Analysis and Modeling of Collaborative Execution Strategies for Heterogeneous CPU-FPGA Architectures

Sitao Huang[1], Li-Wen Chang[2], Izzat El Hajj[3], **Simon Garcia De Gonzalo[1]**, Juan Gómez-Luna[4], Sai Rahul Chalamalasetti[5], Mohamed El Hadedy[6], Dejan Milojicic[5], Onur Mutlu[4], Deming Chen[1], Wen-mei Hwu[1]

[1]University of Illinois at Urbana-Champaign, USA
[2]Microsoft, USA
[3]American University of Beirut, Lebanon
[4]ETH Zürich, Switzerland
[5]Hewlett Packard Labs, USA
[6]California State Polytechnic University, Pomona, USA

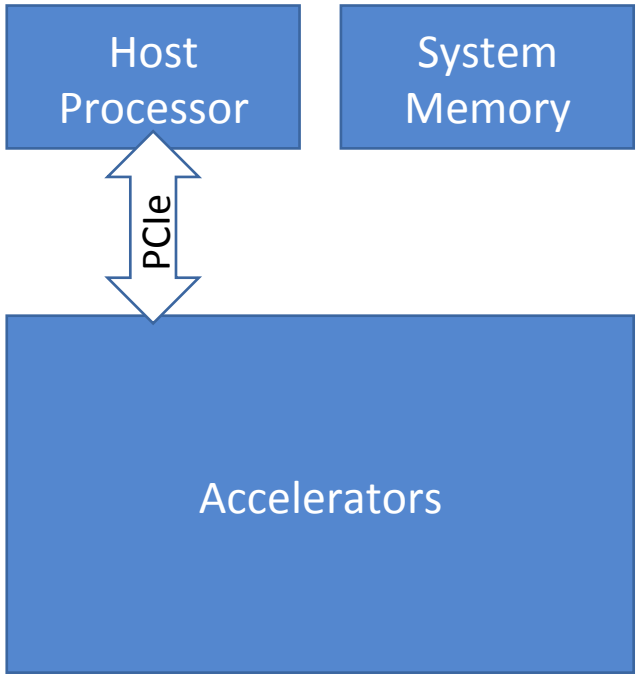
ILLINOIS
Electrical & Computer Engineering
COLLEGE OF ENGINEERING

Outline

- Introduction to Collaborative Computing
- Analytical Models of Collaborative Computing
- Evaluation
  - Evaluation platform
  - Data partitioning vs. task partitioning
  - Impact of kernel replication
- Key Insights
- Chai Benchmarks for CPU-FPGA Systems

Collaborative Computing

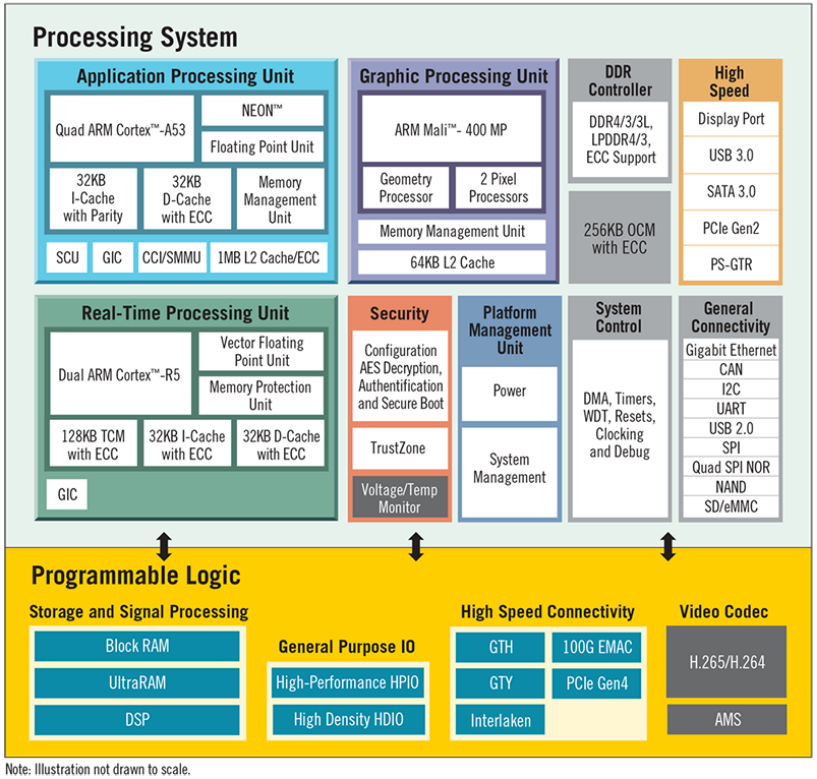
- Traditionally, accelerators (GPUs, FPGAs, etc.) have been used as *offload* engines



Host processor *offloads* computation tasks to accelerators

Collaborative Computing

- Traditionally, accelerators (GPUs, FPGAs, etc.) have been used as *offload* engines
- Heterogeneous architectures moving towards tighter integration
  - Unified memory
  - System-wide atomics

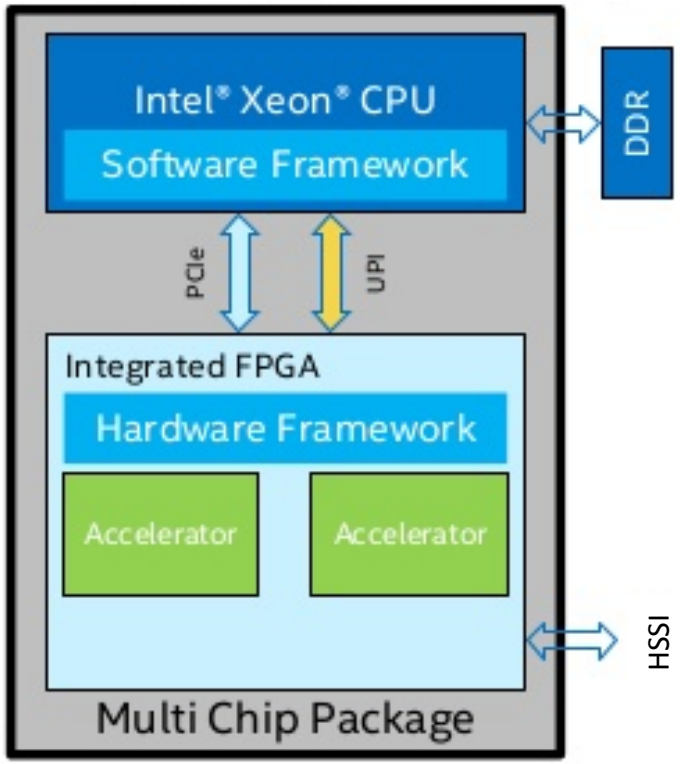


Xilinx Zynq UltraScale+ MPSoC

Collaborative Computing

- Traditionally, accelerators (GPUs, FPGAs, etc.) have been used as *offload* engines
- Heterogeneous architectures moving towards tighter integration
  - Unified memory
  - System-wide atomics
- Tighter integration allows fine-grained collaboration

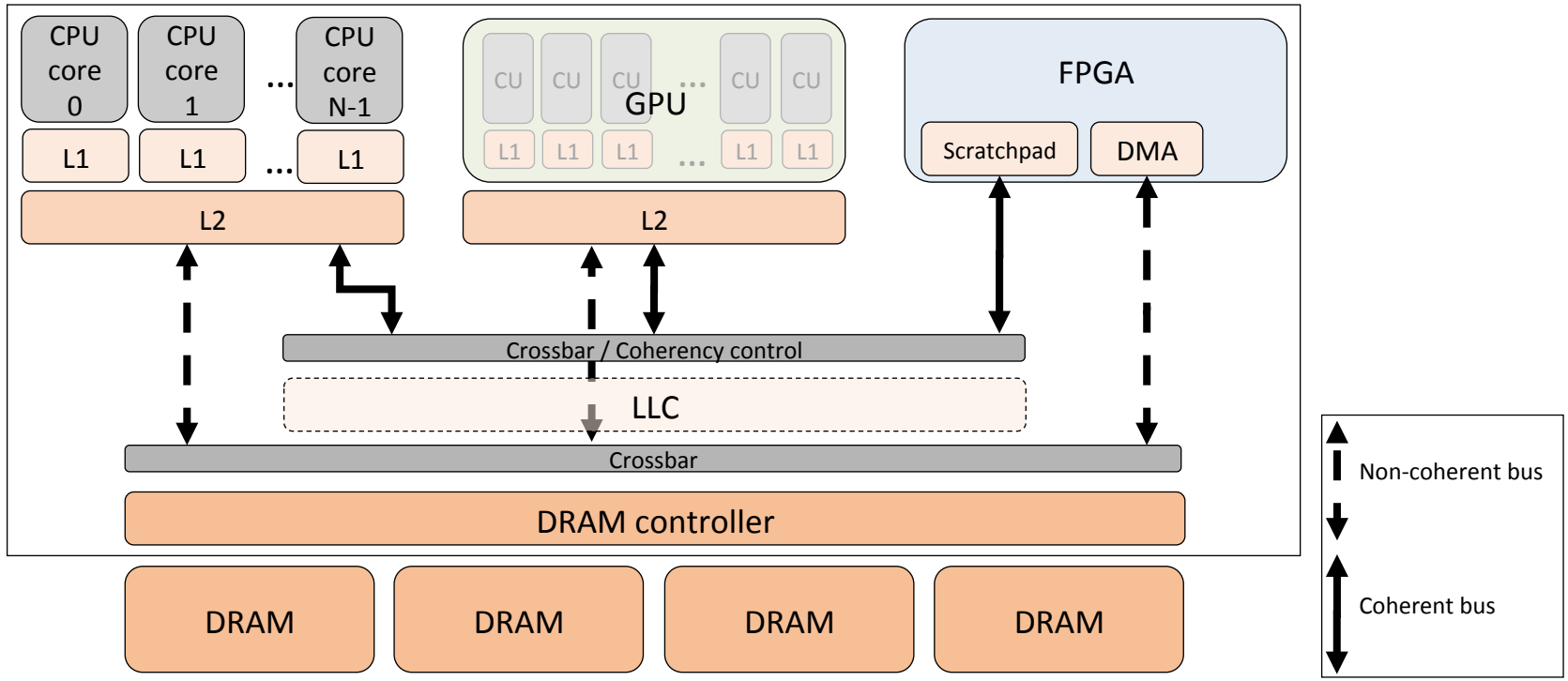
Key challenge: identify the best CPU-FPGA collaboration strategy



