JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

THREE DIMENSIONAL PARITY FORTCAM BASED FPGA FOR ERROR DETECTION AND CORRECTION

Ms.L.Pavithra, Mrs.S.Prabhavathy, Mrs.V.Vaishnavi,Mrs.V.Nithya
Assistant Professor, Department of ECE, CKEC, Coimbatore
Assistant Professor, Department of ECE, CKEC, Coimbatore
Assistant Professor, Department of ECE, CKEC, Coimbatore
Assistant Professor, Department of ECE, CKEC, Coimbatore

Abstract* - Delicate mistakes are a significant concern for current electronic circuits and, specifically, for recollections. A delicate blunder can change the substance of the pieces put away in a memory and cause a framework disappointment. Ternary content addressable memories (TCAMs) are generally utilized in network gadgets to execute bundle characterization. They are utilized, for instance, for bundle sending.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. In any case, the adaptability of FPGAs makes them appealing for SDN executions, and most FPGA sellers give improvement packs to SDN. Those need to help TCAM usefulness and, hence, there is a need to imitate TCAMs utilizing the rationale blocks accessible in the FPGA. As of late, various plans to imitate 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. The recollections can be secured with an equality check to identify mistakes or with a blunder revision code to address them, yet this requires extra memory bits per word. For detecting MBUs in the configuration frames of the FPGA, we propose a technique, namely, interleaved n dimensional (InD) parity. Furthermore, by carefully dividing the frames into several clusters, the proposed technique can detect and correct an MBU affecting several adjacent configuration frames.

***Index Terms* – Field-programmable gate arrays (FPGA),soft errors, ternary content addressable memories (TCAM).**

I.INTRODUCTION:

Ternary Content addressable memories (TCAMs) are broadly utilized in network gadgets to carry out parcel grouping. They are utilized, for instance, for parcel sending, for security, and to execute programming characterized networks (SDNs). TCAMs are normally executed as independent gadgets or as a licensed innovation block that is incorporated on systems administration application-explicit coordinated circuits. Then again, field-programmable entryway exhibits (FPGAs) do exclude TCAM blocks. Be that as it may, the adaptability of FPGAs makes them appealing for SDN executions, and most FPGA merchants give advancement units to SDN. Those need to help TCAM usefulness and, accordingly, there is a need to imitate TCAMs utilizing the rationale blocks accessible in the FPGA. As of late, various plans to imitate TCAMs on FPGAs have been proposed. Some of them exploit the huge number of memory blocks accessible inside current FPGAs to utilize them to execute TCAMs. A difficult when utilizing recollections is that they can be influenced by delicate mistakes that bad the put away pieces. The recollections can be secured with an equality check to identify blunders or with a mistake rectification code to address them, however this requires extra memory bits per word. In this concise, the assurance of the recollections used to imitate TCAMs is thought of. Specifically, it is shown that by misusing the way that solitary a subset of the conceivable memory substance are substantial, most single-digit mistakes can be adjusted when the recollections are secured with an equality bit.

Delicate blunders are a significant worry for current electronic circuits and, specifically, for recollections [1]. A delicate blunder can change the substance.of the pieces put away in a memory and cause a framework disappointment. The delicate mistake rate in earthly applications is low. For instance, in [2], it was assessed that for a 65-nm static irregular access memory (SRAM) memory, the cycle mistake rate was on the request for 10^{-9} blunders per year. That would mean just a single mistake each year for a framework that utilizes 1 Gbit of memory. Nonetheless, even a low blunder rate is a major worry for basic applications like correspondence networks on which the organization components, for example, switches need to give an undeniable degree of dependability and accessibility. Accordingly, delicate mistakes are a significant issue when planning switches or other organization components, and makers consider them and fuse mistake alleviation procedures [3], [4]. For instance, mistake discovery furthermore, amendment codes are normally used to ensure recollections [5]. An equality cycle can be added to every memory word to identify single-digit mistakes, or a solitary blunder amendment (SEC) code can be utilized to address them. These codes require extra pieces per word hence, 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 pieces while an equality check requires just one.

