Pipelined Diagonal Matrix Codes for Error Correction in Embedded Memories

C. H. Kavya
Department of ECE, SV University, Tirupati, India
kavyach.phd@gmail.com (corresponding author)

S. Varadarajan
Department of ECE, SV University, Tirupati, India
varadasouri@gmail.com

Received: 11 May 2025 | Revised: 6 July 2025 and 14 July 2025 | Accepted: 19 July 2025

 DOI: https://doi.org/10.48084/etasr.12069

ABSTRACT

Semiconductor memories are the basic storage elements for advanced FPGAs. However, with technology scaling due to high packing densities, the temperature of the device rises drastically, creating temporary or permanent faults that manifest as errors in the stored data. Permanent errors cannot be corrected, but temporary errors can. If the data in the memory is critical, such as data used during satellite or missile launch, patient data, etc., there is a need for Error-Detecting and Correcting (EDAC) code. Memories are represented as a matrix that stores data in rows. EDAC codes correct random (errors at various distributed locations) and burst errors (a sequence of erroneous bits within a row). The Hamming code represents the basis for any EDAC code. This work focuses on a single code used to identify 8-bit erroneous data and correct for 6 and 7 random bit errors and 8 burst errors. The matrix code utilizes a memory representation and Hamming code to detect and correct errors, taking care to increase the code rate with less area and delay. In addition, a pipelining technique is used to reduce power dissipation, which also helps to increase the speed of the design. The codes were modeled in Verilog HDL and verified for the Zynq 7000 series FPGA using Xilinx Vivado 2023.2. The results were verified for technological parameters, such as area in terms of LUTs, critical path delay, power dissipation, etc., and for non-technological parameters such as code rate, bit overhead, detection capability, correction capability, etc. The proposed pipelined matrix code was better in most aspects compared to existing designs.

Keywords-bit overhead; code rate; errors; Hamming code; multiple cell upset; single error correction; syndrome bits

I. INTRODUCTION

Semiconductor memories use MOSFETs and capacitors to store binary information in an integrated circuit. Logical 1 is stored when the capacitor is charged, and logical 0 is stored when the capacitor is discharged. Faults manifest as errors that may be temporary or permanent and need to be eliminated [1]. The utilization of memories in the past two decades has increased from 20% to 74%, but they keep being sensitive to radiation effects [2]. Soft errors are caused due to radiation that degrades the reliability of the data stored. Error Correction Codes (ECCs) are used to rectify soft errors [3]. A Single Error Correction (SEC) method corrects one error bit of a word by computing the parity bit with marginal effect on area and power consumed [4, 5]. As the number of errors increases, SEC is not sufficient; hence, multibit error correction codes are required. As technology scales, multi-bit errors occur in adjacent cells, manifested due to Multiple Cell Upsets (MCUs) that are alleviated by interleaving, i.e., the bits are rearranged within the same word to reduce their effect [6-7]. The increase in MCUs increases the requirement for efficient ECCs [8-9].

The advent of the latest technologies with non-functional constraints (timeliness, dependability, cost, power, time-to-market, etc) made the design of embedded systems more complex and challenging to implement [10, 11]. In addition, the complexity of such systems is increased by the integration of mixed-criticality distributed applications. Embedded systems evolved from single CPU systems, such as System-on-a-Chip (SoC), to concurrent computing structures, such as Multi-Processor-System-on-a-Chip (MPSoC) and Network-on-a-Chip (NoC) [12, 13]. Errors are a major concern in semiconductor electronics critical applications, such as patient monitoring systems and space programs, where malfunctions are inevitable due to soft errors. Furthermore, even a single failure may cause severe consequences in real-time nuclear systems due to missed deadlines [14, 15].

Many researchers aimed to enhance reliability with low overhead and improved code rates [16, 17]. Error Detection And Correction (EDAC) techniques can use Horizontal-Vertical Parity (HVP) and Diagonal Hamming Code (DHC) for memory protection. This approach combines parity-based

row/column checking with low diagonal redundancy to increase fault coverage [18, 19]. A 3D parity check code can extend traditional parity checking across three dimensions (row, column, diagonal) to enhance soft error resilience and improve multiple-bit error tolerance with a trade-off in bit overhead [20- 22]. A Horizontal-Vertical Double-Bit Diagonal Parity (HVDP) method has been proposed for soft error detection and correction with a balance between low redundancy and high error tolerance [23, 24].

