RISC-V: Data Integrity in High-Performance Data Analysis

RISC-V: State of the Union

By: Abdallah Abdelnaby

Supervisor: Freja Nordsiek

Agenda

- Introduction
- RISC-V in Data Analytics
- RISC-V Architecture
- Problem Statement
- Triple Modular Redundancy (TMR)
- Project Design
- Results
- Pros & Cons
- Conclusion and Future Work

Inroduction

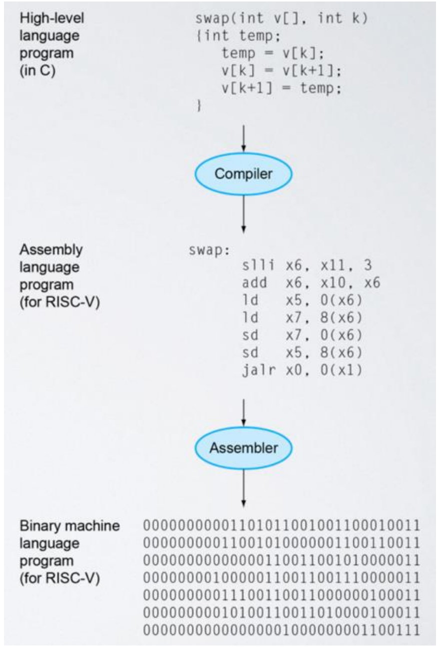
What is RISC-V?
Open-source ISA developed for educational, research and industrial use.
Developed by Krste Asanovi , Andrew Waterman, and Yunsup Lee at UC Berkeley, with input from David Patterson in 2010.
Currently maintained by the RISC-V Foundation: A non-profit corporation (http://riscv.org).

Importance in data analysis:

- Scalability
- Flexibility
- efficiency

Instruction Set Architecture (ISA)

- An abstract interface between the hardware and the lowest-level software
- ISA encompasses all the information necessary for programmers to write a machine language program that will run correctly, including instructions, registers, memory access, I/O devices, etc.
- ISA standardizes instructions, machine language bit patterns, etc.
- The ISA abstract interface enables many implementations (presumably varying in cost and performance) to run identical software (same architecture)
- Modern ISAs: x86/Pentium/K6, PowerPC, MIPS, RISC-V, DEC Alpha, SPARC, HP PA


High-level language program (in C)
Compiler
Assembly language program (for RISC-V)
Assembler
Binary machine language program (for RISC-V)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

```
swap:
      slli x6, x11, 3
      add  x6, x10, x6
      ld   x5, 0(x6)
      ld   x7, 8(x6)
      sd   x7, 0(x6)
      sd   x5, 8(x6)
      jalr x0, 0(x1)
```

```
00000000001101011001001100010011
00000000011001010000001100110011
00000000000000110011001010000011
00000000100000110011001110000011
00000000011100110011000000100011
00000000010100110011010000100011
00000000000000001000000001100111
```

