CS2323: Computer Architecture

Project and Specification Proposal

Design and FPGA Implementation of a 64-bit RISC-V Processor with FPU and UART Interface

Submitted By:

Krishna H. Patil	(EE24BTECH11036)
Arnav Yadnopavit	(EE24BTECH11007)

Indian Institute of Technology Hyderabad

October 7, 2025

1. Project Scope

The objective of this project is to design and demonstrate a 64-bit RISC-V processor on an FPGA board (Arty A7 or Pynq-Z2). The design will evolve in multiple stages—beginning with a single-cycle implementation and progressively extending to a multicycle version with additional components such as a Floating-Point Unit (FPU) and UART interface for communication with a host computer.

Key Features Planned

- Implementation of the RV64I base instruction set (integer operations).
- Support for arithmetic, logical, load/store, and branch instructions.
- Development of a single-cycle CPU initially, followed by a multicycle version for improved resource utilization.
- Addition of a Floating-Point Unit (FPU) supporting IEEE-754 operations.
- Integration of a UART interface for serial communication and demonstration of results.

The final goal is to demonstrate the working of the 64-bit RISC-V processor executing sample programs (e.g., factorial, matrix operations, floating-point arithmetic) and verify outputs via UART on a connected terminal.

Minimum Deliverable

The minimum deliverable planned for the project is a **64-bit multicycle RISC-V CPU** capable of correctly executing the core RV64I instruction set. This includes arithmetic, logical, load/store, and control-flow instructions, verified both through simulation and on FPGA hardware.

2. Implementation Plan

Phase 1 – Single Cycle RISC-V Processor (RV64I)

Design datapath components: Register File, ALU, Control Unit, Immediate Generator, Program Counter, and Instruction/Data Memory. Implement instruction fetch, decode, execute, memory access, and write-back in a single clock cycle. Verify correct execution using simulation for sample programs.

Phase 2 – Multicycle RISC-V Processor

Modify datapath to reuse functional units across multiple cycles. Implement a finite state machine (FSM) for control, reducing hardware duplication and improving timing closure. Validate correctness using the same instruction tests.

Phase 3 – Floating-Point Unit (FPU) Integration

Implement or integrate a basic IEEE-754 compliant single-precision FPU supporting add, subtract, and multiply operations. Extend control logic to recognize floating-point instructions. Test with mixed integer-floating point programs.

Phase 4 – UART Integration

Add UART transmit and receive modules. Implement a memory-mapped I/O interface for UART communication. Demonstrate program output via a PC terminal.(some interface to work with the CPU in a more comfy way)

3. Verification Plan

Stage	Verification Method	Tools Used
Single Cycle	Simulation of instruction sequences (add, branch, load/store)	ModelSim / Viva
Multicycle	FSM state trace verification, timing analysis	Vivado
FPU	Testbench for FP operations vs. software reference (Python / GCC)	ModelSim / Pyth
UART	Loopback and terminal tests for input/output	Tera Term / PuT
Full System	Run compiled RISC-V assembly programs via `$readmemh`	Vivado / FPGA

4. Tools and Resources

- **Hardware Description Language:** Verilog
- **Simulation:** Vivado Simulator / ModelSim
- **Synthesis & Implementation:** Xilinx Vivado
- **FPGA Boards:** Arty A7 / Pynq-Z2
- **Software Toolchain:** RISC-V GCC / `objcopy` for .hex generation
- **Optional Testing:** Python reference model for numerical verification

5. Expected Outcome

- Working 64-bit RISC-V CPU running on FPGA.
- UART output showing computation results (e.g., factorial, floating-point arithmetic).
- Demonstration video or live hardware run during evaluation.