Layered-ECC: A Class of Double Error Correcting Codes for High Density Memory Systems

Abhishek Das and Nur A. Touba
Computer Engineering Research Center,
University of Texas at Austin, TX 78712
abhishekdas@utexas.edu, touba@ece.utexas.edu

Abstract - As memory technology scales, the demand for higher performance and reliable operation is increasing as well. For main memory, e.g., DRAM, a conventional single error correcting double error detecting (SEC-DED) code may not be sufficient. However, existing double error correcting (DEC) codes either have very high decoder latency or high data redundancy. For flash-based memories, e.g., NAND flash, using a highly complex decoding scheme with a large number of clock cycles for the whole procedure creates a performance bottleneck. In this paper, a layered DEC code is proposed with a simple decoding procedure. The codes are shown to strike a good balance between redundancy and decoder complexity. A general construction methodology is presented. Two different decoding schemes can be implemented using the proposed methodology. One is a low latency decoding scheme that is useful for main memories which need high speed decoding for optimal performance. This scheme is shown to achieve better redundancy compared to existing low-latency codes as well as faster decoder latency compared to existing low-redundancy codes. The second decoding scheme is a low complexity decoding scheme which is useful for flash-based memories. This scheme is shown to have considerably less area compared to existing schemes. Also, it is shown that the proposed serial low complexity decoding scheme can take significantly fewer cycles to complete the whole decoding procedure; thus, enabling better performance compared to existing serial decoding schemes.

Keywords*—double error correction, low latency, low complexity, memory*

1. INTRODUCTION

Soft errors are a major reliability concern for high density memories. Soft errors can arise due to numerous mechanisms, e.g., particle strikes, marginal cells, etc. Error correcting codes (ECCs) can be used to tolerate such soft errors and overcome data corruption by adding parity or check bits to each word. These bits are then evaluated after the read process to detect and/or correct errors.

DRAM scaling has been a challenge in recent years. The gap in performance between main memory and processors has triggered research into alternative forms of main memory, specifically emerging non-volatile main memories. Memories with better scalability and low power have been proposed, e.g., phase change memories [Raoux 08] and resistive RAMs [Kawahara 12]. But a true replacement for DRAM is yet to be seen. Meanwhile, industry has continued the scaling of DRAM to nanoscale technology nodes. Some of the current generation of DRAMs are being manufactured as a 10 nm class of memories. These pose new kinds of challenges for manufacturing due to the smaller technology node. For DRAMs, single error correcting double error detecting (SEC-DED) codes have traditionally been sufficient to protect against soft errors. But field studies of more recent DRAM systems [Meza 15] have observed an increasing failure rate with increasing DRAM chip density which necessitates the use of stronger ECCs.

In terms of flash memories, NAND flash has been very successful due to its scalability and low cost per bit. Research has led to the development and manufacturing of multilevel cell (MLC) NAND flash which stores multiple bits per cell [Lee 11]. Soft errors in NAND flash memories are addressed using double error correcting (DEC) Bose-Chaudhuri-Hocquenghem (BCH) codes. These DEC BCH codes have complex decoding logic which takes a high number of clock cycles to decode [Chien 64]. It is possible to parallelize the decoding logic, but that incurs a significant area overhead. The high complexity is mainly due to Galois Field (GF) operations specifically for larger bits per symbol.

In this paper, a layered double error correcting scheme is proposed. These codes are constructed using two layers of parity bits. The first layer of parity bits is used to prune down possible error locations through analysis of the computed syndrome of the received word. The second layer of parity bits is used to compute syndrome bits that are to be matched with the pruned down error locations and get the final bits that are in error. The proposed schemes are shown to have a good tradeoff between data redundancy and decoder complexity or latency. Two different decoding procedures are proposed which either reduce the decoder complexity or the decoder latency. Thus, these codes can be used for various class of memories. The rest of the paper is organized as follows. Section 2 describes the existing schemes. Section 3 describes the proposed scheme and the two types of decoding schemes. Section 4 evaluates the two decoding schemes against the existing schemes. Section 5 provides a conclusion of this work.

2. RELATED WORK

An SEC-DED Hamming code [Hamming 50] has been traditionally used and is still in use for protecting memories. These codes can correct a single error and rely on syndrome matching based decoding i.e. the computed syndrome is directly matched to a particular column of the parity check matrix and the corresponding bit is flipped. These codes also detect all double errors but cannot correct them.

