Multiple-Bit-Flip Detection Scheme for A Soft-Error Resilient TCAM

Infall Syafalni[†], Tsutomu Sasao[‡], and Xiaoqing Wen[§]
[†]Logic Research Co., Ltd., Japan, [‡]Meiji University, Japan, [§]Kyushu Institute of Technology, Japan
[†]infall@logic-research.co.jp, [‡]sasao@cs.meiji.ac.jp, [§]wen@cse.kyutech.ac.jp

***Abstract*—Ternary content addressable memories (TCAMs) are special memories which are widely used in high-speed network applications such as routers, firewalls, and network address translators. In high-reliability network applications such as aerospace and defense systems, soft-error tolerant TCAMs are indispensable to prevent data corruption or faults caused by radiation. This paper proposes a novel soft-error tolerant TCAM for multiple-bit-flip errors using partial don't-care keys (X-keys), called k-TX. k-TX corrects up to k-bit flip errors and significantly enhances the tolerance of the TCAM against soft errors, where k is the maximum number of bit flips in a word of a TCAM. k-TX consists of a TCAM, a preprocessed don't-care-bit index look-up memory (X look-up), and an ECC-SRAM. First, k-TX randomly selects a search key. After that, k-TX detects multiple-bit-flip errors by the generated X-keys using the X look-up. If the keys match the different locations, then a soft error is suspected and k-TX refreshes the TCAM words by using a backup ECC-SRAM. Experimental results show that the soft-error tolerance capability of k-TX outperforms other schemes significantly. Moreover, the hardware overhead of k-TX is small due to the use of only a single TCAM. k-TX can be easily implemented and is useful for fault-tolerant packet classifiers.**

I. Introduction

A ternary content addressable memory (TCAM) is a special memory with three values, *i.e.,* 0, 1, and * (*don't-care*). It simultaneously compares the input vector with the entire list of registered vectors [7]. TCAM is a de facto standard in routers and devices for packet classification in high-speed network applications [2]. Fig. 1 shows a cell of a TCAM. The search bits (SL_1, SL_0) are compared with the stored bits (D_1, D_0). When there is a match, the match line (ML) sends a signal to the priority encoder to produce the match address.

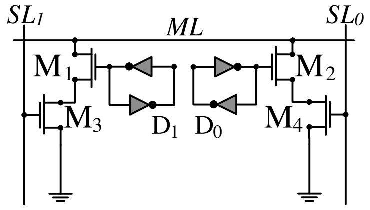


Fig. 1. NOR-type ternary cell

Generally, soft errors are caused by ionizing particles, like alpha particles, protons, heavy ions, neutrons, etc; in some cases, this ionizing particles are generated by radioactive atoms [3]. Several works investigated the effects of scaling down the size of transistors [4], [5]. A work in [5] shows that moving from the 130 nm process to the 22 nm process increases the soft error rates up to 7 times. Furthermore, soft errors tend to cause more serious problems in low power devices [3].

One of the troubling effects of soft errors in memories is that they hit memory cells and may change the values of some cells. Value changes of TCAM cells may lead to errors or data corruption. A soft error does not damage hardware; it only changes the data that is being processed, and it can produce faulty data [6]. TCAMs are more vulnerable to soft errors than SRAMs since TCAMs are more complicated than SRAMs. The bit storage of TCAMs uses SRAM cells which are susceptible to soft errors [7]. Furthermore, high-speed memories with smaller transistor sizes are more vulnerable to soft errors [8]. Thus, in applications that require high-reliability such as finance, aerospace, and defense networks, soft-error tolerant TCAMs are indispensable. Unfortunately, conventional ECC (Error Checking and Correction) techniques used for SRAM is difficult to apply for TCAM [9].

