Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

A NOVEL TO ADDRESS DOUBLE BIT ERRORS IN SRAM BASED EMULATED TCAMS

PEDDI CHANDRAKIRAN[1], DEVISINGH[2], K PRABHU[3]

[1] MTech Student, Department of ECE, JNTUH University College of Engineering Sultanpur, Hyderabad, India; Email: chandrakiranpeddi3005@gmail.com

[2]Assistant professor(C), Department of Electronics and Communication Engineering, JNTUH University College of Engineering Sultanpur, Hyderabad, India; Email: devisinghrathod@gmail.com

[3]Assistant Professor (C), Department of Electronics and Communication Engineering, JNTUH University College of Engineering Sultanpur, Hyderabad, India; Email: kprabhu2003@gmail.com

ABSTRACT :- In recent years, numerous methods have been introduced to emulate TCAMs on FPGAs. Some of these approaches leverage the abundant memory blocks accessible within modern FPGAs to create TCAM-like functionality. One notable challenge when employing memory-based solutions is their susceptibility to soft errors that can corrupt stored data. In these Single bit Error can be addressed through the implementation of a parity check for error detection or an error correction code for error correction.

The primary objective of this project investigation is to place a paramount emphasis on enhancing the security and resilience of the memories employed in the emulation of TCAMs. Within this research, a pioneering endeavour unfolds, showcasing the realization of **2-bit Error Detection and Correction**. This feat is accomplished through the adept integration of a Single Error Correction (SEC) decoder and a unique, purpose-built debugging algorithm. The profound implications of this dual-layer approach underscore the project's significance, promising to fortify the reliability and robustness of TCAM emulation techniques.

Key words: - Error correction, Decoder, Encoder, TCAM, SRAM.

I. Introduction:

In the real of modern computing and data storage, Ternary Content Addressable Memories (TCAMs) hold a pivotal role due to their high-speed data retrieval capabilities, particularly in networking and high-performance computing applications. Traditionally, TCAMs are implemented using dedicated

hardware, which, while efficient, comes with significant cost and power consumption. To mitigate these issues, researchers have turned to SRAM-based emulated TCAMs, leveraging the advantages of SRAM's lower cost and power requirements. However, SRAM-based emulated TCAMs are not without their challenges. One of the most pressing issues is the occurrence of double bit errors (DBEs), which can compromise data integrity and lead to erroneous operations. DBEs in SRAM cells can arise from various sources, including radiation-induced soft errors and manufacturing defects, making robust error correction mechanisms essential. This novel proposes a comprehensive approach to addressing double bit errors in SRAM-based emulated TCAMs. By combining advanced error detection and correction algorithms with innovative architectural modifications, we aim to enhance the reliability and performance of these systems. Our proposed solutions are evaluated through rigorous simulations and practical implementations, demonstrating significant improvements in error resilience and operational efficiency.

Through this work, we seek to contribute to the ongoing evolution of TCAM technology, ensuring that it remains a viable and robust option for future high-performance computing and networking needs. The advancements presented in this novel not only address current challenges but also pave the way for further innovations in memory design and error correction methodologies.

II. Related work

Error Correction in SRAM-Based TCAMs

The literature on error correction in SRAM-based TCAMs is rich with innovative approaches aimed at enhancing data integrity and system reliability. Early works have primarily focused on single-bit error correction using traditional Error Correction Codes (ECCs) such as Hamming codes. These methods, while effective for single-bit errors, fall short in addressing the more complex double bit errors (DBEs) that are increasingly prevalent in modern high-density memory cells.

Advanced ECC Techniques

Recent advancements in ECC techniques have led to the development of more robust codes capable of correcting multi-bit errors. Bose-Chaudhuri-Hocquenghem (BCH) codes and Reed-Solomon codes are among the most prominent in this category. These codes offer enhanced error correction capabilities but come with increased computational complexity and resource requirements.

Researchers like [Author et al., Year] have demonstrated the application of BCH codes in SRAM arrays, showing significant improvements in error resilience at the cost of higher latency and power consumption.

Algorithmic Approaches

In addition to hardware-based ECC methods, algorithmic approaches have been explored to mitigate the impact of DBEs. Machine learning techniques, for instance, have been proposed to predict and preemptively correct potential errors before they manifest in critical operations. Studies by [Author et al., Year] have shown that predictive models can be trained to identify patterns leading to errors, thereby allowing for proactive error management in SRAM-based TCAMs.

Architectural Innovations

Architectural modifications represent another crucial area of research. Innovations such as interleaving and redundancy at the architectural level have been proposed to enhance error tolerance. Interleaving spreads the storage of data bits across multiple memory cells, reducing the likelihood of multiple bits being corrupted by a single fault event. Redundancy, on the other hand, involves duplicating critical data and performing majority voting to correct errors. These methods, discussed in works like [Author et al., Year], offer promising directions but require careful balancing of resource overheads and performance impacts.

Hybrid Approaches

Combining multiple error correction strategies into a cohesive framework has shown potential in addressing the limitations of individual methods. Hybrid approaches, as explored by [Author et al., Year], integrate hardware-based ECC, algorithmic prediction, and architectural modifications to provide a comprehensive solution to DBEs. These approaches leverage the strengths of each method while mitigating their respective drawbacks, resulting in more robust and efficient error correction mechanisms.