For double error correction, a binary BCH code is more prevalent. These codes have low data redundancy but have high decoding complexity. Generally, a BCH code has a serial decoder involving a Chien search algorithm. This algorithm iterates n times where n is the total number of bits in the codeword to detect and correct a certain number of errors. Over the years, numerous different approaches have been proposed to reduce the time taken for decoding. Parallel Chien search has been proposed to perform p GF multiplications in parallel [Chang 02]. This reduces the decoding time to n/p cycles where p is the degree of parallelization. A p-parallel Chien search algorithm is shown in Fig. 1. Low power architectures for a parallel Chien search has also been proposed in [Yoo 16] using a two-step approach, which reduces power by reducing access to the second step.

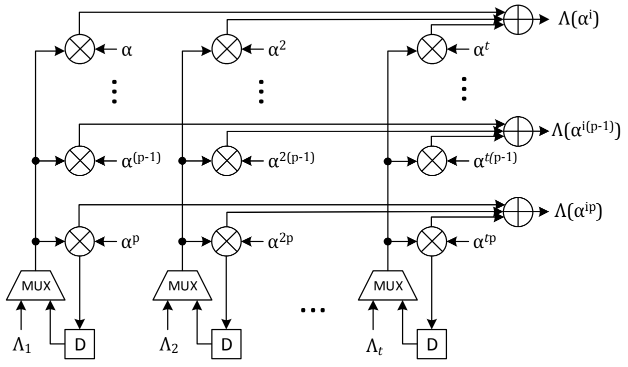

Fig. 1. *p*-parallel Chien search architecture with short critical path

For DEC BCH codes, approaches which store possible double error syndromes in a read-only memory (ROM) and evaluate the computed syndrome against the syndromes in the ROM were proposed in [Lu 96]. [Naseer 08] proposed a direct decoding method through syndrome matching for smaller data bit sizes. [Yoo 14] proposed a search-less DEC BCH decoder which utilized look-up table (LUT) based computations to replace the Chien search. But for more than a single error, all these decoding schemes either take multiple cycles to decode using a serial decoding architecture or involve a significant decoding latency and decoder area for parallel architectures.

Another class of codes that are suitable for random-access memories is the majority logic decodable codes. Orthogonal Latin Square (OLS) codes are one of the best examples for these types of codes [Hsiao 70]. These codes are modular in design and have the basic parity check matrix structure as shown in equation (1). The submatrices $\{M_1, M_2, \dots M_{2t}\}$ can be constructed from mutually orthogonal Latin squares. The basic idea of the majority logic decoding scheme is that in the presence of t errors, each data bit can be reconstructed from $2t$ independent sources excluding the data bit itself. Thus, there are $(2t + 1)$ independent sources for each data bit and in the presence of t errors, $(t + 1)$ of them will be uncorrupted. Thus, a majority vote will always be able to correct any t errors. The disadvantage of these codes lies in its data redundancy required to construct the $2t$ independent sources. But the decoding scheme itself is very simple and has very low decoding latency. The DEC OLS codes have recently been used in SRAM based FPGAs [Reviriego 16]. A second class of majority logic decoding using difference-set codes was proposed in [Reviriego 12]. But for double error correction these codes only support a single block size (data block size of 11, codeword size 21) and cannot be extended to include different sizes. [Liu 18] recently used difference set codes to correct data block sizes of 32 with some additional decoding complexity. But with only two data block sizes, the application of such codes is limited.

$$H = \left(\begin{array}{c|c} M_1 & \\ M_2 & \\ M_3 & I_{2tm} \\ \vdots & \\ M_{2t} & \end{array}\right) \tag{1}$$

Multiple cell upsets (MCUs) can also occur in DRAMs wherein adjacent bits are affected by a single particle strike. MCUs are generally addressed by using word interleaving such that any MCU will at most cause a single error in any word. [Das 18] also addresses this issue by modifying Hamming codes to correct MCUs in SRAMs, which can be extended to DRAMs as well. The focus of this work is on double random errors only and not on MCUs.

A new class of DEC codes are proposed which are shown to have better data redundancy at the expense of higher decoding complexity and higher decoding latency compared to OLS codes. The proposed codes are also shown to have better decoding latency and better decoding complexity, depending on the type of decoding logic, compared to DEC BCH codes. But these benefits come at the cost of additional data redundancy compared to DEC BCH codes.

