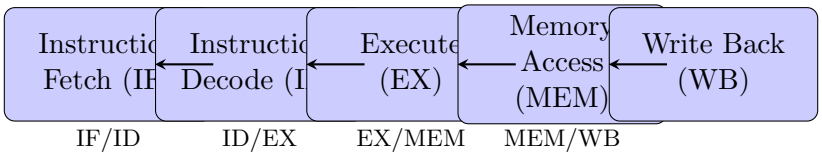
RISC-V RV32I

Pipelined Processor

Technical Datasheet



Version 1.0

June 17, 2025

Advanced Digital Systems Design

Contents

1 Overview

1.1 Introduction

The RISC-V RV32I Pipelined Processor is a 32-bit processor implementation based on the RISC-V instruction set architecture. This design implements a classic 5-stage pipeline with advanced features including branch prediction, hazard detection, and data forwarding to achieve high performance while maintaining compatibility with the RV32I base integer instruction set.

1.2 Key Features

- 32-bit RISC-V RV32I instruction set architecture
- 5-stage pipeline (IF, ID, EX, MEM, WB)
- Branch prediction for improved performance
- Data forwarding to minimize pipeline stalls
- Hazard detection and handling
- 32 32-bit general-purpose registers
- Support for all RV32I base instructions
- Synthesizable Verilog HDL implementation

1.3 Applications

- Educational processor for computer architecture studies
- Embedded system applications
- FPGA-based system implementations
- Research and development platform
- Custom processor applications

2 Architecture

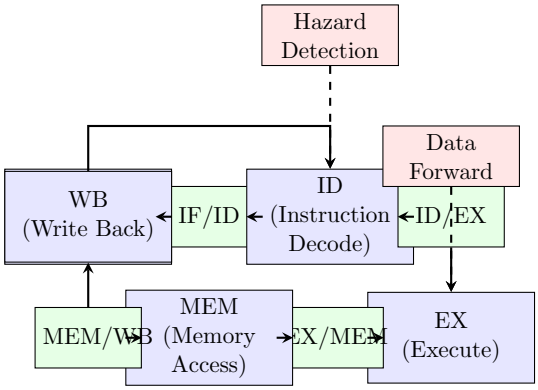
2.1 Pipeline Overview

The processor implements a classic 5-stage RISC pipeline:

Stage	Function
IF (Instruction Fetch)	Fetches instructions from memory, manages program counter
ID (Instruction Decode)	Decodes instructions, reads register file, generates control signals
EX (Execute)	Performs arithmetic/logic operations, calculates branch targets
MEM (Memory Access)	Handles load/store operations, accesses data memory
WB (Write Back)	Writes results back to register file

Table 1: Pipeline Stage Functions

2.2 Block Diagram



3 Functional Specifications

3.1 Instruction Set Support

The processor supports the complete RV32I base instruction set:

Category	Instructions	Description
Arithmetic	ADD, SUB, ADDI	Addition and subtraction operations
Logical	AND, OR, XOR, ANDI, ORI, XORI	Bitwise logical operations
Shift	SLL, SRL, SRA, SLLI, SRLI, SRAI	Left and right shift operations
Compare	SLT, SLTU, SLTI, SLTIU	Set-less-than operations
Branch	BEQ, BNE, BLT, BGE, BLTU, BGEU	Conditional branch instructions
Jump	JAL, JALR	Unconditional jump instructions
Load	LB, LH, LW, LBU, LHU	Load from memory instructions
Store	SB, SH, SW	Store to memory instructions
Upper Immediate	LUI, AUIPC	Upper immediate operations

Table 2: Supported RV32I Instructions

3.2 Register File

- 32 general-purpose registers (x0-x31)
- 32-bit register width
- x0 is hardwired to zero
- Dual-port read, single-port write
- Synchronous write, asynchronous read

3.3 Memory Interface

- 32-bit address space
- Byte-addressable memory
- Support for 8-bit, 16-bit, and 32-bit memory operations
- Separate instruction and data memory interfaces
- Memory-mapped I/O support

4 Performance Characteristics

4.1 Pipeline Performance

Parameter	Value
Pipeline Depth	5 stages
Theoretical CPI	1.0
Branch Penalty	1-2 cycles (with prediction)
Load-Use Penalty	1 cycle
Maximum Clock Frequency	Design dependent

Table 3: Pipeline Performance Parameters

4.2 Hazard Handling

- **Data Hazards**: Resolved through data forwarding and pipeline stalling
- **Control Hazards**: Mitigated using branch prediction and early branch resolution
- **Structural Hazards**: Avoided through proper pipeline design

5 Interface Specifications

5.1 Top-Level Module Interface

Listing 1: TOP_Pipelined_design Module Interface

```
module TOP_Pipelined_design #(parameter size = 32)(
    // Clock and Reset
    input clk,
    input reset,

    // Instruction Memory Interface
    input [size-1:0] instruction_i,
    output [size-1:0] ins_address,

    // Data Memory Interface
    input [size-1:0] MEM_result_i,
    output [size-1:0] RAM_DATA_o,
    output [size-1:0] RAM_Addr_o,
    output [2:0] RAM_DATA_control,
    output RAM_rw
);
```