Ternary substance addressable recollections (TCAMs) are an uncommon sort of substance addressable recollections [6] that help couldn't care less bits (generally signified as "x") that match both a zero and a one. TCAMs are generally utilized in systems administration applications to perform parcel characterization [7]. They can be executed as independent gadgets or then again coordinated as a component of systems administration application explicit incorporated circuits (ASICs) [8]. The TCAM memory cells unique in relation to ordinary 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 shipped off a need encoder that profits the match with the most noteworthy need. This correlation and choice rationale presents a critical overhead as far as zone and force utilization comparative with that of a SRAM memory. Ensuring TCAMs against delicate mistakes is trying as blunder remedy codes (ECCs) are not effectively material since all words are checked in equal. This implies that a decoder for every word would be required, which would lead to a huge region and force overhead. Various plans have been proposed to ensure TCAMs that are based, for instance, on recreating part of the principles or on utilizing different designs.

Field-programmable door clusters (FPGAs) give an adaptable platform to execute frameworks. Specifically, they give a huge sum of rationale and memory assets that can be designed to carry out a given usefulness. This makes them appealing for systems administration applications [11]. Not with standing, they do exclude CAM or TCAM blocks in light of the fact that FPGAs are likewise utilized in numerous applications that are definitely not identified with systems administration. For double CAMs that isn't an issue as they can be effectively copied utilizing cuckoo hashing and RAM recollections with a little expense overhead [8]. TCAMs are additionally copied utilizing the rationale and memory assets, however for this situation, the overheads are a lot bigger (making imitating not serious for ASIC executions). To copy the TCAMs in FPGAs, various plans have been proposed in the writing [12]–[16]. Some of them execute the TCAM memory cells with FPGA flip-lemon and rationale [12]. This approach has restricted versatility regarding the TCAM size, and subsequently, plans dependent on utilizing the SRAM recollections inserted in the FPGA [13]–[16] are liked and carried out by FPGA merchants [17].

At the point when SRAM recollections are utilized to execute a TCAM, an enormous number of pieces are utilized for each TCAM cell. For instance, in [13], it has been shown that in excess of 55 pieces of the Xilinx FPGAs block RAMs (BRAMs) are required for each single TCAM bit. In the event of dispersed RAMs, 6 pieces are required for each TCAM bit.

This implies that an enormous number of memory pieces are utilized and consequently the likelihood of enduring delicate mistakes increments. To ensure them, ECCs can be utilized, however as examined previously, they add extra memory overhead [18]. For TCAMs that are imitated utilizing rationale furthermore, flip-flops, security can be executed by utilizing triple measured repetition that sets of three the flip–tumbles and adds casting a ballot rationale to right mistakes, along these lines requiring an enormous asset overhead.

In this concise, it is shown that the particularity of the substance put away in recollections used to imitate TCAMs can be abused to execute a proficient mistake adjustment strategy. Specifically, at the point when recollections are ensured with an equality touch to distinguish single-piece blunders, the proposed plan will actually want to address a large portion of the single-piece blunders.

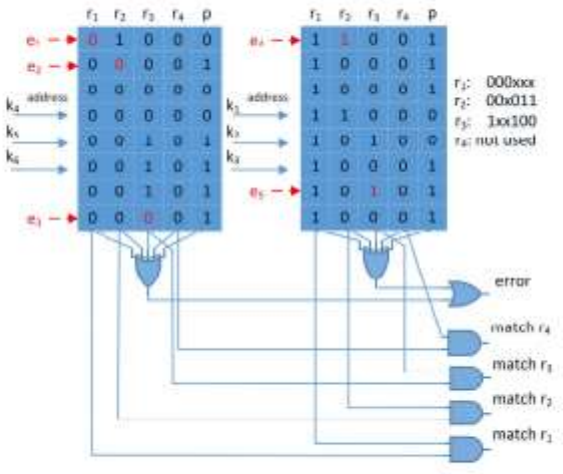


Fig. 1. Examples of single-bit errors on a parity protected TCAM with 6-bit keys and four rules emulated using two SRAMs.

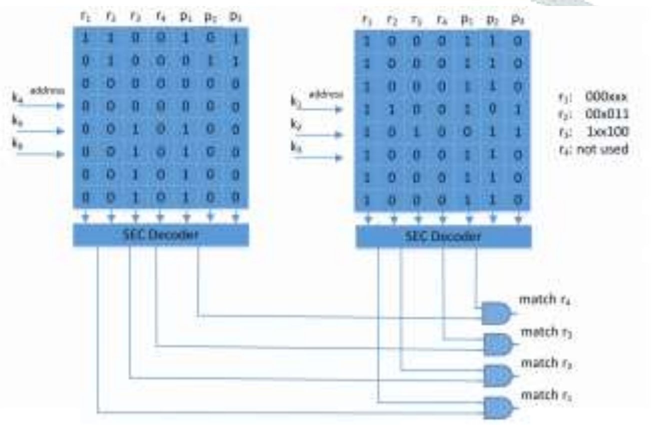


Fig. 2. SEC protected TCAM with 6-bit keys and four rules emulated using two SRAMs.

