THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Exploration of Cache Coherent CPU-FPGA Heterogeneous System

Wei Zhang

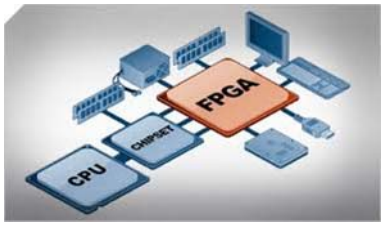
Department of Electronic and Computer Engineering

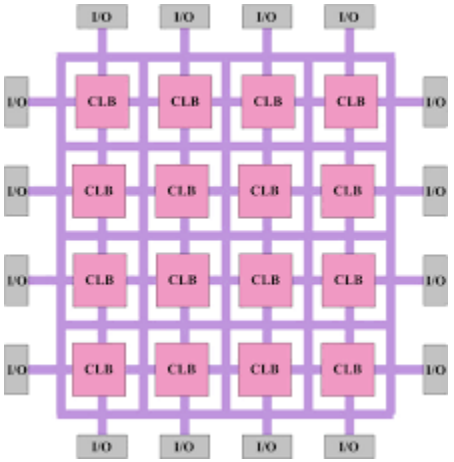
Hong Kong University of Science and Technology

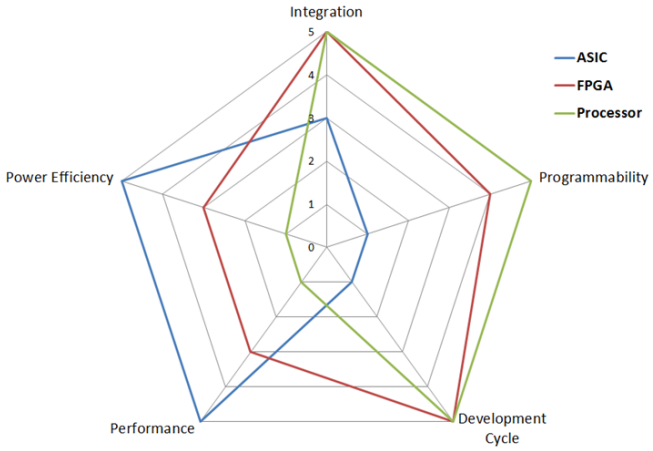
Outline

- Introduction to FPGA-based acceleration
- Heterogeneous CPU-FPGA architecture
- A full system simulator for exploration of various CPU-FPGA architectures
- A static and dynamic combined cache optimization strategy for cache coherent CPU-FPGA system
- Conclusion