Memory-mapped ECC is a cost-effective protection method for last-level caches, with reduced hardware complexity and robust error protection for modern processors [25, 26]. Modified Decimal Matrix Codes (MDMC) for protecting FPGA configuration memory from Multiple-Bit Upsets (MBU) enhance traditional decimal-based multiple error detection and correction with minimal hardware overhead [27-28]. Enhanced Decimal Matrix Code (EDMC), an extension of MDMC, was proposed to improve reliability against MCUs in semiconductor memories with lower redundancy compared to conventional BCH and Reed-Solomon codes in mission-critical embedded systems [29, 30]. Matrix codes [31-32] have been proposed for the design of cost-efficient and reliable memory chips to achieve low power consumption while maintaining strong error correction capability for Triple Modular Redundancy (TMR) in terms of hardware cost and power efficiency, being well-suited for aerospace and safety-critical embedded memory. Matrix-based codes [33] provide adjacent error correction for radiation-hardened space applications, particularly in space and aerospace memory systems, with reduced redundancy compared to conventional Hamming-based approaches. This approach uses a syndrome-based error localization decoding method to accurately detect error position and perform real-time correction [34].

II. DIAGONAL MATRIX CODES

Multibit error correction can be performed by using the Diagonal Hamming EDAC technique, as shown in Figure 1, for memories. The use of diagonal Hamming bits alone can save bit overhead in the coded word and memory used. During encoding, message bits are used to calculate the diagonal Hamming bits. Then, the code word is coded with data and Hamming bits (32+24 bits) and stored in memory.

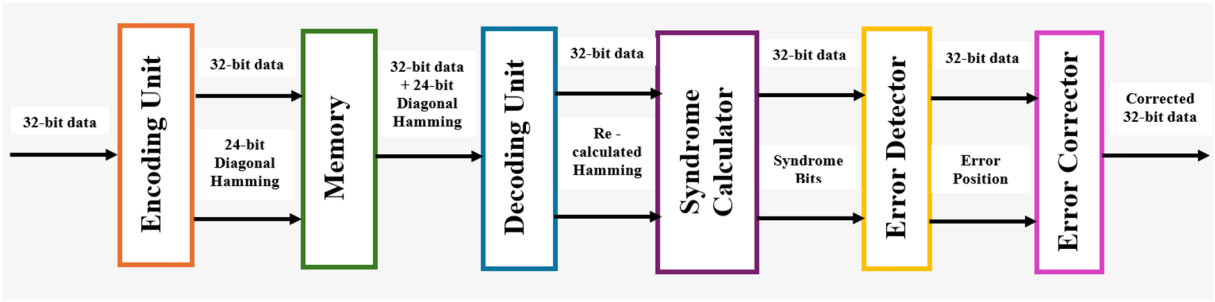

Fig. 1. Diagonal Hamming method.

Errors caused by radiation in the semiconductor memory may be identified and rectified in the decoder. Consider 32-bit data organized in an $m \times n$ matrix form as 4 rows and 8 columns, grouped as shown in Table II. The encoder produces Hamming bits through modulo-2 addition of the grouped data bits. The code word is stored in memory as shown in Figure 2. Hence, the eight diagonals consist of each 4 data bits and 3 Hamming bits.

TABLE I. 32-BIT DATA MESSAGE ORGANIZED IN 8 COLUMNS AND 4 ROWS

m3[7]	m3[6]	m3[5]	m3[4]	m3[3]	m3[2]	m3[1]	m3[0]
m2[7]	m2[6]	m2[5]	m2[4]	m2[3]	m2[2]	m2[1]	m2[0]
m1[7]	m1[6]	m1[5]	m1[4]	m1[3]	m1[2]	m1[1]	m1[0]
m0[7]	m0[6]	m0[5]	m0[4]	m0[3]	m0[2]	m0[1]	m0[0]

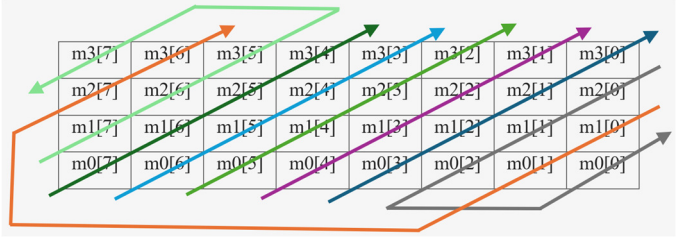

Fig. 2. Hamming bits generation.

TABLE II. CODE WORD

m3[7]	m3[5]	m2[6]	R1[3]	m1[7]	R1[2]	R1[1]
m3[6]	m2[7]	m0[1]	R2[3]	m1[0]	R2[2]	R2[1]
m0[0]	m0[2]	m1[1]	R3[3]	m2[0]	R3[2]	R3[1]
M3[4]	m2[5]	m1[6]	R4[3]	m0[7]	R4[2]	R4[1]
m3[3]	m2[4]	m1[5]	R5[3]	m0[6]	R5[2]	R5[1]
m3[2]	m2[3]	m1[4]	R6[3]	m0[5]	R6[2]	R6[1]
m3[1]	m2[2]	m1[3]	R7[3]	m0[4]	R7[2]	R7[1]
m3[0]	m2[1]	m1[2]	R8[3]	m0[3]	R8[2]	R8[1]

