Contents

1 Error Detection For More Than 1-Bit **2**
1.0.1 STEPS:- 2

2 Problem (1-bit ECC test setup) **4**
2.1 Problem Statement 4
2.2 Introduction 5
2.2.1 BCD to 7-segment display 5
2.2.2 3-bit Decoder 5
2.2.3 3-bit ECC parity generator 5
2.2.4 1-bit ECC 5
2.2.5 1-bit ECC test setup 5
2.3 Truth Table 5
2.3.1 BCD to 7-Segment Display 5
2.3.2 3-bit decoder 6
2.3.3 3-bit ECC parity generator 7
2.4 Logical Expression 7
2.4.1 BCD to 7-segment display 7
2.4.2 3-bit decoder 8
2.4.3 3-bit ECC parity generator 8
2.5 Digital Logical Circuits 8
2.5.1 Part 1 8
2.5.2 Part 2 9
2.5.3 Part 3 9
2.5.4 Part 4 10
2.5.5 Part 5 11

Chapter 1

Error Detection For More Than 1-Bit

Single bit Hamming Code can be extended to 2 bit error detection (but not correction) by adding a single parity bit check over the entire code. For an 11 bit message we can use 4bits for finding 1 bit errors and the 16th bit for the 2 bit errors.
This can be done as follows:

1.0.1 STEPS:-

1. Consider a eleven bit message in a sixteen bit, let the bits at positions 1,2,4,8

$$(c_1, c_2, c_4, c_8) \tag{1.1}$$

and the bit in position 0 (c_0) be used for parity checks.
2. Let c[i] be the state of the i^{th} bit.
3. Positions given below:

$$i = 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15 \tag{1.2}$$

may be used to transmit information.
4. The bit at

$$c_{2^i} i = 0, 1, 2, 3 \tag{1.3}$$

will store the XOR of the values of the positions whose $(i+1)^{th}$ bit is 1.
5. For example c_1 will store the $\oplus$ of the values of the positions whose bit is turned on, that is:

$$c[1] = c[3] \oplus c[5] \oplus c[7] \oplus c[9] \oplus c[11] \oplus c[13] \oplus c[15] \tag{1.4}$$

6. Similarly,

$$c[2] = c[3] \oplus c[6] \oplus c[7] \oplus c[10] \oplus c[11] \oplus c[14] \oplus c[15] \tag{1.5}$$

$$c[4] = c[5] \oplus c[6] \oplus c[7] \oplus c[12] \oplus c[13] \oplus c[14] \oplus c[15] \tag{1.6}$$

$$c[8] = c[9] \oplus c[10] \oplus c[11] \oplus c[12] \oplus c[13] \oplus a[14] \oplus a[15] \tag{1.7}$$

7. Now $a[2^i]$ stores parity bits of the relevant positions, so if there is a 1-bit error then we can extract the exact position of the flipped bit using the four parity bits transmitted.
8. Observe that this can be done by taking:

$$\lambda = \oplus_{i=1}^{15}(i * \dot{a}[i]) \tag{1.8}$$

where a[i] is the received bit .This quantity is the XOR of the positions which are turned on in the received message.This would be 0 if there are no errors, if this is $\neq 0$ then that is the position of the flipped bit.
9. To extend this scheme to detecting (but not correcting) 2-bit errors, we can utilise the bit at c_0.
10. By setting:

$$a[0] = \oplus_{i=1}^{15}(i * \dot{a}[i]) \tag{1.9}$$

(total parity of the original message now becomes 0). We can detect errors by comparing

$$\oplus_{i=1}^{15}(i * \dot{a}[i]) \tag{1.10}$$

(parity of received bits) to λ (same as above):

1. Now $\oplus_{i=1}^{15}(i * \dot{c}[i]) = 1$ then we have a 1-bit error, whose position can be determined using λ

2. If$\oplus_{i=1}^{15}(i * \dot{c}[i]) = 0$,

   (a) $\lambda \neq 0$ this means that we have a 2 bit error (parity continues to be correct but there is some change)

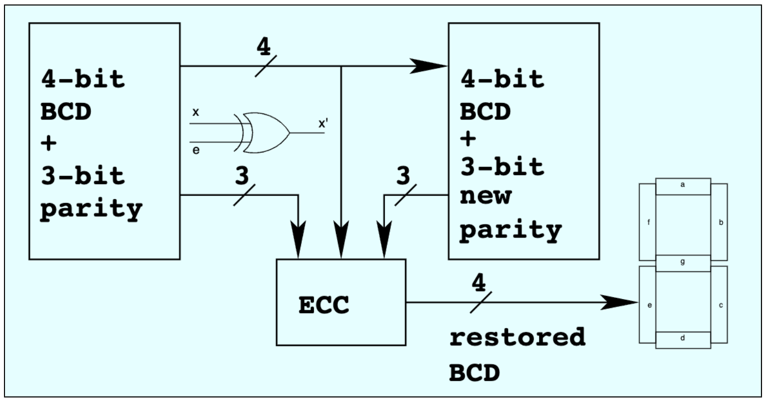
   (b) $\lambda = 0$, this means that we have no error.

