Available *online* at : http://jnte.ft.unand.ac.id/

Jurnal Nasional Teknik Elektro

| ISSN (Print) 2302-2949 | ISSN (Online) 2407-7267 |

Optimized Multiple-Bit-Flip Soft-Errors-Tolerant TCAM using Machine Learning

Infall Syafalni [1] *and Trio Adiono* [2]

[1] *School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, Indonesia*
[2] *University Center of Excellence on Microelectronics, Bandung Institute of Technology, Bandung, Indonesia*

ARTICLE INFORMATION

Received: February 11, 2022
Revised: March 25, 2022
Available online: March 29, 2022

KEYWORDS

TCAM, Soft Error, X-Key, Machine Learning

CORRESPONDENCE

E-mail: infall.s@itb.ac.id

A B S T R A C T

Soft errors from radiations can change the data in electronic devices especially memory cells such as in TCAMs. The soft errors cause bit-flip errors that makes the data are corrupted in the network. This paper presents a novel machine learning for a multiple-bit-flip-tolerant TCAM that address soft errors problem using partial don't-care keys (X-keys). The general methodology is classified into two steps, *i.e.*, statistical training and X-keys matching. First, we train the machine by collecting match probability of a filter by using X-keys that match the same locations as the search key. This method uses statistical training to determine the most efficient of number of don't cares. Moreover, in the statistical training, we also explore the maximum number of don't cares that produce best performance in covering the soft errors. Finally, the X-keys are implemented in the TCAM to correct bit-flip errors. The suitable number of don't cares in X-key is determined from the distribution of match probability of the X-keys so that the best degree of tolerance of the TCAM against soft errors is found. Match probabilities for various filters are shown. Experimental results demonstrate that the soft-error tolerance using statistical data has better soft-error tolerance than other methods. The proposed method is useful for soft-error tolerant TCAMs in routers and firewalls for robust networks.

INTRODUCTION

In the era of artificial intelligent and internet of things (IoT), the demands for high-reliability circuits are constantly increasing. For applications, such as financial transactions, aerospace communications, and military based networks, fault-tolerant circuits are indispensable to avoid faulty data.

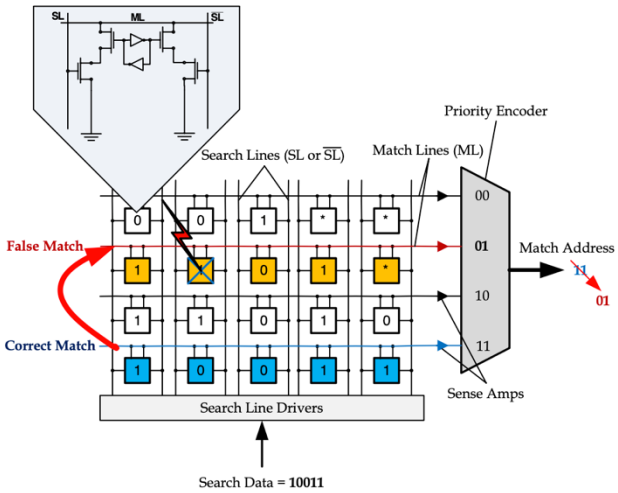

Figure 1 Soft Error in TCAM

In networks, a special type of memory called ternary content addressable memory (TCAM) is often used to perform packet forwarding, access controlling, and filtering [1], [2]. A TCAM has three values, namely 0, 1, and don't-care. Fig. 1 illustrates a TCAM. It compares the input vector with the entire list of registered vectors simultaneously [3]. A TCAM is a *de facto* standard in routers and devices for packet classification in high-speed network applications [2]. Fig. 1 shows a cell in a TCAM. The search bits $(SL, \overline{SL})$ are compared with stored bits. When a match occurs, the match line (ML) sends a signal to the priority encoder to produce the match address. However, a TCAM can be attacked by soft errors caused by radiations that changes the data of the memory cells [4], [5]. The soft errors cause bit-flip errors that makes the data in the network corrupted. Moreover, some hardware applications of soft error tolerant TCAMs using SRAM and GPU are presented in [6] and [7], respectively.

This work is based on partial don't care keys used for detecting bit-flip errors. A novel method by using machine learning is proposed for improving the tolerance performance and the execution speed. The **major contributions** of this work are as follows: 1) The distribution of the match probability of a filter is found; 2) A suitable number of don't care bits for X-keys is determined by the mean from the statistical data; 3) The proposed method uses only one TCAM and requires no modification of the TCAM.