The grouped message bits of Table II represent the first, second, third, fourth, fifth, sixth, seventh and eight diagonal as {(m3[7], m3[5],m2[6], m1[7]), (m3[6], m2[7], m0[1],m1[0]), (m0[0], m0[2], m1[1], m2[0]), (m3[4], m2[5], m1[6], m0[7]), (m3[3], m2[4], m1[5], m0[6]), (m3[2], m2[3], m1[4], m0[5]), (m3[1],m2[2], m1[3], m0[4]), (m3[0], m2[1],m1[2], m0[3])}, respectively, i.e., each having four message bits. For these grouped bits, the Hamming bits are calculated as R1, R2, R3, R4, R5, R6, R7, R8, etc., with each 3 bits wide, as in (1)-(24):

For the first row:

$$R1[1] = XOR(m1[7], m2[6], m3[7]) \quad (1)$$

$$R1[2] = XOR(m1[7], m3[5], m3[7]) \quad (2)$$

$$R1[3] = XOR(m2[6], m3[5], m3[7]) \quad (3)$$

For the second row:

R2[1] = XOR(m1[0], m0[1], m3[6]) (4)

R2[2] = XOR(m1[0], m2[7], m3[6]) (5)

R2[3] = XOR(m0[1], m2[7], m3[6]) (6)

For the third row:

R3[1] = XOR(m2[0], m1[1], m0[0]) (7)

R3[2] = XOR(m2[0], m0[2], m0[0]) (8)

R3[3] = XOR(m1[1], m0[2], m0[0]) (9)

For the fourth row:

R4[1] = XOR(m0[7], m1[6], m3[4]) (10)

R4[2] = XOR(m0[7], m2[5], m3[4]) (11)

R4[3] = XOR(m1[6], m2[5], m3[4]) (12)

For the fifth row:

R5[1] = XOR(m0[6], m1[5], m3[3]) (13)

R5[2] = XOR(m0[6], m2[4], m3[3]) (14)

R5[3] = XOR(m1[5], m2[4], m3[3]) (15)

For the sixth row:

R6[1] = XOR(m0[5], m1[4], m3[2]) (16)

R6[2] = XOR(m0[5], m2[3], m3[2]) (17)

R6[3] = XOR(m1[4], m2[3], m3[2]) (18)

For the seventh row:

R7[1] = XOR(m0[4], m1[3], m3[1]) (19)

R7[2] = XOR(m0[4], m2[2], m3[1]) (20)

R7[3] = XOR(m1[3], m2[2], m3[1]) (21)

For the eight row:

R8[1] = XOR(m0[3], m1[2], m3[0]) (22)

R8[2] = XOR(m0[3], m2[1], m3[0]) (23)

R8[3] = XOR(m1[2], m2[1], m3[0]) (24)

In the encoder, 24 Hamming bits are obtained for 32-bit data, which are stored in memory as a code word in a matrix form, as shown in Figure 3. The code word read from memory is processed in the decoder by recalculating the Hamming bits and evaluating the syndrome bits as in (25)-(48):

For the first row:

S1[1] = XOR(R1[1], m1[7], m2[6], m3[7]) (25)

S1[2] = XOR(R1[2], m1[7], m3[5], m3[7]) (26)

S1[3] = XOR(R1[3], m2[6], m3[5], m3[7]) (27)

For the second row:

S2[1] = XOR(R2[1], m1[0], m0[1], m3[6]) (28)

S2[2] = XOR(R2[2], m1[0], m2[7], m3[6]) (29)

S2[3] = XOR(R2[3], m0[1], m2[7], m3[6]) (30)

For the third row:

S3[1] = XOR(R3[1], m2[0], m1[1], m0[0]) (31)

S3[2] = XOR(R3[2], m2[0], m0[2], m0[0]) (32)

S3[3] = XOR(R3[3], m1[1], m0[2], m0[0]) (33)

For the fourth row:

S4[1] = XOR(R4[1], m0[7], m1[6], m3[4]) (34)

S4[2] = XOR(R4[2], m0[7], m2[5], m3[4]) (35)

S4[3] = XOR(R4[3], m1[6], m2[5], m3[4]) (36)

For the fifth row:

S5[1] = XOR(R5[1], m0[6], m1[5], m3[3]) (37)

S5[2] = XOR(R5[2], m0[6], m2[4], m3[3]) (38)

S5[3] = XOR(R5[3], m1[5], m2[4], m3[3]) (39)

For the sixth row:

S6[1] = XOR(R6[1], m0[5], m1[4], m3[2]) (40)

S6[2] = XOR(R6[2], m0[5], m2[3], m3[2]) (41)

S6[3] = XOR(R6[3], m1[4], m2[3], m3[2]) (42)

For the seventh row:

S7[1] = XOR(R7[1], m0[4], m1[3], m3[1]) (43)

S7[2] = XOR(R7[2], m0[4], m2[2], m3[1]) (44)

S7[3] = XOR(R7[3], m1[3], m2[2], m3[1]) (45)

For the eight row:

S8[1] = XOR(R8[1], m0[3], m1[2], m3[0]) (46)

S8[2] = XOR(R8[2], m0[3], m2[1], m3[0]) (47)

S8[3] = XOR(R8[3], m1[2], m2[1], m3[0]) (48)

If syndrome bits are zero, then data is correct. If syndrome bits are non-zero, then data is corrupted. To correct the data read from memory, the error location is identified for an error in the third row as:

$$(S3[3] * (2^2)) + (S3[2] * (2^1)) + (S3[0] * (2^0)) \quad (49)$$

Then, the error corrector inverts the corresponding bit to correct the error in the data. This process is repeated until all erroneous bits are corrected.

