SOFT ERROR DETECTION AND CORRECTION IN SRAM-BASED TCAM ARCHITECTURES

C. SUCHARITHA[1], B. SREEKANTH REDDY[2]
[1]PG Student, Dept of ECE, G. NARAYANAMMA INSTITUTE OF TECHNOLOGY AND SCIENCE, FOR WOMEN, Hyderabad, TG, India.
[2]Assistant Professor, Dept of ECE, G. NARAYANAMMA INSTITUTE OF TECHNOLOGY AND SCIENCE, FOR WOMEN, Hyderabad, TG, India.

ABSTRACT: Ternary Content Addressable Memories (TCAMs) are built using Static Random Access Memory (SRAM). It plays a vital role for high-speed packet classifications in networking applications like software defined networking (SDN) as well as OpenFlow applications. TCAMs are generally realized as standalone devices combined into application-specific integrated circuits (ASICs). However, these type of memories are prone to soft errors, which might lead to data corruption. This paper presents a unique technique to improve the performance of SRAM-based TCAMs through a single-bit parity mechanism which is designed to detect and correct soft errors. By incorporating this error correction system TCAMs become more resilient to soft errors, in this way improving the data reliability and ensuring the system functions more smoothly and stably. The implementation of the proposed method has been done on Xilinx ISE 14.7 and is simulated in ModelSim 6.5b, demonstrating its efficiency in terms of resource utilization and performance across various TCAM sizes.

Keywords: Field-programmable gate arrays (FPGA), soft errors, Static Random Access Memory (SRAM), Ternary Content Addressable Memory (TCAM).

1. INTRODUCTION

Content Addressable Memories (CAMs) are the type of memories that are designed to enable high-speed data searches by comparing input data against a stored data. Ternary Content Addressable Memories (TCAMs) contains three possible states per bit - 0, 1, and X (don't care) [5]. Don't care state, also known as wild-card state "X" allows TCAMs to perform partial matching, which results in matches with multiple words [4]. In CAMs large amount of content is searched in parallel at high speed, where they can be used in wide range of applications such as pattern recognition, data compression, big-data analytics [3] as well as in network devices.

The FPGAs that are based on SRAMs are vulnerable to single-event upsets (SEUs) [3] due to the disturbances caused by high-energy neutron particles. The on-chip embedded memory is particularly susceptible to SEUs because of the small size and closely packed memory cells which increases their vulnerability [1]. Temporary errors are caused due to single event upsets and remains until the data is corrected. [5]. SEUs can cause errors like single-bit upsets (SBUs), multiple-bit upsets (MBUs) [3]. The TCAM functionality depends on the content which is stored in the configured embedded memory, like Block RAM (BRAM) [1]. A transient error in this memory can lead to a wrong match or mismatch, which in turn results in a match address which is incorrect being returned [2]. When a soft error occurs, the corrupted SRAM word needs to be overwritten to restore accurate matching for lookups. Yet, safeguarding SRAM-based TCAM solutions is difficult, as it must be done without increasing the critical path delay and while preserving high search performance [3].

In this paper, error detection mechanism is implemented with single-bit parity, which introduces insignificant delay as well as logic overhead [1]. The error-correction technique leverages the redundant binary-encoded TCAM table [1], typically used for SRAM based TCAMs to fix soft errors. In this approach search performance is high by allowing the error-correction method to run in the background, enabling simultaneous search operations [1]. The response time is low in this technique, ensuring reliable lookups for TCAM throughout nearly the entire processing duration.

