Diyala Journal of Engineering Sciences

ISSN 1999-8716
Printed in Iraq

Vol. 08, No. 03, pp. 22-37, September 2015

DESIGN OF HAMMING CODE FOR 64 BIT SINGLE ERROR DETECTION AND CORRECTION USING VHDL

Adham Hadi Saleh
Assistant Lecturer, Department of Electronic engineering, College of Engineering, University of Diyala
adham.hadi@yahoo.com
(Received: 16/2/2014; Accepted: 21/9/2014)

ABSTRACT: - Hamming code is an efficient error detection and correction technique which can be used to detect single and burst errors, and correct errors. In communication system information data transferred from source to destination by channel, which may be corrupted due to a noise. So to find original information we use Hamming code.

In this paper, we have described how we can generate 7 redundancy bit for 64 bit information data. These redundancy bits are to be interspersed at the bit positions (n = 1, 2, 4, 8, 16, 32 and 64) of the original data bits, so to transmit 64 bit information data we need 7 redundancy bit generated by even parity check method to make 71 bit data string. At the destination receiver point, we receive 71 bit data, this receives data may be corrupted due to noise. In Hamming technique the receiver will decided if data have an error or not, so if it detected the error it will find the position of the error bit and corrects it. This paper presents the design of the transmitter and the receiver with Hamming code redundancy technique using VHDL. The Xilinx ISE 10.1 Simulator was used for simulating VHDL code for both the transmitter and receiver sides.

Keywords: Hamming code, error correction, error detection, even parity check method, Redundancy bits, VHDL language, Xilinx ISE 10.1 Simulator

INTRODUCTION

The theory of linear block codes is well established since many years ago. In 1948 Shannon's work showed that any communication channel could be characterized by a capacity at which information could be reliably transmitted. In 1950, Hamming introduced a single error correcting and double error detecting codes with its geometrical model (1).

In telecommunication, Hamming code as a class of linear block codes is widely used, Hamming codes are a family of linear error-correcting codes that generalize the

Hamming(7,4)-code. Hamming codes can detect up to two-bit errors or correct one-bit errors. By contrast, the simple parity code cannot correct errors, and can detect only an odd number of bits in error. Hamming codes are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance 3 [2, 3].

Due to the limited redundancy that Hamming codes add to the data, they can only detect and correct errors when the error rate is low. This is the case in computer memory (Error Checking & Correction, ECC memory), where bit errors are extremely rare and Hamming codes are widely used. In this context, an extended Hamming code having one extra parity bit is often used. Extended Hamming codes achieve a Hamming distance of 4, which allows the decoder to distinguish between when at most one bit error occurred and when two bit errors occurred. In this sense, extended Hamming codes are single-error-correcting (SED) and double-error-detecting (DED). The ECC functions described in this application note are made possible by Hamming code, a relatively simple yet powerful ECC code. It involves transmitting data with multiple check bits (parity) and decoding the associated check bits when receiving data to detect errors. The check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several check bits show parity errors after decoding the retrieved codeword. The combination of these check bit errors display the nature of the error. In addition, the position of any single bit error is identified from the check bits [2, 4].

Error detection and correction codes are used in many common systems including: storage devices (CD, DVD, DRAM), mobile communication (cellular telephones, wireless, microwave links), digital television, and high-speed modems. Hamming codes is a Forward Error Correction (FEC), as a fundamental principle of channel coding techniques, provides the ability to correct transmission errors without requiring a feedback channel for a correct retransmission. The exact correction capability of an FEC code varies depending on the coding schemes used [5, 6].

The basic idea for achieving error detection is to add some redundancy bits to the original message to be used by the receivers to check consistency of the delivered message and to recover the correct data. Error-detection schemes can be either systematic or non-systematic: In a systematic scheme the transmitter sends the original data and attaches a fixed number of check bits. That is derived from the data bits by some deterministic algorithm. If only error detection is required a receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits if the values do not match an error has occurred at some point during the transmission. In a system that uses a

non-systematic code the original message is transformed into an encoded message that has at least as many bits as the original message. Error correction & detection Hamming code may perform using Even parity or Odd parity [7, 8].

Suppose, we want to transmit 64 information data bit is "0101010101010101010101010101010101010101010101010101010101010101"which equal in hexadecimal"5555555555555555". For this 64 bit information data we need 7 redundancy bits using even parity method. After generating redundancy bits, add these bits to 64 bit information data for making 71 bit data string for transmission at source end. How we can generate 7 redundancy bits for 64 bit information data for making 71 bit data string for transmission at source end by using even parity method will be discussed in details at communication with even parity method. At destination receiver receives 71 bit data string from channel and check it, is it corrupted or not? If it is corrupted then the receiver find the error location according to parity check method correct it.

2. Error Detection and Correction

For a given practical requirement, detection of errors is simpler than the correction of errors. The decision for applying detection or correction in a given code design depends on the characteristics of the application. When the communication system is able to provide a full duplex transmission (that is, a transmission for which the source and the destination can communicate at the same time, and in a two way mode, as it is in the case of telephone connection, for instance), codes can be designed for detecting errors, because the correction is performed by requiring a repetition of the transmission [3, 8].

These schemes are known as automatic repeat request (ARQ) schemes. In any ARQ system there is the possibility of requiring a retransmission of a given message. There are on the other hand communication systems for which the full-duplex mode is not allowed. An example of one of them is the communication system called paging, a sending of alphanumerical characters as text messages for a mobile user. In this type of communication system, there is no possibility of requiring retransmission in the case of a detected error, and so the receiver has to implement some error-correction algorithm to properly decode the message. This transmission mode is known as forward error correction (FEC) [3, 8].

3. HAMMING CODE

Hamming code is a linear error-correcting code named after its inventor, Richard Hamming. Hamming codes can detect up to two simultaneous bit errors, and correct single-bit error. By contrast, the simple parity code cannot correct errors, and can only detect an odd

number of errors. In 1950 Hamming introduced the (7, 4) code. It encodes 4 data bits into 7 bits by adding three parity bits. Hamming (7, 4) can detect and correct single – bit errors. With the addition of overall parity bit, it can also detect (but not correct) double bit errors. Hamming code is an improvement on parity check method. It can correct 1 error bit only [9].

Hamming code used two methods (even parity and odd parity) for generating redundancy bit. The number of redundancy bits depends on the size of information data bits as shown below [8, 9, 10, 11]:

$$2^r \geq m + r + 1 \qquad (1)$$

Where r = number of redundancy bit.

m = number of information data bits.

According to (1), 7 redundancy bits required for a 64 input data bits. Hamming-based codes are widely used in memory systems for reliability improvements. The algorithm consists of two phases: encoding and decoding. Hamming encoding involves deriving a set of parity check bits over data bits. These parity check bits are concatenated or merged with the data bits. These extra bits are called redundancy bits. We add these redundancy bits to the information data at the source end and remove at destination end. Presence of redundancy bit allows the receiver to detect or correct corrupted bits. The concept of including extra information in the transmission for error detection is a good one. But in place of repeating the entire data stream, a shorter group of bits may be added to the end of each unit. This technique is called redundancy because the extra bits are redundant to the information [8, 12, 13, 14].