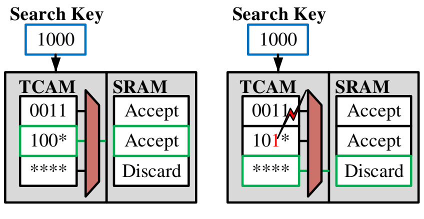
In previous works, hardware and software methods were developed to mitigate soft errors [6], [10]–[13]. In [11], hardware modification with XOR-based conditional keepers are used to overcome noises including soft errors. In [12], a system using bloom filters detects errors in TCAMs. Furthermore, in [6], a parallel system using two TCAMs detects and corrects TCAM words that are attacked by soft errors. However, previous hardware and software methods still suffer from severe drawbacks. Firstly, hardware methods are very costly to implement since they modify the circuits of TCAMs. Secondly, for software methods, researchers are still looking for more efficient way to tackle the soft-error problem in TCAMs. In [10], a TCAM with the optimized scrubbing interval is proposed against soft errors. However, this scheme ignore the fact that some keys are more frequently used than others in the TCAM which can lead to more faults when the frequent keys hit the upset word caused by soft error. In [6], a TCAM checker is proposed that takes into consideration frequent keys through comparing the matched words caused by a soft error. However, this scheme uses two TCAMs, which doubles the hardware overhead and the power dissipation. In [13], the similarity of two TCAMs is checked for detecting soft errors.

Table I shows the TCAM encoding. The stored bits are represented by D_1 and D_0. Note that the values of D_1 and D_0 are not necessarily complementary. The don't-care value (*) is represented by $(D_1, D_0) = (1, 1)$. While the stored bits represent the value of the TCAM cell, the search lines represent a search key bit. In the search lines, the don't-care value (*) is represented by $(SL_1, SL_0) = (0, 0)$. If we refer to the transistor-level representation of the TCAM cell in Fig. 1, then this condition allows transistors M_3 and M_4 to be off, forcing a match of the bits regardless of the stored bits D_1 and D_0. This observation inspires us to propose our method in this paper.

TABLE I
TCAM ENCODING

Value	D_1	D_0	SL_1	SL_0
0	0	1	0	1
1	1	0	1	0
*	1	1	0	0

Fig. 2 shows an example of a soft error. A **false match** is a match that would be a mismatch if no soft error occurred. In term of packet classification, a false match can produce **misclassification**. Fig. 2(b) shows a misclassification which matches the third word of the TCAM, but should match the second word of the TCAM (Fig. 2(a)). The soft error occurs at the third bit of the second word. If we change the search key to 10*0, then the TCAM will match the correct word (*i.e.,* second word). We call this search key as a partial don't-care key (X-key) which will be explained later.



(a) Correct match (b) False match
Fig. 2. A soft error in a TCAM

The **major contributions** of this paper are as follows: 1) A novel scheme, called k-TX, is proposed that can detect and correct multiple-bit-flip soft errors in a TCAM. 2) The k-TX requires no modifications to the TCAM. 3) The k-TX uses only one TCAM. 4) The soft-error tolerance of k-TX outperforms existing schemes. The rest of the paper is organized as follows: Section II defines the basic properties; Section III describes the proposed scheme, k-TX for multiple-bit-flip errors. Section IV shows experimental results; and Section V concludes the paper.

II. Definitions and Basic Properties

A TCAM consists of words. And a word of a TCAM that represents a packet classifier rule consists of several fields.

A. Classification Functions

A classification function is defined as a mapping of fields specified by a set of rules.

Definition 2.1. *A **classification function** with k **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.*

In the field of Internet, a packet classifier is specified by source and destination addresses, source and destination ports, and the protocol type. The source and the destination ports are represented by intervals. Some works on the representation of classification functions can be found in [14], [15]

B. Soft Error in a TCAM Cell

Definition 2.2. *A **soft error** in a TCAM cell is a non-permanent error that changes cell values and may cause misclassification.*

Definition 2.3. *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 $P_e = \frac{u}{w}$ in a TCAM word or wP_e 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.

III. k-TX: A TX For Multiple-Bit-Flip Errors

This section describes our proposed method called k-TX, where k is the maximum number of soft errors in a word of a TCAM, T stands for TCAM and X stands for partial don't-care keys (X-keys).

