International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25449-25456

http://www.ripublication.com

Multiple Error Detection and Correction Using Orthogonal Latin Squares (Ols) and Majority Logic Decoders

Dr. Fazal Noorbasha[1], Yamini Pallapolu[2]
[1]*Associate Professor, Department of ECE, K L University, Guntur, AP, India-522502*
[2] *M.Tech. Student, Department of ECE, K L University, Guntur, AP, India -522502*
fazalnoorbasha@kluniversity.in; p.yamini112@gmail.com;

Abstract

Error detection and correction in encoding and decoding using majority logic is a well known process for 2 and 3 bits but now with the help of "Orthogonal Latin Squares" and MLD's 4 bit and 5 bit error detection and correction made by the instance. This is a three sectional architecture for tracing the error and correcting the same within a given clock cycle period of time. This mainly helps in reducing the clock cycles utilising for detecting and correcting the error in the encoded data bits or word.

Key Words: Error detection, correction, decoding, OLS, MLD.

Introduction

To protect cache memories and advancement to single bit error correction and detection codes[1] is used. Different types of Error correcting codes are Binary codes, Non-binary codes. An H-matrix, generated using the semi-random technique, consists of two parts, Hd and Hp , Hd being a randomly generated form and Hp a deterministic form[1] . The matrix structure makes encoding processes simple because the deterministic form of Hp shown in Figure 1 is a dual-diagonal square matrix.

```
1  0  0  1  0  0 ...   1  0  0
1  1  0  0  1  0 ...   0  1  0
0  0  1  0  0  1 ...   0  0  1
1  0  0  1  0  0 ...   1  0  0
1  1  0  0  1  0 ...   0  1  0
0  0  1  0  0  1 ...   0  0  1
```

Figure 1: Hybrid matrix with parity.

Although an H_d -matrix is randomly generated, some efficient forms of matrices are preferred for better coding performance if possible. Once an H_d -matrix is

generated, the H-matrix is constructed as H = [H_d , H_p]. When a codeword C = [d, p]t , where d and p are information bits and parity bits, respectively, parity bits can be easily calculated .

$$M = \sum_j h_j^d dj \quad (1)$$

Here in equation (1) 'M' is the parity matrix and 'h' is the hybrid matrix and'd' is the diagonal matrix.

Table 1: Algerian representation of orthogonal latin squares

	F1	F2	F3
A1	$t_1 m_1$	$t_1 m_2$	$t_1 m_3$
A2	$t_2 m_1$	$t_2 m_2$	$t_2 m_3$
A3	$t_3 m_1$	$t_3 m_2$	$t_3 m_3$

Orthogonal Latin Squares are generally hard to come by. There are no orthogonal latin squares of order 2 because there are only two latin squares of order 2 in the same symbols and they are not orthogonal. There are orthogonal latin squares of order 3 as exemplified above. Orthogonal latin squares of order 4 exist but won't be exposed without a little struggle[2]. Orthogonal latin squares of order 5, or any odd order, on the other hand, are not so hard to find[1,7,8,9].

Low-density parity-check (LDPC) codes are the most powerful error correction codes (ECCs) and approach the Shannon limit, while having a relatively low decoding complexity. Therefore, the idea of combining LDPC codes and bandwidth-efficient modulation has been widely considered.

Let $A = (i + j)$ be the addition table for the integers modulo $2n + 1$ and let $B = (2i + j)$, entries taken modulo 2n+1. Then A and B are orthogonal latin squares. For, A is a latin square because it is the addition table for the integers modulo 2n+1 and B is a latin square because it is just A with its rows rearranged. To show that A and B are orthogonal, suppose that for some i,j,m,n, the ordered pairs $(i + j, 2i + j)$ and $(m + n, 2m + n)$ are equal. Then $i + j = m + n$ and $2i + j = 2m + n$. Subtracting (modulo 2n+1) the first equation from the last, we get i=m, from which it follows that j=n^3.Hence all ordered pairs from different cells of the two latin squares are distinct and the squares are orthogonal. Here is an example for order 4 and tabulated below.

Table 2: orthogonal latin square matrix with Galios field order 4

0	1	A	a+1
1	0	a+1	a
A	a+1	0	1
a+1	A	1	0

Let A=(s+t) be the above latin square with s and t ranging over {0,1,a,a+1}. As we did earlier with modular arithmetic, but now with this new arithmetic, form the latin

squares B=(as+t) and C=((a+1)s+t). Then, using the same argument used with earlier with the integers modulo 2n+1, the latin squares A, B, and C are mutually orthogonal. Here are B and C.