3.1 Hamming Encoder

In communication system need two main part one of them is the source for sending data and another is the destination to receive the transmitted data. Even parity check method count the number of one`s if number of one`s are even it adds zero (0) otherwise it adds one (1) [8].

At the transmitter the 64 bit information data needs 7 redundancy bit according to equation (1). Suppose, these redundancy bits are R(1),R(2),R(4),r(8),R(16) R(32),R(64),and to calculate these redundancy bits easily done by XORing operation of the original data bit positions as shown below:

$$R(1) = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7 \oplus D9 \oplus D11 \oplus D12 \oplus D14 \oplus D16 \oplus D18 \oplus D20 \oplus D22 \oplus D24 \oplus D26 \oplus D27 \oplus D29 \oplus D31 \oplus D33 \oplus D35 \oplus D37 \oplus D39 \oplus D41 \oplus D43 \oplus D45 \oplus D47 \oplus D49 \oplus D51 \oplus D53 \oplus D55 \oplus D57 \oplus D58 \oplus D60 \oplus D62 \oplus D64 \qquad (2)$$

R(2) = D1⊕ D3 ⊕ D4⊕ D6 ⊕ D7 ⊕ D10 ⊕D11 ⊕ D13⊕ D14 ⊕ D17 ⊕ D18 ⊕ D21 ⊕ D22 ⊕ D25 ⊕ D26 ⊕ D28⊕ D29 ⊕ D32 ⊕ D33 ⊕ D36 ⊕ D37 ⊕ D40 ⊕ D41 ⊕ D44 ⊕ D45 ⊕ D48 ⊕ D49 ⊕ D52 ⊕ D53 ⊕ D56 ⊕ D57 ⊕ D59 ⊕ D60 ⊕ D63⊕ D64. (3)

R(4) = D2⊕ D3 ⊕ D4⊕ D8 ⊕ D9 ⊕ D10 ⊕D11 ⊕ D15⊕ D16 ⊕ D17 ⊕ D18 ⊕ D23 ⊕ D24 ⊕ D25 ⊕ D26 ⊕ D30⊕ D31 ⊕ D32 ⊕ D33 ⊕ D38 ⊕ D39 ⊕ D40 ⊕ D41 ⊕ D46 ⊕ D47 ⊕ D48 ⊕ D49 ⊕ D54 ⊕ D55 ⊕ D56 ⊕ D57 ⊕ D61 ⊕ D62 ⊕ D63⊕ D64. (4)

R(8) = D5⊕ D6 ⊕ D7⊕ D8 ⊕ D9 ⊕ D10 ⊕D11 ⊕ D19⊕ D20 ⊕ D21 ⊕ D22 ⊕ D23 ⊕ D24 ⊕ D25 ⊕ D26 ⊕ D34⊕ D35 ⊕ D36 ⊕ D37 ⊕ D38 ⊕ D39 ⊕ D40 ⊕ D41 ⊕ D50 ⊕ D51 ⊕ D52 ⊕ D53 ⊕ D54 ⊕ D55 ⊕ D56 ⊕ D57. (5)

R(16) = D12⊕ D13 ⊕ D14⊕ D15 ⊕ D16 ⊕ D17 ⊕D18 ⊕ D19⊕ D20 ⊕ D21 ⊕ D22 ⊕ D23 ⊕ D24 ⊕ D25 ⊕ D26 ⊕ D42⊕ D43 ⊕ D44 ⊕ D45 ⊕ D46 ⊕ D47 ⊕ D48 ⊕ D49 ⊕ D50 ⊕ D51 ⊕ D52 ⊕ D53 ⊕ D54 ⊕ D55 ⊕ D56 ⊕ D57. (6)

R(32) = D27⊕ D28 ⊕ D29⊕ D30 ⊕ D31 ⊕ D32 ⊕D33 ⊕ D34⊕ D35 ⊕ D36 ⊕ D37 ⊕ D38 ⊕ D39 ⊕ D40 ⊕ D41 ⊕ D42⊕ D43 ⊕ D44 ⊕ D45 ⊕ D46 ⊕ D47 ⊕ D48 ⊕ D49 ⊕ D50 ⊕ D51 ⊕ D52 ⊕ D53 ⊕ D54 ⊕ D55 ⊕ D56 ⊕ D57. (7)

R(64) = D58⊕ D59 ⊕ D60⊕ D61⊕ D62 ⊕D63 ⊕ D64. (8)

The value of redundancy bits can be calculated using an even parity check method. The value of redundancy bit can be calculated by XORing of different locations of information data bits, as shown in Figure (1). The calculation of redundancy bit of Hamming encoder is done by VHDL code written in Xilinx ISE 10.1 project navigator window as shown in Figure (2).

Suppose, we want to transmit 64 information data bit is "0101010101010101010101010101010101010101010101010101010101010101"which equal in Hexadecimal "5555555555555555".Calculation for redundancy bits, by XORing input bit, according to hamming code with even parity redundancy the transmitted data will be 71 bits "0100 1011 0101 0100 1010 1010 1010 1011 0101 0101 0101 0101 0101 0101 0101 0101 1010 101" which equal in Hexadecimal "25AA5555AAAAAAAA55" as explained in Figure (1).

The simulation of Hamming code generation code for VHDL code for source end as shown in Figure (3) and Figure(4)**.** The schematic circuit diagram is shown in Figure (5)**,** and Hamming encoder design status is shown in Table (1).

3.2 Hamming Decoder

At the receiver side 71 bit information data is received, 64 bit encrypted information data and redundancy 7 bits. At the destination, the receiver receives 71 bit encrypted data and check for any error that may occurred. If any error is occurred, receiver find the error location and corrects it. Hamming decoder detect the error by EXORing data and corrected it by a NOT gate (8).Then the receiver removes the redundancy bit and get the original data information, if there are no error the result of even parity check was (0000000) else it detect the location of error bit as shown in Figure (6).

The detection and correction of a single error bit by Hamming decoder is done by VHDL code written in Xilinx ISE 10.1 project navigator window as shown in Figure (7).

So according the supposed example the received data with no error (noiseless channel) will be “0100 1011 0101 0100 1010 1010 1010 1011 0101 0101 0101 0101 0101 0101 0101 0101 1010 101” which equal in Hexadecimal "25AA5555AAAAAAAA55". As shown in Figure (8) and Figure (9), where 'ded means detection error' and 'ne means no error.