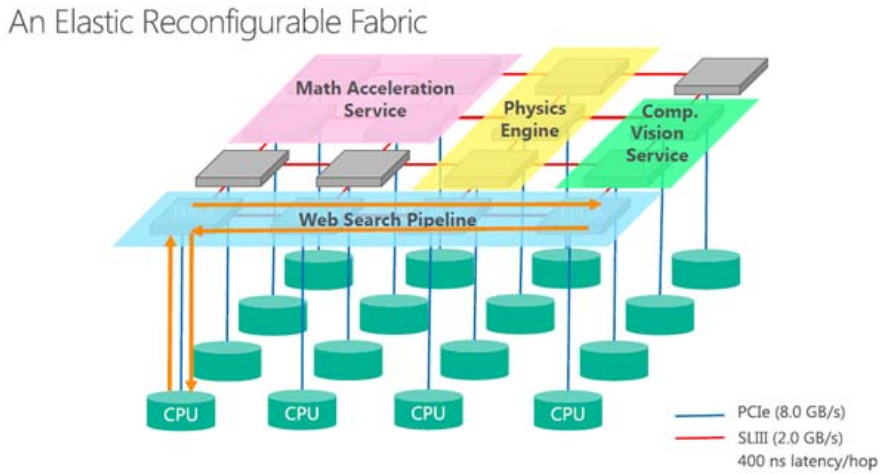
FPGA based Acceleration







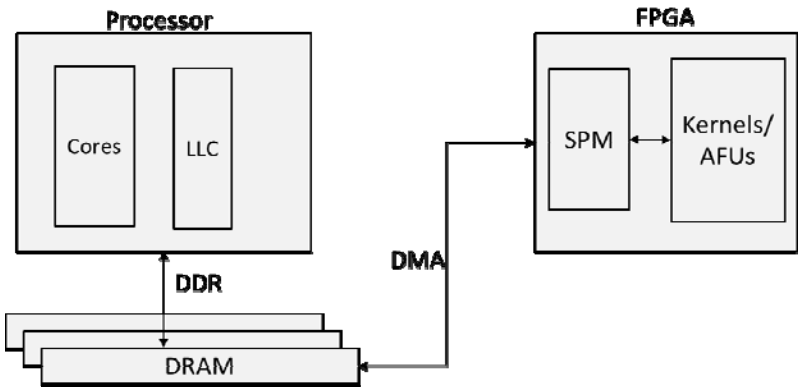
Field Programmable Gate Array



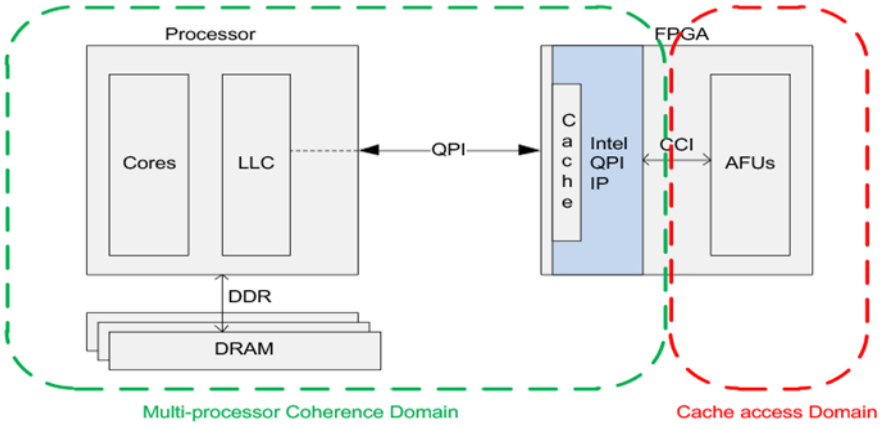
- FPGA accelerator
  - Tens of times faster than CPU
  - One to two orders of magnitude more power efficient than CPU and GPU
  - Post-fabrication reconfigurable
- FPGA usage in data center
  - Microsoft, Baidu, Tencent
  - 30% cost reduction

CPU-FPGA System Architecture

- Traditional DMA based design:
  - Manual efforts for DMA control
  - Coherency handled by user
  - Larger communication overhead through DRAM
  - Less hardware cost

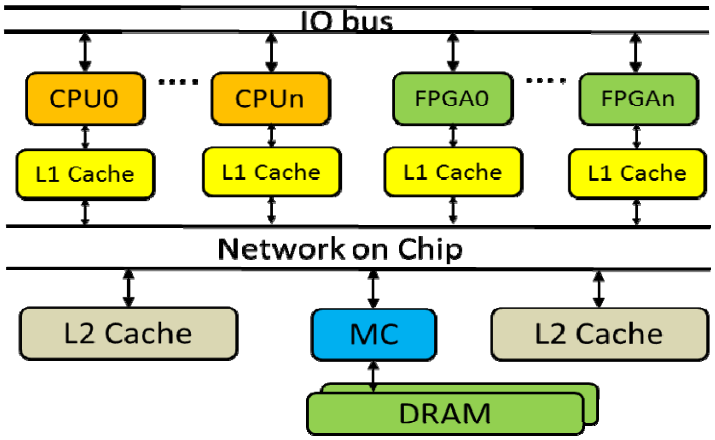


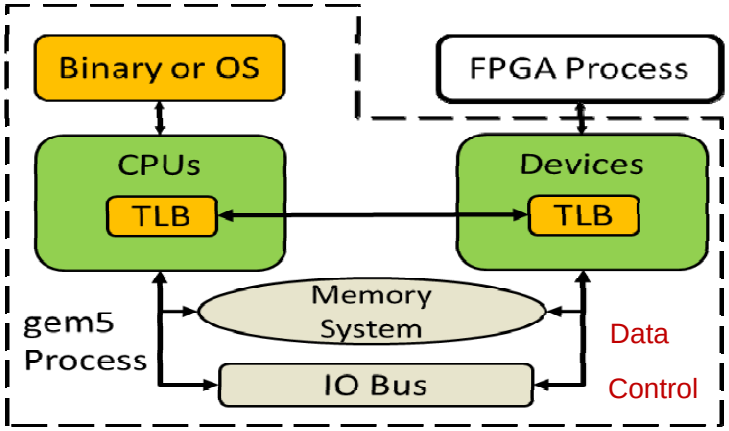
- Cache coherent design:
  - Automatic coherence guarantee
  - Simple programming model
  - Smaller communication overhead through LLC
  - Good for scattered access and large data space
  - Intel Harp and IBM Power 8



