ELC 2137 Lab 8: 4-digit Display

Spencer Stinson

October 21, 2020

Summary

In this experiment, we used the skills we had learned in the previous two labs to apply it to our 4 digit display. This display was able to show us BCD and hex conversions in one device. This device used many different converters that we had previously made to work together for the final product.

Code

Listing 1: mux2 code

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

// Company:
// Engineer:
//
// Create Date: 10/15/2020 11:37:37 AM
// Design Name:
// Module Name: mux2
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////



module mux2 #(
parameter BITS=4
)(
    input [BITS-1:0] in0,
    input [BITS-1:0] in1,
    input sel,
    output [BITS-1:0] out
    );
    assign out= sel ? in1 : in0;

endmodule
```

Listing 2: mux4 code

```
‘timescale 1ns / 1ps
//
   ////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/15/2020 12:08:15 PM
// Design Name:
// Module Name: mux4
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////



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

    always @*
    case(sel)
    2’b00: out = in0;
    2’b01: out = in1;
    2’b10: out = in2;
    2’b11: out = in3;
    endcase
endmodule
```

Listing 3: an decoder code

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

// Company:
// Engineer:
//
// Create Date: 10/15/2020 12:40:58 PM
// Design Name:
// Module Name: an_decoder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////



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

    always @*
    case(in)
    2'b00: out = 4'b1110;
    2'b01: out = 4'b1101;
    2'b10: out = 4'b1011;
    2'b11: out = 4'b0111;
    endcase
endmodule
```

Listing 4: sseg4 code

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

// Company:
// Engineer:
//
// Create Date: 10/15/2020 01:02:18 PM
// Design Name:
// Module Name: sseg4
// Project Name:
```

```
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////


module sseg4(
    input [15:0] data,
    input hex_dec,
    input sign,
    input [1:0] digit_sel,
    output [6:0] seg,
    output dp,
    output [3:0] an
    );
    wire [15:0] in0;
    wire [15:0] m2out;
    wire [3:0] m4out;
    wire [6:0] out0;
    wire [3:0] an_d_out;



    dd11b bcd11(
        .B(data[10:0]),
        .ones(in0[3:0]),
        .tens(in0[7:4]),
        .hundreds(in0[11:8]),
        .thousands(in0[15:12])
    );

    mux2 #(.BITS(16)) m2_seg (
        .in1(data),
        .in0(in0),
        .sel(hex_dec),
        .out(m2out)
    );

    mux4 m4_seg (
        .in0(m2out[3:0]),
        .in1(m2out[7:4]),
        .in2(m2out[11:8]),
        .in3(m2out[15:12]),
        .sel(digit_sel),
        .out(m4out)
    );

    sseg_decoder s_seg(
        .num(m4out),
        .sseg(out0)
    );

     an_decoder ad1 (
     .in(digit_sel),
     .out(an_d_out)
    );

    assign sel = sign & ~an_d_out[3];

    mux2 #(.BITS(7)) m2_seg2 (
        .in1(7'b0111111),
        .in0(out0),
        .sel(sel),
        .out(seg)
    );



    assign dp = 1;
    assign an = an_d_out;

endmodule
```

Listing 5: sseg4 manual code

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

// Company:
// Engineer:
//
// Create Date: 10/19/2020 09:00:35 PM
// Design Name:
// Module Name: sseg4_manual
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////