II.ERROR DETECTION AND CORRECTION IN SRAM-BASED TCAMs

Allow us presently to expect that a solitary piece blunder has happened on a given word and that it is recognized with the equality check. Upon blunder identification, we can check the substance of the memory to attempt to address the blunder. 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 mean 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 assist us with recognizing the mistaken piece as the load for a mistake free guideline must be 0, 1, 2, 4, and 8 for an 8-position memory. To additionally examine the mistake remedy measure, allow us to zero in on the instances of single-piece blunders appeared in Fig. 3. For instance, e3 influences r3 on the furthest left memory by changing its weight from 4 to 3. Since 3 is certifiably not a substantial worth, in the wake of recognizing the equality blunder, we would distinguish that the wrong piece is that in r3also, we would address it. This methodology would be powerful for rules that have a weight bigger than two, i.e., they have at least two "x" bits on the key pieces that compare to that memory. On the other hand, for rules with a lower weight, checking the weight alone may not be sufficient. Allow us presently to think about a standard with weight two. At that point, a mistake that changes a zero to a one will change the load to three also, the mistake will be adjusted. In any case, when a one is changed to a zero (as in e2), at that point the new weight would be one that is a substantial esteem and the mistake can't be adjusted. This, in any case, is more outlandish to happen as just 2 positions have a one. On the off chance that we presently think about a weight one principle, a mistake that sets another piece to one would create a weight of two that is likewise substantial. In any case, not all weight two mixes are conceivable. This is obviously seen when taking a gander at e4. All things considered, the estimations of r2 that are one would relate to key qualities 000 and 011 and those don't relate to a legitimate standard. All in all, as it were places that relate to key qualities that are at distance one from the first worth won't be recognized. Then again, a mistake 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. Assuming that is the situation, the standard is incapacitated and the piece is not in blunder. Something else, the standard had a load of one and the blunder is remedied. At last, a mistake in a standard that had a load of zero can additionally be amended by checking the heaviness of the standard on different recollections.

The past conversation shows that by utilizing the inborn redundancy of the memory substance, many single-digit blunder examples could be remedied. Allow us presently to measure the small part of single-piece mistake designs that can be remedied for each weight in a memory of 2b positions.

1) Weight zero: everything examples can be adjusted.

2) Weight one: all aside from those that set a piece to one for a position

with a location at distance one, this relates to 1 − b/2b.

3) Weight two: everything examples can be rectified with the exception of the two that set a situation with a one to a zero, this relates to 1−2/2b .

4) Weight four or bigger: everything examples can be adjusted.

It very well may be seen that the majority of the mistake designs are revised. This is better found in Table I that delineates the level of correctable designs for segments of various loads. The lone situations where not everything blunders can be amended are weight one and two, and for those, the rate will move toward 100% when b is enormous. The rate of blunders that can be revised for various estimations of b is appeared in Table II. 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 thinking pessimistically. For bigger recollections, the inclusion is more than 95% and gets near 100%. For instance, for b = 9, the inclusion is more than 98% in the most pessimistic scenario. This shows the viability of the proposed plot in remedying single piece blunders when the recollections are secured with an equality bit.

The pseudocode of the proposed amendment calculation is appearedin Algorithm 1. The cycle begins when an equality mistake is distinguished when persuing a word from a square memory. To address the blunder, we need to recognize the piece (or section) influenced by the blunder. To do thus, in the main stage, every one of the situations in the square are perused and the segment loads are figured by adding the ones found in each segment. At that point, the subsequent stage checks various cases for the segment weight to attempt to recognize the mistaken section. On the off chance that that happens, the piece of that section in the word that had the equality blunder is the mistaken spot and it is rectified. In the calculation, this subsequent stage begins by checking if there is a section that has an unlawful weight. As examined previously, the lone legitimate segment loads are: 0, 1, 2i for I = 1, 2,… , b. Hence, on the off chance that a segment has, for instance, weight three, it is the wrong one. In the event that the incorrect piece is discovered, it is rectified and the measure closes. Else, we continue to check sections that have zero weight. Those ought to compare to TCAM passages that are definitely not utilized and ought to have zero load on the wide range of various memory blocks. Accordingly, we check in the event that they have additionally zero load on another square. If not, the blunder has been found and it is revised. On the off chance that every one of the segments with zero weight have likewise weight zero on another square, we proceed to check sections of weight one. For that, we check on the off chance that they have zero weight on another square. On the off chance that that is the situation, that section is the one that endured the mistake and we right it. If not, we continue to the last step in which segments of weight two are checked. To do that, the two addresses of the two places that contain a one are XORed. On the off chance that the result has in excess of a one, the section has endured a mistake and we right it. Assuming that doesn't occur, we have endured one of the barely any mistakes that are not correctable. The general cycle can undoubtedly be executed in a delicate processor that is available in numerous FPGA-based organizing applications to deal with the control capacities [11].

