ELC 2137 Lab 09: ALU with Input Register

Abigail Joseph

October 22, 2020

Summary

In this lab, we constructed two modules, a register and an ALU. The register uses a clock input to store data. The ALU implements various logical and arithmetic operations using one input to determine which operation and two others upon which the operation is done. One ALU and two registers were combined in a top-level module to connect their inputs to switches on the Basys3 and their outputs to LEDs to display the various operations as binary numbers in the LEDs.

Code

Listing 1: Register

```
module register #(parameter N=1)
    (
    input clk,
    input rst,
    input en,
    input [N-1:0] D,
    output reg [N-1:0] Q
    );

    always @(posedge clk, posedge rst)
    begin
        if (rst==1)
            Q <= 8'b00000000;
        else if (en==1)
            Q <= D;
    end


endmodule
```

Listing 2: Register Testbench

```
module register_test();

reg [3:0] D_t;
reg clk_t, en_t, rst_t;
wire [3:0] Q_t;

register #(.N(4)) dut(
    .D(D_t),
```

```
    .clk(clk_t),
    .en(en_t),
    .rst(rst_t),
    .Q(Q_t)
    );

//run clock
always begin
    clk_t= ~clk_t; #5;
end

//test
initial begin
    clk_t=0; en_t=0; rst_t=0; D_t=4'h0; #7;
    rst_t=1; #3;
    D_t=4'hA; en_t=1; rst_t=0; #10;
    D_t=4'h3;     #2;
    en_t=0;       #5;
    en_t=1;       #3;
    D_t=4'h0;     #2;
    en_t=0;       #10;
    en_t=1;       #2;
    D_t=4'h6;     #11;
    $finish;
end

endmodule
```

Listing 3: ALU

```
module alu #(parameter N=8)
    (
    input [N-1:0] in0,
    input [N-1:0] in1,
    input [3:0] op,
    output reg [N-1:0] out
    );

parameter ADD=0;
parameter SUB=1;
parameter AND=2;
parameter OR=3;
parameter XOR=4;

always @*
begin
    case(op)
        ADD: out = in0 + in1;
        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

```
module alu_test();

reg [7:0] in0_t, in1_t;
reg [3:0] op_t;
wire [7:0] out_t;


alu #(.N(8)) dut(
    .in0(in0_t),
    .in1(in1_t),
    .op(op_t),
    .out(out_t)
    );

initial begin
    in0_t= 8'b10010100; in1_t=8'b00011000; op_t=4'b0000; #10
    in0_t= 8'b10000000; in1_t=8'b00000001; op_t=4'b0001; #10
    in0_t= 8'b00111010; in1_t=8'b01011100; op_t=4'b0010; #10
    in0_t= 8'b11000011; in1_t=8'b10110100; op_t=4'b0011; #10
    in0_t= 8'b10100101; in1_t=8'b11001110; op_t=4'b0100; #10
    in0_t= 8'b00000100; in1_t=8'b00000000; op_t=4'b1000; #10
$finish;
end
endmodule
```

Listing 5: Top-Level File

```
module top_lab9(
    input btnU,
    input btnD,
    input [11:0] sw,
    input clk,
    input btnC,
    output [15:0] led
    );

    wire [7:0] Q_i1, out_i1;

register #(.N(8)) my_register0(
    .D(sw[7:0]),
    .clk(clk),
    .en(btnD),
    .rst(btnC),
    .Q(Q_i1)
    );

assign led[7:0] = Q_i1;
```

```
alu #(.N(8)) my_alu(
    .in0(sw[7:0]),
    .in1(Q_i1),
    .op(sw[11:8]),
    .out(out_i1)
    );

register #(.N(8)) my_register1(
    .D(out_i1),
    .clk(clk),
    .en(btnU),
    .rst(btnC),
    .Q(led[15:8])
    );

