ELC 2137 Lab 07: Binary Coded Decimal

Yiting Wang

October 15, 2020

Summary

In the last lab, we implemented a 7-segment display. We input numbers in binary, and it outputs in hex. When you get used to it, hex is fine to read, but its not as easy as decimal, particularly for multi-digit numbers. In this lab, you will make a cheat button that will turn the display to decimal and then back to hex.

Q&A

There is no question in lab7.

Results

Firgure 1 is the simulation waveform and ERT of the Add3.

Time (ns):	0	10	20	30	40	50	60	110	120	130	140	150
num	0000	0001	0010	0011	0100	0101	0110	1011	1100	1101	1110	1111
out	0000	0001	0010	0011	0100	1000	1001	1110	1111	0000	0001	0010

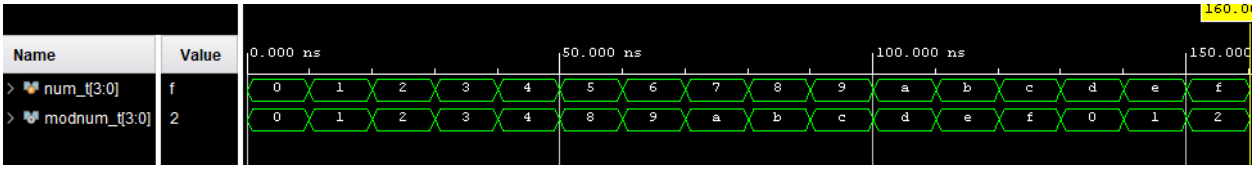

Figure 1: the simulation waveform and ERT of the 4-bit Multiplexer

Firgure 2 is the simulation waveform and ERT of the 6-bit BCD.

Firgure 3 is the simulation waveform and ERT of the 11-bit BCD.

Time (ns):	0	10	20	600	610	620	630
Bin	000000	000001	000010	111100	111101	111110	111111
ones	0000	0001	0010	0000	0001	0010	0011
tens	0000	0000	0000	0110	0110	0110	0110

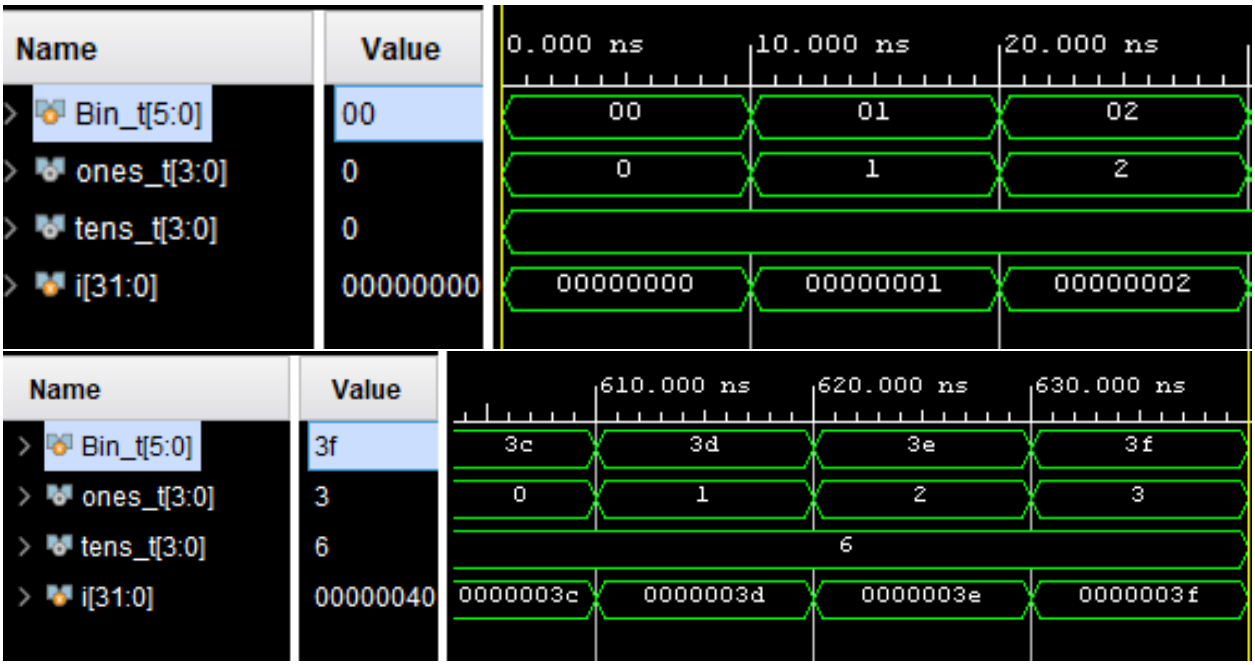

Figure 2: the simulation waveform and ERT of the 6-bit BCD

Code

File Inclusion

Listing 1: Add3 Verilog code

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

// Company: ELC 2137
// Engineer: Yiting Wang
// Create Date: 10/08/2020
//
    //////////////////////////////////////////////////////////////////////////////////



