ELC 2137 Lab 9: ALU with Input Register

Megan Gordon

April 4, 2020

Summary

This lab looks to expand on previous labs by expanding into larger systems with more modules. It allowed us to learn how to create an arithmetic logic unit (ALU) capable of a few operations using hexadecimal values. In completing this lab, the following skills were gained: ability to create and implement a D register with synchronous enable and asynchronous reset, ability to create and implement an arithmetic logic unit (ALU), ability to import and modify modules, and use them to design a modular system.

Results

Below are the simulations with ERTs for 2 modules (register and an ALU) and pictures of the board for each step in the operation list for on-board testing.

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	a	a	a	a	a	a	a	6	6

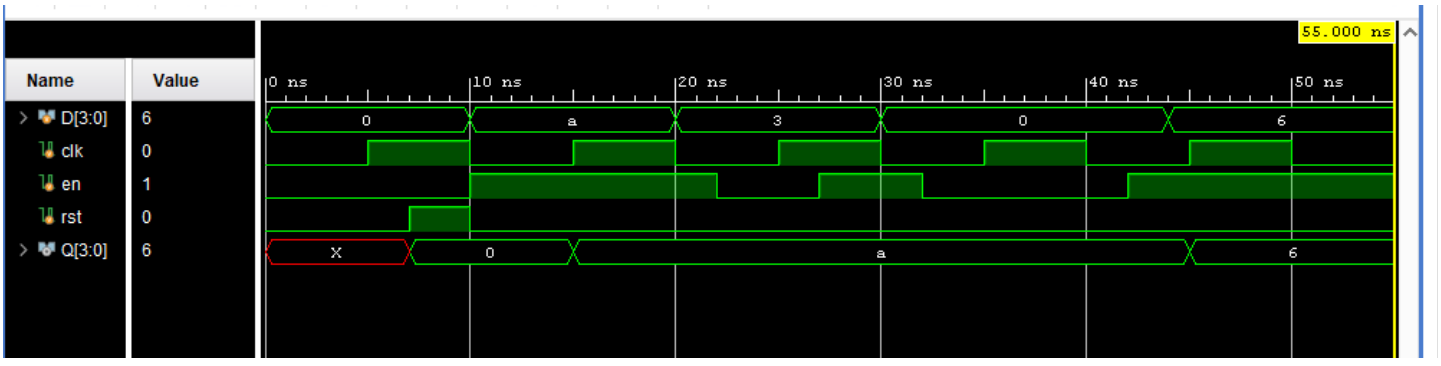

Figure 1: Register ERT and Testbench Results

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	3	3	3	3	3	3
in1	2	2	2	2	2	2
op	0	1	2	3	4	7
out	5	1	6	3	1	3

