

ECE Capstone Project Final Report

Project Title	Design, Verification, and FPGA Bring-Up of a 32-bit Multicycle RV32I RISC-V Processor
Team	Almas Adilgazyuly Amir Zhumagulov Ali Tussupov
Supervisor	Daniele Tosi
Course/Term	ENG 400 Capstone Project, Spring 2026
Date	09-04-2026

Prepared by: Almas Adilgazyuly | Amir Zhumagulov | Ali Tussupov

Reviewed by: Daniele Tosi

TABLE OF CONTENTS

0. ABET Evidence Map, Final Design Snapshot, and Design Evolution Log	4
1. Executive Summary	7
2. Problem Definition, Stakeholders, Context, and Constraints	8
2.1 Problem Definition and Need Statement	8
2.2 Stakeholders	8
2.3 Context and Operating Environment	8
2.4 Constraints and Context Factors	9
3. Final Requirements, Acceptance Criteria, and Traceability	11
3.1 Final Requirements Compliance	11
3.2 Requirements Traceability	12
3.3 Derived, Interface, Environmental, and Regulatory Requirements	13
4. Engineering Standards, Regulations, and Compliance Evidence	15
5. Concept Generation, Trade-off Study, and Selected Concept	17
5.1 Candidate Concepts	17
5.2 Weighted Trade-off Study	17
5.3 Selected Concept and Rationale	18
6. Design Iterations and Decision History	19
7. Final System Architecture	21
8. Detailed Design and Implementation	23
8.1 Hardware design	23
8.2 Software / firmware design	24
8.3 Mechanical / packaging design (if applicable)	24
8.4 Prototype realization and integration	24
8.5 Final bill of materials and design-for-manufacture / assembly notes	24
9. Verification, Validation and Experimental Results	26
9.1 Unit verification results	28
9.2 Full rv32ui regression results	28
9.3 Synthesis results	28
9.4 FPGA hardware validation	29
9.5 Limitations and engineering judgment	29
10. Risk Management, Schedule, Budget, and Procurement Outcomes	29
10.1 Final Risk Register	29
10.2 Milestones Planned vs. Actual	30
10.3 Budget, Procurement, and Substitutions	30
11. Professional Responsibilities and Broader Impacts	32

11.2 Ethical and Professional Responsibilities	32
11.3 Security and Privacy	32
11.4 Accessibility and Usability	32
11.5 Environmental, Economic, Social, Global, and Cultural Factors	32
12. Team Organization, Collaboration, and Individual Contributions	33
13. Use of Prior Course Work and New Knowledge Acquired	35
14. Conclusions and Recommendations	36
15. References	37
Appendix A. Engineering Standards Evidence	38
Appendix B. Multiple-Constraints Matrix	39
Appendix C. Team Organization and Contributions	40
Appendix D. Design Evidence	41
Appendix E. Verification Evidence	42
Appendix F. Project Management Evidence	43

0. ABET Evidence Map, Final Design Snapshot, and Design Evolution Log

This section helps assessors quickly locate evidence of engineering design, verification, teamwork, and the final outcome state. It summarizes what was built, what was verified, and how the design evolved during the capstone.

Table 0-A. Report Evidence Guide.

ABET item / SO	What must be demonstrated	Primary report section	Supporting appendix / evidence ID
SO1	Identification and solution of a complex digital design problem using engineering science and mathematics	Sections 2, 3, 8, 9	EV-RTL-01, EV-UT-01, EV-RG-01, EV-SYN-01
SO2	Engineering design under functional, cost, schedule, and hardware constraints	Sections 2, 5, 6, 10	EV-ARCH-01, EV-ARCH-02, EV-SYN-01, EV-PM-01
SO3	Clear communication to technical and mixed audiences	Sections 1, 7, 9, 14	Full report plus Appendices D-E
SO4	Ethical and professional judgment, including broader impacts	Section 11	EV-PM-01, references, open-source toolchain choices
SO5	Effective teamwork, role ownership, coordination, and objective completion	Section 12	EV-TEAM-01, EV-TEAM-02, EV-PM-01
SO6	Experimentation, analysis, interpretation, and evidence-based conclusions	Section 9	EV-UT-01, EV-RG-01, EV-RG-02, EV-SYN-01
SO7	Acquisition and application of new knowledge	Section 13	EV-UT-01, EV-RG-01, toolchain and workflow references
Criterion 5 design evidence	Standards use, multiple constraints, iteration, and prior course work	Sections 4, 5, 6, 13	Appendices A-B, EV-ARCH-01, EV-SYN-01

Table 0-B. Final Design Snapshot.

Field	Final value / summary
Final prototype / revision	Verilog RTL multicycle RV32I processor with Verilator verification flow, regression scripts, Yosys synthesis evidence, and Tang Primer 20K LED bring-up
Main need addressed	A low-cost, standards-aligned educational processor platform that goes beyond simplified single-cycle examples by demonstrating multicycle control, verification, and hardware realization
Requirements met	42/42 rv32ui regression PASS, 40/40 directed unit-test checks PASS, successful top-level synthesis to 6,447 generic cells, live FPGA LED blink demo
Primary standards applied	RISC-V RV32I instruction behavior, IEEE 1364 synthesizable Verilog practices, Verilator simulation methodology, RISC-V verification-guideline style evidence
Key constraints satisfied	Open-source toolchain, low-cost FPGA target, modular verification flow, hardware-feasible synthesis footprint, reusable regression evidence
Most important trade-off	Multicycle control was selected over pipelining to reduce hardware cost and verification complexity while still enabling full RV32I user-mode verification
Overall result	The project achieved a verified multicycle RV32I core with strong simulation evidence and a real FPGA bring-up demonstration; remaining gaps are mainly device-specific timing/power reports and broader privileged-suite evidence

Table 0-C. Design Evolution Log.

Iteration ID	What changed	Why it changed	Evidence that triggered the change	Impact on performance / cost / schedule / risk	Date
ITR-1	Selected multicycle RV32I architecture over single-cycle and pipelined alternatives	Better balance of hardware reuse, control complexity, and verification effort for a one-term capstone	Trade-off study in Section 5	Reduced integration risk and FPGA resource pressure	January 2026
ITR-2	Added Verilator plus C++ unit-test flow	Needed faster debug loops and objective PASS/FAIL evidence before full integration	Early module-debug issues in ALU, control, and memory bring-up	Reduced late-stage debugging risk	07-02-2026
ITR-3	Added directed unit tests for decoder, ALU decoder, and sequential primitives	More isolation was needed to verify critical low-level interfaces	Unit-test expansion recorded in EV-UT-01	Improved confidence in interfaces and reset/edge behavior	10-02-2026
ITR-4	Added +MEMFILE loading and clarified word-aligned	Reproducible simulation required dynamic program	Instruction-memory testing and workflow issues	Improved portability and regression repeatability	16-02-2026

Iteration ID	What changed	Why it changed	Evidence that triggered the change	Impact on performance / cost / schedule / risk	Date
	memory addressing	loading and consistent address interpretation			
ITR-5	Added CSR trap path, misaligned-access handling, and trap return support	The design needed machine-mode control flow for realistic exception behavior	Control-path integration work in rtl/CSR.v, rtl/Control_Unit.v, rtl/top.v	Increased functionality with moderate area cost	Late February 2026
ITR-6	Added regression scripts, result summaries, and synthesis documentation	Final verification needed project-wide evidence rather than isolated screenshots	EV-RG-01, EV-SYN-01, README evidence package	Improved traceability for final reporting	March 2026
ITR-7	Demonstrated CPU-driven LED blinking on the Tang Primer 20K FPGA	Real hardware execution was needed to validate that the design could run beyond simulation	Team hardware bring-up and demo observation	Added real-world validation; archival media still needs insertion into appendix	April 2026

1. Executive Summary

Open-source RISC-V has become a strong platform for computer-architecture education because it exposes the instruction set, microarchitecture, and verification process without licensing barriers. However, many educational processors stop at highly simplified single-cycle implementations and provide limited evidence beyond a few simulations. This capstone addressed that gap by building and verifying a 32-bit multicycle RV32I processor in synthesizable Verilog, then bringing the design onto a Tang Primer 20K FPGA for a visible LED-based hardware demonstration.

