# Digital System Design using HDL Lab Report

Experiment 8c

Arithmetic logic unit

Submitted by

**Madhu Krishnan A P**

M.Tech VLSI and Embedded Systems

Department of Electronics

Cochin University of Science and Technology

## Contents

## 1 Module description

The ALU Control Module is designed to generate control signals determining which arithmetic or logic operation the ALU performs. These operations include addition, subtraction, logical AND, logical OR and comparison for less than. The control signals are typically generated based on the instruction's opcode and additional function fields that specify the type of operation.

**Inputs**

- **ALUControl2:0 (3-bit)**: This 3-bit control input specifies the desired operation for the ALU. Each value of this input corresponds to a specific arithmetic or logic operation.
- **Data inputs (32-bit)**: The operations are performed on these data inputs.

ALUControl2:0 is a crucial part of the ALU's operation, allowing it to perform different arithmetic and logical functions based on the control signal. These operations enable a processor to perform various tasks, including mathematical calculations, bitwise manipulations, and decision-making processes required for executing programs. The ALUControl2:0 signal is typically controlled by a combination of instruction decoding and the processor's control unit, enabling dynamic selection of the appropriate operation during the execution of an instruction.

## 2 Block diagram

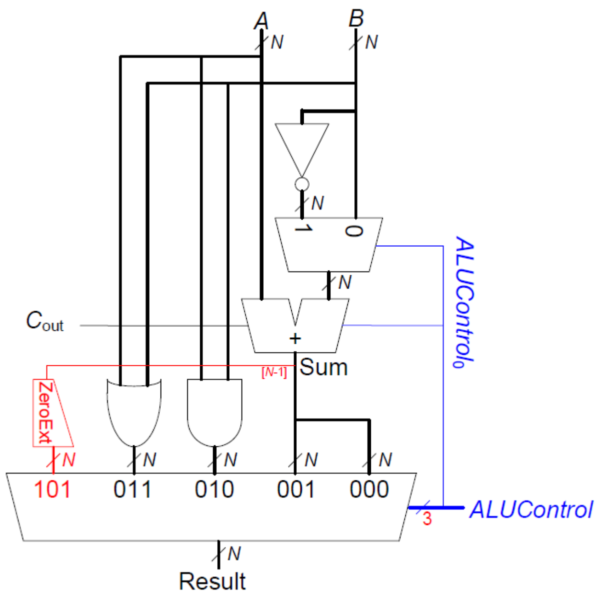


Figure 1: Arithmetic and logic unit

## 3 Operation Logic

- **Addition (000)**: The ALU adds the two input values.
- **Subtraction (001)**: The ALU subtracts the second operand from the first.
- **AND (010)**: The ALU performs a bitwise AND operation on the operands.
- **OR (011)**: The ALU performs a bitwise OR operation on the operands.
- **SLT (101)**: The ALU compares if the first operand is less than the second. It outputs 1 if true and 0 if false, often used in conditional statements.

## 4 Control path

- The ALU Control module receives the opcode or function code from the instruction being processed.
- Based on the opcode or function code, it generates a 3-bit signal (ALUControl2:0) that controls the ALU's operation.
- The ALU uses this signal to perform the specified arithmetic or logic operation.

## 5 Verilog description

The following description is used to describe the system. Behavioural modelling is used to describe the system.

```verilog
module ALU (
    input  logic [31:0] A,              // Operand A
    input  logic [31:0] B,              // Operand B
    input  logic [2:0] ALUControl,      // ALU control signals (2:0)
    output logic [31:0] Result,         // ALU result
    output logic Zero                   // Zero flag (Result == 0)
);

    always @(A, B, ALUControl) begin
    // Perform ALU operations based on ALUControl
    if (ALUControl == 3'b000) begin
        Result = A + B;  // Add
    end

    else if (ALUControl == 3'b001) begin
        Result = A - B;  // Subtract
    end

    else if (ALUControl == 3'b010) begin
        Result = A & B;  // AND
    end

    else if (ALUControl == 3'b011) begin
        Result = A | B;  // OR
    end

    else if (ALUControl == 3'b101) begin
        Result = (A < B) ? 1 : 0;  // Set Less Than (SLT)
    end

    else begin
        Result = 32'b0;  // Default case to handle undefined operations
    end

    // Zero flag is set if the result is zero
    Zero = (Result == 0);
end

endmodule
```