Figure 1 AUC Computer Architecture

Statistics

Total= 96%

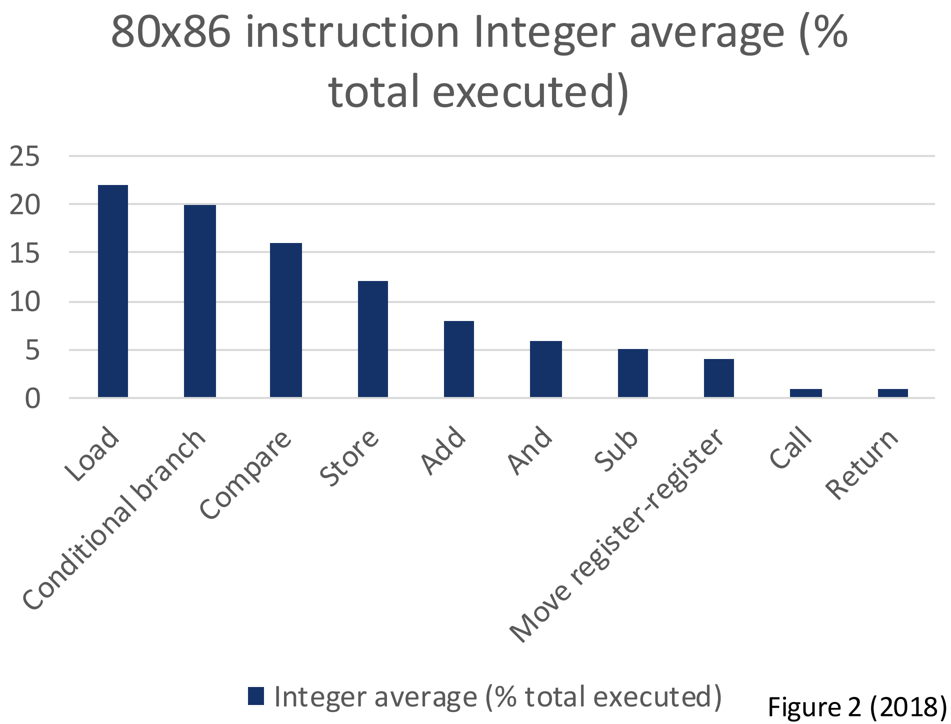


Figure 2 (2018) Computer Architecture: A Quantitative Approach

RISC-V in Data Analytics

RISC-V based SiFive’s P670 processor vs. Arm’s Cortex-A78 processor

- Peak Single-Thread Performance
- Area
- Compute Density

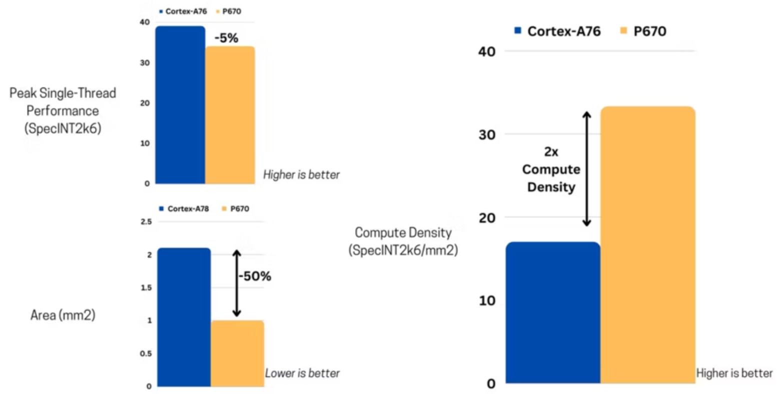
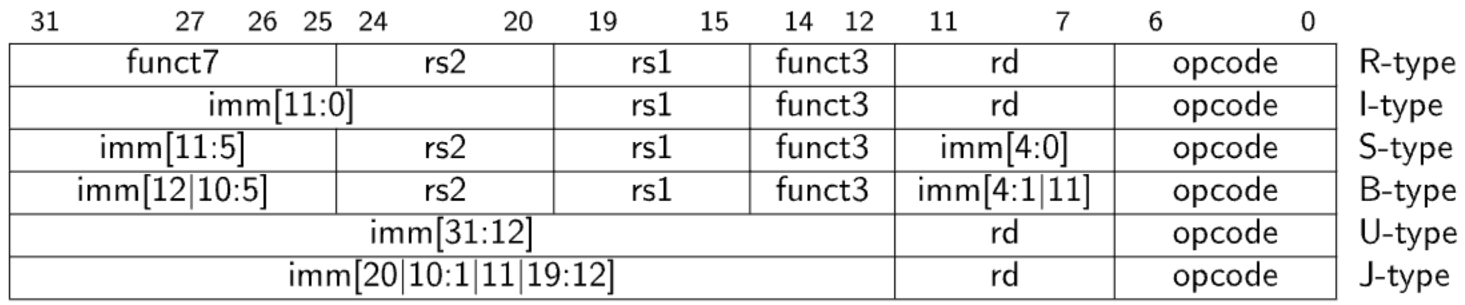


Figure 3: (2022) RISC vs. RISC-V vs. ARM: What Is the Difference?

