ELC 2137 Lab 8: Four Digit Display

CJ Jones, Jake Simmons

March 26, 2020

Summary

This lab exlpored using a Basys3 board to produce a 4-digit display with the ability to switch between hexadecimal and decimal (BCD) output using the previous 2-digit, 7-segment display and BCD converter. Using Verilog, some skills gained in this lab include: Useing parameters to create flexible, reusable modules, import modules, modify modules, use them to design a modular system using constraint files,and creating a design on a board. Overall, this lab demonstrated how to utilize software and programmable logic to produce a hardware output.

Results

Time (ns):	0	10	20	30
in 0	0001	0002	0004	0006
in 1	0000	0003	0005	0007
sel	0	0	1	1
out	0001	0002	0005	0007

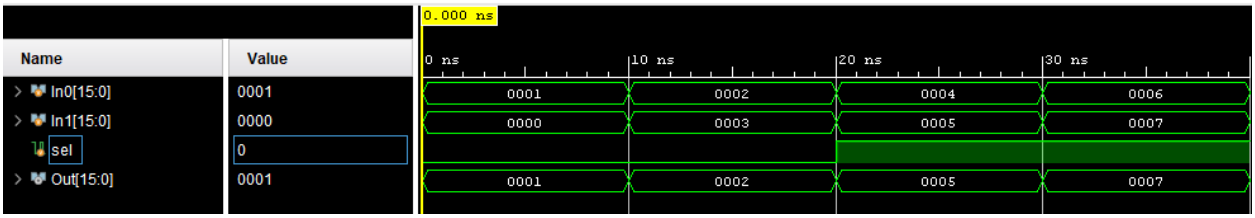

Figure 1: Mux2 simulation waveform and ERT

Time (ns):	0	10	20	30
In0	0	4	8	c
In1	1	5	9	d
In2	2	6	a	e
In3	3	7	b	f
sel	0	1	z	3
Out	0	5	a	f

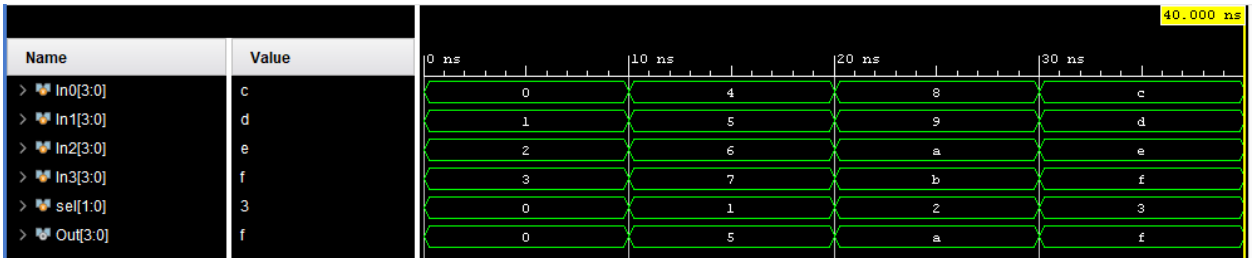

Figure 2: Mux4 simulation waveform and ERT

Time (ns):	0	10	20	30
In	0	1	2	3
Out	e	d	b	7

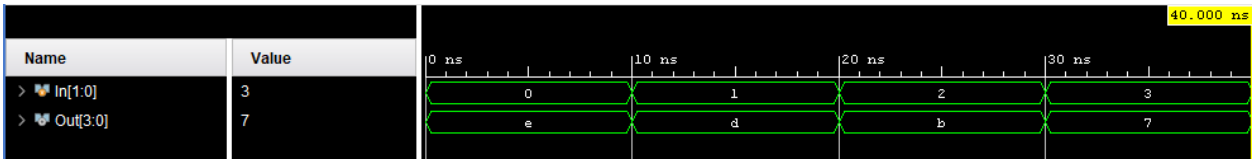

Figure 3: Annode decoder simulation waveform and ERT

```
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will sta
#   }
# }
# run 1000ns
    +--------- Digit 3 (? = incorrect segment values)
    |+-------- Decimal point
    ||+------- Digit 2 (? = incorrect segment values)
    |||+------ Decimal point
    ||||+----- Digit 1 (? = incorrect segment values)
    |||||+---- Decimal point
    ||||||+--- Digit 0 (? = incorrect segment values)
    |||||||+-- Decimal point
    ||||||||
-->       0
-->       C
-->       2
-->     1
-->   0
-->   1
-->     4
-->       0
--> 0
--> 1
--> 0
--> -
-->   4
-->     8
-->       C
$finish called at time : 150 ns : File "C:/Users/School/Desktop/basys3.sv" Line 98
xsim: Time (s): cpu = 00:00:11 ; elapsed = 00:00:06 . Memory (MB): peak = 792.820 ; g
INFO: [USF-XSim-96] XSim completed. Design snapshot 'basys3_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:15 ; elapsed = 00:00:31 . Memory (MB): peak
```

