International Journal of Engineering Technology and ManagementSciences
Website: ijetms.in Issue: 3 Volume No.6 May – 2022
DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

TCAM Network Memory Error Detection and Correction Method

Thirumala Vidya Sagar S[1], B.D.Venkatramana Reddy[2]
[1]M.Tech Student,Dept. of ECE, Viswam Engineering College, Madanapalli, AP, India
[2]Associate Professor,Dept. of ECE, Viswam Engineering College, Madanapalli, AP, India

Abstract: *Ternary content addressable memories (TCAMs) are widelyused in network devices to implement packet classification. They are used, for example, for packet forwarding, for security, and to implement software-defined networks (SDNs). TCAMs are commonly implemented as standalone devices or as an intellectual property block that is integrated on networking application-specific integrated circuits. On the other hand, field-programmable gate arrays (FPGAs) do not include TCAM blocks. However, the flexibility of FPGAs makes them attractive for SDN implementations, and most FPGA vendors provide development kits for SDN. Those need to support TCAM functionality and, therefore, there is a need to emulate TCAMs using the logic blocks available in the FPGA. In recent years, a number of schemes to emulate TCAMs on FPGAs have been proposed. Some of them take advantage of the large number of memory blocks available inside modern FPGAs to use them to implement TCAMs. A problem when using memories is that they can be affected by soft errors that corrupt the stored bits. The memories can be protected with a parity check to detect errors or with an error correction code to correct them, but this requires additional memory bits per word. In this brief, the protection of the memories used to emulate TCAMs is considered. In particular, it is shown that by exploiting the fact that only a subset of the possible memory contents are valid, most single-bit errors can be corrected when the memories are protected with a parity bit.*

Keywords: Field-programmable gate arrays (FPGA), soft errors, ternary content addressable memories (TCAM)

1. INTRODUCTION

Ternary Content Addressable Memory (TCAM) is a special type of memory. For random access memory (RAM), a bit has only two possible values (i.e., 0 or 1), and the input is the index and the output is the content of the memory at that index. Whereas, a bit in TCAM has three possible values: 0, 1, or *, where * means ‘‘don’t care’’. Each entry in TCAMs is an array of 0s, 1s, or *s. TCAMs work in a reverse manner as compared to RAMs: the input to a TCAM chip is a packet header (i.e., a search key) of 0s and 1s, and the output is the index of the *first* entry that the key matches where decisions are stored [1]. A search key matches an entry if and only if their bits of 0s and 1s match. For example, 1,001 matches a TCAM entry 10**. Furthermore, a TCAMs searches the entire memory in one operation, so it is considerably faster than RAMs. In essence, a TCAM is a parallel search machine, not just memory.

TCAMs are the core component of many networking devices such as routers, switches, firewalls and intrusion detection/prevention systems. One major task performed by TCAMs is packet classification, the processing of finding the first rule that a given packet matches in a rule list. In TCAM-based packet classification, rules are stored in the TCAM in ternary format. Thus, when a packet comes, only one TCAM lookup is needed. The result of the TCAM lookup is the index in the memory where packet decisions, such as accept or drop, are stored. This superior constant lookup performance is the key reason that many networking devices use TCAMs to implement their lookup operations.

A TCAM bit may flip to any of the other two values of 0, 1, or * due to environmental impacts. Memory and electronics in general are susceptible to non- persistent faults, called *soft errors*, due to radiation events that generate enough electricity to cause a bit to flip in memory or for a transistor to trigger or fail to trigger [2, 3]. TCAMs are even more susceptible to soft errors than RAMs because of much higher circuit density [4]. Furthermore, the soft errors in TCAMs are more damaging than the soft errors in RAMs. While a soft error in RAM only causes one lookup to fail, a soft error in a TCAM chip can cause many different lookups to fail due to the first matching semantics: the lookup result

of a search key is the index of the first entry that the key matches [5]. For example, if a TCAM entry ‘‘1**’’ changes to ‘‘100’’, then the lookup results of the three keys ‘‘101’’, ‘‘110’’, and ‘‘111’’ may become erroneous. In fact, a single erroneous entry in a TCAM can cause 100 % of search keys to have erroneous lookup results in the worst case. TCAMs are increasingly prone to more soft errors as the area efficiency of memory devices decreases [5]. Furthermore, Mastipuram et al. [5] reported that soft errors induced the highest failure rate of all other reliability mechanisms combined in TCAMs. Consequently, it is important to ensure reliability in TCAM operations because soft errors can potentially jeopardize their application in mission-critical network management applications.

Figure 1 shows different effects of soft errors in TCAMs. Note that the index returned by an erroneous lookup may be smaller or greater than the index returned by a correct lookup. Figure 1a shows an example TCAM table without errors, on which the lookup result for search key 0010 is the second entry. Figure 1b shows animplementation of the table in Fig. 1a where the last bit in the first entry is flipped from 1 to 0. On this table, the lookup result for search key 0010 is the first entry. Figure 1c shows an implementation of the table in Fig. 1a where the third bit in the second entry is flipped from * to 0. On this table, the the lookup result for search key0010 is the third entry. Moreover, some soft errors do not affect the lookup result ofany search key in a TCAM. For example, the first bit in the fourth entry in Fig. 1c erroneously flipped from * to 1. However, this error does not affect any lookup. In this paper, we do not consider such benign errors as they cause no harm. In the rest of this paper, the term TCAM errors refers to harmful TCAM errors.