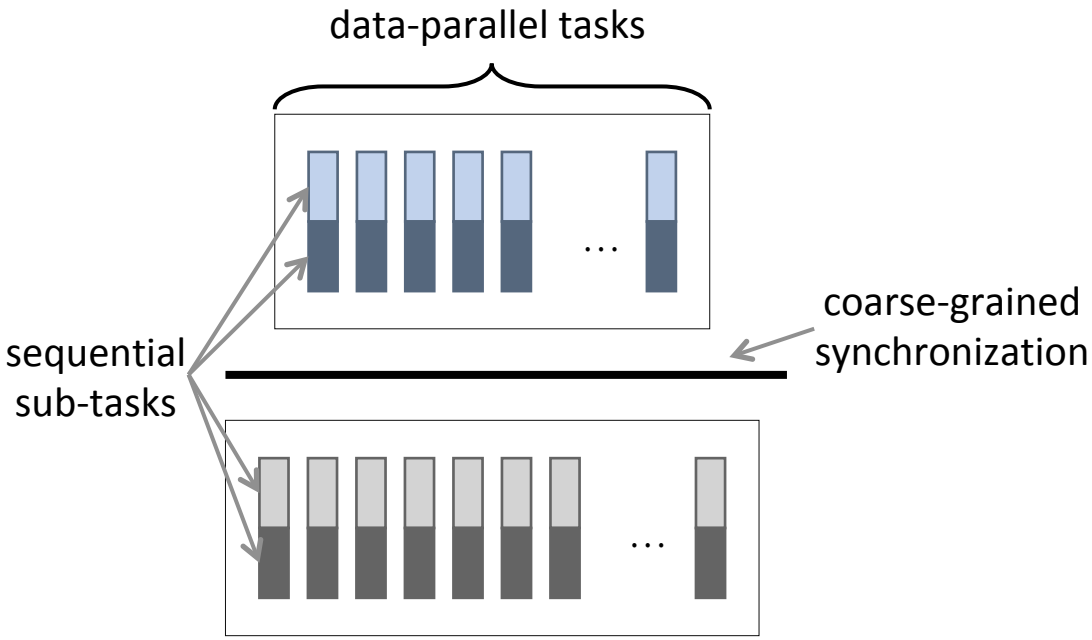
Intel Xeon + FPGA Integrated Platform (MCP)

Integrated Heterogeneous Systems

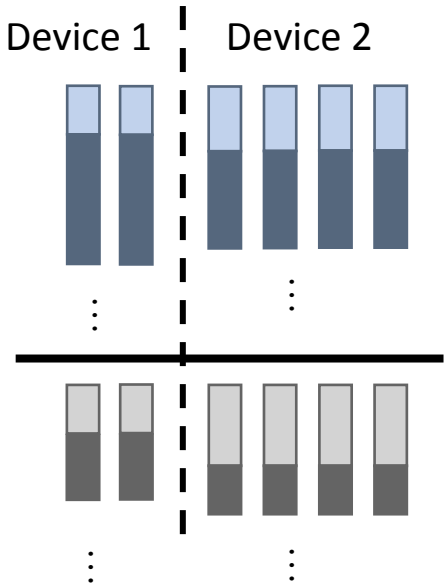
Our vision of an integrated heterogeneous system:



