📘 **Day 11 Topic: Verilog codes for 4 bit ALU Using Cadence (Xcelium)**

🔹 What is an ALU?

An Arithmetic Logic Unit (ALU) is a combinational circuit that performs arithmetic (add, subtract, increment, etc.) and logic (AND, OR, XOR, etc.) operations on binary data.

It is the heart of the CPU, as it executes the actual data processing.

🔹 Block Diagram of a 4-bit ALU

- Inputs:
    - Two 4-bit operands → A[3:0], B[3:0]
    - Control signals (operation select lines) → e.g., sel[2:0]
- Outputs:
    - 4-bit result → Y[3:0]
    - Status flags (optional) → Carry, Zero, Overflow, Negative

---

🔹 Common Operations in a 4-bit ALU

The ALU typically performs both Arithmetic and Logic functions:

1. Arithmetic Operations

- Addition: Y = A + B
- Subtraction: Y = A - B
- Increment: Y = A + 1
- Decrement: Y = A - 1

2. Logic Operations

- AND: Y = A & B
- OR: Y = A | B
- XOR: Y = A ^ B
- XNOR: Y = ~(A ^ B)

- NOT: Y = ~A

3. Shift Operations (sometimes included)

- Logical Shift Left (LSL)
- Logical Shift Right (LSR)

🔹 Example Operation Selection Table

sel	Operation	Output Y
000	A + B	Arithmetic
001	A - B	Arithmetic
010	A & B	Logic (AND)
011	A | B	Logic (OR)
100	A ^ B	Logic (XOR)
101	~A	Logic (NOT)
110	A + 1	Increment
111	A - 1	Decrement

🔹 Flags (Status Indicators)

Most ALUs also generate flags that indicate special results:

- Carry flag (C) → 1 if result generates a carry out.
- Zero flag (Z) → 1 if result is zero.
- Overflow flag (V) → 1 if signed overflow occurs.
- Negative flag (N) → 1 if MSB = 1 (negative in signed).

🔹 **1.1) Design Code**

alu_4bit.v

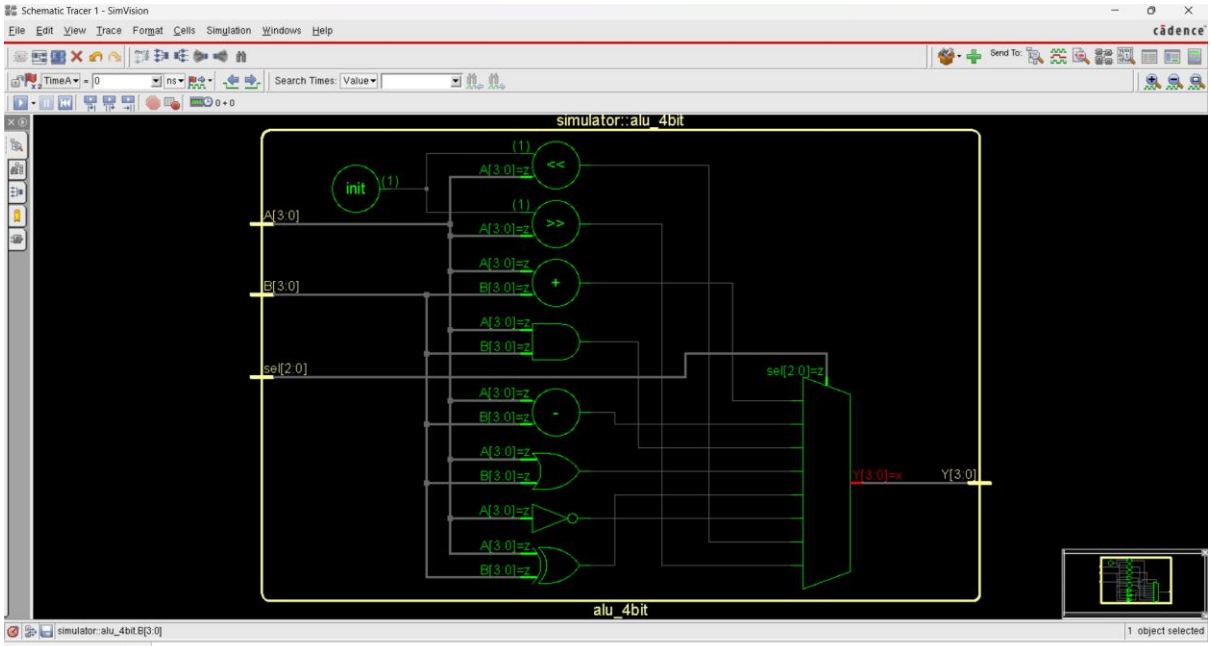
```
module alu_4bit (
    input  [3:0] A,
    input  [3:0] B,
    input  [2:0] sel,
    output reg [3:0] Y
);
    always @(*) begin
        case (sel)
            3'b000: Y = A + B;
            3'b001: Y = A - B;
            3'b010: Y = A & B;
            3'b011: Y = A | B;
            3'b100: Y = A ^ B;
            3'b101: Y = ~A;
            3'b110: Y = A << 1;
            3'b111: Y = A >> 1;
            default: Y = 4'b0000;
        endcase
    end
endmodule
```

