POLITECNICO DI TORINO

Department of Electronics and Telecommunication (DET)

Master Degree Thesis

The Implementation and Verification of Hamming code

Computer and Communication Networks Engineering

Candidate: Majid Kashkouli

Supervisors: Prof. Roberto Garello, Jeroen Boydens

October 2018

Abstract:

Nowadays, the communication systems like telecom communication and etc. already become inevitable in people's life. For providing effective and reliable communication, the error control method must be applied in each communication system.

In the process of each communication system, the transmitted data might be corrupted and in this situation, the signals which is received in the receiver side are different than the initial signal or data which generated by the sender (transmitter) [1].

This master thesis project offered that how the error will be detected on the receiver side and also how does receiver correct this detected error by a method called hamming code. So in this report, there is illustrate with an example that how the receiver can find in which bit an error is occurred and how the detected error will be corrected?

FEC (Forward Error Correction) can increase the ability to receive places to correct an error during a transmission, so the throughput of a data link operation will be improved in a noisy environment [2].

To indicate how the error correction works, the additional information must append to the data bits in a form of parity bits, but with increasing the length of a frame, the transmission process will slow down.

In hamming code method, a block parity mechanism is provided for FEC (forward error correction) which can be implemented, cheaply. In this method, two errors can be detected, while just one error can be corrected in each received codeword. In Hamming codes, a special principle is used called parity principle to correct just one error and find two errors, but the hamming code method is not able to do both error detection and correction at the same time. One can use hamming codes as an error detector to obtain both single and double bit errors or use them to correct single bit error. It will be done by applying more than one parity bits, and each of them will be computed on different and various combination of bits in the data.

Keywords: Hamming code, Resilience, Fault tolerance.

Acknowledgements

Performing my graduation project and writing this thesis has been a long journey. I would like to thank several people for their help and their support during this process.

I would like to express my sincere thanks to my dear supervisor, Professor Roberto Garello at Politecnico di Torino for his continuous help, scientific guidance and valuable support throughout the project. Moreover, I also owe a great deal of appreciation to my entire family, especially my dear father and mother who always support me in my life, and most especially to my wonderful friend Mrs. Alieh Lotfinejad.

Contents

List of figures

List of tables

List of symbols

Symbol	Description
n	Number of total bits
k	Size of the data bits
r	Size of the parity bits
k/n	Rate
n/k	Overhead factor
G	Generator matrix
H	Check matrix
P	Parity matrix
$\mathbf{P^T}$	Transposed of parity matrix
$\mathbf{H^T}$	Transposed of check matrix
GF	Galois Fields technique
S (Si)	syndrome
$\mathbf{X_i}$	System input
$\mathbf{Y_i}$	System output
XOR	Exclusive OR
<<	Shift the bit to the left
>>	Shift the bit to the right
&	Bitwise of AND
^	Bitwise of exclusive
|	Bitwise of OR
R(x)	Received codeword

Chapter 1.

1. Introduction

1.1. Problem Statement

Coding theory is interested in providing a reliability and trustiness in each communication system over a noisy channel. In the more communication systems, the error correction codes are used to find and correct the possible bit changing [2], for example, in wireless phones and etc.

In any environment, Environmental interference, physical fault noise, electromagnetic radiation and other kinds of noises and disturbances in the communication system affect communication direction to corrupted messages, an errors in the received codeword (message) or bit changing might be happened during a transmission of data [1] [2].

So, the data can be corrupted during a transmission and dispatching from the transmitter (sender) to the receiver, but during a data transmission, it might be affected by a noise, then the input data or generated codeword is not the same as received codeword or output data [1]. The error or bit changing which is happened in data bits can change the real value of the bit(s) from 0 to 1 or vice versa.

In the following simple figure, a base structure of communication system is indicated. And this figure indicates that how the binary signal can be affected by a noise or other effects during a transmission on the noisy communication channel:

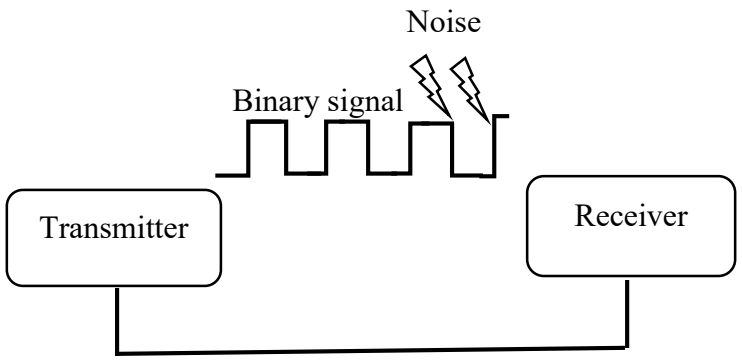

Figure 1. Data Transmission

Generally, there are three types of errors that can be corrupted in data transmission from the sender to the recipient:

- Single bit error: The error is called single bit error when bit changing or an error happens in one bit of the whole data sequence [3].

- Multiple bit errors: the occurred errors are called multiple bit errors, if there are a bit changing or an errors in two or more than two bits of the sequence of forwarded data [3].

- Burst errors: if an errors or bits changing are happened on the set of bits in the transmitted codeword, then the burst error is occurred. The burst error will be calculated from the first bit which is changed until the last changed bit [3].

These types of errors will be discussed in chapter three of this report in detail.

1.2. The General Idea

The main method is using redundancy (parity) to recover messages with an error during transmission over a noisy channel.

As a simple example, an error can happen in human language in both verbal and written communication. For instance, if during a reading of this sentence: “this sentence is miscake”, there is an error and wrong word in this sentence and the error should be found that in this sentence in which word the mistake is occurred and then it must be corrected, so two important things must be achieved here: error detection and error correction, and the principles that must be used to achieve these goals are first in English language the string “miscake” is not accepted word, so in this point, it is obvious that the error is occurred, Secondly, the word “miscake” is closest to the real and correct word “mistake” in English, so it is the closest and the best word which can be used instead of the wrong word [4].

So, it shows how redundancy can be useful in the example of human language, but the goal of this project is how the computers can use some of the same principles to achieve error detection and error correction in the digital communication?

To get the best idea of correction by using a redundancy in digital communication, first of all, it is necessary to model the main scheme contains two main parts called encoding and decoding like the following figure and in the next parts all of these parts will be explained in detail and implemented in C++ language.

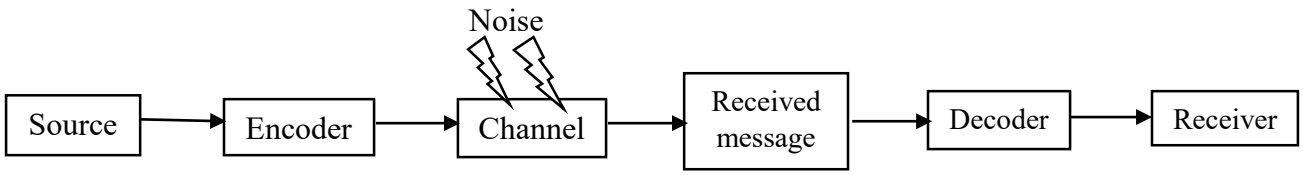

Figure 2. The main scheme of hamming code [2]

According to this figure [2] [3], first of all, the code will be generated by a source, and then to do encoding part, the parity (redundancy) bits will be added by the encoder to the data bits which sent from a source. After that, the generated codeword which is a combination of data bits and parity bits will be transmitted to the receiver side, and during transmission, the error or bit changing might be happened in the communication channel over produced codeword. At the end, the corrupted error must be detected and corrected by decoder on the receiver side.

Chapter 2.

2. Literature Review

2.1. Historical Background related to Hamming Code

One of the proper subset of information theory is called coding theory, but the concept of these theories are completely different [2].

The main subject is began from a seminar paper which presented by Claude Shannon in the mid of 20th century. And in that paper, he demonstrated that good code exist, but his assertions were probabilistic. Then after Shannon's theorem, Dr Hamming [1] [3] and Marcel Golay [5] presented their first error correction codes which called Hamming and Golay codes [2].

In 1947, Dr Hamming introduced and invented the first method and generation of error correction code called hamming code [1] [6]. Hamming code is capable to correct one error in a block of the received message contains binary symbols [7]. After that, Dr Hamming has published a paper [1] in the Bell technical journal with a subject of error detection and error correction code.

In 1960, other methods for error detection and error correction codes were introduced and invented by another inventor, for example, BCH code was invented by Bose, Chaudhuri and Hocquenghem who are a combination of the surname of the initial inventors' of BCH code [7].

Another error detection and error correction code which presented in 1960 was Reed Solomon (RS) code that invented by two inventors called Irving Reed and Gustave Solomon and this method was developed by more powerful computers and more efficient algorithm for decoding part [8].

2.2. Types of Hamming Code

There are three more significant types of hamming codes which called 1. Standard hamming code, 2. Extended hamming code [9], 3. Extended hamming product code, but in this research, the standard hamming code is used and implemented.

2.2.1. Standard Hamming Code

The standard method of hamming code is depended on a minimum hamming distance, it means to design the standard hamming code, minimum hamming distance which is three among any two codewords is needed, for instance, a standard hamming code with a dimension of H (7, 4) which four data bits are encoded into 7 bits by appending 3 redundancy bits [10]. In the next chapter, this kind of hamming code will be explained in detail.

2.2.2. Extended Hamming Code (EH)

In this type of hamming code (EH), the minimum distance will be increased to have one more than the standard form of hamming code, so the minimum distance is equal to four among every two codewords. The main frame and structure of the Hamming code is the same for both binary and extended Hamming code.

In fact, in the extended Hamming code there is an extra bit which is added to the redundant (parity) bit that allows the decoder to recognize between single and double bit errors. Afterwards, on the receiver side, the decoder can detect and correct an error with one bit changing and also a double bit errors can be detected (not corrected) at the same time, and if there is no attempting by a decoder to correct this single bit error, then it also can detect maximum three errors [10].

In the last example above, the extended hamming code defined as H (8, 4) and here there are 4 parity bits which are calculated by a subscription among 8 total bits and 4 data bits.

2.2.3. Extended Hamming Product Codes

Extended Hamming product codes is a code which is built based on the extended Hamming code (EH). This kind of code is consist of the new algorithms and outcomes. The more important issue which is so significant for both researchers and designers is low error rate for the evaluation of efficiency for channel coding scheme. Also, there are two remarkable values called BER (bit error rate) and FER (frame error rate) which some fresh digital radio links are designed for them, because nowadays, using wireless multimedia applications are going to be extended and low-error-rates are so important issue for these applications [11].

There are two more important schemes for coding which has a special performance and proficiency that is very close to Shannon limit, and they are called turbo codes [11], And low-density parity check codes [12]. So the product codes can be more competitive, therewith, this kind of codes can be used for the implementation of the fast parallel decoder.

One of the most substantial and prosperous works and paper to accede the Shannon limit was published in 1993 by three researchers called Berrou, Glavieux and Thitimajshima, at the same time [14]. They also introduced and stated turbo codes, which also known as PCCC (PCCC is an abbreviation of parallel concatenated convolutional codes).

