

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

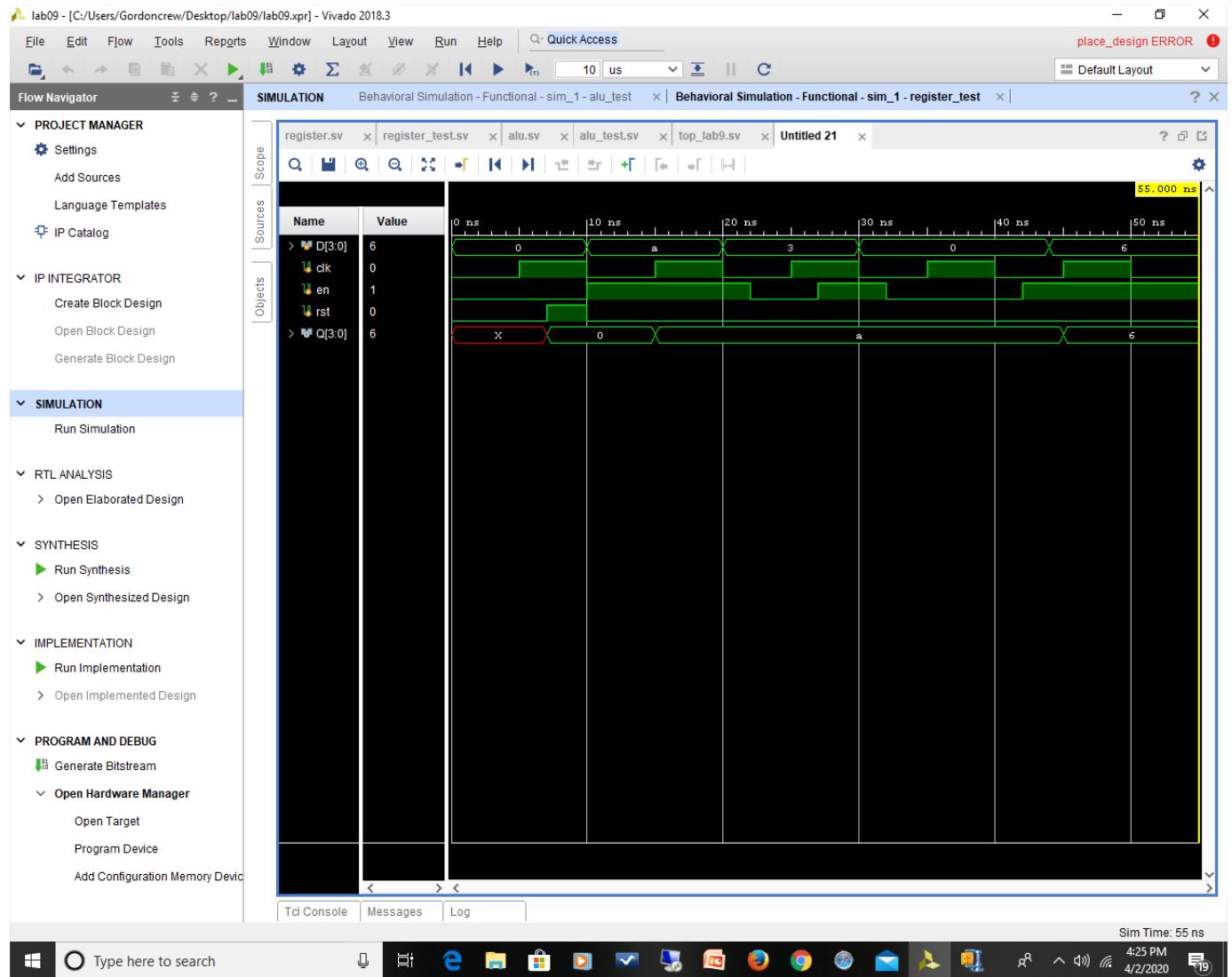


Figure 1: Register ERT and