

# ELC 2137 Lab 9: ALU with Input Register

Ashlie Lackey

April 2, 2020

## Summary

The goal of this lab was to start on building a small calculator by creating an ALU and using two registers to do some mathematical operations including: addition, subtraction, AND, OR, and XOR, along with a default that returns the first number you input. The registers are used to store the numbers when inputting, as one input comes from the switches and the other comes from the register in which it is stored. Additionally, the switches are also used to specify which operation is to be performed. During this lab skills were gained to learn the differences between combinational and sequential logic and how to implement in Verilog, knowledge and some implementation of SR latches, D latches, D flip flops, and D registers, and reuse of modules from previous labs.

## Results

Simulation waveforms, expected results tables, and pictures of the operations testing on the Basys3 board are included below to demonstrate that the ALU built during this lab works correctly.

### Expected results tables

Table 1: *register* expected results table

Time (ns):	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50	50-55
D (hex)	0	0	A	A	3	3	0	0	0→6	6	6
clk	0	1	0	1	0	1	0	1	0	1	0
en	0	0	1	1	1→0	0→1	1→0	0	0→1	1	1
rst	0	0→1	0	0	0	0	0	0	0	0	0
Q (hex)	X	X→0	0	A	A	A	A	A	A	6	6

Table 2: *alu* expected results table skeleton

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	6	0	C	A	0	1
in1	7	2	4	A	3	0
op	default	ADD(0)	SUB(1)	AND(2)	OR(3)	XOR(4)
out	6	2	8	A	3	1