Collaborative Patterns

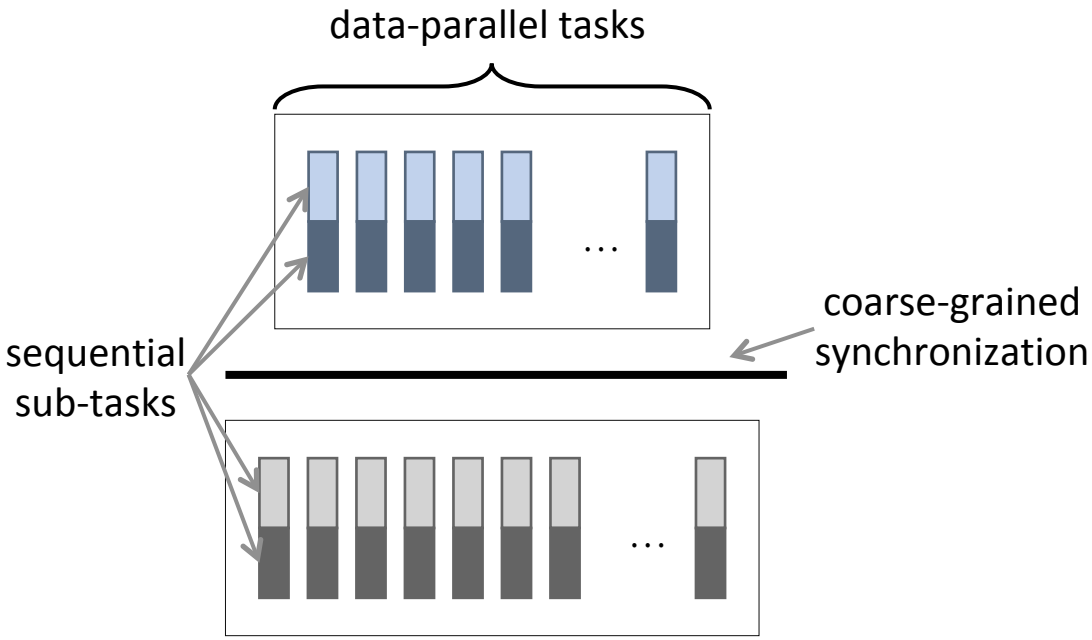


Program Structure

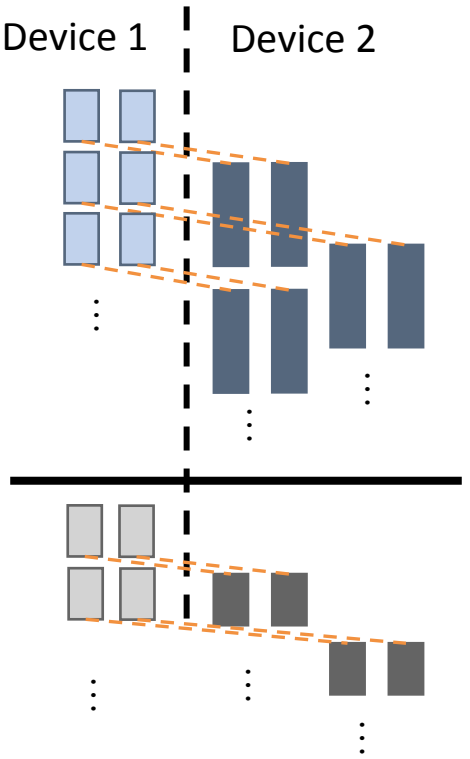


Data Partitioning

Collaborative Patterns

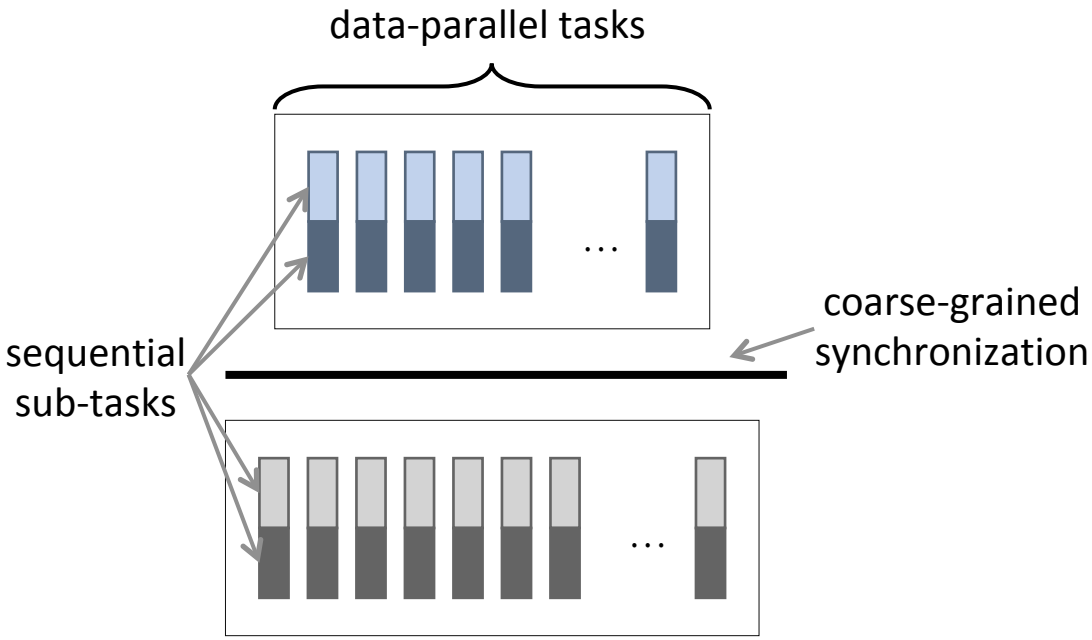


Program Structure

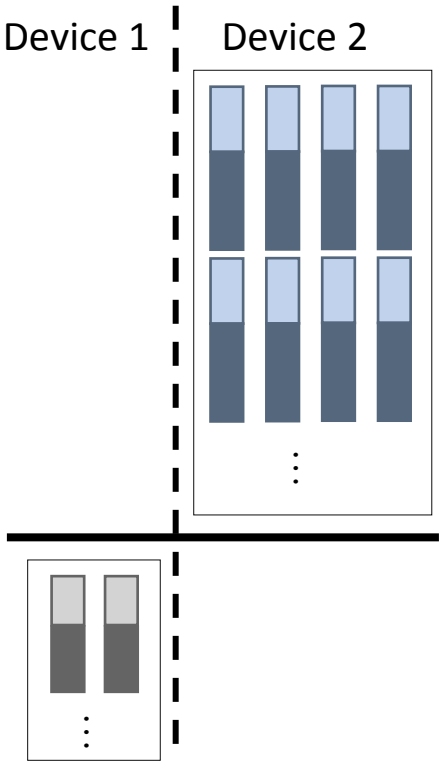


Fine-grained Task Partitioning

Collaborative Patterns



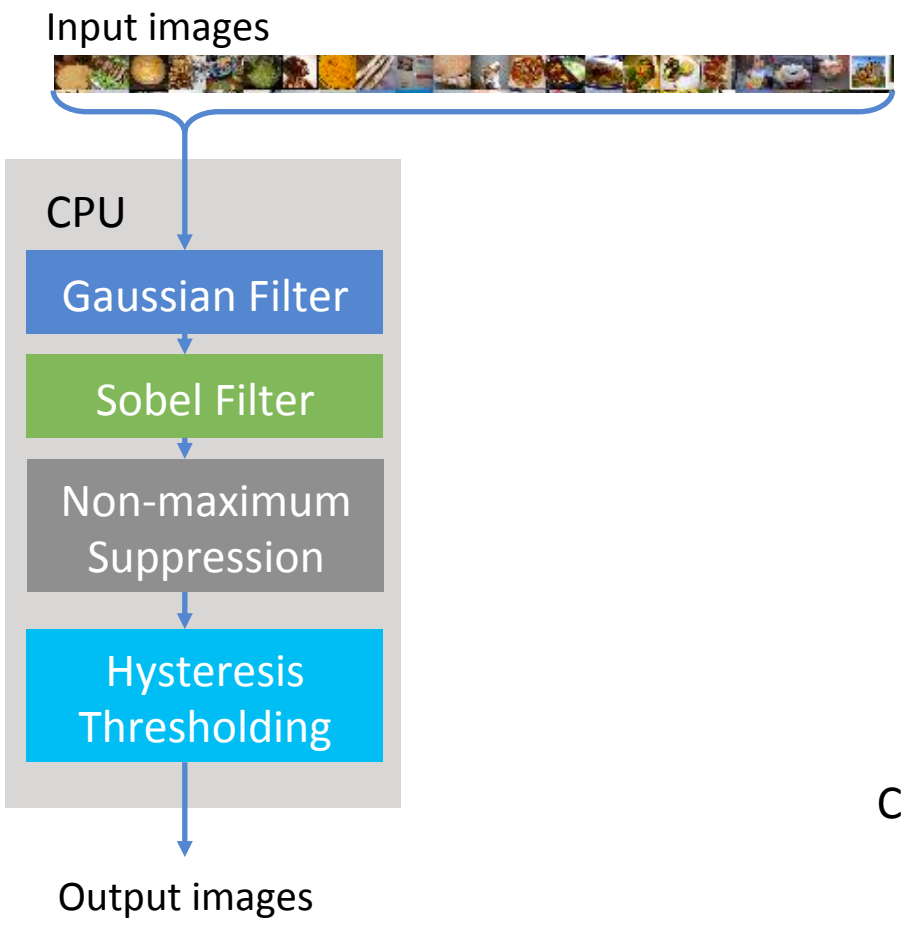
Program Structure



Coarse-grained Task Partitioning

Data Partitioning

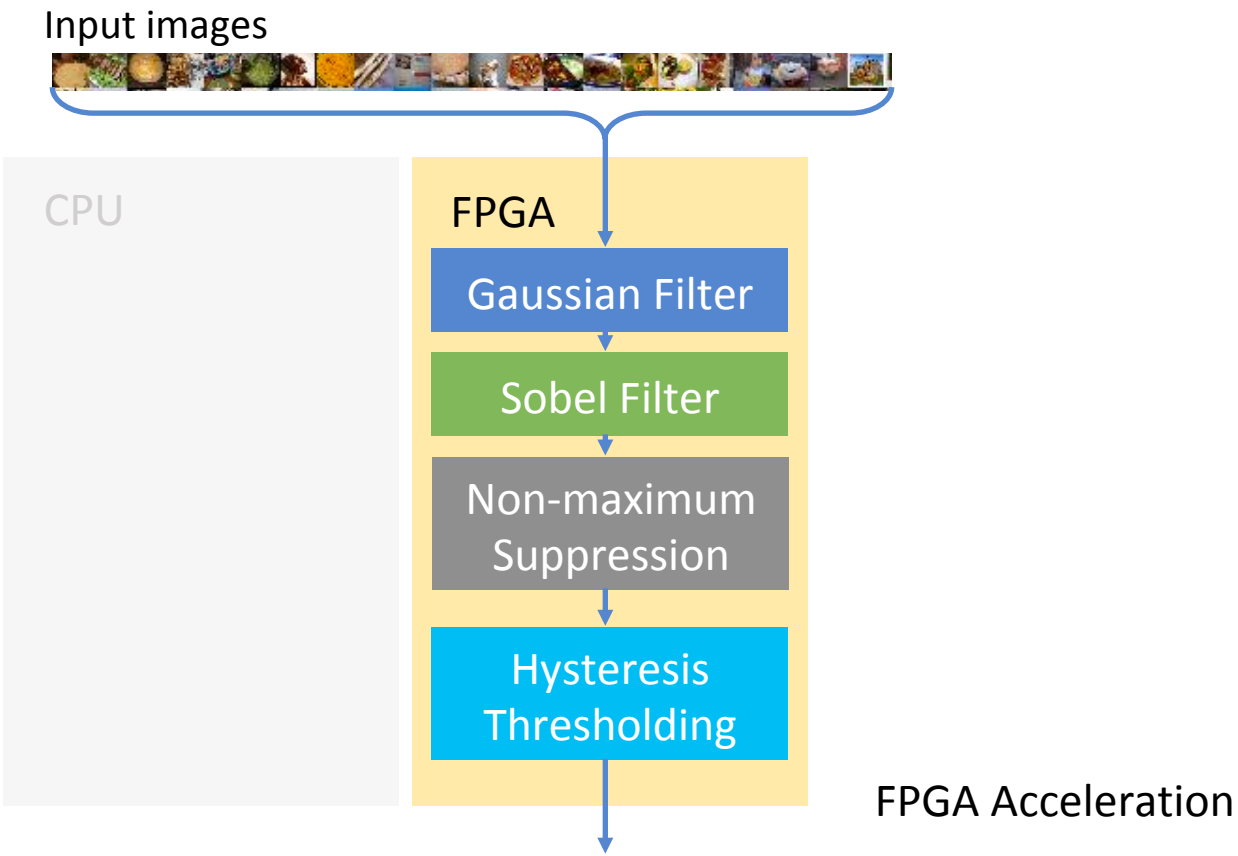
Using Canny Edge Detection (CED) as an example



CPU Implementation

Data Partitioning

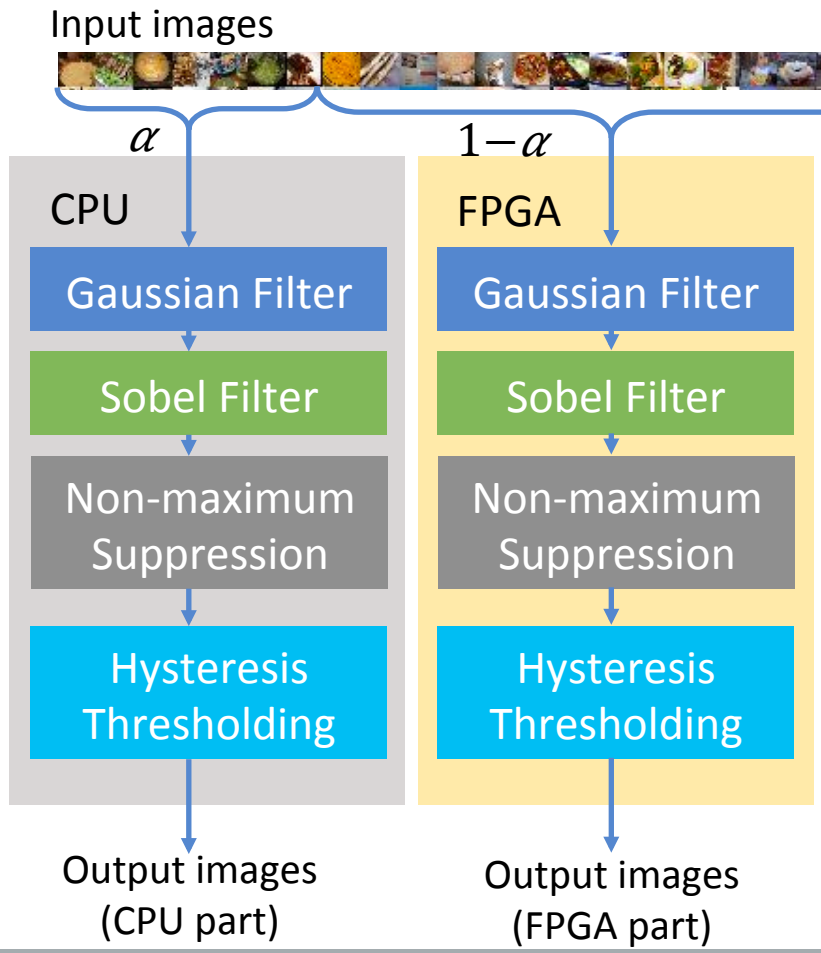
Using Canny Edge Detection (CED) as an example



FPGA Acceleration

Data Partitioning

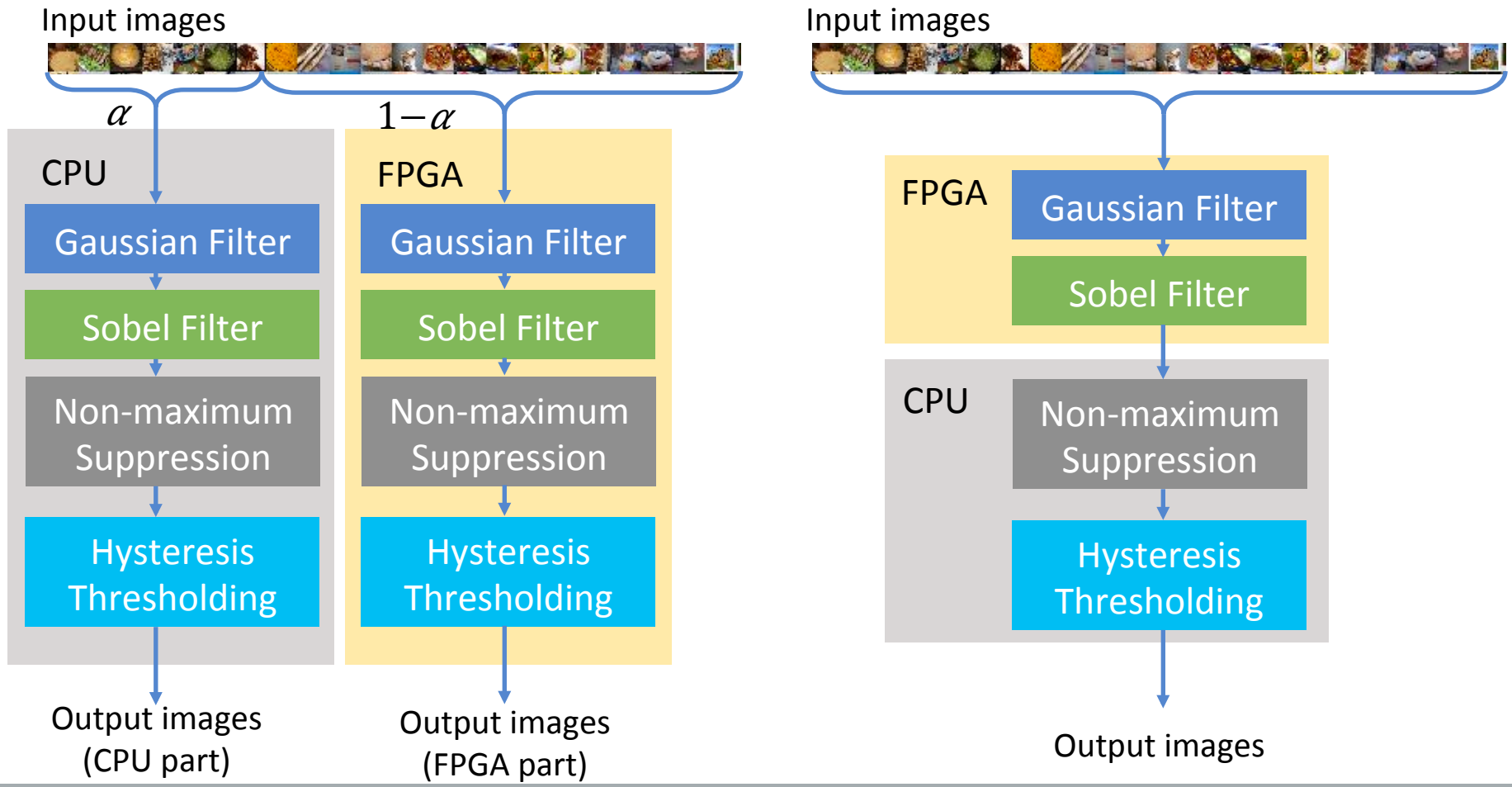
Using Canny Edge Detection (CED) as an example