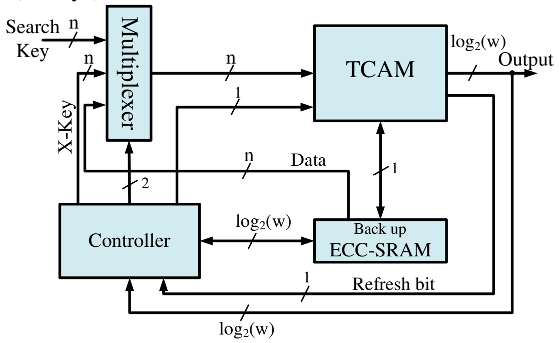


Fig. 3. k-TX: A soft-error tolerant TCAM using partial X-keys

A. Properties the k-TX Scheme

k-TX consists of a preprocessing X-key look-up table (included in the controller), a TCAM, and an ECC-SRAM for handling the refresh the operation as shown in Fig. 3. In the k-TX, no modification to the TCAM is required. The k-TX has two modes: **normal mode** and **test mode**. In the normal mode, the TCAM looks up a search key in parallel. In the test mode, the TX generates partial don't-care keys (X-keys) and detects soft errors. It works sequentially as follows: First, the k-TX starts by applying several X-keys to the TCAM. Don't-care bits are inserted in the X-keys produced using X look-up memory, and they are expected to match the words that contain soft errors. After applying the X-keys to the TCAM, k-TX records all the returned indices. After that, if the indices are equal, then no soft error is detected; otherwise a soft error is detected. Fig. 4 shows the flowchart of the k-TX.

1) 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 operation.

Definition 3.1. *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.*

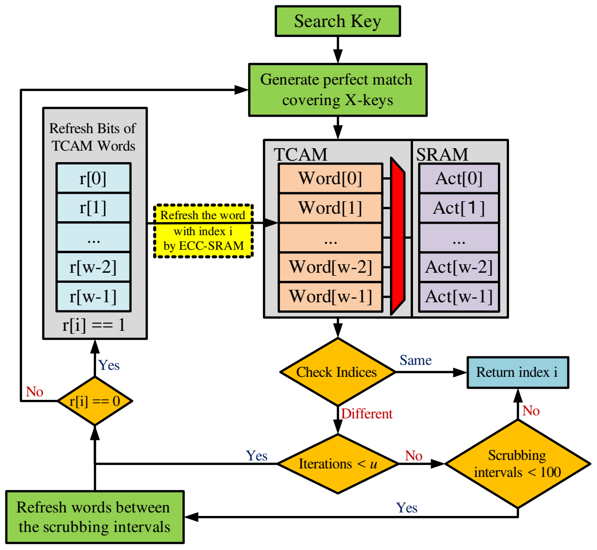


Fig. 4. Generation of partial don't-care keys for k-TX

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. However, if the number of don't-care bits is greater than the number of care bits, then the capability of soft-error detection will decrease.

2) Referesh Bits of TCAM Words: Refresh bits are used to denote refresh TCAM words. Thus, we can efficiently control the refresh time and power using these bits.

Definition 3.2. *A **refresh bit** of a TCAM word has a value 1, when the TCAM word has been refreshed by the ECC-SRAM. Otherwise, it has a value 0. Every word in the TCAM has a refresh bit to indicate the status of refreshing of the word. The total number of refresh bits of TCAM words is w.*

3) Detection of Soft Errors Using Partial Don't-Care Keys: Now, we describe how soft errors can be detected using partial don't-care keys.

Definition 3.3. *Let g and h be logic functions. Then, g **covers** h if and only if $g \cdot h = h$.*

Theorem 3.1. *Soft errors can be detected by X-keys that covers all bits of the search key. If all the X-keys match the same word, then the matched word is correct.*

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

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

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

Example 3.1. *This example illustrates the covering of X-keys to a correct match $x_4x_3\bar{x}_2x_1x_0$ for n (the number of bits) is 5 and q (the number of don't-care bits inserted in the X-keys) is 3. By Lemma 3.2, the maximum number of X-keys is $p = \binom{s}{l} = 10$. Thus, the correct match is $\bigwedge_{i=0}^{9} g_i = x_4x_3 \wedge x_4\bar{x}_2 \wedge x_4x_1 \wedge x_4x_0 \wedge x_3\bar{x}_2 \wedge x_3x_1 \wedge x_3x_0 \wedge \bar{x}_2x_1 \wedge \bar{x}_2x_0 \wedge x_1x_0 = x_4x_3\bar{x}_2x_1x_0$.* ■