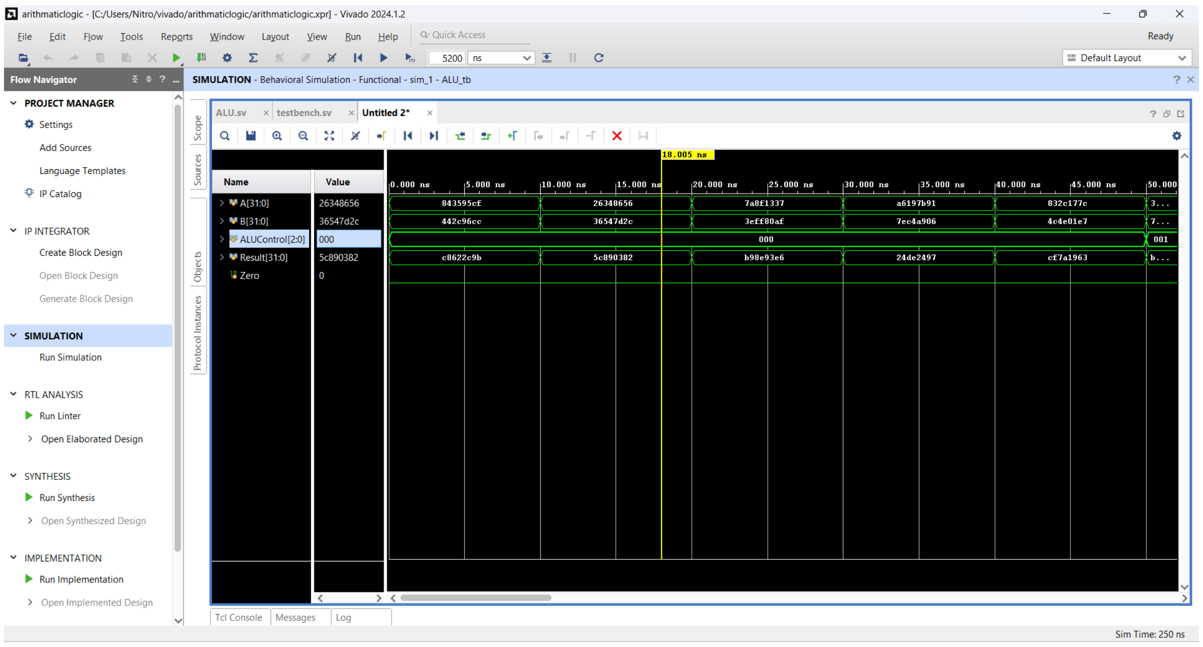
## 6 Testbench

The following testbench does 5 cycles each with every valid input of ALU Control signal. Random numbers are chosen for inputs A and B.

```
module ALU_tb;

    // Testbench signals
    logic [31:0] A, B;              // Operand A and Operand B
    logic [2:0] ALUControl;         // ALU control signals
    logic [31:0] Result;            // ALU result
    logic Zero;                     // Zero flag

    // Instantiate the ALU module
    ALU uut (
        .A(A), .B(B),
        .ALUControl(ALUControl),
        .Result(Result), .Zero(Zero)
    );

    // Test stimulus
    initial begin
        // Initialize signals to known values
        A = 32'b0;
        B = 32'b0;
        ALUControl = 3'b000;

        // Apply random test cases
        for (int j = 0; j <= 3; j = j + 1) begin
            for (int i = 0; i < 5; i = i + 1) begin
            // Generate random values for A and B
            A = $urandom;  // Random 32-bit operand A
            B = $urandom;  // Random 32-bit operand B
            ALUControl = j;
            #10;
            end
        end

        for (int i = 0; i < 5; i = i + 1) begin
            A = $urandom;  // Random 32-bit operand A
            B = $urandom;  // Random 32-bit operand B
            ALUControl = 3'b101;
            #10;
        end

        $finish;    // End simulation
    end

endmodule
```