3. Proposed Scheme

The proposed scheme is made up of two layers of ECC. The first layer is based on a single error correcting OLS code which prunes down possible error location candidates. The second layer is constructed by adding columns to the parity check matrix such that the below mentioned conditions are satisfied.

1. All columns added are unique and are not repeated.
2. The sum of any two columns of the complete parity check matrix should not be equal to the sum of any other two columns of the parity check matrix.

The first condition ensures that double errors produce a syndrome which is non-zero. The second condition basically ensures that the pruned list of possible error locations do not produce the same syndrome thus avoiding the chance of mis-correction.

The first layer of the parity check matrix is created using m groups of m data bits each, where $m = \sqrt{k}$. The rest of the parity check matrix is created in an algorithmic manner satisfying the two conditions mentioned above. An example of the parity check matrix and the corresponding syndrome bits for a data bit size of $k = 16$ is shown in Fig. 2. The syndrome computed from the parity check matrix in this case is also divided into layers. The upper syndrome layer has $2m$ bits (S_0 to S_7 in Fig. 2) which help in selecting possible double error candidates. The lower syndrome layer (S_8 to S_{12} in Fig. 2) simply matches the possible error candidates to the received syndrome to find out the actual bits in error.

$$
\left(\begin{array}{ccccccccccccccccccccccccccccc}
d_0&d_1&d_2&d_3&d_4&d_5&d_6&d_7&d_8&d_9&d_{10}&d_{11}&d_{12}&d_{13}&d_{14}&d_{15}&p_0&p_1&p_2&p_3&p_4&p_5&p_6&p_7&p_8&p_9&p_{10}&p_{11}&p_{12} \\
1&1&1&1& &&&& &&&& &&&& 1 &&&& &&&& &&&& \\
&&&& 1&1&1&1& &&&& &&&& & 1 &&&& &&&& &&& \\
&&&& &&&& 1&1&1&1& &&&& && 1 &&&& &&&& && \\
&&&& &&&& &&&& 1&1&1&1& &&& 1 &&&& &&&& & \\
1&&&& 1&&&& 1&&&& 1&&&& &&&& 1 &&&& &&&& \\
&1&&& &1&&& &1&&& &1&&& &&&& & 1 &&&& &&& \\
&&1&& &&1&& &&1&& &&1&& &&&& && 1 &&&& && \\
&&&1& &&&1& &&&1& &&&1& &&&& &&& 1 &&&& & \\
1&1&&1& &1&&& &&1&1& 1&&1&1& &&&& &&&& 1 &&&& \\
1&&1&& 1&&&1& &&1&1& &1&&1& &&&& &&&& & 1 &&& \\
&1&1&& &&1&& 1&&1&& 1&1&1&& &&&& &&&& && 1 && \\
&&&1& 1&&1&& &1&&1& 1&1&&& &&&& &&&& &&& 1 & \\
&&&& &1&&1& 1&1&&& &&1&1& &&&& &&&& &&&& 1
\end{array}\right)
\rightarrow
\begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \\ S_9 \\ S_{10} \\ S_{11} \\ S_{12} \end{pmatrix}
$$

Fig. 2. Parity check matrix of proposed scheme for k=16

Consider the simple example from Fig. 2 where bits d_0 and d_7 are in error. The corresponding syndrome bits for this case is shown in equation (2). Considering the syndrome bits S_0 through S_3, $S_0 = 1$ suggests that one of the bits in amongst d_0, d_1, d_2 *and* d_3 is in error. Similarly, $S_1 = 1$ suggests that one of the bits amongst d_4, d_5, d_6 and d_7 is also in error. Now, considering the syndrome bits S_4 through S_7, $S_4 = 1$ narrows down the possibility to either d_0 or d_4. Similarly, $S_7 = 1$ narrows down the second error's possibility to d_3 or d_7. Thus, from the upper layer of syndrome bits, we narrowed the set of suspect location pairs to $\{d_0, d_7\}$ and $\{d_3, d_4\}$. We can now simply compute the XOR of both pairs and match it against the second layer of syndromes. It can be easily verified that $d_0 \oplus d_7 = S_{8:12}$, which means that bits d_0 and d_7 have flipped.

$$
\begin{pmatrix} S_0 & S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 & S_{10} & S_{11} & S_{12} \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}
$$

