KU-CSL
Computer Systems Lab

Per-Bank Bandwidth Regulation of Shared Last-Level Cache for Real-Time Systems

Connor Sullivan[§], Alex Manley[§], Mohammad Alian[¶], Heechul Yun[§]

[§]University of Kansas, [¶]Cornell University

KU THE UNIVERSITY OF KANSAS

Cornell University

Memory Level Parallelism

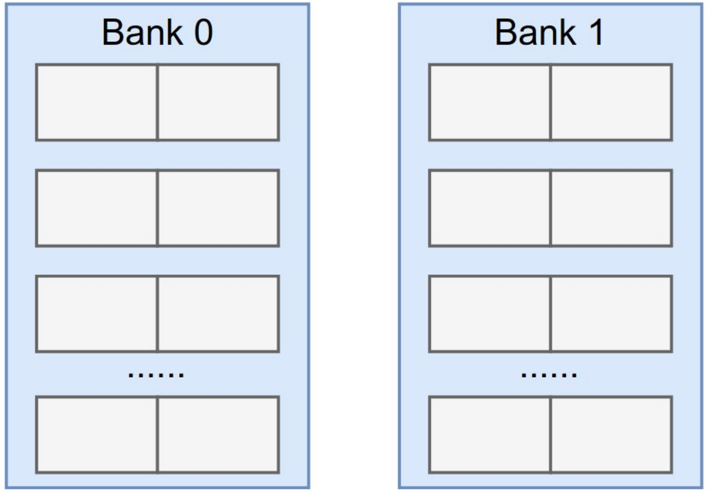
- Essential in modern multi-core processors
- Each core can have multiple memory requests in flight
- A shared last-level cache (LLC) may be composed of multiple independent resources---banks

