

Enabling Ultra-Reliable Memories: A Practical Framework for Zero Mis-correction SEC-DED-DAEC Codes for Safety-Critical Systems

Guixiang Chen, Sheng Liu*, Bo Yuan, Yang Guo

College of Computer Science and Technology

National University of Defense Technology

Hunan, China

{chengguixiang19, liusheng83, yuanbo18, guoyang}@nudt.edu.cn

Abstract—In safety-critical systems such as autonomous driving and aerospace, memory reliability standards are evolving from "high-reliability" to "ultra-reliability," demanding the eradication of all foreseeable, deterministic failure modes. To address the prevalent challenge of Double Adjacent Errors (DAE) induced from radiation, the design of SEC-DED-DAEC codes faces a critical dilemma: efficient but flawed Hsiao-based codes that risk miscorrection, versus correct-by-construction but costly and inflexible OLS-based codes. This trade-off between efficiency and correctness presents a key barrier to designing ultra-reliable systems. To resolve this impasse, this paper introduces MCTS-CDB, a novel Computer-Aided Design (CAD) framework. By integrating a CDCL-inspired search with Monte Carlo Tree Search (MCTS) guidance, it systematically constructs codes that achieve a zero-miscorrection guarantee within the highly-efficient Hsiao architecture. Experimental results validate our approach, showing that compared to a wide range of existing schemes, our generated codes achieve the correctness while reducing average encoding and decoding delays by 24.15% and 13.66%. This work provides a practical solution for designing the ultra-reliable memory subsystems required by next-generation safety-critical applications.

Index Terms—Ultra-Reliable, SEC-DED-DAEC, Zero Miscorrection, Safety-Critical

I. INTRODUCTION

The proliferation of safety-critical systems, from autonomous vehicles governed by SAE J3016[1] to avionics hardware certified under DO-254[2] and ECSS-Q-ST-60-02C[3], has driven a paradigm shift in reliability standards. These domains demand a move from traditional high-reliability to ultra-reliability, an engineering principle focused on the eradication of all foreseeable, deterministic failure modes. This zero-tolerance for predictable risk, while popularized in communications by URLLC[4], is already a cornerstone of functional safety standards like ISO 26262[5], which mandate failure rates low enough to be considered ultra-reliable.

In memory subsystems, this requirement forces a focus on soft errors, whose characteristics evolve with technology scaling[6]. As process nodes enter the sub-nanometer regime, heightened cell density and lower operating voltages make

This work was supported by Key Laboratory of Advanced Microprocessor Chips and Systems.

*Corresponding author.

Double Adjacent Errors (DAEs) the most prevalent multi-bit upset (MBU)[7][8], accounting for over 90% of such events. Consequently, mitigating DAEs is a primary challenge[9]. Viewed through the rigorous lens of ultra-reliability, the known miscorrection issue in DAE codes transforms from a low-probability nuisance into an unacceptable, deterministic flaw.

While Single-Error Correction, Double-Error Detection, Double-Adjacent-Error Correction (SEC-DED-DAEC) codes have become the de facto solution for their balance of protection and efficiency[10], their implementation is hindered by a fundamental design dilemma between two competing methods:

- **Hsiao Code-Based Schemes** promise high efficiency and flexibility, but their construction is an NP-complete problem[11]. This forces a harsh trade-off: either accept low overhead with high miscorrection rates (up to 55.2%)[12] or sacrifice the efficiency advantage for lower miscorrection by adding more check bits[13].
- **Orthogonal Latin Square (OLS) Code-Based Schemes** conversely guarantee zero miscorrection but impose prohibitive overhead, requiring excessive check bits[14] and offering poor scalability due to rigid data-width constraints[15].

This unresolved deadlock between flawed efficiency and costly correctness leaves designers of ultra-reliable systems without a satisfactory solution. This paper resolves this impasse by introducing MCTS-CDB, a novel Computer-Aided Design (CAD) framework that constructs a new class of SEC-DED-DAEC codes. Our work achieves a zero-miscorrection guarantee within the highly efficient Hsiao code framework. This synergy breaks the long-standing trade-off, delivering the correctness of OLS codes while retaining the minimal check bits and data-width flexibility of the Hsiao family.