The encoding procedure of the proposed codes is a one-step low latency procedure. Each parity bit can be computed in parallel by XORing all the data bits which have corresponding 1's in the row of the parity check matrix. The codeword can be formed by appending the parity bits to the received data bits and the codeword can then be stored in memory. For the decoding procedure, there are 3 major cases that need to be considered. These include the cases of no error, a single error, and a double error in the word. These three cases can be deciphered using a combination of the upper layer of 2m syndrome bits as given by the below mentioned equations (3-6).

$$G_{OR1} = S_0 | S_1 | S_2 \dots S_{m-2} | S_{m-1} \tag{3}$$

$$G_{OR2} = S_m | S_{m+1} | S_{m+2} \dots S_{2m-2} | S_{2m-1} \tag{4}$$

$$G_{XOR1} = S_0 \oplus S_1 \oplus S_2 \dots S_{m-2} \oplus S_{m-1} \tag{5}$$

$$G_{XOR2} = S_m \oplus S_{m+1} \oplus S_{m+2} \dots S_{2m-2} \oplus S_{2m-1} \tag{6}$$

For single errors, since the upper 2m rows in the parity check matrix can also be used as a single error correcting OLS code, a simple majority voting logic can be used to correct single errors. For double errors, we propose a double error pattern generator which gets triggered for double error cases. The block diagram of the proposed decoding logic is shown in Fig. 3. The different cases for the decoding logic are as follows:

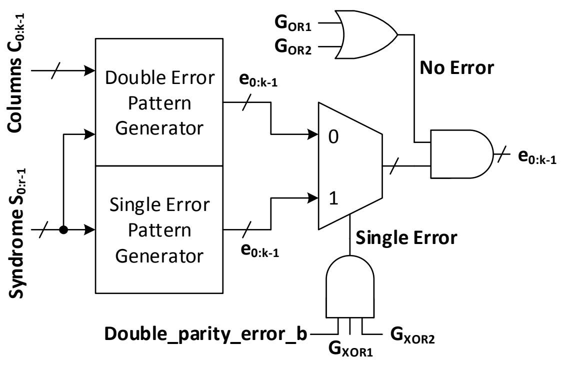

Fig. 3. Block diagram of proposed decoding logic

1. $G_{OR1} = G_{OR2} = 0$: No error.
2. $G_{XOR1} = G_{XOR2} = 1$: This is only possible in case of a single error in one of the data bits. For this case, a simple majority voting logic can be used.
3. $G_{XOR1} = 1$; $G_{XOR2} = 0$: This is possible either in case of a single parity error or a combination of data bit error and a parity bit error. The decoding logic is the same for either case. In this case, exactly one bit in a m-bit group (e.g., d_0 to d_3 is one group in Fig. 2) is in error and each column from the group is then matched to the lower syndrome to get the correct error location. Parity bit errors are ignored as there is no requirement to correct them.
4. $G_{XOR1} = 0$; $G_{XOR2} = 1$: Similar to case-3, this is also possible for either a single parity bit error or a combination of single data bit error and a parity bit error. For this case, a specific column number (whichever of the syndrome bits S_m through S_{2m-1} is 1) of each group can create this syndrome. The column from each group is matched to the lower layer syndrome in this case.
5. $G_{XOR1} = 0$; $G_{XOR2} = 0$: This is a case of a double error with both the errors in the data bits. This involves 3 more cases which are distinguished as described below.
   a. $G_{OR1} = 1$; $G_{OR2} = 0$: This indicates that two syndrome bits among S_0 through S_{m-1} are 1 while $S_{m:2m-1} = 0$. Thus, m column pairs from each of the two groups indicated by $S_{0:m-1}$ are matched to the lower layer syndrome bits to get the correct pair that is in error.
   b. $G_{OR1} = 0$; $G_{OR2} = 1$: This indicates that two syndrome bits among S_m through S_{2m-1} are 1 while $S_{0:m-1} = 0$. Thus, column pairs indicated by $S_{m:2m-1}$ from each of the m groups are matched to the lower layer syndrome bits to get the correct pair that is in error.
   c. $G_{OR1} = 1$; $G_{OR2} = 1$: This indicates errors in distinct groups. Based on the syndrome bits $S_{0:m-1}$, the groups are narrowed down. Then, based on the syndrome bits $S_{m:2m-1}$, specific column pairs are found. Thus, there are exactly two pairs of columns that need to be compared to the lower layer syndrome bits.

