International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

Design and Implemenation Single Error Correction Codes with Bloom Filter

[1]A MOUNIKA, [2]V SWATHANTHRA

[1]Pg Scholar, Department of ECE,Vaageswari college of engineering, Karimnagar

[2]Assoc.Prof, Department of ECE,Vaageswari college of engineering, Karimnagar

ABSTRACT

Bloom filters (BFs) provide a fast and efficient way to check whether a given element belongs to a set. BFs are widely used in various applications like networking, computer architectures and communications. At present these bloom filters are also extended to various scenarios. In advanced electronic circuits, the reliability becomes a challenge due to the increase in radiation, manufacturing variations and reduction of noise margins as technology improves. Here, the BFs are used to detect and correct errors in the associated data set. This gives a reuse of existing BFs to also detect and correct errors such that there is no need to add extra error correction methods which reduces the area of the proposed method.

I. INTRODUCTION

Bloom filter checks whether an element belongs to a set in a simple and efficient way [1]. It is a probabilistic data structure. It is used in various computer architectures and networking applications [2]. The BFs are also used to reduce data lookups in the large data bases for example in Google Bigtable [3]. The extension of the BF basic structures are implementing over years for example counting BFs are implemented in order to remove the elements in the BF [4]. Another extension in order to enhance the transmission in network, compressed BFs was proposed [5]. To achieve error correction in large data sets, Bloom filter (Biff) codes are proposed recently that are based on Bloom filter [6]. Bloom filters are mostly implemented using electronic circuits [7], [8]. In order to get high performance, Bloom filters contents are generally stored in high speed memory and the processing is implemented in a processor or in separate circuitry. The element set which are used to construct Bloom filter is generally stored in low speed memory [9]. As technology scales, the challenging thing for electronic circuits is to maintain reliability. The most common errors occurred due to radiation,

interferences and other effects. So, in order to operate the circuits reliably at various levels the mitigation methods are implemented. The important element in implementation of Bloom filters are memories. By using spare rows and columns methods, the permanent errors are generally corrected for memories. But, soft errors which are occurred due to radiation can affect the memory cell by changing the value during circuit operation. Soft errors are one which doesn‟t damage the memory device but produces a wrong value in the memory cell i.e., by converting zeros to ones or ones to zeros but operates correctly [10]. Generally in memories, the per word parity bit of different higher error correction codes (ECCs) are used to deal soft errors [11]. In electronic circuits to reduce errors the Bloom filters are proposed. For example, to detect faulty words in nanomemory the Bloom filters is used [12]. In [13], the CBF is used to detect and correct errors in content addressable memories (CAMs). Here, the purpose is to find errors in CAM entries where the CBF and CAM are used in parallel. This can be obtained by studying the results of CAM and CBF such that they are uniform. If an error is found, the correction method is started to get the correct value in the modified CAM entry by using the additional copy of its contents. The BFs are included externally in both the cases to detect and correct errors but it doesn‟t exist in the actual design. In Biff codes, the same method applies where the error correction is done by using extended Bloom filters. Therefore, in all the above cases the bloom filters are added externally but actually not present in the original design. This is distinct to the reuse of existing BFs which already exists in system for error correction

II LITERATURE SURVEY.

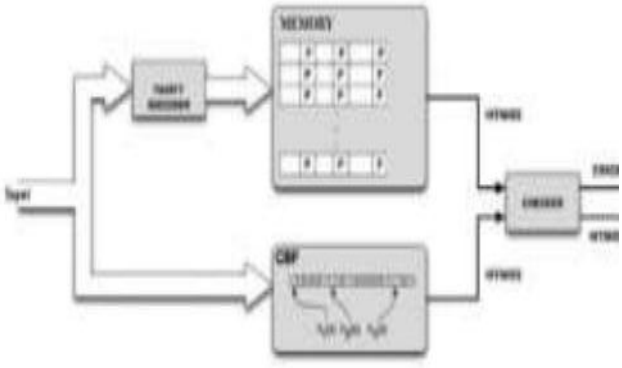
Kazuteru Namba et al., (1) proposed a BCH decoder that corrects double-adjacent and single-bit errors in parallel which resembled a Wilkerson's parallel BCH decoder which is used to correct only single-bit errors. The decoder section is operated serially for a double and double adjacent error where in the memory system the probability of occurrence of single and multi-bit error occurs in the memory system. Amit Kumar et al., (2) presented a design of a (15,k) BCH encoder on FPGA with multi bit error correction control, in which the primitive polynomial is implemented in the Linear Shift Register logic .By the use of the cyclic codes, the reminder parameter b(x) is calculated in a LFSR (15,k) with feedback

connections which corresponds to the coefficient of the polynomial generated. In this technique three encoders are designed to encode the single, double and triple error correcting BCH codes

