### \*\*ALU Module (alu.v)\*\*

verilog

module ALU (

input [3:0] A, B, // 4-bit Inputs

input [2:0] ALU\_Sel, // 3-bit control signal to select operation

output reg [3:0] Result // 4-bit Output

);

- module ALU: Declares a new module named ALU.

- input [3:0] A, B: Defines \*two 4-bit inputs\* (A and B) for arithmetic/logic operations.

- input [2:0] ALU\_Sel: Defines a \*3-bit selection input\* that determines which operation the ALU should perform.

- output reg [3:0] Result: Declares a \*4-bit output\* (Result) to store the operation result. It's a reg because we use an \*always block\* (which needs a register to store the result).

---

verilog

always @(\*) begin

- always @(\*): This means \*anytime\* any input (A, B, or ALU\_Sel) changes, execute the block.

---

verilog

case (ALU\_Sel)

- case (ALU\_Sel): A \*case statement\* is used to perform different operations depending on the value of ALU\_Sel.

---

### \*Operations\*

verilog

3'b000: Result = A + B; // Addition

- If ALU\_Sel is 000, \*addition\* (A + B) is performed.

verilog

3'b001: Result = A - B; // Subtraction

- If ALU\_Sel is 001, \*subtraction\* (A - B) is performed.

verilog

3'b010: Result = A & B; // AND operation

- If ALU\_Sel is 010, the \*bitwise AND\* operation (A & B) is performed.

verilog

3'b011: Result = A | B; // OR operation

- If ALU\_Sel is 011, the \*bitwise OR\* operation (A | B) is performed.

verilog

3'b100: Result = ~A; // NOT operation (only A is used)

- If ALU\_Sel is 100, the \*bitwise NOT\* operation (~A) is performed. (This operation \*ignores B\*