Dynamic Verification of Multicore Architectures with Untrusted Memory Operations.

Shaan Yadav[1], Andrew D. Hilton[1]
[1]Pratt School of Engineering, Duke University, NC, USA
shaan.yadav@duke.edu

Duke UNIVERSITY | ELECTRICAL & COMPUTER ENGINEERING

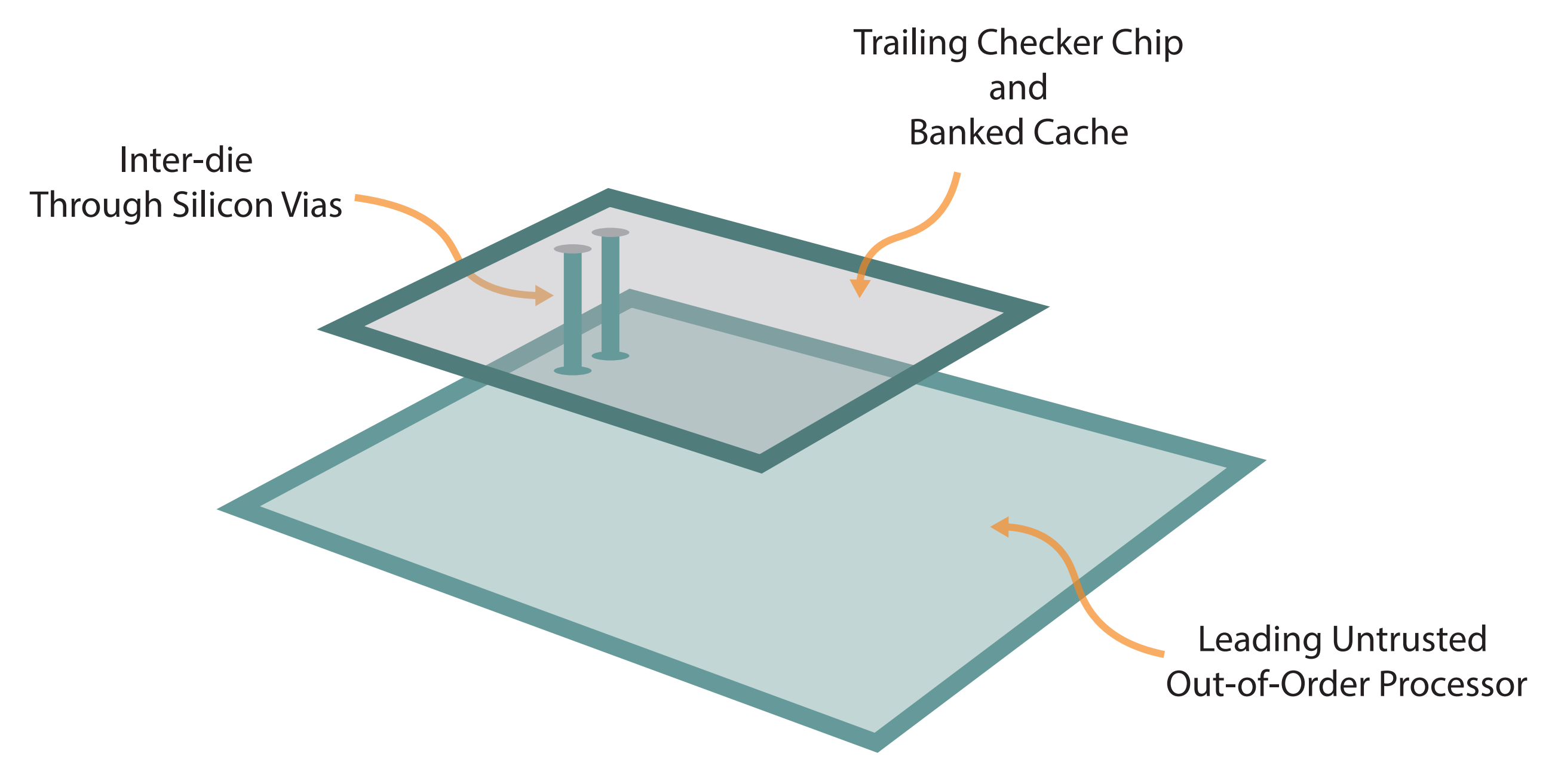
Abstract

Advancements in processor architectures and hardware technologies have led to improved processor performance but have also introduced increased vulnerability to reliability and security issues. Reduction in transistor sizes have made systems more susceptible to transient faults, and the increased complexity of modern multi-core out-of-order (OOO) processors have created more opportunities for security breaches.

One potential solution to mitigate errors is the use of redundancy through an in-order checker processor. In this setup, a "trailing" checker verifies the entire lifecycle of instructions executed by a "leading" untrusted processor to ensure correct execution. Several fault-tolerant models have been proposed before, however, these architectures trust load values from the untrusted processor, leading to potential vulnerabilities.

This research proposes a novel architecture framework that verifies instructions, including load instructions. We 3D stack the checker die on top of the untrusted processor's die for minimal invasiveness during manufacturing as well as power savings.

This new architecture opens several new use cases such as better prevention of fabrication attacks and even allowing for more aggressive microarchitectural designs which can run behind the safety net of this work.


Trailing Checker Chip and Banked Cache
Inter-die Through Silicon Vias
Leading Untrusted Out-of-Order Processor

Physical model of leading out-of-order processor and trailing checker processor with through silicon vias