Our key contributions include:

- Analysis of the minimum for the check bits required in zero-miscorrection SEC-DED-DAEC codes.
- An efficient, Conflict-Driven Clause Learning (CDCL)-inspired parallel search algorithm, enhanced with Monte Carlo Tree Search (MCTS), for rapid check-matrix construction.

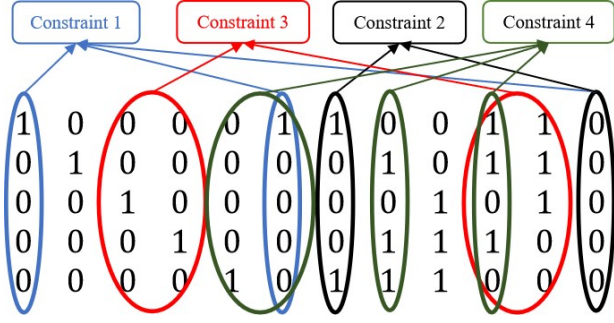


Fig. 1. Examples of Constraints Violation.

- A sliding-window-based Boolean constraint optimization model to further reduce hardware overhead.

II. BACKGROUND

A. SEC-DED-DAEC Codes

SEC-DED-DAEC codes are systematic linear block codes. An (n, k) code of this type transforms a k -bit data word into an n -bit codeword by appending $n-k$ check bits. The encoding transformation is defined by a generator matrix $G_{k \times n}$, and conversely, decoding begins with the syndrome computation, which multiplies the codeword by the check matrix $H_{(n-k) \times n}$. Thus the two matrices are mathematical duals, linked by the relationship in Equation (1).

$$data \cdot (G \cdot H^T) = \mathbf{0} \quad (1)$$

The construction of the check matrix H for a zero-miscorrection SEC-DED-DAEC code is governed by a precise set of combinatorial constraints:

Constraint 1: All columns in H are distinct and non-zero.

Constraint 2: All columns in H have an odd Hamming weight.

Constraint 3: The XOR sums of two adjacent columns in H are all different and also different from all columns.

Specifically, Constraint 1 ensures unique syndromes for single-bit errors, while Constraint 2 guarantees the detectability of all double-bit errors. Building on this, Constraint 3 distinguishes between different adjacent double-bit errors by assigning them unique syndromes. While the three constraints define a functional SEC-DED-DAEC code[16], they are insufficient to prevent all miscorrections as depicted in Fig. 1. To eliminate this critical vulnerability, a crucial constraint is introduced:

Constraint 4 (extra): The XOR sums of any two adjacent columns must differ from those of non-adjacent columns.

B. Minimum for the Required Check Bits

Determining the minimum number of check bits (r) required for a given data width (k) is a central challenge in ECC design. While the minimum number of check bits for standard SEC-DED codes can be estimated, a similar estimation for zero-miscorrection SEC-DED-DAEC codes is insufficient due to the latter's significantly more complex encoding principles.

The primary challenge arises from Constraint 3 and Constraint 4, which demands that the set of adjacent error syndromes be disjoint from the set of non-adjacent error syndromes. All available non-zero even-weight syndromes must be partitioned to accommodate both the $k+r-1$ adjacent error syndromes and all unique syndromes produced by other error types. The main difficulty lies in estimating the number of unique syndromes from these other errors. Simplified by assuming the dominant other errors are non-adjacent double-bit errors, it can be formulated as Equation (2).

$$2^{r-1} \geq k+r + \frac{(k+r-1)(k+r-2)}{2s} \quad (2)$$

In Equation (2), a characteristic parameter s is introduced, which represents the degree of homogeneity, or the average collision rate of non-adjacent double-error syndromes. When $s = 1$, it implies that every non-adjacent double-bit error produces a unique syndrome; under this condition, the inequality degenerates into the requirement for a Double-Error Correcting (DEC) code. For a SEC-DED-DAEC code, it is evident that $s > 1$, and a larger s signifies a higher degree of syndrome collision as well as lower number of check bits. This structural homogeneity is another manifestation of the inherent difficulty in constructing an optimal check matrix.

III. NOVEL FRAMEWORK FOR CONSTRUCTING CHECK MATRIX

The construction of the check matrix requires selecting k optimal odd-weight vectors that satisfy all constraints, which is a combinatorial optimization search problem, and the enormous search space renders brute-force methods computationally intractable.