The organization of the paper is as follows: First, we explain the definition and the basic properties of classification function, soft-error in TCAMs and partial-don't-care keys that are used in the

https://doi.org/10.25077/jnte.v11n1.1007.2022

research. Second, we explain the statistical properties of partial-don't-care keys that is used to optimize the parameters for resolve the bit flips caused by the soft errors. Third, we explain the proposed method which is the random X-keys for bit-flip errors detection using machine learning. Finally, we show the experimental results and conclude the paper.

DEFINITIONS AND BASIC PROPERTIES

First, we explain the definition and the basic properties of classification function, soft-error in TCAMs and partial-don't-care keys that are used in the research.

Classification Functions

A TCAM consists of words, each of them representing a packet classifier rule [2].

Definition 1: A **classification function** with s **fields** is a mapping $F: P_1 \times P_2 \times \cdots \times P_s \rightarrow \{0, 1, 2, \cdots, r\}$ where $P_i = \{0, 1, \cdots, 2^{t_i-1}\}$ (i $= 1, 2, \cdots,$ s). F is specified by a set of r rules. A **rule** consists of s fields, and each field is specified by an interval of t_i bits.

A packet classifier is specified by source and destination addresses, source and destination ports, and the protocol type [2]. The works on the representation of classification functions are [8], [9].

Soft Errors in a TCAM Cell

Definition 2: A **soft error** in a TCAM cell is a non-permanent error that changes the value of the cell temporarily and may cause misclassification.

Definition 3: A **single-bit-flip** error or **multiple-bit-flip** errors is (are) a change of value(s) of a TCAM word caused by a soft error, where the **fault model** is $0 \rightarrow \{1,*\}$, $1 \rightarrow \{0,*\}$, or $* \rightarrow \{0,1\}$. The **probability of bit-flip errors** is $P_e = u/w$, where u is the number of bit-flip words and w is the number of words in a TCAM.

Note that, in this work, the soft errors are assumed to occur randomly with the probability $P_e = u/w$ in a TCAM word, or wP_e words in a TCAM as a whole. Moreover, the errors, *i.e.*, the changes of the stored values are assumed at the beginning of a test.

Partial Don't-Care Keys (X-keys)

Firstly, we define partial don't-care keys (X-keys). The X-keys are derived from the main search key that enters the TCAM for look-up operations.

Definition 4: A **partial don't-care key (X-key)** is a search key with inserted don't-care bits. The key is used to detect a soft error in TCAM words [10].

We can set the number of don't-cares inserted in an X-key according to the estimated amount of soft errors. The ability to detect soft errors increases with the number of inserted don't-cares. In this work, we assume that the number of segmentations (s) is maximum, *i.e.*, the value of $s = n$, where n is the number of bits in a TCAM word and the number of don't-care groups covers l bits.

Lemma 1: The different number of X-keys in a look-up table for multiple-bit-flip errors is:

$$p = \binom{s}{l} = \frac{s!}{l!\,(s-l)!}$$

where s is the number of fields (segmentations) in a TCAM word and l is the number of don't-care groups in the X-keys. In this case, a don't-care group refers to a particular field whose all bits are set as don't-care bits [10].

Overview of the Proposed Method

In this work, we use machine learning to analyze the matching distribution of partial don't-care keys. However, the setting of the filter is found by examining the distributions of matching probability in the subsequent sections. After the trained X-keys are generated, they enter the TCAM, and the matching indices are checked by the index checker. Finally, it performs short scrubbing intervals by 10% of number of rules by referring the ECC SRAM.

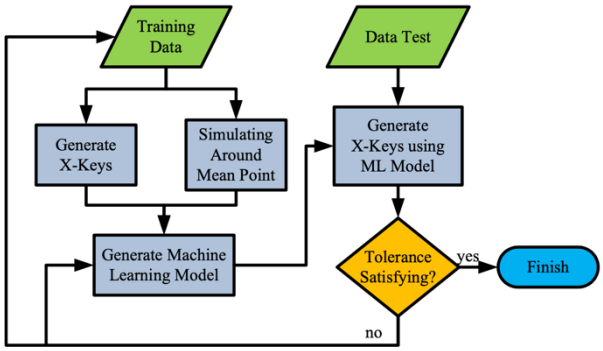

Figure 2 Machine Learning for Optimized Multiple-Bit-Flip-Tolerant TCAM

STATISTICAL PROPERTIES OF PARTIAL DON'T CARE KEYS (X-KEYS)

In this section, we briefly explain our proposed method by analyzing and deriving the statistical properties of X-keys. The mean and the variance of the distribution of filter matching probability help to choose the number of don't cares in the X-keys.

Mean and Variance

First, we define an event of an X-key with one inserted don't-care bit. From this, we find the distribution of filter matching probability.

Definition 5: Let λ_i be the probability of an X-key that is inserted with a don't-care bit at the index i will match as the same index as the search key in a TCAM. Let S_i be the random X-key with a don't-care bit at the index i for various search keys. We assume the event of different index as an independent event. Thus, we have $T(S_i) = \lambda_i$.

