A VLSI IMPLEMENTATION OF HAMMING CODE ALGORITHM USING FPGA ARCHITECTURE

Vijayakumara Y M[#1], Byregowda B K[#2], Pradeep Kumar S[#2], Raghav S[#2], Nataraja M[#2], Dr. S N Sheshappa[#2]

[#1] *Assistant Professor, Department of Information Science and Engineering, Sir MVIT, VTU University*
[#2]*Assistant Professor, Department of Information Science and Engineering, Sir MVIT, VTU University*
[#2]*Assistant Professor, Department of Electronics and Telecommunication, Sir MVIT, VTU University*
[#2]*Associate Professor, Department of Information Science and Engineering, Sir MVIT, VTU University*
[#2]*Assistant Professor, Department of Mechanical Engineering, Sir MVIT, VTU University*
[#2]*Associate Professor, Department of Information Science and Engineering, Sir MVIT, VTU University*
Department of ISE, Sir MVIT, Bangalore – 562157, Karnataka, India
Department of ISE, Sir MVIT, Bangalore – 562157, Karnataka, India
Department of ETE, Sir MVIT, Bangalore – 562157, Karnataka, India
Department of ISE, Sir MVIT, Bangalore – 562157, Karnataka, India
Department of ME, Sir MVIT, Bangalore – 562157, Karnataka, India
Department of ISE, Sir MVIT, Bangalore – 562157, Karnataka, India

ABSTRACT

The communication theory has the issue of error correction, and detection has great practical importance. Error correction codes permit the detection and correction of errors that result from noise or other impairments during transmission from the transmitter to the receiver. Error correction schemes permit error localization and also give the possibility of correcting them. Error correction and detection schemes find use in implementations of reliable data transfer over noisy transmission links, data storage media (including dynamic RAM, compact discs), and other applications where data integrity is important. Error correction avoids retransmission of the data, which can degrade system performance.

Hamming code is an error-correction code that can be used to detect single and double-bit errors and correct single-bit errors that can occur when binary data is transmitted from one device into another. Hamming codes provide for FEC using a "block parity" mechanism that can be inexpensively implemented. In general, their use allows the correction of single-bit errors and detection of two-bit errors per unit data, called a code word.

Keywords: *communication, memory, code correction, Hamming code*

I. INTRODUCTION

A. Error correction codes

In computer science and information theory, the issue of error correction and detection has great practical importance. Error correction codes (ECCs) permit detection and correction of errors that result from noise or other impairments during transmission from the transmitter to the receiver. Given some data, ECC methods enable you to check whether data has been corrupted, providing the difference between a functional and nonfunctional system. Error correction schemes permit error localization and also give the possibility of correcting them. Error correction and detection schemes find use in implementations of reliable data transfer over noisy transmission links, data storage media (including dynamic RAM, compact discs), and other applications where the integrity of data is important. Error correction avoids retransmission of the data, which can degrade system performance [1] [2] [3].

a) RAM Devices

RAM devices do not, as such, support error control codes. There are no mandatory requirements for ECC support on RAM/DRAM devices. Memory suppliers are generally not in favor of implementing a complex logic function like ECC onto a RAM die [4] [5]. It is costly, inefficient, and leads to an expensive memory subsystem. Where enhanced reliability is a requirement, the standard technique is to use a more comprehensive interface. In the context of SDRAMs, DIMMs come in two widths: 64 and 72 bits. The 72-bit DIMMs are targeted for use with ECCs, because of the extra 8 bits. The extra 8 bits are merely extra data bits. In reality, you can use any of the bits. An extra 8 bits of parity on 64 bits of data allows you to employ a two-bit error detection. Single bit correcting Hamming code.

b) Hamming code

Hamming code is an error-correction code that can be used to detect single and double-bit errors and correct single-bit errors that can occur when binary data is transmitted from one device into another. Hamming codes provide for FEC using a"block parity" mechanism that can be inexpensively implemented. In general, their use allows the correction of single-bit errors and detection of two-bit errors per unit data, called a code word.

The fundamental principle embraced by Hamming codes is parity. Hamming codes, as mentioned before, are capable of correcting one error or detecting two errors but not capable of doing both simultaneously. You may choose to use Hamming codes as an error detection mechanism to catch both single and double-bit errors or correct a single error. This is accomplished by using more than one parity bit, each computed on a different combination of bits in the data [6] [7] [8].

