**Error detection and correction by Hamming code**



**Error Detection and Correction**

**by Hamming Code**

**Course Title: Computer Architecture**

**Course Code: CSE 360**

**Section: 1**

**MONDAY, 24 May, 2021**



**Project Title: Error detection and correction by Hamming code.**

**Submitted By: Group (1)**

|  |  |
| --- | --- |
| **Name** | **ID** |
| **Ariful Islam Toufiq** | **2017-2-60-108** |
| **Md. Ahied Mahi Chowdhury** | **2018-1-60-028** |
| **Syeda Tamanna Sheme** | **2018-2-60-010** |

**Submitted To:**

**Assoc. Prof. Dr. Ahmed Wasif Reza**

**Department of Computer Science and Engineering East West University**

# Table of Content

1. **Title Page…………………………………………………………..1**
2. **Group Information………………………………………………..2**
3. **Table of Content…………………………………………………...3**
4. **Executive Summary……………………………………………….4**
5. **Introduction………………………………………………………..5**
6. **Methodology……………………………………………………….5**
7. **The Classical Hamming Code Method……………………5**
8. **The Improved Hamming Code Method…………………..8**
9. **Results……………………………………………………………...9**
10. **Analysis……………………………………………………………11**
11. **Comparison……………………………………………………….12**
12. **Applications, Advantages and Disadvantages…………………..13**
13. **Recommendation………………………………………………….13**
14. **Conclusion…………………………………………………………14**
15. **References…………………………………………………………14**

# Executive Summary

This project is all about transmitting message bits and checking if it has errors or not. Whether it has errors, the hamming code is going to fix it. In this project, we have implemented an algorithm that is faster than the classic Hamming code and it works with single-bit errors. Besides, We compared both classic and improved hamming code. This code is famous for its single-bit error detection & correction capability.

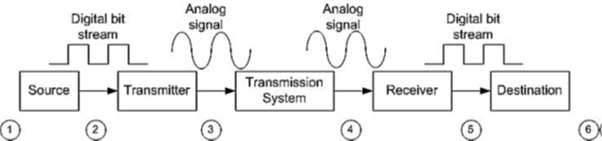
**Keywords:** Error Correction, Hamming Code, Transmission Data, Bit Error, Bit Detection.

# Introduction

**Digital communication is important in a communications infrastructure that is being built, a simple basic communication model is as follows:**



**The data transmission process in detail can be seen in the figure below:**



**Hamming code is the combination of two parts,**

1. The data bits or the massage bits
2. The parity or the redundancy bits

The redundancy bits are used to indicate or to point out the message bits following a specific algorithm. We used the even parity bits system to generate the redundancy bits. For a 7-bit data item, it introduces 4 redundancy bits. These redundancy bits are to be interspersed at bit positions **2n (n = 0, 1, 2, 3)** with the original data bits(for classic hamming code). After error detection & correction, if any, the data bits have to be reassembled by removing the redundancy bits. Scaling it for larger data lengths results in a lot of overhead due to interspersing the redundancy bits and their removal later. In contrast, the proposed method is highly scalable without much overhead as we append the redundancy bits at the end of the data bits. Because of this feature, it is suitable for the transmission of large size data bit-streams with much lower redundancy bits per data bit ratio.

# Methodology

Hamming Code method is one of the simplest methods of error detection and error correction. This method uses logic operation XOR (Exclusive - OR) in the process of error detection, and the process of error correction, while input and output of data from the method of Hamming Code in the form of binary numbers.

## The classical hamming code method

The classical hamming code method inserts multiple pieces of check bits or redundancy bits into data. The number of check bits inserted depending on the length of the data. The calculation formula for calculating the number of check bits that must insert into the data following the rule: **2r >= n + r + 1.**

Where r = number of redundancy bits, n= the number of data bits.

If we want to get **‘r’** much redundancy bits for 7 bits data (n=7) we calculate as follows:

**For r=0, 20 >= 7+0+1**

1>=8 (false)

**For r=1, 21>=7+1+1**

2>=9 (false)

**For r=2, 22>=7+2+1**

4>=10(false)

**For r=3, 23>=7+3+1**

8>=11(false)

**For r=4, 24>=7+4+1**

16>=12(true)

Therefore, **r=4,** the redundancy bits satisfies for 7-bit data massage.

