# Lab 3 – Four bit, Seven Segment Decoder

*Name*: Von Kaukeano *Date*: 9/25/18

*Course*: Digital Circuit Design Lab – ECE 2613 *Section #*: 001

## Summary/Abstract

In this lab, we created two modules that convert binary to hexadecimal and output on the given LED display on our board. The other module is to select which of the four displays that we desire. We created truth tables for the logic that would output onto the LED display and to convert hexadecimal to binary. Matching our truth tables, we enter them in our modules and their corresponding test benches and instantiate them to finish.

## Introduction

This lab designs a decoder using the Hamming Algorithm from the previous labthat converts a 4-bit hexadecimal code to a corresponding 7-bit code to create a display. The design implements a display on code which activates the display. The display creates the hexadecimal corresponding to its binary number and can set it at one of the four displays. Background information on this lab would be how binary numbers are converted to hexadecimals and how logic is implemented in applications.

## Procedure

This truth table determines which display the conversion will be shown on. For example, if both switches are off then the number or letter is shown at the first position [0} at the right most LED. The 4-bit Anode code is selected when the selected position is low. (1101 = second most right bit selected)

| **sw[5]** | **sw[4]** | **Digit Displayed** | **4-bit Anode code: AN3-0=anode[3:0]** |
|---|---|---|---|
| 0 | 0 | 0 | 1110 |
| 0 | 1 | 1 | 1101 |
| 1 | 0 | 2 | 1011 |
| 1 | 1 | 3 | 0111 |

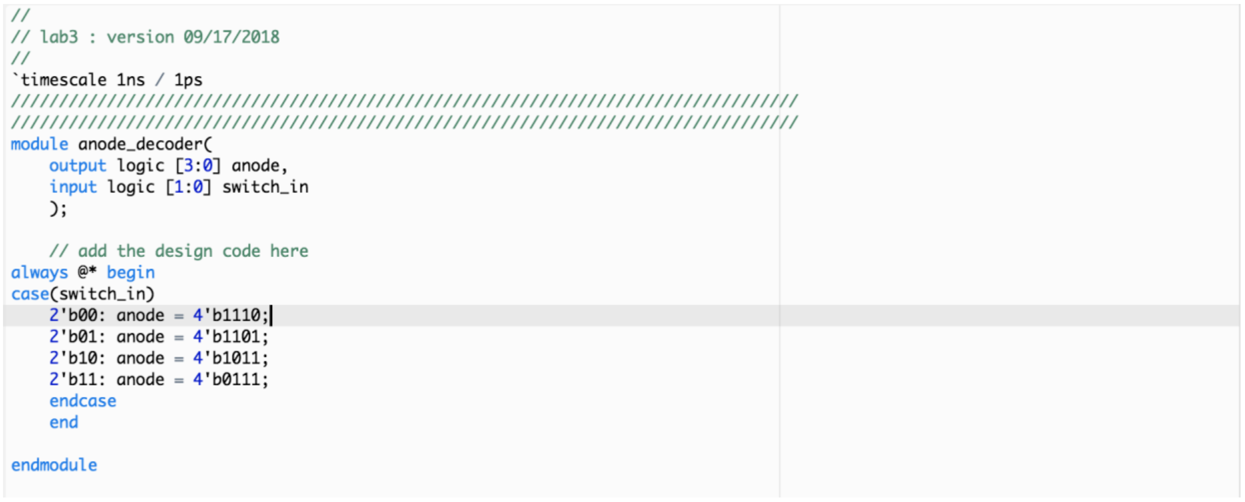
This is the conversion between binary and hexadecimal numbers. The segments that are selected are based off the shape of the numbers and which segments would need to be selected in order to create the number. The number zero, you can see all segments are to be selected except the middle segment because that would create the number 8. The segments are selected when the input is low like the display above. (1000000 = all segments except one are selected.)

| sw[3:0] | Digit Displayed | 7-bit Seg code: gfecdba |
|---|---|---|
| 0000 | 0 | 1000000 |
| 0001 | 1 | 1111001 |
| 0010 | 2 | 0100100 |
| 0011 | 3 | 0110000 |
| 0100 | 4 | 0011001 |
| 0101 | 5 | 0010010 |
| 0110 | 6 | 0000010 |
| 0111 | 7 | 1111000 |
| 1000 | 8 | 0000000 |
| 1001 | 9 | 0010000 |
| 1010 | A | 0100000 |
| 1011 | B | 0000011 |
| 1100 | C | 1000110 |
| 1101 | D | 0100001 |
| 1110 | E | 0000110 |
| 1111 | F | 0001110 |

## Results

I had several mismatches which were occurring because of missing some the lines for the decoder. I also had mismatches because of missing the test benches for the condition of enable = 0. After fixing these issues I had no mismatches.

