Languages & Libraries

Testbench + Design

SystemVerilog/Verilog

UVM / OVM (http://eda-playground.readthedocs.org/en/latest/intro.html#libraries-methodologies)

None

Other Libraries (http://eda-playground.readthedocs.org/en/latest/intro.html#libraries-methodologies)

None
OVL
SVUnit

- [ ] Enable TL-Verilog (http://www.redwoodeda.com)
- [ ] Enable Easier UVM (http://www.doulos.com/easier)
- [ ] Enable VUnit (https://vunit.github.io/index.html)

Tools & Simulators (http://eda-playground.readthedocs.org/en/latest/intro.html#tools-simulators)

Icarus Verilog 12.0

Compile Options

-Wall -g2012

Run Options

Run Options

- [ ] Use **run.bash** shell script
- [x] Open **EPWave** after run
- [ ] Show output file after run

Output File Name

Output Filename

- [ ] Download files after run

Examples

using EDA Playground (/playgrounds?searchString=EDAPEX&language=&simulator=&methodologies=&curated=true)
VHDL (/playgrounds?searchString=&language=VHDL&simulator=&methodologies=&curated=true)
Verilog/SystemVerilog (/playgrounds?searchString=&language=SystemVerilog%2FVerilog&simulator=&methodologies=&curated=true)
UVM (/playgrounds?searchString=&language=SystemVerilog%2FVerilog&simulator=&methodologies=UVM&curated=true)
EasierUVM (/playgrounds?searchString=&language=SystemVerilog%2FVerilog&simulator=&methodologies=&easierUVM=true&curated=true)
SVAUnit (/playgrounds?searchString=&language=&simulator=&methodologies=&libraries=SVAUnit&curated=true)
SVUnit (/playgrounds?searchString=&language=&simulator=&methodologies=&libraries=SVUnit&curated=true)
VUnit (Verilog/SV) (/playgrounds?searchString=&language=&simulator=&methodologies=&vunit=true&curated=true)
VUnit (VHDL) (/playgrounds?searchString=&language=VHDL&simulator=&methodologies=&vunitVhdl=true&curated=true)
TL-Verilog (/playgrounds?searchString=&language=&simulator=&methodologies=&svx=true&curated=true)
e + Verilog (/playgrounds?searchString=&language=Specman+e+%2B+SV%2FVerilog&simulator=&methodologies=&curated=true)
Python + Verilog (/playgrounds?

searchString=&language=Python+%2B+SV%2FVerilog&simulator=&methodologies=&curated=true)

Python Only (/playgrounds?searchString=&language=Python+2.7+only&simulator=&methodologies=&curated=true)

C++/SystemC (/playgrounds?searchString=&language=C%2B%2B+only&simulator=&methodologies=&curated=true)