II IMPLEMENTATION OF SRAM BASED TCAM ON FPGA

Various approaches exist for implementing TCAMs on FPGAs [2]. Small-sized TCAMs are typically realized using flip-flops (FFs) that are accessible in the FPGA's logic resources [10], [11]. In contrast, many FPGA-based TCAM designs predominantly use BRAM or distributed RAM (distRAM) [8], [9]. Using shallow SRAMs for TCAM design leads to higher memory efficiency [9]. TCAM designs that utilize more resources generally exhibit greater sensitivity to soft errors [4] returned. In a 28-nm FPGA device, the resource distribution is as follows: 34% for BRAMs, 5% for LUTRAMs, and 1% for slice registers [13]. This distribution implies that soft errors are less likely in the smaller populations, such as slice registers. Consequently, soft errors are more likely to effect TCAM architectures which are implemented in BRAM compared to those based on distRAM or SRAM. The occurrence of soft errors is frequently quantified by failures in time (FIT) [4]. In the experiments conducted with the Vertex-5 FPGA, the soft error rate per event in BRAMs is measured at 73% ± 9%. The don't care state "X" in TCAMs allows the representation of ranges, meaning an input word can match multiple entries if their ranges overlap. When multiple matches occur, the word which has the lowest priority will be selected. Therefore, TCAM words should be organised in a prioritized list, with high priority words are at lower memory addresses [4], [8]. Updating a TCAM word may alter the priorities and the words are reorganised within the TCAM table. Throughout such updates, search words which are coming as input may be suspended, as they cannot be matched accurately until the rearrangement is complete.

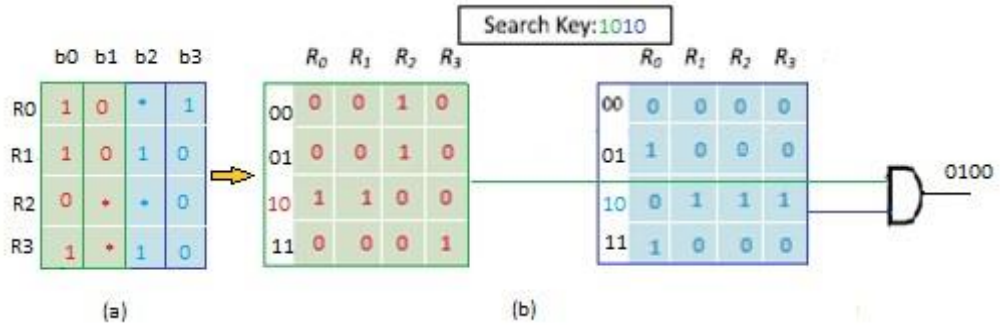

Fig 1: Basic implementation of TCAM using SRAM. (a) 4×4 TCAM table. (b) Implementation of 4× 4 TCAM using two 4 × 4 SRAMs.

TCAMs are specified hardware that are used for high speed data search operations. SRAM-based TCAMs are a popular choice for FPGA implementations due to their flexibility, power efficiency, and lower cost compared to traditional TCAMs. They leverage the SRAM blocks available on FPGAs to mimic the functionality of a TCAM, where each bit stores data using three different inputs: 0, 1, and X. Onchip SRAM memories in contemporary FPGAs are employed to realize TCAM solutions. A 1×1 TCAM can be realized by using a 2×1 RAM [1], where the matched condition meant for value "0" is denoted by storing "1" at RAM 0, for "1" value by way of storing a "1" at RAM 1, also for the "X" state by representing a "1" at RAM 0 and RAM 1 positions [4]. The word C-bit in a TCAM table may be realized by 1-bit RAM with 2^C entries [1]. SRAM address corresponds to the TCAM pattern of C-bit, whereas the SRAM's data holds match or mismatch status for every entry in the TCAM table relative to all available C-bit patterns. Thus, TCAM table with C-bit width and B entries are realized by means of a B-bit wide SRAM with 2^C positions. C bit TCAM patterns [1] are separated into smaller portions with depth D and then 2CxD size SRAMs are AND- Cascaded [8]. In Fig. 1(a), TCAM of 4-word deep with 4-bit patterns is separated into two 4×2 [1] partitions. As illustrated in Fig. 1(b), two 4×4 SRAMs are used to implement these partitions. For instance, if search key (1010) is given as input, the third word in the SRAM 1 (1100) is accessed by the first two bits (10), and the third word [1] in the SRAM 2 (0111) is accessed by the last two bits (10). The outputs from

the SRAMs are then ANDed together to obtain the concluding match result (0100), indicating a match for rule R2 [1].