endmodule
```

Results

Table 1: *r*egister 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	6	6	

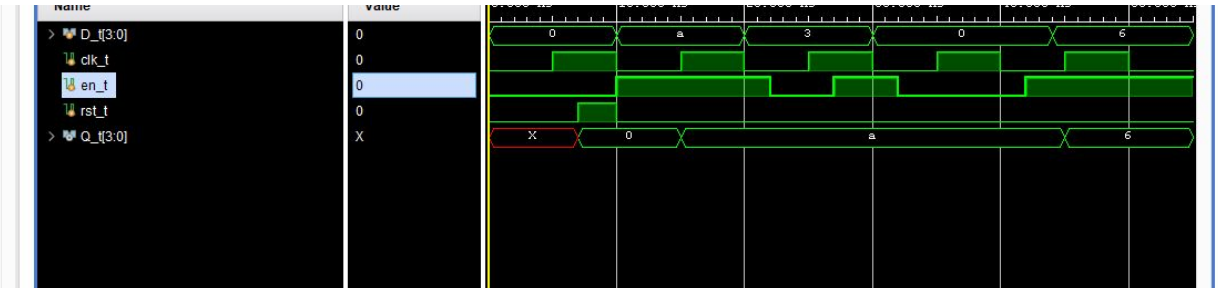

Figure 1: Register Simulation

Table 2: *a*lu expected results table skeleton

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	1001 0100	1111 1111	0011 1010	1100 0011	1010 0101	0000 0001
in1	0011 0000	0000 0001	0101 1100	1011 0100	1100 1110	0000 0000
op	0000	0001	0010	0011	0100	1000
out	1100 0100	1111 1110	0001 1000	1111 0111	0110 1011	0000 0001

