Performance Analysis of Different Error Correction Codes

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering

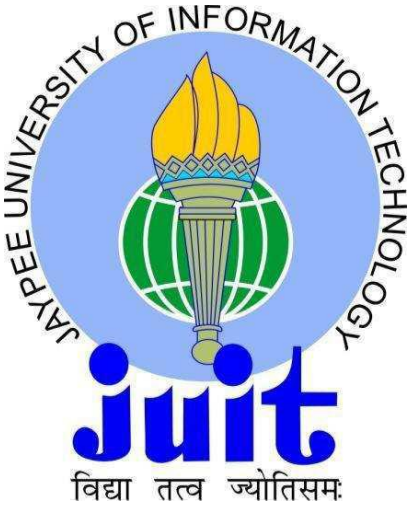
By

Lavanya Kumar (141264)

Under the supervision of

(Dr. Amit Kumar Singh)

To



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled **Performance Analysis of Different Error Correction Codes** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2017 to May 2018 under the supervision of **Dr. Amit Kumar Singh** (Assistant Professor, Senior Grade, Computer Science & Engineering Department).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Lavanya Kumar, 141264

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Amit Kumar Singh

Assistant Professor (Senior Grade)

Computer Science & Engineering Department

Dated:

ACKNOWLEDGEMENT

I owe my profound gratitude to my project supervisor **Dr. Amit Kumar Singh**, who took keen interest and guided us all along in my project work titled **—Analysis of Error Correction Codes**, till the completion of my project by providing all the necessary information for developing the project. The project development helped us in my research and I got to know a lot of new things in my domain. I am really thankful to him.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

ABSTRACT

The issue of having errors is always there when we communicate or send some data because of usual reasons such as jitter, noise, etc. These errors can be on storage devices due to sector's corruptions on the drive or manufacturing defects which may lead to loss of data. There are different error correcting techniques that are available for commercial use as well as for experimental applications. But there are some specific requirements or necessities for different implementations due to which finding an appropriate Error Correcting Code becomes a challenge and you need to find the best suitable code using some performance parameters (like we have used BER). That is why in this project, we want to do a comparative analysis of different ECCs to generate a table that concludes the codes' performance at a place thereby making it easier to choose from different options of codes according to the requirements.

Chapter 1 covers the introduction to error correcting codes, motivation behind the work that why this project was chosen, problem statement explaining what will be done and the methodology explaining how it will all be done.

Chapter 2 covers the literature review where various research papers have been studied telling us what are the various error correction codes, what their algorithms are, how they are implemented and what special requirements are there.

Chapter 3 covers the implementation of various techniques that we have chosen, explaining the encoding and decoding part of the techniques.

Chapter 4 comprises of the proposed diagram and the software requirements related to the project.

Chapter 5 includes the performance analysis part of the various techniques used with the help of Bit Error Rate(BER).

Chapter 6 concludes the observations collected in the project and discusses more about future scope and applications.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO ERROR CORRECTING CODES

Innovation in this day and age has dependably been a major piece of our lives. Regardless of how little or huge it will be, despite everything it has a major effect particularly in interfacing us regardless of whether we are not at a similar place. For us to feel fortified regardless of whether we are inaccessible, it assumes an indispensable part. In any case, on the off chance that we attempt to look behind this whole innovation, we may discover a great deal of information handling and that excessively complex ones. Since information is included, mistakes will likewise be there. To discover them and right them isn't that simple as it shows up. To utilize an ideal system to do as such is a most extreme need in these sorts of situations. That is precisely where its characterization comes as two sorts of these codes.

The error correction codes can be of mainly 2 types:

- Error detection
- Error detection and correction

In error identification, codes for the most part centre around identifying whether there has been error made or change in the bits amid their transmission on a channel. Distinctive codes tell that whether there has been an adjustment in the bits or not, the recognition of bits relies on the sort of code utilized for encoding the message. In error discovery and association, codes are substantially more progressed than the recognition one, they are fit for recognizing and revision errors in bits or change in bits made amid transmission, the quantity of bits that can be remedied by these codes relies upon the sort and multifaceted nature of code used to encode the information before transmission. In this venture we have

concentrated on various mistake adjustment strategies actualized around the innovative world, in various ways. We have contemplated numerous blunder amendment coding plans, their calculation, their execution methods and will analyze their execution in view of BER (Bit error rate). We will draw out a near report for constant applications likewise like how they will deal with information moving in field of live information gushing, where redress isn't middle of the road.

We will analyze different calculations and what can be there future uses in various forthcoming fields and which strategy may be the appropriate for various fields from the arrangement of techniques we have. Like in medicinal practices or saving money area, putting away critical and classified information is the most extreme need where we can't hold up under any misfortune or mistake.

1.2 **MOTIVATION**

We generally discover errors when we convey or exchange electrical signs; these are because of numerous reasons like jitter, clamour, and free associations. Information mistakes are likewise away media gadgets because of part's debasements on the drives or assembling deformities, and which frequently brings about information defilement or miss elucidation of information. Be that as it may, there is an answer of each issue so does it. There are blunder revision techniques which have created over the time and have become progressed as far as innovation with time. These codes simply take up somewhat more information space contingent on the sort or the degree to which we need information to be solid. They help in recognizing information blunder rapidly and now and again even right information bit(s), which is the reason exactly to transfer of touchy and solid information we utilize these information mistake amendment procedures. By examining mistake discovery and amendment strategies we can discover better approaches for how we can execute these procedures in various territories of information exchanging and analyze some mainstream blunder identification and redress methods in view of what number of bits they can recognize, revise, how much space taken and what is its reaction to BER (Bit error rate).

1.3 **PROBLEM STATEMENT**

In our real life scenarios we use these ECCs all the time without much of an issue, but for all those implementing some kind of network or system, this problem always arouses that what should be the optimum course of action for their particular requirements. Finding a suitable, appropriate ECC is tough selection. To make it easier is exactly what we are trying to accomplish here. After the selection, that code will be used in hospital systems, intelligence network systems, etc.

1.4 **OBJECTIVES**

Implement different error correction codes algorithms and compare the performance of these ECC based on different parameters to find the best possible type of ECC can be used in any scenario. Our performance parameters include, the space complexity of the algorithm, how much extra bits it takes to encode the code, what is its time complexity, that is how much time it takes to encode and decode the certain bits of data. Mainly our focus is on the BER (Bit Error Rate), how much BER can a certain code handle and, BER is based on how much noise is there on a certain channel and how many bits can be altered by those bits. Explore possibilities where these codes or techniques can be implemented best in the real life. We aim to create a well-defined table or list to provide with the best possible solution possible for implementation.

1.5 **METHODOLOGY**

Our project is based on the water fall model, in which we have gone step by step, first we have done literature survey of some of the popular error correction codes published earlier, then we handpicked a few of them and implemented them, and after that we have analysed them for different parameters and situations.

CHAPTER 2

LITERATURE REVIEW

Error correction codes are complicated as they depends on many factors for their performance, like clock cycles taken for encryption and decryption. Extra space they take on channel after encryption, and lastly the reliability of the code to the channel noise transmission, that whether the code is able to recover the data encrypted for reliable transmission or it is only able to recover only as partial amount of the data or not even a significant amount of data after transmission.

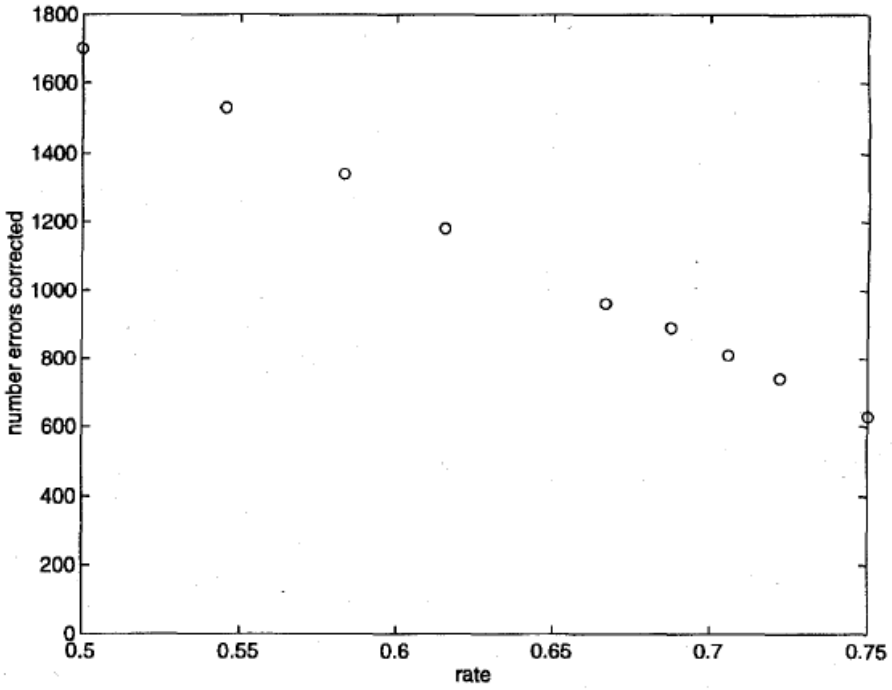
2.1 SUMMARY: SOME EXISTING TECHNIQUES

In [1], the authors have proposed a method for Chinese remainder codes that depend on the development rule got from Chinese remaining portion hypothesis which is a piece of the ring hypothesis. This hypothesis takes help of idea of powerless square plan and double feeble piece outline because of the way that number of components in a piece require not be excessively consistent or rise to, so its frail square plan idea encourages us in making a mistake amendment code more effectively than a general piece outline. In this paper they have connected utilizing a powerless piece outline and built complex codes utilizing this plan. Sun Zi Codes [12] were utilized by them as a source of perspective to build the complex code and to comprehend there ideas. In this technique they essentially interface direct blunder remedy codes and powerless piece plan which is taken as the fundamental establishment of this code. They utilized the idea that if a frail piece configuration can be developed that the comparing straight mistake adjustment code can be additionally built effortlessly, and inverse of this is likewise conceivable that is whether we have a direct blunder redress code then we can build the feeble square plan of that code. So they can state that we can consider feeble piece plan properties by the straight codes and competes a versa. From the purpose of usage it is anything but difficult to develop a straight code