```

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

    sseg4 s4(
     .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
```

Results

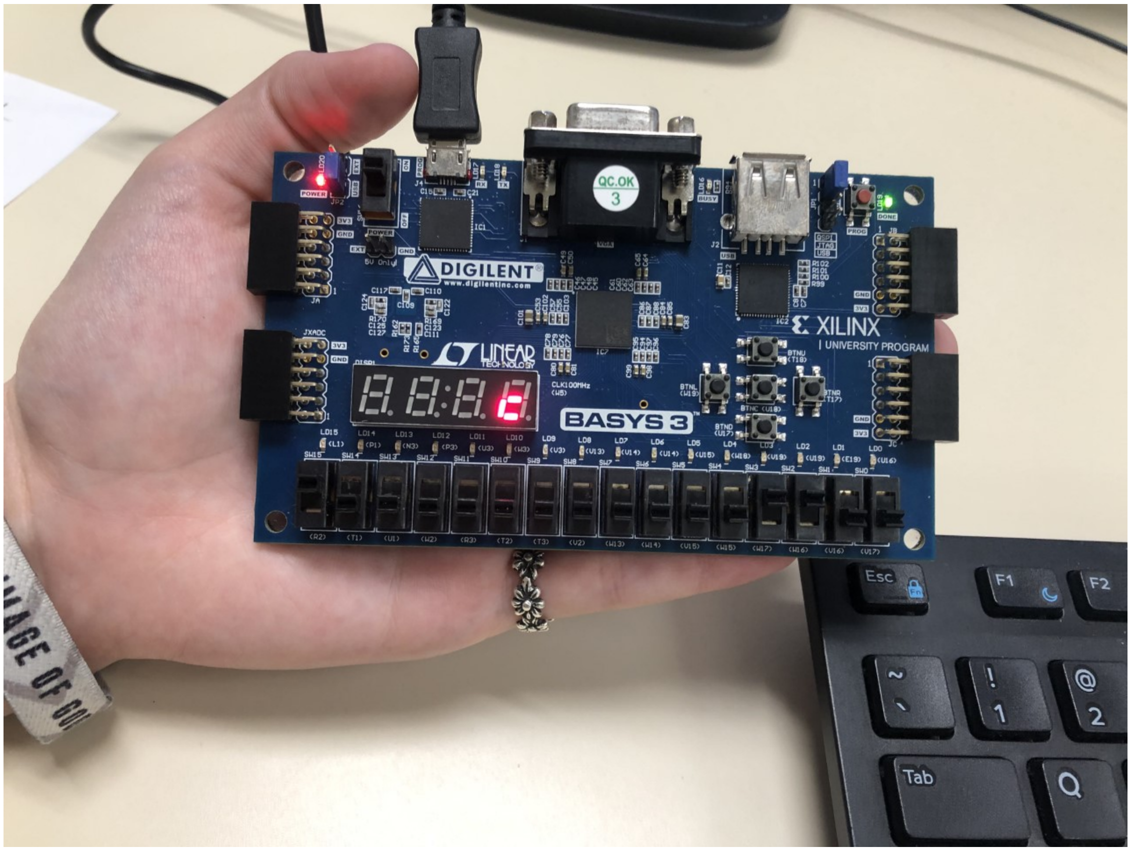


