ELC 2137 Lab 10: 7-Segment Display with Time-Division Multiplexing

Megan Gordon

April 16, 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 a 7-segment multiplexer utilizing counters. 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 (counter and a 7-seg driver) 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
clk	0	1	0	1	0	1	0	1	0	1	0
en	0	0	1	0	1	0	1	0	1	0	1
rst	0	1	0	0	0	0	0	0	0	0	0
count	X	0	0	1	1	2	2	3	3	4	4
clkA	X	0	0	1	1	0	0	1	1	0	0
clkB	X	0	0	0	0	1	1	1	1	0	0
clkC	X	0	0	0	0	0	0	0	0	1	1

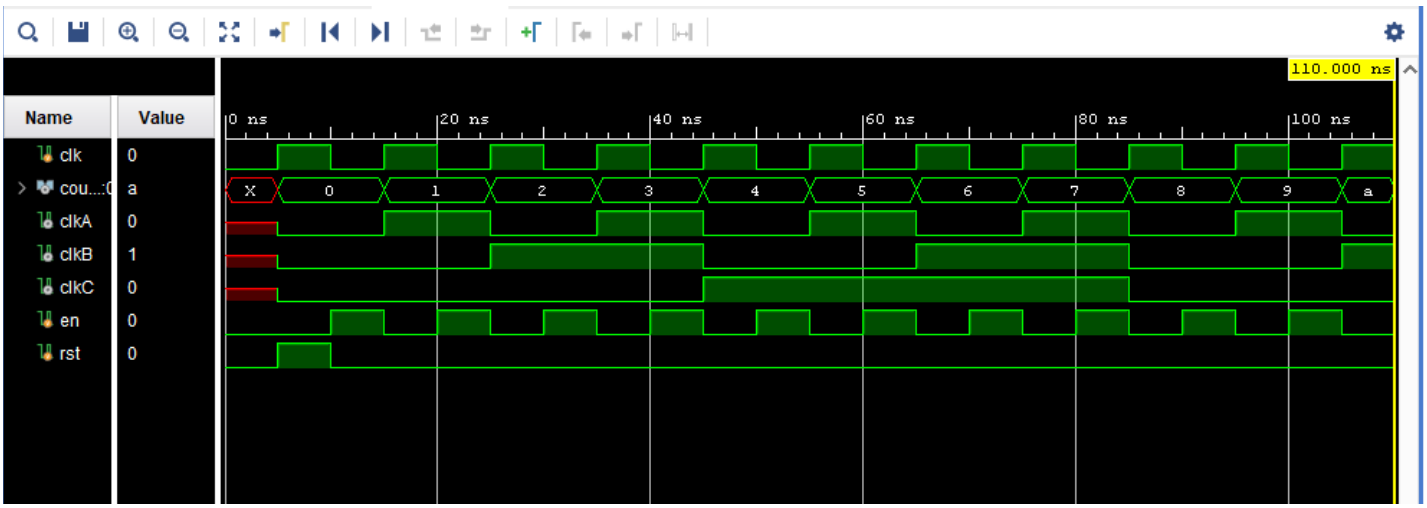

Figure 1: Counter ERT and Testbench Results

Time (ms):	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50	50-55	55-60
hexdec	0	0	0	0	0	1	1	1	1	1	1	1
sign	0	0	0	0	0	0	0	0	0	0	1	1
rst	0	1	0	0	0	0	0	0	0	0	0	1
an[3]	1	1	1	1-0	0-1	1	1	1-0	0-1	1	1	1-0
an[2]	1	1	1-0	0-1	1	1	1-0	0-1	1	1	1-0	0-1
an[1]	1	1-0	0-1	1	1	1-0	0-1	1	1	1-0	0-1	1
an[0]	0	0-1	1	1	1-0	0-1	1	1	1-0	0-1	1	1
seg (hex)	30	78	02	79	30	12	02	79	79	12	02	3E

