DESIGN AND VERIFICATION OF IMPROVED HAMMING CODE (ECC) USING VERILOG

[1]RAYMOND IRUDAYARAJ I., [2]ABDUL LATEEF HAROON P.S, [3]ULAGANATHAN J., [4]SHRIDHAR S. BILAGI

[1,2,3,4]Assistant Professors, Dept. Of ECE, BITM/RYMEC-Ballari-583104
E-mail: 1raymond.jhts@gmail.com, 2abdulharoon27@gmail.com, 3ulgan.81@gmail.com, 4shridharbilagi875@gmail.com

Abstract - This paper describes Improved Hamming Code, At whatever point data is stored or transmitted, some chance one or more bits will "flip" i.e., will change to an incorrect value. Such incorrect values are called errors; they may be because of a changeless shortcoming (broken hardware) or a transient condition. To neutralize this issue and guarantee dependable operation, error correcting codes (ECC) are utilized. Additional bits are sent or stored close by the data bits to give redundant data. With enough bits of deliberately picked redundant data, we can detect or correct the most likely classes of errors. Hamming code error correction is most generally utilized for computer memories. Hamming code with additional parity/redundancy bit can detect and correct single-bit errors and detect two bit errors. Hamming code is normally utilized for transmission of data with little lengths. Scaling it for bigger data lengths, results in a ton of overhead because of interspersing the redundancy bits and their evacuation later. Improved hamming code strategy is exceptionally adaptable without such overhead. Accordingly it is suitable for transmission of huge size data bit-streams with much lower overhead bits per data bit ratio. The project's objective is to design an error correction IP core utilizing improved hamming code. Hamming code with extra parity bit can detect and correct single-bit errors and detect two bit errors. The error correction IP core design endeavored in the paper utilizes improved hamming code error correction strategy. This strategy can detect and correct single-bit errors. In traditional hamming code strategy, extensive quantities of overhead bits are utilized as a part of the procedure of computation of parity/redundancy bits. In improved hamming code system the quantity of overhead bits is significantly decreased. The parity bits are annexed toward the end of data bits. This wipes out the overhead of interspersing the redundancy bits at the sender end and their evacuation at the receiver end. This work is accepted to serve as a decent error correction system for transmission of substantial size data bit-streams the length of there is probability of at the most single-bit error amid transmission.

Keywords - Hamming code, Error correction, IHC

I. INTRODUCTION

Hamming code is an error correction code that is used in the process of detecting single and 2-bit errors and corrects the single-bit error that may happen when binary data is propagated along the way from one unit into another [2].

Showing the design and development of (11, 7, 1) Hamming code utilizing Verilog hardware description language (HDL). Here, "11" compares to the aggregate number of Hamming code bits in a transmittable unit containing data bits and excess redundancy bits, 7 is the quantity of data bits while "1" signifies the most extreme number of error bits in the transmittable unit. This code fits well into little “field-programmable gate array (FPGAs), application-specific integrated circuits (ASICs) and complex programmable logic devices (CPLDs)” and is preferably suited into communication applications that need error-control [1].

The Technique

Utilization of basic parity permits in detecting the single-bit error in a received message. Correcting these obtained errors requires some extra data, since the bit location of the erroneous bit must be recognized in the event that it is to be corrected. (On the off chance that an erroneous bit can be found on the location and it can be re-corrected to original by simply changing inversely or flipping its bit value.) Correction is unrealistic with one parity bit subsequent to any bit error in any bit location creates the very same data, i.e., an error. In the event that more number of bits are incorporated in a source data message, and if that given bits can be organized such that diverse erroneous bits produces distinctive error results, then erroneous bits could be recognized.