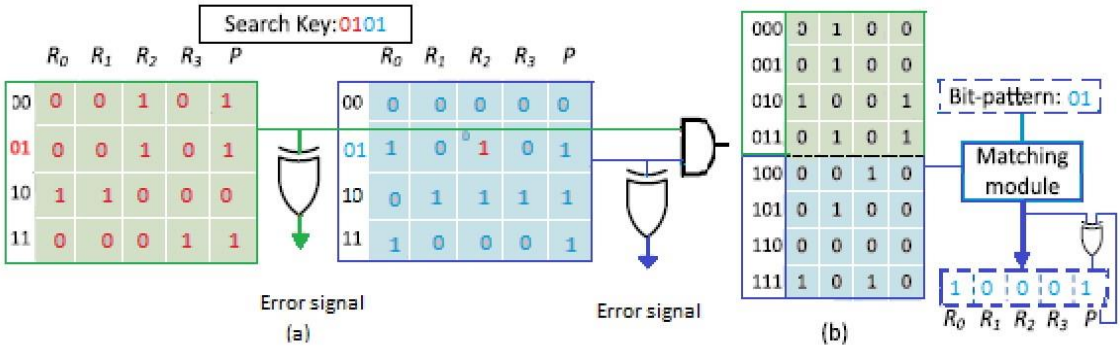

Fig 2: A simplified model of (a) Error detection in TCAMs using parity-protected SRAMs, and (b) Error correction vector generation through binary encoding of the TCAM content.

Figure 2 illustrates the fundamental concept of the error detection and error correction [6] model. The TCAM table is partitioned by two SRAMs and depict an 8 × 4 SRAM that stores binary-encoded TCAM table contents, which provides redundancy for matching information in the SRAMs that implement the TCAM. A shown in Figure 2(a), detection of a single-bit upset (SBU) in the SRAMs, is done by adding a parity bit to each SRAM word. The process of error detection is performed on the words of SRAM that are accessed throughout lookup operations. Upon detecting a fault in a particular word, the redundant information which is maintained in the TCAM table [1] is controlled by TCAM for error correction. Suppose, the search key (0101) given to the SRAMs as in figure 2(a), the second word (00101) in the SRAM 1 is accessed and the second word [1] (10101) in SRAM 2 is accessed. The parity result for the match bits in the third word in the SRAM 2 (10101) do not align with parity which is stored [1], representing that Single Bit Upset has occurred in the SRAM. TCAM sequentially retrieves the words from the SRAM that stores the binary-encoded TCAM contents of the equivalent partition. The SRAM words that are read are compared by means of the suitable bit pattern (01) for generating match bits and the related parity (10001), inclusively referred to as the error correction vector (ECV). This computed ECV is then used for overwriting the SRAM word which has fault, effectively correcting that particular error [1].