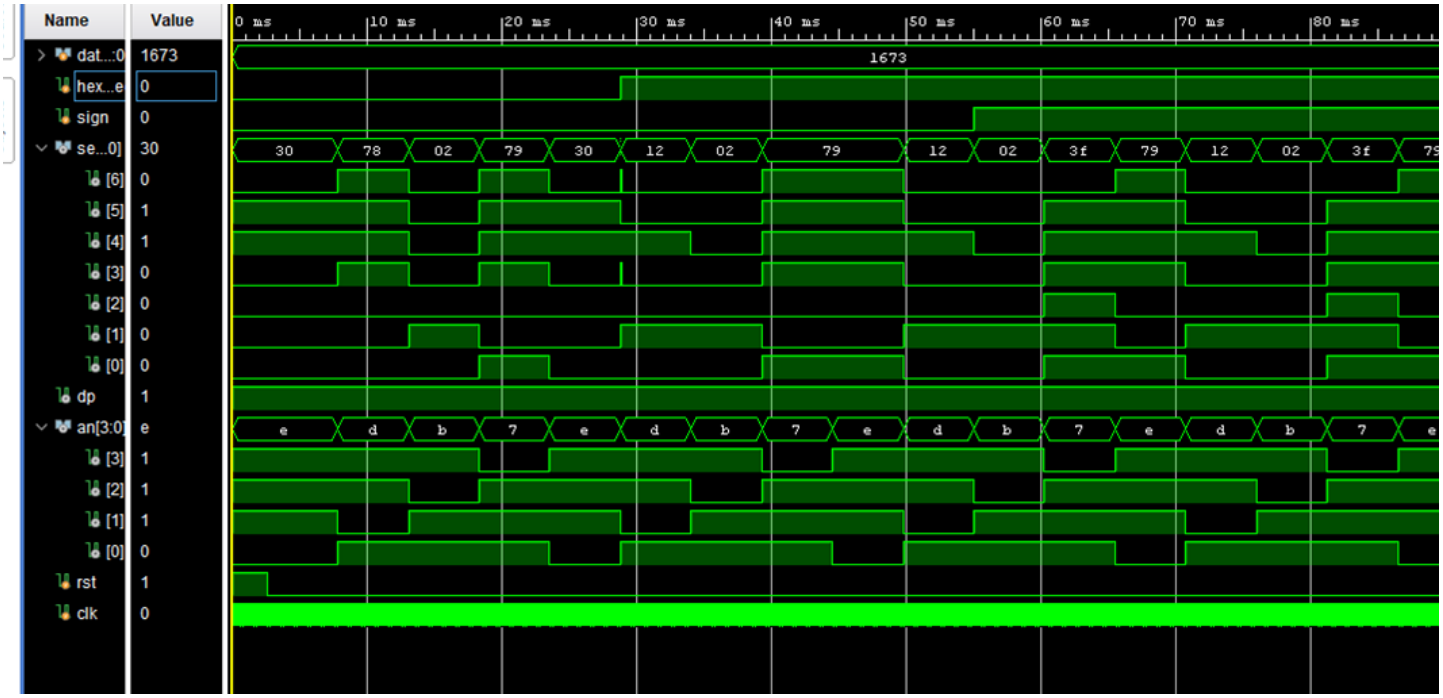

Figure 2: 7-Seg Driver ERT when data is 1673 and Testbench Results

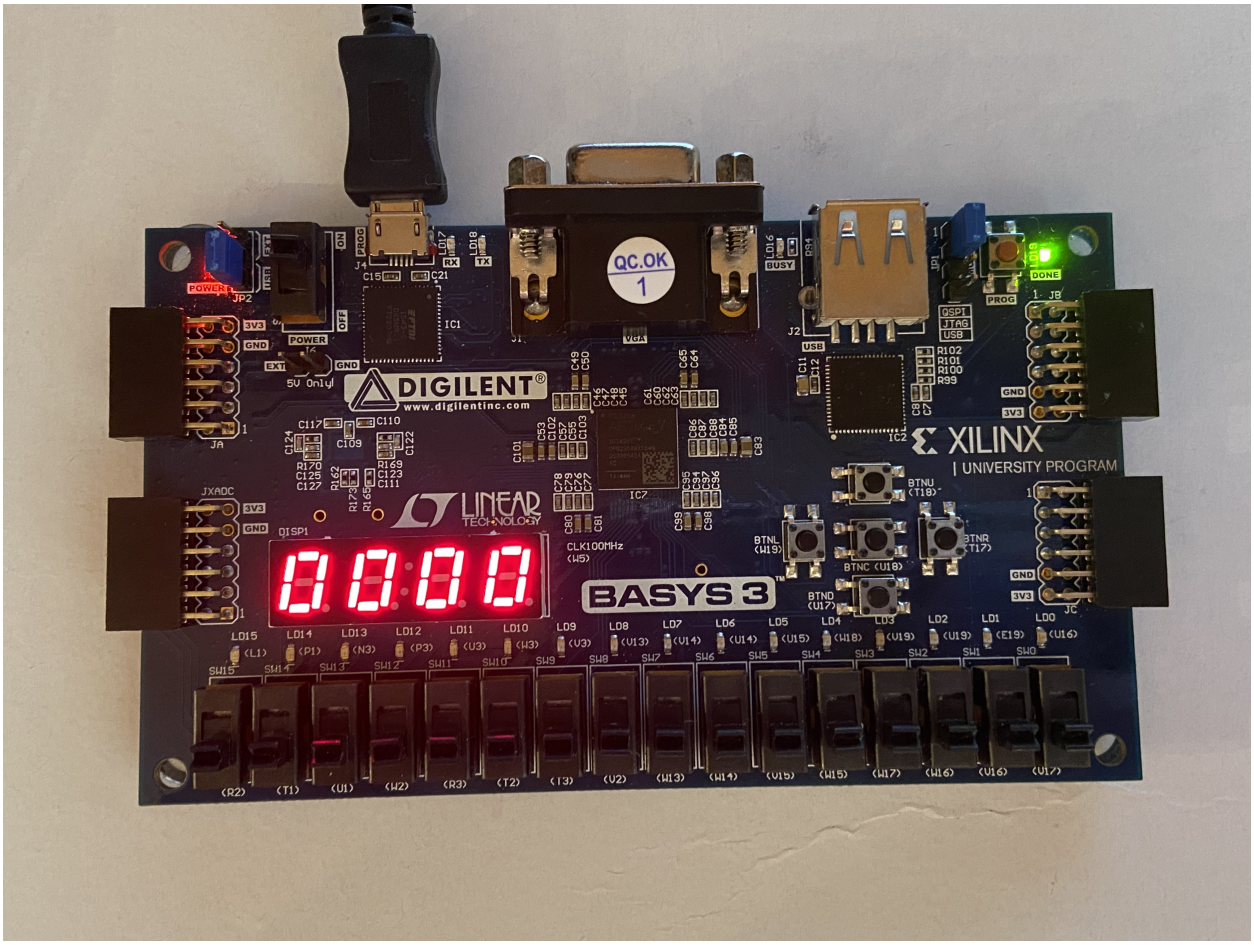

Figure 3: Testing Step 1

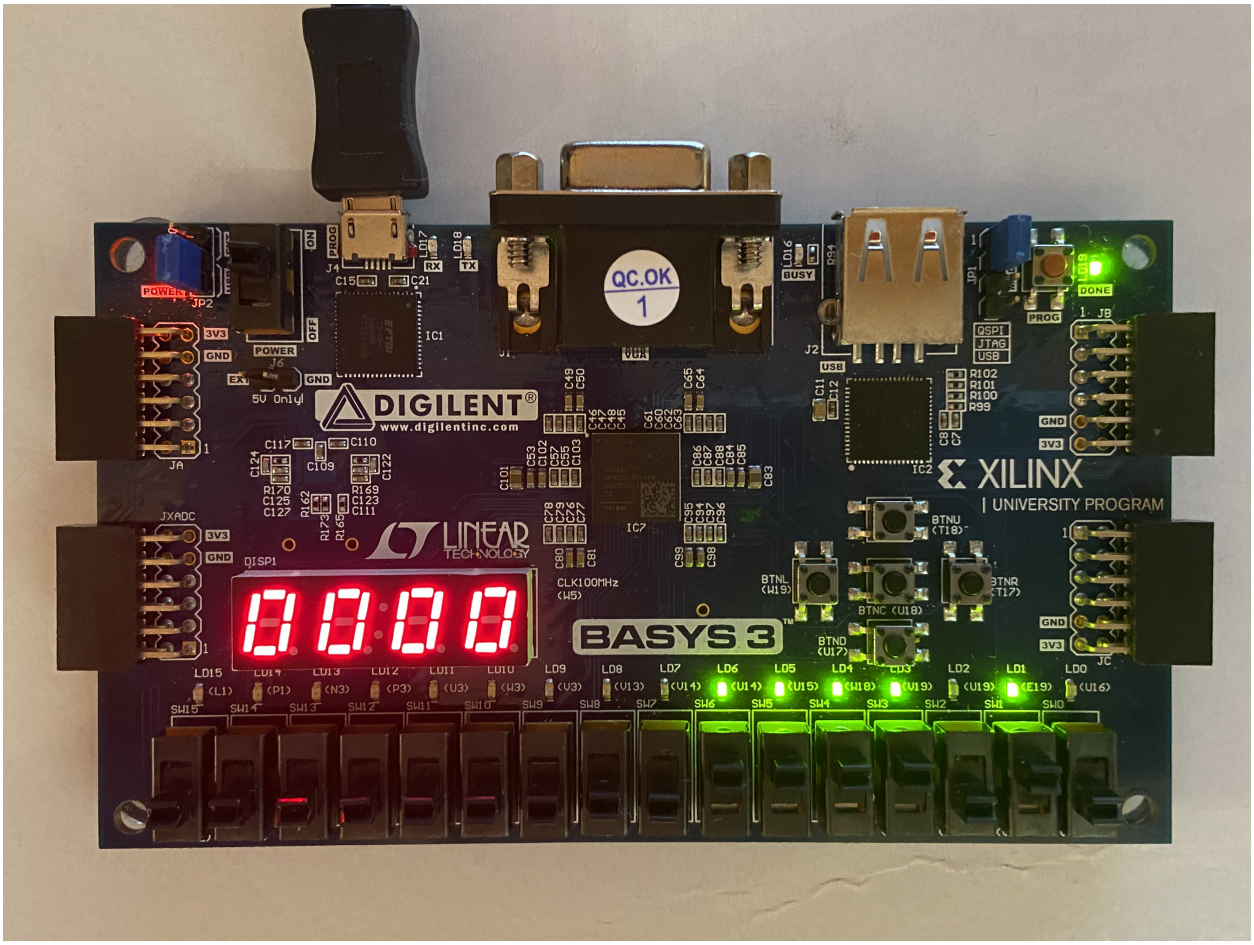

Figure 4: Testing Step 2

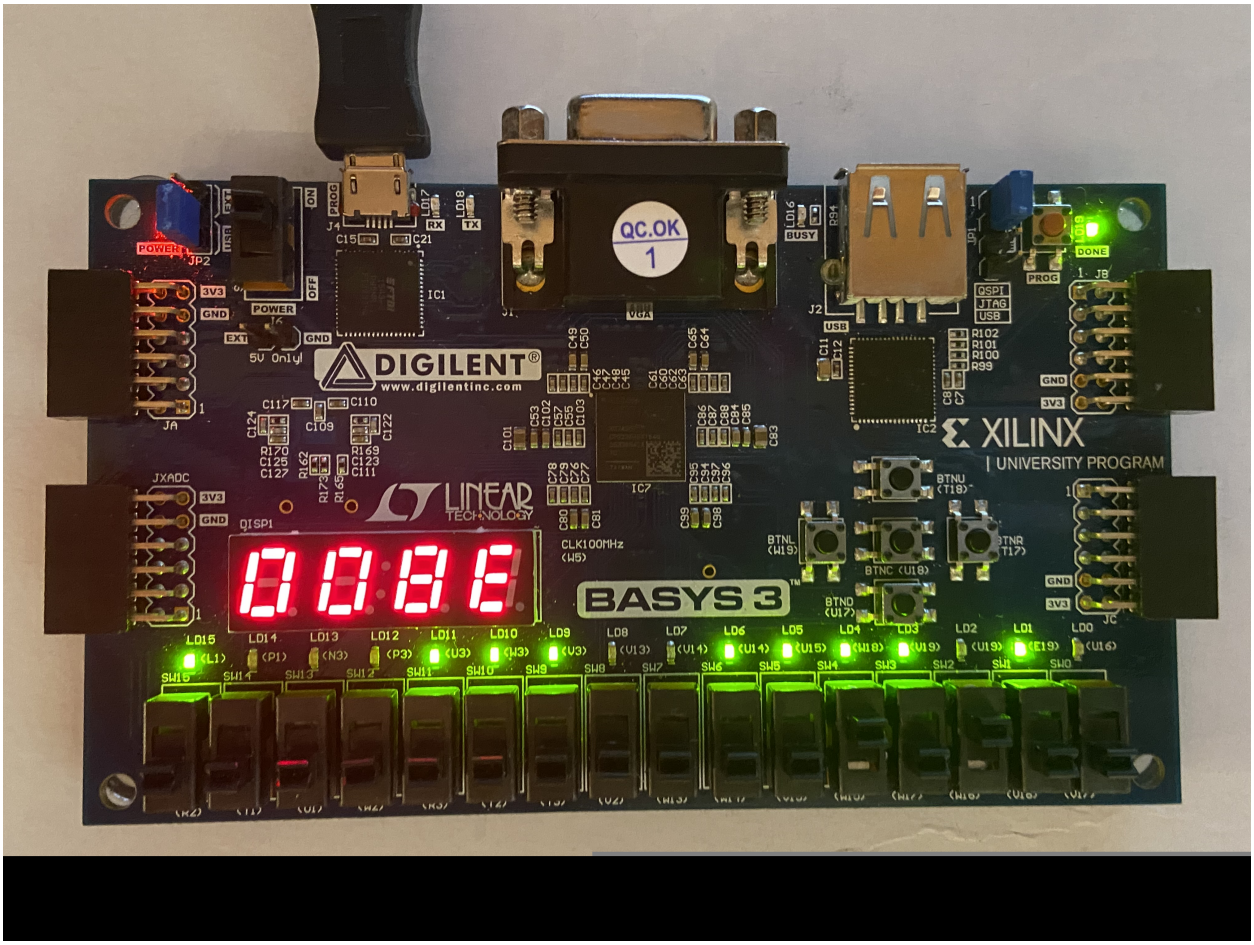

Figure 5: Testing Step 3

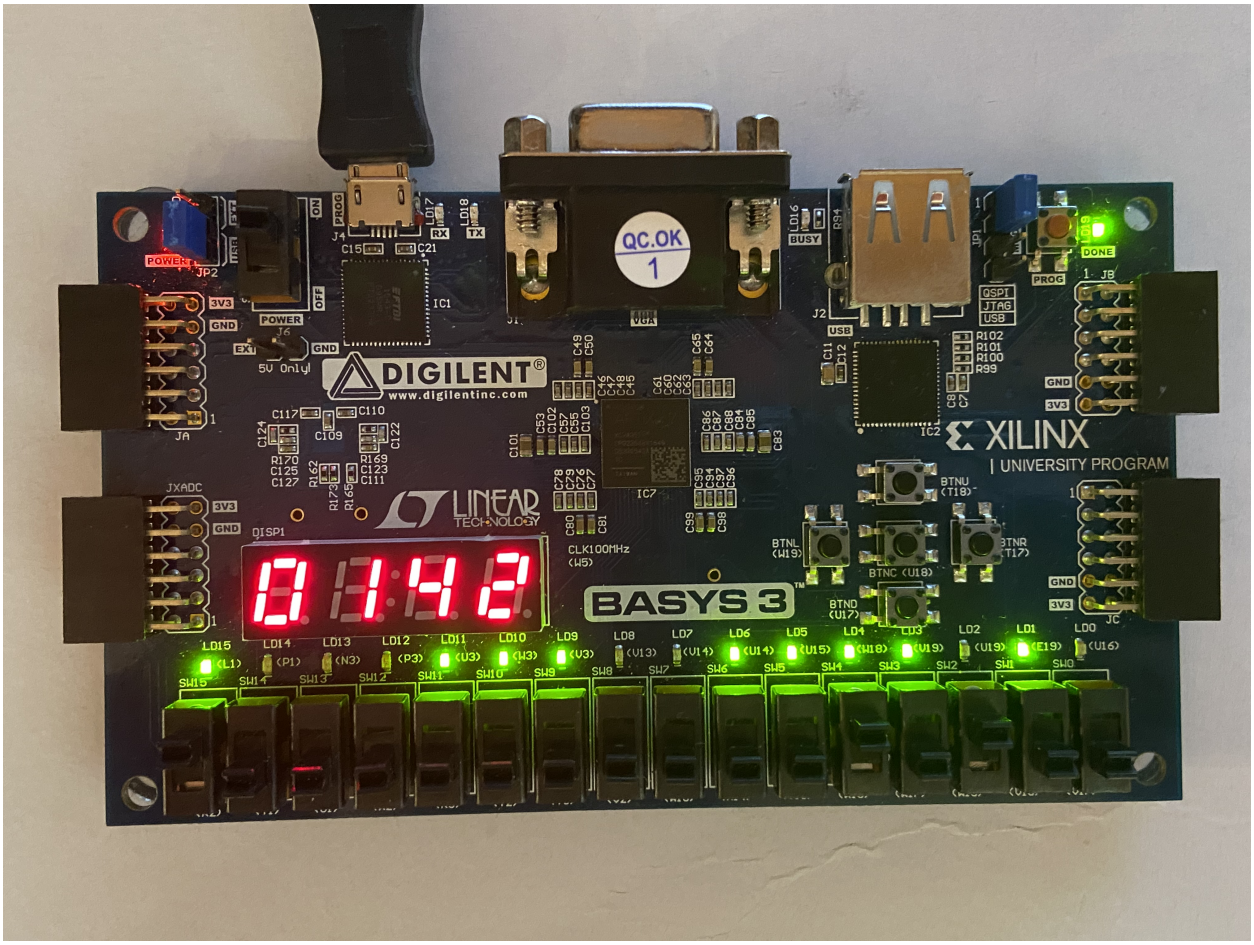

Figure 6: Testing Step 4

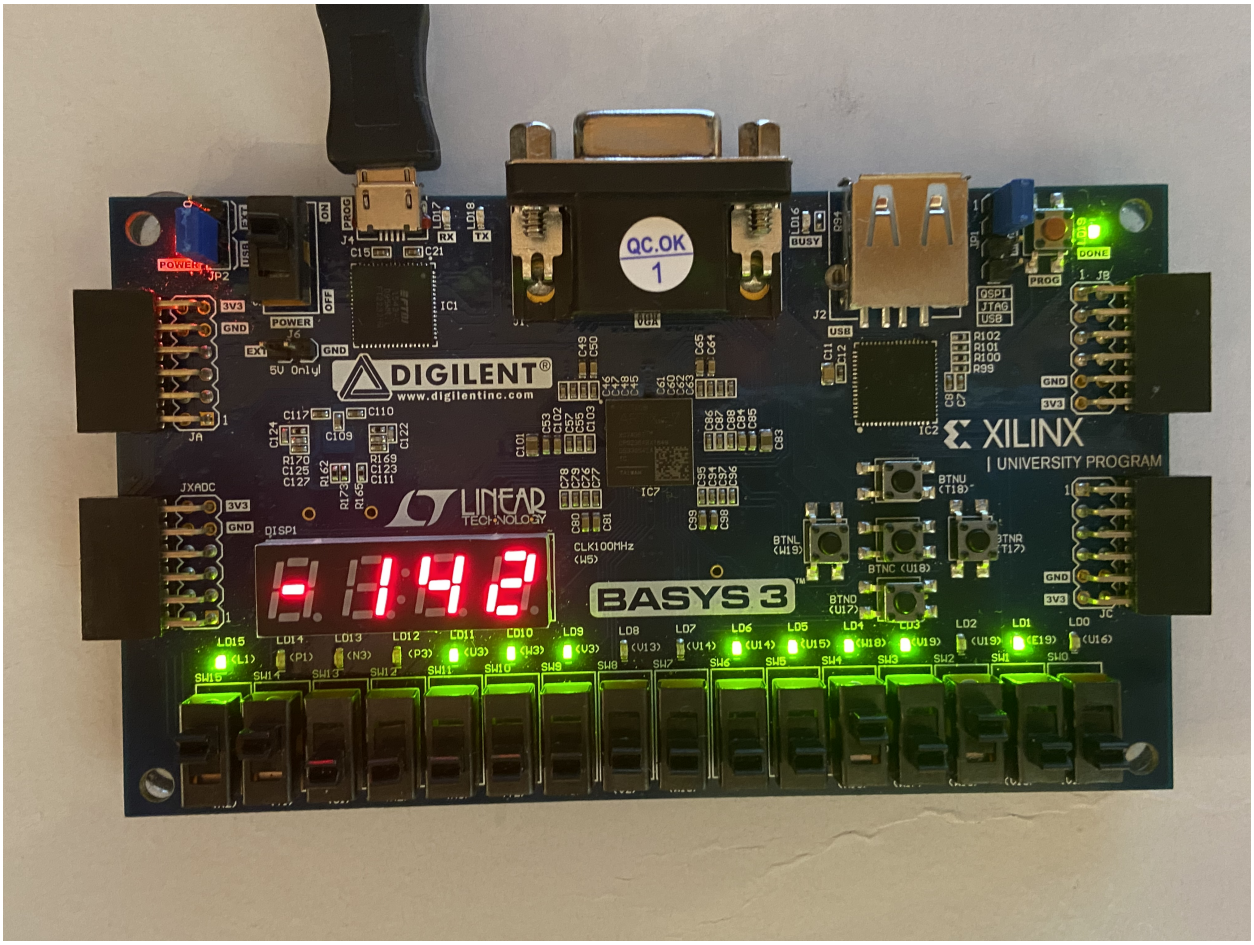

Figure 7: Testing Step 5

Code

Code for source files for Counter, 7-Seg Driver, and Top-Level Calculator and the test bench files for the Counter and Driver.