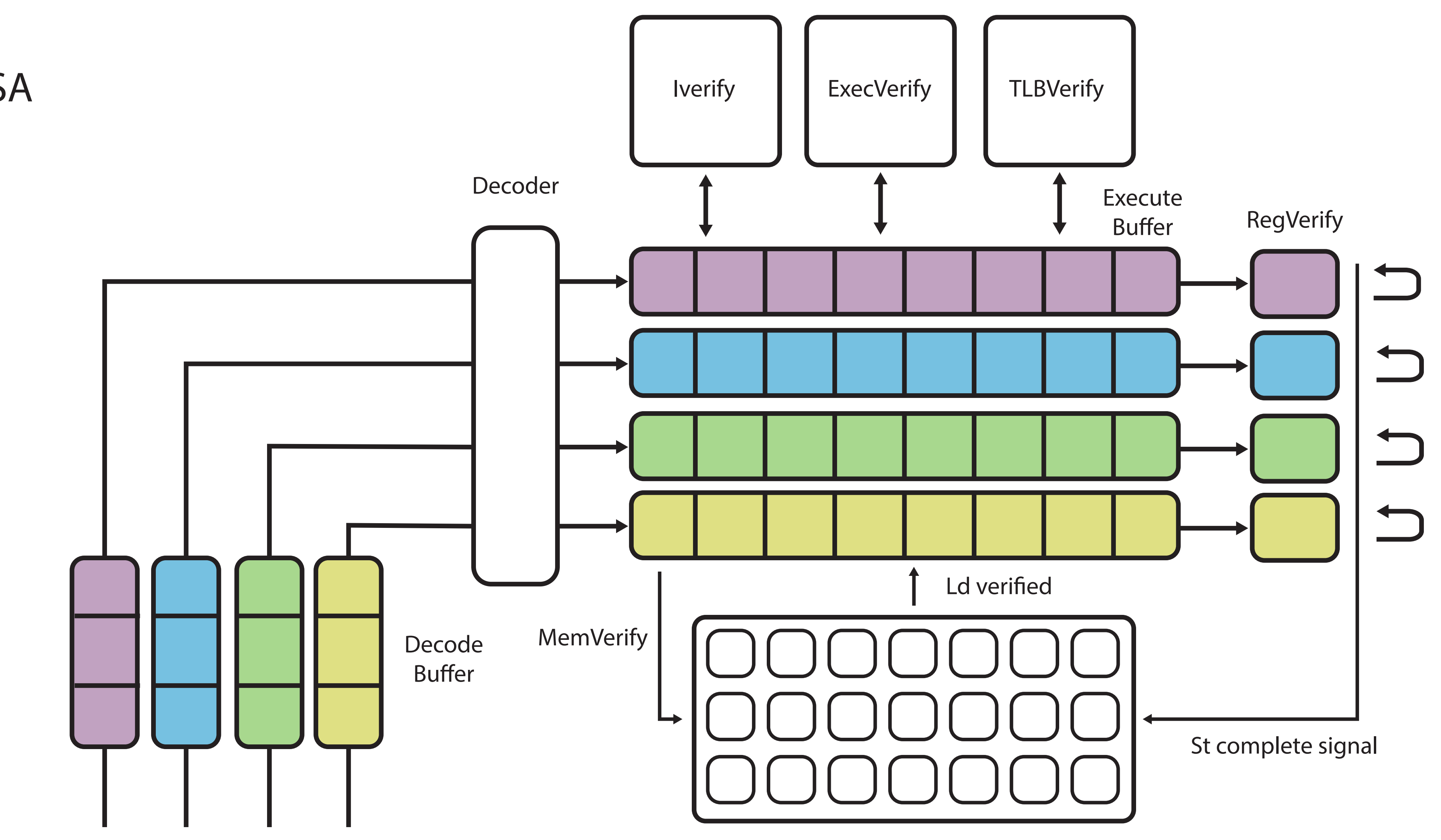


Diagram of Checker Chip Pipeline each color represents instruction

Checker System

- Operates at 1/4 or 1/2 of leading clock frequency
- Per core verification pipeline, verifying each stage of an instruction's execution
- Exploits parallelism of verification to keep up with leading chip's throughput

IVerify

- Inputs: Program Counter, Instruction Word
- Outputs: IVerify bit

  Ensures the correct instruction was fetched for the associated PC value by coalescing requests and verification via the banked cache

ExecVerify

- Inputs: Opcode, Operand, Operation Result
- Outputs: ExecVerify bit

  Confirms that the value resulting from the instruction execution was correct by utilizing a pool of functional units (FUs)

