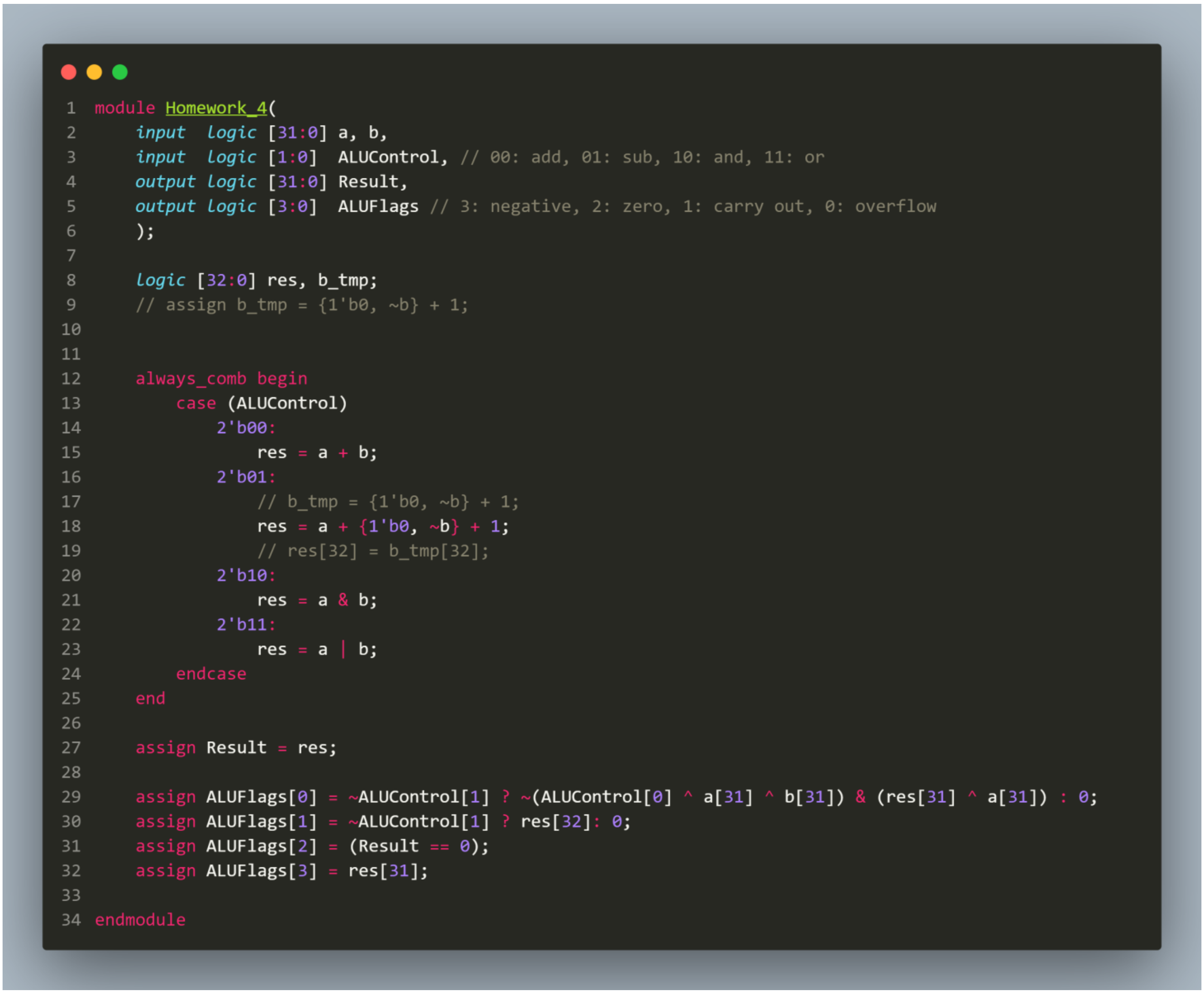
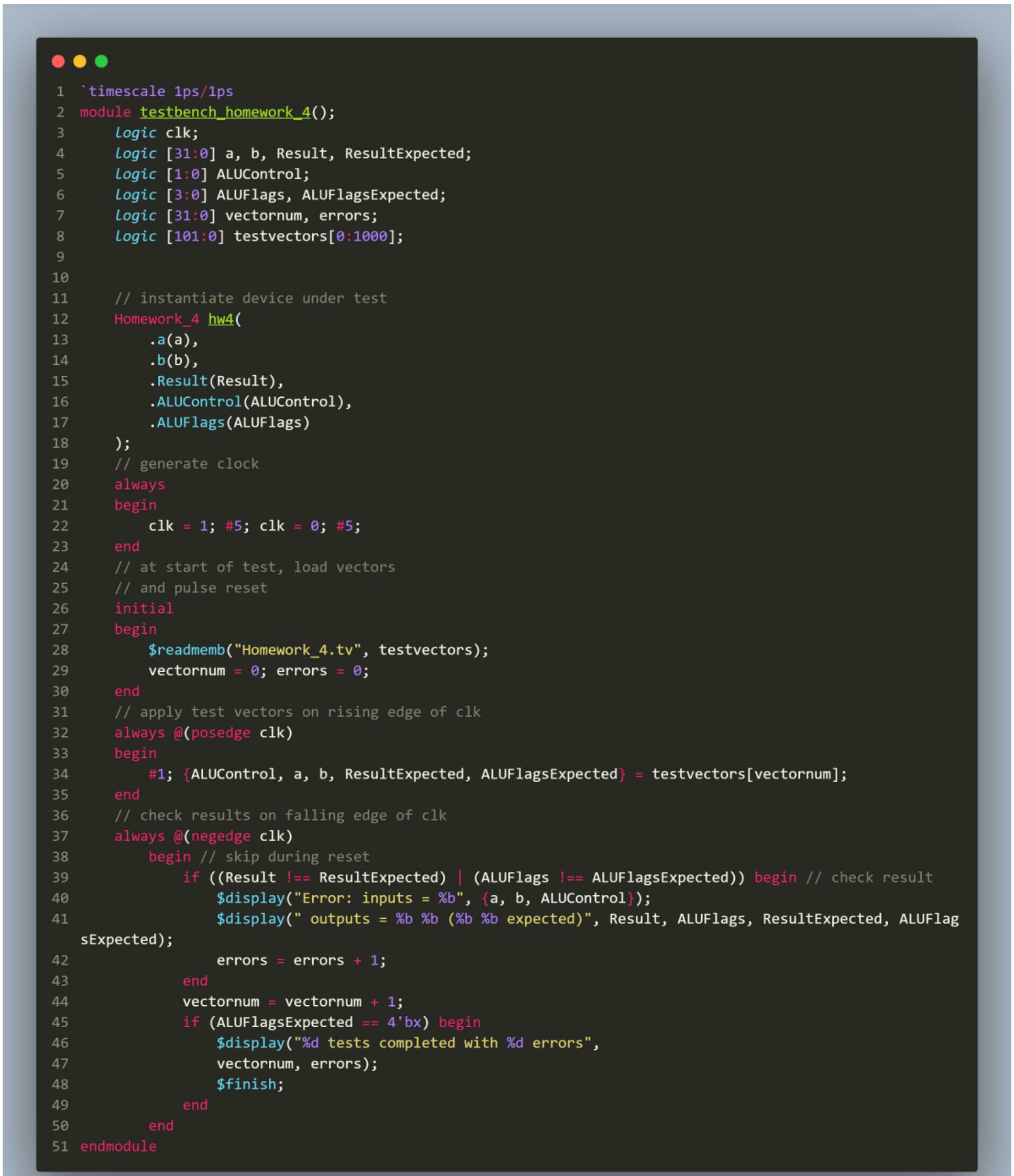
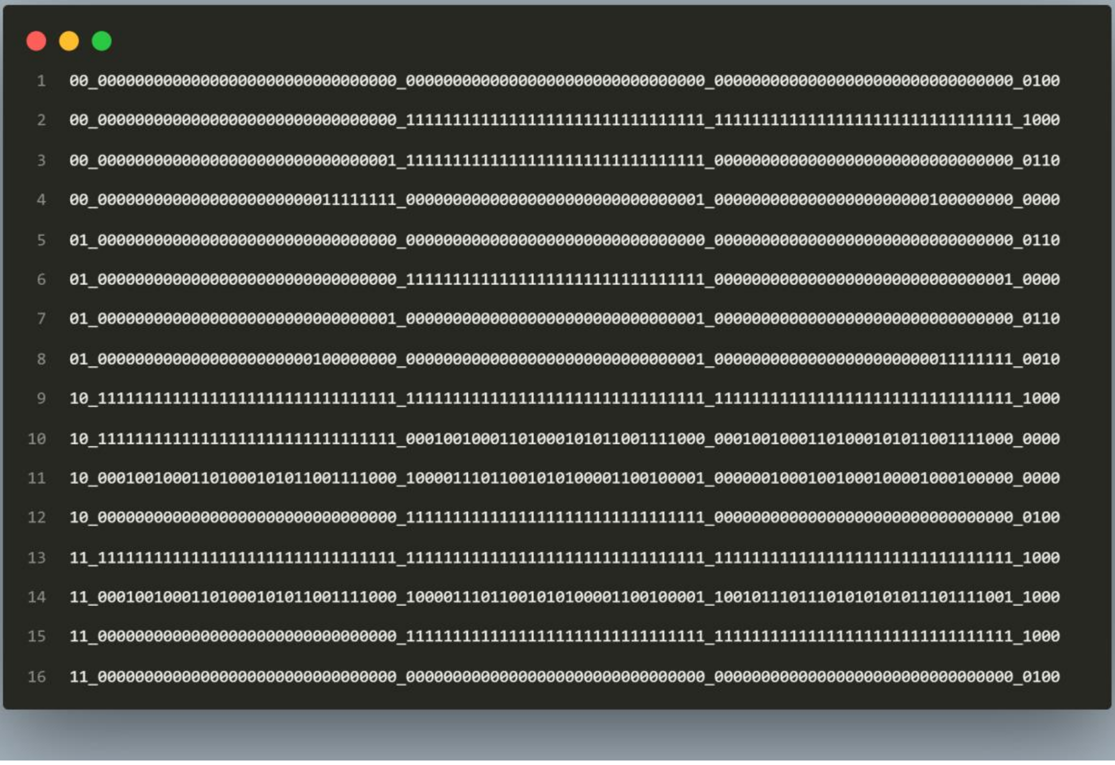


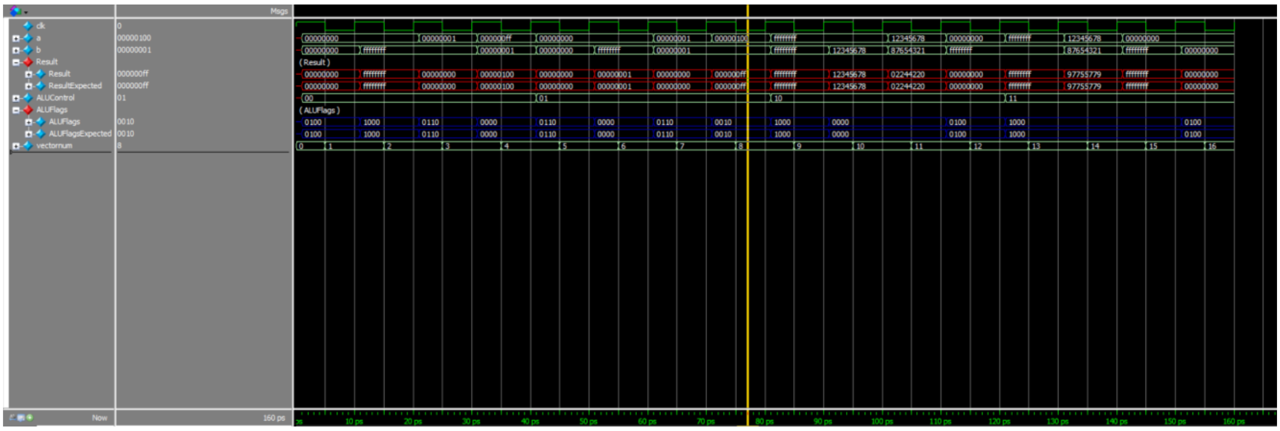
```
module Homework_4(
    input  logic [31:0] a, b,
    input  logic [1:0]  ALUControl, // 00: add, 01: sub, 10: and, 11: or
    output logic [31:0] Result,
    output logic [3:0]  ALUFlags // 3: negative, 2: zero, 1: carry out, 0: overflow
    );

    logic [32:0] res, b_tmp;
    // assign b_tmp = {1'b0, ~b} + 1;


    always_comb begin
        case (ALUControl)
            2'b00:
                res = a + b;
            2'b01:
                // b_tmp = {1'b0, ~b} + 1;
                res = a + {1'b0, ~b} + 1;
                // res[32] = b_tmp[32];
            2'b10:
                res = a & b;
            2'b11:
                res = a | b;
        endcase
    end

    assign Result = res;

    assign ALUFlags[0] = ~ALUControl[1] ? ~(ALUControl[0] ^ a[31] ^ b[31]) & (res[31] ^ a[31]) : 0;
    assign ALUFlags[1] = ~ALUControl[1] ? res[32]: 0;
    assign ALUFlags[2] = (Result == 0);
    assign ALUFlags[3] = res[31];

endmodule
```



```
`timescale 1ps/1ps
module testbench_homework_4();
    logic clk;
    logic [31:0] a, b, Result, ResultExpected;
    logic [1:0] ALUControl;
    logic [3:0] ALUFlags, ALUFlagsExpected;
