RV32I Single-Cycle Processor

Overview

This project implements a 32-bit **RV32I single-cycle RISC-V processor** using Verilog HDL. The design follows a classic single-cycle datapath, where each instruction completes execution in one clock cycle. The processor has been developed for functional correctness and clarity of architecture rather than performance optimizations such as pipelining.

The design is synthesized using **Quartus Prime** and functionally verified through **ModelSim** simulations.

Architecture Description

The processor is organized into a clean and modular structure consisting of a controller and datapath, along with separate instruction and data memories.

Major Components

- **Program Counter (PC)** with reset logic
- **Instruction Memory** (read-only, word-aligned)
- **Register File** (32 registers × 32 bits)
- Two combinational read ports
- One synchronous write port
- Register x0 hardwired to zero
- **Immediate Generator** supporting multiple instruction formats
- **ALU** with arithmetic, logical, comparison, and shift operations
- **Control Unit**
- Main decoder for instruction-level control
- ALU decoder for operation selection
- **Data Memory** supporting byte, halfword, and word accesses
- **Branch and Jump Control Logic**

The design uses a Harvard-style memory organization at the top level, with separate instruction and data memories.

Instruction Set Supported

This processor supports the **RV32I base integer instruction set**. The supported instructions are listed below.

Load Instructions

- LB (Load Byte)
- LH (Load Halfword)
- LW (Load Word)

- LBU (Load Byte Unsigned)
- LHU (Load Halfword Unsigned)

Store Instructions

- SB (Store Byte)
- SH (Store Halfword)
- SW (Store Word)

R-Type Instructions

- ADD
- SUB
- AND
- OR
- XOR
- SLT (Set Less Than, signed)
- SLTU (Set Less Than, unsigned)
- SLL (Shift Left Logical)
- SRL (Shift Right Logical)
- SRA (Shift Right Arithmetic)

I-Type Instructions

- ADDI
- ANDI
- ORI
- XORI
- SLTI
- SLTIU
- SLLI
- SRLI
- SRAI

Branch Instructions

- BEQ
- BNE
- BLT
- BGE
- BLTU
- BGEU

Jump Instructions

- JAL
- JALR

Upper Immediate Instructions

- LUI
- AUIPC

Memory System

Instruction Memory

- Word-addressed, combinational read
- Program loaded using `$readmemh`
- Supports sequential instruction fetch

Data Memory

- Supports byte, halfword, and word accesses
- Implements correct sign-extension and zero-extension for load instructions
- Synchronous write and combinational read behavior

Tools Used

- **Verilog HDL** for design implementation
- **Quartus Prime** for synthesis and compilation
- **ModelSim** for functional simulation and waveform analysis

---

Simulation and Verification

Functional verification is performed using ModelSim. Test programs are loaded into instruction memory using hexadecimal files. Simulation waveforms are used to verify: - Correct instruction execution - Proper register file updates - Accurate ALU operations - Correct branching and jumping behavior - Proper memory read/write functionality

---

Limitations

- Single-cycle design (no pipelining)
- No support for RISC-V extensions (M, C, F, etc.)
- No CSR, ECALL, EBREAK, or exception handling

These limitations are intentional to keep the design focused on the RV32I base architecture.

Author

Ankur Mishra

This project was developed as part of an academic exercise to understand the internal working of the RISC-V RV32I architecture at the RTL level.