IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

Design and Simulation of Error Correction Codes

Chaitra Vardhini H[1], Prof. Madhu Patil[2]

M. Tech Scholar, Department of ECE, NMIT, Bangalore, India[1]

Associate Professor, Department of ECE, NMIT, Bangalore, India[2]

Abstract: Natural interference like EMI, noise, crosstalk can happen over the communication channel like memory, which causes the original data to be different from the stored data. In order to find these errors a few techniques to recognize and correct the error is required. This work is an overview of different Error Correction techniques. These techniques guarantee to find and possibly correct the errors brought about by the stuck-at faults in the memory. The work is mainly focused on Hamming codes, Convolution codes, CRC and Turbo codes.

Keywords: Hamming codes, Convolution codes, CRC - Cyclic Redundancy Check and Turbo codes.

I. INTRODUCTION

Environmental interference and physical defects like noise, EMI and crosstalk in the communication channel can cause random errors during data transmission. The method of detecting and correcting these random errors to ensure information is transferred intact from its source to its destination is called Error coding. Error coding is used for fault tolerant computing in computer memory, satellite communication, space communication, magnetic data storage and optical storage media network communications, cellular telephonic networks, and digital data communication. Error coding encodes the information bits into longer bits called code words for transmitting the data. These code words can be decoded at reception in order to get the original information bits. Redundancy is because of the extra bits that is present in the code word that will allow the reception to use the decoding process to find if the communication medium introduced errors and in some cases correct them so that the data need not be retransmitted.

II. METHODOLOGY

Initially, the input data bits are given to the encoder, where encoder converts input data bits into code words. Then this encoded sequence is stored on to the memory. Memory is introduced with a stuck-at-fault (error) which might be either stuck-at-0 or stuck-at-1. The data in the memory is read and given as the input to the decoder. Decoder detects the fault and possibly the position of the fault present in the memory.

III. HAMMING CODES

Hamming codes is called after its designer; Richard Hamming from Bell Labs is a Linear Error Correction Codes. Hamming codes can detect two bit errors and correct single bit error. If the Hamming distance between the sent data and the received data bit is less than or equal to one then reliable communication is possible. It is not reasonable for transmission circumstance where burst errors happen. Hamming Codes (n, k) adds n-k no of check bits to every k -bits of message data.

DATA INPUT	11000100	10011010
DATA WITH PARITY BITS	P1 P2 1 P4 100 P8 0100	P1 P2 1 P4 001 P8 1010
CALCULATION OF PARITY BITS	P1 XOR of bits (3, 5, 7, 9, 11) P2 XOR of bits (3, 6, 7, 10, 11) P4 XOR of bits (5, 6, 7, 12) P8 XOR of bits (9, 10, 11, 12)	P1 XOR of bits (3, 5, 7, 9, 11) P2 XOR of bits (3, 6, 7, 10, 11) P4 XOR of bits (5, 6, 7, 12) P8 XOR of bits (9, 10, 11, 12)
SET PARITY BITS EVEN = 0 ODD = 1	1+1+0+0+0=0 Even Parity P1=0 1+0+0+1+0=0 Even Parity P2=0 1+0+0+0=1 Odd Parity P3=1 0+1+0+0=1 Odd Parity P4=1	1+0+1+1+1=0 Even Parity P1=0 1+0+1+0+1=1 Odd Parity P2=1 0+0+1+0=1 Odd Parity P3=1 1+0+1+0=0 Even Parity P4=1
CODEWORD	0 0 1 1 1 0 0 1 0 1 0 0	0 1 1 1 0 0 1 0 1 0 1 0
ERROR CODEWORD STORED ON MEMORY	1 0 1 1 1 0 0 1 0 1 0 0	0 1 1 1 0 0 1 0 1 1 1 0
EVALUATE CHECK BITS	C1 XOR of bits (1, 3, 5, 7, 9, 11) C2 XOR of bits (2, 3, 6, 7, 10, 11) C4 XOR of bits (4, 5, 6, 7, 12) C8 XOR of bits (8, 9, 10, 11, 12)	C1 XOR of bits (1, 3, 5, 7, 9, 11) C2 XOR of bits (2, 3, 6, 7, 10, 11) C4 XOR of bits (4, 5, 6, 7, 12) C8 XOR of bits (8, 9, 10, 11, 12)
CHECK BITS PARITY	1+1+1+0+0+0=1 Odd Parity 0+1+0+0+1+0=0 Even Parity 1+1+0+0+0=0 Even Parity 1+0+1+0+0=0 Even Parity	0+1+0+1+1+1=0 Even Parity 1+1+0+1+1+1=1 Odd Parity 1+0+0+1+0=0 Even Parity 0+1+1+1+0=1 Odd Parity
ERROR BIT POSITION	C8C4C2C1 = 0001	C8C4C2C1 = 1010