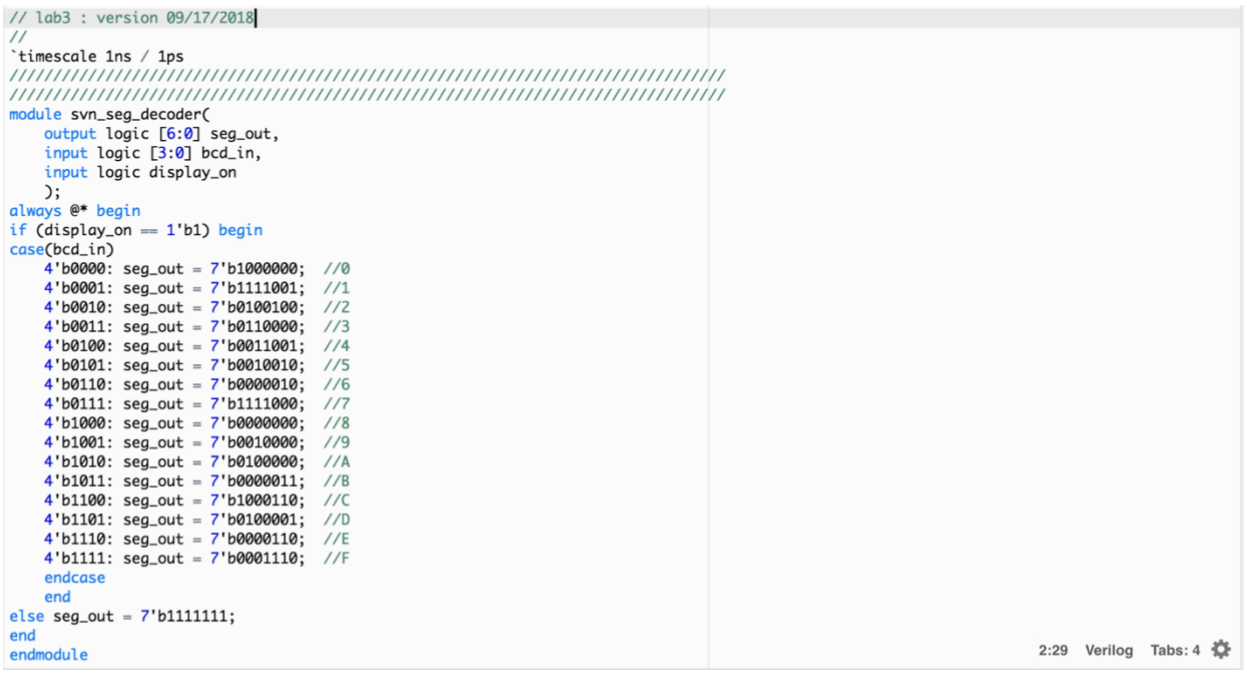
## Design Code



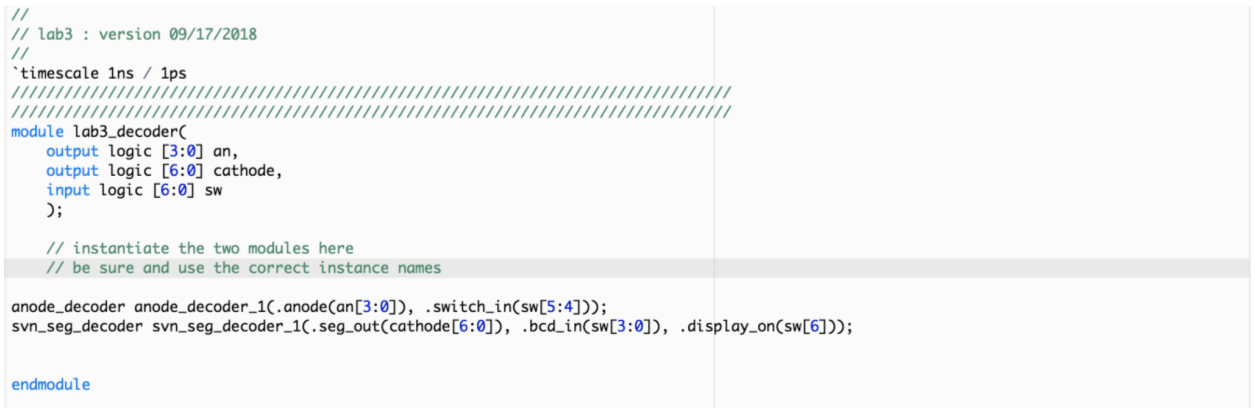
```
//
// lab3 : version 09/17/2018
//
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
module anode_decoder(
    output logic [3:0] anode,
    input logic [1:0] switch_in
    );

    // add the design code here
always @* begin
case(switch_in)
    2'b00: anode = 4'b1110;
    2'b01: anode = 4'b1101;
    2'b10: anode = 4'b1011;
    2'b11: anode = 4'b0111;
    endcase
    end

endmodule
```



```
// lab3 : version 09/17/2018
//
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
module svn_seg_decoder(
    output logic [6:0] seg_out,
    input logic [3:0] bcd_in,
    input logic display_on
    );
always @* begin
if (display_on == 1'b1) begin
case(bcd_in)
    4'b0000: seg_out = 7'b1000000;  //0
    4'b0001: seg_out = 7'b1111001;  //1
    4'b0010: seg_out = 7'b0100100;  //2
    4'b0011: seg_out = 7'b0110000;  //3
    4'b0100: seg_out = 7'b0011001;  //4
    4'b0101: seg_out = 7'b0010010;  //5
    4'b0110: seg_out = 7'b0000010;  //6
    4'b0111: seg_out = 7'b1111000;  //7
    4'b1000: seg_out = 7'b0000000;  //8
    4'b1001: seg_out = 7'b0010000;  //9
    4'b1010: seg_out = 7'b0100000;  //A
    4'b1011: seg_out = 7'b0000011;  //B
    4'b1100: seg_out = 7'b1000110;  //C
    4'b1101: seg_out = 7'b0100001;  //D
    4'b1110: seg_out = 7'b0000110;  //E
    4'b1111: seg_out = 7'b0001110;  //F
    endcase
    end
else seg_out = 7'b1111111;
end
endmodule
```



