© 2018 IJSRST | Volume 4 | Issue 3 | Print ISSN : 2395-6011 | Online ISSN: 2395-602X

National Conference on Advances in Engineering and Applied Science (NCAEAS)
29th January 2018
Organized by : Anjuman College of Engineering and Technology (ACET) Nagpur, Maharashtra, India, In association with
International Journal of Scientific Research in Science and Technology

Review on Detection of Error and Correction of Corrupted Code Using Fpga Implementation

Yogeshwari Paidlewar[1], Shagufa Anjum[2], Zubaida Khatoon[2], Naghma Tarrannum[2],Prof. Mohsina Anjum[2]
[1]B.E. Student, Electronics & Telecommunication Engineering, Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur, Maharashtra, India
[2]Asst. Prof., Electronics & Telecommunication Engineering, Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur, Maharashtra, India

ABSTRACT

In this paper, the error detection and correction of the corrupted code is done by FPGA implementation using Xilinx and Modelsim software. In information theory and coding theory with applications in computer science and telecommunication, error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels. When data is stored, compressed, or communicated through a media such as cable or air, sources of noise and other parameters such as EMI, crosstalk, and distance can considerably affect the reliability of these data. Error detection and correction techniques are therefore required. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data in many cases. In coding theory ,hamming (7,4) is a linear error-correcting code that encodes four bits of data into seven bits by adding three parity bits.Hamming's (7,4) algorithm can correct any single bit error ,or detect all single-bit and two-bit errors.

Keywords: Error detection and correction, FPGA, Hamming code, Xilinx and Modelsim.

I. INTRODUCTION

In information and communication technology, when data is stored, compressed, communicated through a media such as cable or air, sources of noise and other parameters such as EMI, crosstalk, and distance can considerably affect the reliability of these data. Error detection and correction techniques are therefore required. Error coding is a method of detecting and correcting these errors in a wide range of communication systems in computer memory, magnetic and optical data storage media, satellite and deep space communications, network communications, cellular telephone networks, and almost any other form of digital data communication Digital data is transmitted over a channel and there is often noise in the channel. The noise may distort the messages to be sent. Therefore, what the receiver receives may not be the same as what the sender sends. The goal of error coding is to improve the reliability of digital communication by error detection and error correction. In this paper we have written Verilog code for finding error location and correct the bit which is corrupted. At the destination, we receive 7-bit of data with 4 redundancy bits. This received data may be corrupted due to noise. To remove this noise we find the address of the error bit then correct them. To find the location of error bit and correct them we write code in Verilog language. This paper is organized as follows: the concept of hamming code along with the application of it, Verilog language, Why Verilog is preferred over VHDL, Implementation of hamming code, Performance and experimental results, conclusion and references.

1. HAMMING CODE

In telecommunication, 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 without detection of uncorrected 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 of three.

To improve system reliability, a designer may wish to provide an automatic error detection and correction circuit. One such example is the data communicated from the microprocessor to peripheral memory devices. This document describes a flow-through method for doing data SECDED. In this design, multiple parity bits are added to the data word upon a write to memory. With multiple parity bits, both single and double data errors can be detected upon reading the word from memory and correct single data errors. The system provides a 2-bit error output flag for the microprocessor to handle detected double errors.

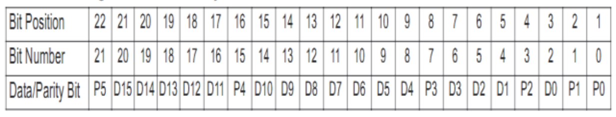
The SECDED design described here is the combinational logic for data communication between the microprocessor and memory. The data bus from the processor is 16-bit wide data, while the data written to memory is a 22-bit data word. When data is read back from the memory device, the stored parity bits are compared with a newly created set of parity bits from the read data. The result of this comparison, called the syndrome, will indicate the incorrect bit position in a single data error.

This design is a model of the Hamming code developed by R. Hamming. SECDED for N bits of data requires K parity bits to be stored with the data where:

N<=2K −1−K

If the bits are numbered in sequence, those bit positions that represent powers of two are dedicated to parity bits. Table 1 illustrates how the 16-bit data word and parity bits are stored in memory.