Figure 1: c displayed on first digit

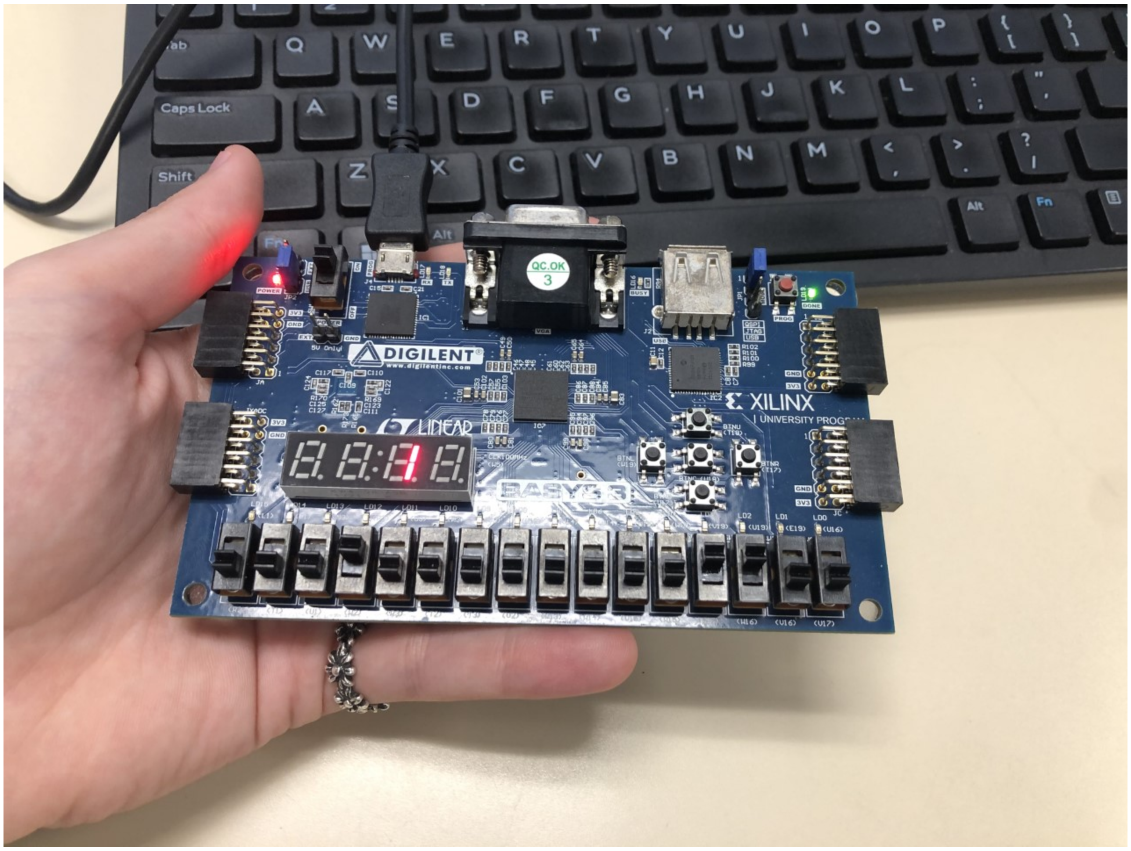


Figure 2: 1 displayed on second digit

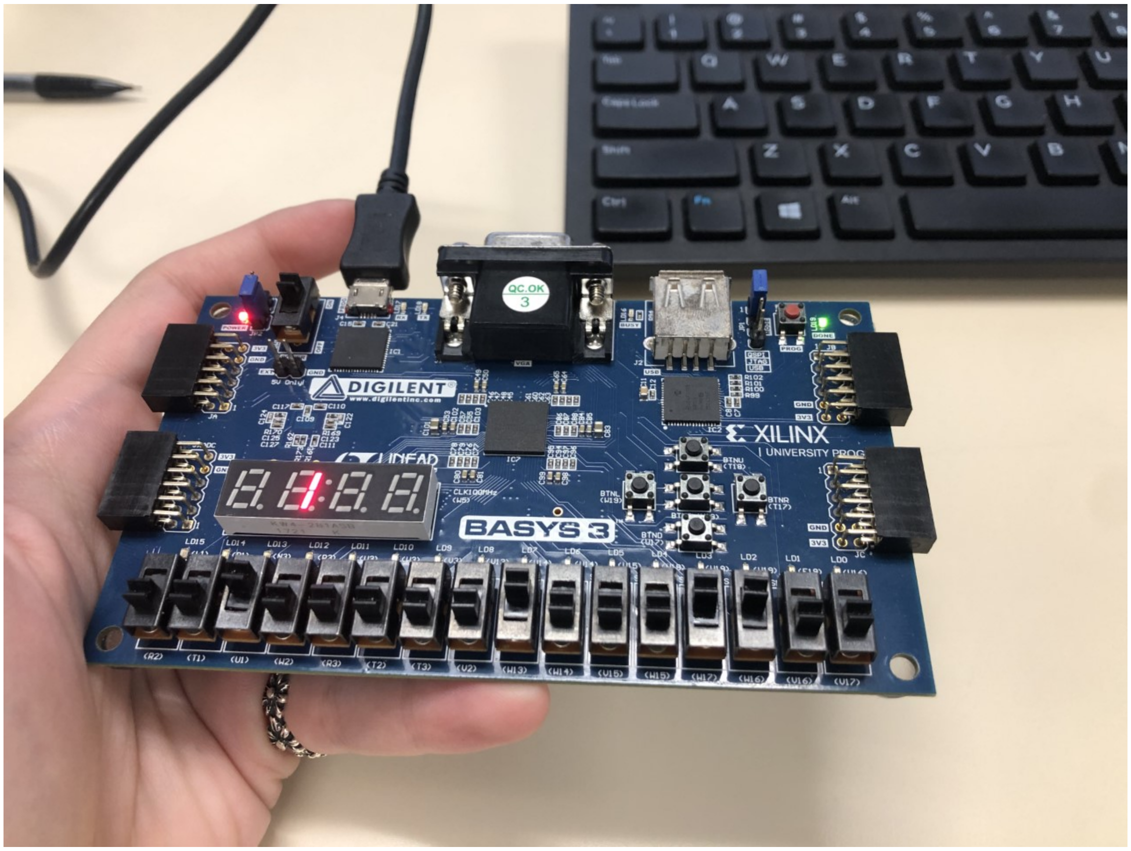


Figure 3: 1 displayed on third digit

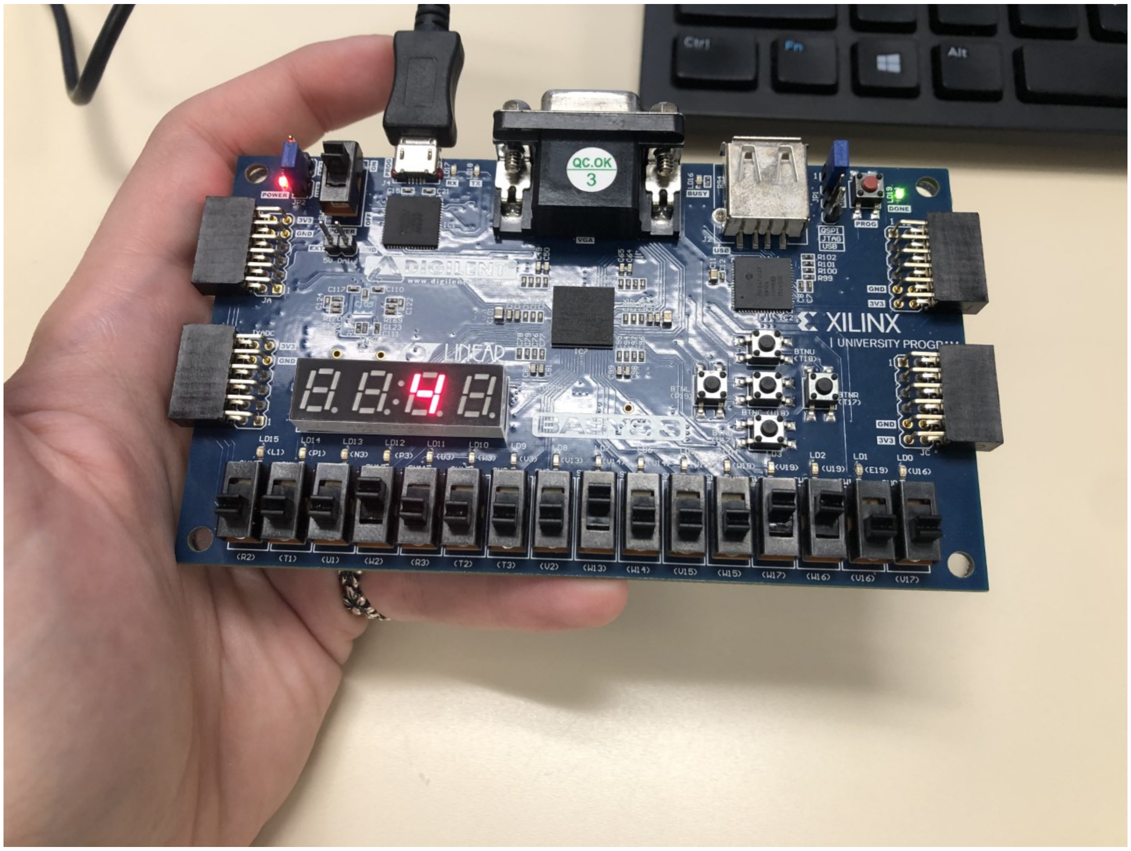


