Error Correction codes Derived from Orthogonal Latin Square codes

Ande Bhargav, Y.Varthamanan

Abstract— The developments in IC technology and rapid increase of transistor densities and scaling factor, the use of ECC's acquired prominence. Multiple bit errors in memories due to technology scaling demands advanced error correction codes. SEC-DEC, DEC, burst error detection, Golay code, Reed Solmon codes etc. have much decoding complexity and latency. The above drawbacks can be reduced with OLS codes. OLS codes with majority logic decoding technique, modular construction and simple decoding mechanisms it enables low delay improvements. MBU'S can be addressed using OLS-MLD codes. This paper presents a detail study of developments in multibit ECC's using OLS-MLD mechanism.

Keywords: -- component; Error correction codes; Single error correction – double error detection; orthogonal latin squares; majority logic decodable.

I. INTRODUCTION

Communication systems of deep space exploration are one of the impressive applications of ECC's. On the other hand ECC's are extensively used in storage device systems. A typical audio CD can accept one and a half million bits to represent just one second of music. At the same instant 6 billion or so bits are added to protect the recorded music form scratches, dust… etc. Due to scaling of technology multiple bit upsets came into existence as the floor planning and routing became more complex. When technology scaled down from 180nm to 22nm, the ratio of multiple bit upsets (MBU) to single event upset (SEU) increases from 0.5% to 3.9% [11]. Soft errors lead to loss of information, which ultimately effects the function of the devices, especially in avionics and space applications. Internal radiation can also cause soft errors (bit change).

Early developments like SEC's, DEC's, SEC-DED, are unable to address the multiple bit upsets due to their decoding complexity. Memories like SRAM is not exceptional form MBU's. Decoding and encoding has to be done in parallel with SRAM memories for fast access of data. Developments like Hamming code, Bose – chaudhuri – Hocquenghem (BCH) codes, and Reed Solmon codes are more complex in parallel decoding [3][14]. Due to delay overheads and high area requirements the codes are limited to smaller applications. And most of the ECC's have serial decoding circuitry. An alternative for parallel decoding limitation is one step majority logic decodable (OS-MLD) codes such as Euclidean geometry (EG) codes, Difference set (DS) codes [13]. But both the EG, DS codes have limited number of error correction capabilities and data size [8]. In addition to that, available data bits are not in the powers of two [3]. So OS-MLD is not suitable for SRAM memories as the data size is generally in the powers of two like 16, 32. This limitation has opened a new class of codes i.e. orthogonal latin square (OLS) codes. OLS codes are derived from latin squares [1]. OLS codes are also a type of OS- MLD codes with wide range of word sizes and error correction capabilities. However it requires more number of parity bits compared to other ECC's. With increase in number of error correction bits, parity bits increase rapidly. Research is being done to reduce the parity bits and a method has been proposed recently [6]. In this paper different advancements in OLS-MLD codes are present.

The rest of the paper is organized as follows. Introduction and basic implementation of OLS-MLD code is included in Section II. Recent developments of OLS-MLD codes are discussed in Section III. Section IV deals with a comparative study of different parameters. Conclusion and References are included in Section V.

II. ORTHOGONAL LATIN SQUARE CODES

Before A Latin Square is a matrix structured grid with the numbers '1 to n' occurs only once in each row and column. For example: Latin Square of order 4.

$$\begin{matrix} 2 & 3 & 1 & 4 \\ 4 & 1 & 3 & 2 \\ 1 & 4 & 2 & 3 \\ 3 & 2 & 4 & 1 \end{matrix}$$

Two Latin Squares are said to be mutually orthogonal when two squares are superimposed, every order of co-ordinates should occur exactly once. The concept of mutually orthogonal latin squares (MOLS) can extend to a set of more than two squares, provided that every square must be mutually orthogonal to every other one.

$$\begin{matrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{matrix} \quad \text{and} \quad \begin{matrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{matrix}$$

Revised Version Manuscript Received on August 19, 2019.