TLBVerify

- Inputs: Virtual Address, Physical Address
- Outputs: TLBVerify bit

  Validates the address translation by a per-checker core ITLB and DTLB

RegVerify

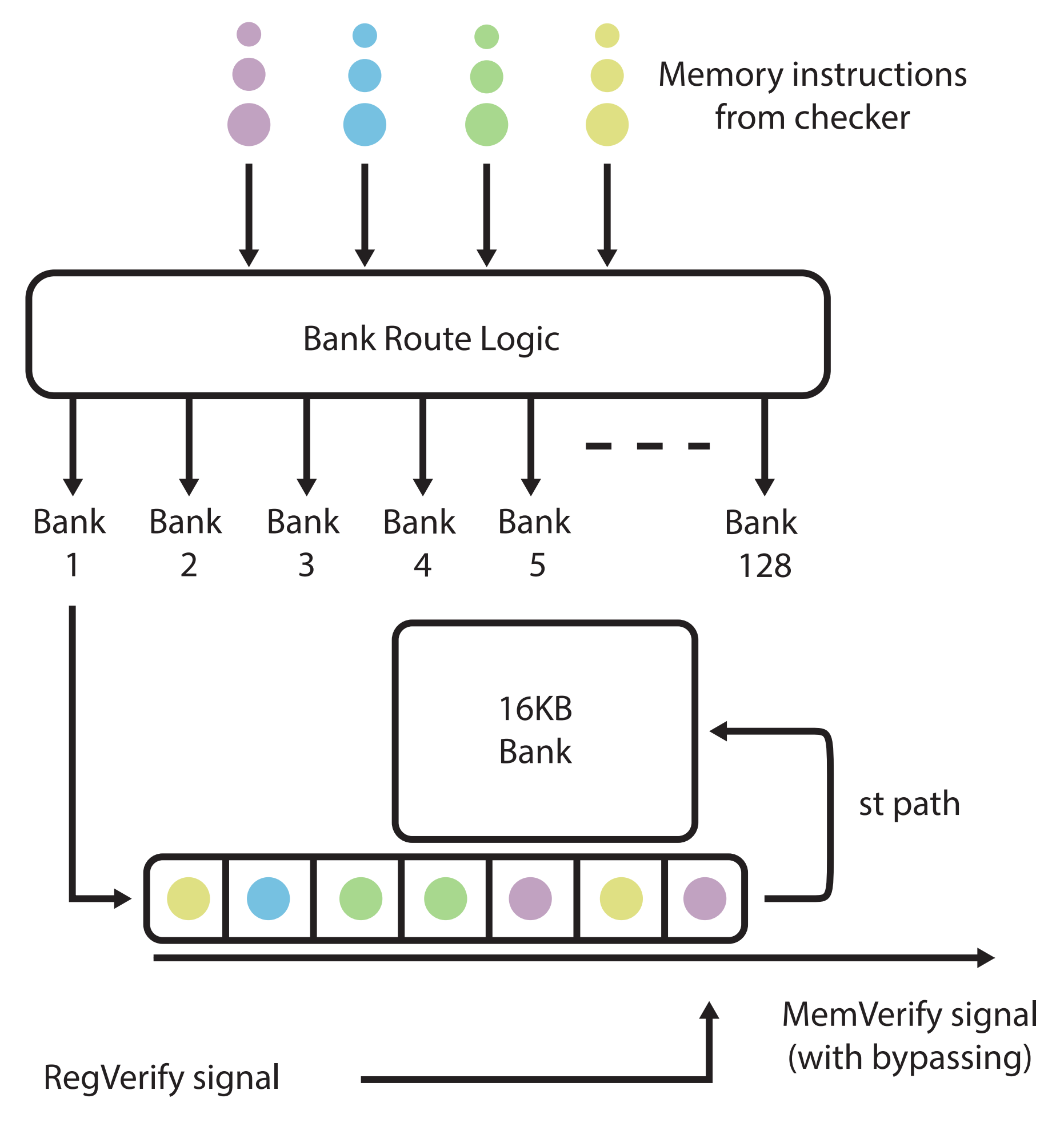
- Inputs: Register Values, Verification bits
- Outputs: Store Complete Signal

  Certifies values used for registers were correct by reading out checker's regfile and updates with new register values

MemVerify

- Inputs: Physical Address, Ld/St data
- Outputs: MemVerify bit

  Verifies value loaded from memory through the checker's banked cache and performs store complete upon instruction validation