Chapter 2

Problem (1-bit ECC test setup)

2.1 Problem Statement

•Wire up the designed components as indicated in the figure with the provision to introduce a single bit error in any of the seven lines using the EXOR gates in conjunction with the 3-bit decoder, displaying the restored data using the 7-segment display
•Test that it works by applying appropriate inputs and checking the outputs
•Label the terminals to reflect their roles
•Save it as a regular circuit (logic file), reopen and retest
•How would you extend this to allow error detection for more than 1-bit (without correction)? You need not realise this in the tool, only report the paper design in your PDF report



2.2 Introduction

We have divided our experiment into 5 parts:-(namely)

2.2.1 BCD to 7-segment display

For the BCD number $\langle x_3, x_2, x_1, x_0 \rangle$ form the circuits to light up the LEDs $a..g$ of the 7-segment display to indicate the decimal number properly

2.2.2 3-bit Decoder

A 3bit-8 Line Decoder which takes in 3-bit binary representation and converts into its corresponding decimal equivalent.

2.2.3 3-bit ECC parity generator

For the BCD number given as $\langle x_3, x_2, x_1, x_0 \rangle$ form the truth tables and the corresponding circuits for the three parity bits required for 1-bit Hamming ECC

2.2.4 1-bit ECC

Design a ECC module that takes the 4-bit received data $\langle x_3, x_2, x_1, x_0 \rangle$, 3-bit received parity $\langle p_2, p_1, p_0 \rangle$ and 3-bit recomputed parity $\langle p'_2, p'_1, p'_0 \rangle$ to generate the 4-bit restored data.

2.2.5 1-bit ECC test setup

Wire up the designed components as indicated in the figure with the provision to introduce a single bit error in any of the seven lines using the EXOR gates in conjunction with the 3-bit decoder, displaying the restored data using the 7-segment display

2.3 Truth Table

2.3.1 BCD to 7-Segment Display

Decimal	**BCD**				**7 Segment LED Display**						
	A	**B**	**C**	**D**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	1	1	0	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Table 2.1: **BCD to 7 segment display**

2.3.2 3-bit decoder

Input			**Output**							
A	**B**	**C**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Table 2.2: **3 bit decoder**

2.3.3 3-bit ECC parity generator

Decimal	**4 BIT BINARY**				**3-bit ECC parity**		
	X1	**X2**	**X3**	**X4**	**P1**	**P2**	**P3**
0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1
2	0	0	1	0	0	1	1
3	0	0	1	1	1	0	0
4	0	1	0	0	1	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	1	1	0
7	0	1	1	1	0	0	1
8	1	0	0	0	1	1	0
9	1	0	0	1	0	0	1
10	1	0	1	0	1	0	1
11	1	0	1	1	0	1	0
12	1	1	0	0	0	1	1
13	1	1	0	1	1	0	0
14	1	1	1	0	0	0	0
15	1	1	1	1	1	1	1

Table 2.3: **3-bit ECC Parity Generator**

2.4 Logical Expression

2.4.1 BCD to 7-segment display

The input to the Circuit is a 4-Bit BCD. Let's Assume the digits as A,B,C and D. We assume the 7-Segment Display to Represent all digits a,b,c,d,e,f and g. Now using **Karnaugh map (K-MAP)** for simplifying the Circuits.