Suppose, transmitter of source end transmit data is “0100 1011 0101 0100 1010 1010 1010 1011 0101 0101 0101 0101 0101 0101 0101 0101 1010 101” which equal in Hexadecimal "25AA5555AAAAAAAA55" and at destination receiver received error data is “0110 1011 0101 0100 1010 1010 1010 1011 0101 0101 0101 0101 0101 0101 0101 0101 1010 101” which equal in Hexadecimal "35AA5555AAAAAAAA55", Hamming decoder at first detect the error location by even parity checking method and corrected it as shown in Figure (10)

According to Hamming detection method take even parity check to get the address of error location is = 0000011 (the third bit at the input data). After getting the location of error bit, the receiver correct, that error bit by replacing zero by one and one by zero. To produce the actual transmitted data.

We write VHDL code to find the error bit location, correction it and decrypt this encrypted data. Simulated results for destination end shown in Xilinx ISE 10.1 Simulation window which shows 71 bit receives encrypted data string and 64 bit actual error free information data string after correction the errors, as shown in Figure (11) and Figure (12). Where Figure (13) show schematic circuit diagram of Hamming decoder. The design status of Hamming decoder is shown in Table (2).

4. CONCLUSION

As a conclusion, Hamming code error detection and correction with even parity check method can be design using 64 bits data string in VHDL and can be implemented in FPGA. it

speed up the communication as we can encode the total data bits as a whole and send as soon, so there are no need for data splitting, therefore more combination (more information in a single frame) of data can be transmitted easily. The complexity of circuit also reduced for regenerating actual information data from encrypted corrupt received data at destination end by using of the same method at the source end, so the original data can be correctly recovered.

5. REFERENCES

1) Joanne Gomes and B. K. Mishra "DOUBLE ERROR CORRECTING LONG CODE" International Journal of Computer Networks & Communications (IJCNC) Vol.2, No.5, September 2010.
2) en.wikipedia.org/wiki/Hamming code.
3) Moon, Todd K. "Error Correction Coding", New Jersey, John Wiley & Sons, (2005).
4) Simon Tam "Single Error Correction and Double Error Detection" XILINIX, XAPP645 (v2.2) August 9, 2006.
5) Brajesh Kumar Gupta, M. Tech Scholar and Associate prof. Rashmi Sinha " Novel Hamming code for error correction and detection of higher data bits using VHDL" International Journal of Scientific & Engineering Research, Volume 4, Issue 4, April, 2013.
6) Patrick Faraj, Jochen Leibrich and Werner Rosenkranz "Coding Gain of Basic FEC Block-Codes in the Presence of ASE Noise" University of Kiel, Chair for Communication, Kaiserstr. 2, 24143 Kiel, Germany, IEEE, 2003.
7) Jatinder Singh and Jaget Singh "A Comparative study of Error Detection and Correction Coding Techniques" Second International Conference on Advanced Computing & Communication Technologies, IEEE, 2012.
8) Behrouz A. Forouzan, "Data communication and networking" 3nd edition McGraw Hill publication.
9) Brajesh Kumar Gupta, M. Tech Scholar and Associate prof. Rashmi Sinha "30 BIT Hamming Code for Error Detection and Correction with Even Parity and Odd Parity Check Method by using VHDL" International Journal of Computer Applications (0975 – 8887) Volume 35– No.13, December 2011.
10) Mr. Brajesh Kumar Gupta and Prof. Rajeshwar Lal Dua "COMMUNICATION BY 31 BIT HAMMING CODE TRANSCEIVER WITH EVEN PARITY AND ODD PARITY CHECK METHOD BY USING VHDL" International Journal Of Computational Engineering Research / Vol. 2 | Issue No.2 |278-288, Mar-Apr 2012

11) U. K. Kumar, and B. S. Umashankar "Improved Hamming Code for Error Detection and Correction" IEEE, 2007.

12) Davit Mirzoyan "Fault-Tolerant Memories in FPGA based Embedded Systems" M.SC thesis, Dept. of Information and Communication Technology, Royal Institute of Technology (KTH), Sweden, Stockholm , June 2009.

13) Brajesh Kumar Gupta Rajeshwar Lal Dua "Various Methodologies used for 25 Bit Information Data String Communication through Hamming Code" International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 2– No.2, May 2012.

14) S. Balochi, T. Arslani, and A. Stoicaiii "Efficient Error Correcting Codes for On-Chip DRAM Applications for Space Missions" IEEE, 2005.

Table (1): Hamming encoder design status.

HAMMINGENCODE64 Project Status			
Project File:	HAMMINGENCODE64.ise	Current State:	Synthesized
Module Name:	hamenc	• Errors:	No Errors
Target Device:	xc3s200-4ft256	• Warnings:	No Warnings
Product Version:	ISE 10.1 - WebPACK	• Routing Results:	
Design Goal:	Balanced	• Timing Constraints:	
Design Strategy:	Xilinx Default (unlocked)	• Final Timing Score:	

HAMMINGENCODE64 Partition Summary	[-]
No partition information was found.	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	28	1920	1%
Number of 4 input LUTs	50	3840	1%
Number of bonded IOBs	135	173	78%

Table (2): Hamming decoder design status.

HAMMINGdecoder64 Project Status			
Project File:	HAMMINGdecoder64.ise	Current State:	Synthesized
Module Name:	hamdec	• Errors:	No Errors
Target Device:	xc3s200-4ft256	• Warnings:	68 Warnings
Product Version:	ISE 10.1 - WebPACK	• Routing Results:	
Design Goal:	Balanced	• Timing Constraints:	
Design Strategy:	Xilinx Default (unlocked)	• Final Timing Score:	

HAMMINGdecoder64 Partition Summary	[-]
No partition information was found.	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	135	1920	7%
Number of 4 input LUTs	245	3840	6%
Number of bonded IOBs	137	173	79%
Number of GCLKs	1	8	12%