Ande Bhargav, Reseach Scholar, Dept. of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, India

Dr.Y.Varthamanan, Assoc.Prof, Dept. of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, India

The superimposing of two latin squares is shown below:

Mutually OLS condition:	11	22	33
	23	31	12
	32	13	21

OLS codes are constructed using these mutually orthogonal latin squares. Primary considerations for OLS codes are as follows

Order of the matrix = m
Number of error correcting bits = t
Number of data bits that a code can protect are $k = m^2$
Number of check bits / parity bits = $2tm$

In encoding technique each bit involves $2t$ parity bit computations and each other bit participates in at most one of those parity bits. This creates a simple correction when the number of bits in error is t or less. Decoding starts with recomputing the parity bits. The recomputed check parity bits are fed to one step majority logic and a majority vote is taken. If the majority value is one, it results in bit error and correction is needed else the decoded bit is true. If $t + 1$ bits need to be corrected then simply $2m$ parity bits are added to the code.

Modeling a parity check matrix H:

Parity check matrix is modeled depends on error correction capabilities of the code. For single error correction (SEC) using OLSC. Consider an H- matrix for $m = 4; k = 16$ data bits, t=1 error correcting capability, $2tm = 8$ parity / check bits.

$$H = \begin{bmatrix} M_1 & \\ & I_{2tm} \\ M_2 & \end{bmatrix}$$

Where I_{2m} is identity matrix of size $2m$. M_1, M_2 are matrices of size $m \times m^2$. The matrix M_1 has m ones in each row. For the rth row the one are at positions:

$((r-1) * m + 1, (r-1) * m+2, \dots (r-1) * m+m-1, (r-1) * m + m$

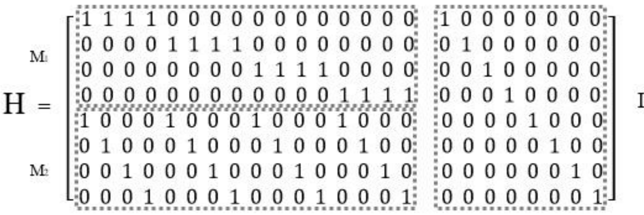
The matrix $M2$ is constructed as shown below:

$M_2 = [\ I_m \quad I_m \quad I_m \quad \dots I_m\ ]$

The encoding matrix G is just the H matrix on which the check bits are removed.

$$G = \begin{bmatrix} M_1 \\ \\ M_2 \end{bmatrix}$$

In brief, the encoder with k=m2 data bits will generate 2tm check bits using generator matrix G, derived from OLS's.

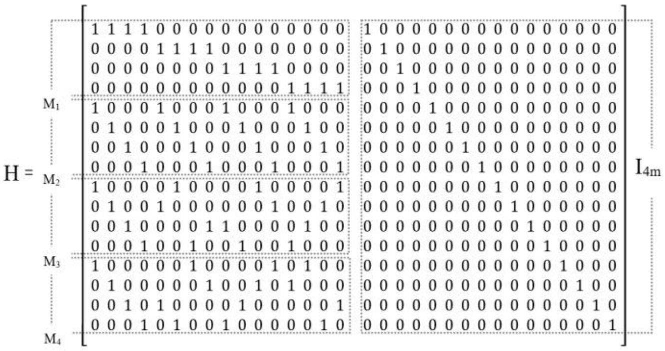


$$\begin{array}{cc} & \\ M_1 & \\ H = & \\ & \\ & \\ & \\ M_2 & \\ & \end{array} \left[\begin{array}{cccccccccccccccc|cccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&1&0&0&0&0\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1
\end{array}\right] I_{2tm}$$

Fig. 1. Parity check matrix for OLSC (24, 16) having k=16 and t=1

Generation of parity bits:

Each row of the matrix M should have only m number of one's. As shown in the Fig. 1. OLS (24, 16) parity bits are calculated by performing an XOR operation of data bits present in each row of generator matrix. P_0 is calculated by XOR of first 4 data bits (d_0, d_1, d_2, d_3) of row one in M_1. P_1 is calculated by XOR operation between d_4, d_5, d_6, d_7 of 2nd row in M_1. Likewise all other parity bits are obtained with XOR operation of data bits in respective rows of parity check matrix. Total 8 parity bits will be obtained as $2tm = 8$ in OLS (24, 16) code. In the decoding process: each group of data bits involved in encoding mechanism will be verified against their respective parity bits to generate syndrome bits S_i. S_0 is an XOR operation of data bits along with its parity bit d_0, d_1, d_2, $d_{3,}$ P_0. Thus obtained syndrome bits are fed to majority logic decodable gate for error detection and correction.



$$\begin{array}{c} \\ \\ \\ \\ M_1 \\ \\ \\ H = \quad M_2 \\ \\ \\ \\ M_3 \\ \\ \\ \\ M_4 \end{array} \left[\begin{array}{cccccccccccccccc|cccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
\hline
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
\hline
1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{array}\right] I_{4m}$$