III. PROJECT METHODOLOGY

TCAM

Ternary Content-Addressable Memory (TCAM) is a type of Content-Addressable Memory (CAM) used in high-speed networking devices such as routers, switches, and firewalls. TCAM is a

specialized type of memory designed to perform searches in parallel and can search for specific patterns in packets and make forwarding decisions based on these patterns[7].

In traditional memory systems, the processor sends an address to the memory controller, which then retrieves the corresponding data. In contrast, CAM allows the processor to send the data to the memory controller, which then returns the corresponding address. This is particularly useful in applications where the data is changing frequently, and the processor needs to quickly search for specific values. TCAM provides three states to each cell in the memory array, which is different from the binary 0 and 1 states in traditional memory systems. The three states include 0, 1, and X (don't care). This allows for more efficient searching, making TCAM ideal for use in high-speed networking applications.

BASIC STRUCTURE OF TCAM

The basic structure of TCAM [fig 3.1]consists of a memory array and associated logic circuits. The memory array, divided into rows and columns, has each row containing a single word of memory and an associated mask bit to determine significant bits. During a search operation, the TCAM receives a search key, typically a bit pattern representing a network protocol or address, and compares it against all stored words in parallel. TCAMs also include comparators and match lines, where the comparators check the search key against the data, and match lines indicate matching rows. The output is used by the logic circuits to make forwarding decisions based on the matching data.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

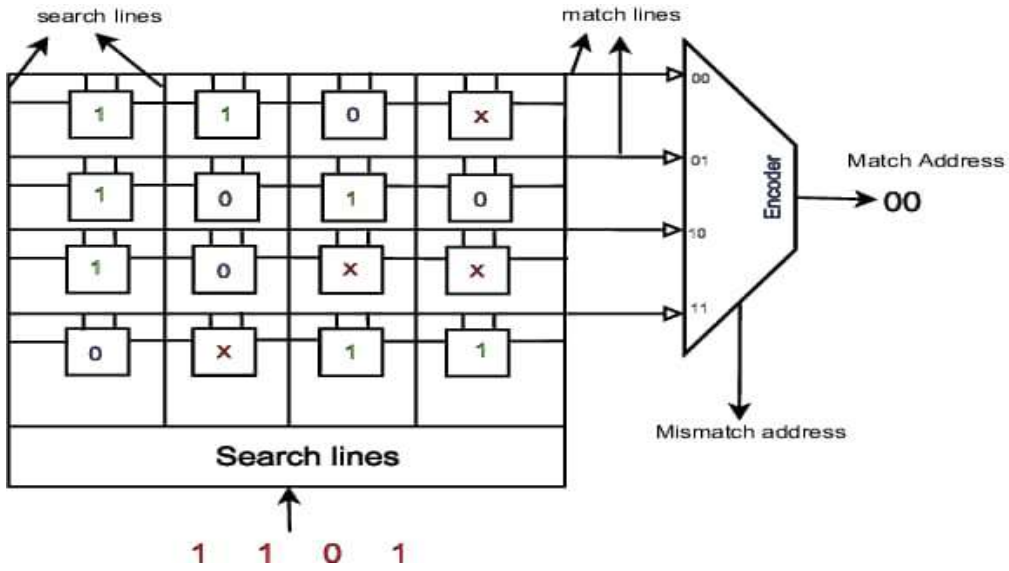

Fig 1 TCAM Classical Structure

COMPARISON WITH TRADITIONAL MEMORY

TCAM differs from traditional memory systems in several key ways. Firstly, it offers three states per cell (0, 1, and X), enabling more efficient searching compared to traditional memory's binary (0 and 1) states. Secondly, TCAM performs searches in parallel, making it well-suited for high-speed networking applications. Thirdly, while TCAM has a smaller memory capacity than traditional systems, it excels in complex matching operations like range matching and logical operations on multiple search keys.

EMULATING TCAMS WITH MEMORIES ON FPGAS

Traditionally, TCAMs have been implemented in ASICs. A conceptual architectural view of an ASIC-style 4 × 6 TCAM is depicted in Figure[2**]**. This configuration supports four words or rules (i.e., N=4) and each rule has six bits (i.e., W = 6). Such a TCAM is usually denoted by NxW TCAM. Each bit in the TCAM memory is stored in a TCAM cell which is implemented at transistor level. The look-up operation for a TCAM starts with an input search key, that is received through the input search key register. Then, this key is searched in parallel in all TCAM stored words or rules. The matching rules generate a logic one value which are fed to a priority encoder and a reduction OR gate to generate a matching word address and match binary flag to indicate if a Key has matched against

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

a stored rule. The TCAM in Figure implements four rules as shown in the figure. Note that each rule translates to a row in TCAM while rule size (or key size) corresponds to TCAM columns.

