ELC 2137 Lab 07: Binary Coded Decimal

Maddie Vorhies

October 14, 2020

Summary

The goal of this lab was to implement a 7-segment display, with binary as the input and hex as the output. To do this, I started out by using the double-dabble algorithm to convert hex values to BCD. I then drew the picture for an 11-bit double-dabble circuit and built it. I then created a simulation to test the circuit to make sure that it was properly built. I then took the mux and the 7-segment decoder from Lab 06 and modified it to fit into this lab. Then I created the sseg1 that connected all of those together. Lastly, I created a wrapper to make the circuit appear cleaner and more organized. I then hooked up my board to test out my completed circuit. Overall, I was able to successfully build the circuit and have it display the proper values on the board.

Results

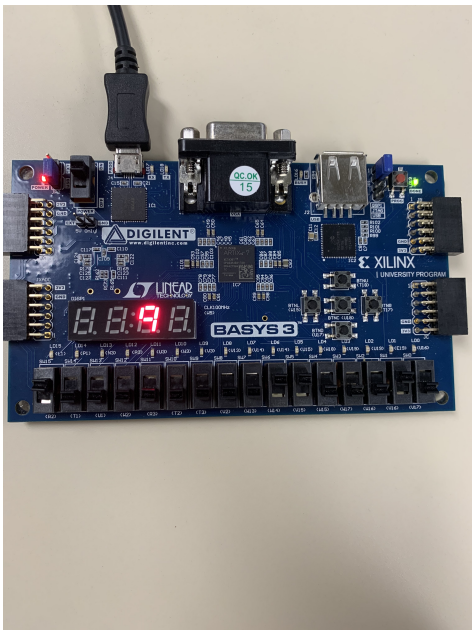
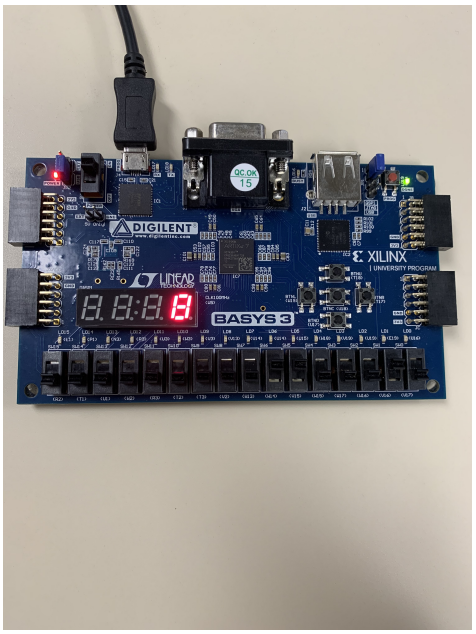
First Digit	Second Digit
	

Table 1: Board Pictures

Time (ns)	Input	Output
0	0000	0000
10	0001	0001
20	0010	0010
30	0011	0011
40	0100	0100
50	0101	1000
60	0110	1001
70	0111	1010
80	1000	1011
90	1001	1100
100	1010	1101
110	1011	11100
120	1100	1111
130	1101	0000
140	1110	0001
150	1111	0010

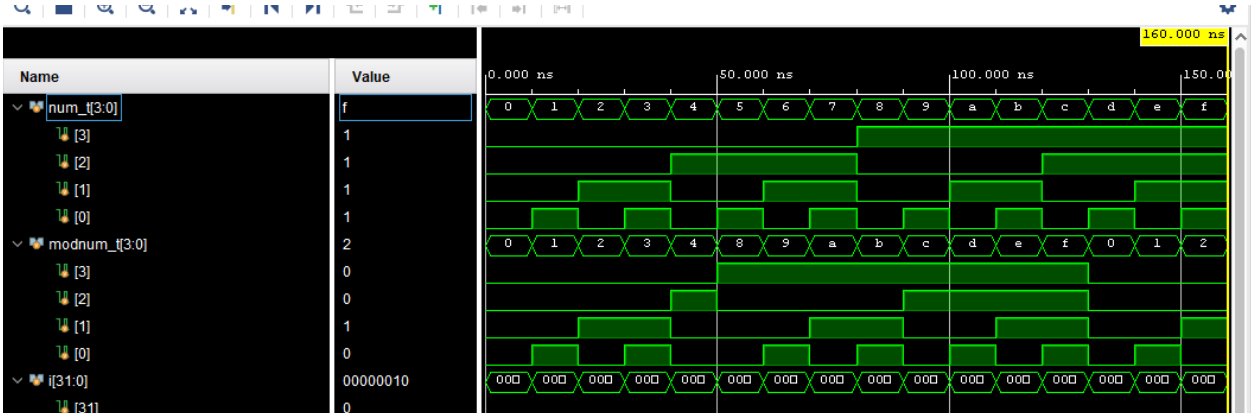


Figure 1: Simulation Waveform and ERT of Add3

Time (ns)	Input	Ouput	decimal
0	100100	00110110	36
10	001101	00010011	13
20	111001	01010111	57