Testbench Results

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

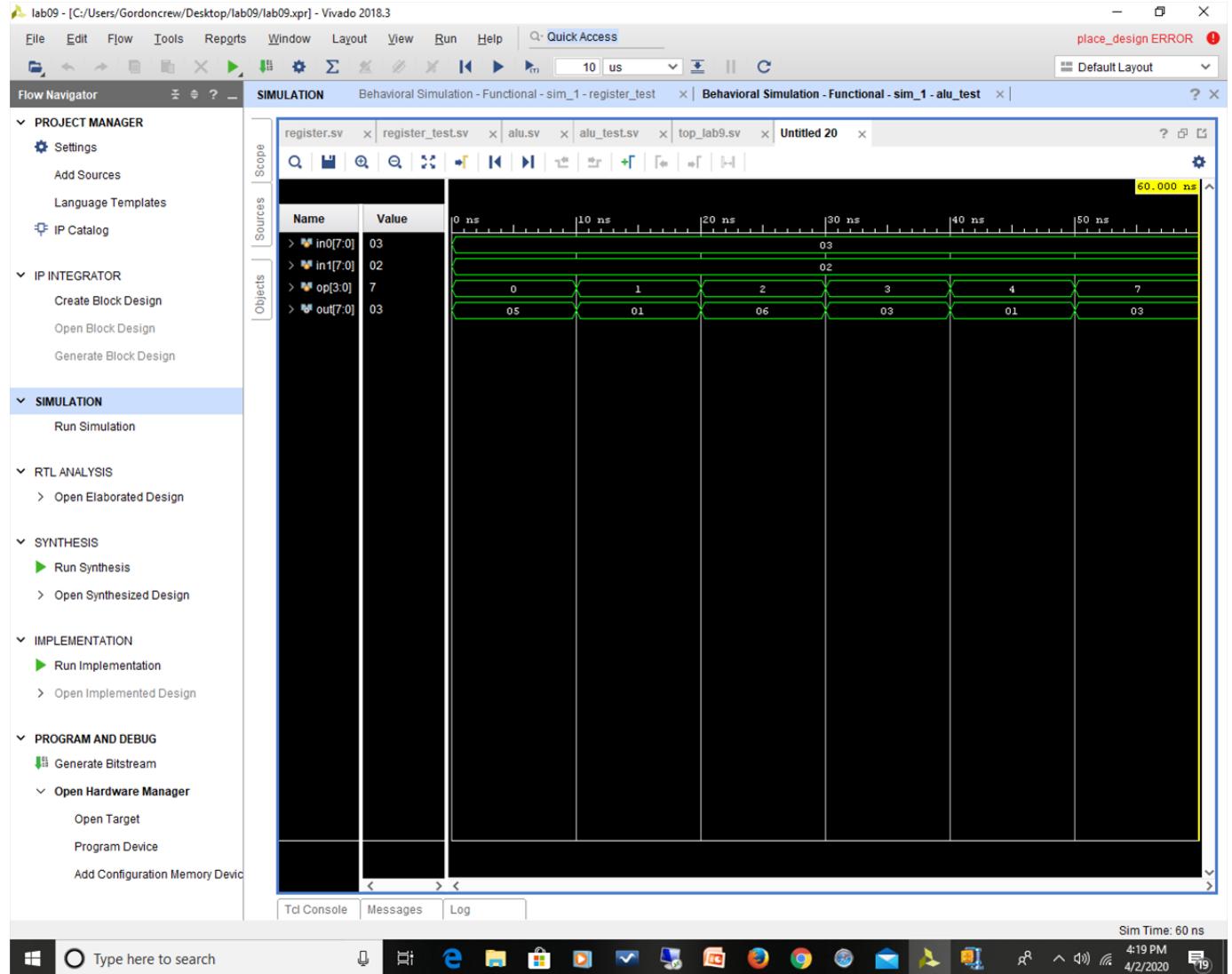