This project design and development of (11, 7, 1) Hamming code using the VLSI coding method. Here, '11' corresponds to the total number of Hamming code bits in a transmittable unit comprising data bits and redundancy bits, 7 is the number of data bits, while '1' denotes the maximum number of error bits in the transmittable unit. This code fits nicely into small field-programmable gate arrays (FPGAs), complex programmable logic devices (CPLDs), and application-specific integrated circuits (ASICs). It is ideally suited to communication applications that need error-control [9][10][11].

B. Error correction

The use of simple parity allows the detection of single-bit errors in a received message. Correcting these errors requires more information since the position of the corrupted bit must be identified if it is to be corrected. (If a corrupted bit can be detected, it can be corrected by simply complementing its value.) Correction is not possible with one parity bit since any bit error in any position produces precisely the same information, i.e., error. If more bits are included in a message, and if those bits can be arranged such that different corrupted bits produce different error results, then corrupted bits could be identified. [12].

a) Forward error correction (FEC)

Digital communication systems, particularly those used in the military, need to perform accurately and reliably even in noise and interference. Among many possible ways to achieve this goal, forward error correction coding is the most effective and economical. Forward error correction coding (also called 'channel coding') is a type of digital signal processing that improves the data's reliability by introducing a known structure into the data sequence prior to transmission.

This structure enables the receiving system to detect and possibly correct errors caused by corruption from the channel and the receiver. As the name implies, this coding technique enables the decoder to correct the mistakes without requesting the original information. Hamming code is a typical example of forwarding error correction.

In a communication system that employs forward error-correction coding, the digital information source sends a data sequence to an encoder. The encoder inserts redundant (or parity) bits, thereby outputting a more extended sequence of code bits, called a ‘code word.' These code words can then be transmitted to a receiver, which uses a suitable decoder to extract the original data sequence.

II. IMPLEMENTATION

In Hamming's code, ***p*** is interpreted as an integer, which is 0 if no error occurred, and otherwise is the 1-origined index of the bit that is in error. Let k be the number of information bits, and m the number of check bits used. Because the m check bits must check themselves as well as the information bits, the value of ***p***, interpreted as an integer, must range from 0 to which is distinct values. Because m bits can distinguish cases, we must have

$$2m \geq m+k+1$$

Where

k = Number of “information” or “message” bits.

m = Number of parity-check bits (“check bits,” for short).

n = Code length, n = m + k.

u = Information bit vector, ***u***0, ***u***1, … ***u***k–1

p = Parity check bit vector, ***p***0, ***p***1, …, ***p***m–1.

s = Syndrome vector, ***s***0, ***s***1, …, ***s***m–1.

This is known as the Hamming rule. It applies to any single error correcting (SEC) binary FEC block code in which all of the transmitted bits must be checked.

A. PARITY BITS

A parity bit is the extra bit included to make the total number of 1’s in the resulting codeword either even or odd. For a 7-bit number, there are 7 possible one-bit errors. 64 different binary permutations can be recognized in a string of length 6 bits. However, another state is needed to represent the case when a detectable error has not occurred. The number of parity bits, *m*, needed to detect and correct a single-bit error in a data string of length *n* is given by the following equation:

$$m = \log 2n + 1$$

The ECC block uses the Hamming code with an additional parity bit, which can detect single and double-bit errors, and correct single-bit errors. The extra parity bit applies to all bits after the Hamming code check bits have been added. This extra parity bit represents the parity of the codeword. If one error occurs, the parity changes. If two errors occur, the parity stays the same. In general, the number of parity bits, *m*, needed to detect a double-bit error or detect and correct a single-bit error in a data string of length *n*, is given by the following equation:

$$m = \log 2n + 2$$

B. METHODOLOGY OF OPERATION OF A SIMPLE (7, 4, 1) HAMMING CODE

The purpose of Hamming codes is to create a set of parity bits that overlap such that a single-bit error (the bit is logically flipped in value) in a data bit or a parity bit can be detected and corrected. While multiple overlaps can be created, the general method is presented and shown in Table 2.2.1

Table 2.2.1 Parity Bits cover

Bit #	1	2	3	4	5	6	7
Transmitted bit	p_1	p_2	d_1	p_3	d_2	d_3	d_4
p_1	Yes	No	Yes	No	Yes	No	Yes
p_2	No	Yes	Yes	No	No	Yes	Yes
p_3	No	No	No	Yes	Yes	Yes	Yes

Table 2.2.1 describes the parity bits cover, which transmitted bits in the encoded-word. For example, p2 covers bits 2, 3, 6, & 7. It also details which transmitted by which parity bit by reading the column. For example, d1 is covered by p1 and p2 but not p3.