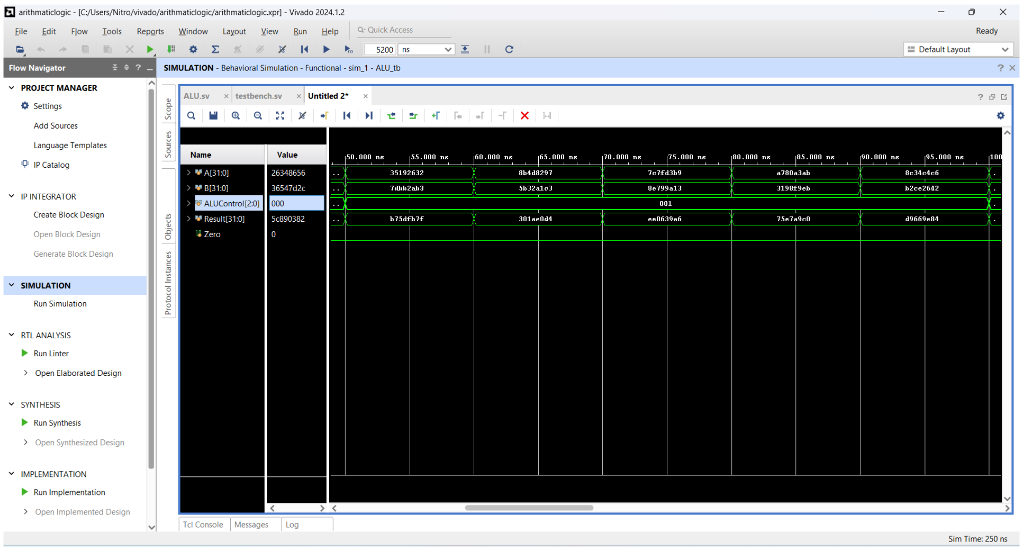
## 7 Simulation result

The following result is obtained by performing the simulation on the testbench for the design of ALU.



| Name | Value | 0.000 ns | 5.000 ns | 10.000 ns | 15.000 ns | 20.000 ns | 25.000 ns | 30.000 ns | 35.000 ns | 40.000 ns | 45.000 ns | 50.000 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A[31:0] | 26348656 | 843595cf | | 26348656 | | 7a8f1337 | | a6197b91 | | 832c177c | | 3... |
| B[31:0] | 36547d2c | 442c96cc | | 36547d2c | | 3eff80af | | 7ec4a906 | | 4c4e01e7 | | 7... |
| ALUControl[2:0] | 000 | 000 | | | | | | | | | | 001 |
| Result[31:0] | 5c890382 | c8622c9b | | 5c890382 | | b98e93e6 | | 24de2497 | | cf7a1963 | | b... |
| Zero | 0 | | | | | | | | | | | |

Figure 2: ALU: Simulation result: Addition



| Name | Value | 50.000 ns | 55.000 ns | 60.000 ns | 65.000 ns | 70.000 ns | 75.000 ns | 80.000 ns | 85.000 ns | 90.000 ns | 95.000 ns | 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A[31:0] | 26348656 | 35192632 | | 8b4d8297 | | 7c7fd3b9 | | a780a3ab | | 8c34c4c6 | | . |
| B[31:0] | 36547d2c | 7dbb2ab3 | | 5b32a1c3 | | 8e799a13 | | 3198f9eb | | b2ce2642 | | . |
| ALUControl[2:0] | 000 | 001 | | | | | | | | | | . |
| Result[31:0] | 5c890382 | b75dfb7f | | 301ae0d4 | | ee0639a6 | | 75e7a9c0 | | d9669e84 | | . |
| Zero | 0 | | | | | | | | | | | |

