RESEARCH ARTICLE OPEN ACCESS

VLSI Implementation of Encryption and Decryption System Using Hamming Code Algorithm

Fazal Noorbasha*, Harikishore Kakarla, Sri Ramya R, G Venkata Maruthi Manoj, Siva Prakash U and Varalakshmi G
VLSI Systems Research Group, Department of ECE, KL University, Vaddeswaram, Guntur (District), AP, India PIN 522 502

ABSTRACT
In this paper, we propose an optimized VLSI implementation of encryption and decryption system using hamming code algorithm. In the present field of communication has got many applications, and in every field the data is encoded at the transmitter and transfer on a communication channel and receive at the receiver after data is decoded. During the broadcast of data it might get degraded because of some noise on the channel. So it is crucial for the receiver to have some function which can recognize and correct the error in the received data. Hamming code is one of such forward error correcting code which has got many applications. In this paper the algorithm for hamming code is discussed and then implementation of it in verilog is done to get the results. Hamming code is an upgrading over parity check method. Here a code is implemented in verilog in which 4-bit of information data is transmitted with 3-redundancy bits. In order to do that the proposed method uses a Field Programmable Gate Array (FPGA). It is known that FPGA provides quick implementation and fast hardware verification. It gives facilities of reconfiguring the design construct unlimited number of times. The HDL code is written in verilog, Gate Level Circuit and Layout is implemented in CMOS technology.
Keywords **-** Hamming Code, FPGA, CMOS, Encryption, Decryption

I. INTRODUCTION

When digital data is transmitted or stored in nonvolatile memory, it is crucial to have a mechanism that can detect and correct a certain number of errors. Error correction code (ECC) encodes data in such a way that a decoder can identify and correct errors in the data. Typically, data strings are encoded by adding a number of redundant bits to them. When the original data is reconstructed, a decoder examines the encoded message to check for any errors [1].
There are two basic types of ECC:

A. Block codes

These codes are referred to as "n" and "k" codes. A block of k data bits is encoded to become a block of n bits called a code word. In block codes, code words do not have any dependency on previously encoded messages. NAND Flash memory devices typically use block codes.

B. Convolution codes

These codes produce code words that depend on both the data message and a given number of previously encoded messages. The encoder changes state with every message processed. Typically, the length of the code word is constant.

II. HAMMING CODE DATA ENCRYPTION AND DECRYPTION SYSTEM

Hamming codes are the most widely used linear block codes. Typically, a Hamming code is defined as (2n - 1, 2n - n - 1), where:

- n is equal to the number of overhead bits.
- 2n - 1 is equal to the block size.
- 2n - n - 1 is equal to the number of data bits in the block.

All Hamming codes can detect three errors and one correct one. Common hamming code sizes are (7, 4), (15, 11), and (31, 26). All have the same hamming distance. The hamming distance and the hamming weight are useful in encoding. When the hamming distance is known, the capability of a code to detect and correct errors can be determined [2].

In this paper to implement the hamming code data encryption and decryption system, we have used a Verilog HDL code for FPGA. Also we have designed the gate level circuit and implemented CMOS layout in three different nanometer technologies. We explained all the steps involved in this system implementation with results. Fig. 1 shows the block diagram of hamming code data encryption and decryption system.

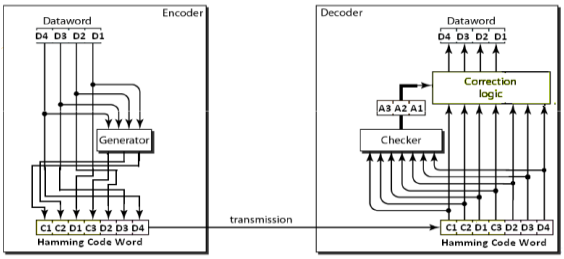

Fig.1 Block Diagram of Hamming Code Data Encryption and Decryption System

III. IMPLEMENTATION

The Hamming code can be used for data words of any length. In general, for *k* check bits and *n* data bits, the total number of bits, $n + k$, that can be in a coded word is at most 2^k -1. In other words, the relationship $n + k \leq 2^k - 1$must hold. This relationship gives $n \leq 2^k - 1 - k$ as the number of bits for the data word [3].