The turbo codes is a special technique and method of the coding theory which used to prepare a reliable and reputable communication over a noisy or messy communication channel. The turbo codes are a special class of FEC (forward error correction) codes, and they are used in 3G or 4G mobile communications, for instance, in LTE. The turbo codes gain their considerable efficiency and performance with relatively low complication algorithms for encoding and decoding parts. The turbo codes can attain BER or bit-error-rate levels which should be around 10^{-5} at the code rate which is completely close to the specifically related capacity with advisable complexity and

convolution for decoding. And the most important key that made it to the successful algorithm was using decoding algorithm which is called soft-in soft-out [14] [15]. In the last years, other similar codes such as SCCCS, low-density parity check codes and also the block product codes are introduced and studied, but the product codes are using a high level of rating and degree in parallelization, for instance, PCCC [15].

In this method (turbo codes), the main goal is preparing a perfect and complete set of techniques and analytical methods to have the performance related to the low-error-rate for the extended Hamming product codes. For doing analytical approximation three important parameters are required which must be evaluated for the extended Hamming (EH) product codes, these parameters are called the code performance at low-error-rates, the exact knowledge of the code minimum distance and its multiplicities, respectively [16].

2.2.3.1. Binary linear code

Linear codes are determined by the special alphabets called Σ which are finite fields. All over, it will be denoted by F_q which means the finite fields with q elements (q is a primer power and F_q is {0, 1, . . . , q − 1}) [16].

If a required field is Σ and C ⊂ Σ^n is a subset of Σ^n, then C will be called a linear code. And because C is a subset, so there are several basis c (like: c1, c2, . . . , c_k which k is a dimension of the subset), and each generated codeword can be declared as the linear combination of these basis vectors and these vectors can be written In the dominant of proper matrix as the columns of a n×k matrix and this proper matrix is called G matrix (generator matrix) [16].

According to the binary linear code which is written in the form of C (n, k), also there are some values which are described in the following:

There are k data bits or information frame and r parity bits, also, n is the total number of bits of a codeword where n = (k + r). A particular symbol $W_H(.)$ is the Hamming weight of a vector [16].

u = $(u_1, u_2, \dots, u_k)$

c = $(c_1, c_2, \dots, c_n)$

c = (u | P) where the first k bits are a data bits which is generated by u and P is a specific vector which is contained r parity bits and these parity bits will be placed after the first k data bits [16].

A. Weight Enumerating Functions (WEF)

Weight enumerating function (WEF) admits the superlative description and complete information for a weight structure. The turbo codes consist of weight enumerating function of the component codes, for instance, the traditional trellis search [13]. There are three distinguished WEF for a code [16], the weight enumerating function is like the following formula:

$$W_{C(y)} = \sum_{c \in C} y^{w_H(C)} = \sum_{i=0}^{n} A_i y^i$$

Where A_i is the number of codewords in which the weight is $W_H(c)$ = i. And another significant formula will be $IW_{C(y)}$ which is called the information weight of an enumerating function and it will be written like the downward formula with some changed parameters in comparison to above formula [16]:

$$IW_{C(y)} = \sum_{c \in C} W_H(u) y^{w_H(C)} = \sum_{i=0}^{n} W_i y^i$$

In this formula W_i is a data (information) multiplicity and it used instead of A_i because it's the summation of the Hamming weights of A_i and u is a specific data frame which the codeword with a weight of $W_H(c)$ = i will be generated by it [16].

The third significant function is IOWEF (or the input output weight enumerating function) which is written like the underneath formula:

$$IOW_C(x, y, X, Y) = \sum_{c \in C} x^{k - W_H(u)} y^{W_H(u)} X^{r - W_H(P)} Y^{W_H(P)}$$

In this formula the number of codewords c = (u | P) which $W_H(u)$ is equal to w and $W_H(P)$ is equal to p, so if we replace w and p instead of $W_H(u)$ and $W_H(P)$ in above formula, respectively [16], then the result will be like the following formula called $IOW_C(x, y, X, Y)$:

$$IOW_C(x, y, X, Y) = \sum_{w=0}^{k} \sum_{p=0}^{r} A_{(wp)} x^{k-w} y^w X^{r-p} Y^p$$

So, the minimum non zero $A_i = \sum_{w+p=i} A_{(wp)}$ and w_i = w. $\sum_{w+p=i} A_{(wp)}$

B. BER and FER Performance for Maximum Likelihood Decoding:

In this part, the main foundation related to the evaluation of analytical code performance is at the SNR like, low error rates. According to the binary linear codes C (n, k) which can be transmitted by a binary averse constellation, for example, a 2-PAM, Gray labeled 4-PSK and etc. upon the increasable White Gaussian Noise channel. So it will be so clear that BER and FER performance for maximum likelihood decoding which are corresponded to the specific ratio among the energy of data bits and the spectral density.

The bit error rate or BER is the number of bits containing an errors in each time, so the ratio of bit error will be calculated by the number of bits containing an error over the total number of bits which are transferred over a communication channel during the time interval (the result can be expressed as a percentage).

The frame error rate (FER) is a ratio of errors for a received data bits, it can be used for evaluating a quality of the signal connection. If the result of FER is so high, it means there are lots of errors among the whole received bits, and in this situation, the connection can be rejected and dropped [16].

$$\text{FER} <= \sum_{i=d_{min}}^{n} \frac{1}{2} A_i \, erfc\left(\sqrt{i \frac{k}{n} \frac{E_b}{N_0}}\right)$$

$$\text{BER} <= \sum_{i=d_{min}}^{n} \frac{1}{2} \frac{w_i}{k} erfc\left(\sqrt{i \frac{k}{n} \frac{E_b}{N_0}}\right)$$

Where E_b is the energy of each data (information) bit and N_0 is the noise of spectral density and the ratio between them ($\frac{E_b}{N_0}$) is significant and related to the BER and FER performance for maximum likelihood decoding.

When the SNR is very high, then the error rate will be very low and actually the code performance will be concurred with the union band truncated for contribution of d_{min}, so the following formula can be written according to the mentioned condition and the above formulas for very high SNR and FER_{EF} and BER_{EF} are called the code error floor [16]:

$$\text{FER} \approx FER_{EF} \triangleq \frac{1}{2} A_{min} erfc\left(\sqrt{d_{min} \frac{k}{n} \frac{E_b}{N_0}}\right)$$

$$\text{BER} \approx BER_{EF} \triangleq \frac{1}{2} \frac{w_{min}}{k} erfc\left(\sqrt{d_{min} \frac{k}{n} \frac{E_b}{N_0}}\right)$$

C. BER properties:

There are some main problems which are related to the computation of multiplicity, so the new theoretical results are indicated in the below for solving these problems. A value of d_{min} (minimum distance) and its multiplicity or A_{min} are explicit and clear for many codes, but the value of w_{min} is very difficult to compute.

So, pursuant to the above formula correspond to BER_{EF}, we will have: $\frac{\mathrm{w_{min}}}{\mathrm{k}} \approx \mathrm{A_{min}}.\frac{\mathrm{d_{min}}}{\mathrm{n}}$ then the mentioned error floor is connected by $\mathrm{BER_{EF}} \approx \mathrm{FER_{EF}}.\frac{\mathrm{d_{min}}}{\mathrm{n}}$ (in fact, it's a relation between $\mathrm{BER_{EF}}$and $\mathrm{FER_{EF}}$ error floor). $\mathrm{BER_{EF}} \approx \mathrm{FER_{EF}}.\frac{\mathrm{d_{min}}}{\mathrm{n}}$ will be satisfied by some codes with equality, so in this situation, they satisfy and process BER property, and if all of the multiplicities which called A_i and the data multiplicities which called w_i can be connected by this property: $\mathrm{w_i} = \mathrm{A_i}.\mathrm{i}.\frac{\mathrm{k}}{\mathrm{n}}$, but both multiplicity and the properties correspond to BER are still unsolved and open problem, in the next parts some solution called extended hamming code and extended hamming product code will be explained for these properties [16].

2.2.4. Extended Hamming code vs. Extended Hamming product code

In the extended Hamming code $\mathrm{EH_r(n,k)}$, the minimum distance will be increased to have one more than the standard form of the Hamming code, so the minimum distance is equal to four among every two codewords. The main frame and structure of the Hamming code is the same for both binary and extended Hamming code. In fact, in the extended Hamming code there is an extra bit which is added to the redundant (parity) bit that allows the decoder to recognize between single and double bit errors. So as mentioned above by adding an additional parity bit to the Hamming code, $\mathrm{EH_r(n,k)}$ will be characterized by $\mathrm{n} = 2^r$, $\mathrm{k} = 2^r - r - 1$ and r is a parity bit which must be an integer numbers greater than two. Also, the form of multiplicity for extended Hamming code can be found by the following lemma [16]:

$$\mathrm{A_{2i}} = \frac{\binom{\mathrm{n}}{\mathrm{2i}} + (-1)^{\mathrm{i}} \times (\mathrm{n}-1) \times \binom{\frac{\mathrm{n}}{2}}{\mathrm{i}}}{\mathrm{n}}, \quad \mathrm{i} = 2, 3, \ldots$$

For introducing an extended hamming product codes, first of all, if C_1 is a block code like (n_1, k_1) and C_2 is a block code with a form of (n_2, k_2), then a proper product code which is equal to the multiplication between first and second block code $C_P(n_P, k_P) = (C_1 \times C_2)$, so the result will be equal to: $(n_1\, n_2, k_1 k_2)$ code [15].

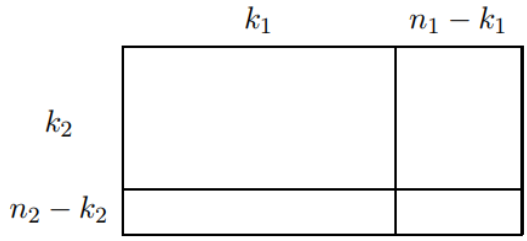

Figure 3. Relation among data and parity bits in extended hamming product code [15]

The systematic encoder for the product code $C_P(n_P, k_P) = (C_1 \times C_2) = EH_{r_1}(n_1, k_1) \times EH_{r_2}(n_2, k_2)$ will be obtained by three important cases:

1) First of all the proper matrix contains data bits must be written, to create this matrix, first the data symbols $k_1 k_2$ will be arranged into $k_2 \times k_1$ array.
2) At the second step, k_1 rows will be encoded by using code C_2 and then $(n_2 - k_2)$ parity bits can be added to each row of this matrix.
3) Finally, all n_2 columns must be encoded by using code C_1 which can add $(n_1 - k_1)$ parity bits to the end of each column in this matrix.

Then it's so clear that the minimum distance of the product code $C_P(n_P, k_P) = (C_1 \times C_2) = EH_{r_1}(n_1, k_1) \times EH_{r_2}(n_2, k_2)$ is $d_{Pmin} = d_{1min} \times d_{2min}$ and the values of its multiplicity A^P_{min} and W^P_{min} are the product of C1 and C2, so the minimum distance of product code is equal to 16 ($A^P_{min} = A^P_{16} = A^1_4 \times A^2_4$). Also, each column of a matrix is a codeword of C1. But if both C1 and C2 are the linear codes, then all rows in this matrix are codewords of C2. So according to these definitions, systematic encoders for C1 and C2 are assumed. And according to this property.