The mean [13], [14] of the probability of an X-keys that is inserted with a don't-care bit at the index i will match as the same index as the search key in a TCAM is derived as follows:

First, the variance is derived as follows:

Lemma 2: Let $T(S_i)$ be a matching probability of an X-keys S_i. Let μ be the mean of the probabilities $T(S_1), T(S_2), \cdots, T(S_n)$. Then, the mean of X-keys matches the same index in TCAM as the original search key is given by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} T(S_i).$$

Second, the variance is derived as follows:

Lemma 3: Let $T(S_i)$ be the matching probability of an X-key S_i. Let μ be the mean of the probabilities $T(S_1), T(S_2), \cdots, T(S_n)$. Then, the variance is given by

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (T(S_i) - \mu)^2.$$

Match Probability

Since it is assumed that the event of matching of an X-key is independent, we have the probability of X-keys with i number of don't-care bits as follows:

Theorem 1: The probability of X-keys matching is

$$P = \sum_{i=1}^{l} \lambda_i,$$

where l is the number of don't-care bits in an X-key.

Proof. By Definition 5, the independent event probability is assumed in the matching X-key for a don't-care bit at i index. Then, we have the probability of i number of don't-care bits by summing the smaller events. Thus, we have the theorem.

The generation of arrays with random X-keys is required to detect the soft errors by covering the errors.

Definition 6: Let g and h be logic functions. Then, g **covers** h if and only if $g \cdot h = h$ [10].

As in [10], [11], and [12], we have the following theorem:

Theorem 2: Soft errors can be detected by X-keys that cover all bits of the search key. If all the X-keys match the same word and the word is already refreshed, then the matched word is correct.

Proof. The correct match is represented by the intersection of all matched X-keys:

$$f = \bigvee_{i=0}^{p-1} g_i,$$

where p is the number of X-keys, f is the correct match and g_i is an X-key with relations $f \subset g_i$ such that $g_i \not\subset g_j$, $i \neq j$. If all of the X-keys match f, then f is proven to be the correct match, otherwise, some error(s) may have occurred.

Finding the Setting for the Filter using Statistical Properties

In this subsection, we find statistical properties of the filter by using random X-keys. Fig. 3 shows the flowchart of evaluation of a filter. First, we analyze the distribution by the X-keys. Then, from the distribution, we determine the number of X-keys and don't-care bits (l). Then, we generate the random X-keys by the determined number of don't cares. Finally, if the tolerance does not meet the requirement, we increase the number of X-keys or change the number of don't cares.

Algorithm 1 shows the way to find the filter distribution. There is $NSKeys$ search keys and we add the number of don't care bits form 1 to $s-1$ bits. Next, it generates a random X-key (Line 7) by using $RandXKeys$. If $xKey$ matches the same index as $searchKey$, we increment the correct counter. Finally, the match probability for s bits filter is P.

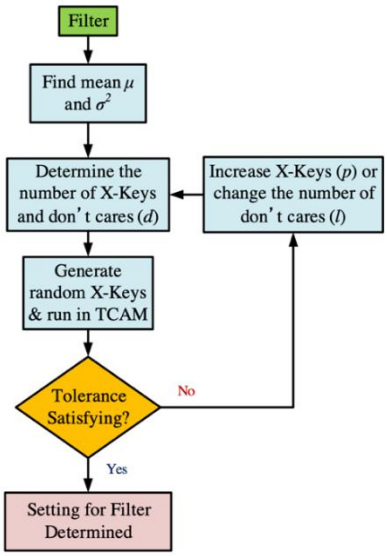

Figure 3 Evaluating Filter

Algorithm 1 Filter Distribution ($FilterDist()$)

```
\*Input: searchKey and xKey, where searchKey is the input to match a TCAM word, and xKey is the partial don't-care keys generated from searchKey. Output: Match probability arrays P.*/
01: Number of search keys is NSKeys
02: Number of bits is s
03: Number of correct match is NCorr ← 0
04: Match probability arrays is P
05: for (i = 0, i < s, i++) do
06: for (j = 0, j < NSKeys, j++) do
07: Generate a random X-key;
        xKey ← RandXKeys(searchKey, NBit, i)
08: if (TCAM(xKey) == TCAM(searchKey)) then
09: Increment the number of correct match;
            NCorr ← NCorr + 1
10: else
11: NCorr ← NCorr
12: endif
13: endfor
14: P[i] ← NCorr/NSKeys
15: endfor
16: Return P
```

RANDOM X-KEYS FOR BIT-FLIP ERRORS DETECTION USING MACHINE LEARNING

This section first describes previous methods in [10], [11], and [12]. It consists of random generation of X-keys, a TCAM, and

https://doi.org/10.25077/jnte.v11n1.1007.2022

an ECC-SRAM for handling the refresh the operation. In this paper, we improve the method by using statistical properties to determine the suitable random X-keys setting.

