

Lab 2: FPGA Tools & Combinational Logic Design SOLUTIONS

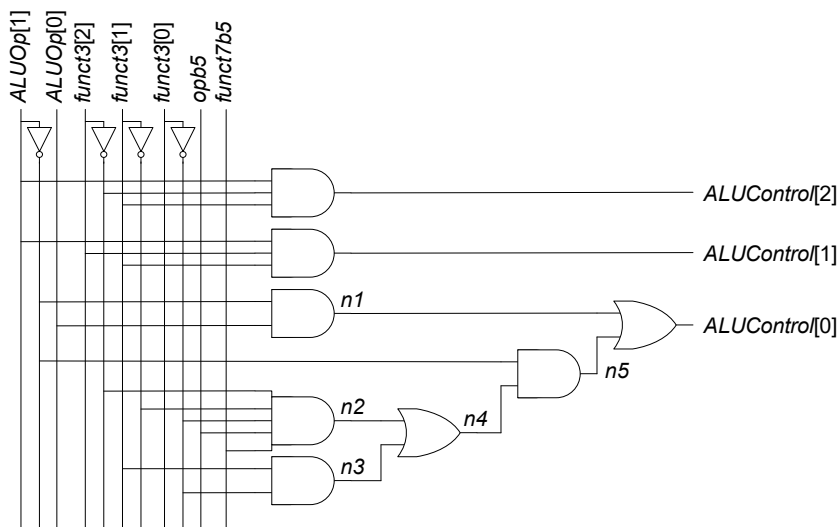
Digital Design and Computer Architecture: RISC-V Edition (Harris & Harris, Elsevier © 2021)

1. Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught.

2. [2] Boolean equations for your ALU Decoder

```
ALUControl[2] = ALUOp[1] & ~funct3[2] & funct3[1]
ALUControl[1] = ALUOp[1] & funct3[2] & funct3[1]
ALUControl[0] = ~ALUOp[1] & ALUOp[0] | ALUOp[1] &
                (~funct3[2] & ~funct3[1] & ~funct3[0] & opb5 & funct7b5 |
                funct3[1] & ~funct3[0]);
```

3. [1] Gate-level schematic of your ALU Decoder



4. [2] Structural Verilog code for your ALU Decoder

```
module aludec(input logic opb5,
              input logic [2:0] funct3,
              input logic funct7b5,
              input logic [1:0] ALUOp,
              output logic [2:0] ALUControl);

    logic [3:0] funct3b; // Inverse of funct3 - Verilog won't implicitly declare arrays
    logic n1, n2, n3, n4, n5; // Best practice to explicitly declare all internal signals
    logic ALUOpb1b;

    not not1(ALUOpb1b, ALUOp[1]);
    not not2(funct3b[2], funct3[2]);
    not not3(funct3b[1], funct3[1]);
    not not4(funct3b[0], funct3[0]);
    and a1(ALUControl[2], ALUOp[1], funct3b[2], funct3[1]);
    and a2(ALUControl[1], ALUOp[1], funct3[2], funct3[1]);
    and a3(n1, ALUOpb1b, ALUOp[0]);
    and a4(n2, funct3b[2], funct3b[1], funct3b[0], opb5, funct7b5);
    and a5(n3, funct3[1], funct3b[0]);
```

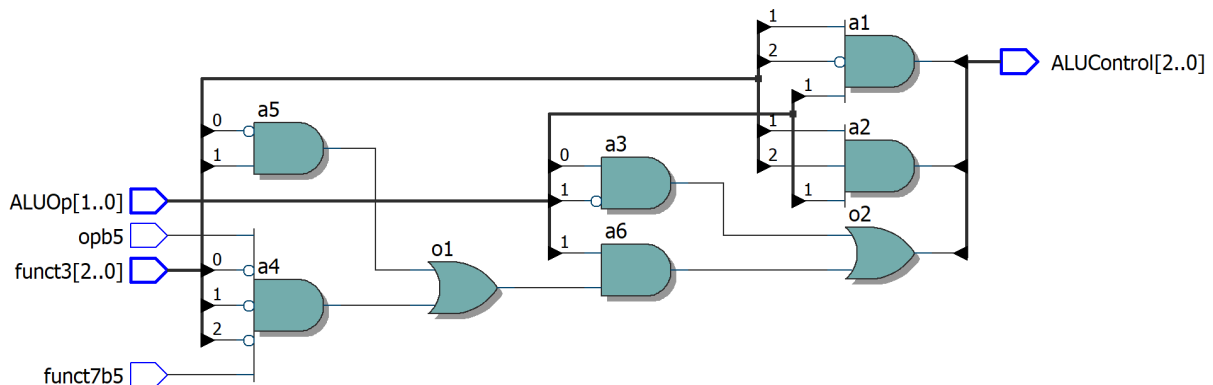
```

    or o1(n4, n2, n3);
    and a6(n5, ALUOp[1], n4);
    or o2(ALUControl[0], n1, n5);
endmodule

```

Graders: please give full credit for using inline NOTs (e.g. ~func3[2] in place of fb[2]), although this is technically mixing structural and behavioral Verilog.