Algorithm 1 Perfect Match Generator ($PerfectXKeyGen()$)

```
/* Input: Search Key (searchKey), $n$, $s$, and $l$, where $n$ is the
   number of bits, $s$ is the number of fields in a TCAM word,
   and $l$ is the number of don't-care groups in the X-keys. */
1:  Generate index combinations with the number of X-Keys $p = \binom{s}{l}$.
2:  if $s \mod l \neq 0$, $g = s$, else $g = 1$.
3:  Find perfect matches with the number of perfect covering
    of X-keys being $u = \frac{p}{g \times l}$
4:  while $c \neq u$ do
5:      while $CoveringVector[idx]! = g$ do
6:          Invoke the combinations of indices and find the perfect
            match. Every combination of indices can be used only
            once in order to compact the $XLookUp$ memory.
7:          if $Contradict$ then
8:              Backtrack by subtracting the previous addition op-
                eration in $CoveringVector$.
9:          else
10:             Write the perfect match pairs in the $XLookUp$
                memory.
11:         end if
12:     end while
13: end while
14: Return $XLookUp$
```

B. Algorithm and Time Complexity of k-TX

A merit of a TCAM is the matching search keys simultaneously. However, in the test mode, the TX uses a sequential TCAM look-up. Since the operation time of the TX is significantly influenced by the TCAM look-up time, we briefly describe it.

Lemma 3.1. *The time for the sequential TCAM is $\mathcal{O}(wn)$, where n is the number of bits in a word and w is the number of words in the TCAM.*

Proof: Checking the TCAM sequentially requires n steps for each word and there are w words. □

1) Generating X-Keys for Look-Up Table: Detection of multiple-bit-flip errors (k-flip) requires more than k fields in a word of the TCAM.

Lemma 3.2. *The maximum 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 (segmentation) in a TCAM word and l is the number of don't-care groups in the X-keys.

Proof: Consider s baskets and l balls. The number of combinations of having l balls in the baskets is $p = \binom{s}{l}$. □

Algorithm 2 Multiple-Bit-Flip Detection and Correction Using X-Keys

```
/* Input: A TCAM with w words and n bits each word, a search key ($searchKey$), an ECC-SRAM as a back-up of the TCAM for refresh operation, $s$ (the number of partition), and $l$ (the number of groups of don't-care). */
1: Determine the prediction of the maximum number of multiple-bit-flip in a word of TCAM.
2: Preprocess $XLookUp \Leftarrow PerfectXKeyGen(n, s, l)$.
3: $Prob \Leftarrow rand()$
4: if $Pc \geq Prob$ then
5:    $r \Leftarrow 0$
6:    $difIdx \Leftarrow 1$
7:    $iterations \Leftarrow 0$
8:    while $(difIdx) \wedge (iterations < u)$ do
9:       $XKeys \Leftarrow XLookUp(searchKey, iterations + 1)$
10:      for $i = (0, \cdots, numKey - 1)$ do
11:         $Idx[i] \Leftarrow TCAM(XKeys[i])$
12:         $lowIdx \Leftarrow \min_{\forall i} Idx$
13:         $highIdx \Leftarrow \max_{\forall i} Idx$
14:      end for
15:      $subIdx \Leftarrow \min_{\forall i}(highIdx - lowIdx)$
16:      $difIdx \Leftarrow CheckIdx(Idx, numKey)$
17:      if $difIdx$ then
18:         for $i = (0, \cdots, numKey - 1)$ do
19:            if $r[Idx[i]] == 0$ then
20:               Refresh the TCAM word with index $Idx[i]$
21:               $r[Idx[i]] \Leftarrow 1$
22:               $Idx[i] \Leftarrow TCAM(XKeys[i])$
23:            end if
24:         end for
25:         $difIdx \Leftarrow CheckIdx(Idx, numKey)$
26:      end if
27:      $iterations + +$
28:   end while
29: end if
30: if $difIdx$ then
31:    If $subIdx$ is less than 100, refresh the scrubbing interval.
32: end if
33: Return $TCAM(searchKey)$
```