Random Generation of X-Keys

Algorithm 2 shows the procedure for obtaining a random X-key. If the number of don't-care bits is larger than a half of the total bits, we just randomize the care bits. Inside $RandXKey$, the arrays are checked whether the bit has already been assigned (value 1) or not (value 0). When it randomizes the index, it puts the don't-care or the search key bit in the X-key. Finally, it returns the X-key. The collection of the random arrays of X-keys is stored in the $xKey$. It iterates $RanXKey()$ function for $numKey$ times, where $numKey$ is the desired number of X-keys.

Algorithm 2 Arrays of Random X-keys($ArrayRandXKeys$)

```
/*Input: searchKey, s, l and numKey, where searchKey is the input to match a TCAM word, xKey is X-keys generated from searchKey, s is the number of bits, and l is the number of don't-care bits, as well as numKey is the number of X-keys. Output: Random x-keys xKeys.*/
01: The arrays of X-keys will be stored in xKeys.
02: for (i = 0; i < numKey; i++) do
03: xKeys[i] ← RandXKey(searchKey, s, l, numKey)
04: endfor
05: Return xKeys
```

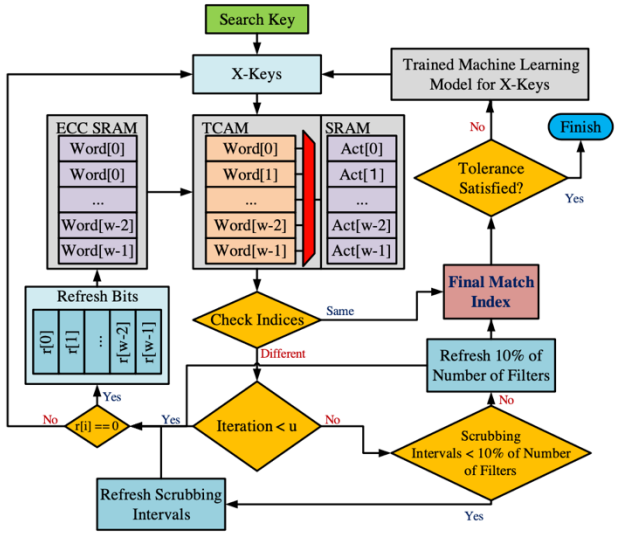

Figure 4 Proposed MLTCAM using Random X-Keys

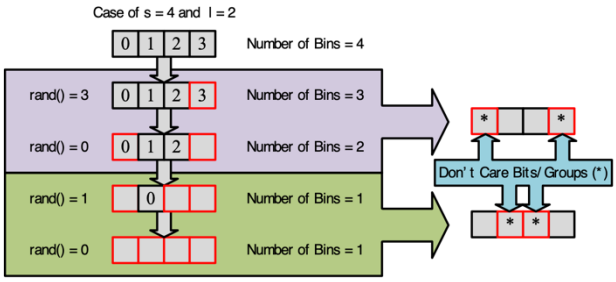

Figure 5 Perfect Match using Random Index Example

Algorithm 3 and Fig. 4 show the proposed ML-TCAM scheme. First, we train the X-keys using statistical properties and enter the X-keys to the TCAM memory. If the indices of the matching X-keys are the same, thus perfect match is reached, otherwise we detect the bit flip errors from the different indices (Line 12). Next, some filters are refreshed by using the ECC SRAM to recover the errors (Line 18). Fig. 5 shows the steps for covering whole bits by random X-keys. We reduce the number of random ranges when a bit has already been assigned to a don't care bit.

Algorithm 3 ML-TCAM($SearchKeys()$)

```
/*Input: searchKeys and TCAM, where searchKeys is the search keys and the TCAM word, and l is the number of don't-care groups in the X-keys. Output: Corrected TCAM.*/
01: finish ← 0
02: while (finish ≠ 1) do
03: P ← 0
04: while !(P ~ 0.5) do
05: Generate X-Keys;
    xKeys ← ArrayRandXKeys(searchKey, s, l, numKey)
06: P ← FilterDist(xKeys)
07: endwhile
08: idx ← TCAM(xKeys[0])
09: numErr ← 0
10: while (finish != 1) do
11: for (i = 1; i < numX; i++) do
12: if (idx ≠ TCAM(xKeys[i])) then
13: Soft Error Detected; numErr ← numErr + 1
14: endif
15: endfor
16: if (numErr == 0) then
17: Tolerance satisfied; finish ← 0
18: else
19: Refresh TCAM with scrubbing intervals 10%
20: endif
21: endwhile
22: endwhile
23: Return TCAM
```

EXPERIMENTAL RESULTS

We evaluated the proposed method by packet classification benchmarks generated by ClassBench [15]. Then, we used these benchmarks and evaluated by the computer program in OSX with i7 Intel machine and 8 GB memory.