Multi-banked LLC

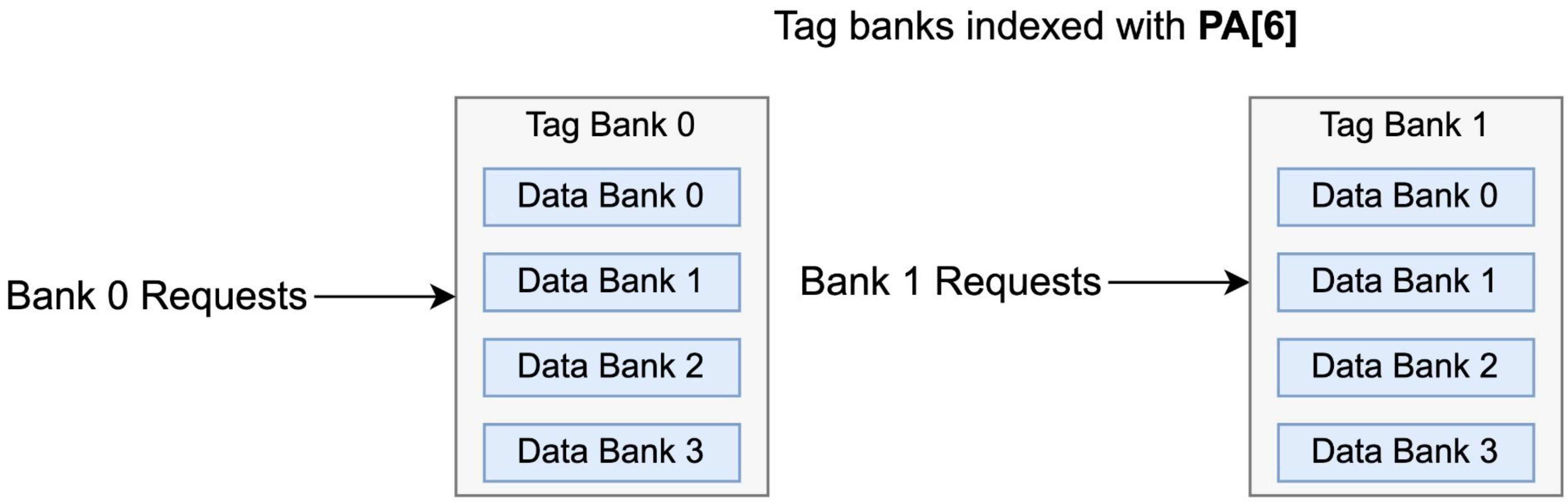
- Each bank is like a mini cache

- Independent of one another

- Separate cache sets

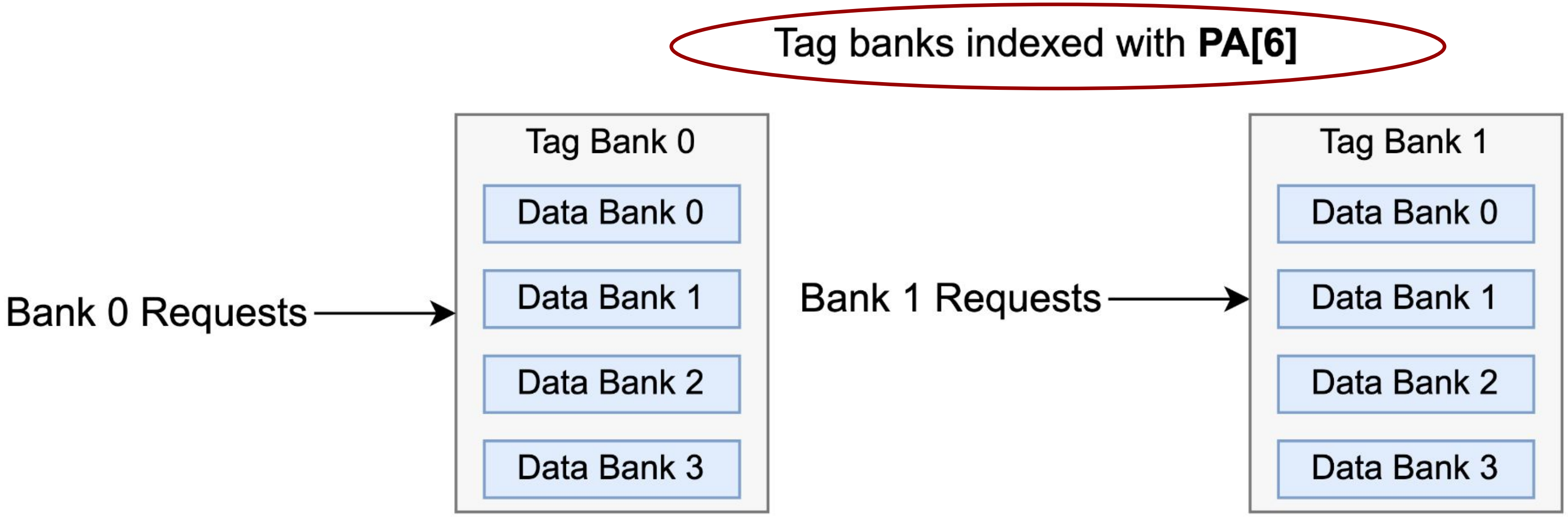


ARM Cortex A72 LLC



https://developer.arm.com/documentation/100095/0003/ – Cortex-A72 TRM

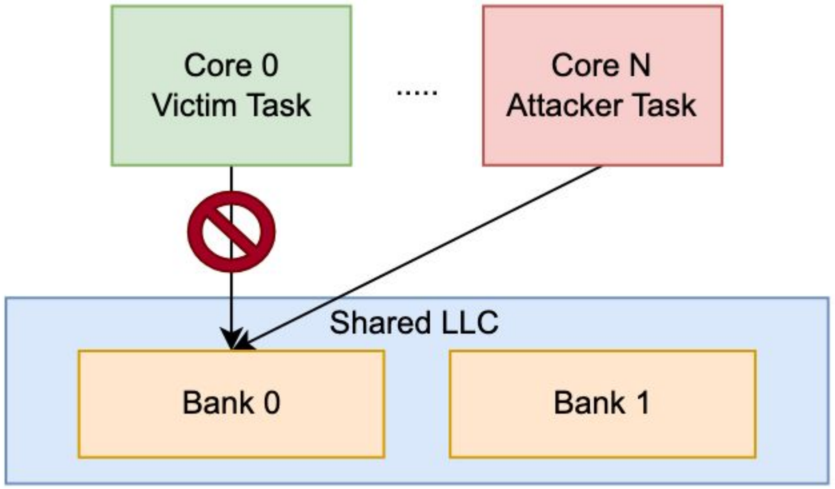
ARM Cortex A72 LLC



Cache Bank Attack

- Attackers use bank mapping knowledge to hammer a bank with requests[1]

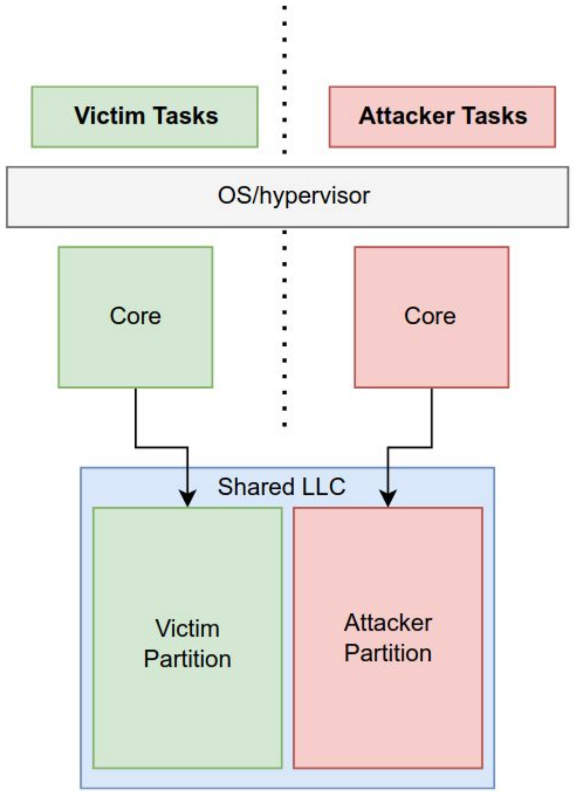
- Create bank contention



[1]M. Bechtel et al. “Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors” RTAS’23

Threat Model[1]

- Attacker best-effort tasks (red) restricted to user space
- Victim -> real-time tasks (green)
- System has a shared multi-bank LLC
- LLC is space partitioned



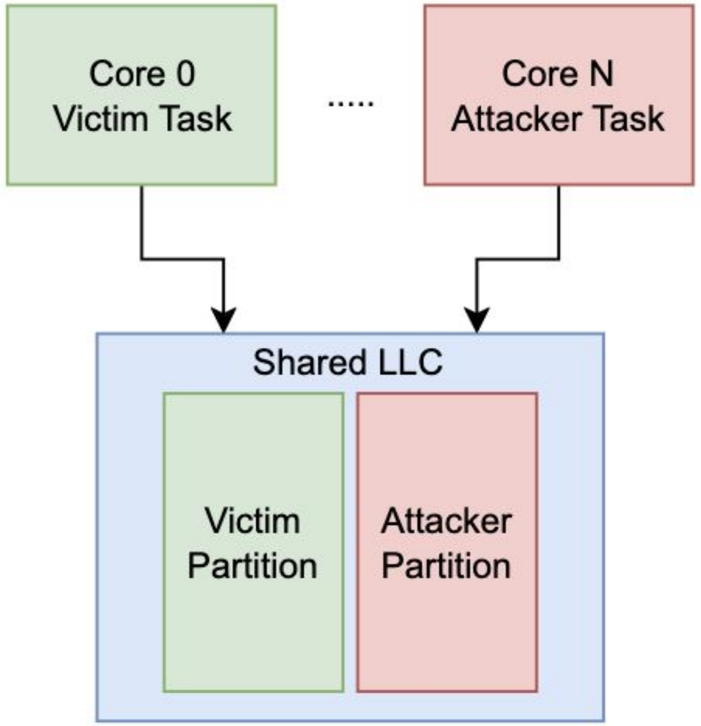
[1]M. Bechtel et al. "Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors" RTAS'23

Cache Space Partitioning

- Give attackers and victim separate partitions in LLC

- Ensures attackers don't evict victim cache lines

- Page coloring (PALLOC[1])

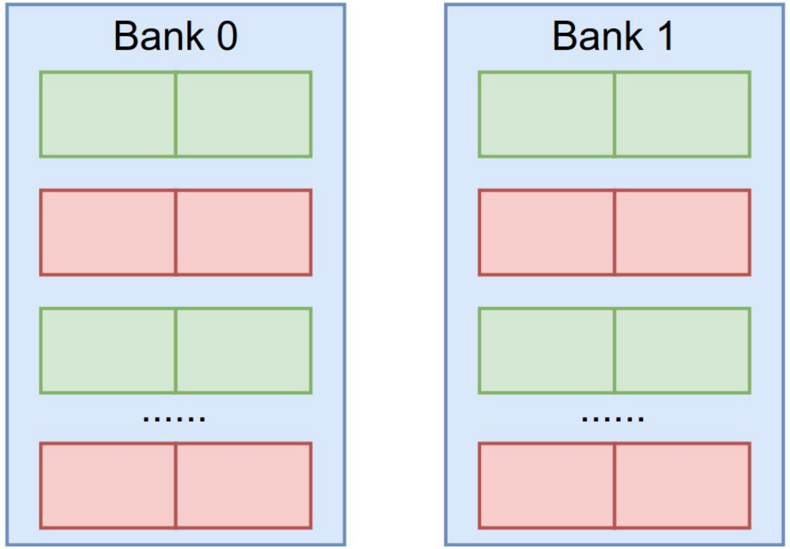


[1]H. Yun et al. PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms" RTAS'14