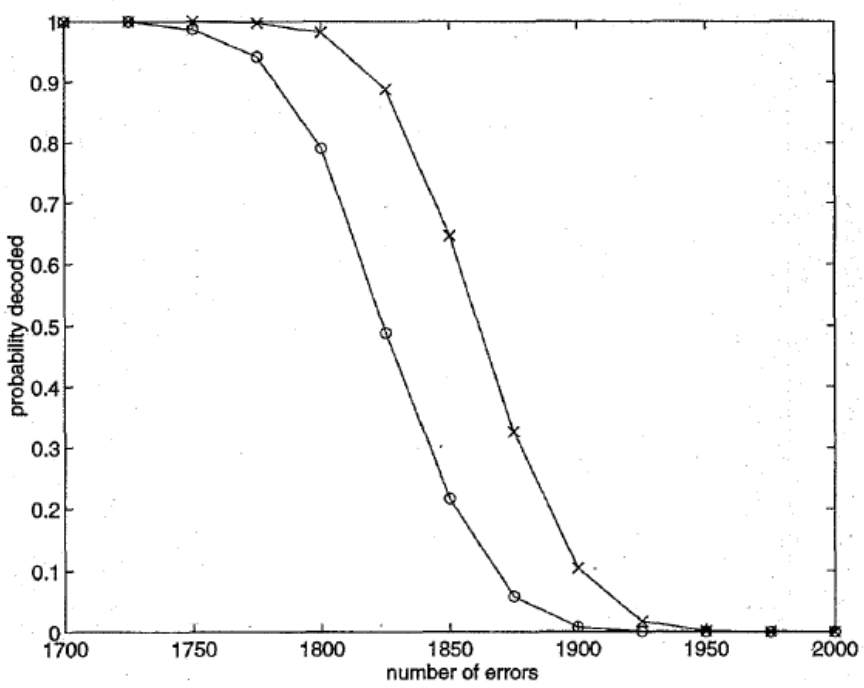
and after that establish that whether a code with these parameters exist or not. Zhang [13] in 1999 presented an idea leftover portion codes named as Chinese leftover portion codes. These codes depended on the ideas of powerless piece outline of the Chinese leftover portion hypothesis. Sun Zi codes are a unique instance of Chinese leftover portion codes, and Chinese leftover portion codes have considerably more broad logarithmic structure than Sun Zi codes. In this paper they have summed up codes sum up Sun Zi codes as a general commutative ring with a personality component from the number ring. Distinctive feeble piece outlines can be developed or acquired with generally prime thoughts and commutative rings, and determine another class of powerless square plans with an assortment of straight mistake rectifying codes. Chinese remainder codes development can be joined with the techniques for utilizing relative geometry in mix with the projective geometry to build new sorts of direct codes and blunder amending codes.

In [2], the authors have proposed a method for expander codes where they have built up another group of asymptotically great, straight error adjustment codes in view of the idea of expander diagrams. These codes are exceedingly productive as they have straight time successive deciphering calculation and logarithmic time parallel unravelling calculation which just uses a direct number of processors. They have additionally demonstrated the trial consequences of arbitrarily made codes and indicated how great they perform, in actuality, application. In this paper they have made an amazing failure – thickness – equality check code presented by Gallager [14], these are totally in view of expander charts so they are named as expander codes, these codes can be effortlessly decoded in direct time, in the development of these codes as recommended by Gallager [14] they have utilized contiguousness framework of an arbitrarily picked low-degree bipartite diagram as the equality check lattice of an error– remedy code. Gallager appeared in his paper that when a code is actualized by this approach then this kind of code has a rate and least separation most likely close to the Gilbert-Varshamov bound. They have

developed straight codes over letters in order {0, 1}, essentially parallel digits yet it is anything but difficult to grow it or sum it up to bigger fields. By taking a code of square length n and rate r, by definition a code which has n images and rn are the message images which can be specific or irregular. And afterward if a code has least relative separation a then each combine of words in the code varies in at any rate images. In the beneath figures they have thought about the execution version of the successive deciphering calculation in which they have actualized negative advance flips with an adaptation in which they didn't. They have chosen a ½ rate expander code which is of length 40000. At that point they have performed 2000 tests for every calculation and every last error in it, and after that tallied how often the errors were successfully amended.



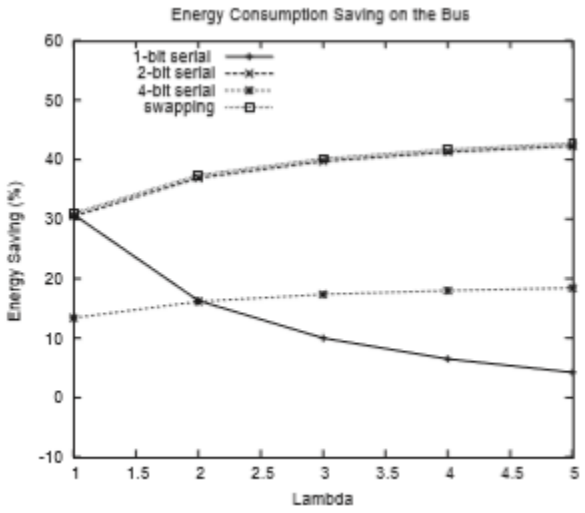
Graph 1 - the point in the upper left-hand corner of the figure indicates that an expander code of rate 1/2 corrected all of the 50000 patterns of 1720 errors on which it was tested.



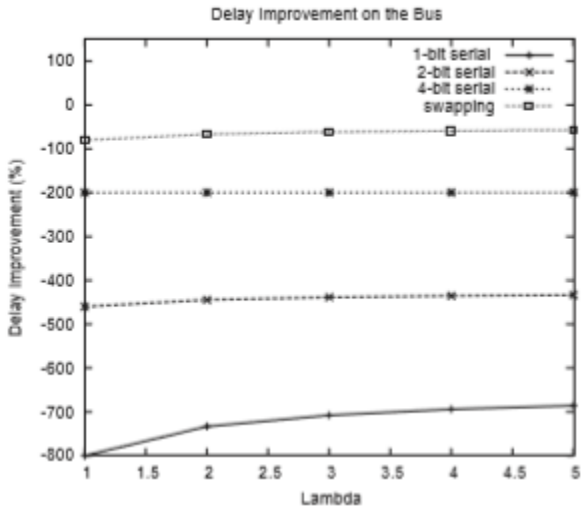
Graph 2 - a rate 1/2 expander code of length 40000. The points marked by ‘o’ indicate the probability that the standard sequential decoding algorithm corrected a given number of errors. The points marked by x indicate the probability when the algorithm was allowed to flip up to 700 variables that appeared in one more satisfied than unsatisfied constraint.

In [3], the authors have proposed a method for serial error correction code in which the primary issue which excites the need of this exploration paper is crosstalk on the framework transport which really alludes to a marvel where an adjustment in a VLSI circuit or wires makes or rather initiates an impact in adjacent neighbouring circuits or wires on account of the after effect of capacitive coupling. Since these days, the thickness of VLSI circuits is expanding step by step; this crosstalk issue gets greater and greater. As it impacts vitality, postponement and unwavering quality of the framework for transmission of codes, it turns out to be much greater worry to determine this. Some coding plans [15],

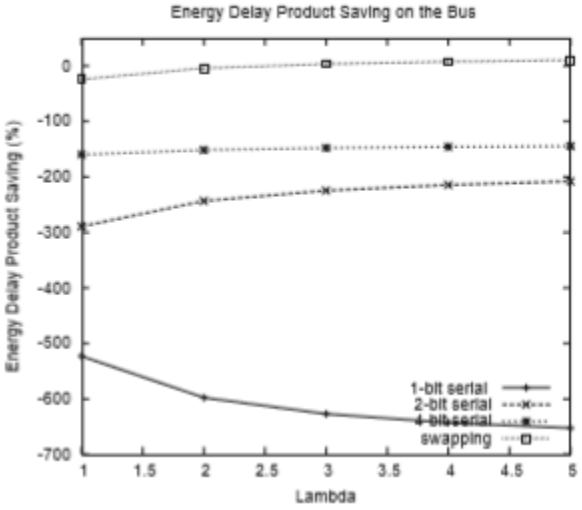
[16], [17] were organized and assembled particularly to limit the vitality and a few plans [18], [19], [20] for postpone lessening as it were. Be that as it may, every one of them were based on a parallel transport structure. In any case, to keep away from crosstalk from the root specifically as opposed to going for its distinctive effects independently, an aggregate plan was required which was found in serialized structure. Utilizing this serial strategy, an ECC is proposed to diminish crosstalk and its effects. This coding plan begins from Hamming Code utilizing its natural property which winds up in integrating a line swapping method. After the development of this code and its circuit execution, execution investigation is improved the situation a 8-bit transport and 16-bit transport and the discoveries demonstrate that they have prevailing with regards to diminishing the vitality and postponement. Given underneath is the examination for various serial piece plans utilizing a diagram to demonstrate the reduction in the parameters in figure 3, 4 and 5. After the execution examination, it ended up obvious that they prevailing with regards to accomplishing the true objective or fundamental goal for thinking about this issue articulation.