Table 1 shows the number of search keys as well as the number of rules of ACL1-5, FW1-5, and IPC1-2 filters. We derived match probability of the filters for different numbers don't-care bits from 1 to 103. In this case, the number of bits in the TCAM is 104 which consists of source address (32 bits), destination address (32 bits), source port (16 bits), destination port (16 bits), and protocol type (8 bits). First, we compare our proposed method with KX-TCAM [12] for $l = \mu$. In [12], all-possible perfect match X-keys are generated by the permutation of s and l, where s is the number of bits and l is the number of don't-cares in the X-keys. As shown in Table 1, the proposed perfect match using random X-keys is faster than the perfect match X-keys [12] up to 7.65 times.

Next, we evaluated the filter distributions with Algorithm 1. To obtain more accurate and smooth distributions, we iterated the samples 100 times. Fig. 6 shows the match probability of the

filters. We approximated the distribution by the cumulative distribution of normal function by calculating the mean μ.

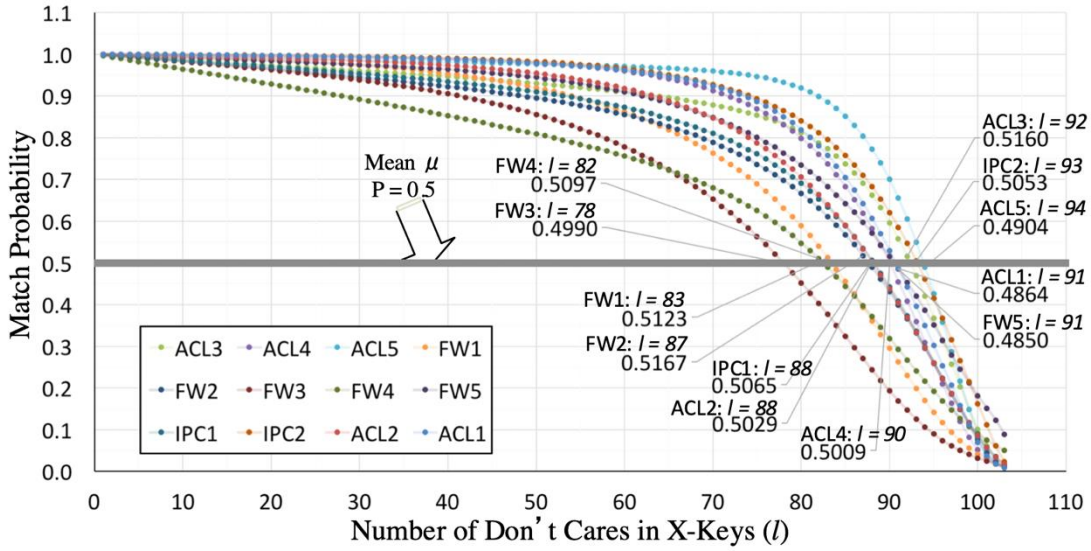

Figure 6 Match Probability of Filters

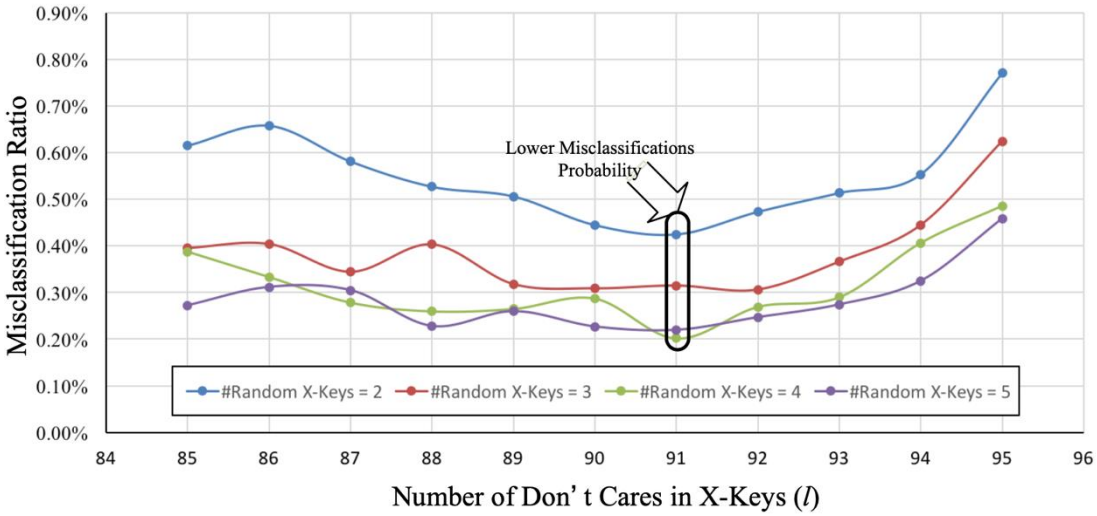

Figure 7 Comparison of performance (misclassification) for various number of X-keys in ACL5 filter