Prior to MLD's reed-Solomon codes, Bose- Chaudari- Hacqenghem codes were used with maximum complex iterations[3]. To reduce the complex iteration LDPC were followed on later these were applied by the distance calculation methods such as Hamming distance and Euclidean distance methods for tracing an error of 2 or 3 bits error detection and correction is possible for a maximum of 2 bits.

So to break this montage OLSMLD is introducing for correcting maximum number of data bits with n^2 value of error detection and correction capability[6].

Proposed Work

Table 3: Proposing code parameters are K and n-K

K	n-K
24	16
32	16

The proposed architecture is consisting of 1X2 network i.e, 16X32 encoder and the OLSMLD is for (32, 16) decoding strategy. Figure 2 indicating the block schematic of error correcting and detecting.

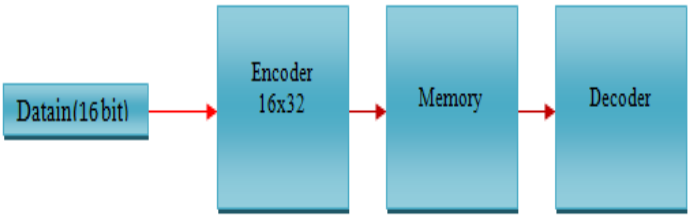


Figure 2: Block Schematic of Encoder and decoder of error correction and detection

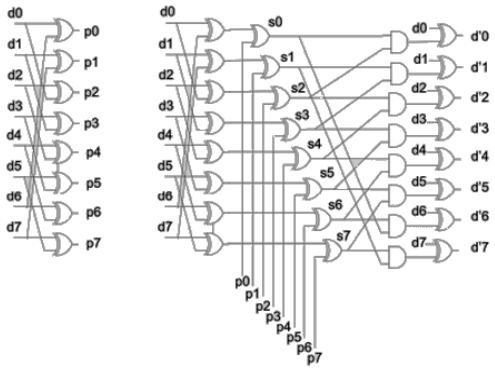


Figure 3: Encoder and decoder of OLS scheme with error correction capability.

Figure 3 indicates an example of 8 bit data encoding and decoding scheme with error correction capability along with the syndrome test example.

Encoding Process

The entire encoding process is depends on OLS codes and is mathematically representing as follows in below equation prior to that the encoder architecture for this mode is consists of 8 sections each section comprises with another 4 blocks. The example representation is as below for 32 bit encoding process for 16 bit input. Here N = 32 and n = 16. [5]

Encoded Output [0] = DataIn[0] xor DataIn[1] xor DataIn[2] xor DataIn[3]
Encoded Output [1] = DataIn[4] xor DataIn[5] xor DataIn[6] xor DataIn[7]
Encoded Output [2] = DataIn[8] xor DataIn[9] xor DataIn[10] xor DataIn[11]
Encoded Output [3] = DataIn[12] xor DataIn[13] xor DataIn[14] xor DataIn[15]
Encoded Output [31] = DataIn[15] xor DataIn[0] xor DataIn[1] xor DataIn[2]

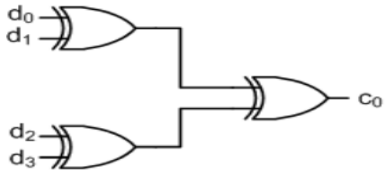


Figure 4: A Single bit OLS encoder.

Decoding Process

In decoding process we are accessing MLD logic with the combination of OLS codes

Algorithm 1: MLD algorithm for detecting errors
Input: 32 bit encoding data and original data is 16 bit
STEP 1: N = 16 bit
STEP 2: Calculate H_d with respect matrix generation
STEP 3: If H_d <= T then all the decoded values are equals to '0'.
STEP4: Else all the decoded values are equals to '1'.

