Fault Tolerant Ternary Content Addressable Memory Based Field Programmable Gate Array for Network Applications

N. Janani[1] Dr. T. V. P. Sundararajan[2] E. Mythili[3] N. Renugadevi[4]
[2]Professor [3]Assistant Professor
[1,2,3,4]Sri Shakthi Institute of Engineering and Technology, India

Abstract— Ternary Content Addressable Memory (TCAMs) are generally utilized in organize gadgets to execute bundle arrangement. They are utilized, for instance, for packet sending, for security, and to actualize Software Defined Networks (SDNs). TCAMs are usually actualized as independent gadgets or as a protected innovation obstruct that is coordinated on systems administration application-explicit incorporated circuits. On the other hand, Field-Programmable Gate Array (FPGAs) do exclude TCAM squares. In any case, the adaptability of FPGAs makes them appealing for SDN executions, and most FPGA sellers give improvement units for SDN. Those need to help TCAM usefulness and, in this manner, there is a need to copy TCAMs utilizing the rationale squares accessible in the FPGA. Lately, various plans to copy TCAMs on FPGAs have been proposed. Some of them exploit the enormous number of memory squares accessible inside present day FPGAs to utilize them to execute TCAMs. An issue when utilizing recollections is that they can be influenced by delicate mistakes that degenerate the put away bits. The recollections can be ensured with an equality check to recognize errors or with a mistake adjustment code to address them, yet this requires extra memory bits per word. In this concise, the insurance of the recollections used to imitate TCAMs is considered. Specifically, it is demonstrated that by misusing the certainty that lone a subset of the conceivable memory substance are substantial, most single-bit error can be remedied when the recollections are ensured with parity bits.

Keywords: Field Programmable Gate Array (FPGA), Soft Error, Ternary Content Addressable Memory(TCAM), Static Random Access Memory

I. INTRODUCTION

Soft errors are a significant worry for present day electronic circuits and, specifically, for memories [1]. A soft error can change the substance of the bits put away in a memory and cause a framework disappointment. The soft error rate in earthly applications is low. For instance, in [2], it was evaluated that for a 65-nm Static Random Access memory (SRAM) memory, the bit blunder rate was on the request for 10−9 errors per year. That would mean just a single mistake for every year for a framework that utilizations 1Gbit of memory. Be that as it may, even such a low error rate is a major worry for basic applications, for example, correspondence organizes on which the system components, for example, switches need to give a significant level of unwavering quality and accessibility. In this manner, soft errors are a significant issue when structuring switches or other system components, and producers consider and join error relief methods [3], [4]. For instance, mistake recognition what's more, redress codes are ordinarily used to ensure recollections [5]. An equality bit can be added to every memory word to recognize single-piece blunders, or a single Error correction(SEC) code can be utilized to address them. These codes require extra bits per word in this manner, expanding the memory size and furthermore some rationale to compose and peruse from the memory. For instance, for a 16-bit word, a SEC code requires 5 bits while an equality check requires just one.

Ternary Content addressable Memories (TCAMs) are a unique sort of substance addressable recollections [6] that help couldn't care less bits (generally indicated as "x") that match both a zero and a one. TCAMs are broadly utilized in systems administration applications to perform packet order [7]. They can be executed as independent gadgets or then again incorporated as a component of systems administration Application Specific Integrated Circuits (ASICs) [8]. The TCAM memory cells not the same as should be expected SRAM cells, in which they check the approaching an Incentive for a match to the put away worth that can be for each piece 0, 1, or x. The outcomes from every one of the words are then sent to a need encoder that profits the match with the most elevated need. This examination and determination rationale presents a huge overhead as far as zone and power utilization comparative with that of a SRAM memory. Securing TCAMs against delicate blunders is trying as Error Correction Codes (ECCs) are not effectively appropriate on the grounds that all words are checked in parallel. This implies a decoder for each word would be required, which would lead to an exceptionally enormous region and power overhead. Various plans have been proposed to secure TCAMs that are based, for instance on reproducing some portion of the standards or on utilizing different structures, for example, Blossom channels to check the consequences of the TCAM look [9], [10].