Figure 3: ALU: Simulation result: Subtraction

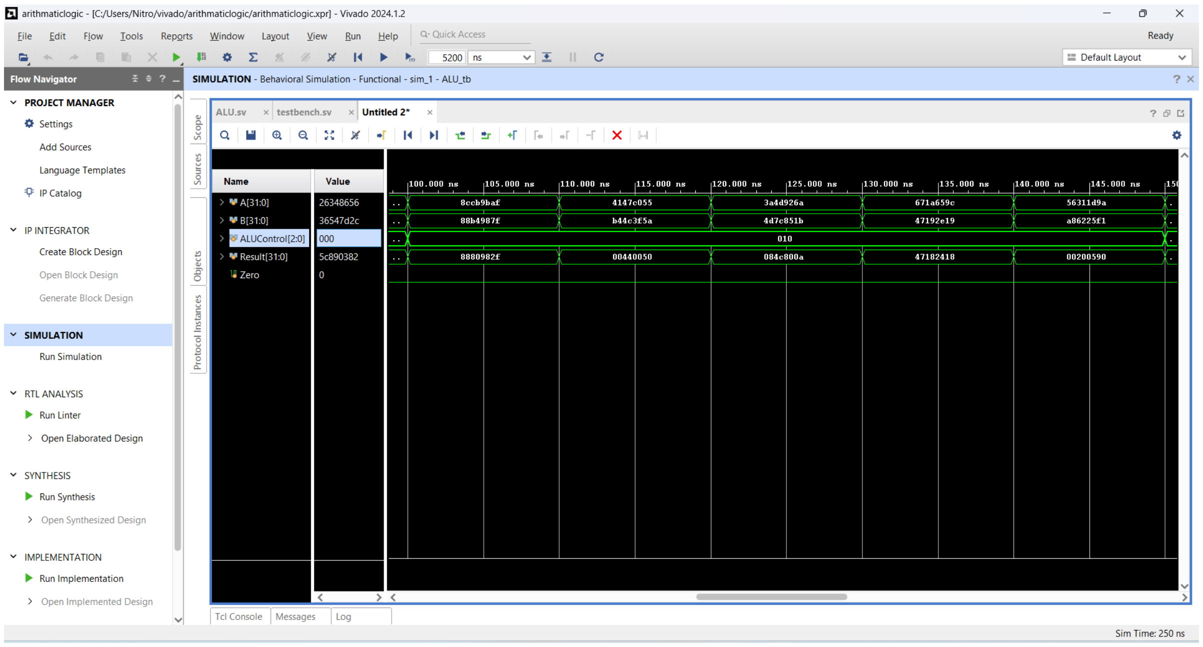

```
arithmaticlogic - [C:/Users/Nitro/vivado/arithmaticlogic/arithmaticlogic.xpr] - Vivado 2024.1.2
SIMULATION - Behavioral Simulation - Functional - sim_1 - ALU_tb

Name             Value      |100.000 ns  |105.000 ns  |110.000 ns  |115.000 ns  |120.000 ns  |125.000 ns  |130.000 ns  |135.000 ns  |140.000 ns  |145.000 ns
A[31:0]          26348656   ..  8ccb9baf                  4147c055                  3a4d926a                  671a659c                  56311d9a      .
B[31:0]          36547d2c   ..  88b4987f                  b44c3f5a                  4d7c851b                  47192e19                  a86225f1      .
ALUControl[2:0]  000        ..                                                       010                                                            .
Result[31:0]     5c890382   ..  8880982f                  00440050                  084c800a                  47182418                  00200590      .
Zero             0

Sim Time: 250 ns
```