Fig. 2. Parity check matrix for OLSC (32, 16) having *k=16* and *t=2*

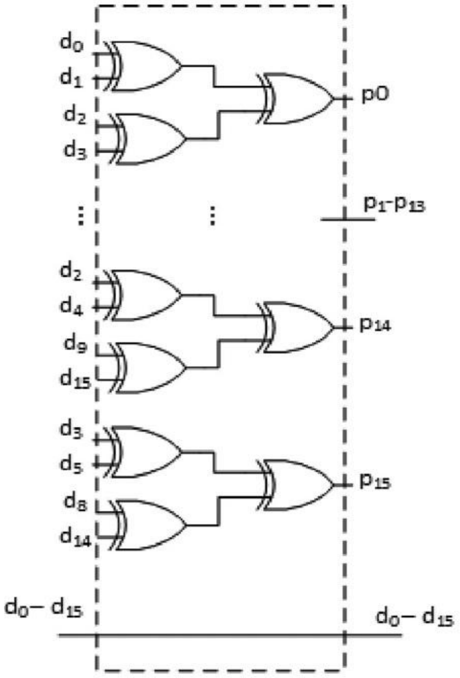


Fig. 3. Encoder for OLS code *k=16* and *t=2*

The above figure shows that three XOR gates are required to generate one parity bit. Encoder of OLS (32, 16) requires 48 two- input XOR gates [2]. As number of error correction bits increases, parity bits increases, that results in increase in XOR gates. Developments are going on to reduce the number of parity bits and logic gates which will be discussed later. The basic decoder along with correction logic of OLS code computes the syndrome Si as shown in Fig.4.

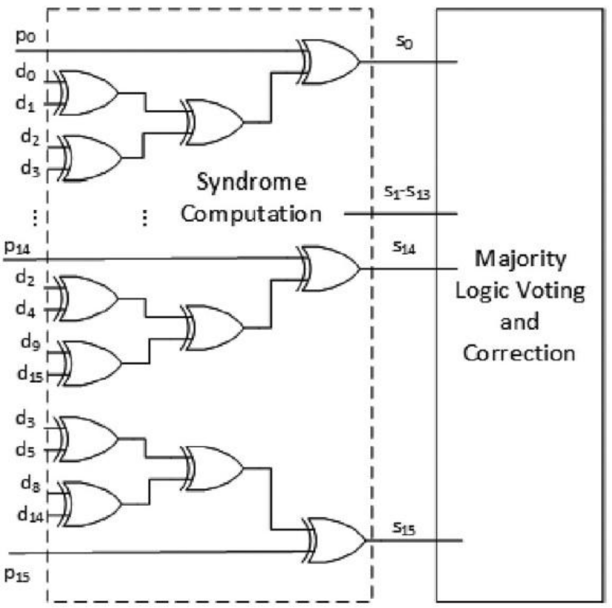


Fig. 4. Decoder for OLS code k=16 and t=2

Syndrome bits thus calculated are connected to majority logic and correction module. To detect / correct a data bit di, syndrome bits having the same di as one of its input, are given to the OS-MLD gate [1] [6]. Inputs of OS-MLD correction logic for data bit d0 are shown in Fig.5. Encoder requires 2tm(m-1) two input XOR gates and syndrome computation requires 2tm2 two input XOR gates [9].

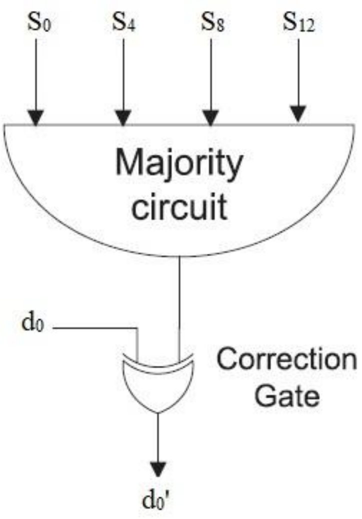


Fig. 5. OLS-OS-MLD gate with correction logic

OLS codes can be implemented for different values of m, t, and a few are listed in the Table-1. It is clear that with increase in correctable errors, parity bits needed to encode the data also increase in the same phase. With increase in parity bits, parameters like area, power and delay may get effected.

