ELC 2137 Lab 7: MUX and 7-segment Decoder

CJ Jones, Jake Simmons

March 4, 2020

Summary

This lab explored using a Basys3 board to produce an 8-bit number on a 7-segment display through a MUX combinational logic design. This lab produced a cheat button that turns the display from hex to decimal, and back because of the difficulty of reading the decimal numbers. Using Verilog, some skills gained in this lab include: writing a multiplexer utilizing the conditional operator, use the double-dabble algorithm to convert hex values into BCD, using multi-bit signals, using constraint files,and creating a design on a FPGA board. Overall, this lab demonstrated how to utilize software and programmable logic to produce a hardware output.

Q&A

1. How many wires are connected to the 7-segment display?

   The current seven segment display includes two 7-segment displays, so in total we used 10 wires. This is because the only wires we are "adding" in our current method are the anode wires while the segments are connected, effectively, across and don't provide any additional wires. Thus we have 8 wires that take care of all the segments, and 2 anodes for each sepearate digit display, making 10 in total.

2. If the segments were not all connected together, how many wires would there have to be?

   If the segments were not all connected together, there would have to be separate wire systems for each individual 7-segment display. For the two (out of four) display numbers we used, each display would need 9 wires, totalling to 18 wires in total. Each display would have 8 wires for each segment, and the dp, and a wire for its anode, totalling to 9 per individual display, 18 all together.

3. Why do we prefer the current method vs. separating all of the segments?

   We prefer the current method because it uses less wires. As the circuits become larger, the displays can only account for so many wires before it becomes egregiously expensive, and/or the chip breaks because of heating issues. Therefore, in using our current display for production, less wires means a cheaper circuit that is safer(less likely to fry), thus allowing for more displays to safely be possible. Therefore, the current method is preferred.

Results

1. Not saving files before running again

2. Continually getting Z's when trying to run simulations
3. Forgetting to instantiate sseg1 test in the testbench file.

Time (ns):	0	10	20	30
in 0	0000	0001	0010	0010
in 1	0001	0010	0001	0001
sel	0	1	0	1
out	0000	0010	0010	0001

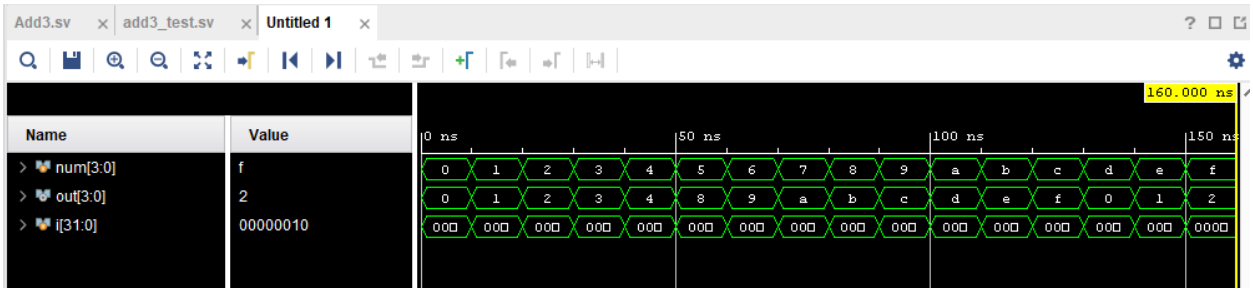

Figure 1: Add 3 simulation waveform and ERT

Time (ns):	0	10	20	30
in 0	0000	0001	0010	0010
in 1	0001	0010	0001	0001
sel	0	1	0	1
out	0000	0010	0010	0001