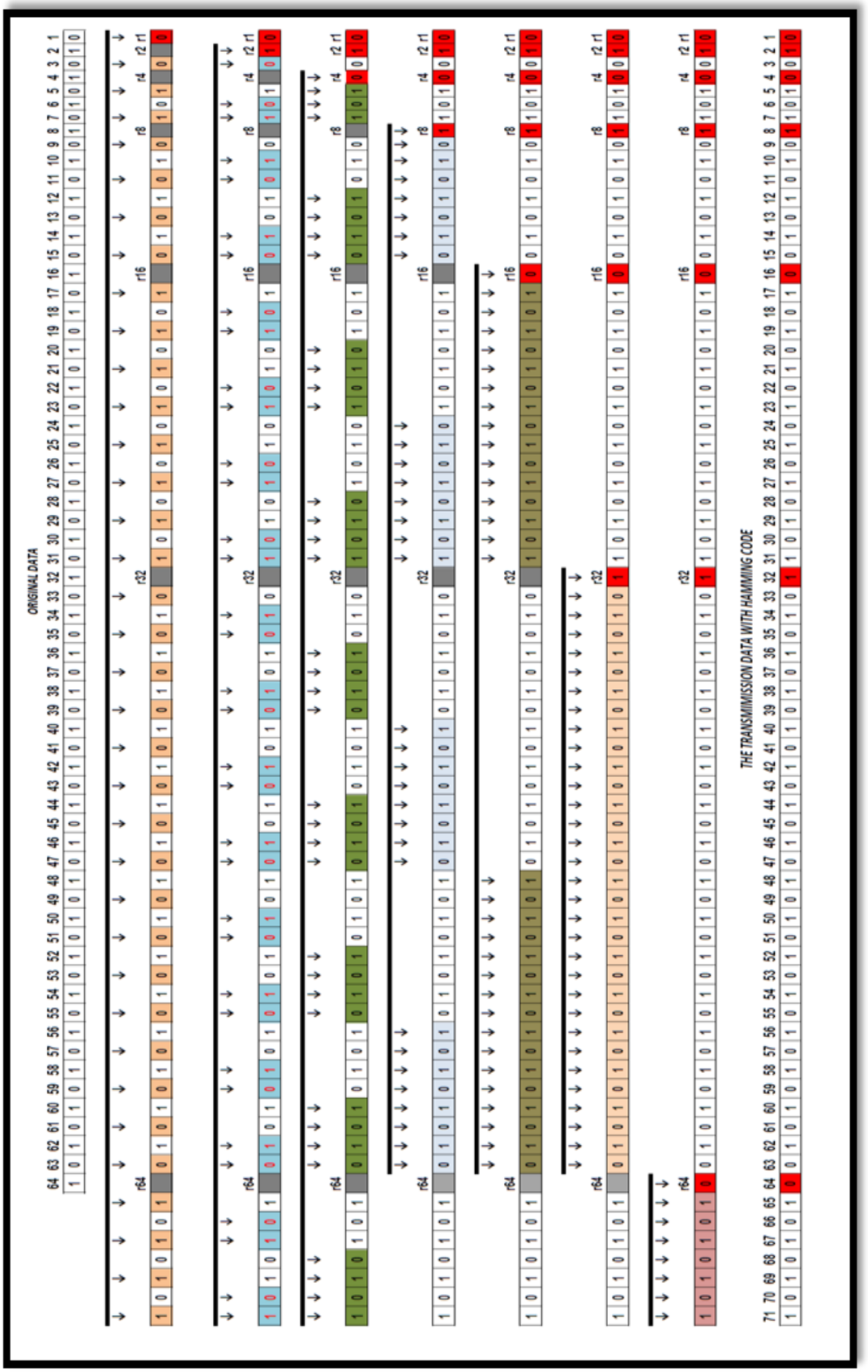

Figure (1): Hamming Code Generation for 64 Bits.

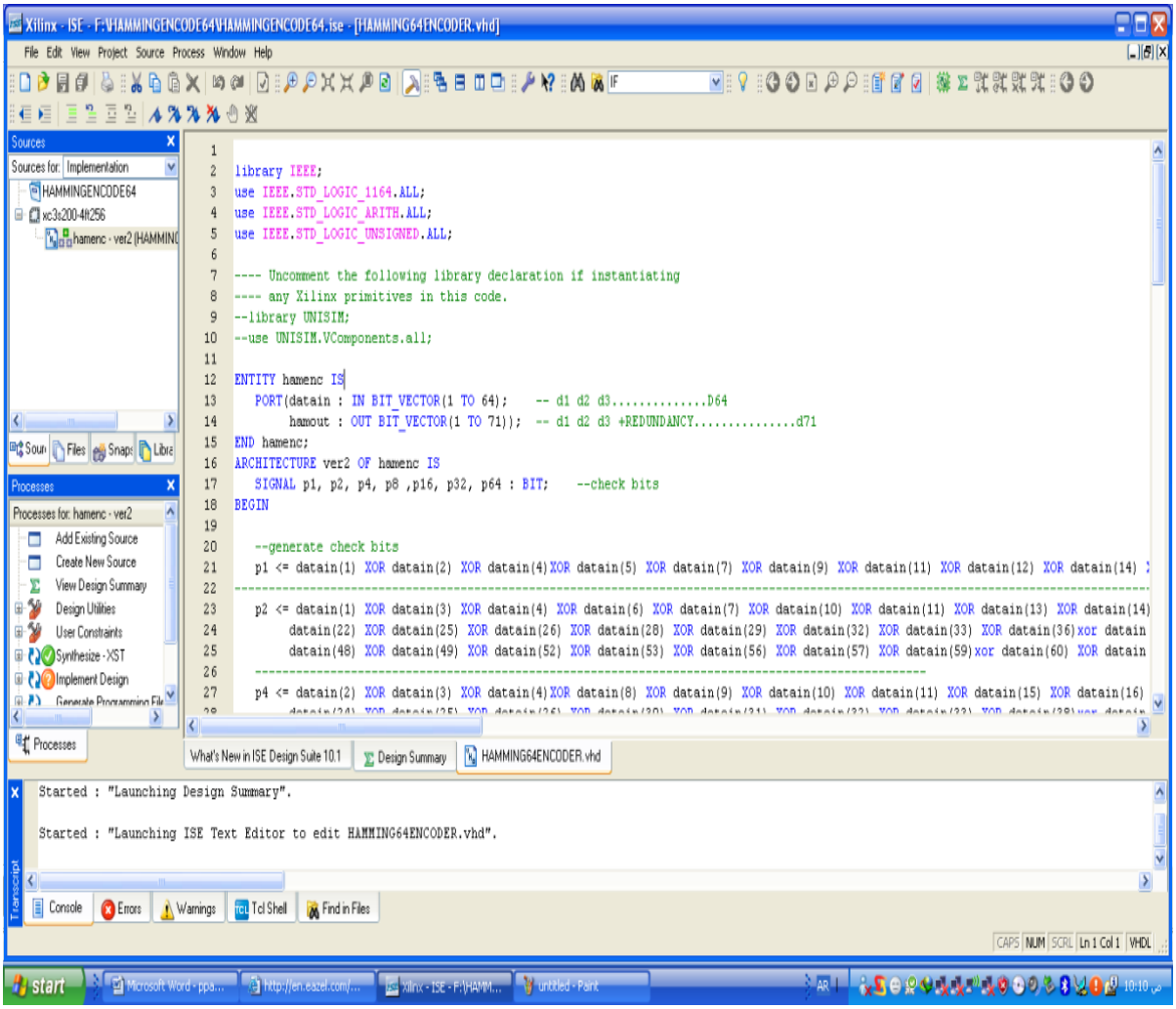

Figure (2): Hamming Encoder in VHDL Using ISE 10.1.