The forward error correction (FEC) in the digital communication facility systems, especially those utilized as a part of military, need to perform precisely and dependably even in the vicinity of noise and interference. Among numerous conceivable approaches to accomplish this objective, forward error-correction coding is the best and efficient. The forward error correction systemic coding (additionally called 'channel coding') is a kind of digital signal processing so as to handle that enhances dependability of the data a known structure into the data arrangement before transmission. This structure empowers the accepting system that is used to detect and perhaps correct errors brought on by debasement from the provided data channel and the applied receiver at the other end. As this systems name suggests, this employed coding procedure empowers the decoder to correct the found errors by not taking a chance of asking for re-transmission of the original data. Hamming code is a run of the mill illustration of forward error correction. In any event related to the

communication system that utilizes forward error-correction coding, in the first stage the digital data source sends a message data succession to a systematic data encoder unit. The encoder embeds redundant bits (or parity bits), consequently yielding a more drawn out succession of code bits, which is called a 'code word.' These coded bits in the formation of words can be later propagated to a recipient, which utilizes a proper related decoder to extricate the original message data sequence on the go.

II. METHODOLOGY

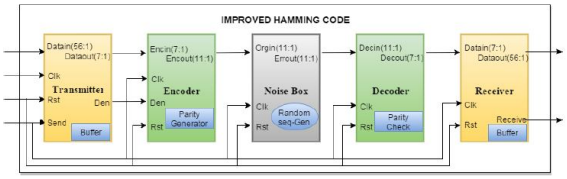

Fig 1: Block diagram of Improved Hamming code system

The above figure shows the detailed block diagram of Improved Hamming code system. It is composed of 5 stages as shown, such as Transmitter, Encoder, Noise Box, Decoder, and Receiver. The major role for the operation is given to the encoder on the source side and to the decoder on the receiver end.

The noise box is introduced only for the demonstration purpose to induce the error assuming it to be in the transmission path. It has a random sequence generation, which in turn determines the position of the bit to be flipped.

In improved Hamming code, the redundancy bits are put toward the data's end bits. This significantly decreases considerable measure of overhead because of interspersing the redundancy bits and their evacuation later at the receiver end. Further the quantity of overhead bits included during the process of calculating redundancy bits is less [1]. Therefore this improved hamming code strategy can be used for the transmission of huge size message data bit-streams. But the overhead in this strategy is much lower on bits per data bit ratio comparatively.

Comparative to the traditional hamming code method, the improved hamming code employs the equal number of redundancy bits in this technique for the some value of n. In any case, now and again, the required number of redundancy bits would be just one increased in its count more than Hamming code [1]. The quantity of redundancy bits, "r" to be attached to n-bit data to acquire single error correction is as per mathematical statement 2.

$$2(r-1) - 1 >= n \quad (2)$$

For instance, if the accessible space for code word is just 16 bits, then data bit ought to be just 10 bit wide and the quantity of redundancy bits will be 6 to get single error correction and two error detection. These six bits are set at areas 15, 14, 13, 12, 11 and 10. The parity bits are ascertained by calculating according to the equations got from reality table given in table

Bit position of data	P[3]	P[2]	P[1]	P[0]
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0

Fig 2: Truth table of erroneous bit position

P[0] is having value "1" at bit positions 1, 3, 5, 7 and 9. Consequently P[0] is chosen such that there is even parity at these locations (XXX1 <= 10) [1]. P[1] is chosen such that there is even parity at locations 2, 3, 6, 7 and 10 (XX1X <= 10). P[2] is chosen such that there is even parity at locations 4, 5, 6 and 7 (X1XX <= 10). P[3] is chosen such that there is even parity at locations 8, 9 and 10 (1XXX <= 10). P[4] is chosen such that there is even parity at the bit positions of redundancy bits.

P[3:0]. P[5] is chosen such that there is even parity in all the bit locations including the redundancy bits P[4:0]. These parity bits are scattered in positions 15, 14, 13, 12, 11 and 10. For the computation of parity bits, even parity checks were per-shaped on 5, 5, 4, 3, 4 & 16 bits respectively. In this way a sum of 37 bits are included during the process of hamming bits computation. The code word format for an example data 10'b1100110011 is appeared in figure 2. Parity bits are appeared in bold format.

0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1
15	**14**	**13**	**12**	**11**	**10**	9	8	7	6	5	4	3	2	1	0

Fig 3: Code word format for sample data

This code word shown is transmitted or stored in the memory. At the receiving end, the parity bits are uprooted. A parity check is performed between the transmitted parity and parity that is in the received code word. The result of examination decides the way of error. On the off chance that single bit error has been happened, then a mask will be produced and the data will be corrected. The error-detection and correction process in hamming code is as outlined in table 4.