Graph 3 – Energy saving with respect to (8,4) Hamming code



Graph 4 – Delay saving with respect to (8,4) Hamming code



Graph 5 – Energy-delay-product saving with respect to (8,4) Hamming code

In [4], the authors have proposed a method for exponential Golomb and Rice error correction codes. Before this proposed code appeared openly, there were a few codes that were as of late proposed like UEC (Unary Error Correction) and EGEC (Elias Gamma Error Correction) codes which were doing likewise sort of work that propositions codes will do however there were a few inadequacies. They were working at close limit yet the images from expansive arrangement of letters in order were chosen at a low unpredictability. They worked for a constrained scope

of image. That is the reason the need of a substantially more extensive and adaptable coding plan excited. In this paper, they expand both UEC and EGEC and consolidate them together to frame a class of Rice and Exponential Golomb ECCs which even have the capacity of utilizing the image esteems from H.265 codec likewise which takes after Zipf's law that expresses that given a substantial example of words utilized, the recurrence of any word is contrarily relative to its rank in the recurrence table. In the event that this broadened code gets actualized or gets for all intents and purposes useful, it won't just permit a persistent stream of image esteems yet additionally upgrade them and moreover the encoding and translating of even huge arrangements of letters in order should be possible just by utilizing settled length framework parts.

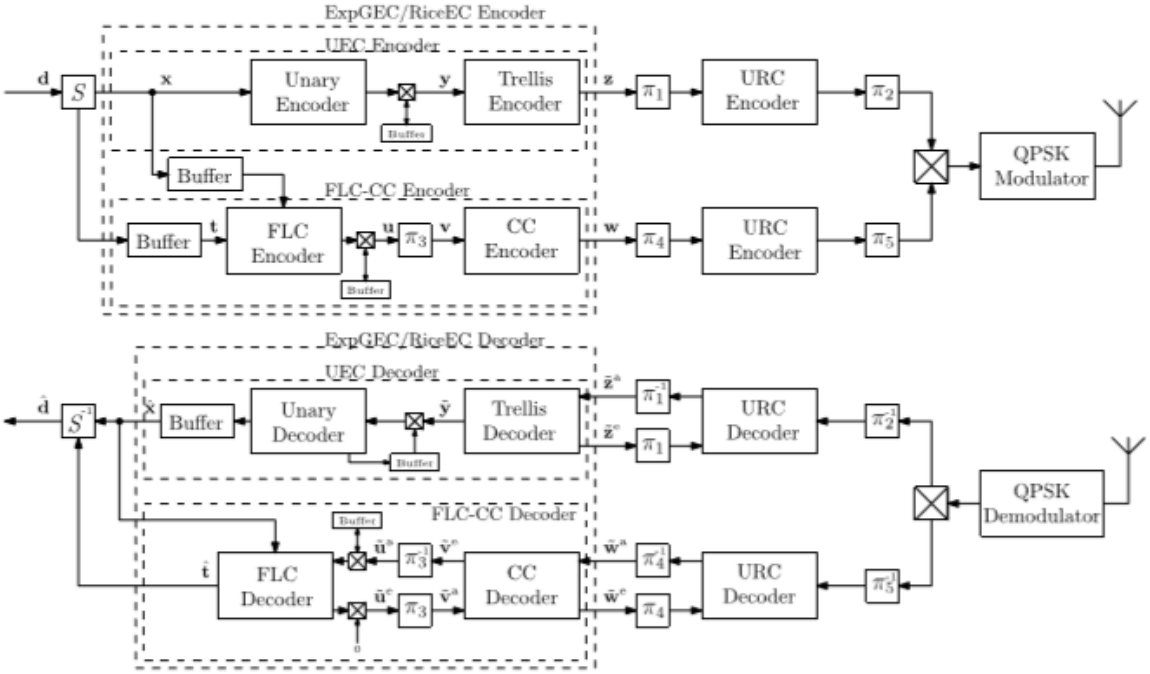


Figure 1 – Proposed ExpGEC/RiceEC schemes

The above diagram shows the proposed ExpGEC/RiceEC schemes. Here the buffers facilitate operation on the basis of a stream of source symbols, while maintaining fixed-length designs for the interleavers $\pi_1 - \pi_5$. The proposed code

not only provides wider range and good flexibility but also showcase the same error correction capability as well as transmission-energy, transmission-duration and transmission-bandwidth as showcased by best of the several state-of-the-art benchmarkers.

In [5], the authors have proposed a method for reordered Elias gamma error correction code. Like the past paper, the plan proposed here likewise expands the Unary Error Correction code (UEC) and Exponential Gamma Error Correction code (EGEC). Be that as it may, on the off chance that we look at the proposed conspire and these two coding plans, we find that image esteems are chosen or picked indiscriminately with the assistance of a monotonic likelihood conveyance. Moreover, EGEC (Exponential Gamma Error Correction code) has an alternate synthesis, an entangled one including two pieces where it is needed unequal mistake security with a specific end goal to kill or adjust the two pieces. This is finished with the assistance of a particular parameterization that must be joined with the essential information/source circulation. Rather than that, Reordered Elias Gamma Error Correction code (REGEC) needs to adjust the two pieces as it doesn't have two of them. There is just a single part. As a result of that it doesn't experience from any deferral and synchronization issue that occurs with the EGEC code. Amid execution examination stage under a particular down to earth situation the coding plan depicted in this paper demonstrates a pick up of up to 0.9 db with the regard to correlation finished with the best of JSCC (Joint Source Channel Coding) and SSCC (Separate Source Channel Coding) benchmarkers. The additions here are not accomplished by causing any extra cost that is without expanding the important transmits' span, data transfer capacity and vitality or unraveling unpredictability.

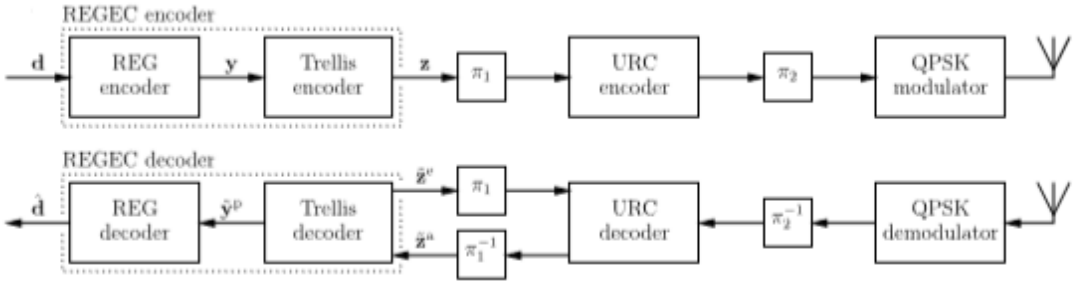
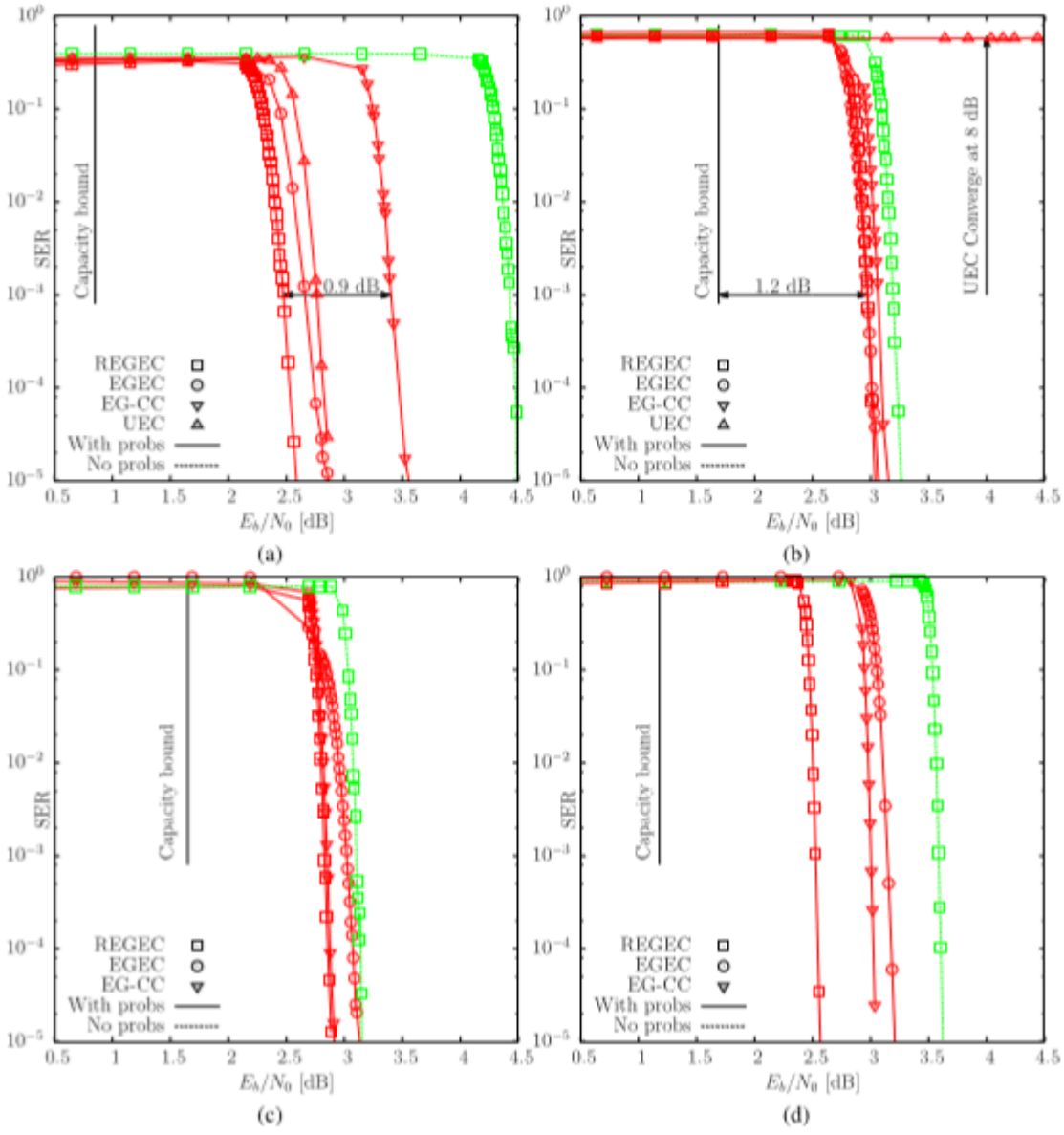


