UNIVERSITY OF CALIFORNIA IRVINE

EECS 31L: Introduction to Digital Logic Laboratory (Spring 2025)

Lab 2: Arithmetic Logic Unit (ALU)

1. Overview

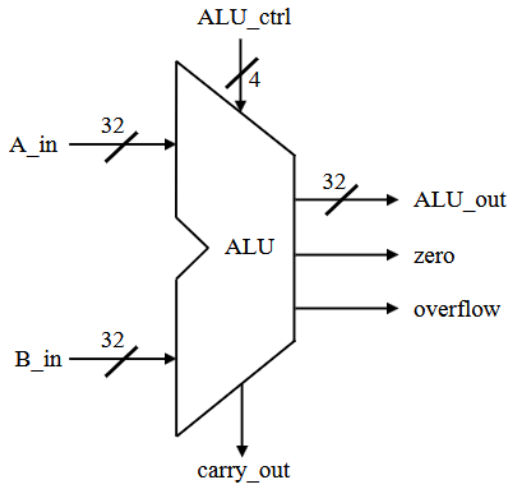

Figure 1: Arithmetic Logic Unit Block Diagram

This lab focused on designing a 32-bit arithmetic logic unit (ALU) in Verilog. The ALU performs arithmetic and logical operations on binary numbers. It accepts two 32-bit inputs A_in and B_in and a 4-bit control signal ALU_ctrl that decides which operation to perform. The ALU produces a 32-bit output ALU_out and sets three output flags: zero, overflow, and carry_out.

The ALU_ctrl selects between seven operations: AND, OR, Add, Subtract, Set Less Than (slt), NOR, and Equal Comparison. Based on this control code, the ALU routes the appropriate operation to compute the result. The zero flag indicates whether the result is all zeroes, the carry_out flag indicates that there is an unsigned overflow from the most significant bit during addition, and the overflow flag indicates that there was a signed arithmetic overflow.

2. Hardware Design

The 32-bit ALU was implemented as a combinational logic block using an always @(*) statement. Its behavior is controlled by a 4-bit input signal ALU_ctrl, which selects the operation that will be performed. An operation is chosen based on the value of ALU_ctrl via a case statement.

For bitwise operations such as AND, OR, and NOR, the ALU performs a direct logical operation on each bit between the two 32-bit inputs A_in and B_in. For arithmetic operations such as addition and subtraction, the ALU detects overflow. In addition, the sum and carry-out are concatenated together. The

overflow signal is determined based on the signs of the operands and the sum. In subtraction, overflow is checked and the carry-out output is set to zero since carry-out is only relevant for addition.

Comparison operations use signed arithmetic. In the Set Less Than (SLT) operation, the operands are cast to signed integers using the $signed() function to ensure correct signed comparison. The Equal Comparison operation checks whether the two inputs are equal and produces either a 32-bit 1 (32'b1) or a 32-bit 0 (32'b0).

The zero output is computed by checking if the 32-bit result ALU_out is all zeros after any operation. If so, the zero flag is set to 1 or remains a 0 otherwise.

A snippet of the Verilog code used for addition is shown below:

```
// Add
4'b0010: begin
    {carry_out, ALU_out} = A_in + B_in;
    overflow = (~A_in[31] & ~B_in[31] & ALU_out[31]) | (A_in[31] & B_in[31] & ~ALU_out[31]);
end
```

Figure 2.1: Addition Code Snippet (ALU.v)

This block adds the two 32-bit inputs, takes the carry-out from the most significant bit, and detects overflow based on the sign bits of the operands and sum.

3. Simulation Results

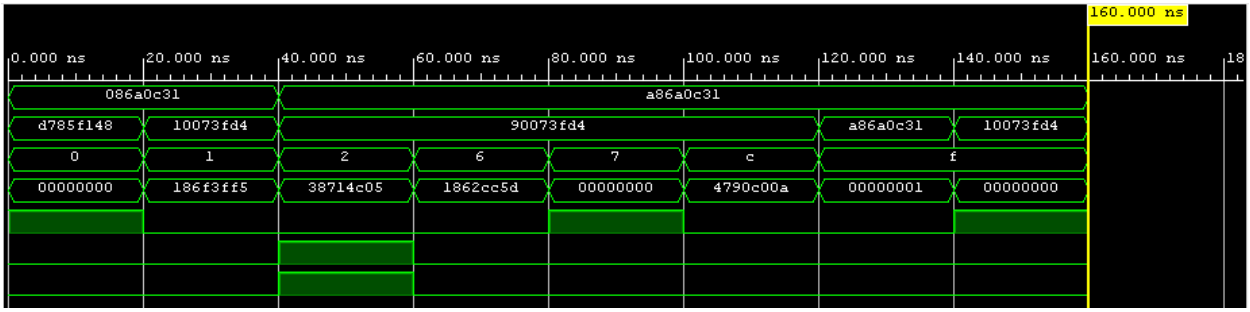

Figure 3.1: ALU Waveform

The 32-bit ALU design was verified using a Verilog testbench ALU_tb.v that applied eight test cases to check each of the specified ALU operations. Two 32-bit inputs, A_in and B_in were applied for each case with the corresponding ALU_out, zero, carry_out, and overflow outputs being observed. Each test case ran for 20 nanoseconds before applying the next set of inputs. The simulation was monitored using a $monitor statement for textual output and a waveform file was created using $dumpfile and $dumpvars to inspect the transitions after each interval.

The simulation waveform is shown in Figure 3.1. Each block of 20 nanoseconds corresponds to one ALU operation. The results matched the expected behavior for each operation:

- (0-20 ns) The AND operation produced all zeros.
- (20-40 ns) The OR operation produced the bitwise OR of the two inputs.
- (40-60 ns) The addition and (60-80 ns) subtraction operations handled signed arithmetic properly and set the carry_out and overflow flags when appropriate.
- (80-100 ns) The Set Less Than (SLT) operation compared the signed values.
- (100-120 ns) The NOR operation inverted the result of the OR operation.
- (120-140 ns) The Equal Comparison identified when two inputs were equal or not.

No discrepancies were observed and the ALU performed as expected for all provided test cases.