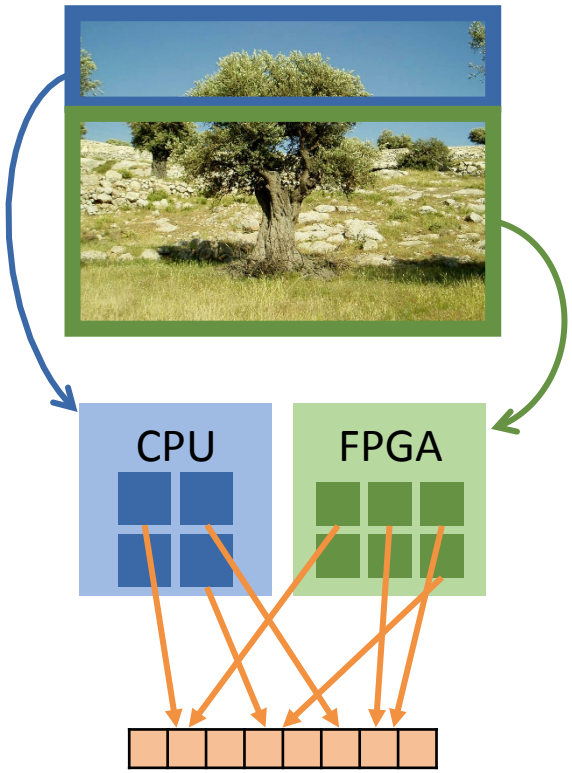
CPU-FPGA Collaboration

Data Partitioning vs. Task Partitioning

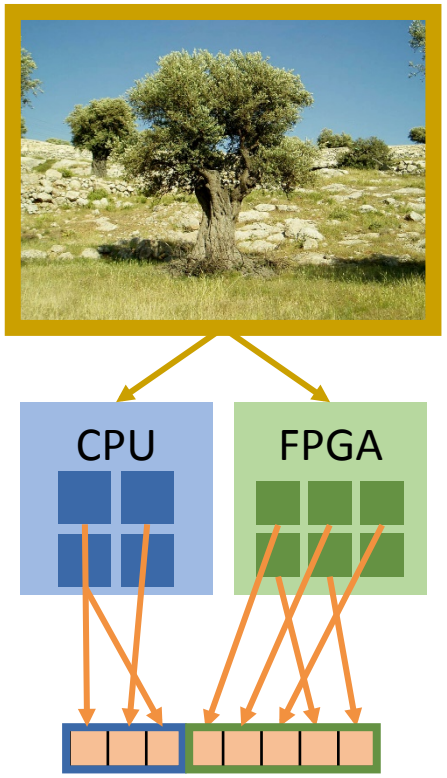
Using Canny Edge Detection (CED) as an example



Another Data Partitioning Example: Image Histogram



Input pixels distributed across devices



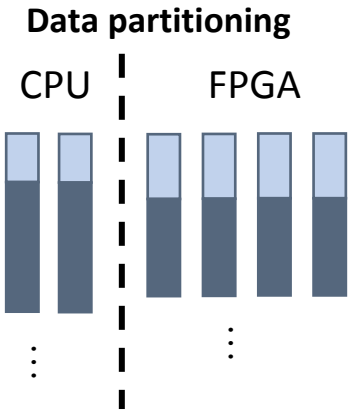
Output bins distributed across devices

Analytical Models

- N: Number of data parallel tasks in the application
- $t\downarrow i, C$: Execution time of sub-task i by a CPU worker
- $t\downarrow i, F$: Execution time of sub-task i by an FPGA worker
- $w\downarrow C$: Number of available CPU workers
- $w\downarrow F$: Number of available FPGA workers
- β: Distribution and aggregation overhead factor
- α: Fraction of data parallel tasks assigned to CPU

Analytical Models

- N: Number of data parallel tasks in the application
- $t\downarrow i, C$: Execution time of sub-task i by a CPU worker
- $t\downarrow i, F$: Execution time of sub-task i by an FPGA worker
- $w\downarrow C$: Number of available CPU workers
- $w\downarrow F$: Number of available FPGA workers
- β: Distribution and aggregation overhead factor
- α: Fraction of data parallel tasks assigned to CPU



Data partitioning

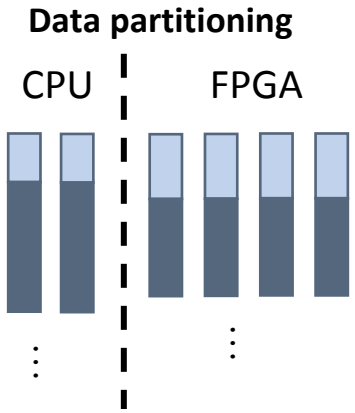
The total execution time is

Total CPU execution time (sequential execution)

Total FPGA execution time (sequential execution)

$$t\downarrow data, total = \beta\downarrow data \cdot \max(\alpha N \sum i\uparrow\ t\downarrow i,C \ / w\downarrow C \ , \ (1-\alpha) N \sum i\uparrow\ t\downarrow i,F \ / w\downarrow F \ )$$

Analytical Models



- N: Number of data parallel tasks in the application
- $t_{i,C}$: Execution time of sub-task i by a CPU worker
- $t_{i,F}$: Execution time of sub-task i by an FPGA worker
- w_C : Number of available CPU workers
- w_F : Number of available FPGA workers
- β: Distribution and aggregation overhead factor
- α: Fraction of data parallel tasks assigned to CPU

Data partitioning

The total execution time is

$$t_{data,total} = \beta_{data} \cdot \max\left(\frac{\alpha N \sum_i t_{i,C}}{w_C}, \frac{(1-\alpha) N \sum_i t_{i,F}}{w_F}\right)$$

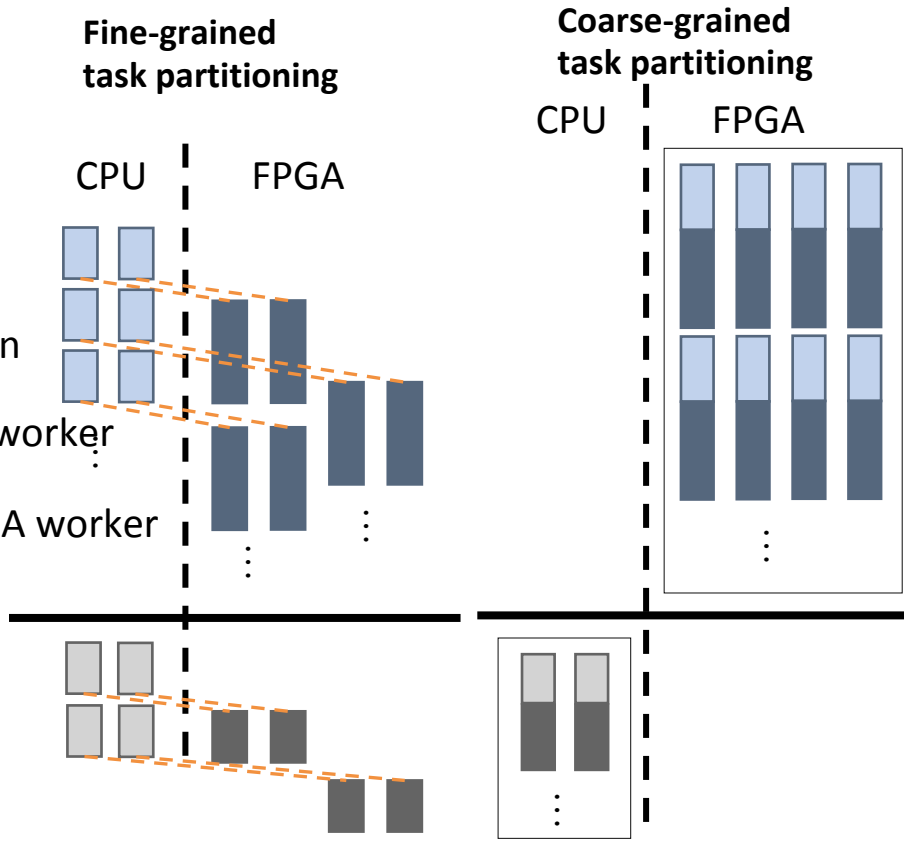
Fixing all the variables except α, the optimal α (global minimum point) is

$$\alpha^* = \frac{\sum_i t_{i,F} / w_F}{\sum_i t_{i,C} / w_C + \sum_i t_{i,F} / w_F}$$

Workloads of CPU and FPGA workers are balanced

Analytical Models

- N: Number of data parallel tasks in the application
- $t\downarrow i, C$: Execution time of sub-task i by a CPU worker
- $t\downarrow i, F$: Execution time of sub-task i by an FPGA worker
- $w\downarrow C$: Number of available CPU workers
- $w\downarrow F$: Number of available FPGA workers
- β: Distribution and aggregation overhead factor