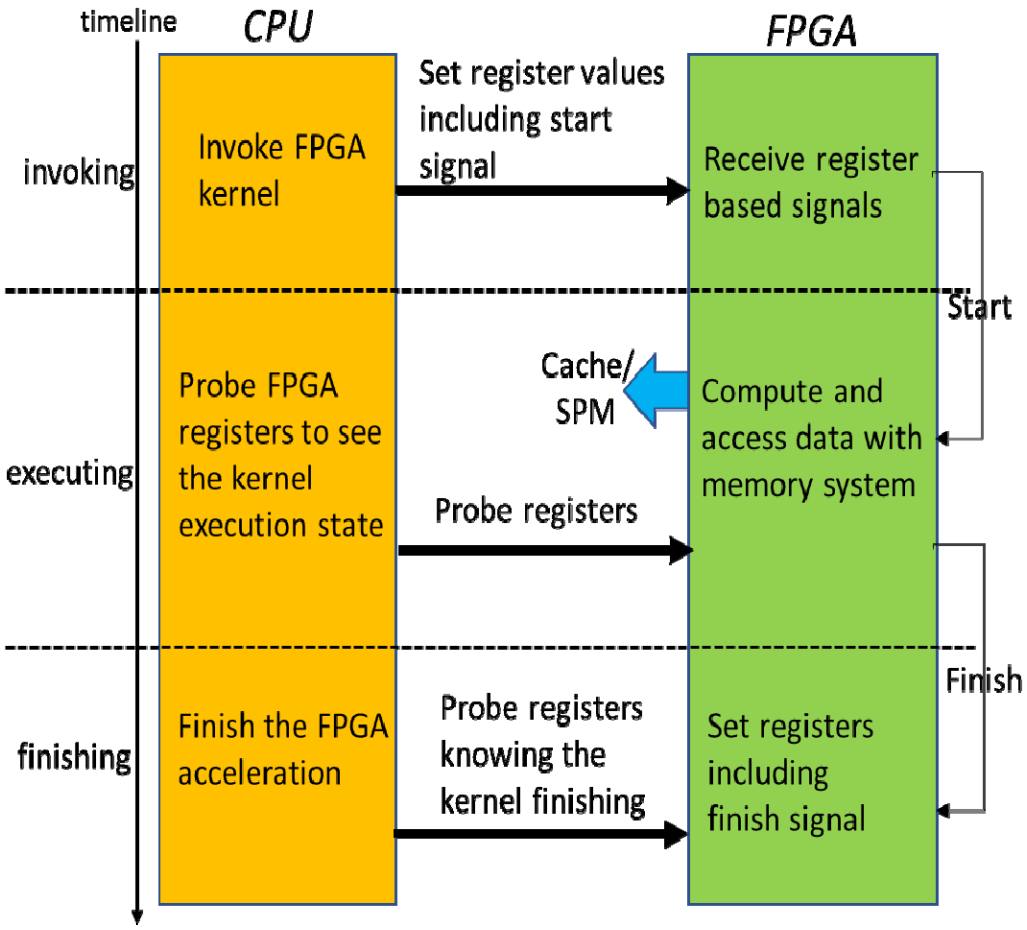
Gem5-HDL: Heterogeneous System Simulator

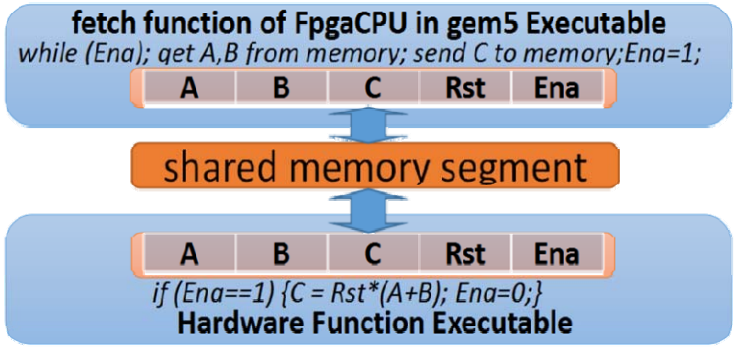
- Powerful support of gem5-HDL
  - Cycle-Accurate Co-Simulation of hardware accelerators and CPUs
    - ✓ Based on gem5 and Verilator
    - ✓ Accelerators described by Verilog or C/C++
    - ✓ FPGA model: Functionality, timing, IO signals
  - Shared coherent caches between CPUs and accelerators on FPGA
  - Runtime Control on accelerators
  - Flexible Architecture for on-chip interconnection work, NoC or BUS
  - Cooperation between separate processes based on Inter-Process Communication