The engineering problem was not only to make instructions execute correctly, but to produce a design process that was testable, traceable, and realistic under capstone constraints. The final solution is a modular multicycle core composed of a top-level datapath, a 13-state control unit, ALU, ALU decoder, instruction decoder, register file, instruction/data memory model, CSR subsystem, and supporting flip-flop primitives. Verification was performed using Verilator, directed C++ unit testbenches, regression scripts, archived PASS logs, and a Yosys synthesis pass to confirm hardware feasibility.

Final performance achieved:

- 42/42 official rv32ui user-mode regression programs passed with no archived FAIL or UNDEFINED banners (EV-RG-01).
- 40/40 directed unit-test checks passed across seven RTL modules (EV-UT-01).
- The top-level design synthesized successfully to 6,447 generic cells in Yosys (EV-SYN-01).
- The processor was brought up on the Tang Primer 20K FPGA and used to blink the on-board LEDs under CPU program control (EV-FPGA-01, media pending insertion).

Requirements status:

- Met: functional RV32I user-mode execution, module-level verification, reproducible verification workflow, synthesizable top-level design, and FPGA LED bring-up.
- Partially met: formal trap/CSR compliance is implemented and exercised by the project flow, but full privileged-suite coverage is not archived in the repository.
- Not fully verified in the repository: board-specific timing closure, measured power, logic-analyzer capture, and complete rv32mi evidence.

The main limitation is not the core RTL itself, but the completeness of external hardware evidence. The next recommended step is to attach the Tang Primer 20K photo/video, any board tool reports available from the final bring-up, and then extend regression coverage to archived rv32mi or privileged-mode test evidence.

2. Problem Definition, Stakeholders, Context, and Constraints

2.1 Problem Definition and Need Statement

The project solves the problem of creating a fully understandable, standards-aligned RISC-V processor implementation that is small enough for a capstone but complete enough to demonstrate realistic engineering practice. A useful educational processor must do more than compile or blink a light. It must expose microarchitectural decisions, verification methods, exception behavior, synthesis feasibility, and the difference between a simulation-only artifact and hardware that can run on an actual FPGA.

This is an engineering design problem rather than a pure research problem because the team had to balance functional correctness, implementation time, tool availability, FPGA feasibility, documentation, and testability. The final design needed to satisfy measurable acceptance criteria, operate within budget and schedule, and produce traceable evidence for both simulation and hardware bring-up.

2.2 Stakeholders

The project serves the capstone team, the supervising faculty, the ECE department, future students who may reuse the workflow, and the wider open-source RISC-V education community.

Table 2-A. Stakeholder Analysis.

Stakeholder	Role/Interest	Needs/Success criteria	How engaged	Impact of project
Capstone team	Designers, verifiers, and integrators	A working and understandable RV32I processor plus credible evidence for course assessment	Weekly team coordination, code ownership, simulation, synthesis, and FPGA testing	Direct effect on academic success and technical skill development
Project supervisor	Technical and academic oversight	Sound architecture, traceable verification, and defensible conclusions	Progress review, report review, milestone guidance	Ensures quality and alignment with capstone outcomes
ECE department	Assessment and curriculum stakeholder	Visible application of digital design, architecture, verification, and teamwork	Course milestones and final report evaluation	Demonstrates program outcome attainment
Future students	Secondary users of the repository and report	Reproducible workflow and readable documentation	Indirect through documentation quality and artifact reuse	Increases educational value of the project
Open-source RISC-V ecosystem	Indirect reference stakeholder	Standards-aligned design practice using open tools	Indirect through tool and test selection	Supports accessible engineering education

2.3 Context and Operating Environment

The project was developed in an academic capstone environment with limited time, limited hardware budget, and a strong need for demonstrable evidence. The technical environment consisted of synthesizable Verilog RTL, Verilator-based simulation, C++ unit testbenches, a Python dump-to-hex converter, shell regression scripts, GTKWave-compatible waveform generation, and Yosys-based synthesis reporting.

The design target was a Tang Primer 20K FPGA development board for final bring-up. This made the operating environment low voltage, USB powered, and suitable for safe laboratory demonstration. The board was used as a real-world validation platform rather than as the primary source of quantitative timing or power data because the repository currently archives simulation and synthesis artifacts more completely than board-tool outputs.

The project assumptions were:

- The instructional goal favored architectural clarity and verification rigor over maximum throughput.
- Open-source tools were preferred whenever practical.
- A multicycle control structure was acceptable if it reduced hardware duplication and debugging complexity.
- The final submission would include a board photo or short video to document the LED demonstration.

2.4 Constraints and Context Factors

Table 2-B. Constraints Matrix.

Constraint category	Metric / target	Final estimate or measurement	Gap / residual risk	Owner	Evidence / next action
Functional correctness	All archived rv32ui tests pass with no FAIL/UNDEFINED banners	42/42 PASS in results/summary.csv	Full privileged-suite evidence not archived	Team	EV-RG-01; extend archived suites if needed
Verification rigor	100% pass on directed unit tests for critical RTL blocks	40/40 checks passed across 7 modules	Directed tests do not replace exhaustive formal proof	Team	EV-UT-01; add more scenario tests if scope expands
Cost	Use low-cost board and open-source tools	BOM remains limited to board, cables, and jumpers; major tools are free	No issue	Team	Section 10.3 and Appendix F
Schedule	Completion within one academic term	Final regression, synthesis, and hardware demo achieved within Spring 2026	Exact PM timestamps should be added if required by instructor	Team	Section 10.2
FPGA feasibility	Top-level synthesis should show hardware-realizable footprint	6,447 generic cells from Yosys	No archived device-specific place-and-route report	Team	EV-SYN-01; add Gowin report if available
Hardware observability	Real-world output visible on board LEDs	CPU-driven LED blinking demonstrated on Tang Primer 20K	Repo does not yet include media attachment	Team	Insert EV-FPGA-01 photo/video in Appendix D/E
Safety and welfare	Safe low-voltage demonstration with no hazardous actuation	Development used low-voltage FPGA board and software tools only	Usual lab ESD and USB-power precautions remain	Team	Section 11.1

3. Final Requirements, Acceptance Criteria, and Traceability

3.1 Final Requirements Compliance

The final requirements below are evidence-based end-state statements rather than future plans. Each requirement maps to a verification method, a target threshold, an observed result, and an evidence ID.

