Novel Hamming code for error correction and detection of higher data bits using VHDL

Brajesh Kumar Gupta , M.Tech Scholar
Department of Electronics and Communication Engg.
National Institute of technology, Jamshedpur (India)

Associate prof. Rashmi Sinha
Department of Electronics and Communication Engg.
National Institute of technology, Jamshedpur (India)

Abstract— **Hamming code is familier for its single-bit error detection & correction capability. To provide such a capability, it introduces 4 redundancy bits in a 7-bit data item. These redundancy bits are to be interspersed at bit positions 2^n (n = 0, 1, 2, 3) with the original data bits. After error detection & correction, if any, the data bits have to be reassembled by removing the redundancy bits. In the proposed improvement, the redundancy bits will be 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 subsequent removal. In contrast, the proposed method is highly scalable without such overhead. Hence, this feature is suitable for transmission of large size data bit-streams with much lower redundancy bits per data bit ratio. In this paper we have proposed an efficient and novel Hamming Code algorithm, for finding error location and correction to optimize the load on receiver end and reduce delay in the circuit. Simulation is done using Xilinx 9.2**

Index Terms— **Hamming code, Novel Hamming code, error-correction, error-detection, even parity check method, Redundancy bits, VHDL language, Xilinx ISE 9.2 Simulator**

1 INTRODUCTION

When data is transmitted from one location to another there is always the possibility that an error may occur HAMMING code is well known for its single-bit error detection & correction capability. 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 errors[1][2][3]; thus, reliable communication is possible when the Hamming distance between the transmitted and received bit patterns is less than or equal to one. Error correction & detection Hamming code may perform using Even parity or Odd parity. Here we are comparing the novel Hamming coding approach with basic Hamming coding using Even parity Hamming code is based on the principle of adding 'r' redundancy bits to 'n' data bits such that

$$2^r \geq n + r + 1$$

For example, it needs 4 redundancy bits for a 7-bit data item and 6 redundancy bits for a 56-bit data stream. Let us say, the 7-bit data to be transmitted is 0111001. Then the 11-bit data actually transmitted is 011P100P1PP, where the P's refer to the Hamming bits that are to be calculated and interspersed at bit positions 1, 2, 4, & 8 (n = 0, 1, 2, 3) with the original data bits. The calculation of the Hamming bits is as illustrated below:

The Hamming bit at bit position 1 is selected such that there is even parity at bit positions 1, 3, 5, 7, 9, & 11 Thus the Hamming bit at position 1 will be 1.

The Hamming bit at bit position 2 is selected such that there is even parity at bit positions 2, 3, 6, 7, 10, & 11 .Thus the Hamming bit at position 2 will be 1.

The Hamming bit at bit position 4 is selected such that there is even parity at bit positions 4, 5, 6, & 7 . Thus the Hamming bit at position 4 will be 1.

The Hamming bit at bit position 8 is selected such that there is even parity at bit positions 8, 9, 10, & 11. Thus the Hamming bit at position 8 will be 0

Thus the 11-bit data actually transmitted is 011**0**100**1**1**11**. For the calculation of Hamming bits at positions 1, 2, 4, & 8, even-parity checks were performed on 6, 6, 4, & 4 bits respectively. Thus a total of 20 bits are involved in the process of Hamming bits calculations. The error-detection and correction process in Hamming code is as illustrated in Table 1 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, It is not efficient for higher data bit

TABLE 1
ERROR DETECTION AND CORRECTION USING HAMMING CODE

Recieved information include Hamming bits											Status of Parity check				Conclusion
11	10	9	8	7	6	5	4	3	2	1	S3	S2	S1	S0	
0	1	1	0	1	0	**1**	1	1	1	1	T	F	T	F	Error at bit position 5
0	1	1	**1**	1	0	0	1	1	1	1	F	T	T	T	Error at bit position 8
0	1	1	0	1	0	0	1	1	1	1	T	T	T	T	No Error

Suppose the received information is 0110101**1**1111, with an error at bit position 5. There is no even parity error at bit positions 8, 9, 10, & 11. So the status of parity check S3 is shown as T (true) in the table below. There is even parity error at bit positions 4, 5, 6, & 7. So the status of parity check S2 is shown as f (false). Similarly there is no even parity error at bit positions 2, 3, 6, 7, 10 & 11 and even parity occurs at bit positions 1, 3, 5, 7, 9, & 11. So the status of parity check S1 is shown as T and S0 is shown as F. Interpreting F as 1 and T as 0 in the status of parity check, we find that the error is at bit position 0101 i.e. 5. Bit 5, which was received as 1, is corrected as 0. Previous approach of Hamming coding [] is not efficient for error correction and detection of higher bit data items