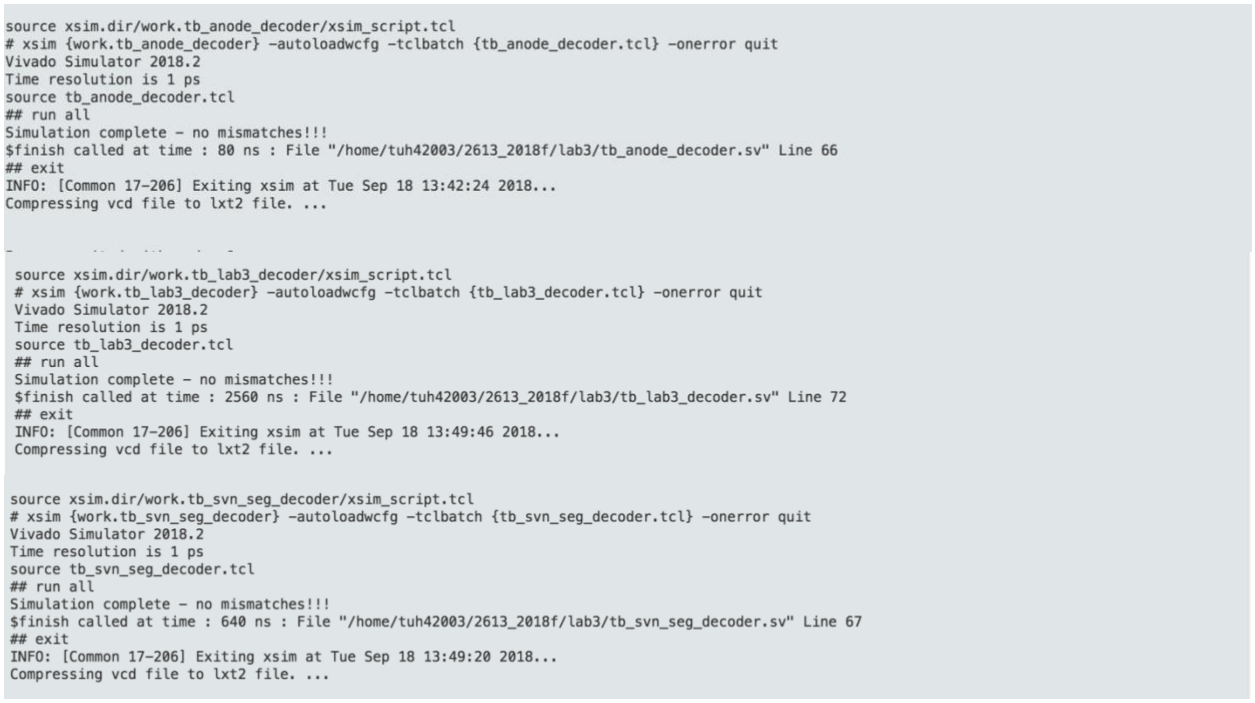
```
//
// lab3 : version 09/17/2018
//
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
module lab3_decoder(
    output logic [3:0] an,
    output logic [6:0] cathode,
    input logic [6:0] sw
    );

    // instantiate the two modules here
    // be sure and use the correct instance names

anode_decoder anode_decoder_1(.anode(an[3:0]), .switch_in(sw[5:4]));
svn_seg_decoder svn_seg_decoder_1(.seg_out(cathode[6:0]), .bcd_in(sw[3:0]), .display_on(sw[6]));


endmodule
```