RISC-V 32I Instruction Types



31 27 26 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

Figure 4 (2021) Five EmbedDev

RISC-V 32I Architecture

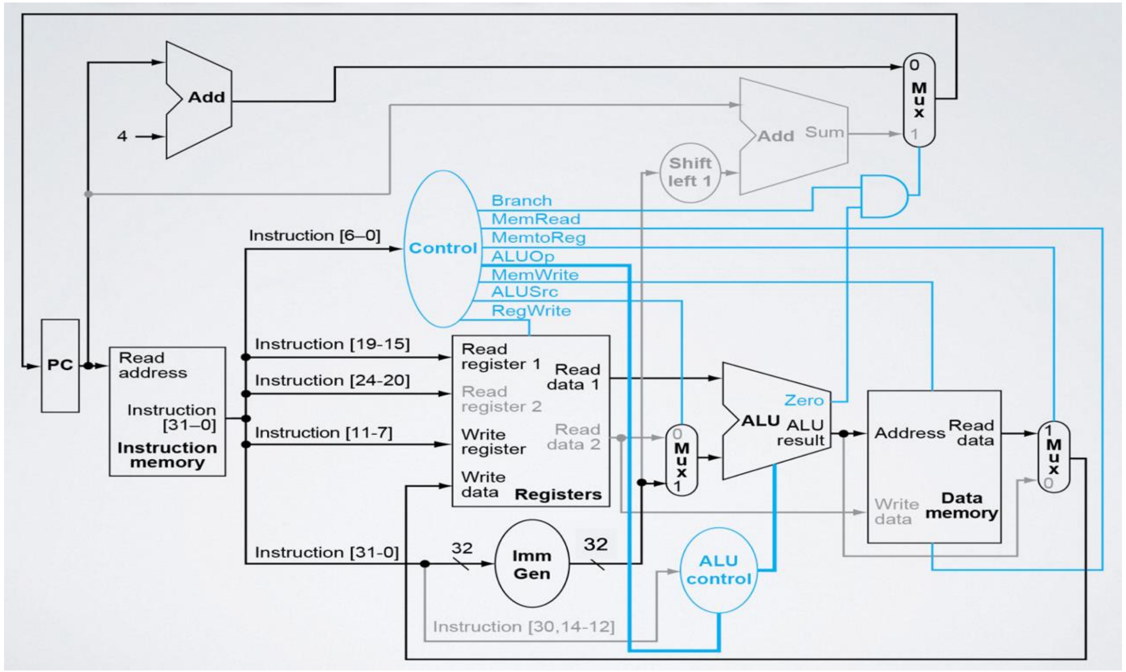


Figure 5 AUC Computer Architecture

Program Counter (PC)

4 Bit Register Example

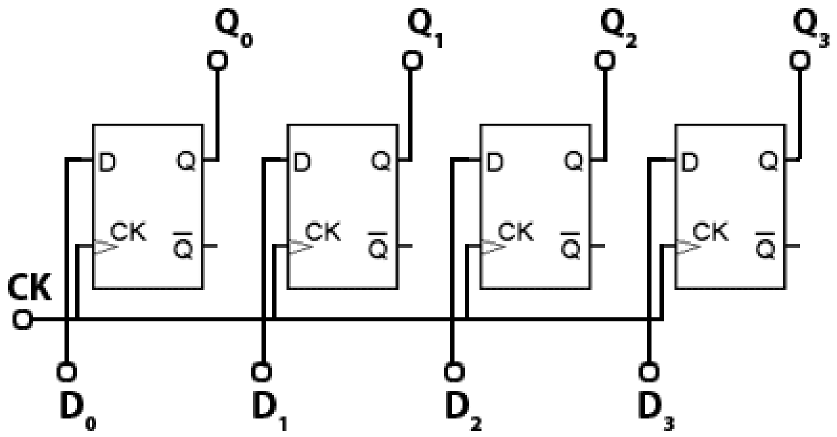


Figure 6 (2020) Performance Analysis of MAC

Sequential Elements

- Register with write control
  - Only update on clock edge when write control input is 1
  - Used when the stored value is required later

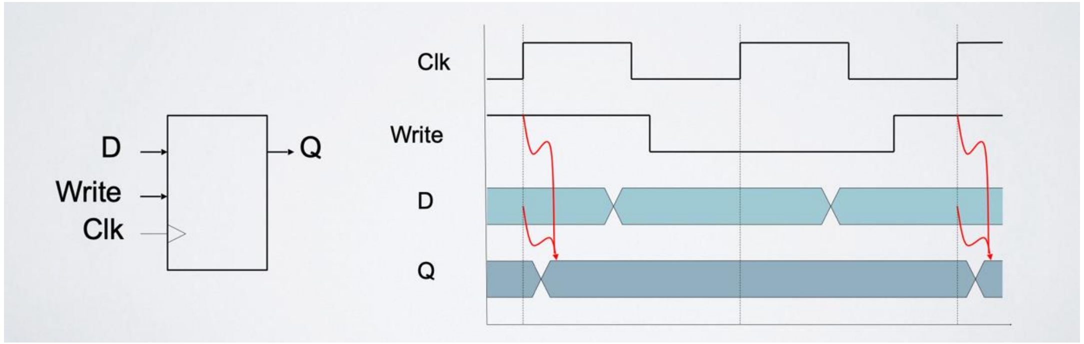
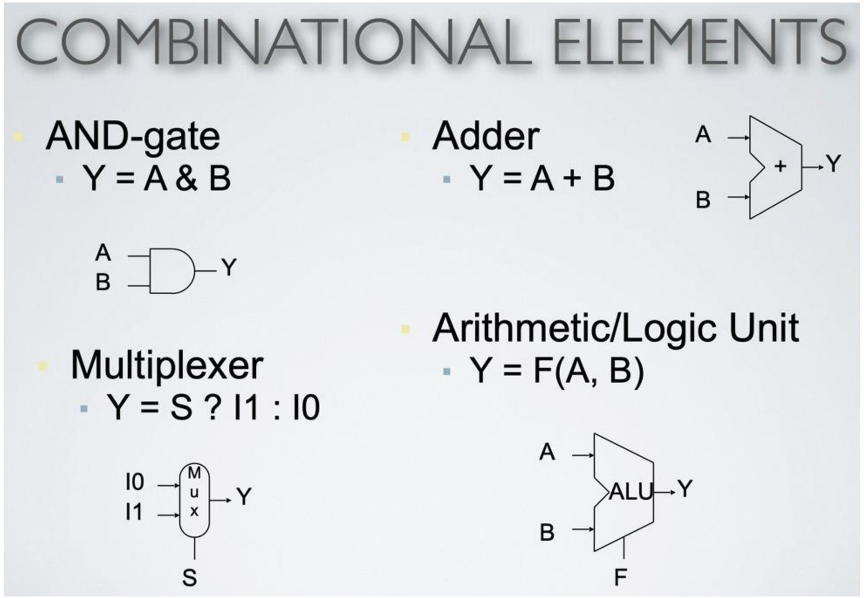