Fine-grained task partitioning

The total execution time is

$$t\downarrow task, total = \beta\downarrow task\, N \cdot \max\left(\sum i\in S\downarrow C \uparrow \square t\downarrow i,C \ / w\downarrow C \ , \sum i\in S\downarrow F \uparrow \square t\downarrow i,F \ / w\downarrow F \ \right)$$

(Assume sub-tasks are very fine-grained)

Coarse-grained task partitioning

The total execution time is

$$t\downarrow task, total = \beta\downarrow task\, N \cdot \left(\sum i\in S\downarrow C \uparrow \square t\downarrow i,C \ / w\downarrow C + \sum i\in S\downarrow F \uparrow \square t\downarrow i,F \ / w\downarrow F \ \right)$$

Chai Benchmark Suite

Chai: **C**ollaborative **H**eterogeneous **A**pplications for **I**ntegrated-architectures

- Chai benchmark suite:

  chai-benchmarks.github.io

  - 14 benchmarks covers data partitioning, fine-grain task partitioning, and coarse-grain task partitioning patterns
  - OpenCL, C++ AMP, and CUDA versions
  - Unified memory and system-wide atomic versions and traditional discrete architecture versions

Evaluated Chai Benchmarks

Benchmark	Description	Strategy
CED-D	Canny Edge Detection	Data Partitioning
CED-T	Canny Edge Detection	Task Partitioning
RSC-D	Random Sample Consensus	Data Partitioning
RSC-T	Random Sample Consensus	Task Partitioning
BS	Bézier Surface	Data Partitioning
HSTO	Image Histogram	Data Partitioning
SSSP	Single-Source Shortest Path	Task Partitioning
TQ	Task Queue System (Synthetic)	Task Partitioning
TQH	Task Queue System (Histogram)	Task Partitioning

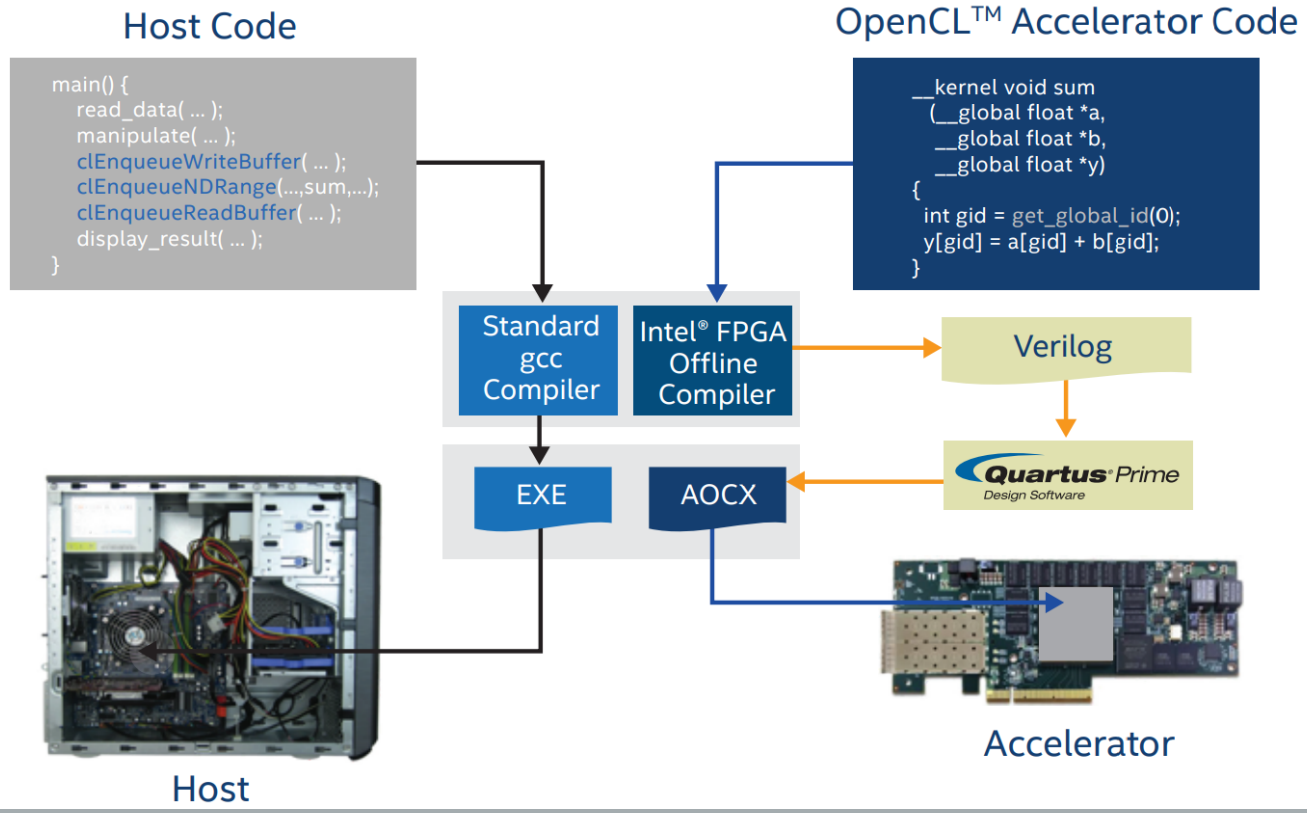
OpenCL-D (OpenCL discrete architecture) versions of these benchmarks are used.

Evaluation Platforms

	Platform A	Platform B
FPGA Board	Terasic DE5-Net	Nallatech 510T
FPGA Chip	**Intel Stratix V GX**	**Intel Arria 10 GX**
On-Board Memory	4 GB (DDR3)	8 GB (DDR4)
Host CPU	Intel Xeon E3-1240 v3	Intel Xeon E5-2650 v3
Host Memory	8 GB (DDR3)	96 GB (DDR4)
Interface	PCIe gen3.0 x8	PCIe gen3.0 x8

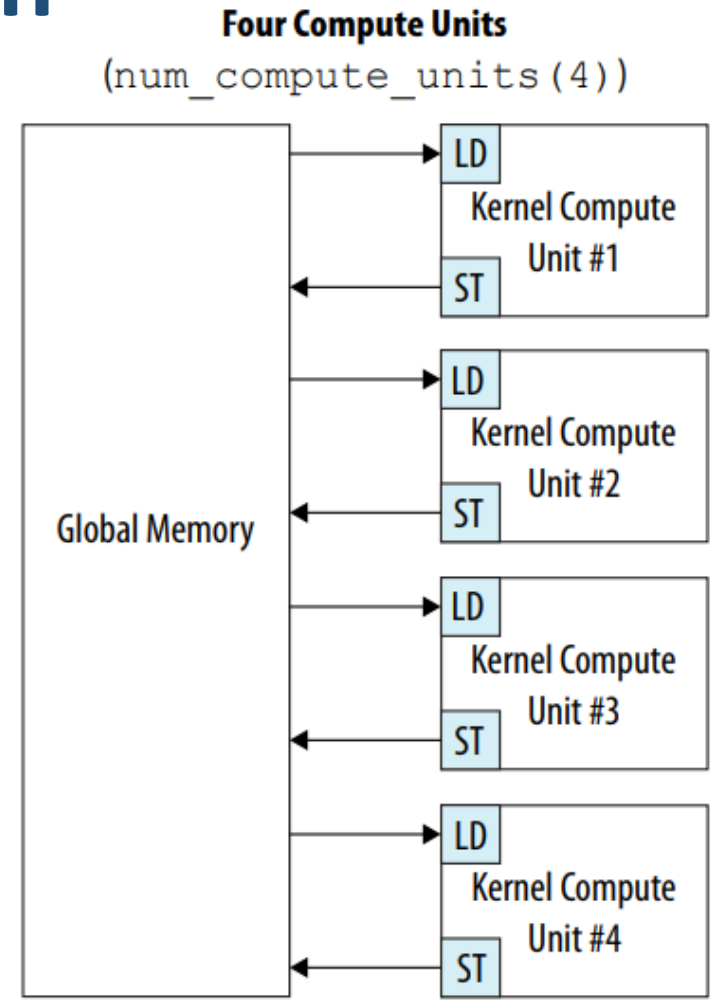
Intel OpenCL SDK for FPGA

- Intel OpenCL SDK for FPGA is used to compile and synthesize host executable and FPGA design



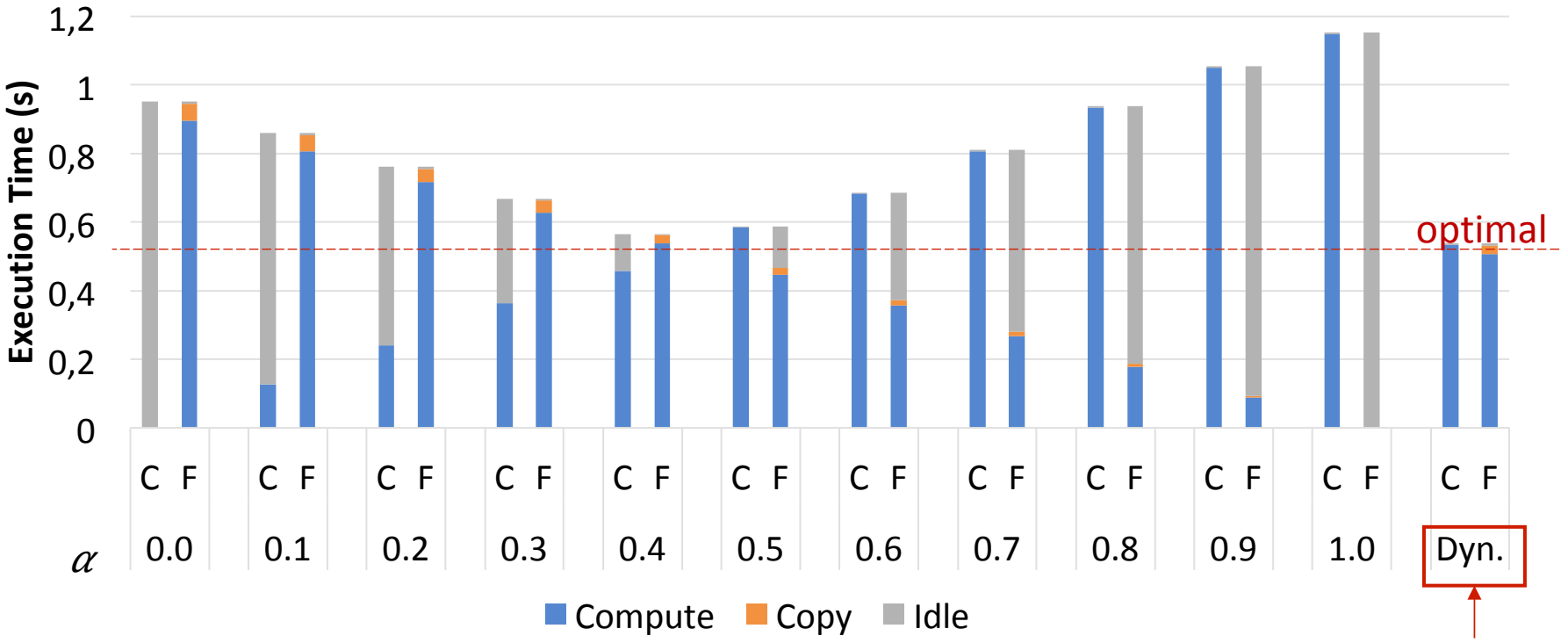
Compute Unit Replication

- OpenCL kernels are synthesized to compute units on FPGA
- The compute units on FPGA can be replicated by adding `num_compute_units` attribute in the OpenCL kernel code
- `num_compute_units` attribute modifies the **number of compute units to which work-groups can be scheduled**, which also modifies the **number of times a kernel accesses global memory**
- We evaluate the impact of compute unit replication