Figure 4: 4 displayed on second digit

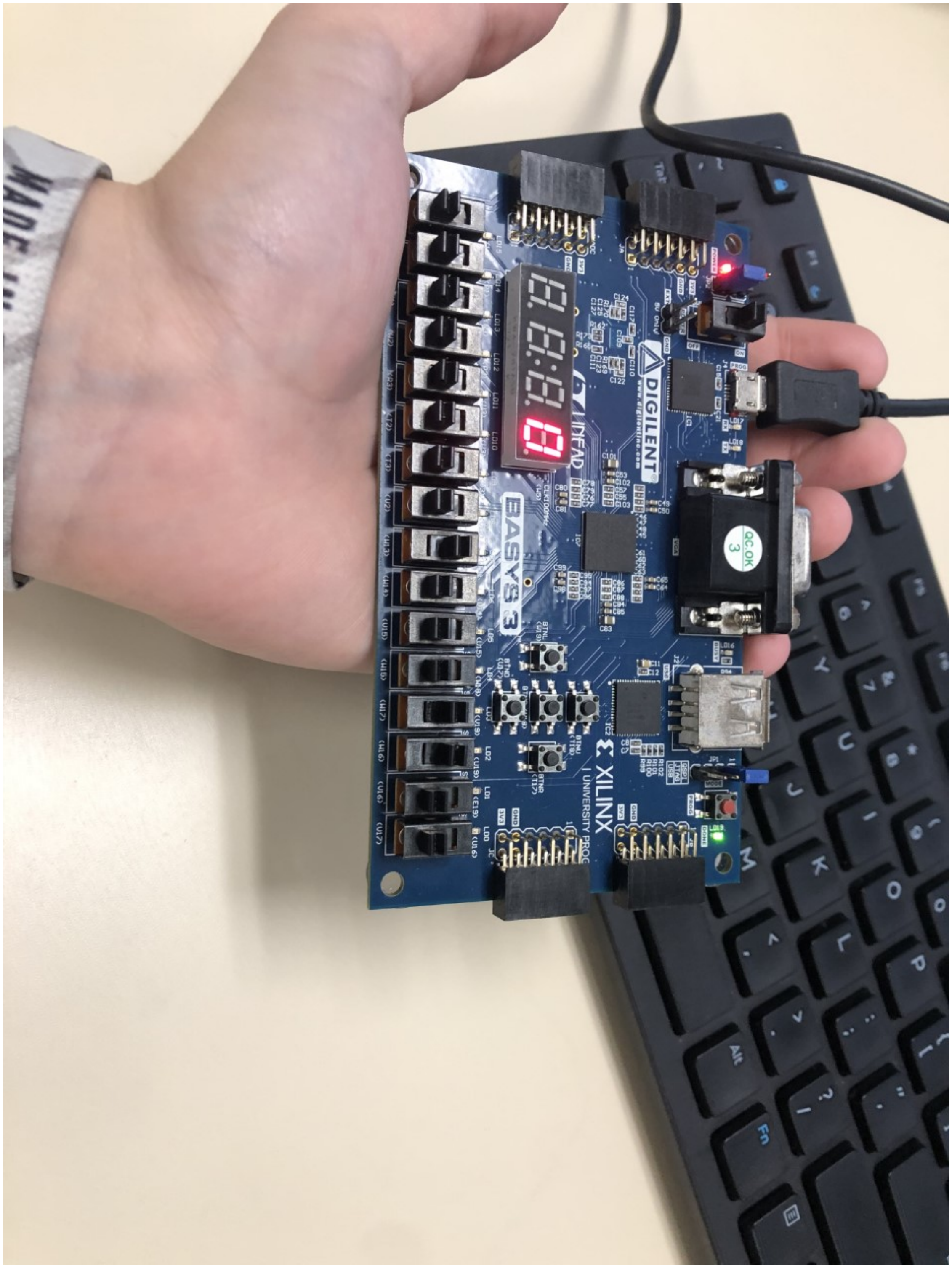


Figure 5: 0 displayed on first digit

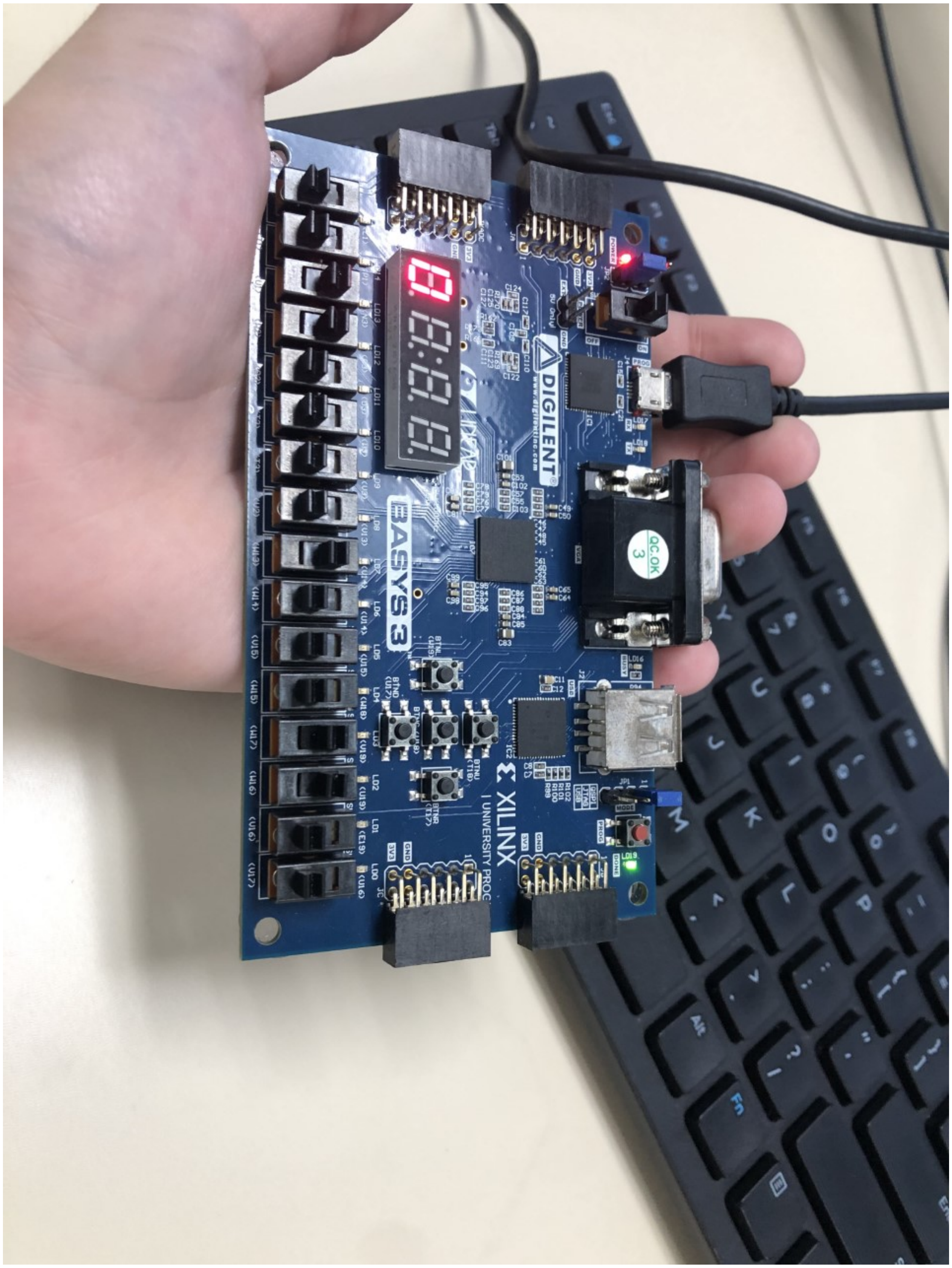


Figure 6: 0 displayed on fourth digit

