SoCET VIP Course - Tapeout Configuration and Testing

Katelyn Shah, Minghan Wang, Asavari Deshmukh
Advisor(s): Cole Nelson

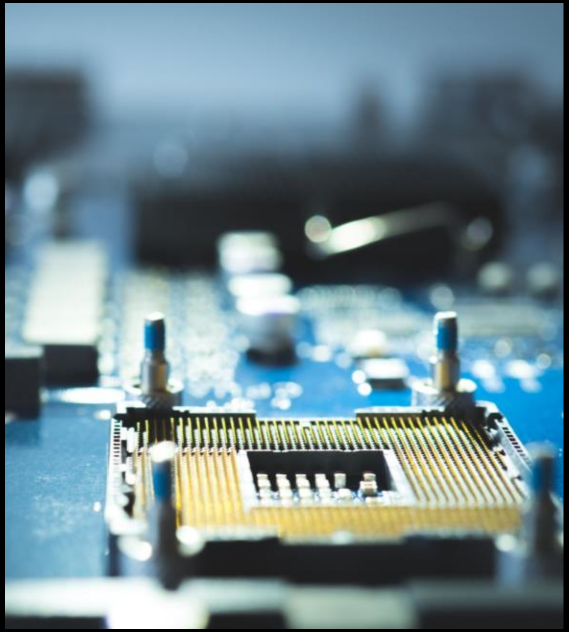
PURDUE UNIVERSITY | School of Electrical and Computer Engineering

Background

- Ensure cache sizes, ISA + extensions, and synthesis flow are correct and optimized before fabrication
- Using AFTx07 as reference of testing and validating tapeout flow and configuration for upcoming AFTx08 tapeout
- Helps figure out potential issues/fixes for upcoming AFTx08 tapeout
- Reduces time, cost, and risk of errors before fabrication stage

Semester Goals and Methods

- Adjust number, sizing and presence of components on chip (ex. cache sizing)
- Adjusting memory latency (ex. 4, 11, 132 cycles) → simulate longer delays for off-chip memories
- Optimizing performance speed through benchmarking and simulations of CPU data



Methodology

- Used previous chip design as test platform for safely testing new configurations
- Evaluated configurations:
  - 1-8KB cache sizes for d- and i-cache
  - ISA extensions (ex. zba, zbb, zbs) and base ISAs (ex. RV32I and RV32E)
- Ran synthesis scripts with Genus to get area estimates for different configurations