**All Steps to Detect and Correct the Error:**

1. Generating parity or redundancy bits on sending a message and intersperse them into the data bits.
2. Receive the massage and generate redundancy bits on the received data message only and intersperse them into the data bits.
3. Compare these two redundancy bits data set using Ex-or operation and get the parity status.
4. Convert the parity status into a decimal number and thus we get the error position.
5. Correct the error.

**The bits are represented as follows:**

**Sending-message:**



**In Hamming Code, each r bit is the parity bit for one combination of data bits as shown below:**

 **r1: bits** 1,3,5,7,9,11 = 1 0 1 0 1 such that even parity is 1 **r2: bits** 2,3,6,7,10,11 =1 1 1 0 1 such that even parity is 0 **r4: bits** 4,5,6,7 = 0 1 1 such that even parity is 0  **r8: bits** 8,9,10,11 = 0 0 1 such that even parity is 1

**So, parity bits for sending data bits are: 1 0 0 1**

**The bits are represented as follows:**

**Receiving-message:**



**The received message is:**



**Now we calculate redundancy bits looping over data bits only (excluding the received redundancy bits):**

 **r1: bits** 1,3,5,7,9,11 = 1 0 0 0 1 such that even parity is 0  **r2: bits** 2,3,6,7,10,11 = 1 1 0 0 1 such that even parity is 1  **r4: bits** 4,5,6,7 = 0 1 0 such that even parity is 1  **r8: bits** 8,9,10,11 = 0 0 1 such that even parity is 1

**So,** parity bits newly generated for received data bits are **1 1 1 0**

**Now comparing these two parity bits set using Ex-or operation, we get the parity status:**



This is the sending parity bits.



This is the received parity bits.

**We get parity status as below:**



Converting these binary bits to decimal we get the position where the error is which is 7.

**Correction:**



We replace the bit position 7 as binary bit 1.

Thus, a total of 20 bits are involved in the process of Hamming bits calculations (including parity bits).

**PROPOSED IMPROVED HAMMING CODE**

We get the total parity bits number following the same rule which is **2r >= n + r + 1**.

All steps are briefly given below to detect and correct the error following the proposed improved way:

1. Generating parity or redundancy bits on sending a message and appending them at the end of the data bits.
2. Receive the massage and generate redundancy bits on the received data message only.
3. Compare these two redundancy bits data set using Ex-or operation and get the parity status.
4. Convert the parity status into a decimal number and thus we get the error position.
5. Correct the error.

**The bits are represented as follows: Sending-message:**



**In the improved Hamming Code, each r bit is the parity bit for one combination of data bits as shown below: (We loop over till the data bits):**  **r1: bits** 1,3,5,7= 1 1 0 1 such that even parity is 1 **r2: bits** 2,3,6,7=0 1 0 1 such that even parity is 0 **r3: bits** 4,5,6,7 = 1 0 0 1 such that even parity is 0



**r4: bits** 8,9,10 = If the data bits are not available, the discovered redundancy bits will be

the combination. So, bits are r1, r2, r3 = 1 0 0 = 1

So, parity bits for sending data bits are **1 0 0 1**

**The bits are represented as follows:**

**Receiving-message:**



**In the improved Hamming Code, each r bit is the parity bit for one combination of data bits as shown below: (We loop over till the data bits):**

**r1: bits** 1,3,5,7= 1 0 0 1 such that even parity is 0 **r2: bits** 2,3,6,7= 0 0 0 1 such that even parity is 1 **r3: bits** 4,5,6,7= 1 0 0 1 such that even parity is 0



**r4: bits** 8,9,10 = If the data bits are not available, the discovered redundancy bits will be

the combination. So, bits are r1, r2, r3 = 1 0 0 = 1

So, parity bits for received data bits are **1 0 1 0**

**Now comparing these two parity bits set using Ex-or operation, we get the parity status**



This is the parity bits for sending-message



This is the parity bits for received-message.

**So, we get parity status as,**



Converting these binary bits to decimal we get the position where the error is Which is 3.

**Correction:**



Here the red-colored cell is the error bit position.

**We replace the bit position 3 as binary bit 1.**

**So, the corrected message is,**



Thus, a total of 19 bits are involved in the process of Hamming bits calculations (including parity bits).

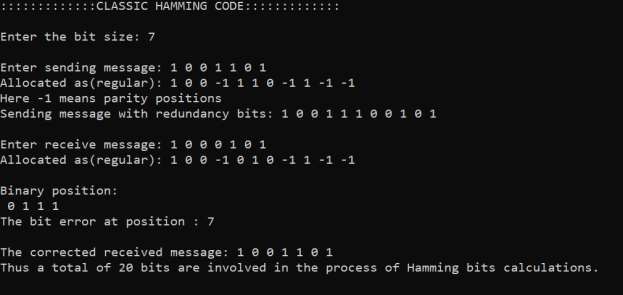
# Results

Now I am representing the Codeblock output results of my Classic and Improved Hamming Code.