III. Implementation

The proposed method represents that CBF in addition to structure that performs fast membership check to element set, also provide a redundant representation of the element set. Hence, this redundancy can be used for error detection and correction. To analyze this method, general implementations of CBFs where the slow memory is stored with elements of set and faster memory is used to store the CBF. Generally, it is considered that elements of set are stored in DRAM and CBF is stored in cache [9]. The reason behind this is that elements of set are accessed only when elements are read, added or removed hence access time is not an issue but CBF needs to be accessed frequently hence it requires fast access time to increase its performance. Since entire element set is stored in slow memory, no incorrect deletion will be present as it would be found when the element is getting removed from slow memory. Hence, false negative in CBF is not an issue in our method. Generally, memories are protected with per word parity bit or with single bit error correction code [11]. This is considered based on observation that most errors affect single bit or even if they affect various bits, the errors can be divided among various words by use of interleaving [16]. The time between the soft errors is mostly large because they are rare events [10]. Generally these errors are considered as isolated due to arrival rate of terrestrial applications is of at least days or weeks. Hence, if the soft error arrives by a time any previous error can be detected or corrected. This is the consideration needed when single bit error correction codes are used. In this method, it is considered that both DRAM and cache are protected with per word parity bit which can detect single errors. Hence, by using single bit error correction codes it is considered that error are isolated. The objective for this method implementation is to correct single bit errors using CBF. Hence, without having the cost of adding an ECC to memories, the CBF will give single bit error correction. The initial step to perform error correction is to detect errors. It is performed by checking parity bit while accessing either DRAM or caches. To have earlier detection of errors, the use of scrubbing to periodically read memories is considered [15]. If the error

gets detected, then the correction procedure is implemented. Hence, the error can occur either in CBF itself or the element set which is stored in memory. If error occurs in CBF, then it is corrected by clearing entire CBF and reconstructed it using element set. If error occurs in element set, then the procedure is complex and it is divided in two phases that are represented in the following methods. First the fast and simple procedure is used and if it is unable to correct error then and advanced and complex procedure for error correction is used.



IV CONCLUSION

In this brief, a new application of BFs is proposed. The idea is to use BFs in existing applications to also detect and correct errors in their associated element set. In particular, it shows that CBFs can be used to correct errors in associated element set. This gives a cost efficient solution to reduce soft errors in applications which use CBFs. The configuration considered here is that the memory protected with per word parity bit for which the CBF is used to get single bit error correction. This show how existing CBFs can be used to get error correction in addition to perform the membership checking function. The general idea can also be used when memory is protected with more advanced codes. For example, if a SEC-DED code is used, the CBF is used to correct double errors. In addition, the simplest part of error correction method can be applied to BFs to get some degree of error detection and correction.

References

[1] B. Bloom, “Space/time tradeoffs in hash coding with allowable errors, ”Commun. ACM, vol. 13, no. 7, pp. 422–426, 1970. .

[2] A. Broder and M. Mitzenmacher, “Network applications of bloom filters: A survey,” in Proc. 40th Annu. Allerton Conf., Oct. 2002, pp. 636–646.

[3] C. Fay et al., “Bigtable: A distributed storage system for structured data,” ACM TOCS, vol. 26, no. 2, pp. 1–4, 2008.

[4] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, “An improved construction for counting bloom

filters," in Proc. 14th Annu. ESA, 2006, pp. 1–12.

[5] M. Mitzenmacher, "Compressed bloom filters," in Proc. 12th Annu. ACM Symp. PODC, 2001, pp. 144– 150.

[6] M. Mitzenmacher and G. Varghese, "Biff (Bloom Filter) codes: Fast error correction for large data sets," in Proc. IEEE ISIT, Jun. 2012, pp. 1–32.

[7] S. Elham, A. Moshovos, and A. Veneris, "L-CBF: A low-power, fast counting Bloom filter architecture," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 6, pp. 628–638, Jun. 2008.

[8] T. Kocak and I. Kaya, "Low-power bloom filter architecture for deep packet inspection," IEEE Commun. Lett., vol. 10, no. 3, pp. 210–212, Mar. 2006.

[9] S. Dharmapurikar, H. Song, J. Turner, and J. W. Lockwood, "Fast hash table lookup using extended bloom filter: An aid to network processing," in Proc. ACM/SIGCOMM, 2005, pp. 181–192.

[10] M. Nicolaidis, "Design for soft error mitigation," IEEE Trans. Device Mater. Rel., vol. 5, no. 3, pp. 405–418, Sep. 2005.

[11] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," IBM J. Res. Develop., vol. 28, no. 2, pp. 124–134, 1984. [12] G. Wang, W. Gong, and R. Kastner, "On the use of bloom filters for defect maps in nanocomputing," in Proc. IEEE/ACM ICCAD, Nov. 2006, pp. 743–746.

[13] S. Pontarelli and M. Ottavi, "Error detection and correction in content addressable memories by using bloom filters," IEEE Trans. Comput.,vol. 62, no. 6, pp. 1111–1126, Jun. 2013.

[14] A. Reddy and P. Banarjee, "Algorithm-based fault detection for signal processing applications," IEEE Trans. Comput., vol. 39, no. 10, pp. 1304– 1308, Oct. 1990.

[15] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of scrubbing recovery-techniques for memory systems," IEEE Trans. Rel., vol. 39, no. 1, pp. 114–122, Apr. 1990.

[16] P. Reviriego, J. A. Maestro, S. Baeg, S. J. Wen, and R. Wong, "Protection of memories suffering MCUs through the selection of the optimal interleaving distance," IEEE Trans. Nucl. Sci., vol. 57, no. 4,pp. 2124–2128, Aug. 2010.

[17] Pedro Reviriego, Salvatore Pontarelli, "A synergetic use of bloom filters for error

detection and correction", IEEE Transactions on VLSI systems, VOL. 23, NO. 3, March 2015.

AUTHOR'S PROFILE:

A MOUNIKA, Pg Scholar, Department of ECE,Vaageswari college of engineering, Karimnagar

V SWATHANTHRA, Assoc.Prof, Department of ECE,Vaageswari college of engineering, Karimnagar