Figure 2: ALU ERT and Testbench Results

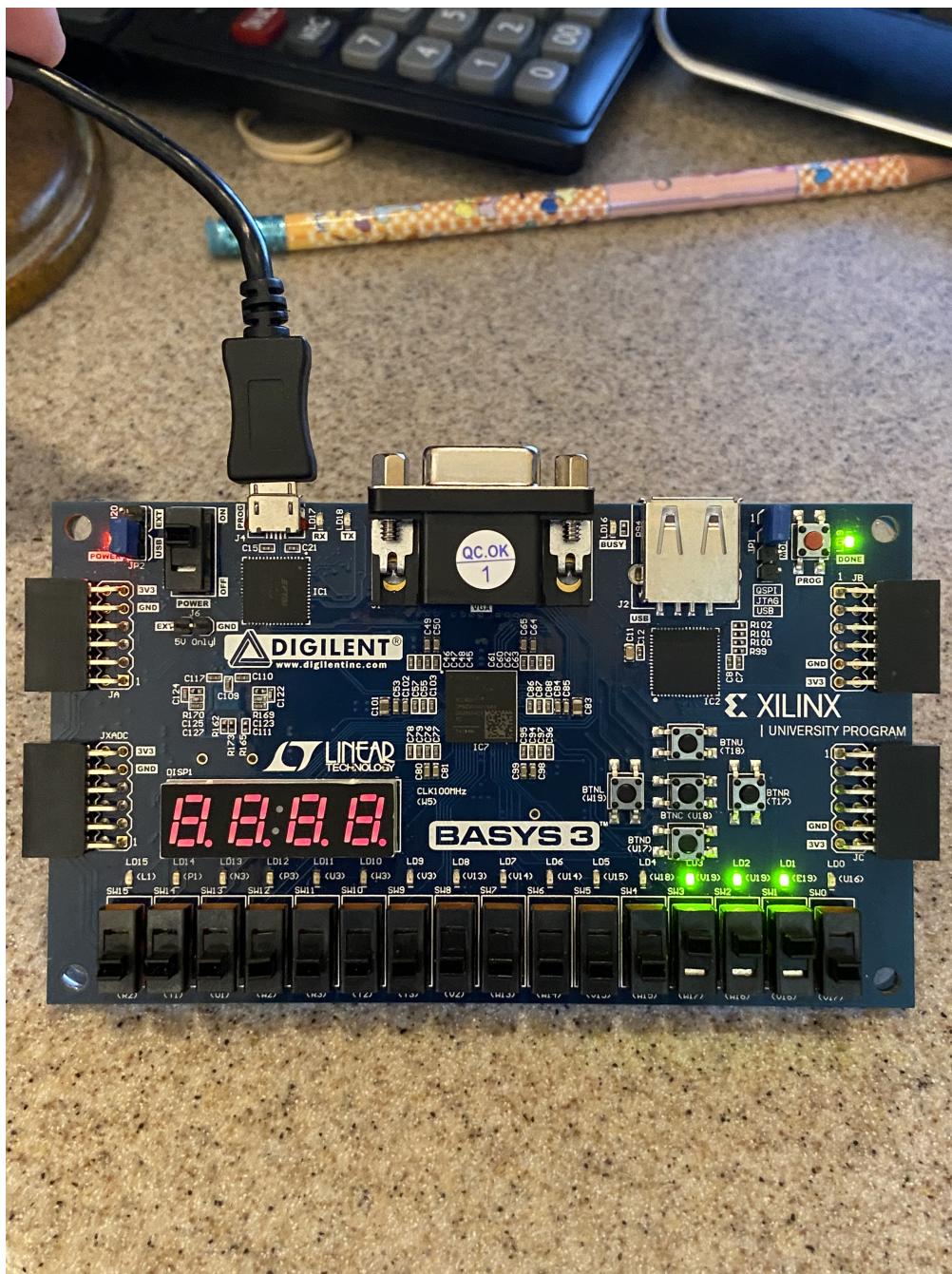


Figure 3: Operation 1-Displaying hexadecimal "14"