Figure 4: ALU: Simulation result: Logic AND

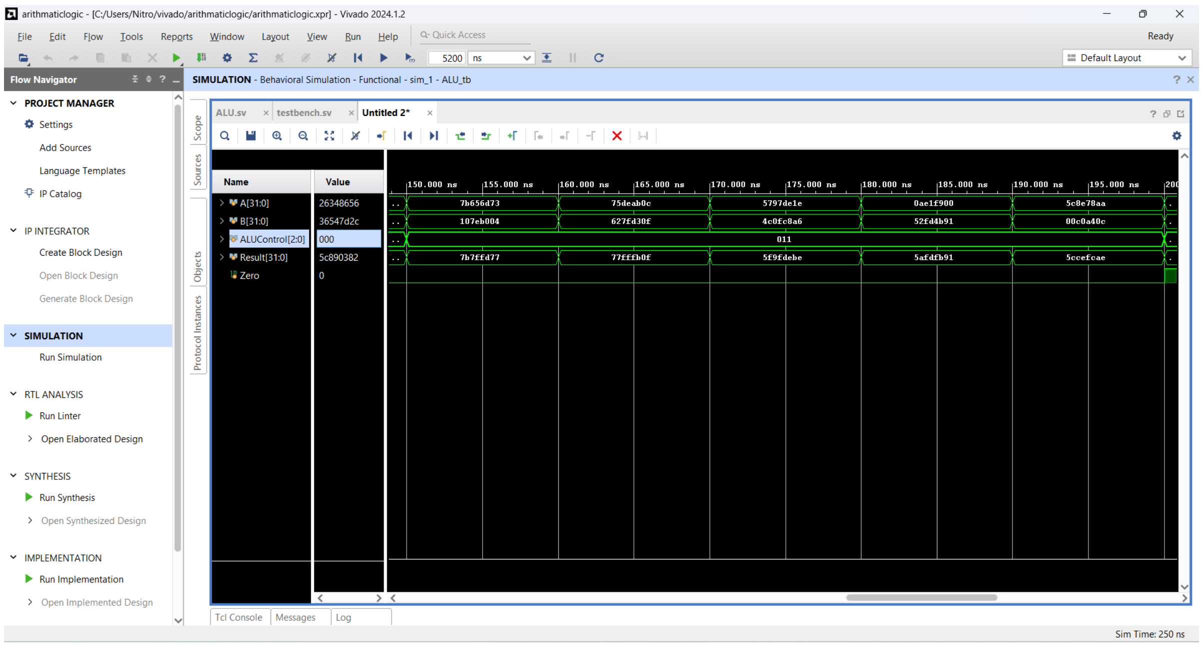
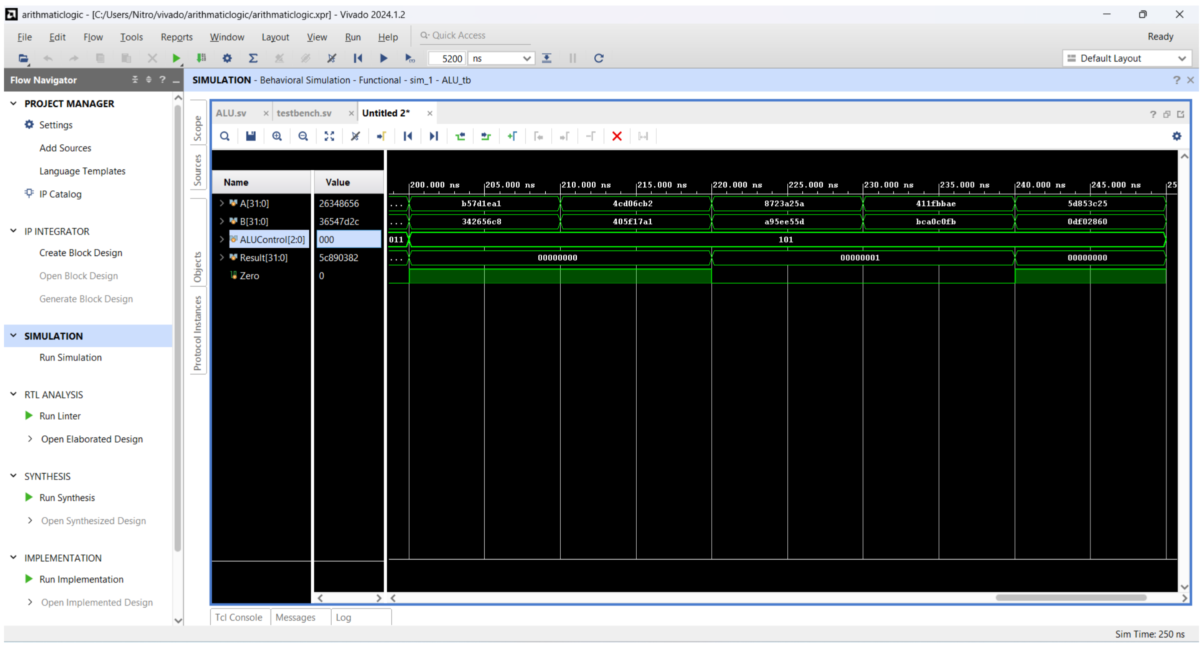

```
arithmaticlogic - [C:/Users/Nitro/vivado/arithmaticlogic/arithmaticlogic.xpr] - Vivado 2024.1.2
SIMULATION - Behavioral Simulation - Functional - sim_1 - ALU_tb

Name             Value      |150.000 ns  |155.000 ns  |160.000 ns  |165.000 ns  |170.000 ns  |175.000 ns  |180.000 ns  |185.000 ns  |190.000 ns  |195.000 ns
A[31:0]          26348656   ..  7b656d73                  75deab0c                  5797de1e                  0ae1f900                  5c8e78aa      .
B[31:0]          36547d2c   ..  107eb004                  627fd30f                  4c0fc8a6                  52fd4b91                  00c0a40c      .
ALUControl[2:0]  000        ..                                                       011                                                            .
Result[31:0]     5c890382   ..  7b7ffd77                  77fffb0f                  5f9fdebe                  5afdfb91                  5ccefcae      .
Zero             0

Sim Time: 250 ns
```

Figure 5: ALU: Simulation result: Logical OR



SIMULATION - Behavioral Simulation - Functional - sim_1 - ALU_tb

```
Name              Value      200.000 ns  205.000 ns  210.000 ns  215.000 ns  220.000 ns  225.000 ns  230.000 ns  235.000 ns  240.000 ns  245.000 ns
A[31:0]           26348656   ...  b57d1ea1         4cd06cb2         8723a25a         411fbbae         5d853c25
B[31:0]           36547d2c   ...  342656c8         405f17a1         a95ee55d         bca0c0fb         0df02860
ALUControl[2:0]   000        011  101
Result[31:0]      5c890382   ...  00000000                          00000001                          00000000
Zero              0
```

Figure 6: ALU: Simulation result: SLT