alu\_project/

│

├── alu\_defines.vh // macros and parameters

├── alu.v // ALU implementation

├── alu\_tb.v // testbench

File Name : alu\_defines.vh ( Verilog header file )

// Define Bit-width (choose ONE)

`define WIDTH\_4

// `define WIDTH\_8

// `define WIDTH\_16

`ifdef WIDTH\_4

`define DATA\_WIDTH 4

`elsif WIDTH\_8

`define DATA\_WIDTH 8

`elsif WIDTH\_16

`define DATA\_WIDTH 16

`else

`define DATA\_WIDTH 8 // default

`endif

// Operation Modes

`define OP\_ADD 3'b000

`define OP\_SUB 3'b001

`define OP\_AND 3'b010

`define OP\_OR 3'b011

`define OP\_XOR 3'b100

alu.v – Main ALU Module

`include "alu\_defines.vh"

module ALU (

input [`DATA\_WIDTH-1:0] a, b,

input [2:0] op\_code,

output reg [`DATA\_WIDTH-1:0] result );

always @(\*) begin

case (op\_code)

`OP\_ADD: result = a + b;

`OP\_SUB: result = a - b;

`OP\_AND: result = a & b;

`OP\_OR: result = a | b;

`OP\_XOR: result = a ^ b;

default: result = {`DATA\_WIDTH{1'b0}};

endcase

end

endmodule

alu\_tb.v – Testbench with timescale

`timescale 1ns/1ps

`include "alu\_defines.vh"

module alu\_tb;

reg [`DATA\_WIDTH-1:0] a, b;

reg [2:0] op\_code;

wire [`DATA\_WIDTH-1:0] result;

ALU uut (.a(a), .b(b), .op\_code(op\_code), .result(result));

initial begin

$display("Testing ALU with data width: %0d bits", `DATA\_WIDTH);

a = 'd3; b = 'd2; op\_code = `OP\_ADD; #10;

$display("ADD: %d + %d = %d", a, b, result);

a = 'd5; b = 'd1; op\_code = `OP\_SUB; #10;

$display("SUB: %d - %d = %d", a, b, result);

a = 'd15; b = 'd6; op\_code = `OP\_AND; #10;

a = 'd9; b = 'd3; op\_code = `OP\_OR; #10;

a = 'd12; b = 'd5; op\_code = `OP\_XOR; #10;

$finish;

end

endmodule