Benchmark Setup

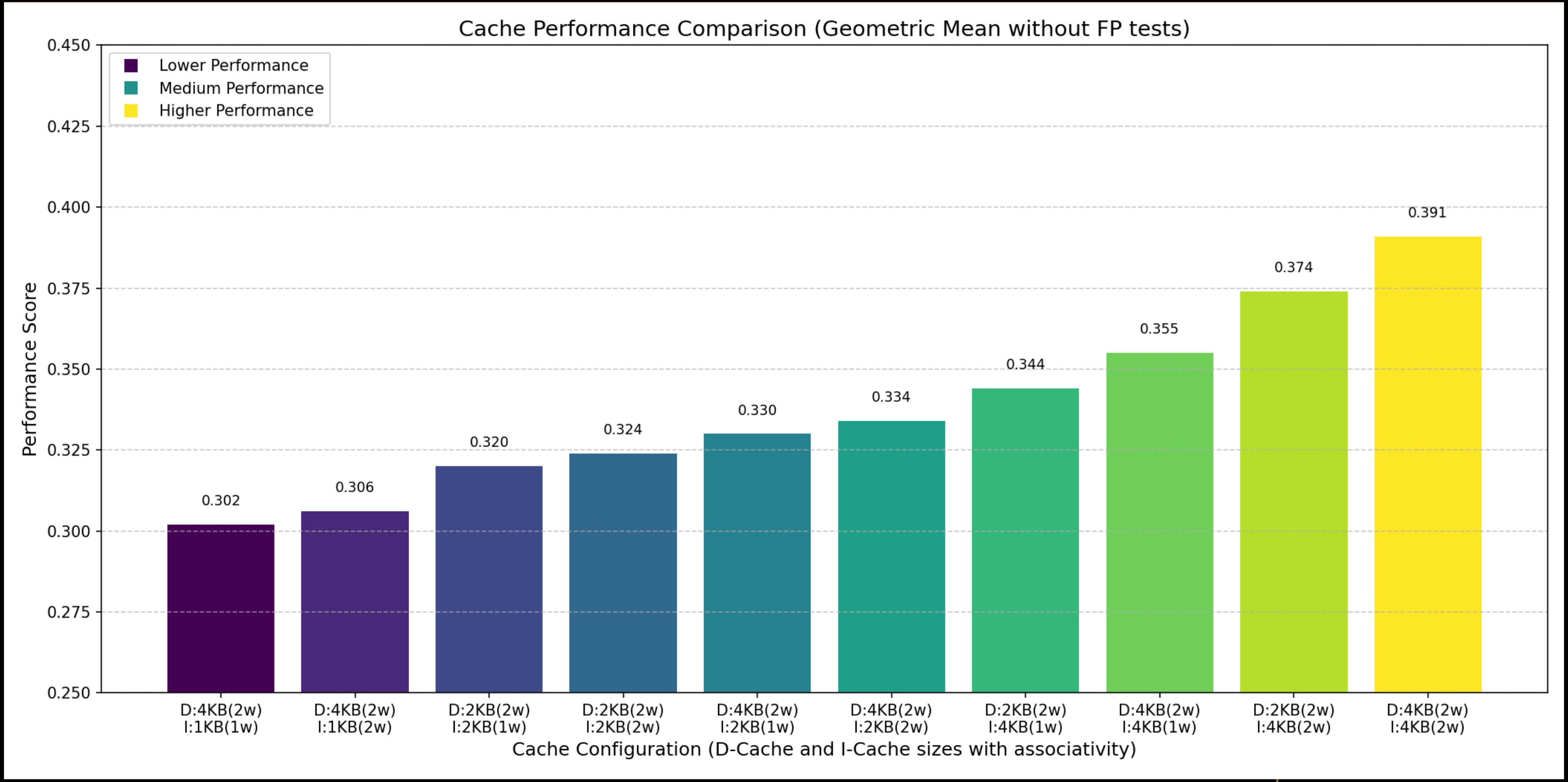
- Platforms
  - Verilator simulations (via FuseSoc)
- Intro to Embench:
  - Suite of open-source C benchmarks that run on embedded systems (>= 64kB ROM/RAM)
  - Focuses on real-world C programs, not synthetic tests, to represent embedded workloads
  - aha-mont64 (64-bit Montgomery multiplication), crc32 (checksum)
- Metric: Embench score
  - Running benchmarks gives a speed score (geo mean of performance ratios from each benchmark relative to reference platform), performance range (geo standard deviation)
  - GM > 1 → faster than ref. platform
  - Optimize for smaller GSD