An example of each of the above cases for k = 16 with the parity check matrix in Fig. 2, the corresponding syndrome bits $S_{0:2m-1}$ and the possible errors or pairs of errors has been shown in Table 1. The worst-case possibility for all the cases is comparing a combination of m pairs of syndromes to the lower layer syndrome bits. Based on the cases above, there are two types of decoding procedures that can be followed. The first is a low latency decoding scheme, which enumerates all the cases above and is based on syndrome matching for each case. The second is a low complexity decoding scheme which instead of operating on individual columns operates on the index of columns instead. This lowers the complexity of the decoding logic as well as the number of cycles needed for decoding compared to a serial decoding BCH scheme.

Apart from the above cases, 2 additional check bits are required to distinguish between a single error and a double error in parity. The two check bits basically compute the parity of each group of parities to detect errors in the parity bits. In theory, it is also possible to extend these codes to multiple bits per symbol and construct a symbol based layered ECC. This can be useful for emerging multilevel cell memories or even for double byte error correction. The details of the extension to include multiple bit symbols is beyond the scope of this paper.

A. *Low Latency Decoding*

The low latency decoding procedure involves a syndrome matching based decoding. The upper layer syndromes $S_{0:2m-1}$ are directly enumerated and depending on these syndrome values, a certain number of combination of pairs of syndromes are matched to the lower layer of syndromes. The critical path in this case depends on the selection of a particular combination of the upper layer of syndromes. The total number of relevant syndromes in a double error correcting code [Naseer 08] is given by equation (7), where n is the total number of bits in the codeword and k is the number of data bits. Comparatively, the low latency decoding procedure has fewer syndromes to be enumerated as shown in equation (8). The comparison of the total number of syndromes for different data bit sizes has been shown in Fig. 4.

$$\#syndromes = kn \qquad (7)$$

$$\#syndromes = 2m({}^{m}C_1) + 2m({}^{m}C_2) + 2({}^{m}C_2)^2 \qquad (8)$$

As the number of data bits increases, the number of syndromes goes up considerably. This causes a considerable increase in the decoding complexity. But, since the syndrome matching is done in parallel, the rise in decoding latency is much slower. This type of decoding method is suitable for mostly random-access memories which need low decoding latency and high throughput.

B. *Low Complexity Decoding*

An alternative decoding procedure is the low complexity decoding procedure. Compared to the low latency decoding, this procedure operates on the indices of data bits instead of the columns of parity check matrix. Based on the different cases described previously, the first index or pair of indices is

TABLE 1. EXAMPLE OF SYNDROME VALUES AND ERROR CANDIDATES FOR DIFFERENT ERROR TYPES

Syndrome Bits $S_{0:7}$	**Possible (pairs of) error candidates (data bits only, parity bits ignored)**
00000000	No error
00010001	Single data error (majority vote)
00000001	Possible single error in $\{d_3, d_7, d_{11}, d_{15}\}$
00100000	Possible single error in $\{d_8, d_9, d_{10}, d_{11}\}$
10000011	Possible single error in $\{d_2, d_3\}$
11000001	Possible single error in $\{d_3, d_7\}$
11000000	$\{(d_0, d_4), (d_1, d_5), (d_2, d_6), (d_3, d_7)\}$
00000110	$\{(d_1, d_2), (d_5, d_6), (d_9, d_{10}), (d_{13}, d_{14})\}$
11000011	$\{(d_2, d_7), (d_3, d_6)\}$

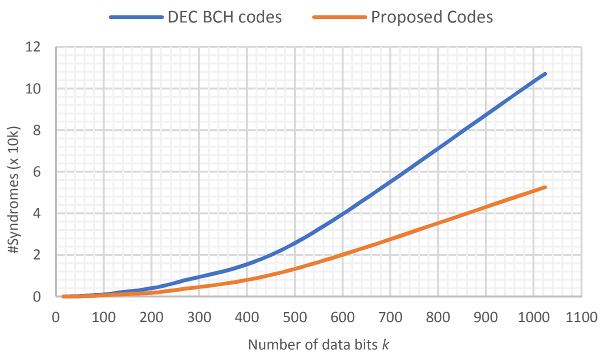

Fig. 4. Comparison of number of syndromes for different data bit sizes between proposed scheme and a DEC BCH code

