Enhancing an Out-Of-Order Processor for Latency-Critical Cloud Applications

Abraham Gonzalez, Ronald MacMaster, Barak Lidsky, Justin Nguyen, Ryan Syed
Mentors: Mattan Erez and Haishan Zhu
November 20th, 2017

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Background and Objective

- **Datacenters** consolidate computational power to a single physical location, usually for power and cost savings.
- Many datacenters underutilize or waste resources in order to ensure **quality of service (QoS)** (high deadline completion rates) for **latency-critical tasks**.
- **Objective:** Enhance resource utilization for datacenters so that latency-critical tasks and latency-noncritical tasks can run at the same time and share resources.

Design Summary

- In order to replicate datacenter behavior, our team decided to modify **ZSIM**, an academic, timing-based, processor simulator.
- Our solution consists of two parts: implementing **simultaneous multi-threading (SMT)** on ZSIM and designing different **arbitration schemes** for increased utilization.
- Specifications to fulfill these tasks are outlined below in Figure 1 .

Specification	Description
Processes per Simulator Core (PPSC)	PPSC ≥ 2
Combined Thread Time ($Cycles_{comb}$)	$Cycles_{comb} \leq Cycles_{thread1} + Cycles_{thread2}$
High Workload Punctuality (P_w)	$P_w \geq 0.7$ or 70%
High Workload Reliability (R_w)	$R_w \geq 0.8$ or 80%

Figure 1: Design Specifications

Simultaneous Multi-threading

- **Simultaneous multi-threading** is a type of multiprocessing that can run multiple instruction threads at the same time by sharing a single processor's resources.
- In order to have a single core receive two instruction threads, our team added the new **simultaneous multi-threading window (SMT Window)** shown in Figure 2.
- Additionally, changes were added to ZSIM's timing model to simulate contention.
- As shown below, all changes were built into a new simulator core called the **SMT Core.**

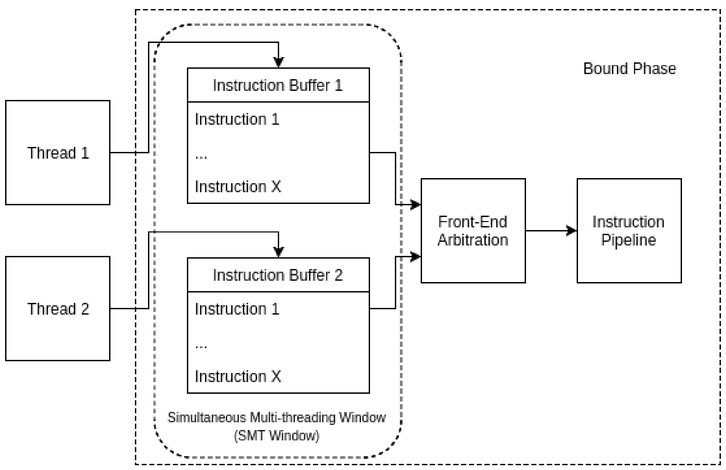

Figure 2: SMT Core and SMT Window

Arbitration Schemes

- After the new SMT Core is implemented , we will need to design more arbitration schemes to improve resource utilization and performance.
- **Arbitration scheme** – an algorithm that determines which task receives resources when multiple tasks contend with each other on a single core.
- Shown in Figure 3, we implemented a **fair arbitration scheme** that equally filled the SMT Core with instructions from two different threads and ran the instructions one after the other in a round-robin fashion.
- Allows resources to be equally split between the two threads.