Received information including Hamming bits	Status of parity check	Conclusion
0000111100110011	000000	No Error
0000111100110111	100011	Single Error at position 2
0000111000110011	101010	Single Error at position 9

Fig 4: Error detection and correction using improved hamming code

Operational Pipeline

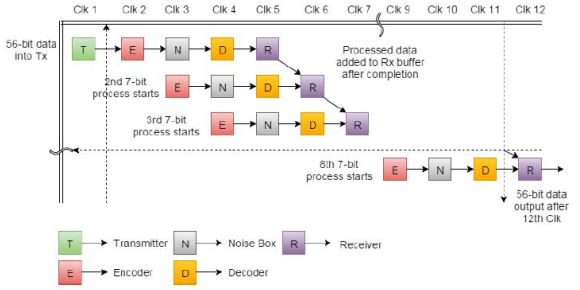

Fig 5: IHC Operational Pipeline

The above figure shows the operational pipeline of the improved hamming code. And it consists of a data acceptance, data encoding using parity generation stage, noise introduction stage, data decoding using parity check stage and data stream packing and output stage.

At the end of the execution sequence, the receiver collects all 8 groups of 7-bits data packet and arranges it back to a 56-bit data stream reforming the original message bits.

Since the system is built with the pipelining structure the overall clock pulse took for the data appear in the output is 12 clocks (4 processing clocks and 8 data clocks).

There is also an indicator for sending the data ready signal at the output. This signal is given the name as receive signal in this project. When the original data stream is reforms and made available in the output, this receive signal goes high.

III. RESULTS AND DISCUSSIONS

Suppose for a 56-bit data (1111000-1111110-1100000-1010000-1001000-1000100 1000010-1000000) transmission, the data is split into 8 packets of 7 bit each in the first stage. One packet is forwarded per cycle onto every next stage. The bit size in a packet is increased in the encoder stage, where 4 extra bits called as parity bits are added up to the packet wrapping the data for redundancy. Hence for every 7-bit, there is an added 4-bit of redundancy. Therefore for the overall 56-bits of data the overhead bits count is 32-bits. So the payload is only 56-bit for a whole of 88 bits and the remaining 32 bits are the redundancy bits. In percentage 63.6 % is a payload. Initially when beginning the data transmission, the send signal is activated indicating the data ready on the input. There are a number of signals at the input end. Like Input data, Clk, Rst and Send signals.

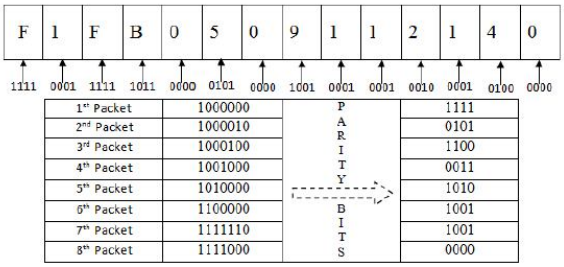

Fig 6: Bit splitting and parity check table

I.e. Payload = No. of Data Bits / Total number of bits transmitted

$$\text{I.e. Payload} = \frac{56 \text{ bits}}{88 \text{ bits}} \text{ X } 100$$

Therefore payload = 63.6 %
And remaining 36.4 % is redundancy bits

56-bits data input with an initial system reset

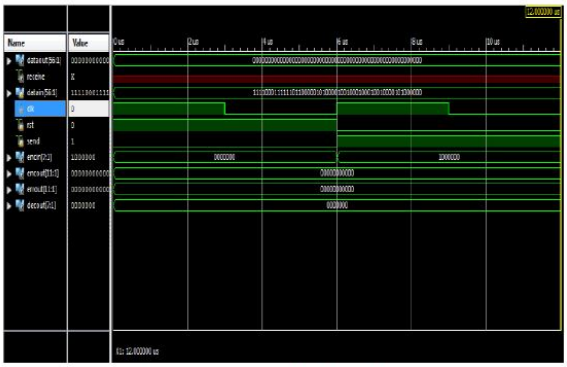

Fig 7: Data input of all 56-bits at once

The above figure shows system reset and feeding of the input bits for the transmission. The 56-bits are split into 8 groups of equal bit size, where each group has 7-bits and then all the split bits are stored in the transmitter buffer. For each clock pulse, one 7-bit data packet is passed on to the next operational stage. This is done till all the 8 groups are passed to the next stage, i.e., the encoder stage.