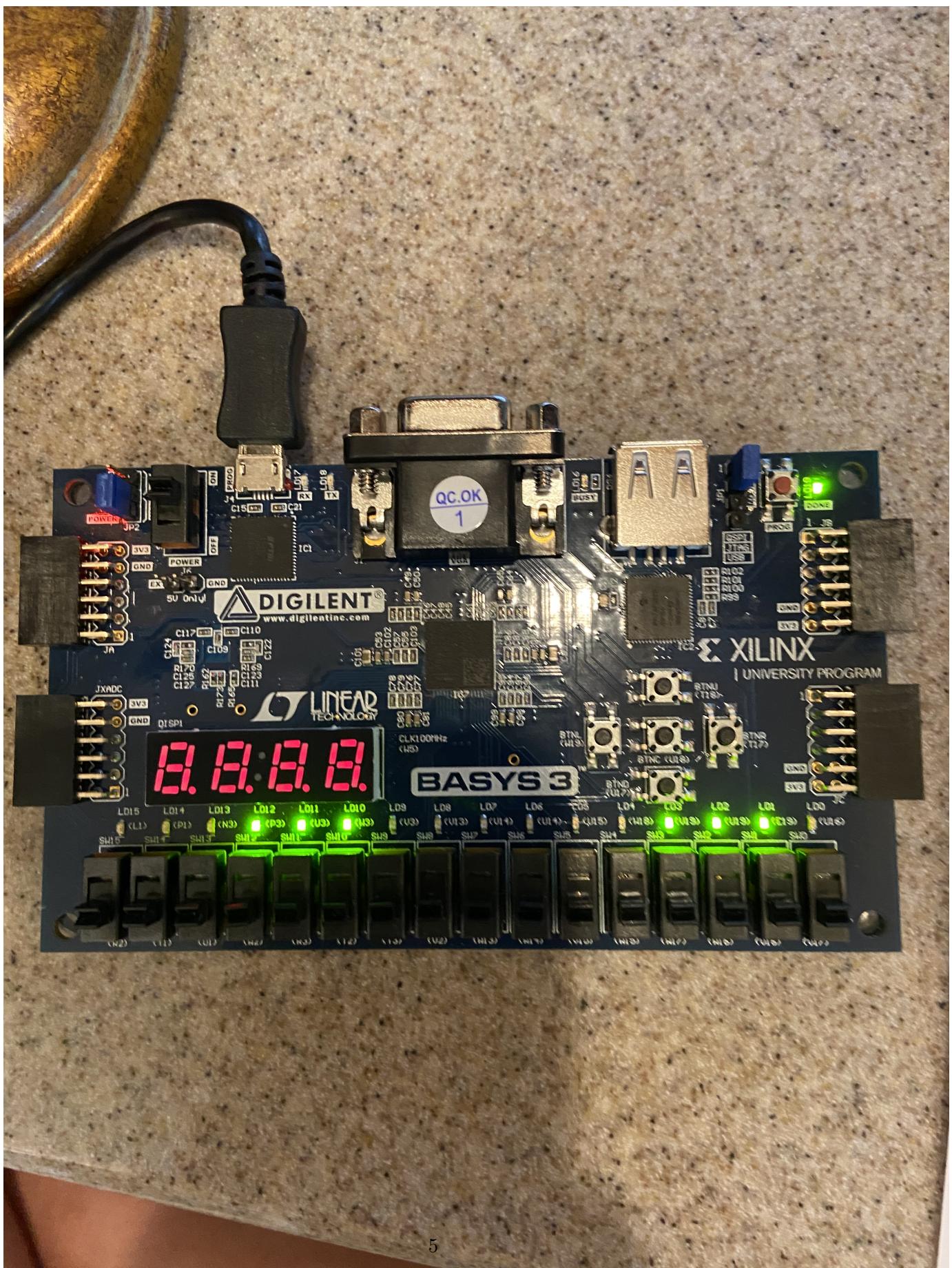


Figure 4: Operation 2-Displaying hexadecimal "14" on LEDs 15-8 and 7-0

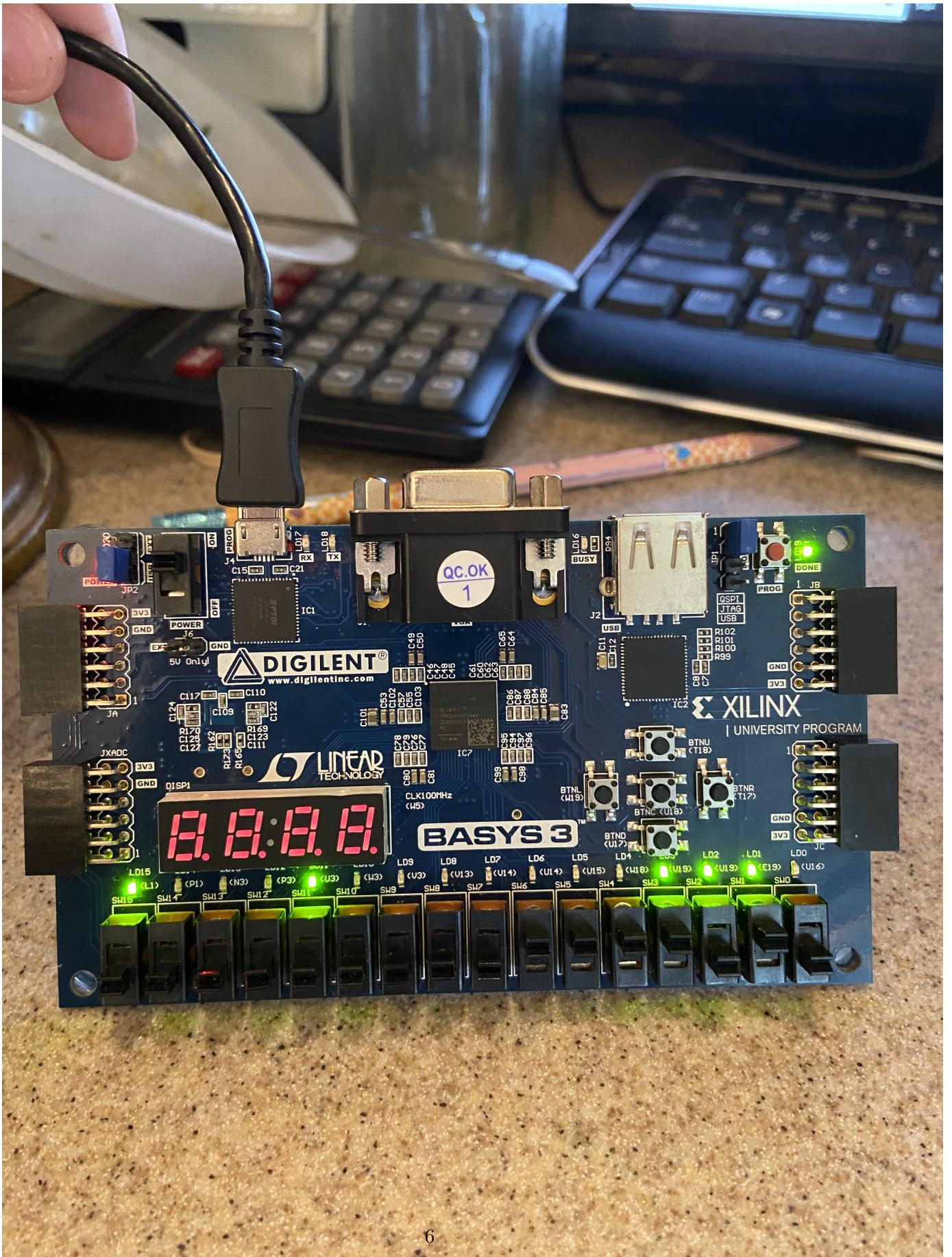


Figure 5: Operation 3-Displaying sum of hexadecimal "14+7A"

Code

Code for mux2, mux4, anddecode, sseg4, sseg4manual is included below.

