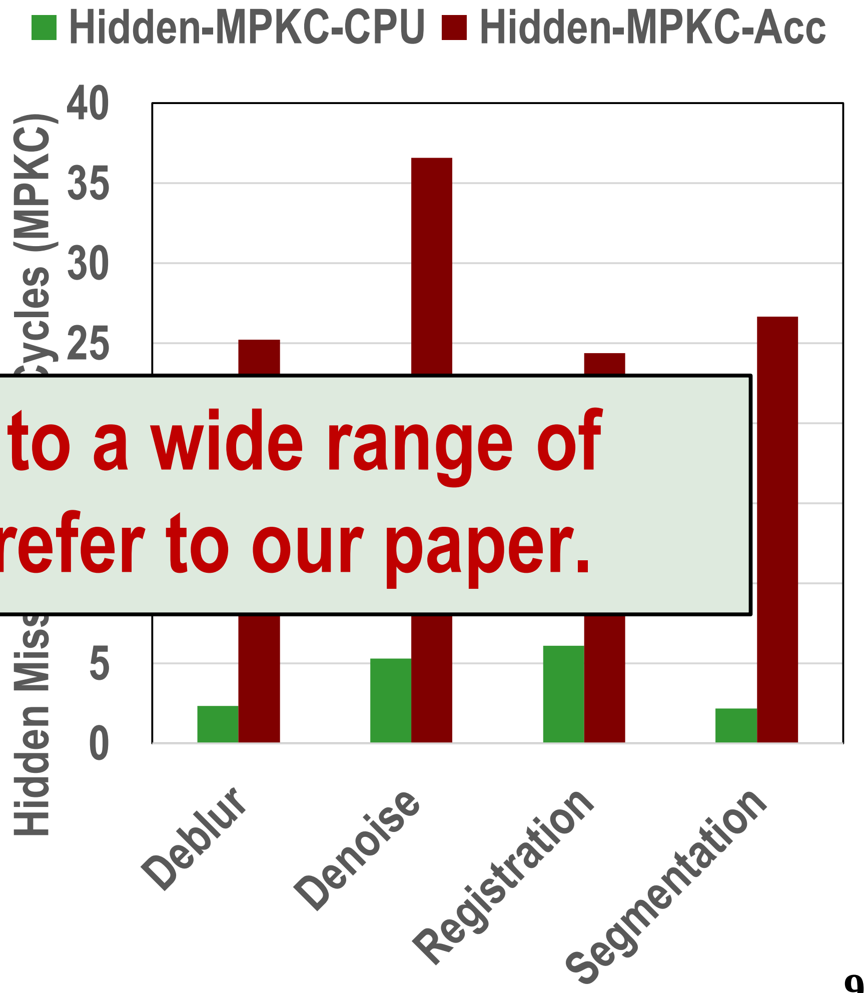
Time

Gains from Memory Customization

#3 Memory access reduction is not the key!