As a proper conclusion for a product code which is a construction of placing the data bits into the matrix, the rows and columns of this matrix will be encoded separately by using the linear block codes, this kind of encoder for the product code is drawn in the underneath figure which shows a typical encoding procedure for a product code when a block code is used for encoding rows and columns of a matrix [15]:

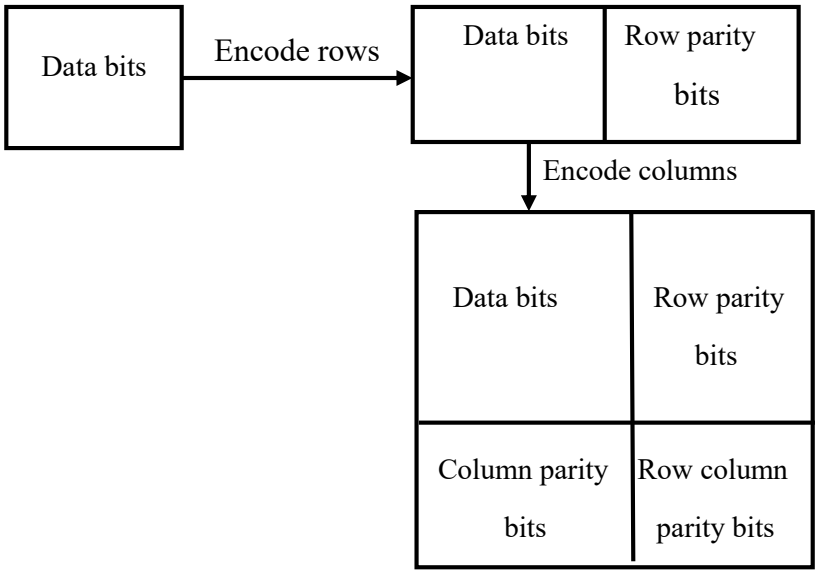

Figure 4. Typical encoding procedure for a product code [15]

2.2.5. Codes with the property of BER and a group of transitive automorphism

In this part, the goal is evaluation and proofing that when a code has a transitive automorphism group, then the specific multiplicities and the property of BER will be a part of this code. After that, another issue must be proved which is the issue that the extended Hamming product codes and the paronymous and derived form statement for their data multiplicity [16].

A group of transitive automorphism is a group with a special property that any two specified non-identity elements of the group, there is an automorphism of this group which is sending the first to the second. According to the binary code C (n, k) which is described completely in the last part, a permutation of the coordinates is a symmetry of C, if it maps one codeword in another codeword [16].

Theorem A: A binary codes has C (n, k) will satisfy the multiplicity property(w_i), if it has a transitive automorphism group, so: $w_i = \frac{i\, A_i k}{n}$.

There are codewords which called $A_i$where I is a weight of the codewords. First of all, they should be put in a special matrix called M which the number of rows and columns of matrix M are called A_i and n, respectively. Also, the first k columns of this matrix are contained the number of ones because C is defined as a systematic [16] [17].

If i and j are two different coordinates (i≠j and 1 <= i <= n) and then the permutation p is a member of Aut (C) which can map i into j where Aut (C) is transitive, and the main responsibility of permutation p is mapping any row and column of a matrix into another row and column.

Transitive automorphism group also exist in the extended Hamming code, so after that they will contain the multiplicity and BER property, so the data multiplicity for $\mathrm{EH}_r(n, k)$ or an extended hamming code is:

$$\mathrm{w_{2i}} = \frac{2_{ik}\left[\binom{\mathrm{n}}{2\mathrm{i}} + (-1)^{\mathrm{i}} \times (\mathrm{n}-1) \times \binom{\frac{\mathrm{n}}{2}}{\mathrm{i}}\right]}{\mathrm{n}^2} \text{ where i = 2, 3 , …}$$

Theorem B: if there are 2 binary codes C_1 and C_2, then it must be proved that the transitive automorphism group will be like Aut ($\mathrm{C}_1 \times \mathrm{C}_2$) [38] [39].

According to $\mathrm{C_P}(n_P, k_P) = (\mathrm{C}_1 \times \mathrm{C}_2) = \mathrm{EH}_{r_1}(n_1, k_1) \times \mathrm{EH}_{r_2}(n_2, k_2) = (n_1\, n_2,\ k_1 k_2)$, any codeword will be contained some coordinates ($\mathrm{x_i}$, $\mathrm{y_i}$) related to its position in this matrix [16][17].

Then for simplicity, if 1 <= i <= n^P, then i ≙ ($\mathrm{x_i}$, $\mathrm{y_i}$) and j ≙ ($\mathrm{x_j}$, $\mathrm{y_j}$) and for each of them, there are two permutation symmetry p1 of C_1 which will map $\mathrm{x_i}$ in $\mathrm{x_j}$ and permutation p2 of C_2 that map $\mathrm{y_i}$ into $\mathrm{y_j}$, then the first permutation symmetry will be applied to the rows of specified matrix and second permutation symmetry will be applied to the columns of this matrix, so for each couple of coordinates, the proper permutation symmetry will be built to map coordinate i into coordinate j and after that, it's possible to create transitive automorphism group [16] [17].

So, the dominant multiplicity for $\mathrm{C_P}(n_P, k_P) = \mathrm{EH}_{r_1}(n_1, k_1) \times \mathrm{EH}_{r_2}(n_2, k_2)$ is $W_{min}^{P} = \frac{k_1(n_1-1)(n_1-2)k_1(n_2-1)(n_2-2)}{36}$. In the following table all dominant and information multiplicity of square extended Hamming product codes are shown and by looking this table, it's obvious that product codes are contained a large minimum distance by using r which is in the range among 3 to 9 and also they have very large multiplicities [16].

C_P	(n_P, k_P)	$R_P = \frac{k_P}{n_P}$	d_{min}^P	A_{min}^P	W_{min}^P
$(EH_3)^2 = (8,4)^2$	(64, 16)	0.250	16	196	784
$(EH_4)^2 = (16,11)^2$	(256, 121)	0.473	16	19600	148225
$(EH_5)^2 = (32,26)^2$	(1024, 676)	0.660	16	1537600	16240900
…	…	…	…	…	…
$(EH_9)^2 = (512,502)^2$	(262144, 252004)	0.961	16	30910041702400	475430551096900

Table 1. Dominant and information multiplicity [16]

2.2.6. Error floors of extended Hamming product codes

Conforming to table 1, it's explicit that a large minimum distance which is equal to 16 is achieved by product code. Also, the multiplicities will be so large for high code rates and this situation does not match with the behavior of turbo codes that always offer a low rate of multiplicities and also minimum distance, in the following figure the BER and FER error floors are drawn, separately for SEHPC (or the square extended hamming product code) [16] [17].

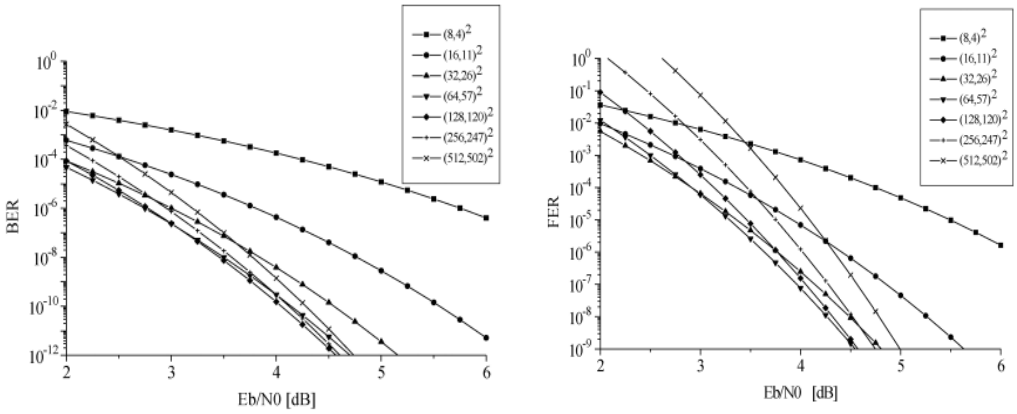

Figure 5. BER and FER error floor [16]

On the other hand, the analytically computed error floors are shown below for a code performance at low error rate. To reach this goal the special optimal algorithm called BCJR (Bahl, Cocke, Jelinek and Raviv) is required for minimizing the probability of bit error, but there is one drawback in this algorithm which can prohibit high rate execution is related to its complexity [17].

The main goal of the BCJR algorithm is minimizing the BER or bit error rate by calculating APP (a posteriori probability) of a special bits among the bits of the codeword (in fact, minimizing BER will be done by maximizing APP and exactly for this reason BCJR decoder is called MAP which is abbreviation of Maximum Posteriori Probability decoder) [17]. In the underneath figure the proper performance of BER and FER related to the extended Hamming code is drawn for $(EH_5)^2$=$(32, 26)^2$ as an example by applying both BCJR and chase algorithms [15] [16].

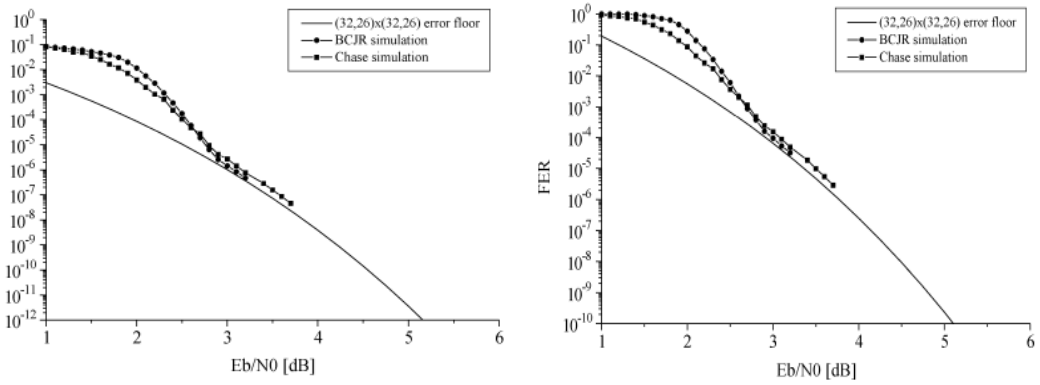

Figure 6. BER and FER by applying BCJR and Chase algorithms [16]

In this figure, the drawn curves are obtained with 15 iterations and to progress iterative decoding of this kind of hamming code a fixed coefficient (multiplier) weighting outer data is applied during these iterations.

2.2.7. Algorithm for Computing an error floor for shortest extended Hamming product codes

In this case, two more significant operations are considered which are called shortening or puncturing. If C is a linear binary code with a length of n. then a shortened code is a special collection of whole codewords of C that are equivalent to zero at the constant coordinate with the coordinate which is eliminated. So the codewords in C which are with one at that coordinate must be eliminated from C. Thus the coordinates which are non-zero for whole codewords must be deleted from C. Matching to binary linear code C (n, k) and an integer number (s) which is less than the number of data bits (k), then C_S (n_S, k_S) is a shortened code where $k_S = k - s$ and $n_S = n - s$, also encoding and decoding for the shortened code is so easy by using C (n, k) circuits, so it is a sub-code of binary linear code, exactly, for this reason, the original minimum distance will be kept by this kind of code [16] [17] [18].