Various existing error detection and correction codes were compared, and, as shown in Table III, the diagonal hamming method was better. The diagonal hamming method has maximum error correction of 8 bits with a code rate of 57% and bit overhead of 75%, which is minimum among all existing methods. Table II shows the comparison of technology parameters between existing codes. The proposed code has almost the same or slightly higher values than those of existing codes when comparing encoders. Table V compares existing methods for decoders. Although DHC requires more area,

delay, and power than existing codes, its correction capability is higher, i.e., 8 bits out of 32-bit data can be corrected.

TABLE III. COMPARISON IN TERMS OF BIT OVERHEAD AND CODE RATE

ECCs	Data bits (d)	Redundant bits (h)	Maximum error correction	Bit overhead (%)	Code rate (%)
DMC	32	36	5	112.5	47
MDMC	32	32	16	100	50
HVD	32	38	3	118.5	45.75
DH	32	24	8	75	57

TABLE IV. COMPARISON OF THE PROPOSED 32-BIT ENCODER CODE WITH EXISTING

Method	No. of slices	Delay	Total logic power	Total data power	Total IO's power
DMC	40	1.411	0.00007	0.00141	0.02372
MDMC	32	1.380	0.00004	0.00147	0.02297
HVD	34	1.287	0.00007	0.00045	0.01197
DH	38	1.668	0.00006	0.00144	0.02854

TABLE V. COMPARISON OF THE PROPOSED 32-BIT DECODER CODE WITH EXISTING ONES

Method	No. of slices	Delay	Total logic power	Total data power	Total IO's power
DMC	64	1.412	0.00011	0.00050	0.01145
MDMC	89	1.439	0.00014	0.00071	0.00625
HVD	80	1.276	0.00001	0.00033	0.01129
DH	150	2.397	0.00031	0.00455	0.06193

III. PIPELINED DIAGONAL MATRIX CODES

The Pipelined Diagonal Hamming Code (PDHC) is an advanced EDAC technique, specifically designed for high-speed and low-power memory systems. It combines DHC with a pipelining architecture to efficiently detect and correct soft errors in memory cells caused by radiation-induced Single-Event Upsets (SEUs) and Multiple-Bit Upsets (MBUs), which are common in VLSI systems and FPGA-based memory architectures. DHC is beneficial because the traditional Hamming code uses horizontal parity for single-bit error detection and correction. DHC extends parity checks along the diagonal lines of the memory matrix. DHC is effective in detecting multiple-bit errors that occur in adjacent cells or diagonal patterns, unlike traditional row-based parity checks. The pipelined architecture improves the throughput and speed of error detection and correction by breaking the process down into multiple pipeline stages. This allows concurrent error detection and correction operations for high-speed real-time applications, such as FPGA and ASIC-based memories, as shown in Figure 5.

The key architecture of PDHC includes:

- Stage 1: The encoding stage generates horizontal, vertical, and diagonal parity bits.
- Stage 2: Error syndrome detection identifies error location through syndrome vector analysis.
- Stage 3: The pipeline register for error correction corrects Single- and Double-Bit Errors (SEC-DED).
- Stage 4: Output error-free data forward corrected data to the memory controller.

The mathematical model includes the parity check matrix (H) for DHC:

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (50)$$

and the syndrome vector:

$$S = H \times C^T \quad (51)$$

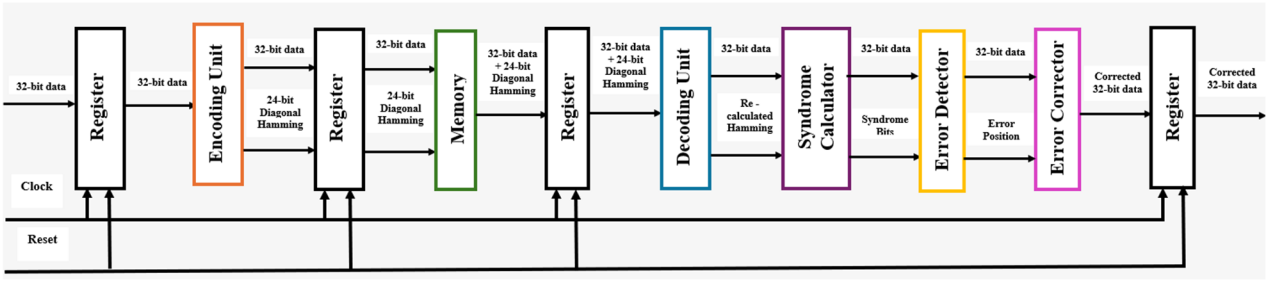

Fig. 3. Pipelined Diagonal Hamming Code (PDHC).

