Microelectronics Reliability 72 (2017) 85–89

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

A method to protect Cuckoo filters from soft errors

P. Reviriego [a,*], S. Pontarelli [b], J.A. Maestro [a]

[a] ARIES Research Center, Universidad Antonio de Nebrija, C. Pirineos 55, Madrid, Spain
[b] CNIT- National Inter-University Consortium for Telecommunications, University of Rome "Tor Vergata" Via del Politecnico 1, 00133, Rome, Italy

ARTICLE INFO

Article history:
Received 14 December 2016
Received in revised form 17 March 2017
Accepted 19 March 2017
Available online xxxx

Keywords:
Soft errors
Cuckoo filters
Error correction

ABSTRACT

Soft errors that corrupt the value of bits stored in registers or memories are a major issue for modern electronic systems. To ensure that they do not cause failures, error detection and correction codes are commonly used to protect memories. When memories are used for a specific application, sometimes it is possible to optimize the protection based on the knowledge of the application. One example is the memories used in network devices for packet processing. In particular, the protection of approximate membership check structures such as Bloom filters has been recently studied showing that it is possible to optimize the protection. Cuckoo filters are an alternative to Bloom filters for approximate membership check that has been recently proposed. Cuckoo filters are interesting as they are competitive in terms of memory usage for low false positive rates and also support the removal of elements. In this paper, the protection of Cuckoo filters against soft errors is studied showing that it can be enhanced by exploiting its structure and using the knowledge of the effects on an error in a Cuckoo filter.

1. Introduction

Soft errors caused by radiation are a major concern for modern electronic circuits [1]. Soft errors can alter the bits stored in memories and registers causing data corruption and failures. In applications where reliability is important, memories are typically protected with either a parity bit to detect single bit errors or with a Single Error Correction Double Error Detection (SEC-DED) code to correct single bit errors and detect double errors [2]. This is the case for network devices that need high levels of availability and reliability [3].

In network devices a variety of internal memories are used to store data needed for example for packet routing and classification. For example, Bloom filters [4] are widely used in networking to perform approximate membership checks and speed up packet processing. Recently, Cuckoo filters have been proposed as an alternative to Bloom filters [5]. They are attractive as they are more efficient than Bloom filters in terms of memory usage when the false positive rate is low. Another interesting feature of Cuckoo filters is that they support the removal of elements.

In any case, both Bloom and Cuckoo filters make use of memories to store information that is then checked when a packet arrives. As mentioned before, those memories can be protected with either a parity bit or a SEC-DED code. The protection of Bloom filters against soft errors has been studied in [6,7]. The first work focuses on the protection of the hash functions used in the Bloom filter while the second deals with the protection of the memories. In particular, it is shown that the protection of the memories can be optimized by exploiting the asymmetric effect of errors in the Bloom filter. A Bloom filter based approximate membership check by nature can produce false positives but not false negatives. Therefore, any soft error that only causes false positives produces only a degradation of the filter but does not change its fundamental behavior. On the other hand, an error that can create false negatives changes the nature of the filter and should be corrected. This was exploited in [7] to perform only error detection and when an error is detected on a memory word, it is set to all ones such that any false negative is avoided and the error only produces a small increase in the false positive rate of the filter. The protection of Fast Counting Bloom filters has also been optimized recently [8]. In this case a method to provide error detection when the memory used for the filter has no protection is presented. In this paper, the protection of Cuckoo filters is studied showing how it is also possible to optimize their protection based on the different effects that errors have on the filter and by using the filter structure to implement a parity bit with no memory cost.

2. Cuckoo filter overview

A Cuckoo filter is built around an array of m buckets each of which can hold up to b fingerprints. This array is typically stored in memory and thus prone to suffer soft errors. The fingerprints are obtained from the elements when they are stored in the filter using a hash function $fp = h_f(x)$. A new element can be inserted on buckets $a_1 = h_a(x)$ or $a_2 = a_1 \; xor \; h_b(fp)$ and when both buckets are full, an element stored in one of the buckets is removed and the new element is inserted on its place. Then the removed element is inserted and the process continues possibly moving several elements until an empty place is found [5]. To

* Corresponding author.
E-mail address: previrie@nebrija.es (P. Reviriego).

http://dx.doi.org/10.1016/j.microrel.2017.03.040
0026-2714/

search for an element x, first its fingerprint is computed using a hash function $fp = h_f(x)$. Then bucket a_1 is read and the fingerprints stored in that bucket are compared with fp. When there is a match the operation ends. If there is no match, then bucket a_2 is read and the fingerprints stored in that bucket are compared with fp. In this case, the operation ends with a match or with a miss as only two buckets are accessed. The removal operation is the same as a search but once found, the stored fingerprint is removed. A diagram of a Cuckoo filter is shown on Fig. 1. In this case, $b = 4$ as proposed in [5]. This configuration can achieve approximately 95% occupancy before the insertion of an element fails.

The false positive rate (fpr) of a Cuckoo filter can be easily estimated. When f bits are used for the fingerprints and the occupancy is l, the rate will be approximately:

$$fpr = \frac{8 \cdot l}{2^f} \tag{1}$$

This is obtained as on average the search will compare against $8 \cdot l$ fingerprints each having a probability of giving a false positive of 2^{-f}.