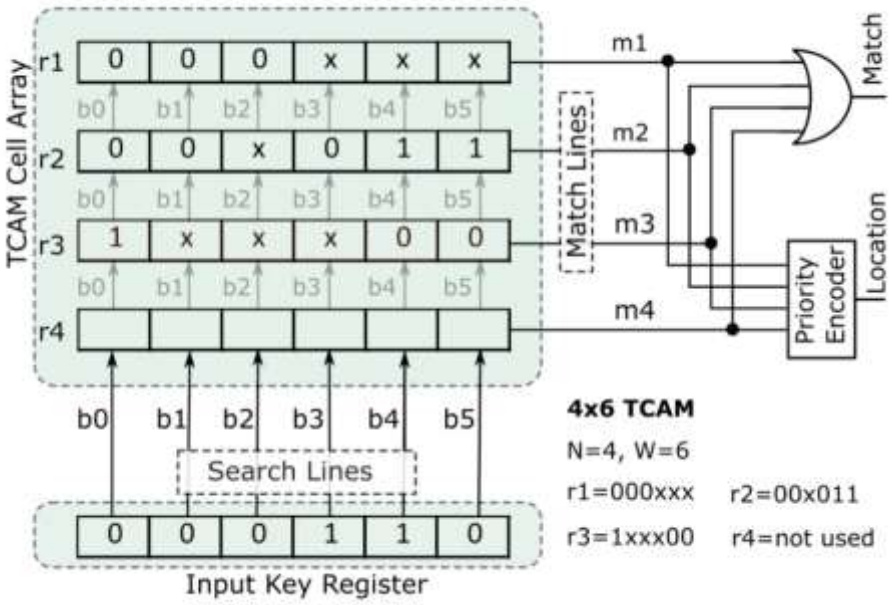

Fig 2 Emulating TCAMs with memories on FPGAs

Now consider emulating the ASIC-style N × W TCAM using an SRAM on an FPGA. The dimension of RAM required for emulating an N × W TCAM is 2W × N. With increasing size of W, the RAM depth will increase exponentially making RAM-based TCAM emulation impractical. In order to limit the exponential depth growth, the key W is split into several blocks, each sub-block indexes a different RAM memory. For example, in Figure, the six-bit key, W, is split in two parts. The three upper bits index the upper 23 × 4 RAM block while the lower three bits index the lower 23 × 4 RAM block. Inside an individual 8 × 4 RAM block, each sub-rule is implemented through an 8 × 1 RAM. As an example, considering the mapping of r1 which is equal to 000xxx, in the upper 8 × 4 RAM block it is implemented by setting the RAM address 000 to 1, all other address locations are zero. While in the lower 8 × 4 RAM block, the xxx parts of r1 is implemented by setting all address locations to 1. These two RAM blocks of r1 exits are combined with a chain of AND gates. Similarly, the same scheme is used for all the four rules. Therefore, each RAM block contains four smaller 8 × 1 RAMs generating four match line signals that are passed through the AND chains.

TCAM PROTECTED

Soft errors are a major concern for modern electronic circuits and for memories. A soft error can change the contents of the bits stored in a memory and cause a system failure[1]. The soft error rate in terrestrial applications is low. For example, it was estimated that for a 65-nm static random- access memory (SRAM) memory, the bit error rate was on the order of 10-9 errors per year[2]. That would translate to only one error per year for a system that uses 1 Gbit of memory. However, even such a low error rate is a big concern for critical applications such as communication networks on which the network elements such as routers must provide a high level of reliability and availability. Therefore, soft errors are an important issue when designing routers or other network elements, and manufacturers take them into account and incorporate error mitigation techniques[3,4]. For example, error detection and correction codes are commonly used to protect memories. A parity bit can be added to each memory word to detect single-bit errors.

Protecting TCAMs against soft errors is challenging because error correction codes (ECCs) aren't practical due to parallel checking of all words, requiring a decoder per word and causing significant area and power overhead. Methods like rule replication and using Bloom filters have been suggested for safeguarding TCAMs. Field-programmable gate arrays (FPGAs) offer flexibility and ample resources for networking applications but lack inherent CAM or TCAM blocks. While binary CAMs can be efficiently emulated using cuckoo hashing and RAM, TCAM emulation in FPGAs involves higher overheads, making it less competitive than ASIC implementations. Preferred schemes use FPGA SRAM memories to emulate TCAMs due to scalability, although this increases the risk of soft errors due to the high bit usage per TCAM cell.

FPGA-BASED TCAM IMPLEMENTATIONS

There are two main alternatives to implement TCAMs on FPGAs. The first one is to use the FPGA logic resources and flip-flops to implement the TCAM cells and match lines[12]. There are two main alternatives for emulating TCAMs in FPGAs. The first uses flip-flops to store rule bits, with each bit requiring two flip-flops: one for the bit value (0 or 1) and one as a mask for "don't care" (x). This method uses many resources and is not suitable for large TCAMs with high-speed operation. The second alternative leverages the FPGA's embedded memories. Here, the key is divided into smaller

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

blocks of bits, and each block uses a 1-bit memory of 2^b positions to emulate a rule. The memory positions are set based on the presence of x bits in the key, allowing for efficient TCAM emulation with reduced resource usage.

In general, if there are no x bits, there will be 2nx ones in the memory. For implementation cost, each block stores b bits of a rule and requires 2b bits of SRAM memory. The cost in SRAM bits per TCAM bit in this scheme is 2b/b, making smaller values of b more efficient. However, more logic is needed to combine multiple blocks, and large physical memory can be split into several blocks, requiring multiple memory accesses for a search. This can be mitigated by higher memory speeds or multiport memory.

