ELC 2137 Lab 9: ALU

Spencer Stinson

October 28, 2020

Summary

In this lab, we created an arithmetic logic unit capable of 4 operations, and with room to program more operations into the program. By using our program, we are able to save a value so that we can perform these operations with just a few pushes of a button. We were able to do this by use of sequential and combinational logic and by using new latches and flip flops that we have been taught.

Results

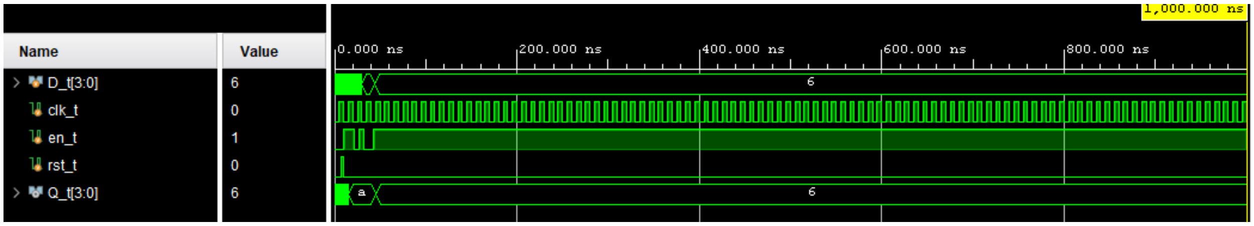

Figure 1: register simulation waveform