3. Protection of Cuckoo filters

A soft error that affects the memory used to store the fingerprints of a Cuckoo filter will corrupt one or more fingerprints. Let us assume for now that only single bit errors occur and therefore only one fingerprint is affected by the error. A corrupted fingerprint has the following effects when searching for an element in the Cuckoo filter:

1. It will create false positives for elements that match the corrupted fingerprint.
2. It will create false negatives for elements that matched the original fingerprint and now do not match the corrupted fingerprint.

As discussed before, having more false positives degrades the performance of the filter but does not change its nature. On the other hand having false negatives does change the expected filter behavior and should be avoided. Therefore, to avoid false positives, it would seem that an error correction code such as a SEC-DED code is needed so that the bit in error can be corrected. These codes require a number of additional bits per memory word and thus increase the cost. The read and write operations also need to perform a decoding and encoding operation that add delay and power over an unprotected memory.

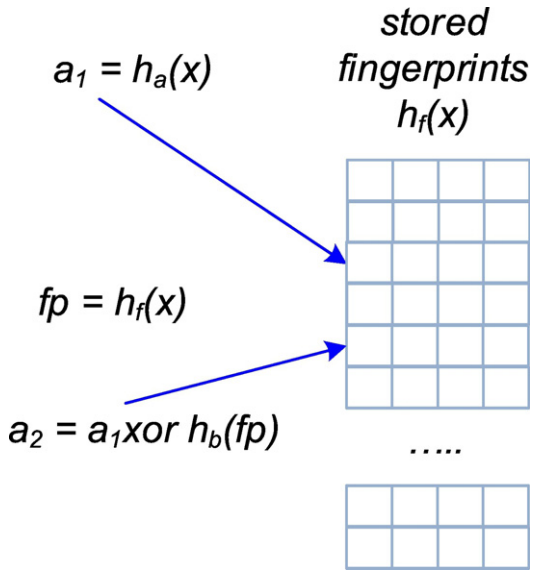

Fig. 1. Illustration of a Cuckoo filter.

A lower cost protection would be the use of a parity bit but that only detect errors. Let us assume that the memory that stores the fingerprints is protected with a parity bit. Then, when a memory word is read during a search operation and a parity error is detected, a possible solution is to return positive regardless of the contents of the word. This strategy is similar to the one proposed in [7] for Bloom filters and ensures that the error will not create false negatives at the cost of increasing the false positive rate. Assuming that a memory word corresponds to a bucket in the Cuckoo filter, the increment will be approximately $2/m$ as now one of the m positions always returns a positive. This increment in the false positive rate would be negligible for large filters. The same scheme could be applied to a memory protected using a SEC-DED code when a double error is detected. However, more refined schemes can be designed to minimize the impact of soft errors on the filter false positive rate. Two such schemes are presented in the following subsections the first providing protection against single errors and the second against double errors.

3.1. Single Error Protection (SEP) scheme

To protect against single errors, one option is to use a memory that has a parity bit and when there is a parity error use distance one comparisons for the fingerprints. Such comparisons give a match when the fingerprints are the same or they differ only in one bit. Therefore, if a stored fingerprint has suffered an error and it is compared with its initial value, it would give a match as there will only be one bit that is different (the bit that has changed by the error). Therefore, this scheme cannot produce false negatives. An f bit fingerprint will now match also f distance one values (corresponding to an error on each of its bits). This causes an increment in the false positive rate of approximately:

$$\frac{4 \cdot l \cdot f}{2^f} \tag{2}$$

when that bucket is selected on a query. As each query selects two buckets, the probability that the bucket in error is selected will be approximately $2/m$. Therefore the increment due to one error will be approximately:

$$\frac{4 \cdot l \cdot f}{2^f} \cdot \frac{2}{m} = \frac{8 \cdot l}{2^f} \cdot \frac{f}{m} \tag{3}$$

To ensure that soft errors have a negligible effect on the overall false positive rate, the increment given by Eq. (3) has to be much smaller than the initial value given by Eq. (1) which occurs when:

$$m \gg f \tag{4}$$

This is the case in most practical scenarios as the number of fingerprint bits f is small with typical values in the rage of 8 to 16 bits while the number of buckets in the filter is large with values easily exceeding 16 K buckets and reaching several millions in some cases [5]. The previous calculations have been done considering that there is only one soft error in the memory at a given point in time. This is a reasonable assumption as soft errors are rare events that occur with low frequency in terrestrial applications. For example, a per bit soft error rate of $8.8 * 10^{-9}$ per year has been estimated for some 65 nm memories [9]. In the case that more than one soft error affect the memory, the condition to ensure that the impact on the false positive rate is small will be:

$$m \gg f \cdot e \tag{5}$$

where e is the number of soft errors.

The distance one comparison requires a more complex hardware than an exact match comparison but as will be shown later it is simpler than a SEC-DED decoder. In the rest of the paper, this refined scheme based on distance one comparisons is used.