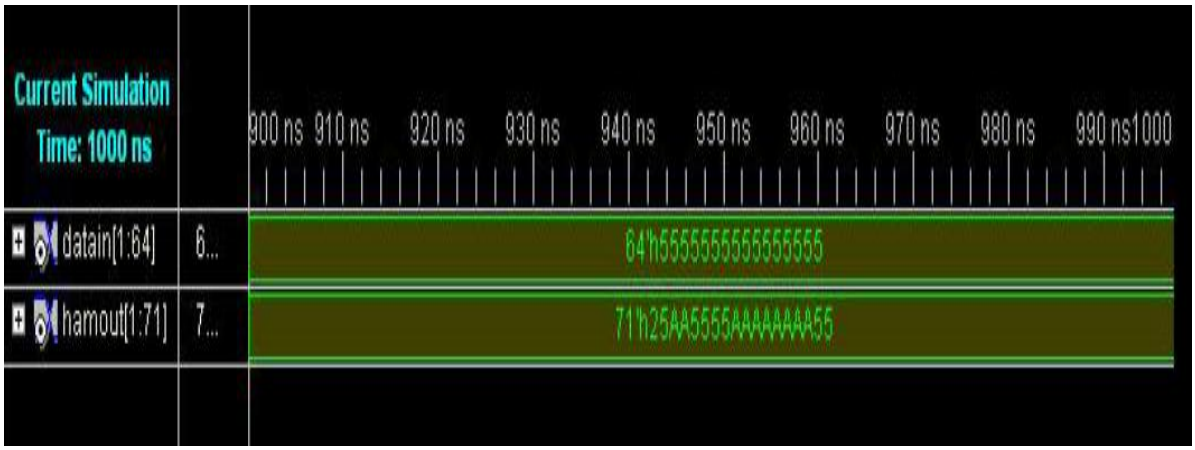

Figure (3) Hamming Code Generation for 64 Bits in Hexadecimal Form

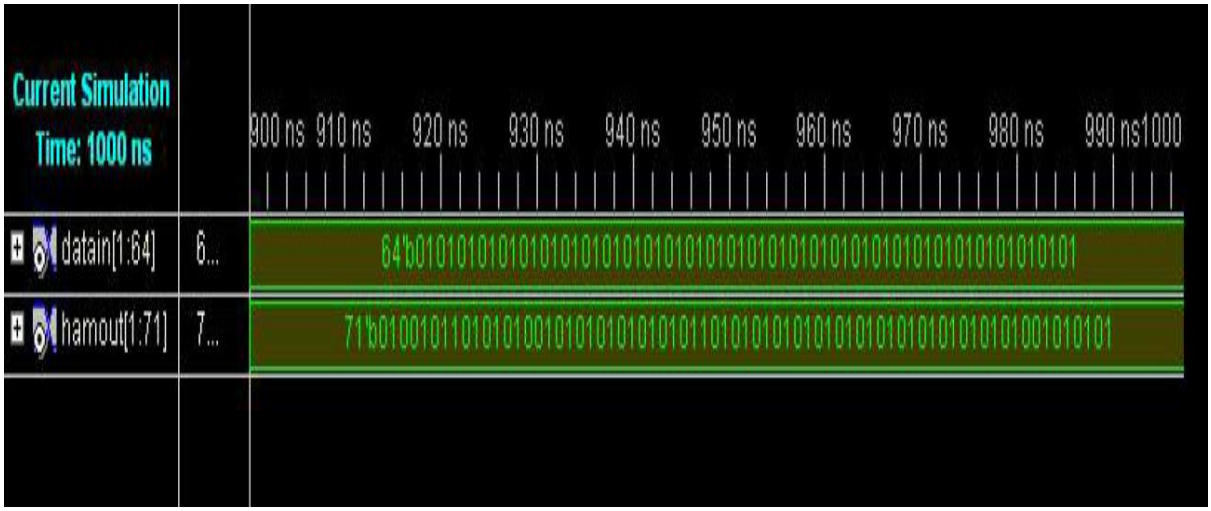

Figure (4): Hamming Code Generation for 64 Bits in Binary Form.

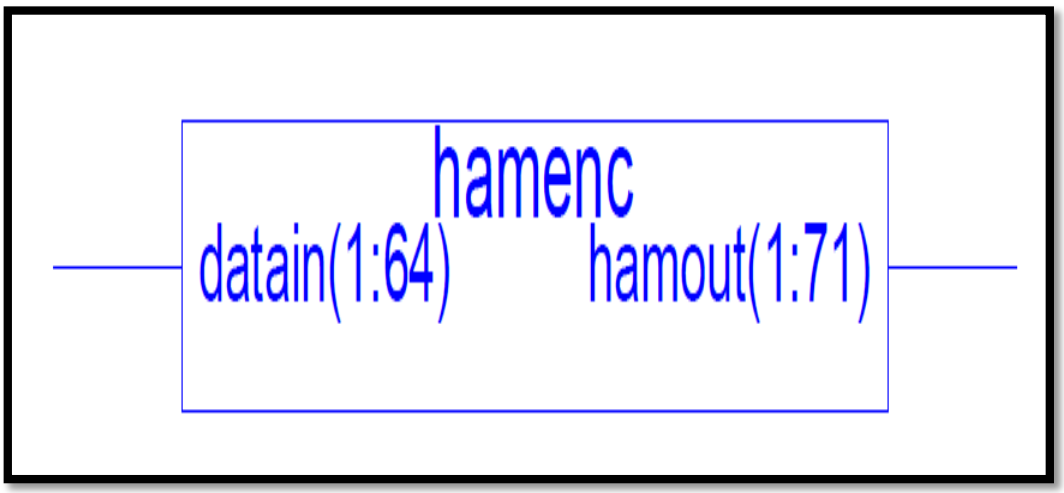

Figure (5): Schematic Circuit Diagram of Hamming Encode.