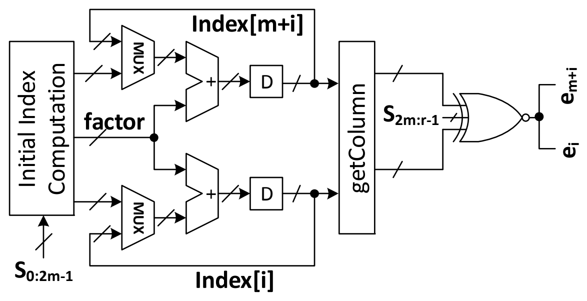

Fig. 5. Block diagram of error pattern generation using the serial low complexity decoding procedure

computed. Also, based on the upper layer of syndromes an addition factor is computed. This addition factor basically constructs all other m possible error indices by adding to the previous index or pair of indices. For the case of a double error in separate groups i.e. case-5c, the 2 possible pairs of indices are directly assigned.

The m possible error indices can be computed in m clock cycles which lowers the decoding time considerably. Once all the indices are computed, the corresponding columns are then matched with the lower layer of syndromes to get the final error location or pair of locations. Fig. 5 shows the partial block diagram of a double error pattern generator for the low complexity serial decoding procedure. Fig. 6 shows an un-rolled parallel version of the double error pattern generator in the low complexity decoding procedure. The use of arithmetic addition and ripple computation of indices causes a long critical path for the un-rolled version as can be seen in Fig. 6. This

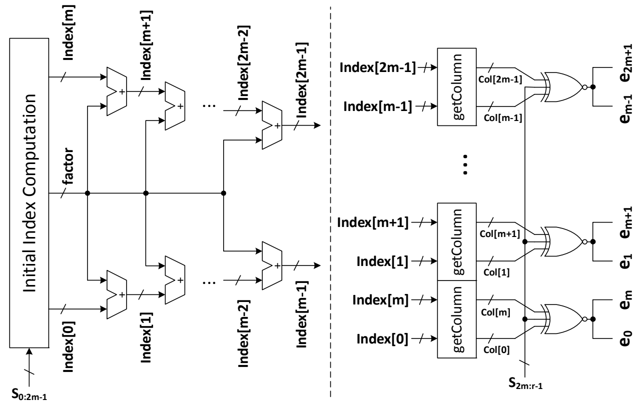

Fig. 6. Block diagram of un-rolled version of error pattern generation for

longer critical path comes at the expense of considerable reduction in decoding complexity. Thus, this type of decoding is more suitable for NAND flash based memory devices which can tolerate a higher decoding latency but can benefit from the highly reduced complexity.

4. EVALUATION

The proposed scheme was implemented using Verilog for both the low latency and low complexity decoder versions for different data bit sizes. The codes were synthesized using the NCSU FreePDK45 45nm library and Synopsys Design Compiler. OLS codes were also implemented and synthesized for different data bit sizes. A double error correcting BCH code with syndrome matching based decoding [Naseer 08] was implemented and synthesized as well. The proposed decoding schemes were compared to the existing schemes in terms of decoder complexity, decoder latency, total dynamic power consumption and data redundancy.

Table 2 shows the comparison of the low latency decoding scheme with OLS codes and the syndrome matching based DEC BCH codes [Naseer 08]. As seen in Table 2, OLS codes have the lowest decoder latency but it comes at the expense of very high data redundancy. DEC BCH codes on the other hand have very low redundancy but have considerably higher decoder latency instead. The proposed low latency decoding scheme balances between the two existing schemes. The data redundancy of the proposed scheme is much less than OLS with up to 45% reduction. Similarly, the decoding latency and dynamic power consumption of the proposed scheme is much lower than the BCH codes with up to 68% and 83% reduction respectively. The decoder area of the proposed scheme compared to the existing BCH codes is also comparatively less since it uses lesser number of syndromes as shown in Fig. 4.

A fully-parallel BCH decoder was also implemented to compare the proposed low complexity decoder. For the DEC BCH code, the error location computation was done via direct error location polynomial computation i.e. the error location polynomial coefficients were computed directly from equation (9). The Chien search algorithm was unrolled and the lower critical path version was implemented as described in [Chang 02]. The scheme in [Wilkerson 10] was also implemented and synthesized to make a comparison to the proposed low complexity serial decoder.

$$\Lambda_1 = S_1 \text{ and } \Lambda_2 = \frac{S_3 + (S_1)^3}{S_1} \tag{9}$$

Table 3 shows the comparison of the low complexity decoding scheme with the existing decoding scheme of BCH codes in [Wilkerson 10]. The additional check-bit compared to BCH codes comes from the computation of a parity bit. The decoding scheme is a one-step decoding procedure for single errors and takes multiple clock cycles for double errors. As seen in Table 3, the number of cycles taken to decode, area and power consumption for the proposed decoder is significantly lesser with up to 71%, 95% and 76% reduction respectively.