**For classic Hamming code: 7-bit data, sending bit:1 0 0 1 1 0 1**

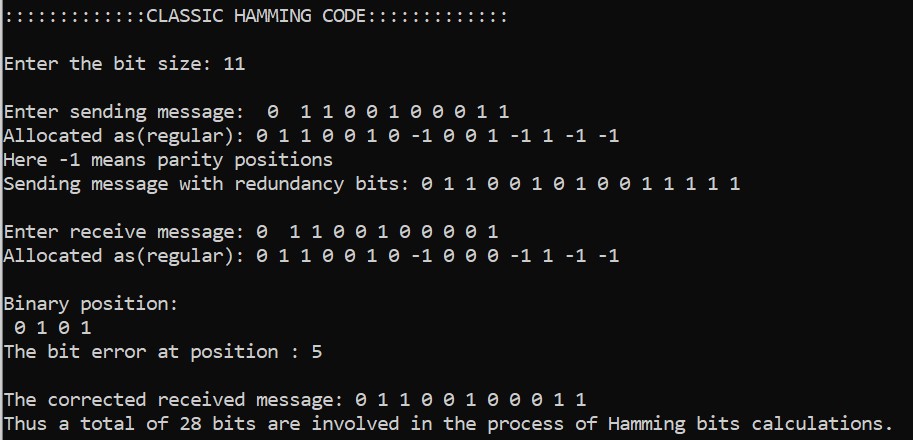
**Received bit:1 0 0 0 1 0 1**

**The result is,**



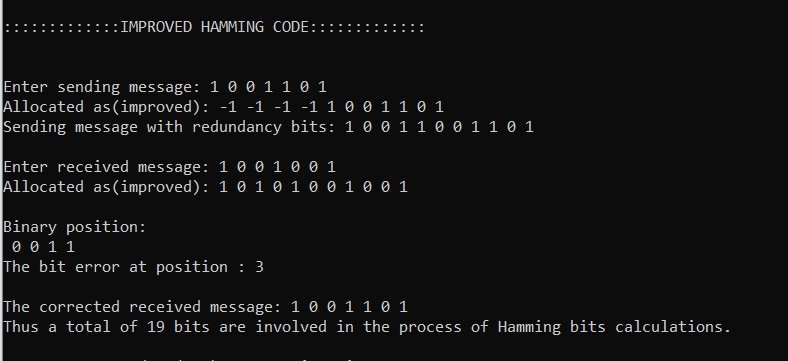
**For Classic Hamming Code, 11-bit data, sending bit:0 1 1 0 0 1 0 0 0 1 1**

**Received bit:0 1 1 0 0 1 0 0 0 0 1**



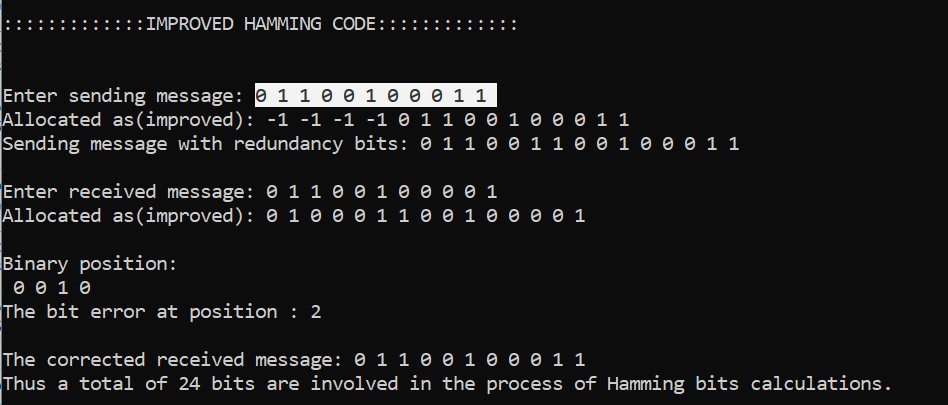
**For improved Hamming code:** **7-bit data, sending bit:1 0 0 1 1 0 1**

**Received bit:1 0 0 0 1 0 1**



**For improved Hamming code:** **11-bit data, sending bit:0 1 1 0 0 1 0 0 0 1 1**

**Received bit:0 1 1 0 0 1 0 0 0 0 1**



# Analysis

The result demonstrates that for 7 bits and 11 bits data in classic hamming code, a total of 20 bits and 28 bits are involved in the process of Hamming bits calculations wherein improved hamming code, a total of 19 bits and 24 bits are involved in the process of calculation of redundancy bits.