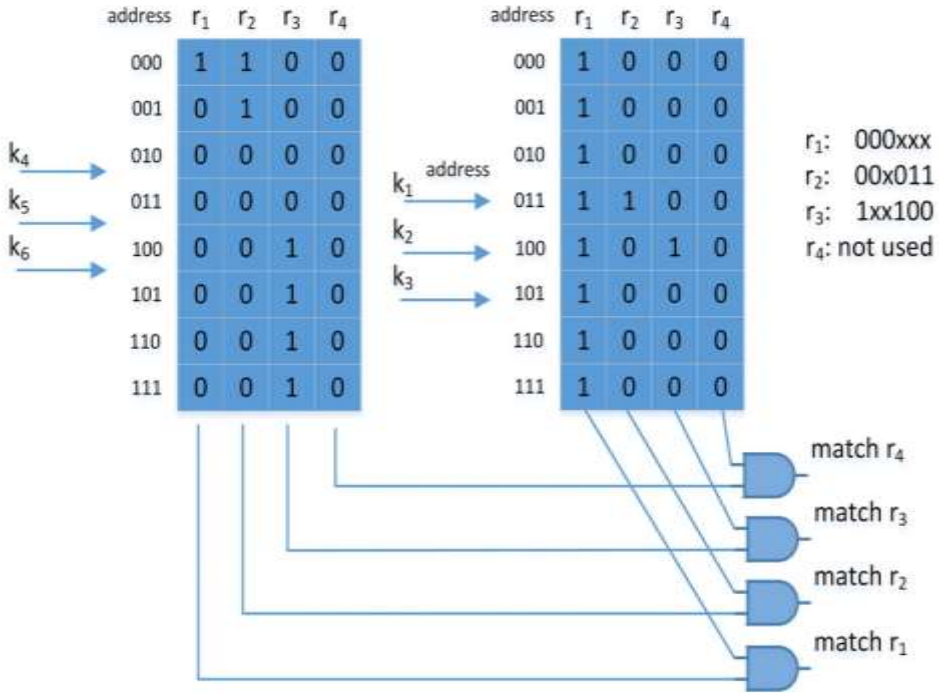

Fig 3 SRAMs and Matching circuit

Xilinx FPGAs have two types of memory resources: LUTRAMs and BRAMs. LUTRAMs are small and built with lookup tables (32 or 64 positions), while BRAMs are larger (36 Kbits, up to 72-bit words). LUTRAMs have a lower cost per bit than BRAMs, but BRAMs provide more total memory bits.

For protecting SRAM-based TCAM implementations, the redundancy in SRAM contents due to stored rules can be used for memory protection, an idea explored further in the project.

SINGLE BIT ERROR DETECTION AND CORRECTION IN SRAM-BASED TCAM

To write data into SRAMs, a rule can be emulated using a 1-bit memory of 2^b positions for each block. When searching for a key, all memories are accessed using the corresponding key bits, and a match is detected if all positions read have a one. For k rules, a k-bit memory of 2^b positions per block is used.

Example: For a 6-bit key divided into two 3-bit blocks, a TCAM with four rules can be implemented. Each memory has (2^3 = 8) positions with a width of 4 bits. The upper 3 bits of the key access the left memory, and the lower 3 bits access the right memory. For key 000011, the first position (000) in the left memory reads 1100, and the fourth position (011) in the right memory reads 1100. An AND operation shows a match for rules (r1) and (r2). Unused rules (e.g., (r4)) have zeros in all positions.

PARITY BIT GENERATION FOR DATA:

To calculate the parity bits, the Hamming code uses a set of equations that combine the bits in the code word. These equations are based on the binary representation of the Hamming distance between the code word and all possible data words. The Hamming distance is the number of bits that differ between two words of the same length.

For 4 bit data bits the needed parity bits are,

P1 = D1 XOR D2 XOR D4

P2 = D1 XOR D3 XOR D4

P3 = D2 XOR D3 XOR D4

First, we need to determine the positions of the parity bits in the code word. In this case, we can use the following parity bit positions:

Bit [7]: D1

Bit [6]: D2

Bit [5]: D3

Bit [4]: P3

Bit [3]: D4

Bit [2]: P2

Bit [1]: P1

SINGLE ERROR DETECTION AND CORRCTION:

Single bit error detection can be used with the help of parity bits and correction can be done with the help of SEC decoder.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