In the (7, 4) extended Hamming code, the equations can be pre-computed as:

Data In Word Bits 4-bit (DI):

$$DI = D_1\ D_2\ D_3\ D_4 \quad (1)$$

Code Data Bits 3-bit (C):

$$C_1 = D_1 \oplus D_2 \oplus D_4 \quad (2)$$

$$C_2 = D_1 \oplus D_3 \oplus D_4 \quad (3)$$

$$C_3 = D_2 \oplus D_3 \oplus D_4 \quad (4)$$

Hamming Code Data 7-bit (HC): **{**$HC = HC_7\ HC_6\ HC_5\ HC_4\ HC_3\ HC_2\ HC_1$**}**

$$HC = C_1\ C_2\ D_1\ C_3\ D_2\ D_3\ D_4 \quad (5)$$

Check Data 3-bit (A):

$$A_1 = C_1 \oplus D_1 \oplus D_2 \oplus D_4 \quad (6)$$

$$A_2 = C_2 \oplus D_1 \oplus D_3 \oplus D_4 \quad (7)$$

$$A_3 = C_3 \oplus D_2 \oplus D_3 \oplus D_4 \quad (8)$$

In the data transmission side, this system allows only a 4-bit input data by using this input data a 3-bit Code data bits will be generated and it will added with 4-bit input data, lastly 7-bit hamming coded data will come out. This is shown in the equations from 1 to 5.

TABLE 1
ERROR DETECTION AND CORRECTION OF HAMMING CODE DATA USING CHECK DATA BITS

Check Data	Hamming Code Error Bit Logic Value Toggling		
$A_3A_2A_1$	Data Bit	Received Data	Corrected Data
000	No Error	0	0
		1	1
001	HC_1	0	1
		1	0
010	HC_2	0	1
		1	0
011	HC_3	0	1
		1	0
100	HC_4	0	1
		1	0
101	HC_5	0	1
		1	0
110	HC_6	0	1
		1	0
111	HC_7	0	1
		1	0

In the receiving face, a 3-bit check data (A) will be generated by using the receiving 7-bit hamming code data. If A = "000" received data is correct and system will decoded original transmitted 4-bit word data from the received 7-bit hamming code bits. This is shown in the equations from 6 to 8. If there is only one bit error in the received 7-bit hamming code 'A' will show some value with respect to this value, appropriate one hamming code data bit logic will toggle (0 to 1 or 1 to 0). Error detection and correction of hamming code data using error detection data bits shown in the TABLE 1. From this corrected 7-bit hamming code data, system will generate original transmitted 4-bit data.

A. Verilog HDL for FPGA

We have used verilog HDL as developing tool for the FPGA for the proposed design. The hamming code top-level system has two models, transmitting (***hamingtx***) and receiving (***hamingrx***) sections. Fig. 2 shows the FPGA RTL schematic view of hamming code data encryption and decryption system. The input information data size for the *hamingtx* is 4-bit; the output of this *hamingtx* is encoded 7–bit hamming coded data. The output of *hamingtx* is input for the *hamingrx* section, it will decode the main 4-bit input data from the 7-bit hamming code data. Even if there is any one bit error it will give the corrected data. Device utilization report is shown in TABLE 2 and data in pad to hamming code pad delay is shown in TABLE 3.

