SEAN INSTITUTE
Sharing Knowledge
http://infor.seaninstitute.org/index.php/infokum/index
JURNAL INFOKUM, Volume 9, No. 2,Juni 2021 ISSN : 2302-9706

BIT CHECK IN ERROR DETECTION ON TEXT DATA TRANSMISSION USING HAMMING CODE ALGORITHM

Pilipus Tarigan

Medan College of Computer Science (STIKOM Medan)
Jln. Jamin Ginting No. 285 P. Moon Medan
e-mail: pilipustarigans@gmail.com

Abstract

Article Info
Received 12 June 2021
Revised 28 June 2021
Accepted 30 June 2021

When data or information is transmitted via wireless or via cable channels, errors may occur while the data is transmitted. One of the efforts made is to apply error control coding. Hamming code is an example of an existing error control coding technique. Hamming code performance is distinguished by the number of parity bits it has. Ontelecommunications allows everyone to communicate with each other quickly over long distances though. Data that is transmitted or sent in the form of text data can fail (error). Errors cause changes in the contents of the data transferred to the recipient (Receiver) to change or fail. One way to detect simple errors is to use Hamming Code with single error correction. In the detection, this algorithm uses the EX-OR (Exclusive–OR) operation in the error detection process.In testing the data sent is not the same as the result received, the bit has experienced an error, and the system will correct the position where the bit has an error.
Keywords: information system, web-based, thesis defense schedule and assessment

1. INTRODUCTION

Submission of data at the time of transmission or transmission of text data can fail (error). Errors cause changes in the contents of the data transferred to the recipient (Receiver) to change or fail. In computer science, there are various kinds of logic to detect and correct these errors. One way to detect simple errors is to use Hamming Code with single error correction.

*Hamming Code*is an error detection algorithm that is able to detect several errors, but is only able to correct one error (single error correction). This error detection algorithm is very suitable to be used in situations where there are several random errors. The Hamming Code algorithm inserts (n + 1) check bits into 2n data bits. This algorithm uses the EX-OR (Exclusive–OR) operation in the error detection process. The input and output data of the Hamming Code algorithm are binary numbers.

Based on the description above, the author intends to design an application that is able to explain error detection techniques with the Hamming Code algorithm

2. METHODS

Step-The steps for making this system include:

1) Literature review
   The literature study method is by reading some of the literature and references on error detection algorithms.Gather expert opinions in supporting journals related to previous error detection.
2) Analyzing error detection techniques from the Hamming Code algorithm.
3) Designing the application interface using the Visual Basic 6.0 programming language.

4) Implementing error detection techniques using the Hamming Code algorithm.
5) Testing the error detection application using the Visual Basic 6.0 programming language.

3. RESULTS AND DISCUSSION

1. System Modeling

In modeling this system the author uses *Unified Modeling Language* (UML) in designing and designing Bit Check In Error Detection Application In Text Data Transmission With Single Error Correction Using Hamming Code Algorithm. The UML that will be used are use case diagrams and Activity diagrams.

The following is a picture of the use of the system which is described in the form of a Use Case diagram.