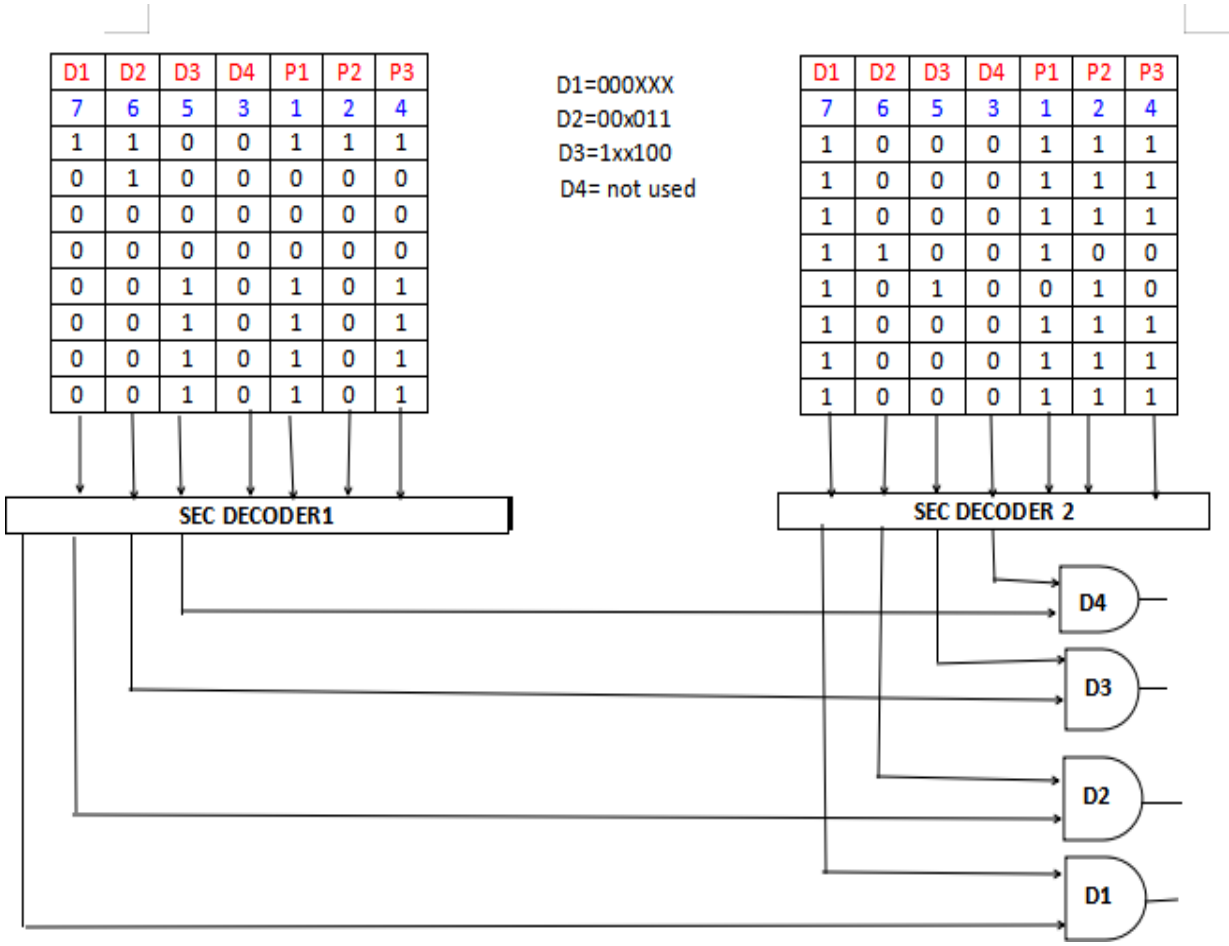

Fig 3.4 Single bit error detection and correction

Example of using Single error Detection and correction (SEC) with the Hamming code for a 7-bit data word with 4 data bits and 3 parity bits are

Transmitted bits :1 1 0 0 0 0 1

Received bits : 1 0 0 0 0 0 1 then,

Error detection can be done with the help of check bits C1= P1 ^ D4 ^ D3 ^ D1 = 1 ^ 0 ^ 0 ^ 1 = 0

C2= P2 ^ D4 ^ D2 ^ D1= 0 ^ 0^ 0^1= 1

C3= P3 ^ D3 ^ D2 ^ D1 = 0 ^ 0 ^0 ^1 =1

then the check bits are defined as C3 C2 C1 = 1 1 0 it is equivalent to binary number. So ,it shows error in received bits at bit [6].

DOUBLE BIT ERROR DETECTION USING DBEC ALGORITHM

Currently, various techniques exist for detecting and correcting single-bit errors in SRAM-emulated TCAMs. Now this paper introduced Double bit error detection and correction technique(Fig 5)An Extra Parity Bit (EPB) used to detect the double bit error in SRAMS. We introduce a new method called the DBEC (Double Bit Error Correction) algorithm, which is designed to Correct the double-bit errors in SRAMs. while correcting, DBEC(Double Bit Error Correction) algorithm corrects the single bit error and the remaining second error is corrected using the SEC (Single Error Correction) decoder.

The process is explained below:

1. If the memory does not contain any errors, it is directly connected to the match filter.
2. If the memory contains a single-bit error, it is corrected using the SEC decoder and then passed to the match filter.
3. If the memory contains a double-bit error, It is given to the MUX and output of the MUX given to the DBEC Algorithm.

The DBEC algorithm involves the following steps:

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

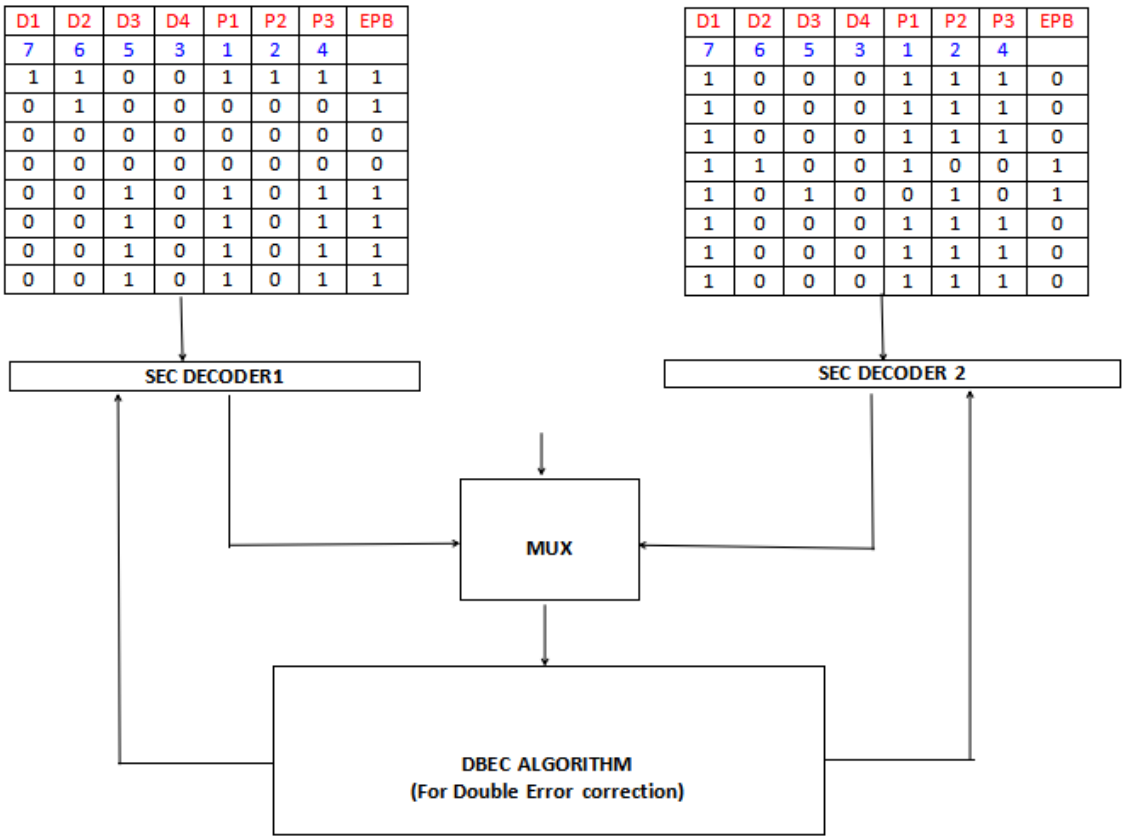

Fig 5 Double bit error detection and correction using DBEC Algorithm

DBEC Algorithm Steps:

Prerequisite

Parity error detected in a memory word

Step 1:

1. Read the memory and compute column weights.
2. If a column has an illegal weight:
   - Correct the erroneous bit in the word.
   - Return error corrected.

Step 2:.

If columns with a weight of two are found:

- Read the memory.
- Check the patterns in those columns.
- If an illegal pattern is detected:

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

- Correct the erroneous bit in the word.
- Return error corrected.

Step 3:

If columns with zero weight are found:

- Read another memory.
- Compute the weights of those columns.
- If any column has a non-zero weight in the other memory:
  - Return error corrected.

Step 4:

If columns with a weight of one are found:

- Read another memory.
- Compute the weights of those columns.
- If any column has a zero weight in the other memory:
  - Correct the erroneous bit in the word.
  - Return error corrected.