Source: Embench-IoT README, GitHub

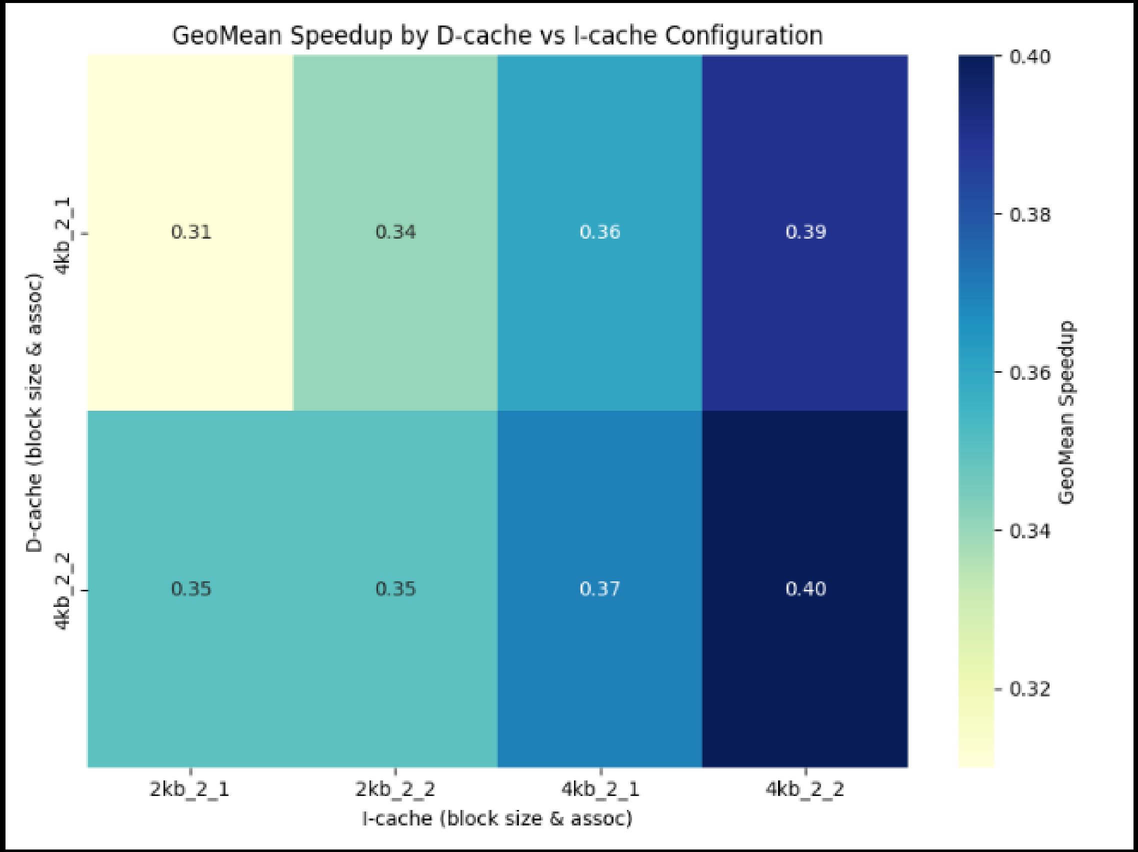
Benchmark Speed Comparison – cache size (2KB vs. 4KB vs. 8KB)

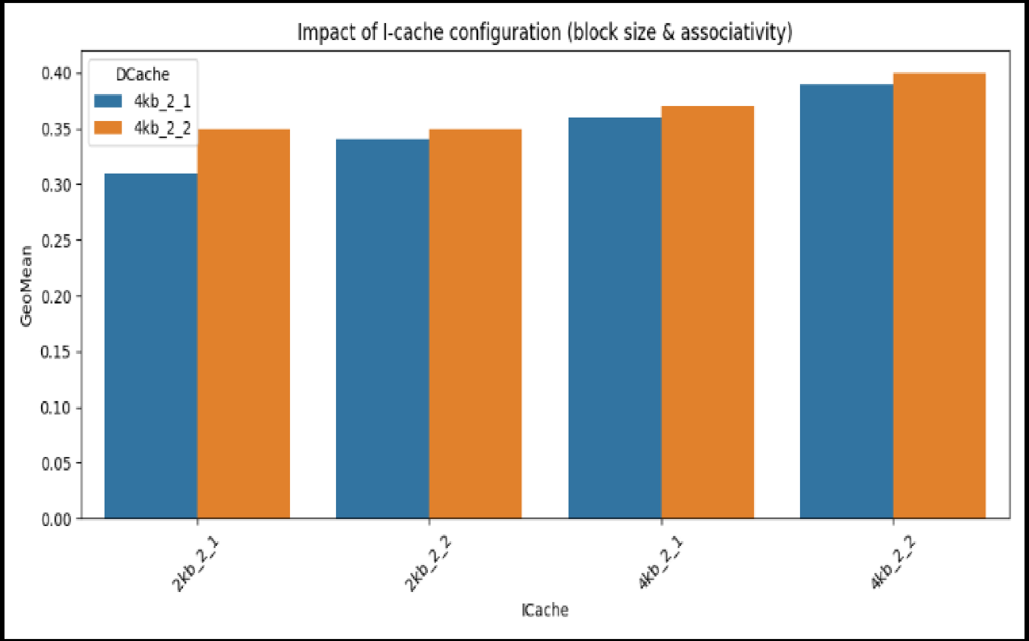
Test	D-Cache (size/B-blk/assoc)	I-Cache (size/B-blk/assoc)	Geo Mean	Geo SD	Geo Range
1	1KB / 2B / 1	1KB / 2B / 1	0.32	1.68	0.35
2	4KB / 8B / 4	1KB / 2B / 1	0.28	1.70	0.31
3	2KB / 2B / 1	2KB / 2B / 1	0.34	1.70	0.38
4	2KB / 4B / 2	2KB / 4B / 2	0.33	1.76	0.39
5	4KB / 4B / 2	4KB / 4B / 2	0.37	1.74	0.43
6	4KB / 2B / 1	4KB / 2B / 1	0.38	1.68	0.41
7	8KB / 16B / 8	8KB / 16B / 8	0.23	1.80	0.28
8	**8KB / 2B / 1**	**8KB / 2B / 1**	**0.44**	**1.63**	**0.44**