Accordingly, we propose MCTS-CDB, a four-stage framework composed of: preprocessing and vector pool construction, MCTS for ordering of candidates, a CDCL-inspired backtrack solver for systematic construction, and local refinement with a Boolean constraint model.

A. Pre-process : Construct Odd-weight Vectors Pool

Constraint 2 dictates that the check matrix must be constructed exclusively from odd-weight vectors. While this task could be modeled at the low-level Boolean variable level, doing so would introduce numerous auxiliary constraints and lead to a computationally intractable problem size.

Therefore, proposed framework adopts a more efficient, higher-level abstraction by operating directly at the BitVector level. The crucial first step in this process is to generate a high-quality pool of all valid candidate vectors. Algorithm 1 details our combinatorial method for constructing this pruned odd-weight vector pool.

B. Monte Carlo Tree Search Evaluating Candidate Vectors

Although operating at the BitVector level simplifies the model, the construction problem remains intractable due to the enormous search space and tightly coupled constraints. For instance, on a moderately-sized instance such as $k = 16$ and

Algorithm 1 Generate Odd-Weight Vector Pool

Input: Number of Check Bits r
Output: Odd-Weight Vector Pool V_{odd}

```
1:  $V_{odd} \leftarrow \emptyset$  ▷ Initialization
2: for  $w \leftarrow 1$  to  $r$  step 2 do ▷ Iterate odd weight
3:    $P \leftarrow \{0, 1, \dots, r-1\}$ 
4:    $C \leftarrow \text{all\_combinations}(P, w)$ 
5:   for each combination  $c \in C$  do
6:      $v \leftarrow \text{zero\_vector}(r)$ 
7:     for each index  $j \in c$  do
8:        $v[j] \leftarrow 1$ 
9:     end for
10:    if  $w == 3$  and  $\text{has\_adjacent\_ones}(v)$  then ▷
11:       $vec_{w=3}$  with adjacent ones violate constraint 3 or 4
12:      continue
13:    else
14:       $V_{odd} \leftarrow V_{odd} \cup \{v\}$ 
15:    end if
16:  end for
17: return  $V_{odd}$ 
```

$r = 8$, state-of-the-art SMT solvers like CVC5[17] and Z3[18] fail to produce a feasible solution within practical time limit.

To this end, Monte Carlo Tree Search (MCTS) is employed, a powerful technique renowned for navigating vast decision spaces. Drawing on its core philosophy, MCTS is adapted to perform a resource-limited, preliminary exploration of the solution space. The goal is not to exhaustively find the final matrix, but rather to provide crucial heuristic guidance by identifying and prioritizing the most promising construction paths.

The MCTS algorithm introduced in this paper adheres to the standard four-stage pipeline—*selection*, *expansion*, *simulation*, and *back-propagation*—as illustrated in Fig. 2. However, since MCTS functions as a heuristic preprocessing stage rather than a complete solver, its simulation phase is adapted. Instead of yielding a definitive success/failure verdict on the final matrix construction, a simulation’s outcome is determined by a new success criterion based on search progress. As defined in Equation (3), a simulation is considered successful simply if the current construction depth reaches a predefined threshold. This threshold is not arbitrary; it can be derived by relaxing the parameter s in Equation (2), effectively defining a smaller, more tractable sub-problem for MCTS to explore.

$$tag_{win} = (level_{curr} == level_{set}) \ ? \ 1 : 0 \quad (3)$$

During the selection phase, MCTS iteratively descends the search tree by selecting the most promising child node at each level. This decision is guided by the Upper Confidence bound applied to Trees (UCT) shown in Equation (4).

$$UCT_{vector_i} = \frac{\omega_i}{n_i} + C * \sqrt{\frac{\log(N_i)}{n_i}} \quad (4)$$

The first term $\frac{\omega_i}{n_i}$, represents exploitation, calculating the current average success rate of a given node $vector_i$, where ω_i is the number of successful simulations (“wins”) initiated from that node, and n_i is its total number of visits. Conversely, the second term provides the exploration incentive; it favors nodes that have been visited less frequently (n_i) relative to the total number of simulations run from the parent node, N_i . The balance between these two components is tuned by the exploration parameter C , a constant that adjusts the weight given to exploring less-certain paths. At each step of the selection process, the node with the highest UCT score is chosen to continue the descent.