Figure 2: Schematics of REGEC code



Graph 6 - SER performance of REGEC scheme with UEC, EGEC and EG-CC Benchmarkers

In [6], the authors have proposed a method for variable data length error correction codes. With the expanding extensive scale incorporation that is coordinating as much as circuits on little region of chip with forceful scaling of working voltage to reach to a higher thickness and lower control have enormously diminished the unwavering quality of on-chip memory which implies more possibility or likelihood of mistake is there. We have been utilizing ECCs in a wide range of recollections from quite a while now to give an equal security to every one of the information put away inside the memory. Yet, since the ECCs utilized have a settled information length with which they deal with the entire information. In the event that there are blunders in different bits, at that point they need to work additionally expanding the overhead because of rehashed encoding and translating of rationale and equality bits. This is precisely where VL-ECC code proposed in this paper comes into the photo where information length of ECC will be powerfully changed (i.e. at the run time) in like manner to ensure higher need bits first. It is executed in a way which naturally arranges its information length when it sees number of blunders surpassing the mistake remedy limit of the blunder rectification code. For the most part it is fairly diminished to give careful consideration to just high need bits, along these lines lessening the corruption of value because of various piece blunders surpassing the greatest limit.

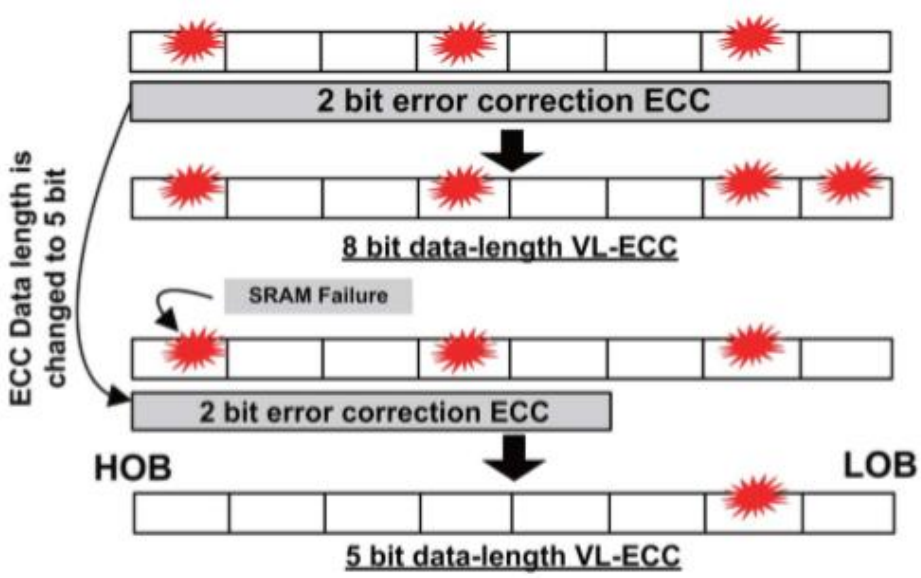
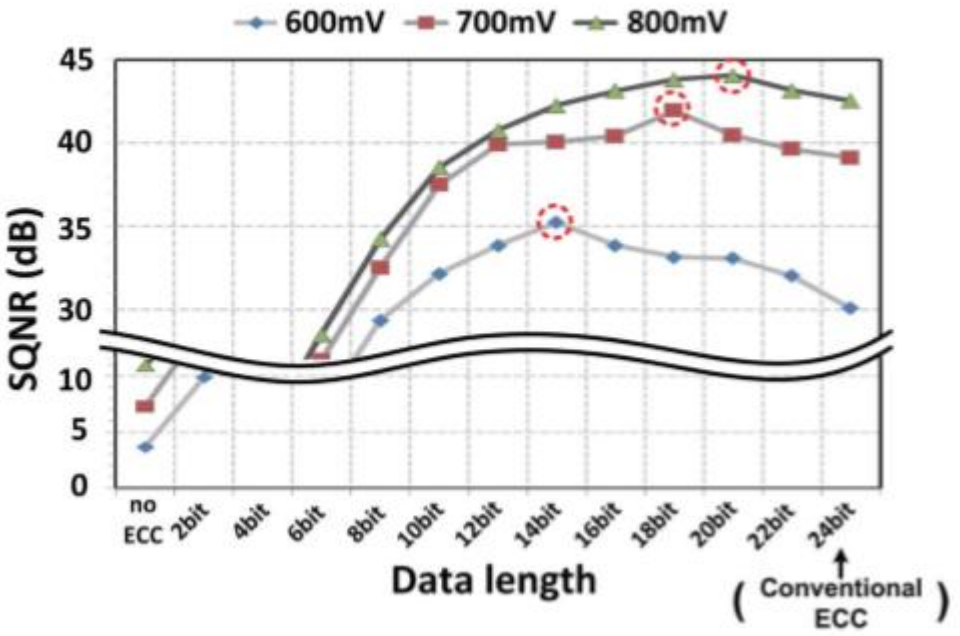


Figure 3: Overview of the proposed VL-ECC approach



Graph 7 - SQNR plots of the FFT processor outputs when VL-ECC is used with varying data length

In [7], the authors have proposed a method for forward error correction code for optical signs. This plan utilizes the properties of convolutional code alongside FWM (Four Wave Mixing) in a HNLF (Highly Non-Linear Fiber) to figure an ECC method in correspondence to SNR (Signal to Noise Ratio) between the diverse hubs in an optical system.