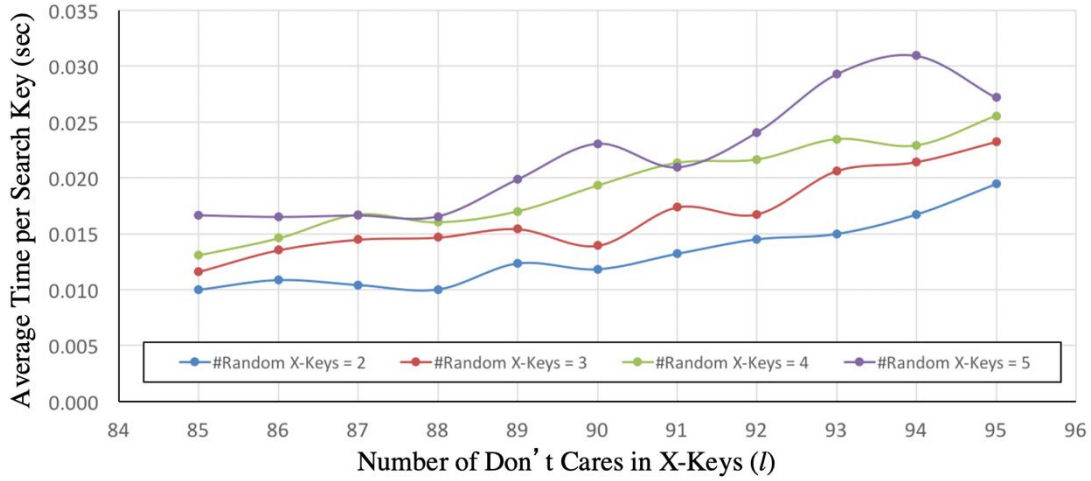

Figure 8 Comparison of performance (time) for various number of X-keys in ACL5 filter