Once a simulation in a given path terminates, either by reaching the depth threshold or a dead end, the back-propagation phase begins. The outcome is propagated back up the search tree, updating the evaluation scores of all visited nodes according to Equation (5). This update rule assigns a reward based on the achieved search depth, $level_{end}$, ensuring that paths leading to deeper, more promising constructions accumulate a higher value. This score, in turn, dynamically influences future selections via the UCT calculation, guiding the search toward more fruitful regions of the solution space.

$$value_i += \frac{1}{level_{set} - level_{end} + 1} \quad (5)$$

Due to resource constraints, the goal of this entire MCTS stage is not to find a single, final solution, but rather to identify a set of high-quality candidates for the next stage. Consequently, after the search budget is exhausted, all explored vector sequences are ranked based on their accumulated value scores. Only the top- n sequences—representing a diverse yet high-potential set of starting points—are retained and passed to the next stage.

To make this extensive exploration computationally feasible, the entire MCTS process is parallelized (P-MCTS). The primary challenge in this parallel implementation lies in the synchronization of shared variables, as the UCT scores and other heuristic values must be kept consistent across all threads to maintain the integrity of the search. Although this synchronization introduces some overhead and caps the theoretical maximum performance, the substantial overall gain from parallel exploration far outweighs this manageable cost.

C. Parallel BitVector Conflict-Driven Backtrack

While conventional heuristic search is inefficient due to chronological backtracking, modern CDCL-based SAT solvers are also suboptimal for operating at the bit-level.

To bridge this methodological gap, Parallel, Bit-Vector Conflict-Driven Backtrack (P-CDB) is introduced, it extends the powerful principles of CDCL to operate directly at the BitVector level and performs conflict analysis and non-chronological backtracking on vectors, guided by a heuristic function to efficiently converge on hardware-optimal solutions.

1) *Heuristic Decision:* The design of the heuristic function is pivotal to the efficiency of P-CDB, guiding its search toward solutions that are not only valid but also hardware-optimal. Our

P-MCTS

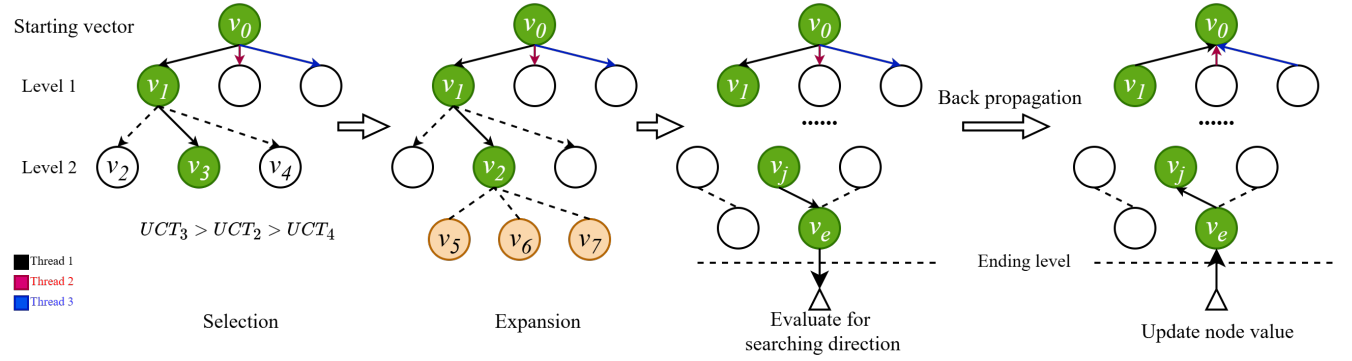


Fig. 2. Multi-thread Monte Carlo Tree Search

heuristic function assesses both the quality of the current partial solution and the feasibility of future search paths.

The quality of the current state is evaluated based on three hardware-centric metrics:

- **Total Matrix Weight:** Correlates directly to the total number of XOR gates, impacting the overall circuit area.
- **Maximum Row Weight:** Determines the critical path delay of the encoder/decoder.
- **Row Weight Deviation:** Reflects the timing balance and slack across the circuit.

$$value_{heuristic} = v_1 * eval_{curr} + v_2 * eval_{pred} \quad (6)$$

$$eval_{curr} = \omega_1 * SUM + \omega_2 * MAX + \omega_3 * DEVIATION \quad (7)$$

Simultaneously, the function estimates future feasibility by considering the number of remaining valid vectors in the candidate pool. By combining the evaluation of the current hardware metrics with the prediction of future search potential, the heuristic function effectively prunes the search space.