C. HAMMING MATRICES

Hamming codes can be computed in linear algebra terms through matrices because Hamming codes are linear codes. For Hamming codes, two Hamming matrices can be defined as the code generator matrix G and the parity-check matrix H and shown as

$$\mathbf{G} := \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{H} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

As mentioned above, rows 1, 2, & 3 of G should look familiar as they map the data bits to their parity bits.

1. $p1$ covers $d1$, $d2$, $d4$
2. $p2$ covers $d1$, $d3$, $d4$
3. $p3$ covers $d2$, $d3$, $d4$

The remaining rows (4, 5, 6, 7) map the data to their position in encoded form, and there is only 1 in that row, so it is an identical copy. These four rows are linearly independent and form the identity matrix (by design, not coincidence). Also, as mentioned above, the three rows of H should be familiar. These rows are used to compute the syndrome vector at the receiving end, and if the syndrome vector is the null vector (all zeros), then the received word is error-free; if non-zero, then the value indicates which bit has been flipped.

The 4 data bits assembled as a vector P and is pre-multiplied by G (i.e., Gp) and taken modulo 2 to yield the encoded value that is transmitted. The original 4 data bits are converted to 7 bits [hence the name "Hamming(7,4)"] with 3 parity bits added to ensure even parity using the above data bit coverage.

D. CHANNEL CODING

Suppose we have to transmit this data over a noisy communications channel. Specifically, a binary symmetric channel meaning that error corruption does not favor either zero or one (it is symmetric in causing errors). Furthermore, all source vectors are assumed to be equiprobable. We take the product of G and p, with entries modulo 2, to determine the transmitted codeword x:

$$\mathbf{x} = \mathbf{Gp} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

E. PARITY CHECK

Any error occurs during transmission, and then the received codeword r is identical to the transmitted codeword x:

$$r = x$$

The receiver multiplies H and r to obtain the syndrome vector Z, which indicates whether an error has occurred and for which codeword bit. Performing this multiplication (again, entries modulo 2) :

$$\mathbf{z} = \mathbf{Hr} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Since the syndrome z is the null vector, the receiver can conclude that no error has occurred. This conclusion is based on the observation that when the data vector is multiplied by, a change of basis occurs into a vector subspace that is the kernel of. As long as nothing happens during transmission, it will remain in the kernel, and the multiplication will yield the null vector.

F. ERROR CORRECTION

Otherwise, suppose a single bit error has occurred. Mathematically, we can write.

$$\mathbf{r} = \mathbf{x} + \mathbf{e}_i$$

modulo 2, where e_i is the i^{th} unit vector, that is, a zero vector with a 1 in the i^{th}, counting from 1.

$$c_2 = \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\end{pmatrix}$$

Thus the above expression signifies a single bit error in the i^{th} place.

Now, if we multiply this vector by H:

$$\mathbf{Hr} = \mathbf{H}(\mathbf{x} + \mathbf{e}_i) = \mathbf{Hx} + \mathbf{He}_i$$

Since x is the transmitted data, it is without error, and as a result, the product of H and x is zero. Thus

$$\mathbf{Hx} + \mathbf{He}_i = \mathbf{0} + \mathbf{He}_i = \mathbf{He}_i$$

Now, H's product with the i^{th} standard basis vector picks out that column of H, we know the error occurs in the place where this column of H occurs. For example, suppose we have introduced a bit error on bit #5

$$\mathbf{r} = (\mathbf{x} + \mathbf{e}_5) = \left(\begin{pmatrix}0\\1\\1\\0\\0\\1\\1\end{pmatrix} + \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\end{pmatrix}\right) = \begin{pmatrix}0\\1\\1\\0\\1\\1\\1\end{pmatrix} = \begin{pmatrix}0\\1\\1\\0\\1\\1\\1\end{pmatrix}$$