Table 1. Hamming code data and parity bits



Bit Position	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Bit Number	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data/Parity Bit	P5	D15	D14	D13	D12	D11	P4	D10	D9	D8	D7	D6	D5	D4	P3	D3	D2	D1	P2	D0	P1	P0

The parity bits P0-P4 are created for single error detection and correction and are created as follows:

$PO=D15\oplus D13\oplus D11\oplus D10\oplus D8\oplus D6\oplus D4\oplus D3\oplus D1\oplus D0$

$P1=D13\oplus D12\oplus D10\oplus D9\oplus D6\oplus D5\oplus D3\oplus D2\oplus D0$

$P2=D15\oplus D14\oplus D10\oplus D9\oplus D8\oplus D7\oplus D3\oplus D2\oplus D1$

$P3=D10\oplus D9\oplus D8\oplus D7\oplus D6\oplus D5\oplus D4$

$P4=D15\oplus D14\oplus D13\oplus D12\oplus D11$

One additional parity bit, P5, detects double errors that are not correctable. This extra parity bit is an overall parity bit and is comprised by XOR-ing all the data bits, D15-D0 and parity bits, P0- P4.

The syndrome is created upon a memory read and provides the ability to correct single bit errors. The syndrome is created by XOR-ing the parity bits read out of memory with the newly created set of parity bits from the data stored in memory. The value of the syndrome will indicate the bit position in error (if a single error has occurred). Table 2 illustrates the value of the syndrome and the overall parity bit in detecting both single and double errors.

Table 2. Error detection flags

error_out[1:0]	Description
00	No error has occurred.
01	Single error has been detected. Syndrome holds value of erroneous bit.
10	Double error has been detected. Not correctable
11	Parity error has occurred. Correctable.

Figure 1 illustrates the block level design of the SECDED. The left side of the diagram illustrates the processor interface. This interface consists of the 16-bit processor data bus, u_data [15:0], the read/write control signal, rw_n, and the error flag signal, error_out [1:0]. The right hand side describes the memory component interface, consisting of the memory data bus, mem_data [21:0].

The rw_n control signal from the processor switches between read and writes cycles. The rw_n signal will be equal to "1" for a processor read cycle and equal to "0" for a processor write cycle.

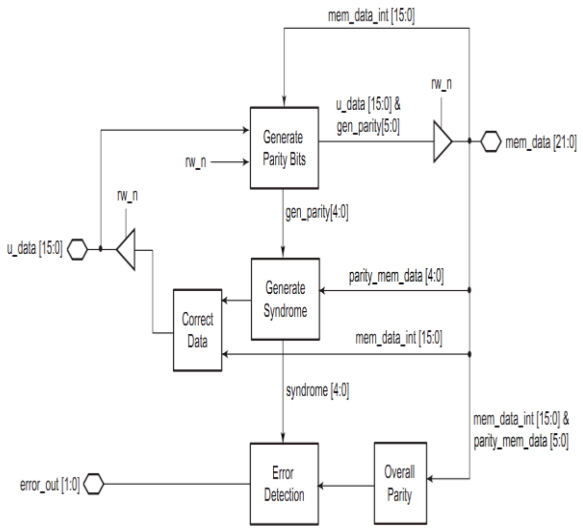
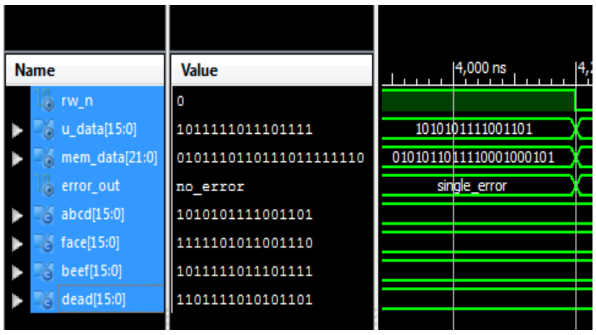


Figure 1. Block diagram of SECDED

The "Generate Parity Bits" block creates the parity bits to store with the processor data (u_data [15:0]) during a write cycle. In a read cycle, this block is also responsible for creating one of the inputs in generating the syndrome; this block creates the parity bits with the data word stored in memory.

The "Error Detection" block generates the error_out [1:0] flag based on the syndrome and the overall parity created from the data in memory.

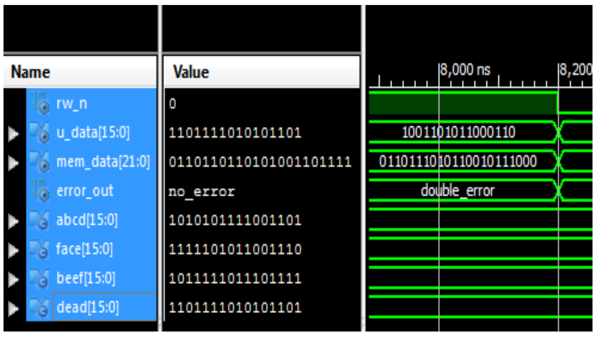
2.SINGLE BIT ERROR: The term single-bit error means that only one bit of given data unit (such as a byte, character, or data unit) is changed from 1 to 0 or from 0 to 1.The following fig shows the
detection of single bit error on the software.



```
Name            Value                    4,000 ns
rw_n            0
u_data[15:0]    1011111011101111         1010101111001101
mem_data[21:0]  0101110110111011111110   0101011011110001000101
error_out       no_error                 single_error
abcd[15:0]      1010101111001101
face[15:0]      1111101011001110
beef[15:0]      1011111011101111
dead[15:0]      1101111010101101
```

