EXPERIMENT O3
SINGLE BIT ECC WITH DISPLAY

SINGLE BIT ECC FOR 4-BIT BINARY STRINGS

In general, a code is said to be *error-correcting* if the correct code word can always be deduced from the erroneous word. If the minimum distance of a code is three, then any single error changes a valid code word into an invalid one, which is distance one away from the original code word and distance two from any other valid code word. Therefore, in a code with minimum distance three, any single error is correctable or any double error detectable.

Consider the 4-bit binary string $b_4b_3b_2b_1$. Let us define three parity bits p_3, p_2 and p_1 in the following way.

$$p_3 = b_4 \oplus b_3 \oplus b_2$$

$$p_2 = b_4 \oplus b_3 \oplus b_1$$

$$p_1 = b_4 \oplus b_2 \oplus b_1$$

$b_4b_3b_2b_1$ are the *data bits* or the *information bits*, that contain the actual information that needs to be transmitted. p_3, p_2 and p_1 are the *check bits* or the *parity bits*, that are characteristic to every 4-bit binary string.

Consider the following *Boolean* table. In this table a cell (i, j) contains 1 if the value of j depends on i and 0 otherwise.

	p_3	p_2	p_1
b_4	1	1	1
b_3	1	1	0
b_2	1	0	1
b_1	0	1	1
p_3	1	0	0
p_2	0	1	0
p_1	0	0	1

Now treat every 0-1 pattern corresponding to each of the 7 bits (4 information and 3 parity pits) as a binary string and convert them to their decimal equivalent. The decimal value of a 3-bit binary string ranges from 0 to 7. Each of these

decimal values is known as the *position number* associated to the corresponding bit.

	p_3	p_2	p_1	
b_4	1	1	1	7
b_3	1	1	0	6
b_2	1	0	1	5
b_1	0	1	1	3
p_3	1	0	0	4
p_2	0	1	0	2
p_1	0	0	1	1

	p_3	p_2	p_1	
b_4	1	1	1	7
b_3	1	1	0	6
b_2	1	0	1	5
p_3	1	0	0	4
b_1	0	1	1	3
p_2	0	1	0	2
p_1	0	0	1	1

The 7 bits are transmitted as a single packet and they are assembled according to their position numbers.

7	6	5	4	3	2	1
b_4	b_3	b_2	p_3	b_1	p_2	p_1

Let the *received* 7-bit binary string be $b_4'b_3'b_2'p_3b_1'p_2p_1$; and $b_4'b_3'b_2'p_3'b_1'p_2'p_1'$ be another 7-bit binary string with the same data bits but *recomputed* parity bits. Let us first understand the model when there are errors in at most one bit. Consider another 3-bit binary string $e_3e_2e_1$ whose bits are defined in the following way.

$$e_i = p_i \oplus p_i' \qquad for\ \ i \in \{1,2,3\}$$

In other words, e_i is 1 if and only if there is an error in the ith parity bit p_i and 0 otherwise. *The decimal equivalent of $e_3e_2e_1$ ranges from 0 to 7, which gives the position number of the bit that has got corrupted during transmission.*

In case the decimal value of $e_3e_2e_1$ is 0, no error has occurred and all bits have stayed intact over transmission. Besides, if the decimal value of $e_3e_2e_1$ is 1, 2 or 4, the error has occurred only in the parity bits and the information bits have stayed intact.

MINIMUM HAMMING DISTANCE

For a single bit error correction, the binary string $b_4b_3b_2p_3b_1p_2p_1$ is transmitted as a single packet that contains both the data and the parity bits. Note that all the 7-bits in the string are not independently chosen. If that was the case, the

minimum *Hamming* distance would have been 1, and no error could have been detected, let alone corrected. The data bits $b_4b_3b_2b_1$ however are independently chosen and the parity bits are strategically chosen based on them so that the *Hamming* distance between any two codewords is at least 3, so that the single-bit error can be corrected. Let us check the minimum *Hamming* distance for the single bit error correcting code.

- Consider the information string $b_4b_3b_2b_1$ and all other 4-bit binary strings at a *Hamming* distance of 1. There are 4 such possible strings, each one formed by flipping any one of the bits in $b_4b_3b_2b_1$.

Bit Flipped	Total number of bits flipped in the codeword
b_1	Each of these bits determine exactly 2 parity bits. When simply one of these is flipped and others are kept intact, exactly 2 parity bits will be flipped. Therefore, overall, in the codeword exactly 3 bits are flipped.
b_2	
b_3	
b_4	This bit determines all the three parity bits. When this bit is flipped while others are kept intact, all 3 parity bits will get flipped. Therefore, overall, in the codeword exactly 4 bits are flipped.