TABLE 2
DEVICE UTILIZATION TABLE

Hamming Code Encoding System		
Sl. No.	Name	Number
1	Slices	3
2	LUTs	3
3	IOs	15
Hamming Code Decoding System		
Sl. No.	Name	Number
1	Slices	15
2	LUTs	32
3	IOs	37

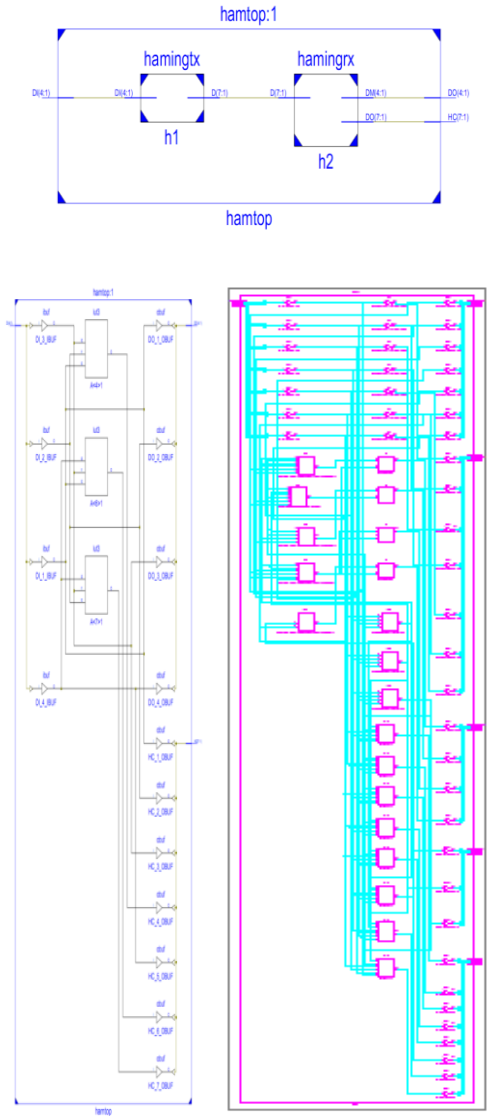

Fig.2 FPGA RTL Schematic View of Hamming Code Data Encryption and Decryption System

B. CMOS Layout Design

For more than 40 years, CMOS device technologies have been improving at a dramatic rate. A large part of the success of the MOSFET is due to the fact that it can be scaled to increasingly smaller dimensions, which results in higher performance. In the past 40 years, the MOSFET gate length has scaled from 10 m to 45 nm. The ability to consistently improve performance while decreasing power consumption has made CMOS architecture the dominant technology for integrated circuits [4].

We have implemented the layout of hamming code data encryption and decryption system in 90nm, 70nm and 50nm technologies. The CMOS layout is shown in fig. 3. The comparison of number of metals, operation voltage, layout width, layout height and total layout surface with nanometer technology is shown in the table 4. In this system implementation 262 PMOS and 226 NMOS transistors are utilized.

TABLE 3
PAD TO PAD DELAYS OF DATA IN TO HAMMING CODE SWAP

Sl. No.	Data In Pad (Tx)	Hamming Code Pad	Delay (ns)
1	DI<1>	HC<1>	5.643
2	DI<1>	HC<4>	6.271
3	DI<1>	HC<6>	6.179
4	DI<2>	HC<2>	5.551
5	DI<2>	HC<4>	6.649
6	DI<2>	HC<7>	6.165
7	DI<3>	HC<3>	5.604
8	DI<3>	HC<4>	6.409
9	DI<3>	HC<6>	6.320
10	DI<3>	HC<7>	6.170
11	DI<4>	HC<5>	5.611
12	DI<4>	HC<6>	6.196
13	DI<4>	HC<7>	6.315

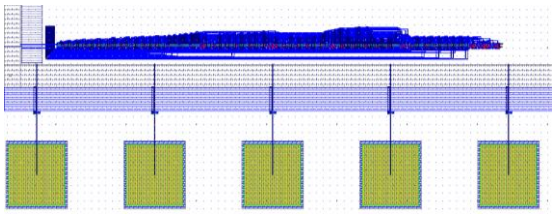

Fig.3 CMOS Layout of Hamming Code Data Encryption and Decryption System

TABLE IV
CMOS LAYOUT COMPARISON IN 90NM, 70NM AND 50NM TECHNOLOGY

Nanometer Size	Number of Metals	Operation Voltage	Total Layout Surface
90nm	6	1 V	6764.5µm2
70nm	6	0.7 V	6277.8µm2
50nm	7	0.5 V	3203.0µm2

IV. SIMULATION RESULT

After successful implementation of FPGA and CMOS layouts of the hamming code data encryption and decryption system, we had tested all the possible inputs and outputs. We got the accurate results and even one bit error is present in the received data this system is recovered the original data from the error data. If the error bits are more than one this system shows only error is available in the received data but not recover the original data. All the possible input condition test results of data encryption and data decryption are shown in the fig. 4 and fig. 5.