Now,

$$\mathbf{z} = \mathbf{Hr} = \begin{pmatrix}1&0&1&0&1&0&1\\0&1&1&0&0&1&1\\0&0&0&1&1&1&1\end{pmatrix}\begin{pmatrix}0\\1\\1\\0\\1\\1\\1\end{pmatrix} = \begin{pmatrix}3\\4\\3\end{pmatrix} = \begin{pmatrix}1\\0\\1\end{pmatrix}$$

which corresponds to the fifth column of H. Furthermore, since the general algorithm used was the intention in its construction. The syndrome of 101 corresponds to the binary value of 5, indicating the fifth bit was corrupted. Thus, an error has been detected in bit 5 and corrected (simply flip or negate its value):

$$\mathbf{r}_{corrected} = \begin{pmatrix}0\\1\\1\\0\\1\\1\\1\end{pmatrix} = \begin{pmatrix}0\\1\\1\\0\\0\\1\\1\end{pmatrix}$$

This corrected received value indeed now matches the transmitted value X from above.

G. DECODING

Once the received vector has been determined to be error-free or corrected if an error occurred (assuming only zero or one-bit errors are possible), the received data needs to be decoded back into the original 4 bits. First, define a matrix R :

$$\mathbf{R} = \begin{pmatrix}0&0&1&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{pmatrix}$$

Then the received value, pr is

$$\mathbf{p_r} = \mathbf{Rr}$$

and using the running example from above

$$\mathbf{p_r} = \begin{pmatrix}0&0&1&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{pmatrix}\begin{pmatrix}0\\1\\1\\0\\0\\1\\1\end{pmatrix} = \begin{pmatrix}1\\0\\1\\1\end{pmatrix}$$

Which is the same as the transmitted 4 Bit data? Consider another message having four data bits (D), which is transmitted as a 7-bit codeword by adding three error control bits. This would be called a (7,4) code. The three bits to be added are three EVEN Parity bits (P), where the parity of each is computed on different subsets of the message bits, as shown below.

7	6	5	4	3	2	1	
D	D	D	P	D	P	P	7-BIT CODEWORD
D	-	D	-	D	-	P	(EVEN PARITY)
D	D	-	-	D	P	-	(EVEN PARITY)
D	D	D	P	-	-	-	(EVEN PARITY)

H. BITS

The three parity bits (1,2,4) are related to the data bits (3,5,6,7), as shown. In this diagram, each overlapping circle corresponds to one parity bit and defines the four bits contributing to that parity computation. For example, data bit 3 contributes to parity bits 1 and 2. Each circle (parity bit) encompasses four bits, and each circle must have EVEN parity. Given four data bits, the three parity bits can easily be chosen to ensure this condition.

It can be observed that changing any one bit numbered 1..7 uniquely affects the three parity bits. Changing bit 7 affects all three parity bits, while an error in bit 6 affects only parity bits 2 and 4, and an error in a parity bit affects only that bit. The location of any single bit error is determined directly upon checking the three parity circles.

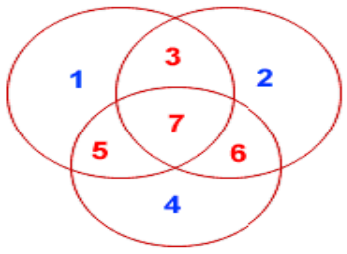

Figure 1. Parity Circle

I. Hamming Distance

The Hamming Code allows error correction because the minimum distance between any two valid codewords is 3. In the figure below, two valid codewords in 8 possible 3-bit codewords are arranged

to distance 3 between them. It takes 3-bit changes (errors) to move from one valid codeword 000 to the other 111. Suppose the codeword 000 is transmitted, and a single bit error occurs. In that case, the received word must be one of {001,010,100}, any of which is easily identified as an invalid codeword, and which could only have been 000 before transmission.

J. The Distance Argument

Looking again at the Venn diagram (above), it can be observed that a change in any of the data bits (3,5,6,7) necessary changes at least two other bits in the codeword. For example, given a valid Hamming codeword, a change in bit 3 changes three bits (1,2,3) such that the new codeword is a distance (d=3) from the initial word. The Hamming code words' clever arrangement ensures that this is the case for every valid codeword in the set.