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

    always @*
```

Time (ns):	0	10	20	30	40	50	20430	20440	20450	20460
Bin	000	001	002	003	004	005	7fb	7fc	7fd	7fe
ones	0000	0001	0010	0011	0100	0101	0011	0100	0101	0110
tens	0000	0000	0000	0000	0000	0000	0100	0100	0100	0100
hund	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
thou	0000	0000	0000	0000	0000	0000	0010	0010	0010	0010

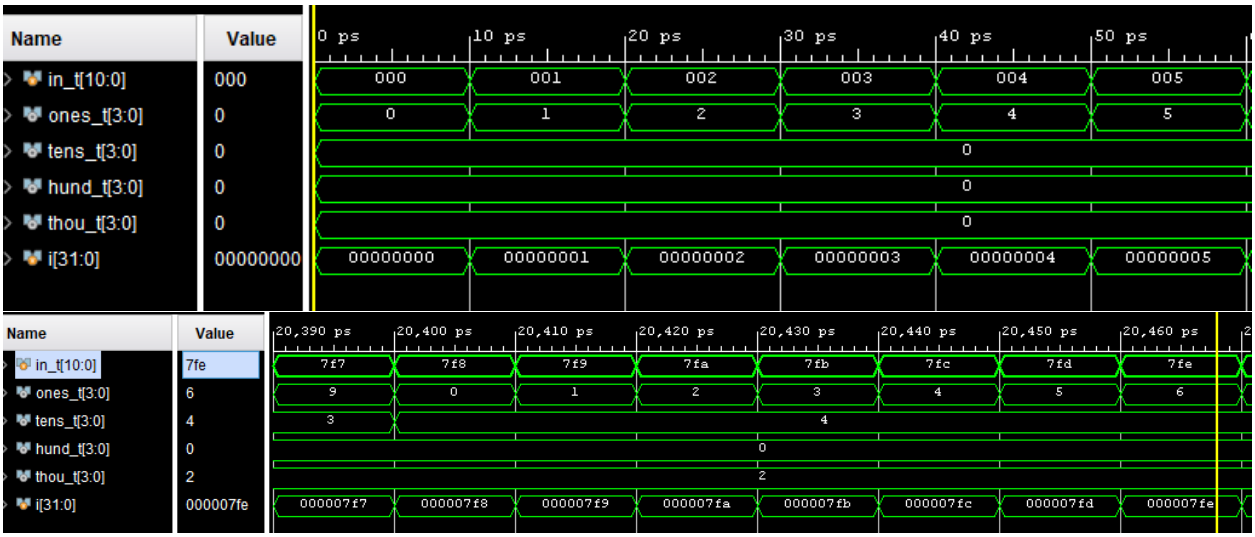

Figure 3: the simulation waveform and ERT of the 11-bit BCD

```
        if (num > 4'h4)
            mod = num + 4'h3;
        else
            mod = num;

endmodule
```

File Inclusion

Listing 2: Add3 Test Benches Verilog code

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

// Company: ELC 2137
// Engineer: Yiting Wang
// Create Date: 10/08/2020
//
    //////////////////////////////////////////////////////////////////////////////////



module add3_test();
```

```
    reg [3:0] num_t;
    wire [3:0] mod_t;

    add3 dut(
        .num(num_t),
        .mod(mod_t)
    );


    integer i;

    initial begin
        for (i=4'h0; i<=4'hf; i=i+1) begin
            num_t = i; #10;
        end
        $finish;
    end

endmodule
```

File Inclusion

Listing 3: 6-bit BCD Verilog code

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

// Company: ELC 2137
// Engineer: Yiting Wang
// Create Date: 10/08/2020
//
   //////////////////////////////////////////////////////////////////////////////////



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

    wire [2:0] w1, w2;
    wire [6:1] Dout;

    add3 C1(
        .num({ 0, Bin[5:3] }),
        .mod({ Dout[6], w1 })
    );

    add3 C2(
        .num({ w1, Bin[2] }),
        .mod({ Dout[5], w2})
    );
```

```
    add3 C3(
        .num({ w2, Bin[1] }),
        .mod(Dout[4:1])
    );

    assign ones = {Dout[3:1], Bin[0]};
    assign tens = {0, Dout[6:4]};

endmodule
```

File Inclusion

Listing 4: 6-bit BCD Test Benches Verilog code

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

// Company: ELC 2137
// Engineer: Yiting Wang
// Create Date: 10/08/2020
//
   //////////////////////////////////////////////////////////////////////////////////



module bcd6_test();

    reg [5:0] Bin_t;
    wire [3:0] ones_t, tens_t;

    bcd6 dut_bcd6(
    .Bin(Bin_t),
    .ones(ones_t), .tens(tens_t)
    );

    integer i;

    initial begin
        for(i=6'h0; i<=6'h3f; i=i+1) begin
            Bin_t = i; #10;
        end
        $finish;
    end

endmodule
```

