ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 6, Issue 11, November 2017

A bloom filters based data management with error correction and detection

Vaisanthi.P

Department of ECE

PSNA CET

Dindigul,Tamil Nadu,India

ABSTRACT:

Bloom filters have been used to reduce the delay in networking and computing applications when a set membershipcheck is to be applied. Error sources can affect the behavior of Bloom filters resulting in a wrong outcome of this membershiptest and a possible effect in the system's output. Single event transients are a type of temporary errors altering the operation ofcombinational logic. A single event transient affecting the hash generation logic of a hardware-implemented Bloom filter canproduce errors such as false negatives. The main drawback of the existing system is that counting bloom filter and compressed bloom filter. In order to overcome the comparator technique. The compares incoming data and storage data.

KEYWORDS:Bloom filters, Error detection,counter filter,compressed filter,modelsim,comparator.

1.INTRODUCTION:

Bloom filters are probabilistic data structuresused in computing and networking applications whenit is necessary to check if an element belongs to a set.They can produce false positives, but false negativesare not possible. Some examples of the application of BFs include data storage systems such as Google Bigtable , different networking scenarios and privacy protection systems.

when the system implemented in hardware, it may suffer reliability problems such as soft errors andmanufacturing defects. One of these types of errors areSingle Event Transients , which are temporary errorsthat may affect the combinational logic of a system.SETs may be caused by a number of reasons, includingradiation sources. Hash generation circuits included in aBloom filter may suffer from temporary errors due toSETs.

An SET can cause a membership query to return thewrong result, indicating that an element is in the set whenit is not or stating that an element is absentwhen it is in the set . As explained before,false positives are allowed in Bloom filters, and the effectwould only be a small reduction in performance. However,false negatives may cause the system to misbehave and they should be prevented. To do so, a simple approachcould be to replicate the circuitry using traditionalschemes such as Triple Modular Redundancy andDual Modular Redundancy .

However, in manycases, the increase in power and area consumption ofthese schemes makes necessary to find other approaches.The protection of Bloom filters against errors has

beenThe studied in different works. In , permanent errors producedin hash generation circuits of a Bloom filter werestudied and a solution oriented to this type of errors wasproposed. The scheme uses additional spare hash unitsthat are activated when a possible false negative occurs toverify if there is a faulty unit to replace it with one of thespare ones. In , a scheme to protect the content of theBloom filter is presented.

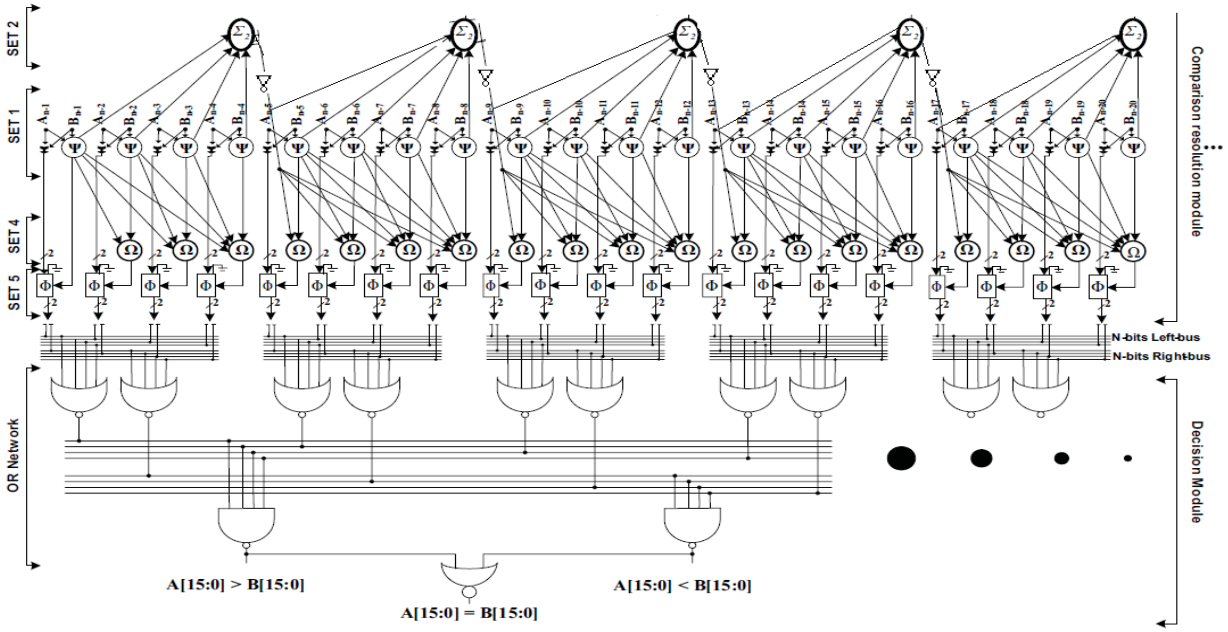
The scheme focuses in thememory protection against single event upsets anddoes not provide solutions for the hash circuitry. It can becombined with the schemes presented in this paper tobuild a fault tolerant solution. Bloom filters have alsobeen used to identify defects or errors in other structuressuch as nano-memories , Content Addressable Memories, or the data set associated with the filterThis paper presents and compares different schemes toprotect Bloom filter implementations against SETs. Therest of the paper is organized as follows: first, a descriptionof the architecture and operation of Bloom Filters.