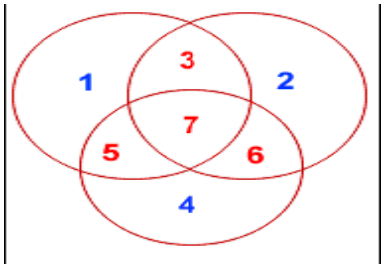

Figure 2: Venn Diagram

III. BLOCK DIAGRAM

Encoding is performed by multiplying the original message vector by the generator matrix; decoding is performed by multiplying the codeword vector by the parity check matrix H. All additions are performed modulo 2. In hardware, this process equates to XORing a particular set of data elements and is computationally inexpensive. Suppose an error occurs, and one of the parity or data bits change during transmission. In that case, the ECC decoder-corrector gives the bit syndrome of the data bit affected by recalculating the parity bits and XORing them with the transmitted parity bits (computing the syndrome). Then the decoder-corrector allows the correction of a single-bit error. ECC detects errors through the process of data encoding and decoding. For example, when ECC is applied in a transmission application, data read from the source are encoded before being sent to the receiver. The output (code word) from the encoder consists of the raw data appended with the number of parity bits. The exact number of parity bits appended depends on the number of bits in the input data. The generated code word is then transmitted to the destination.

The receiver receives the code word and decodes it. Information obtained by the decoder determines whether or not an error is detected. The decoder detects single-bit and double-bit errors, but can fix only single-bit errors in the corrupted data. This kind of ECC is called Single Error Correction Double Error Detection (SECDED).

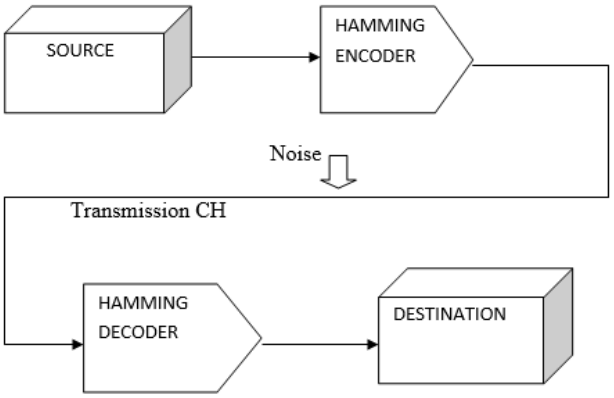

Figure 3: Block Diagram of Hamming encoder and decoder

This Hamming code design provides two modules, HAMMING ENCODER and HAMMING DECODER, to implement the ECC functionality. The data input to the HAMMING ENCODER module is encoded to generate a code word that is a combination of the data input and the generated parity bits. The generated code word is transmitted to the HAMMING DECODER module for decoding just before reaching its destination block.

The HAMMING DECODER module generates a syndrome vector to determine if there is any error in the received code word. It fixes the data if and only if the single-bit error is from the data bits.

A. DESCRIPTION OF THE HAMMING ENCODER

The HAMMING_ENCODER Module takes in and encodes the data using the Hamming Coding scheme. The Hamming Coding scheme derives the parity bits and appends them to the original data to produce the output code word. The number of parity bits appended depends on the width of the data. Table 2.2 shows the number of parity bits appended for different ranges of data widths. The Total Bits column represents the total number of input data bits and appended parity bits.

B. DESCRIPTION OF THE HAMMING DECODER

The HAMMING_DECODER Module decodes the input data (codeword) by extracting the parity bits and data bits from the code word. The parity bits and data bits are recalculated based on the Hamming Coding scheme to generate a syndrome code. The generated syndrome code provides the status of the data received. The ECC detects single-bit and double-bit errors, but only single-bit errors are corrected.

The incoming 7-bit data along with the 4-bit parity are XOR'd together to generate the 4-bit syndrome (S1 through S4). In order to correct a single bit error, a 7-bit correction mask is created. Each bit of this mask is generated based on the result of the

syndrome from previous stage. When no error is detected, all bits of the mask become zero. When a single bit error is detected, the corresponding mask masks out the rest of the bits except for the error bit. The subsequent stage then XORs the mask with the original data. As a result, the error bit is reversed (or corrected) to the correct state. If a double bit error is detected, all mask bits become zero. The error type and corresponding correction mask are created during the same clock cycle.