Listing 1: Mux2 Module Code

```
'timescale 1ns / 1ps
// Ashlie Lackey and Megan Gordon, ELC 2137, 2020 -03 -05
module mux2 #(parameter N=2)(input [N-1:0]in0, [N-1:0]in1,
    input sel,
    output [N-1:0]out);

    assign out = sel?in1:in0;
endmodule
```

Listing 2: Mux4 Module Code

```
'timescale 1ns / 1ps
// Ashlie Lackey and Megan Gordon, ELC 2137, 2020 -03 -05

module mux4 #(parameter N=4)(input [N-1:0] in3,
    input [N-1:0] in2,
    input [N-1:0] in1,
    input [N-1:0] in0,
    input [1:0] sel,
    output reg [N-1:0] out);

    always @(*)
    begin
        case(sel)
            0: out = in0;
            1: out = in1;
            2: out = in2;
            default: out = in3;
        endcase;
    end
endmodule
```

Listing 3: andecoder Module Code

```
'timescale 1ns / 1ps
// Ashlie Lackey and Megan Gordon, ELC 2137, 2020 -03 -05

module an_decode(input [1:0] in,
    output reg [3:0] out);

    always @*
    begin
        case(in)
            0: out = 4'b1110;
            1: out = 4'b1101;
            2: out = 4'b1011;
            default: out = 4'b0111;
        endcase
    end
endmodule
```

Listing 4: sseg4 Module Code

```
'timescale 1ns / 1ps
// Ashlie Lackey and Megan Gordon, ELC 2137, 2020 -03 -05

module sseg4(input [15:0] data,
              input hex_dec,sign,
              input [1:0] digit_sel,
              output reg [7:0] seg,
              output reg dp,
              output reg [3:0] an);

    wire [15:0] bcd11out;
    bcd11 sseg4_bcd11(.B(data[10:0]), .Boutfinal(bcd11out));

    wire [15:0] mux2_1_out;
    mux2 #(N(16)) sseg4_mux2_1(.in0(bcd11out), .in1(data[15:0]), .sel(hex_dec), .out(mux2_1_out));

    wire [3:0] mux4_out;
    mux4 sseg4_mux4(.in0(mux2_1_out[3:0]), .in1(mux2_1_out[7:4]), .in2(mux2_1_out[11:8]), .in3(mux2_1_out[15:12]), .sel(digit_sel), .out(mux4_out));

    wire [6:0] sseg_decoder_out;
    sseg_decoder sseg4_decode(.num(mux4_out), .sseg(sseg_decoder_out));

    wire [3:0] decoder_out;
    an_decode an_decode_sseg4(.in(digit_sel), .out(decoder_out));

    wire mux22_in;
    assign mux22_in = ~decoder_out[3] & sign;
    mux2 #(N(7)) sseg4_mux2_2(.in0(sseg_decoder_out), .in1(7'b0111111), .sel(mux22_in), .out(seg));

    assign dp = 1;
    assign an = decoder_out;

endmodule
```

Listing 5: sseg4 manual Module Code

```
'timescale 1ns / 1ps
// Ashlie Lackey and Megan Gordon, ELC 2137, 2020 -03 -05

module sseg4_manual(input [15:0] sw,
                     output [6:0] seg,
                     output dp,
                     output [3:0] an);

    sseg4 boardconnect(.data({4'b0000, sw[11:0]}), .hex_dec(sw[15]), .sign(sw[14]), .digit_sel(sw[13:12]), .seg(seg), .dp(dp), .an(an));
endmodule
```