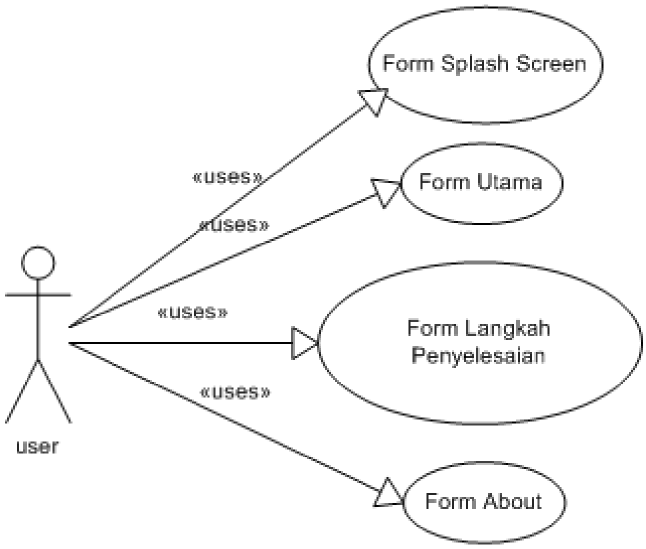


Figure 1. Use Case diagram of the system

2 Check Bit Insertion Model Activity

System	User

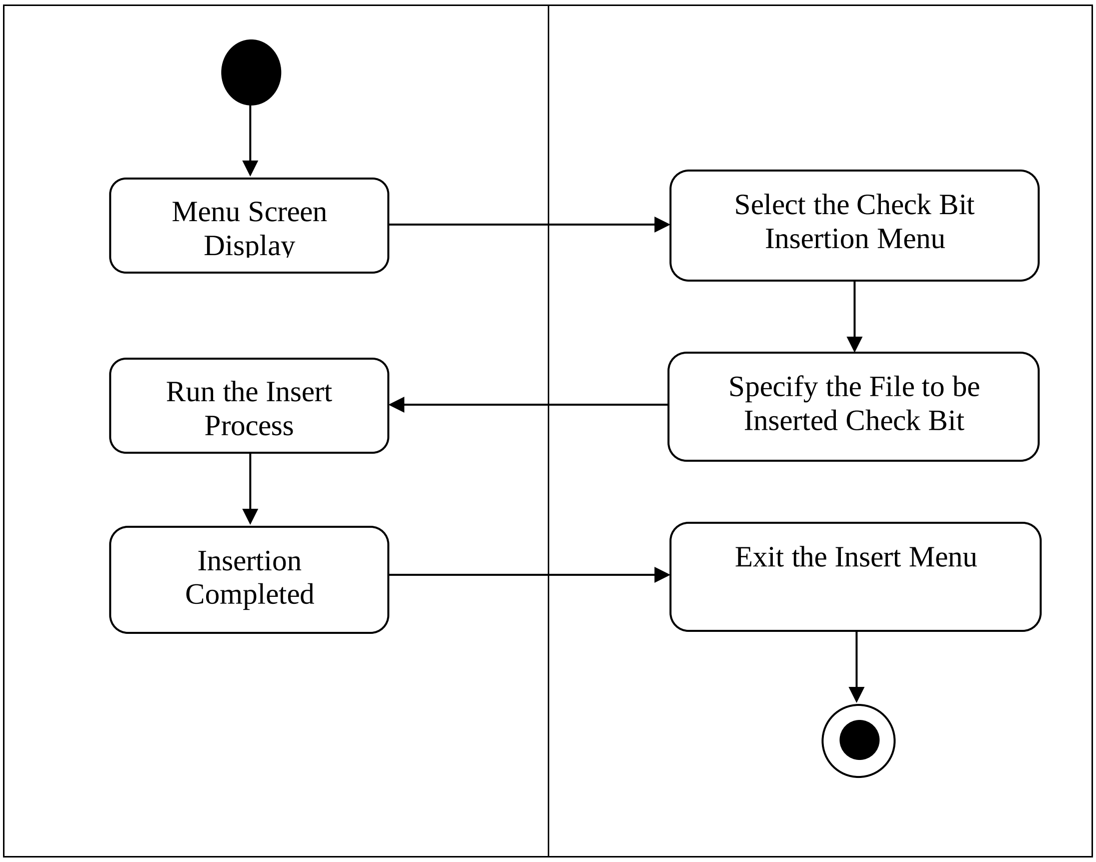


Figure 2. *Activity* Check Bit Insertion Process Diagram