In the data correction stage, the mask is XOR'd together with the original incoming data to flip the error bit to the correct state, if needed. When there are no bit errors or double bit errors, all the mask bits are zeros. As a result, the incoming data goes through the ECC unit without changing the original data.

IV. RESULTS

The Coding is implanted and results are verified using Verilog HDL coding Technique for the Hamming code encoder and decoder. Encoder (11,7,1) converts a 7-bit ASCII code into an 11-bit code word while the Decode is the (11, 7, 1) Hamming code decoder that converts an 11-bit code word back into a 7-bit ASCII code after correcting the single bit error, if any. Both these programs have been developed in Verilog HDL and simulated using Vivado software and code is downloaded to FPGA Spartan 3E for further implementation.

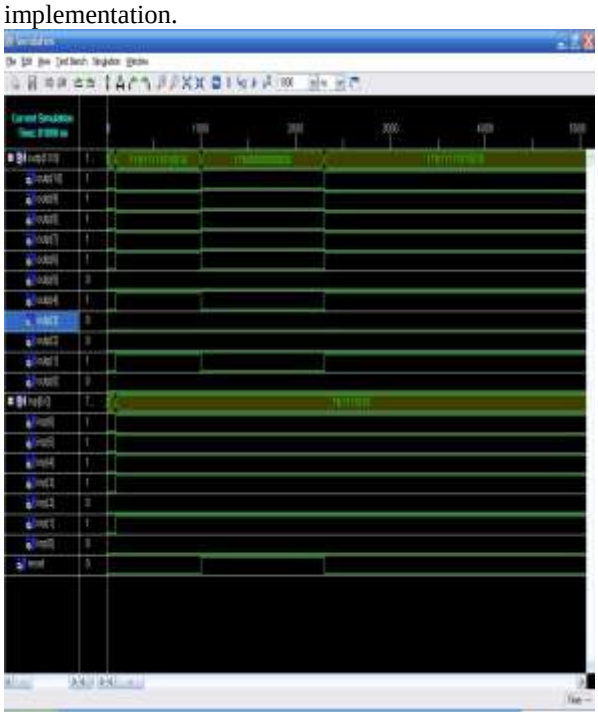

Figure 4: Simulation Result of Encoder

The figure 4 shows the simulation result of Hamming encoder for the original 8 bit input word. No error while transmitting the original code bits.

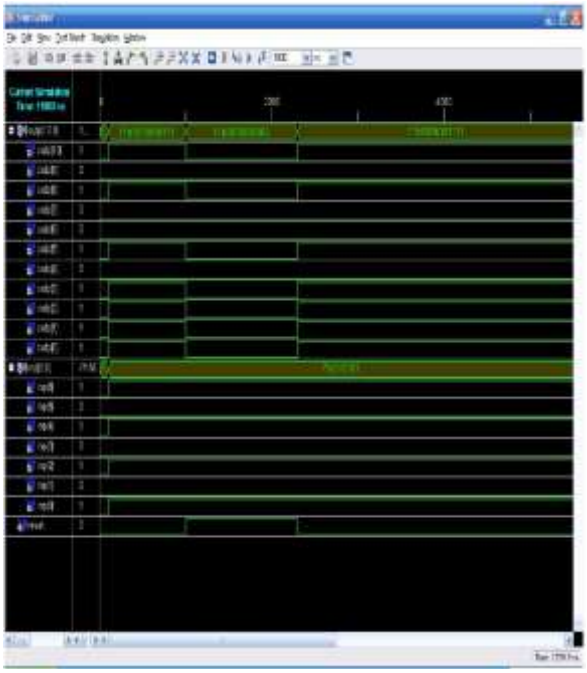

Figure 5: Simulation Result of Decoder

The figure 5 shows the simulation result of Hamming Decoder for the Received data and no error is occurred while receiving the data.

Similar way the transmission can be done using different bits with errors and without errors and further the code is downloaded for FPGA Spartan 3E for Validation and efficient transmission purpose.

V. CONCLUSION

Error Correction Code (ECC) is a method of error detection and correction in digital data transmission. This project presented design and development of (11, 7, 1) Hamming code using Verilog hardware description language (HDL). Here, '11' corresponded to the total number of Hamming code bits in a transmittable unit comprising data bits and redundancy bits, 7 was the number of data bits while '1' denoted the maximum number of error bits in the transmittable unit.