Cache Space Partitioning

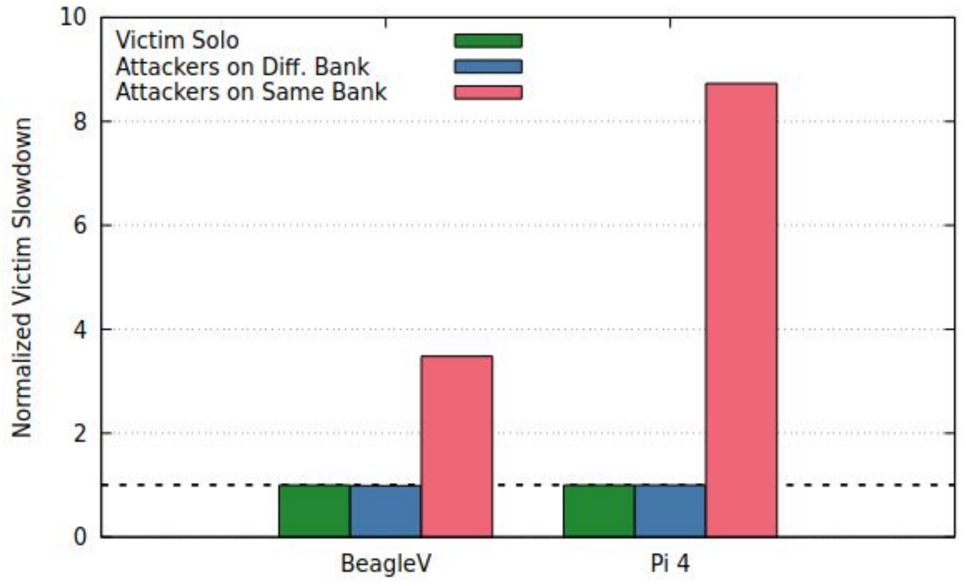
- Banks may still be shared

- Partition the sets via physical address bits



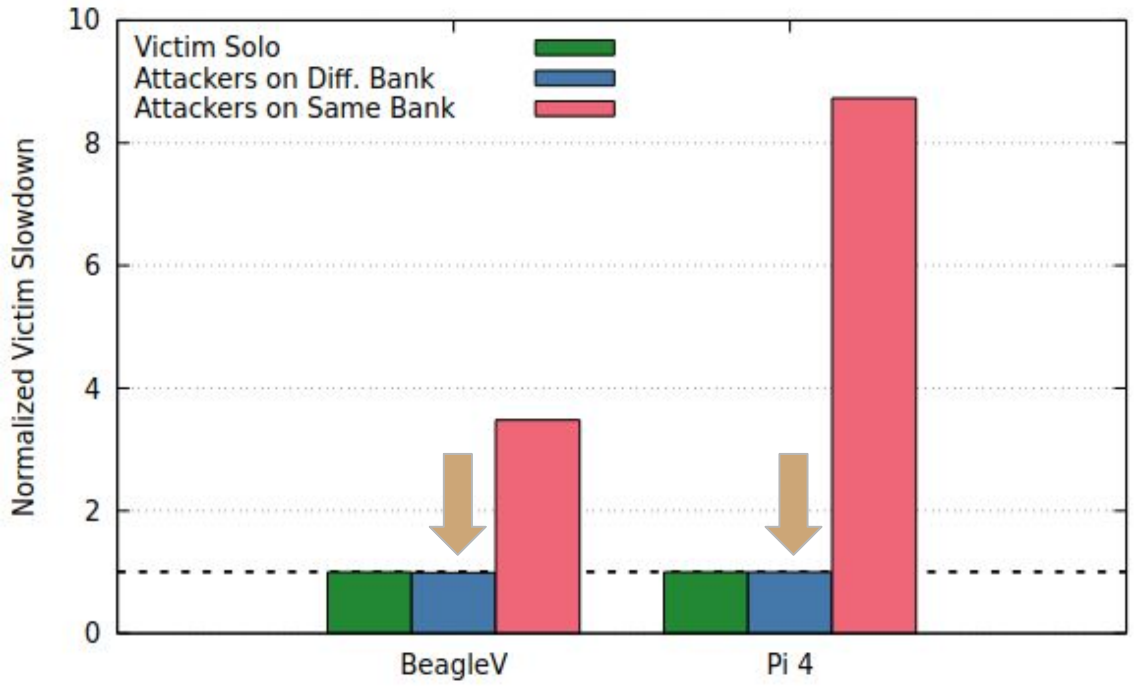
Impact of Cache Bank Attack

- Up to 8.7x cross-core slowdown
- Demonstrated on ARM[1] and RISC-V based embedded multicore SoCs

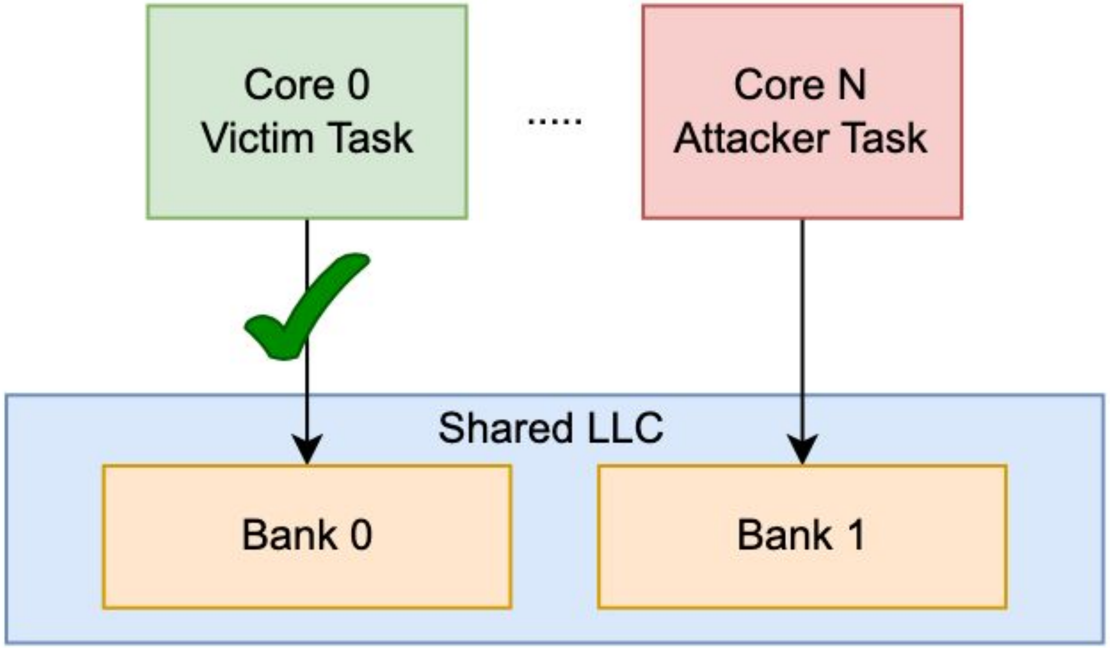


[1]M. Bechtel et al. “Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors” RTAS’23

Where is the Contention?