2. PROPOSED NOVEL HAMMING CODE

Let us consider the same example. The 7-bit data to be transmitted is 0111001. The number of redundancy bits, 'r' to be appended to n-bit data is obtained such that the relation

$$(2^{r-1} - 1) \geq n$$

is satisfied. The number of redundancy bits in this method is same as that for Hamming code for some values of n. But in some cases, it will be just one more redundancy bit than needed in the Hamming code[4][8][9]. For 7-bit data, the number of redundancy bits required will be 4. Then the 11-bit data actually transmitted is PPPP0111001, where the P's refer to the redundancy bits that are to be calculated and appended at bit positions 8, 9, 10, & 11. The calculation of these bits in the proposed method is as illustrated below.

The bit at bit position 8 is selected such that there is even parity at bit positions 1, 3, 5, 7 and bit position 8. Thus the bit at position 8 will be 0.

The bit at bit position 9 is selected such that there is even parity at bit positions 2, 3, 6, 7 and bit position 9. Thus the bit at position 9 will be 1.

The bit at bit position 10 is selected such that there is even parity at bit positions 4, 5, 6, 7 and bit position 10. Thus the bit at position 10 will be 1.

The bit at bit position 11 is selected such that there is even parity considering only the redundancy bits. Thus the bit at position 11 will be 0.

Thus the 11-bit data actually transmitted is **0110**0111001. For the calculation of bits at positions 8, 9, 10, & 11, even- parity checks were performed on 5, 5, 5, & 4 bits respectively. Thus a total of 19 bits are involved in the process of calculation of redundancy bits. The error-detection and correction process in the novel method is as illustrated in Table 2.

TABLE 2
ERROR DETECTION AND CORRECTION USING IMPROVED METHOD

Recieved information including redundancy bits											Status of Parity check				Conclusion
11	10	9	8	7	6	5	4	3	2	1	S3	S2	S1	S0	
0	1	1	0	0	**0**	1	1	0	0	1	T	F	F	T	Error at bit position 6
0	1	1	**1**	0	1	1	1	0	0	1	F	T	T	F	Error at bit position 8
0	1	**0**	0	0	1	1	1	0	0	1	F	T	F	T	Error at bit position 9
0	**0**	1	0	0	1	1	1	0	0	1	F	F	T	T	Error at bit position10
1	1	1	0	0	1	1	1	0	0	1	T	F	T	T	Error at bit position11
0	1	1	0	0	1	1	1	0	0	1	T	F	T	T	No Error

TABLE 3
COMPARISON OF NUMBER OF BITS INVOLVED IN THE CALCULATION OF REDUNDANCY BITS

Length of data stream	Number of bits involved in calculation of Redundancy bits in **Hamming code**	Number of bits involved in calculation of Redundancy bits in **Novel Hamming code**
7 bit data	20 bits involved	19 bits involved
31 bit data	93 bits involved	90 bits involved
56 bit data	182 bits involved	172 bits involved

Suppose the received information is 01100**0**11001, with an error at bit position 6. There is no even parity error at bit positions 1, 3, 5, 7 & 8. So the status of parity check S0 is shown as T in the table below. There is even parity error at bit positions 2, 3, 6, 7 & 9. So the status of parity check S1 is shown as F. Similarly there is even parity error at bit positions 4, 5, 6, 7 & 10 and even parity occurs at bit positions 8, 9, 10 & 11. So the status of parity check S2 is shown as F and S3 is shown as T. Interpreting F as 1 and T as 0 in the status of parity check, we find that the error is at bit position 0110 i.e. 6. Bit 6, which was received as 0, is corrected as 1

3. EXPERIMENTAL RESULTS

As we increase the number of bits , pad to pad delay decreases in Novel Hamming encoder and decoder with respect to the Hamming encoder and decoder. Which is shown below in TABLE 4 & 5

TABLE 4
COMPARISON OF PAD TO PAD DELAY FOR 7 & 31 BITS ENCODER

CPLD TIMING REPORT		PAD TO PAD DELAY IN CIRCUIT
7 BITS	HAMMING ENCODER	9.700 ns
	NOVEL HAMMING ENCODER	11.400 ns
31 BITS	HAMMING ENCODER	30.300 ns
	NOVEL HAMMING ENCODER	30.300 ns