204

testbench.sv

```
// tb_basic_alu.v
// Test bench for basic_alu module
// Designed for EDA Playground with waveform dumping

module tb_basic_alu;

    // Declare signals that will connect to the ALU's ports
    reg  reg [7:0] A_tb;  wire  [7:0] // Test bench input for A
    Result[7:0] B_tb;    wire // Test bench input for B
    CarryOut_tb; [2:0] OpCode_tb;wire // Test bench input for OpCode
    Zero_tb;                  // Test bench output for Result
                              // Test bench output for CarryOut
                              // Test bench output for Zero

    // Instantiate the Design Under Test (DUT) - our basic_alu
    basic_alu dut (
        .A        (A_tb),
.B        (B_tb),         .OpCode
  (OpCode_tb),         .Result
(Result_tb),           .CarryOut
(CarryOut_tb),         .Zero
(Zero_tb)
    );

    // Initial block to apply stimulus and configure waveform dumping
    initial begin
        // --- Waveform Dumping Configuration (IMPORTANT for EDA Playground EPWave) --

        $dumpfile("basic_alu.vcd");    // Specify the VCD file name
         $dumpvars(0, tb_basic_alu); // Dump all signals within tb_basic_alu hierarchy
                                     // This includes signals in 'dut' (basic_alu) as
well.

               // ------------------------------------------------------------------------

                  $display("------------------------------------------------------------------");
                         $display("Starting Basic ALU Test Bench (for EDA Playground)");
                         $display("Time\tA\tB\tOpCode\tResult\tCarry\tZero\tOperation");
                  $display("------------------------------------------------------------------");

                                        // Test Case 1: ADD (5 + 3 = 8)
                                                 A_tb = 8'd5;
                                                 B_tb = 8'd3;
                                              OpCode_tb = 3'b000; // ADD
                                    #10; // Wait 10 time units for combinational logic to settle
      $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tADD (5 + 3 = 8)", $time, A_tb, B_tb, OpCode_tb, Result_tb, CarryOut_tb,
                                                 Zero_tb);
       if (Result_tb !== 8'd8 || CarryOut_tb !== 1'b0 || Zero_tb !== 1'b0) $display("Test Failed for ADD (5+3)!");

                                  // Test Case 2: ADD (250 + 10 = 260, but 8-bit overflow)
                                                   A_tb = 8'd250; // 0xFA
                                                   B_tb = 8'd10;  // 0x0A
                                                OpCode_tb = 3'b000; // ADD
                                                         #10;
    $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tADD (250 + 10 = 260 -> 4, Carry)", $time, A_tb, B_tb, OpCode_tb, Result_tb,
                                               CarryOut_tb, Zero_tb);
      if (Result_tb !== 8'd4 || CarryOut_tb !== 1'b1 || Zero_tb !== 1'b0) $display("Test Failed for ADD (250+10)!");

                                              // Test Case 3: SUB (10 - 3 = 7)
                                                       A_tb = 8'd10;
                                                       B_tb = 8'd3;
                                                    OpCode_tb = 3'b001; // SUB
                                                             #10;
      $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tSUB (10 - 3 = 7)", $time, A_tb, B_tb, OpCode_tb, Result_tb, CarryOut_tb,
                                                    Zero_tb);
      if (Result_tb !== 8'd7 || CarryOut_tb !== 1'b1 || Zero_tb !== 1'b0) $display("Test Failed for SUB (10-3)!"); //
                                               CarryOut=1 for no borrow
```

```
        // Test Case 4: SUB (3 - 10 = -7, 8-bit underflow)
        A_tb = 8'd3;
        B_tb = 8'd10;
        OpCode_tb = 3'b001; // SUB
        #10;
$display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tSUB (3 - 10 = -7 -> 0xF9, Borrow)", $time, A_tb, B_tb, OpCode_tb, Result_tb,
        CarryOut_tb, Zero_tb);
    // -7 in 8-bit two's complement is 11111001 (0xF9)
 if (Result_tb !== 8'hF9 || CarryOut_tb !== 1'b0 || Zero_tb !== 1'b0) $display("Test Failed for SUB (3-10)!"); //
        CarryOut=0 for borrow

    // Test Case 5: AND (0xF0 & 0x0F = 0x00)
        A_tb = 8'hF0; // 11110000
        B_tb = 8'h0F; // 00001111
        OpCode_tb = 3'b010; // AND
        #10;
    $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tAND (0xF0 & 0x0F = 0x00)", $time, A_tb, B_tb, OpCode_tb, Result_tb,
        CarryOut_tb, Zero_tb);
    if (Result_tb !== 8'h00 || CarryOut_tb !== 1'b0 || Zero_tb !== 1'b1) $display("Test Failed for AND (0xF0 &
        0x0F)!");

    // Test Case 6: OR (0xF0 | 0x0F = 0xFF)
        A_tb = 8'hF0;
        B_tb = 8'h0F;
        OpCode_tb = 3'b011; // OR
        #10;
    $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tOR (0xF0 | 0x0F = 0xFF)", $time, A_tb, B_tb, OpCode_tb, Result_tb,
        CarryOut_tb, Zero_tb);
if (Result_tb !== 8'hFF || CarryOut_tb !== 1'b0 || Zero_tb !== 1'b0) $display("Test Failed for OR (0xF0 | 0x0F)!");

    // Test Case 7: NOT (0xAA = 0x55)
        A_tb = 8'hAA; // 10101010
    B_tb = 8'h00; // B is ignored for NOT
        OpCode_tb = 3'b100; // NOT
        #10;
 $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tNOT (~0xAA = 0x55)", $time, A_tb, B_tb, OpCode_tb, Result_tb, CarryOut_tb,
        Zero_tb);
  if (Result_tb !== 8'h55 || CarryOut_tb !== 1'b0 || Zero_tb !== 1'b0) $display("Test Failed for NOT (~0xAA)!");

    // Test Case 8: NOT (0x00 = 0xFF), check Zero flag
        A_tb = 8'h00;
        B_tb = 8'h00;
        OpCode_tb = 3'b100; // NOT
        #10;
 $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tNOT (~0x00 = 0xFF)", $time, A_tb, B_tb, OpCode_tb, Result_tb, CarryOut_tb,
        Zero_tb);
  if (Result_tb !== 8'hFF || CarryOut_tb !== 1'b0 || Zero_tb !== 1'b0) $display("Test Failed for NOT (~0x00)!");

    // Test Case 9: ADD resulting in Zero
        A_tb = 8'd255;
        B_tb = 8'd1;
        OpCode_tb = 3'b000; // ADD
        #10;
 $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tADD (255 + 1 = 0, Carry, Zero)", $time, A_tb, B_tb, OpCode_tb, Result_tb,
        CarryOut_tb, Zero_tb);
  if (Result_tb !== 8'h00 || CarryOut_tb !== 1'b1 || Zero_tb !== 1'b1) $display("Test Failed for ADD (255+1)!");

    // Test Case 10: Unassigned OpCode (should result in 'x' or default)
        A_tb = 8'h12;
        B_tb = 8'h34;
        OpCode_tb = 3'b111; // Unassigned
        #10;
 $display("%0t\t%h\t%h\t%b\t%h\t%b\t%b\tUnassigned OpCode", $time, A_tb, B_tb, OpCode_tb, Result_tb, CarryOut_tb,
        Zero_tb);
 // For unassigned, we expect 'x' values, so checking specific values isn't ideal here in Verilog 'x' propagation.

    $display("----------------------------------------------------------------");
        $display("Basic ALU Test Bench Finished.");
        $finish; // End simulation
    end

endmodule
```

```
// basic_alu.v
// Basic Arithmetic Logic Unit (ALU)
// Supports ADD, SUB, AND, OR, NOT operations
// Designed for 8-bit operands

module basic_alu (
    input  [7:0] A,          // 8-bit input operand A
    input  [7:0] B,          // 8-bit input operand B
    input  [2:0] OpCode,     // 3-bit operation code
    output reg [7:0] Result,   // 8-bit output result
    output reg CarryOut,     // Carry out for addition/subtraction
    output reg Zero          // Zero flag (1 if Result is 0, 0 otherwise)
);

    //CHANGE:sum_sub_resultmustbea'reg'becauseit'sassignedinsideanalways block
    reg [8:0] sum_sub_result; //Changedfromwiretoreg

    always @(*) begin
        // Default values to prevent latches and ensure defined outputs
        Result = 8'b0;
        CarryOut = 1'b0;
        Zero = 1'b0;

        case (OpCode)
            3'b000: begin //ADD:Result=A+B
                sum_sub_result = {1'b0, A} + {1'b0, B}; //Performadditionwith9 bits to capture carry
                Result = sum_sub_result[7:0];            //Assignlower8bitsto Result
                CarryOut = sum_sub_result[8];            //CarryOutisthe9thbit
            end
            3'b001: begin //SUB:Result=A-B(implementedasA+(~B)+1'b1,using two's complement)
                sum_sub_result = {1'b0, A} + {1'b0, ~B} + 1'b1; //Performaddition with 9 bits
                Result = sum_sub_result[7:0];         //Assignlower8bits to Result
                // CarryOut for subtraction typically indicates no borrow (CarryOut = 1 if no borrow)
                CarryOut = sum_sub_result[8];
            end
            3'b010: begin //AND:Result=A&B(BitwiseAND)
                Result = A & B;
            end
            3'b011: begin //OR:Result=A|B(BitwiseOR)
                Result = A | B;
            end
            3'b100: begin //NOT:Result=~A(BitwiseNOTofA,Bisignored)
                Result = ~A;
            end
            default: begin //ForanyunassignedOpCode,setResultto'x'for unknown/undefined
                Result = 8'hXX; //'x'forunknown/undefinedbehavior
                CarryOut = 1'bX;
                Zero = 1'bX;
            end
        endcase

        // Update Zero flag based on the computed Result
        // This check is outside the case statement to apply to all valid operations.
        if (Result == 8'b0) begin
            Zero = 1'b1;
        end else begin
            Zero = 1'b0;
        end

    end

endmodule
```