Table 4 shows the comparison between the fully-parallel version of the BCH codes [Chang 02] and the un-rolled version of the proposed low complexity decoder. As can be seen in

TABLE 2. COMPARISON OF PROPOSED LOW LATENCY DECODER WITH EXISTING SCHEMES

Data bits	OLS codes				BCH codes [Naseer 08]				Proposed Low Latency Decoder			
	Check bits	*Area (μm^2)*	*Latency (ns)*	*P_{dyn} (mW)*	*Check bits*	*Area (μm^2)*	*Latency (ns)*	*P_{dyn} (mW)*	*Check bits*	*Area (μm^2)*	*Latency (ns)*	*P_{dyn} (mW)*
16	16	647.63	0.5	0.53	10	3567.15	1.71	2.05	15	1576.38	0.98	0.77
32	28	1307.94	0.63	1.19	12	9249.43	2.17	6.15	20	5117.25	1.09	1.45
64	32	2599.92	0.78	2.98	14	27213.77	2.78	15.18	25	12205.08	1.32	2.48
128	64	5182.95	1.09	6.51	16	80312.72	3.60	23.35	34	51656.79	1.78	4.77
256	64	10280.02	1.2	15.45	18	252717.11	5.09	29.61	45	139603.61	1.60	10.39

TABLE 3. COMPARISON OF PROPOSED LOW COMPLEXITY SERIAL DECODER WITH EXISTING SCHEMES

Data bits	Serial BCH Codes [Wilkerson 10]				Proposed Low Complexity Serial Decoder			
	#Checkbits	*Area (μm^2)*	*Latency (cycles)*	*P_{dyn} (mW)*	*#Checkbits*	*Area (μm^2)*	*Latency (cycles)*	*P_{dyn} (mW)*
16	11	2341.34	27	2.59	15	1236.61	4	1.19
32	13	3800.39	45	4.50	20	2162.53	6	1.99
64	15	6141.73	79	7.14	25	3471.88	8	2.90
128	17	10800.00	145	12.48	34	6168.48	12	4.99
256	19	18927.34	275	18.42	45	5925.38	16	4.58
512	21	36757.92	533	32.79	61	10389.83	23	7.65

TABLE 4. COMPARISON OF PROPOSED UN-ROLLED LOW COMPLEXITY DECODER WITH EXISTING SCHEMES

Data bits	Fully-Parallel BCH codes [Chang 02]				Proposed Low Complexity Decoder (Unrolled Version)			
	#Checkbits	*Area (μm^2)*	*Latency (ns)*	*P_{dyn} (mW)*	*#Checkbits*	*Area (μm^2)*	*Latency (ns)*	*P_{dyn} (mW)*
16	10	12790.30	4.76	24.73	15	1546.34	2.04	1.75
32	12	29944.62	5.19	69.62	20	3514.12	3.12	4.60
64	14	74161.60	5.36	181.91	25	8150.33	4.75	10.40
128	16	197717.50	6.50	490.45	34	20920.46	6.43	23.43
256	18	406500.15	6.92	906.75	45	48519.05	8.32	41.77
512	20	955485.41	7.20	2155.30	61	111293.55	9.33	90.81

Table 4, the proposed decoding procedure is able to achieve a much lower decoder area and considerably lower dynamic power consumption compared to the fully-parallel BCH codes with up to 87% and 95% reduction respectively. This benefit comes at the expense of a slightly increased decoder latency. But this latency increase means a few more clock cycles in practice and will not cause any major impact to the performance of the memory system. Both the serial and parallel codes in Tables 3 and Table 4 also show that the proposed codes incur a redundancy overhead in order to provide the benefits of lower number of decoding cycles, lesser power consumption and lower decoder complexity.

5. CONCLUSION

In this paper a new class of layered double error correcting codes are presented along with two different decoding procedures for the codes: a low latency decoding scheme based on syndrome matching and a low complexity decoding scheme based on error location index evaluation. The schemes are compared with existing schemes and are shown to provide a good trade-off between data redundancy and decoding latency or complexity depending on the type of decoder logic. The low latency scheme can be used for high density random-access memories while flash memories can benefit from the low complexity scheme. Thus, these codes are able to provide a balanced data redundancy and decoder latency/complexity tradeoff and can be used for high density memory systems.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under Grant No. CCF-1617665.