Table 3-A. Final Requirements Compliance Matrix.

Req. ID	Requirement statement	Source	Verification method	Target threshold	Measured result	Verdict	Evidence ID
FR-1	The processor shall execute the archived official rv32ui user-mode programs with no stored FAIL or UNDEFINED result banners.	Project goals, RISC-V ISA compliance intent	Regression log review	42/42 PASS	42/42 PASS in results/summary.csv	Met	EV-RG-01
FR-2	The critical RTL modules shall pass directed unit tests with 100% success.	Verification plan	Unit-test report review	40/40 checks passed	40/40 checks passed across ALU, control, memory, decoder, ALUdec, dff_clk, dff_en	Met	EV-UT-01
FR-3	The design shall implement machine-mode trap storage and control signals for mtvec, mepc, mcause, mstatus, trap entry, and mret return.	Architecture and exception-handling goals	RTL inspection plus regression-context review	CSR registers and trap path present in top-level design	Implemented in rtl/CSR.v, rtl/Control_Unit.v, and rtl/top.v	Met for implemented design	EV-RTL-01
FR-4	The verification workflow shall support deterministic test selection, dump-to-hex conversion, build, execution, and archived result files.	Capstone reproducibility goal	Workflow inspection	End-to-end scripted flow present	Makefile, converter.py, run_regression.sh, and 42 archived hex/result pairs are present	Met	EV-FLOW-01
FR-5	The complete top-level processor shall synthesize successfully into a hardware-feasible FPGA-oriented netlist.	Hardware bring-up objective	Yosys synthesis report review	Successful top-level synthesis with reported cell count	Successful synthesis to 6,447 generic cells	Met	EV-SYN-01
FR-6	The design shall support visible hardware bring-up on the Tang Primer 20K	Hardware validation goal	Live hardware demonstration	On-board LEDs visibly toggle under	Team reports successful CPU-driven LED blink	Met, archival media pending insertion	EV-FPGA-01

	through CPU-controlled LED output.			processor control	on Tang Primer 20K		
FR-7	The design shall reset into a known state suitable for deterministic operation.	Reliability requirement	Unit tests and testbench reset review	Reset path present and checked in directed tests	Reset semantics validated in dff_clk, dff_en, control reset checks, and testbench reset sequence	Met	EV-UT-01, EV-FLOW-01
FR-8	The final evidence package shall include board-specific timing and power reports for the Tang Primer 20K.	Original progress goals	Evidence review	Archived board-tool timing and power reports	Not present in current repository snapshot	Not fully verified	None archived

3.2 Requirements Traceability

The traceability matrix shows how stakeholder needs were translated into measurable requirements, constrained by standards and project limits, and finally supported by design features and verification results.

Table 3-B. Requirements Traceability Matrix.

Stakeholder need	Requirement(s)	Constraint(s)	Standard clause(s)	Design feature(s)	Test ID(s)	Result section
Working educational processor with real execution evidence	FR-1, FR-5, FR-6	Functional correctness, FPGA feasibility, hardware observability	RV32I instruction behavior, IEEE 1364 synthesizability	Multicycle datapath, FSM control, LED demo	T1, T5, T7	Sections 7, 8, 9
Trustworthy verification results	FR-2, FR-4, FR-7	Verification rigor, schedule	Verilator methodology, verification-guideline evidence	Directed unit tests, scripted flow, archived logs	T2, T4, T6	Sections 8, 9
Trap-capable architecture	FR-3	Functional correctness	RISC-V machine CSR behavior	CSR.v, trap entry/return path, misalignment detection	T3	Sections 8, 9
Affordable capstone implementation	FR-5, FR-6	Cost, schedule	Open-source tool use	Low-cost FPGA target, open tools	T5, T7	Sections 5, 10
Safe and accessible	FR-6, FR-7	Safety, welfare, usability	Low-voltage lab practice, readable	Reset logic, LED output,	T4, T7	Sections 11, 14

Stakeholder need	Requirement(s)	Constraint(s)	Standard clause(s)	Design feature(s)	Test ID(s)	Result section
educational platform			documentation	reusable report		

3.3 Derived, Interface, Environmental, and Regulatory Requirements

Derived requirements:

This project was low risk from a physical-safety perspective because it used a low-voltage development board and did not drive motors, high-current loads, or hazardous equipment. The main safety considerations were USB power integrity, preventing shorts during board handling, and following ESD-safe lab practice. The final demonstration used visible LEDs as a non-hazardous output, which is appropriate for an educational processor prototype.

- The simulation environment must support +MEMFILE=<path> so test programs can be loaded without editing RTL between runs.
- The trap path must retain a saved PC and exception cause so mepc and mcause can support trap entry and return.
- The regression process must archive each selected test as both an instruction image and a corresponding result log.

Interface requirements:

- The core-to-memory interface uses address, write data, read data, and write enable signals in a simple SRAM-like style.
- The design requires a single clock and reset input for global sequencing.
- The Verilator flow requires DPI-compatible PASS/FAIL monitoring through a0 and mcause.
- The FPGA bring-up requires a visible LED output path driven by processor behavior.

Environmental requirements:

- Development and demonstration occur in a low-voltage indoor laboratory setting on a USB-powered FPGA board.
- The design assumes normal classroom/lab ambient conditions and standard ESD-safe handling.

Regulatory and safety requirements:

- RTL coding follows synthesizable IEEE 1364-style constructs.
- Instruction behavior is based on the RV32I base integer model used by the archived regression suite.
- The board demonstration must remain within the normal safe operating conditions of the development board and USB power source.

4. Engineering Standards, Regulations, and Compliance Evidence

Only standards and references that directly affected requirements, RTL implementation, synthesis, or verification were retained in the final report. Progress-stage references that did not support a final design or test claim were not carried forward as final evidence.

Table 4-A. Standards Application Matrix.

ID	Standard / regulation	Clause used	Why applicable	Design decision affected	Verification / inspection method	Evidence produced	Compliance status
STD-1	RISC-V RV32I base instruction behavior	Arithmetic, logical, branch, jump, load/store, and immediate semantics used by the rv32ui suite	Defines what the processor must execute correctly	ALU op set, decoder behavior, control sequencing, memory access rules	rv32ui regression review and RTL inspection	EV-RG-01, EV-RG-02, EV-RTL-01	Applied
STD-2	RISC-V machine CSR and trap concepts as implemented in project scope	mtvec, mepc, mcause, mstatus, trap entry, mret	Needed for ECALL/E BREAK and trap-aware control flow in this design	CSR module contents, trap-entry state, trap return path	RTL inspection and project result review	EV-RTL-01	Applied within implemented scope
STD-3	IEEE 1364-2005 synthesizable Verilog practice	Combinational always @(), edge-triggered sequential logic, reset structure, avoid unintended latches	Ensures synthesizability and maintainability	FSM coding style, ALU logic, decoder logic, storage elements	Yosys synthesis plus code review	EV-SYN-01, RTL files	Applied
STD-4	Verilator simulation methodology	Cycle-accurate C++ model, DPI-C integration, waveform tracing	Defines how simulation evidence is generated	Testbench design, PASS/FAIL hook, waveform generation	Build/run workflow review and archived logs	EV-FLOW-01, EV-RG-02	Applied
STD-5	RISC-V verification-guideline style evidence	Directed tests, regression evidence, traceable logs, artifact-based review	Supports credible capstone verification claims	Unit-test organization, summary artifacts, regression scripting	Unit-test report and regression archive review	EV-UT-01, EV-RG-01	Applied

5. Concept Generation, Trade-off Study, and Selected Concept

5.1 Candidate Concepts

Table 5-A. Candidate Concepts.