$$a = A + C + BD + \bar{B}\bar{D} \tag{2.1}$$

$$b = \bar{B} + \bar{C}\bar{D} + CD \tag{2.2}$$

$$c = B + \bar{C} + D \tag{2.3}$$

$$d = A + \bar{B}\bar{D} + \bar{B}C + C\bar{D} + B\bar{C}D \tag{2.4}$$

$$e = \bar{B}\bar{D} + C\bar{D} \tag{2.5}$$

$$f = A + B\bar{C} + B\bar{D} + \bar{C}\bar{D} \tag{2.6}$$

$$g = A + B\bar{C} + \bar{B}C + C\bar{D} \tag{2.7}$$

2.4.2 3-bit decoder

$$a = \bar{A}\bar{B}\bar{C} \tag{2.8}$$

$$b = \bar{A}\bar{B}C \tag{2.9}$$

$$c = \bar{A}B\bar{C} \tag{2.10}$$

$$d = \bar{A}BC \tag{2.11}$$

$$e = A\bar{B}\bar{C} \tag{2.12}$$

$$f = A\bar{B}C \tag{2.13}$$

$$g = AB\bar{C} \tag{2.14}$$

$$h = ABC \tag{2.15}$$

2.4.3 3-bit ECC parity generator

The Logical Expression for P_1, P_2 and P_3 are:

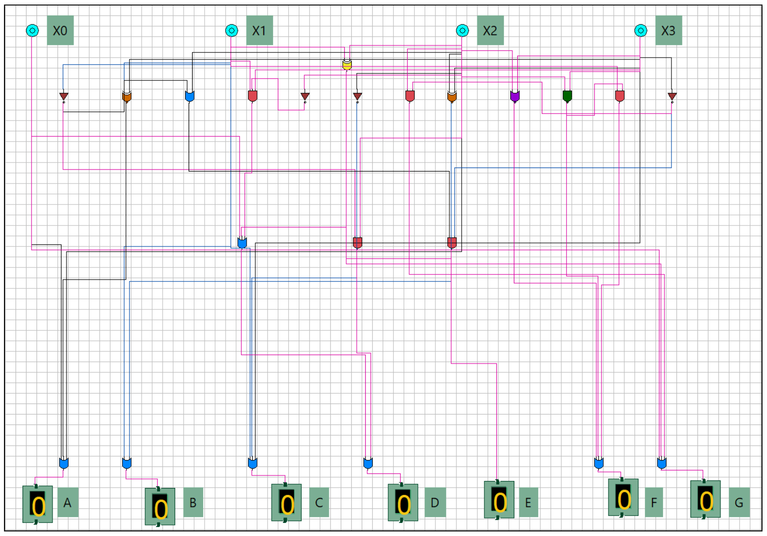
$$p_1 = x_1 \oplus x_2 \oplus x_4 \tag{2.16}$$

$$p_2 = x_1 \oplus x_3 \oplus x_4 \tag{2.17}$$

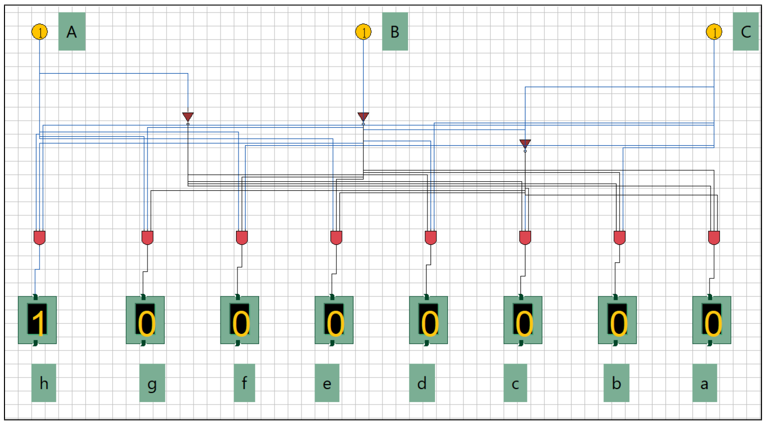
$$p_3 = x_2 \oplus x_3 \oplus x_4 \tag{2.18}$$