Listing 1: Counter Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020-04-09

module counter #(parameter N=1)
(input clk, rst, en,
output [N-1:0] count,
output tick
);

//internal signals
reg [N-1:0] Q_reg, Q_next;

//register (state memory)
always @(posedge clk, posedge rst)
begin
if (rst)
Q_reg <= 0;
else
Q_reg <= Q_next;
end

//next-state logic
always @*
begin
if (en)
Q_next = Q_reg + 1;
else
Q_next = Q_reg; //no change
end

//output logic
assign count = Q_reg;
assign tick = (Q_reg=={N{1'b1}}) ? 1'b1 : 1'b0;

endmodule //counter
```

Listing 2: 7-Seg Driver Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020-04-09

module sseg4_TDM(input [15:0] data,
input hex_dec,sign,
output reg [6:0] seg,
output reg dp,
output reg [3:0] an,
input rst, clk);

wire [1:0] digit_sel;
wire tick1;

counter #(.N(18))timer(.clk(clk), .rst(rst), .en(1'd1), .tick(tick1));


counter #(.N(2))counter2(.clk(tick1), .rst(rst), .en(1'd1), .count(
    digit_sel));

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(data[15:0]), .in1(bcd11out), .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 3: Top-Level Calculator Code

```
‘timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020-04-09

module calc_lab10(
input btnU,
input btnD,
input [15:0]sw,
input clk,
input btnC,
output [15:0]led,
output [6:0] seg,
output dp,
output [3:0] an);

sseg4_TDM disp_unit(.data({8'b00000000, led[15:8]}),
.hex_dec(sw[15]), .sign(sw[14]), .clk(clk), .rst(btnC),
.seg(seg), .dp(dp), .an(an));

top_lab9 calc_unit(.btnC(btnC), .btnD(btnD), .btnU(btnU), .clk(clk),
.sw(sw), .led(led));

endmodule
```

Listing 4: Counter Test Bench Code