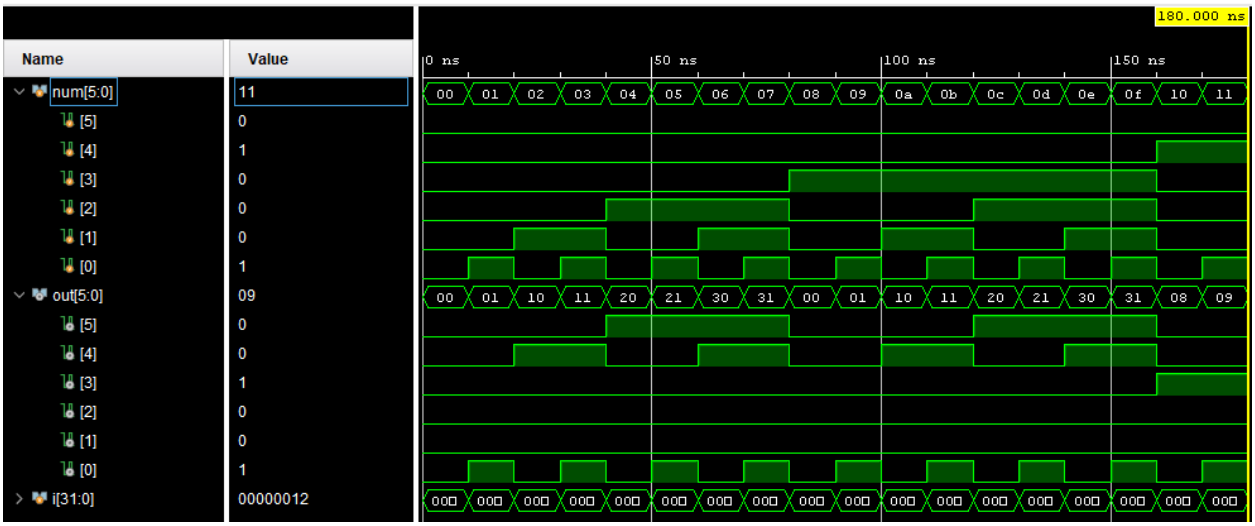

Figure 2: BCD simulation waveform and ERT

Code

Listing 1: Add 3 Module Code

```
module Add3(
input  [3:0] Cin,
output reg [3:0] Cout);

always @*
begin
if (Cin > 4'd4)
Cout = Cin + 4'd3;
else
Cout = Cin;
end

endmodule //Add3
```

Listing 2: BCD6 Module Code

```
module BCD6(
input [6:0] B,
output [6:0] Out
);
wire W1, W2, W3;
wire W4, W5, W6;

Add3 C1(.Cin(B[6:3]), .Cout({ Out[6],W3 , W2, W1}));

Add3 C2(.Cin({W3,W2,W1,B[2]}), .Cout({Out[5],W4,W5,W6}));

Add3 C3(.Cin({W4,W5,W6,B[1]}), .Cout({Out[4], Out[3], Out[2], Out[1]}));

assign Out[0] = B[0];

endmodule
```

Listing 3: BCD11 Module Code

```
module BCD11(
input [11:0] in11,
output [15:0] out11
);

wire [6:0] W1;
wire W2, W3, W4;
BCD6 C1( .B({0, in11[8:3]}), .Out(W1)
);

BCD6 C2( .B({W1[3:0], in11[2:0]}), .Out({W2, W3, W4,out11[3:0]})
);
```

```
BCD6 C3( .B({0,W1[6:4], W2, W3, W4}), .Out(out11[10:4])
);

assign out11[15:11] = 0;
endmodule
```

Listing 4: sseg1 Module Code

```
module sseg1(
input [15:0] sw,
output [3:0] an,
output [6:0] seg,
output dp
);

wire [3:0] muxwire;
wire [7:0] W1;

BCD11 BD11(
.in11(sw[10:0]), .out11(W1));

mux2_4b m1(
.in1(W1[7:4]), .in0(W1[3:0]),
.sel(sw[15]), .out(muxwire[3:0])
);

sseg_decoder sseg1(
.sseg(seg), .num(muxwire[3:0])
);

assign an[3:2] = 3;
assign dp = 1;
assign an[1] = ~sw[15];
assign an[0] = sw[15];
endmodule
```