Journal of VLSI circuits and systems
RESEARCH ARTICLE
Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 8, No. 1, 2026 (pp. 1-8)
WWW.VLSIJOURNAL.COM

An Efficient Error Resilient Ternary Content Addressable Memory Architecture

Sirisha Mallaiah[1], M. Vinodhini[2]*
[1,2]Department of Electronics and Communication Engineering, Amrita School of Engineering, Bengaluru, Amrita Vishwa Vidyapeetham, India.

KEYWORDS:
Ternary Content Addressable Memory (TCAM),
Soft error in data and parity bits,
2D parity,
3D parity,
Hamming Codebased Error Correction,
Error Correction Capability (ECC)

ARTICLE HISTORY:
Received: 19.09.2025
Revised: 16.10.2025
Accepted: 05.11.2025

DOI:
https://doi.org/10.31838/JCVS/08.01.01

Abstract
High-speed memories like ternary content addressable memory (TCAM) are widely employed in high-throughput search applications like network routers. Using application-specific integrated circuits (ASIC) to construct TCAM memories allows for a higher search rate at the expense of increased power and resource requirements. However, safeguarding the TCAM from soft errors while maintaining good search speed and minimizing critical path time is a difficult task. In this paper, we present a TCAM architecture with a multipumping technique that incorporates the correction of multiple bits using the Hamming code. Different sizes, such as 4×4, 16×8, and 32×16, of the proposed TCAM architecture are simulated and implemented in 45 nm technology. The proposed work evaluates the TCAM in comparison to the look-up table (LUT) based on a priority encoder in TCAM architecture, two-dimensional (2D) parity, three-dimensional (3D) parity, and Hamming code-based error correction techniques with a multiplexer block in TCAM architecture. The results demonstrate that the suggested TCAM has a lesser delay compared to the LUT-based and higher error correction capability, including the parity bits.

Authors' e-mail ID: sirishamallaiah@gmail.com and m_vinodhini@blr.amrita.edu

Authors' orcid ID: 0009-0009-5443-2793; 0000-0002-3718-2122

How to cite this article: Sirisha Mallaiah and M. Vinodhini, An Efficient Error Resilient Ternary Content Addressable Memory Architecture, Journal of VLSI Circuits and System, Vol. 8, No. 1, 2026 (pp. 1-8).

Introduction

Content addressable memory (CAM) enables concurrent searches of stored data in only one cycle, resulting in great search performance. Only "0" and "1" can be stored and sought in a binary CAM. Ternary content addressable memory (TCAM) stores information using three different representations: 1, 0, and X.[1-5] Its search operation is so fast that it can complete the search operation in just one clock cycle, making it useful in high-speed applications like networking, radar signal tracking, data compression image processing, etc. TCAM is the inverse of random-access memory (RAM) because it does not access data in the same way that RAM does, by providing a precise memory address to where the data are stored. While RAM requires an address to read data, TCAM takes data as input and sends back an address.[6-12]

Single event upsets (SEUs) in SRAM-based field-programmable gate arrays (FPGAs) refer to the phenomenon where a single ionizing particle or radiation strike can cause a temporary or permanent change in the state of a memory cell within the SRAM configuration memory of the FPGA.[13,14] This change can lead to errors in the functionality of the FPGA, which can be particularly problematic in safety-critical applications such as aerospace, automotive, and medical devices.[15] TCAM uses SRAM because of its quick access times, capacity for parallel comparisons, and low standby power consumption. It allows rapid data retrieval in real time and performs flexible pattern-matching tasks.[14] Large capacity entries can be stored by TCAMs because of the high integration density of SRAM, which is also one of the reasons why it is prone to soft errors which can be either single-bit or multi-bit upsets.[16] To address these concerns, many error detection and correction techniques such as SEC-DED,[17] look-up table, 2D parity,[18] and 3D parity are present. The conventional TCAM takes the shape of a