Runtime Control Interface

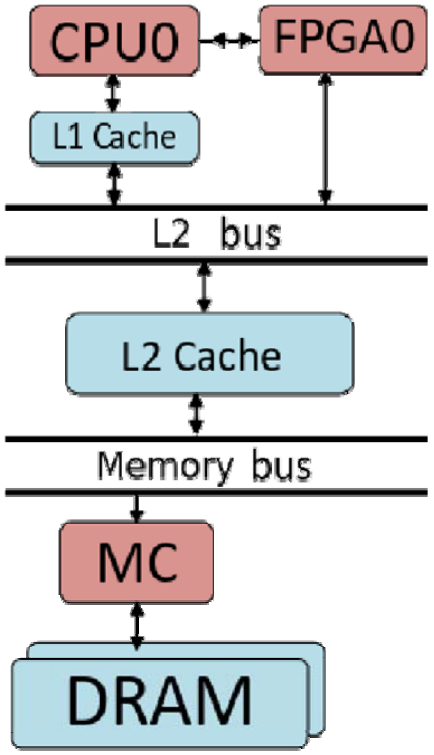


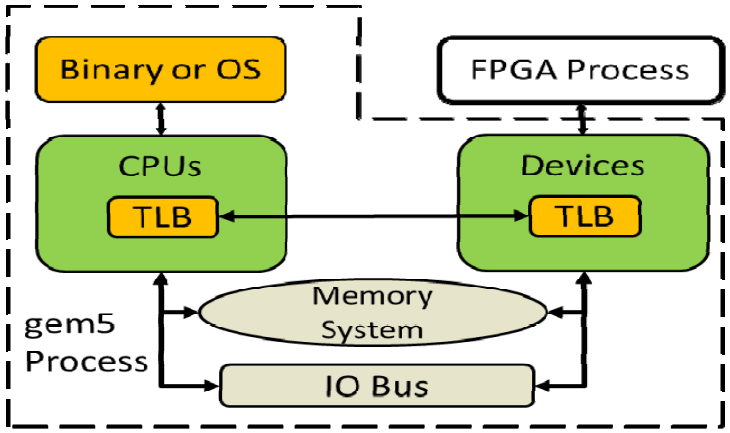


ReadBase	Base virtual address of input data
WriteBase	Base virtual address of output data
MemoryRange	Amount of input data
MemorySize	Size of a single input/output data
RunState	The running state of FPGA
Terminate	Terminate the process of FPGA
OccupyFPGA	Set to reserve FPGA
TaskHash	Hash code of Task which occupys FPGA

Memory Management

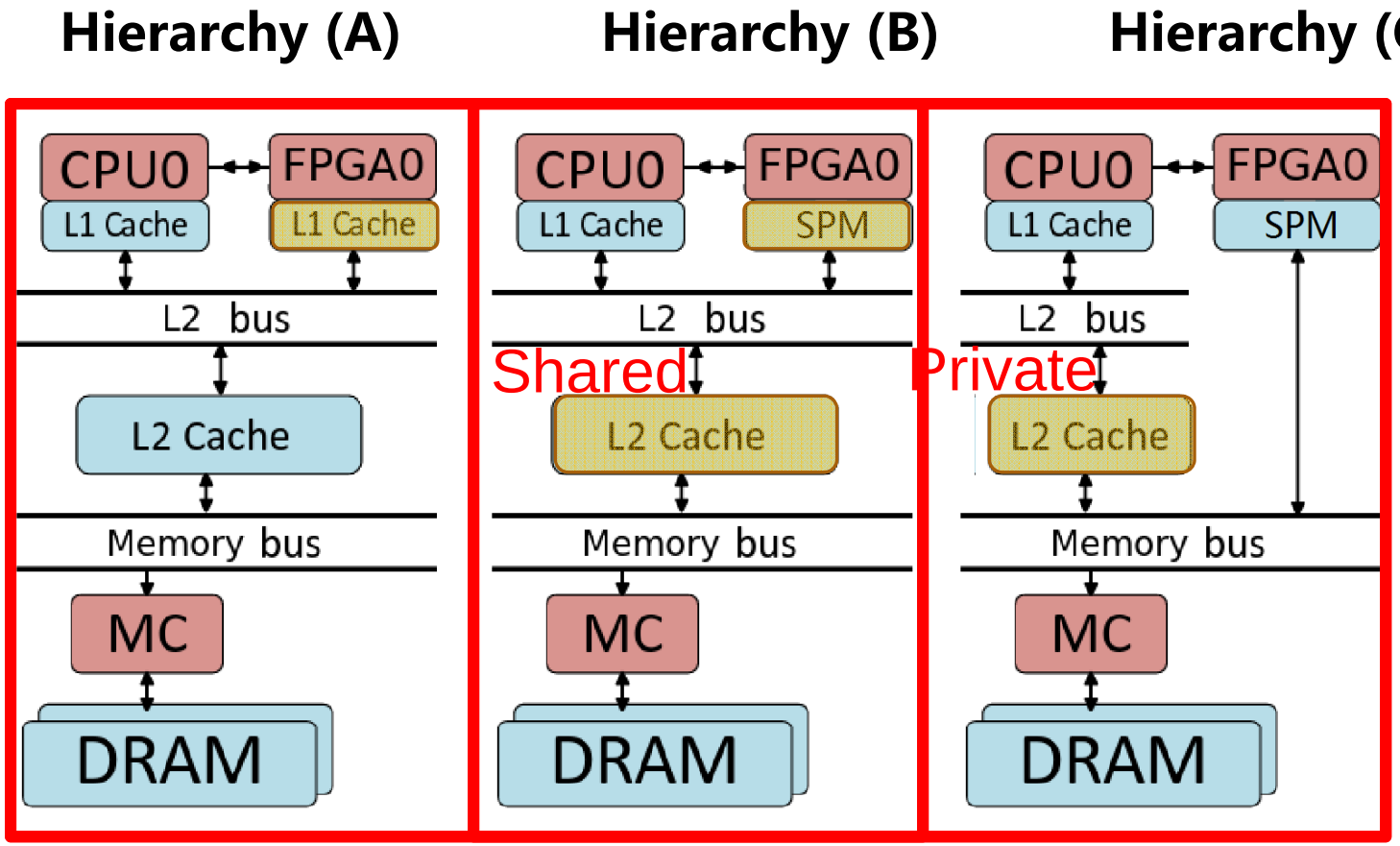
- Data Coherence
  - Support both virtual address and physical address
  - Protocol - MOESI
  - Accelerator Coherent Port
    - ✓FPGA-> L2 bus probe L1 of CPU
  - Shared TLB
    - ✓CPU allocates a space of virtual address for FPGA