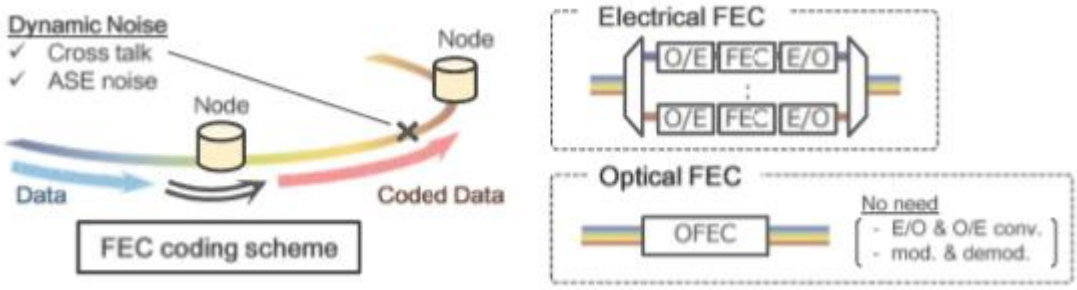


Figure 4 - Conceptual image of all-optical FEC Coding Scheme for dynamic noise compensation technologies

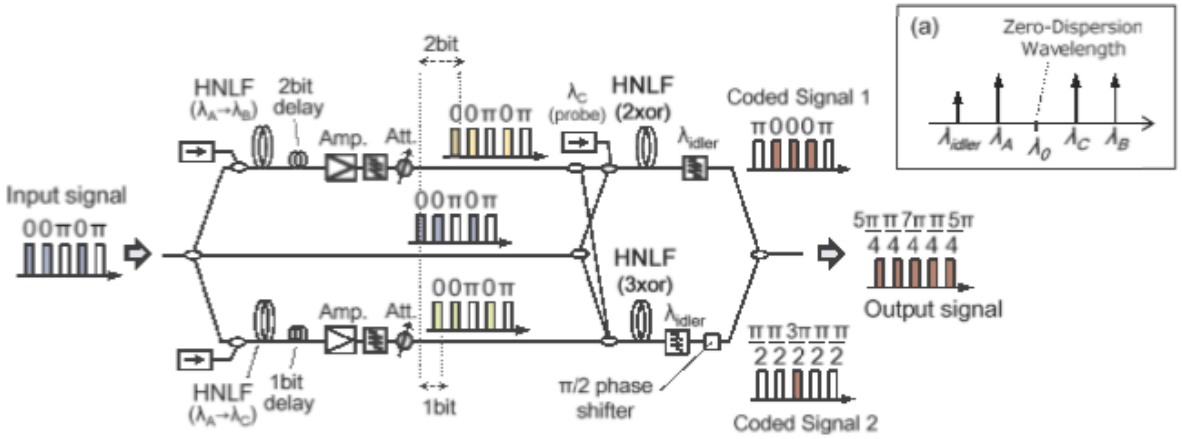


Figure 5 - Diagram of the optical FEC coding circuit with $(5, 7)_8$ convolutional code.

In this paper they have numerically assessed the execution of optical XOR activities for various mixes of all the distinctive parameters in regards to a flag like fiber lengths, flag wavelengths and flag powers. Amid the examination, the discoveries likewise showed this Forward blunder redress code gives high calibre and passableness while performed on contribution amid an activity.

In [8], the authors have proposed a method for BCH code where they make an execution investigation of Bose-Chaudhari-Hocquenghem (BCH) code by utilizing Bit Error Rate (BER) and Signal Energy to clamour control thickness proportion (Eb/No) as the execution parameter. They are utilizing two codes here – BCH (15, 7, 2) and BCH (55, 231, 3). Segment II includes foundation of BCH code. This area discusses BCH Encoder and enlightens us concerning what ought to be the piece length, what number of equality check bits ought to be there and what ought to be the base separation. The majority of the Error revising codes that are from the class of effective cyclic codes work significantly on intense arithmetical structures called limited fields, otherwise called Galois fields after Pierre Galois. So to accomplish the BCH code, we first need to manufacture the group of Galois field.

After that BCH decoder is categorised into three steps:

- Computation of syndromes

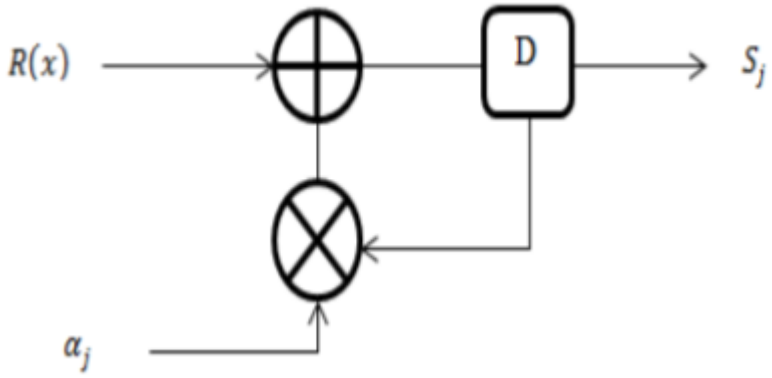


Figure 6 - Basic syndrome calculator cell

- Berlekamp-Massey algorithm
- Detection of error position using Chien Search block

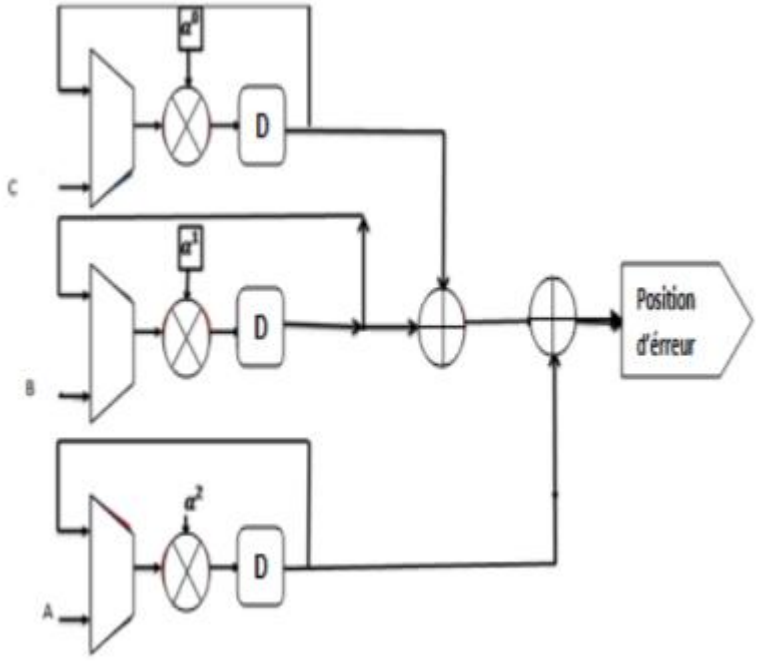
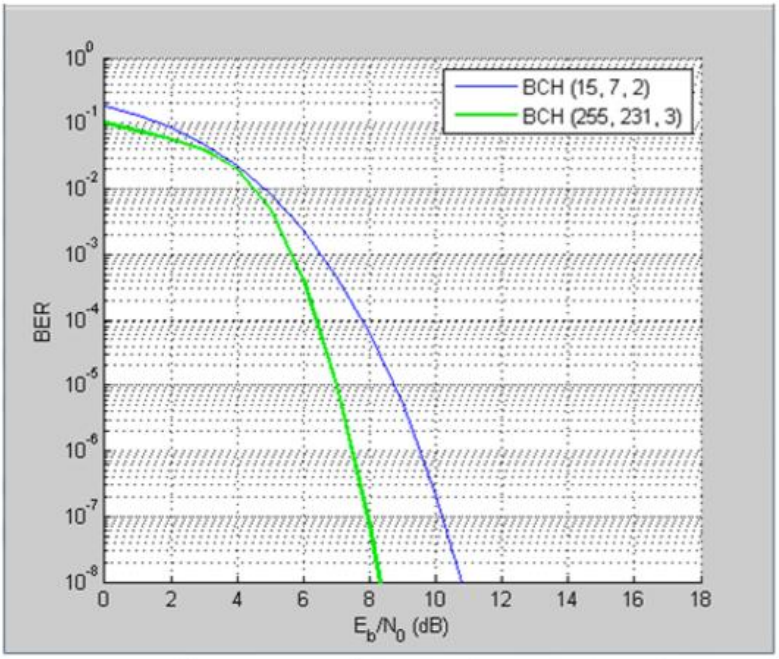


Fig. 7 - Scheme of Chien Search Block

After that performance analysis was conducted where the channel and modulation type are AWGN & PSK respectively and the result is given below.



Graph 8 – Performance of BCH(15,7,2) and BCH(255,231,3) using BER

It is observed that gain is 1, 4 db for a BER of 10^{-4} when length changes from 15 to 255.

In [9], the authors have proposed a method for hamming codes where they are explaining the entire procedure of Hamming code and 7-bit show alongside little prologue to speculations of information transmission by means of wired link and remote. Wired link can be of a few sorts, for example, Unshielded Twisted Pair (UTP) which can be utilized over a PC arrange like in the workplace, Coaxial link or Fiber Optics in which information goes through light waves. Remote correspondence can be of various kinds like radio, microwave, infrared, Bluetooth, NFC, and so forth. Hamming code utilizes equality bits to rectify mistakes and as per the quantity of bits to be encoded, number of equality bits increments. These equality or check bits are situated at specific spots with the first info relying upon information length. It is finished by:

$$cb = 2^n - (n-1) > p$$

Where:

cb : check bit value
n : total check bits
p : bit length

Check-Bit	Total Bit	Position
1	0	1
2	1	2
3	4	4
4	11	8
5	26	16
6	57	32
7	120	64
8	247	128

Table 1 - Check Bit and Total Bit

This paper concludes that hamming code can detect up to 1 bit. That is why it is also called single bit error detection. It also gives a future project to other people of finding a way to implement this technique to detect multiple errors.

In [10], the authors have proposed a method for cyclic redundancy codes which uses binary division to correct errors. Apart from that it also discusses types of error.

There are two types of error:

•Single bit error: When only one bit gets changed during sending the message. Usually happens during parallel transmission.

•Burst error: If more than one bit is changed during sending of message, it is known as burst error. This generally occurs during serial transmission.