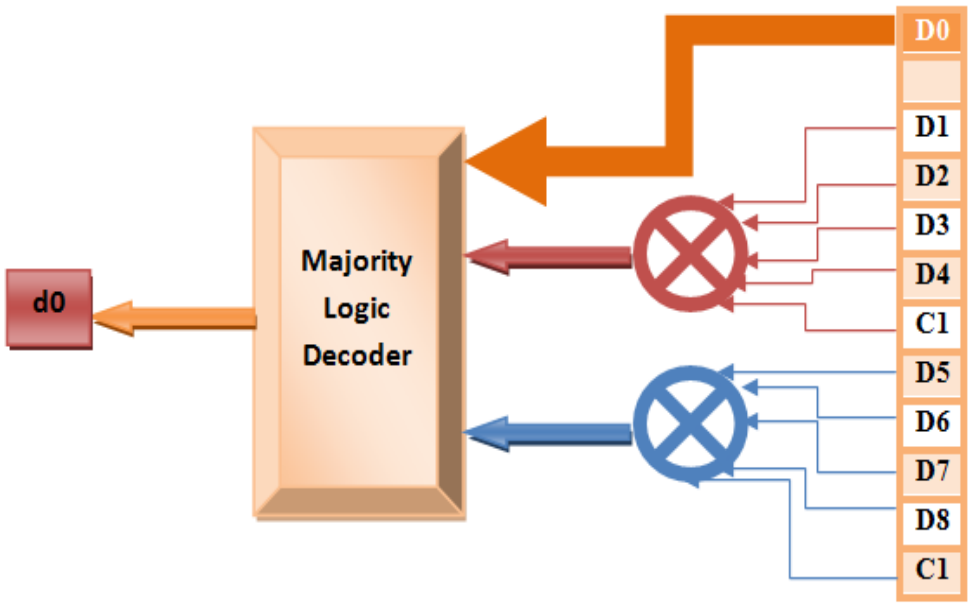


Figure 5: Decoder of detecting single error correcting code for data bit d0 using single sectional approach.

Algorithm: OLSMLD Algorithm

Step 1: The syndrome is computed as usual by shifting the received polynomial r(X) into the syndrome register.

Step 2: The f parity-check sums orthogonal on e1 are formed by taking proper sums of the syndrome digits. These J check-sums are fed into a f -input majority-logic gate.

Step 3: The first received digit is read out of the buffer register and is corrected by the output of the majority gate. At the same time the syndrome register is also shifted once (with gate 2 on), and the effect e1 on the syndrome is removed (with gate 3 on). The new contents in the syndrome register form the syndrome of the altered received vector cyclically shifted one place to the right.

Step 4: The new syndrome formed in step 3 is used to decode the next received digit r2. The decoder repeats steps 2 and 3. The received digit r2 is corrected in exactly the same manner as the first received digit r1 was corrected.

Step 5: The decoder decodes the received vector r2 digit by digit in the same manner until a total of n shifts of the buffer and the syndrome registers.

d'0 = D1 xor D2 xor D3 xor D4 xor C1

d'1 = D1 xor D2 xor D3 xor D4 xor C1

Results

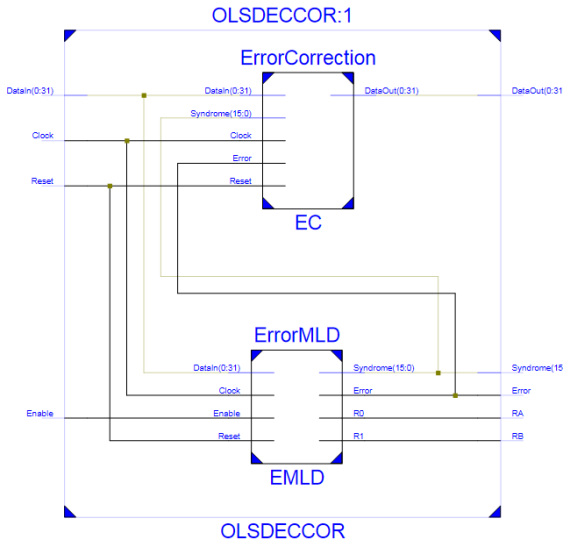
Through syndrome test error detection is performed and corrected by using MLD approach. Three sectional approaches are following for correcting the data bits.Here Enc(Encoder) and Dec(Decoder) data.

Table 4: Delay estimation (In nano seconds)

	PROPOSED			SEC-DED-DAEC			
K	n-K	Enc	Dec	K	n-K	Enc	Dec
24	16	0.065	0.068	16	12	0.22	0.47
32	16	0.065	0.078	-	-	-	-

The area and delay results only for the encoders and decoders are presented in Table 4. The decoders for the proposed codes are simpler and faster than those of existing codes. The results show the delay is also greatly reduced (k = 24 and k = 32, respectively).