TABLE 1. Orthogonal Latin square codes

Latin square size (m)	No. of errors (t)	Data bits (m^2)	Parity bits (2tm)	Total bits (m^2 + 2tm)	No. of XOR gates (encoder)
3	2	9	12	21	36
4	1	16	8	24	24
4	2	16	16	32	48
5	2	25	20	45	60
8	2	64	32	96	96
16	2	256	64	320	960

The main drawback of OLS codes is, number of parity bits increase in increase with data bits. To generate these parity bits more number of XOR gates are required which effects the area and delay of the encoder and decoder modules. Having this in view more research is being done to reduce the number of parity bits and enhancing other parameters. In the next section recent advancements in OLS codes discussed.

$$H = \left[\begin{array}{cccccccccccccccc|cccc|cccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{array}\right]$$

Fig. 6. Parity check matrix for OLSC (36, 20) for *m=4* and *t=2*

III. ADVANCEMENTS IN OLS CODES

An extended method in OLS codes have been proposed to protect the large number of data blocks without changing parity bits. For example, OLS code with *m=4, t=2* will have we have *k=16* data bits. For same *m=4, t=2, k=20* data bits have been proposed in [6]. In both the cases parity bits *2tm=16* are same, but the number of data bits increased by 4. Extended parity check matrix with same of parity bits is shown in Fig. 6. Design of encoder and decoder will be same but the input of encoder will increase by one number.

TABLE.2. Extended DEC-OLS Parameters

K_{OLS}	$K_{Extended}$	n-k	m
16	20	16	4
64	72	32	8
256	336	64	16

OLS codes are being studied to protect memories against MCUs. For 32 bit data it is not possible to implement OLS codes as data bits are defined by $k=m^2$ and 32 is not a perfect square. In [1] it was proposed to reduce the (77, 49) OLS code to (60,32) OLS code. And extended (55, 32) OLS code was proposed for 32- data bits in [7]. Extended OLS codes for 32-bit data block (54, 32), (53, 32) are discussed and decoder was modified accordingly [16].

Reduction of parity bits is the prominent problem to work with OLS codes. A new scheme was developed in this aspect [2]. A self – checking encoder and syndrome generator was designed in [9]. In this method two additional bits are generated with XOR of selected parity bits at encoder and the same was verified at syndrome computation of the receiver for *S* and *t=1*. Parity prediction in this scheme was proposed as

Retrieval Number: B12050882S819/2019©BEIESP
DOI:10.35940/ijrte.B1205.0882S819

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

$r_1 = s_1 \oplus s_2 \oplus s_3 \oplus \ldots \oplus s_{2tm}$

$r_2 = c_1 \oplus c_2 \oplus c_3 \oplus \ldots \oplus c_{2tm}$

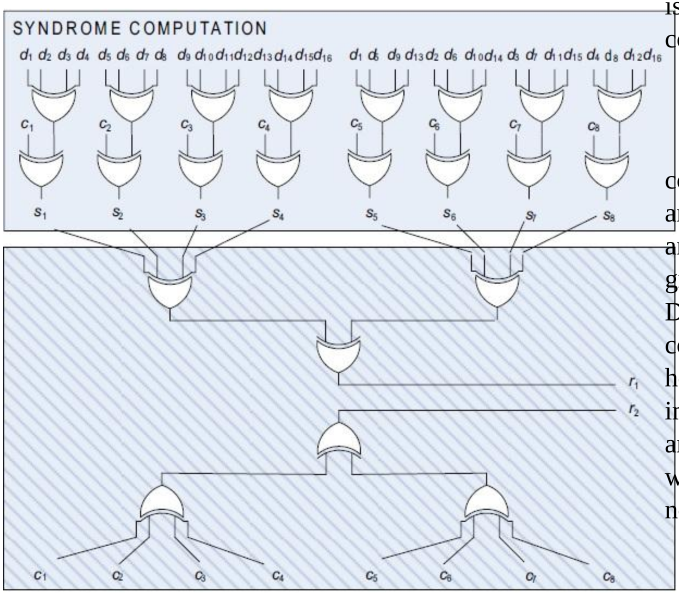

SYNDROME COMPUTATION
d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16
d1 d5 d9 d13 d2 d6 d10 d14 d3 d7 d11 d15 d4 d8 d12 d16
c1 c2 c3 c4 c5 c6 c7 c8
s1 s2 s3 s4 s5 s6 s7 s8
r1
r2
c1 c2 c3 c4 c5 c6 c7 c8

Fig. 7. Syndrome computation for OLSC (24, 16) for m=4 and t=1

$\{r_1, r_2\}$ will take the values 11 or 00 for valid inputs and 10 or 01 for an erroneous input at the checker. *2tm-1* and *4tm-2* two input XOR gates are required for encoder and syndrome computation respectively. Therefore this implementation suffers with overhead due to increased number of XOR gates compared to [1][2][6].