Concept	Description	Strengths	Weaknesses	References / evidence
Single-cycle RV32I processor	Every instruction completes in one long clock cycle using more duplicated datapath behavior	Conceptually simple, straightforward control	Long critical path, weaker hardware reuse, less realistic for resource-constrained FPGA target	Computer architecture texts, early project study
Multicycle RV32I processor	A compact datapath reuses the ALU and sequences work over multiple states	Lower area, better hardware reuse, easier capstone-scale verification than pipelining	Lower throughput than pipelining and more control states than single-cycle	Selected project architecture, Sections 7-9
Five-stage pipelined RV32I processor	Overlapped IF/ID/EX/MEM/WB execution with higher throughput potential	Better theoretical throughput and extensibility	Hazard handling, forwarding, control complexity, and verification burden exceed project constraints	Trade-off study, schedule and verification limits

5.2 Weighted Trade-off Study

Table 5-B. Weighted Trade-off Matrix

Criterion	Metric / target	Weight	Single-cycle	Multicycle	Pipelined	Evidence source
FPGA resource usage	Minimize logic cost	0.25	6	9	5	Architecture reasoning, Yosys feasibility goal
Clock feasibility	Reach practical FPGA clock rates without excessive control burden	0.20	5	9	7	Control-path complexity and area trade-off
Power / efficiency	Favor lower switching activity and reused hardware	0.15	6	9	6	Multicycle hardware reuse rationale
Development time and verification effort	Finish within capstone	0.25	8	9	4	Unit-test and regression strategy

	term with strong evidence					
Scalability for future work	Leave room for later extension	0.15	6	8	9	Architecture comparison
Total		1.00	6.35	8.70	6.00	

Table 5-C. Bring-Up Platform Comparison.

Platform	Benefit	Limitation	Outcome
ASIC	Most realistic final silicon target	High cost, long turnaround, cannot patch bugs after fabrication	Rejected for capstone scope
CPLD	Simple and non-volatile	Lower logic density and memory suitability for this processor project	Rejected
Commercial hardware emulator	Fast hardware-style validation	Too expensive for project budget	Rejected
FPGA	Reprogrammable, low-cost, supports real I/O and fast iteration	Requires board-specific constraints and tool outputs for full quantitative closure	Selected for real-world testing

5.3 Selected Concept and Rationale

The multicycle RV32I processor was selected because it offered the best compromise among area, verification effort, and educational value. It allowed the ALU to be reused for PC updates, address calculation, and arithmetic logic, which kept the design compact enough for a low-cost FPGA target. At the same time, it avoided the forwarding, stalling, and branch-hazard logic that would have significantly increased the verification burden of a pipelined design.

The team also selected FPGA validation as the real-world bring-up platform because it enabled a practical bridge between simulation and hardware. Unlike ASICs, the FPGA allowed rapid changes and repeated testing. Unlike purely software simulation, it provided visible confirmation that the processor could drive external hardware behavior.

6. Design Iterations and Decision History

The final solution emerged through staged refinement, targeted debugging, and evidence-driven workflow changes. Architectural changes and process improvements were adopted only when they reduced risk or improved the strength of final evidence.

Table 6-A. Major Design Iterations

Iteration / version	Issue discovered	Alternatives considered	Decision made	Trade-off accepted	Evidence of improvement
V1 - baseline RTL	Need for a realistic but manageable processor architecture	Single-cycle, multicycle, pipelined	Selected multicycle architecture	Lower throughput than pipelining	Trade-off totals in Section 5
V2 - simulation bring-up	Ad hoc simulation gave weak evidence and slow debug	Manual wave inspection only vs scripted flow	Added Verilator plus C++ testbench and archived logs	Extra tooling work early in project	EV-FLOW-01, EV-RG-02
V3 - module isolation	Integration debugging risk was too high	Only full-system tests vs isolated module tests	Added directed unit tests for seven modules	More up-front verification effort	EV-UT-01
V4 - memory portability	Hard-coded program handling reduced repeatability	Static memory contents vs dynamic image loading	Added +MEMFILE and converter-based program flow	Workflow complexity increased slightly	EV-FLOW-01, EV-UT-01
V5 - trap support	Basic core needed more realistic control flow	No trap path vs CSR-backed trap path	Added CSR module, trap entry, misalignment handling, and mret support	Added RTL area and control complexity	EV-RTL-01, EV-SYN-01
V6 - project-level evidence	Isolated tests were not enough for final-report claims	Manual reruns vs archived regression summary	Added regression scripts, summary CSV, synthesis documentation, and figures	More documentation work	EV-RG-01, EV-SYN-01, EV-ARCH-01, EV-ARCH-02
V7 - FPGA bring-up	Simulation alone was insufficient for real-world validation	Stop at synthesis vs demonstrate on board	Used Tang Primer 20K to run CPU-driven LED blink	Board-specific archival evidence still needs insertion	EV-FPGA-01

7. Final System Architecture

The final system architecture links the RTL processor core, the instruction-image preparation flow, the Verilator/DPI verification environment, and the Tang Primer 20K hardware bring-up path.

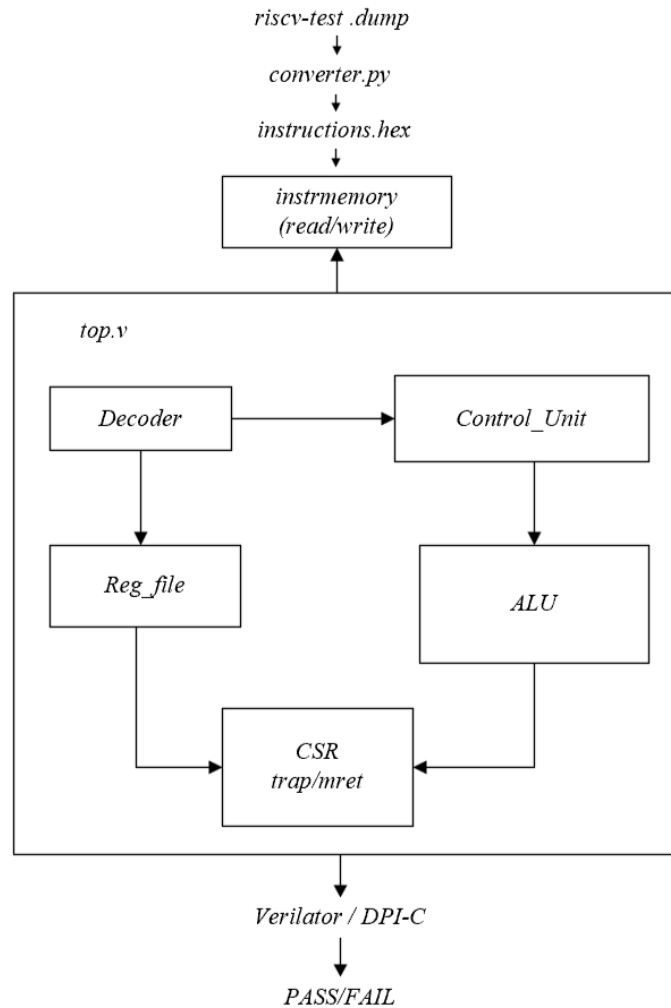


Figure 1. Final system block diagram

Final hardware bring-up path:

Top-level FPGA integration -> Tang Primer 20K LEDs

Table 7-A. Interface Summary.