https://doi.org/10.25077/jnte.v11n1.1007.2022

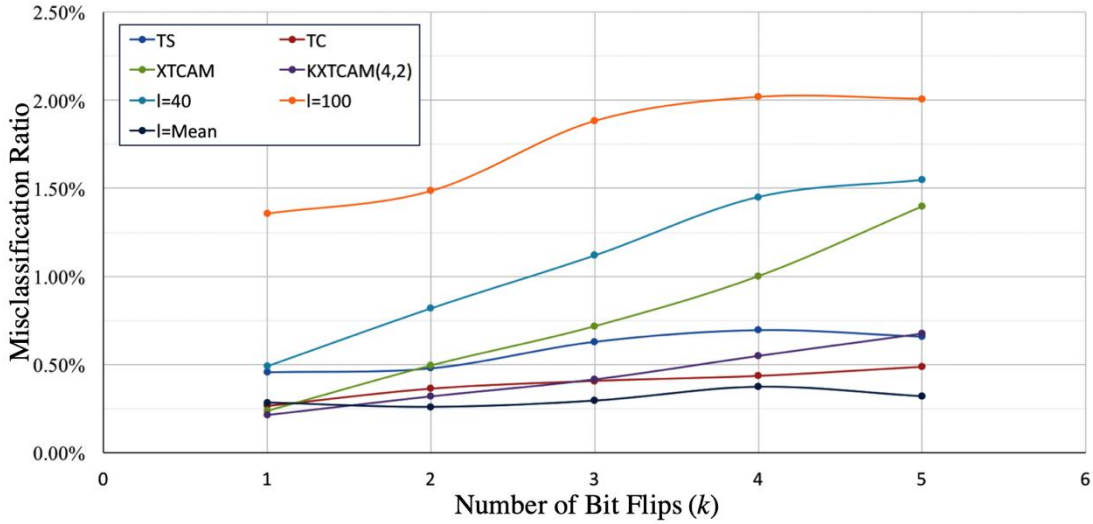

Figure 9 Comparison with other methods

Table 1. The Number of Rules and Search Keys for Various Filters

Type	**ACL1**	**ACL2**	**ACL3**	**ACL4**
# Rules	100	100	100	100
# Search Keys	10002	10003	10005	10057
μ Values	91	88	92	90
KX-TCAM (sec) [12]	50.65	44.28	26.13	36.36
ML-TCAM (sec)	18.40 (2.75×)	32.36 (1.37×)	4.92 (5.31×)	16.09 (2.26×)
Type	**ACL5**	**FW1**	**FW2**	**FW3**
# Rules	100	99	98	98
# Search Keys	10001	10136	10415	10370
μ Values	94	83	87	78
Distribution (sec) [12]	20.15	83.50	30.20	88.72
ML-TCAM (sec)	4.54 (4.44×)	19.48 (4.29×)	31.91 (0.95×)	19.24 (4.61×)
Type	**FW4**	**FW5**	**IPC1**	**IPC2**
# Rules	98	99	100	92
# Search Keys	10507	10194	10002	10453
μ Values	82	91	88	93
KX-TCAM (sec) [12]	139.41	32.36	34.91	21.84
MLTCAM (sec)	18.23 (7.65×)	21.87 (1.48×)	16.66 (2.10×)	13.96 (1.56×)

Figure 6 shows the characteristics of the filters. Thus, we can determine a suitable number of don't care keys for the X-keys. The statistical result is very useful for finding a right setting for each filter, so that the X-keys detect the soft errors at desired tolerance of soft errors. The resume of the mean μ values are shown in the bottom row of Table 1. The match probability is given by:

$$\gamma = \frac{\#\ matched\ indices\ of\ X-Keys\ equal\ to\ search\ key}{\#\ search\ keys\ of\ the\ filter}$$

The ratio of misclassifications is given by:

$$\sigma = \frac{\#\ misclassifications\ at\ each\ filter}{\#\ search\ keys\ of\ the\ filter}$$

Furthermore, we evaluated the ACL5 filter in particular. This is because the misclassification ratios of ACL5 is higher than others. As shown in Fig. 6, the ACL5 drops at the largest number of don't-care bits. It is also indicated by the largest μ value which is 94. In this experiment, we assigned the number of X-keys as 2, 3, 4, and 5. Then, we ran Algorithm 3 when the numbers of don't-care bits l were 85 to 95. As shown in Fig. 7, the lowest misclassification ratio was obtained when the number of don't-care bits was 91. It also indicates that more random X-keys will increase the tolerance of the TCAM. However, for the case where the number of X-keys is 4, the tolerance is similar to the case where the number of X-keys is 5. Fig. 8 shows the time performance.

Finally, we compare our proposed method with other existing methods. In this experiment, we used the filters in Table 1 and compared the performance of other methods: TCAM Scrubbing (TS) [16], TCAM Checker (TC) [17], XTCAM [9], KX-TCAM(4,2) [10], [11], [12], and random X-Keys whose number of don't-care bits $l = 40$ and $l = 100$, as well as the proposed random X-Keys whose number of don't-care bits Mean (μ). The value of the misclassification ratio shows the average of all filters (ACL, FW, IPC). As shown in Fig. 9, random X-Keys with $l = \mu$ performs betters in higher bit flips ($1 < k$) up to more than 2 times compared to TCAM Scrubbing (TS) and TCAM Checker (TC), and 6 times better than other settings of number of don't cares for the random X-Keys. This indicates that by knowing the characteristic of the filters from their statistical data, we can improve the soft-error tolerance of the TCAM.

CONCLUSIONS

This paper has proposed a soft-error-tolerant TCAM for multiple-bit-flip errors using randomly generated partial don't-care keys (X-keys). First, we obtained the statistical data of X-keys that match the same locations as the search key. After that, we assumed the distribution of filters to estimate the number of don't-

care bits in X-keys. Finally, the X-key were implemented in TCAM to correct bit-flip errors. Moreover, the suitable number of don't-care bits in an X-keys was determined from the distribution of X-keys so that the appropriate degree of tolerance of the TCAM against soft errors was found. Match probabilities for various filters were shown. Experimental results demonstrated that the soft-error tolerance using statistical data has better soft-error tolerance than other methods up to more than 6 times and it is faster up to 7.65 times compared to other methods. Moreover, our method simplifies the generation of X-keys and helps to find the proper number of don't-care bits in the X-keys. The proposed method is useful for soft-error tolerant TCAMs in routers for robust communications. In the future, researches on the implementations of soft-error tolerant systems on embedded devices such as FPGAs, GPUs, or even specific ASICs are open to be explored.

ACKNOWLEDGMENT

The works is supported by P2MI ITB Program of Electronics Research Group, STEI, ITB. Infall Syafalni also would like to thank Prof. Tsutomu Sasao and Prof. Xiaoqing Wen for the guidance in the research.