Table VI describes the key features of pipelining, error detection capability, redundancy, low power architecture, and real-time error correction. The advantages over the traditional Hamming codes are shown in Table VII. The major advantages include improvement in error handling capability from single- to multi-bit error correction, higher power efficiency with low latency, and area-efficient utilization in FPGAs.

TABLE VI. KEY FEATURES OF PDHC

Feature	Description
Pipelining	Reduces latency and increases throughput
Multi-bit error detection	Detects both Single-Bit and Double-Bit Errors (SEC-DED)
Low redundancy	Lower hardware overhead compared to traditional Hamming
Low-power architecture	Suitable for FPGA and VLSI-based memory systems
Real-time error correction	High-speed error recovery in memory chips

TABLE VII. ADVANTAGES OVER TRADITIONAL HAMMING CODE

Feature	Traditional Hamming code	Pipelined Diagonal Hamming Code (PDHC)
Error handling capability	Single error correction	Multi-bit error correction
Redundancy overhead	High	Low
Power efficiency	Moderate	High
Latency	High	Low
FPGA implementation	Complex	Area-efficient

The applications of PDHC include FPGA-based memories for real-time error detection, VLSI-based cache memory for multi-bit error tolerance, space applications for radiation-hardened memory, NoC interconnects for low-redundancy error control, biomedical memory systems for low-power high-speed correction, etc.

The hardware architecture for FPGA implementation requires Verilog modeling of components, i.e., the Hamming encoder module, syndrome vector generation, pipeline register for error detection, corrector block (SEC-DED), fault detection flag, etc. The benefits of the pipeline-based architecture are tabulated in Table VIII, with major improvements such as reductions in latency by 40%, power consumption by 35%, area overhead by 30%, and increased error detection speed by 50%.

TABLE VIII. ADVANTAGES OVER TRADITIONAL HAMMING CODE

Aspect	Improvement
Latency	Reduced by 40%
Power consumption	Reduced by 35%
Area overhead	Reduced by 30%
Error detection speed	Increased by 50%

Therefore, PDHC is an optimized VLSI architecture for low-power and high-speed error detection and correction, suitable for real-time FPGA implementation, memory chips, NoC interconnects, and radiation-tolerant systems, as it outperforms conventional Hamming and SEC-DED techniques in terms of error resilience and energy efficiency.

IV. RESULTS AND DISCUSSION

Error detection and correction codes were modeled in Verilog HDL and implemented in Xilinx Vivado Tool 2023.2 version for a 28 nm Zynq 7000 series FPGA with part number xc7z020clg484-2. The DHC design was synthesized for the above-mentioned device, and its RTL schematic is shown in Figure 4. It occupies 70 cells, 67 I/O ports, and 143 nets.

The proposed PDHC design was synthesized for the same device, and its RTL schematic is shown in Figure 5. It occupies 74 cells, 67 I/O ports, and 335 nets.

The simulation results were verified for memory without any read errors, and the same data was obtained at the output, as shown in Figure 6. The simulation results were also verified for a memory with 8-bit adjacent read errors, and still, the actual data encoded is obtained at the output, as shown in Figure 9.

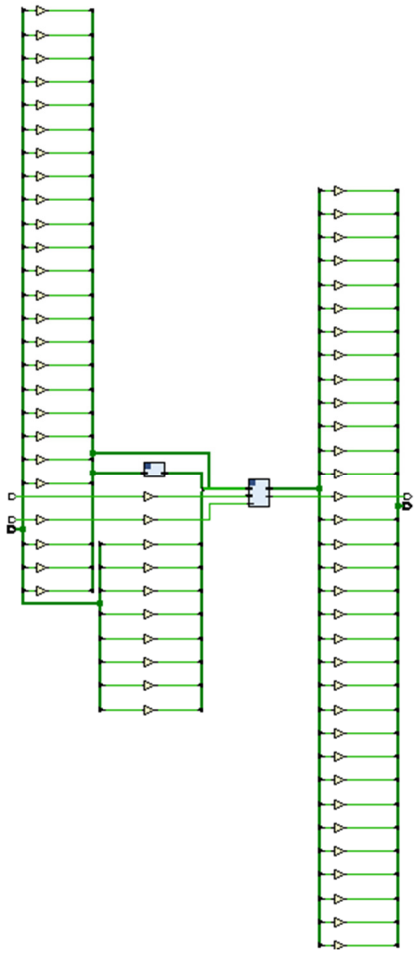

Fig. 4. RTL schematic of DHC.

