Soft-Error Tolerant TCAMs for High-Reliability Packet Classifications

Infall Syafalni[†], Tsutomu Sasao[‡], Xiaoqing Wen[†], Stefan Holst[†], and Kohei Miyase[†]
[†]Kyushu Institute of Technology, Email: {infall@aries30., wen@, k_miyase@}cse.kyutech.ac.jp, holst@ci.kyutech.ac.jp
[‡]Meiji University, Email: sasao@cs.meiji.ac.jp

Abstract—**In the internet, packets are classified by source and destination addresses, ports, and protocol type. Ternary content addressable memories (TCAMs) are often used to perform this operation. However, high-reliability packet classifiers in aerospace network applications require soft-error tolerant TCAMs to prevent system from errors caused by environmental factors such as high-energy radiations. This paper shows a soft-error tolerant TCAM (STTCAM), which enhances the reliability of TCAMs for soft-errors. The STTCAM randomly selects a search key to be evaluated. Then, parallel TCAMs detect single-bit flip errors. When the search key matches the last word, the STTCAM calculates the similarity of the search key to the TCAM word. If 99% of similarity is detected, then a suspected error is found and the STTCAM refreshes the TCAM words by using a backup ECC-SRAM. Experimental results show that the STTCAM improves TCAMs reliability significantly than the existing scheme. The STTCAM can be easily implemented and is useful for fault-tolerant packet classifiers.**

I. Introduction

Packet classifications [1] are used in many network applications such as router and firewall. This technology uses ternary content address memories (TCAMs) [3] to perform high-performance packet forwarding. Because of it's high speed, TCAMs have been a *de facto* standard for lookup devices in the network industries. Unlike a random access memory (RAM) which only supports two possible values 0 and 1, a TCAM supports three values 0, 1, and $*$ (don't care). While a RAM searches the input address and produces the content of the memory, a TCAM compares the input vector with the content of the memory and returns the first satisfied (matched) index.

In network applications such as routers for internet, packet classifiers consist five fields: source address (SA), destination address (DA), source port (SP), destination port (DP), and the protocol type (PO). Table I shows an example of a packet classifier, where the classifier consists of three **rules**, and each rule consists of five **fields**. In this example, the rules are specified by 8-bit number source and destination addresses, 4-bit number source and destination ports, and 2-bit number protocol type. However, in IPV4, an internet address is specified by a 32-bit number, while the ports are specified by intervals of 16-bit numbers. The protocol is specified by an 8-bit number.

In Table I, the value of each field represented by an integer before the symbol $/$. An integer after the symbol $/$ denotes the number of non-don't care bits in the TCAM representation. For example, the value 44/6 means in the binary representation there are two don't care bits at the end of the vector. As shown in Table II, the ternary representation of 44/6 is 001011**, that can be represented by the interval $[11 \times 4, 11 \times 44 + 3] = [44, 47]$. Thus, the value 0/0 means the whole bits in the field are *'s, which can match all the values. The fields often have *, which denotes *don't care*. If the result of the classification is *Accept*, then the corresponding packet is sent to the next destination. Otherwise, the packet is discarded.

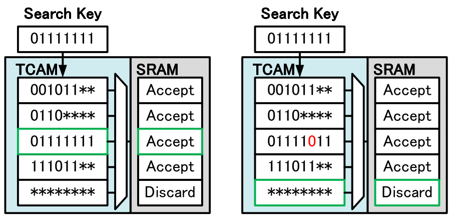
TABLE I
Example of rules in packet classifier

Index	SA	DA	SP	DP	PO	Action
1	44/6	146/7	[0,5]	15/4	2/2	Accept
2	192/6	48/6	0/0	0/4	0/2	Accept
3	0/0	0/0	0/0	0/0	0/0	Discard

TABLE II
TCAM representation

Index	SA	DA	SP	DP	PO	Action
1	001011**	1001001*	00**	1111	10	Accept
2	001011**	1001001*	010*	1111	10	Accept
3	1100****	001100**	****	0000	00	Accept
4	********	********	****	****	**	Discard