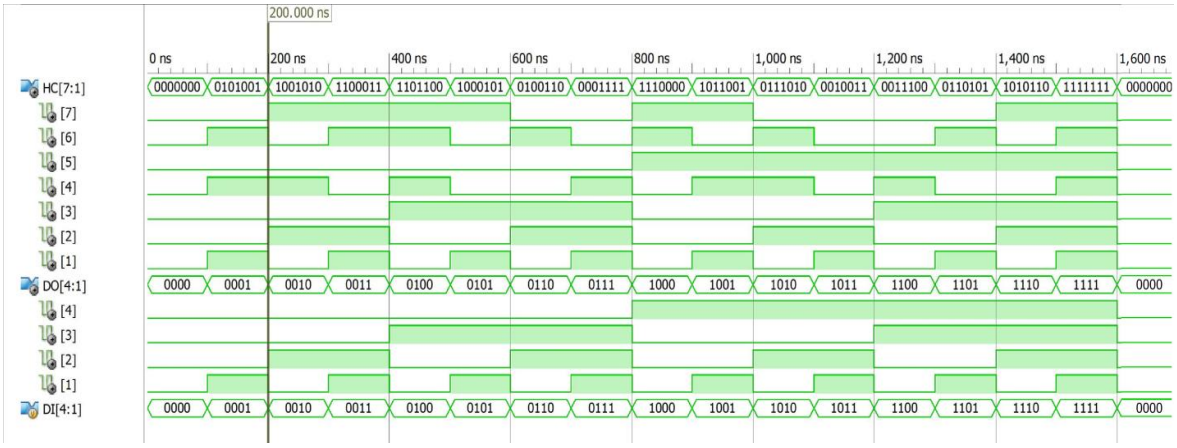

Fig.4 Simulation timing diagram of hamming code data encryption system

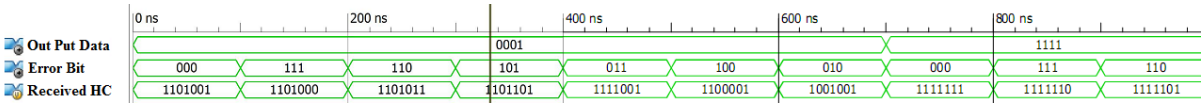

Fig.5 Simulation timing diagram of hamming code data decryption system

V. CONCLUSION

We have implemented the FPGA for hamming code data encryption and decryption system and layout in a typical 90 nm, 70nm and 50nm CMOS process. We have simulated and tested the system and got an excellent performance at 50GHz. We have found that, reliability considerations aside, at a voltage of 1 V, 0.7V, and 0.5 V this device offer the best performance. On the other hand, when consideration is given to device layout surface, the 90 nm device is 6764.5 µm2, 70nm device is 6277.8 µm2 and 50nm device is 3203.0 µm2.

REFERENCES

[1] Z. Al-Ars and A. J. van de Goor, "Soft faults and importance of stresses in memory testing," Design, Automation and Test in Europe Conference and Exhibition, pp. 1084–1089, Feb. 2004.

[2] R. W. Hamming, "Error detecting and error correcting code," Bell System Technical Journal, vol. 26, pp. 147–160, Apr.1950.

[3] P. K. Lala, P. Thenappan, and M. T. Anwar, "Single error correcting and double error detecting coding scheme," IET Electronics Letters, vol. 41, Issue 13, pp. 758–760, Feb. 2005.

[4] Rick Ma and Samuel Cheng "The Universality of Generalized Hamming Code for Multiple Sources", IEEE Transactions on Communications, VOL. 59, NO. 10, PP. 2641- 2647, OCTOBER 2011