Pictorial Example



Bandwidth Regulation

- Software-based solutions (Memguard[1])
  - High overhead
  - Up to 300x best-effort (non-real time) task slowdown[2]
- Hardware based solutions
  - Industry: Intel RDT[3], ARM MPAM[4]
  - Research: BRU[5]
  - Low overhead

- All above regulate bandwidth as one resource (bank unaware)....

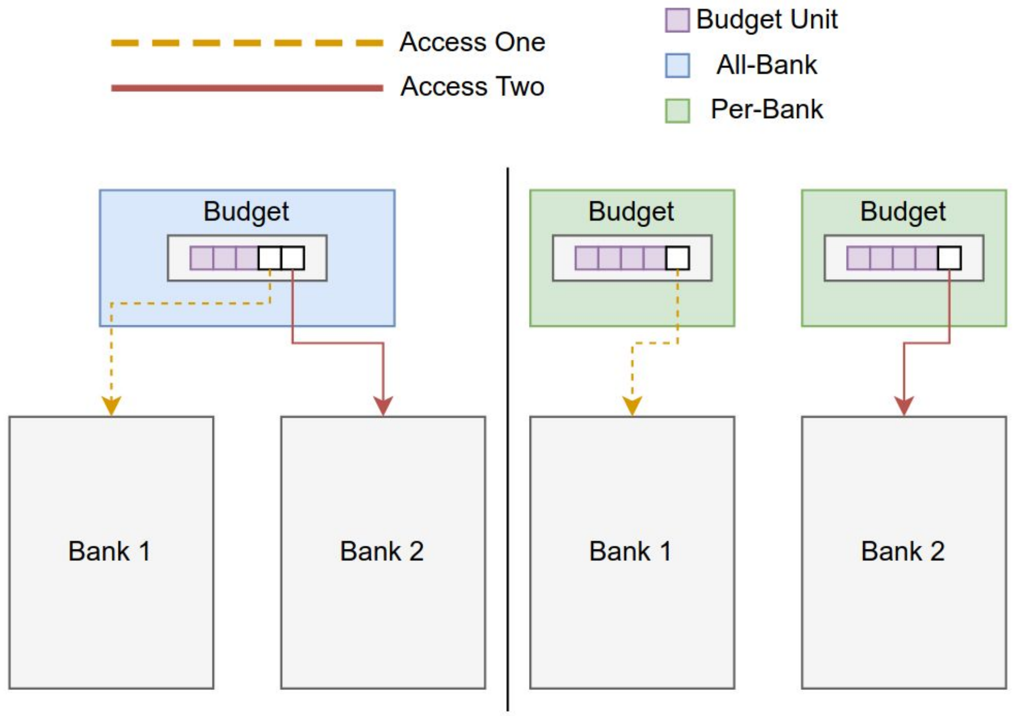
[1]H. Yun et al. "Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms" RTAS'13
[2]M. Bechtel et al. "Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors" RTAS'23
[3]Intel® Resource Director Technology (Intel® RDT) Framework
[4]Arm Memory System Resource Partitioning and Monitoring (MPAM) System Component Specification
[5]F. Farshchi et al. "BRU: Bandwidth Regulation Unit for Real-Time Multicore Processors" RTAS'20

All-bank vs Per-bank Regulation

- All-bank (Bank unaware) regulation
  - Ignores underlying structures

- Per-bank (Bank aware) regulation
  - Takes underlying structures into account

Intuitive Example



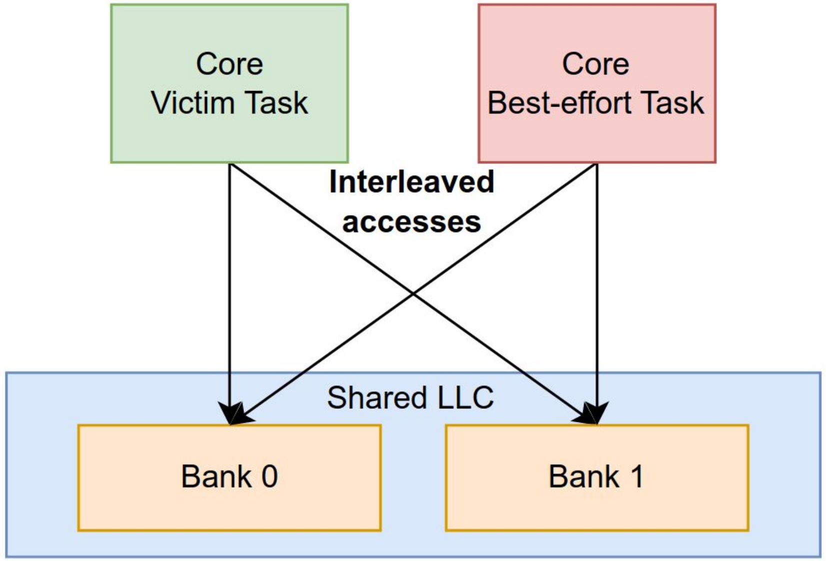
All-bank Regulation Limitations

- We know that contention is at the bank level
- All-bank assumes all accesses are to the same bank
- Bad assumption for best-effort throughput