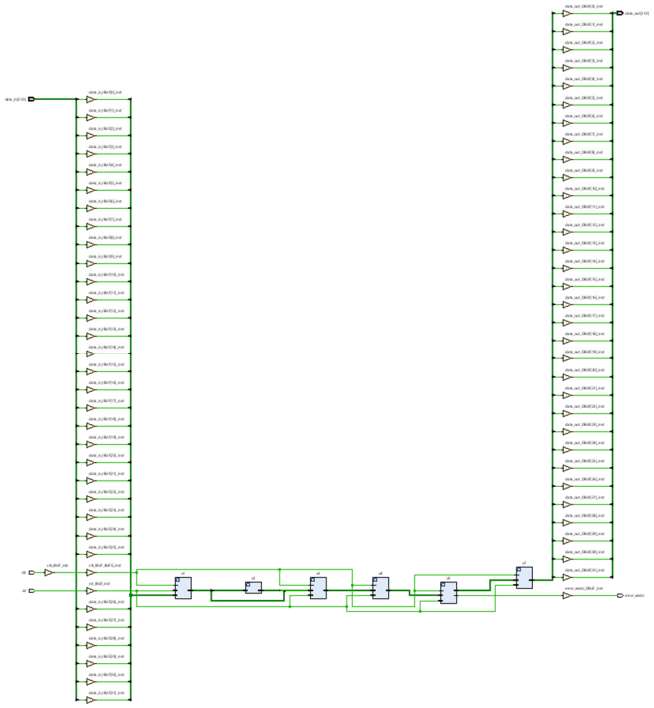

Fig. 5. RTL schematic of PDHC.

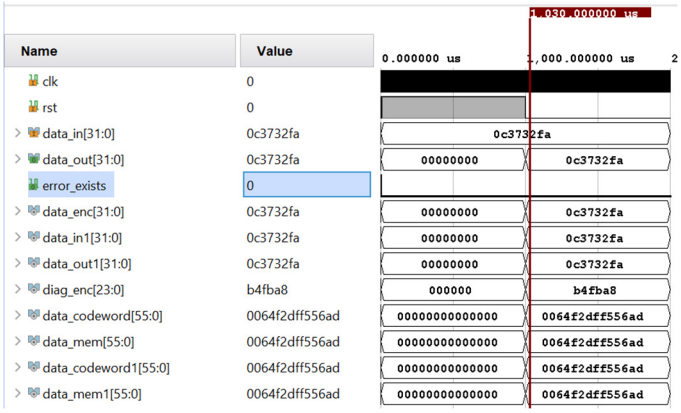

Fig. 6. Simulation results without data bit errors.

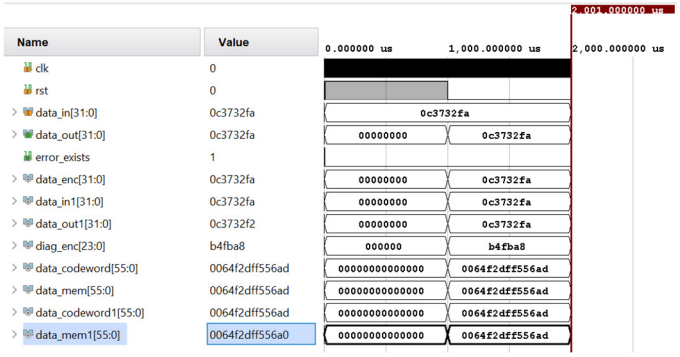

Fig. 7. Simulation results with 8-bit data errors.

The overall implementation results are shown in Table IX. PDHC reduces slice LUTs by 20.81%, delay by 20.27%, and logic power by 36.71%. However, the number of slice registers used increases by 62.03%. In addition, the code can be modified for reduced area utilization and improved code rate of at least 65% using a smaller number of parity bits.

TABLE IX. DEVICE UTILIZATION SUMMARY OF DIAGONAL AND PIPELINED DIAGONAL HAMMING CODES

Parameter	DHC	PDHC
Cells	72	74
I/O ports	67	67
Nets	183	335
Slice LUTs (out of 53200)	221	175
Slice registers (out of 106400)	88	232
Register as flip-flop	56	232
Register as latch	32	0
Delay (ns)	34.928	27.848
Logic power (mW)	5.755	3.642

V. CONCLUSION

Error correction is the main problem for digital data transmission in memories. To overcome memory errors, the best technique that can be used is the DHC method. The main idea behind DHC is to decrease the maximum number of errors during the transmission of bits in memory. Pipelining is used to reduce latency and improve throughput in real-time scenarios with multibit error detection and correction. The existing designs do not adapt techniques such as parallel processing, pipelining, etc. The proposed PDHC involves pipelining at four places, before and after the encoder and decoder, to synchronize and improve throughput. The proposed PDHC reduces the area and delay by 91.76% and 84.18%, respectively, with a correction capability of 8 bits in 32 data bits compared to the 3D parity code. PDHC reduces slice LUTs by 20.81%, delay by 20.27%, and logic power by 36.71%. However, the number of slice registers used increases by 62.03%. In the future, these codes can be optimized for area, with minimum bit overhead and a code rate of at least 65%.