Field-Programmable Gate Arrays (FPGAs) give an adaptable stage to execute frameworks. Specifically, they give an immense sum of rationale and memory assets that can be designed to actualize a given usefulness. This makes them alluring for systems administration applications [11]. In any case, they do exclude CAM or TCAM squares on the grounds that FPGAs are additionally utilized in numerous applications that are definitely not identified with systems administration. For twofold CAMs that isn't an issue as they can be effectively imitated utilizing cuckoo hashing and RAM recollections with a little cost overhead [8]. TCAMs are additionally imitated utilizing the rationale and memory assets, yet for this situation, the overheads are a lot bigger (making copying not aggressive for ASIC usage). To copy the TCAMs in FPGAs, various plans have been proposed in the writing [12]–[16]. Some of them actualize the TCAM memory cells with FPGA flip-failures and rationale [12]. This approach has restricted adaptability as far as the TCAM size, and in this way, plans dependent on utilizing the SRAM recollections implanted in the FPGA [13]–[16] are liked and actualized by FPGA sellers [17].

When SRAM recollections are utilized to execute a TCAM, a huge number of bits are utilized for each TCAM cell. For instance, in [13], it has been demonstrated that in excess of 55 bits of the Xilinx FPGAs Block RAMs (BRAMs) are required for each single TCAM bit. In the event of appropriated RAMs, 6 bits are required for each TCAM bit.

This implies an enormous number of memory bits are utilized and in this manner the likelihood of enduring delicate mistakes increments. To secure them, ECCs can be utilized, however as talked about previously, they include extra memory overhead [18]. For TCAMs that are copied utilizing rationale what's more, flip-flops, security can be executed by utilizing triple measured repetition that triplicates the flip–flounders and adds casting a ballot rationale to right mistakes, in this manner requiring an enormous asset overhead. In this short, it is demonstrated that the explicitness of the substance put away in recollections used to copy TCAMs can be misused to execute a productive blunder remedy strategy. Specifically, at the point when recollections are ensured with an equality bit to recognize single bit errors, the proposed plan will have the option to address the greater part of the single-piece blunders. This makes the system alluring to improve the unwavering quality of FPGA-based TCAM usage without acquiring enormous overheads in asset use. In more detail, the proposed usage diminishes the FPGA cuts required for insurance by at any rate half when contrasted with a SEC insurance for normal TCAM sizes.

Rules used to store bits in memory is shown below:

r1:000xxx
r2:00x011
r3:1xx100
r4:not used

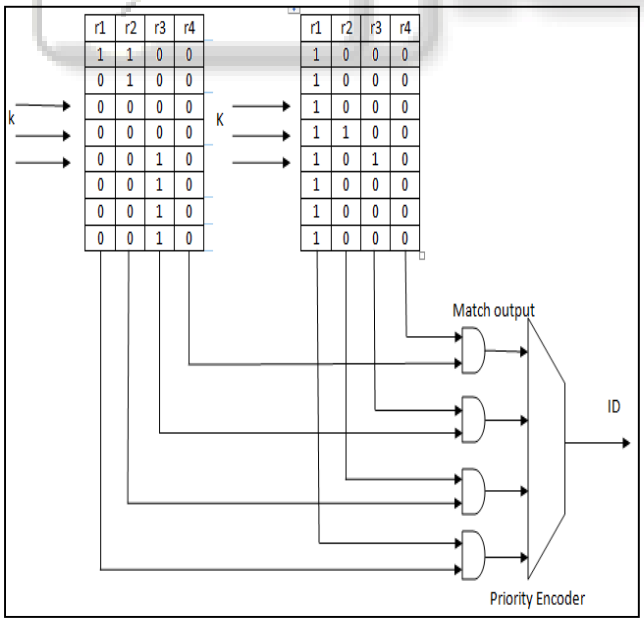

Fig. 1: TCAM with 6 bit Keys

This brief is sorted out as pursues. Segment II talks about FPGA-based TCAM usage. The proposed blunder remedy plan is displayed in Section III and assessed in Section IV. At last, the conclusion are abridged in Section V.

II. FPGA BASED TCAM MPLEMENTATIONS

There are two fundamental choices to actualize TCAMs on FPGAs. The first is to utilize the FPGA rationale assets and flip-lemon to execute the TCAM cells and match lines. The second is to utilize the square recollections inside the FPGA [13].

In the primary option, the bits of the principles are put away in flip-flops. As examined previously, each piece can take three potential qualities: 0, 1, what's more, x. For instance, a flip-failure can be utilized to store if the bit is 0 or 1 and another flip-flop that goes about as a cover and is set when the bit is couldn't care less [12]. At that point, the programmable rationale can be utilized execute the correlation against the key. This elective employments numerous assets per rule and, accordingly, can't be utilized to execute huge TCAMs with a huge number of standards of in excess of 100 bits that work at fast.