Data cache size vs. Instruction cache size



Block Size vs. Associativity





RISC-V Instruction Set Architecture (ISA)

- What is RISC-V?
  - Reduced Instruction Set Computing——fifth version
  - RISC-V is a set of open standards for instruction sets.
- What is an ISA?
  - ISA defines a set of instructions a processor can execute, registers, data types and so on.
  - Interface between software and hardware.
- Base and Extension Architecture
  - Base ISA → RV32I, RV32E
  - ISA Extensions → RV32IMC (base ISA + multiply + compress)
  - Ex. RV32IM_zba_zbb_zbc → (Z- and B- extension set; not full B extension set)

RISC-V Instruction Set Architecture (ISA)

- Base ISA (Each RISC-V implementation must include one base ISA)
  - RV32I – 32-bit base integer ISA
  - RV64I – 64-bit base integer ISA

- ISA Extensions
  - RV32E - embedded
  - RV32IMCB- 32-bit base + Integer multiply + Compressed + Bit manipulation
  - Z-extensions – smaller extensions that focus on specific ISA

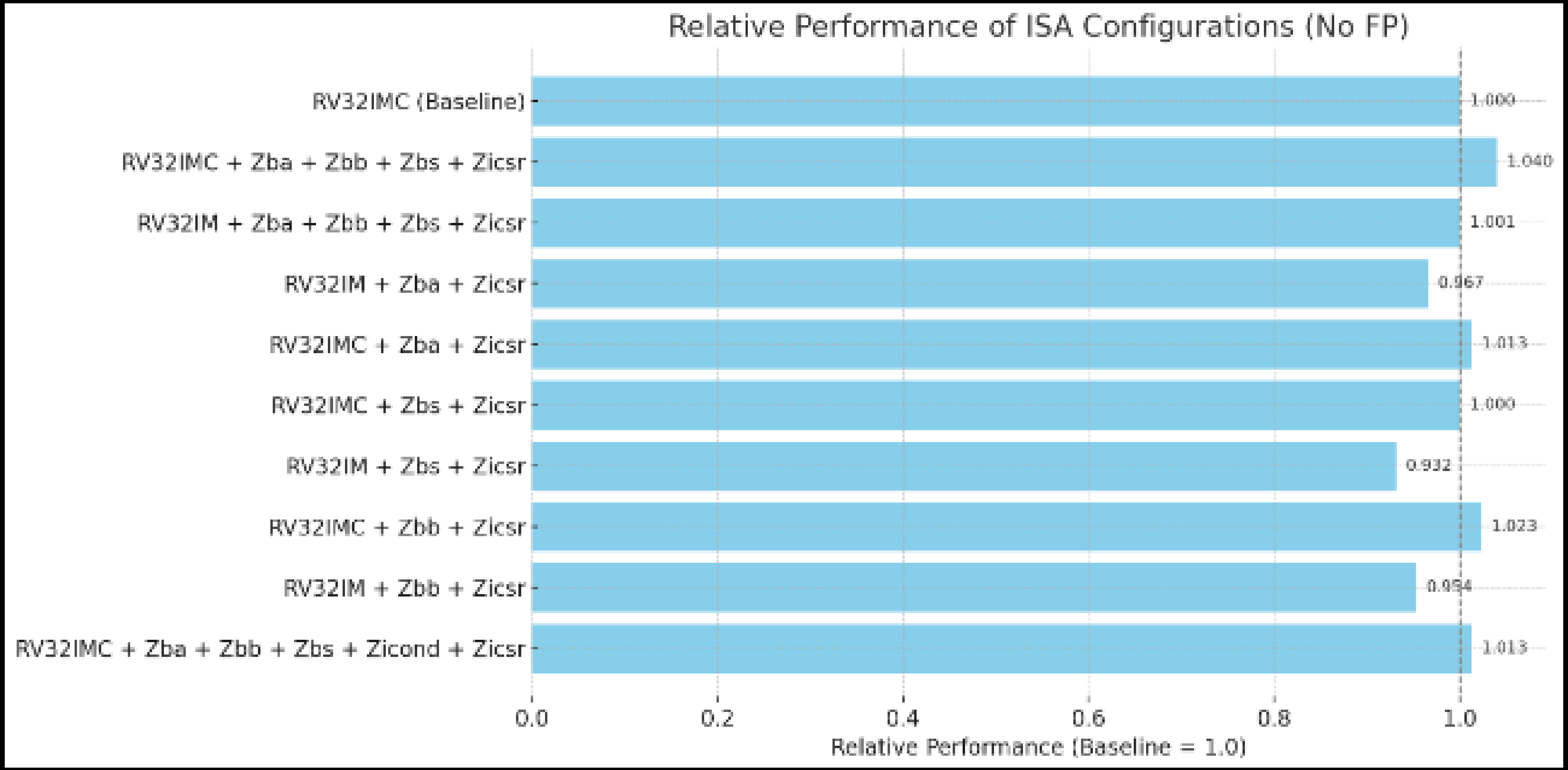
Extension	Name
M	Integer Multiply/Divide
A	Atomic Instructions
F	Single-Precision Floating-Point
D	Double-Precision Floating-Point
C	Compressed Instructions
B	Bit Manipulation
Zicsr	Control and Status Registers
Zifencei	Instruction-Fetch Fence
Zicond	Integer Conditional Operations