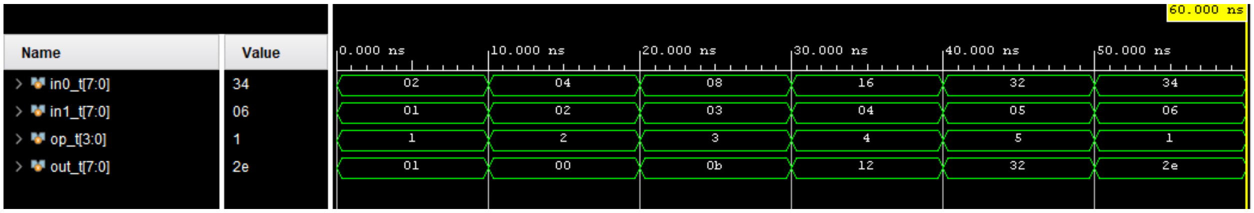

Figure 2: alu simulation waveform

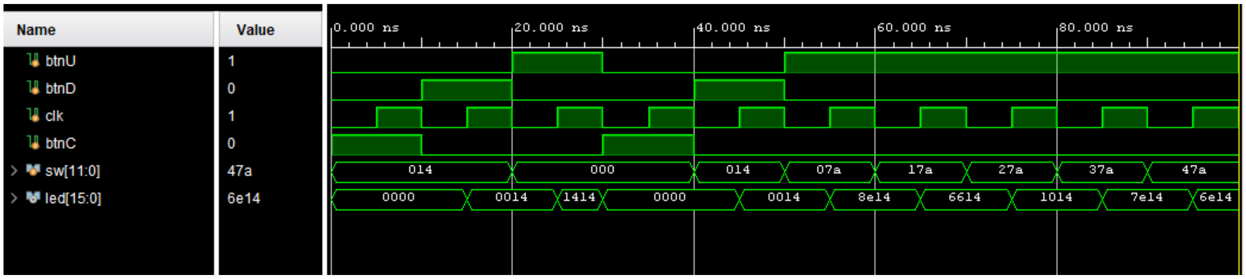

Figure 3: top level simulation waveform

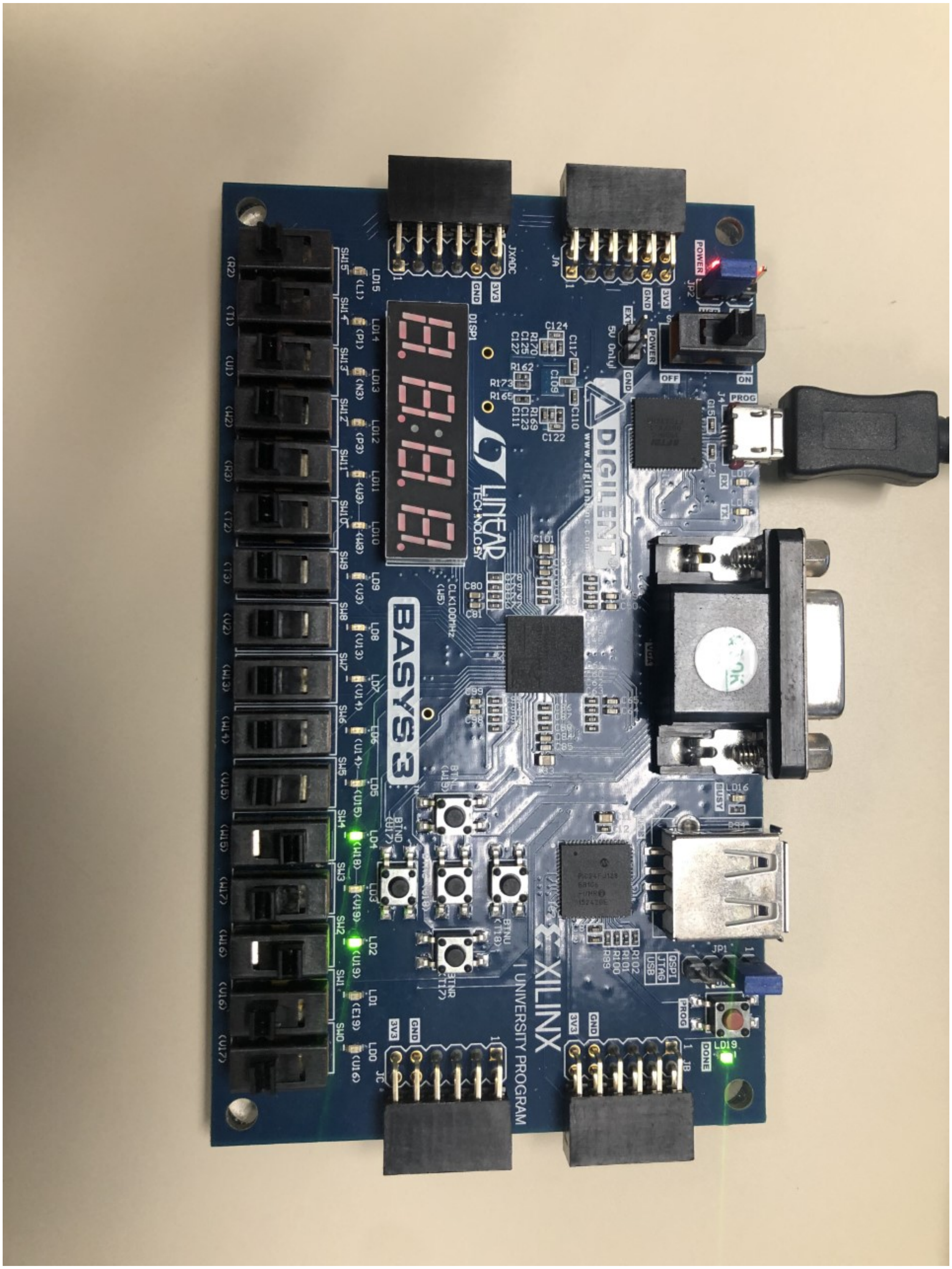

Figure 4: 14 displayed in hexidecimal

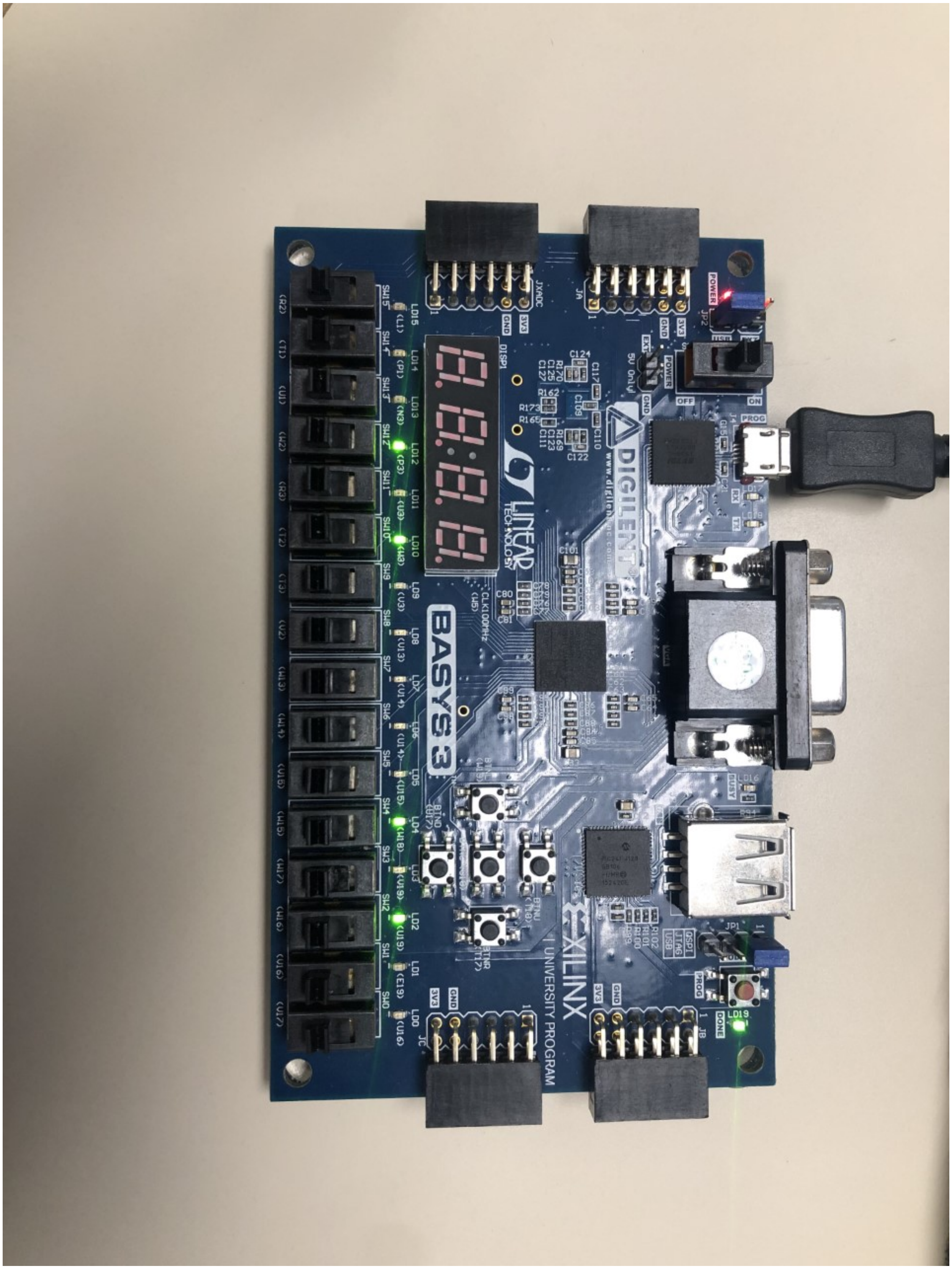

Figure 5: 1414 displayed in leds

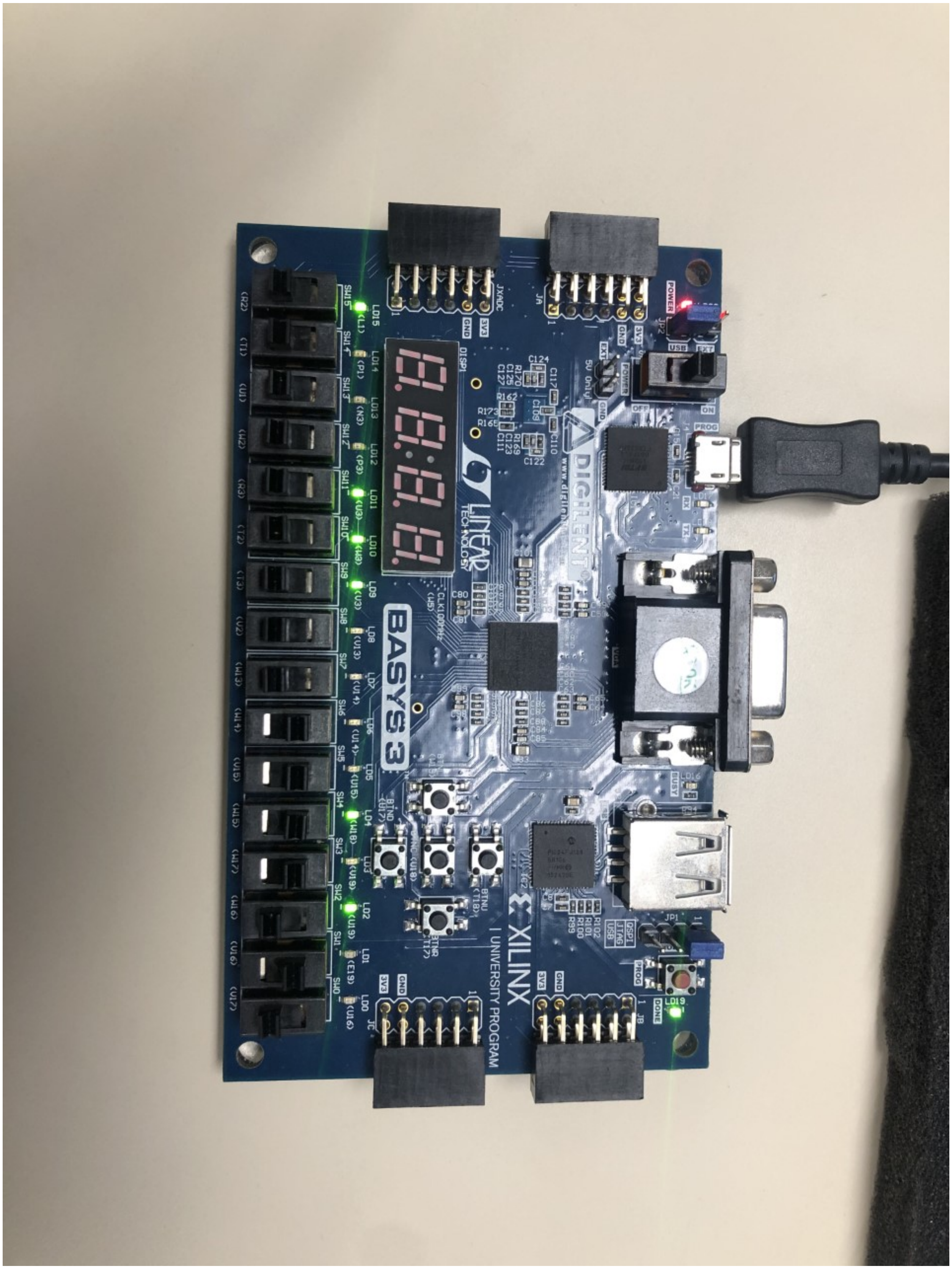

Figure 6: addition displayed on leds

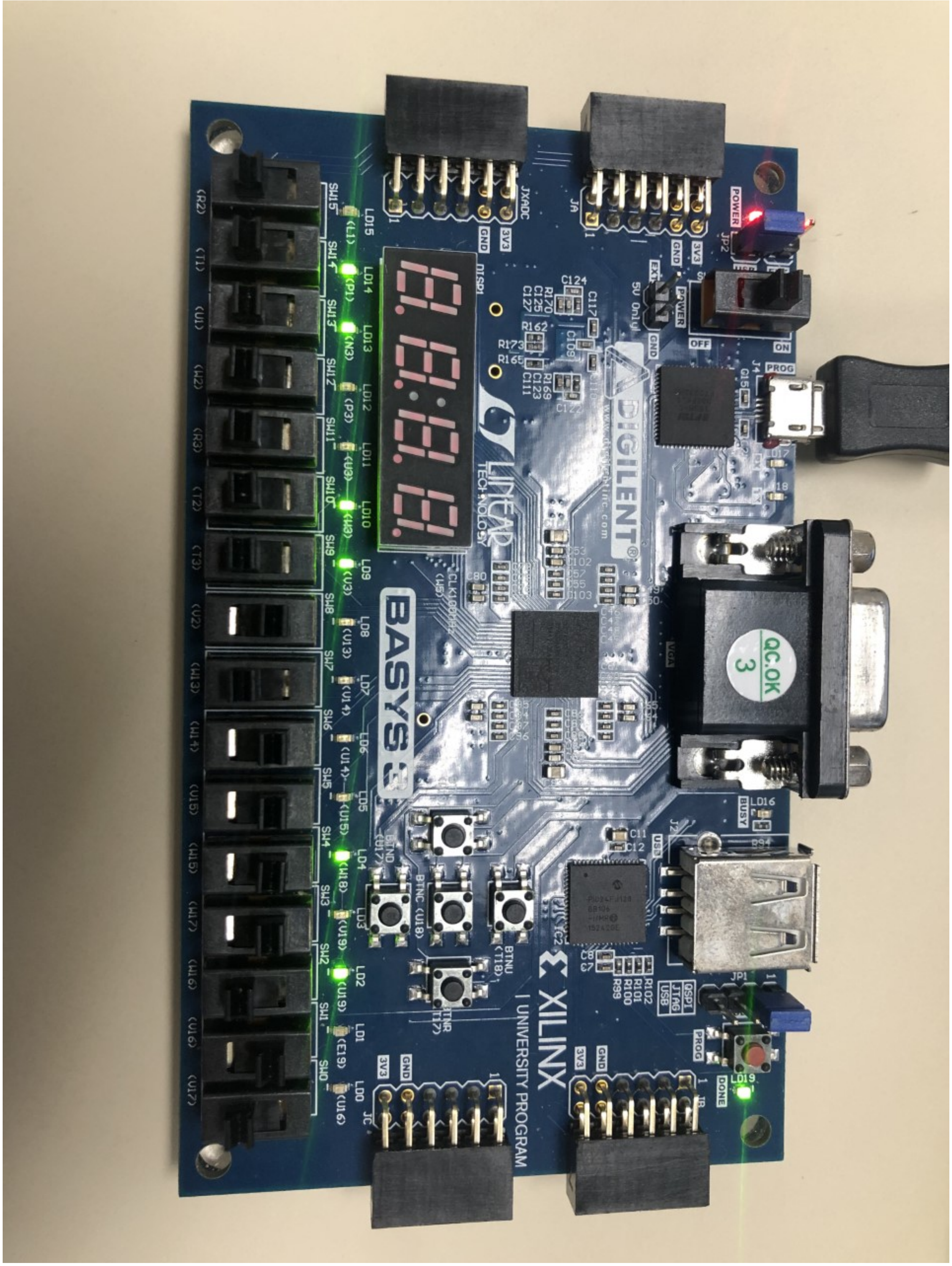

Figure 7: register simulation waveform

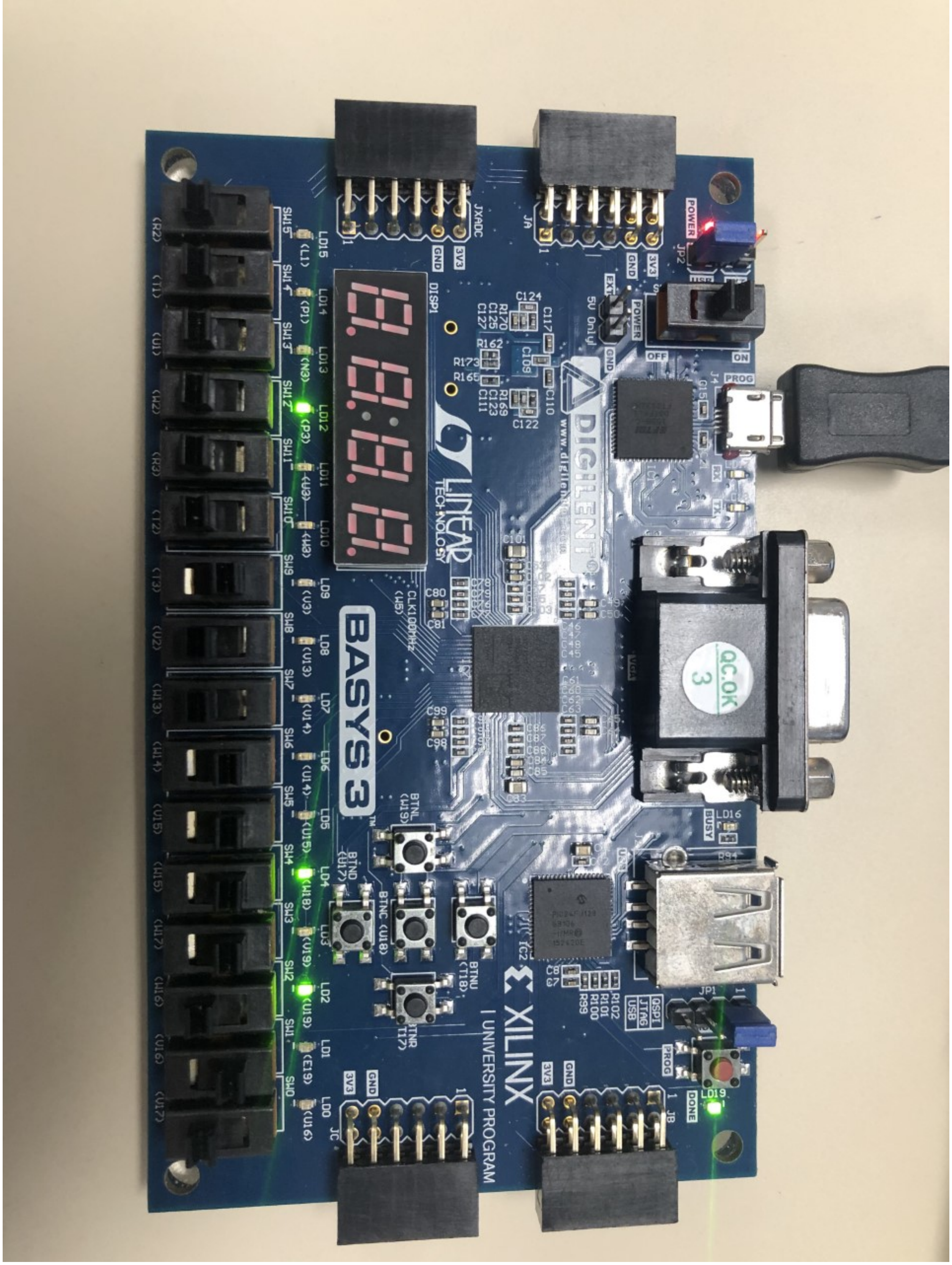

Figure 8: register simulation waveform