The subsequent option depends on the utilization of the inserted recollections accessible in the FPGA. To do as such, the key is partitioned into littler squares of b bits. At that point, a standard can be imitated utilizing a 1-piece memory of 2b positions for each square. When looking for a key, all the recollections are gotten to utilizing the relating key bits and assuming all the positions read have a one, a match is identified. As a rule, k rules can be actualized by utilizing a k − bit memory of 2b positions for each square. This is best outlined with a model. Allow us to consider a key of 6 bits that is separated into two squares of 3 bits. At that point, a TCAM with four principles can be executed as appeared in Fig. 1. It tends to be seen that the recollections have 23 = 8 positions and a width of 4 bits. The furthest left memory is gotten to utilizing the upper 3 bits of the key and the other with the lower three. Those bits are used to decide the location of the position read from the memory. The principles put away in each piece are likewise appeared in "Fig. 1". Allow us to consider a quest for key: 000011. We would get to the main position (address 000) on the furthest left memory perusing 1100 and the four position (address 011) on the other memory perusing 1100. Subsequent to performing Also, there would be a match just for rules r1 and r2. Looking at this point at the guidelines, it very well may be seen that decides that are not utilized (r4) have zeros in every one of the recollections and positions. For the remainder of the guidelines, the quantity of ones of every a given memory relies upon the quantity of x bits that the standard has on the key bits utilized as addresses on that memory. When there are no x bits, just one position has a one, when there is one x bit, two positions have a one, when there are two x bits, four positions have a one, etc. When all is said in done, if there are nx bits that are x, there will be 2nx ones on the memory.

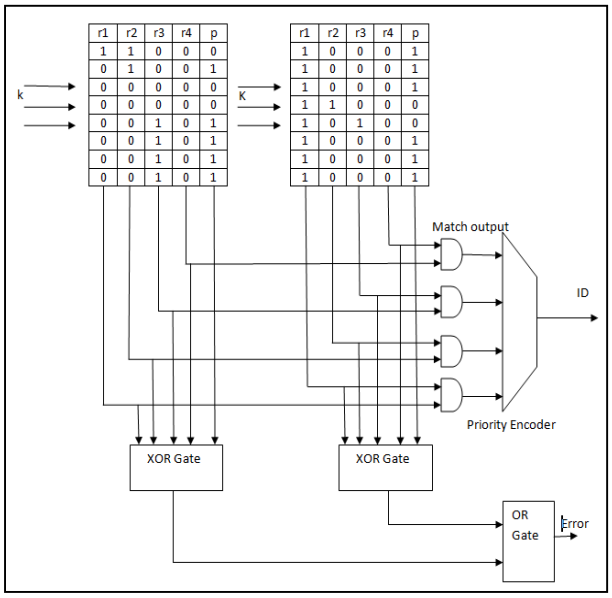

Fig. 2: Parity Protected TCAM with 6 bit keys

Considering now the execution cost, since each square stores b bits of a standard and requires 2b bits of SRAM memory. The expense in SRAM bits per TCAM bit in this plan is 2b/b [13]. In this way, no doubt littler estimations of b are increasingly effective. Be that as it may, this isn't totally valid as the rationale expected to consolidate the squares together increments with the quantity of squares. It ought to likewise be referenced that a huge physical memory can be part into a few squares with the goal that everyone executes b bits of a standard. At that point, a few memory gets to are expected to finish a hunt activity, this can be relieved by working the memory at a bigger speed or utilizing multiport recollections [16].

For Xilinx FPGAs, there are two sorts of memory assets: query table arbitrary access recollections (LUTRAMs) and BRAMs. The initial ones are worked with a similar query tables (LUTs) that are utilized to actualize the rationale and are commonly little with 32 or 64 positions. Then again, BRAMs are bigger having 36 K bits that can be designed with various word estimates, the biggest being 72 bits that compares to 512 positions. Along these lines, LUTRAMs have a much lower cost for every piece (25/5) than BRAMs (29/9). In any case, the all-out number of memory bits accessible is bigger for BRAMs than for LUTRAMs.

A key perception for the assurance of the SRAM-based TCAM usage is that the substance of the SRAMs are resolved by the guidelines put away and that lone a couple of mixes of all conceivable qualities are utilized. This implies the SRAM substance have a characteristic repetition that could possibly be utilized to ensure the recollections. This thought is investigated in the remainder of this brief.

III. ERROR DETECTION AND CORRECTION IN SRAM BASED TCAMS

The plan proposed to secure the recollections used to copy the TCAM utilizes a for each word equality bit to identify single-bit error. At that point, when a blunder is distinguished, the characteristic excess of the memory substance is utilized to attempt to address the error. The execution of the equality security is appeared in "Fig. 2" where p compares to the equality bit. It tends to be seen that notwithstanding the match signal, a mistake sign is produced when there is a crisscross between the put away equality and the recomputed one. This is a standard equality security that can recognize all single-piece blunders [5]. Distinguishing the blunder on each entrance is pivotal to maintain a strategic distance from inaccurate outcomes on search tasks.