In a packet classification, rules are applied from the top to the bottom. Thus, in Table I, if we have a search key with the source address 46, the destination address 147, the source port 4, the destination port 15, and the protocol type 2, then the second rule is satisfied, and the packet is sent to the next address (Accept). If the upper rule is not satisfied, then the lower rule is checked until the last rule. Since the last rule has * in all the fields, the last rule is always satisfied. In this case, the packet is discarded. Thus, the packet classification in Table I can be considered as a five-field classification function.



(a) Correct TCAM (b) TCAM with an error

Fig. 1. Soft-error in TCAM

Soft-error is an erroneous signal that typically caused by radioactive atoms, alpha particles, cosmic rays or high-energy neutrons. It hits a memory cell and changes the value of the cell.

A soft-error will not damage hardware, but it only changes the data that is being processed, and it can produce faulty data [5]. TCAMs have higher possibility to be attacked by soft-errors than RAMs, since TCAMs have more transistors than RAMs. Moreover, the bit storage of TCAMs uses SRAM cells which are susceptible to soft-errors [4]. However, in environments that requires high-reliability network applications such as finance, aerospace, and defense networks, soft-error tolerant TCAMs or packet classifiers are necessary.

Fig. 1 shows an example of a single-bit flip error caused by a soft-error which leads to a misclassification. Suppose that a search key has a value in binary 01111111. Fig. 1(a) shows the correct match of the TCAM, while Fig. 1(b) shows a soft-error effect causing a misclassification. In this case, one bit was flipped at the third word of the TCAM that changes the value from 0111111 to 01111011. In this case, misclassification occurs, if the action of the rule where the soft-error occurred is different from the correct action. This paper shows a soft-error tolerant TCAM to enhance the reliability of TCAMs.

II. Definition and Basic Properties

A packet classifier is a form of classification function which consists of fields. In a packet classifier, we represent a field by prefix sum-of-products expressions (PreSOPs).

A. Prefix Sum-of-Products

Definition 2.1: ${x_i}^{a_i}$ denotes x_i when $a_i = 1$, and $\bar{x}_i$ when $a_i = 0$. x_i and $\bar{x}_i$ are **literals** of a variable x_i. The AND of literals is a **product**. The OR of products is a **sum-of-products expression** (SOP).

Definition 2.2: A **prefix SOP** (PreSOP) is an SOP consisting of products having the form $x^*_{n-1}x^*_{n-2}\cdots x^*_{m+1}x^*_m$, where x^*_i is x_i or $\bar{x}_i$ and $m \leq n-1$.

Example 2.1: $f(x_2,x_1,x_0) = \bar{x}_2\bar{x}_1x_0 \vee \bar{x}_2x_1 \vee x_2\bar{x}_1 \vee x_2x_1\bar{x}_0$ is a PreSOP. $f(x_2,x_1,x_0) = \bar{x}_2x_0 \vee x_1\bar{x}_0 \vee x_2\bar{x}_1$ is an SOP, for the same function, but is not a PreSOP. ∎

In high-speed network applications, PreSOPs are used instead of SOPs, since PreSOPs can be quickly generated from the binary decision trees of the functions.

B. Classification Functions

A classification function is defined as a mapping of fields specified by a set of rules. Each rule is a conjunction of fields that can be represented by PreSOPs.

Definition 2.3: A **classification function** with k **fields** is a mapping $F : P_1 \times P_2 \times \cdots \times P_k \to \{0,1,2,\cdots,r\}$, where $P_i = \{0,1,\cdots,2^{t_i}-1\}$ $(i = 1,2,\cdots,k)$. F is specified by a set of r rules. A **rule** consists of k fields, and each field is specified by an interval of t_i bits.

In IPV4, SP and DP are represented by intervals.