REFERENCES

[1] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE JSSC*, vol. 41, No. 3, pp. 712-727, March 2006.

[2] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computing Surveys*, vol. 37, no. 3, pp. 238-275, Sept. 2005.

[3] K. Pagiamtzis, N. Azizi, and F. N. Najm, "A soft-error tolerant content addressable memory (CAM) using an error-correcting-match scheme," *IEEE CICC*, pp. 301-304, Sept. 2006.

[4] C. Slayman, "Soft errors -- Past history and recent discoveries," *International Integrated Reliability Workshop Final Report*, pp. 25-30, Oct. 2010.

[5] A. Dixit and A. Wood, "The impact of new technology on soft error rates," *IEEE International Reliability Physics Symposium*, pp. 5B.4.1-5B.4.7, April 2011.

[6] P. Reviriego, S. Pontarelli and A. Ullah, "Error Detection and Correction in SRAM Emulated TCAMs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 2, pp. 486-490, Feb. 2019.

[7] S. Greenberg, T. Sheps, D. A. Leon and Y. Ben-Shimol, "Packet Classification Using GPU and One-Level Entropy-Based Hashing," *IEEE Access*, vol. 8, pp. 80610-80623, 2020.

[8] I. Syafalni, T. Sasao, X. Wen, S. Holst, and K. Miyase, "Soft-error tolerant TCAMs for high-reliability packet classification," *IEEE APCCAS*, pp. 471-474, Nov. 2014.

[9] I. Syafalni, T. Sasao, X. Wen, S. Holst, and K. Miyase, "A soft-error tolerant TCAM using partial don't-care keys," *IEEE ETS*, pp. 1-2, May 2015.

[10] I. Syafalni, T. Sasao, and X. Wen, "Multiple-bit-flip detection scheme for a soft-error resilient TCAM," *IEEE ISVLSI*, pp. 679-684, July 2016.

[11] I. Syafalni, T. Sasao, and X. Wen, "A soft-error tolerant TCAM for multiple-bit flips using partial don't care keys," *IWLS-2015*, Mountain View, CA, pp. 11-18, June 2015.

[12] I. Syafalni, T. Sasao and X. Wen, "A Method to Detect Bit Flips in a Soft-Error Resilient TCAM," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1185-1196, June 2018.

[13] C. L. Liu, *Introduction to Combinatorial Mathematics*, McGraw-Hill, 1968.

[14] C. M. Grinstead, W. P. Peterson, and J. L. Snell, *Probability Tales*, American Mathematical Society, 2011.

[15] D. E. Taylor and J. S. Turner, "ClassBench: a packet classification benchmark," *IEEE/ACM TON*, vol. 3, no. 15, pp. 499-511, June 2007.

[16] S. Baeg, S. Wen, and R. Wong, "Minimizing soft errors in TCAM devices: A probabilistic approach to determining scrubbing intervals," *IEEE TCAS*, vol. 57, no. 4, pp. 814-822, April 2010.

[17] M. Z. Shafiq, C. Meiners, Z. Qin, K. Shen, and A. X. Liu, "TCAMChecker: A software approach to the error detection and correlation of TCAM-based networking system," *Journal of Network and System Management*, vol. 21, no. 3, pp. 335-352, Sept. 2013.

AUTHOR(S) BIOGRAPHY

Infall Syafalni
received the B.Eng. degree in electrical engineering from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 2008, the M.Sc. degree in electronic engineering from the University of Science Malaysia (USM), Penang, Malaysia, in 2011, and the Dr. (Eng.) degree in engineering from the Kyushu Institute of Technology (KIT), Iizuka, Fukuoka, Japan, in 2014. From 2014 to 2015, he has held a research position with KIT. From 2015 to 2018, he has held an ASIC Engineer with the ASIC Development Group, Logic Research Company Ltd., Fukuoka. In 2019, he joined the ITB, where he is currently an Assistant Professor with the School of Electrical Engineering and Informatics and a Researcher with the University Center of Excellence on Microelectronics, ITB. His current research interests include logic synthesis, logic design, VLSI design, and efficient circuits and algorithms.

Trio Adiono
received the B.Eng. degree in electrical engineering and the M.Eng. degree in microelectronics from the Institut Teknologi Bandung, Indonesia, in 1994 and 1996, respectively, and the Ph.D. degree in VLSI design from the Tokyo Institute of Technology, Japan, in 2002. He holds a Japanese Patent on a high-quality video compression system. He is currently a Professor with the School of Electrical Engineering and Informatics, and also works as the Head of the IC Design Laboratory, Microelectronics Center, Institut Teknologi Bandung. His research interests include VLSI design, signal and image processing, VLC, smart cards, and electronics solution design and integration.

 https://doi.org/10.25077/jnte.v11n1.1007.2022