2.RELATED WORK:

Bloom filters have an essential role in network services and consequently the growing importance of operations such as information retrieval, distributed databases, packet content inspection, and cooperative caching results in the wide applications of Bloom filters that provide set-membership queries based on a relatively easy hardware implementation. The main design tradeoffs are the number of hash functions used, the size of the filter and the error ratecounting Bloom filter generalizes a Bloom filter data structure so as to allow membership queries on a set that can be changing dynamically via insertions and deletions. A pipelined Bloom filter consists of two groups of hash functions. The first stage always computes the hash values. By contrast, the second stage of hash functions only compute the hash values if in the first stage there is a match between the input and the signature sought. compressing Bloom filters might lead to significant bandwidth savings at the cost of higher memory requirements and some additional computation time to compress the filter that is sent across the network. We do not detail here all theoretical and practical issues analyzed.

3.PROPOSED WORK:

A bloom filter is a binary data management filter used to compare a incoming data with data stored in memory. In my proposed method I am going to use comparator technique. If I give a value which already store means it will not add, if it is not have the value will add.



Details of Implementation Architecture

Set 1 compares the N-bit operands A and B bit-by-bit, using a single level of N ψ type cells. The ψ type cells provide a termination flag D_k to cells in sets 2 and 4, indicating whether the computation should terminate. these cells compute (where $0 \leq k \leq N-1$).

$$\psi: D_k = A_k B_k. \qquad (1.1)$$

Set 2 consists of Σ_2 type cells, which combine the termination flags for each of the four ψ type cells from set 1 (each 2-type cell combines the termination flags of one 4-b partition) using OR-logic to limit the fan-in and fan-out to a maximum of four. The Σ_2 type cells either continue the comparison for bits of lesser significance if all four inputs are 0s, or terminate the comparison if a final decision can be made. For $0 \leq m \leq N/4–1$, there is a total of $N/4$ Σ_2 type cells, all functioning in parallel .