At encoder $O_{enc} = \frac{(2tm-1)}{(2tm\,(m-1))}$

Syndrome computation $O_{syn} = \frac{(4tm-2)}{(2tm^2)}$

Overhead of encoder and syndrome for (24,16) OLS code is 29.17% and 43.75 respectively, which are less when compared other error correction codes [9].

IV. COMPARATIVE STUDY OF ECC'S & RESULTS

This section illustrates the synthesis reports of error correction codes. Parameters such as Area, Delay, and Power are compared for recent developments in ECCs. In Encoder and Decoder designs of OLS codes delay was optimized at a great level when compared to Area and power. Area and Delay were optimized in OLS codes when compared to conventional ECCS. 16 and 32 bit data bits were compared here because the conventional codes are not flexible to implement for larger data block sizes. Encoders and Decoders are included as a part of memory design to correct the data while reading the information from it because memories are not an exception from error.

TABLE.3. Parameters comparison of Conventional Codes

Conventional Error Correcting Codes			**Area (μm^2)**	**Delay (ns)**
16 - bit	SEC_DEC_ BCH	[4]	4257.2	3.38
	DEC_ BCH		13606	5.30
32 - bit	SEC_DEC_ BCH		7975.2	3.54
	DEC_ BCH		57954.4	6.14

TABLE.4. Parameters comparison of synthesized OLS encoders and decoders (16-bit)

Advancements of 16-bit OLS-MLD codes		**Area (μm^2)**		**Delay (ns)**		**Power (mW)**	
		Encoder	Decoder	Encoder	Decoder	Encoder	Decoder
OLS (32,16) Area optimized	[2]	-	775	-	1.82	-	1.48
Delay optimized encoder		700	-	0.28	-	0.34	-
Delay optimized decoder		-	1738	-	0.71	-	2.08
OLS codes	[6]	299	921	0.17	0.36	-	-
DEC OLS Extended codes (20 bits)		382	1097	0.20	0.38	-	-
OS-MLD	[8]	-	2852.04	-	0.36	-	-
Modified OS-MLD		-	3122.99	-	0.51	-	-
Standard OLS DEC (32,16)	[10]	-	1198	-	0.35	-	-
Proposed OLS-SEFI plus SEU (32,16)		-	3649	-	0.57	-	-
Unprotected OLS codes	[9]	184.6	237.7	1.26	1.32	-	-
With CED		236.2	344.1	1.65	1.75	-	-

TABLE.5. Parameters comparison of synthesized OLS encoders and decoders (32-bit)

Advancements of 32-bit OLS-MLD codes			**Area (μm^2)**		**Delay (ns)**		**Power (mW)**	
			Encoder	Decoder	Encoder	Decoder	Encoder	Decoder
For (t=2)	OLS (60,32)	[7]	970.8	4408.8	0.34	0.77	0.56	6.16
	Proposed (55,32)		1016.4	3884.0	0.34	0.77	0.60	5.71
For (t=3)	OLS (76, 32)		1417.6	6674.0	0.36	0.77	0.81	8.94
	Proposed (68, 32)		1445.6	6388.8	0.36	0.77	0.85	8.94

CONCLUSION

This paper has presented recent advancements of ECCs. Today technology has scaled below 20nm and the reduced spacing in the logic circuit causes soft errors due to internal radiation. Memories (SRAM) are not an exception from these soft errors. Design of powerful ECCs is the only solution for this problem. Multiple bit upsets (MBUs) are gradually increasing due to scaling of the technology. Earlier codes like Hamming code, BCH code, Golay code, Reed Solmon codes couldn't address recent problems due to their decoding complexity and limited data block size. Triple adjacent errors, single byte errors… etc. MBUs have increased gradually. This created a necessity to develop more efficient ECCs. . In this scenario, OLS codes created a wide range of opportunity to develop error correction codes for MBUs.

OLS codes with their modular construction and Expandable data block size contributed much in development of effective ECCs. Recent developments of OLS-MLD codes are described in this letter. At the cost of parity bits OLSC reduces the delay in decoding. Advancements has shown the possibility to control the parity bits. By reducing the parity bits and with flexibility of data block size OLS codes are efficient to design effective error correcting codes.