The main drawback of shortened extended Hamming codes is that by using this code, a transitive automorphism group and the multiplicity property never reach and hold. Then the algorithm called extended McWilliams theorem [18] to dual code IOWEF will be introduced and stated for computing the data multiplicity for them.
In fact, C (n, k) is a code with a generator matrix and $C^{\perp}$ (n, n-k) is a dual of C with a proper check matrix, then the extended weight of $C^{\perp}$ will be appointed and determined by the extended weight of C and vice versa by using the following formula [16]:

$\mathrm{IOW}_{C^{\perp}}(x, y, X, Y) = \frac{1}{|C|}\mathrm{IOW}_{C}(x + y, x - y, X + Y, X - Y)$ where $|C|$ is the number of codewords, so the underneath algorithm will be used for computing the multiplicity of a shortening code. So, consider SC: $C^{\perp}$(n, n-k), first of all, the $\mathrm{IOW}_{C^{\perp}}$ which is an abbreviation of input or output weight of enumerating function must be specified and calculated. Then for computing the IOW_{SC} the extended MacWilliams identity should be used, and at the end, compute the coefficients of SC which are A_i and W_i [16] [18].

2.2.8. Error floor of punctured code

The most significant famous way for incrementing the rate of convolutional codes is called puncturing. Its process will be done by a particular pattern of puncturing to mother code C (n, k) which is the original code and the pattern of eliminating symbols is known as a puncturing pattern [19]. So consider a mother code, a puncturing code $C'(n', k)$ will be reached by applying a special puncturing pattern to each codeword and it's able to remove d bits which is equal to $n - n'$ [16] [19]. It is very useful for applications that are reconfigurable which are required for Reiterated changes of code rates, so for this reason, it can enhance the amount of code rate without changing the length of a data-rate. And the minimum distance or dmin can be decreased by puncturing, and according to the following lemma, this important issue will be proved [16].

Lemma: According to C (n, k) which its minimum distance is equal to 16 and the puncturing code $C'(n', k)$ with d = n - n' bits, for example for d=1, 2, 3, the minimum distance for punctured code will be equal to 16 – d [16].
An EH code is a dual of Reed-Muller code which has a transitive automorphism group and for each triplex of coordinates, then the weight will be equal to -4 codewords. If d ≤ 3, then there are d bits with a clear weight -16 codewords, but if d > 3, then the minimum distance will be computing according to the puncturing pattern. The new notion of uniform puncturing in needed for analyzing a performance of a code [16].

As it is told puncturing method is a special process to eliminate certain parity bits in whole bits of the codeword relevant to the matrix called puncturing matrix [20]. And by using a puncturing method, the code rate will be incremented, but without growing the complexity of code-rate [20]. But by using the uniform puncturing, the average multiplicity and also the curves of the average analytical error floor will be calculated, analytically [16].
In the following, the specific analytical formula is provided, and the multiplicity of a puncturing code $C'(n', k)$ will be calculated by this formula:

$$\overline{A'_{(wp')}} = \frac{1}{\binom{r}{d}} \sum_{p=p'}^{p'+d} A_{(wp)} \binom{p}{p'} \binom{r-p}{t-p'}$$

Where $0 \leq w \leq k$, $0 \leq p' \leq t$ and parity bit $r = n - k$ and other transmitted bits to the punctured code $t = r - d$ and $\overline{A'_{(wp')}}$ is the average IO multiplicity of punctured code upon all available pattern of puncturing which is $\binom{r}{d}$ [16].

The average multiplicity or $\overline{A'_i} = \frac{1}{\binom{r}{d}} \sum_{w+p'=i} \sum_{p=p'}^{p'+d} A_{(wp)} \binom{p}{p'} \binom{r-p}{t-p'}$ and the average information multiplicity or $\overline{w'_i} = \frac{1}{\binom{r}{d}} \sum_{w+p'=i} w. \sum_{p=p'}^{p'+d} A_{(wp)} \binom{p}{p'} \binom{r-p}{t-p'}$.

Chapter 3.

3. Theoretical Description of Hamming Code

3.1. Description of Hamming Code

A hamming code is a simple method of error detection and error correction which is used frequently [22]. A hamming code always check the error for all the bits which exist in the codeword (in fact, it checks all bits of a codeword from the first bit to the last available bit) [22]. And exactly for this reason the hamming code is called linear code for error detection and error correction that can detect maximum two synchronic bit errors and it is capable to correct only single bit error (can correct just one bit error) [3].

This method works by appending a special bits called parity or redundancy bits. Several piece of these extra bits is installed and inserted among the data bits at the desired positions, but the number of these extra bits is depend on the number of data bits [22].

The hamming code initially introduced code that enclosed [1], for example, to construct the Hamming (7, 4) that the total bits are equal to seven, which four bits are data bits into seven bits by adding three parity bits. And the number of each bit must start from one, like 1, 2, 3, 4, 5, 6, 7 and to write them in binary: 001, 010, 011, 100, 101, 110, 111.

The parity bits are calculated at the proper positions that are calculated by powers of two: 1, 2, 4 and etc. And the data bits must be located at another remained positions which in this example are: 3, 5, 6, 7.

Each data bit is contained in the calculation of two or more parity bits. In particular, parity bit one (r1) is calculated from those bits that their positions has the least considerable or important set of bits which are: 1, 3, 5 and 7 (or in binary 001, 011, 101 and 111).

Parity bit two or r2 (at index two, or 10 in binary) is calculated from the bits that their index has the second least important set of bits which are: 2, 3, 6, 7 (or 010, 011, 110, 111 in binary).

Parity bit three or r3 (which must be located at position four or 100 in binary) is calculated from the bits where their positions has the third least considerable set of bits which are: 4, 5, 6 and 7 (or 100, 101, 110, 111).

The code sends message bits padded with specific parity or redundancy bits in the form of is the block size or a whole number of bits and k is the number of data bits in the generated codeword [1]. All the procedure which is required for adding parity bits to the data bits to encode and create a proper codeword will be explained in next parts of this report.

In the following table, all the possible Hamming codes are indicated:

Parity bits	Total bits	Data bits	Name	Rate	Overhead factor
3	7	4	H (7, 4)	4/7 = 0.571	7/4=1.75
4	15	11	H (15, 11)	11/15 = 0.733	15/11=1.36
5	31	26	H (31, 26)	26/31 = 0.839	31/26=1.19
6	63	57	H (63, 57)	57/63 = 0.905	63/57=1.10
7	127	120	H (127, 120)	120/127=0.945	127/120=1.05
8	255	247	H (255, 247)	247/255=0.969	255/247=1.03
…..					
r	$n = 2^r - 1$	$k = 2^r - r - 1$	H (n, k) = H ($2^r - 1, 2^r - r - 1$)	k/n	n/k

Table 2. A dimension of possible Hamming codes

According to table 1, if there are r parity bits, it can cover bits from 1 up to $2^r - 1$ which called total bits (or n), and also the number of parity bits should be more than one (r > 1). For each dimension of hamming code, there is a rate which is equal to a division of the number of data bits and number of total bits (rate $= \frac{k}{n}$) and always its result will be less than 1, another important factor is called overhead factor which is calculated by a division among total bits or n and data bits or k (overhead factor $= \frac{n}{k}$), and its result is always more than one [2] [3].

3.2. An Alternative Description of the Hamming Code

In the following figure which indicated another description of the Hamming code. There are three nested circles in this figure and they are related to the three parity equation defining the Hamming code, also there are seven areas among these available circles in this figure that are related to the seven bits in a final codeword [2]. The number of the circle can be extended for each dimension of hamming code.

When the single bit error happens, during a transmission process, this error will be fixed by flipping this bit (single bit) in the proper area, related to the position where the error happens.

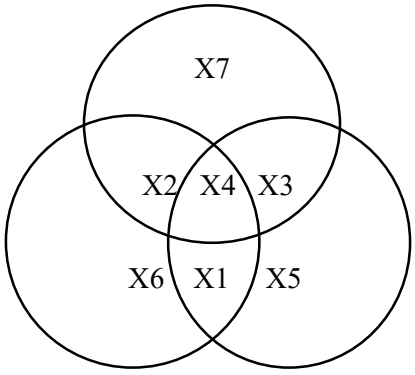

Figure 7. An additional description for the Hamming code [2]

For example: For H (7, 4) which encodes four data bits by appending three parity or redundancy bits to generate a proper codeword which is a combination of both parity and data bits.

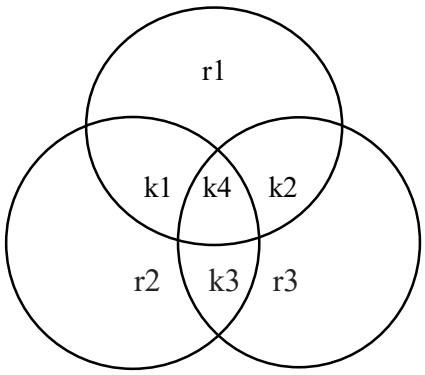

Figure 8. Graphical description of H (7, 4) with 4 data bits (k) and 3 parity bits (r) [2]

In the upper figure which describe [7, 4] Hamming code, graphically, each parity bits can cover just its adjacent bit positions which can be common with other parity bits in this figure:

For instance, parity bit 1 (r1) covers data bits: k1, k2, k4.
And parity bit 2 (r2) covers data bits: k1, k3, k4.
And parity bit 3 (r3) covers data bits: k2, k3, k4.

3.3. Theoretical Description for Encoding Part of Hamming Code

Hamming code is a specific linear code which can correct just one error by adding r parity bits to k data bits that generated by the source to have a codeword with a length of n which is equal to k + r. And the data bits or k is equal to $2^r - r - 1$ to generate a codeword with a length of $2^r - 1$ [1] [6] [7].

General algorithm for the encoding part of hamming code is described in the following:

1. r parity or redundancy bits are combined to k data bits to creating a proper codeword which is contains r + k bits.
   So the bit position sequence is from the position number 1 to r + k.

   In fact, for decoding part on the receiver side which is contained detection and correction step for an occurred error, extra bits is needed to send some extra bits with data bits which called parity (redundant) bits and these parity bits are added by the sender (by encoding) and they always removed by the receiver on the receiver side (by decoding) [23].

But how to merge the data bits and parity bits into a codeword?
The parity bit positions are always numbered in powers of two, reserved and prepared for the parity bits and another bit positions are related to the data bits [23].

The first parity bit is located in the bit position: $2^0 = 1$.
The second parity bit is located in a bit position: $2^1 = 2$.
The third parity bit is located in a bit position: $2^2 = 2$ and so on.

In the following simple table, the position of each parity bits (r) presented in a grey colored cells and they are reserved for the parity purpose.

Bit position	1	2	3	4	5	6	7	…
Parity bits	r1	r2		r3				…

Table 3. Parity bits position [22]

After that, the data bits (k) are copied to the other free and remaining positions where they are not reserved before. And the data bits will be appended in the same order as they are appeared in the main data bits which generated by source.

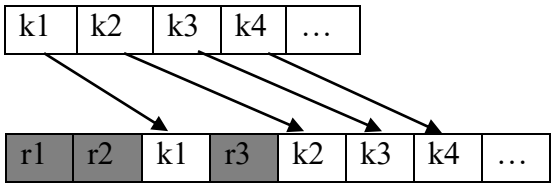

Figure 9. Merging of data and parity bits [22]

So, the transmitter adds parity bits through a process that creates a relationship between parity bits and specified data bits.

Then the receiver will check the two set of bits to detect and correct the errors which will explain in the decoding part [23], but the following figure can present the main idea of coding that is used in the hamming code.