Interface	Direction	Protocol / signal type	Rate / timing	Voltage / power	Owner subsystem	Notes
Core to memory	Bidirectional	Simple 32-bit address/data interface with write enable	One memory access per active state; multicycle sequencing	Internal FPGA logic domain	top.v and instrmemory.v	Unified instruction/data-style memory model in current design
Clock / reset to core	Input	Global clock plus reset	Global sequencing for states and storage elements	FPGA board supply domain	Top-level integration	Reset is applied at simulation start and in hardware bring-up
CSR / trap path	Internal bidirectional	CSR read/write plus trap-entry and mret control signals	Updated on trap events and return events	Internal FPGA logic domain	CSR.v and Control_Unit.v	Stores mtime, mepc, mcause, mstatus
Verilator / DPI verification interface	Bidirectional	Verilator model, DPI-C check() function, waveform tracing	Cycle-accurate simulation; archived results per test	Host computer only	Testbench and DPI layer	Produces PASS/FAIL/UNDEFINED banners and waveform.vcd
LED output interface	Output	FPGA on-board LED drive from processor-controlled logic	Human-visible blink/toggle behavior	Tang Primer 20K user-LED domain	FPGA integration wrapper	Used for final hardware bring-up demonstration

Timing / latency budget: The processor uses a 13-state multicycle FSM, so correctness depends on sequenced control behavior rather than aggressive single-cycle timing. Strong functional evidence is archived, but no final vendor-specific place-and-route timing report is present.

Power / resource budget: The Yosys result of 6,447 generic cells demonstrates a hardware-feasible footprint for a low-cost FPGA-oriented prototype. Final board-specific power estimation or measurement was not archived.

8. Detailed Design and Implementation

This section documents the final design from four viewpoints: hardware architecture, supporting verification software, physical prototype context, and staged realization/integration.

8.1 Hardware design

The processor is organized as a modular multicycle datapath with a centralized control FSM. The top-level module integrates:

- top.v: datapath integration, muxing, register staging, PC update, memory interface, and module interconnect
- Control_Unit.v: 13-state controller (s0 to s12) that sequences fetch, decode, memory, ALU writeback, branch, trap entry, and trap return
- ALU.v: arithmetic, logical, shift, and compare operations
- ALUdec.v: operation decode from instruction function fields and control state
- decoder.v: register-field extraction and immediate generation
- Reg_file.v: 32 x 32-bit register file
- instrmemory.v: simulation memory model with program loading and write support
- CSR.v: machine-mode trap storage and return support
- dff_clk.v and dff_en.v: sequential primitives for staging and gated updates

Key hardware choices:

- The ALU is reused for both arithmetic/logic operations and address/PC-related computation, which reduces hardware duplication.
- The controller uses explicit states rather than a deeply nested purely combinational controller, which makes waveform interpretation and debugging easier.
- Trap support was implemented through CSRs and dedicated control states rather than leaving exceptions entirely outside the architecture.
- The design uses stored intermediate registers to separate state transitions and datapath updates.

Module-level observations from synthesis:

- Total synthesized cells: 6,447
- Register file: 3,167 cells
- ALU: 1,461 cells
- CSR: 489 cells
- Control unit including ALU decoder: 282 + 35 cells
- Decoder: 69 cells

The register file dominates the area because dual-read register access requires substantial multiplexing compared with the control logic itself. This is expected for a compact educational core.

8.2 Software / firmware design

Although the processor itself is hardware, the project depended on a supporting software verification stack:

- `test/converter.py` converts riscv-tests dump files into hex images used by the memory model.
- Makefile orchestrates compilation, Verilator elaboration, simulation execution, and result capture.
- `test/tb/testbench.cpp` drives reset, clock toggling, and waveform dumping.
- `test/tb/dpi_functions.c` inspects `a0` and `mcause` to print PASS, FAIL, or UNDEFINED.
- `scripts/run_regression.sh` automates a sweep of ISA tests.
- `scripts/summarize_results.py` converts archived result files into Markdown or CSV summaries.

This software layer was essential to the project because it transformed the hardware from a hand-driven experiment into a reproducible engineering workflow.

8.3 Mechanical / packaging design (if applicable)

No custom enclosure, PCB, or mechanical package was required. The final physical prototype used the Tang Primer 20K development board as the hardware platform. Mechanical design considerations were therefore limited to safe lab handling, stable USB power delivery, and clear visibility of the on-board LEDs during the demonstration.

8.4 Prototype realization and integration

Prototype realization occurred in four layers:

RTL modules were implemented and tested independently.

The modules were integrated into `top.v` and exercised through Verilator simulation.

Regression and synthesis artifacts were generated to build objective evidence.

The design was brought onto the Tang Primer 20K FPGA and used to blink the on-board LEDs under CPU control.

The integration strategy favored staged risk reduction. Instead of immediately relying on the full core, the team first verified arithmetic, control, memory, decode, and sequential behavior in isolation. Only then were project-wide regression and synthesis evidence collected. This sequencing reduced the chance that a late-stage failure would be impossible to localize.

8.5 Final bill of materials and design-for-manufacture / assembly notes

The final bill of materials remained intentionally small because the project used a reusable development board and open-source tools rather than custom hardware fabrication. Assembly considerations were limited to safe USB power, normal lab handling, and optional jumper access for future GPIO work.

Table 8-A. Bill of Materials

Part No.	Description	Qty	Unit cost	Total	Supplier / source	Status
FPGA-1	Tang Primer 20K FPGA development board	1	25,000 KZT	25,000 KZT	YX Electronic Components	Procured
USB-1	USB-C cable for programming and power	2	2,000 KZT	4,000 KZT	YX Electronic Components	Procured
ACC-1	Breadboard wires / jumpers for optional GPIO work	1	2,500 KZT	2,500 KZT	YX Electronic Components	Procured
TOOL-1	Verilator, Yosys, GNU RISC-V toolchain, Python, make	-	0 KZT	0 KZT	Open-source	In use

9. Verification, Validation and Experimental Results

The final V&V section combines requirement-level results, a test-campaign summary, and focused interpretation of the unit-test campaign, the full rv32ui regression archive, synthesis evidence, and FPGA hardware validation.

Table 9-A. Requirements-to-Test Results Matrix.