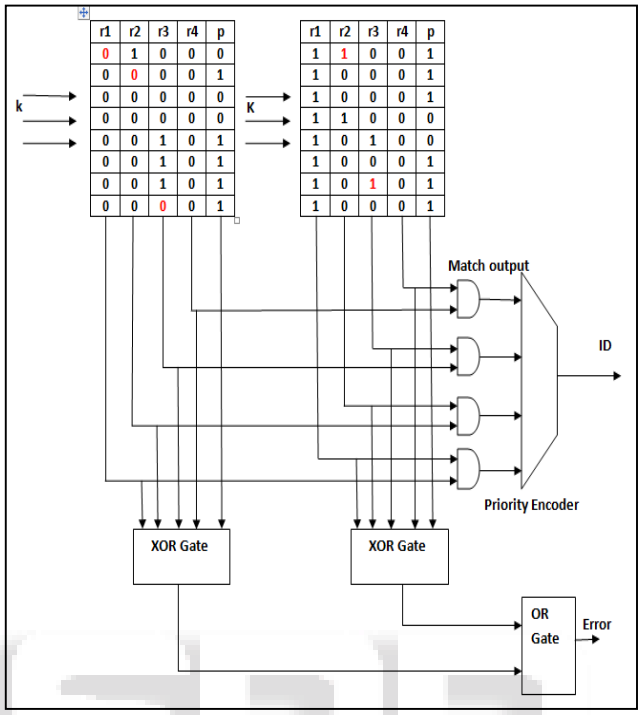

Fig. 3: Single but error in parity protected TCAM with 6 bit keys

Let us now a chance to accept that a single bit error has happened on a given word and that it is distinguished with the equality check. Upon mistake recognition, we can check the substance of the memory to attempt to address the error. A first endeavor could be to peruse every one of the words in the memory also, tally the quantity of places that have a one for each standard. Let us indicate that number as the heaviness of the standard in that memory. For model, in the furthest left memory of "Fig. 2", r1 would have a weight of 1, r2 of 2, and r3 of 4. This can enable us to recognize the incorrect piece as the weight for a mistake free principle must be 0, 1, 2, 4, and 8 for an 8-position memory. To further talk about the blunder amendment process, give us a chance to concentrate on the instances of single-piece blunders appeared in "Fig. 3". For instance, third error influences r3 on the furthest left memory by changing its weight from 4 to 3. Since 3 is certainly not a legitimate worth, subsequent to distinguishing the equality mistake, we would recognize that the wrong bit is that in r3 furthermore, we would address it. This methodology would be viable for rules that have a weight bigger than two, i.e., they have at least two "x" bits on the key bits that compare to that memory. On the other hand, for rules with a lower weight, checking the weight alone may not be sufficient. Give us now a chance to think about a standard with weight two. At that point, an error that changes a zero to a one will change the weight to three furthermore, the error will be redressed. Be that as it may, when a one is changed to a zero (as in second error), at that point the new weight would be one that is a substantial esteem and the error

can't be amended. This, nonetheless, is more outlandish to happen as just 2 positions have a one. On the off chance that we currently think about a weight one principle, a error that sets another piece to one would deliver a weight of two that is additionally legitimate. In any case, not all weight two blends are conceivable. This is obviously observed when taking a gander at fourth error. All things considered, the estimations of r2 that are one would compare to key qualities 000 and 011 and those don't compare to a legitimate principle. Then again, an error that sets to zero the position that was one out of a weight one guideline can be amended by checking if the standard has zero load on the other recollections. In the event that that is the situation, at that point the standard is debilitated and the bit is not in error. At long last, an error in a standard that had a load of zero can additionally be redressed by checking the heaviness of the standard on the other recollections.

Weight/b	5	6	7	8	9
1	100	100	100	100	100
2	84.5	90.9	94.7	97	98.7
3	94	97.1	98.5	99.5	99.7
>=4	100	100	100	100	100

Table 1: Percentage of Correctable Error For Block Memories

The past discourse demonstrates that by utilizing the characteristic excess of the memory substance, many single bit error examples could be remedied. Give us now a chance to measure the part of single-bit error designs that can be redressed for each weight in a memory of 2b positions.