Suppose for a 56-bit data (1111000-1111110-1100000-1010000-1001000-1000100 1000010-1000000) transmission, the data is split into 8 packets of 7 bit each in the first stage. One packet is forwarded per cycle onto every next stage. The LSB 7 bits "1000000" is already available on the encoder input as seen on the above figure.

First packet on the second stage (Encoder)

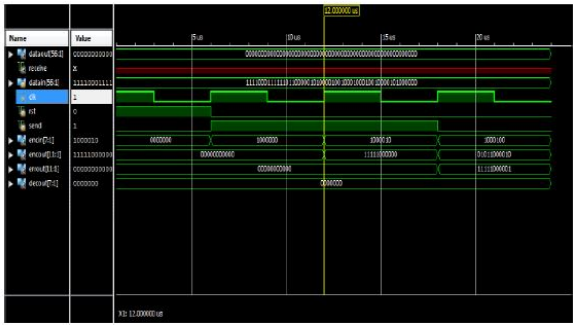

Fig 8: First packet in the encoder input

In the above figure, the first packet "1000000" is processed at the encoder stage. In the encoder stage, the parity bits are calculated for the available packet. Here in this case the parity bits calculated are "1111". Now the 7-bit data is concatenated with the 4-bit parity on the MSB side making the whole packet size of 11-bits which is available on the encoder output. At the same instant of time the next data packet is available at the encoder input.

The encoder stage is a redundancy bits generation stage. When the data of 7-bits are received by the encoder, the 4-bit parity generation is done for the given 7-bits of data. The 4-bit parity and 7-bit data summed up together makes the encoder output to be an 11-bit output. The payload over the 11-bit encoded data is only 63.6%.

Introduction of noise in the transmitted first packet

Fig 9: Introduction of noise in first packet

After the encoding the data packet consisting of 11-bits including the redundancy bits, that is passed on through a noise box for introducing the noise for demonstration purpose. As we see, the encoded packet is "11111000000". For each clock pulse, a random bit position is chosen for introducing the error or rather say flipping the bit. For this instance, the randomly chosen location is bit position 1. Then the erroneous output out of the noise box is "11111000001". This is shown in the errout signal in the above figure.

Error detection and correction on decoder stage

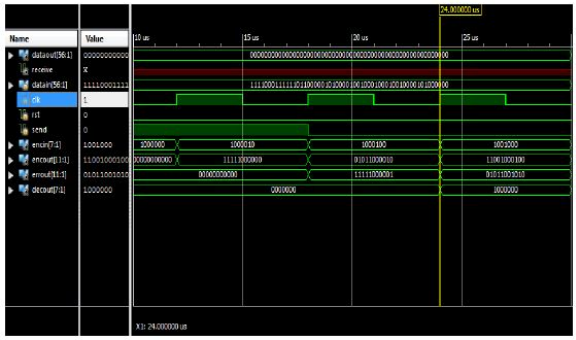

Fig 10: Error detection and correction using parity check in decoder stage

Redundancy check and correction of the erroneous bit happens in this stage. So, the received bit on the decoder input is "11111000001". The decoder calculates the parity check and finds the error position and then flips the bit in the found location correcting it. With respect to the received data, the error location calculated is "001". Hence the bit in the position 1 is flipped and the redundancy bits are removed from the packet making the packet size back to 7-bits.
The obtained data is not immediately made available at the output. But it is stored at the receiver buffer till all the 56-bits are received.