Figure 1: (12, 8) Hamming Codes

IV. CONVOLUTIONAL CODES

Convolutional codes are designed using two parameters:

1. The code rate
2. The constraint length

Code rate – (k/n) is defined as the ratio of the number of bits input to the encoder to the number of bits output by the encoder.

Constraint length (k) denotes the length of convolutional encoder, i.e. how many k-bit stages are available to input the combinatorial logic that produces the output code words.

'm' indicates the number of shift registers in the encoder design. It is simply the memory length of the encoder.

Convolutional codes are used in real time communication systems for error correction. The code words depend on the present 'k' message bits and some past input bits. The

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

main decoder combination of convolutional encoder is Viterbi Algorithm.

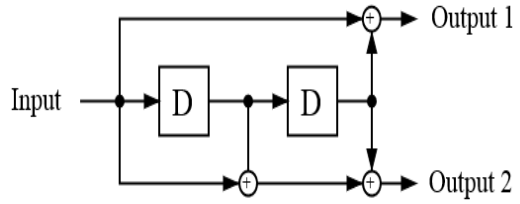

Figure 2: k=1, n=2 and r=1/2 Convolutional Encoder.

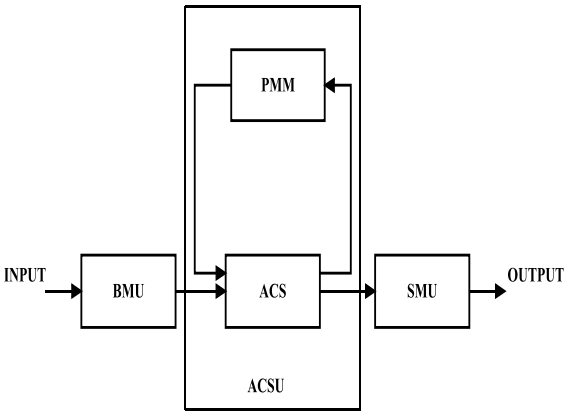

Figure 3: Viterbi Decoder Block Diagram.

DATA INPUT	1000	1011	0001	1100
INPUT TO THE ENCODER	00100000	00101100	00000100	00110000
OUTPUT OF THE ENCODER	C1: 101000 C2: 111000	C1: 100111 C2: 110001	C1: 000101 C2: 000111	C1: 111100 C2: 100100
CODEWORD GENERATED	110110000000	110100101010	000000110111	111010110000
CODEWORD STORED ON FAULTY SRAM	110010000000	100100101010	000000010111	111010100000
INPUT TO THE DECODER	C1: 1010000 C2: 1000000	C1: 100111 C2: 100001	C1: 000001 C2: 000111	C1: 111100 C2: 100000
OUTPUT OF THE DECODER	00100000	00101100	00000100	00110000
CODEWORD OF THE RETRIEVED DATA	110110000000	110100101010	000000110111	111010110000
XORING CODEWORD OF SRAM AND DATA RETRIEVED	110110000000 XOR 110010000000	110100101010 XOR 100100101010	000000110111 XOR 000000010111	111010110000 XOR 111010100000
FAULTY SRAM POSITION	000100000000	010000000000	000000100000	000000010000

Figure 4: Example of Convolution Codes.

