DESIGN SPEC DOCUMENT

Milestone 5

ECE-593: Fundamentals of Pre-Silicon Validation
Maseeh College of Engineering and Computer Science
Winter, 2025

Portland State UNIVERSITY

Project Name: RISC-V ALU Design and Validation

Members: Angelo Maldonado-Liu,Saishree Lnu, Niko Nikolov

Date: 03/04/2025

Project Name	RISC-V ALU Design And Validation
Location	
Start Date	01/27/2025
Estimated Finish Date	03/01/2025
Completed Date	

Prepared by: Team Number	
Prepared for: Prof. Venkatesh Patil	
Team Member Name	Email
Angelo Maldonado-Liu	maldon7@pdx.edu
Saishree Lnu	saishree@pdx.edu
Niko Nikolov	nnikolov@pdx.edu

Design Features:
1. **Operand Selection Mechanism** ❖ **Operand A Selection:** ➢ Primary input from first source register (rs1) ➢ Secondary input from immediate value when needed ❖ **Operand B Selection:** ➢ Primary input from second source register (rs2) ➢ Secondary input from immediate value based on operation type 2. **ALU Core Operations** ❖ **Arithmetic Operations** ➢ ADD: Addition of two 32-bit operands ➢ SUB: Subtraction of two 32-bit operands ❖ **Logical Operations**

- ➢ AND: Bitwise AND operation
  - ➢ OR: Bitwise OR operation
  - ➢ XOR: Exclusive OR operation
- ❖ **Shift Operations**
  - ➢ SLL: Logical left shift
  - ➢ SRL: Logical right shift
  - ➢ SRA: Arithmetic right shift (sign-preserving)
- ❖ **Comparison Operations**
  - ➢ SLT: Set less than (signed)
  - ➢ SLTU: Set less than unsigned
  - ➢ EQ: Equality comparison
  - ➢ NEQ: Not equal comparison
  - ➢ GE: Greater than or equal (signed)
  - ➢ GEU: Greater than or equal unsigned

Project Description:

The RISC-V ALU module implements a comprehensive arithmetic logic unit supporting the RV32I base instruction set. The design focuses on efficient computation and logical operations with the following key aspects:

Core Functionality

1. **Operand Processing**
   - 32-bit input operand handling
   - Immediate value support
   - Sign extension handling for signed operations
2. **Computation Engine**
   - Dedicated arithmetic unit for ADD/SUB operations
   - Optimized logic unit for AND/OR/XOR
   - Barrel shifter for shift operations
   - Comparison logic for all comparison operations
3. **Result Generation**
   - 32-bit result output

- Zero flag generation
- Overflow detection for arithmetic operations
- Sign flag for comparison results

Important Signals/Flags

Core Control Signals:

```
input  logic    i_clk;      // System clock
input  logic    i_rst_n;    // Active-low reset
input  logic    i_ce;       // Clock enable
```

Operation Control Signals:

```
input  logic [`ALU_WIDTH-1:0] i_alu;     // ALU operation selection
input  logic [2:0]            i_funct3;  // Function type specifier
input  logic [`OPCODE_WIDTH-1:0] i_opcode; // Instruction opcode
```

Data Input Signals:

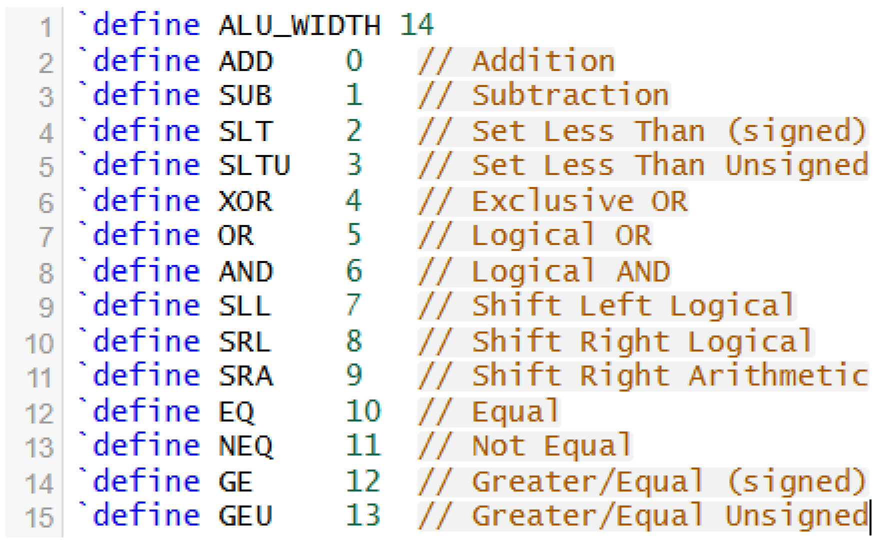
```
input  logic [4:0]  i_rs1_addr;        // Source register 1 address
input  logic [31:0] i_rs1;             // Source register 1 value
input  logic [31:0] i_rs2;             // Source register 2 value
input  logic [31:0] i_imm;             // Immediate value
input  logic [31:0] i_pc;              // Program counter value
input  logic [4:0]  i_rd_addr;         // Destination register address
```

Data Output Signals:

```
output logic [31:0] o_y;               // ALU result
output logic [31:0] o_rd;              // Value for destination register
output logic [4:0]  o_rd_addr;         // Destination register address
output logic        o_rd_valid;        // Destination register value valid
output logic        o_wr_rd;           // Write enable for destination register
```