Here, improved hamming code takes fewer bits to calculate. Therefore, the improved hamming code is faster than the classic one.

Also the improved hamming code can detect and correct the single-bit error quickly as it iterates in the range of only the last few redundancy bits, in contrast, the classic one has to iterates through the whole array to find parity bits and then can do the detection and correction part of it.

# Comparison

There are some differences between Improved hamming code and classic hamming code. The differences are given below:-

|  |  |
| --- | --- |
| **Hamming Code** | **Improved Hamming Code** |
| **1.** Redundancy bits are interspersed in the data bits. | **1.** Redundancy bits are appended at the end of the data bits. |
| **2.** Takes much bits to calculate. | **2.** Takes fewer bits than old hamming code. |
| **3.** Tough to separate redundancy bits from data bits. | **3.** Simple and easy to separate redundancy bits from data bits. |
| **4.** Takes more iteration while comparing the redundancy bits. | **4.** Takes the same number of redundancy bits to iterate. |
| **5.** Slower and inefficient to scale large number of data bits. | **5.** Easily scalable for larger data lengths. |

# Application, Advantages and Disadvantages

**The applications of Hamming Code are:** .

They are extensively used in telecommunication industry.

They are used in [computer memory,](https://electricalfundablog.com/read-only-memory-rom/) modems and [embedded](https://electricalfundablog.com/embedded-system-characteristics-types-advantages-disadvantages/) processors.

They are used in Nano Satellites.

**Advantages of Hamming code:**

Hamming code method is effective on networks where the data streams are given for the single-bit errors.

Hamming code not only provides the detection of a bit error but also helps you to indent bit containing error so that it can be corrected.

The ease of use of hamming codes makes it best them suitable for use in computer memory and single-error correction.

**Disadvantages of Hamming code:**

Single-bit error detection and correction code. However, if multiple bits are founded error, then the outcome may result in another bit which should be correct to be changed. This can cause the data to be further errored.

Hamming code algorithm can solve only single bits issues.

# Conclusion

In the proposed improvement the redundancy bits are appended at the end of data bits. This eliminates the overhead of interspersing the redundancy bits at the sender end and their removal at the receiver end after checking for single-bit error and consequent correction, if any. Further the effort needed in identifying the values of the redundancy bits is lower in the proposed novel method. Hamming code is normally used for transmission of 7-bit data item. Scaling it for larger data lengths results in a lot of overhead due to interspersing the redundancy bits and their removal later. In contrast, the proposed method is highly scalable without such overhead. We see that there is only 7 bit overhead for a 56-bit data stream, which is much less compared to 4 bit overhead for a 7-bit data. Because of this feature this new method is suitable for transmission of large size data bit-streams as long as there is likelihood of at the most single-bit error during transmission.

# Recommendations

The improved hamming code is unable to detect and correct multiple bits error and burst bit error. If we can research on it, hope we can eradicate this limitation.

# Reference

1. U. K. Kumar and B. S. Umashankar, “Improved hamming code for error detection and correction,” in 2007 2nd International Symposium on Wireless Pervasive Computing, 2007, pp.

498–500, doi: 10.1109/ISWPC.2007.342654.

**Link:** [https://ieeexplore.ieee.org/document/4147113?arnumber=4147113.](https://ieeexplore.ieee.org/document/4147113?arnumber=4147113)

1. Deepika, A. Kumar, and Gurusiddayya, "A Study on Error Coding Techniques,"

International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 4, no. 4, pp. 825-828, 2016

1. R. Ullah, J. Khan, S. Latif and I. Ullah, "Indication of Efficient Technique for Detection of Check Bits in Hamming Code," International Journal of Computer Science (IJCSI), vol. 8, no. 5, pp. 241-246, 2011

1. R. Hamming, "Error Detecting and Error Correcting Codes," The Bell System Technical Journal, vol. 29, no. 2, 1950.

1. **Link:**<https://www.tutorialspoint.com/error-correcting-codes-hamming-codes>

**Link:** https://www.electronicshub.org/error-correction-and-detection-codes/