Log Share

```
[2025-07-23 05:44:07 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.o
VCD info: dumpfile basic_alu.vcd opened for output.
--------------------------------------------------------------------
Starting Basic ALU Test Bench (for EDA Playground)
Time    A       B       OpCode Result Carry    Zero    Operation
--------------------------------------------------------------------
10      05      03      000     08      0       0       ADD (5 + 3 = 8)
20      fa      0a      000     04      1       0       ADD (250 + 10 = 260 -> 4, Carry)
30      0a      03      001     07      1       0       SUB (10 - 3 = 7)
40      03      0a      001     f9      0       0       SUB (3 - 10 = -7 -> 0xF9, Borrow)
50      f0      0f      010     00      0       1       AND (0xF0 & 0x0F = 0x00)
60      f0      0f      011     ff      0       0       OR (0xF0 | 0x0F = 0xFF)
70      aa      00      100     55      0       0       NOT (~0xAA = 0x55) NOT (~0x00 =
80      00      00      100     ff      0       0       0xFF) ADD (255 + 1 = 0, Carry,
90      ff      01      000     00      1       1       Zero) Unassigned OpCode
100     12      34      111     xx      x       0
--------------------------------------------------------------------
Basic ALU Test Bench Finished.
testbench.sv:121: $finish called at 100 (1s)
Finding VCD file...
./basic_alu.vcd
[2025-07-23 05:44:08 UTC] Opening EPWave...
Done
```