Req ID	Requirement	Test ID	Verification method	Equipment / tools and accuracy	Target	Measured result	Verdict	Evidence ID
FR-1	rv32ui user-mode correctness	T1	Archived regression review	Verilator, riscv-tests, converter, shell scripts	42/42 PASS	42/42 PASS	Met	EV-RG-01
FR-2	Unit-level correctness	T2	Directed unit-test campaign	Verilator, C++ testbenches, VCD traces	40/40 checks PASS	40/40 PASS	Met	EV-UT-01
FR-3	CSR and trap-path implementation	T3	RTL inspection plus result-context review	RTL source, regression artifacts	CSR registers and trap path present	Implemented and integrated	Met for implemented scope	EV-RTL-01
FR-4	Reproducible flow	T4	Workflow inspection	Makefile, scripts, archived outputs	End-to-end deterministic flow	Present and reusable	Met	EV-FLOW-01
FR-5	Synthesis feasibility	T5	Synthesis report review	Yosys	Successful top-level synthesis	6,447 cells	Met	EV-SYN-01
FR-6	Tang Primer LED demo	T6	Hardware demonstration	Tang Primer 20K board, user LED output	Visible CPU-driven blink	Demonstrated	Met, media pending insertion	EV-FPGA-01
FR-7	Deterministic reset behavior	T7	Unit-test and reset-flow review	Unit tests, testbench reset logic	Known reset state	Reset semantics verified in directed tests and testbench	Met	EV-UT-01, EV-FLOW-01

Table 9-B. Test Campaign Summary.

Test / experiment	Date	Owner(s)	Setup	No. of trials	Key result	Observed issues	Follow-up action
T1 rv32ui regression archive	Spring 2026	Team	Verilator plus riscv-tests hex flow	42 archived programs	42/42 PASS	Archived cycle count is fixed by current testbench bound, so it is not a benchmark metric	Keep using PASS/FAIL result rather than cycle value for performance claims
T2 directed unit tests	Spring 2026	Team	Module-local Makefiles and C++ testbenches	40 directed checks	40/40 PASS	Directed coverage is strong but not exhaustive	Expand only if future scope increases
T3 trap/CSR integration review	Spring 2026	Team	Review of CSR.v, Control_Unit.v, top.v, plus result flow	Design inspection	Trap entry, mret, and CSR registers are integrated	Full privileged-suite archive not present	Add archived privileged tests later
T5 synthesis run	Spring 2026	Team	Yosys top-level synthesis	1 archived run	6,447 generic cells	No archived target-specific place-and-route report	Add Gowin report if available
T6 FPGA LED bring-up	April 2026	Team	Tang Primer 20K hardware demo	Demonstration event	CPU-driven LED blinking observed	Repo does not yet contain photo/video or board reports	Insert EV-FPGA-01 media in appendix

9.1 Unit verification results

The unit-test campaign verified seven RTL modules in isolation:

- ALU
- Control Unit
- Instruction memory
- Decoder
- ALU decoder
- dff_clk
- dff_en

The archived summary reports 40/40 checks passed. This matters because it confirms not only arithmetic and decode logic, but also the small sequential behaviors that often cause integration bugs in multicycle designs. The strongest directed coverage appears in the ALU decoder and decoder blocks, where control-field interpretation is most branch-heavy.

9.2 Full rv32ui regression results

The archived regression summary in results/summary.csv shows 42 result files and 42 matching program images, with every result marked PASS. This is the strongest direct evidence that the integrated processor correctly executes the user-mode instruction subset exercised by the project.

Important interpretation note:

- Every archived result log ends with Simulation complete. Total cycles: 5000.
- This value comes from the current testbench running to a fixed simulation bound (MAX_SIM_TIME = 10000, which corresponds to 5000 clock cycles), not from early program completion timing.
- Therefore, the archived cycle field is useful as a consistency record but not as a meaningful performance benchmark.

9.3 Synthesis results

The Yosys report shows a successful top-level synthesis to 6,447 generic cells. The main hierarchy breakdown is:

- 3,167 cells in the register file
- 1,461 cells in the ALU
- 489 cells in the CSR block
- 282 cells in the control unit
- 35 cells in the ALU decoder
- 69 cells in the instruction decoder

This confirms that the design is hardware-feasible and that the resource-intensive parts are where expected: storage and datapath, not control. The result supports the claim that the processor is suitable for low-cost FPGA-oriented realization, even though the repository does not archive a final vendor-specific place-and-route report.

9.4 FPGA hardware validation

The final hardware validation step was a Tang Primer 20K bring-up in which a RISC-V program running on the processor visibly blinked the on-board LEDs. This is important because it demonstrates that the project progressed beyond a simulation-only model and achieved observable physical execution.

Current evidence status for hardware validation:

- The team reports the demonstration as completed.
- The final report reserves EV-FPGA-01 for the board photo or short video plus a short written description.
- The repository snapshot used for this report does not yet contain the media file itself, so the appendix entry is marked as pending insertion.

9.5 Limitations and engineering judgment

The final evidence package is strong on simulation and synthesis, but several claims were intentionally left conservative:

- No archived Tang Primer 20K timing-closure report is present, so frequency claims are not treated as final verified results.
- No archived board power estimate is present, so power-budget claims remain unverified in the repository.
- No archived logic-analyzer capture is present, so internal FPGA debug visibility is not claimed as final evidence.
- Full privileged-suite or rv32mi archival evidence is not present, so exception-handling claims remain limited to the implemented project scope and available artifacts.

These limitations do not invalidate the project result. They define the boundary between what has been demonstrated and what should be added for a stronger final evidence package.

10. Risk Management, Schedule, Budget, and Procurement Outcomes

10.1 Final Risk Register

Table 10-A. Risk Register with Outcomes.

ID	Risk	L	I	RPN	Mitigation	Did it occur?	How handled	Residual risk / limitation
RS K-1	Late RTL logic bug in ALU, decoder, or FSM	4	5	20	Unit-test early and isolate modules	Yes, as normal debug pressure	Added directed unit tests and waveform-backed debug	Deep corner cases beyond current suite could still exist
RS K-2	Toolchain mismatch or build instability	3	4	12	Keep simple open-source flow and reusable scripts	Minor workflow friction occurred	Standardized Makefile and converter-based flow	Vendor-specific hardware-report flow is not archived

ID	Risk	L	I	RPN	Mitigation	Did it occur?	How handled	Residual risk / limitation
RS K-3	Incomplete ISA / trap coverage	4	4	16	Use rv32ui suite and trap-aware design	Partially	Achieved full archived rv32ui PASS and implemented trap path, but broader privileged evidence remains limited	Full rv32mi archive still recommended
RS K-4	Long debugging cycles caused by multicycle control complexity	3	5	15	Expose state logic and keep modular design	Yes	FSM documentation, unit tests, and staged integration reduced the problem	Board-level debug artifacts are still limited

[RPN = Risk Priority Number ($L \times I$) $\rightarrow$ higher = more important to manage early]

10.2 Milestones Planned vs. Actual

Table 10-B. Milestones and Completion Evidence.