III.MULTI-BIT ERROR DETECTION AND CORRECTION USING THREE DIMENSIONAL PARITY FOR TCAM BASED FPGA

The main idea of the proposed work is to exploit erasure codes to recover the contents of the erroneous configuration frame. Nevertheless, since eliminated codes cannot detect errors, an effective detection technique is required as well. Hence, every arrangement outline must be furnished with a minimal effort blunder recognition code. A scrubber unit regularly investigates the configuration frames for feasible errors. Once an error is detected, by assuming that the erroneous frame is erased, its contents are recovered using. Considering the way that the whole arrangement edge could be recuperated utilizing an eradication code, the recognizable proof of the specific area of wrong pieces isn't of our premiumt,rather an ease mistake discovery strategy with a high location inclusion is required. In such manner, we present an exceptionally effective mistake identification coding strategy called InD equality for MBU location in the design outlines.

In memory arrays of typical microprocessors such as cache units, the parity bits working for error detection in each memory entry (i.e., word) are enumerate during each memory access. Hence, from the performance perspective, it is crucial that each memory entry has its own error detection coding. Otherwise, during each memory access, all other entries that are involved in the computation of the corresponding common parity bit(s) must be obtain as well. However, the errors in FPGAs are detected during the regular scrubbing where the error checking unit can access the entire contents of a configuration frame. Therefore, a parity bit could be enumerate based on multiple entries. In the proposed error detection technique, we exploit the fact that the sizes of large MBU patterns are typically much smaller than the size of a configuration frame. Since an MBU incident affects several bits in a localized manner, the bits that are find far enough cannot be simultaneously affected with one MBU incident. Therefore, having separate parities for such bits in configuration frames neither increases the error detection capability nor improves the performance, rather only imposes unnecessary area overhead.

In order to increase the cost-efficiency of error detection for the configuration frames, we introduce the idea of InD parity. The main idea is to use the same parity bit for the bits that are separated by a constant distance to minimize the area overhead (i.e., interleaved parity). In addition, the parity bits are distributed at several elements to increase the detection coverage with respect to probable MBU patterns as there are some MBU patterns that cannot be find using one or two elements, no matter how many parity bits are employed. The number of parity bits in each element theoretically has to be at most equal to the largest MBU spread on that element. However, in practice, large MBU patterns are typically find using the parity bits on the other dimensions. Consequently, the number of parity bits required by this technique is always compact than this theoretical limit. It should be mentioned that the proposed InD parity technique is conceptually different from the existing error detection techniques used for an interleaved memory such as cache units.

In the last case, each word has its own blunder discovery code and furthermore, the memory words are genuinely interleaved to diminish the progressions of having more than one incorrect bit in each word for an MBU incidence. In any case, this doesn't diminish the zone overhead of the mistake recognition strategy. Conversely, the proposed InD equality coding is a virtual interleaving strategy, which doesn't need any progressions to the orchestrated edge structure. What's more, on account of an enormous setup outline, the quantity of repetitive pieces could be definitely diminished in examination with the total interleaved equality coding.

In the complete (traditional) 2-D parity, a parity bit is associated for each row (column) which is constructed by XORing all the bits in that particular row (column). In the I2D parity technique, each horizontal (vertical) parity bit is the XOR of the bits in multiple rows (columns). All the more explicitly, all lines (sections) that are isolated by a steady distance of h (v) structure an interleaving gathering and every one of the pieces inside that interleaving bunch are covered by only one horizontal (vertical) parity bit. For a f × g memory array, the complete 2-D parity have f +g parity bits while I2D parity requires only h + v parity bits. In addition, the number of XOR operations in both schemes is of the same order.

In general, there might be some MBU patterns, which affect an even number of cells in each row and column. This sort of examples can't be distinguished by the I2D equality method since equality spot can just identify an odd number of mistaken pieces. Examples of MBU patterns, in this technology, with significant occurrence probabilities that cannot be find by either I2D or traditional 2-D parity technique.To distinguish such MBUs, an all the more impressive MBU discovery strategy is required. In such manner, we propose I3D equality procedure that has an extra arrangement of equality bits for diagonals.