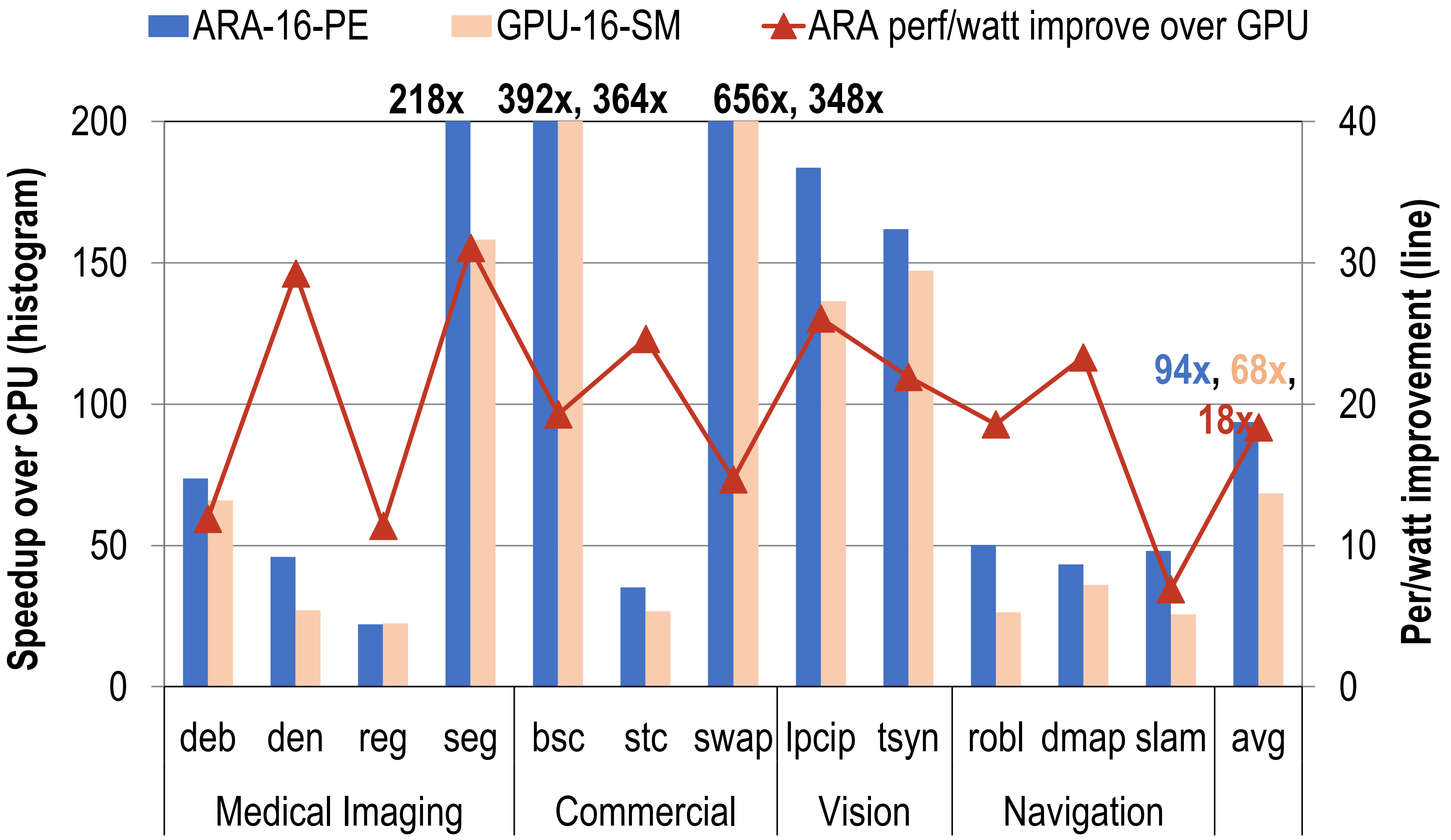


#4 Memory-level parallelism improvement is the key!



These insights apply to a wide range of applications, please refer to our paper.

ARA (Multi-PE) vs. GPU

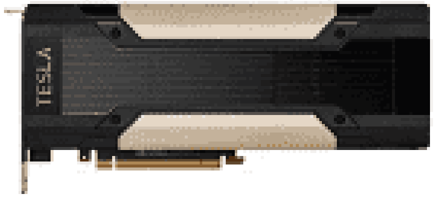


Programmable Accelerators: FPGA vs. GPU

Although their performance and energy advantages are clear, ASICs have high design cost and lack flexibility