2) Vector-pool Pruning and Non-chronological Backtrack:

When the candidate vector results in a constraint violation (conflict), P-CDB employs two powerful, techniques to accelerate the search as depicted in Figure 3.

- **Vector-pool pruning.** It performs conflict analysis to identify the minimal set of previously placed vectors that are inconsistent with the current choice. Based on this analysis, vector-pool pruning is performed to remove not only the failing candidate but also all other vectors in the pool that would induce the identical conflict.
- **Non-chronological backtracking.** Instead of retreating chronologically to the previous level, P-CDB uses the identified conflict set to perform non-chronological backtracking. It intelligently jumps directly back to the highest decision level responsible for the conflict, effectively bypassing large, irrelevant portions of the search tree and eliminating futile exploration.

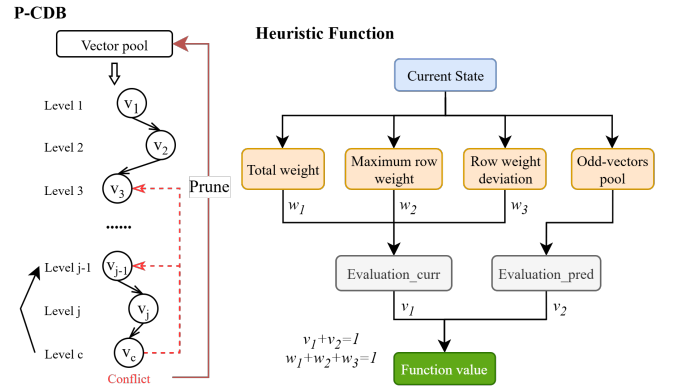


Fig. 3. BitVector Conflict-Driven Backtrack with Heuristic Function

3) **Restart Policy:** To enhance the robustness of the search and prevent it from getting trapped in unproductive regions of the solution space, P-CDB incorporates a restart policy. If it exhausts all search possibilities originating from the initial top- n sequences provided by P-MCTS without finding a solution, a restart is triggered, and P-MCTS is then reinvoked to generate the subsequent batch of n promising candidate sequences.

D. Local Search with Sliding-Window Based Boolean Constraints

A deeper analysis reveals that Constraint 2 is a widely-used heuristic to satisfy a more fundamental property: ensuring any three columns are linearly independent, which guarantees the minimum distance of 4 required for SEC-DED codes. To further reduce hardware overhead, certain high-weight odd vectors can be replaced with lower-weight even vectors by relaxing the strict odd-weight constraint and directly enforcing the more fundamental linear independence constraint instead. While globally searching this expanded solution space is intractable, performing a local search around the high-quality solution already found by P-CDB is computationally feasible.

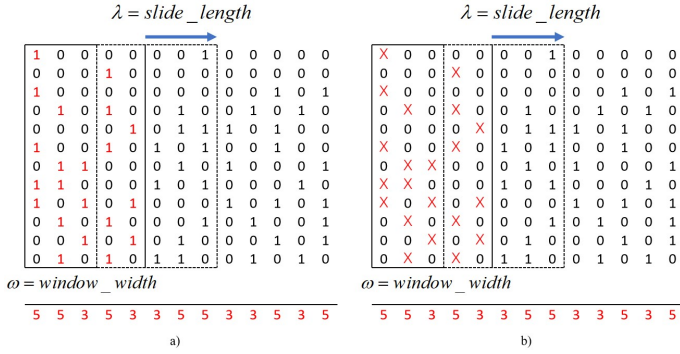


Fig. 4. Local Search with Sliding-window Based Boolean Constraints

To implement this local search, the full set of encoding principles is first translated into a constraint matrix C . Each column of C represents a single instance of a condition that, if met, would invalidate the check matrix H . These constraint vectors are systematically generated from an all-zero vector of length $n = k + r$ according to the forbidden patterns:

- Column Uniqueness ($h_i = h_j$). To check for a violation where two columns are identical, a constraint vector is created with 1s at positions i and j only.
- Linear Dependency ($h_i \oplus h_j \oplus h_k = \mathbf{0}$). To check for a linear dependency among any three columns, a constraint vector is created with 1s at positions i , j , and k .
- XOR Sum Uniqueness: To ensure the syndrome of a correctable adjacent-pair error ($h_i \oplus h_{i+1}$) is unique, it must not be identical to the syndrome of any other double-bit error. Such a collision is represented by a constraint vector with 1s at the four corresponding column indices.