V. CYCLIC REDUNDANCY CHECK – CRC

CRC - Cyclic Redundancy Check is a technique for detecting errors in digital information transmission, however not for error correction after errors are recognized. In the CRC strategy, some number of check bits is added. These check bits are called checksum. They are connected to the information bits that are being stored. The beneficiary figures out whether the check bits concur with the information bits. On the off chance that an error has happened then fault location in the memory is detected and replaced with the fault free memory.

Dataword	Codeword	Dataword	Codeword
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Figure 5: Example of Generated code words using CRC.

VI. TURBO CODES

In 1993 Berrou, Glavieux and Thitimajshima proposed "another class of convolution codes called as turbo codes". It a class of Forward Error Correction Codes widely used in communication system.

A turbo code is a parallel concatenation of many RSC codes. The encoder consists of two identical RCS – Recursive Systematic Encoder in parallel. The output of encoder1 and encoder2 are got from the same massage bits. The output code rate of both the encoders is at a rate R=1/3 code.

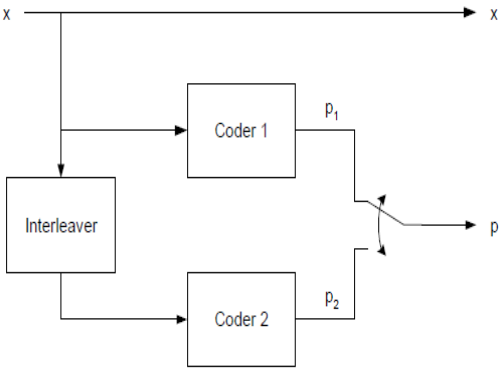

Figure 6: Turbo Encoder Block Diagram.

DATA INPUT	1110
ENCODER 1 INPUT	01110
ENCODER 1 OUTPUT	0011101100
INTERLEAVER INPUT	1110
INTERLEAVER OUTPUT	0111
ENCODER 2 INPUT	00111
ENCODER 2 OUTPUT	0000111011

Figure 7: Turbo Encoder example

DOI 10.17148/IJARCCE.2016.5728

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

VII. SIMULATION RESULTS

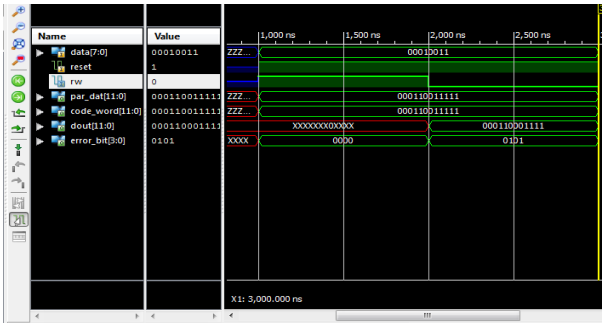

Figure 8: Hamming codes implemented for SRAM.

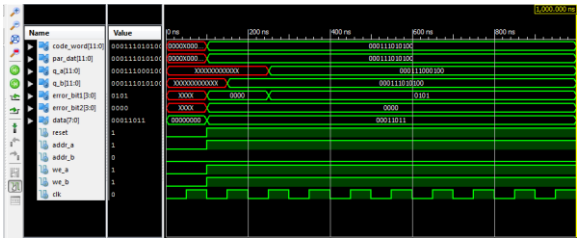

Figure 9: Hamming codes implemented for Dual Port Memory.

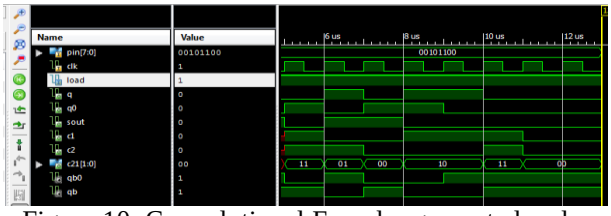

Figure 10: Convolutional Encoder generated code words.