    logic [31:0] vectornum, errors;
    logic [101:0] testvectors[0:1000];


    // instantiate device under test
    Homework_4 hw4(
        .a(a),
        .b(b),
        .Result(Result),
        .ALUControl(ALUControl),
        .ALUFlags(ALUFlags)
    );
    // generate clock
    always
    begin
        clk = 1; #5; clk = 0; #5;
    end
    // at start of test, load vectors
    // and pulse reset
    initial
    begin
        $readmemb("Homework_4.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 // skip during reset
            if ((Result !== ResultExpected) | (ALUFlags !== ALUFlagsExpected)) begin // check result
                $display("Error: inputs = %b", {a, b, ALUControl});
                $display(" outputs = %b %b (%b %b expected)", Result, ALUFlags, ResultExpected, ALUFlagsExpected);
                errors = errors + 1;
            end
            vectornum = vectornum + 1;
            if (ALUFlagsExpected == 4'bx) begin
                $display("%d tests completed with %d errors",
                vectornum, errors);
                $finish;
            end
        end
endmodule
```



```
00_00000000000000000000000000000000_00000000000000000000000000000000_00000000000000000000000000000000_0100
00_00000000000000000000000000000000_11111111111111111111111111111111_11111111111111111111111111111111_1000
00_00000000000000000000000000000001_11111111111111111111111111111111_00000000000000000000000000000000_0110
00_00000000000000000000000011111111_00000000000000000000000000000001_00000000000000000000000100000000_0000
01_00000000000000000000000000000000_00000000000000000000000000000000_00000000000000000000000000000000_0110
01_00000000000000000000000000000000_11111111111111111111111111111111_00000000000000000000000000000001_0000
01_00000000000000000000000000000001_00000000000000000000000000000001_00000000000000000000000000000000_0110
01_00000000000000000000000100000000_00000000000000000000000000000001_00000000000000000000000011111111_0010
10_11111111111111111111111111111111_11111111111111111111111111111111_11111111111111111111111111111111_1000
10_11111111111111111111111111111111_00010010001101000101011001111000_00010010001101000101011001111000_0000
10_00010010001101000101011001111000_10000111011001010100001100100001_00000010001001000100001000100000_0000
10_00000000000000000000000000000000_11111111111111111111111111111111_00000000000000000000000000000000_0100
11_11111111111111111111111111111111_11111111111111111111111111111111_11111111111111111111111111111111_1000