3. Activity Model Error Detection

System	User

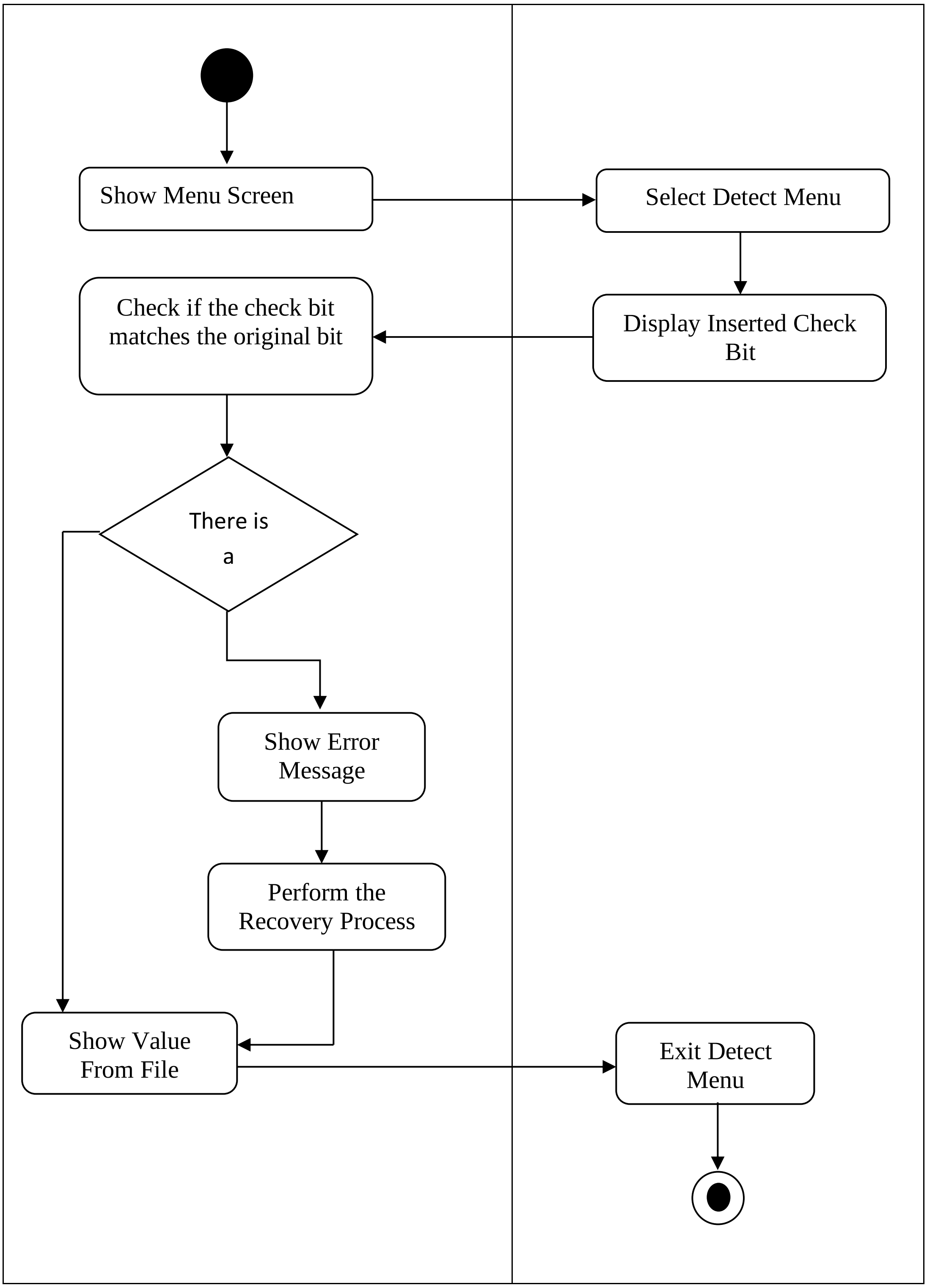


Figure 3. Activity Diagram Error Detection

4. Activity Model Process View About the Program

System	User

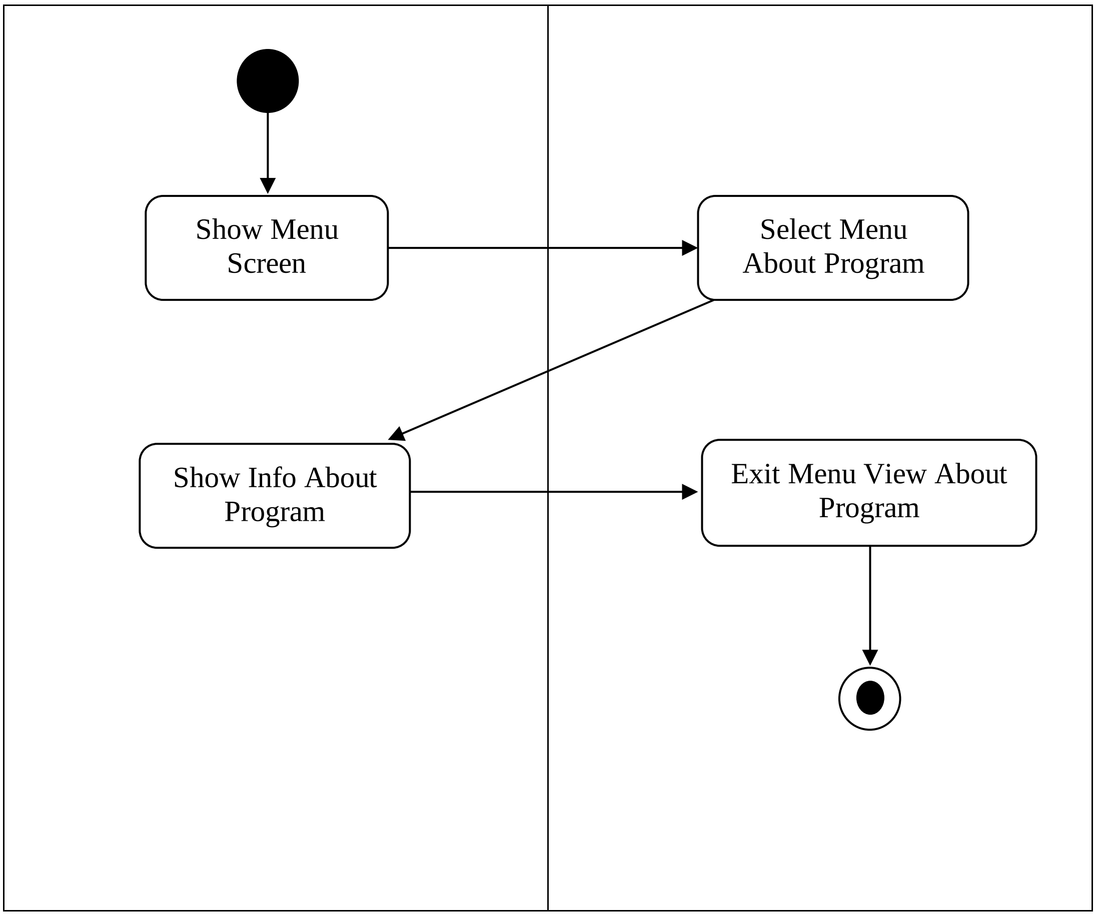