Concept behind CRC technique is that multiplication of data and CRC bits when divided by generating function should give a value equal to zero to prove that data is not corrupted.

$$\frac{\text{[Message][CRC bits]}}{\text{Generating function}} = 0$$

Fig. 8 – Equation for CRC check

There are two critical strides of CRC procedure:

CRC generator-procedure of figuring the CRC bits at sender side.

CRC checker - procedure of figuring the CRC bits at beneficiary side.

Calculation for CRC generator

1. Information the quantity of information bits and CRC bits.

2. Enter creating capacity bits, its esteem and information bits esteem.

3. Perform Adder task on the variable entered in step-1.

4. Store information bits, creating capacity bits and information unit (information bits + CRC bits) to be worked in various clusters.

5. Rehash step-6 until the point when each piece of information unit is prepared.

6. Check information unit to be worked

if(1st bit==0)

XOR the information unit with 0 and store the outcome in same cluster of information unit.

else

XOR the information unit with producing capacity and store the outcome in same exhibit of information unit.

7. Show CRC bits and information unit to be transmitted over the system to recipient end in the wake of attaching CRC bits to information bits.

Calculation for CRC checker

1. Info the quantity of information unit bits (number of information bits + number of CRC bits) got from the sender.

2. Enter producing capacity bits, its esteem and information unit bits esteem.

3. Store producing capacity bits and information unit (information bits + CRC bits) got from sender in various clusters.

4. Rehash step-5 until the point when each piece of information unit is prepared.

5. Check information unit got from sender-

if(1st bit==0)

XOR the information unit with 0 and store the outcome in same exhibit of information unit.

else

XOR the information unit with creating capacity and store the outcome in same cluster of information unit.

6. Show CRC bits, perform snake activity on them and store the outcome in a number variable.

7. Check the estimation of whole number variable (var)-

in the event that (var ==0)

Information unit is right and acknowledged in the wake of disposing of the CRC bits.

else

Information unit got is debased and disposed of.

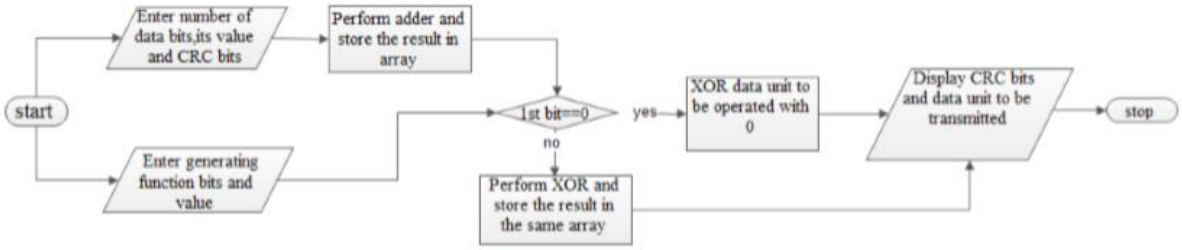


Figure 9 - CRC generator

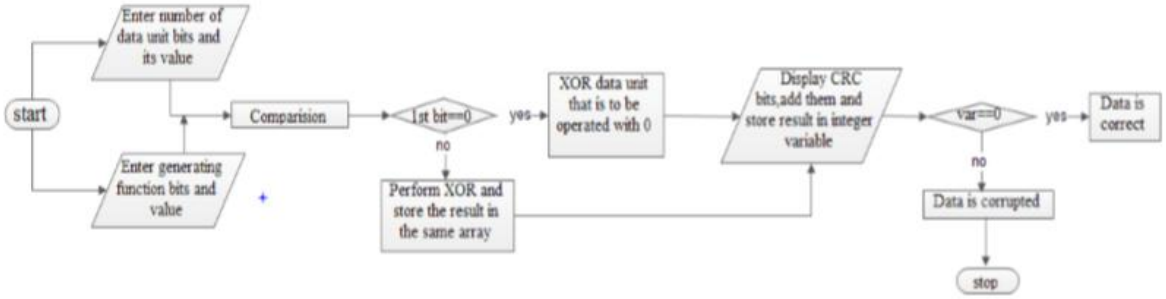


Figure 10 - CRC checker

In [21], the authors have proposed a method for Reed-Solomon codes. In the decades since their revelation, Reed-Solomon codes have appreciated incalculable applications, from minimized disc players in lounges everywhere throughout the planet to rocket that are presently well past the circle of Pluto. Reed-Solomon codes have been a necessary piece of the media communications insurgency in the last 50% of the twentieth century. This book has been composed trying to catch the power and utility of these codes, and in addition the historical backdrop of their utilization. Every part has been composed by experts in computerized correspondences who have utilized Reed-Solomon codes in their building outlines or have made the investigation and usage of Reed-Solomon codes a focal point of their exploration. The parts can be approximately assembled into four classes: history, code development, applications, and unravelling systems. The section promptly following this presentation is a joint exertion by Irving Reed and Gus Solomon. In this part they depict the occasions that hinted at the revelation of Reed-Solomon codes and their encounters a while later. It is essentially authentic in nature and gives some superb bits of knowledge into an incredible occasion in the historical backdrop of computerized interchanges innovation. The rest of the parts manage Reed-Solomon codes in a more itemized, specialized way. It is in this manner fitting that the peruser be set up with a touch of early on material. The rest of this section exhibits the three fundamental procedures for building Reed-Solomon codes and quickly talks about some ordinary applications and the translating issue. All the while, the rest of the sections in this book are presented.

The CD player is only the first of the numerous business, mass applications that Reed-Solomon codes can hope to appreciate in the coming years. The business world is winding up progressively portable, while all the while requesting solid, quick access to deals, promoting, and bookkeeping data. Tragically the portable channel is a frightful situation in which to impart, with profound blurs an ever-show wonder. Reed-Solomon codes are the absolute best arrangement; there is no other blunder control framework that can coordinate their unwavering quality execution in the portable condition. The optical channel gives another arrangement

of issues through and through. Shot commotion and dispersive, loud medium torment observable pathway optical frameworks, making clamour blasts that are best taken care of by Reed-Solomon codes. As optical strands see expanded use in rapid multiprocessors, we can hope to see Reed-Solomon codes utilized there also. In more specific, single-utilize applications, for example, the periodic profound space test, Reed-Solomon codes will keep on being utilized to constrain correspondence framework execution nearer and nearer to the line drawn by Shannon. These are however a couple of the applications demonstrating that Irving Reed and Gus Solomon's codes have presented to us far and that we can anticipate that them will be with us for quite a while to come.

CHAPTER 3

Implementation of various Error Correcting Techniques

3.1 Bose-Chaudhari-Hocquenghem Error Correcting Technique

BCH codes frame a class of cyclic mistake remedying codes that are developed utilizing polynomials over a limited field(also called Galois field). BCH codes were designed in 1959 by French mathematician Alexis Hocquenghem, and freely in 1960 by Raj Bose and D. K. Beam Chaudhari.

Algorithm:

Encoding steps:

- The 7 message bits are applied to the parallel to serial shift register.
- The output of parallel to serial shift register will be sent to the encoder.
- Using these message bits, parity bits are computed and sent to serial to parallel shift register.
- Then parity bits are appended to original message bits to obtain 15 bit encoded data.

Decoding steps:

- Calculate the syndromes for the received vector.
- Determine the no. of errors and error locator polynomial from the syndromes.
- Calculate the roots of the error location polynomial to find the error locations.
- Calculate the error values at those error locations.
- Correct the errors.

```
void generate_gf() {
    int i, mask;
    mask = 1;
    alpha_to[m] = 0;
    for (i = 0; i < m; i++) {
        alpha_to[i] = mask;
        index_of[alpha_to[i]] = i;
        if (p[i] != 0)
            alpha_to[m] ^= mask;
        mask <<= 1;
    }
    index_of[alpha_to[m]] = m;
    mask >>= 1;
    for (i = m + 1; i < n; i++) {
        if (alpha_to[i - 1] >= mask)
            alpha_to[i] = alpha_to[m] ^ ((alpha_to[i - 1] ^ mask) << 1);
        else
            alpha_to[i] = alpha_to[i - 1] << 1;
        index_of[alpha_to[i]] = i;
    }
    index of[0] = -1;
```

Figure 11 - Code

Galois field is a field that contains a limited number of components. Similarly as with any field, a limited field is a set on which the activities of addition, multiplication, subtraction and division are characterized and fulfil certain fundamental principles. The most well-known cases of limited fields are given by the whole number's mod p when p is a prime number. The above figure represents the Galois field generator.

```
void encode_bch() {
    int i, j;
    int feedback;
    for (i = 0; i < length - k; i++)
        bb[i] = 0;
    for (i = k - 1; i >= 0; i--) {
        feedback = data[i] ^ bb[length - k - 1];
        if (feedback != 0) {
            for (j = length - k - 1; j > 0; j--)
                if (g[j] != 0)
                    bb[j] = bb[j - 1] ^ feedback;
                else
                    bb[j] = bb[j - 1];
            bb[0] = g[0] & feedback;// g[0] && feedback
        } else {
            for (j = length - k - 1; j > 0; j--)
                bb[j] = bb[j - 1];
            bb[0] = 0;
        }
    }
```