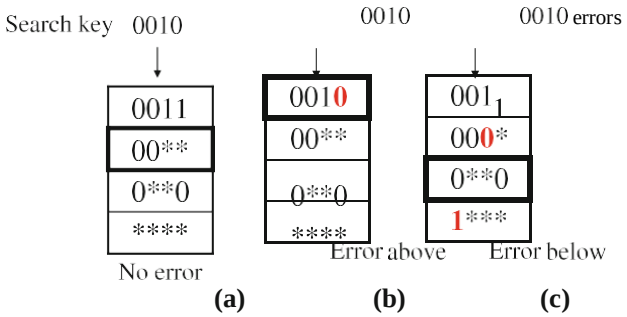

Fig. 1 Effects of TCAM soft

2. RELATED WORK

RAM-based associative content-addressable memory (CAM) was presented in a U.S. patent. Its memory requirement increases exponentially with the increase in the pattern bits of CAM. For large bit patterns, it does not scale well in terms of the memory requirement, power consumption, and cost. Thus, it is not feasible to implement it on ASIC or FPGA platforms. Contrary to this our proposed work is memory-efficient and achieves a reduction in the dynamic power consumption by reducing the exponential increase in RAM memory to a linear increase. This is achieved by partitioning the large width TCAM bit patterns and then implementing them as a cascade of SRAM blocks in the proposed architecture.

A set-associative memory architecture implemented in hardware using the well-known hashing method was presented in. It used RAM memory to implement CAM. However, in order to support TCAM functionality, the architecture suffered from inefficient memory utilization as it required two bits to encode ternary bits. On the contrary, our proposed solution does not encode ternary bits as two bits. The proposed solution addresses RAM memory with TCAM content, and each RAM word corresponds to a specific TCAM data pattern. In order to implement “don’t care” bits, more than one RAM word is selected.

DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

The proposed solution maps all possible TCAM data as RAM addresses. Two types of FPGA implementations: a CAM based on Xilinx BRAM resource and a TCAM based on 16-bit shift registers (SRL16E)-are presented in Xilinx application note [20]. This emulation of TCAM on FPGA consumes one SRL16E for implementing two bits of TCAM. The shift register based TCAM works efficiently for smaller TCAMs whereas, for implementing large storage capacity TCAMs, its design experience routing congestion and timing challenges.

The proposed solution implements TCAM using the embedded SRAM memory blocks available on FPGAs and scales well in terms of speed and power consumption for implementing TCAMS of large storage capacity. A low-power SRAM-based CAM design implementation on FPGA was presented in a previous work [17]. It performs a hierarchical lookup of the design-configured SRAM blocks. It achieves low power consumption by stopping subsequent SRAM lookup operations if a match is found in the present SRAM block. Although it reduces the average power consumption of the design, its worst-case power consumption remains high, which is not beneficial for the designed hardware as the hardware is designed based on the worst-case power consumption budget. In contrast, proposed work achieves a significant reduction in the worst-case power consumption of the implemented TCAM design.

Algorithmic solutions of TCAM implemented in SRAM-based pipelines on FPGAs suffer from non-deterministic throughput, long latencies, and inefficient memory usage. A resource-efficient SRAM-based TCAM design was presented in, which stored the validation information of the TCAM words in distributed RAM blocks and the address information in the sub-blocks of the embedded SRAM memory blocks on FPGA. It completed the lookup operation for the incoming TCAM word via multiple high-speed sequential reads from the sub-blocks of its SRAM blocks and thus resulted in a degraded system throughput. The multiple high-frequency SRAM read operations per input word's lookup resulted in higher dynamic power consumption of the design. Proposed work stores the validation and address information of the TCAM words in a single SRAM memory block and a single read operation is performed for completing the incoming TCAM word's lookup. Thus, achieves higher throughput with reduced overall power consumption.

The design methodologies presented in employs multiple distinct SRAM blocks for implementing TCAM functionality. These stores the TCAM word's existence and address information separately in distinct sets of SRAM blocks. The Input TCAM word is applied to the first set of SRAM blocks to read its existence information and the address information is read from the second set of SRAM blocks. These TCAM design methodologies using multiple distinct SRAM blocks utilizes excessive SRAM memory. These works suffered from higher power consumption as the entire used excessive SRAM memory is activated for the incoming TCAM word lookup. In contrast, our proposed work stores the TCAM word's existence and address information in a single RAM, thus realizing efficient memory usage. Moreover, proposed work achieves substantially reduced power consumption by activating at most one row of SRAM blocks for incoming TCAM words.

The SRAM-based design methodology with efficient storage efficiency in previous work stored the existence and address information of TCAM word in a single SRAM memory block. A similar approach implemented an SRAM-based TCAM design using Xilinx BRAM or distributed RAM resource and provided an in-depth theoretical analysis in. However, these works energize all used SRAM memory blocks in the design for each incoming TCAM word's lookup. Thus, consumes higher power consumption. In contrast proposed TCAM architecture selectively energizes a part of SRAM memory blocks used, achieving a substantial reduction in the overall dynamic power consumption of the design.

A recent work presented in uses multi-pumping enabled multi-ported SRAM memory for implementing memory efficient TCAM design on FPGA. It stores the sub-blocks of

DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

partitioned TCAM table in the shallow sub-blocks of BRAMs configured as multi-ported memories on FPGA.