Let's look at more programmable accelerators

GPU



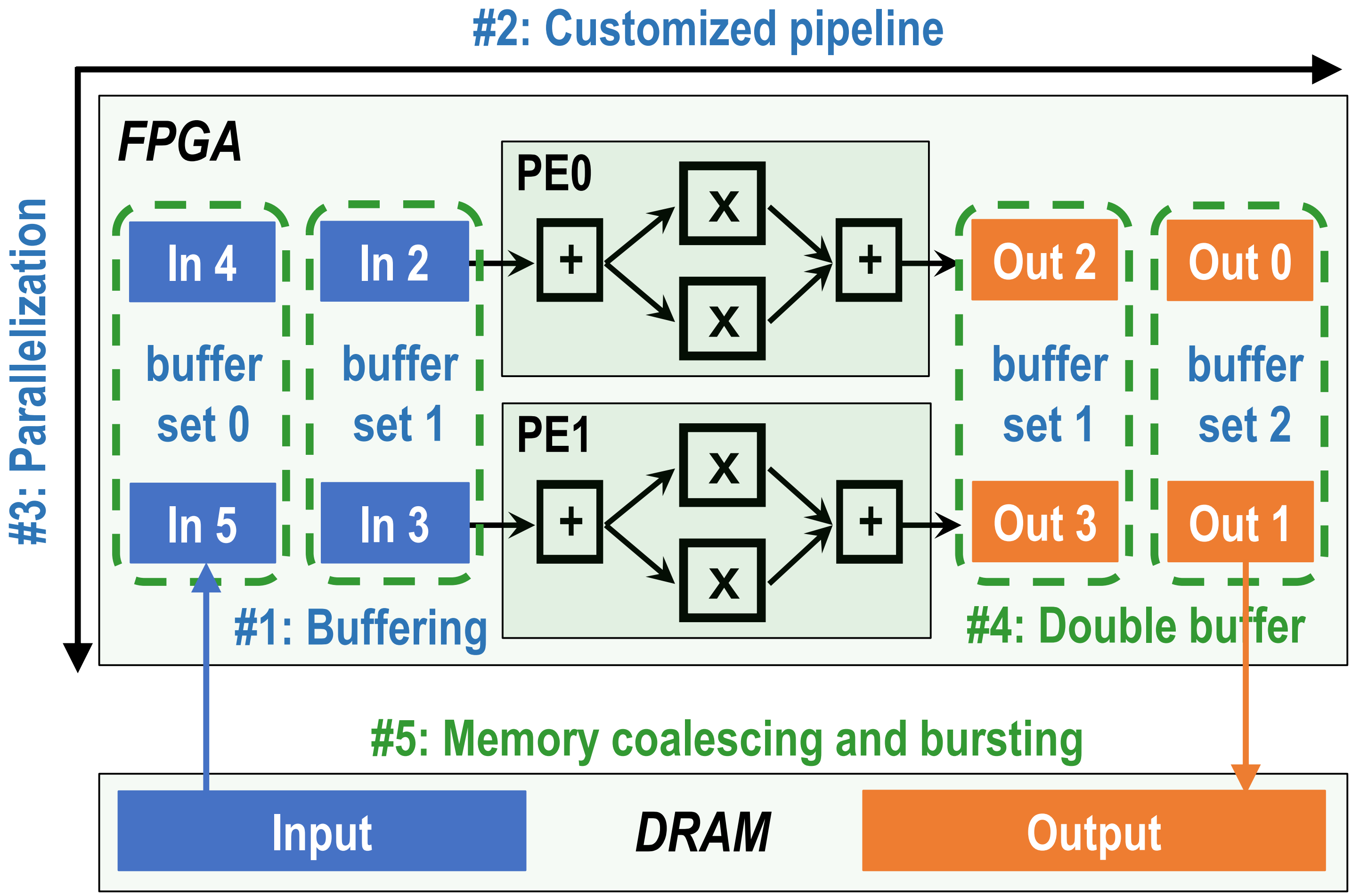
e.g., Nvidia Tesla GPUs

vs.