Exploration of Memory Hierarchy

- Comparison of three different memory hierarchies



Experimental Setup

- Parameters for Memory Hierarchy

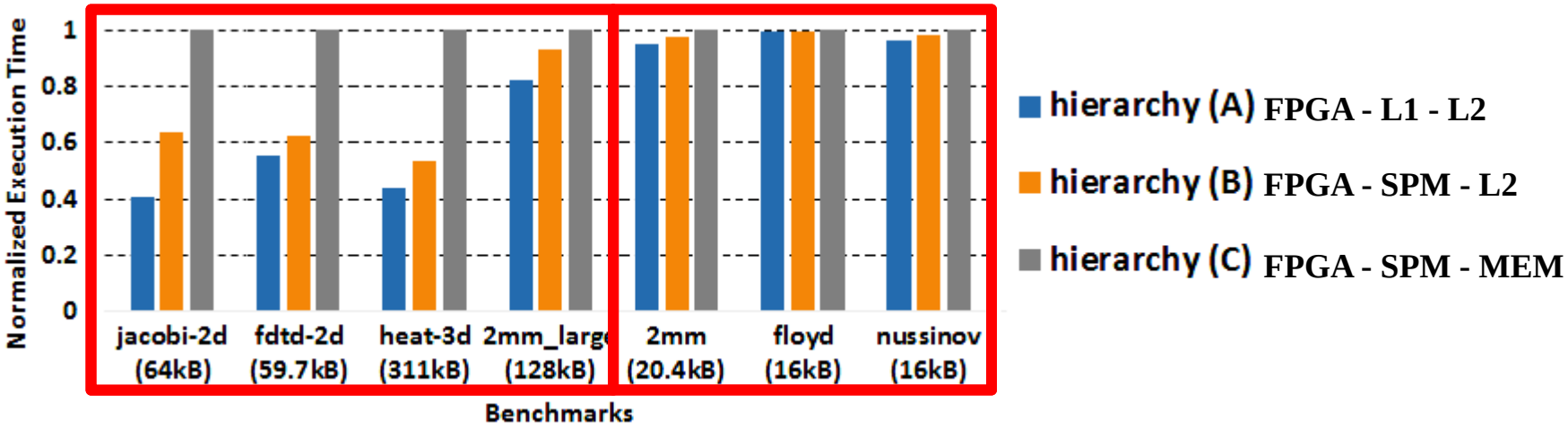
CPU			FPGA
Frequency	Architecture	Execution	Frequency
2.67GHz	X86	Out-of-Order	150MHz

FPGA SPM	32 KB-private, 4 cycles
L1 Cache	32 KB-private, 8-way associate, 4 cycles
L2 Cache	128KB-shared, 8-way associate, 12 cycles
Main Memory	512 MB, DDR3-1600

- Characteristics of PolyBench Benchmarks

Benchmark	2mm	floyd	nussinov	Jacobi-2d	fdtd-2d	heat-3d
# Data (KB)	20.4	16	16	64	59.7	211