Intel® FPGA SDK for OpenCL™ Best Practices Guide

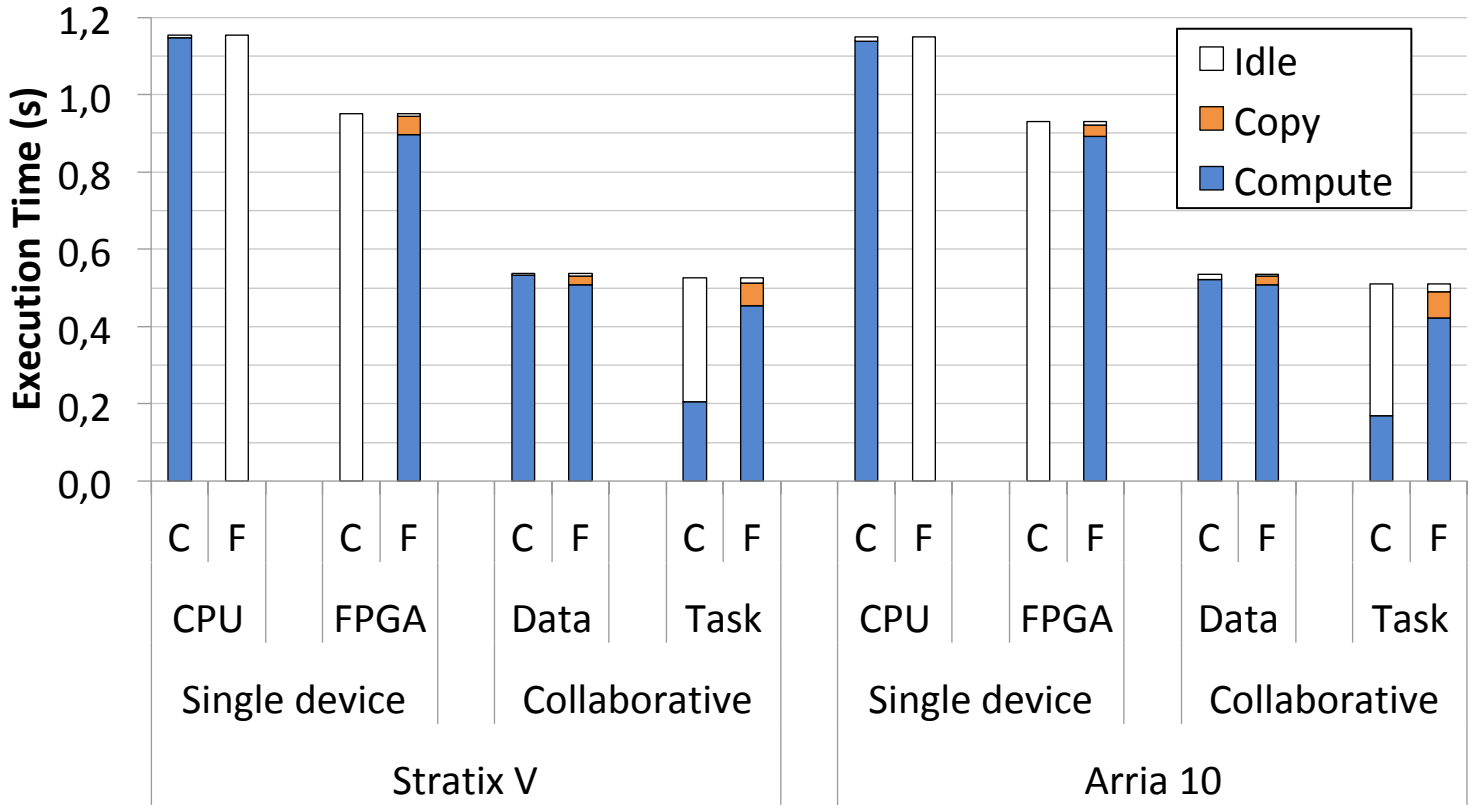
Evaluation: Canny Edge Detection (Data Partitioning)



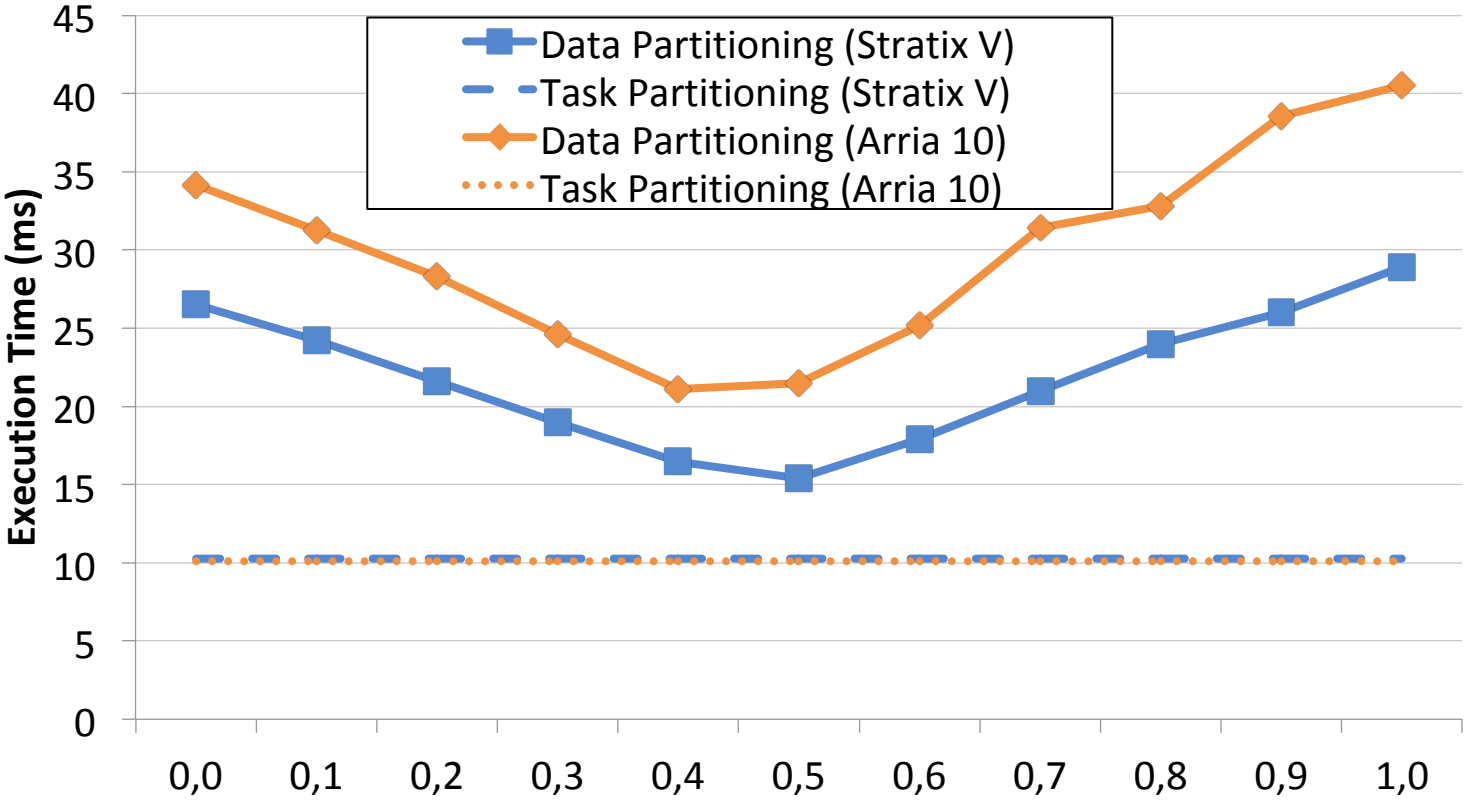
- C: CPU; F: FPGA
- α: Fraction of data parallel tasks assigned to CPU

Dynamic data partitioning: assigning batch of data parallel tasks to idle devices, achieving dynamic workload balance