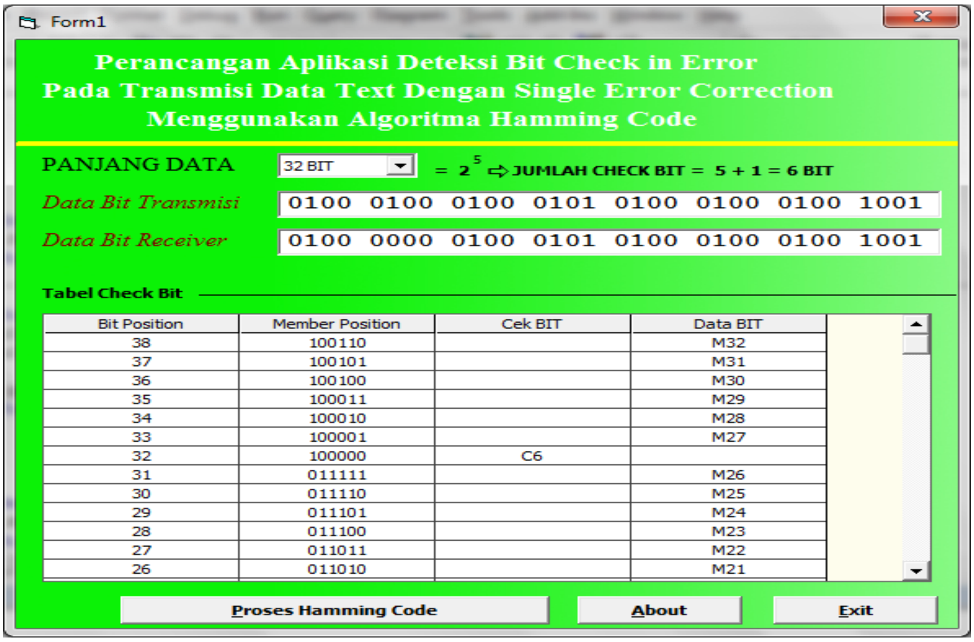
Figure 4. Activity Diagram of the Process See About the Program

5, Error Detection with Hamming Code Algorithm

Suppose the length of the input and output data = 32 bits, the data to be transmitted is the DEDI character.