File Inclusion

Listing 5: 11-bit BCD Verilog code

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

```
//
   //////////////////////////////////////////////////////////////////////////////////

// Company: ELC 2137
// Engineer: Yiting Wang
// Create Date: 10/08/2020
//
   //////////////////////////////////////////////////////////////////////////////////



module bcd11(
    input [10:0] in,
    output [3:0] ones,
    output [3:0] tens,
    output [3:0] hund,
    output [3:0] thou
    );

    wire [12:1] Y;
    wire [3:0] w1, w2, w3, w4, wa, w5, wb, w6, wc, w7, wd, wA;

    add3 C1_bcd11(
        .num({ 0, in[10:8] }),
        .mod(w1)
    );

    add3 C2_bcd11(
        .num({ w1[2:0], in[7] }),
        .mod(w2)
    );

    add3 C3_bcd11(
        .num({ w2[2:0], in[6] }),
        .mod(w3)
    );

    add3 C4_bcd11(
        .num({ w3[2:0], in[5] }),
        .mod(w4)
    );

    add3 C5_bcd11(
        .num({ w4[2:0], in[4] }),
        .mod(w5)
    );

    add3 C6_bcd11(
        .num({ w5[2:0], in[3] }),
        .mod(w6)
    );

    add3 C7_bcd11(
        .num({ w6[2:0], in[2] }),
```

```
        .mod(w7)
    );

    add3 C8_bcd11(
        .num({ w7[2:0], in[1] }),
        .mod(Y[4:1])
    );

    add3 Ca_bcd11(
        .num({ 0, w1[3], w2[3], w3[3] }),
        .mod(wa)
    );

    add3 Cb_bcd11(
        .num({ wa[2:0], w4[3] }),
        .mod(wb)
    );

    add3 Cc_bcd11(
        .num({ wb[2:0], w5[3] }),
        .mod(wc)
    );

    add3 Cd_bcd11(
        .num({ wc[2:0], w6[3] }),
        .mod(wd)
    );

    add3 Ce_bcd11(
        .num({ wd[2:0], w7[3] }),
        .mod(Y[8:5])
    );

    add3 CA_bcd11(
        .num({ 0, wa[3], wb[3] , wc[3] }),
        .mod(wA)
    );

    add3 CB_bcd11(
        .num({ wA[2:0], wd[3] }),
        .mod(Y[12:9])
    );

    assign ones = {Y[3:1], in[0]};
    assign tens = Y[7:4];
    assign hund = Y[11:8];
    assign thou = {2'b00, wA[3],Y[12]};

endmodule
```

File Inclusion

Listing 6: 11-bit BCD Test Benches Verilog code

```
‘timescale 1s / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company: ELC 2137
// Engineer: Yiting Wang
// Create Date: 10/08/2020
//
   //////////////////////////////////////////////////////////////////////////////////



module bcd11_test();

    reg [10:0] in_t;
    wire [3:0] ones_t, tens_t, hund_t, thou_t;

    bcd11 dut_bcd11(
        .in(in_t),
        .ones(ones_t), .tens(tens_t), .hund(hund_t), .thou(thou_t)
    );

    integer i;

    initial begin
        for (i=11'h0; i<=11'h7ff; i=i+1) begin
            in_t = i; #10;
        end
    end

endmodule
```

File Inclusion

Listing 7: sseg1 BCD Verilog code

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

// Company: ELC 2137
// Engineer: Yiting Wang
// Create Date: 10/08/2020
//
   //////////////////////////////////////////////////////////////////////////////////



module sseg1_BCD(
    input [15:0] sw,
    input clk,//do nothing with it but borad needs it to run
    output [3:0] an,
    output dp,
    output [6:0] seg
```

```
    );

    wire tens_bcd11, ones_bcd11, hund_bcd11, thou_bcd11, out_mux;

    bcd11 my_bcd11_sseg1(
        .in(sw[10:0]),
        .tens(tens_bcd11), .ones(ones_bcd11), .hund(hund_bcd11), .thou(
           thou_bcd11)
    );

    mux2_4b my_mux_sseg1(
        .in1(tens_bcd11), .in0(ones_bcd11), .sel(sw[15]),
        .out(out_mux)
    );

    sseg_decoder my_sseg_decoder_sseg1(
        .num(out_mux),
        .sseg(seg)
    );

    assign dp = 1;
    assign an[3:2] = 2'b11;
    assign an[1] = ~sw[15];
    assign an[0] = sw[15];

endmodule
```