- Now consider all 4-bit binary strings at a *Hamming* distance of 2 from $b_4b_3b_2b_1$. There are 6 such possible strings, each one formed by flipping any two of the bits in $b_4b_3b_2b_1$.

Bits Flipped	Total number of bits flipped in codeword
b_2b_1	In each of the pairs, both the bits determine exactly two parity bits. Besides, there is a parity bit that is determined by both of them. Eg- b_2 determines p_3 and p_1 while b_1 determines p_2 and p_1. Therefore, flipping both the bits in any of these pairs, flips exactly 2 parity bits. Therefore, overall, in the codeword exactly 4 bits are flipped.
b_3b_1	
b_3b_2	
b_4b_1	In each of the pairs, b_4 is present which determines all 3 parity bits. Flipping b_4 will flip all 3 parity bits and then flipping another information bit among b_3, b_2, b_1 will restore two of them. Hence, exactly one parity bit is flipped and overall in the codeword exactly 3 bits are flipped.
b_4b_2	
b_4b_3	

- Now consider all 4-bit binary strings at a *Hamming* distance of 3 or 4 from $b_4b_3b_2b_1$. Each one of these is formed by flipping at least three bits in $b_4b_3b_2b_1$. Either way, at least three bits have been flipped in the data bits

itself. Therefore, the total number of bits flipped in the codeword cannot be less than 3.

This implies that in this code, the minimum *Hamming* distance is 3. Therefore, it is not only possible to correct single bit errors but it should also be possible to detect double bit errors. Is there any way to increase the minimum *Hamming* distance of this code from 3 to 4, so that both of them can be done simultaneously?

HOW TO INCREASE THE MINIMUM HAMMING DISTANCE ?

The minimum *Hamming* distance of the code can be increased by adding another parity bit p_4 at the prefix of all the codewords in the previous code. This new parity bit is defined as follows.

$$p_4 = b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus p_3 \oplus p_2 \oplus p_1$$

VERIFYING THE "NEW" MINIMUM HAMMING DISTANCE

Revisit the section where the calculations of the minimum *Hamming* distance for a *single bit error correcting code* are discussed. The minimum *Hamming* distance came out to be 3. All we need to check is the "improvement" in the case when the *Hamming* distance between the former 7-bit codewords is 3; because when it is 4 or more, the purpose is already solved.

The eighth bit p_4 added in the new set of codewords is an XOR operation of the other 7-bits. If the Hamming distance between two 7-bit codewords is 3, it means that three bits have to be flipped in going from one to another. Flipping an odd number of bits in a chained XOR operation, flips the result of the operation too. Therefore, the eighth bit p_4 will also be flipped during this transformation and hence in the new system, *the minimum Hamming distance has indeed increased to 4.*

ACHIEVEMENTS OF THE "NEW" HAMMING CODE.

CASE I $\boldsymbol{p_4 \oplus p_4' = 1}$

The detection and hence the correction of single bit errors in the new *Hamming* code will remain the same. If there is a single bit error, the error in the most significant parity bit p_4 will detect it. Recall that p_4 was defined in the following way.

$$p_4 = b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus p_3 \oplus p_2 \oplus p_1$$

If the single bit error corrupts the parity bit p_4 itself, then definitely the value of $p_4 \oplus p_4'$ will be 1 (both must be unequal). Otherwise, if the single bit error has corrupted one of the other parity bits or the information bits, then also the value of p_4 will be flipped (by the above formula), hence making $p_4 \oplus p_4'$ = 1 again. ***This implies that $\boldsymbol{p_4 \oplus p_4' = 1}$ if and only if a single bit error occurs.***

Now once the binary string/packet is received by the receiver and the four parity bits p_4', p_3', p_2', p_1' are recomputed; provided the value of $p_4 \oplus p_4'$ comes out to be 1, the most significant bit p_4/p_4' can be dropped and the same procedure can be reused with the rest of the 7 bits as is discussed for the *single bit error correction*. Therefore, after the value of $p_4 \oplus p_4'$ establishes that only a single bit error has occurred, each and every step of the single bit error correction remains intact and the original binary code can be restored. This is how the new *Hamming* code supports detection and correction of single bit errors.