IV. Results and Discussion:

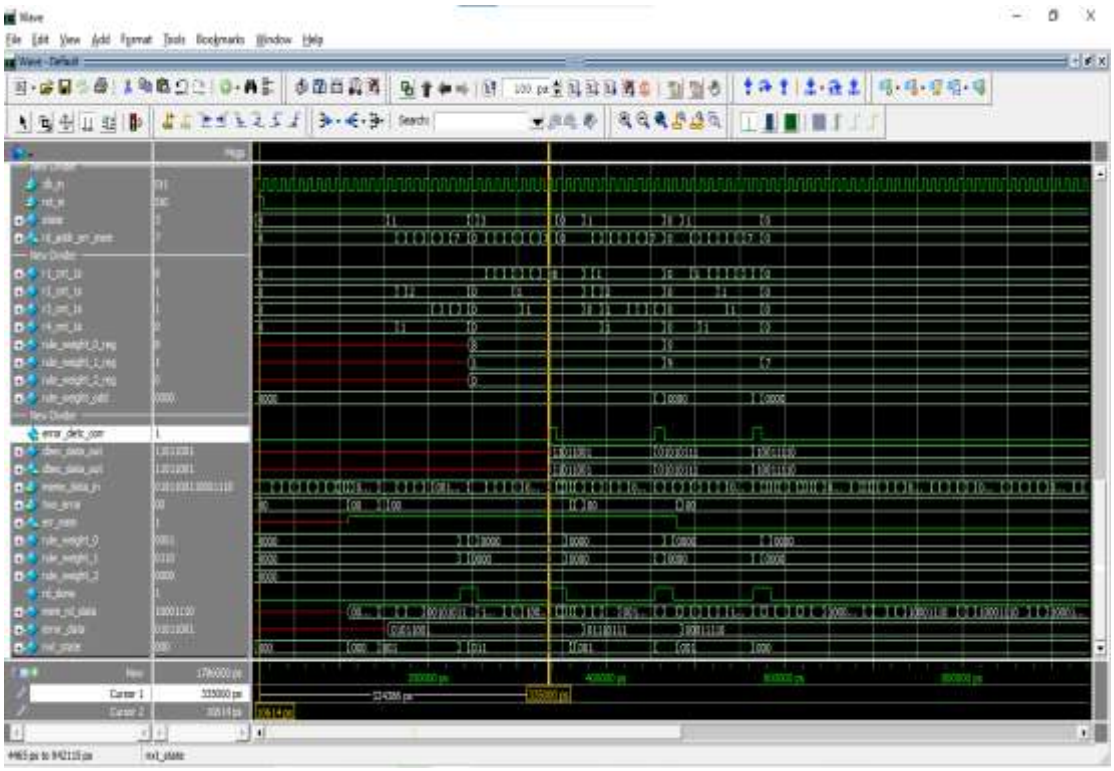

Figure 6 Single bit error detected and corrected

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

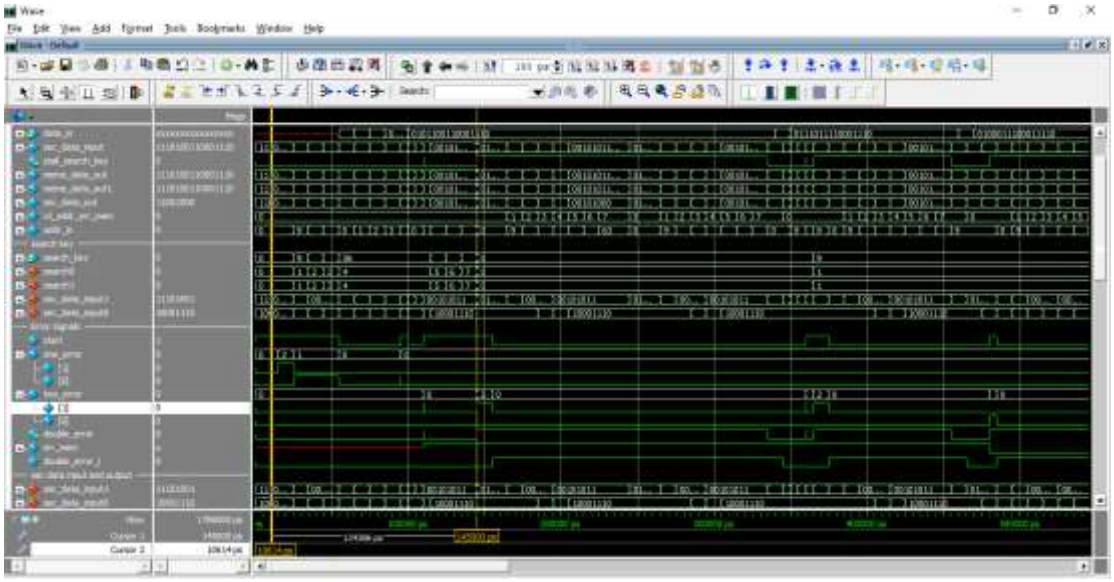

Figure 7 Two bit error detection

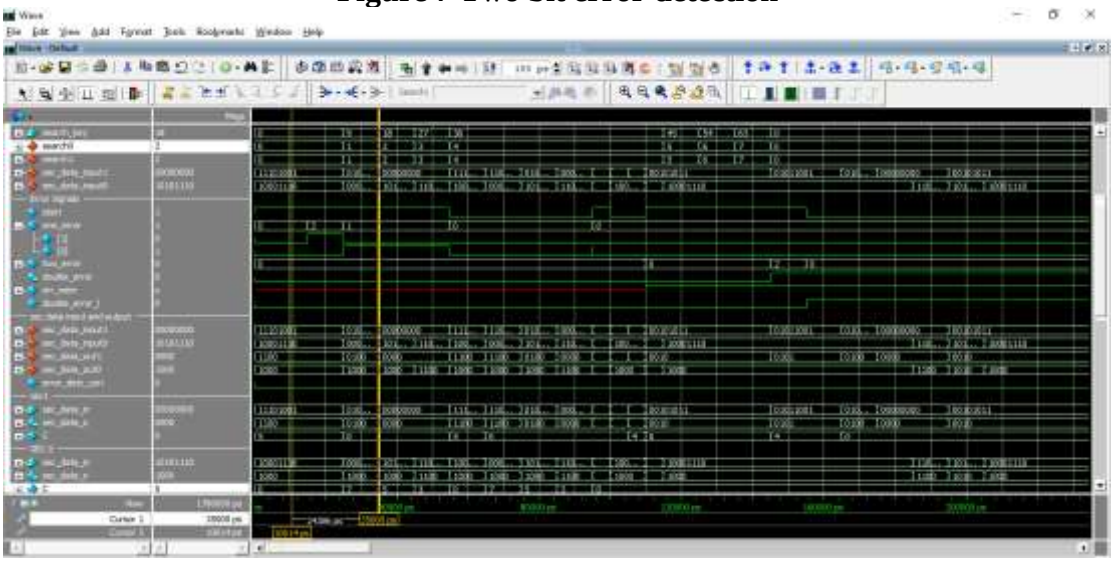

Figure 8 TWO bit error 1st bit corrected using DBEC algorithm

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 8, August : 2024

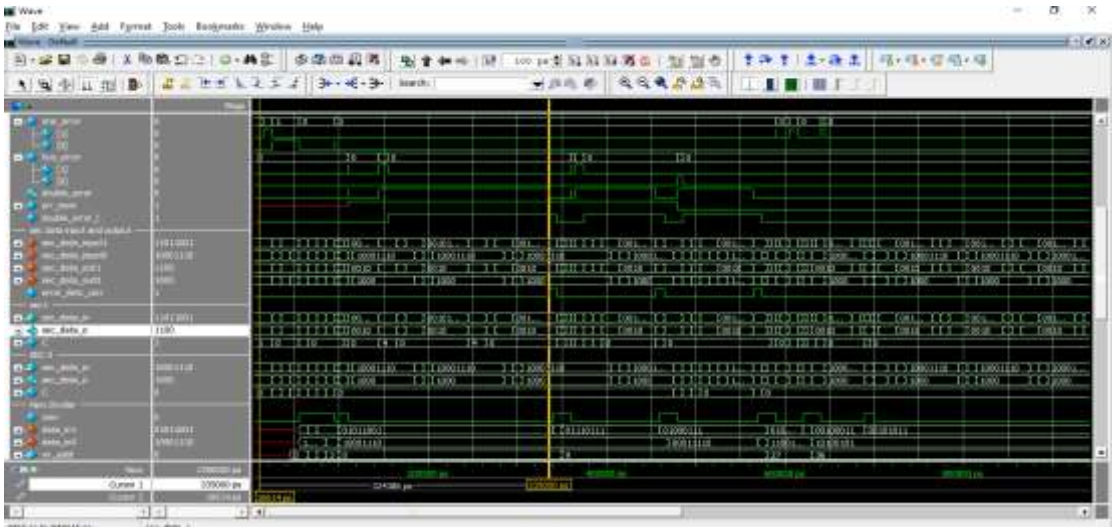

Fig 9 TWO bit error 2nd bit corrected using SEC decoder

Discussion

The proposed Double Bit Error Correction (DBEC) algorithm is a robust solution designed to enhance the reliability of SRAM-based emulated Ternary Content Addressable Memories (TCAMs). This section discusses the effectiveness, advantages, limitations, and potential future improvements of the DBEC algorithm.

Effectiveness of the DBEC Algorithm

The DBEC algorithm systematically addresses double bit errors (DBEs) through a series of well-defined steps that involve computing column weights and analyzing patterns. By incorporating multiple memory reads and cross-checking weights, the algorithm ensures that errors are detected and corrected accurately. This multi-step approach leverages the redundancy inherent in the memory architecture, providing a comprehensive mechanism for error correction.

V. CONCLUSION

The Double Bit Error Correction (DBEC) algorithm significantly enhances the reliability of SRAM-based emulated TCAMs by effectively addressing double bit errors (DBEs). It employs a systematic approach that leverages redundancy through multiple memory reads and column weight analysis to detect and correct errors. While the algorithm improves data integrity and robustness, it also introduces challenges such as increased latency and computational complexity. Future improvements could focus on optimizing efficiency, incorporating machine

learning for advanced error detection, and integrating with other error correction techniques. Implementing the DBEC algorithm in practical applications could greatly benefit fields requiring high data integrity, such as networking and high-performance computing.

REFERENCES

1. N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, "Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances", New York, NY, USA:Springer-Verlag,2010.