Definition 2.4: Let A and B be integers such that $A < B$. An **open interval** (A,B) denotes the set of integers X such that $A < X < B$. Note that endpoints are not included. The **size** of an open interval (A,B) is $C = B - A - 1$.

Definition 2.5: An n-input **open interval function** is

$$IN_0(n : A,B) = \begin{cases} 1, & \text{if } A < X < B \\ 0, & \text{otherwise.} \end{cases}$$

An interval function can be represented by a PreSOP.

Theorem 2.1 [6]: Let $\vec{a} = (a_{n-1},a_{n-2},\cdots,a_1,a_0)$ and $\vec{b} = (b_{n-1},b_{n-2},\cdots,b_1,b_0)$ be the binary representations of A and B, respectively, and $A < B$. Let s be the largest index such that $a_s \neq b_s$. Then, $IN_0(n : A,B)$ can be represented by a PreSOP:

$$\bigvee_{i=s-1}^{0}\left[\left(\bigwedge_{j=n-1}^{i+1} x_j^{a_j}\right)x_i\bar{a}_i \vee \left(\bigwedge_{j=n-1}^{i+1} x_j^{b_j}\right)\bar{x}_ib_i\right].$$

The number of products is $\sum_{i=0}^{s-1}(\bar{a}_i + b_i)$.

Example 2.2: Let $A = 0$ and $B = 7$. The binary representations of A and B are $\vec{a} = (0,0,0)$ and $\vec{b} = (1,1,1)$, respectively. Thus, by Theorem 2.1, the PreSOP for $IN_0(3 : 0,7)$ is $\bar{x}_2\bar{x}_1x_0 \vee \bar{x}_2x_1 \vee x_2\bar{x}_1 \vee x_2x_1\bar{x}_0$. ∎

C. Soft-Error in TCAM Cell

Definition 2.6: A **soft-error** in a TCAM cell is a non-permanent error that changes cell values and can cause misclassification.

Definition 2.7: A **single-bit flip** is a change of a cell value of a TCAM word caused by a soft-error. The probability of single-bit flip is $P_e = \frac{u}{N}$, where u is the number of bit-flip words and N is the number of words in a TCAM.

III. Soft-Error Tolerant TCAMs

In this section, first, we illustrate the algorithm of the STTCAM. Then, we analyze it probabilistically.

A. STTCAM Algorithm

The STTCAM uses an SRAM with error-correcting code (ECC) [4] and two TCAMs to detect soft-errors. First, the packet classification function is represented by PreSOPs. SA, DA, and PO are directly represented by PreSOPs, while SP and DP, which are intervals, are represented by Theorem 2.1. In this case, the size of the rules increases due to the interval functions. All bits of the fields of each rule are concatenated and stored in an ECC-SRAM and two TCAMs.

Fig. 2 shows the flowchart for the STTCAM. First, we inject soft-errors that cause bit-flips to both TCAMs with probability P_e. When the value of randomized value $Prob$ is less than P_c, the STTCAM is operable. Fig. 3 shows the pseudocode for the STTCAM. The search key enters both TCAMs and they are checked whether the indices are the same or different. If the indices are different, then at least one TCAM has an error. Thus, the STTCAM refreshes the words of the smaller indices of both TCAMs by using ECC-SRAM which backs up all the rules, and performs a look up operation again. If the actions are the same, the STTCAM checks the indices and check whether one or both indices are equal to $N-1$, where $N-1$ denotes the index of the last word of both TCAMs. When both indices are different from $N-1$, the STTCAM returns the smaller index of two TCAMs and do the action.

Otherwise, we assume that two TCAMs have mismatched. In this case, the words of the same indices of two TCAMs are injected by soft-errors so that the search key, that should have been matched at both TCAM words, misses both words and matches only the last words. In this case, we read the TCAMs from the beginning until we find similarity with the search key. If we find a word with 99% similarity or more, then we find the suspected error, and the STTCAM will refresh the words of the index of the both TCAMs by using ECC-SRAM. Finally, we perform a look up operation again and return the smaller index. Lines 6 to 10 show how to find a smaller index. The refresh process is in the lines 11 and 27. The similarity check is performed in lines 17 to 33.