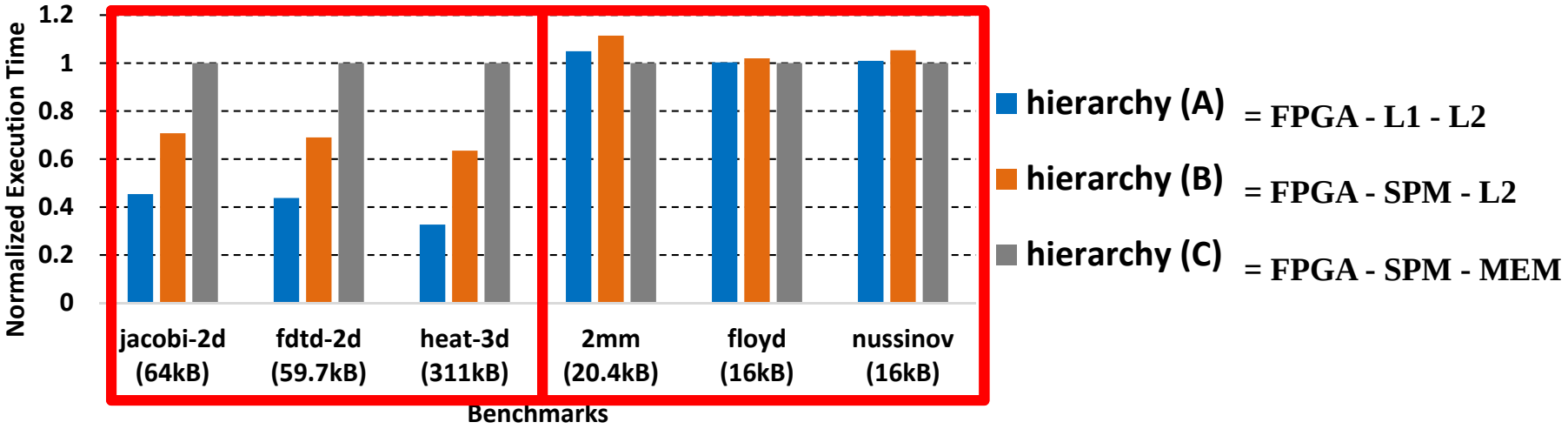
Experimental Results



Data generated by CPU

- L1 Cache vs. SPM
  - Frequent communication between CPU and FPGA
  - If the amount of data > the size of SPM
  - the performance of FPGA with L1 > the performance of FPGA with SPM
  - SPM costs less area and energy than L1 Cache
  - L1 Cache√