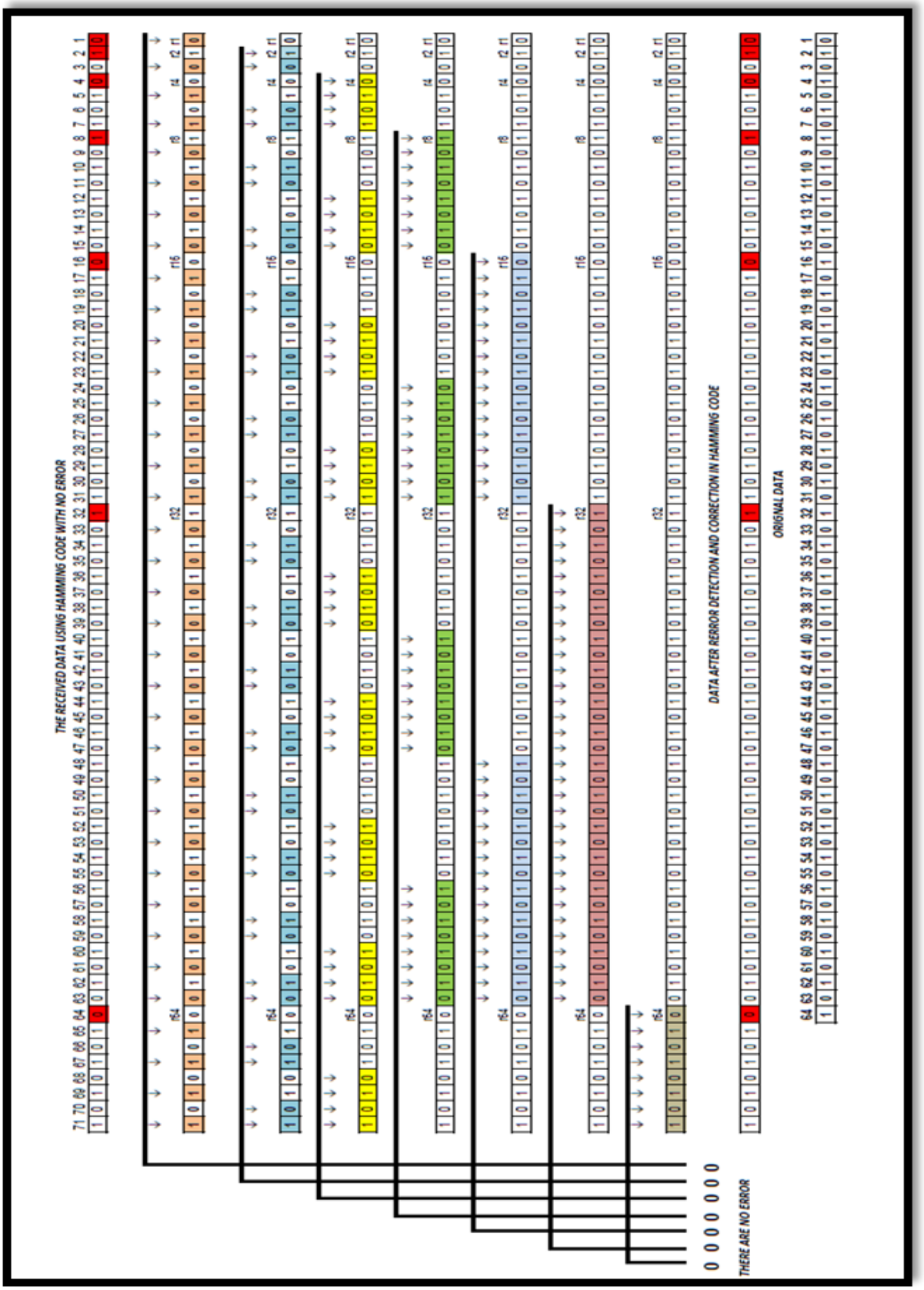

Figure (6): Hamming Code Detection Method for 64 Bits with no Error State

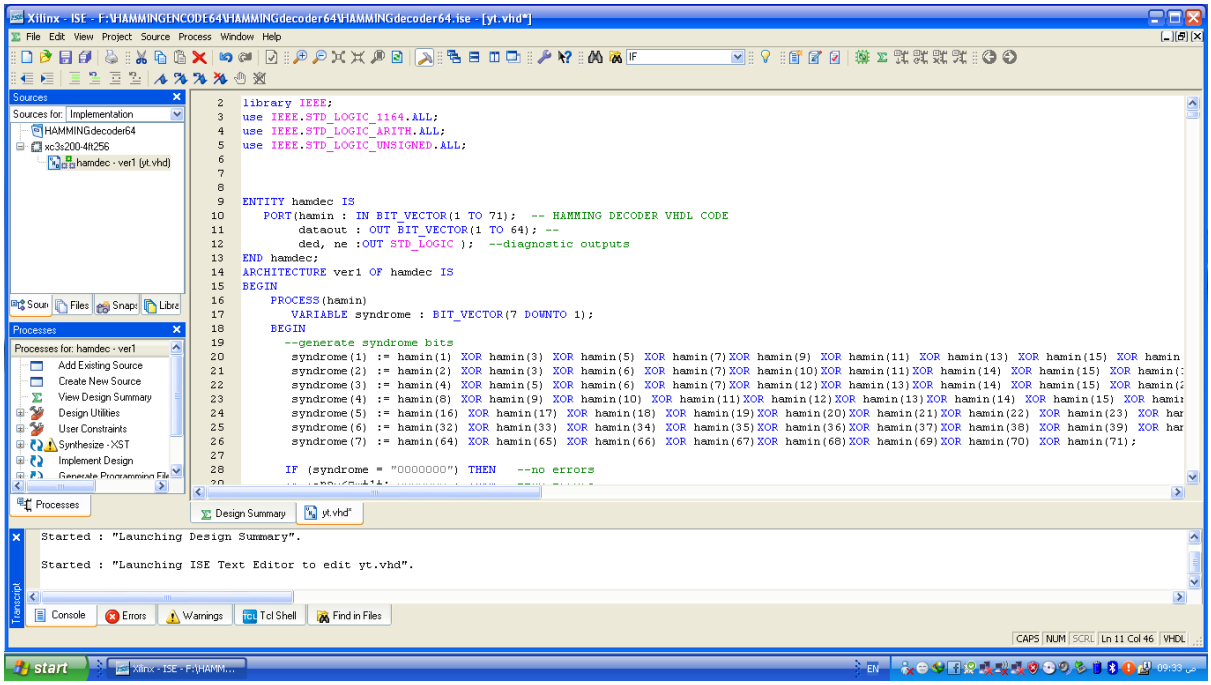

Figure (7): Hamming Decoder in VHDL Using ISE 10.1.

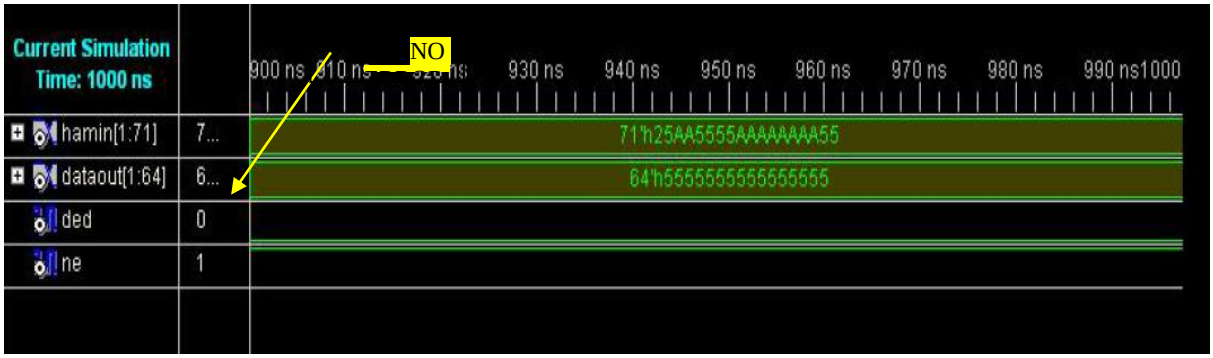

Figure (8): Hamming Code Error Detection and Correction for a Single Error in Hexadecimal Form (With no error state).

