Lab 2 –Four bit, Error Correction Code using Hamming(7,4)

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

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

Summary/Abstract

In this lab, we are introduced to the Hamming[7:4] algorithm that detects one and two bit errors. Using the given truth table we assigned seven output bits to four input bits. The encoded milti-bit output signal has three input code bits interleaved with the data bits. The truth table below shows which LED lights should turn on according to what switch is high or low.

Introduction

The background information needed is the knowledge of Verilog programming language that associate with bits and gate primitives along with how the exclusive OR gates determine the output by their logic truth tables. These determine how the Basys3 board responds to the switches being used by turning on the LED lights accordingly. Four input switches will be able to control seven output LED lights.

Procedure

In this lab we were given the logic truth table and the schematic for the code. Using the truth table below we were able to determine the LED light from being on based on the positions of the switches.

d[4:1]	e[7:1]
4'b0000	7'b0000000
4'b0001	7'b0000111
4'b0010	7'b0011001
4'b0011	7'b0011110
4'b0100	7'b0101010
4'b0101	7'b0101101
4'b0110	7'b0110011
4'b0111	7'b0110100
4'b1000	7'b1001011
4'b1001	7'b1001100
4'b1010	7'b1010010
4'b1011	7'b1010101
4'b1100	7'b1100001
4'b1101	7'b1100110
4'b1110	7'b1111000
4'b1111	7'b1111111

Using this truth table, we assigned three input parity bits and a seven output parity bit. They are assigned by the position of the switches using two exclusive OR gates between them. The LED lights at positions one, two, and four are assigned these two exclusive OR gates. The exclusive OR gate has an output logic of one only if one or the other is one but not both and not both 0. The first line of code assigns p1 high if 1, 2, or 4 are high. It runs through the first gate from 1 and 2. If switch 1 or 2 is high but not both p1 is high. If 1 is high and 4 is high then p1 is low. This runs the same for 2 because it goes through two gates. If both 1 and 2 are high or low and 4 is high, then the light at position 1 is high.

Results

It runs through the first gate from 1 and 2. If 1 is high or 2 is high but not both p1 is high. If 1 is high and 4 is high, then p1 turns low. This runs the same for 2 because it goes through two gates. If both 1 and 2 are high or low and 4 is high, then the light at position 1 turns high. I had no mismatches and was able to get the board to communicate with the bit file with one attempt.

Design Code

```
//
// lab2 : version 09/11/2018
//
`timescale 1ns / 1ps
module hamming7_4_encode(
    output logic [7:1] e,
    input logic [4:1] d
    );


    // enter your code here

logic p1, p2, p3;

assign p1 = d[1] ^ d[2] ^ d[4];

assign p2 = d[1] ^ d[3] ^ d[4];

assign p3 = d[2] ^ d[3] ^ d[4];

assign e[7:1] = {d[4], d[3], d[2], p3, d[1], p2, p1};

endmodule
```

6:23 Verilog Tabs: 4

Simulation Results

```
INFO: [Common 17-206] Exiting Webtalk at Fri Sep 14 18:22:22 2018...

****** xsim v2018.2 (64-bit)
  **** SW Build 2258646 on Thu Jun 14 20:02:38 MDT 2018
  **** IP Build 2256618 on Thu Jun 14 22:10:49 MDT 2018
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

source xsim.dir/work.tb_hamming7_4_encode/xsim_script.tcl
# xsim {work.tb_hamming7_4_encode} -autoloadwcfg -tclbatch {tb_hamming7_4_encode.tcl} -onerror quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_hamming7_4_encode.tcl
## run all
Simulation complete - no mismatches!!!
$finish called at time : 320 ns : File "/home/tuh42003/2613_2018f/lab2/tb_hamming7_4_encode.sv" Line 65
## exit
INFO: [Common 17-206] Exiting xsim at Fri Sep 14 18:22:33 2018...


Process exited with code: 0
```

Hardware Implementation

My design was demonstrated to Catherine on the afternoon of 9/11 during the lab period.

Conclusion

Four parity bits control seven parity bits. Within the logic of the Verilog code, the position of the switches determines the output of seven LED lights to turn them low and high. This algorithm is used to detect all single and two bit errors and identify to correct a single bit error. The next lab will expand on this topic and show that our logic is correct.