REFERENCES

[Chang 02] H. C. Chang, C. C. Lin and C. Y. Lee, "A low power ReedSolomon decoder for STM-16 optical communications", in *Proc. of IEEE Asia-Pacific Conference on ASIC*, pp. 351-354, 2002..

[Chien 64] R. Chien, "Cyclic decoding procedures for Bose- Chaudhuri-Hocquenghem codes," in *IEEE Transactions on Information Theory*, vol. 10 , no. 4, pp. 357-363, Oct. 1964.

[Das 18] A. Das and N.A. Touba, "Low Complexity Burst Error Correcting Codes to Correct MBUs in SRAMs", in *Proc. of ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 219-224, 2018.

[Hamming 50] R. W. Hamming, "Error detecting and error correcting codes," in *Bell System Technical Journal*, vol. 26, no. 2, pp. 147-160, Apr. 1950.

[Hsiao 70] M. Y. Hsiao, D. C. Bossen, and R. T. Chien, ''Orthogonal Latin Square codes,'' in *IBM Journal of Research and Development*, vol. 14, no. 4, pp. 390–394, Jul. 1970.

[Kawahara 12] A. Kawahara, R. Azuma, Y. Ikeda, K. Kawai, Y. Katoh, K. Tanabe, T. Nakamura, Y. Sumimoto, N. Yamada, N. Nakai, S. Sakamoto, Y. Hayakawa, K. Tsuji, S. Yoneda, A. Himeno, K. Origasa, K. Shimakawa, T. Takagi, T. Mikawa and K. Aono, "An 8Mb MultiLayered Cross-Point ReRAM Macro with 443MB/s Write Throughput," in *Proc. of IEEE International Solid-State Circuits Conference*, paper 25-6, 2012.

[Lee 11] K. Lee and S. Choi, "A Highly Manufacturable Integration Technology of 20nm Generation 64Gb Multi-Level NAND Flash Memory," in Proc. of *IEEE Symposium on VLSI Technology*, pp. 70-71, 2011.

[Liu 18] S. Liu, J. Li, P. Reviriego, M. Ottavi and L. Xiao, "A Double Error Correction Code for 32-Bit Data Words With Efficent Decoding," in *IEEE Transactions on Device and Materials Reliability*, vol. 18, no. 1, pp. 125-127, Mar. 2018.

[Lu 96] E. H. Lu and T. Chang, "New decoder for double-error-correcting binary BCH codes," in *IEE Proceedings – Communications*, vol. 143, no. 3, pp. 129 – 132, Jun. 1996.

[Meza 15] J. Meza, Q. Wu, S. Kumar and O. Mutlu, "Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field," in *Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 415-426, 2015.

[Naseer 08] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in Sub-100nm technologies," in *Proc. of IEEE International Conference on Electronics, Circuits and Systems*, pp. 586-589, 2008.

[Raoux 08] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S. H. Chen, H. L. Lung and C. H. Lam, "Phase-change random access memory: A scalable technology," in *IBM Journal of Research and Development*, vol. 52 , no. 4/5, pp. 465-479, Sep. 2008.

[Reviriego 12] P. Reviriego, M. F. Flanagan, S. F. Liu and J. A. Maestro, "Multiple Cell Upset Correction in Memories Using Difference Set Codes," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2592-2599, Nov. 2012.

[Reviriego 16] M. Demirci, P. Reviriego and J. A. Maestro, "Implementing Double Error Correction Orthogonal Latin Squares Codes in SRAM-based FPGAs," in *Microelectronics Reliability*, vol. 56, pp. 221-227, Jan. 2016.

[Wilkerson 10] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S. Lu, "Reducing cache power with low-cost, multi-bit error-correcting codes," in *Proc. of ACM annual international symposium on Computer architecture,* pp. 83–93, 2010.

[Yoo 14] I. Yoo and I. C. Park, "A search-less DEC BCH decoder for low-complexity fault-tolerant systems," in *Proc. of IEEE Workshop on Signal Processing Systems*, pp. 1-6, 2014.

[Yoo 16] H. Yoo, Y. Lee and I. C. Park, "Low-Power Parallel Chien Search Architecture Using a Two-Step Approach," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 3, pp. 269-273, Mar. 2016