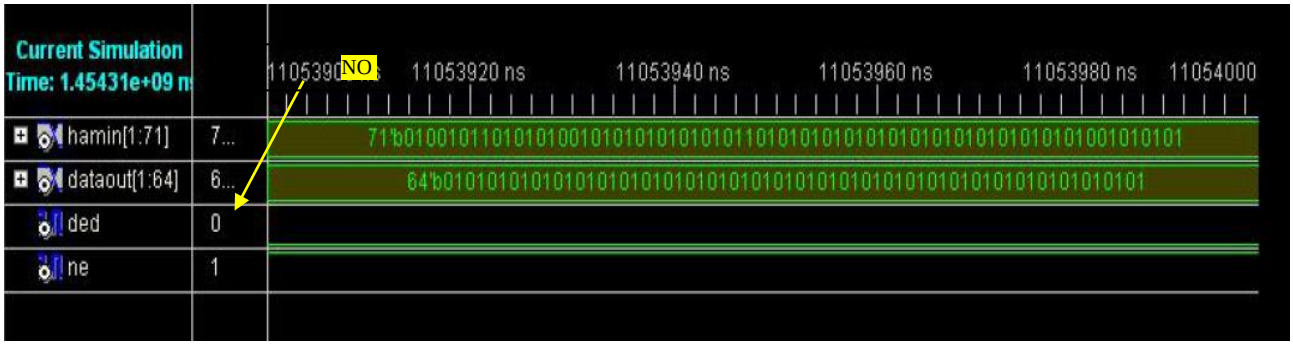

Figure (9): Hamming Code Error Detection and Correction for a Single Error in Binary Form (With no Error state).

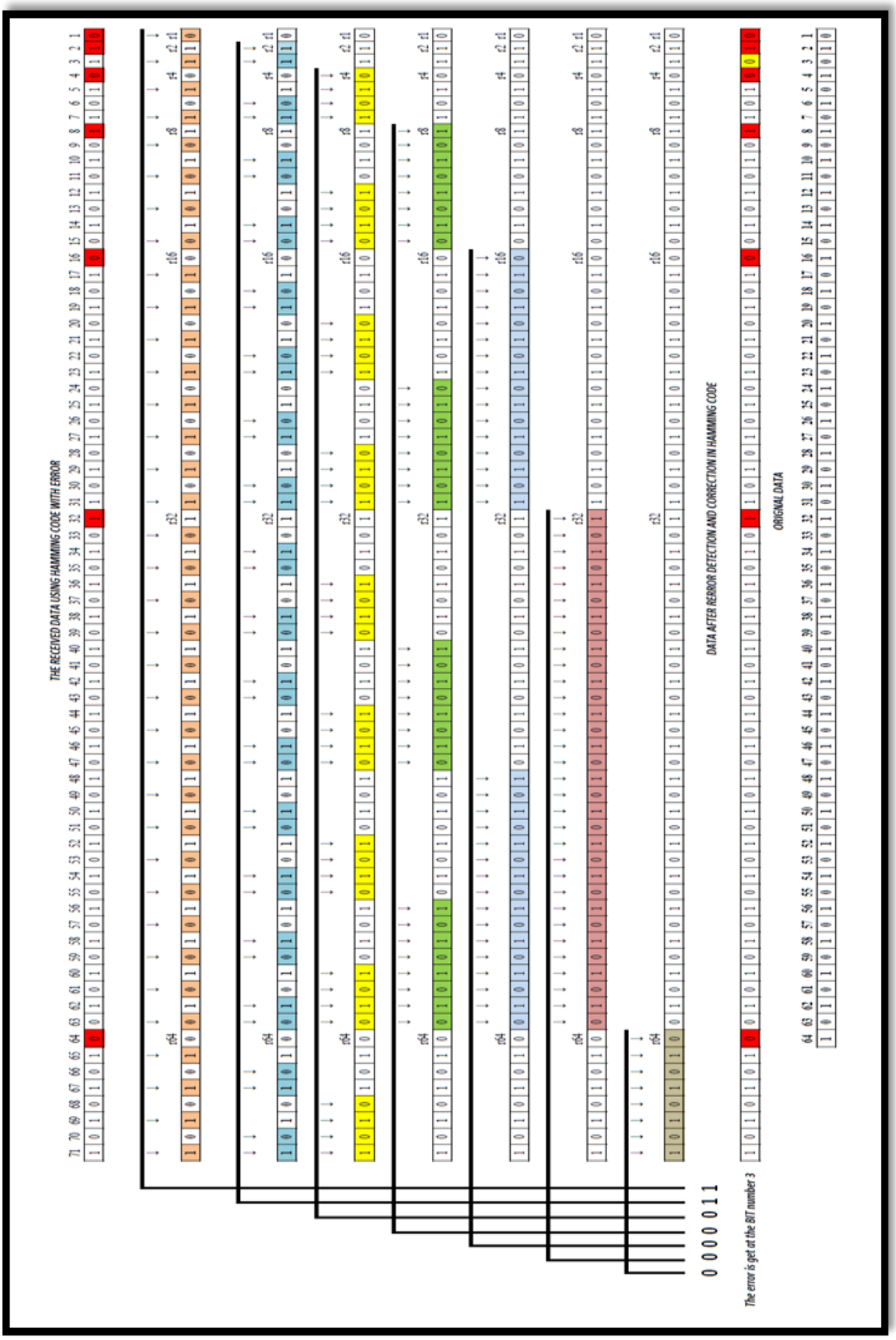

Figure (10): Hamming Code Detection Method for 64 Bits with Error State.

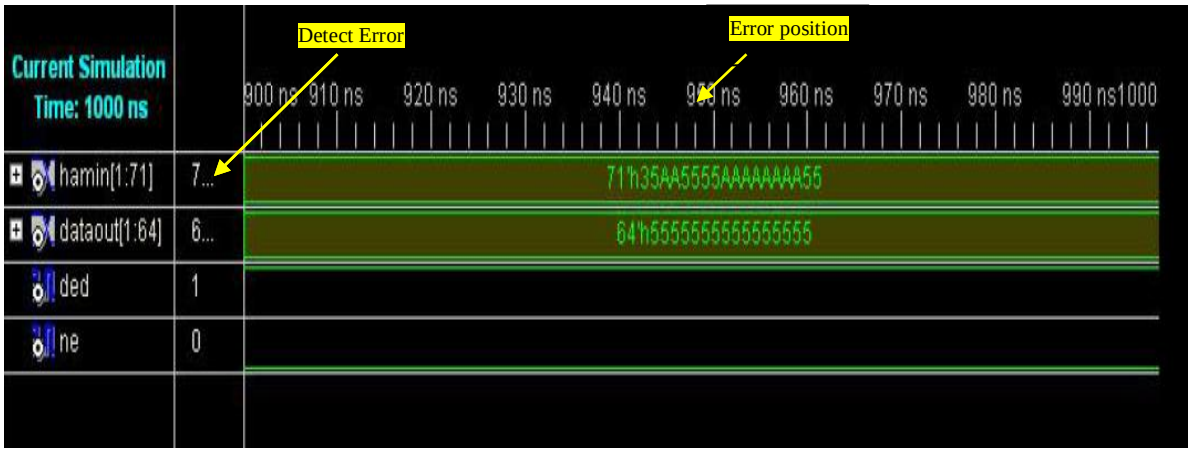

Figure (11): Hamming Decoder for a Single Error with Error Received Data (Error at Third Bit) in Hexadecimal Form.