Figure 7 AUC Computer Architecture

Combinational Elements



COMBINATIONAL ELEMENTS

- AND-gate
  - Y = A & B
- Multiplexer
  - Y = S ? I1 : I0
- Adder
  - Y = A + B
- Arithmetic/Logic Unit
  - Y = F(A, B)

A
B
Y
I0
I1
M
u
x
Y
S
A
B
+
Y
A
B
ALU
Y
F

Figure 8 AUC Computer Architecture

Register File

- x0: the constant value 0
- x1: return address
- x2: stack pointer
- x3: global pointer
- x4: thread pointer
- x5 – x7, x28 – x31: temporaries
- x8: frame pointer
- x9, x18 – x27: saved registers
- x10 – x11: function arguments/results
- x12 – x17: function arguments

Memory

Main memory used for composite data Arrays, structures, dynamic data to apply arithmetic operations

- Load values from memory into registers
- Store results from register to memory

- Memory is byte addressed
- Each address identifies an 8-bit byte
- RISC-V is Little Endian (like x86)
- RISC-V does **NOT** require words to be aligned in memory unlike MIPS for example

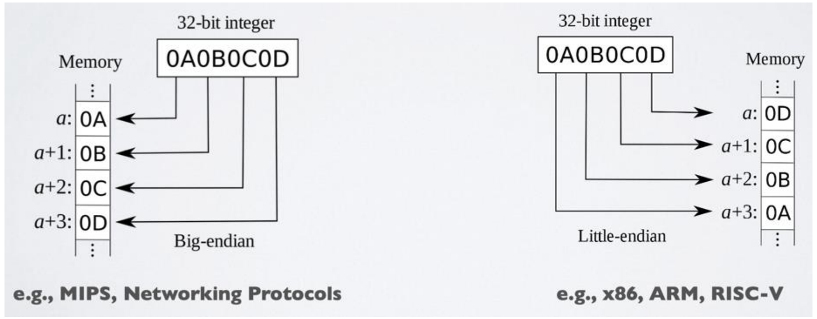


Figure 9: (2017) Computer Organization and Design: RISC-V Edition

Data Loss Issues

Problems with flip-flops and memory

- Bit flips in the program counter (PC)
- Data memory corruption

Impact on data integrity and processing accuracy

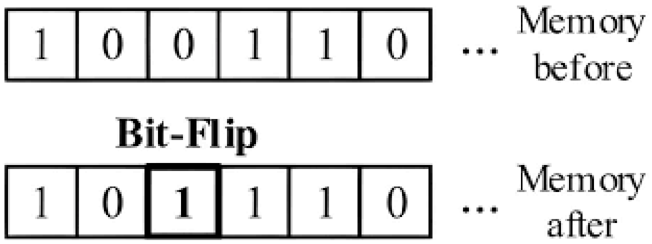


Figure 10 (2024) ScienceABC

Cosmic rays bit flip cases

- Belgium's national election in 2003. (Radiolab)
- Qantas Flight Incident in 2008. (physicsworld)

- Imagine with your bank account!!
  - Become a billionaire or be broke

Triple Modular Redundancy (TMR)

What is TMR?
How TMR works: replication and majority voting
Benefits of using TMR in critical components like PC and data memory