III DESIGN ARCHITECTURES FOR ERROR DETECTION AND ERROR CORRECTION IN TCAMS

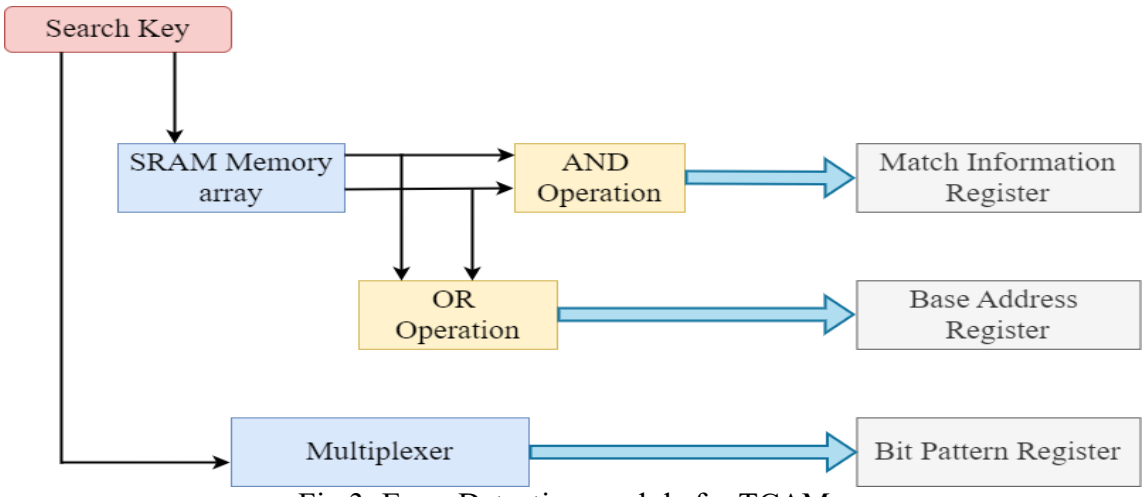

Fig 3: Error Detection module for TCAM

For SRAM-based TCAM solutions it is essential to ensure a soft error protection method, particularly one which does not impact the search throughput throughout the design. Error detection is achieved by means of single-bit parity checking, which incurs minimal overhead in terms of storage, logic, and delay [2]. Figure 3 illustrates the error detection architecture for TCAM. To generate an error signal, the corresponding SRAM words bits are XORed when search key [9] will be given for lookup. In the TCAM design, error signals which are present in the N SRAMs are then encoded to generate a log2N-bit error code, which identifies the SRAM that is corrupted [1]. Simultaneously, ANDed outputs of the SRAMs are provided to the match information register. The error code, along with the associated search-key bit patterns, is then sent for error-correction [1].

Figure 4 presents the error correction module for TCAM, where SRAM stores the contents of binary-encoded TCAM table, along with an ECV computation unit, an address generation unit (AGU), and also a read/write controller [1]. The MOD D counter is used to generate a new sequence of log2D bits for each round. The address of SRAM has been configured so as the log2N bits in the SRAM ID make up the most significant bits, indicating to the beginning of the equivalent intermediate block in the SRAM, while the subordinate log2D bits in the counter chooses specific SRAM words within that subordinate block. The arrangement allows AGU to access all the binary-encoded words [1] from the relevant partitioned TCAM table.

The words in TCAM that are read, will be coordinated against the C-bit pattern to generate a matching bit, which requires D clock cycles for computing both the bits that are matched as well as the parity bit that is associated, together form the ECV. The write-enable high signal is generated by the read/write controller for the equivalent SRAM, which allows the ECV that is computed to overwrite the incorrect SRAM word [1].

Throughout the error-correction method, the TCAM continues to support search operations, because the SRAMs implementing the TCAM task are accessible for search operations. These SRAMs are configured as simple dual-port RAMs, enabling simultaneous read and write operations within the clock cycle. When ECV is calculated, it is stored by means of the SRAM's write interface, allowing the error-correction method to fully overlap with ongoing search operations in the TCAM [1].