In a communication system that employs forward error-correction coding, the digital information source sends a data sequence to an encoder. The encoder inserts redundant (or parity) bits, thereby outputting a longer sequence of code bits, called a code word. These code words can then be transmitted to a receiver, which uses a suitable decoder to extract the original data sequence.

The Verilog HDL code is implemented well into small field-programmable gate arrays (FPGAs), complex programmable logic devices (CPLDs) and application specific integrated circuits (ASICs) and therefore is ideally suited to communication applications that need error-control.

Acknowledgments

We would like to thank Mr Pradeep kumar S, Dr Seshappa and Mr Bhyregowda B K for continuous support and financial assistance for developing a project.

VI. REFERENCES

[1] E. Khan , S. Lehmann ; H. Gunji , M. Ghanbari, "*Iterative error detection and correction of coded video for wireless networks"* IEEE Transactions on Circuits and Systems for Video Technology Year: 2004, Volume: 14, Issue: 12 .

[2] P. Perry ; Mingche Li ; Mao-Chao Lin ; Zhen Zhang, *"Runlength limited codes for single error-detection and single error-correction with mixed type errors",* IEEE Transactions on Information Theory Year: 1998, Volume: 44, Issue: 4.

[3] Ashwini kumari P, , Byregowda B K, Vijayakumara Y M, Ravikumar H R , Dr S N Sheshappa, Pradeep kumar S*"A Hardware Implementation Of Hazardous Gases Detection Using Robot"* International Journal of Engineering Trends and Technology 67.7 (2019): 24-30.

[4] Liwen Liu, YiqiZhuang, LiZhang, HualianTang, SiwanDong, *"Proactive correction coset decoding scheme based on SEC-DED code for multibit asymmetric errors in STT-MRAM",* Elsevier, Microelectronics Journal Volume 82, December 2018, Pages 92-100

[5] C. W. Slayman, *"Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations," in* IEEE Transactions on Device and Materials Reliability, vol. 5, no. 3, pp. 397-404, Sept. 2005, doi: 10.1109/TDMR.2005.856487.

[6] S. S. Sarnin, N. F. Nairn and W. N. S. W. Muhamad, *"Performance evaluation of phase shift keying modulation technique using BCH code, Cyclic code and Hamming code through AWGN channel model in communication system,"* The 3rd International Conference on Information Sciences and Interaction Sciences, Chengdu, 2010, pp. 60-65, doi: 10.1109/ICICIS.2010.5534715.

[7] J. Metzner, "*Correction of Two (or Often More) Vector Symbol Errors With the Outer Structure of a Hamming Single Error Correcting Code*," *in* IEEE Communications Letters, vol. 18, no. 12, pp. 2069-2072, Dec. 2014, doi: 10.1109/LCOMM.2014.2363665.

[8] S. G and R. N, "*VLSI design of Parity check Code with Hamming Code for Error Detection and Correction*," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 15-20, doi: 10.1109/ICCS45141.2019.9065790.

[9] Anlei Wang and N. Kaabouch, "*FPGA based design of a novel enhanced error detection and correction technique*," 2008 IEEE International Conference on Electro/Information Technology, *Ames*, IA, 2008, pp. 25-29, doi: 10.1109/EIT.2008.4554262.

[10] J. Singh and J. Singh, "A Comparative Study of Error Detection and Correction Coding Techniques," *2012 Second International Conference on Advanced Computing & Communication Technologies*, Rohtak, Haryana, 2012, pp. 187-189, doi: 10.1109/ACCT.2012.2.

[11] T. Anwar, P. K. Lala and J. P. Parkerson, "A novel FPGA Architecture with Built-in Error Correction," 2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007, Warsaw, 2007, pp. 1-4, doi: 10.1109/IMTC.2007.379193.

[12] S. Muppalla and K. R. Vaddempudi, "*A novel VHDL implementation of UART with single error correction and double error detection capability*," 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, 2015, pp. 152-156, doi: 10.1109/SPACES.2015.7058236.

[13] R.Divyasharon and Dr.D.Sridharan, "I*mplementation of low power wireless sensor node with fault tolerance mechanism*" SSRG International Journal of Electronics and Communication Engineering 3.4 (2016): 1-5.