The aggregation of all such vectors, representing every possible constraint violation for the given n , forms the complete constraint matrix C . This matrix is then used in the Boolean optimization model of Equation (8).

$$\min_H \sum_{i,j} H_{i,j} \quad (8a)$$

$$\text{subject to } H \cdot c_j \pmod{2} \neq \mathbf{0}, \quad \forall c_j \in C \quad (8b)$$

To overcome the computational complexity of this model for large data widths, a sliding-window approach is employed. As illustrated in Fig. 4, this technique divides the global optimization into a series of simpler, localized sub-problems that are solved iteratively based on a defined window size and step length.

IV. EVALUATION

A. Analysis of Algorithm Performance

To validate both the solving efficiency and the solution optimality of the proposed framework, a comprehensive benchmark against a suite of state-of-the-art solvers and classical methods is performed, including SMT solvers (CVC5 and Z3), SAT solvers (Kissat[19] and CaDiCaL[19]), and two widely-used heuristics (a genetic algorithm and a greedy algorithm).

TABLE I
PERFORMANCE OF SEVERAL SOLVERS

Scheme	Runtime(s)	Memory(MB)
CVC5	65.29	3332.33
Z3	71.94	3576.14
Kissat	291.32	1661.52
Cadical	191.83	5008.20
Genetic	<i>Unsolved</i>	<i>Unsolved</i>
Greedy	6.57	2755.20
MCTS-CDB	1.8	28.45

TABLE II
RESULTS SOLVED BY SOLVERS

Scheme	Total weight	Row Maximum	Row Deviation
CVC5	416	44	18
Z3	370	62	42
Kissat	414	42	13
Cadical	384	37	10
Greedy	270	27	8
MCTS-CDB	246	22	4

1) *Environment Setup*: All methods were tasked with constructing a check matrix for the $k = 64$, $r = 12$ instance, executed on an Ubuntu 20.04 virtual machine configured with 4 CPU cores and 8 GB of RAM. The results are evaluated with two categories of metrics, collected in Table I. Solver performance is measured by runtime and peak memory consumption, and solution quality is assessed with three hardware-centric criteria for the generated matrix: total weight (circuit area), maximum row weight (critical path delay), and row weight deviation.

2) *Results*: The results demonstrate the limitations of conventional solvers on this problem. As expected, SMT solvers outperform SAT solvers, as the latter's reliance on bit-blasting discards the problem's high-level structural information. Classical heuristics prove unreliable with the genetic algorithm failing to converge and the greedy algorithm producing suboptimal solutions.

As shown in Table I and Table II, the proposed framework achieves better performance in both runtime and solution quality. By performing conflict analysis and non-chronological backtracking directly at the BitVector level, it executes a complete and efficient search that consistently converges on high-quality, hardware-optimal matrices. This systematic process, guided by MCTS, ensures completeness while consistently converging on matrices with optimized hardware characteristics, and the necessity of this approach is proven on the most challenging instances ($k = 16, r = 8$ and $k = 64, r = 11$), where MCTS-CDB is the only solver to find a solution within the time limit. This confirms that the MCTS stage is a crucial component for solving these intractable cases.

B. Comparison with Existing Schemes

This section validates the hardware performance of the MCTS-CDB-generated codes against a comprehensive set of state-of-the-art SEC-DED-DAEC schemes. All designs were