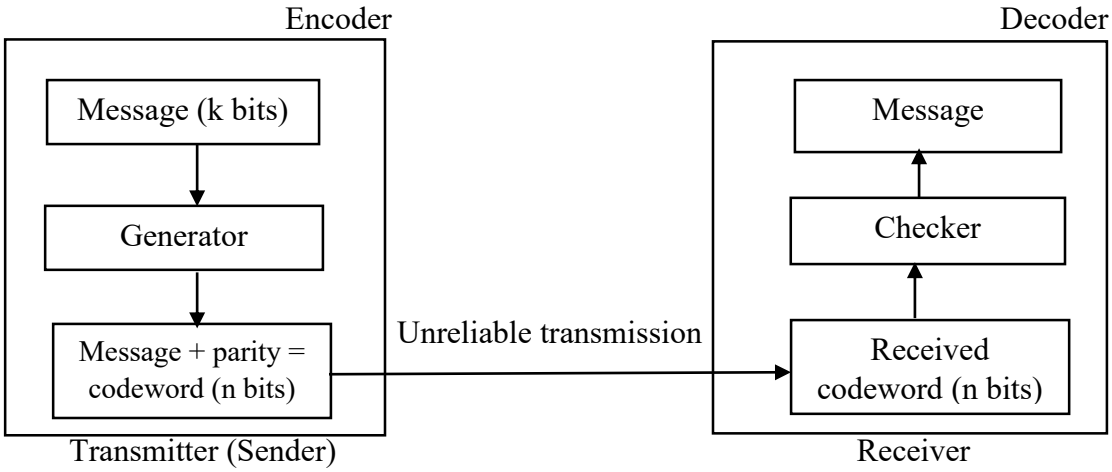

Figure 10. The Structure of Encoder and Decoder [23]

2. The value of parity bits are calculated by XOR operation of some combination of data bits [24].

A combinational of data bits are indicated in the downward table which all parity bits and data bits and their position that should be calculated according to the rules of encoding part of the hamming code.

For example, for calculating the value of parity bit number one which called r1, the XOR calculation must begin from r1, then check one bit, and skip one bit until the last bit which exists in the sequence of bits in the codeword.

Bit position	1	2	3	4	5	6	7	…
Bit type	r1	r2	k1	r3	k2	k3	k4	…
r1	*		*		*		*	
For the first parity bit (r1): Begin from r1, then check 1 bit, skip 1 bit,… (This process will be continued until the last available bit). r1=XOR of bits 3, 5, 7, ... (r1=k1⊕k2⊕k4⊕…)								
r2		*	*			*	*	…
For the second parity bit (r2): Begin from r2, then check 2 bits, skip 2 bits,… (This process will be continued until the last available bit). r2=XOR of bits 3, 6, 7, … (r2=k1⊕k3⊕k4⊕…)								
r3				*	*	*	*	…
For the third parity bit (r3): Begin from r3, then check 4 bits, skip 4 bits,… (This process will be continued until the last available bit). For r3, 4bits must be checked, because r3 is located on 4th bit of codeword. r3=XOR of bits 5, 6, 7, … (r3=k2⊕k3⊕k4⊕…)								

Table 4. Calculation method of parity bits [24]

The result of XOR operation is zero if both two bits are the same, otherwise, if two bits are different then the result will be one (according to the following table).

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

Table 5. Result of XORing of two single bits

The XOR operation in table 4 indicates the odd function. If the number of 1's among the variables X and Y are odd number, then the result of XORing among these two single bits are equal to one, otherwise if the number of 1's are even numbers, then the result will be zero.

3.4. Theoretical description for decoding part of Hamming Code

In block coding, the message is divided into blocks and each block in contains a data bits. According to encoding part of hamming code in the last part, r parity bits will be added to each block and their data bits, so the total length of each block (n) is equal to the summation of data bits (k) and parity bits (r) and the resulting n bits block is called codeword.

For instance: The 4 bits data word with the three parity bits as a seven bits combined word containing binary numbers which they should be written in the specific memory. (For replacing the three parity bits in their proper positions, the seven bits mixed word written into memory must be catch) [3] [24].

After all definitions of encoding part and calculating a proper codeword, In the decoding part, two important steps are desired called error detection and error correction.

Bit position	1	2	3	4	5	6	7	…
Bits	r1	r2	k1	r3	k2	k3	k4	…

Table 6. Bit composite word written into memory [6]

3.4.1. Construction of G and H Matrix

1. The first required matrix for decoding part of the hamming code is a generator matrix (G) that the standard form for this matrix is $G=(I_K|P)$ where I_K is identity matrix of $k \times k$ and P is a parity matrix of $k \times r$.

Parity matrix (P) is a matrix extracted from the last three columns of the generator matrix and it will be calculated according to the bits in each row of generator matrix, contains data bits [23].

In each rows of this matrix the codewords are the linear combinations of the rows of this generator matrix, in fact, it means each row of this matrix which is a combination of data bits and parity bits is the codeword, so the three last binary numbers of each row are the parity bits which are calculated according to the data bits which are in the first four bits in each row.

For instance: $G = (I_K|P) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ where $P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

2. The second required matrix for decoding part of hamming code is check matrix (H). This matrix describes the linear relevance that must be satisfied by the components in each codeword [25]. It can be used to decide whether a particular vector is a codeword and is also used in decoding algorithms, also check matrix is given by $H=(-P^T|I_{n-k})$ where P^T is the transposed matrix of the parity matrix with exchanging rows and column of this matrix [25].

The following two matrices are the available types of check matrix for [7, 4] Hamming codes:

First type of check matrix: $H = (-P^T|I_{n-k}) = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ and the parity matrix or $P^T = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$

This kind of check matrix which is presented above is the standard form of check matrix (H), And It has among its columns each non-zero triple from exactly once [26].

But in this project another kind of check matrix is used like the next matrix:

Second type of check matrix: $H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$ and this matrix is a binary lookup table in vertical order which the column contains all zero numbers is eliminated.

Among these two check matrixes which showed above, the first matrix is the check matrix for a code which is contained a standard form of generator matrix (G). And the second matrix checks a code is not contained a generator matrix (G) which is exist in the standard form [26].

3.4.2. Types of Error

There are three kinds of errors which may occur during a transmission of codeword or data from the sender to the receiver.

3.4.2.1. Single bit Error

Single bit error means that there is an error occurred just in one bit over transmitted codeword during a transmission from the sender to the receiver [3], so the binary bit may change from 1 to 0 or from 0 to 1 as shown in figure 7 as an example:

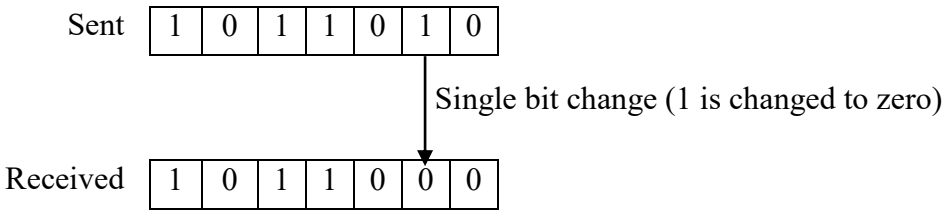

Figure 11. Single bit error

3.4.2.2. Multiple bit Errors

This kind of error is occurred, when there are more than one error in the bits of transmitted codeword during a transmission, so, the codeword is received with more than one bits error [3] [7].

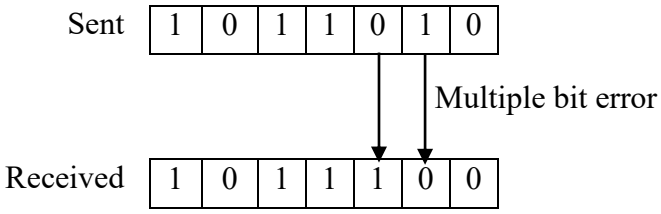

Figure 12. Multiple bit error

3.4.2.3. Burst Error

Burst error means there is an error occurred in two or more sequential bits of transmitted codeword during a transmission from the sender to the receiver.
The burst error will be calculated from the first bit which is changed until the last changed bit [2] [7].

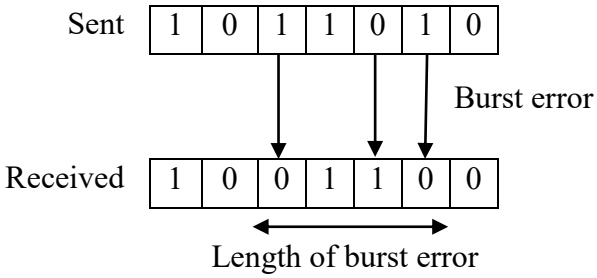

Figure 13. Burst error

3.4.3. Error Detection and Error Correction

As described in the last part, in any communication system, first of all, the data (message) in the sender side must be generated and then the generated data bits must be encoded in the block of binary bits containing a value of zero and one which called codeword which the procedure and the rules for creating a proper codeword were discussed before, after that a codeword which is a binary vector, should be forwarded and transmitted to the receiver side and the generated message must pass from a communication channel between transmitter (sender) and the receiver and during a transmission on the channel, an error or bit changing may happens [7].

If a codeword which generated on the transmitter side and the received codeword on the receiver side are both the same, then the received codeword is accepted and there is no error, otherwise, the detected error must be corrected [25], for this reason in any communication system with a variety, the error correction codes are used for detecting and correcting a probable errors in a transmitted codeword [1] [25]. But how to detect and correct in decoding part?

For a decoding part of a hamming code, there are two significant matrices called generator matrix (G) and check matrix (H) that are presented in the following respectively:

$$G=\begin{bmatrix}1&0&0&0&1&1&0\\0&1&0&0&1&0&1\\0&0&1&0&0&1&1\\0&0&0&1&1&1&1\end{bmatrix},\ H=\begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix},\ H^{T}=\begin{bmatrix}0&0&1\\0&1&0\\0&1&1\\1&0&0\\1&0&1\\1&1&0\\1&1&1\end{bmatrix}$$

In this project for decoding a received codeword (C), just check matrix (H) and its transposed matrix (H^T) is used and they will be enough for syndrome calculation. So, when the codeword with an error received by the receiver, first the received codeword must multiply by a transposed of check matrix to do error detection which this multiplication is called syndrome. If the syndrome result is equal to the zero vector (the vector with all zero elements), there is no error, but if there is an element(s) with a value equal to one, there is error occurred in received codeword [13] [25].

Syndrome (S) = $[C]$. $[H^T] = 0$ where $H^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ and the codeword (C) is given by the standard matrix product C = m * G = $m * (I_K|P) = [c_1, c_2, \ldots, c_k, c_{k+1}, \ldots, c_n]$ which G is generator matrix that is equal to k * n. And m is message part with k bits and p is parity matrix with n - k bits and $c_1, \ldots, c_k$ is message part and $c_{k+1}, \ldots, c_n$ is parity part.

For error correction, first must look at the syndrome result, and then the result must be compared to the columns of check matrix. Then the error will be corrected and fixed, just by changing a value of bit where the error occurred in corresponding bit in the received codeword [24].

Hamming code has a capacity to detect maximum two errors, but just can correct one error in each received message that means the Hamming code cannot correct burst errors if more than one error occurs during transmission of data and this issue is a disadvantage of hamming code, also another disadvantage of hamming code is that parity bits are also send to the receiver side by appending to data bits therefore more bandwidth is needed to send the generated data [24] [25].

3.5. Example of Hamming Code

As an example for Hamming code, the encoding and decoding procedure of H (7, 4) will be evaluated in the following parts.