2.5 Digital Logical Circuits

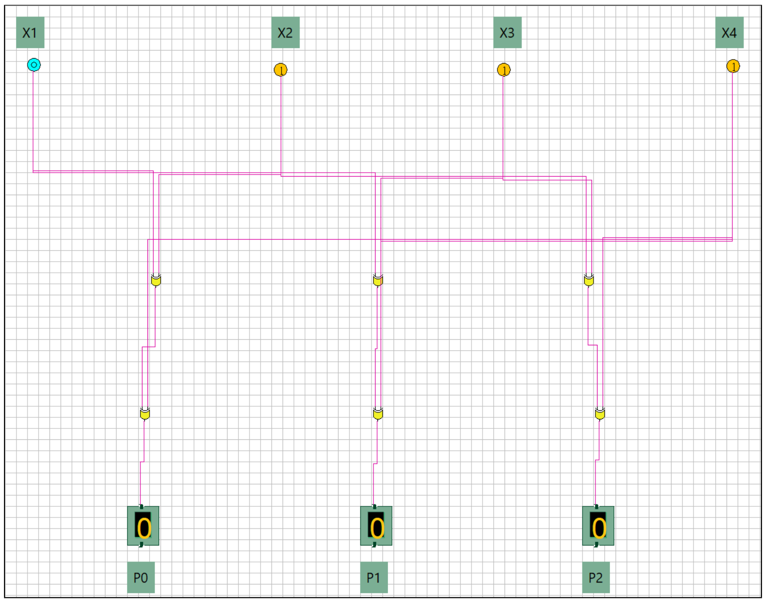
2.5.1 Part 1



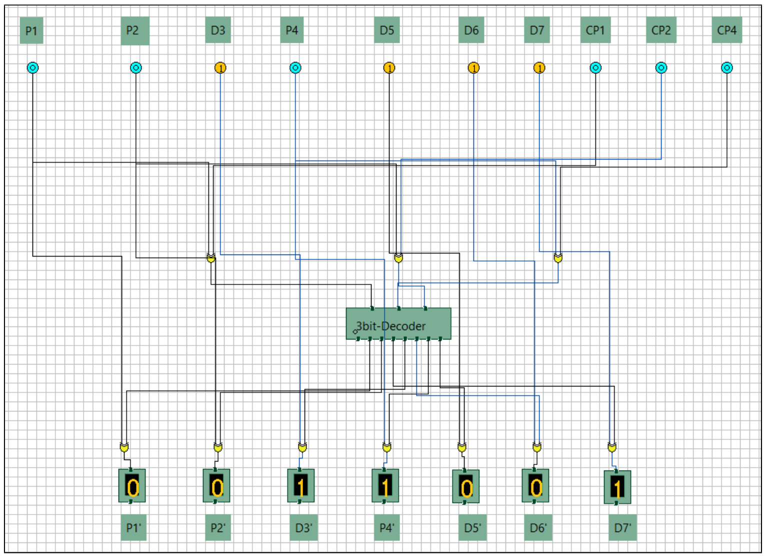
2.5.2 Part 2



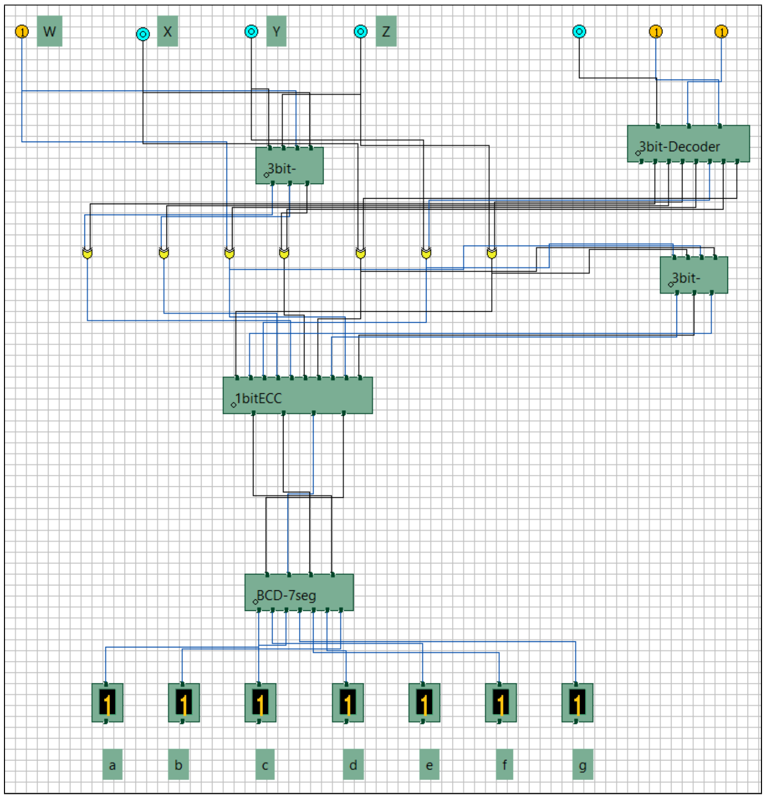
2.5.3 Part 3



2.5.4 Part 4


P1
P2
D3
P4
D5
D6
D7
CP1
CP2
CP4
3bit-Decoder
0
0
1
1
0
0
1
P1'
P2'
D3'
P4'
D5'
D6'
D7'

2.5.5 Part 5


W
X
Y
Z
3bit-Decoder
3bit-
3bit-
1bitECC
BCD-7seg
a
b
c
d
e
f
g