1. Introduction

This project presents the design, verification, and FPGA implementation of a five-stage pipelined RV32I processor. The objective was to develop a functionally correct and timing-aware RISC-V CPU core while gaining practical experience with processor microarchitecture, hazard handling, and FPGA-based implementation.

The processor was described using VHDL, verified against a software-based golden model, and synthesized using Intel Quartus Prime. Emphasis was placed on both architectural correctness and realistic timing closure.

---

2. Processor Architecture Overview

The processor implements the RV32I base integer instruction set and follows a classic five-stage pipeline organization:

1. **Instruction Fetch (IF)**
2. **Instruction Decode (ID)**
3. **Execute (EX)**
4. **Memory Access (MEM)**
5. **Write Back (WB)**

Pipeline registers separate each stage, enabling concurrent execution of multiple instructions. A Harvard-style interface is used, with independent instruction and data memory ports.

---

3. Microarchitectural Design

3.1 Register File

The processor includes a 32-entry register file, with each register being 32 bits wide. Two read ports and one write port are provided. Register x0 is hardwired to zero, as required by the RISC-V specification.

3.2 Control Unit

A centralized control unit decodes instruction opcodes and generates control signals such as register write enable, memory access control, branch control, and ALU operation selection.

3.3 ALU and ALU Control

The Arithmetic Logic Unit supports arithmetic, logical, and comparison operations required by RV32I. ALU control signals are generated based on instruction function fields and are precomputed to reduce critical path delay.

3.4 Immediate Generator

Immediate values are extracted and sign-extended during the decode stage. This reduces combinational complexity in later pipeline stages.

3.5 Hazard Detection and Forwarding

To maintain correctness in the presence of data hazards:

- A load-use hazard detection unit inserts pipeline stalls when required
- A forwarding unit resolves most data hazards by bypassing results from later pipeline stages

3.6 Branch Handling

Branch conditions are evaluated using ALU comparison results. Branch decisions are registered to simplify control logic and improve timing predictability.

---

4. Verification Methodology

Verification was performed using both HDL simulation and software-based validation.

4.1 Simulation-Based Verification

The processor was simulated using ModelSim. Waveform inspection confirmed:

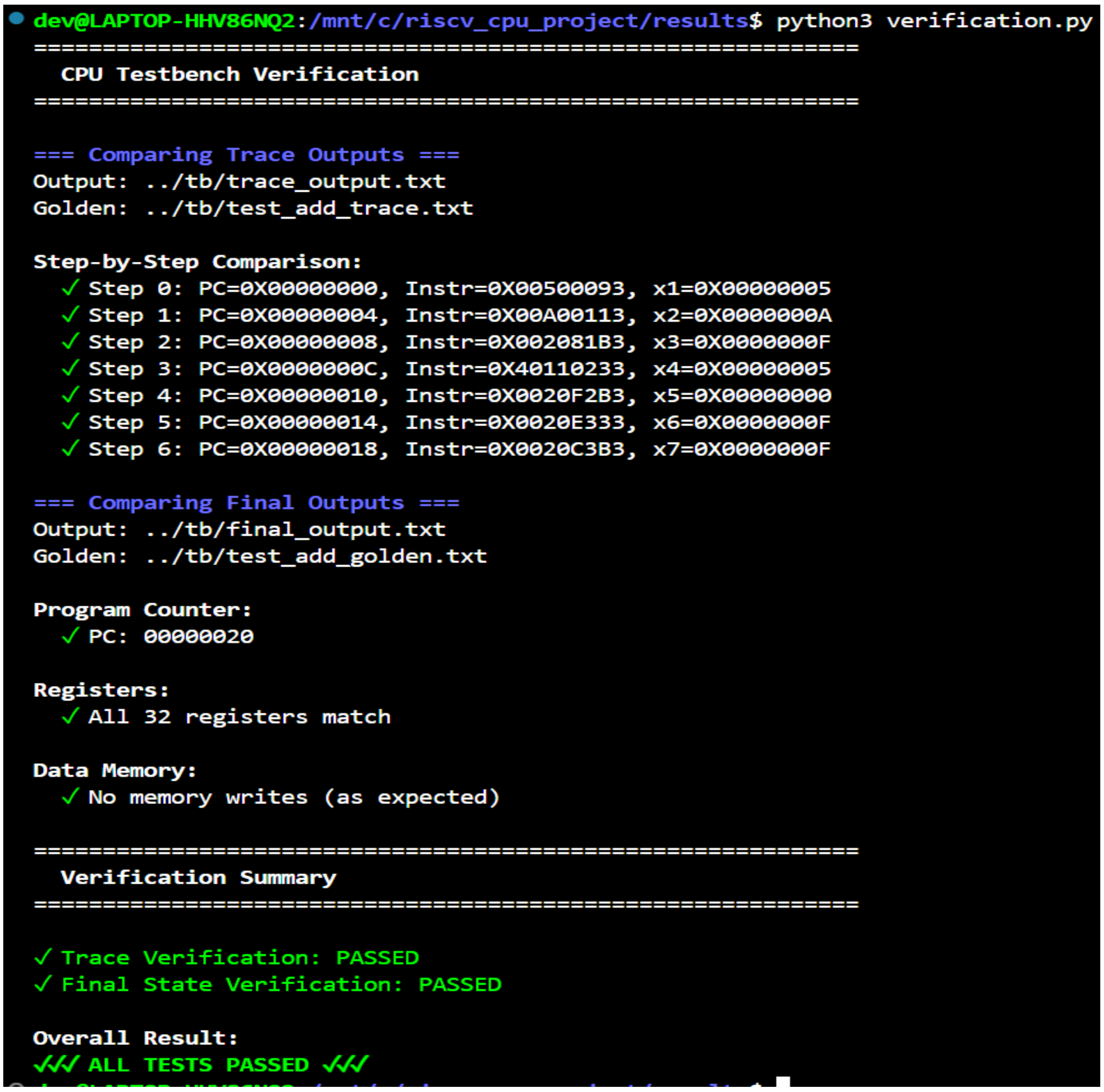
- Correct pipeline progression
- Proper hazard resolution
- Correct program counter updates
- Absence of spurious register or memory writes