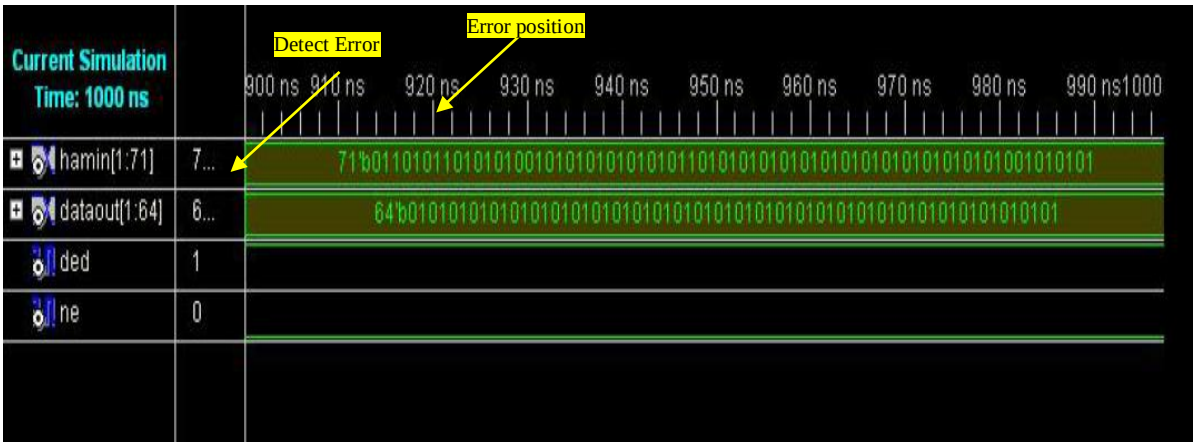

Figure (12): Hamming Decoder for a Single Error with Error Received Data (Error at Third Bit) in Binary Form.

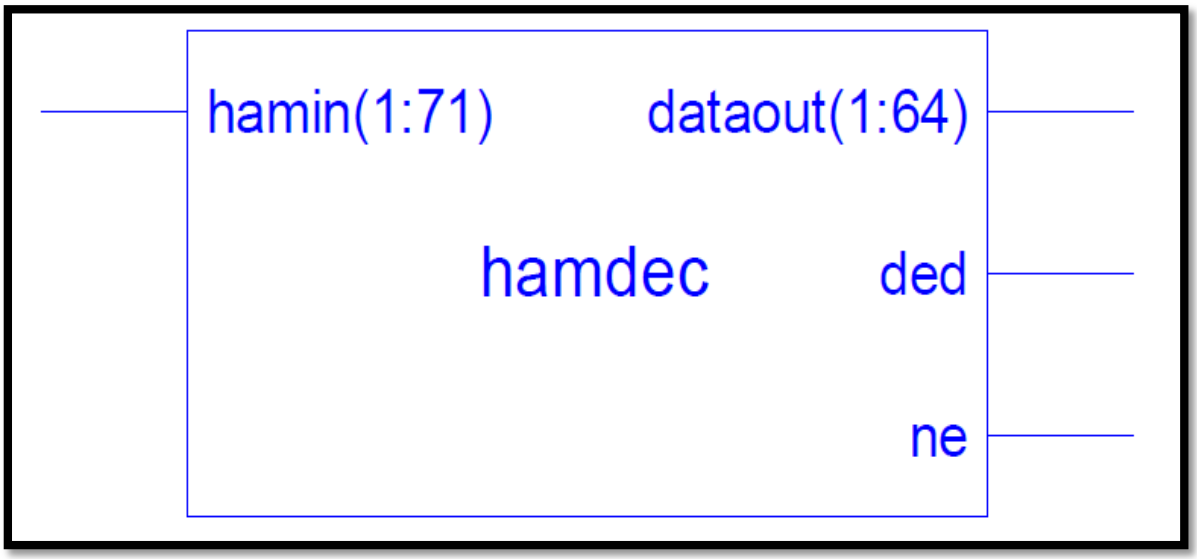

Figure (13): Schematic Circuit Diagram of Hamming Decoder.

تصميم شفرة (Hamming) لـ 64 بت لاكتشاف ومعالجة الخطأ المفرد باستخدام لغة (VHDL)

أدهم هادي صالح

مدرس مساعد، قسم الهندسة الالكترونية، كلية الهندسة، جامعة ديالى

الخلاصة :-

شفرة (Hamming) هي تقنية لاكتشاف الخطأ وتصحيحه والتي يمكن أن تستخدم لكشف خطأ واحدة أو عدة أخطاء، أن قابلية هذه التقنية في الكشف عن الخطأ الذي يحدث لبت واحد ومعالجته تتم بكفاءة عالية و التي يمكن أن تحدث عندما يتم نقل البيانات الثنائية من جهاز إلى آخر. في أنظمة الاتصالات يتم نقل البيانات من المرسل إلى المستلم خلال الوسط الناقل ، والتي قد تعرض للتلف بسبب الضوضاء. تعمل شفرة (Hamming) على العثور على المعلومات الأصلية من خلال اكتشاف الخطأ ومعالجته. هذا البحث، يصف كيف يمكننا توليد 7 بت أضافية و أضافتها للبيانات الأصلية ذات ال 64 بت. هذه البتات الاضافية ستتخلل المواقع (ن = 1، 2، 4، 8، 16، 32، 64) مع بتات البيانات الأصلي. يتم نقل البيانات والمعلومات(64 بت مع 7) لجعل سلسلة البيانات 71 بت بتقنية اكتشاف المزدوجات الثنائية عند المرسل. عند نقطة الاستلام، قد تصل البيانات ذات71 بت وجزء من البيانات تالفة بسبب الضوضاء. تتم عند المستقبل و باستخدام تقنية (Hamming) تحديد إذا كان هناك خطأ في البيانات أم لا، وفي حال الكشف عن الخطأ سيتم العثور على موقع الخطأ و تصحيحه . هنا استخدم XILINX ISE 10.1 لمحاكاة VHDL . وهو مترجم يستخدم لمحاكاة لغة VHDL ولرسم مخطط الرسم البياني، في هذا البحث تم تصميم المرسل والمستقبل لشفرة (Hamming) بتقنية التكرار باستخدام (VHDL) وهي لغةَ وصفيةِ تستخدم لتصميم الدوائر الالكترونية.

الكلمات الرئيسية: شفرة (Hamming)، تصحيح الخطأ، اكتشاف الخطأ، طريقة اكتشاف الخطا بالمزدوجات الثنائية، بت التكرار، لغة VHDL، XILINX ISE 10.1 المحاكي.