FPGA

e.g., Xilinx Alveo FPGAs

Applying the Insights into FPGA Accelerators



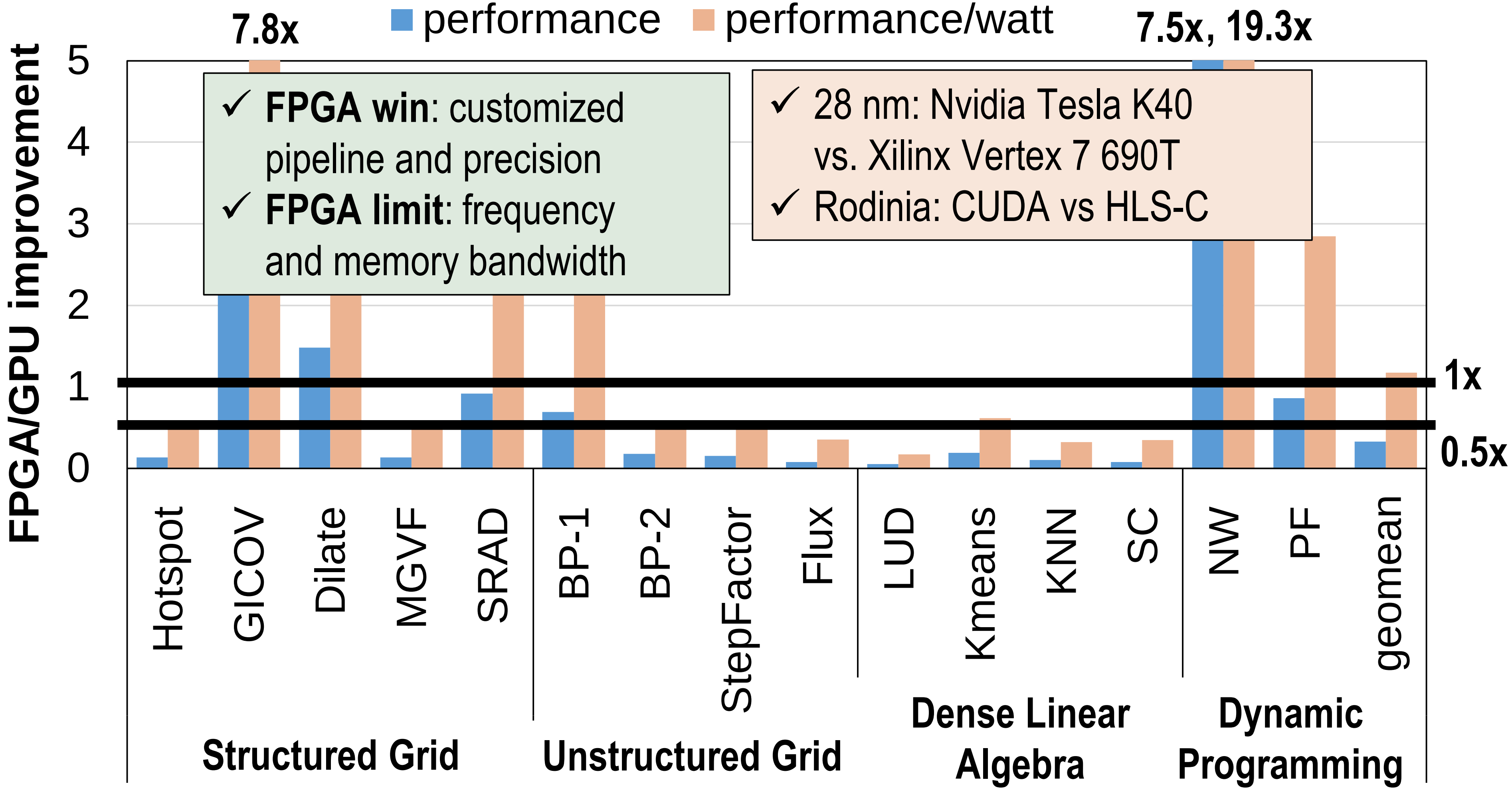
Kernel design in HLS C

CPU-FPGA communication in OpenCL

Xilinx SDAccel 2016.4 (updating)

For fair comparison, we port the widely recognized GPU benchmark suite Rodinia to FPGA using HLS C, and apply the prior insights during porting

Preliminary GPU-FPGA Comparison [FCCM 2018]



- ✓ **FPGA win**: customized pipeline and precision
- ✓ **FPGA limit**: frequency and memory bandwidth

- ✓ 28 nm: Nvidia Tesla K40 vs. Xilinx Vertex 7 690T
- ✓ Rodinia: CUDA vs HLS-C

Performance: out of 15 kernels, 3 FPGA kernels win, 3 kernels comparable

Performance/watt: 6 FPGA kernels win, 4 kernels is > 2x worse than GPU

Conclusion and Future Directions

The power wall has led to the trend of heterogeneous accelerator-rich architectures (ARAs)

The performance gains of ARAs come from

- Computation customization: 1) customized accelerator pipeline, and 2) coarse-grained parallelism
- Memory customization (often more important): 1) memory access reduction and 2) improved memory level parallelism (often the key)

Future directions

- Better understand when apps run better on FPGAs, when on GPUs
- Near data acceleration architectures and systems, with corresponding programming, compiler, and runtime support

Thank You!