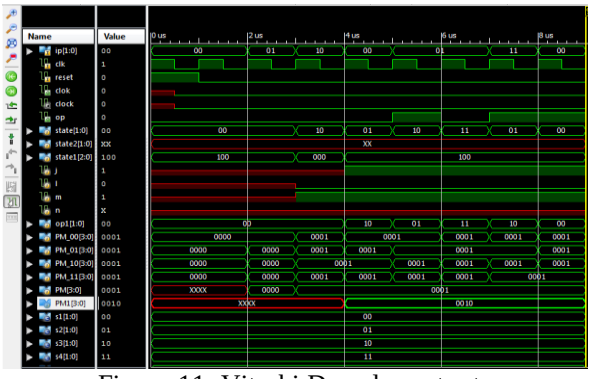

Figure 11: Viterbi Decoder output.

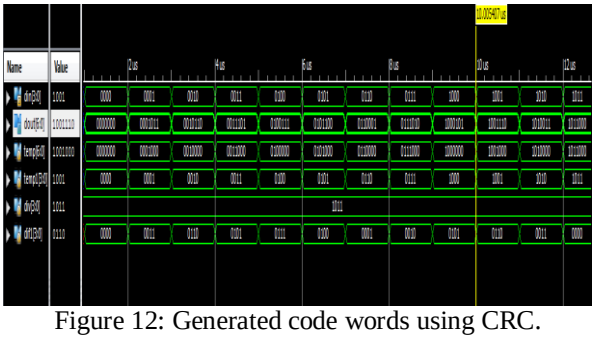

Figure 12: Generated code words using CRC.

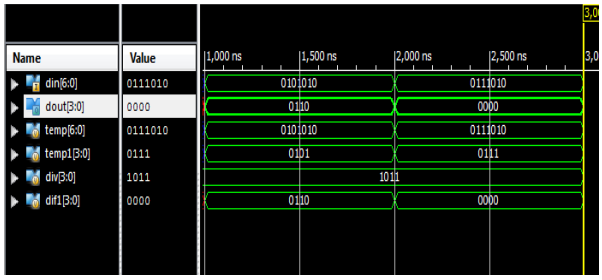

Figure 13: CRC code words stored on Memory with error.

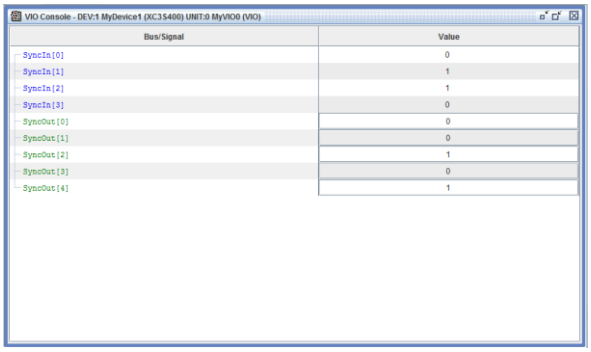

Figure14: Output of CRC using Chipscope Pro

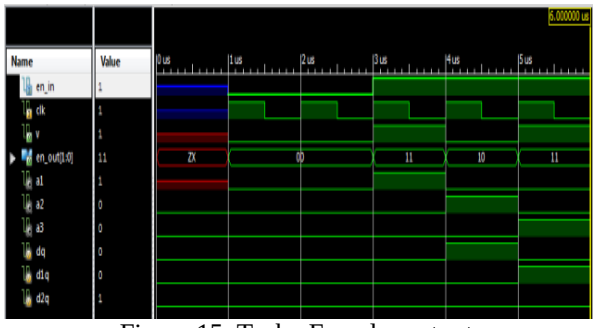

Figure 15: Turbo Encoder output.