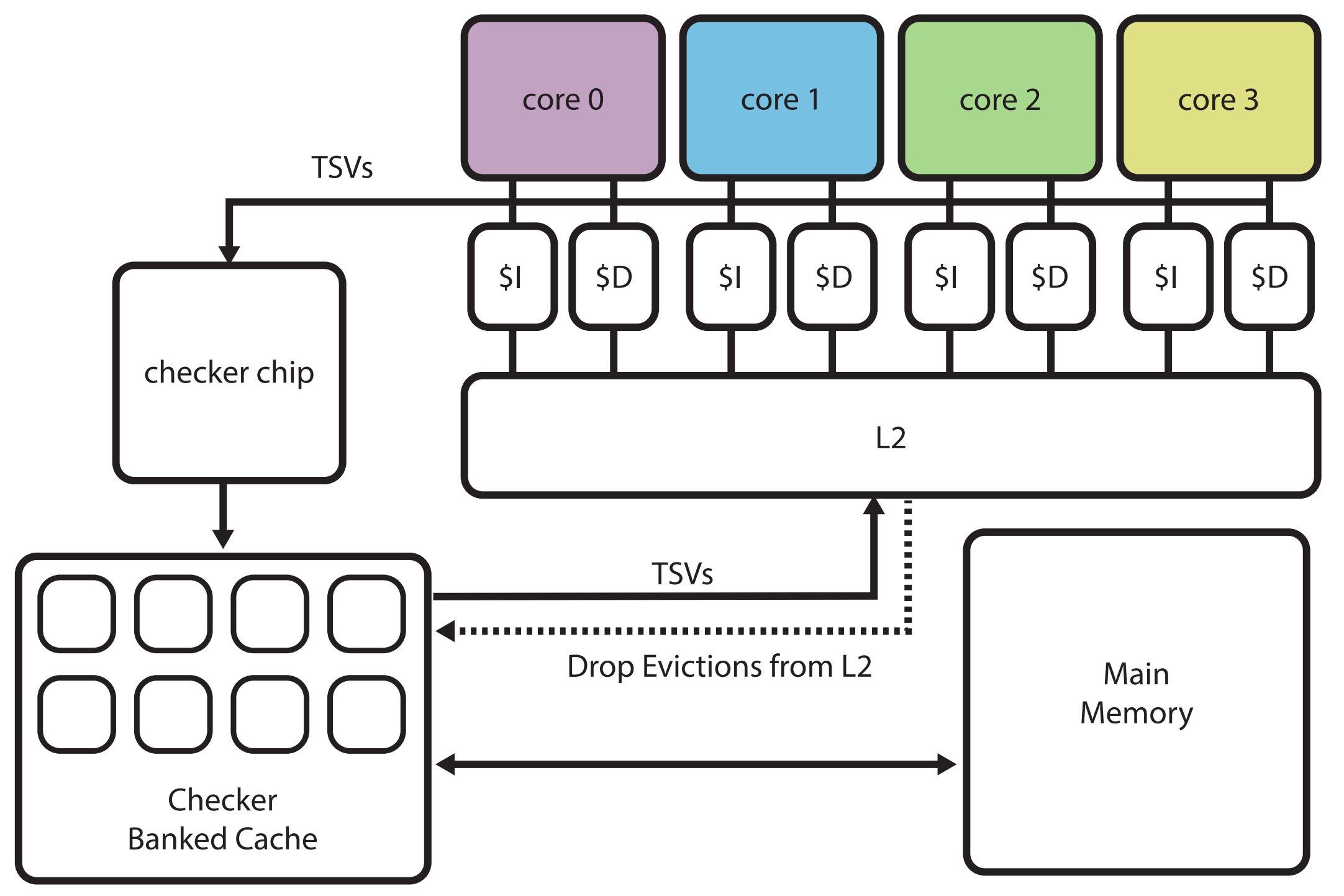


Banked Cache

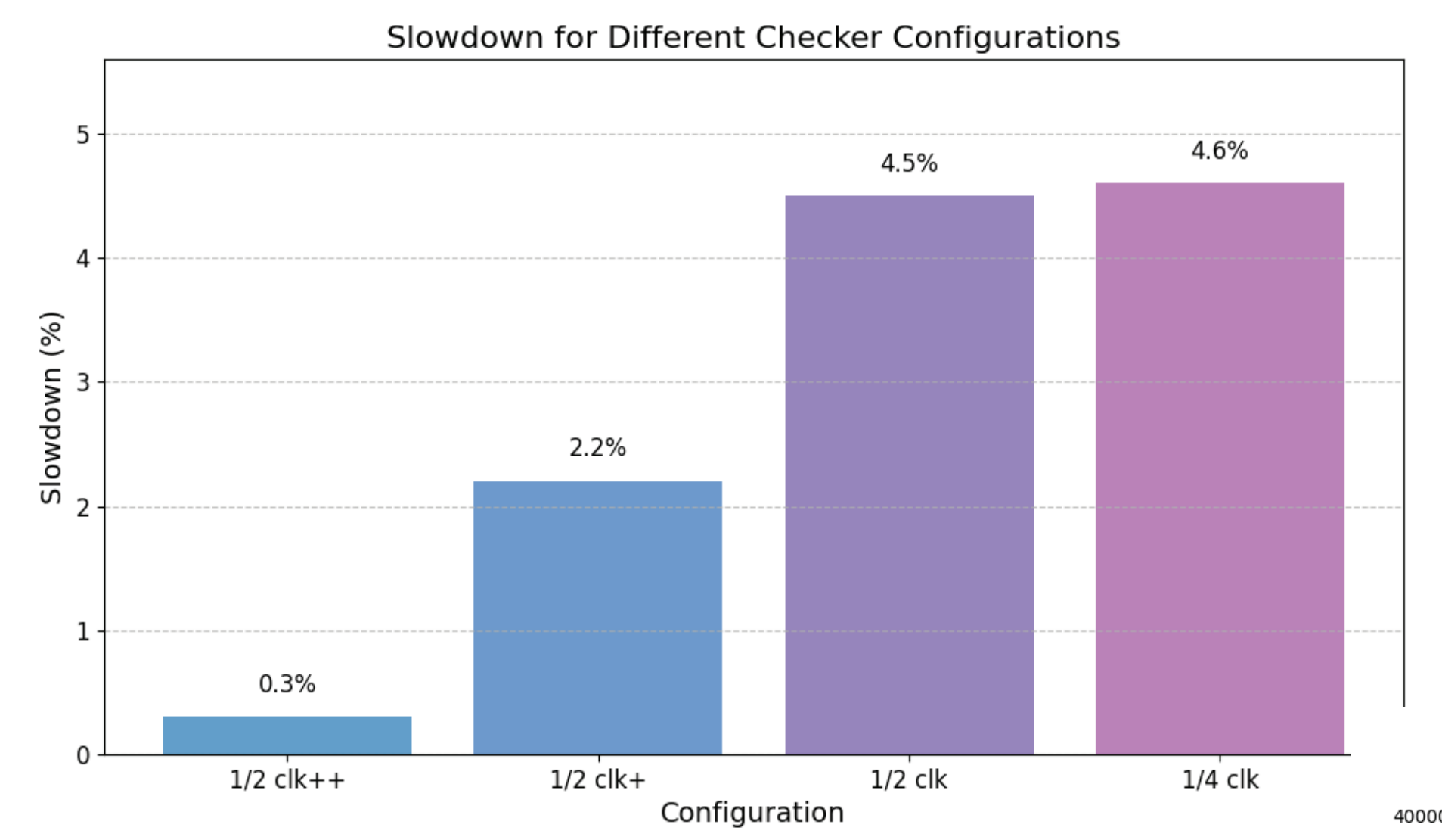
- 2MB, 128 Banks (16KB each)
- Requests from checker get routed to appropriate bank queue
- Bank queue holds ld/st instruction
- Ld insts verify contents with bank
- St insts complete once RegVerify has occurred

Checker Integration

- Instructions transported to checker at commit phase
- Evictions from L2 Cache dropped
- All completions to main memory happen through checker