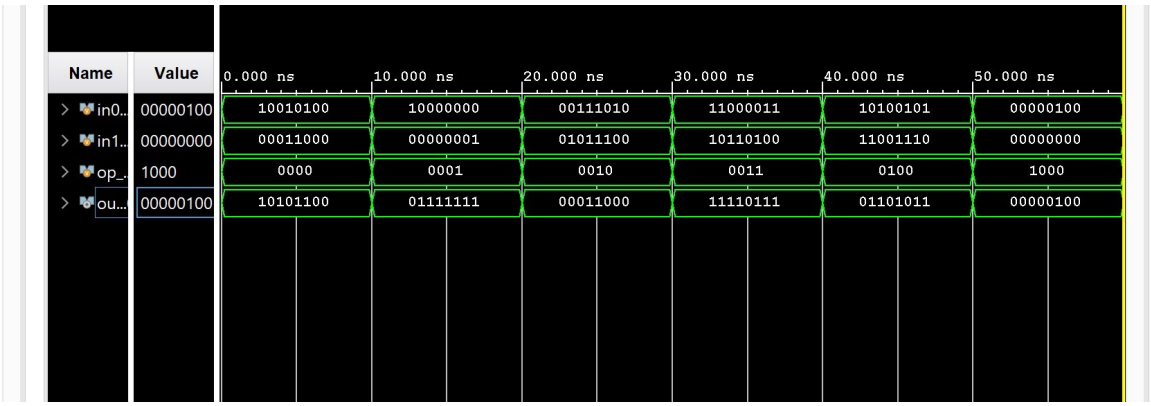

Figure 2: ALU Simulation

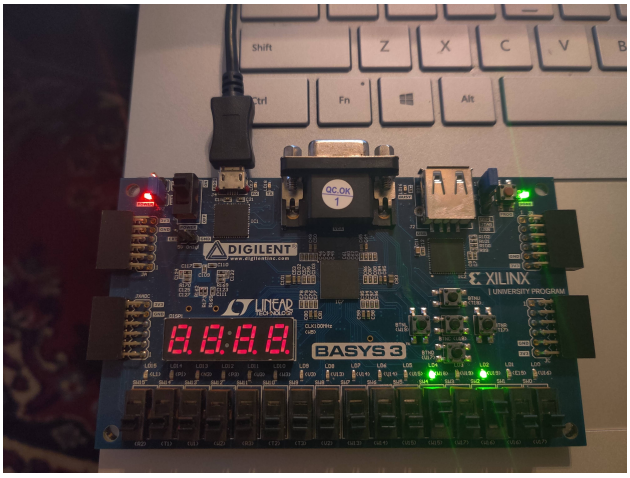

Figure 3: 10100

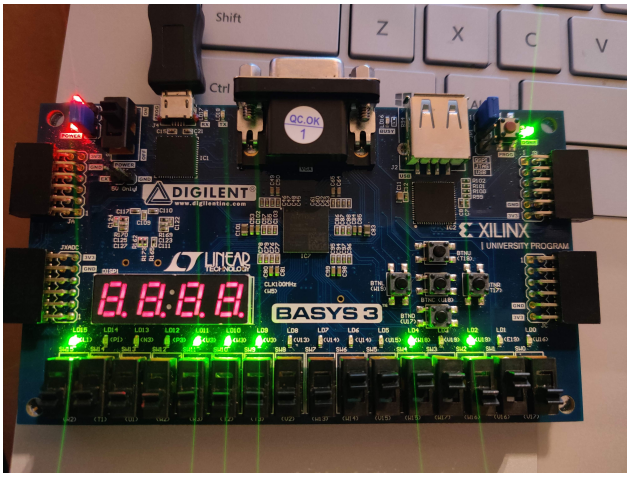

Figure 4: 10100+1111010

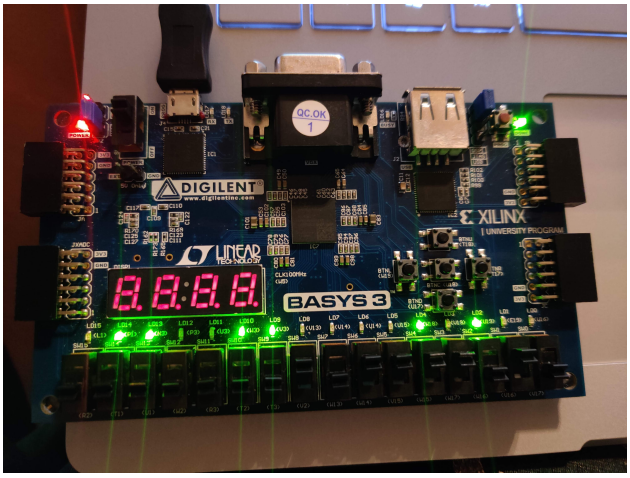

Figure 5: 10100-1111010

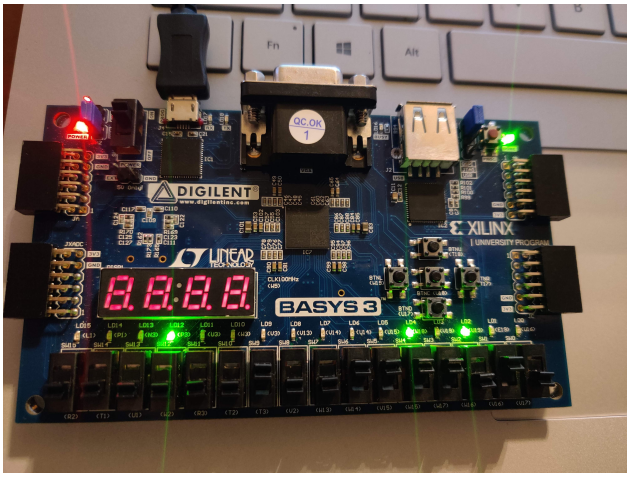

Figure 6: 10100 AND 1111010

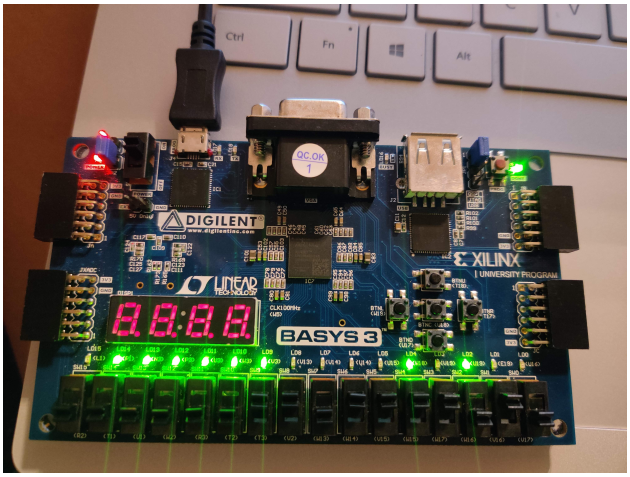

Figure 7: 10100 OR 1111010

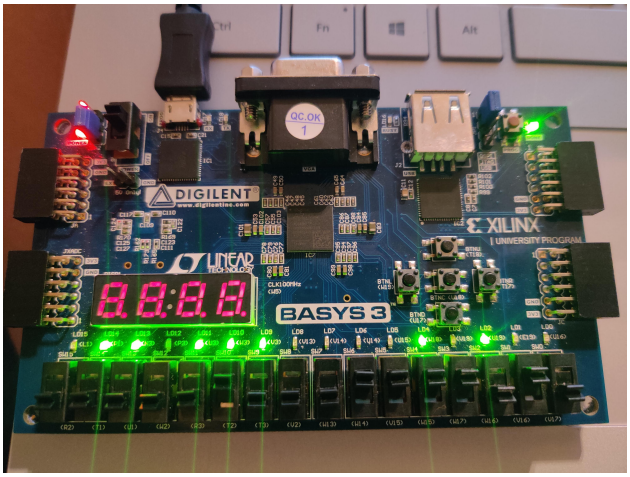

Figure 8: 10100 XOR 1111010

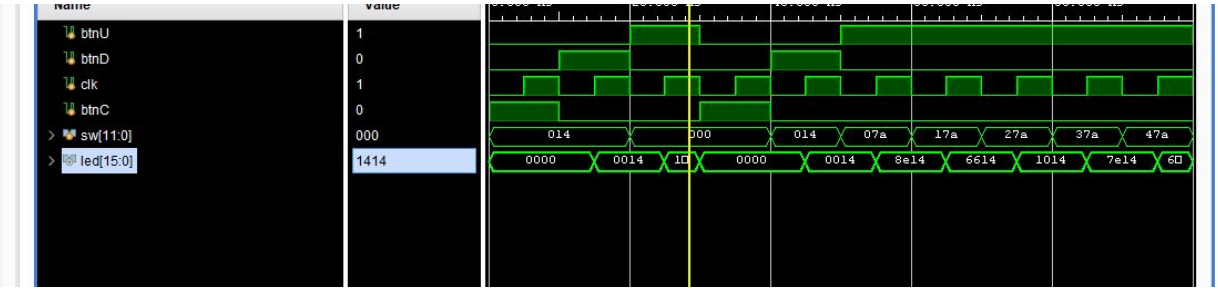

Figure 9: Basys3 Simulation