3.5.1. Hamming Encoding Example

An Encoding of Hamming (7, 4) will be done by encoding procedure of the Hamming code that is a composition of logical functions which can convert an incoming binary message that here is equal to 4 binary digits to the proper codeword before transferring to the receiver side [27].

There are two kinds of bits for the hamming code encoding:

a) Data bits that are generated and transmitted from the transmitter (sender) to the receiver, for example: 1 0 0 1.

b) Parity bits (redundant bits) which is extra bit stream or extra binary digits that are generated in special way and combined with data bits (the combination of data bits and parity bits is called codeword) and then the codeword must be transferred to the receiver.

To calculate the number of parity bits the following method must be used:

$2^r >= k + r + 1$ where (r = parity bit which is more than 1, k = data bit) [28].

- If r = 2 then: $(2^2 >= 4 + 2 + 1) = (4 >= 7)$ that is not correct, so certainly this value is refused. Then the value of r must be increased.

- If r = 3 then: $(2^3 >= 4 + 3 + 1) = (8 >= 8)$ that is full hamming code and it is correct, so it accepted.

Total number of bits (n) = Number of data bits (k) + number of parity bits (r) = 4 + 3 = 7.

c) After specifying the number of parity bits, then the position of each parity is calculated by 2^n like the following where (n = 0, 1, 2, 3, …, n) [28].

$r1=2^0=1$ (parity bit 1 or r1 must put in position 1).

$r2=2^1=2$ (parity bit 2 or r2 must put in position 2).

$r3=2^2=4$ (parity bit 3 or r3 must put in position 4).

In the next figure, the combination of data bits and parity bits which explained in the last chapter is shown in the frame of this example for H (7, 4) that the sequence of data bits is 1 0 0 1.

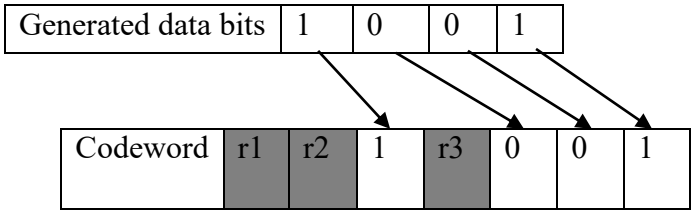

Figure 14. Example of merging data bits and parity bits

Afterwards, the value of each parity bit will be calculated by XOR operation which explained in the section 3.3.

	1	2	3	4	5	6	7
	r1	r2	k1=1	r3	k2=0	k3=0	k4=1
r1	*		*		*		*
For calculating a value of r1: begin from r1, then check 1 bit, skip 1bit, etc. , so: (r1 = 1 ⊕ 0 ⊕ 1 = 2 (even) = 0)							
r2		*	*			*	*
For calculating a value of r2: begin from r2, then check 2 bits, skip 2 bits, etc. , so: (r2 = 1 ⊕ 0 ⊕ 1 = 2 (even) = 0)							
r3				*	*	*	*
For calculating a value of r3: begin from r3, then check 4 bits, skip 4 bits, etc. , so: (r3 = 0 ⊕ 0 ⊕ 1 = 1 (odd) = 1)							

Table 7. Calculation of parity bits for hamming code example [24]

Matching to above table, the final codeword is:

Bit Position	1	2	3	4	5	6	7
Bit name	r1	r2	k1	r3	k2	k3	k4
Value of each bit	0	0	1	1	0	0	1

Table 8. Shows the value of each bit in the final codeword

The total series of bits of Hamming code for a dimension of H (7, 4) is shown in the following table, in truth, correspondent to the lookup table which contain all combination of data bits, when there are four generated data bits, in the lookup table there are fifteen different cases and states which are contained different combinational of binary numbers (as a underneath table in the column of data bits), then the proper parity bits will be located at the proper place among data bits and the whole bits are displayed in the column called codeword:

Bit #	Data bits				Parity bits			Codeword (data bits + parity bits)						
	k1	k2	k3	k4	r1	r2	r3	r1	r2	k1	r3	k2	k3	k4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	0	1	0	0	1
2	0	0	1	0	0	1	1	0	1	0	1	0	1	0
3	0	0	1	1	1	0	0	1	0	0	0	0	1	1
4	0	1	0	0	1	0	1	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0	1	0	1
6	0	1	1	0	1	1	0	1	1	0	0	1	1	0
7	0	1	1	1	0	0	1	0	0	0	1	1	1	1
8	1	0	0	0	1	1	0	1	1	1	0	0	0	0
9	1	0	0	1	0	0	1	0	0	1	1	0	0	1
10	1	0	1	0	1	0	1	1	0	1	1	0	1	0
11	1	0	1	1	0	1	0	0	1	1	0	0	1	1
12	1	1	0	0	0	1	1	0	1	1	1	1	0	0
13	1	1	0	1	1	0	0	1	0	1	0	1	0	1
14	1	1	1	0	0	0	0	0	0	1	0	1	1	0
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 9. The total series of bits of Hamming code for a dimension of H (7, 4)

3.5.2. Hamming Decoding Example

The specific reserved process to encode a data or message which can recover the main and original generated data which is generated by a source on the sender side.

Specific calculation or operation which exists at the receiver side and the first one is calculating check matrix (H) and transposed of check matrix (H^T) and then error detection by using syndrome (S):

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad , \quad H^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$ then for instance: if an error (bit changing) happened on the third bit of the codeword: C = [0 0 0 1 0 0 1].

Then syndrome (S) = $[C]$. $[H^T]$ = [0 0 0 1 0 0 1] . $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ = [2 1 1] = [0 1 1] and this vector is non-zero vector, so it means during a transmission from sender to the receiver an error or bit changing is occurred, and then all the rows of matrix H^T should be checked to find [0 1 1] among the rows of H^T. In this example [0 1 1] is exist on the third row of H^T, after that for correcting an error the third position of matrix C = [0 0 0 1 0 0 1] must change to: C = [0 0 1 1 0 0 1].

Chapter 4.

4. Alternatives for Hamming Code

4.1. BCH Code

A BCH code is used to correct a multiple random errors, and this code is a multi-level cyclic digital error correcting code, so BCH code is powerful multiple error correction code with a mathematical properties. And the main advantage of BCH code is flexibility for block length and code rate [29] [30].

For positive pair of integer's r > = 3 and t, (n, k) BCH code is contained the parameters like:

Total bits (n) is: $n = 2^r - 1$

Number of parity or redundancy bits: n – k < = r * t

Minimum distance (which is called d_{min}) : d_{min} > = 2t + 1

Capability of correcting an error for the BCH code: t errors

And t < $(2^r - 1) / 2$ which is random error detected and corrected which also called t error correcting BCH code [29].

4.1.1. Advantages of BCH code

- BCH code has a strong capability for correcting an errors which is ability to find (detect) and correct an errors at the same time, and this is the main and most significant and substantial property of BCH codes [30] [31].

- Construction of BCH is applied without very high provisions for a computer.

- Also, a process of coding is simpler than other methods and finding an efficient method for decoding part of BCH is simple [32].

- The considerable features for the BCH codes is that the accurate and exact control exist over the number of error bits by this code, specially, it is possible to implement a BCH code (in binary) which the multiple bit errors can be corrected by this code [32].

4.1.2. Disadvantages of BCH codes

- In BCH code, when the code length increases, then the process of decoding part will be more complex [31].

4.1.3. Encoding Instructions of BCH Code

- In (15, 7) BCH encoder, for instance, if the message or the codeword contains seven bits $[m_0,\ m_1,\ \dots,\ m_6]$ are used and applied in parallel to have a serial shift register [21].

- Parity bits are computed and then send to these serial parallel shift register by using these message bits and then these parity bits are appended to the original message bits to have encoded data contains 15 bits [29] [33].

4.1.4. Decoding Instructions of BCH Code

- First of all, a syndrome (Si) where (i = 1, 2, 3, …, 2t) will be calculated from the received codeword r(x).

- At the second step of BCH decoding, the error location polynomial S(x) will be determined.
- And at the end, a root of S(x) will be found, and then the occurred error will be fixed and corrected [33].

4.1.5. Overview of BCH code design

The figure below represents the proper overview of BCH code design which contains an example, and its good pattern for implementing this code [31].

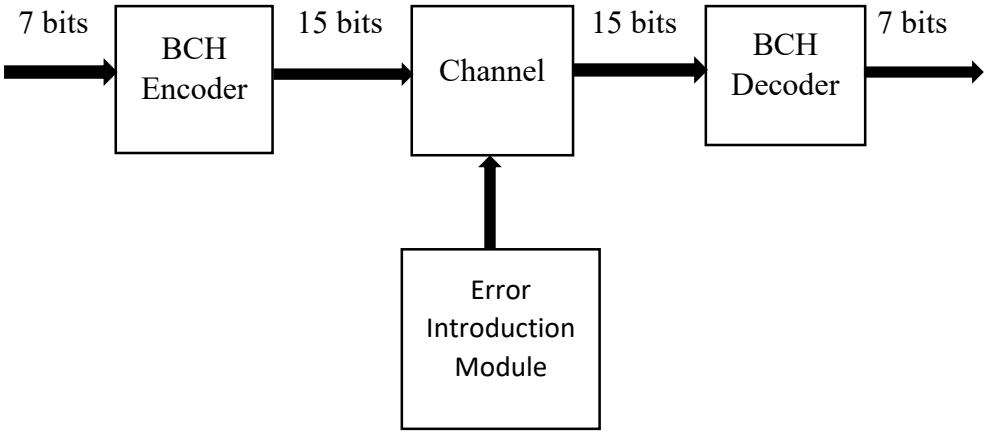

Figure 15. Overview of BCH code design [32]

In above figure the number of generated data bits are equal to seven bits, the reason of choosing seven bits is an indication of ASCII characters will be easier. Then these seven bits will be encoded by BCH encoder to create a proper codeword which is contained fifteen bits [32].

Thereupon, the generated codeword will be transmitted to the channel, and the channel might be affected by a module which is called error introduction module that creates and inject an errors to one or more bits of a codeword, randomly. Then, a codeword with one or more errors is reached to the BCH decoder [32].

In the decoding module, the syndrome formula will be calculated to find and distinguish the position where an error bits are located there. At the end, the detected errors will be corrected by flipping the error bits and at the output of the decoder, the main data bits which were generated at the first step will be retrieved [32].

In the underneath figure the structure of error introduction module is drawn and displayed.

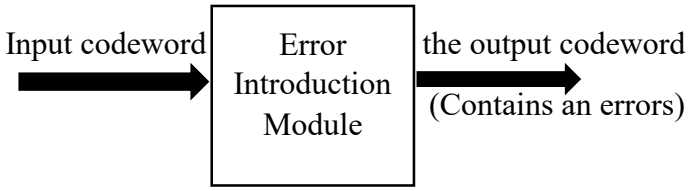

Figure 16. The module of error injection [32]

4.2. Reed Solomon Code Definition

Reed Solomon (RS) codes are another kind of error correction codes which are block-based code and the applications of RS codes are so wide and large range in digital communications and storage. And they are used to correct errors in some systems including Wireless or mobile communications, Satellite communications and etc. [8] [34].