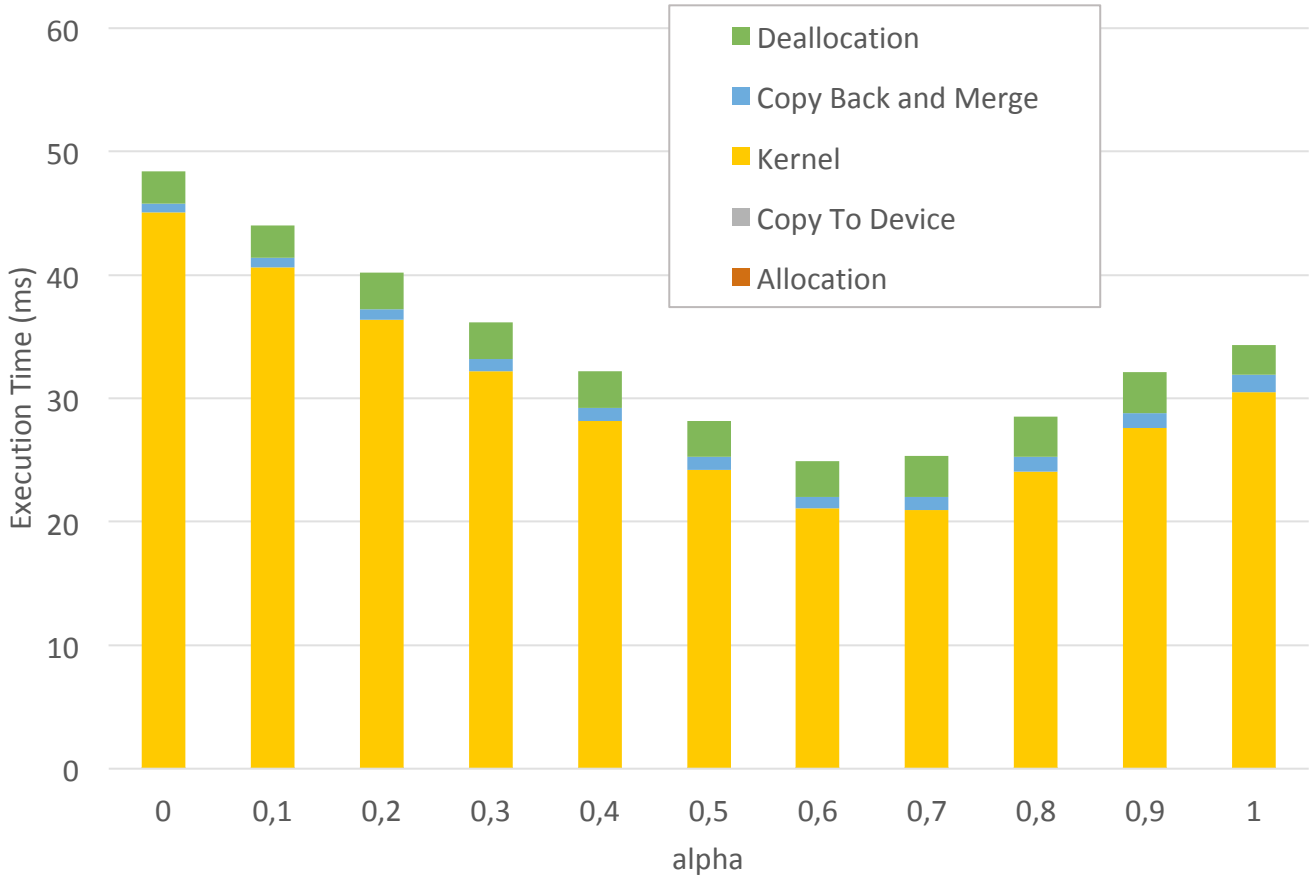
Evaluation: Canny Edge Detection (Data Partitioning and Task Partitioning)



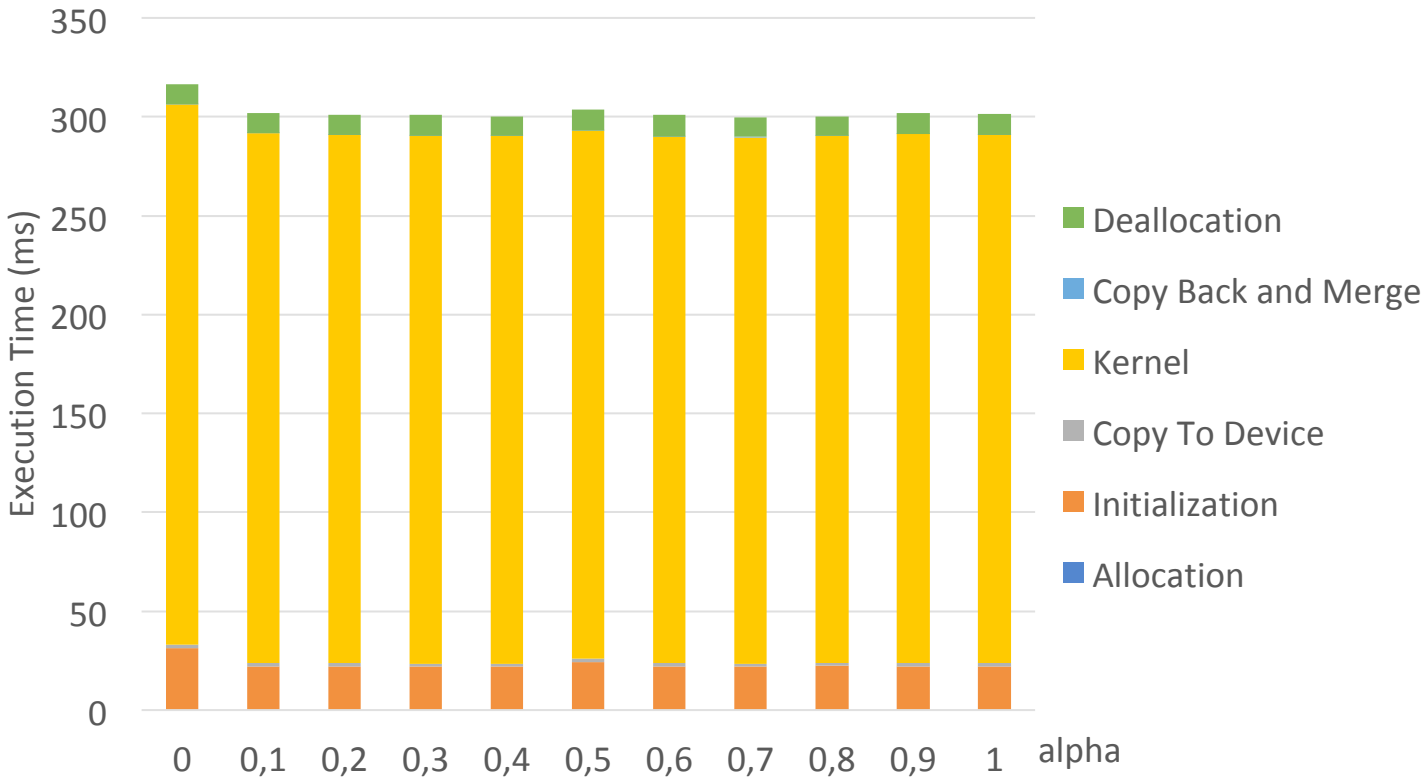
Evaluation: Random Sample Consensus (Data Partitioning and Task Partitioning)



Bézier Surface (BS, Data Partitioning)

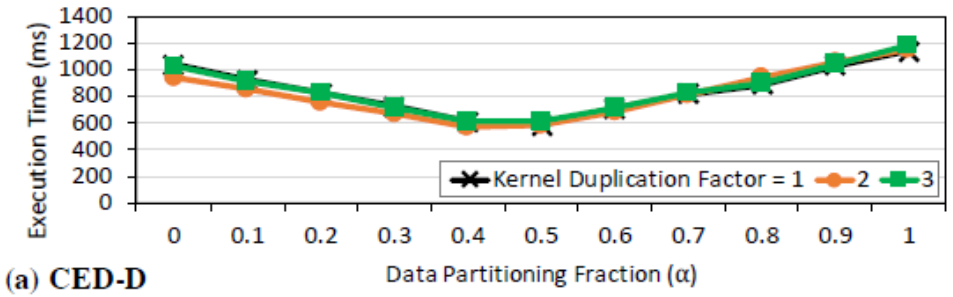


Histogram (HSTO, Output Data Partitioning)

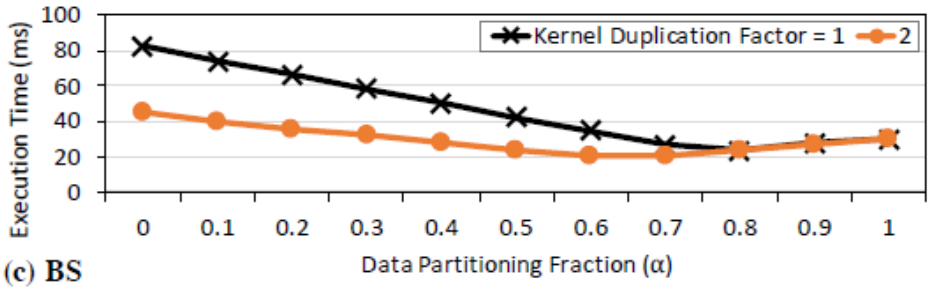


Kernel Replication – Data Partitioning

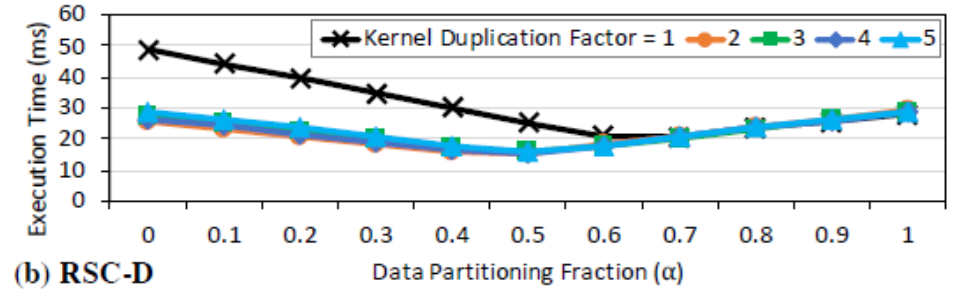
We evaluated the performance under different kernel replication factors.