Definition 3.4. *Let the covering of X-Keys be f, where g_i is an X-key, p is the number of X-keys, and h is a TCAM word (product). The **perfect match** is described as $h = f$.*

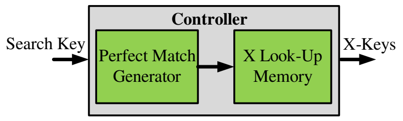


Fig. 5. Controller Illustration

Alg. 1 shows the method for generating perfect match indices which to be stored in the X look-up memory. The perfect match generator is a preprocess routine, and can be set based on the prediction of the maximum number of bit flips in a word in a TCAM. In this case, we use the array approach to find the perfect match. Fig. 5 shows the controller that contains the preprocessing unit: the perfect match generator and the X look-up memory.

2) Segmentation of TCAM Word:

Definition 3.5. *$(h_1, h_2, \ldots, h_s)$ is a **segmentation** of a TCAM word if and only if $h_1 h_2 \cdots h_s = h$, where s is the number of the parts, h_i is the function represented by the i-th part, and h is the function represented by the whole word.*

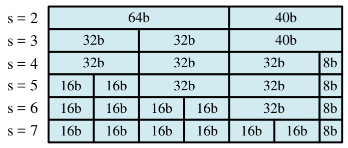


Fig. 6. Segmentation TCAM words

Consider the case of $s = 5$ fields classification function: 32-bit source and destination addresses, 16-bit source and destination ports, and 8-bit protocol, and the total number of bits in the TCAM is $n = 104$ bits. Fig. 6 shows the segmentation of the word.

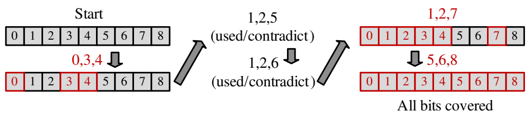


Fig. 7. Example of perfect matching in TCAM word

For example, if we have k-bit-flip errors in a word, it requires $s > l \geq k$ to detect all the errors, where s is the number of partitions and l is the number of don't-care groups in the X-keys. Let $k = 4$, and we choose $s = 7$ and $l = 4$. Thus, we have $p = 35$ X-keys to detect the errors. However, based on our observation, if the condition $l/s > 0.57$ holds (*e.g.,* $(s = 6, l = 4)$ or $(s = 7, l = 5)$), the X-keys cannot detect the soft errors effectively, since the numbers of don't cares in the X-keys are too large. In this case, the X-keys will likely match only the upper part of the TCAM. Fig. 7 shows the example of the perfect match covering in the TCAM. In this case, for the best detection of soft errors, the index of an X-key generated by combination can be only used once. Thus, when it contradicts, we have to reorder the X-keys.

3) Scrubbing Interval: To reduce the time overhead, we can only use small values for s and l. Thus, a short scrubbing interval is used in the k-TX for improving the tolerance of the k-TX against soft errors. First, the k-TX compares the lowest and the highest indices of matched X-keys. If the difference between the lowest matched index and the highest matched index is less than the previous stored scrubbing interval, the k-TX will store the scrubbing interval (Alg. 2, line 15). Finally, if the X-keys match different indices at the end of the routine, the TCAM will be refreshed based on the stored scrubbing interval.