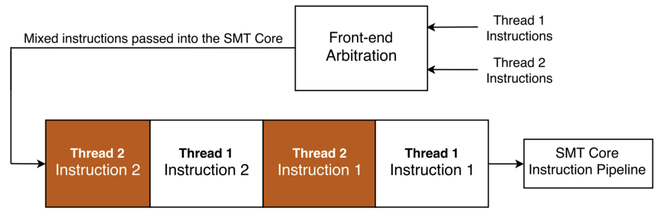

Figure 3: Arbitration and Instruction passing

Testing Methods

- Tested the simulator using several microbenchmarks.
- Tests with names ending in "good" were our baseline tests that do not intentionally cause instruction cache misses, data cache misses, or branch mispredictions.
- Tests with names ending in "miss" stressed these misses and mispredictions to see how well our simulator performs under unusual conditions.
- Each microbenchmark was run by itself on the original ZSIM Core (also known as the OOO Core) and with a duplicate of itself on the SMT Core.
- Each micro-benchmark test on the OOO Core and SMT Core was run with different instruction cache, data cache, and reorder buffer sizes through a series of configuration files shown in Figure 4-5.

```
sys = {
    coreA = { type = "OOO"; };
    caches = {
        l1dA = { size = 65536; };
        l1iA = { size = 32768; };
        l2 = { size = 2097152; };
    };
};

process0 = {
    command = "benchmark/i_cache_miss";
};
```

```
sys = {
    coreA = { type = "SMT"; };
    caches = {
        l1dA = { size = 65536; };
        l1iA = { size = 32768; };
        l2 = { size = 2097152; };
    };
};

process0 = {
    command = "benchmark/i_cache_miss";
    rob = 32;
};

process1 = {
    command = "benchmark/i_cache_miss";
    rob = 32;
};
```

Figure 4-5: OOO and SMT configuration files

Testing Results

- Our new SMT Core is generally better than ZSIM's original OOO Core.
- As shown in Figure 6, all of the SMT Core cycle counts are less than those of the OOO Core, with the exception of the branch misprediction microbenchmark. This indicates that the SMT Core is properly emulating simultaneous multi-threading.
- Figure 7 shows the percentage speedup of our SMT Core over the OOO Core. It shows this speedup for each microbenchmark and two different reorder buffer (ROB) sizes. It shows that our SMT Core is particularly better than the OOO Core when there is a low miss rate for the data and instruction caches and high miss rate for data cache.

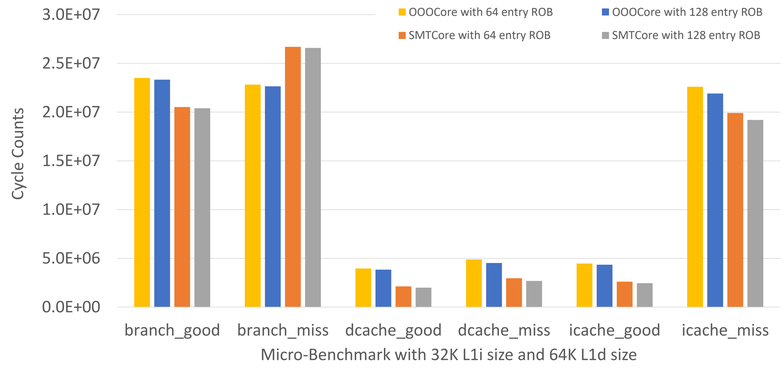

Figure 6: OOO Core vs SMT Core Comparison

	branch_good	branch_miss	dcache_good	dcache_miss	icache_good	icache_miss
64 entry ROB	12.5%	-17.4%	48.0%	39.4%	43.4%	12.4%
128 entry ROB	12.7%	-17.0%	46.4%	39.4%	41.6%	11.9%

Figure 7: Percentage Speedup from OOO Core to SMT Core

Conclusion

- Implemented a working version of simultaneous multi-threading on ZSIM.
- We were able to run smaller micro-benchmarks on the new core but experienced issues with running larger commercial benchmarks such as SPEC.
- Only designed a fair arbitration scheme due to timing constraints and difficulties in implementing simultaneous-multithreading.
- Cleaned up and refined the code base so that we could release our results to the ZSIM GitHub repository. Our work will be useful since few academic simulators have simultaneous multi-threading implemented.

Future Work

- With simultaneous multi-threading completed, future work can be focused on designing new arbitration schemes for latency-critical task performance.
- One could increase the number of processes running on each SMT Core and have multiple cores run concurrently to match current datacenter behavior.
- A cycle-based simulator could be used instead ZSIM, a timing-based simulator, for cycle-level accuracy and modeling.

Acknowledgements

We would like to thank Dr. Erez for his guidance and the opportunity to learn more about the field of parallel computer architecture. Additionally, we would like to thank Haishan Zhu for providing the computing resources and advice necessary to work on the project.

References

ZSIM: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems", Sanchez and Kozyrakis, ISCA-40, June 2013