TABLE III
SYNTHESIS RESULTS OF EXISTING SCHEMES

Scheme	Data Bits	Check Bits	Miscorrection Rate	Delay(ns)		Area(μm^2)		Power(mW)	
				Encode	Decode	Encode	Decode	Encode	Decode
Dutta[20]	32	7	53.40%	0.45	0.95	1174.66	1556.63	1.98	3.85
Reviriego[12]	32	7	55.20%	0.51	1.12	983.14	1673.94	1.82	3.24
Rahul[21]	32	7	5.13%	0.48	1.02	971.45	1795.50	1.91	5.36
Jun[13]	32	10	4.88%	0.48	0.91	1142.74	2042.88	1.86	2.80
Neale[22]	32	10	9.00%	0.5	0.85	977.28	1549.18	1.81	3.27
OLS:Reviriego[14]	64	24	0	0.35	0.45	1983.38	1178.06	4.76	6.56
OLS:Ahmed[15]	64	23	0	0.35	0.46	1994.54	1176.16	4.71	6.50
Proposed	32	10	0	0.36	0.83	1064.53	1472.31	2.75	3.60
		11	0	0.37	0.83	1070.65	1571.53	2.56	3.08
	64	11	0	0.44	0.95	2068.42	2696.71	4.84	8.50
	64	12	0	0.45	0.96	2076.93	2595.10	4.40	6.64

synthesized using Synopsys Design Compiler with a standard 45nm CMOS technology library. The comparative results for area, delay, and power are presented in Table III, leading to the following key insights.

1) *Comparison with Hsiao Code-Based Schemes:* The data reveals a clear trade-off in prior work. Traditional low-cost designs[20], while using only 7 check bits for 32-bit data, suffer from prohibitively high miscorrection rates. More advanced designs[21][13][22] reduce this rate to 5-9% but at the cost of increasing the check bits to 10. The proposed 10-bit solution not only achieves zero miscorrection rate but also demonstrates an average of 24.15% and 13.66% reduction in encoding and decoding delay respectively. While maintaining a comparable encoding area, it also reduces the decoding area by an average of 13.67%.

2) *Comparison with OLS Code-Based Schemes:* It is important to note that OLS-based codes are highly optimized for performance, featuring extremely low decoding delay and a compact decoder logic area. However, they provide a zero-miscorrection guarantee but with significant overhead. For 64-bit data, the OLS scheme requires 23 check bits, but the proposed codes achieve the same guarantee with only 11-12 check bits instead. Our method presents a strategic trade-off with a more complex decoder to break the rigid structural and data-width constraints of the OLS framework. The result is a dramatic reduction in check bit overhead, making our solution a more practical and scalable choice for area- and power-constrained ultra-reliable systems.

In summary, the synthesis results demonstrate that the MCTS-CDB framework can generate codes that are simultaneously correct (zero-miscorrection), efficient (fewer check bits and lower hardware overhead), and flexible (arbitrary data widths). This capability effectively addresses the long-standing trade-off within the design of high-reliability DAEC codes, making it highly suitable for modern ultra-reliable memory systems.

V. CONCLUSION

This paper introduces MCTS-CDB, a novel CAD framework that resolves the long-standing trade-off between flawed,

efficient Hsiao codes and correct but costly OLS codes for zero-miscorrection SEC-DED-DAEC design. Integrating a CDCL-inspired search with MCTS guidance and Boolean optimization, our framework systematically generates codes that are simultaneously correct, efficient, and flexible. Experimental validation against prior Hsiao-based codes confirms that this approach achieves zero-miscorrection with an average reduction of encoding and decoding delay by 24.15% and 13.66%, respectively. This work thus provides a practical, high-performance path to designing the ultra-reliable memory subsystems required by next-generation safety-critical applications.

APPENDIX

A. *H*-matrix for proposed (42,32) code

$$\begin{pmatrix} 10000000001010100100100101000100101000101 \\ 010000000000101010010010010010001001001001 \\ 001000000010010100100001000001010100010111 \\ 000100000010100010001010100100000010011010 \\ 000010000001010001010001001010100001010000 \\ 000001000000101000001000010100010101100110 \\ 000000100010000101010100001001000100001010 \\ 000000010010001010001001001000101010100101 \\ 000000001010000001000100101010010101011001 \\ 000000000101000000100010010101001010101101 \\ 0000000000101000000100010010101001010110110 \end{pmatrix}$$

B. *H*-matrix for proposed (75,64) code (Hexadecimal)

$$\begin{pmatrix} 80140c8515235561c54 \\ 401211224a474d05126 \\ 20094004ab32b2fc9f6 \\ 10142202c45062a2ad8 \\ 08124059211d2d2a53a \\ 040092a0015eb8dfd2 \\ 021121509484a354342 \\ 0108c429424ad4a482c \\ 00830a44929914a95e4 \\ 00443408a94ccb4a64a \\ 002889920aa10a1228a \end{pmatrix}$$