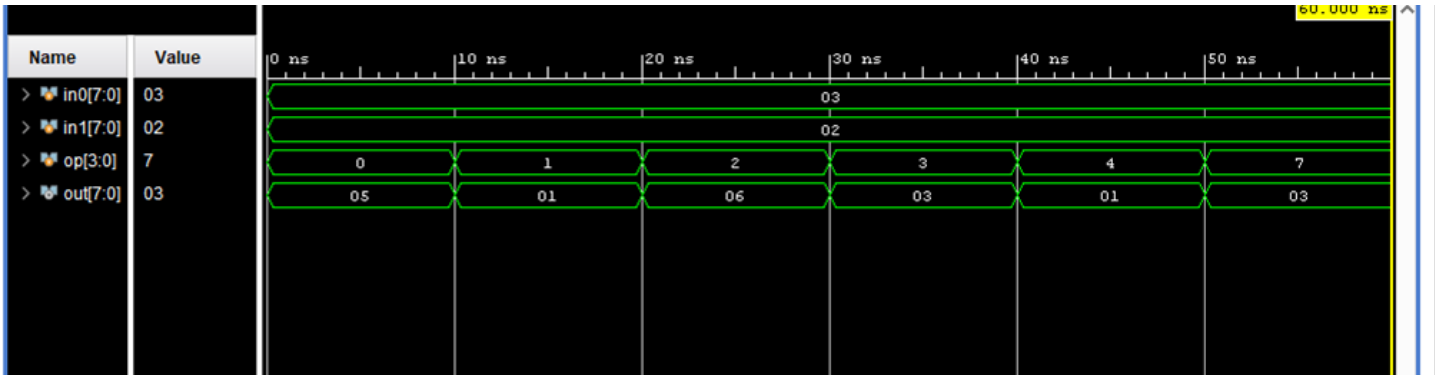

Figure 2: ALU ERT and Testbench Results

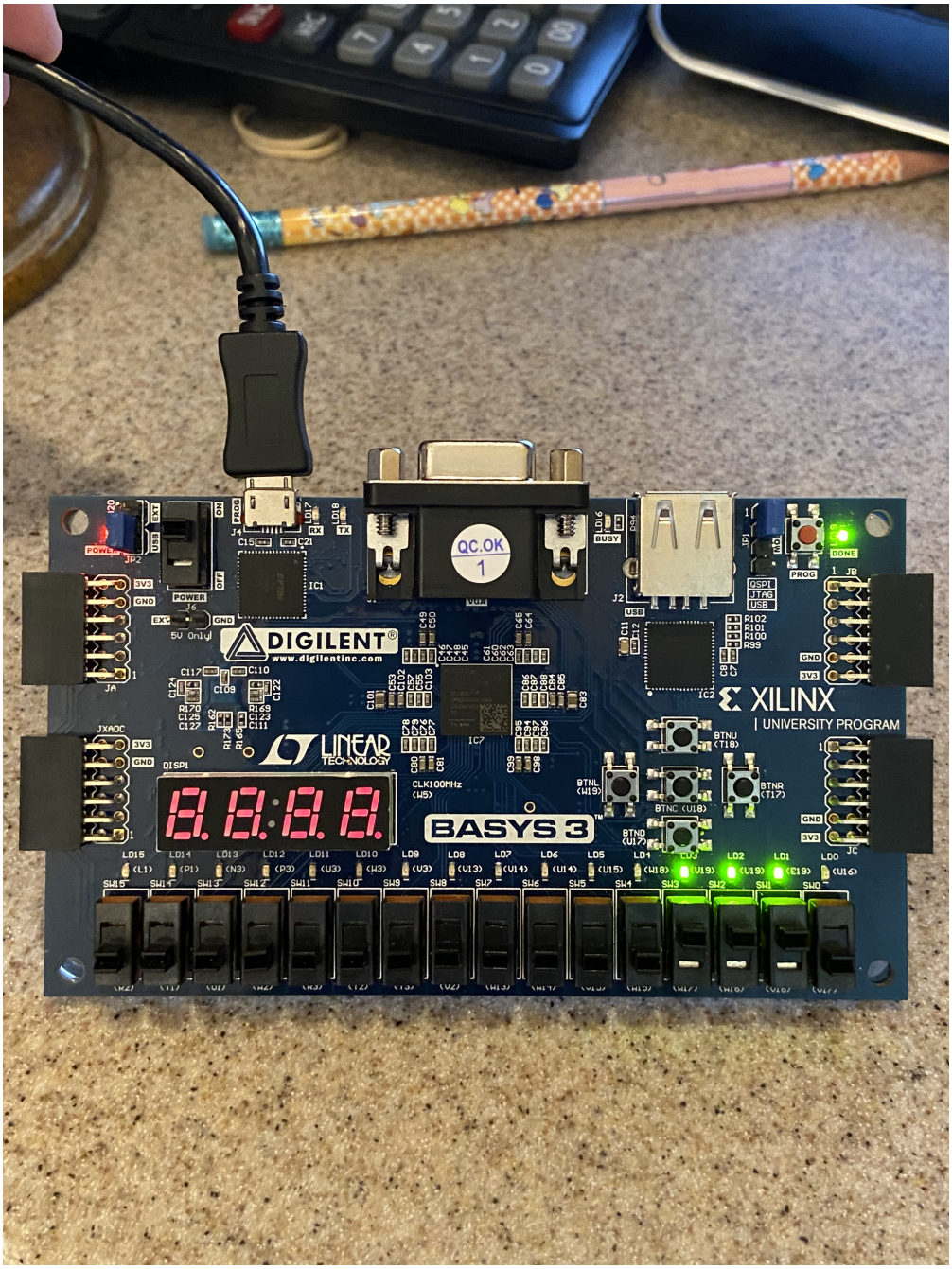

Figure 3: Operation 1-Displaying hexadecimal "14"