4.2 Trace-Based Verification

A Python-based verification framework was developed to compare the processor's execution trace against a golden reference generated by a C-based model.

The comparison was performed on a step-by-step basis, validating:

- Program counter values
- Executed instructions
- Register write-back values

- 

```
● dev@LAPTOP-HHV86NQ2:/mnt/c/riscv_cpu_project/results$ python3 verification.py
============================================================
  CPU Testbench Verification
============================================================

=== Comparing Trace Outputs ===
Output: ../tb/trace_output.txt
Golden: ../tb/test_add_trace.txt

Step-by-Step Comparison:
  ✓ Step 0: PC=0X00000000, Instr=0X00500093, x1=0X00000005
  ✓ Step 1: PC=0X00000004, Instr=0X00A00113, x2=0X0000000A
  ✓ Step 2: PC=0X00000008, Instr=0X002081B3, x3=0X0000000F
  ✓ Step 3: PC=0X0000000C, Instr=0X40110233, x4=0X00000005
  ✓ Step 4: PC=0X00000010, Instr=0X0020F2B3, x5=0X00000000
  ✓ Step 5: PC=0X00000014, Instr=0X0020E333, x6=0X0000000F
  ✓ Step 6: PC=0X00000018, Instr=0X0020C3B3, x7=0X0000000F

=== Comparing Final Outputs ===
Output: ../tb/final_output.txt
Golden: ../tb/test_add_golden.txt

Program Counter:
  ✓ PC: 00000020

Registers:
  ✓ All 32 registers match

Data Memory:
  ✓ No memory writes (as expected)

============================================================
  Verification Summary
============================================================

✓ Trace Verification: PASSED
✓ Final State Verification: PASSED

Overall Result:
✓✓✓ ALL TESTS PASSED ✓✓✓
```

4.3 Final State Verification

At the end of execution:

- All 32 registers matched the golden model
- Program counter value matched the expected result
- No unintended memory writes were observed

All verification tests passed successfully.

---

5. Simulation and Synthesis Top-Level Separation

For clarity and efficiency, separate top-level modules were used for simulation and synthesis.

The simulation top-level included additional debug signals, register dumps, and trace-generation logic required for functional verification and automated testing.

The synthesis top-level excluded all non-synthesizable and debug-only constructs, ensuring accurate resource utilization and timing analysis.

Both versions shared the same internal processor core logic, guaranteeing functional equivalence between simulation and synthesized implementations.

---

6. FPGA Implementation

The processor core was synthesized and implemented using Intel Quartus Prime. The design was mapped to a Cyclone V FPGA device. To enable realistic timing analysis, all pipeline stages were fully registered.

External instruction and data memory interfaces were modeled separately, allowing the processor core to be analyzed independently of memory timing.

---

7. Timing Analysis

Timing analysis was performed after place-and-route using the Quartus Timing Analyzer.