Data Input DEDI = 0100 0100 0100 0101 0100 0100 0100 1001

While the output data received is 0100 0000 0100 0101 0100 0100 0100 1001


Form1

Perancangan Aplikasi Deteksi Bit Check in Error
Pada Transmisi Data Text Dengan Single Error Correction
Menggunakan Algoritma Hamming Code

PANJANG DATA 32 BIT = 2^5 ⇨ JUMLAH CHECK BIT = 5 + 1 = 6 BIT

Data Bit Transmisi 0100 0100 0100 0101 0100 0100 0100 1001

Data Bit Receiver 0100 0000 0100 0101 0100 0100 0100 1001

Tabel Check Bit

Bit Position	Member Position	Cek BIT	Data BIT
38	100110		M32
37	100101		M31
36	100100		M30
35	100011		M29
34	100010		M28
33	100001		M27
32	100000	C6	
31	011111		M26
30	011110		M25
29	011101		M24
28	011100		M23
27	011011		M22
26	011010		M21

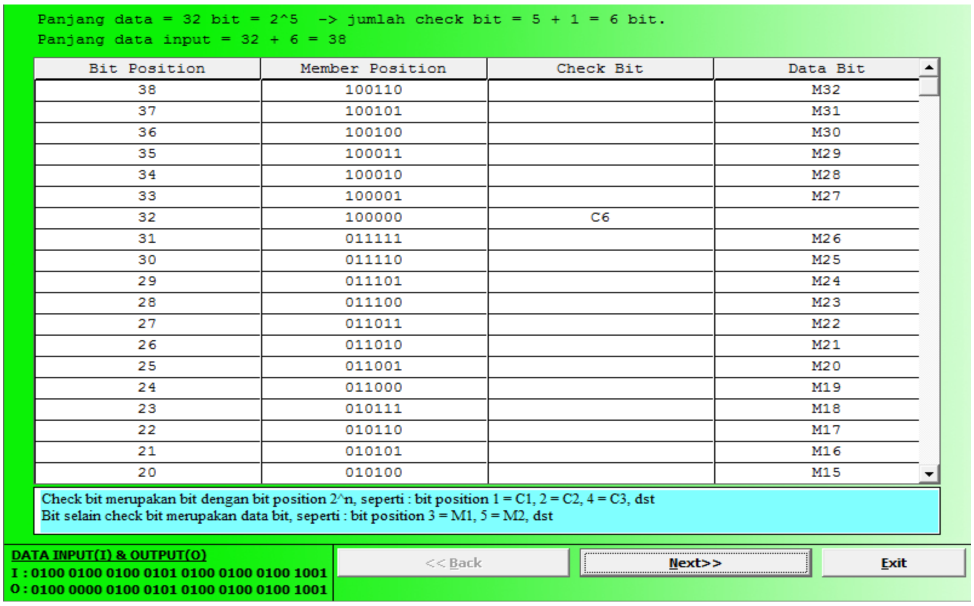
Proses Hamming Code | About | Exit

Figure 5. Main Input and Output Form

In Figure 5 above the user can choose the length of the data to be transmitted then input the transmitted data (Input Data) and re-enter the received data (Data Output) as comparison data for detecting the position where the bit has an error.