Example 3.2. *Consider the TCAM in Fig. 8 and assume that the search key is 0111, where $n = 4$, $s = 4$ and $l = 2$. Perform detection and correction of a soft error using X-keys. First, the X look-up memory is filled by perfect match pairs of indices. The number of generated X-keys is $p = 6$. The X-keys are 01**, **11, 0*1*, *1*1, *11* and 0**1. The X-keys are applied to the TCAM sequentially and the TCAM returns the matched indices.*

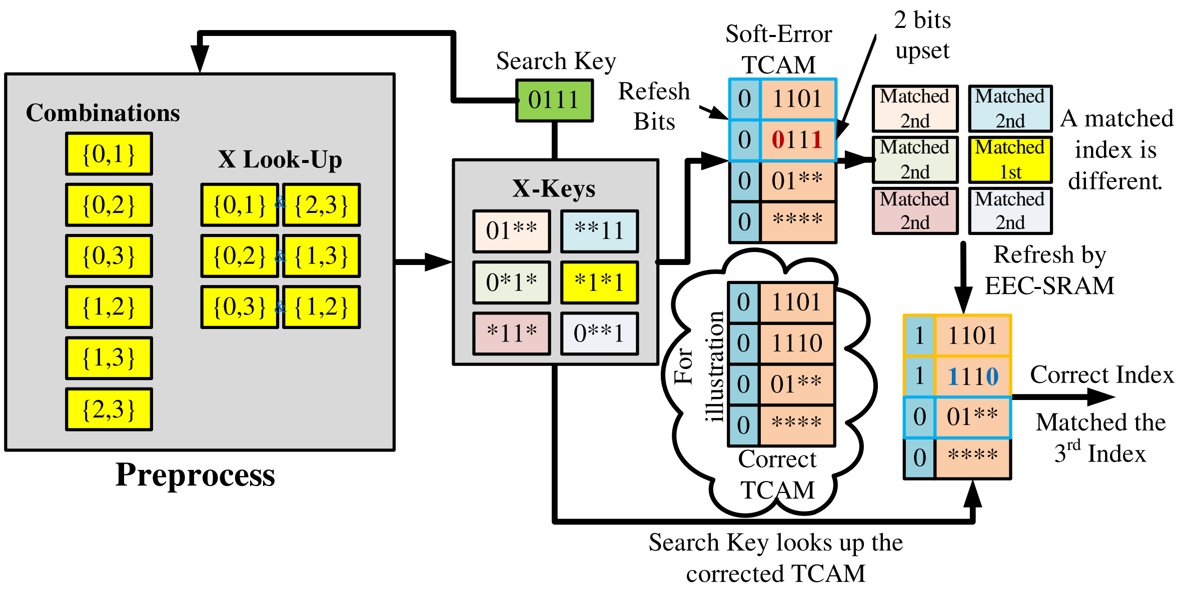


Fig. 8. Detection of Multi-Bit-Flip Errors ($k = 2$) using X-keys

The 1st, 2nd, 3rd, 5th and 6th X-keys match the 1st TCAM word and only the 4th X-key matches the 2nd TCAM word. After that, it checks the indices and returns 1 (line 15, Alg. 2), which means the indices have different values (a soft error is detected). Next, the refresh bits of TCAM words are checked with the corresponding indices (line 19, Alg. 2). If the refresh bit is 0, then a refresh operation for the corresponding TCAM word is performed using the ECC-SRAM. In this case, the 1st and 2nd TCAM words are refreshed. Finally, the search key is applied to the TCAM and matches the 3rd word. ■

When the TCAM has entries with smaller *Hamming* distance, the k-TX may suspect that there is a soft error in the TCAM by the X-keys. However, the k-TX will check the refresh bits and refresh all the unrefreshed TCAM words with the corresponding X-key matched indices. Thus, the TCAM words are updated, **regardless of the existence of any soft error**. Finally, the k-TX applies the original search key to the corrected TCAM (by Return TCAM(searchKey) procedure in Alg. 1, line 32). The complete procedure of the k-TX is described in Alg. 2.