Data reception on the receiver end

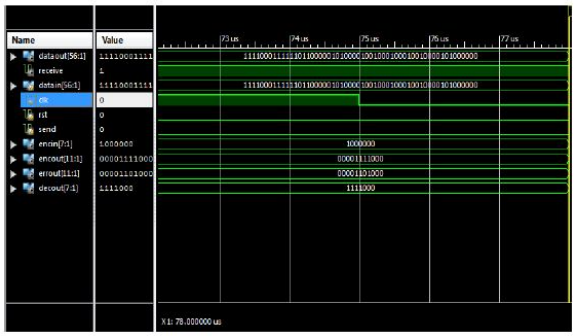

Fig 11: Final data reception and receive signal enabling

For 56-bit data size, exactly after the 12 clock intervals the all the 8 packet are received and are processed at the decoder and they are stored in the receiver buffer.

Once when all the data packets are done, the all the 8 7-bit data packets are put together forming a 56-bit data likely to the input message data and that is projected on the output as shown in the above figure.

And there is an indication signal called receive signal for intimating the user that the data is available in the output line.

CONCLUSIONS

In improved hamming code technique the quantity of overhead bits is incredibly diminished. The parity bits are attached toward the other side of data bits stream. This strategy will wipe out the overhead at the sender end to intersperse the redundancy bits and the redundancy bits elimination at the receiver end. This work is accepted to serve as a decent error correction system for transmission of substantial and variable size data bit-streams, the probability of the length is at the most corrects only single-bit error amid transmission. Further the exertion required in distinguishing the redundancy's values bits is lower in the proposed novel technique.

In the traditional Hamming code, if the data lengths are to be scaled for larger data lengths, that results in a considerable measure of overhead because of interspersing the redundancy bits and their evacuation later. In contrast, the improved hamming code proposed strategy [1] can be scaled highly without any of such overhead. Therefore this improved hamming code strategy can be used for the transmission of huge size message data bit-streams. But the overhead in this strategy is much lower on bits per data bit ratio comparatively.

This work is accepted to serve as a decent error correction system for transmission of substantial and variable size data bit-streams, the probability of the length is at the most corrects only single-bit error amid transmission.

FUTURE SCOPE

Since the proposed system is the improvement for the traditional hamming code method, the only drawback which can be improved is the single bit error correction. This proposed system can be further improvised to detect and correct more than one bit error.

ADVANTAGES AND APPLICATIONS

- In Improved hamming code, in the senders end or transmitter the overhead of redundancy bit interspersing and their removal at the receiver end is eliminated.
- Process payload for obtaining the encoded data is reduced.
- Can be used on any domain by just feeding the 56-bit data splitting
- Easy to use the IP with only inputting the data. That is the only job to be taken care of user. Since it has interface lines.
- The scalability has improved up to good large number of data size.
- Used in communication systems
  a) Tele-text systems
  b) Satellite communication system
  c) Broadcasting (radio and digital TV)
  d) Telecommunications
- Used in information systems
  a) Semiconductor memories
  b) Magnetic disks
  c) Optic reading disks
- Used in audio and video systems
  a) Video switching and splitter units
  b) Audio mixer jumper box units.

REFERENCES

[1] B. Umashankar. "Improved Hamming Code for Error Detection and Correction", 2007 2nd International Symposium on Wireless Pervasive Computing, 02/2007

[2] Kythe. "Extended Hamming Codes", Algebraic and Stochastic Coding Theory, 2012.

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

[4] 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.

[5] 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)].

[6] Wyner, "Recent results in the Shannon theory", IEEE Trans. Inf. Theory, vol. 20, pp. 2-10, 1974

[7] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction", Proc. DCC, pp. 158-167

[8] V. Stankovic, A. D. Liveris, Z. Xiong and C. N. Georghiades, "On code design for the Slepian-Wolf problem and lossless multiterminal networks", IEEE Trans. Inf. Theory, vol. 52, no. 4, pp. 1495-1507, 2006

[9] S. Pradhan and K. Ramchandran, "Generalized coset codes for distributed binning", IEEE Trans. Inf. Theory, vol. 51, no. 10, pp. 3457-3474, 2005.

★★★