Figure 2. Single Bit Error Detection

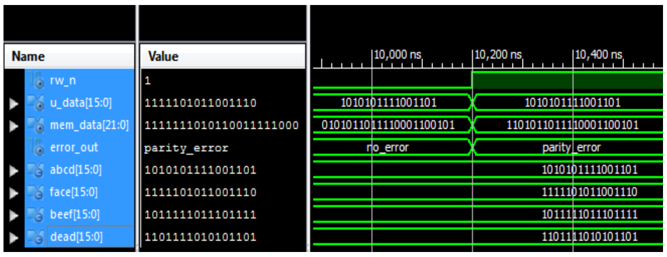
3.DOUBLE BIT ERROR: Double bit errors in a Hamming code cause trouble. Two bit errors will always be detected as an error, but the wrong bit will get flipped by the correction logic, resulting in gibberish. If there are more than two bits in error, the received codeword may appear to be a valid one (but different from the original), which means that the error may or may not be detected.



Name	Value
rw_n	0
u_data[15:0]	1101111010101101
mem_data[21:0]	0110110110101001101111
error_out	no_error
abcd[15:0]	1010101111001101
face[15:0]	1111101011001110
beef[15:0]	1011111011101111
dead[15:0]	1101111010101101

Figure 3. Double Bit Error Detection

4.PARITY ERROR: A parity bit, or check bit, is a bit added to a string of binary code to ensure that the total number of 1-bitsin the string is even or odd. Parity bits are used as the simplest form of error detecting code.



Name	Value
rw_n	1
u_data[15:0]	1111101011001110
mem_data[21:0]	1111111010110011111000
error_out	parity_error
abcd[15:0]	1010101111001101
face[15:0]	1111101011001110
beef[15:0]	1011111011101111
dead[15:0]	1101111010101101

Figure 4. Parity Error Detection

FPGA:

After a design passes basic the functional validations, it must be synthesized into a netlist of components of a target library. The target library is the specification of the hardware that the design is being synthesized to. A field-programmable gate array (FPGA) is an integrated circuit (IC) that can be programmed in the field after manufacture. FPGAs are similar in principle to, but have vastly wider potential application than, programmable read-only memory (PROM) chips. FPGAs are used by engineers in the design of specialized ICs that can later be produced hard-wired in large quantities for distribution to computer manufacturers and end users. Ultimately, FPGAs might allow computer users to tailor microprocessors to meet their own individual needs. We have show some results of detection above. The following figure shows the flow of the project.

II. CONCLUSION

In this paper, the conclusion can be made as this is the case in computer memory, where bit errors are extremely rare and hamming codes can be widely used. Extended hamming codes can achieve a hamming distance of four, which allows the decoder to distinguish between when at most one bit error occurs and when any two-bit error occurs. In this sense, extended hamming codes are single error correcting and double error detecting, abbreviated as SECDED.

III. REFERENCES

[1]. Nuh Aydin: An Introduction to Coding Theory via Hamming Codes. Department of Mathematics Kenyon College.

[2]. Ranpara, S.; Dong Sam Ha, 1999. A low-power Viterbi decoder design for wireless communications applications. IEEE Proceedings of the Twelfth Annual IEEE International Int. ASIC Conference 1999, Washington, DC, 15-18 Sept. 1999, pp. 377-381.

[3]. Leena, Mr. Subham Gandhi and Mr. Jitender Khurana, "Implementing (7,4) Hamming Code using CPLD on VHDL" International Journal of New Trends in Electronics , Vol. 1, Issue 1, Aug. 2013.

[4]. Xilinx "Synthesis and Simulation Design Guide", Xilinx Tech UG626 2012.

[5]. Ming- Bo Lin "Digital System Design and Practices using Verilog HDL and FPGA", Wiley-India, ISBN: 978-81-265-3694-8 .

[6]. Information Theory Coding and Cryptography by Ranjan Bose.

[7]. http://www.xillinx.com/training/xillinx-training- courses.pdf.

[8]. Verilog HDL: A guide to digital design and synthesis, second edition by Sameer Palnitkar.