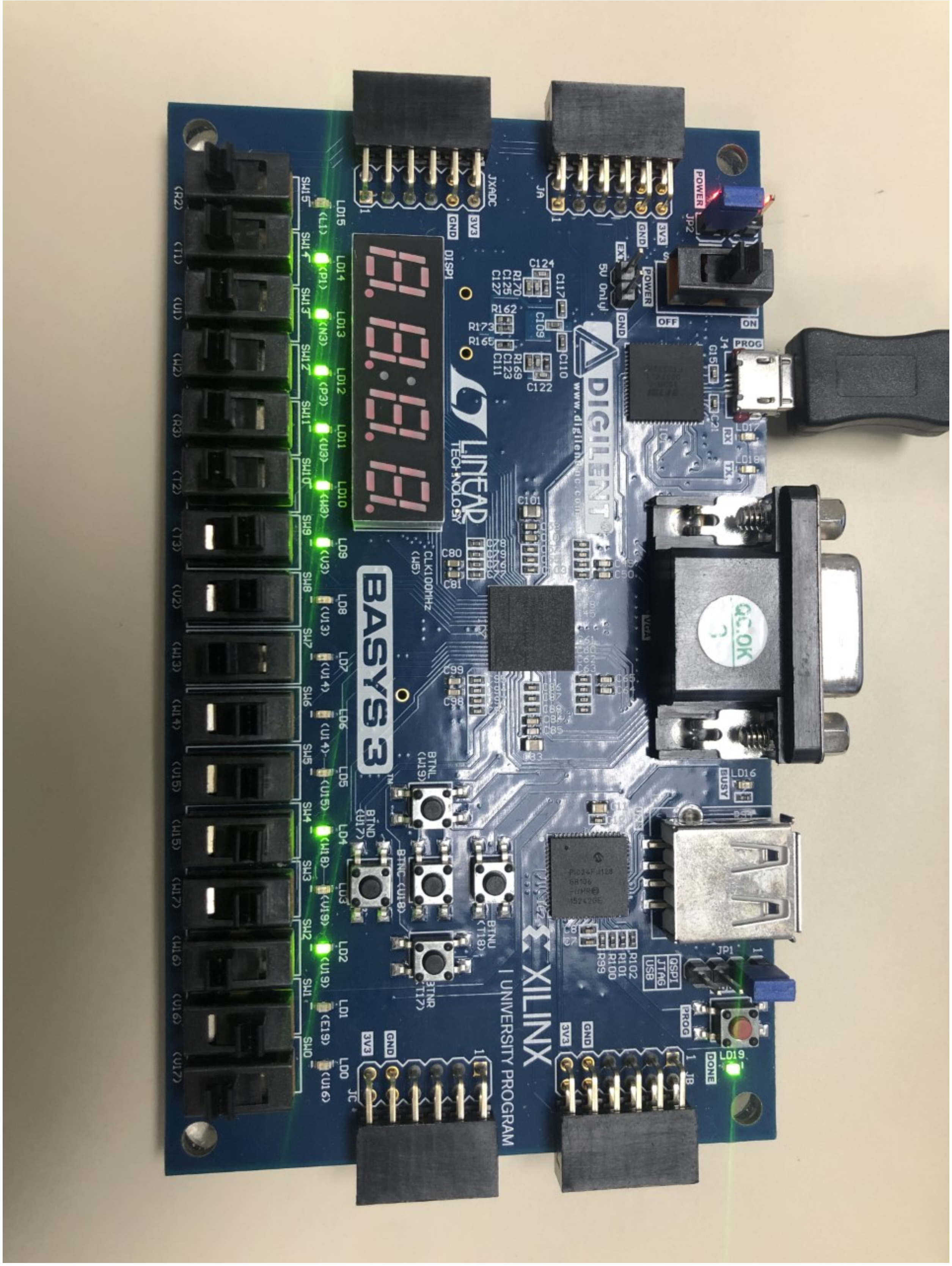

Figure 9: register simulation waveform

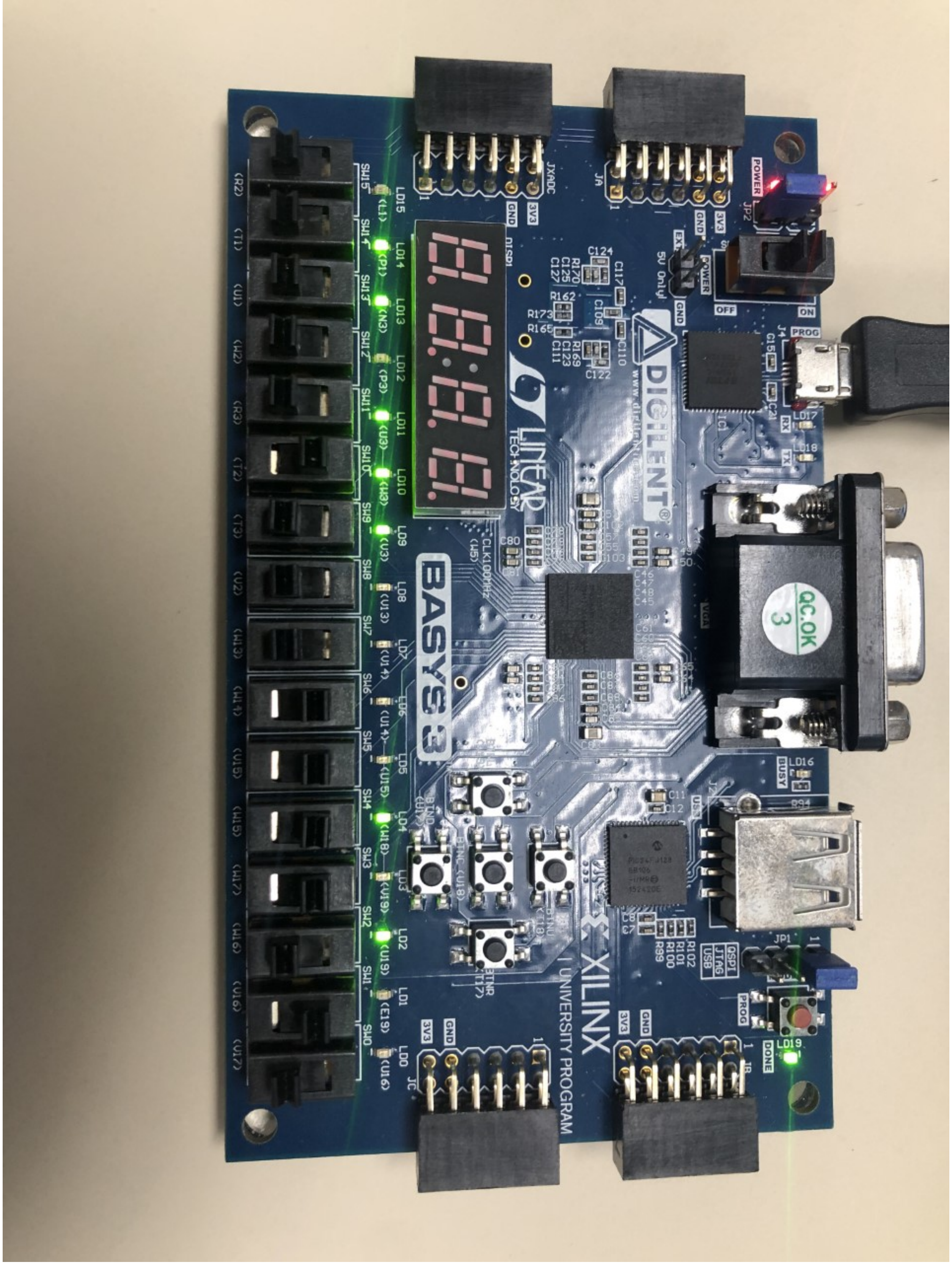

Figure 10: register simulation waveform

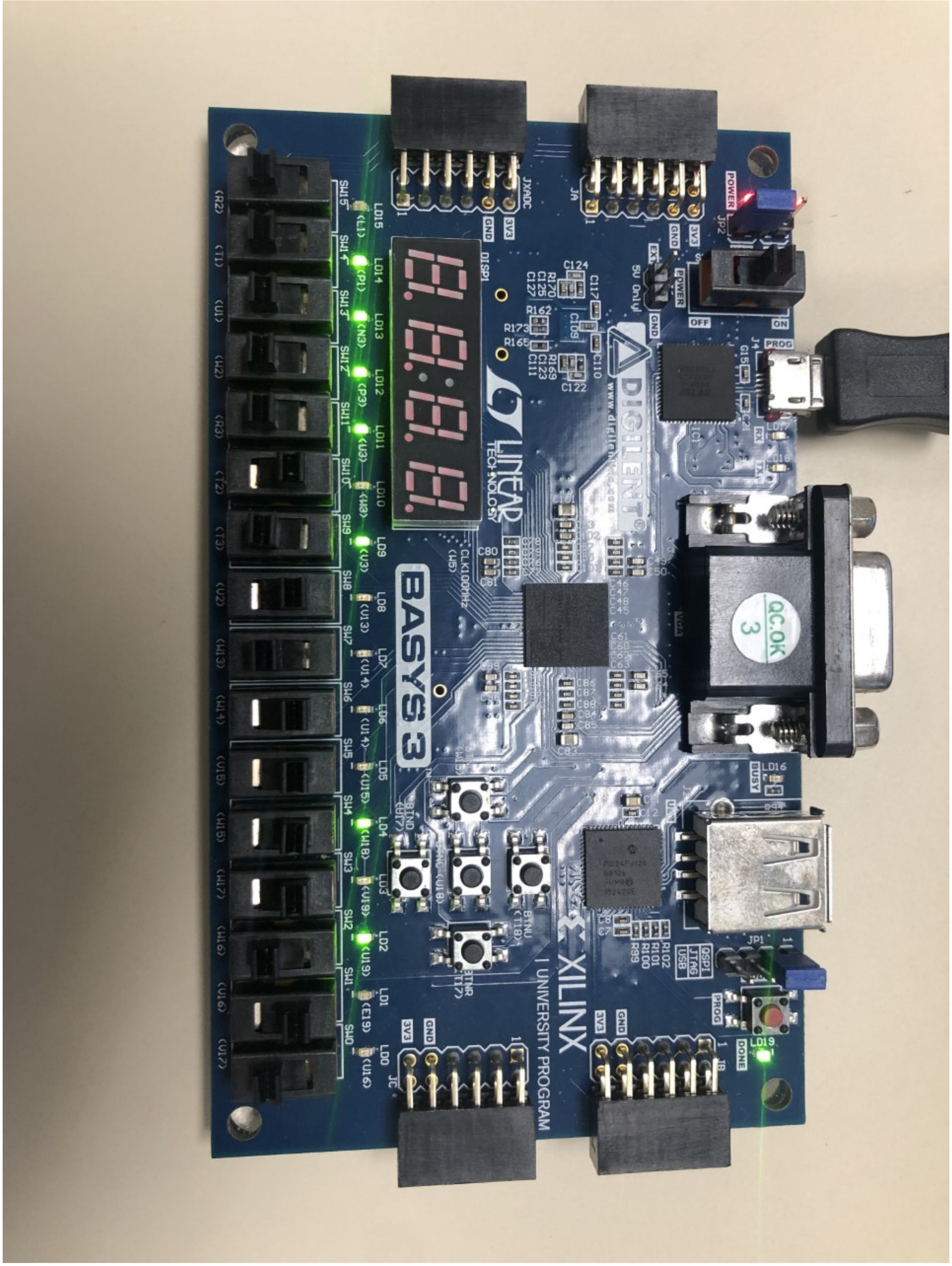

Figure 11: register simulation waveform

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	A	A	A	A	A	A	6	6	6

Table 2: *alu* expected results table skeleton

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0 0	2	4	8	16	32	34
in1 0	1	2	3	4	5	6
op 0	1	2	3	4	5	1
out 0	1	0	11/b	12	32	2e

Code

Listing 1: register code

```