REFERENCES

1. *J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* Warrendale, PA: SAE International, Apr. 2021.
2. RTCA, Inc. *DO-254, Design Assurance Guidance for Airborne Electronic Hardware* tech. rep. RTCA/DO-254 (Radio Technical Commission for Aeronautics, Washington, D.C., Apr. 2000).
3. *ECSS-Q-ST-60-02C: Space product assurance - ASIC and FPGA development* Noordwijk, The Netherlands: ECSS - European Cooperation for Space Standardization, July 2013.
4. 3GPP. *Study on scenarios and requirements for next generation access technologies (Release 14)* tech. rep. TR 38.913 V14.3.0 (3rd Generation Partnership Project (3GPP), June 2017).
5. *ISO 26262-1:2018, Road vehicles – Functional safety – Part 1: Vocabulary* Geneva, CH: International Organization for Standardization, Dec. 2018.
6. Das, A. & Touba, N. A. A New Class of Single Burst Error Correcting Codes with Parallel Decoding. *IEEE Transactions on Computers* **69**, 253–259 (2020).
7. Maity, R. K., Samanta, J. & Bhaumik, J. An Improved Single and Double-Adjacent Error Correcting Codec with Lower Decoding Overheads. *Journal of Signal Processing Systems* **95**, 1–13 (May 2023).
8. Dong, M. *et al.* A Universal, Low-Delay, SEC-DECTAEC Code for State Register Protection. *IEEE Access* **10**, 57665–57673 (2022).
9. Sullivan, M. B. *et al.* Characterizing and Mitigating Soft Errors in GPU DRAM. *IEEE Micro* **42**, 69–77 (2022).
10. Kumar, K. N., Reddy, N. A., Shanmukh, P. & Vinodhini, M. *Matrix based Error Detection and Correction using Minimal Parity Bits for Memories* in *2020 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)* (2020), 100–104.
11. Liu, H., Xiao, L., Wang, T., Li, J. & Li, J. Error Correction Codes for Double Burst Errors Correction in Memories. *IEEE Access* **13**, 116621–116631 (2025).
12. Reviriego, P., Martínez, J., Pontarelli, S. & Maestro, J. A. A Method to Design SEC-DED-DAEC Codes With Optimized Decoding. *IEEE Transactions on Device and Materials Reliability* **14**, 884–889 (2014).
13. Jun, H.-y. & Lee, Y.-s. Single error correction, double error detection and double adjacent error correction with no mis-correction code. *IEICE Electronics Express* **10**, 20130743–20130743 (Oct. 2013).
14. Reviriego, P., Pontarelli, S., Evans, A. & Maestro, J. A. A Class of SEC-DED-DAEC Codes Derived From Orthogonal Latin Square Codes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **23**, 968–972 (2015).
15. Abood Ahmed, R. & Samsudin, K. Double Adjacent Error Correction Codes for Ultra-Fast Cache Memories. *IEEE Access* **13**, 36626–36636 (2025).
16. Bhargavi, C., Nishanth, D. V. R., Nikhita, P. & Vinodhini, M. *H-Matrix Based Error Correction Codes for Memory Applications* in *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)* (2021), 1–5.
17. Barbosa, H. *et al.* *cvc5: A Versatile and Industrial-Strength SMT Solver* in *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022* (eds Fisman, D. & Rosu, G.) **13243** (Springer, 2022), 415–442.
18. De Moura, L. & Bjørner, N. *Z3: An Efficient SMT Solver* in *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008, Proceedings* **4963** (Springer, 2008), 337–340.
19. Biere, A., Fazekas, K., Fleury, M. & Heisinger, M. *CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020* in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions* (eds Balyo, T. *et al.*) **B-2020-1** (University of Helsinki, 2020), 51–53.
20. Dutta, A. & Touba, N. A. *Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code* in *25th IEEE VLSI Test Symposium (VTS'07)* (2007), 349–354.
21. Rahul, K. & Yachareni, S. *Deterministic Algorithm to generate SEC-DED-DAEC H-Matrix for SRAMs in FPGAs for reliable space applications* in *2020 5th International Conference on Computing, Communication and Security (ICCCS)* (2020), 1–5.
22. Neale, A. & Sachdev, M. A New SEC-DED Error Correction Code Subclass for Adjacent MBU Tolerance in Embedded Memory. *IEEE Transactions on Device and Materials Reliability* **13**, 223–230 (2013).