```
‘timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020-04-09

module counter_test();

reg clk;
wire [3:0] count;
wire clkA;
wire clkB;
wire clkC;
reg en, rst;

assign clkA = count[0];
assign clkB = count[1];
assign clkC = count[2];

counter #(.N(4)) r(.clk(clk),
.en(en), .rst(rst), .count(count));

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

initial begin
clk=0; rst=0; en=0; #5;
rst=1; #5; //reset
rst=0; en=1; #5;
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5; en=1; #5
en=0; #5;
$finish;
end

endmodule
```

Listing 5: 7-Seg Driver Test Bench Code

```
`timescale 1ns / 1ps
//Megan Gordon, ELC 2137, 2020-04-09

module sseg4_TDM_test();


reg [15:0] data;
reg hex_dec,sign;
wire [6:0] seg;
wire dp;
wire [3:0] an;
reg rst, clk;


sseg4_TDM testTDM(.clk(clk), .rst(rst), .data(data), .seg(seg), .dp(dp), .
    an(an),
.hex_dec(hex_dec), .sign(sign));

always begin
clk = ~clk; #10;
end //clock constantly runs

initial begin
clk=0; rst=1;
hex_dec=0; sign=0;
data=16'b0001011001110011;
#2621440;
rst=0; #26214400;
hex_dec=1; #26214400;
sign=1;
end

endmodule
```