Milestone	Planned date	Actual date	Owner	Exit criteria	Status	Evidence
M1: Architecture and requirements finalization	Week 3	January 2026	Almas	Baseline architecture and measurable requirements defined	Completed	Sections 2-5, progress-report baseline
M2: Initial RTL and simulation bring-up	Week 6	January-February 2026	Amir, Ali	Core modules compile and run basic simulation correctly	Completed	EV-UT-01, EV-FLOW-01
M3: Full functional verification and synthesis review	Week 10	Spring 2026	Team	rv32ui regression archive collected and synthesis evidence generated	Completed	EV-RG-01, EV-SYN-01
M4: Hardware demonstration and final report	Week 14	April 2026	Team	FPGA demo completed and final report assembled	Completed	EV-FPGA-01, this report

10.3 Budget, Procurement, and Substitutions

Table 10-C. Bill of Materials and Procurement Summary.

Part No.	Description	Qty	Unit cost	Total	Alt. part / substitution	Supplier / lead time	Impact of substitution
FPGA-1	Tang Primer 20K FPGA development board	1	25,000 KZT	25,000 KZT	None recorded	YX Electronic Components	None
USB-1	USB-C cable	2	2,000 KZT	4,000 KZT	Generic equivalent acceptable	YX Electronic Components	None
ACC-1	Jumpers / optional GPIO accessories	1	2,500 KZT	2,500 KZT	Generic equivalent acceptable	YX Electronic Components	None
TOOL-1	Open-source EDA and build tools	-	0 KZT	0 KZT	N/A	Open-source	No cost impact

Budget status vs. plan:

- The project remained within the expected budget.
- The largest physical expense was the FPGA board.
- Open-source tooling kept software cost at zero.

Change-control summary:

- The major project changes were architectural and verification-process improvements rather than procurement-driven substitutions.
- This kept the cost stable while still improving the quality of evidence.

11. Professional Responsibilities and Broader Impacts

11.1 Public Health, Safety, and Welfare

This project was low risk from a physical-safety perspective because it used a low-voltage development board and did not drive motors, high-current loads, or hazardous equipment. The main safety considerations were USB power integrity, preventing shorts during board handling, and following ESD-safe lab practice. The final demonstration used visible LEDs as a non-hazardous output, which is appropriate for an educational processor prototype.

11.2 Ethical and Professional Responsibilities

The project was performed for educational and research purposes only. It was not represented as suitable for safety-critical, medical, or security-sensitive deployment. The team relied heavily on open-source tools and test suites and therefore had an obligation to attribute those sources properly and to distinguish between verified claims and assumptions.

AI-assisted drafting and editing tools may be used to improve report structure or prose quality, but the engineering responsibility remains with the team. All technical claims in this report were checked against local project artifacts before inclusion. This distinction is important for academic integrity and professional accountability.

11.3 Security and Privacy

The project handled no personal data and was not deployed as a connected product. Security considerations therefore focused mainly on protecting source files, build artifacts, and backups. The repository-centered workflow supports reproducibility and reduces the risk of undocumented changes. Since the design is educational and local, privacy concerns were minimal.

11.4 Accessibility and Usability

The final report is structured for export to searchable, selectable text. Figures are named explicitly, evidence IDs are used throughout, and tables are intended to make the report easier to review by different audiences. The LED demonstration also provided a simple and accessible hardware output that could be understood without specialized instrumentation.

11.5 Environmental, Economic, Social, Global, and Cultural Factors

RISC-V and open-source EDA tools lower economic barriers to processor education. This makes the project more globally accessible than a proprietary-tool or proprietary-ISA workflow. Environmentally, the project avoided fabrication-intensive prototype paths by using simulation, synthesis, and a reusable FPGA board. Socially, the repository and final report can be reused by future students who want an example of how to combine architecture design with verifiable evidence.

12. Team Organization, Collaboration, and Individual Contributions

This section provides direct evidence of team organization, role ownership, collaboration, and individual technical contributions in the final capstone work.

Table 12-A. Team Roles and Responsibilities.

Member	Primary role	Secondary role	Major technical ownership	Leadership responsibility	Evidence ID
Almas Adilgazyuly	Verification workflow and documentation	ALU design/verification, build environment	ALU verification, build/simulation flow, final documentation support	Coordinating reproducible evidence package	EV-TEAM-01
Amir Zhumagulov	Control-path development	Architecture integration	Control unit, FSM behavior, trap/control logic support	Control-path review and system integration support	EV-TEAM-01
Ali Tussupov	RTL module verification and memory-related work	FPGA integration support	Instruction memory, module verification expansion, regression/support work	Verification expansion and hardware support	EV-TEAM-01

Table 12-B. Teamwork Evidence Matrix.

SO5 element	Project-specific evidence	Where shown in report / appendix	What changed because of team interaction
Leadership	Clear module ownership and milestone completion	Sections 10 and 12	Work was split into manageable technical areas rather than handled ad hoc
Collaborative environment	Shared verification artifacts and common report structure	Sections 9 and 12	Evidence became easier to review and combine
Goals	Common focus on a verified multicycle core plus FPGA demo	Sections 1, 3, 14	The team prioritized measurable results over feature creep
Task planning	Milestones, risk register, and staged integration	Section 10, Appendix F	Verification was performed earlier, reducing late risk
Objectives met	Regression PASS archive, synthesis report, and hardware bring-up	Sections 9 and 14	The project reached a more complete finish than a simulation-only core

Table 12-C. Individual Contribution Log.

Member	Key tasks completed	Deliverables produced	Dates / period	Supervisor verification / evidence
Almas Adilgazyuly	Built ALU verification flow, supported build environment, contributed to documentation and synthesis/report packaging	ALU artifacts, workflow artifacts, documentation contributions	Spring 2026	Progress dashboard ownership, EV-FLOW-01

Member	Key tasks completed	Deliverables produced	Dates / period	Supervisor verification / evidence
Amir Zhumagulov	Implemented and verified control-related behavior, supported integration of multicycle sequencing and trap flow	Control-unit artifacts and integration support	Spring 2026	Progress dashboard ownership, EV-RTL-01
Ali Tussupov	Supported instruction-memory work, broader RTL verification, and supporting test evidence	Instruction-memory artifacts, module verification, support evidence	Spring 2026	Progress dashboard ownership, EV-UT-01

Team collaboration summary:

- The team used module ownership to reduce overlap while keeping integration decisions shared.
- Meetings and reviews were used to align on requirements, debugging priorities, and final evidence.
- When uncertainty existed, the preferred next step was to run a test, inspect a waveform, or archive a result rather than argue from assumption.

13. Use of Prior Course Work and New Knowledge Acquired

Table 13-A. Prior Course Work Applied.

Design task	Prior course / topic used	How it was applied	Evidence in design
FSM-based controller design	Digital logic and finite-state-machine design	Used to structure the multicycle control states and transitions	Control_Unit.v, EV-ARCH-02
ALU and datapath design	Computer organization and architecture	Applied to arithmetic, logic, and branch-support datapath behavior	ALU.v, top.v
Register and reset behavior	Sequential logic design	Applied to staged registers and reset semantics	dff_clk.v, dff_en.v, EV-UT-01
Verilog implementation	HDL and FPGA-oriented design practice	Applied to synthesizable module structure and integration	RTL directory, EV-SYN-01
C/C++ programming	Systems programming	Applied to Verilator testbenches and DPI interaction	test/tb/testbench.cpp , test/tb/dpi_functions.c

Table 13-B. New Knowledge Acquired and Applied.