5.2 Signal Descriptions

6 Design Implementation

6.1 File Organization

The design is organized in a hierarchical structure:

- **digital_top/**: Top-level integration module
- **instruction_fetch/**: IF stage components
- **instruction_decode/**: ID stage components

Signal	Direction	Description
clk	Input	System clock signal
reset	Input	Asynchronous reset (active high)
instruction_i	Input	32-bit instruction from instruction memory
ins_address	Output	Instruction memory address
MEM_result_i	Input	Data from data memory (for loads)
RAM_DATA_o	Output	Data to be written to memory (for stores)
RAM_Addr_o	Output	Data memory address
RAM_DATA_control	Output	Memory operation control (byte/half/word)
RAM_rw	Output	Memory read/write control

Table 4: Top-Level Signal Descriptions

- **execute/**: EX stage components
- **mem/**: MEM stage components
- **write_back/**: WB stage components
- **pipeline_register/**: Inter-stage registers
- **hazard/**: Hazard detection and forwarding
- **common/**: Shared utility modules
- **testbench/**: Verification components

6.2 Key Components

6.2.1 Program Counter (PC_new.v)

Enhanced program counter with:

- Branch prediction support
- JALR instruction handling
- Pipeline stall capability
- Misprediction correction

6.2.2 Functional Unit (FU.v)

Main execution unit containing:

- Arithmetic unit for ADD/SUB operations
- Logic unit for bitwise operations
- Shifter for shift operations
- Comparison logic

6.2.3 Data Forwarding Unit

Handles data hazards by:

- Detecting register dependencies
- Providing bypass paths
- Controlling multiplexer selection

7 Timing Specifications

7.1 Clock Requirements

- Minimum clock period: Technology dependent
- Clock duty cycle: 50% 5%
- Clock skew: ¡ 10% of clock period
- Setup time: Technology dependent
- Hold time: Technology dependent

7.2 Reset Specifications

- Reset type: Asynchronous assert, synchronous deassert
- Reset duration: Minimum 2 clock cycles
- Reset recovery time: 1 clock cycle after deassertion

8 Power Specifications

8.1 Power Consumption

Power consumption varies based on:

- Clock frequency
- Activity factor
- Technology node
- Supply voltage
- Temperature

8.2 Power Management

- Clock gating for unused modules
- Pipeline bubbling to reduce activity
- Optional low-power modes

9 Verification and Testing

9.1 Testbench Coverage

The design includes comprehensive testbenches:

9.2 Test Programs

Specialized test programs verify:

- Basic instruction functionality
- Branch and jump operations
- Load/store operations
- Pipeline hazard handling
- Forwarding mechanisms

Testbench	Purpose
Pipeline_tb.v	Complete processor functionality testing
RegisterFile_tb.v	Register file read/write operations
PC_tb.v	Program counter functionality
Controller_tb.v	Control unit verification
Datapath_tb.v	Datapath component testing
NVRAM_tb.v	Memory interface verification

Table 5: Testbench Components

10 Synthesis and Implementation

10.1 Synthesis Requirements

- Verilog-2001 compliant code
- Synthesizable constructs only
- No behavioral delays in synthesizable code
- Proper reset handling

10.2 Resource Utilization

Typical resource requirements (technology dependent):

- Logic cells: Moderate complexity
- Memory blocks: For register file implementation
- DSP blocks: Not required
- I/O pins: Configurable based on memory interface

11 Application Notes

11.1 Memory System Integration

- Instruction and data memories can be separate or unified
- Memory latency affects pipeline performance
- Cache integration possible for improved performance
- Memory-mapped I/O supported

11.2 Debug and Trace

- Internal signals accessible for debugging
- Pipeline state visibility
- Program counter trace capability
- Register file monitoring

11.3 Customization Options

- Parameterizable data width (default 32-bit)
- Configurable register file size
- Optional branch predictor enhancements
- Custom instruction extensions possible

12 Design Files

12.1 Compilation Order

Use the provided .f files for proper compilation:

Listing 2: Compilation Command Example

```
vlog -f rv32i_pipeline_processor.f +define+PROJ_ROOT="/path/to/project"
```

12.2 Simulation Scripts

- **run_sim.tcl**: ModelSim/QuestaSim simulation script
- **run_sim.bat**: Windows batch script for simulation
- **rv32i_pipeline_processor.f**: Master compilation file

13 Revision History

Version	Date	Changes
1.0	June 17, 2025	Initial datasheet release

Table 6: Revision History

14 Contact Information

For technical support, questions, or additional information about this RISC-V RV32I Pipelined Processor design, please refer to the project documentation and source code comments.

This datasheet describes the RISC-V RV32I Pipelined Processor design implementation. All specifications are subject to the actual HDL implementation and target technology characteristics.