RTL Schematic



Device Utilization Summary:

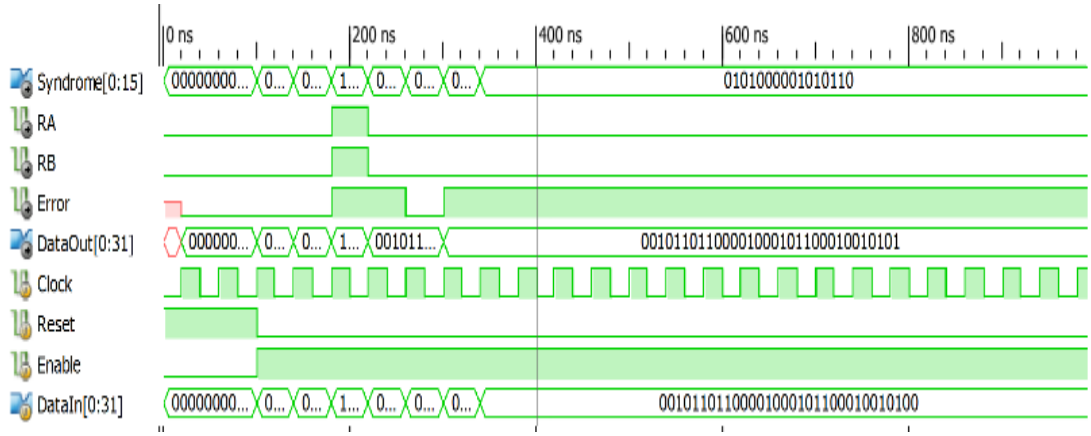
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	56	4656	1%
Number of Slice FlipFlops	33	9312	0%
Number of 4input LUTs	101	9312	1%
Number of bonded IOBS	86	232	37%
Number of GCLKs	1	24	4%

Cell Usage:

LUT3	99
LUT4	64
MUXF5	7

VCC	1
FlipFlops/Latches	32
FDRE	32
Clock Buffers	1
BUFGP	1
IO Buffers	84
IBUF	33
OBUF	51

OLS Decoder Simulation Results



Conclusion

To increase the speed of error identification and reduce the utilisation of clock cycles in identifying error OLSMLD scheme performs with a best under two different tests i.e., M(24, 16) and M(32, 16). This approach is low complex in mathematical computation and speed in computation this reduces the power utilisation. An efficient scheme for error identification and error correction is proposed with this architecture for multiple bits.

References

[1] Reviriego, P, Pontarelli, S, Sanchez-Macian, A, Maestro, JA,"A method to extend orthogonal latin square codes",Very Large scale Integration (VLSI) Systems, IEEE transactions on Vol.22, issue 7. Page 1635-1639, 07 August 2013.

[2] L. Ping, W. K. Leung, and N. Phamdo, "Low Density Parity Check Codes with Semi-Random Parity Check Matrix," IEE Electronics Lett., vol. 35, Jan. 1999, pp. 38-39.

[3] R.Meenakshi Sundhari, C.Sundarrasu, M. Karthikkumar, "An efficient Majority logic fault detection to reduce the accessing time for memory application", International Journal of Scientific and Research Publications, Volume 3, Issue 3, March 2013

[4] Reviriego, Maesstro, Flanagan, "Error detection and correction by Majority logic decoder using Euclidean geometry Low Density parity check (EG-LDPC)", Very large scale integration systems, IEEE transactions on Vol. 20, Jan 2013.

[5] Sudhakar kongala, Kishore kumar, "Concurrent error detection and correction using orthogonal latin square encoders and syndrome computation", International Journal of Engineering Research & Technology (IJERT), Vol. 3 Issue 9, September- 2014.

[6] A. Dutta, "Low cost adjacent double error correcting code with complete elimination of miscorrection within a dispersion window for multiple bit upset tolerant memory," in Proc. IEEE/IFIP 20th Int. Conf. VLSI SoC, 2012, pp. 287–290.

[7] pedro reviriego, salvatore pontarelli, and juan antonio maestro,"concurrent error detection for orthogonal latin squares encoders and syndrome computation", IEEE transactions on very large scale integration (vlsi) systems, vol. 21, no. 12, december 2013.

[8] Pedro Reviriego, Salvatore Pontarelli, Juan Antonio Maestro, and Marco Ottavi, "A Method to Construct Low Delay Single Error Correction Codes for Protecting Data Bits Only", IEEE transactions on computer-aided design of integrated circuits and systems, vol. 32, no. 3, march 2013.

[9] M. Demirci, P. Reviriego, J. A. Maestro, "Implementing Double Error Correction Orthogonal Latin Squares Codes in Xilinx FPGAs", Median-2014, Dresden, Germany, March 28th, 2014.

[10] M.Y. Hsiao, D.C. Bossen, and R.T. Chien, ''Orthogonal Latin Square Codes,'' IBM Journal of Research and Development, vol. 14, no. 4, pp. 390-394, 1970.

[11] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in *Proc.* IEEE VLSI Test Symp., May 2007, pp. 349–354.

[12] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for nanoMemory applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 473–486, Apr. 2009.