Project using compiler directives

```
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
```

Project using compiler directives

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
```