Figure 12 - Code

The above figure is the main logic of the BCH encoder where parallel to serial shift in message bits takes place and encoding is done.

```
void decode_bch() {
    int i, j, q;
    int elp[] = new int[3], s[] = new int[5], s3;
    int count = 0, syn_error = 0;
    int loc[] = new int[3], err[] = new int[3], reg[] = new int[3];
    int aux;
    /* first form the syndromes */
    System.out.printf("s[] = (");
    for (i = 1; i <= 4; i++) {
        s[i] = 0;
        for (j = 0; j < length; j++)
            if (recd[j] != 0)
                s[i] ^= alpha_to[(i * j) % n];
        if (s[i] != 0)
            syn_error = 1; /* set flag if non-zero syndrome */

        System.out.printf("%3d ", s[i]);
    }
    System.out.printf(")\n");
    if (syn_error != 0) { /* If there are errors, try to correct them */
        if (s[1] != -1) {
            s3 = (s[1] * 3) % n;
            if (s[3] == s3) /* Was it a single error ? */
            {
                System.out.printf("One error at %d\n", s[1]);
                recd[s[1]] ^= 1; /* Yes: Correct it */
            } else {
                if (s[3] != -1)
                    aux = alpha_to[s3] ^ alpha_to[s[3]];
                else
                    aux = alpha_to[s3];
                elp[0] = 0;
                elp[1] = (s[2] - index_of[aux] + n) % n;
                elp[2] = (s[1] - index_of[aux] + n) % n;
                System.out.printf("sigma(x) = ");
                for (i = 0; i <= 2; i++)
                    System.out.printf("%3d ", elp[i]);
                System.out.printf("\n");
```

Figure 13 - Code

This figure of the code is calculating the first form of syndromes with the help of the received vector. Then it is calculating the error location polynomial to find the errors.

```
                System.out.printf("\n");
                System.out.printf("Roots: ");
                /* find roots of the error location polynomial */
                for (i = 1; i <= 2; i++)
                    reg[i] = elp[i];
                count = 0;
                for (i = 1; i <= n; i++) { /* Chien search */
                    q = 1;
                    for (j = 1; j <= 2; j++)
                        if (reg[j] != -1) {
                            reg[j] = (reg[j] + j) % n;
                            q ^= alpha_to[reg[j]];
                        }
                    if (q == 0) { /* store error location number indices */
                        loc[count] = i % n;
                        count++;
                        System.out.printf("%3d ", (i % n));
                    }
                }
                System.out.printf("\n");
                if (count == 2)
                    /* no. roots = degree of elp hence 2 errors */
                    for (i = 0; i < 2; i++)
                        recd[loc[i]] ^= 1;
                else
                    /* Cannot solve: Error detection */
                    System.out.printf("incomplete decoding\n");
            }
        } else if (s[2] != -1) /* Error detection */
            System.out.printf("incomplete decoding\n");
    }
```

Figure 14 - Code

In the above figure, to find the roots of error location polynomial Chien search algorithm is used. With the help of the roots, one will be able to find the position of the errors. Then store those locations to help remove those errors in next step.

3.2 Hamming Error Correcting Technique

Hamming codes are a family of linear error-correcting codes that generalize the Hamming (7,4) code, and were invented by Richard Hamming in 1950. Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors.

Algorithm:

- Number the bits starting from 1: bit 1, 2, 3, 4, 5, etc.
- Write the bit numbers in binary: 1, 10, 11, 100, 101, etc.
- All bit positions that are powers of two (have a single 1 bit in the binary form of their position) are parity bits: 1, 2, 4, 8, etc. (1, 10, 100, 1000)
- All other bit positions, with two or more 1 bit in the binary form of their position, are data bits.
- Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.
- Parity bit 1 covers all bit positions which have the least significant bit set: bit 1 (the parity bit itself), 3, 5, 7, 9, etc.
- Parity bit 2 covers all bit positions which have the second least significant bit set: bit 2 (the parity bit itself), 3, 6, 7, 10, 11, etc.
- Parity bit 4 covers all bit positions which have the third least significant bit set: bits 4–7, 12–15, 20–23, etc.
- Parity bit 8 covers all bit positions which have the fourth least significant bit set: bits 8–15, 24–31, 40–47, etc.
- In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

```
static int[] generateCode(int a[])
    {
        int b[];
        int i=0, parity_count=0 ,j=0, k=0;
        while(i<a.length)
            {
                if(Math.pow(2,parity_count) == i+parity_count + 1)
                    {
                        parity_count++;
                    }
                else
                    {
                        i++;
                    }
            }
        b = new int[a.length + parity_count];
        for(i=1 ; i<=b.length ; i++)
            {
                if(Math.pow(2, j) == i)
                    {
                        b[i-1] = 2;
                        j++;
                    }
                else
                    {
                        b[k+j] = a[k++];
                    }
            }
        for(i=0 ; i<parity_count ; i++)
            {
                b[((int) Math.pow(2, i))-1] = getParity(b, i);
            }
        return b;
    }
```

Figure 15 - Code

This is the main logic of hamming code encoding technique by inserting parity bits on the position of multiples of 2.

3.3 Repetition Error Correcting Technique

In coding hypothesis, the redundancy code is a standout amongst the most essential mistake remedying codes. Keeping in mind the end goal to transmit a message over an uproarious channel that may degenerate the transmission in a couple of spots, the possibility of the redundancy code is to simply rehash the message a few times. The expectation is that the channel defiles just a minority of these reiterations. Along these lines the beneficiary will see that a transmission blunder happened since the got information stream isn't the redundancy of a solitary message, and additionally, the collector can recuperate the first message by taking a gander at the got message in the information stream that happens frequently.

```
System.out.println("Enter the size of message(<10)");
Scanner s = new Scanner (System.in);
int n= s.nextInt();
System.out.println("Enter the message");

Scanner t = new Scanner (System.in);
for(i=0; i<n ; i++)
{
   arr[i] = t.nextInt();
}

System.out.println("message is encoding......");
System.out.println("\n");

j=0;
for(i=0; i<3*n; i=i+3)
{
    brr[i]=arr[j];
    brr[i+1]=arr[j];
    brr[i+2]=arr[j];
    j=j+1;
```

Figure 16 - Code

This figure shows how input is taken and stored in array to be used as an input to the encoder part of repetition codes.

3.4 Reed-Solomon Error Correcting Technique

Reed-Solomon codes are cases of blunder rectifying codes, in which excess data is added to information with the goal that it can be recuperated dependably notwithstanding mistakes in transmission or capacity and recovery. The blunder revision framework utilized on CD's and DVD's depends on a Reed-Solomon code.

These codes are likewise utilized on satellite connections and different correspondences frameworks.

```
public int[] doFFT(int[] a, int gen) {
    if(a.length == 1) return a;
    int wn = gen;
    GF257 w = new GF257(1);

    int[] U = new int[a.length/2];
    int[] V = new int[a.length/2];
    for(int i = 0; i < a.length; i++) {
        if(i % 2 == 0) U[i/2] = a[i];
        else V[i/2] = a[i];
    }

    int[] y0 = doFFT(U, (new GF257(gen)).mult(gen).val);
    int[] y1 = doFFT(V, (new GF257(gen)).mult(gen).val);
    int[] y = new int[a.length];

    for(int i = 0; i < a.length; i++ ) {
        y[i] = w.mult(y1[i % (a.length/2)]).add(y0[i % (a.length/2)]).val;
        w = w.mult(gen);
    }
    return y;
}
```

Figure 17 - Code

```
public char[] decode(char[] allC, HashSet<Integer> bad) {
    GF28 wn = new GF28((char)1);
    char[][] A = new char[k][k];
    char[] c = new char[k];
    char[] m = new char[k];
    int cnt = 0;

    for(int i = 0; i < allC.length; i++) {
        if(i != 0) wn = wn.mult(gen);
        if(bad.contains(i)) continue;

        GF28 cur = new GF28((char)1);
        for(int j = 0; j < k; j++) {
            A[cnt][j] = cur.val;
            cur = cur.mult(wn);
        }
        c[cnt] = allC[i];

        cnt++;
        if(cnt >= k) break;
    }

    m = gaussianElimination(A,c);

    return m;
}
```