2D array, with cells in the same row sharing a matching line and cells in the same column sharing a select line.[19] Though it provides accurate output for smaller designs, there is exponential memory expansion for larger designs making them bulky, complex, and expensive. Therefore, for efficient resource utilization, the multipumping- enabled multiported SRAM-based TCAM is implemented in FPGA.[20] The multipumping technique effectively doubles the number of ports on a dual-ported SRAM module by timing the SRAM block internally as integral multiples of the system's external clock. The TCAM is organized into subblocks, each of which is timed at a rate *p* times higher than the system clock.[20] The output of each subblock is then passed to the priority encoder after the search key is provided. To determine the final address, the outputs from each priority encoder are fed into an overall priority encoder (OPE), which chooses the lowest address line. The flip-flop storage of the matching logic is replaced by the rules with LUTs, and partial reconfiguration (PR) capabilities of the FPGA are used to alter the rules, thereby improving the efficiency of the implementation.[17],[21] SRAM-based TCAMs are prone to soft errors, and data bits can get corrupted that must be recovered.[22] Error detection and correction processes came into the picture to retrieve the data bits that were corrupted because of soft errors. Figure 1 implements a 4×6 TCAM size using two 8×4 SRAM blocks with a multipumping factor of 2.[21,23] For error checking, even parities and legal weights (the number of ones in each column) are computed and compared to the current parities; if there is a discrepancy, an error is generated. *P* represents a parity bit. Intrinsic redundancy will be used for error correction.

A soft-error-resilient SRAM-based TCAM using a multipumping technique and binary encoded TCAM LUT for error detection and correction is presented in references.[24,25] To achieve TCAM, SRAM blocks can be used in both FPGAs and ASICs. For instance, 1×1 TCAM may be produced with the help of 2×1 RAM. If a "0" is present in the 1×1 TCAM, logic 1 is stored in RAM [0], and if a "1" is present, logic 1 is put in RAM.[1] If "X" is in TCAM, then both RAM [0] and RAM [1] will be overwritten with "1". A TCAM may be constructed from N bits of SRAM input data using an SRAM of 1-bit width with 2N locations. Using M-bit wide SRAM in 2N locations, it is possible to create a TCAM with a size of M by N. A TCAM of size 4×4 is divided into two parts, which will be implemented using two 4×4 SRAMs, as shown in Figure 2.[24]-[26]

For example, when 1001 is entered into TCAM as the search key, the values stored in the position "10" of the SRAM to the left and the position "01" of the SRAM to the right will be sent into the AND gate, as shown in Figure 2(b). A physical address for the search key may be obtained by using the AND operation, with

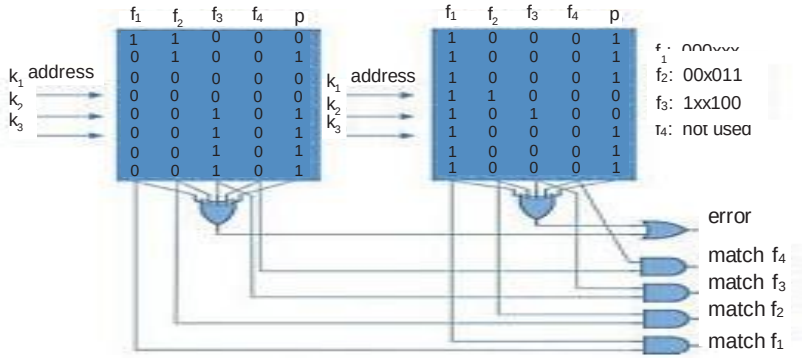

Fig. 1: Parity protected TCAM using two SRAM blocks

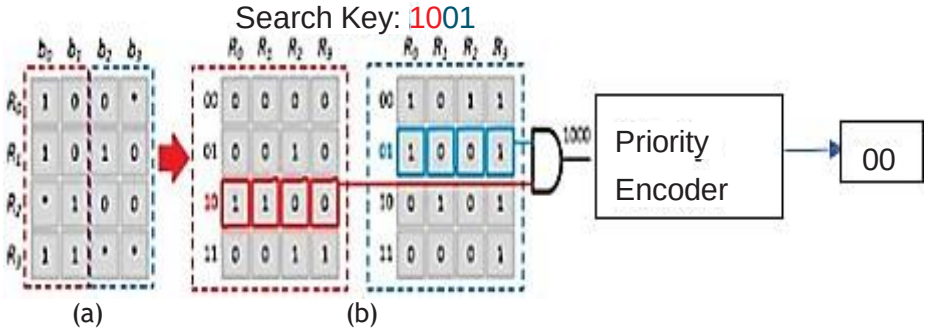

Fig. 2: Implementation of 4×4 TCAM using SRAM blocks. (a) TCAM of size 4×4. (b) Using two 4×4 SRAM blocks to implement 4×4 TCAM and providing the address of the search key using a priority encoder

the resultant " 1000" being transmitted to the priority encoder. To get the parity bits, TCAM reads position 01 in the left-sided SRAM and location 10 in the right-sided SRAM when the search key is 0110.[24] Parity is calculated for each row and all the parity bits put together denote the error signal. The error signal is then given to the encoder which gives the output as 110. MSB bit 1 represents the SRAM at which the error is present, while the LSB bit 10 represents the row that consists of the corrupted bit. The values of reconfigurable LUT are encoded by replacing the TCAM values as follows: 0 - 00, 1 - 01 and x - 10. By looking for the matching value concerning the LSB bits in the LUT, the correction vector can be computed. The whole row in which the error is present is replaced by the correction vector to restore the faulty bit. A single bit of data can be detected and corrected.