For each incoming TCAM word, the existing SRAM-based TCAM architectures energize the entire SRAM memory of their architectures, resulting in excessive power consumption. In this work, we present a pre-classifier-based architecture for an energy-efficient SRAM-based TCAM design. We first classify a TCAM table into several TCAM sub-tables, which are further partitioned vertically. Each partitioned TCAM sub-table is implemented as a row of cascaded SRAM blocks in the proposed architecture. For each input TCAM word, at most one row of SRAM blocks is activated in the proposed design, significantly reducing the dynamic power consumption compared with the existing SRAM-based TCAMs.

The demand for a high-speed flexible (re-configurable) and adaptable (easy for integration) TCAM configurations renders the embedded memories BRAMs on modern SRAM-based FPGAs attractive for the design of TCAMs. FPGAs implement TCAM using SRAM, by addressing SRAM with TCAM contents, and stores information for all data of TCAM table. Each SRAM word stores the existence and address information for a specific TCAM pattern. Existing SRAM-based TCAMs on FPGAs suffer from higher energy consumption as they consume excessive power to energize the entire SRAM memory used per lookup. For example, the SRAM-based TCAM design methodologies presented in recent works [13,17] consumed 2.5 W and 3.2 W to implement 89 kb and 150 kb TCAM tables using BRAMs on FPGA, respectively. The higher power consumption of SRAM-based TCAM designs becomes more severe for larger capacities.

The demand for a low power configurable and easy to integration TCAM design on FPGA makes the use of pre-classification approach practical for designing an energy-efficient SRAM-based TCAM (EE-TCAM). It works as follows: First, a TCAM table is partitioned into several sub-tables of balanced size in the classification stage. Second, in the SRAM-based implementation stage, each resultant TCAM sub-table is mapped to a separate row of cascaded SRAM blocks in the architecture. The proposed architecture selectively activates at most one row of SRAM blocks for each incoming TCAM word, thus attaining a substantial reduction in the overall dynamic power consumption.

3. PROPOSED METHOD

The conventional 6T SRAM cell uses two cross coupled inverters and two access transistors as shown in Figure 1. These access transistors connect the cell to the outside world. The inverters are the storage element and reinforce the data bit within the cell as long as the power is supplied (VDD).

International Journal of Engineering Technology and ManagementSciences
Website: ijetms.in Issue: 3 Volume No.6 May – 2022
DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

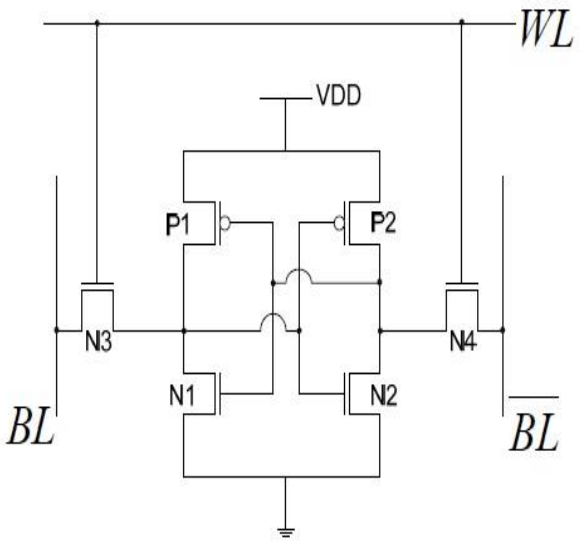

Fig 4-1 Conventional 6T SRAM Cell

P1, P2 are PMOS transistors and N1, N2, N3, N4 are NMOS transistors. N3 and N4 are the access transistors (or pass-transistors) connecting the cell to the Bit Lines (The different modes of operation of the SRAM cell are detailed below.

A. Standby/Hold Mode

When the Word Line (WL) is at logic '0', the access transistors N3 and N4 disconnect the cell from the Bit Lines $(BL\ and\ \overline{BL})$. The two cross coupled inverters in the cell continue to reinforce the data bit present in the cell as long as power is supplied (VDD). The current drawn in this mode from VDD is termed as standby current.

B. Write Mode

The cell gamma ratio should be sufficiently high for a successful write operation to take place. Cell gamma ratio is defined as the ratio of the strength (drive current) of the pass transistor (access transistor) to that of the pull up transistor. The value to be written into the cell is applied to the Bit Lines. Complementary Bit Lines are used $(BL\ and\ \overline{BL})$. on either side of the cell to ensure that the load to charge is reduced for each access transistor. This enables us to use smaller access transistors on each side rather than one single large access transistor. For a write operation, the Word Line (WL) is set to logic '1' enabling the access transistors N3 and N4 thereby connecting the cell to the outside world $(BL\ and\ \overline{BL})$. The contents from the Bit Lines $(BL\ and\ \overline{BL})$. are then transferred into the cell via the access transistors. After a successful write operation, the Word Line (WL) is set to logic '0'. For example, to write logic '0' into the cell, initially is set to logic '0' and is set to logic '1'. The Word Line (WL) is then set to logic '1' thereby allowing the access transistors to pass the logic value from into the cell $(BL\ and\ \overline{BL})$. After the data bit is successfully written into the cell, the Word Line (WL) is set to logic '0' thereby disconnecting the cell from the outside world $(BL\ and\ \overline{BL})$.