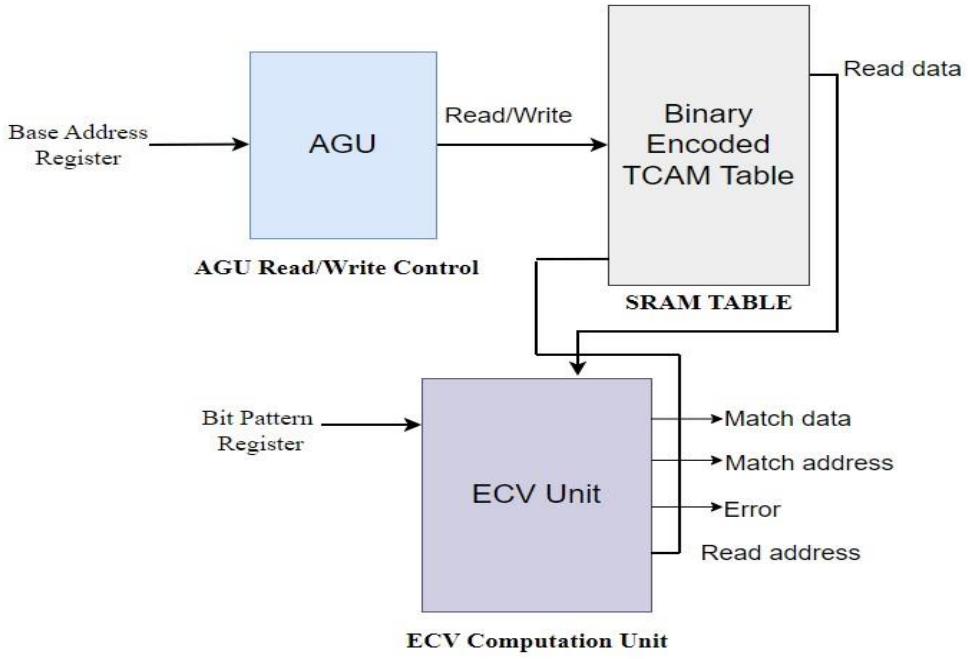

Fig 4: Error-Correction module for TCAM

IV EXPERIMENTAL RESULTS AND ANALYSIS

The following section outlines the results from experiments conducted for various TCAM implementations on Xilinx ISE 14.7, targeting the Virtex-5 FPGA device XC5VLX330T-FF1738. Simulations were performed using ModelSim software. The design was simulated for BRAM configurations of 64x40, 512x40, 1024x40, and 2048x40. A detailed comparison of performance metrics across these configurations is provided to highlight the efficiency and scalability of each implementation.

Table 1: Performance metrics of various TCAM sizes

TCAM size	LUT	Occupied Slices	Delay (ns)	Power (W)
64x40	63	29	1.744	3.541
512x40	78	35	2.004	3.543
1024x40	82	37	2.005	3.537
2048x40	85	38	2.314	3.537

Table 1 presents an overview of the performance metrics for various TCAM sizes from 64x40 to 2048x40. The key parameters such as LUTs, occupied slices, delay and power consumption are shown. As TCAM size increases, there is gradual rise in LUT ranging from 63 to 85, the occupied slices increase from 29 to 38. The delay also slightly increases with TCAM size, from 1.744 ns for 64x40 to 2.314 ns for 2048x40. However, the power consumption slightly decreases as the TCAM size increases. The TCAMs 1024x40 and 2048x40 consumes slightly less power compared to the smaller TCAM sizes. This could be due to smaller TCAMs have high leakage currents leading to high power consumption and it could also be due to fixed power overheads. This implies that the larger TCAMs are not only more resource-efficient but also more power-efficient. Therefore, it is a great option for applications where both power consumption and capacity are important considerations.

V CONCLUSION

This project successfully demonstrates the TCAM error-resiliency technique, which utilizes the binary-encoded TCAM table within TCAMs which are based on SRAM for fault correction. By employing single-bit parity for fault detection, TCAM ensures marginal logic overhead while maintaining the availability of SRAMs for search processes. The background error-correction procedure does not interfere with data path processing, allowing uninterrupted search functionality. The study, conducted on a Virtex-5 FPGA, highlighted key trade-offs between resource utilization, delay, and power consumption as TCAM size varies. These insights are valuable for optimizing TCAM designs according to specific application requirements. Future work could look into improvement of TCAM efficiency as well as exploring the implications of these findings across different platforms or practical scenarios, thereby extending the impact and applicability of this work in FPGA-based TCAM design.

References

[1] Inayat Ullah, Joon-Sung Yang, and Jaeyong Chung, “ER-TCAM: A Soft Error Resilient SRAM-based Ternary Content Addressable Memory for FPGAs. (VLSI) Syst., vol. 28, no. 4, pp. 1084-1088, April 2020.