TMR Majority Voting

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 1

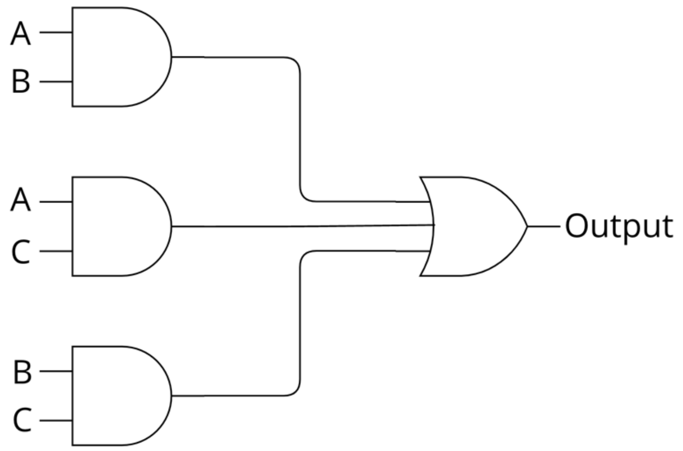


Figure 11

Output = AB + AC + BC

TMR Applied to Program Counter, Registers File and Data Memory

Implementation details

- Design changes in Verilog-HDL
- Integration with existing architecture

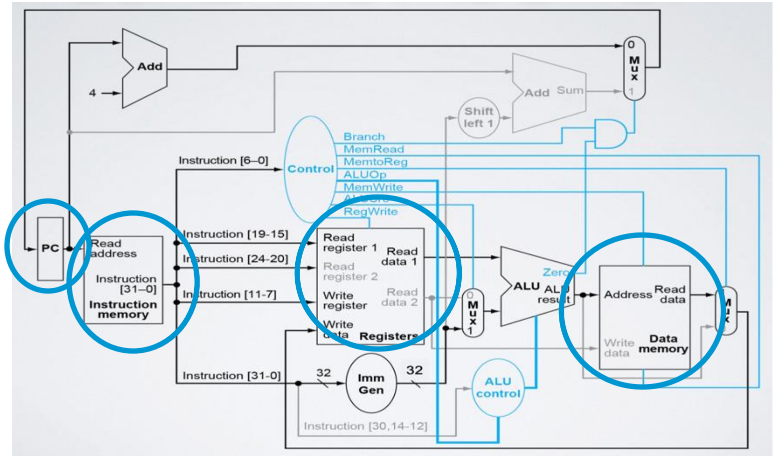


Figure 12 AUC Computer Architecture

Project Implementation and Simulation

Data Memory size= 256 Bytes

Instruction Memory size = 4 KB

Development environment

- Vivado - Verilog-HDL
- Simulation board: Alpha-Data ADM-PCIE-7V3 (xc7vx690tffg1157-2)

Results

- Opt_design disabled Logic Delay increased by 121.4%

- Opt_design enabled Logic Delay decreased by 0.05%

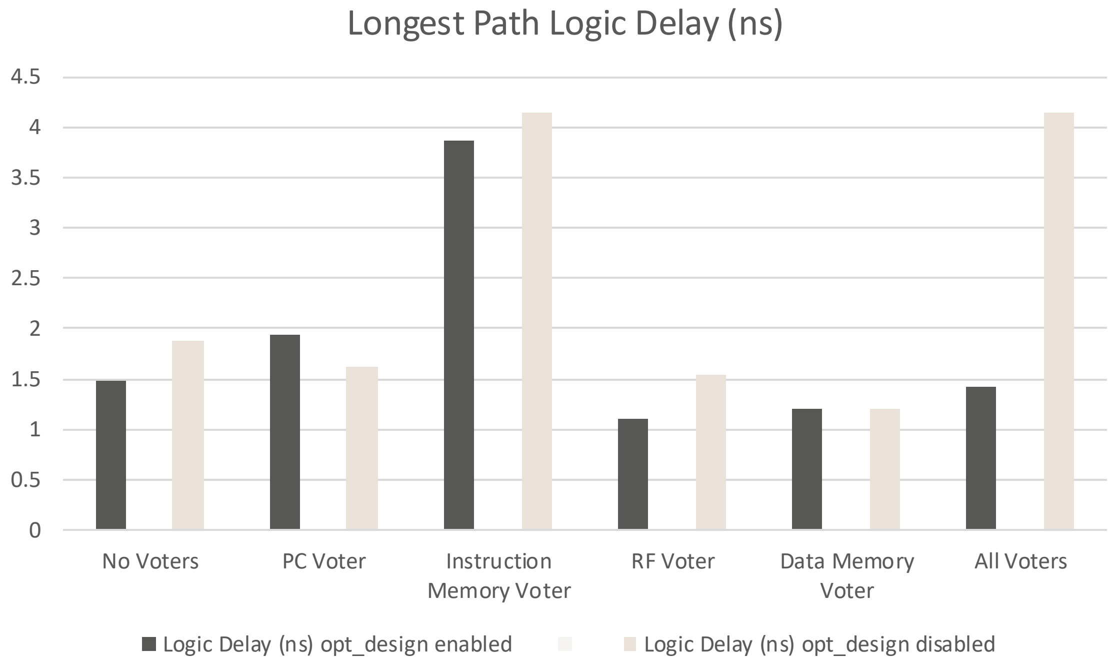


Figure 13

Results

- Opt_design disabled Net Delay increased by 38.4%
- Opt_design enabled Net Delay increase by 63.9%