REFERENCES

1. M.Y. Hsiao, D.C. Bossen, R.T. Chien, Orthogonal Latin square codes, IBM J. Res. Dev., Vol.14, pp.390–394, July 1970.
2. Pedro Reviriego, Shanshan Liu, Alfonso Sanchez-Macian, Liyi Xiao, Juan Antonio Maestro, 'A scheme to Reduce the Number of Parity Check Bits in Orthogonal Latin Square Codes', IEEE Transactions on Reliability, Vol.66, pp. 518-528, June 2017
3. Mustafa Demirci, Pedro Reviriego, Juan Antonio Maestro, 'Unequal Error Protection Codes Derived from Double Error Correction Orthogonal Latin Square Codes', IEEE Transactions on Computers, Vol.65, pp.2932-2938, Iss.9, DOI 10.1109/TC.2015.2498547, sep.2016.
4. Shanshan Liu, Pedro Reviriego, Liyi Xiao, 'Evaluating Direct Compare for Double Error-Correction Codes', IEEE Transactions on Device and Materials Reliability, Vol.17, Iss.4, pp.802-804, Dec.2017.
5. Shanshan Liu, Jiaqiang Li, Pedro Reviriego, Macro Ottavi, Liyi Xiao, 'A Double Error Correction Code for 32-bit Data Words with Efficent Decoding', IEEE Transaction on Device and Materials Reliability, Vol.18, Iss.1, pp.125-127, Jan.2018.
6. Pedro Reviriego, Salvatore Pontarelli, Alfonso Sanchez-Macian, Juan Antonio Maestro, 'A Method to Extend Orthogonal Latin Square Codes', IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol.22, No.7 July 2014
7. Shanshan Liu, Liyi Xiaohigang Mao, 'Extend Orthogonal Latin Square Codes for 32-bit data Protection in Memory Applications, Microelectronics Reliability, Elsevier, 2016.
8. Pedro Reviriego, S.Liu, J.A.Maestro, S.Lee, N.A Touba, R.Datta, 'Implementing Triple Adjacent Error Correction in Double Error Correction Orthogonal Latin Squares Codes' IEEE International Symposiumon Defect and Fault Tolerance in VLSI and Nanotechnolgy Systems (DFTS)
9. Pedro Reviriego, Salvatore Pontarelli, Juan Antonio Maestro, 'Concurrent Error Detection for Orthogonal Latin Squares Encoders and Syndrome Computation', IEEE Transaction on very large scale integration (VLSI) systems, Vol.21, No.12, pp.2334-2338, Dec.2013.
10. Alfonso Sanchez-Macian, Pedro Reviriego, Juan Antonio Maestro, ' Combined SECS and SEFI Protection for Memories using Orthogonal Latin Square Codes' IEEE Transactions on circuits and systems -1, Vol.63, No.11, pp.1933-1943,Nov.2016.
11. I be E, Taniguchi H, Yahagi Y, et al. Impact of Scaling on Neutron-Induced Soft Error in SRAMs From a 250nm to a 22 nm Design Rule [J]. IEEE Transactions on Electron Devices, 2010, 57(7):1527-1538.
12. J. Dénes and A. D. Keedwell, "Latin Squares and their Applications", Academic Press, 1974.
13. P. Reviriego, M. Flanagan, S. Liu and J.A. Maestro, "Multiple Cell Upset Correction in Memories Using Difference Set Codes", IEEE Trans. on Circuits and Systems I, vol. 59, no 11, pp. 2592-2599, Nov. 2012.
14. S. Lin and D. J. Costello, "Error Control Coding" (2nd Ed.), Englewood Cliffs, NJ: Prentice-Hall. 2004.
15. J.E.McCluskey "Design techniques for testable embedded error checkers", IEEE computer, Vol.23, no.7, pp.84-88, Jul.1990.
16. Shanshan Liu, P. Reviriego, Alfonso Sanchez-Macian, Juan Antonio Maestro, Liyi Xiao, " Comments on - Extended orthogonal latin square codes for 32-bit data protectionin memory applications", Microelectronics Reliability, Elsevier, 2016.
17. C. L. Chen and M. Y. Hsiao, Error-correcting codes for semiconductor memory applications: A state of the art review, IBM J. Res. Develop., vol. 28, No. 2, pp. 124-134, 1984.