All-bank Regulation Limitations

- Must regulate to protect victim in worst case scenario

- Consider the throughput impact of regulating a best-effort task



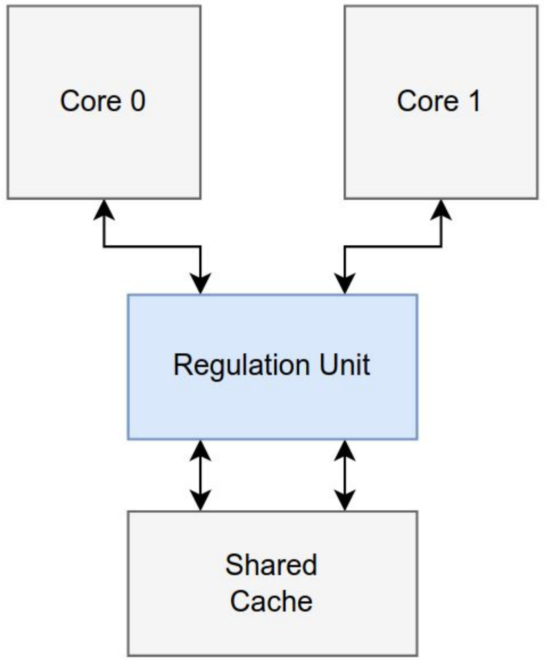
Goals

1. Demonstrate hardware implemented bandwidth regulation as a solution to the cache-bank attack

2. Improve on previous regulation implementations through more fine grained (per-bank) regulation

Design Overview

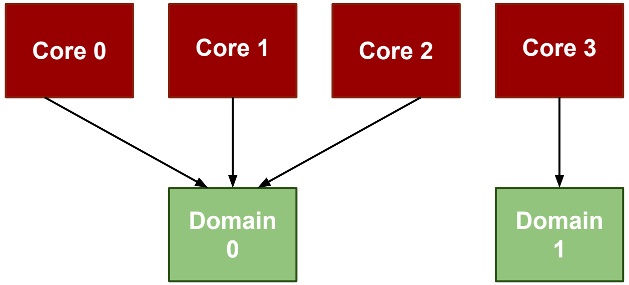
- Use drop-in BRU[1] as baseline
- Sits between cores and shared memory
- No modifications to the shared cache



[1]F. Farshchi et al. “BRU: Bandwidth Regulation Unit for Real-Time Multicore Processors” RTAS’20

Design Overview

- Enables grouping of cores into arbitrary domains[1]
- Bandwidth regulation done over a fixed period with a fixed number of accesses per period
- BW "budget" is given to each bank



$$BW = \frac{MaxAccesses}{Period} \cdot TS \cdot f_{clk}$$

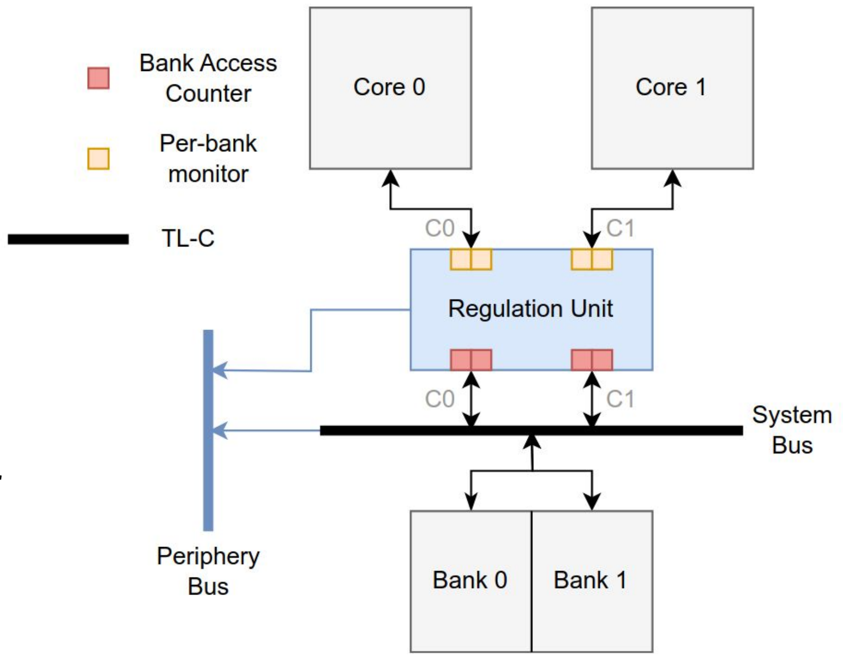
$TS : Transaction\ Size$

$f_{clk} : Clock\ frequency$

[1]F. Farshchi et al. "BRU: Bandwidth Regulation Unit for Real-Time Multicore Processors" RTAS'20

Implementation

- Integrate with Rocket Chip SoC[1]
- Leverage TileLink interconnect channels to regulate bandwidth
- Regulator is a Chisel[2] generator enabling support for any number of banks



[1]K. Asanovic et al. “The Rocket Chip Generator” UC Berkeley Tech. Rep. 2016
[2]J. Bachrach et. al “Chisel: Constructing hardware in a Scala embedded language” DAC’12

Evaluation Platform

- Use FireSim[1] for simulation
  - Synthesizable RTL
  - Simulates at ~100MHz, cycle accurate
  - Run locally on Xilinx UltraScale+ VCU118 FPGA

https://www.xilinx.com/products/boards-and-kits/vcu118.html

[1]S. Karandikar et. al “FireSim: FPGA-accelerated Cycle-exact Scale-out System Simulation in the Public Cloud” ISCA’18

Simulated SoC

- Bank attack requires out-of-order cores -> BOOM[1]
- Can't fit four Large BOOM cores on our FPGA platform without optimizations
  - Also BOOM has a bug...(https://github.com/riscv-boom/riscv-boom/issues/690)
- Use in-order Rocket[2] cores "enhanced" with Mempress[3] (on chip accelerator)
  - Traffic generator allowing parallel access to shared memory

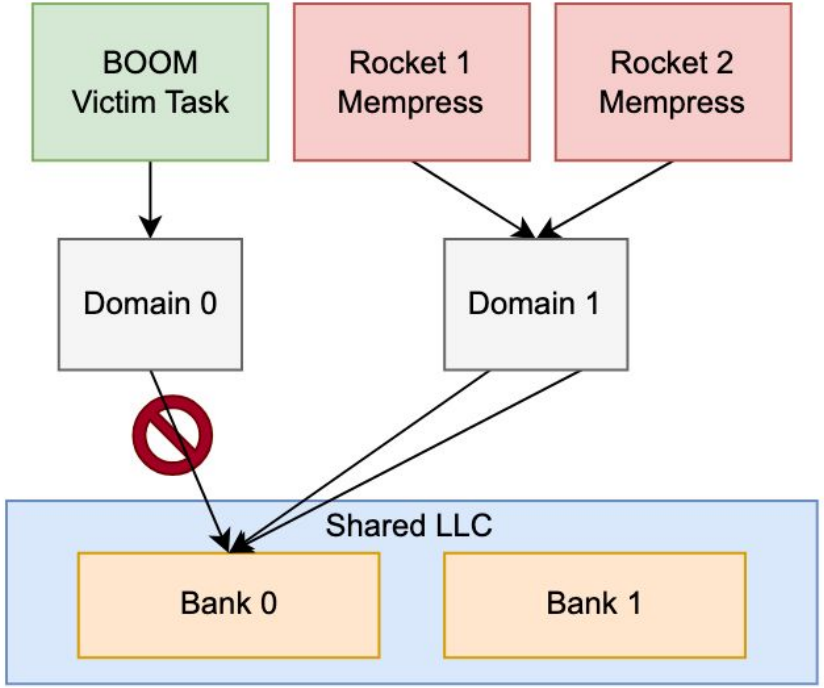
Cores	1xLargeBoom, 1GHz, out-of-order, 3-wide, ROB: 96, LSQ: 24/24 2xRocket, 1GHz, in-order, enhanced with Mempress
BOOM Private L1 Cache	32KB(I) - 32KB(D), 8-way
Shared L2 Cache (LLC)	1MB (16-way)
Memory	4GB DDR3, FR-FCFS