C. Read Mode

For a successful and stable read operation, the cell beta ratio should be sufficiently high. However, the drawback of a high beta ratio is lowered performance. Cell beta ratio is defined as the ratio of the strength (drive current) of the pull-down transistor to that of the pass transistor (access transistor). In order to read the data bit present in the cell, Bit Lines $(BL\ and\ \overline{BL})$ are initially pre charged to VDD i.e., logic '1'. The Word Line (WL) is then enabled by setting it to logic '1'. Depending on the data bit stored in the cell, a Bit Line $(BL\ and\ \overline{BL})$ gets discharged slightly (and slowly) via an access transistor and pull-down transistor thereby developing a differential voltage drop between the Bit Lines

DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

$(BL\ and\ \overline{BL})$. This small differential voltage drop across $(BL\ and\ \overline{BL})$. is then detected by a Sense Amplifier which determines the data bit stored in the cell. Charging and discharging of large capacitive loads is done by the Sense Amplifier. The higher the sensitivity of the sense amplifier, the faster the read operation. However, the differential voltage should not be too large as it could flip the state of the cross coupled inverters.

Readback scrubbing is considered as an effective mechanism to correct errors in Static-RAM (SRAM)-based Field Programmable Gate Arrays (FPGAs). However, current solutions have a low error correction percentage per unit area overhead. This paper proposes two new error detection/correction mechanisms that combine frame readback scrubbing with error correction codes (ECCs) that are applied in multiple directions, to achieve a high error correction percentage per unit area overhead. Experiments conducted show that the proposed schemes have an excellent error correction percentage (over 99%), especially for multi-bit upsets, while using up to 59.37% lesser area overhead compared with other state-of-the-art.

Methodology:

A CMOS SRAM cell is made up of six MOSFETS which has lower power consumption in standby mode and a greater immunity to transient noise and voltage variation than 4T resistive load cell just because it is preferred over resistive load cell for high-speed low power operation. The storage cell has two stable states denotes by 0 and 1. Other than two inverters two additional access transistors serve to control the access to a storage cell during read and write operations. Access to the cell is enabled by the word line (WL) which controls the two access transistors M5 and M6 which, in-turn, control whether the cell should be connected to the bit lines:BL and BLB.

Although it is not strictly necessary to have two bit-lines, global bit-line arrangement is helpful to maintain the stability of the cell and reduce the voltage swing which have initial impact on the power dissipation of the cell. Another advantage is it reduce the complexity of the SRAM cell.

During read access, Pre-charge circuitry is used for differential sense amplifier atthe end of the bit-line which are actively driven high and low by the inverters in theSRAM cell. This improves SRAM bandwidth compared to DRAMs. The symmetric structure of SRAM allows for differential signaling, which makes small voltage swings more easily detectable.

Cell has three different states:

Standby: Idle circuit

If the word line is not active, then M5 and M6 disconnect the cell from the bit lines and the two cross coupled inverter will continue to reinforce each other as long as they are connected to the supply.

Reading: Data has been requested

It totally depends on pre-charging concept, as both the bit-lines and word line are active high. Sometimes a small delta delay is occurred across the bit-lines which will resolved by attaching a sense amplifier at the end of the cell. The higher the sensitivity of sense amplifier, the faster the speed of read operation.

Writing: Updating the contents

Whatever the value we want to write same value is applied to the bit-lines. Bit-line input drivers are designed to be much stronger than the relatively weak transistors in the cell itself, so that they can easily override the previous state of the cross coupled inverters. Careful sizing of the transistors in an SRAM is needed to ensure proper operation.

Modern Static-RAM (SRAM)-based Field Programmable Gate Arrays (FPGAs) are gaining significant importance in space applications due to their operational capacity and performance. Moreover, these devices can be reconfigured after launch depending on various

DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

functional requirements and changes in the device environment [1]. However, due to the technological developments leading to denser chips, these devices become vulnerable to radiation effects called Single Event Upsets (SEUs) that are common in space environments. SEUs can inadvertently change the configuration of the SRAM bits, thereby changing the functionality of the circuit implemented [2]. If SEUs affect only one bit, this effect is known as a Single-Bit Upset (SBU) or single error. On the other hand, if several bits are consecutively affected, this effect is known as a Multiple-Bit Upset (MBU) or burst errors (Figure 1). Several mechanisms have been proposed to mitigate SEUs in SRAM-based FPGAs. The most common solution explores spatial/hardware redundancy [3]. Triple Modular Redundancy (TMR) [4] [5] replicates three times the hardware module to be protected and votes on their outputs, identifying the right results and a possible faulty module. However, this approach imposes a great overhead in terms of area and power consumption. Moreover, this method does not avoid error propagation if more than one component produces erroneous output. Duplication With Compare (DWC) [6] is an alternative approach to reduce the TMR overhead. It compares the output results of duplicated modules in order to identify the errors. However, it cannot correct them, but can trigger the suitable operations to do that, such as a full re-execution. Blind scrubbing is another traditional method of fault mitigation. This mechanism does not detect the existence of faults, but instead, periodically rewrites the configuration frames on to the FPGA, overwriting possible faulty bits caused by SEUs [7] [8]. An external memory with continuous access is required to store the configuration frames, frequently called as golden copy. In order to minimize the faults' impact, the scrubbing frequency must be greater than the expected SEU frequency. However, determining this frequency requires in advance a deep knowledge about fault susceptibility of the device technology, as well as the environmental conditions to which it would be exposed. The balance between error correction and the overhead required has been a challenging problem faced by researchers. This paper proposes two new error correction schemes based on frame readback and ECCs. Depending on the balance of error correction required and area overhead acceptable, one of the proposed solutions can be used. The proposed schemes detect and correct errors in each individual frame of the FPGA by mapping each frame to a 2D matrix and then computing hamming or parity bits for the matrix in different directions (rows, columns and diagonals). The mechanism adopted for computing the ECCs ensures a very good error correction performance, especially for burst errors and also uses lesser area overhead as compared with other state-of-the-art.