TABLE 5
COMPARISON OF PAD TO PAD DELAY FOR 7 & 31 BITS DECODER

CPLD TIMING REPORT		PAD TO PAD DELAY IN CIRCUIT
7 BITS	HAMMING DECODER	17.100 ns
	NOVEL HAMMING DECODER	17.100 ns
31 BITS	HAMMING DECODER	36.000 ns
	NOVEL HAMMING DECODER	30.300 ns

The calculation of redundancy bit is done by VHDL code written in Xilinx ISE 9.2 project navigator window. And simulate this VHDL code by using Xilinx ISE 9.2 Simulator and get the value of redundancy bit for 7 bits Hamming encoder 0111 and for novel Hamming encoder 0110 which is shown in table 1 & 2 [5][6][7]. the encrypted data waveform of 31 bits for both Hamming code and novel Hamming code which is shown below

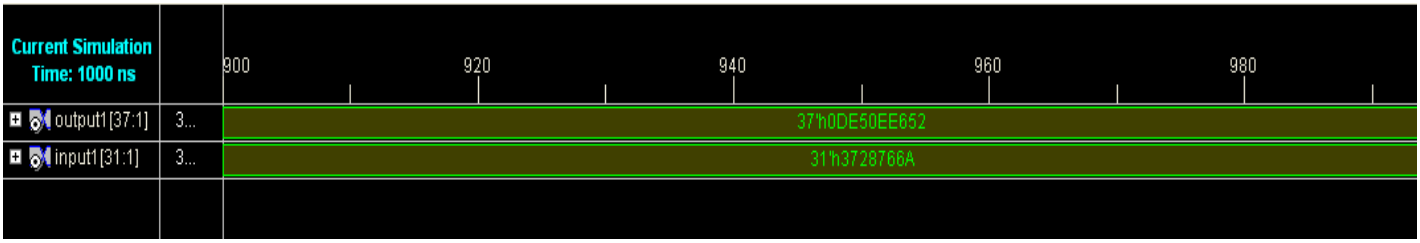
Hamming code

We are transmitting the information data of 31 bits 0110111001010000111011001101010(31'h3728766A) at Hamming encoder output the data in encrypted form 01101**1**11001010000111**1**1100110**0**101**0**0**10** (37'h0DE50EE652) . Now we generate the error at bit position 28 at the output of decoder we are able to collect the actual information data

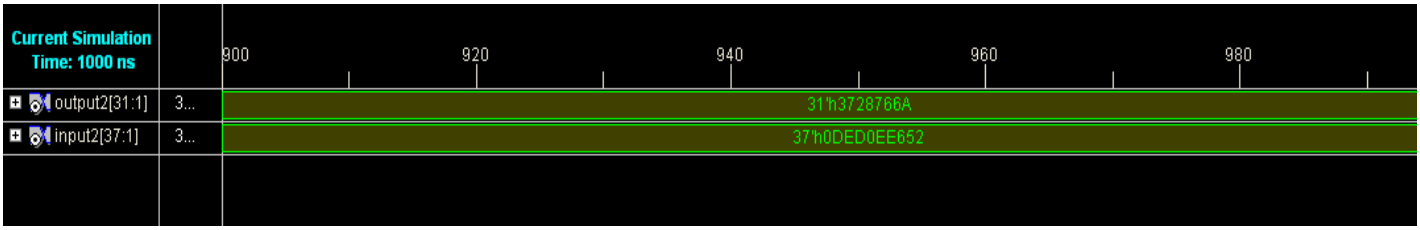
Novel hamming code

We are transmitting the information data of 31 bits 0110111001010000111011001101010(31'h3728766A) at Novel Hamming encoder output the data in encrypted form is **110011**0110111001010000111011001101010 (37'h98728766A) . Now we generate the error at bit position 20 at the output of decoder we are able to collect the actual information data

