Lab 4 – Four bit, Error Correction Code (ECC) using Hamming(7,4) Code Decoding

Name: Von Kaukeano *Date*: 10/4/18

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

Summary/Abstract

In this lab, we designed a decoder using the Hamming (7,4) algorithm in reverse. Using the seven bits as the input we receive the original four bits and a one-bit error detection as output. The given truth table determines when there is a bit error. If the input on the board matches the truth table then the output is zero, if it does not match then the output provides the bit that is in error on the seven segment decoder. The LED lights above the switches are a four-bit output that matches the corresponding original input.

Introduction

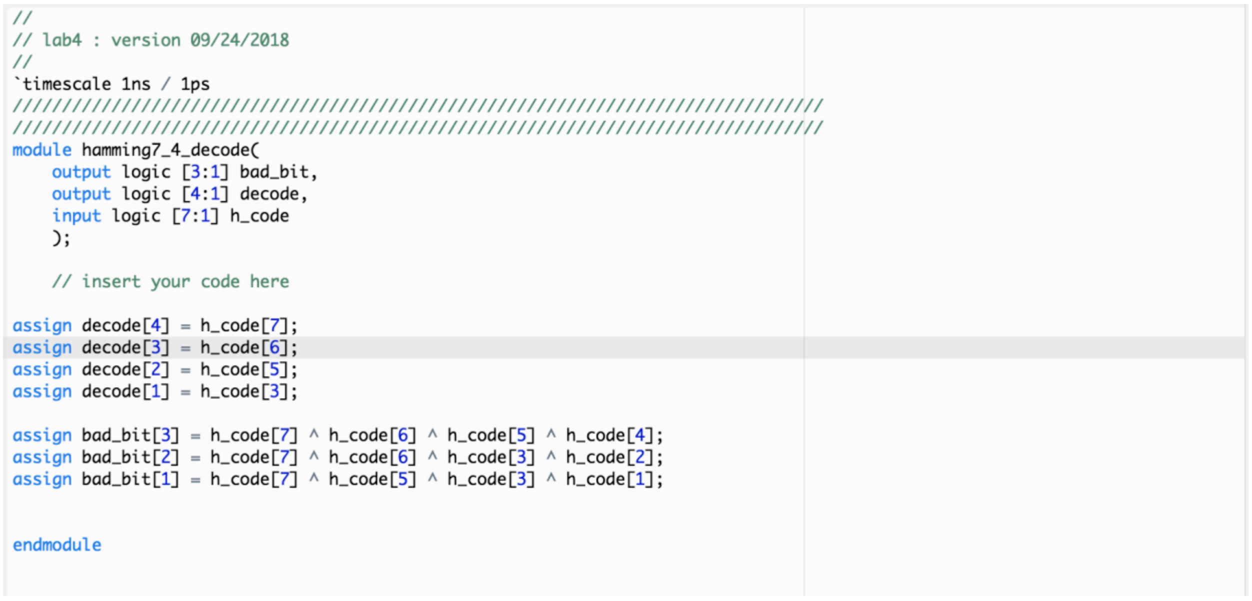
Understanding how the Hamming(7,4) code and parity bits work are an important ground in order to succeed in designing this lab. The algorithm detects and outputs a single bit error based on the truth table that was given that represents the Hamming(7,4) code. The LED lights are also based on the Verilog logic determining the original input signal.

Procedure

Using the code from the Hamming(7,4) algorithm and the given truth table for the seven inputs we received outputs of four bits. The exclusive or logic gates create a parity bit checker that determines the output error bit. Building the Verilog and then instantiating it creates the bit file needed for this lab.

Results

Design Code



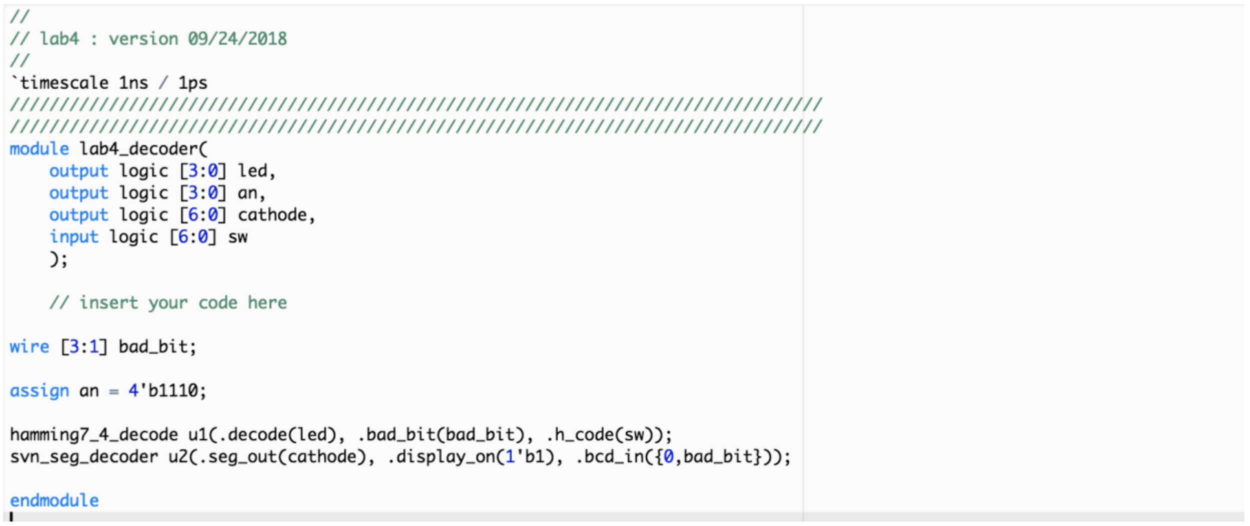
```
//
// lab4 : version 09/24/2018
//
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
module hamming7_4_decode(
    output logic [3:1] bad_bit,
    output logic [4:1] decode,
    input logic [7:1] h_code
    );

    // insert your code here

assign decode[4] = h_code[7];
assign decode[3] = h_code[6];
assign decode[2] = h_code[5];
assign decode[1] = h_code[3];

assign bad_bit[3] = h_code[7] ^ h_code[6] ^ h_code[5] ^ h_code[4];
assign bad_bit[2] = h_code[7] ^ h_code[6] ^ h_code[3] ^ h_code[2];
assign bad_bit[1] = h_code[7] ^ h_code[5] ^ h_code[3] ^ h_code[1];


endmodule
```



```
//
// lab4 : version 09/24/2018
//
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
module lab4_decoder(
    output logic [3:0] led,
    output logic [3:0] an,
    output logic [6:0] cathode,
    input logic [6:0] sw
    );

    // insert your code here

wire [3:1] bad_bit;

assign an = 4'b1110;

hamming7_4_decode u1(.decode(led), .bad_bit(bad_bit), .h_code(sw));
svn_seg_decoder u2(.seg_out(cathode), .display_on(1'b1), .bcd_in({0,bad_bit}));

endmodule
```