The steps for single error detection with Hamming Code are as follows:

1. Create a check bit table.


Panjang data = 32 bit = 2^5 -> jumlah check bit = 5 + 1 = 6 bit.
Panjang data input = 32 + 6 = 38

Bit Position	Member Position	Check Bit	Data Bit
38	100110		M32
37	100101		M31
36	100100		M30
35	100011		M29
34	100010		M28
33	100001		M27
32	100000	C6	
31	011111		M26
30	011110		M25
29	011101		M24
28	011100		M23
27	011011		M22
26	011010		M21
25	011001		M20
24	011000		M19
23	010111		M18
22	010110		M17
21	010101		M16
20	010100		M15

Check bit merupakan bit dengan bit position 2^n, seperti : bit position 1 = C1, 2 = C2, 4 = C3, dst
Bit selain check bit merupakan data bit, seperti : bit position 3 = M1, 5 = M2, dst

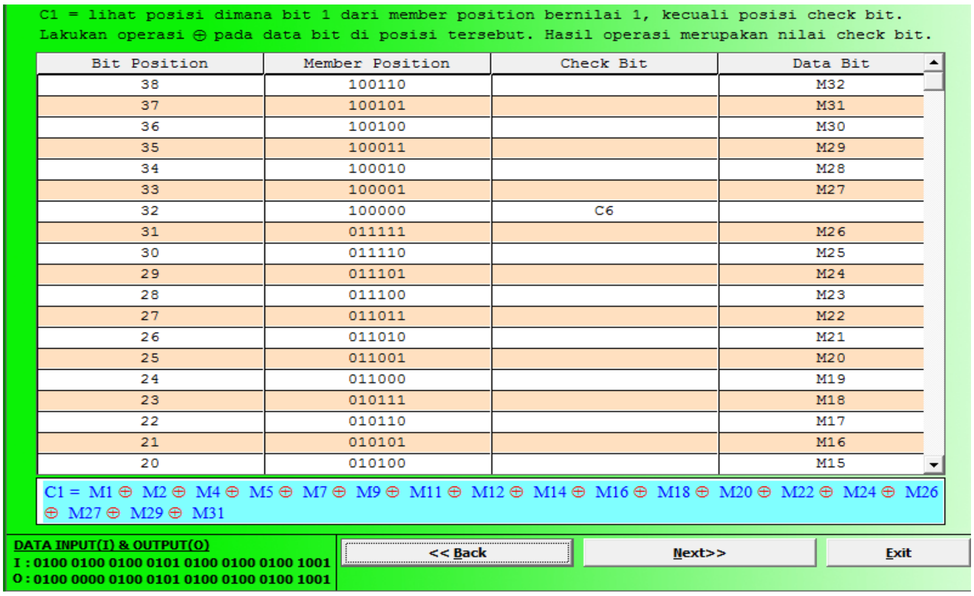
DATA INPUT(I) & OUTPUT(O)
I : 0100 0100 0100 0101 0100 0100 0100 1001
O : 0100 0000 0100 0101 0100 0100 0100 1001

<< Back | Next>> | Exit

Figure 6. Creating a check bit table for input and output data

After the input data and output data are inputted, the next step is to create a check bit table according to the length of the data that has been obtained.

2. Find the formula of the check bit - 1.



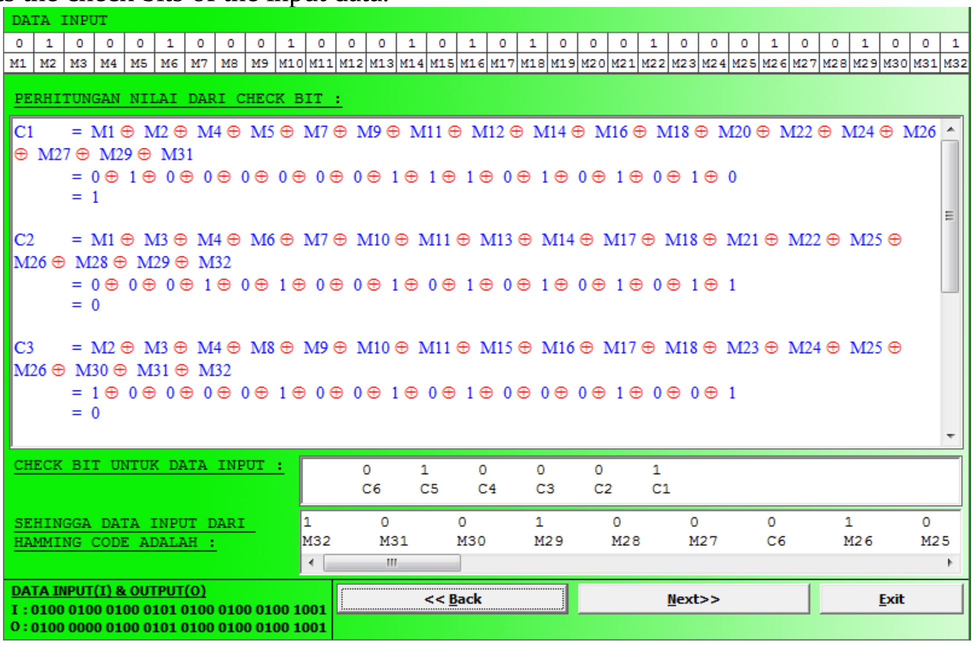
C1 = lihat posisi dimana bit 1 dari member position bernilai 1, kecuali posisi check bit.
Lakukan operasi ⊕ pada data bit di posisi tersebut. Hasil operasi merupakan nilai check bit.