## Simulation Waveforms

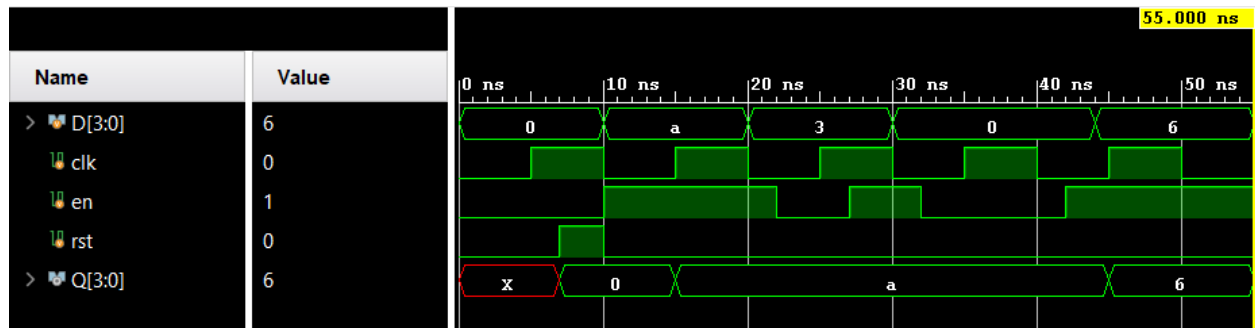


Figure 1: *register testbench* Simulation Waveform

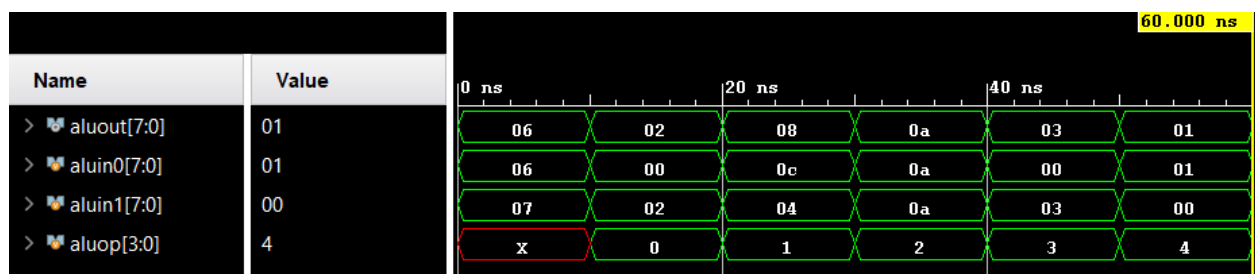


Figure 2: *ALU testbench* Simulation Waveform

## Operation On-board Testing

## Code

Listing 1: register Verilog Code

---

```
'timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020 -03 -26
module register #( parameter N=1)
    (input clk , rst , en ,
     input [N -1:0] D,
     output reg [N -1:0] Q);

    always @(posedge clk , posedge rst)
    begin
        if (rst ==1)
            Q <= 0;
        else if (en==1)
            Q <= D;
        end

    // Notes:
    // - Reset is asynchronous , so this
    //   block needs to execute when rst
    //   goes high.
    // - We want enable to be synchronous
    //   (i.e. only happens on rising
    //   edge of clk), so it is left out
    //   of "sensitivity" list.
endmodule
```

---

---

### Listing 2: register testbench Verilog Code

---

```
'timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020 -03 -26
module register_test ();
    reg [3:0] D;
    reg clk , en , rst;
    wire [3:0] Q;
    register #(.N(4)) r(.D(D), .clk(clk),
        .en(en), .rst(rst), .Q(Q) );
    // clock runs continuously
    always begin
        clk = ~clk; #5;
    end
    // this block only runs once
    initial begin
        clk=0; en=0; rst =0; D=4'h0; #7;
        rst = 1; #3; // reset
        D = 4'hA; en = 1; rst = 0; #10;
        D = 4'h3; #2;
        en = 0; #5;
        en = 1; #3;
        D = 4'h0; #2;
        en = 0; #10;
        en = 1; #2;
        D = 4'h6; #11;
        $finish;
    end
endmodule
```

---

### Listing 3: ALU Verilog Code

---

```
'timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020 -03 -26
module alu #( parameter N=8)(output reg[N -1:0] out ,
    input [N -1:0] in0 ,
    input [N -1:0] in1,
    input [3:0] op);
    // Local parameters
    parameter ADD = 0;
    parameter SUB = 1;
    parameter AND = 2;
    parameter OR = 3;
    parameter XOR = 4;
    always @*begin
        case(op)
            ADD: out = in0 + in1; // add the remaining commands
            SUB: out = in0 - in1;
            AND: out = in0 & in1;
            OR: out = in0 | in1;
            XOR: out = in0 ^ in1;
            default: out = in0;
        endcase
    end
endmodule
```

---

Listing 4: ALU testbench Verilog Code

---

```

`timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020 -03 -26
module alu_test();
    wire [7:0] aluout;
    reg [7:0] aluin0;
    reg [7:0] aluin1;
    reg [3:0] aluop;
    alu #(N(8)) alutest(.out(aluout), .in0(aluin0), .in1(aluin1), .op(
        aluop));

    initial begin
        aluin0 = 8'h6; aluin1 = 8'h7; #10;
        aluin0 = 8'h0; aluin1 = 8'h2; aluop = 4'h0; #10;
        aluin0 = 8'hC; aluin1 = 8'h4; aluop = 4'h1; #10;
        aluin0 = 8'hA; aluin1 = 8'hA; aluop = 4'h2; #10;
        aluin0 = 8'h0; aluin1 = 8'h3; aluop = 4'h3; #10;
        aluin0 = 8'h1; aluin1 = 8'h0; aluop = 4'h4; #10;
        $finish;
    end
endmodule

```

---

Listing 5: top-lab9 Verilog Code

---

```

`timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020 -03 -26
module top_lab9(input btnU, btnD,
    input [11:0] sw,
    input clk, btnC,
    output [15:0] led);

    wire [7:0] regout1;
    register #(N(8)) reg1(.D(sw[7:0]), .clk(clk), .en(btnD), .rst(btnC), .
        Q(regout1));

    wire [7:0] aluout;
    alu #(N(8)) alutest(.out(aluout), .in0(sw[7:0]), .in1(regout1), .op(sw
        [11:8]));

    register #(N(8)) reg2(.D(aluout), .clk(clk), .en(btnU), .rst(btnC), .Q
        (led[15:8]));

    assign led[7:0] = regout1;
endmodule

```

---