- **1.2) Test Bench Code**

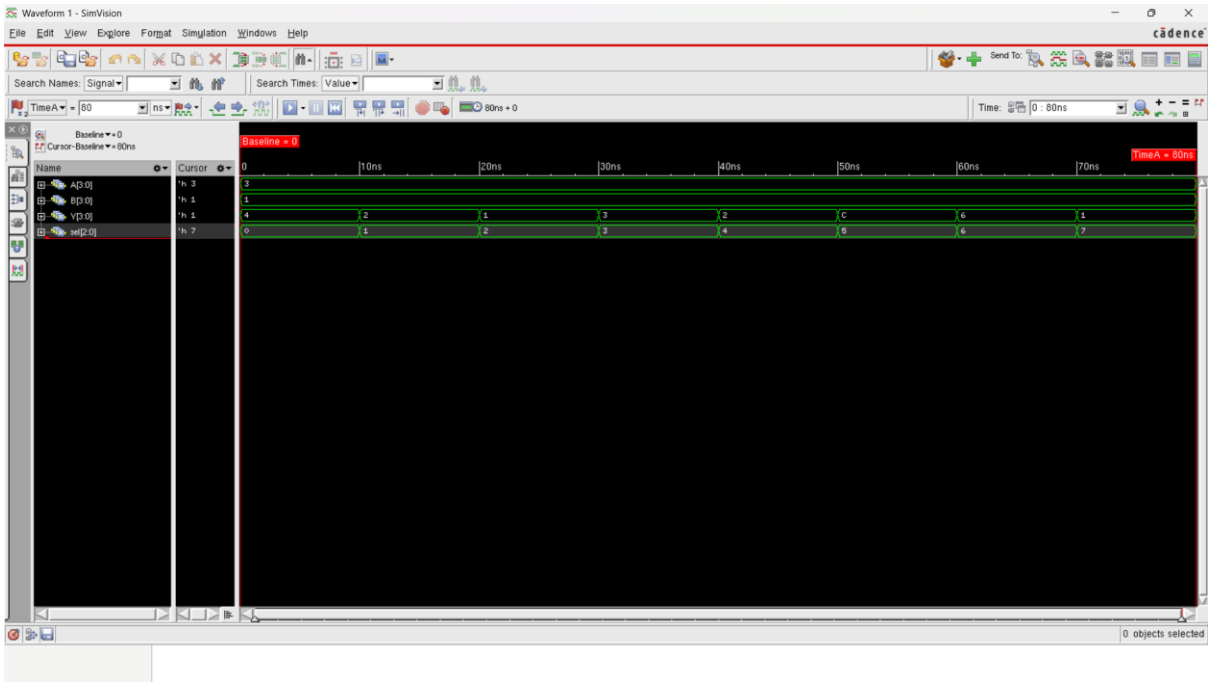
alu_4_tb.v

```
module tb_alu_4bit;
    reg [3:0] A, B;
    reg [2:0] sel;
    wire [3:0] Y;

    alu_4bit uut (
        .A(A), .B(B), .sel(sel), .Y(Y)
    );

    initial begin
        $display("A\tB\tsel\tY\tOperation");

        A = 4'b0011; B = 4'b0001;

        sel = 3'b000; #10; $display("%b\t%b\t%03b\t%b\tADD", A, B, sel, Y);
        sel = 3'b001; #10; $display("%b\t%b\t%03b\t%b\tSUB", A, B, sel, Y);
        sel = 3'b010; #10; $display("%b\t%b\t%03b\t%b\tAND", A, B, sel, Y);
        sel = 3'b011; #10; $display("%b\t%b\t%03b\t%b\tOR",  A, B, sel, Y);
        sel = 3'b100; #10; $display("%b\t%b\t%03b\t%b\tXOR", A, B, sel, Y);
        sel = 3'b101; #10; $display("%b\t--\t%03b\t%b\tNOT A", A, sel, Y);
        sel = 3'b110; #10; $display("%b\t--\t%03b\t%b\tSHIFT LEFT", A, sel, Y);
        sel = 3'b111; #10; $display("%b\t--\t%03b\t%b\tSHIFT RIGHT", A, sel, Y);

        $finish;
    end
endmodule
```

- **1.3) Schematic**



- **1.4) Wave Forms**



- **Applications of ALU**
  - **Microprocessors → every CPU has ALU for executing instructions.**
  - **DSP (Digital Signal Processing) → arithmetic-heavy applications.**
  - **Control systems → decision making based on arithmetic comparisons.**
  - **Cryptography → heavy bitwise operations.**
  - **Embedded systems → implementing arithmetic logic at hardware level.**