[1]C. Celio et al. "The Berkeley Out-of-Order Machine (BOOM)" UC Berkeley Tech. Rep. 2015
[2]K. Asanovic et al. "The Rocket Chip Generator" UC Berkeley Tech. Rep. 2016
[3]https://github.com/ucb-bar/mempress/tree/main

Attack Setup

- Attackers are the two Mempress units

- BOOM core runs victim (real-time) task

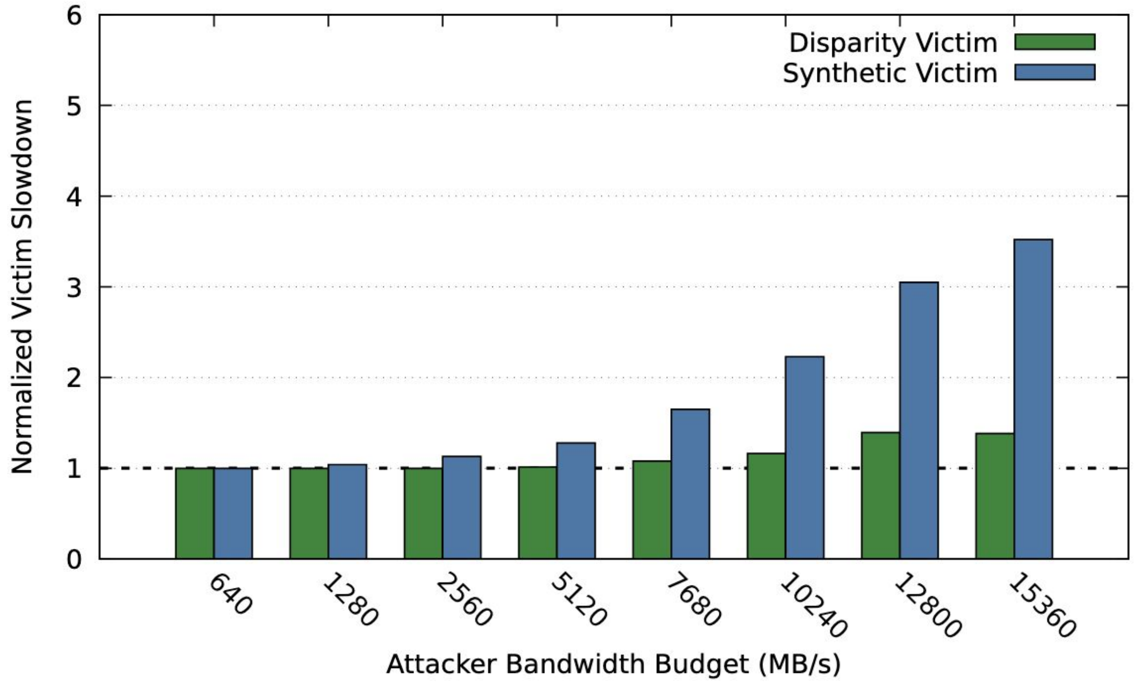


Exp. 1: Isolation Impact on RT Tasks

- Synthetic victim is BankPLL[1] workload run on BOOM core
  - Target specific bank
- Real-world victim is Disparity from SD-VBS[2]
  - We measured Disparity to have highest LLC bandwidth of all SD-VBS workloads
- Vary attacker bandwidth budget
  - Examine change in victim slowdown

[1]M. Bechtel et al. “Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors” RTAS’23
[2]S. K. Venkata et al. "SD-VBS: The san diego vision benchmark suite" IISWC'09

Isolation Impact of Regulation



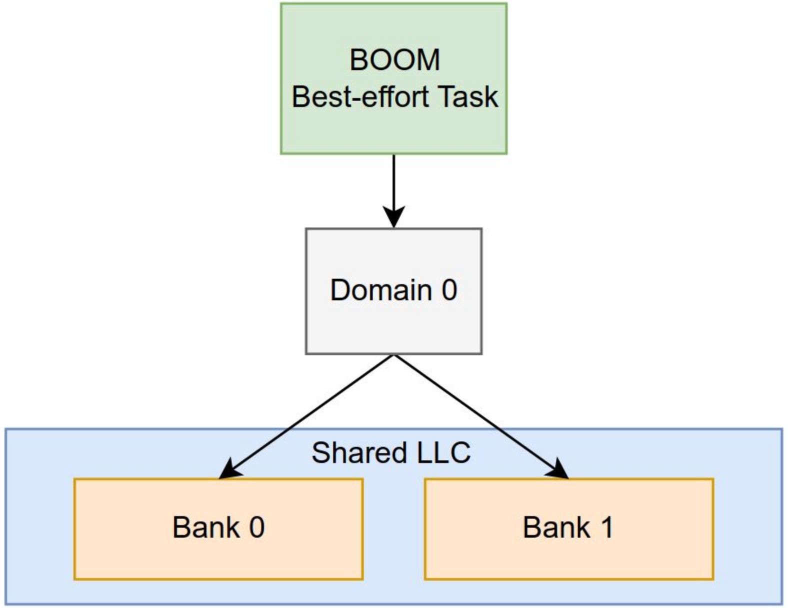
- Results hold for all-bank (BRU) and per-bank (ours) regulation
- At 1.28GB/s budget, victim slowdown is ~1.03x

Exp. 2: Throughput Impact on BE Tasks

- Examine the throughput impact of regulation on the benign best-effort tasks

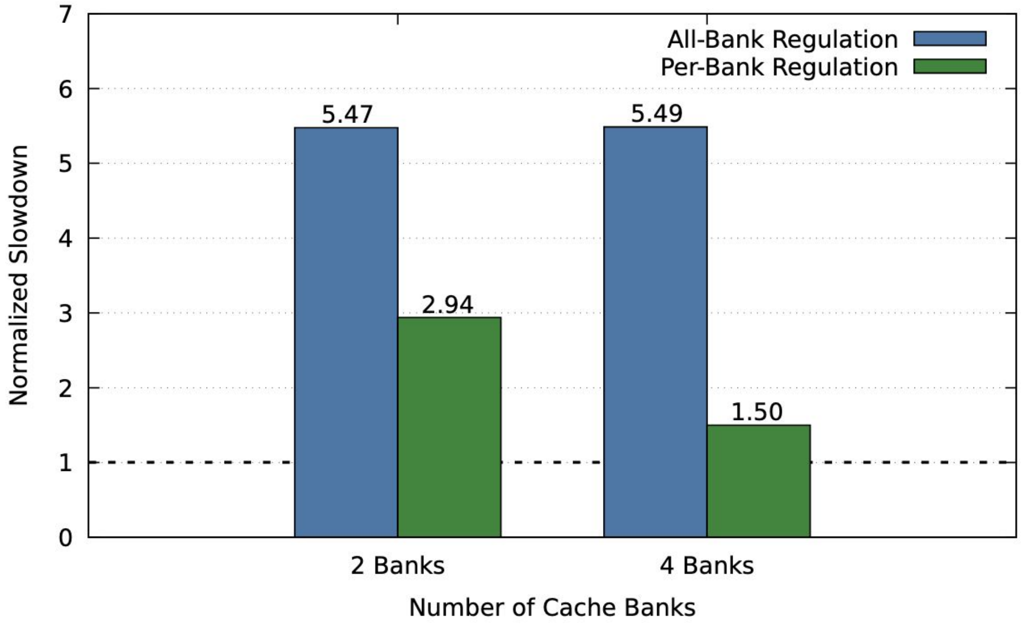
- Use *bandwidth* from IsolBench[1]