Experimental Results



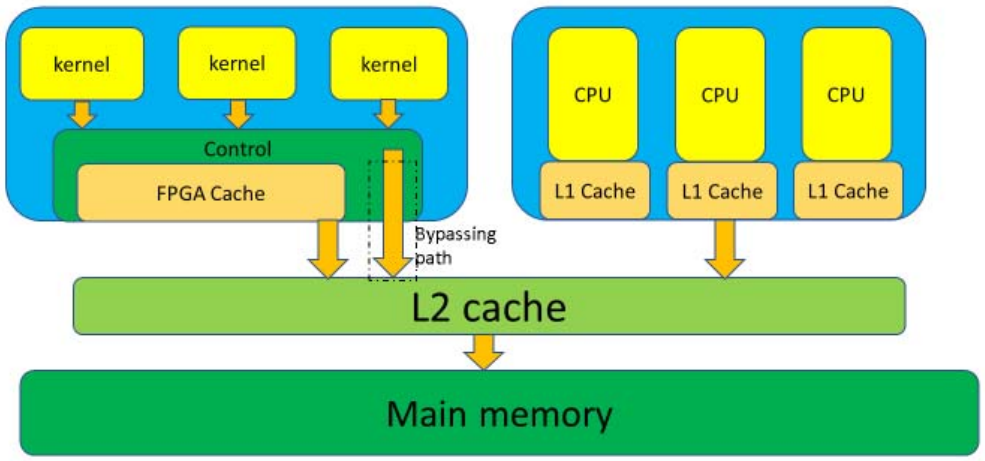
Data are pre-stored in main memory

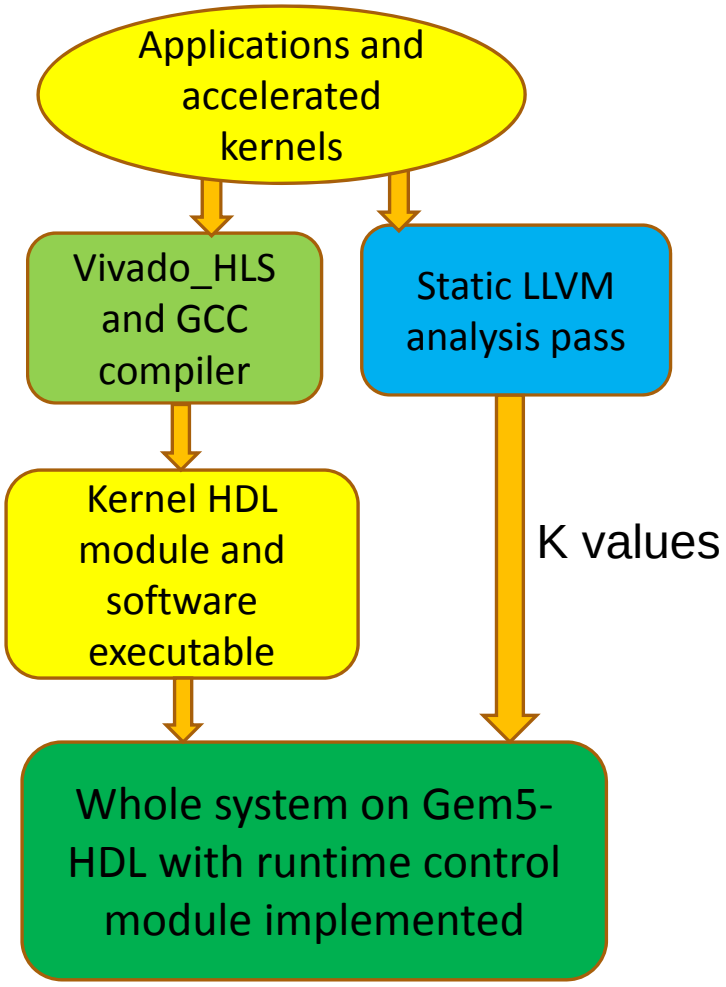
- Cache-coherent vs. DMA
  - If the amount of data > the size of SPM
  - Besides loading/unloading SPM, there are still many FPGA accesses to cache or main memory; Cache helps to reduce the average latency of accesses
  - Hierarchy (A) or (B) √
  - If the amount of data <= the size of SPM
  - Except loading/unloading SPM, there is no FPGA access to cache or main memory; Using DMA can avoid the overhead of caches.
  - Hierarchy (C) √

FPGA L1 Cache Optimization Strategy

- Static analysis of accelerated functions and dynamic cache bypassing control
- Each kernel own one partition of the cache to alleviate the contention
- Dynamic control handles cache bypassing and cache partitioning