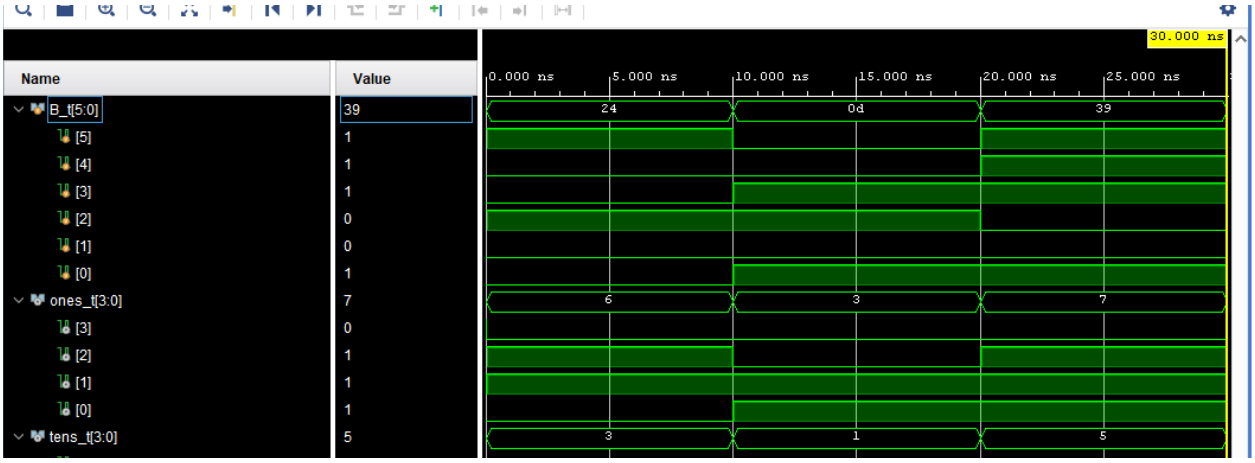
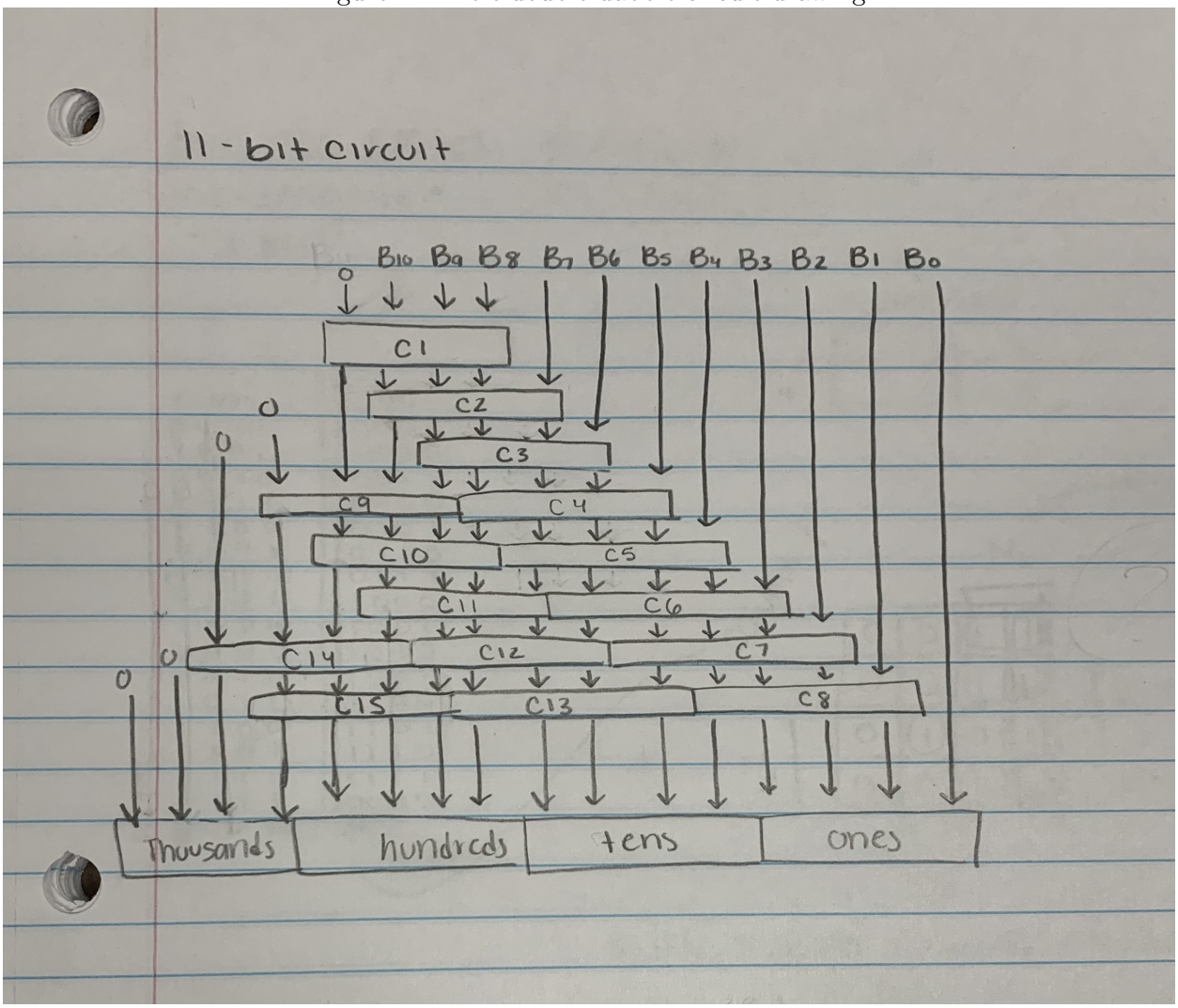