SOFT ERRORS in SRAM:

Technology scaling dramatically increases the sensitivity of the semiconductor devices to radiation. Due to large number of cells with minimized dimensions, SRAM arrays often are the densest circuitry on a chip. The large bit count contributes to the probability that an ionizing particle will hit a sensitive node in the array and corrupt the stored data. The minimum layout dimensions reduce the storage node capacitance and thus, the critical charge Qcrit that can be injected by radiation and upset the SRAM cell.

The shrinking supply voltages reduce the Qcrit even further. These factors contribute to the radiation-induced data errors that complicate building reliable SRAM arrays in nano-scaled technologies. Radiation can create localized ionization events in the semiconductor devices either directly or as secondary reaction products. Many of these radiation-induced events create enough electron-hole pairs to upset the storage nodes of SRAM cells. Such an upset is called a "soft" error. While such an upset can cause a data error, the device structures are not permanently damaged. If the voltage disturbance on a storage node of an SRAM cell is smaller than the noise margin of that node, the cell will continue to operate properly maintaining its data integrity.

However, if the noise margin of a cell is not sufficient to withstand the disturbance caused by ionizing radiation, a “soft” error will result. Soft errors originally were discovered to be a problem for DRAMs with planar storage capacitors in the late 1970s. Such capacitors stored their charge in large area two-dimensional p-n junctions. Due to the high collection efficiency of the radiation-induced charge, the large planar reverse-biased junctions made early DRAM cells highly susceptible to soft errors. Technology scaling and the push to improve the high soft error rate (SER) and poor pause/refresh time ratios of DRAMs necessitated the development of more compact three-dimensional storage capacitors. Compared to the old 2D planar capacitors, the new 3D capacitors had significantly smaller junction collection efficiency due to the reduced volume of the p/n junction. Despite the reduction of the Qcrit of a DRAM cell due to VDDscaling, it was more than compensated by the aggressive junction volume scaling of the storage capacitor. As a result, the SER of a DRAM bit cell is reducing by about 4x per technology generation.

This DRAM SER reduction is, however, being offset by the similar DRAM bit count growth rate at a system level. Because larger DRAM arrays are statistically more susceptible to soft errors, the resulting DRAM SER at the system level has remained relatively stable over many recent technology generations. Early SRAMs were more robust to soft errors than DRAMs due to the feedback mechanism that helps to maintain the state of an SRAM cell. In an SRAM cell both the capacitance of the storage node and the restoring current provided by the pull-up or driver transistors contribute to the cell critical charge.

With technology scaling, the SRAM cell area and thus, the junction area of the storage nodes are shrinking (Fig S1). While it could reduce the cell junction leakage, it also reduced the storage node capacitance. Simultaneously, the transition from the constant voltage scaling to the constant electric field scaling resulted in aggressive VDD scaling. Both these factors directly contribute to the reduction of the resulting Qcrit and the increasing probability of soft errors leading to higher SER levels. With each successive technology generation, the reductions in cell collection efficiency due to shrinking cell depletion volume was compensated by the reductions of VDD and storage node capacitance.

Therefore, soft errors have recently become a growing issue in ultra-high densitylarge embedded SRAMs operating at low voltages. The SRAM bit SER is shown to have saturated for technology nodes beyond 0.25 μm due to the saturation in VDD scaling, reductions in junction collection efficiency of highly doped p-n junctions and the increased charge sharing between the neighboring nodes. However, the exponential growth of SRAM cell count in modern CPUs and DSP processors has led the SRAM system SER to increase with each technology generation.

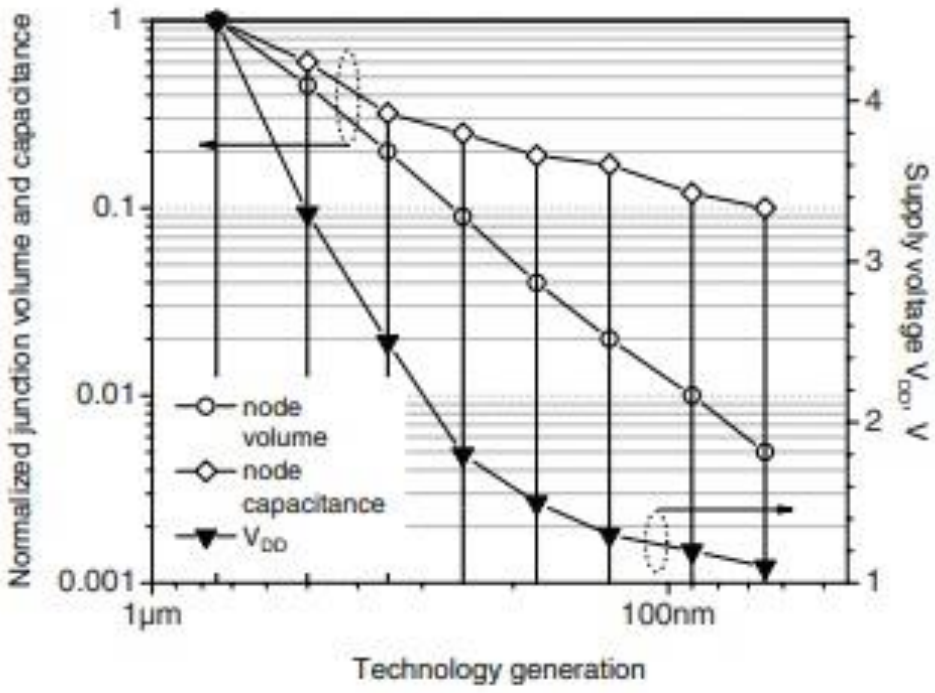

Fig :The normalized junction volume and capacitance of SRAM storage nodes and the power supply voltage VDD scaling as a function of the technology generation.