REFERENCES

[1] A. J. Olazábal and J. P. Guerra, "Multiple Cell Upsets Inside Aircrafts. New Fault-Tolerant Architecture," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 1, pp. 332–342, Oct. 2019, https://doi.org/10.1109/TAES.2018.2852198.

[2] J. Gracia-Morán, L. J. Saiz-Adalid, D. Gil-Tomás, and P. J. Gil-Vicente, "Improving Error Correction Codes for Multiple-Cell Upsets in Space Applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 2132–2142, Jul. 2018, https://doi.org/10.1109/TVLSI.2018.2837220.

[3] "IEEE Standard for Error Correction Coding of Flash Memory Using Low-Density Parity Check Codes." IEEE, https://doi.org/10.1109/IEEESTD.2019.8654228.

[4] J. Samanta, J. Bhaumik, and S. Barman, "Compact and power efficient SEC-DED codec for computer memory," *Microsystem Technologies*, vol. 27, no. 2, pp. 359–368, Feb. 2021, https://doi.org/10.1007/s00542-019-04366-7.

[5] A. M. A. Hamdoon, Z. G. Mohammed, and E. A. Mohammed, "Design and implementation of single bit error correction linear block code system based on FPGA," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 4, pp. 1785–1795, Aug. 2019, https://doi.org/10.12928/telkomnika.v17i4.12033.

[6] S. Liu, P. Reviriego, J. A. Maestro, and L. Xiao, "Fault tolerant encoders for Single Error Correction and Double Adjacent Error Correction codes," *Microelectronics Reliability*, vol. 81, pp. 167–173, Feb. 2018, https://doi.org/10.1016/j.microrel.2017.12.017.

[7] A. Sánchez-Macián, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement," *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 1, pp. 574–576, Mar. 2014, https://doi.org/10.1109/TDMR.2012.2204753.

[8] A. Dutta and N. A. Touba, "Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code," in *25th IEEE VLSI Test Symmposium (VTS'07)*, Berkeley, CA, USA, May 2007, pp. 349–354, https://doi.org/10.1109/VTS.2007.40.

[9] P. Reviriego, J. Martínez, S. Pontarelli, and J. A. Maestro, "A Method to Design SEC-DED-DAEC Codes With Optimized Decoding," *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 3, pp. 884–889, Sep. 2014, https://doi.org/10.1109/TDMR.2014.2332364.

[10] W. Zhou, H. Zhang, H. Wang, and Y. Wang, "Designing scrubbing strategy for memories suffering MCUs through the selection of optimal interleaving distance," *International Journal of Computational Science and Engineering*, vol. 19, no. 1, pp. 53–63, Jan. 2019, https://doi.org/10.1504/IJCSE.2019.099639.

[11] C. Argyrides, D. K. Pradhan, and T. Kocak, "Matrix Codes for Reliable and Cost Efficient Memory Chips," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 3, pp. 420–428, Mar. 2011, https://doi.org/10.1109/TVLSI.2009.2036362.

[12] M. S. Sunita and V. S. K. Bhaaskaran, "Matrix Code Based Multiple Error Correction Technique for N-Bit Memory Data," *International Journal of VLSI Design & Communication Systems*, vol. 4, no. 1, pp. 29–37, Feb. 2013, https://doi.org/10.5121/vlsic.2013.4103.

[13] J. Gracia-Moran, L. J. Saiz-Adalid, J. C. Baraza-Calvo, and P. Gil, "Correction of Adjacent Errors with Low Redundant Matrix Error

Correction Codes," in *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*, Foz do Iguaçu, Brazil, Oct. 2018, pp. 107–114, https://doi.org/10.1109/LADC.2018.00021.

[14] V. Badole and A. Udawat, "Implementation of multidirectional parity check code using hamming code for error detection and correction," *International Journal of Research in Advent Technology*, vol. 2, no. 5, pp. 1–6, 2014.

[15] S. Tambatkar, S. N. Menon, V. Sudarshan, M. Vinodhini, and N. S. Murty, "Error detection and correction in semiconductor memories using 3D parity check code with hamming code," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, Apr. 2017, pp. 0974–0978, https://doi.org/10.1109/ICCSP.2017.8286516.

[16] T. Maheswari and M. P. Sukumar, "Error Detection and Correction in SRAM Cell Using Decimal Matrix Code," *IOSR Journal of VLSI and Signal Processing*, vol. 5, no. 1, pp. 9–14, Jan. 2015.

[17] J. Guo, L. Xiao, Z. Mao, and Q. Zhao, "Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 1, pp. 127–135, Jan. 2014, https://doi.org/10.1109/TVLSI.2013.2238565.