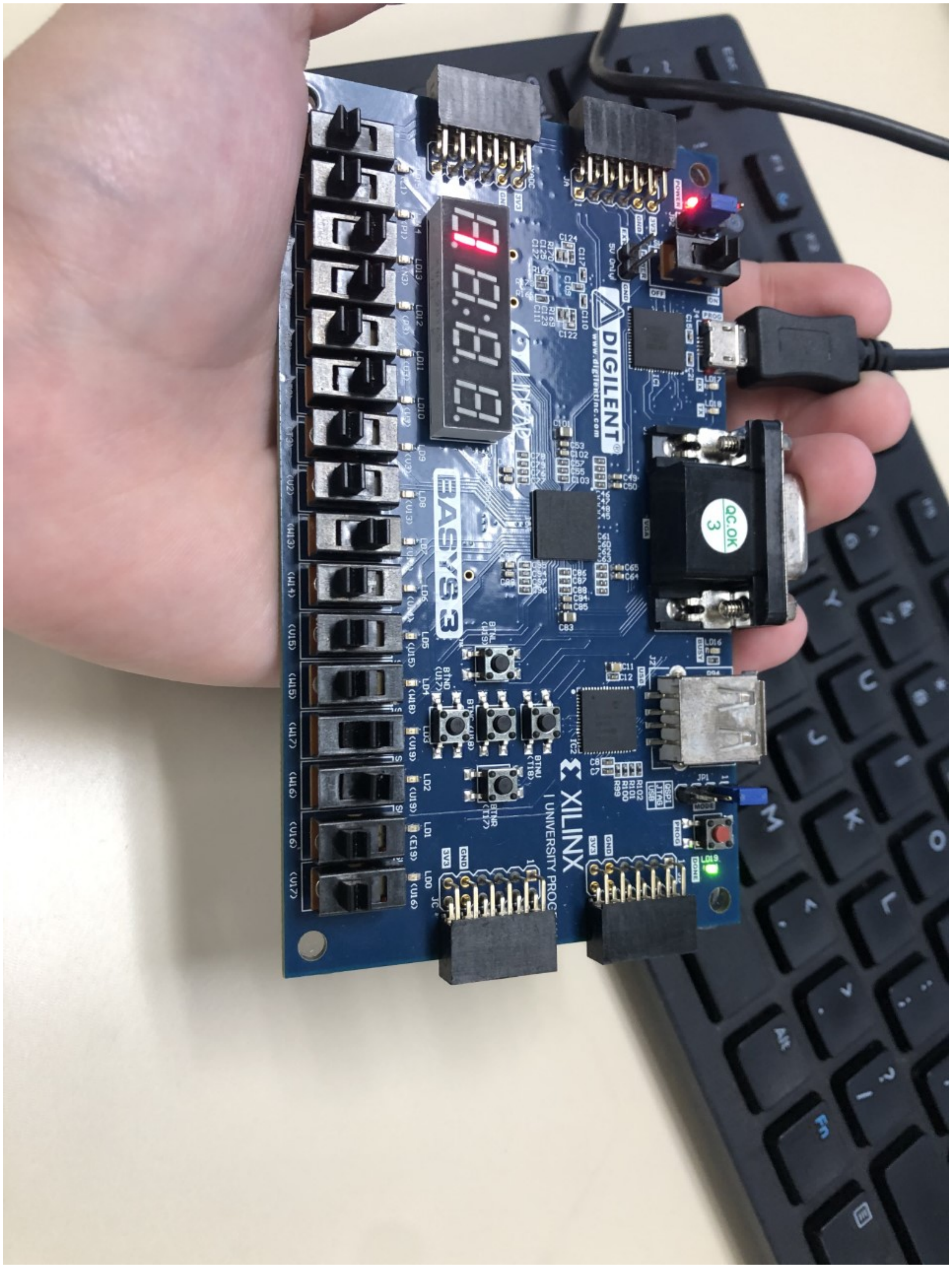


Figure 7: 1 displayed on fourth digit

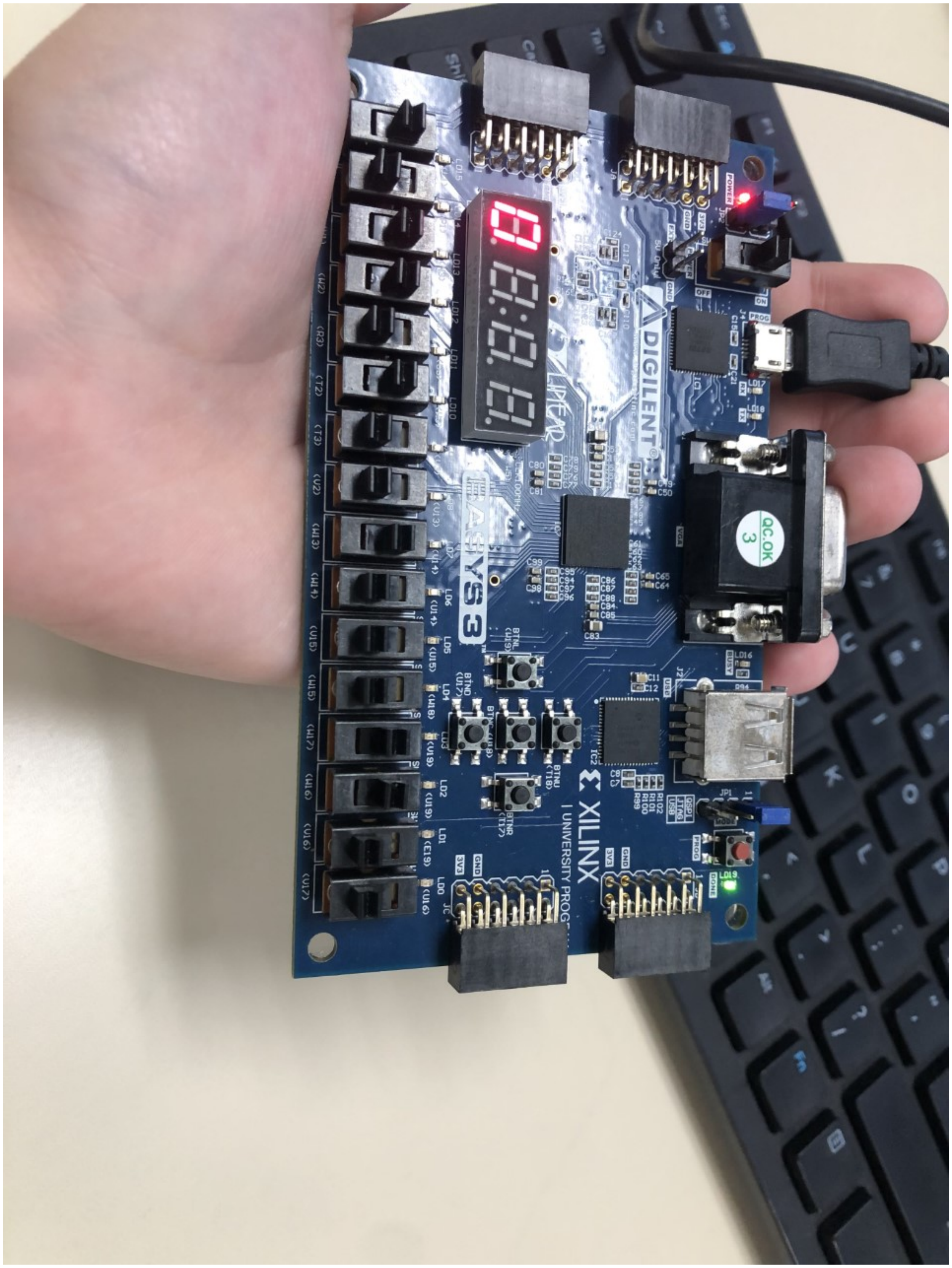


Figure 8: 0 displayed on fourth digit

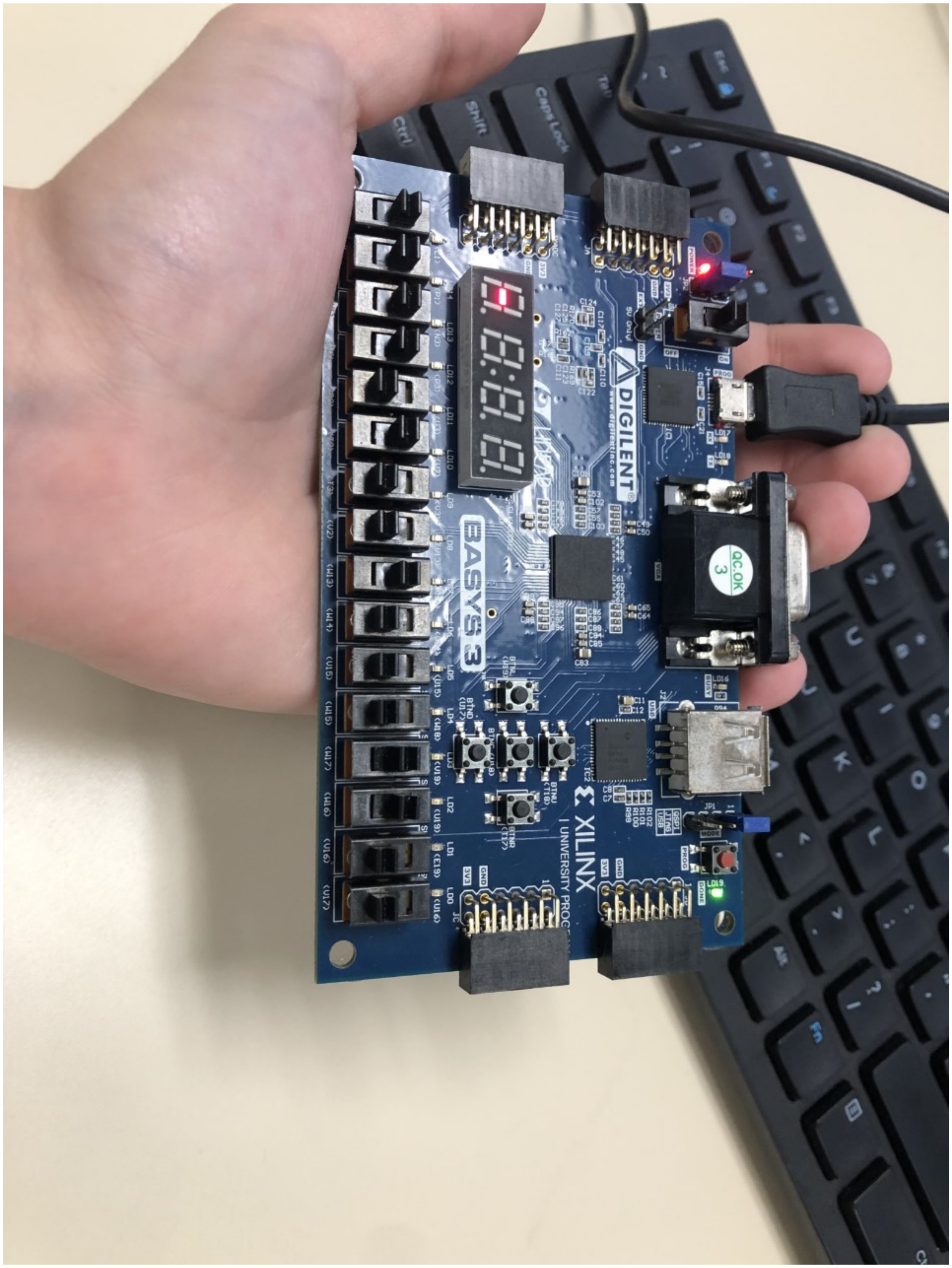