- **Clock domain:** Single synchronous clock
- **Operating corner:** Slow process, 1100 mV, 85°C
- **Reported metric:** Core maximum operating frequency (core Fmax)

Timing Result

Core Fmax = 104.4 MHz

This value represents the maximum frequency at which the internal pipeline registers of the processor core can operate with positive setup slack. External memory timing was not included in this measurement.

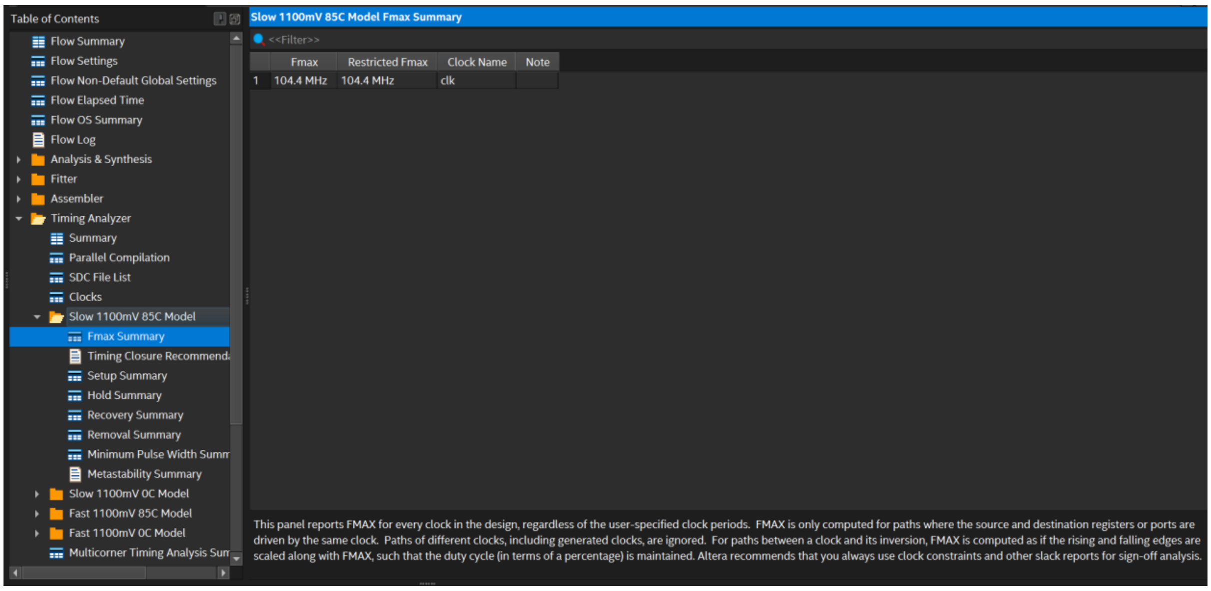


Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Assembler
- Timing Analyzer
  - Summary
  - Parallel Compilation
  - SDC File List
  - Clocks
  - Slow 1100mV 85C Model
    - Fmax Summary
    - Timing Closure Recommenda
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Width Summ
    - Metastability Summary
  - Slow 1100mV 0C Model
  - Fast 1100mV 85C Model
  - Fast 1100mV 0C Model
  - Multicorner Timing Analysis Sum

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	104.4 MHz	104.4 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera recommends that you always use clock constraints and other slack reports for sign-off analysis.

8. Resource Utilization

Post-synthesis resource utilization is summarized below:

- Logic utilization: ~508 ALMs (< 1%)
- Registers: ~578
- Block memory: < 1%
- DSP blocks: 0

The low utilization indicates a compact and efficient implementation of the processor core.

---

9. Results and Discussion

The processor achieved full functional correctness, as validated by automated trace-based verification and final state comparison. Microarchitectural optimizations enabled the design to reach a core operating frequency exceeding 100 MHz under worst-case timing conditions.

The project highlights the trade-offs between pipeline depth, hazard complexity, and FPGA resource constraints. While deeper pipelines improve frequency, they also increase design complexity and fitting difficulty on smaller devices.

---

10. Conclusion

A fully functional five-stage pipelined RV32I processor core was successfully designed, verified, and implemented on FPGA. The processor achieved correct execution of test programs and met realistic timing constraints with a core Fmax of 104.4 MHz.

This project demonstrates practical experience in CPU microarchitecture design, hazard handling, verification, and FPGA-based implementation.

---

11. Future Work

Possible extensions include:

- Instruction and data caches
- CSR and interrupt support
- Memory-mapped I/O
- External memory controllers
- Performance benchmarking with larger programs