More info at http://www.sfu.ca/~zhenman

Past Sponsors

CDSC Center for Domain-Specific Computing

CFAR

idre **Postdoc Fellowship**

Current Sponsors

SFU **Simon Fraser University**

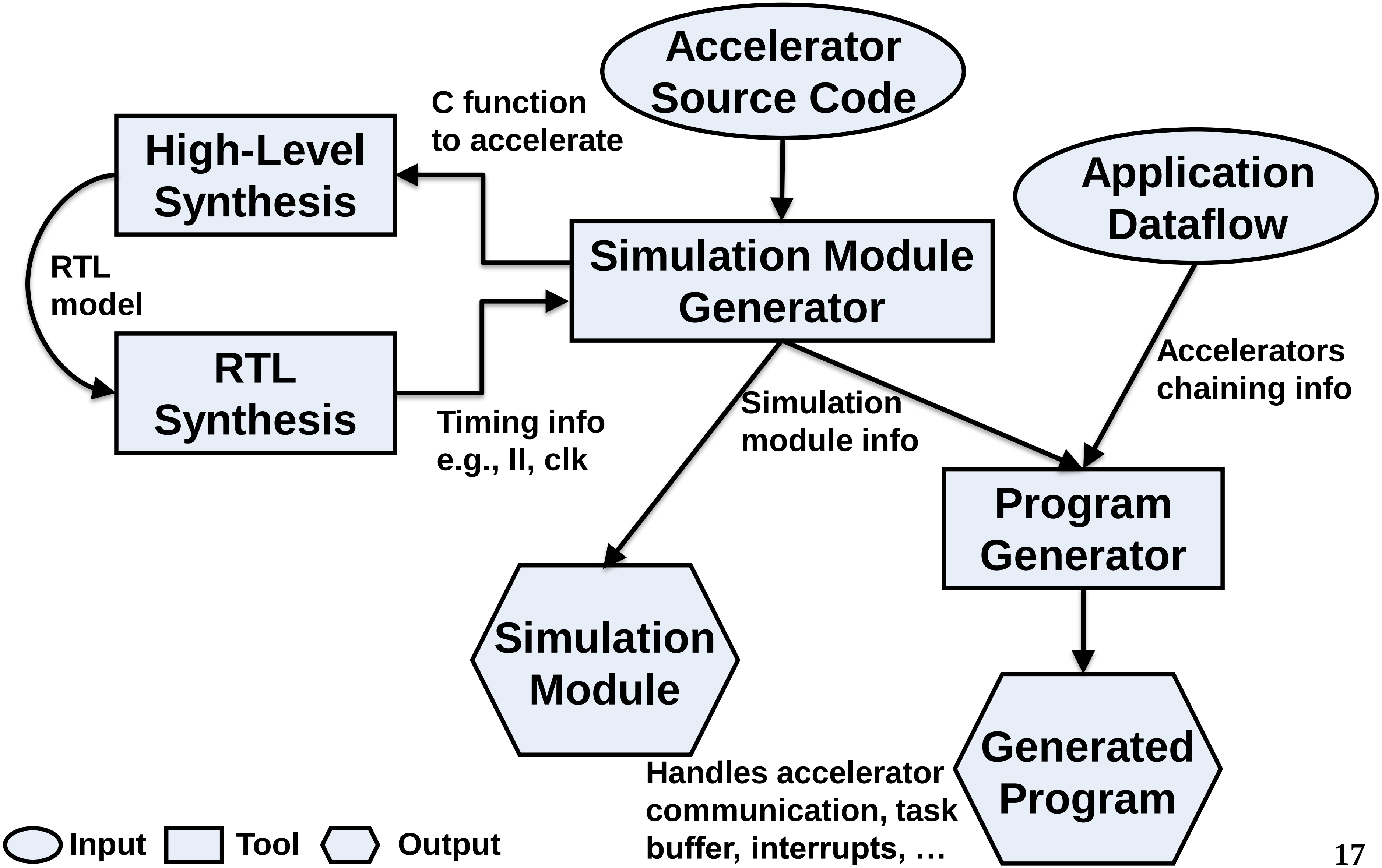
NSERC CRSNG

XILINX®

NVIDIA®

Backup Slides

HLS-based Automatic Accelerator/App Generation



Accelerator Source Code

Application Dataflow

High-Level Synthesis

C function to accelerate

RTL model

RTL Synthesis

Simulation Module Generator