Bit Position	Member Position	Check Bit	Data Bit
38	100110		M32
37	100101		M31
36	100100		M30
35	100011		M29
34	100010		M28
33	100001		M27
32	100000	C6	
31	011111		M26
30	011110		M25
29	011101		M24
28	011100		M23
27	011011		M22
26	011010		M21
25	011001		M20
24	011000		M19
23	010111		M18
22	010110		M17
21	010101		M16
20	010100		M15

Figure 7. Finding the formula from check bit - 1 for input and output data

After the "Next" button is pressed In Figure 7 above, the search form for the 1st check bit formula will appear and will continue until the nth bit corresponds to the number of check bits obtained in the length of the data.

3. Counts the check bits of the input data.



DATA INPUT

M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29	M30	M31	M32
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	1

Figure 8. Calculating check bits from input data

After the Check bit formula is obtained, the process of calculating the value of the input check bit is carried out using XOR logic. The "Next" button is used to display the next output data check bit calculation.

4. Count the check bits of the output data.

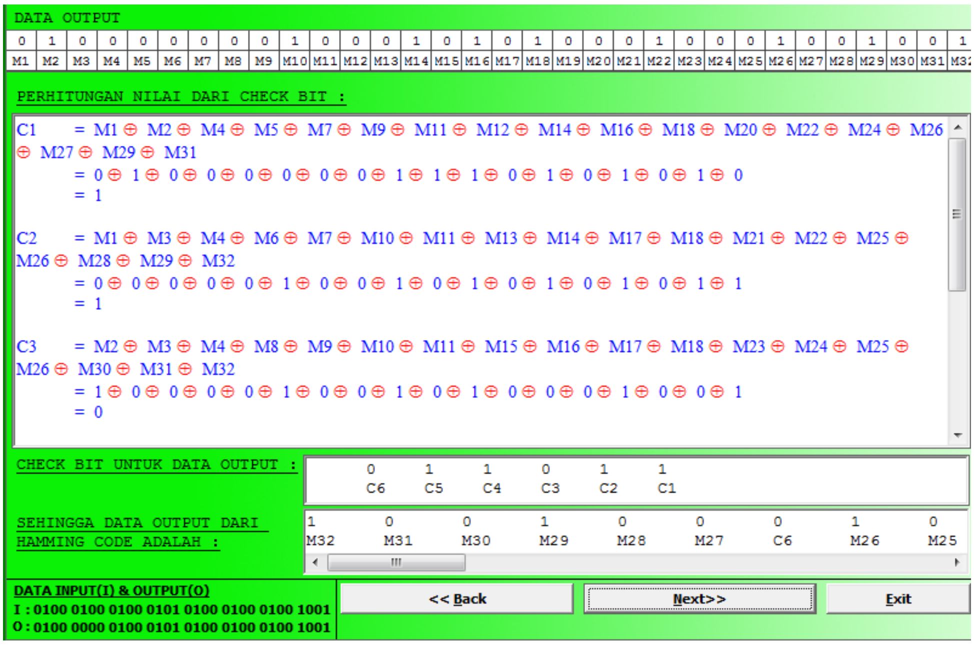


Figure 9. Calculating the check bit of the output data

At this stage, after the Check bit formula is obtained, the process of calculating the value of the output check bit is carried out using XOR logic as a comparison whether the results of the calculation are the same or not, otherwise the data bit has experienced an error.