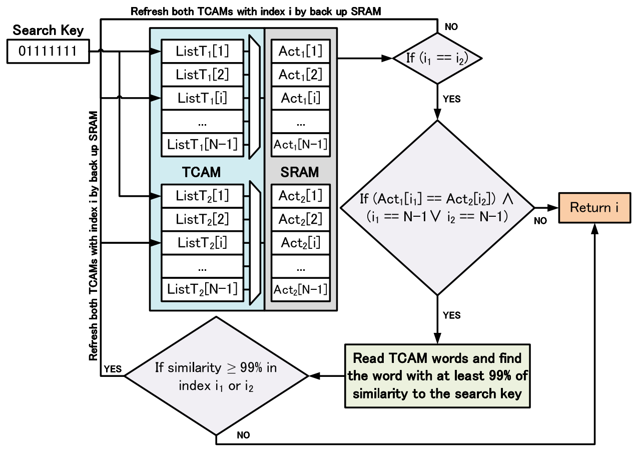

Refresh both TCAMs with index i by back up SRAM
Search Key
01111111
ListT₁[1]
ListT₁[2]
ListT₁[i]
...
ListT₁[N−1]
Act₁[1]
Act₁[2]
Act₁[i]
...
Act₁[N−1]
TCAM
SRAM
ListT₂[1]
ListT₂[2]
ListT₂[i]
...
ListT₂[N−1]
Act₂[1]
Act₂[2]
Act₂[i]
...
Act₂[N−1]
If (i₁ == i₂)
NO
YES
If (Act₁[i₁] == Act₂[i₂]) ∧ (i₁ == N−1 ∨ i₂ == N−1)
NO
YES
Return i
Read TCAM words and find the word with at least 99% of similarity to the search key
If similarity ≥ 99% in index i₁ or i₂
YES
NO
Refresh both TCAMs with index i by back up SRAM

Fig. 2. Flowchart of STTCAM

B. Probabilistic Analysis

In this paper, we only consider single-bit flip errors. Let P_e be the probability of a single-bit flip. Let N be the number of words in TCAM. Then, the probability that errors occur in u words, where $u \leq N$ is given by:

$$P(u) = \binom{N}{u} P_e^u (1-P_e)^{(N-u)}$$

The expected number of bit-flips is $E[u] = NP_e$. If we have two TCAMs, and each TCAM has the same probability of errors in the N locations, then the probability that the same locations have errors is:

$$P_d = \left(\frac{E[u]}{N}\right)^2 \times N = P_e^2 N.$$

Let c be the scale of the search keys size [7] (*i.e.,* $c = 1000$). Then, cN is the total number of search keys. Clearly, we cannot neglect the case where two TCAMs have errors at the same indices. By considering the total search keys, the total number of errors is $cN \times P_d = cN^2 P_e^2$. Although P_e^2 is quite small, cN^2 is very large. Fig. 4 shows an example where two TCAMs have errors in the same locations.