A TCAM architecture is introduced where the priority encoder in reference[24] is replaced with a mux block in reference[27] to efficiently make use of the power by dynamically switching on and off of the muxes. Hence, the result of the AND gate 1000 is given as input to the mux block. In the first phase, only one of the available multiplexers will be activated, reducing power consumption. The first stage multiplexer chooses the match status line with the lowest address among the logic "1" lines. The main mux in the second stage chooses the output of the first stage mux whose enable signal is 1, as depicted in Figure 3.[27] The use of 2D and 3D parity for error detection and correction, which enables the correction of multiple bits, is employed in references.[27,28] In a 2D technique, the horizontal and vertical parities are calculated. Horizontal parity is computed using an EXOR operation for rows, while vertical parity is computed using a column-wise EXOR operation.[27]

The horizontal and vertical parities will be recalculated and compared to previously obtained values during the search process. If there is a discrepancy between the two, it means there is a bug in the SRAM.

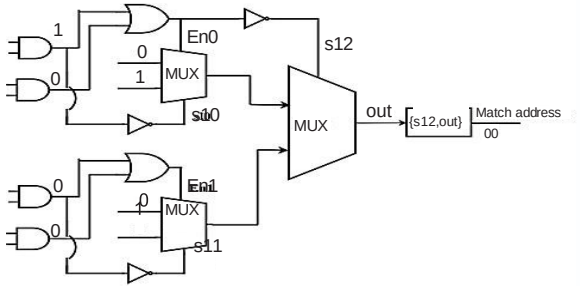

Fig. 3: Multiplexer block

The condition will aid in identifying the source of the mistakes. The syndrome can be determined by performing an EXOR on the parities that have been stored and the parities that have been calculated during the search process. The NOT operation will reset the traced error bit. Three-dimensional parity involves the calculation of horizontal, vertical, and diagonal parities to perform error correction and detection.[27] The HP and VP are computed by EXORing row-wise and column-wise, respectively, while the diagonal parity is computed using an EXOR operation on all data bits in the diagonals. The search process involves recalculating and comparing with the existing parities. During the search operation, the syndrome is determined by performing an EXOR operation on the parities that have been stored and those that have been calculated.

In the left SRAM block, the calculated horizontal, vertical, and diagonal parities are 0010, 1000, and 0100000, while in the right SRAM block, they are 1111, 0011, and 0001001. This results in the syndrome being calculated, as it means there are errors present in the blocks of SRAM. Horizontal, vertical, and diagonal syndromes of 0110, 0101, and 0001100, respectively, convey there are two errors in the left SRAM section. Similarly, 0110, 0101, and 0001100, respectively, are the horizontal, vertical, and diagonal syndromes of the right block of SRAM, and this means that there are two errors in the right SRAM block. The errors can be fixed by performing an inversion operation on the faulty bits.

This paper proposes a novel error detection and correction technique for TCAM architecture. The Hamming code technique is used, which helps in detecting and correcting one error in a row, either the data bit or the parity bit. Hence, the design ensures data integrity and enhances system reliability, making it particularly relevant for mission-critical search applications. The findings of this study offer valuable insights into the design and efficacy of TCAM architectures, especially in mitigating multiple soft errors, optimizing power consumption, and preserving search speed.

Further, the paper presents the following details. The section "proposed TCAM architecture using Hamming code for error detection and correction" discusses the proposed Hamming code–based error detection and correction technique for the TCAM architecture using a mux block. The section "Simulation Results and Performance Analysis" discusses the performance analysis of the proposed and the existing TCAM design. The section "Conclusion and Future Scope" deals with the conclusion and future scope.