[2] P. Reviriego, S. Pontarelli, and A. Ullah, “Error detection and correction in SRAM emulated TCAMs,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 2, pp. 486–490, Feb. 2019.

[3] T. Li, H. Liu, and H. Yang, "Design and characterization of SEU hardened circuits for SRAM-based FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 6, pp. 1276–1283, Feb. 2019.
[4] Afzaal. U, Lee. J. A, Ullah. I, and Ullah. Z, "DURE: An energy-and resource-efficient TCAM architecture for FPGAs with dynamic updates," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 6, pp. 1298– 1307, Mar. 2019.
[5] Y. -J. Chang, "Don't-Care Gating (DCG) TCAM Design Used in Network Routing Table," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 11, pp. 1599-1607, Nov. 2010, doi: 10.1109/TVLSI.2009.2025951.
[6] A. Varada and S. Agrawal, "An Efficient SRAM-Based Ternary Content Addressable Memory (TCAM) with Soft Error Correction," 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 2021.
[7] Ramos, R. G. Toral, P. Reviriego, and J. A. Maestro, "An ALU protection methodology for soft processors on SRAM-based FPGAs," *IEEE Trans. Comput.*, vol. 68, no. 9, pp. 1404–1410, Mar. 2019.
[8] Gupta. P, Shah. D and "Fast updating algorithms for TCAM," IEEE Micro, vol. 21, no. 1, pp. 36–47, Jan. 2001.
[9] Baeg. S, Ilgon. K, and Ullah. Z, "Hybrid SRAM-based ternary material addressable partitioned memory," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 12, pp. 2969–2979, Dec. 2012.
[10] *Ternary Content Addressable Memory (TCAM) Search IP for SDNet V1.0; Xilinx Product Guide PG190*, Xilinx, San Jose, CA, USA, Nov. 2017.
[11] W. Jiang, "Scalable ternary content addressable memory implementation using FPGAs," in *Proc. 9th ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2013, pp. 71–82.
[12] M. Irfan and Z. Ullah, "G-AETCAM: Gate-based area-efficient ternary content-addressable memory on FPGA," *IEEE Access*, vol. 5, pp. 20785–20790, 2017.
[13] Lee. J. A, Ullah. I, and Ullah. Z, "Efficient TCAM design based on multipumping-enabled multiported SRAM on FPGAA," IEEE Access, vol. 6, pp. 19940–19947, 2018.
[14] Irfan. M and Ullah. Z, "G-AETCAM: Gate-based area-efficient ternary content addressable memory on FPGA," IEEE Access, vol. 5, pp. 20785–20790, 2017.
[15] Jia. Y, Li. T, Yang. H, Wang. N, Wei. Y, and Zhao. H, "Investigation into SEU effects and hardening strategies in SRAM based FPGA," in Proc. 17th Eur. Conf. Radiat. Effects Compon. Syst. (RADECS), 2019, pp. 1–5.
[16] Maestro. J. A, Ramos. A, Reviriego. P, and Toral. R. G, "An ALU security methodology for SRAM-based soft processors FPGAs," IEEE Trans. Comput., vol. 68, no. 9, pp. 1404–1410, Mar. 2019.
[17] Keller, A. M and Wirthlin. M. J, "Impact of soft errors on largescale FPGA cloud computing," in Proc. ACM/SIGDA Int. Symp. FieldProgram. Gate Arrays, 2019, pp. 272–281.
[18] F. Syed, Z. Ullah and M. K. Jaiswal, "Fast Content Updating Algorithm for an SRAM-Based TCAM on FPGA," in IEEE Embedded Systems Letters, vol. 10, no. 3, pp. 73-76, Sept. 2018.[19] Pontarelli. S, Reviriego. P, and Ullah. A, "PR-TCAM: On Xilinx FPGAs, efficient TCAM emulation using partial reconfiguration," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 8, pp. 1952–1956, Mar. 2019.