STTCAM for classification function:

```
/* Input: $\{ListT, Act\}$ that store all the factors and actions for the correct TCAM,
   $\{ListT_1, Act_1\}$ and $\{ListT_1, Act_1\}$ that store all the factors and actions for
   TCAMs with probability of injected soft-errors $P_e$, and $\{SearchK\}$ that contains
   list of search keys for evaluation. */
/* Output: $\{index\}$ that is the matched index of the TCAM. */
 1: $index_1 = TCAM_1(SearchK)$.
 2: $index_2 = TCAM_2(SearchK)$.
 3: Randomize the value of $Prob$ with range $[0.0, 1.0]$.
 4: if $P_c \geq Prob$ then
 5:     while $((index_1 \neq index_2) \vee ((Act_1[index_1] == Act_2[index_2]) \wedge$
        $(index_1 == N-1 \vee index_2 == N-1))) \wedge (i < LIMIT)$ do
 6:         if $(index_1 < index_2)$ then
 7:             $index = index_1$.
 8:         else
 9:             $index = index_2$.
10:         end if
11:         $ListT_1[index] = ListT_2[index] = ListT[index]$.
12:         $index_1 = TCAM_1(SearchK)$.
13:         $index_2 = TCAM_2(SearchK)$.
14:         $i++$.
15:     end while
16:     if $index == N-1$ then
17:         while $(Similarity < 99\%)$ do
18:             $Similarity = Compare(SearchK, ListT_1, ListT_2)$.
19:             if $(index_1 < index_2)$ then
20:                 $index = index_1$.
21:             else
22:                 $index = index_2$.
23:             end if
24:             if $(i > LIMIT)$ then
25:                 Break.
26:             else if $(Similarity \geq 99\%)$ then
27:                 $ListT_1[index] = ListT_2[index] = ListT[index]$.
28:                 $Act_1[index] = Act_2[index] = Act[index]$.
29:             else
30:                 $i++$.
31:             end if
32:         end while
33:     end if
34:     Return $index$.
35: end if
36: Terminate.
```

Fig. 3. Pseudocode for STTCAM

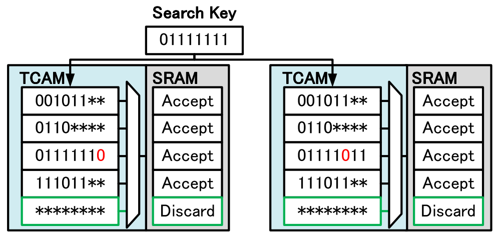


Fig. 4. Errors at two TCAMs with the same locations

IV. Experimental Results

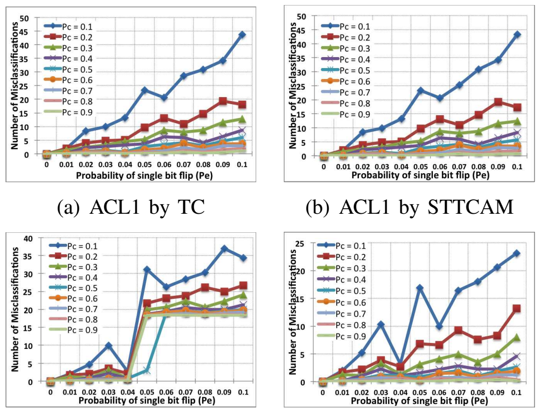
First, we generated ACL filters by ClassBench [7]. There are 5 types of ACL filters and each filter has about $N = 100$ rules. These filters can be generated by using filter seeds. Moreover, we also generated search keys lists consisting of about $cN = 100000$ ($c = 1000$) header words for each filter by using trace filters. Table IV shows the number of rules and the number search keys.

Fig. 5 compares the numbers of misclassifications of TCAM Checker (TC) [5] with that of STTCAM for ACL1 and ACL2 filters. P_c is the probability of TC and STTCAM being used

TABLE III
ACL FILTERS AND THEIR SEARCH KEYS

Type	ACL1	ACL2	ACL3	ACL4	ACL5
# Rules	99	100	100	100	98
# Search Keys	99038	100019	10000	10000	98000

for the given search keys. P_e is the probability of single-bit flip errors in the TCAMs. In this case, TC [5] and the STTCAM have similar results on the number of misclassifications. This is because, there are only a few search keys, that should match to the words, miss both TCAMs having errors for the same indices.