Timing info e.g., II, clk

Simulation module info

Accelerators chaining info

Program Generator

Simulation Module

Generated Program

Handles accelerator communication, task buffer, interrupts, …

Input

Tool

Output

Customize Your Own Accelerator (e.g., Denoise)

Denoise core computation: $1/\sqrt{\sum_{i=0}^{5}(x_c - x_i)^2}$

Auto-generated accelerator for each ABB function

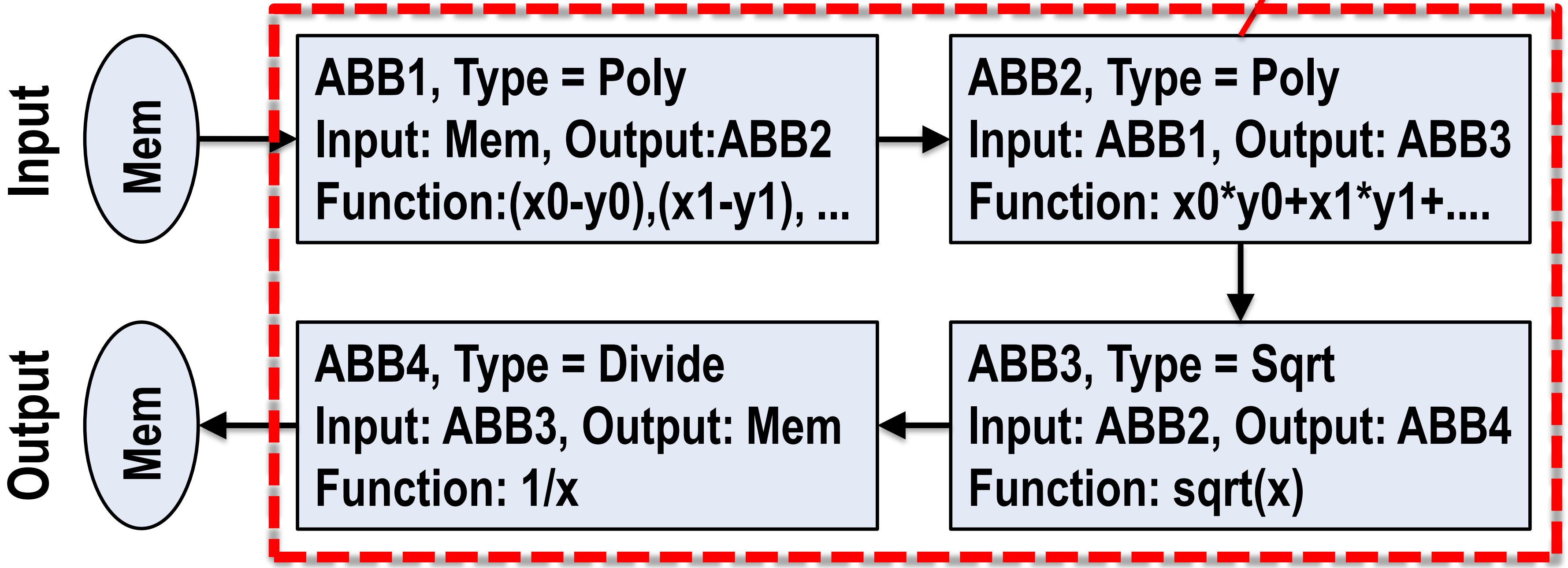
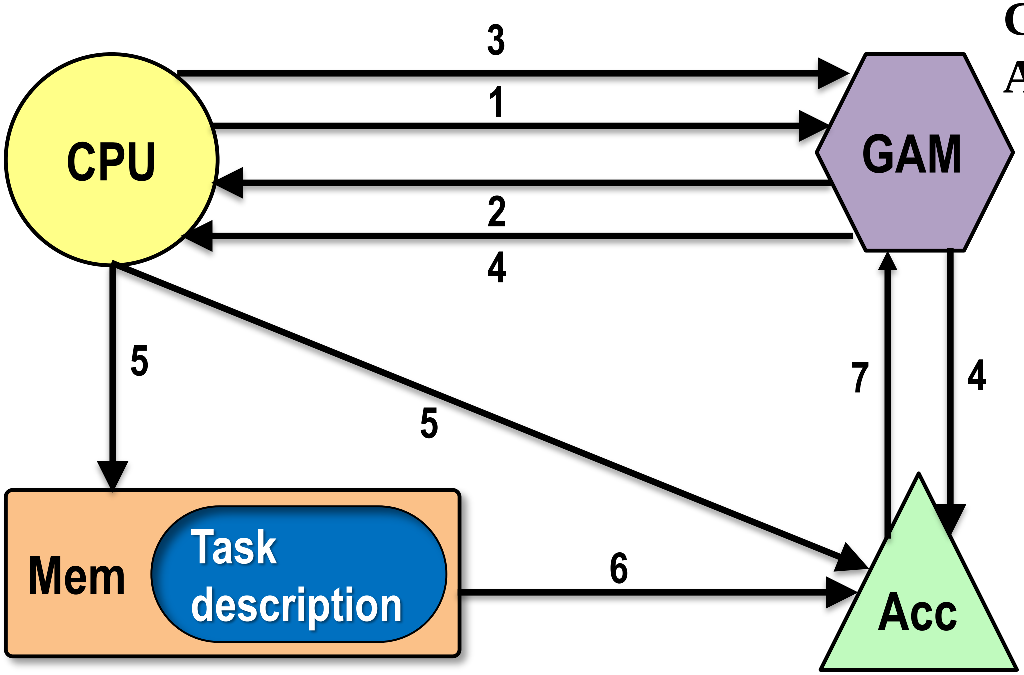