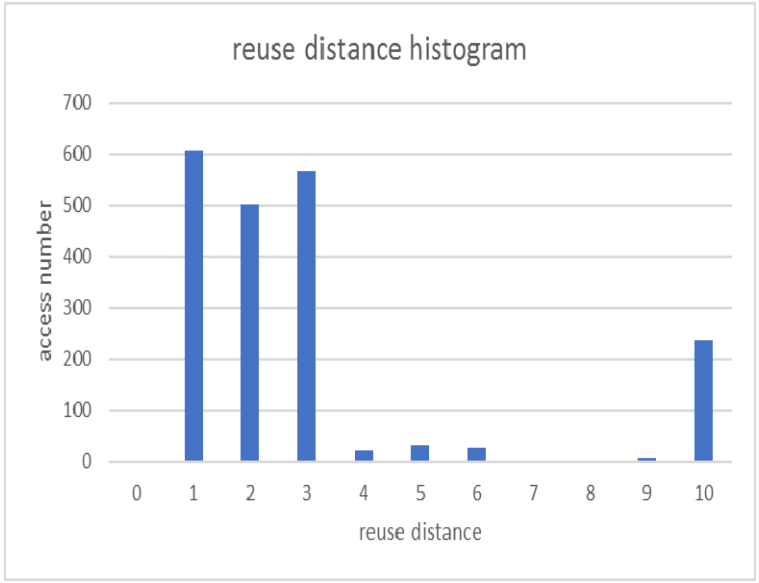
System Architecture





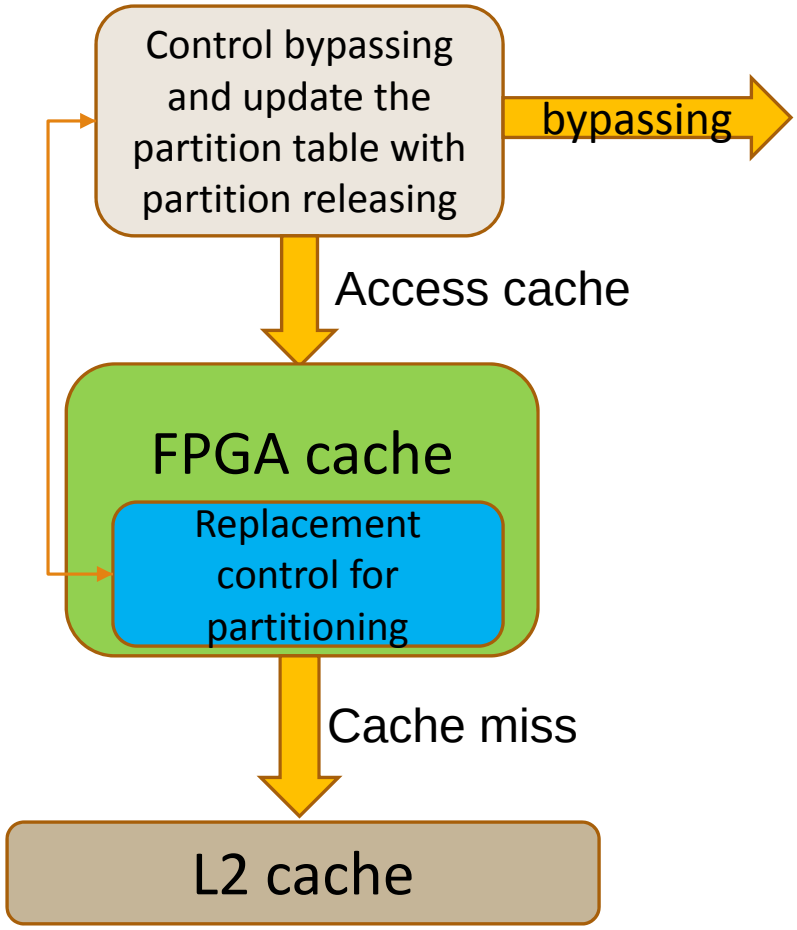
Static Analysis and Dynamic Control

Reuse Distance Analysis

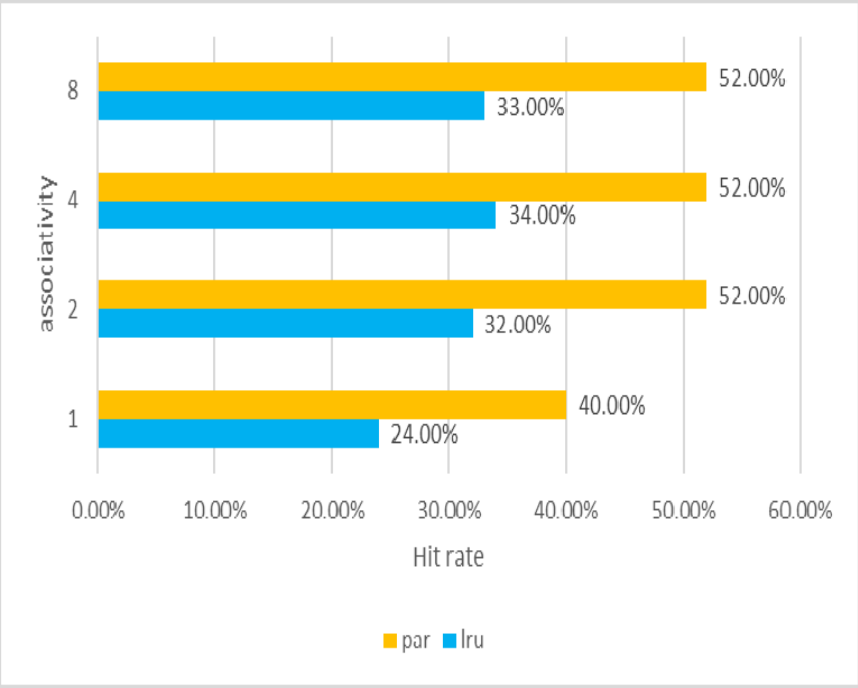


- A set of K values obtained from static analysis to describe the cache utilization information of each kernel
- Cache partition is assigned according to K value

Partition and Bypassing Control



Experiment Results



PolyBench: floyd_warshall + covariance + gesummv
Average improvement: 18.25%

- The improvement on cache hit rate is %10~%20 for the tested benchmarks compared with LRU
- The total performance is improved by 1.5-6.5%.
- Resource overhead
  - ✓ Partition table: 4 associativity and 16KB cache only need 165B additional resource, only 1%
- Latency overhead
  - ✓ No additional delay overhead for partition and bypassing
  - ✓ Negligible compared with large miss latency

Conclusion

- Develop a cycle-accurate full system simulator, gem5-HDL for heterogeneous CPU-accelerator system
  - Flexible run-time control
  - Support various memory hierarchy
  - Enable design space exploration

- Propose a static and dynamic combined cache optimization strategy for optimizing the cache performance in cache coherent CPU-FPGA system
  - Static analysis of data reuse in each kernel
  - Dynamically assign partition and control cache bypassing
  - Significantly improve the hit rate
  - Negligible resource and latency overhead