Theorem 3.2. *The k-TX, which uses a sequential TCAM look-up, requires $\mathcal{O}(pwn)$ time, where p is the number of fields in a TCAM word, l is the number of don't-care groups in the X-keys, n is the number of bits in a word, and w is the number of words in the TCAM.*

Proof: The maximum number of iterations is p. Moreover, each X-key must be applied to the TCAM. If we use sequential TCAM look-up, then by Lemma 3.1, the total time complexity of the TX is $\mathcal{O}(pwn)$. □

IV. Experimental Results

For evaluation, we used packet classification benchmarks generated by ClassBench [16]. First, we generated rules and search keys for 5 types ACL filters, 5 types FW filters, and 1 type IPC filter by ClassBench. Then, we used these benchmarks and evaluated by the computer program in OSX with i7 Intel machine and 8 GB memory. In this case, we compared our proposed scheme with two representative schemes from recent works: the TCAM scrubbing (TS) [10] and the TCAM checker (TC) [6], as well as the TX for a single-bit flip [17] by implementing them in the same environment. Table II shows the numbers of rules and the numbers of search keys.

TABLE II
ACL filters and their search keys

Type	ACL (5 types)	FW (5 types)	IPC (1 type)
# Rules	4739	4356	674
# Search Keys	431604	475542	67402

Table III compares the overall times for 9769 rules and 974548 search keys of ACL, FW and IPC filters run by TS, TC, TX, and k-TX. As shown in Table III, for k-TX, we can choose the trade-off between time and performance by selecting s and l. k-TX requires more time, since it has the complexity proportional to $\binom{s}{l}$ and performs in a short scrubbing interval.

Figs. 9(a)-(c) compare misclassifications for TS, TC, TX, and k-TX, where the probability of bit flips in a word is $P_e = 0.1$ and the probability of a scheme being used is $P_c = 0.1$. The horizontal axis denotes the maximum number of bit flips in a word of TCAM, while the vertical axis denotes the ratio of misclassifications. The ratio of misclassifications is given by:

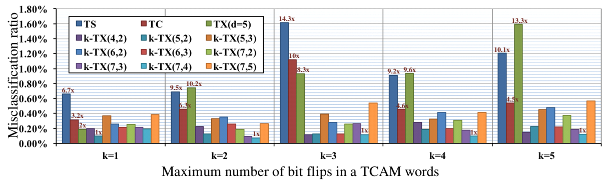
$$\sigma = \frac{\sum \textit{Misclassifications at each type of the same filters}}{\sum \textit{Numbers of search keys of the filters}}.$$

TX is designed to solve single-bit-flip errors by using tolerance degree d, where the number of X-keys is at most $p = 2^d$ [17]. Since it is only targeted for the single-bit-flip errors in the TCAM words, the number of don't-care group in the X-keys (l) is one. In this experiments, we use tolerance degree $d = 5$ for TX. As shown in Fig. 9, we can see the significant tolerance

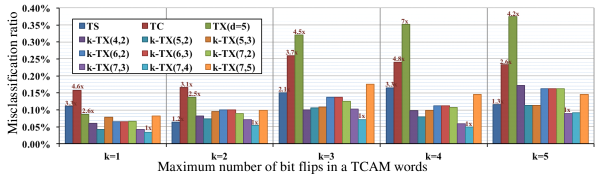
TABLE III

COMPARISON OF OVERALL DETECTION AND CORRECTION TIME FOR MULTIPLE-BIT-FLIP ERRORS

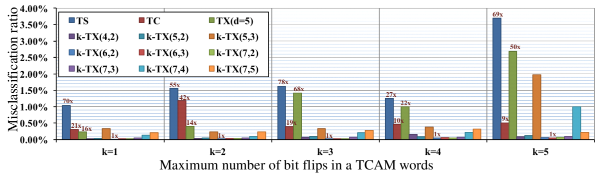
Filter	TS (sec)	TC(sec)	TX(sec)	k-TX(sec)								
			($d = 5$)	($s = 4$) ($l = 2$)	($s = 5$) ($l = 2$)	($s = 5$) ($l = 3$)	($s = 6$) ($l = 2$)	($s = 6$) ($l = 3$)	($s = 7$) ($l = 2$)	($s = 7$) ($l = 3$)	($s = 7$) ($l = 4$)	($s = 7$) ($l = 5$)
ACL	13.156	28.291	45.245	26.471	30.879	34.293	32.493	39.402	40.995	68.041	81.296	56.506
FW	21.455	43.355	119.713	46.221	56.336	79.074	62.874	82.177	79.899	129.523	130.144	75.316
IPC	0.859	1.751	5.406	2.040	2.379	2.618	2.975	3.735	3.197	5.856	5.775	4.334



(a) ACL



(b) FW



(c) IPC

Fig. 9. Comparison of Misclassification Ratio for TS, TC, TX($d = 5$), and k-TX (s, l)

improvement for the proposed method k-TX. We can see also, the TX cannot handle multiple-bit-flip errors.