Simulation Results

```
# xsim {work.tb_hamming7_4_decode} -autoloadwcfg -tclbatch {tb_hamming7_4_decode.tcl} -onerro
r quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_hamming7_4_decode.tcl
## run all
Simulation complete - no mismatches!!!
$finish called at time : 2560 ns : File "/home/tuh42003/2613_2018f/lab4/tb_hamming7_4_decode.
sv" Line 67
## exit
INFO: [Common 17-206] Exiting xsim at Tue Sep 25 13:13:39 2018...
Compressing vcd file to lxt2 file. ...
```

```
source xsim.dir/work.tb_lab4_decoder/xsim_script.tcl
# xsim {work.tb_lab4_decoder} -autoloadwcfg -tclbatch {tb_lab4_decoder.tcl} -onerror quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_lab4_decoder.tcl
## run all
Simulation complete - no mismatches!!!
$finish called at time : 2560 ns : File "/home/tuh42003/2613_2018f/lab4/tb_lab4_decoder.sv" L
ine 64
## exit
INFO: [Common 17-206] Exiting xsim at Tue Sep 25 13:39:28 2018...
Compressing vcd file to lxt2 file. ...
```

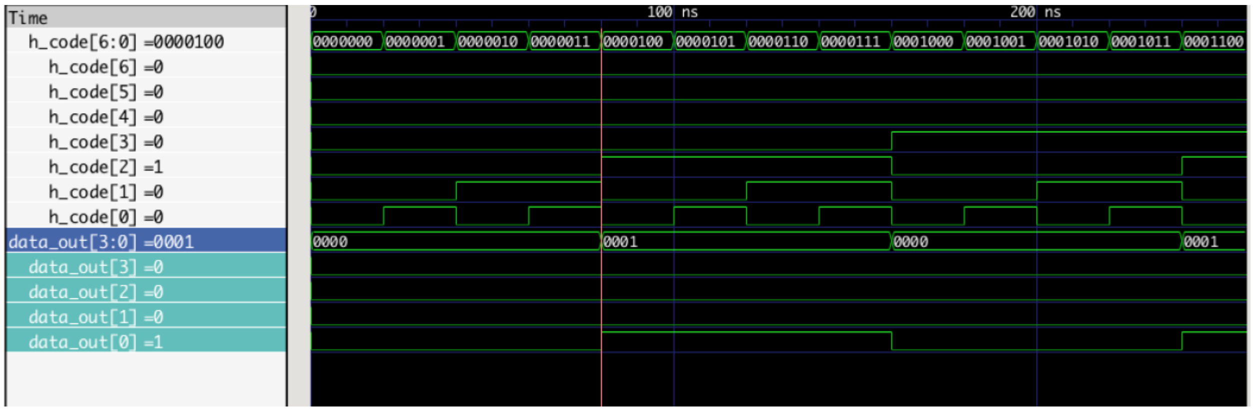
Hardware Implementation

My design was demonstrated to Catherine on the afternoon of 9/25 during the lab period.

Conclusion

This lab has strengthened my knowledge on how to use a decoder algorithm. It also helped me understand the parity bits and how it is able to detect these. This decoder only has the logic to detect only a single bit, and if there were two error bits then the correct error bit will not be displayed.

Appendix


Time
h_code[6:0] =0000100
h_code[6] =0
h_code[5] =0
h_code[4] =0
h_code[3] =0
h_code[2] =1
h_code[1] =0
h_code[0] =0
data_out[3:0] =0001
data_out[3] =0
data_out[2] =0
data_out[1] =0
data_out[0] =1
0
100 ns
200 ns
0000000
0000001
0000010
0000011
0000100
0000101
0000110
0000111
0001000
0001001
0001010
0001011
0001100
0000
0001
0000
0001