Readback scrubbing has been seen by the researchers as an effective mechanism to provide error correction in SRAM-based FPGAs [10]–[14]. Three categories of readback scrubbers can be found in the literature. The first category enables fault detection with a direct comparison between read frames and the golden copy [13]. The fault is corrected by overwriting the faulty frame with the respective golden copy frame. The second category does not use an exhaustive comparison with the golden copy [14]. It detects the faults matching the online computed error detection codes (EDCs) with the original ones, previously computed and externally stored for each frame. Similar to the previous category, the fault recovery is performed using the frame's golden copy. The third category enables the fault detection, computing ECCs for each frame [10]–[12]. The ECCs allow the faults' detection, similar to the previous category. However, upon fault detection in a frame, the faults can be easily recovered using the ECCs. Once the fault is recovered, the frame is written back into the FPGA. Different error detection schemes, combined with different ECCs have been proposed in the literature. However, they are not really efficient to handle burst errors. Lanuzza et al. [12] proposes a scheme to correct burst errors in SRAM-based FPGAs by applying hamming codes to a data word obtained by frame bit interleaving. The bit interleaving technique reduces the probability to have several bit-faults in the same data word, thereby increasing the correction efficiency. However, the error correction is limited by the amount of bit interleaving, and hence might not be suitable if a very high error correction efficiency is required. Argyrides et al. [10] introduce a scheme called Matrix Code (MC), that uses hamming codes combined with parity codes to enable the detection and correction of multiple errors in an FPGA configuration frame. A frame word is mapped into a matrix of sub words. Errors are corrected by computing hamming codes for each row, providing Single Error Correction Double Error Detection (SECDED) and computing parity codes for each column. As a result, this scheme is not efficient in handling MBUs, since if more than two errors occur in the same row, they are not detected by the ECC code. Park et al. [11] propose a built-in 2-D Hamming Product Code (2-D HPC) scheme. This technique is able to perform SECDED by using hamming codes built from arranging the FPGA configuration frame in a

DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

2-D array. Therefore, hamming codes are computed for each row and for each column of the 2-D array. Like the previous work, this scheme is not able to detect more than two errors that occur in the same row or column, and hence not efficient in handling burst errors.
proposes two novel schemes for error detection and correction in SRAM-based FPGAs, using the readback scrubbers. Both these schemes are based on existing ECCs, i.e., parity codes and hamming codes, which are applied to the FPGA frames. A frame is the lowest reconfigurable granularity that can be found in an FPGA. In particular, the FPGA configuration data frames F Rc contain information about the circuit design to be implemented on the FPGA. These schemes map sequentially the content of each frame fri ∈ F Rc in a 2D data matrix mfri with nr rows and nc columns. ECC codes are then computed for this 2D data matrix and stored in an internal or external storage. During runtime, the frames are periodically read and mapped to the 2D data matrix to see if they contain any errors. Errors are identified by comparing the ECCs to the ones stored in the memory. In the case of an error, the proposed algorithm attempts to rectify the affected bits. Once the errors are rectified, the corrected frame is then written back into the FPGA board.

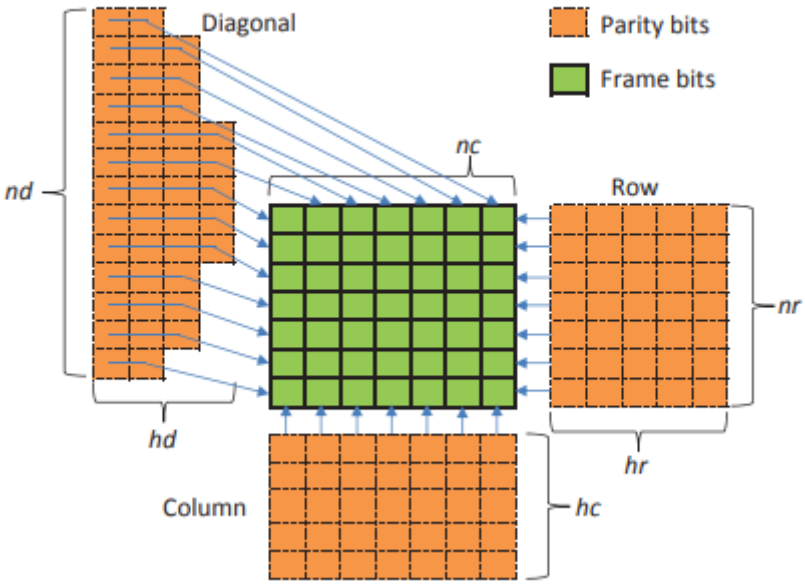

Fig. ECCs used in H3 scheme.