CASE II $\boldsymbol{p_4 \oplus p_4' = 0}$

If the value of $p_4 \oplus p_4'$ (the XOR operation of the received and the recomputed most significant parity bit) is zero, there must have had occurred an error in either zero or two bits. How can this be proven?

For the trivial case in which no errors occur, it is easy to comment that the fourth parity bit will remain unaltered. For the case of double bit errors, however, the following two situations can occur.

SITUATION I : Error in $\boldsymbol{p_4}$ and in one of the other 7 bits

If there is an error in the fourth parity bit (received) p_4, then-

$$p_4 = \backsim p_{t4} = \backsim (b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus p_{t3} \oplus p_{t2} \oplus p_{t1})$$

Now if there is another error in any one of the other 7 bits (say b_2) then the recomputed parity bit p_4' will look like-

$$p_4' = (b_4 \oplus b_3 \oplus \sim b_2 \oplus b_1 \oplus \sim p_{t3} \oplus p_{t2} \oplus \sim p_{t1})$$
$$= \backsim (b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus p_{t3} \oplus p_{t2} \oplus p_{t1})$$

Clearly $p_4 \oplus p_4'$ is zero because both are equal.

SITUATION II : Error in two of the other 7 bits

p_4 is a XOR operation of 7 bits. If exactly two of these 7 bits are flipped, obviously there XOR operation will stay intact (this is in fact valid for any even number of flips). Therefore, in this situation also, $p_4 = p_4'$ and hence, $p_4 \oplus p_4'$ is zero.

Finally, it can be established that in the new Hamming code, $p_4 \oplus p_4'$ will be zero if and only if there are errors in either zero or two bits.

But it is important to differentiate between zero errors and double bit errors; otherwise, an erroneous code can be mixed with a valid code and vice-versa. This is actually possible. When two errors occur; then the overall parity check is satisfied but the position number (determined as before from the other seven bits) will indicate an error. In case of no errors, the string $e_3e_2e_1$ will simply be 000 indicating no error.

Therefore, ***if $p_4 \oplus p_4'$ is zero and at least one of the three bits in $e_3e_2e_1$ is 1, a double bit error has occurred.***

If $p_4 \oplus p_4'$ is zero and all of the three bits in $e_3e_2e_1$ are 0, no error has occurred.

This is how the *Hamming* code can be successful in detecting errors in more than 1 bit. Note that in case of double bit errors, the position of the bits that are corrupted cannot be computed. In other words, one can conclude that the string is erroneous with two errors but cannot restore the original string without any errors.

IMPLEMENTATION / EXTENDING THE ORIGINAL CIRCUIT

Now all we have to do is assemble all the results in form of a *control flow* to choose different paths at different stages of computations.

STAGE O1 - TRANSMISSION

The transmitter transmits an 8-bit binary vector ($p_{t4}b_4b_3b_2p_{t3}b_1p_{t2}p_{t1}$). The most significant bit is the fourth parity bit p_{t4} and the other 7 bits are the ones discussed in the first section (4 information bits and 3 other parity bits).