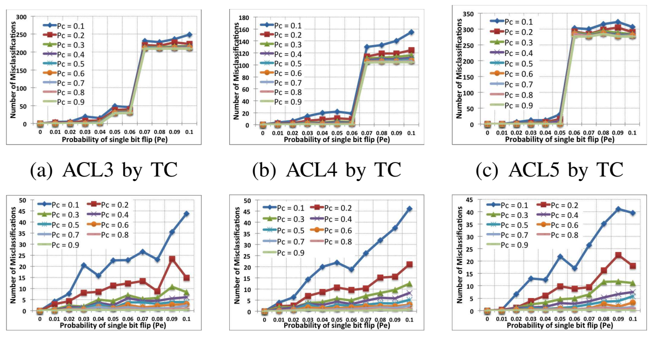
(a) ACL1 by TC (b) ACL1 by STTCAM

(c) ACL2 by TC (d) ACL2 by STTCAM

Fig. 5. Comparison of number of misclassification for ACL1 and ACL2 filters

On the other hand, Fig. 6 compares the numbers of misclassifications of TC with that of the STTCAM for ACL3, ACL4 and ACL5 filters. In this case, in TC, many frequent search keys, that should match the words, miss both TCAMs having errors at the same indices (Fig. 4). While TC is unable to resolve this problem, the STTCAM can solve it. As shown in Figs. 6(c) and 6(f), the number of misclassifications for TC with $P_c = 0.1$ and $P_e = 0.1$ is 300, while that for the STTCAM with $P_c = 0.1$ and $P_e = 0.1$ is only 40. However, before the TCAM word is refreshed by ECC-SRAM, the STTCAM needs to do similarity checks including reading and comparing processes that take more time. After the error correction, the frequent search keys always match the correct words in the TCAMs.

Finally, we compare the overhead of STTCAM with that of TC and TCAM scrubbing (TS) [2]. In term of the number of misclassifications, TC and TS were compared in [5]. Table IV compares the overheads of TS, TC and STTCAM in run-time and area complexities. STTCAM runs three time loops varied by number of TCAM words, number of comparing operations (when the indices are different) and number of reading operations performed to calculate 99% similarity. The area complexity refers to the size of TCAMs to perform the scheme, where N is the number of words in a TCAM.



(a) ACL3 by TC (b) ACL4 by TC (c) ACL5 by TC

(d) ACL3 by STTCAM (e) ACL4 by STTCAM (f) ACL5 by STTCAM

Fig. 6. Comparison of number of misclassification for ACL3, ACL4 and ACL5 filters

TABLE IV
OVERHEAD COMPARISONS

Type	TS	TC	STTCAM
Run-time	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$
Area	N	$2N$	$2N$

V. CONCLUSION

This paper showed a soft-error tolerant TCAM (STTCAM), which enhances the reliability of TCAMs. The STTCAM randomly selects a search key to be evaluated. Then, the parallel TCAMs are used to detect an error. When the search key matched the last word, the STTCAM calculates the similarity of the search key to the TCAM word. If 99% of similarity is detected, then a suspected error is found and the STTCAM refreshed the TCAM words by using a backup ECC-SRAM. Experimental results showed that STTCAM improved TCAMs reliability significantly than the existing scheme.

ACKNOWLEDGMENTS

This work is partially supported by the Japan Society for the Promotion of Science (JSPS), JST-NSC Grant-in-Aid for Japan-Taiwan Joint Research on the Testing of Nano Devices and the Adaptable and Seamless Technology Transfer Program through target-driven R&D, JST.

REFERENCES

[1] F. Baboescu and G. Varghese, “Scalable packet classification,” *IEEE/ACM TON*, vol.13, no.1, pp. 2-14, Feb. 2005.
[2] 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.
[3] 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.
[4] 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, September 2006.
[5] 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, September 2013.
[6] I. Syafalni and T. Sasao, “On the numbers of products in prefix SOPs for interval functions,” *IEICE Transaction on Information and System*, vol. E96-D, no 55, pp. 1086-1094, May 2013.
[7] D. E. Taylor, J. S. Turner, “ClassBench: a packet classification benchmark,” *IEEE/ACM TON*, vol. 3, no. 15, pp. 499-511, 2007.