RS code is non binary linear code and they are used for burst error correcting. A typical system of RS code is shown in the following figure:

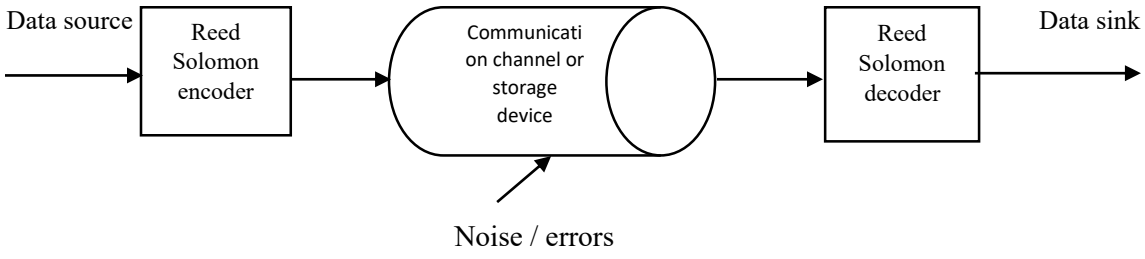

Figure 17. Typical system of Reed Solomon code [34]

So, the RS codes are written according to the following formula [35]:

$$\text{RS (n, k)} = (2^r - 1, 2^r - 1 - 2t)$$

Where k is the length of symbol, n is the length of block which $n = (2^r - 1)$, There are n-k parity symbols of s bits each. Also, up to t symbol errors can be corrected by A decoder of RS code where $t = \frac{n-k}{2}$ that contain errors in a code word, so 2t parity bits are needed to correct t errors (t is the number of true symbols) : 2t = n-k [34] [35].

As well as, the minimum distance for RS code is: $d_{min} = n - k + 1$.

For encoding and decoding parts of Reed Solomon code, applying a special mathematics called finite fields or Galois Fields (GF) are needed [4] [35]. There is a particular feature for a finite field which the specific arithmetic operations like: addition (+), subscription (-), multiplication (*) and division (/) on the finite elements will have an outcome in the field. Therefore, for this reason, the encoder and decoder part of RS code need to perform these operations, and a particular software or hardware functions are required for implementing these arithmetic operations.

The following figure shows a typical structure of RS codeword:

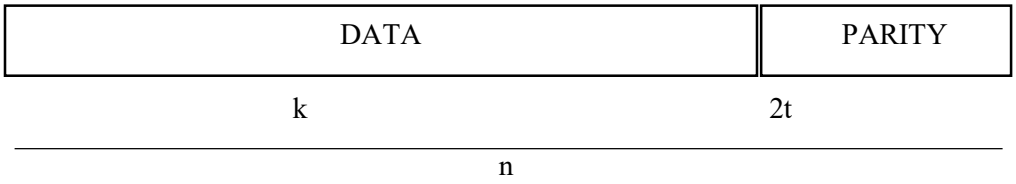

Figure 18. Typical Reed Solomon codeword structure [34]

RS code is a subset of a BCH codes and linear block codes that are contained s bit symbols specified as RS (n, k) which n is the total number of surrounded symbols (the length of the codeword) and k is the number of data. It means that the encoder catches k data symbols of s bits and then it appends the symbols correspond to the parity bits for generating n symbols which called codeword [36].

All the procedure and instructions for the encoding step of Reed Solomon code and BCH code are the same and there is no difference among them [31].

The only difference between RS code and BCH code is that in RS code there are several kinds of ways and methods for decoding part, like, iterative and Euclid algorithms [33].

4.2.1. Advantages of Reed Solomon Code

A Reed Solomon code has a special features and characteristics and because of these special positive points, RS code will be so popular and unique.

- In the Reed Solomon, code an efficient and powerful algorithm is used which is called Berlekamp Massey algorithm that is a bounded distance decoding algorithm [33].
- Also, there is a significant ability for RS code to correct both random error and burst error [34].
- A Reed Solomon code has a very high and superior coding rate, and this property is suitable for using in several applications, for instance, transmission, storage and etc. Thus because of the high coding rate of Reed Solomon code, RS code can be more effective and more powerful than hamming code for a data transmission in the communication systems channels [33] [34].

$$\text{Coding rate} = \frac{k}{n}$$

4.2.2.Disadvantage of Reed Solomon Code

- The implementation of RS code is more complex than binary BCH code [33].

4.2.3. Example of Reed Solomon Code

The most popular example for Reed Solomon code is RS (255, 233). In this example each generated codeword is contained 255 bytes of codeword that from these 255 bytes, 233 bytes are as a data symbols and 32 bytes are redundancy (parity) symbols.

In this example: RS (n, k) = $(2^r + 2^r - 1, 2^r - 1 - 2t)$ = RS (255, 233) which total bits of a prepared codeword or n = 255, data bits or k = 223, 2t = 32 then t = $\frac{32}{2}$ =16 and s = 8 is a number of bits which are as a symbol. And according to RS code, in this example the decoder is able to correct all 16 symbol errors in the received codeword, because in this dimension of RS code 16 bytes of errors can be detected in the received codeword.

Chapter 5.

5. Alternatives for error correction

5.1. N-modular redundancy and Triplication (N-modular redundancy with n=3)

To build fault tolerant system, there is a popular technique which called TMR (triple modular redundancy). In this technique there are three module units, and the output of these three units must be comprised by a voter [37].

The fault-tolerant structure which used is given in the following Figure. The result of a vote of the outputs of redundant components C1, C2 ... Cn for each input is system output where n = 3, 5, 7, … .

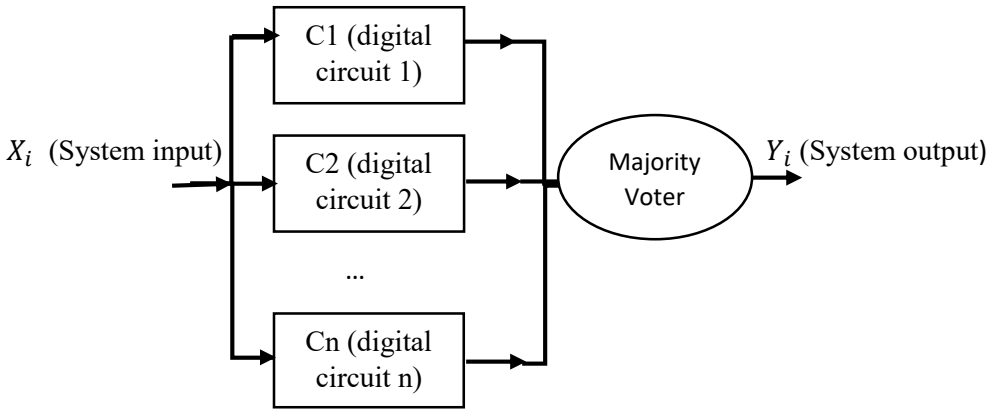

Figure 19. N-Modular redundant structure [27]

The original idea of TMR is naive and simple, a triple modular redundant or TMR R (3, 0) is a typical and usual N modular redundant example which exist three modules, without any spares. So these three modules are used with the serial voter [38] [39]. Also, for TMR in the last figure, there are three parallel modules (components) after the three parallel components a voter (serial voter) exist. And these three parallel systems indicate a process and the result will be processed by a system called voting system to generate and produce a single output. In this system, if one of these three systems fails or drop, then the error will be corrected and masked by other two systems which exist correctly [37] [39].

5.2. N version programming (NVP)

N-version (or multi-version) programming is a proposed method which can provide fault tolerance in software [28]. And NVP is a software analogy of *N*-modular redundancy which used in fault-tolerant that defined the last part and can tolerate both software and hardware faults [40].

For NVP there are some evidence which the concept of all of them are the same and they use a different algorithm in their systems, but the result must be the same.

In the structure of NVP, there are N implementation like N-versions of an application which developed separately and executed in parallel (specification of a number of versions are necessary to ensure acceptable levels of software reliability) and each version is a complete implementation of a specification and then all of their outputs will compared by decider (or voter) [39] [40].

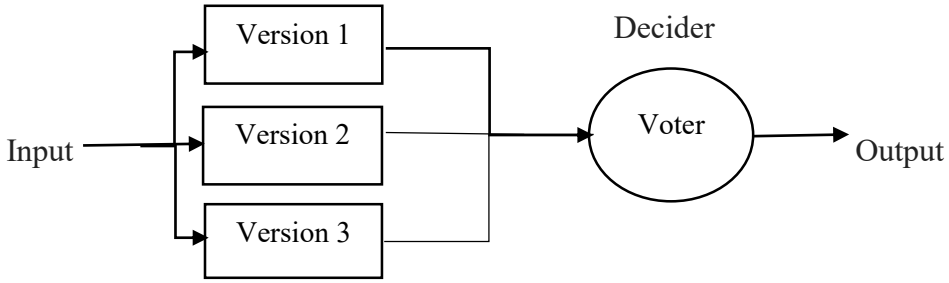

Figure 20. NVP structure (example N = 3) [39]

Chapter 6.

6. Comparative Study (Between Hamming, BCH and RS Codes)

Hamming code can only correct a single bit errors, In fact, hamming code has a capability to detect up to two errors (or maximum three parallel and adjacent errors), but can correct just one error in each message, and this inefficiency of hamming code that cannot handle to correct multiple bit errors can be compensated by Reed Solomon code which can correct more than one error and can be used on many of the current controllers, and BCH code is powerful error correcting code which can correct multiple bit errors and is going to become more popular, because their efficiency improved over RS code. Thus both hamming and RS codes are kinds and under the category of the BCH code.

The following table displays the comparative analysis among these error correcting codes:

No.	Name of error correction code	Capabilities of error detection and error correction
1.	Hamming code	Double error detection and single error correction
2.	BCH code	Multiple error correction
3.	Reed-Solomon code	Correct multiple random errors

Table 10. Comparative analysis among error correcting codes

Chapter 7.

7. Verification and Validation of Hamming code

The code of the proposed implementation has been written in C++ programming language and tested and simulated using visual studio software (version 2017). The result of the Hamming code implementation, for instance, Hamming (31, 26) is given as the following figures and for another dimension of hamming code, the same codes are replaced and just some values are changed, like the number of data bits, the number of rows in the check matrix and some other values and the result of all dimensions are working well.

7.1. Verification and Validation for Encoding part of Hamming Code

In the following screenshot of encoding part result, first a generated random binary numbers for H(31, 26) is printed out, so the input information (data) bit has a width of 26 bits (26 random binary numbers), and after that, the next steps are calculated according to this generated sequence of binary numbers, like calculation of the number of redundancy bits, the calculated parity bits and their proper positions, a calculation of total number of bits in the generated codeword which is consist of generated binary data bits

and calculation of parity bits. And at the end, the final codeword which is ready to transmit to the receiver side will be printed out in the local windows debugger, correctly.