As shown in Fig. 9, in terms of tolerance performance, k-TX with $s = 7$ and $l = 4$ is the best. For the ACL filter, k-TX outperforms TS up to $14\times$(k-TX(7,4), $k = 3$) and TC up to $10\times$(k-TX(7,4), $k = 3$). For the FW filter, k-TX outperforms TS up to $3\times$(k-TX(7,4), $k = 4$) and TC up to $5\times$(k-TX(7,4), $k = 4$). And for the IPC filter, k-TX outperforms TS up to $78\times$(k-TX(6,2), $k = 3$) and TC up to $42\times$(k-TX(6,2), $k = 2$). However, if we consider the time complexity, we can choose k-TX(4,2) or k-TX(5,2), so that we can get shorter runtime.

V. CONCLUSIONS

This paper has proposed a novel soft-error tolerant TCAM for multiple-bit-flip errors using partial don't-care keys (X-keys), called k-TX. k-TX corrects up to k-bit flip errors and significantly enhances the tolerance of the TCAM against soft errors, where k is the maximum number of bit flips in a word of a TCAM. k-TX consists of a TCAM, a preprocessed don't-care bit index look-up memory (X look-up), and an ECC-SRAM. First, k-TX randomly selects a search key. After that, k-TX detects multiple-bit-flip errors using the generated X-keys by X look-up. If the keys match the different locations, then a soft error is detected and k-TX refreshes the TCAM words by using a backup ECC-SRAM. Experimental results show that soft-error tolerance of k-TX outperforms other schemes up to more than $70\times$. Moreover, the hardware overhead of k-TX is small. k-TX can be easily implemented and is useful for fault tolerant packet classifiers.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Numbers 25280016, 26330072, 15K12003 and JSPS Grant-in-Aid for the Promotions of Bilateral Joint Research Projects (Japan-Germany) as well as travel assistance from Logic 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] C. Slayman, "Soft errors-Past history and recent discoveries," *International Integrated Reliability Workshop Final Report*, pp. 25-30, Oct. 2010.

[4] 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.

[5] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," *IEEE Transactions on Electron Devices*, vol. 57, no. 7, pp. 1527-1538, July 2010.

[6] M. Z. Shafiq, C. Meiners, Z. Qin, K. Shen, and A. X. Liu, "TCAM-Checker: 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.

[7] 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.

[8] W. Leung, F. Hsu, and M. E. Jones, "The ideal SoC memory: 1T-SRAM," *13th ASIC/SOC Conference*, pp 32-36, Sept. 2000.

[9] H. Noda, K. Dosaka, H. J. Mattausch, T. Koide, F. Morishita, and K. Arimoto, "A reliability-enhanced TCAM architecture with associated embedded DRAM and ECC," *IEICE Transactions on Electronics*, vol. E89-C, no. 11, Nov. 2006.

[10] 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.

[11] P.-T. Huang and W. Hwang, "A 65 nm 0.165 fJ/bit/search 256 × 144 TCAM macro design for IPv6 lookup tables," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 2, pp. 507-519, Feb. 2011.

[12] S. Pontarelli, M. Ottavi, A. Evans, and S.-J. Wen, "Error detection in ternary CAMs using bloom filters," *DATE*, pp. 1474-1479, March 2013.

[13] 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.

[14] I. Syafalni and T. Sasao, "A TCAM generator for packet classification," *IEEE ICCD*, pp. 322-328, Oct. 2013.

[15] I. Syafalni and T. Sasao, "Head-tail expression for interval functions," *IEICE Trans. Fund.*, vol. E97-A, no. 10, pp. 2043-2054, Oct. 2014.

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

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