Proposed TCAM Architecture using Hamming Code for Error Detection and Correction. The TCAM architecture with the MUX block, which dynamically switches ON and OFF, is adopted as there is a reduction in the power. In this design, the data bits are encoded using the Hamming technique before they are stored in SRAM. The generated parity bits are also stored in SRAM. Because of radiation and noise effects, the bits stored in SRAM alter their value, which introduces soft errors. To introduce the soft errors in the design, the bits stored in SRAM are randomly flipped, and this flipping is done to both data and parity bits. By flipping the bits, the soft error bits are injected in the design, and by using the Hamming techniques, the error bits are detected and corrected to validate the proposed technique. Soft errors not only affect the stored data bits but also might corrupt the stored parity bits. The Hamming code technique is proposed for error detection and correction, which not only checks for errors in data bits but also detects and corrects parity bit errors.[29,30]

To detect and fix errors, this algorithm guarantees that there are enough parity bits to cover all the data bits and the parity bits are placed into the code at predetermined locations, usually every two places (1, 2, 4, 8, etc.). The Hamming code uses a total of seven bits in a row with four actual data bits and three parity bits, as shown in Figure 4. The values of the parity bits are P 1 = D3 ⊕ D5 ⊕ D7, P 2 = D3 ⊕ D6 ⊕ D7, and P 4 = D5 ⊕ D6 ⊕ D7 while the values of the syndrome, which helps in correcting the errors, are given by S1 = D3 ⊕ D5 ⊕ D7 ⊕ P 1, S2 = D3 ⊕ D6 ⊕ D7 ⊕ P 2, and S4 = D5 ⊕ D6 ⊕ D7 ⊕ P 4.

Each row of the SRAMs is given its own error detection and repair syndrome, and hence three parity bits are generated for each row, as seen in Figure 5. Therefore, this technique aids in finding and fixing the corruption of a single bit in each row. So, counting both SRAMS, it can detect and repair eight bits of error.

For example, a two-bit error is introduced in each SRAM; one data bit and one parity bit, as in Figure 6. Now the recalculated parities for the left SRAM are 101, 101, 001, and 110 and for the right SRAM are 001, 001, 101, and 011, respectively.

By comparing the received code with the expected code based on the parity bits, the syndrome can be determined. The syndrome pattern pinpoints the location of the error bit, enabling it to be fixed as in Figure 7. The syndrome values for the left SRAM are 101, 000, 000, 010 and for the right are SRAM 000, 101, 001, and 000. Based on the syndrome values, the first row-third column intersection bit and P2 of the fourth row in the left SRAM and second row-third column intersection bit, and P1 of the third row of the right SRAM are corrupted bits. The main advantage of the Hamming code is that the error correction capability is increased and also the soft error in parity bits is addressed.

D7	D6	D5	P4	D3	P2	P1

Fig. 4: Position of data and parity bits

Simulation Results and Performance Analysis

The coding of the three existing designs[14,16] and the proposed design is done using Verilog HDL, and the Xilinx Vivado Design Suite is used for simulation. These four TCAM designs are also implemented in various sizes like 4×4 with multipumping factor 2, 16×8 multipumping with factor 4, and 32×16 with multipumping factor 8. The simulation results of the proposed Hamming code design of size 4×4 with multi pumping factor as 2 are shown in Figures 8 and 9. Totally, eight errors are introduced, three each in each SRAM and two parity bits.

Figure 8 shows that the data are loaded into the 4×4 TCAM, which is implemented using two 4×4 SRAMs. The

D7	D6	D5	D3	P4	P2	P1
0	0	0	0	0	0	0
0	0	1	0	1	0	1
1	1	0	0	0	0	1
0	0	1	1	1	1	0

D7	D6	D5	D3	P4	P2	P1
1	0	1	1	0	0	1
1	0	0	1	1	0	0
0	1	0	1	1	0	1
0	0	0	1	0	1	1

Fig. 5: Hamming code protected SRAM blocks implementing 4×4 TCAM

D7	D6	D5	D3	P4	P2	P1
0	0	1	0	0	0	0
0	0	1	0	1	0	1
1	1	0	0	0	0	1
0	0	1	1	1	0	0

D7	D6	D5	D3	P4	P2	P1
1	0	1	1	0	0	1
1	0	1	1	1	0	0
0	1	0	1	1	0	0
0	0	0	1	0	1	1

Fig. 6: Multiple errors in parity bits and SRAM blocks implementing 4x4 TCAM

S4	S2	S1	Error Bit
0	0	0	No Error
0	0	1	P1
0	1	0	P2
0	1	1	D3
1	0	0	P4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Fig. 7: Syndrome Based Error Correction

search key is 1100; the 11 rows of the left SRAM and the 00 rows of the right SRAM are considered for the search operation. Parity bits for each row are represented by "r" and syndrome values by "s>" Figure 9 shows the values of the SRAMs, in which errors are introduced and finally corrected based on the syndrome. It can be seen that at each signal line, a single-bit error is injected and corrected.