Figure 18 - Code

CHAPTER 4

System Development

4.1 **SOFTWARE REQUIREMENTS**

- Eclipse
- JDK (Java Development Kit) 1.7 or above

4.2 **PROPOSED MODEL**

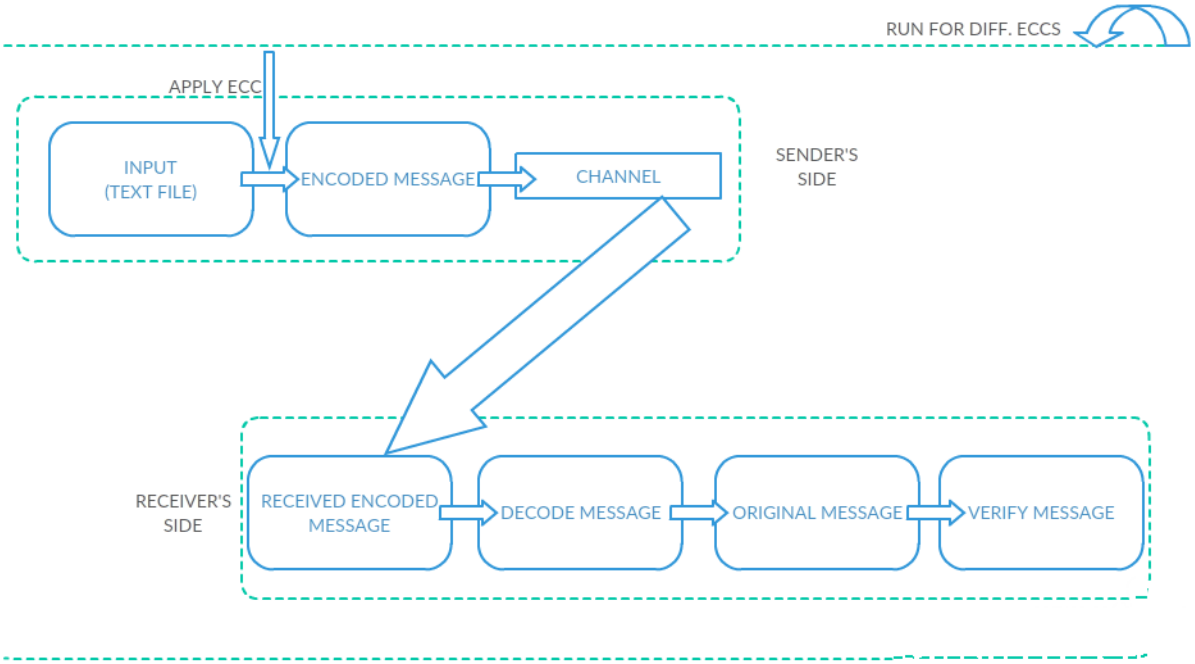


Figure 19 – Proposed diagram

Our proposed framework peruses contribution from a content record through the assistance of a program which will be usage of an ECC. At that point it will apply the ECC on to the info, changing over it into the encoded message. After that point we will put that message on to the channel. Then the channel will send it to beneficiary. It will be received on the recipient's side, message will be gotten. Then we will unravel the message to separate the first message. Subsequent to extricating the first message, we will confirm it for any mistakes. This entire procedure will keep running for various ECCs one by one.

CHAPTER 5

PERFORMANCE ANALYSIS

5.1 **PERFORMANCE PARAMETER**

We are using BER (Bit Error Rate) as our performance parameter.

$$\text{Bit Error Rate, BER} = \frac{\text{Number of errors}}{\text{Total number of bits sent}}$$

Figure 20 – Formula for BER

5.2 **PERFORMANCE TESTING**

We tried to test the code for different size of input message to calculate its BER.

Hamming code

No. of Chars	BER
5 char	0.2
15 char	0.167
50 char	0.128
100 char	0.097
1000 char	0.079

Table 2 – Test outputs

BCH Code

No. of Chars	BER
5 char	0
15 char	0.0037
50 char	0.0149
100 char	0.0035
1000 char	0.0016

Table 3 – Test outputs

Repetition Code

No. of Chars	BER
5 char	0
15 char	0.034
50 char	0.184
100 char	0.268
1000 char	0.443

Table 4 – Test outputs

Reed Solomon Code

No. of Chars	BER
5 char	0
15 char	0.074
50 char	0.0184
100 char	0.0218
1000 char	0.0231

Table 5 – Test outputs

No. of Chars	Hamming Code	Repetition Code	Reed Solomon Code	BCH Code
10	20%	2.83%	5.671%	2.627%
60	14.45%	11.65%	1.84%	1.645%
200	11.43%	32.78%	2.186%	0.321%
400	9.76%	37.42%	2.248%	0.259%
800	8.73%	42.03%	2.293%	0.184%

Table 6 – BER comparison of different codes

CHAPTER 6

CONCLUSION

The conclusion or after-effect of our observation is a table involving a correlation of various ECCs for various sizes of information messages. The table would turn into a reason for future reference for those individuals who wish to execute an ideal code to their framework which will have the capacity to fulfil every one of their prerequisites. Along these lines it will regard pick the suitable code for utilization without having much inconvenience. The comparison between various ECCs will educate us regarding how extraordinary their BER is for various size of info message. So for various situations which code ought to be utilized, we will have the capacity to recognize it as indicated by this current table's information.

Out of the final concluding table, BCH code found out to be better than the others.

6.1 **FUTURE SCOPE**

We will try to improve BER for the codes we have picked. To do so, we can combine two codes together to get better performance. For example, BCH code and repetition code can be merged together to form a single code, thereby improving the BER.

We will try to search for more codes for specific purposes. And even try to test it for some other performance parameter if possible.

6.2 **APPLICATIONS**

All the codes we will be taking in this project can be applied at any real time system. Be it hospitals, offices, departments, etc. In fact they will find it easy, choosing codes now.

REFERENCES

[1] Zhang Aili and Liu Xiufeng, “Chinese remainder codes” Frontiers of Mathematics in China, Volume 1, Issue 3, pp 452-461, September 2006

[2] Michael Sipser and Daniel A. Spielman, “Expander Codes” IEEE Transaction of Information Theory, Volume 42, No. 6, pp 1710-1722, November 1996

[3] Ge Chen and Saied Nooshabadi, “Analysis and Design of Serial Error Correction Code ith Crosstalk Avoidance Technique” IEEE 30th Canadian Conference on Electrical and Computer Engineering, 2017

[4] Matthew F. Brejza, Tao Wang, Wenbo Zhang, David Al-Khalili, Robert G. Maunder, Bashir M. Al-Hashimi and Lajos Hanzo, “Exponential Golomb and Rice Error Correction Codes for Generalized Near-Capacity Joint Source and Channel Coding” IEEE Access, Open Access Journal, Volume 4, November 2016

[5] Tao Wang, Matthew F. Brejza, Wenbo Zhang, Robert G. Maunder and Lajos Hanzo,” Reordered Elias Gamma Error Correction Codes for the Near-Capacity Transmission of Multimedia Information” IEEE Access, Open Access Journal, Volume 4, October 2016

[6] Jangwon Park, Jongsun Park and Swarup Bhunia, “VL-ECC: Variable Data-Length Error Correction Code for Embedded Memory in DSP Applications” IEEE Transactions on Circuits and Systems-II: Express Briefs, Volume 61, No. 2, February 2014

[7] Yohei Aikawa and Hiroyuki Uenohara, “Numerical Investigation of All-Optical Forward-Error-Correction Coding Scheme With Convolutional Code” IEEE Photonics Journal, Volume 8, No. 2, April 2016

[8] Elghayyaty Mohamed, Hadjoudja Abdelkader, Omar Mouhib, El Habti El IdrissiAnas, Mahjoub Chakir, “Performance Study of BCH Error Correcting Codes Using the Bit Error Rate Term BER” International Journal of Engineering Research and Application, ISSN : 2248-9622, Vol. 7, Issue 2, (Part -2), pp.52-54, February 2017

[9] Debalina Roy Choudhury and Krishanu Podder, “Design of Hamming Code Encoding and Decoding Circuit Using Transmission Gate Logic” International

Research Journal of Engineering and Technology (IRJET), Volume 2, Issue 7, October 2015

[10] Sukirty Jain and Siddharth Singh Chouhan, "Cyclic Redundancy Codes: Study and Implementation" International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014

[11] Li Ping, Xiaoling Huang and Nam Phamdo, "Zigzag Codes and Concatenated Zigzag Codes" IEEE Transaction of Information Theory, Volume 47, No. 2, February 2001

[12] Liu X F, Zhang A L, "Affine manifold codes and projective manifold codes" Journal of Southwest Jiaotong University, 33(4): 470–474, 1998 (in Chinese)

[13] Zhang A L, "Study on weak block design and a new class of linear codes" Journal of Southwest Jiaotong University, 33: 68–77, 1998

[14] R. G. Gallager, Low Density Paripcheck Codes Cambridge, MA. MIT Press, 1963

[15] P. Sotiriadis and A. Chandrakasan, "Bus energy reduction by transition pattern coding using a detailed deep submicrometer bus model," IEEE Trans. on Circuits and Systems I], vol. 50, no. 10, pp. 1280–1295, Oct. 2003.

[16] M. Stan and W. Burleson, "Bus-invert coding for low-power i/o," IEEE Trans. on, Very Large Scale Integration (VLSI) Systems, vol. 3, no. 1, pp. 49–58, Mar 1995.

[17] J. Natesan and D. Radhakrishnan, "Shift invert coding (sinv) for low power vlsi," in Euromicro Sym. on Digital System Design, (DSD), pp. 190–194, Aug. 2004.

[18] G. Chen and S. Nooshabadi, "Optimization of on-chip interconnect signaling for low energy and high performance," Journal of Low Power Electronic, vol. 8, no. 1, pp. 30–38, February 2012.

[19] "An adaptive scheme for bus error detection," in 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1–4, Aug 2011.

[20] “A hybrid equivalent-bit spacing scheme for low energy and high performance for bus signalling”, 53rd IEEE International Midwest Symposium on Circuits and Systems, pp. 209–212, Aug 2010.
[21] Stephen B. Wicker, Vijay K. Bhargava, “An introduction to Reed-Solomon codes”, Wiley-IEEE Press, edition 1, 1994