Intro to B- and C-extensions

- B extension (Bit manipulation)
  - Shifts, rotations, and population count
  - Efficiency → Reduces instruction count for bit-level algorithms
  - Modular Z extensions: zba (add + shift, logical operations, carry-less multiplication, clear)
  - Uses: compression, digital signal processing, cryptography
- C extension (Compressed)
  - 16-bit instruction encoding for 32-bit instructions
  - 25-30% smaller binaries
  - Energy efficient → Less memory fetches

Extension	Name
M	Integer Multiply/Divide
A	Atomic Instructions
F	Single-Precision Floating-Point
D	Double-Precision Floating-Point
C	**Compressed Instructions**
B	**Bit Manipulation**
Zicsr	Control and Status Registers
Zifencei	Instruction-Fetch Fence
Zicond	Integer Conditional Operations

Evaluating the Impact of RISC-V Bitmanip (B) Extensions on AFTx08 Performance



School of Electrical and Computer Engineering

Evaluating the Impact of RISC-V Compressed (C) Extensions on AFTx08 Performance

Comparison Pair	Description	With C	Without C	C - No C
rv32imc_zba_zbb_zbs_zicsr vs rv32im_zba_zbb_zbs_zicsr	**All B extensions + Zicsr**	**0.3961**	**0.3809**	**+0.0152**
rv32imc_zba_zicsr vs rv32im_zba_zicsr	Only Zba + Zicsr	0.3857	0.3680	+0.0177
rv32imc_zbs_zicsr vs rv32im_zbs_zicsr	Only Zbs + Zicsr	0.3808	0.3547	+0.0261
rv32imc_zbb_zicsr vs rv32im_zbb_zicsr	Only Zbb + Zicsr	0.3896	0.3631	+0.0265

Conclusions

- C extension improves i-cache performance due to better code density (less cache misses)
- B extension improves speed and efficiency of instructions
- Lower dynamic power and memory used → less bits for each instruction

Future Work

- Update GCC version to evaluate impact of zicond
- Test area estimation scripts for other configurations
- Benchmark with RV32E ISA
- Integrate branch predictor

References

- [1] Embench contributors, *Embench-IoT benchmark suite documentation*, Free and Open Source Silicon Foundation, GitHub, 2020. [Online]. Available: https://github.com/embench/embench-iot/blob/master/doc/README.md

THANK YOU!