The synthesis of the existing and proposed designs is performed using the logic synthesis Cadence Genus tool at 45 nm technology and TSMC standard cell library. As per this library, the process is fast, the voltage is 1.1V, and the temperature is −40 degrees Celsius. The analysis is done using the performance metrics, namely, area, power, critical path delay, and error correction capability for different sizes—4×4, 16×8, and 32×16 TCAM. Table 1 shows the area, power, and critical path delay of 4×4 TCAM for the existing designs (LUT-based (i), 2D parity (ii), and 3D parity (iii)) and the proposed design (Hamming code). Similarly, Tables 2 and 3 show the performance comparison for the TCAM architectures of size 16×8 and 32×16, respectively.

From Table 1, it is observed that the proposed design for 4×4 has a lower area compared to the existing designs. With the LUT-based, 2D parity, and 3D parity designs, there is a reduction of 28.54%, 3.79%, and 27.82%, respectively.

It also shows that for the power comparison, LUT-based, 2D parity, and 3D parity techniques have higher power consumption actually proposed. They utilize more power than the Hamming code–based 4×4 TCAM architecture. There is a reduction of power of about 16.71%, 3.93%, and 24.47%, respectively. The power consumption is higher for 2D and LUT-based designs. The proposed design has a lesser critical path delay compared to other techniques and the reduction percentage is 5.08%, 11.54%, and 6.92%, respectively.

Tables 2 and 3 depict the values for 16×8- and 32×16-sized TCAMs, respectively. There is a gradual increase in area but a trade-off between power and timing can be noticed. For higher sizes, the proposed design has a lesser area than 3D parity–based but more than the other architectures, as the computation of parity bits increases as the size increases. The stored parity bits can also be affected because of soft errors, which can be detected and corrected in the proposed design. The proposed work has a higher error correction capability than the existing designs.[14,16]

In 4×4, 16×8, and 32×16 TCAM architectures, the word size is 4, 8, and 16, respectively. The proposed techniques can correct a single-bit error per word. In 4×4, 16×8, and 32×16 TCAM architectures, 8, 64, and 256 words can be stored explicitly. Hence, in these architectures, 8, 64, and 256 total error bit correction is

Table 1: Performance Metrics Values of Existing and Proposed 4×4 TCAM Designs

Parameter	I[12]	Existing work II[13]	III[13]	Proposed work IV
Area (um^2)	1450.066	1076.993	1435.542	1036.135
Power (mW)	27.563	22.054	30.392	22.956
Critical path delay (ps)	3133	3362	3195	2974
No. of parity bits	8	16	30	24
Error correction	1-bit detection	6-bit detection	6-bit detection	8-bit detection
capability	and correction	2-bit correction	and correction	and correction
Parity bits correction	No	No	No	Yes

Table 2: Performance metrics values of existing and proposed 16×8 TCAM designs

Parameter	I[12]	Existing work II[14]	III[14]	Proposed work IV
Area (um^2)	8427.342	8644.954	11188.285	10726.778
Power (mW)	155.729	164.178	61.731	63.397
Critical path delay (ps)	7551	3304	4361	6612
No. of parity bits	16	80	156	192
Error correction	1-bit detection	12-bit detection	12-bit detection	64-bit detection
capability	and correction	4-bit correction	and correction	and correction
Parity bits correction	No	No	No	Yes

Table 3: Performance metrics values of existing and proposed 32×16 TCAM designs.

Parameter	I[12]	Existing work II[14]	III[14]	Proposed work IV
Area (um^2)	32273.235	32947.273	44472.081	42112.411
Power (mW)	151.163	169.745	231.037	239.393
Critical path delay (ps)	8487	8019	6351	6796
No. of parity bits	32	288	568	768
Error correction	1-bit detection	24-bit detection	24-bit detection	256-bit detection
capability	and correction	16-bit correction	and correction	and correction
Parity bits correction	No	No	No	Yes