## Simulation Results



```
source xsim.dir/work.tb_anode_decoder/xsim_script.tcl
# xsim {work.tb_anode_decoder} -autoloadwcfg -tclbatch {tb_anode_decoder.tcl} -onerror quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_anode_decoder.tcl
## run all
Simulation complete - no mismatches!!!
$finish called at time : 80 ns : File "/home/tuh42003/2613_2018f/lab3/tb_anode_decoder.sv" Line 66
## exit
INFO: [Common 17-206] Exiting xsim at Tue Sep 18 13:42:24 2018...
Compressing vcd file to lxt2 file. ...
```

```
source xsim.dir/work.tb_lab3_decoder/xsim_script.tcl
# xsim {work.tb_lab3_decoder} -autoloadwcfg -tclbatch {tb_lab3_decoder.tcl} -onerror quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_lab3_decoder.tcl
## run all
Simulation complete - no mismatches!!!
$finish called at time : 2560 ns : File "/home/tuh42003/2613_2018f/lab3/tb_lab3_decoder.sv" Line 72
## exit
INFO: [Common 17-206] Exiting xsim at Tue Sep 18 13:49:46 2018...
Compressing vcd file to lxt2 file. ...
```

```
source xsim.dir/work.tb_svn_seg_decoder/xsim_script.tcl
# xsim {work.tb_svn_seg_decoder} -autoloadwcfg -tclbatch {tb_svn_seg_decoder.tcl} -onerror quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_svn_seg_decoder.tcl
## run all
Simulation complete - no mismatches!!!
$finish called at time : 640 ns : File "/home/tuh42003/2613_2018f/lab3/tb_svn_seg_decoder.sv" Line 67
## exit
INFO: [Common 17-206] Exiting xsim at Tue Sep 18 13:49:20 2018...
Compressing vcd file to lxt2 file. ...
```

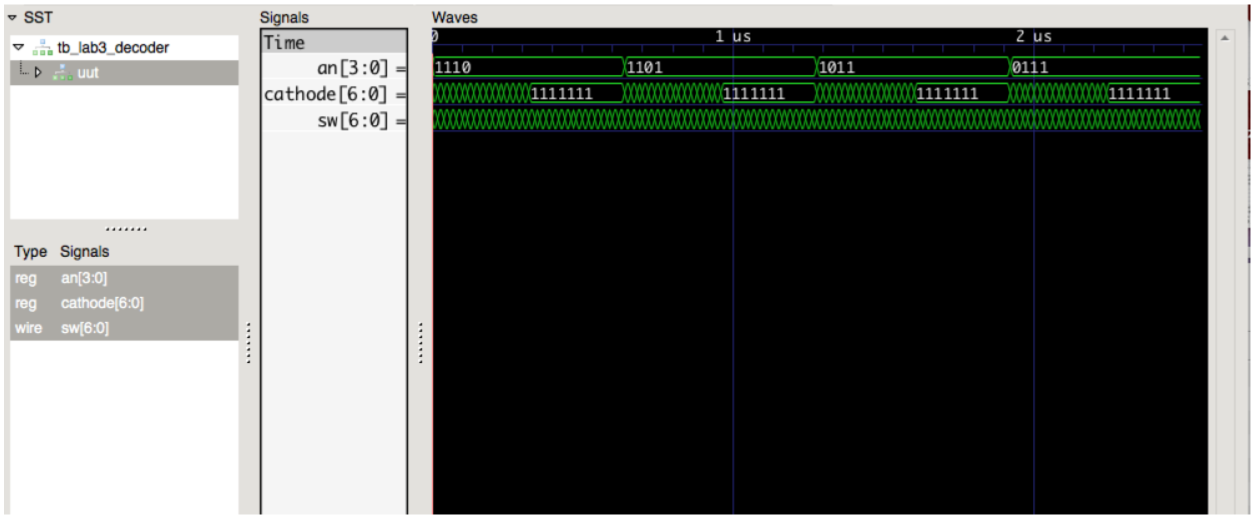
## Hardware Implementation

*My design was demonstrated to Catherine on the afternoon of 9/18 during the lab period.*

## Conclusion

This lab sums most of the past labs up until this point between bits and decoders, instantiation, and even binary to hexadecimal conversions. The lab is a great hand on experience to understand Verilog and the logic implementations. The debugging tool gtkwave is useful for line by line debugging especially if there are more than 32 lines of code necessary.

## Appendix


SST
tb_lab3_decoder
uut
Type Signals
reg an[3:0]
reg cathode[6:0]
wire sw[6:0]
Signals
Time
an[3:0] =
cathode[6:0] =
sw[6:0] =
Waves
0
1 us
2 us
1110
1101
1011
0111
1111111
1111111
1111111
1111111