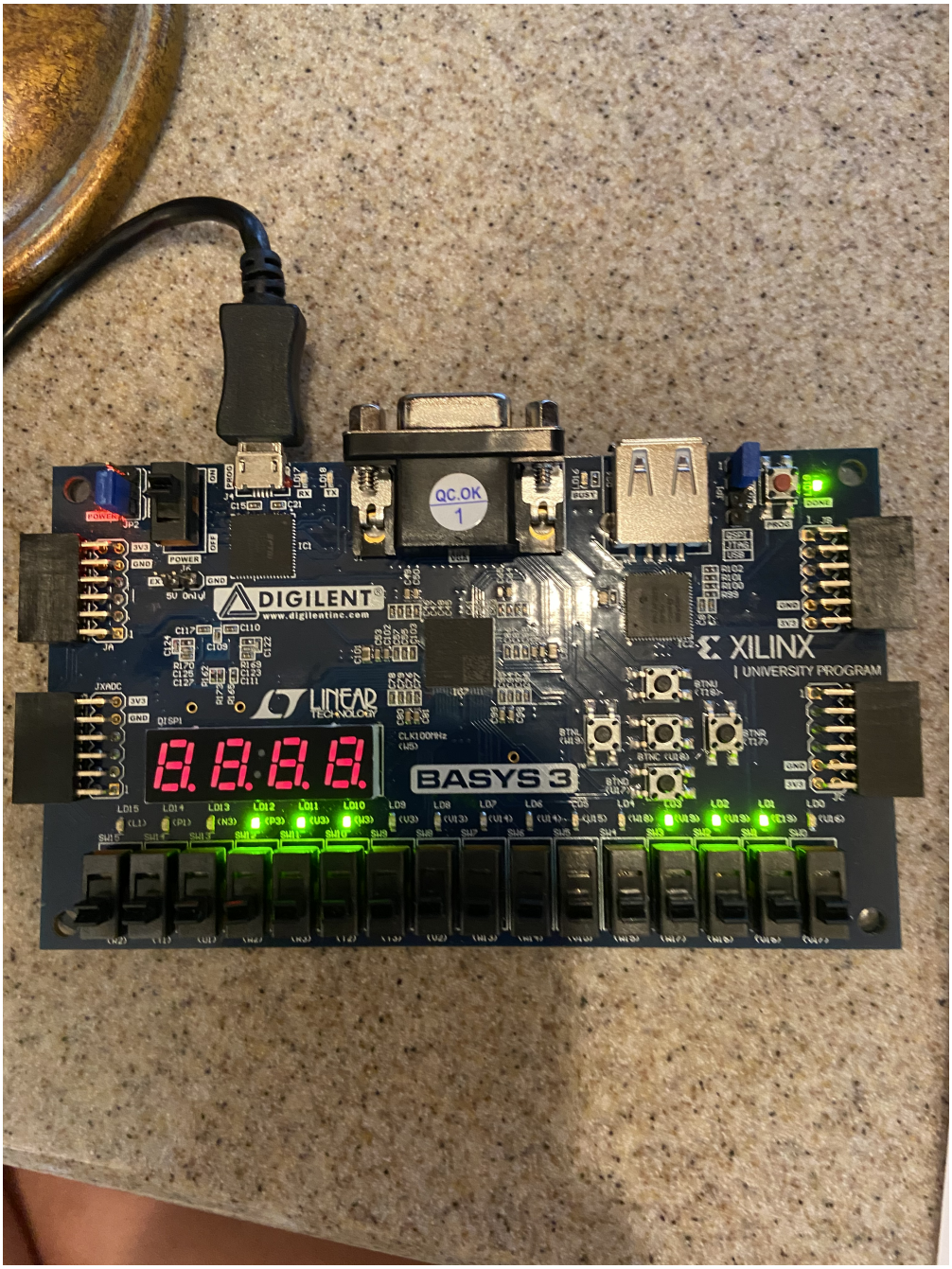

Figure 4: Operation 2-Displaying hexdecimal "14" on LEDs 15-8 and 7-0

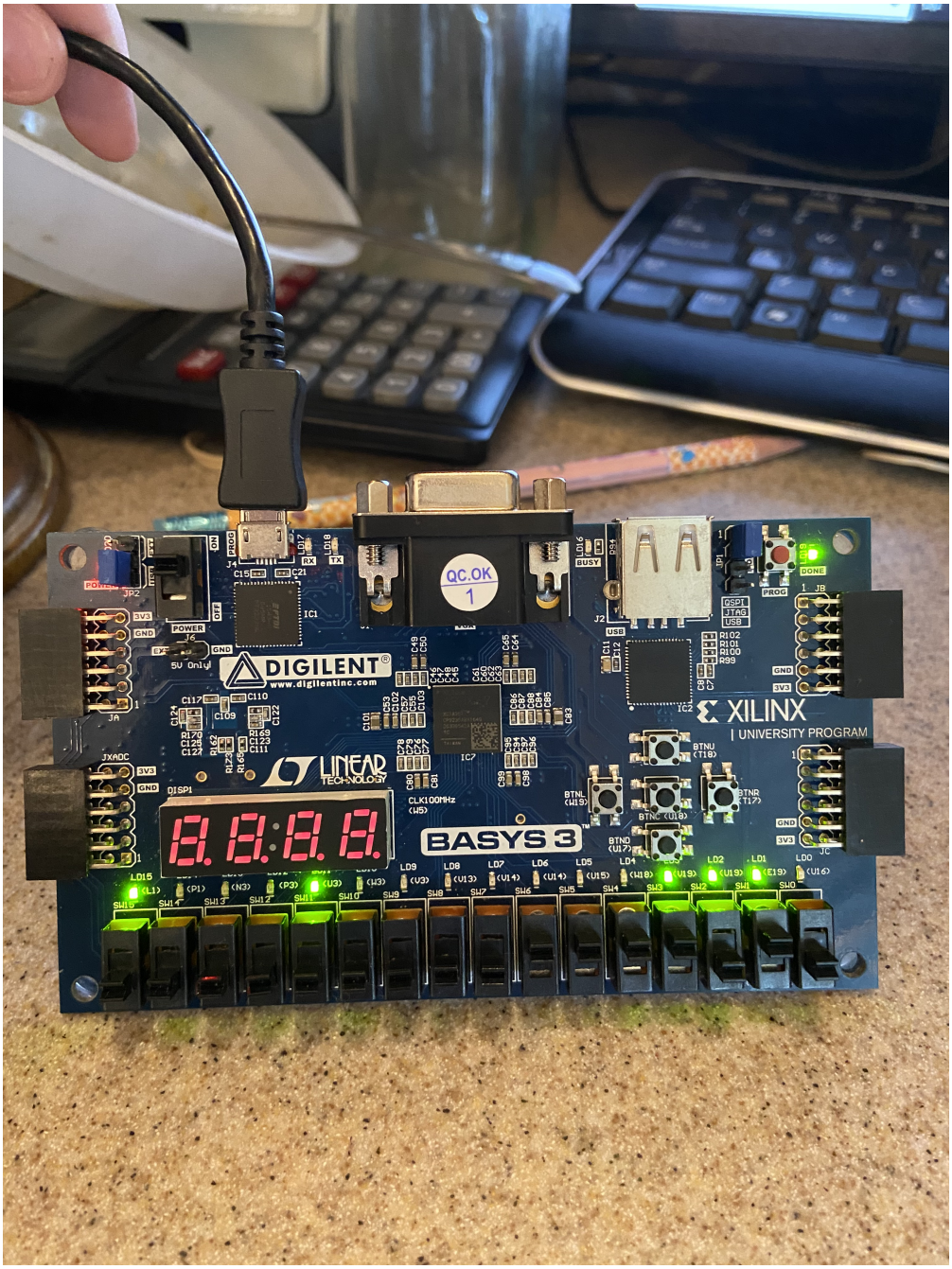

Figure 5: Operation 3-Displaying sum of hexadecimal "14+7A"

Code

Code for source files for Register, ALU, and Top-Lab9 and for test bench files for Register and ALU is included below.

Listing 1: Register Module Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020 -04 - 02

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 <= D;
            else if (en==1)
            Q <= D;
         end
endmodule
```

Listing 2: ALU Module Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020 -04 - 02

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
   );

   //parameters
   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 3: Top-Lab9 Module Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020 -04 - 02

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

wire [7:0] Q1;
wire [7:0] out;
wire [7:0] Q2;
wire [7:0] sw1;
wire [3:0] sw2;

assign sw1 = sw[7:0];
assign sw2 = sw[11:8];


register #(.N(8)) r1(.D(sw1), .clk(clk),
      .en(btnD), .rst(btnC), .Q(Q1));

alu #(.N(8)) (.in0(sw1), .in1(Q1),
      .op(sw2), .out(out));

register #(.N(8)) r2(.D(out), .clk(clk),
      .en(btnU), .rst(btnC), .Q(Q2));

assign led [7:0] = Q1;
assign led [15:8] = Q2;


endmodule
```

Listing 4: Register Test Bench Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020 -04 - 02

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));

   always begin
      clk = ~clk; #5;
   end //clock constantly runs

   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 5: ALU Test Bench Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020 -04 - 02

module alu_test();

reg [7:0] in0;
reg [7:0] in1;
reg [3:0] op;
wire [7:0] out;

alu #(.N(8)) r(.in0(in0), .in1(in1),
      .op(op), .out(out));

   initial begin
      in0=3; in1=2; op=0; #10;
      in0=3; in1=2; op=1; #10;
      in0=3; in1=2; op=2; #10;
      in0=3; in1=2; op=3; #10;
      in0=3; in1=2; op=4; #10;
      in0=3; in1=2; op=7; #10;
      $finish;
   end

endmodule
```