ABB: Accelerator Building Block

Auto-generated application using accelerator chaining data flow

(Automated) Application Execution on ARA



GAM: Global Acc Manager

Extended ISA	
acc-req	type
acc-rsrv	id, time
acc-cmd	id, cmd, addr
acc-free	id

1. Request available accelerators (acc-req)
2. Response available ones & waiting time
3. Request reservation (acc-rsv) and wait
4. Reserve accelerator, send it the core ID
5. The core shares a task description and start the accelerator (acc-cmd)
6. Read task & start work
7. Work done, notify the GAM
8. Free accelerators (acc-free)

Users don't have to worry about these, we provide a dataflow language and tool to automatically generate the library

FPGA vs GPU Results

✓ Ported a comprehensive set of 15 kernels from widely-used GPU benchmark suite Rodinia to FPGA using HLS C

- Performance: 3 FPGA kernels win, 3 kernels comparable
- Perf/watt: 6 FPGA kernels win, 4 kernels is > 2x worse than GPU

✓ Proposed an analytical model with new metrics (pipe_OPC and e_para_factor) to analyze FPGA and GPU performance

- FPGAs often have better pipe_OPC due to their pipeline customization
- FPGAs often lose in e_para_factor due to off-chip BW limitation
- With higher BW, 9 out of 15 FPGA kernels achieve > half of GPU perf

✓ Future work: port to the latest Amazon F1 instance for FPGA and P3 instance for GPU, compare and analyze perf and perf/dollar