Error Model There are two different error models considered for this work. Both these models focus on SEUs, i.e., transient soft errors due to a single particle strike and which affect the configuration (both the logic and routing) bits of the FPGAs. The first model considers SBUs (single errors) and the second considers MBUs (burst errors). Both error models are illustrated in Figure 1. According to the literature, the latter model is more realistic since it is very common that a single particle strike might affect one or more neighboring bits [12]. We try to detect and correct both single errors as well as burst errors in this work. The following sub-sections discuss these schemes in detail with examples to illustrate the working of each. B. H3 Scheme H3 scheme applies hamming codes to the frame matrix at the design time in three directions, rows, columns and in one of the diagonals as shown in the Figure 2. With this scheme, single error correction (SEC) is available for each row rj ∈ mfri, for each column cj ∈ mfri and for each line of one of the diagonals dj ∈ mfri.

International Journal of Engineering Technology and ManagementSciences
Website: ijetms.in Issue: 3 Volume No.6 May – 2022
DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

The pseudocode presented in Algorithm 1 illustrates the H3 scheme behavior. The matrix frame is received as input and SEC is sequentially applied to the rows, columns and diagonal. While errors are detected in the matrix frame, SEC is continuously applied. When no errors are detected the mfri is returned. Figure 3 illustrates the working of the H3 scheme. In the first iteration, the algorithm computes the hamming codes for the rows, columns and diagonals respectively and compares it to the ones already stored in memory. As can be seen from the figure, errors are found in rows 2 and 7 (matrix A1), columns 1, 2, 4, and 6 (matrix A2) and diagonal 2 and −3 (matrix A3). These error bits are highlighted in yellow color. All the errors are corrected at the end of the first iteration.

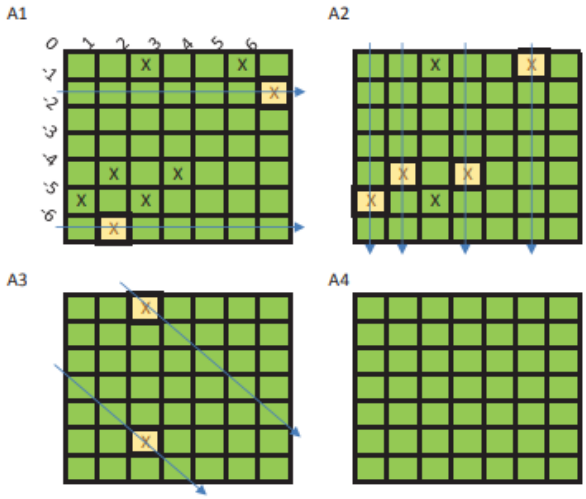

Fig. 3. Example of H^3 scheme.

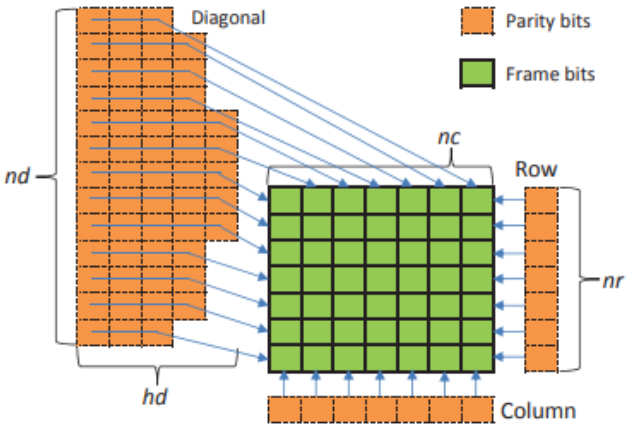

Fig. 4. ECCs used in P^2H scheme.

DOI:10.46647/ijetms.2022.v06i03.019 ISSN: 2581-4621

Algorithm 1 H^3 scheme.

Require: mfr_i
Ensure: mfr_i
1: $errorExists = true$
2: **while** $errorExists = true$ **do**
3: **for all** $r_j \in mfr_i$ **do** $sec(r_j)$;
4: **for all** $c_j \in mfr_i$ **do** $sec(c_j)$;
5: **for all** $d_j \in mfr_i$ **do** $sec(d_j)$;
6: **update** $errorExists$;
7: **end while**
8: **return** mfr_i;

sec – single error correction

ECC Overhead: By definition, for SEC, the number of hamming bits h required for a n-bit word is given by the equation, n +1+ h ≤ 2h. This way, the ECC overhead can be given by Equation 1, H3oh = nd j=1 hdj + (nr × hr)+(nc × hc) (1) where, hdj , the number of hamming bits for each diagonal j, is given by |dj | +1+ hdj ≤ 2hdj ; hr, the number of hamming bits for the rows, is given by nc +1+ hr ≤ 2hr; hc, the number of hamming bits for the columns, is given by nr +1+hc ≤ 2hc; nd, the number of diagonals of the matrix, is given by nd = nr + nc − 1; dj is the set of elements of the diagonal j.

Algorithm 2 P^2H scheme.

Require: mfr_i
Ensure: mfr_i
1: $errorExists = true$
2: $diagErrLoc = []$; $rowErrLoc = []$;
3: $colErrLoc = []$; $errLocs = []$;
4: **while** $errorExists = true$ **do**
5: **for all** $d_j \in mfr_i$ **do** $sec(d_j)$;
6: **for all** $d_j \in mfr_i$ **do** $diagErrLoc+ = ded(d_j)$;
7: **for all** $r_j \in mfr_i$ **do** $rowErrLoc+ = ped(r_j)$;
8: **for all** $c_j \in mfr_i$ **do** $colErrLoc+ = ped(c_j)$;
9: $eci(diagErrLoc, rowErrLoc, colErrLoc)$;
10: **update** $errorExists$;
11: **end while**
12: **return** mfr_i;

ded - double error detection
ped - parity error detection
eci - error correction by intersection

multiple errors, but to also correct them. The pseudocode of this scheme is presented in Algorithm 2.