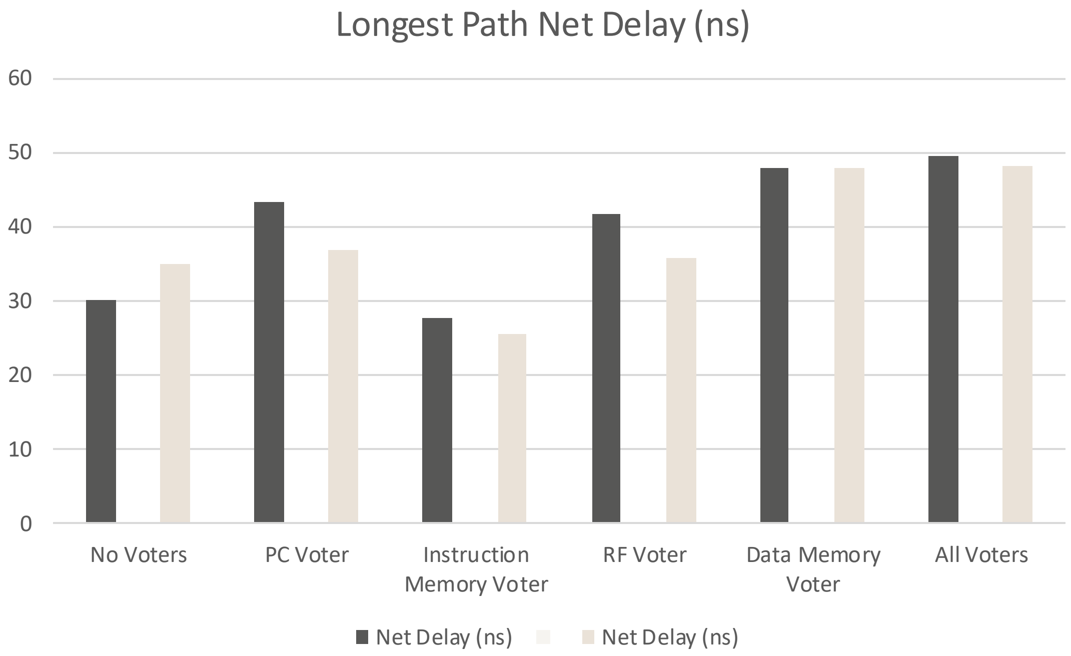


Figure 14

Results

- Opt_design disabled Total Delay increased by 42.7%

- Opt_design enabled Total Delay increase by 60.6%

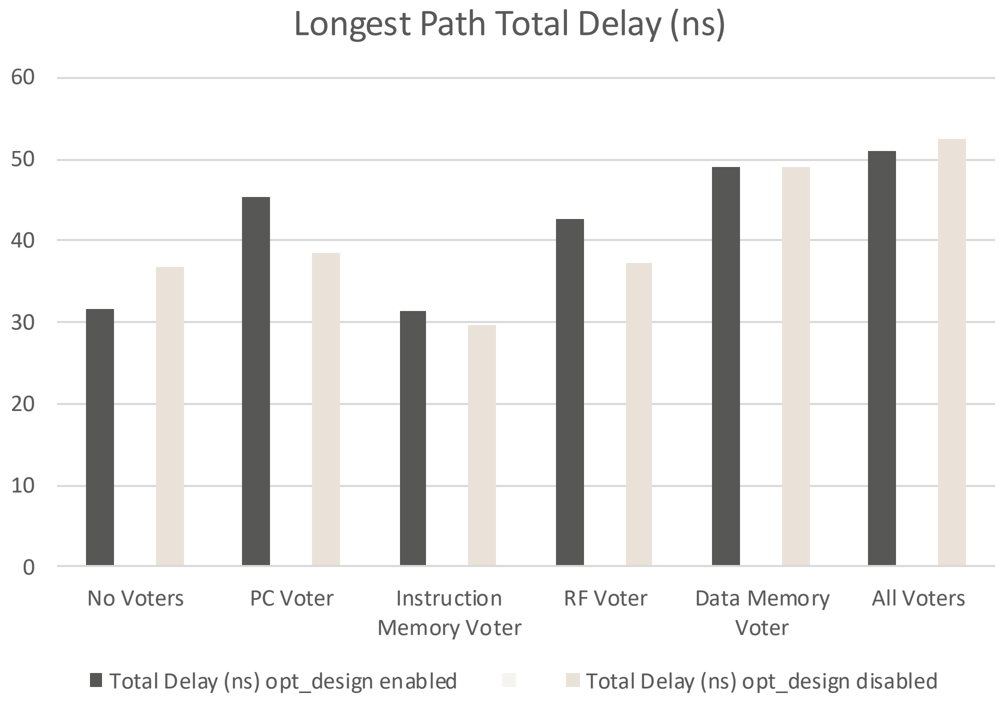


Figure 15

Power Consumption Results

