1 SystemVerilog code

```
1  module Homework_4(input logic [31:0] a, b,
2                    input logic [1:0]    ALUControl,
3                    output logic [31:0]  Result,
4                    output logic [3:0]   ALUFlags);
5
6      // variables for the adder
7      logic [31:0] b_add, Sum;
8      logic cout;
9      assign b_add = ALUControl[0] ? ~b : b;
10
11     // adder output
12     assign {cout, Sum} = a + b_add + ALUControl[0];
13
14     always_comb
15         // result of MUX4
16         case(ALUControl)
17             2'b11 : Result = a | b;
18             2'b10 : Result = a & b;
19             2'b01 : Result =  Sum;
20             2'b00 : Result =  Sum;
21         endcase
22
23     // results for ALUFlags
24     assign ALUFlags[0] = (~(ALUControl[0] ^ a[31] ^ b[31])
25                           & (a[31] ^ Sum[31]) & (~ALUControl[1]));
26     assign ALUFlags[1] = ~ALUControl[1] & cout;
27     assign ALUFlags[2] = Result == 32'b0 ? 1 : 0;
28     assign ALUFlags[3] = Result[31];
29
30 endmodule
```

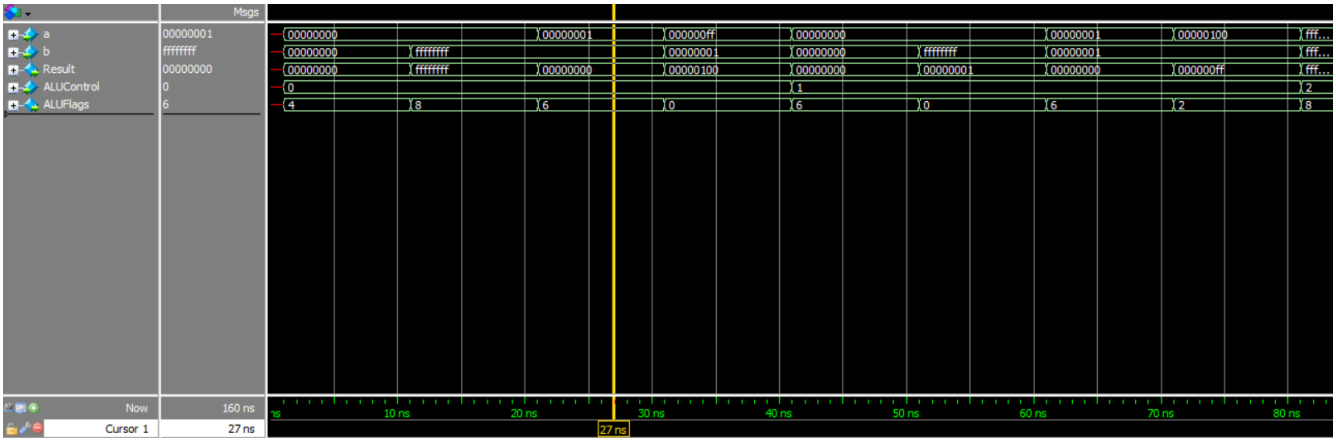
2 Testbench code

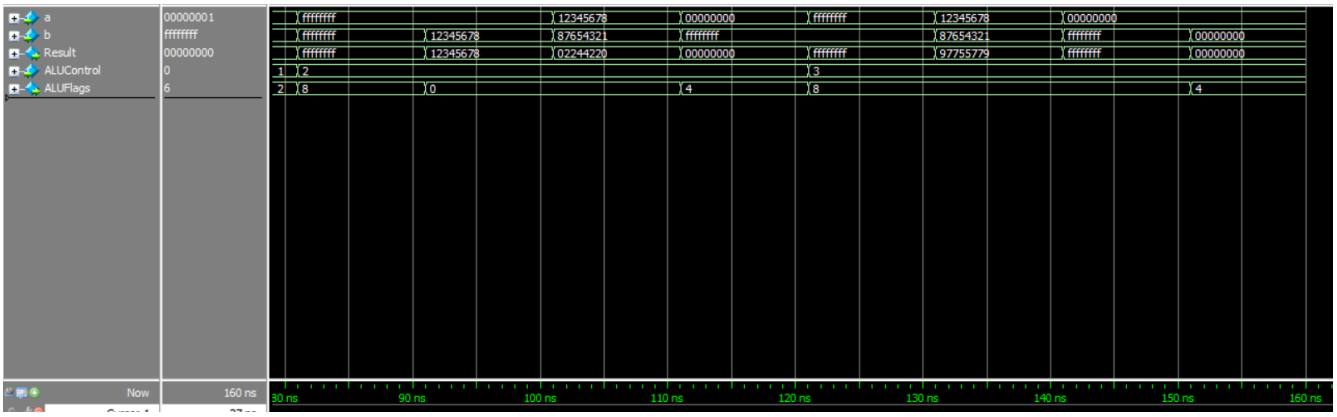
```
module Homework_4_tb();

logic [31:0] a, b, Result, Resultexpected;
logic [1:0]  ALUControl;
logic [3:0]  ALUFlags, ALUFlagsexpected;

logic clk;

logic [31:0] vectornum, errors;
logic [103:0] testvectors [0:15];  //26 numbers encoded in Hex so 4 bits

// instantiate device under test
Homework_4 dut(a, b, ALUControl, Result, ALUFlags);

always
    begin
        #5 clk = ~clk;
    end

// at start of test, load vectors
initial
    begin
        clk = 1;
        $readmemh("alu.tv", testvectors);
        vectornum = 0; errors = 0;
    end

// apply test vectors on rising edge of clk
always @(posedge clk)
    begin
        #1; {ALUControl, a, b, Resultexpected, ALUFlagsexpected} = testvectors[vectornum];
    end

// check results on falling edge of clk
always @(negedge clk)
    begin
        if ((Result !== Resultexpected) | (ALUFlags !== ALUFlagsexpected)) begin
            $display("Error: a = %h, b = %h, ALUControl = %h", a, b, ALUControl);
            $display("Result = %h (%h expected), ALUFlags = %b (%b expected)",
                     Result, Resultexpected, ALUFlags, ALUFlagsexpected);
            errors = errors + 1;
        end
        vectornum = vectornum + 1;
        if (testvectors[vectornum] === 104'hx) begin
            $display("%d tests completed with %d errors",
                vectornum, errors);
        end
    end
endmodule
```

3 ALU vector file

```
0_00000000_00000000_00000000_4
0_00000000_FFFFFFFF_FFFFFFFF_8
0_00000001_FFFFFFFF_00000000_6
0_000000FF_00000001_00000100_0
1_00000000_00000000_00000000_6
1_00000000_FFFFFFFF_00000001_0
1_00000001_00000001_00000000_6
1_00000100_00000001_000000FF_2
2_FFFFFFFF_FFFFFFFF_FFFFFFFF_8
2_FFFFFFFF_12345678_12345678_0
2_12345678_87654321_02244220_0
2_00000000_FFFFFFFF_00000000_4
3_FFFFFFFF_FFFFFFFF_FFFFFFFF_8
3_12345678_87654321_97755779_8
3_00000000_FFFFFFFF_FFFFFFFF_8
3_00000000_00000000_00000000_4
```

4 ModelSim screenshots




a
b
Result
ALUControl
ALUFlags
00000001
ffffffff
00000000
0
6
Now
160 ns
30 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
12345678
87654321
02244220
97755779
00000000
2
8
0
4
3