- Regulate under both per-bank and all-bank at 1.28GB/s



[1]https://github.com/CSL-KU/IsolBench

Throughput Impact of Regulation



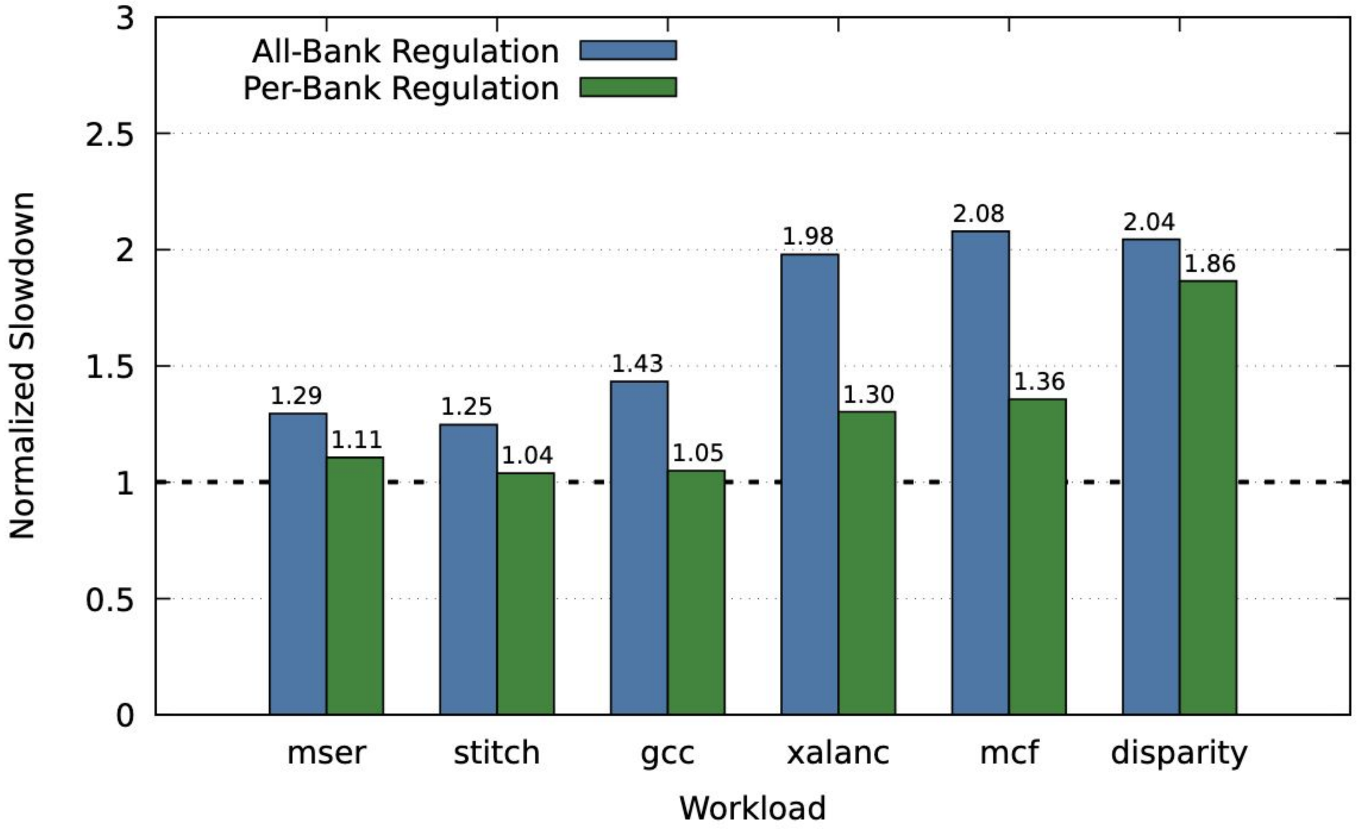
- Two-bank case sees a 1.86x improvement when using per-bank
- Four-bank case sees 3.66x

Exp. 3: Impact on Real-world Apps.

- Demonstrate throughput improvement for real-world workloads
- Select SD-VBS and SPEC2017 workloads as best-effort tasks
- Pin workload to BOOM core and regulate at 1.28GB/s
- Run with all-bank and per-bank, with two and four-bank LLCs