Power Consumption increased by 17.1%

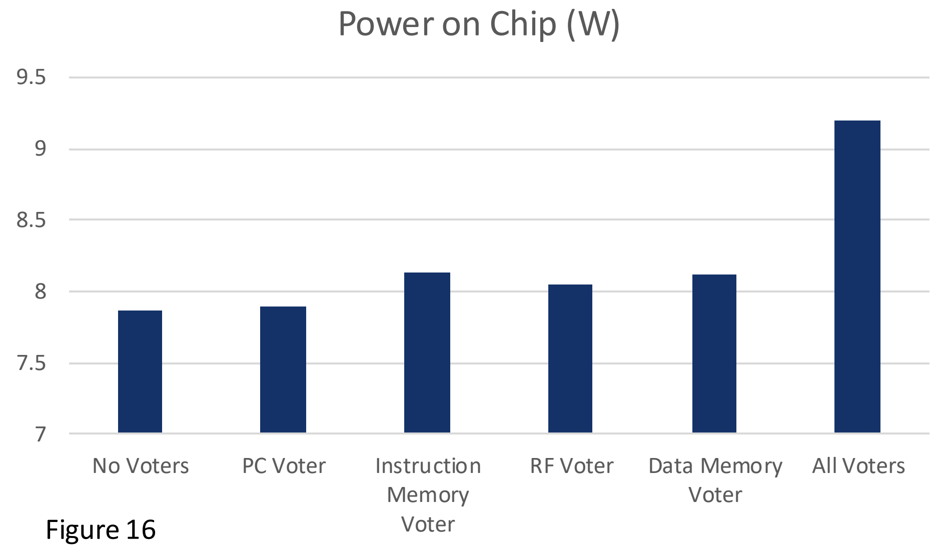
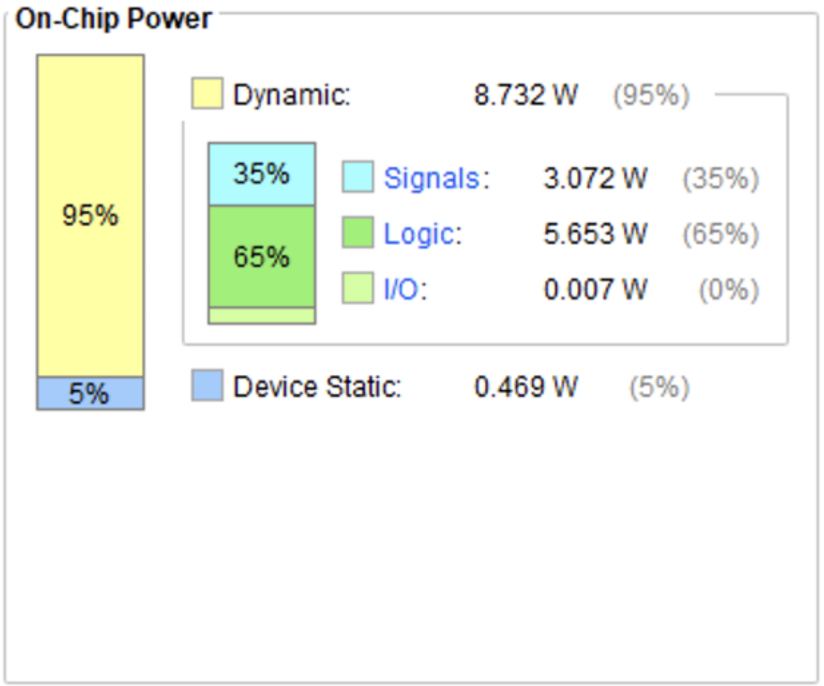


Figure 16

Power Distribution


On-Chip Power

Dynamic:	8.732 W	(95%)	
	Signals:	3.072 W	(35%)
	Logic:	5.653 W	(65%)
	I/O:	0.007 W	(0%)
Device Static:	0.469 W	(5%)	