`timescale 1ns / 1ps
//
   ////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/22/2020 11:19:22 AM
// Design Name:
// Module Name: register
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////
```

```
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)
        if (rst==1)
            Q <= 0;
       else if (en==1)
            Q <= D;
endmodule
```

Listing 2: register testbench code

```
‘timescale 1ns / 1ps
//
   ////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/22/2020 11:23:18 AM
// Design Name:
// Module Name: register_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////



module register_testbench();

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

register #(.N(4)) dut(
    .clk(clk_t),
    .rst(rst_t),
    .en(en_t),
    .D(D_t),
    .Q(Q_t)
);

always begin
    clk_t = ~clk_t; #5;
    end

 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’b0; #2;
    en_t = 0;   #10;
    en_t = 1;   #2;
    D_t = 4’h6; #11;
    end

endmodule
```

Listing 3: alu code

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 11:44:04 AM
// Design Name:
// Module Name: alu
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


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 code

```
`timescale 1ns / 1ps
//
   ////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/22/2020 11:23:18 AM
// Design Name:
// Module Name: register_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////



module alu_testbench();
    reg [7:0] in0_t;
    reg [7:0] in1_t;
    reg [3:0] op_t;
    reg [7:0] out_t;


alu #(.N(8)) dut(
    .in0(in0_t),
    .in1(in1_t),
    .op(op_t),
    .out(out_t)
);
initial begin