5. Look for the position of the error (bad bit).

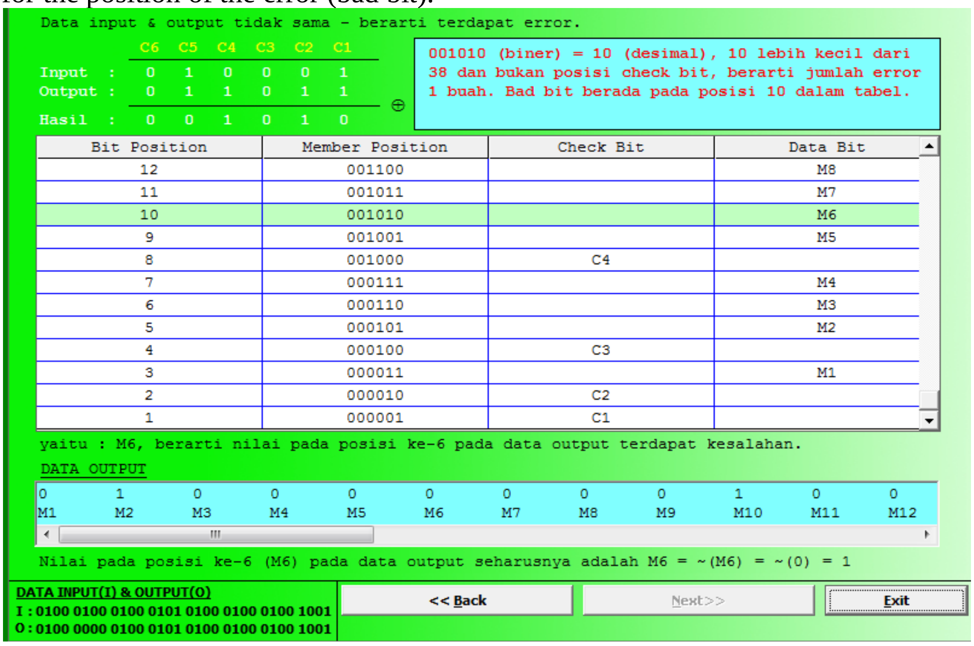


Figure 10. Finding the error position (bad bit) for Input and Output data

In Figure 10 above is the last stage for detecting bit errors and knowing where the bits have errors. The "Back" button is used to show the steps before the bad bit search process, while the "Exit" button is to exit the program when the button is pressed.

4. CONCLUSION

From the previous discussion, it can be concluded:

Bit check-in error detection can be done if the text data sent or transmitted to the receiver has changed so that the data sent is not the same as the data received by the receiver. In its application the Hamming Code Algorithm in detecting the check-in error bit is only capable ofperform one-bit damage detection or called single-bit error detection. On the other hand, Hamming code algorithm cannot detect if there is more than one bit error. If there is such a case, only one bit error is detected.

Reference

1. Jogyanto HM, "Analysis & Design", ANDI Publisher, Yogjakarta, 2005
2. Ariyus, Doni & Andri, Rum KR, “Data Communication”, Andi Publisher, Yogyakarta, 2008
3. Albar, Ahmad Alfi, Poltak and Sani, Arman. Designing Error Detection System And Error Correction System Using Hamming Code Method In Text Data Delivery.
4. Gupta, Brajesh K. and Rajeshwar Lal dua. 2012. 30 BIT Hamming Code For Error Detection and Correction with even parity and odd parity Check Method by using VHDL.
5. Saragih, Arlando Saragi and Hanapi Gunawan, 2011, Simulation of Concealing Errors in Image Using Multi Directional Interpolation (MDI) Method.
6. Purnomo, Galih, "The Hamming Method".
7. Maharani, Tamara, Pratiarso, Aries and Arifin, "Simulation of Sending and Receiving Information Using BCH Code".