11_00010010001101000101011001111000_10000111011001010100001100100001_10010111011101010101011101111001_1000
11_00000000000000000000000000000000_11111111111111111111111111111111_11111111111111111111111111111111_1000
11_00000000000000000000000000000000_00000000000000000000000000000000_00000000000000000000000000000000_0100
```



Msgs

clk 0
a 00000100
b 00000001
Result
Result 000000ff
ResultExpected 000000ff
ALUControl 01
ALUFlags
ALUFlags 0010
ALUFlagsExpected 0010
vectornum 8

Now 160 ps

00000000 00000001 000000ff 00000000 00000001 00000100 ffffffff 12345678 00000000 ffffffff 12345678 00000000
00000000 ffffffff 00000001 00000000 ffffffff 00000001 ffffffff 12345678 87654321 ffffffff 87654321 ffffffff 00000000
(Result)
00000000 ffffffff 00000000 00000100 00000000 00000001 00000000 000000ff ffffffff 12345678 02244220 00000000 ffffffff 97755779 ffffffff 00000000
00000000 ffffffff 00000000 00000100 00000000 00000001 00000000 000000ff ffffffff 12345678 02244220 00000000 ffffffff 97755779 ffffffff 00000000
00 01 10 11
(ALUFlags)
0100 1000 0110 0000 0110 0000 0110 0010 1000 0000 0100 1000 0100
0100 1000 0110 0000 0110 0000 0110 0010 1000 0000 0100 1000 0100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ps
10 ps
20 ps
30 ps
40 ps
50 ps
60 ps
70 ps
80 ps
90 ps
100 ps
110 ps
120 ps
130 ps
140 ps
150 ps
160 ps