P2H Scheme P2H scheme provides error detection and correction through the use of both parity and hamming codes. Parity codes are applied for each row rj ∈ mfri and for each column cj ∈ mfri, while hamming codes are applied for each line of one of the diagonals dj ∈ mfri, as shown in Figure 4. Since parity code can only detect the presence of a single error, this scheme also employs an efficient algorithm to not only detectThe first operation detects and corrects any single-bit error for each diagonal dj ∈ mfri through the function sec (dj). This first operation can be observed in Figure 6 (A1 − A2). Note the frame burst errors can be easily corrected on this first operation. Once all the single bit errors in the diagonals are removed, it is not possible to correct any more errors using only a single direction. Therefore, the next operation set identifies the diagonals, rows and columns with errors and matches

them in order to identify their exact location. ded(dj) identifies the diagonals with double errors, through the hamming codes. ped (rj) and ped(cj) identify the rows and the columns with errors, through the parity codes. These locations are submitted to the function eci, which match them and produce a set of intersections in the matrix frame. These intersections are potential bit errors. This way, several operations are then performed to identify and correct the errors. Figure 5 presents a flowchart, which describes the operation flow performed by P2H scheme. In particular, the shaded area describes the operation flow performed in the scope of eci function.

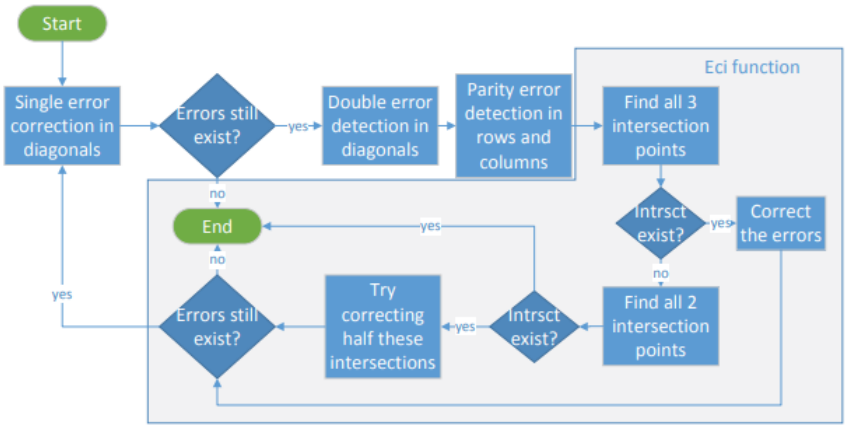

Fig. 5. Flowchart of P^2H scheme.

4. CONCLUSION

In this brief, a technique to protect the SRAMs used to emulate TCAMs on FPGAs has been proposed. The scheme is based on the observation that not all values are possible in those SRAMs, and thus, there is some intrinsic redundancy of the memory contents. This redundancy is used to correct most single-bit error patterns when the memories are protected with a parity bit to detect errors. The proposed technique reduces significantly the resources needed to protect the memories and can be an interesting option for designs on which reliability is a concern, but resources are limited.The idea presented in this brief can be extended to other memory configurations. For example, it can be used to detect errors on an unprotected memory by periodically scrubbing the contents to check their correctness. It could also be used when the memory is protected with a more powerful code that can detect several bit errors to correct multiple bit errors. For example, for a memory protected with an SEC code, double-bit error patterns could be detected and then use the intrinsic redundancy of the memory contents to correct them.

5. 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 mem- ory (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] P. Bosshart et al., "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," in Proc. ACM SIG-COMM, 2013, pp. 99–110.

[9] 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.

[10] 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.

[11] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 Gbps as research commodity," IEEE Micro, vol. 34, no. 5, pp. 32–41, Sep./Oct. 2014.

[12] Naik, S.B., Mahesh, B. and Koilakuntla, K.D., 2021. Evaluating Malware Detection System using Machine Learning Algorithms.

[13] W. Jiang, "Scalable ternary content addressable memory implementation using FPGAs," in Proc. ACM ANCS, San Jose, CA, USA, Oct. 2013, pp. 71–82.

[14] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "E-TCAM: An efficient SRAM-based architecture for TCAM," Circuits, Syst., Signal Process., vol. 33, no. 10, pp. 3123–3144, Oct. 2014.

[15] A. Ahmed, K. Park, and S. Baeg, "Resource-efficient SRAM-based ternary content addressable memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4, pp. 1583–1587, Apr. 2017.

[16] Mahesh, B. "Cost Optimization Techniques in Cloud Computing [J]." *International Journal of Computer Sciences & Engineering* 6.1 (2018): 375-380.

[17] Ternary Content Addressable Memory (TCAM) Search IP for SDNet: SmartCORE IP Product Guide, PG190 (v1.0), Xilinx, San Jose, CA, USA, Nov. 2017.

[18] V. Gherman and M. Cartron, "Soft-error protection of TCAMs based on ECCs and asymmetric SRAM cells," Electron. Lett., vol. 50, no. 24, pp. 1823–1824, 2014.