Figure 17

Temperature Results

Junction Temperature increased by 5.3%

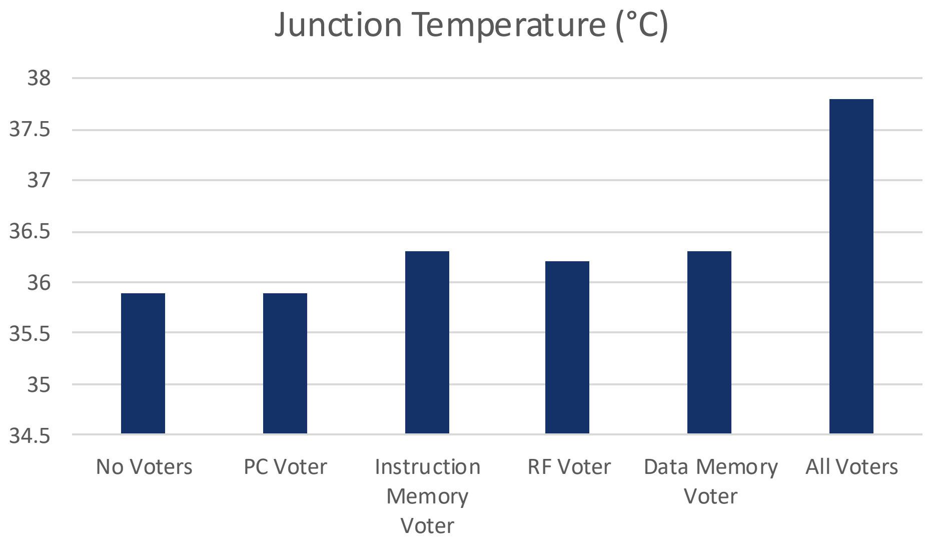


Figure 18

Conclusion

Pros:

- Reliability improvements
- Data integrity
- Overall system robustness

Cons:

- Additional logic
- Power consumption
- More hardware ➔ Higher Cost
- Higher Temperature

Future Work

Pipelined RISC-V

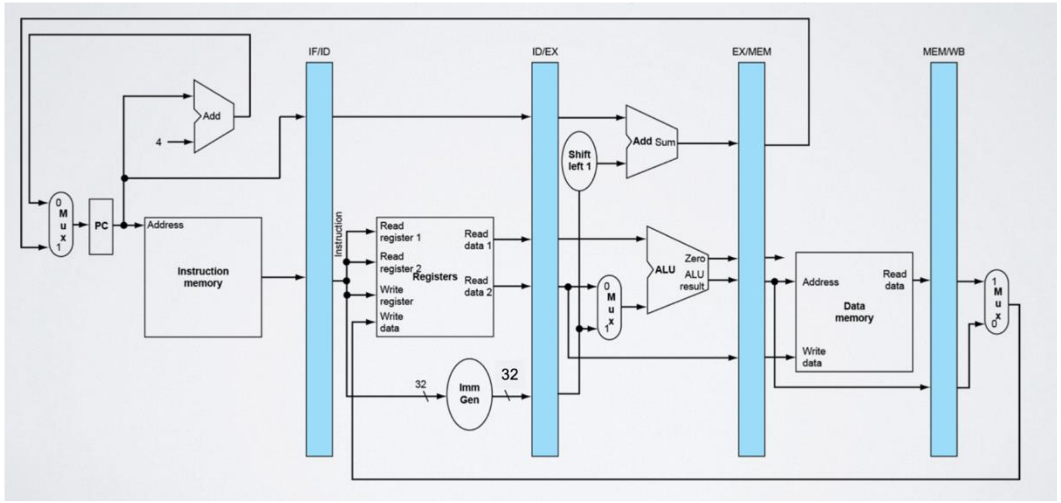


Figure 19 AUC Computer Architecture

References

- Ali, W. (2023). Exploring Instruction Set Architectural Variations: x86, ARM, and RISC-V in Compute-Intensive Applications. Eng OA, 1(3), 157-162.
- David Patterson and John L. Hennessy, *Computer Organization and Design: RISC-V Edition*, Morgan Kaufmann, 2017
- John L. Hennessy and David Patterson, *Computer Architecture: A Quantitative Approach*, Sixth Edition, Morgan Kaufmann, 2018
- Krishna V, Nitin. (2020). Performance Analysis of MAC Unit using Booth, Wallace Tree, Array and Vedic Multipliers. 10.13140/RG.2.2.11002.93127.
- https://riscv.org/technical/specifications/
- https://www.opastpublishers.com/open-access-articles/exploring-instruction-set-architectural-variations-x86-arm-and-riscv-in-computeintensive-applications.pdf
- https://www.wevolver.com/article/risc-v-vs-arm
- https://www.microcontrollertips.com/risc-v-vs-arm-vs-x86-whats-the-difference/
- https://blog.paessler.com/risc-v-vs-arm-who-wins
- https://www.makeuseof.com/risc-vs-arm-what-is-the-difference/#:~:text=The%20ARM%20ISA%20allows%20Arm,chips%20without%20paying%20license%20fees.
- https://techjourneyman.com/blog/arm-vs-risc-v/
- https://catalog.aucegypt.edu/preview_course_nopop.php?catoid=36&coid=82235