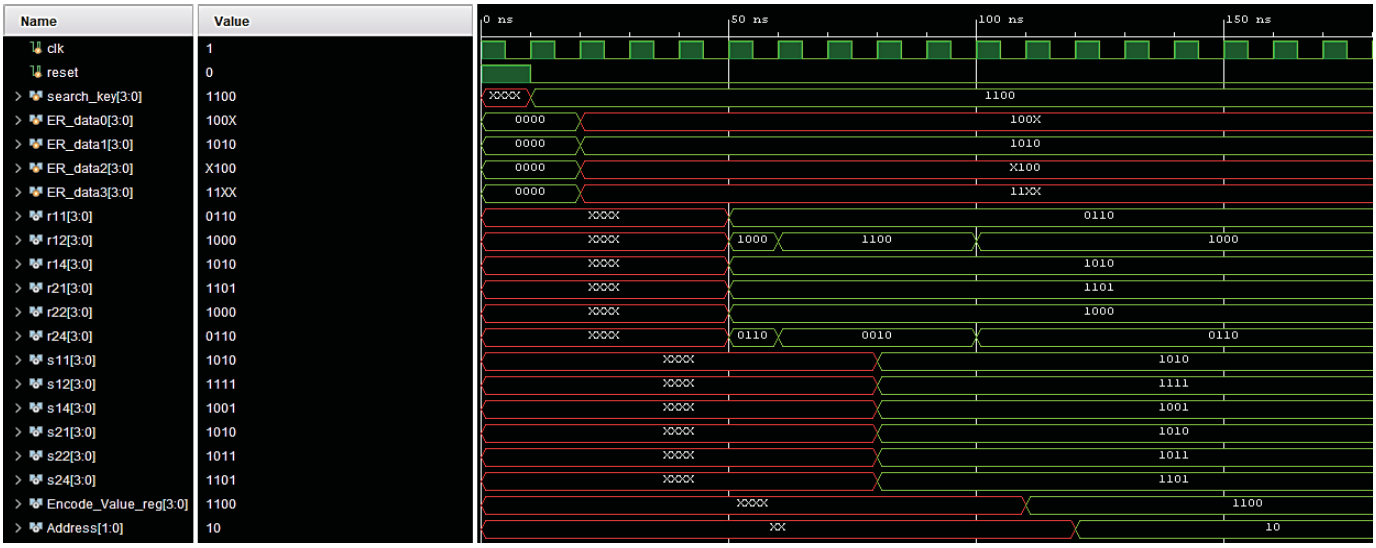

Fig. 8: Loading of data into the TCAM, giving the search key, parity, and syndrome calculation, getting the address as the output at the end and correcting the corrupted parity bits as well

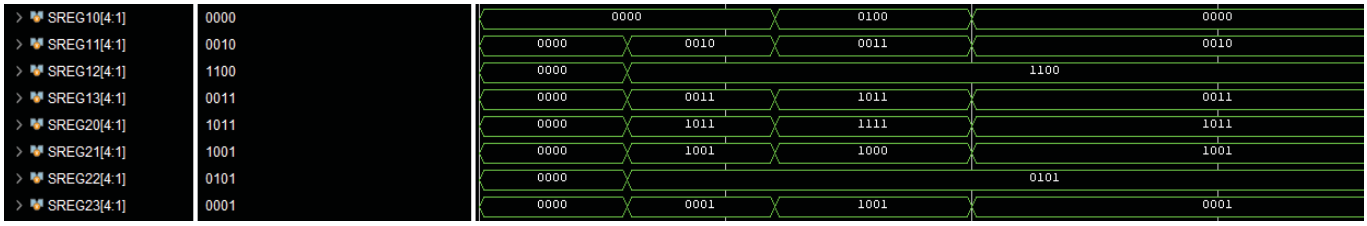

Fig. 9: Error introduction and correcting the corrupted data bits

possible if an error occurs in a word. If an error occurs in the parity bit of that corresponding word, instead of the word itself, even that kind of error can be corrected by this technique. For increasing word size, the error correction capability can be improved with an increase in the number of parity bits. Further, the area occupied increases with a higher word size when compared to 2D and LUT-based techniques. With increased word size, when comparing the proposed technique with 2D and LUT-based techniques, if power increases, the critical path delay decreases and vice versa. So, there is a trade-off between power and critical path delay. Therefore, the proposed TCAM architecture outperforms in terms of error correction capability with increasing word size at the cost of either power or delay.

Conclusion and Future Scope

A soft error prevention approach with multipump operation is given. The TCAM architecture uses a multiplexer block,[16] which reduces the power consumption. The error detection and correction are done using a syndrome-based Hamming code. The existing TCAM architectures, such as priority encoders with LUT-based, 2D parity, and 3D parity- based designs, are implemented and compared with the proposed technique.