1) Weight 0: all errors can be corrected.
2) Weight 1: all except from those that set a bit to one for a position with a address at distance one, this compares to 1 − b/2b.
3) Weight 2: all errors can be corrected with the exception of the two that set a position with a one to a zero, this relates to 1−2/2b .
4) Weight >=4: all errors can be corrected.

It very well may be seen that the vast majority of the error examples are redressed. The main situations where not all errors can be adjusted are weight one and two, and for those, the rate will approach 100% when b is enormous. The rate of blunders that can be adjusted for various estimations of b is appeared in Table I. It very well may be seen that in any event, for little recollections (b = 5 relates to 32 positions), the mistake inclusion is near 90% in the assuming the worst possible scenario. For bigger recollections, the inclusion is over 95% and gets near 100%. For instance, for b = 9, the inclusion is over 98% in the most pessimistic scenario. This demonstrates the viability of the proposed plan in remedying single piece blunders when the recollections are secured with an equality bit.

IV. EVALUATION

To assess the advantages of the proposed plan, it has been actualized utilizing Viva do Design Suite 2016.3 in a Xilinx Artix-7 xc7a100tcsg324 FPGA that is a piece of a Nexys4 twofold information rate board. Both appropriated memory (LUTRAMs) and BRAMs executions are considered. In the two cases, the memory estimates that limit the memory bits required per TCAM bit are utilized. This relates to a setup of 32 positions LUTRAMs comparing to five key bits for each memory, and for the subsequent choice, the BRAM is arranged as 512 places of 72 bits.

For BRAMs, every memory has 512 places that can cover 9 bits of the key, and in this way, just five recollections are required. it very well may be seen that the proposed plan lessens by the greater part the measure of assets required both as far as LUTs utilized for rationale and cuts contrasted with a SEC insurance, For the situation of LUTRAMs and BRAMs, the decreases are lower. This can be clarified as the quantity of equality check bits expected to execute SEC isn't enormous, and moreover, for BRAMs, they have a 72-piece width so that in all cases there are extra bits that can be utilized for the equality check bits. Another preferred position of the proposed plan is that it has a lower sway on delay than the utilization of SEC insurance. At last, it is fascinating to relate the assets utilized with those accessible in the FPGA. In more detail, the gadget utilized has 63 400 LUTs of which just 19 000 can be designed as LUTRAMs and 135 BRAMs. This implies a 2048 × 40 TCAM utilizes practically all LUTRAMs in the LUTRAM execution while account of BRAMs, all BRAMs are utilized and part of the memory squares should be imitated utilizing LUTRAMs. In the two cases, a SEC ensured execution leaves many less LUTs to actualize the rest of the framework than the proposed plan.
In summary, the proposed plan can be an intriguing choice to improve the security of the recollections used to copy TCAMs on FPGAs. Be that as it may, it ought to be noticed that the proposed procedure doesn't right all single bit errors and that it requires some overhead in the product controller of the TCAM to perform revision once an mistake is identified. Consequently, regardless of whether to utilize it or not will rely upon the structure prerequisites as far as dependability and on the assets accessible.

V. CONCLUSION

In this brief, a procedure to secure the SRAMs used to copy TCAMs on FPGAs has been proposed. The plan depends on the perception that not all qualities are conceivable in those SRAMs, and in this way, there is some characteristic repetition of the memory substance. This excess is utilized to address most single-piece mistake designs when the recollections are secured with an equality bit to distinguish mistakes. The proposed method decreases fundamentally the assets expected to secure the recollections and can be an intriguing alternative for structures on which dependability is a worry yet assets are constrained.

The thought exhibited in this brief can be stretched out to other memory designs. For instance, it tends to be utilized to identify mistakes on an unprotected memory by occasionally scouring the substance to check their accuracy. It could likewise be utilized when the memory is secured with an all the more dominant code that can identify a several bit error to address multi bit error. For instance, for a memory ensured with an SEC code, two bit errors examples could be distinguished and after that utilization the inborn excess of the memory content.

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), pp. 1–7, Mar. 2012.
[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] P. Bosshart et al., "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," in Proc. ACM SIGCOMM, pp. 99–110,2013.
[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), pp. 1474–1479, Mar. 2013,.
[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] 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] W. Jiang, "Scalable ternary content addressable memory implementation using FPGAs," in Proc. ACM ANCS, San Jose, CA, USA, pp. 71–82, Oct. 2013.
[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] I. Ullah, Z. Ullah, and J.-A. Lee, "Efficient TCAM design based on multipumping-enabled multiported SRAM on FPGA," IEEE Access, vol. 6, pp. 19940–19947, 2018.
[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.