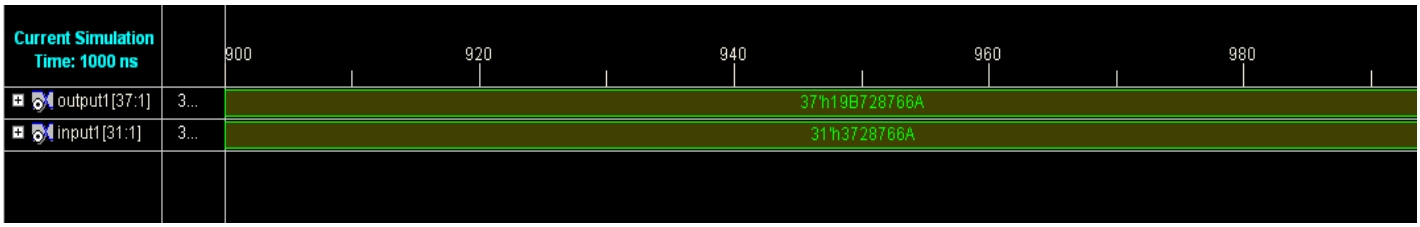
HAMMING ENCODER



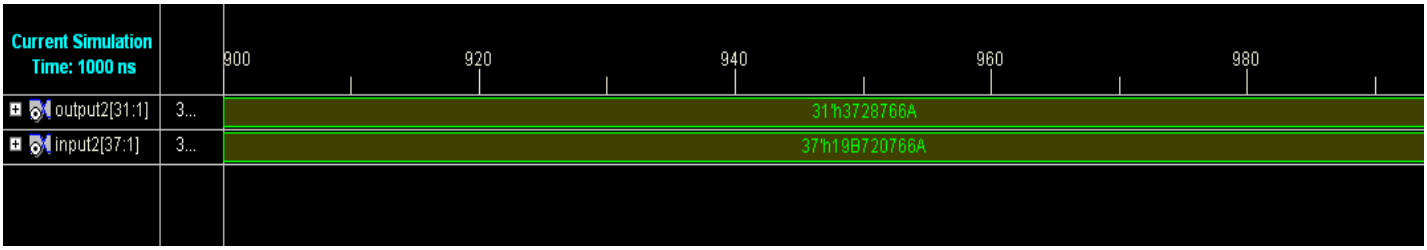
HAMMING DECODER (ERROR AT BIT POSITION 28)



NOVEL HAMMING ENCODER



NOVEL HAMMING DECODER (ERROR AT BIT POSITION 20)



4. APPLICATION

An application of hamming code for error detection and correction with even parity check and odd parity check method is that by this method there is no need to transmit data string again by transmitter at source end, if only single bit error is occurred by noisy channel because at destination end receiver can regenerate the original data string which was transmitted by transmitter at source end. 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 (ADSL, xDSL) and also used to protect storage cell which is effected by SEUs

5. CONCLUSION AND FUTURE SCOPE

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.

6.REFERENCES

[1] Data communication and networking , Behrouz A. Forouzan , 2nd edition Tata McGrawHill publication.

[2] Communication Networks, available at: http://www.pragsoft.com/books/CommNetwork.pdf

[3] Xilinx ISE 8 Software Manuals and Help: http://www.eng.uwaterloo.ca/~tnaqvi/downloads/DOC/sd192/ISE8_1i_manuals.pdf

[4] W. Xiong, and D. W. Matolak, "Performance of Hamming Codes in Systems Employing Different Code Symbol Energies," IEEE Communications Society, pp. 1055–1058 [Wireless and Communications and Networking Conference (WCNC)].

[5] ISE 10.1 Quick Start Tutorial, available at http://www.xilinx.com/itp/xilinx10/books/docs/qst/qst.pdf

[6] VHDL (VHSIC hardware description language): http://en.wikipedia.org/wiki/VHDL

[7] VHDL Tutorial, available at http://www.vhdl-online.de/tutorial

[8] T. Fujiwara et al., "Error Detecting Capabilities of the Shortened Hamming Codes Adopted for Error Detection in IEEE Standard 802.3," IEEE Trans. Communications, vol. 37, no. 9, pp. 986–989, Sep 1989.

[9] W. W. Peterson and E. J. Weldon, Jr., Error-Correcting Codes (2nd ed.). Cambridge, MA: MIT Press, 1972

7. AUTHOPRS PROFILE

1. Brajesh Kumar Gupta : M.Tech Scholar , Department of Electronics and Communication Engineering , National Institute of technology, Jamshedpur (India)
Email : bkgupta8bu@gmail.com

2. Associate Prof. Rashmi Sinha : Department of Electronics and Communication Engineering , National Institute of technology, Jamshedpur (India)
Email : rsinha.ece@nitjsr.ac.in