Figure 2: Simulation Waveform and ERT of 6-bit double dabble circuit

Time (ns)	Input	Ouput	decimal
0	10010011101	0001000110000001	1181
10	00110110110	0000010000111000	438
20	11100110001	0001100001000001	1841

Figure 3: Simulation Waveform and ERT of 11-bit double dabble circuit

Figure 4: 11-bit double dabble circuit drawing



Code

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

// Company:
// Engineer: Maddie Vorhies
//
// Create Date: 10/08/2020 11:42:38 AM
// Design Name:
// Module Name: add3
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module add3(
    input [3:0] num,
    reg [3:0] modnum
    );

    always @*

       if (num > 4)
          modnum = num + 3;
       else
          modnum = num;

endmodule
```

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

// Company:
// Engineer:
//
// Create Date: 10/08/2020 11:57:11 AM
// Design Name:
// Module Name: add3_test
// Project Name:
// Target Devices:
```

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



module add3_test();
    reg [3:0] num_t;
    reg [3:0] modnum_t;
    integer i;

    add3 dut (
       .num(num_t),
       .modnum(modnum_t)
    );

    initial begin
       for (i = 0; i <= 4'b1111 ; i = i + 1) begin
          num_t = i;
          #10;
       end
       $finish;
    end




endmodule
```

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

// Company:
// Engineer: Maddie Vorhies
//
// Create Date: 10/08/2020 12:20:57 PM
// Design Name:
// Module Name: bcd6
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module bcd6(
    input [5:0] B,
    output [3:0] ones,
    output [3:0] tens
    );

    reg [3:0] C1_out;
    reg [3:0] C2_out;
    reg [3:0] C3_out;

    add3 C1 (
       .num({1'b0, B[5:3]}),
       .modnum(C1_out[3:0])
    );

    add3 C2 (
       .num({C1_out[2:0], B[2]}),
       .modnum(C2_out[3:0])
    );

    add3 C3 (
       .num({C2_out[2:0], B[1]}),
       .modnum(C3_out[3:0])
    );

    assign tens[3:0] = {1'b0, C1_out[3], C2_out[3], C3_out[3]};
    assign ones[3:0] = {C3_out[2:0], B[0]};


endmodule
```

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

// Company:
// Engineer: Maddie Vorhies
//
// Create Date: 10/08/2020 01:03:19 PM
// Design Name:
// Module Name: bcd6_test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
```

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module bcd6_test();
    reg [5:0] B_t;
    wire [3:0] ones_t;
    wire [3:0] tens_t;

    bcd6 dut (
        .B(B_t),
        .ones(ones_t),
        .tens(tens_t)
    );

    initial begin

        B_t = 6'b100100; #10;
        B_t = 6'b001101; #10;
        B_t = 6'b111001; #10;

        $finish;

    end


endmodule
```

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

// Company:
// Engineer: Maddie Vorhies
//
// Create Date: 10/14/2020 07:00:18 PM
// Design Name:
// Module Name: bcd11
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
```