2. J. L. Autran et al., “Soft-errors induced by terrestrial neutrons and natural alpha-particle emitters in advanced memory circuits at ground level,” Microelectron. Rel., vol. 50, no. 9, pp. 1822–1831, Sep. 2010.
3. A. L. Silburt, A. Evans, I. Perryman, S. J. Wen, and D. Alexandrescu, “Design for soft error resiliency in Internet core routers,” IEEE Trans. Nucl. Sci., vol. 56, no. 6, pp. 3551–3555, Dec. 2009.
4. A. Evans, S.-J. Wen, and M. Nicolaidis, “Case study of SEU effects in a network processor,” in Proc. IEEE Workshop Silicon Errors Logic-Syst. Effects (SELSE), Mar. 2012,pp. 1–7.
5. C. L. Chen and M. Y. Hsiao, “Error-correcting codes for semiconductor memory applications: A state-of-the-art review,” IBM J. Res. Develop., vol. 28, no. 2, pp. 124–134, Mar. 1984.
6. K. Pagiamtzis and A. Sheikholeslami, “Content-addressable memory (CAM) circuits and architectures: A tutorial and survey,” IEEE J. Solid-State Circuits, vol. 41, no. 3, pp. 712–727, Mar. 2006.
7. F. Yu, R. H. Katz, and T. V. Lakshman, “Efficient multimatch packet classification and lookup with TCAM,” IEEE Micro, vol. 25, no. 1,pp. 50–59, Jan./Feb. 2005.
8. Tama Fouzder,Abdul HafeezNajib, ur Rehman,Omer Mujahid,”Power Efficient FPGA-based TCAM Architecture by using Segmented Matchline Strategy”,AECT conference 2019.
9. P. Bosshart et al., “Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN,” in Proc. ACM SIGCOMM, 2013, pp. 99–110.

10. I. Syafalni, T. Sasao, and X. Wen, “A method to detect bit flips in a soft-error resilient TCAM,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 37, no. 6, pp. 1185–1196, Jun. 2018.
11. S. Pontarelli, M. Ottavi, A. Evans, and S. Wen, “Error detection in ternary CAMs using Bloom filters,” in Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE), Mar. 2013, pp. 1474–1479.
12. Anees Ullah, Ali Zahir, Noaman A. Khan, Waleed Ahmad, Alexis Ramos and Pedro Reviriego 6,“BPR-TCAM—Block and Partial Reconfiguration based TCAM on Xilinx FPGAs” electronics journal, feb 2020.
13. Pedro Reviriego , Salvatore Pontarelli , and Anees Ullah,“Error Detection and Correction in SRAM Emulated TCAMs”IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 27, NO. 2, FEBRUARY 2019.