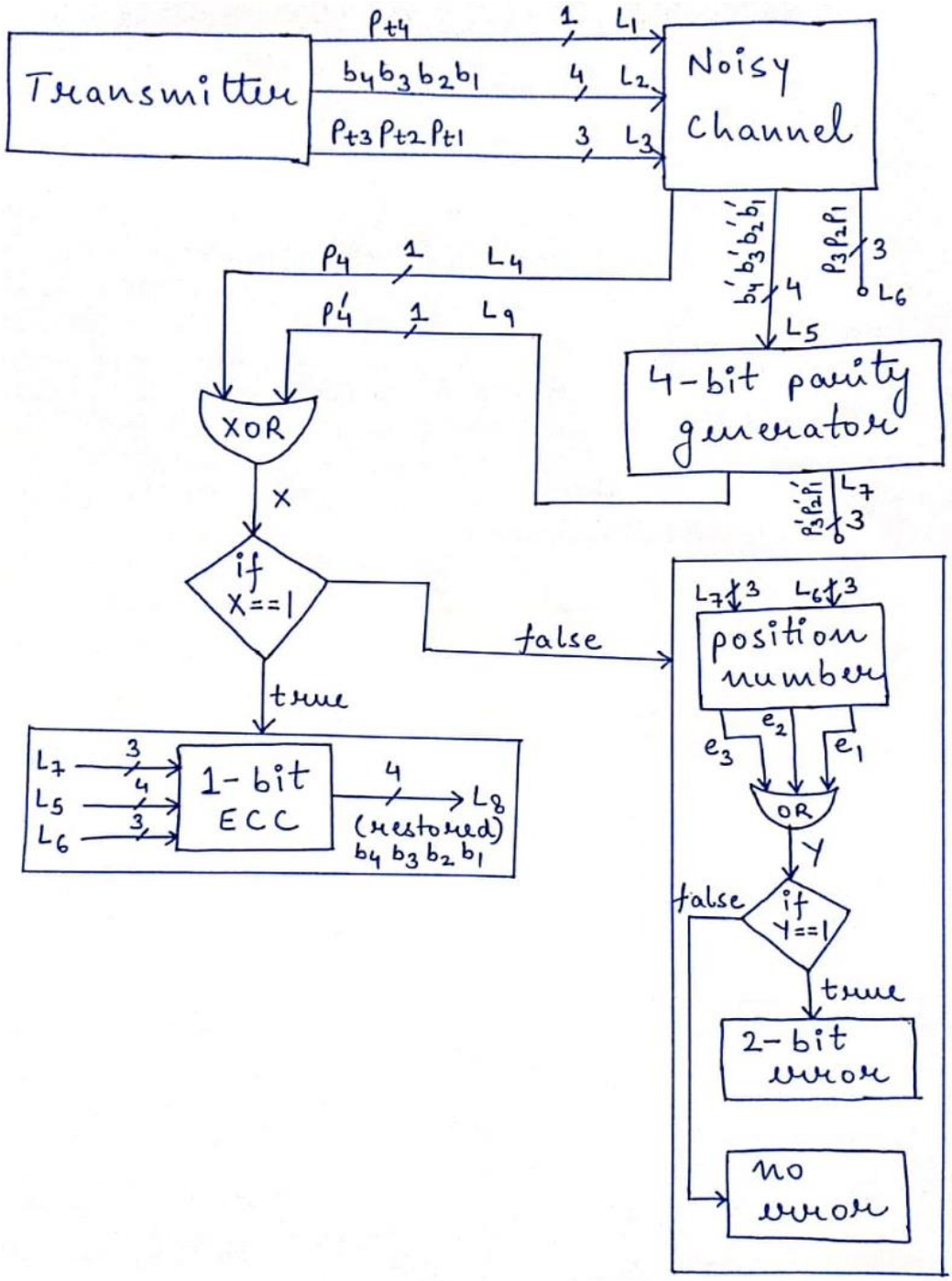
STAGE O2 – NOISE INTRODUCTION

The noisy channel introduces some errors in the original codeword. In this model we have dealt with errors in at most two bits. The noisy channel transforms the

8-bit binary vector $p_{t4}b_4b_3b_2p_{t3}b_1p_{t2}p_{t1}$ to $p_4b_4'b_3'b_2'p_3b_1'p_2p_1$. The latter is the one that reaches the receiver.

STAGE O3 – RECOMPUTATION OF PARITY BITS

The 4 information bits $b_4'b_3'b_2'b_1'$ are extracted from the received codeword and the parity bits associated with this codeword are recomputed (using the formulae as discussed in the first and the third sections). Let the recomputed parity bits be p_4', p_3', p_2', p_1'.



STAGE O4 – CHECKING THE VALUE OF $p_4 \oplus p_4'$

The received parity bit p_4 and the recomputed parity bit p_4' are checked for equality (using XOR operation). ***IF*** the two bits are unequal ($\mathrm{p}_4 \oplus \mathrm{p}_4' = 1$)-

THEN –

It is confirmed that the codeword has a single bit error. The 1-bit ECC component can be reused to correct that fault and restore the original information bits.

ELSE –

It is confirmed that the codeword has an even number of errors (in either zero or two bits). The 3-bit binary representation of the position number ($e_3e_2e_1$) is computed using the received parity bits $p_3p_2p_1$ and the recomputed parity bits $p_3'p_2'p_1'$ as discussed in the first section. ***IF*** decimal value of $e_3e_2e_1$ is non-zero (OR operation of the three bits is 1)-

THEN –

It is confirmed that the codeword has an error in two bits. The exact position of the erroneous bits can not be deduced, but the double-bit errors are anyway detectable.

ELSE –

It is confirmed that the codeword has no errors.

GROUP MEMBERS

Group ID	Roll No.	Name
09	19CS10011	Anish Sofat
09	19CS10031	Abhishek Gandhi
09	19CS10044	Nakul Aggarwal
09	19CS10051	Sajal Chhamunya
09	19CS10053	Satvik Bansal