    in0_t =  2; in1_t = 1; op_t = 1; #10;
    in0_t =  4; in1_t = 2; op_t = 2; #10;
    in0_t =  8; in1_t = 3; op_t = 3; #10;
    in0_t = 8'b00010110; in1_t = 4; op_t = 4; #10;
    in0_t = 8'b00110010; in1_t = 5; op_t = 5; #10;
    in0_t = 8'b00110100; in1_t = 6; op_t = 1; #10;
    $finish;
end

endmodule
```

Listing 5: top lab9 code

```
‘timescale 1ns / 1ps
//
   ////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/22/2020 12:33:10 PM
// Design Name:
// Module Name: top_lab9
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////



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

    wire [7:0] reg1_out;
    wire [7:0] alu_out;

    register #(.N(8)) Reg1 (
        .clk(clk),
        .rst(btnC),
        .en(btnD),
        .D(sw[7:0]),
        .Q(reg1_out)
    );
    assign led [7:0] = reg1_out;

    alu     ALU (
        .in0(sw[7:0]),
        .in1(reg1_out),
        .op(sw[11:8]),
        .out(alu_out)
    );

    register #(.N(8)) Reg2 (
        .clk(clk),
        .en(btnU),
        .rst(btnC),
        .D(alu_out),
        .Q(led [15:8])
    );

endmodule
```