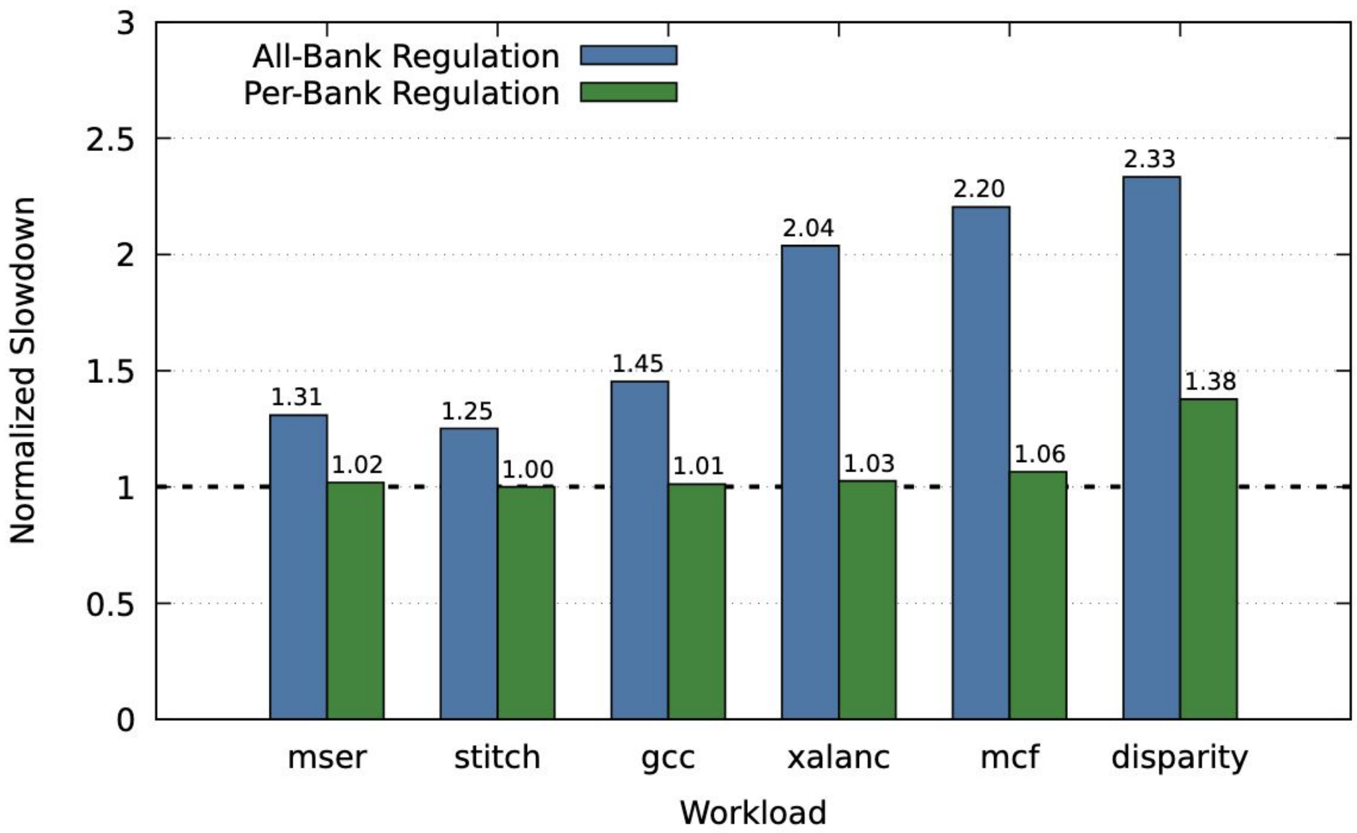
Two-bank LLC

1.31x average improvement



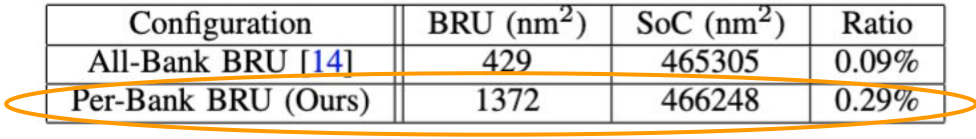
Four-bank LLC

1.61x average improvement



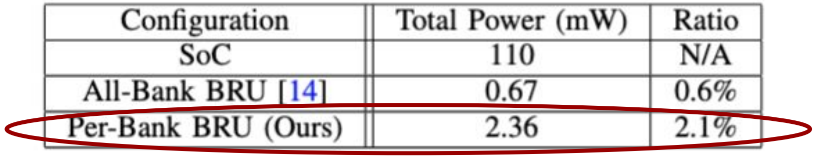
Area and Power Overhead

- Synthesis and place and route for ASAP7 7nm[1]
- Minimal area-overhead, < 0.3%
- Minimal power-overhead, 2.1%



Configuration	BRU (nm^2)	SoC (nm^2)	Ratio
All-Bank BRU [14]	429	465305	0.09%
Per-Bank BRU (Ours)	1372	466248	0.29%

TABLE V: Comparative Area Analysis



Configuration	Total Power (mW)	Ratio
SoC	110	N/A
All-Bank BRU [14]	0.67	0.6%
Per-Bank BRU (Ours)	2.36	2.1%

TABLE VI: Comparative Power Analysis

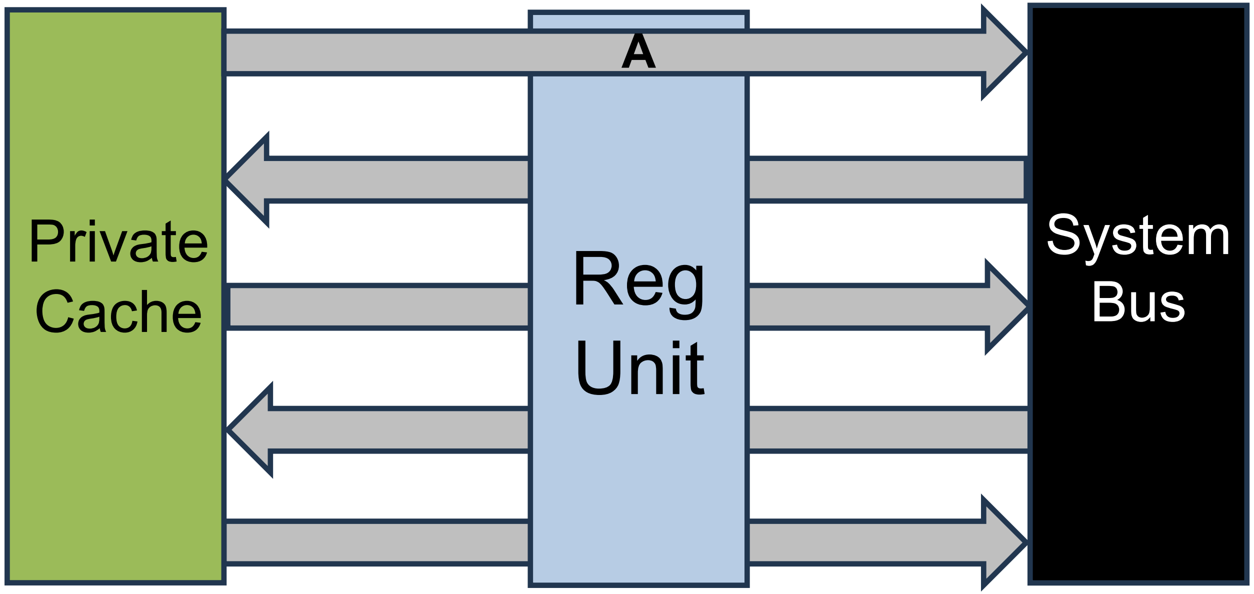
[1]L. T. Clark et. al "ASAP7: A 7-nm finFET predictive process design kit" Microelectronics Journal'16

Conclusion

- Multi-banked LLC in modern multicores may be vulnerable to cache bank contention attacks
- All-bank (bank unaware) regulation is highly inefficient to defend against cache bank contention
- Per-bank regulation provides higher (up to 3.66x on 4-bank LLC) throughput over all-bank regulation while providing the same isolation guarantees

Regulation with TileLink

- Access (read) to LLC occurs on Channel A
- Count these reads
- Extract destination address to examine target bank


A
Private Cache
Reg Unit
System Bus