Preliminary Results



Slowdown of system from different checker configurations

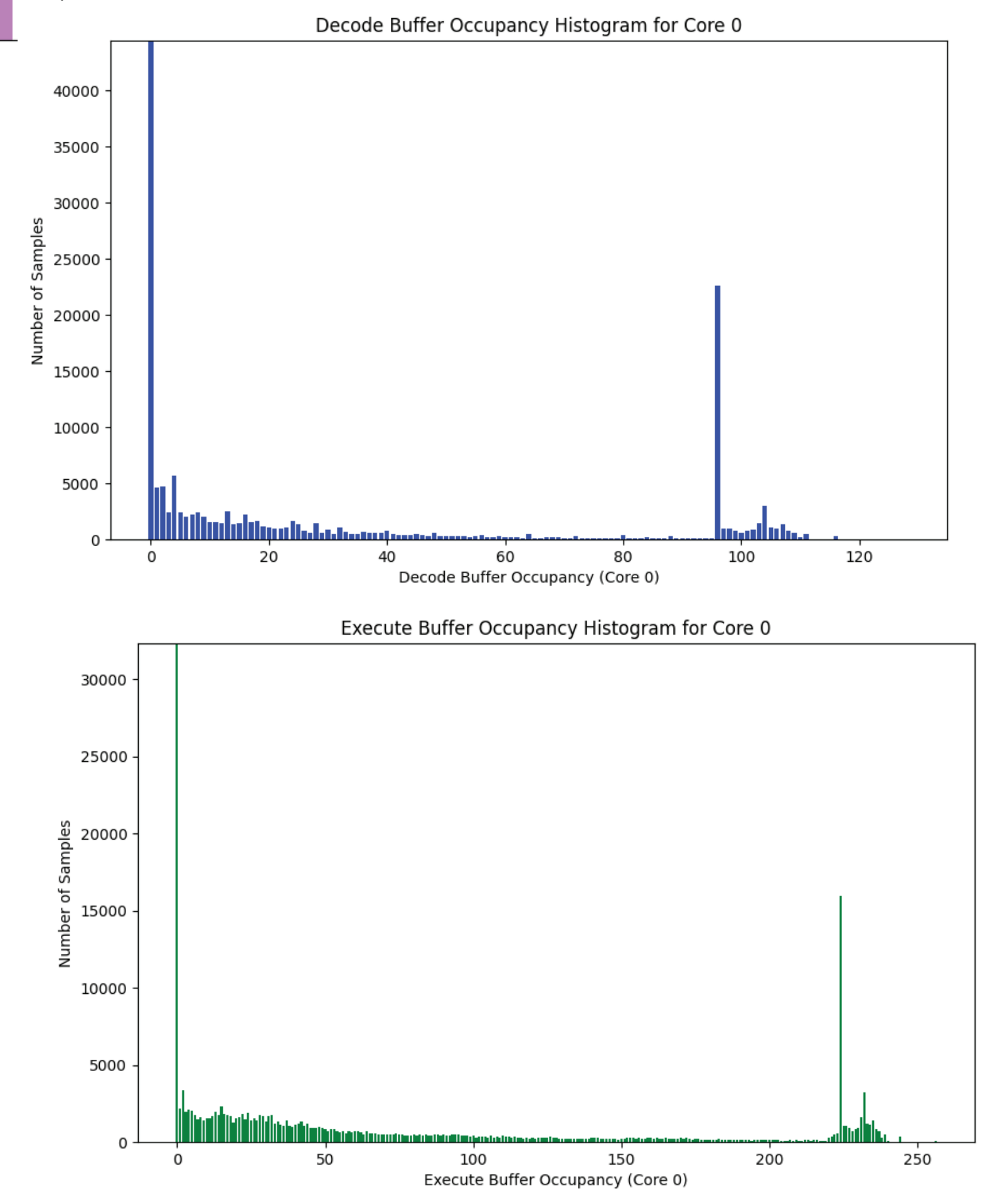
- Checker chip model developed using gem5
- Slowdown of processor throughput measured with different configurations of checker

1/4 clk
: Initial checker model

1/2 clk
: Less hardware, faster clock

1/2 clk +
: Decode/RegVerify have double the 1/2 clk bandwidth

1/2 clk ++
: Components of 1/4 clk, clocked at double the frequency



Occupancy of decode and execute buffers while operating in configuration 2

Future Work

Architecture Improvement

- Coalesce memory requests for Iverify and TLBVerify
- Perform bank queue bypassing to speed up MemVerify
- Optimal configuration (number of FUs, banks etc)

Rigorous Evaluation

- Fault injection and recovery methods
- Comprehensive benchmarking with SPEC2017

References & Acknowledgements

Thank you to Professor Hilton for his guidance and mentorship through this project

[1] N. Madan and R. Balasubramonian, "Leveraging 3D Technology for Improved Reliability". In 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007), Chicago, IL, USA, 2007, pp. 223-235, doi: 10.1109/MICRO.2007.31.

[2] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 simulator. SIGARCH Comput. Archit. News 39, 2 (May 2011), 1–7. https://doi.org/10.1145/2024716.2024718