Figure 9: - displayed on fourth digit

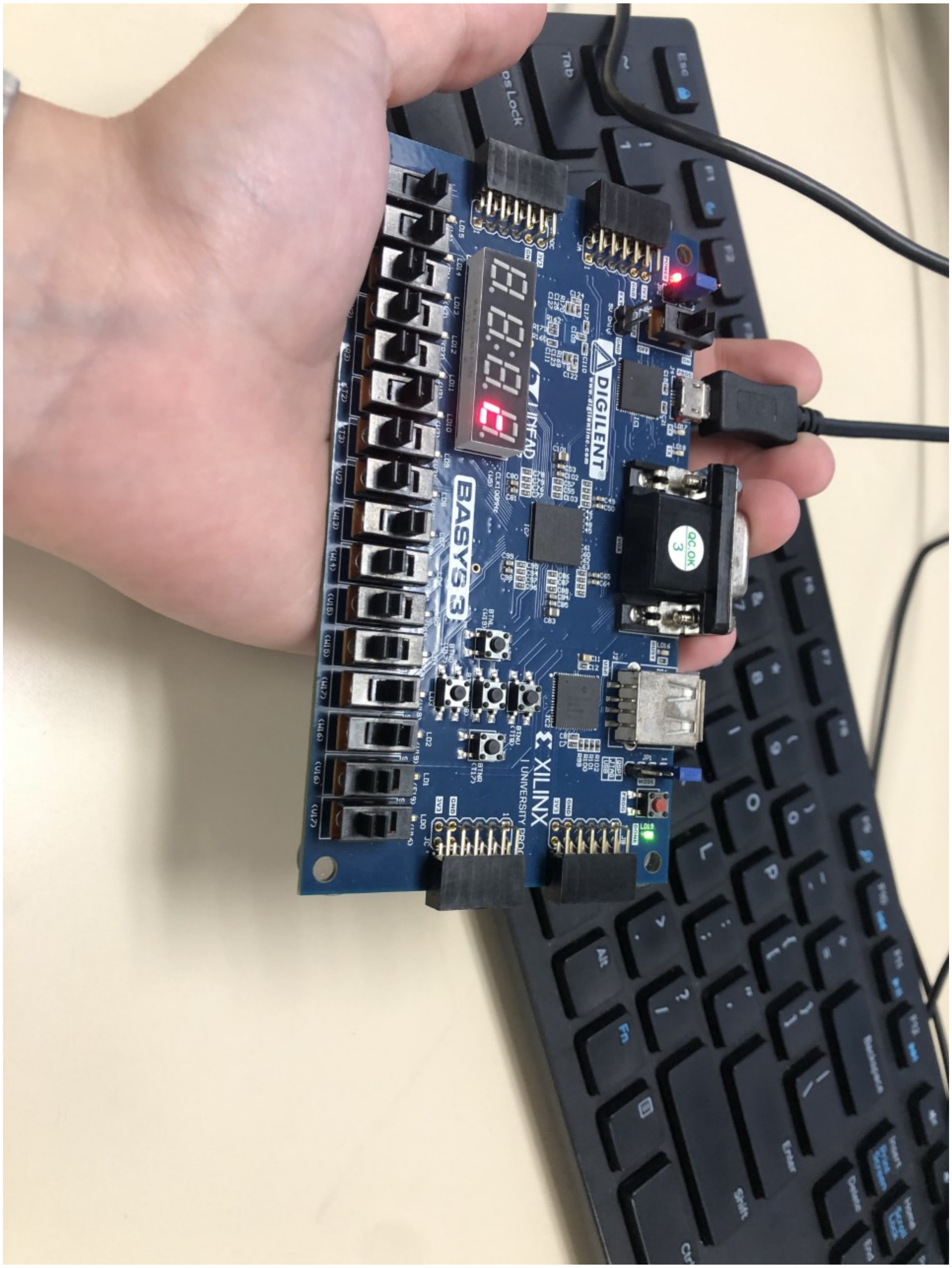
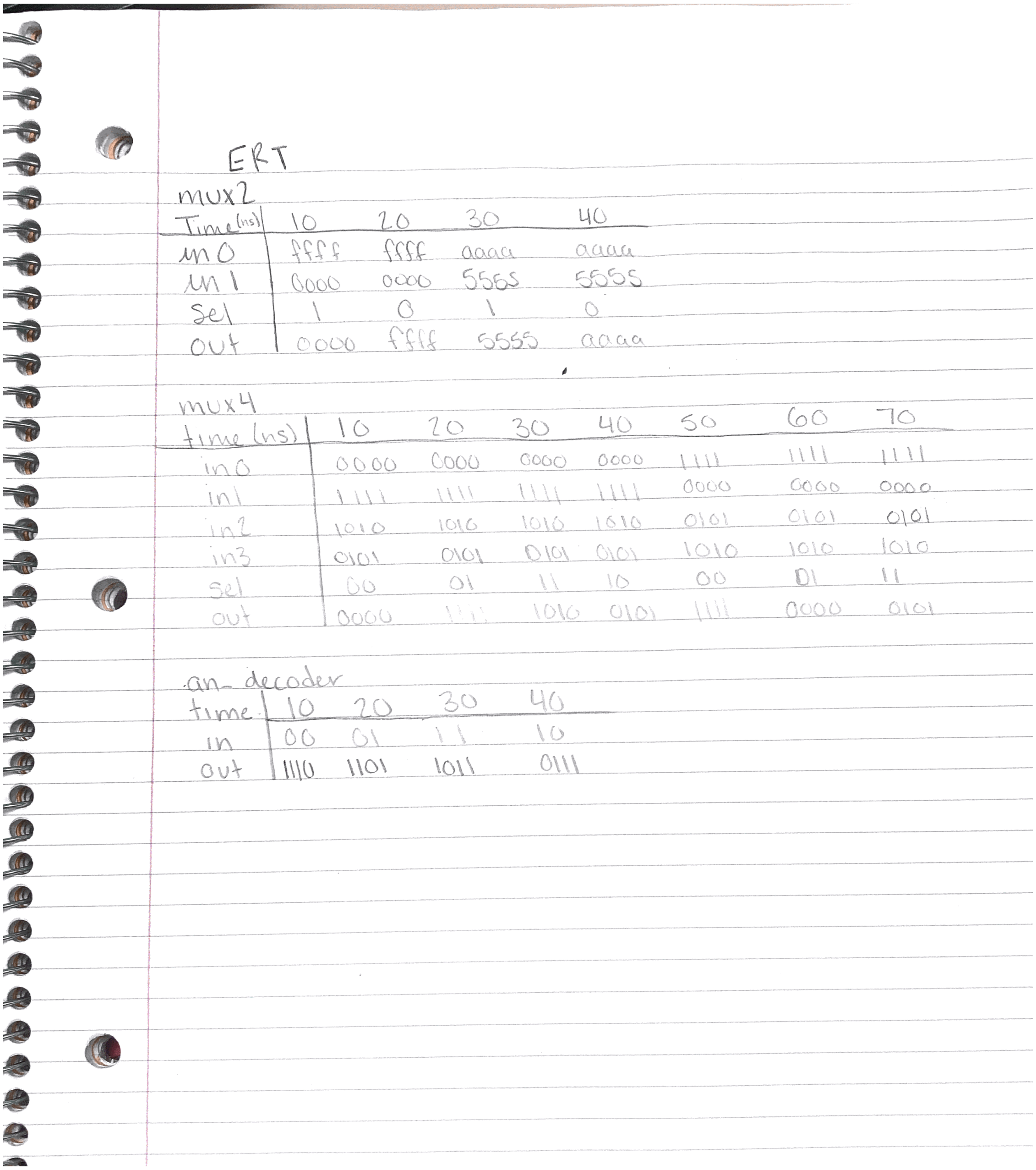


Figure 10: c displayed on first digit



ERT

mux2

Time(ns)	10	20	30	40
in0	ffff	ffff	aaaa	aaaa
in1	0000	0000	5555	5555
sel	1	0	1	0
out	0000	ffff	5555	aaaa

mux4

time (ns)	10	20	30	40	50	60	70
in0	0000	0000	0000	0000	1111	1111	1111
in1	1111	1111	1111	1111	0000	0000	0000
in2	1010	1010	1010	1010	0101	0101	0101
in3	0101	0101	0101	0101	1010	1010	1010
sel	00	01	11	10	00	01	11
out	0000	1111	1010	0101	1111	0000	0101

an_decoder

time	10	20	30	40
in	00	01	11	10
out	1110	1101	1011	0111

Figure 11: ERTs