The proposed technique has better timing compared to the LUT-based technique for all three sizes—4×4, 16×8, and 32×18 with a decrease of 5.07%, 12.45%, and 19.92%, respectively. The 3D parity-based design is larger in area than the proposed Hamming code technique for all the sizes—4×4 (27.82%), 16×8 (4.12%), and 32×16 (5.30%). For larger sizes, the area increases as the computation of parity bits increases. Also, there is a trade-off between critical path delay and power consumption. The proposed technique detects soft errors in both data bits and the stored parity bits, correcting one bit in each row of the SRAM, thereby having higher correction capability. Further, there exists potential for enhancing error correction capability concerning both data and parity bits while minimizing area, power, and critical path delay.

Acknowledgement

The authors are grateful to Sri Mata Amritanandamayi Devi (AMMA), Chancellor, Amrita Vishwa Vidyapeetham, for her inspiration and for providing financial support for the article processing charge (APC) of this publication.

References

1. Ternary Content Addressable Memory (TCAM) Search IP for SDNet V1.0; Xilinx Product Guide PG190, Xilinx, San Jose, CA, USA, Nov. 2017.
2. Kim, S. M., et al. (2023). A digital processing in memory architecture using TCAM for rapid learning and inference based on a spike location dependent plasticity. IEEE Access, 11, 3416–3430, https://doi.org/10.1109/ACCESS.2023.3234323.
3. Hsin-Tsung, L., & Pi-Chung W. (2023). TCAM-based packet classification for many-field rules of SDNs. Computer Communications, 203, 89–98. https://doi.org/10.1016/j.comcom.2023.03.001.
4. Garzoón, E., Lanuzza, M., Teman, A., & Yavits, L. (March 2023). AM4: MRAM crossbar based CAM/TCAM/ACAM/AP for in-memory computing. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 13(1),408–421. https://doi.org/10.1109/JET- CAS.2023.3243222.
5. Reviriego, P., Levy, G., Kadosh, M. & Pontarelli, S. (September 2022). Algorithmic TCAMs: implementing packet classification algorithms in hardware. IEEE Communications Magazine, 60(9), 60–66. https://doi.org/10.1109/MCOM.001.2100923.
6. Irfan, M., Yantır, H. E., Ullah, Z., & Cheung, R. C. C. (June 2022). Comp-TCAM: An adaptable composite ternary content-addressable memory on FPGAs. IEEE Embedded Systems Letters, 14(2),63–66. https://doi.org/10.1109/LES.2021.3124747.
7. Wan, Y., Song, H., & Liu, B. (March 2022). GreedyJump: A fast TCAM update algorithm. IEEE Networking Letters, 4(1),25–29. https://doi.org/10.1109/LNET.2021.3074148.
8. Bazzi, J., Sweidan, M. E. Fouda, R. Kanj, & Eltawil, A. M. (2024). Variability-aware design of RRAM-based analog CAMs. IEEE Access, 12,55859–55873. https://doi.org/10.1109/AC- CESS.2024.3388730.
9. Vakili, S., & Zarei, A. (June 2024). DSCAM: Latency-guaranteed and high-capacity content-addressable memory on FPGAs. IEEE Embedded Systems Letters, 16(2),182–185. https://doi.org/10.1109/LES.2023.3334288.
10. öztekin, H., Lazzem, A., & Pehlivan, ı. (2023). Using FPGA-based content-addressable memory for mnemonics instruction searching in assembler design. The Journal of Supercomputing, 79,17386–17418. https://doi.org/10.1007/s11227-023-05357-2
11. Aniruddh, A., & Prabhu, E. (2023). Error Resolving in SRAM Emulated FPGA using CRC. Third International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, 2023, pp. 647–651. https://doi.org/10.1109/ICSCCC58608.2023.10176518
12. Swaroop, B.S., et al. Implementation of successive cancellation polar decoder architecture. (2023). IEEE 20th India Council International Conference (INDICON), Hyderabad, India, 2023, pp. 103–108. https://doi.org/10.1109/INDICON59947.2023.10440946.
13. Li, T., Liu, H., & Yang, H. (Feb. 2019) Design and characterization of SEU hardened circuits for SRAM-based FPGA. IEEE Transaction on Very Large Scale Integrated (VLSI) Systems, 27(6)1276–1283.
14. Li, T., Yang, H., Zhao, H., Wang, N., Wei, Y., & Jia, Y. (2019). Investigation into SEU effects and hardening strategies in SRAM based FPGA. 17th European Conference on Radiation Effects on Components and Systems (RADECS), pp. 1–5.
15. Ramos, , Toral, , Reviriego, P., & Maestro, (Mar. 2019). An ALU protection methodology for soft processors on SRAM-based FPGAs. IEEE Transaction on Computers, 68(9), 1404–1410.
16. Keller, M., & Wirthlin, (2019). Impact of soft errors on largescale FPGA cloud computing. Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 272–281.
17. Likhith, R. S. R., Sooraj, R., Sasikamal, V., & Vinodhini, M. (2025). Optimized reliable content addressable memory. 8th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, pp. 399–403. https://doi.org/10.1109/ICOEI65986.2025.11013533.
18. Rao, Q., & He, C. A new 2-D parity checking architecture for radiation hardened by design SRAM. Asia Pacific Conference on Postgraduate Research in Microelectronics Electronics (Prime Asia), pp. 360–363, https://doi.org/10.1109/PRIMEASIA.2009.5397372, 2009.
19. Ullah, Z., Jaiswal, M. K., Chan, Y. C., & Cheung, (2012). FPGA Implementation of SRAM-based Ternary Content Addressable Memory, IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, Shanghai, pp. 383–389. https://doi.org/10.1109/IPDPSW.
20. Ullah, Z., & Lee, J. A. (2018). Efficient TCAM design based on multi pumping-enabled multiported SRAM on FPGAA. IEEE Access, 6, 19940–19947.
21. Reviriego, P., Ullah, A., & Pontarelli, S. (Aug. 2019). PR-TCAM: Efficient TCAM emulation on Xilinx FPGAs using