Also, the bits of final codeword is arranged and ordered from the left side to the right side.

```
a generated random binary numbers for Hamming(31,26) are:
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1 1 0 1 0
 The no of redundancy bits to add r : 5

The proper parity bits and their positions:

1 1
2 1
4 0
8 1
16 1
Total number of bits(contain parity and data bits) is: 31

the whole bits(or codeword) for a final hamming code to transmit from TX to RX after encoding is:
1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 0
```

Figure 21. The encoding result of hamming code

7.2. Verification and Validation for Implementation of Check Matrix

Conforming with all definitions and the steps of implementation for the check matrix, the subsequent result which presents a required check matrix in the receiver side for decoding is printed out, in fact, the check matrix (H) is a binary lookup table which always using in hardware engineering and the logical circuit, but it is designed and implemented in vertical, and also the first column of this matrix which contains all zero elements is removed to reach a desired check matrix.

```
The proper check matrix (H) for Hamming (31, 26) is:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
```

Figure 22. The result of check matrix for Hamming (31, 26)

Refer to a section 7.2.2.3, in the following screen capture, the desired output is printed out for implementation of both check matrix (H) and transposed of check matrix (H^T) which is explained in the section of implementation of check matrix.

```
check matrix (H) is:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

Transposed matrix of check matrix (HT) is:

0 0 0 0 1
0 0 0 1 0
0 0 0 1 1
0 0 1 0 0
0 0 1 0 1
0 0 1 1 0
0 0 1 1 1
0 1 0 0 0
0 1 0 0 1
0 1 0 1 0
0 1 0 1 1
0 1 1 0 0
0 1 1 0 1
0 1 1 1 0
0 1 1 1 1
1 0 0 0 0
1 0 0 0 1
1 0 0 1 0
1 0 0 1 1
1 0 1 0 0
1 0 1 0 1
1 0 1 1 0
1 0 1 1 1
1 1 0 0 0
1 1 0 0 1
1 1 0 1 0
1 1 0 1 1
1 1 1 0 0
1 1 1 0 1
1 1 1 1 0
1 1 1 1 1
```

Figure 23. The result of check matrix (H) and transposed of check matrix for Hamming (31, 26)

7.3. Verification and Validation for Decoding part of Hamming Code

The verification and validation of this part is totally depended on applying the framework and error injection part, but all the codes which are written and appended to the decoding part are tested and compiled in another file of visual studio by inserting

some extra variables and printing some output for checking the working reliability of this part of code, and all functions are worked, correctly.

In this part, an error will be injected on the desired position in the transmitted codeword and then the syndrome will be calculated to find a position where an error is occurred, then the error will be corrected with flipping the position of this error. And at the end of the code, an initial generated data bits which were generated by a source in the sender side will be printed out in the output of decoding part.

Chapter 8.

8. Direction for Future

In the hamming code method which is a simpler system of error detection and error correction code that uses a special bits called parity or redundancy bits to correct an error or bit changing [28] [44], the main and considerable drawback of hamming code which can be improved in the next research in the future work is extending the capability of error correction from the single bit error correction in hamming code to the larger number of error correction by using another codes or methods, like BCH, Reed Solomon (RS) and some other designed codes, there are several kind of systems for correcting an error(s) in a data bits which are more efficient than hamming code, because the hamming code is limited to correct just one bit error or bit changing and multiple errors and burst error are not supported by this error correction code, and this inefficiency of hamming code can be improved and compensated by these alternative methods.

Chapter 9.

9. Conclusion

In this thesis project, the structure of hamming code and the evaluation of its construction are tested and examined successfully and the implementation of hamming code is done in the software environment of Visual Studio 2017 and C++ language.

From the content of this report, it is clearly indicated that how to detect an error and bit changing in the received sequence of bits (codeword) by checking the value of parity or redundancy bits. If the bits of a codeword in the sender side are not the same and matched with the value of a received sequence of bits in the receiver side, it shows the received codeword has an error and if the values are matched means the received codeword or signal has no error.

After detecting an error, the bit containing an error will find by adding the sequence numbers of parity bits which calculation of the value and the position of each parity bits has specific procedure which is explained before in encoding part of this report, and on the receiver side, to detect an error, the more significant point is check matrix (H) and transposed of check matrix (H^T) which is used in the syndrome calculation for multiplying with received codeword, the result shows the sequence number contains an error and incorrect bit. After finding the sequence number of the incorrect bit, the detected error must be corrected by changing the value of a wrong bit.

References

[1] HAMMING, R. W. “Error Detecting and Error Correcting Codes.” Bell System Tech. Jour., 29 (1950): 147-160.

[2] Aydin Nuh, "An Introduction to Coding Theory via Hamming Codes: A Computational Science Model" (2007).

[3] Hamming, Richard Wesley (1950). "Error detecting and error correcting codes". Bell System Technical Journal. 29 (2): 147–160.

[4] Isaac Woungang, Sudip Misra, Subhas Chandra Misra, “Selected Topics in Information and coding theory” Shetter, W. Z. 2000. This essay is redundant.

[5] Golay, M. J. E. 1949, “Notes on digital coding”, Proc. IRE. 37: 637

[6] A. Ahmadpour, A. Ahadpour Sha, M. Ziabari, “A novel formulation of Hamming Code”, IEEE Trans. Inf. Theory, 26 June 2009.

[7] William Rurik, Arya Mazumdar**, “**Hamming codes as error-reducing codes”,IEEE Trans. Information Theory Workshop (ITW), 27 October 2016.

[8] Gregory, Mitchell, “Investigation of Hamming, Reed-Solomon, and Turbo Forward Error Correcting Codes.” ARL-TR-4901, July 2009.

[9] Jonathan I. Hall, “Notes on coding theory”, 9 September 2010.

[10] Sreelatha P, P Pradeep Kumar, S V Mohankumar, “A Lab VIEW Based Extended (10, 5) Binary Hamming Code Generator for Telecommanding Applications”, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-4, Issue-2, May 2014.

[11] Yaqi Wang, Jun Lin, Zhongfeng Wang "A New Soft-input Hard-output decoding algorithm for Turbo Product Codes," IEEE Trans. Inf. Theory, pp. 192-197, 06 August 2002.

[12] R. Gallager, "low-density parity-check code", IRE Transactions on Information Theory, Volume: 8, Issue: 1, January 1962.

[13] Xiaoling Huang, N. Phamdo, Li Ping, "Recursive method for generating weight enumerating functions of trellis codes", IEEE Trans. Inf. Theory, pp. 773-774, Vol. 37 No. 12, 7th June 2001.

[14] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," Proc. ICC'93, Geneva, Switzerland, May 1993, pp. 1064-1070.

[15] Orhan Gazi, Ali Özgür Yılmaz, "Turbo Product Codes Based on Convolutional Codes", ETRI Journal, Volume 28, Number 4, August 2006.

[16] Franco Chiaraluce, Roberto Garello, "Extended Hamming Product Codes Analytical Performance Evaluation for Low Error Rate Applications", IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 3, NO. 6, NOVEMBER 2004.

[17] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. Inform. Theory, vol. 20, pp. 284–287, Mar. 1974.

[18] F. J. MacWilliams and N. J. A. Sloane, "Theory of Error-Correcting Codes", Vol. I, North-Holland, 1977.

[19] Habong CHUNG, Hwanseok JANG, Jinwoo SEONG, "Sufficient Conditions for an (n, 1) Mother Code and Its Puncturing Pattern to generate a Given Convolutional Code", IEEE Trans. Inform. Theory, 14-18 Sept. 2015

[20] Ravindra M. Deshmukh and 2 S.A. Ladhake, "Analysis of Various Puncturing Patterns and Code Rates: Turbo Code", International Journal of Electronic Engineering Research ISSN 0975- 6450 Volume 1 Number 2 (2009) pp. 79–88.

[22] Wirda Fitriani, Andysah Putera, Utama Siahaan, "Single-Bit Parity Detection and Correction using Hamming Code 7-Bit Model", International Journal of Computer Applications (0975 – 8887), Volume 154 – No.2, November 2016.

[23] Behrouz A Forouzan, "Data communication and networking", 4th edition Tata McGraw-Hill Publication

[24] A. K. Singh. "Error detection and correction by hamming code." In Proc. Information Computing and Communication (ICGTSPICC) 2016 Int. Conf. Global Trends in Signal Processing, pages 35-37, December 2016.

[25] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes", IEEE Trans. Inform. Theory, vol. 47, pp. 638–656, Feb. 2000.

[26] Stefan Scholl, Norbert When, "Efficient architectures for parity check matrix generation" Signal Processing and Information Technology (ISSPIT) 2016 IEEE International Symposium on, pp. 280-285, 2016.

[27] Barry, Paton, (March 1998 Edition). Fundamentals of Digital Electronics (pp. 2 - 3). Dalhousie University.

[28] Debalina Roy Choudhury, Krishanu Podder, "Design of Hamming Code Encoding and Decoding Circuit Using Transmission Gate Logic", IRJET, volume: 02, Issue: 07, Oct. 2015.

[29] D. Augot, P. Charpin, and N. Sendrier, "Studying the locator polynomials of minimum weight code words of BCH codes", IEEE Trans. Inf. Theory, vol. 38, no. 3, pp. 960–973, May 1992.

[30] W T Penzhorn, "A Fast Algorithm for the Decoding of Binary BCH Codes", Dept. of Electr. & Electron. Eng., Pretoria Univ., South Africa, Aug. 1993

[31] Zhang Xinyu, “A basic research on Forward Error Correction”, IEEE Trans. Inf. Theory, pp. 462-465, 08 September 2011.

[32] Eshtaartha Basu1, Dhanush.P2, Kishor S3, Geethashree A4, “ERROR CONTROL CODING USING BOSE-CHAUDHURIHOCQUENGHEM (BCH) CODES”, International journal of electronics and communication engineering and technology (IJECET), Volume 5, Issue 8, pp. 86-96, August (2014),

[33] D. Augot, P. Charpin, and N. Sendrier, “Studying the locator polynomials of minimum weight code words of BCH codes,” IEEE Trans. Inf. Theory, vol. 38, no. 3, pp. 960–973, May 1992.

[34] Sanjana P. Choudhari and Megha B. Chakole, “Reed Solomon Code for WiMAX Network”, IEEE Transaction on information theory, April 6-8, 2017.

[35] J. Chen and P. Owsley, “A burst error correcting algorithm for Reed Solomon codes,”, vol. 38, IEEE Transaction on information theory, Nov.1992, pp. 1807-1812.

[36] C. Shannon, “A mathematical theory of communication,” Bell System Technical Journal, vol. 27, pp. 2, 1948.

[37] Masashi Hamamatsu, Tatsuhiro Tsuchiya, Tohru Kikuno, “On the Reliability of Cascaded TMR Systems”, IEEE Transaction on information theory, pp. 184-190, 28 January 2011.

[38] Umar Afzaal, Jeong A Lee, “A Self-Checking TMR Voter for Increased Reliability Consensus Voting in FPGAs”, IEEE Transaction on information theory, 09 April 2018.

[39] Banki H., Babamir S., Farokh A., Morovati M., "Enhancing Efficiency of Software Fault Tolerance Techniques Satellite Motion System", Journal of Information Systems and Telecommunication, Vol. 2, September 2014.

[40] Liming Chen; Avizienis, A. “N-Version Programming: A Fault-Tolerance Approach to Reliability of Software Operation, Fault-Tolerant Computing.”, 1995, ' Highlights from Twenty-Five Years'., Twenty-Fifth International Symposium on, Vol., Iss., 27-30 Jun 1995.

[41] Karla Steinbrugge Chauveau, “Working with C++”, IEEE Trans. Inf. Theory, pp. 192-197, 06 August 2002.

[42] Leendert Ammeraal, “C++ for Programmers”, John Wiley & Sons Ltd., England, 1991.

[43] Naba Barkakati, "Object-Oriented Programming in C++", SAMs, Carmel, IN, 1992.

[44] Z. Ramadhan and A. P. U. Siahaan, "Stop-and-Wait ARQ Technique for Repairing Frame and Acknowledgment Transmission," International Journal of Engineering Trends and Technology, vol. 38, no. 7, pp. 384-387, 2016