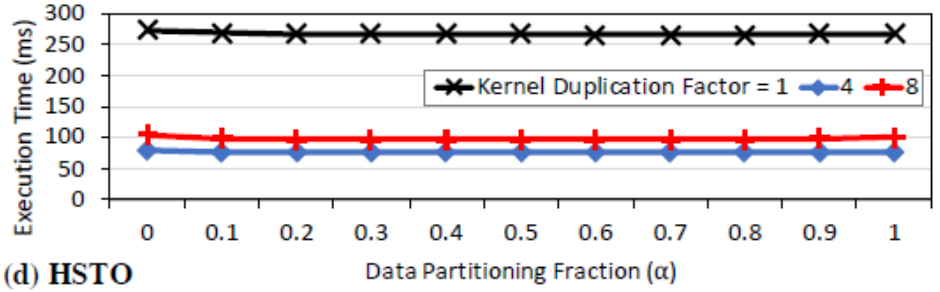
(a) CED-D



(c) BS



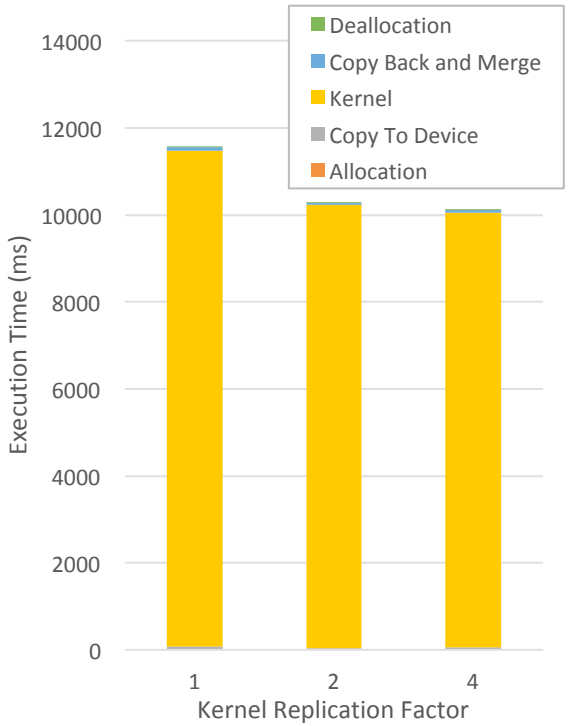
(b) RSC-D



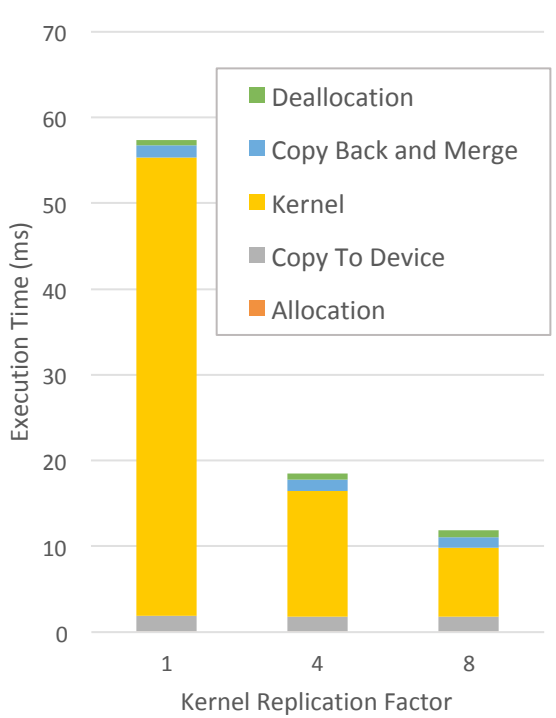
(d) HSTO

Kernel Replication – Task Partitioning

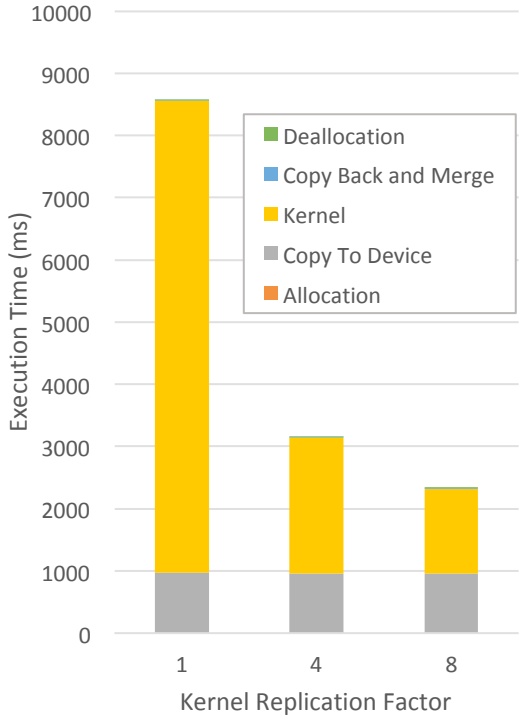
We evaluated the performance under different kernel replication factors.



SSSP



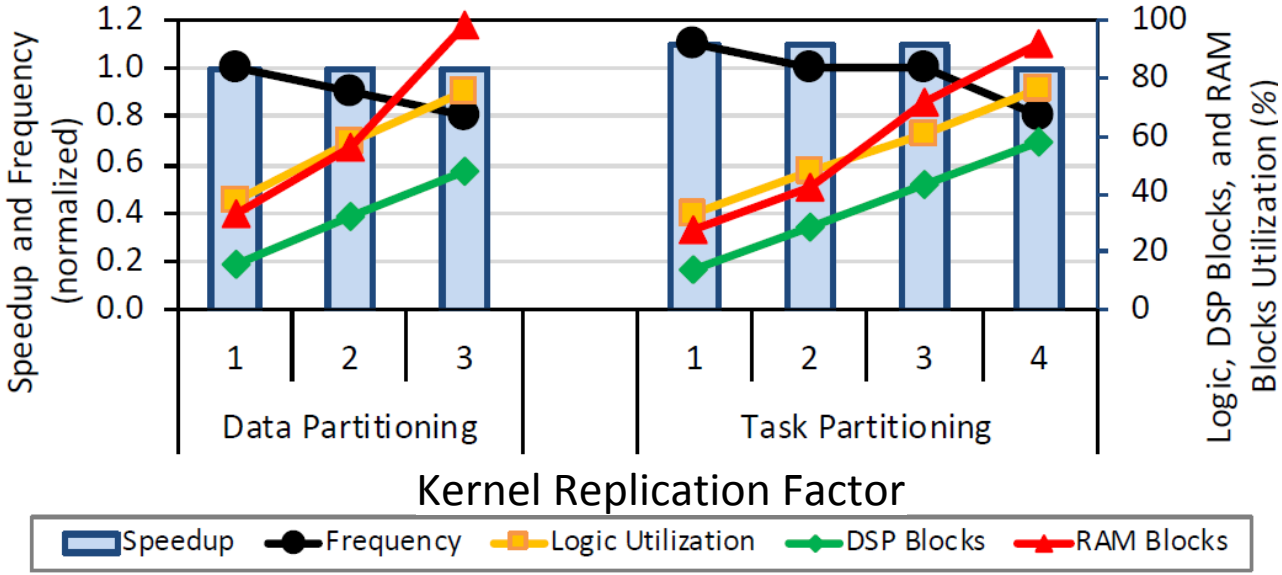
TQ



TQH

Impact of Replication

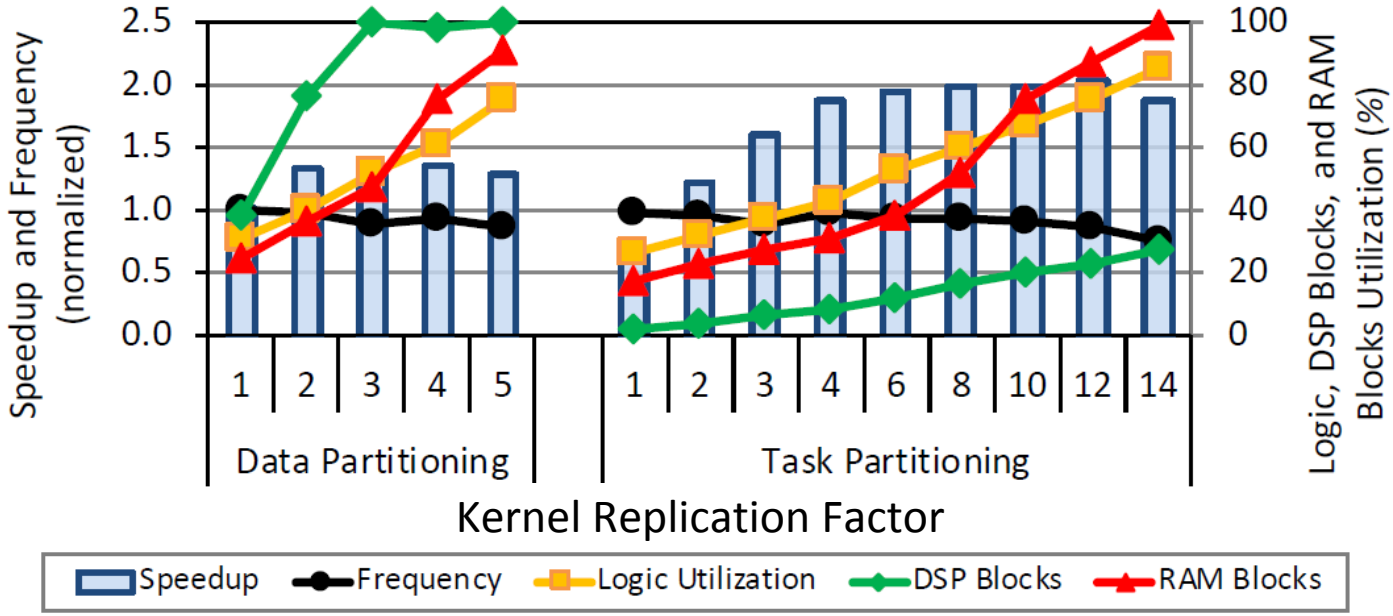
Canny Edge Detection



- Replication factor for this application has little impact on performance
- Further profiling reveals the reason of performance saturation is the saturation of the memory bandwidth
- Task partitioning can afford a larger replication factor

Impact of Replication

Random Sample Consensus



- Replication improves performance of this application
- Bounding resource: DSP blocks
- Task partitioning releases the pressure on DSP block and thus can afford a larger replication factor

Key Insights

- Collaborative execution is beneficial
- Data partitioning requires careful choice of partitions to provide the highest performance
- Task partitioning generally enables more kernel replication on the FPGA than data partitioning
- Data partitioning inflicts less burden on programmers and has less communication overhead than task partitioning
- OpenCL stack provides a convenient programming model while there is still room for better programmability and higher performance

Chai Project

- Papers:
  - **Analysis and Modeling of Collaborative Execution Strategies for Heterogeneous CPU-FPGA Architectures.** *ICPE'19.* (this work)
  - **Collaborative Computing for Heterogeneous Integrated Systems.** *ICPE'17 Vision Track*.
  - **Chai: Collaborative Heterogeneous Applications for Integrated-architectures.** *ISPASS'17*.
- **Chai** Benchmark Suite:
  - Website: chai-benchmarks.github.io
  - Code: github.com/chai-benchmarks/chai
  - Online Forum: groups.google.com/d/forum/chai-dev

Analysis and Modeling of Collaborative Execution Strategies for Heterogeneous CPU-FPGA Architectures

Sitao Huang[1], Li-Wen Chang[2], Izzat El Hajj[3], **Simon Garcia De Gonzalo[1]**, Juan Gómez-Luna[4], Sai Rahul Chalamalasetti[5], Mohamed El Hadedy[6], Dejan Milojicic[5], Onur Mutlu[4], Deming Chen[1], Wen-mei Hwu[1]

[1]University of Illinois at Urbana-Champaign, USA
[2]Microsoft, USA
[3]American University of Beirut, Lebanon
[4]ETH Zürich, Switzerland
[5]Hewlett Packard Labs, USA
[6]California State Polytechnic University, Pomona, USA

Thanks!

ILLINOIS
Electrical & Computer Engineering
COLLEGE OF ENGINEERING