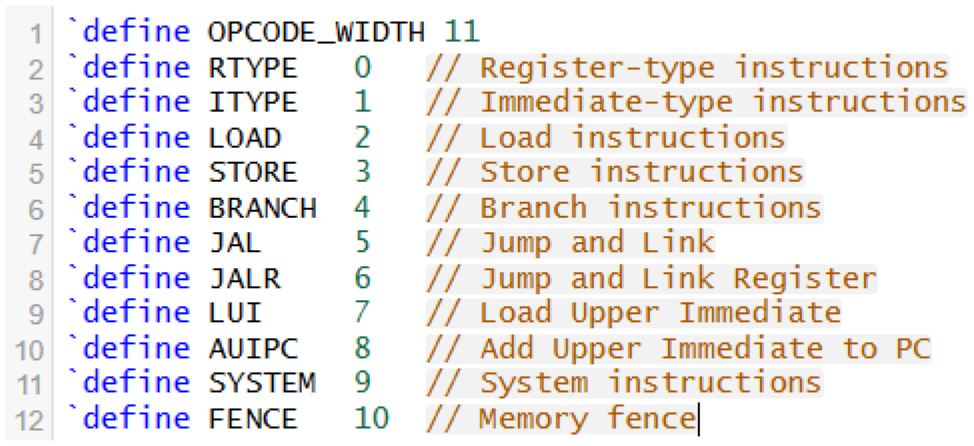
Design Signals

ALU Operation Encodings



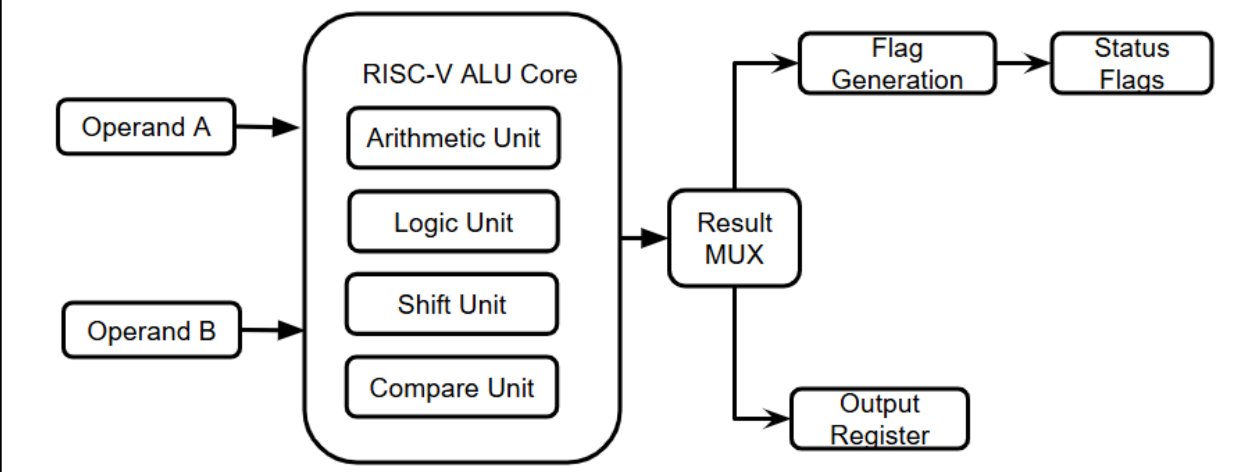
```
`define ALU_WIDTH 14
`define ADD     0    // Addition
`define SUB     1    // Subtraction
`define SLT     2    // Set Less Than (signed)
`define SLTU    3    // Set Less Than Unsigned
`define XOR     4    // Exclusive OR
`define OR      5    // Logical OR
`define AND     6    // Logical AND
`define SLL     7    // Shift Left Logical
`define SRL     8    // Shift Right Logical
`define SRA     9    // Shift Right Arithmetic
`define EQ      10   // Equal
`define NEQ     11   // Not Equal
`define GE      12   // Greater/Equal (signed)
`define GEU     13   // Greater/Equal Unsigned
```

Opcode Encodings



```
`define OPCODE_WIDTH 11
`define RTYPE    0    // Register-type instructions
`define ITYPE    1    // Immediate-type instructions
`define LOAD     2    // Load instructions
`define STORE    3    // Store instructions
`define BRANCH   4    // Branch instructions
`define JAL      5    // Jump and Link
`define JALR     6    // Jump and Link Register
`define LUI      7    // Load Upper Immediate
`define AUIPC    8    // Add Upper Immediate to PC
`define SYSTEM   9    // System instructions
`define FENCE    10   // Memory fence
```

Block Diagram



References/Citations

- Alan-Tony. (n.d.). *GitHub - Alan-Tony/RISC-V-Datapath-and-Control: A basic ALU and ALU Control Unit Implementation using Verilog*. GitHub. https://github.com/Alan-Tony/RISC-V-Datapath-and-Control
- AngeloJacobo. (n.d.). *GitHub - AngeloJacobo/RISC-V: Design implementation of the RV32I Core in Verilog HDL with Zicsr extension*. GitHub. https://github.com/AngeloJacobo/RISC-V
- Deng, L. (2024). Design a 5-stage pipeline RISC-V CPU and optimise its ALU. *Applied and Computational Engineering*, *34*(1), 237–244. https://doi.org/10.54254/2755-2721/34/20230334
- *Design and implementation of 32-bit RISC-V processor using Verilog*. (2024, October 18). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10750638
- Eymay. (n.d.). *GitHub - eymay/ALU: A fast ALU design for RISC-V hardware*. GitHub. https://github.com/eymay/ALU
- Patterson, D. A., & Hennessy, J. L. (2020). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann.
- Robert Baruch. (2018, June 22). *LMARV-1 (Tangible RISC-V) Part 3: Designing the ALU* [Video]. YouTube. https://www.youtube.com/watch?v=KGBUtKRBKZs