VIII. COMPARISON OF ERROR CORRECTION CODES TECHNIQUES

Logic Utilization	SRAM	DUAL PORT MEMORY
No. of Slice Latches	8(1%)	15(1%)
No. of occupied Slices	13(1%)	18(1%)
Total no. of 4 input LUTs	25(1%)	21(1%)
Maximum path delay	8.348ns	6.530ns
Total On-Chip Power	69.18mW	60.66mW

Figure 16: Hamming Codes comparison with SRAM and Dual Port Memory.

Logic Utilization	SRAM	DUAL PORT MEMORY
No. of Slice Latches	8(1%)	368(5%)
No. of occupied Slices	8(1%)	336(9%)
Total no. of 4 input LUTs	14(1%)	325(4%)
Maximum path delay	10.991ns	10.290ns
Total On-Chip Power	59.79mW	60.00mW

Figure 17: CRC comparison with SRAM and Dual Port Memory

DOI 10.17148/IJARCCE.2016.5728

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

Logic Utilization	CRC SRAM	HAMMING SRAM
No. of Slice Latches	8(1%)	8(1%)
No. of occupied Slices	8(1%)	13(1%)
Total no. of 4 input LUTs	14(1%)	25(1%)
Maximum path delay	10.991ns	8.348ns
Total On-Chip Power	59.79mW	69.18mW

Figure 18: Comparison of SRAM of CRC and SRAM of Hamming Codes

Logic Utilization	HAMMING DUAL PORT MEMORY	CRC DUAL PORT MEMORY
No. of Slice Latches	15(1%)	368(5%)
No. of occupied Slices	18(1%)	336(9%)
Total no. of 4 input LUTs	21(1%)	325(4%)
Maximum path delay	6.530ns	10.290ns
Total On-Chip Power	60.66mW	60.00mW

Figure 19: Comparison of CRC codes and Hamming codes using Dual Port Memory

Logic Utilization	CONVOLUTIONAL ENCODER	VITERBI DECODER
No. of Slices	7 out of 3584	89 out of 3584
No. of occupied Slices	6(1%)	92(2%)
Total no. of 4 input LUTs	12(1%)	171(2%)
Maximum path delay	8.094ns	21.370ns
Total On-Chip Power	60.60mW	60.55mW

Figure 20: Comparison of Convolutional Encoder and Viterbi Decoder.

Logic Utilization	TURBO ENCODER
No. of Slice Flip Flops	2 out of 7168(1%)
No. of occupied Slices	2(1%)
Total no. of 4 input LUTs	2(1%)
Time from CPU to Xst	25.49ns
Total On-Chip Power	60.00mW

Figure 21: Design Summary of Turbo Encoder.

IX. CONCLUSION

The aim of this work is to mainly concentrate on Error Correction Coding techniques like Hamming codes, Convolution codes, Cyclic Redundancy Check and turbo codes. These codes are designed with Verilog language and simulated on SPARTAN-3 using XILINX ISE 14.2 and Chipscope Pro.

Though it is very difficult to tell which the best technique is, it is better to run different techniques in parallel and take up the reliability and quality of different techniques to get best throughput.

REFERENCES

[1] Rubal Chaudhary, Vrinda Gupta, "Error Control Techniques and Their Applications", International Journal of Computer Applications in Engineering Sciences.

[2] Prof. Siddeeq Y. Ameen, Mohammed H. Al-Jammas and Ahmed S. Alenezi,"FPGA Implementation of Modified Architecture for Adaptive Viterbi Decoder" IEEE,2011.

[3] Wael. M. El-Medany, "FPGA Implementation of CRC with Error Correction", ICWMC 2012, The Eighth International Conference on Wireless and Mobile Communications.

[4] Miss.Rupinder Kaur, Mr.Shakti Raj Chopra. "Iterative Decoding of Turbo Codes", International Journal of Scientific & Engineering Research, Volume 4, Issue 6, June-2013.

[5] Eltayeb S. Abuelyaman and Abdul-Aziz S. Al-Sehibani, "Optimization of the Hamming Code for Error Prone Media", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.3, March 2008.