Knowledge / skill learned	Why it was needed	Source used	How it changed the design or test process
Verilator-based module verification	Needed faster iteration and objective simulation evidence	Verilator workflow and project experimentation	Enabled reusable directed unit tests
DPI-C based PASS/FAIL reporting	Needed automated result classification instead of manual waveform-only review	Project experimentation with Verilator DPI	Simplified regression review
riscv-tests conversion and regression management	Needed realistic program images and repeatable suites	riscv-tests integration and converter flow	Enabled the archived rv32ui PASS package
Yosys synthesis reporting	Needed hardware-feasibility evidence	Yosys workflow and synthesis script use	Added area/resource evidence to the final report
FPGA bring-up workflow	Needed real-world validation beyond simulation	Tang Primer 20K hardware testing	Produced final LED-based hardware evidence

14. Conclusions and Recommendations

The final project achieved its central goal: a 32-bit multicycle RV32I processor that was not only implemented, but also verified with traceable evidence and brought onto real FPGA hardware. The strongest results are the 42/42 archived rv32ui PASS outcome, the 40/40 directed unit-test summary, the successful Yosys synthesis result of 6,447 cells, and the Tang Primer 20K LED demonstration.

The most important engineering trade-off was the decision to use a multicycle architecture instead of a pipelined one. That decision sacrificed throughput, but it significantly improved hardware reuse, reduced verification complexity, and made the project achievable within capstone constraints while still producing credible evidence.

The main remaining limitation is the completeness of the external hardware evidence package, not the absence of a working design. The next recommended steps are:

- attach EV-FPGA-01 media to document the Tang Primer 20K LED demo,
- archive any available vendor timing and power reports,
- expand archived exception evidence with privileged or rv32mi test artifacts if course scope requires them.

15. References

- [1] K. Asanovic and D. A. Patterson, "Instruction Sets Should Be Free: The Case for RISC-V," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, 2014.
- [2] RISC-V International, "RISC-V Design Verification Guidelines," technical recommendation, 2024.
- [3] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition. Morgan Kaufmann, 2017.
- [4] D. Harris and S. Harris, Digital Design and Computer Architecture, RISC-V Edition. Morgan Kaufmann, 2022.
- [5] IEEE, IEEE Std 1364-2005: Verilog Hardware Description Language, 2005.
- [6] C. Wolf, "Yosys - A Practical Open Source Framework for RTL Synthesis," in Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip), 2013, pp. 84-87.
- [7] Verilator documentation and user guide, used for simulation methodology, waveform tracing, and DPI-C integration.
- [8] riscv-software-src, riscv-tests GitHub repository, used as the archived instruction-regression source for rv32ui user-mode programs.

Appendix A. Engineering Standards Evidence

Appendix A summarizes the standards and technical references that directly influenced requirements, design choices, and verification in the final project.

Table A.1. Engineering Standards Evidence.

Standard ID and title	Clause(s) used	Why applicable?	Design decision affected	How verified	Status
RISC-V RV32I base ISA	Integer arithmetic, branch, jump, immediate, and memory behaviors	Core functionality depends on ISA correctness	ALU, decoder, control flow, memory behavior	EV-RG-01, EV-RG-02	Applied
RISC-V machine CSR / trap concepts	Trap vector, exception PC, cause storage, return path	Needed for exception-aware architecture	CSR module and trap states	EV-RTL-01	Applied in implemented scope
IEEE 1364-2005	Synthesizable combinational and sequential RTL structure	Required for clean synthesis	Coding style and module structure	EV-SYN-01	Applied
Verilator methodology	Cycle-accurate simulation, VCD tracing, DPI usage	Required for regression and unit tests	Testbench and automated checking	EV-FLOW-01, EV-UT-01	Applied

Appendix B. Multiple-Constraints Matrix

Appendix B collects the final multi-constraint view in one place to show how functionality, cost, schedule, feasibility, and evidence completeness affected the final design.

Table B.1. Multiple-Constraints Matrix.

Constraint	Metric / target	Rationale	Trade-offs considered	Outcome / status	Evidence
Functional correctness	Pass archived rv32ui suite	Ensures processor usefulness	Simplicity vs coverage breadth	Achieved	EV-RG-01
Verification rigor	100% directed-unit-test success	Reduces integration risk	Up-front effort vs late debugging	Achieved	EV-UT-01
Cost	Low-cost hardware and free tools	Capstone budget limit	FPGA vs more expensive platforms	Achieved	Section 10.3
Schedule	Finish within Spring 2026 term	Academic milestone constraint	Feature scope vs depth of evidence	Achieved	Section 10.2
FPGA feasibility	Successful synthesis and live bring-up	Needed for real-world validation	Simulation only vs synthesis and board demo	Achieved with partial external evidence gap	EV-SYN-01, EV-FPGA-01
Timing / power documentation	Archive board-specific reports	Supports stronger hardware claims	Stronger evidence vs available time and files	Not fully verified in repo	None archived

Appendix C. Team Organization and Contributions

- Team roles and ownership are summarized in Section 12.
- The team operated with distributed module ownership and shared integration review.
- Weekly coordination and evidence-driven debugging were used to keep scope under control.

Appendix D. Design Evidence

Appendix D lists the main design artifacts referenced throughout the report. Where media or figures are not embedded directly in this document, the table identifies the source artifact and its purpose.

Evidence ID	Artifact	Purpose
EV-ARCH-01	fig/alu_diagram.png	ALU structure visual
EV-ARCH-02	fig/fsm_diagram.png	Multicycle FSM visual
EV-RTL-01	rtl/top.v, rtl/Control_Unit.v, rtl/CSR.v	Core architecture and trap-path implementation
EV-FPGA-01	Tang Primer 20K board photo or short video showing CPU-driven LED blink	Hardware bring-up evidence; pending insertion

Appendix E. Verification Evidence

Appendix E lists the principal verification artifacts that support the claims in Section 9, including unit-test evidence, regression summaries, a sample PASS log, synthesis output, and the scripted verification flow.

Evidence ID	Artifact	Purpose
EV-UT-01	unit_tests/UNIT_TEST_REPORT.md	Consolidated module-verification evidence
EV-RG-01	results/summary.csv	Regression summary showing 42/42 PASS
EV-RG-02	results/rv32ui-p-add-result.txt	Example archived PASS log
EV-SYN-01	fig/synthesis_report.txt	Synthesis and area evidence
EV-FLOW-01	Makefile, test/converter.py, scripts/run_regression.sh, scripts/summarize_results.py, test/tb/testbench.cpp, test/tb/dpi_functions.c	Reproducible verification workflow evidence

Appendix F. Project Management Evidence

Appendix F records the project-management artifacts reused in final-outcome form, including milestone tracking, team-ownership evidence, and contribution traceability.

Evidence ID	Artifact / source	Purpose
EV-PM-01	Progress-report milestone, risk, and BOM content reused in final-outcome form	Project-management traceability
EV-TEAM-01	Team ownership information from progress dashboard and final report Section 12	Contribution tracking
EV-TEAM-02	Final report Section 12 tables	Teamwork evidence for assessment