[18] A. Ahilan and P. Deepa, "Modified Decimal Matrix Codes in FPGA configuration memory for multiple bit upsets," in *2015 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, Jan. 2015, pp. 1–5, https://doi.org/10.1109/ICCCI.2015.7218146.

[19] S. Manoj and C. Babu, "Improved error detection and correction for memory reliability against multiple cell upsets using DMC & PMC," in *2016 IEEE Annual India Conference (INDICON)*, Bangalore, India, Dec. 2016, pp. 1–6, https://doi.org/10.1109/INDICON.2016.7839094.

[20] S. Sharma and P. Vijayakumar, "An HVD based error detection and correction of soft errors in semiconductor memories used for space applications," in *2012 International Conference on Devices, Circuits and Systems (ICDCS)*, Coimbatore, Mar. 2012, pp. 563–567, https://doi.org/10.1109/ICDCSyst.2012.6188771.

[21] Md. S. Rahman, M. S. Sadi, S. Ahammed, and J. Jurjens, "Soft error tolerance using Horizontal-Vertical-Double-Bit Diagonal parity method," in *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, Savar, Bangladesh, May 2015, pp. 1–6, https://doi.org/10.1109/ICEEICT.2015.7307411.

[22] W. Toghuj, "Modifying Hamming code and using the replication method to protect memory against triple soft errors," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 5, pp. 2533–2542, Oct. 2020, https://doi.org/10.12928/telkomnika.v18i5.13345.

[23] S. Dey, Aurangozeb, and M. Hossain, "Low-Latency Burst Error Detection and Correction in Decision-Feedback Equalization," *IEEE Open Journal of Circuits and Systems*, vol. 2, pp. 91–100, 2021, https://doi.org/10.1109/OJCAS.2020.3039256.

[24] J. Li, P. Reviriego, L. Xiao, C. Argyrides, and J. Li, "Extending 3-bit Burst Error-Correction Codes With Quadruple Adjacent Error Correction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 221–229, Oct. 2018, https://doi.org/10.1109/TVLSI.2017.2766361.

[25] J. Athira and B. Yamuna, "FPGA Implementation of an Area Efficient Matrix Code with Encoder Reuse Method," in *2018 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India Apr. 2018, pp. 0254–0257, https://doi.org/10.1109/ICCSP.2018.8524371.

[26] L. J. Saiz-Adalid, J. Gracia-Morán, D. Gil-Tomás, J. C. Baraza-Calvo, and P. J. Gil-Vicente, "Ultrafast Codes for Multiple Adjacent Error Correction and Double Error Detection," *IEEE Access*, vol. 7, pp. 151131–151143, 2019, https://doi.org/10.1109/ACCESS.2019.2947315.

[27] T. A. Gulliver and V. K. Bhargava, "A systematic (16,8) code for correcting double errors and detecting triple-adjacent errors," *IEEE Transactions on Computers*, vol. 42, no. 1, pp. 109–112, Jan. 1993, https://doi.org/10.1109/12.192220.

[28] K. Neelima and C. Subhas, "Half diagonal matrix codes for reliable embedded memories," *International journal of health sciences*, pp. 11664–11677, May 2022, https://doi.org/10.53730/ijhs.v6nS2.8117.

[29] N. Koppala and C. Subhas, "Low overhead optimal parity codes," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 20, no. 3, Jun. 2022, Art. no. 501, https://doi.org/10.12928/telkomnika.v20i3.23301.

[30] N. K and C. Subhas, "Proficient Adjacent Error Correcting Codes," in *2023 IEEE 3rd International Conference on Applied Electromagnetics, Signal Processing, & Communication (AESPC)*, Bhubaneswar, India, Nov. 2023, pp. 1–5, https://doi.org/10.1109/AESPC59761.2023.10389891.

[31] K. Neelima and C. Subhas, "Modified Proficient Adjacent Error Correcting Codes," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 5, Sep. 2023, Art. no. 100277, https://doi.org/10.1016/j.prime.2023.100277.

[32] N. Koppala, N. A. Kumar, S. Satyam, and N. V. Teja, "Proficient matrix codes for error detection and correctionin 8-port network on chip routers," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 29, no. 3, Mar. 2023, Art. no. 1336, https://doi.org/10.11591/ijeecs.v29.i3.pp1336-1344.

[33] N. K and C. Subhas, "Modified Matrix Codes for Shielding Memories against Adjacent Errors," *ASEAN Engineering Journal*, vol. 14, no. 2, pp. 19–25, May 2024, https://doi.org/10.11113/aej.v14.20428.

[34] L. Jordanova, L. Laskov, and D. Dobrev, "Influence of BCH and LDPC Code Parameters on the BER Characteristic of Satellite DVB Channels," *Engineering, Technology & Applied Science Research*, vol. 4, no. 1, pp. 591–595, Feb. 2014, https://doi.org/10.48084/etasr.394.