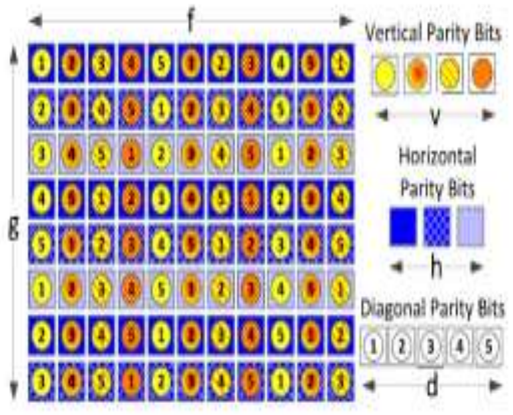


Fig.3. Example of I3D with vertical, horizontal, and diagonal distance of 4, 3, and 5, respectively.

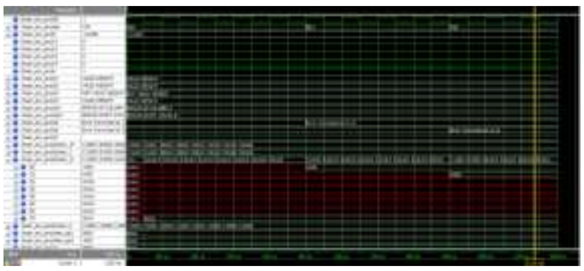
Computation of the horizontal and the vertical parity bits in I3D parity follows the same rules explained for I2D parity. Similar to the interleaving technique employed in I2D parity to reduce the number of horizontal and vertical parity bits, in I3D parity, several interleaving groups are formed for the diagonals as well. Then for each group, only one parity bit is computed. In order to uniformly distribute d diagonal parity bits for a configuration frame of size g × f , the interleaving group for the bit position of (i, j) could be computed by [i+ f ·(j−1)] mod d. An example of the I3D parity technique with the vertical, horizontal, and diagonal interleaving distance of 4, 3, and 5, respectively, is shown in Fig. 3.

IV.Evaluation

ModelSim is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be utilized freely, or related to Altera Quartus or XilinxISE. simulation is performed utilizing the graphical UI (GUI), or naturally utilizing contents. The ModelSim environment is shown in Fig.4.2.ModelSim is offered in multiple editions, such as ModelSim PE, ModelSimSE, and ModelSim XE. ModelSim SE offers superior and progressed investigating abilities, while ModelSim PE is the section level test system for specialists and understudies. ModelSim SE is utilized in enormous multi-million door plans, and is upheld on Microsoft Windows and Linux, in 32-bit and 64-digit designs.

ModelSim XE represents Xilinx Edition, and is uniquely intended for incorporation with Xilinx ISE. ModelSim XE empowers testing of HDL programs composed for Xilinx Virtex/Spartan arrangement FPGA's without required actual equipment.ModelSim can likewise be utilized with MATLAB/Simulink, utilizing Link for ModelSim. Connection for ModelSim is a quick bidirectional co-recreation interface among Simulink and ModelSim. For such plans, MATLAB gives a mathematical simulation toolset, while ModelSim gives devices to check the equipment execution and timing qualities of the plan. ModelSim facilitates the way toward discovering configuration absconds with a shrewdly designed troubleshoot climate. The ModelSim investigate climate effectively shows plan information for examination and troubleshoot of all dialects. ModelSim permits many troubleshoot and investigation capacities to be utilized post-recreation on saved outcomes, just as during live simulation runs.

V.RESULTS:



simulation result for proposed multi-bit error detection and correction

VI.Conculsion

A technique to protect the SRAMs used to emulate TCAMs on FPGAs is 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. For detecting MBUs in the TCAM, a low cost technique, using three dimensional parity analyses is used. Along with row and column detection, the diagonal values are also considered which leads to more error detection and correction. The idea presented in the proposed method 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.

Reference:

[1] 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.

[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] P. Bosshart et al., “Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN,” in Proc. ACM SIGCOMM, 2013, pp. 99–110.

[4] 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.

[5] 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.

[6] V. Gherman and M. Cartron, “Soft-error protection of TCAMs based on ECCs and asymmetric SRAM cells,” Electron.Lett., vol. 50, no.

[7] 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.

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

[9] 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.

[10] 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.

[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] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.