5. [1] RTL Viewer schematics of your synthesized ALU Decoder



6. [1] Self-checking test bench for your ALU Decoder with a test vector file

```

module testbench_aludec();
    logic      clk, reset;
    logic      opb5;
    logic [2:0] func3;
    logic      func7b5;
    logic [1:0] ALUOp;
    logic [2:0] ALUControl, ALUControlexpected;
    logic [31:0] vectornum, errors;
    logic [9:0] testvectors[10000:0];

    // instantiate device under test
    aludec dut(opb5, func3, func7b5, ALUOp, ALUControl);

    // generate clock
    always
    begin
        clk = 1; #5; clk = 0; #5;
    end

    // at start of test, load vectors
    // and pulse reset
    initial
    begin
        $readmemb("aludec.tv", testvectors);
        vectornum = 0; errors = 0;
        reset = 1; #22; reset = 0;
    end

    // apply test vectors on rising edge of clk
    always @(posedge clk)
    begin
        #1; {ALUOp, func3, opb5, func7b5, ALUControlexpected} =
        testvectors[vectornum];
        vectornum++;
    end
end

```

```

// check results on falling edge of clk
always @(negedge clk)
    if (~reset) begin // skip during reset
        if (ALUControl != ALUControlExpected) begin // check result
            $display("Error: inputs = %b, %b, %b, %b", ALUOp, funct3, opb5,
funct7b5);
            $display("          outputs      = %b      (%b      expected)",ALUControl,
ALUControlExpected);
            errors = errors + 1;
        end
        vectornum = vectornum + 1;
        if (testvectors[vectornum] === 10'bx) begin
            $display("%d tests completed with %d errors",
                vectornum, errors);
            $stop;
        end
    end
end
endmodule

```

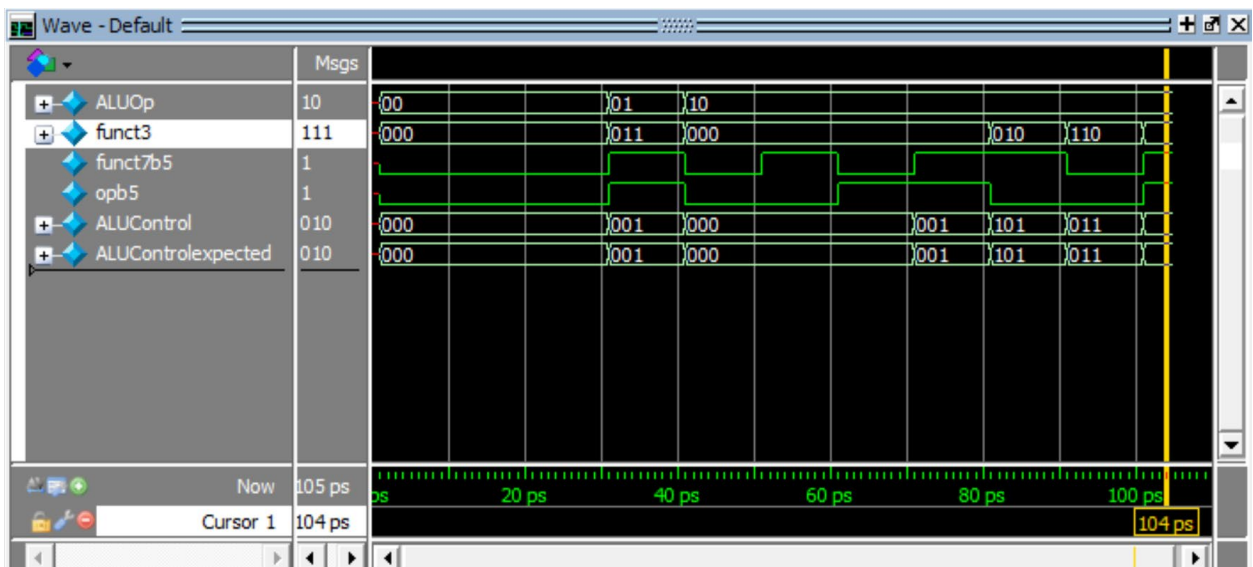
Test vectors

```

// ALUOp_funct3_opb5_funct7b5_ALUControlExpected
00_000_00_000
01_011_11_001
10_000_00_000
10_000_01_000
10_000_10_000
10_000_11_001
10_010_01_101
10_110_00_011
10_111_11_010

```

7. [2] Simulation waveforms showing the ALU Decoder simulation. Display the signals in binary in this order (top to bottom): ALUOp, funct3, opb5, funct7b5, ALUControl, ALUControlExpected. Did it work correctly? **YES**



8. [1] Did the ALU Decoder function correctly on the DE0-CV Board? **YES**