```
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module bcd11(
    input [10:0] B,
    output [3:0] ones,
    output [3:0] tens,
    output [3:0] hundreds,
    output [3:0] thousands
    );

    reg [3:0] C1_out, C2_out, C3_out, C4_out, C5_out;
    reg [3:0] C6_out, C7_out, C8_out, C9_out, C10_out;
    reg [3:0] C11_out, C12_out, C13_out, C14_out, C15_out;

    add3 C1 (
       .num({1'b0, B[10:8]}),
       .modnum(C1_out[3:0])
    );

    add3 C2 (
       .num({C1_out[2:0], B[7]}),
       .modnum(C2_out[3:0])
    );

    add3 C3 (
       .num({C2_out[2:0], B[6]}),
       .modnum(C3_out[3:0])
    );

    add3 C4 (
       .num({C3_out[2:0], B[5]}),
       .modnum(C4_out[3:0])
    );

    add3 C5 (
       .num({C4_out[2:0], B[4]}),
       .modnum(C5_out[3:0])
    );

    add3 C6 (
       .num({C5_out[2:0], B[3]}),
       .modnum(C6_out[3:0])
    );

    add3 C7 (
       .num({C6_out[2:0], B[2]}),
       .modnum(C7_out[3:0])
    );
```

```
    add3 C8 (
        .num({C7_out[2:0], B[1]}),
        .modnum(C8_out[3:0])
    );

    add3 C9 (
        .num({1'b0, C1_out[3], C2_out[3], C3_out[3]}),
        .modnum(C9_out[3:0])
    );

    add3 C10 (
        .num({C9_out[2:0], C4_out[3]}),
        .modnum(C10_out[3:0])
    );

    add3 C11 (
        .num({C10_out[2:0], C5_out[3]}),
        .modnum(C11_out[3:0])
    );

    add3 C12 (
        .num({C11_out[2:0], C6_out[3]}),
        .modnum(C12_out[3:0])
    );

    add3 C13 (
        .num({C12_out[2:0], C7_out[3]}),
        .modnum(C13_out[3:0])
    );

    add3 C14 (
        .num({1'b0, C9_out[3], C10_out[3], C11_out[3]}),
        .modnum(C14_out[3:0])
    );

    add3 C15 (
        .num({C14_out[2:0], C12_out[3]}),
        .modnum(C15_out[3:0])
    );

    assign ones[3:0] = {C8_out[2:0], B[0]};
    assign tens[3:0] = {C13_out[2:0], C8_out[3]};
    assign hundreds[3:0] = {C15_out[2:0], C13_out[3]};
    assign thousands[3:0] = {2'b00, C14_out[3], C15_out[3]};

endmodule
```

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

// Company:
// Engineer: Maddie Vorhies
```

```
//
// Create Date: 10/14/2020 07:26:53 PM
// Design Name:
// Module Name: bcd11_test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module bcd11_test();

    reg [10:0] B_t;
    wire [3:0] ones_t;
    wire [3:0] tens_t;
    wire [3:0] hundreds_t;
    wire [3:0] thousands_t;

    bcd11 dut (
      .B(B_t),
      .ones(ones_t),
      .tens(tens_t),
      .hundreds(hundreds_t),
      .thousands(thousands_t)
   );

    initial begin

      B_t = 11'b10010011101; #10;
      B_t = 11'b00110110110; #10;
      B_t = 11'b11100110001; #10;

      $finish;

   end


endmodule
```

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

// Company:
```

```
// Engineer: Maddie Vorhies
//
// Create Date: 10/14/2020 07:36:17 PM
// Design Name:
// Module Name: sseg1_BCD
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module sseg1_BCD(

   input [15:0] switches,
   output [3:0] out,
   output dp,
   output [6:0] sseg
);

   wire [3:0] num;
   wire [3:0] ones;
   wire [3:0] tens;
   wire [3:0] hundreds;
   wire [3:0] thousands;

   bcd11 bcd(
      .B(switches[10:0]),
      .tens(tens[3:0]),
      .ones(ones[3:0]),
      .hundreds(hundreds[3:0]),
      .thousands(thousands[3:0])
   );

   mux2_4b mux (
      .in1(tens[3:0]),
      .in0(ones[3:0]),
      .sel(switches[15]),
      .out(num)
   );

   sseg_decoder prototype (
      .num(num),
      .sseg(sseg)
   );
```

```
   assign out [0] = switches [15];
   assign out [1] = ~switches [15];
   assign dp = 1'b1;
   assign out [3:2] = 2'b11;

endmodule
```

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

// Company:
// Engineer:
//
// Create Date: 10/14/2020 08:03:57 PM
// Design Name:
// Module Name: sseg1_wrapper
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module sseg1_wrapper(
   input [15:0] sw,
   input clk,
   output [3:0] an,
   output dp,
   output [6:0] seg
);

   sseg1_BCD my_7seg1 (
      .switches(sw),
      .out(an),
      .dp(dp),
      .sseg(seg)
    );

endmodule
```