partial reconfiguration. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, 27(8),1952–1956.
22. Reviriego, P., Pontarelli, S., & Ullah, A. (Feb. 2019). Error detection and correction in SRAM emulated TCAMs. IEEE Transaction on Very Large-Scale Integration (VLSI) Systems. 27(2), 486–490.
23. Al-Yateem, N., Ismail, L., & Ahmad, M. (2024). A comprehensive analysis on semiconductor devices and circuits. Progress in Electronics and Communication Engineering, 2(1), 1–15. https://doi.org/10.31838/PECE/02.01.01
24. Ullah, I., Yang, J., & Chung, J. (April 2020). ER-TCAM: A soft-error-resilient SRAM based ternary content-addressable memory for FPGAs. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, 28(4), 1084–1088.
25. Santhosh and Sonali Agrawal. (2020). Multi pumping-enabled multi ported SRAM based Efficient TCAM design. 4th International Conference on Electronics Materials Engineering and Nano-Technology (IEMENTech), pp. 1–6.
26. Rahim, R. (2024). Scalable architectures for real-time data processing in IoT-enabled wireless sensor networks. Journal of Wireless Sensor Networks and IoT, 1(1), 44–49. https://doi.org/10.31838/WSNIOT/01.01.07
27. Varada, & Agrawal, S. (2021). An efficient SRAM-based ternary con tent addressable memory (TCAM) with soft error correction. 5th International Conference on Electronics, Materials Engineering Nano-Technology (IEMENTech), Kolkata, India, pp. 1–6. https://doi.org/10.1109/IEMENTech53263.2021.9614696
28. Neelima, K., & Subhas, C. (2020). Efficient adjacent 3D parity error detection and correction codes for embedded Memories. IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1–5. https://doi.org/10.1109/CONECCT50063.2020.9198452.
29. S. Kumar R T, Y. N and R. S R. (2024). Rectifying the impact of multiple cell upsets in memory devices. 2024 International Conference on Computer, Electronics, Electrical Engineering and their Applications (IC2E3), Srinagar Garhwal, Uttarakhand, India, pp. 1–5. https://doi.org/10.1109/IC2E362166.2024.10827113.
30. Nakul, K. S., Reddy, M. K. C., Abhiram, P., & Vinodhini, M. (2024). Row-wise Hamming code for memory applications. 5th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, pp. 30–34. https://doi.org/10.1109/ICESC60852.2024.10689934.
31. Muralidharan, J. (2024). Optimization techniques for energy-efficient RF power amplifiers in wireless communication systems. SCCTS Journal of Embedded Systems Design and Applications. 1(1), 1–6. https://doi.org/10.31838/ESA/01.01.01
32. Kavitha, M. (2023). Beamforming techniques for optimizing massive MIMO and spatial multiplexing. National Journal of RF Engineering and Wireless Communication, 1(1), 30-38. https://doi.org/10.31838/RFMW/01.01.04
33. Yang, C. S., Lu, H., & Qian, S. F. (2024). Fine tuning SSP algorithms for MIMO antenna systems for higher throughputs and lesser interferences. International Journal of Communication and Computer Technologies, 12(2), 1–10. https://doi.org/10.31838/IJCCTS/12.02.01