Figure 4: TCL Console output

Code

Listing 1: Mux 2 Source File

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

module mux2 #(parameter N=2)(
input [N-1:0] in0, in1,
input sel,
output reg [15:0] out
);

always @*
case(sel)
0: out = in0 ;
default: out = in1 ;
endcase
endmodule
```

Listing 2: Mux 2 Test Bench Code

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

module mux2_Test( );

reg [15:0] In0, In1;

reg sel;

wire [15:0] Out;

mux2 #(.N(16))  m1(

.in0(In0), .in1(In1),

.out(Out), .sel(sel)

);


initial begin

In0 = 16'b0000000000000000; In1 = 16'b0000000000000001; sel = 1'd0; #10;

In0 = 16'b0000000000000010; In1 = 16'b0000000000000011; sel = 1'd0; #10;

In0 = 16'b0000000000000100; In1 = 16'b0000000000000101; sel = 1'd1; #10;

In0 = 16'b0000000000000110; In1 = 16'b0000000000000111; sel = 1'd1; #10;
```

```
$finish;

end

endmodule
```

Listing 3: Mux 4 Source File

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

module mux4 #(parameter N =4)(
input [N-1:0] in0, in1, in2, in3,
input [1:0] sel,
output reg [3:0] out
);

always @*
case(sel)
0: out = in0;
1: out = in1;
2: out = in2;
default: out = in3;
endcase
```

Listing 4: Mux 4 Test Bench Code

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

module mux4_Test();
reg [3:0] in0, in1, in2, in3;
reg [1:0] sel;
wire [3:0] out;

mux4 m4(
.in0(in0), .in1(in1), .in2(in2), .in3(in3), .sel(sel), .out(out)
);

initial begin
in0 = 4'b0000; in1 = 4'b0001; in2 = 4'b0010; in3 = 4'b0011; sel = 2'b00;
   #10;
in0 = 4'b0100; in1 = 4'b0101; in2 = 4'b0110; in3 = 4'b0111; sel = 2'b01;
   #10;
in0 = 4'b1000; in1 = 4'b1001; in2 = 4'b1010; in3 = 4'b1011; sel = 2'b10;
   #10;
in0 = 4'b1100; in1 = 4'b1101; in2 = 4'b1110; in3 = 4'b1111; sel = 2'b11;
   #10;

$finish;
end
endmodule
```

Listing 5: Annode Decoder Source File

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

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

always @*
case(in)
0: out = 4'b1110;
1: out = 4'b1101;
2: out = 4'b1011;
default: out = 4'b0111;
endcase
endmodule
```

Listing 6: Annode Decoder Test Bench Code

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

module annode_decoder_test();
reg [1:0] in;
wire [3:0] out;

annode_decoder ant(

.in(in), .out(out)
);

initial begin
in = 2'b00; #10;
in = 2'b01;  #10;
in = 2'b10;  #10;
in = 2'b11;  #10;

$finish;
end
endmodule
```

Listing 7: Sseg4 Source File

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

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] W1 ;
wire [15:0] W2 ;
wire [3:0] W3;
wire [6:0] W4;
wire [3:0] W5;
wire W6;
BCD11_2 B1( .in11(data[10:0]), .out11(W1));
mux2 #(.N(16)) B2( .in0(W1), .in1(data), .sel(hex_dec), .out(W2));
mux4 B3( .in0(W2[3:0]), .in1(W2[7:4]), .in2(W2[11:8]), .in3(W2[15:12]), .
   sel(digit_sel), .out(W3));
sseg_decoder B5( .num(W3), .sseg(W4));
mux2 #(.N(7)) B6( .in0(W4), .in1(7'b0111111), .out(seg) , .sel(W6));
and G2( W6, sign, ~W5[3]);
annode_decoder B7( .in(digit_sel), .out(W5));
assign dp = 1;
assign an = W5;
endmodule
```

Listing 8: Sseg4 Manual Source File

```
// Jake Simmons and Chris Jones , ELC 2137, 2020 -3-24

module sseg4_manual(
input [15:0] sw,
output [6:0] seg,
output dp,
output [3:0] an
);
sseg4 Sseg4(
.digit_sel(sw[13:12]), .sign(sw[14]), .hex_dec(sw[15]),
.data({4'b0000, sw[11:0]}), .seg(seg), .dp(dp), .an(an)
);
endmodule
```