Set 3 consists of Σ_3 type cells, which are similar to Σ_2-type cells, but can have more logic levels, different inputs, and carry different triggering points. A Σ_3-type cell provides no comparison functionality; the cell's sole purpose is to limit the fan-in and fan-out regardless of operand bit-width. To limit the Σ_3-type cell's local interconnect to four, the number of levels in set 3 increases if the fan-in exceeds four. Set 3 provides functionality similar to set 2 using the same NOR logic to continue or terminate the bitwise comparison activity. If the comparison is

terminated, set 3 signals set 4 to set the left bus and right bus bits to 0 for all bits of lower significance. For $0 \leq m \leq N/4 - 1$, there is a total of $N/4$ Σ_3-type cells per level, with cell function and number of levels as

$$\sum_3 : C_{3,m} = \qquad (1.2)$$

$$\text{Level}S_{set3} = ([\log_{16}(N)]) \qquad (1.3)$$

From left to right, the first four Σ3-type cells in set 3 combine the partition comparison outcomes from the one, two, three, and four partitions of set 2. Since the fourth Σ_3-type cell has a fan-in of four, the number of levels in set 3 increases and set 3's fifth Σ_3-type cell combines the comparison outcomes of the first 16 MSBs with a fan-in of only two and a fan-out of one.

The simulation-based analysis of leakage power dissipation showed that, whereas the percentage contribution of leakage power increases with each new technology generation, the increase effect is not significant enough to nullify the savings in dynamic power dissipation in near-future technologies. Future work will include additional circuit optimizations to further reduce the power dissipation by adapting dynamic and analog implementations for the comparator resolution module and a high-speed zero-detector circuit for the decision module. Given that our comparator is composed of two balanced timing modules, the structure can be divided into two or more pipeline stages with balanced delays, based on a set structure, to effectively increase the comparison throughput at the expense of increased power and latency.

4.SIMULATION RESULTS:

Bloom filters output:

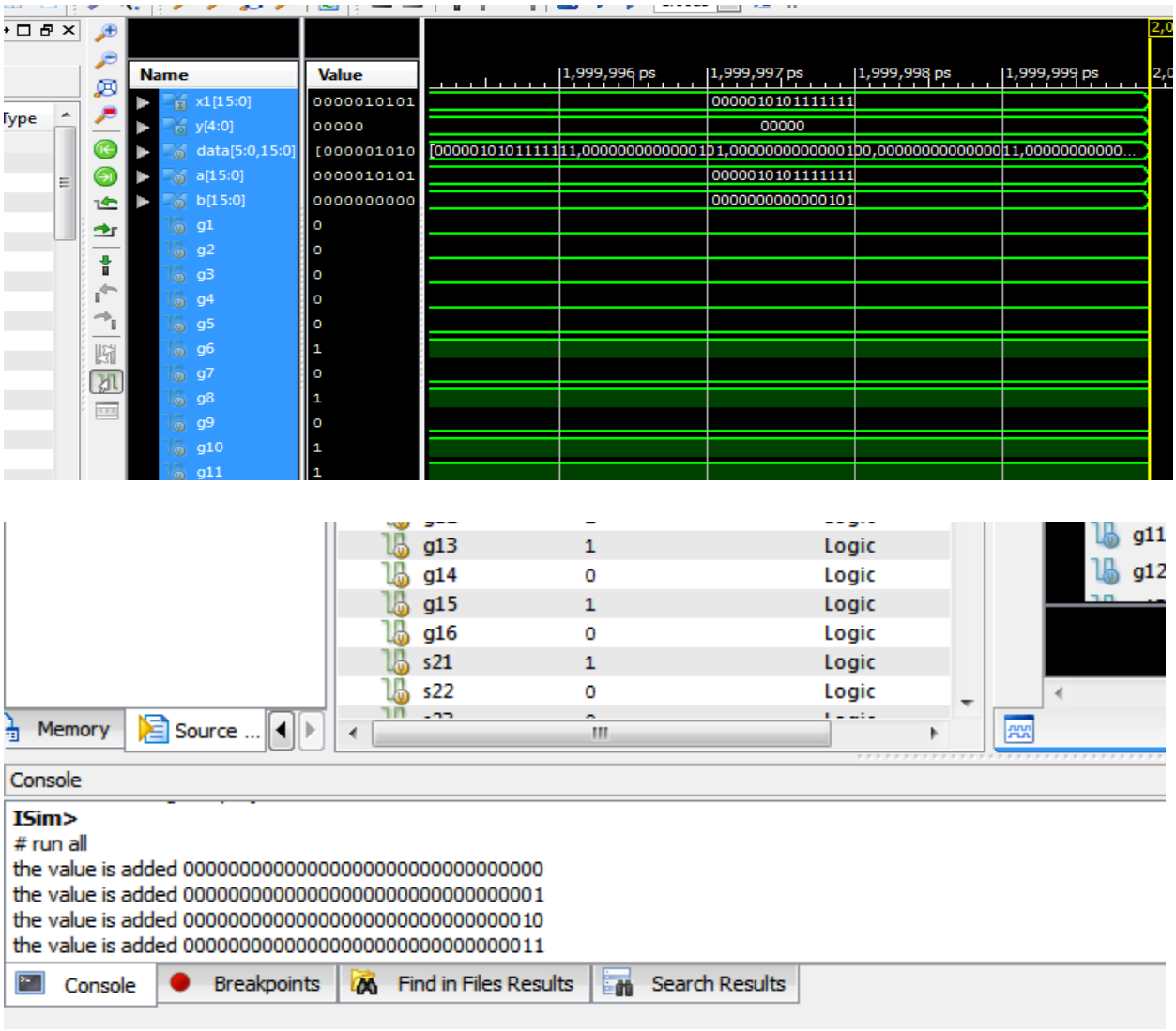


Figure: Data Insertion

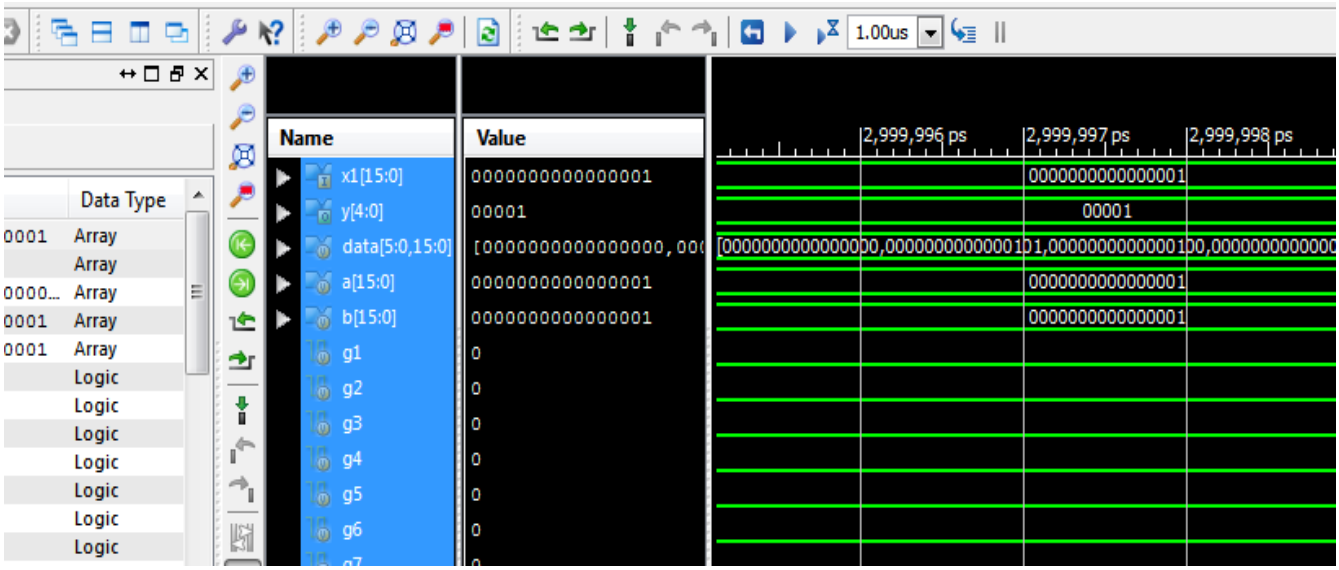


Figure:Bloom filters output

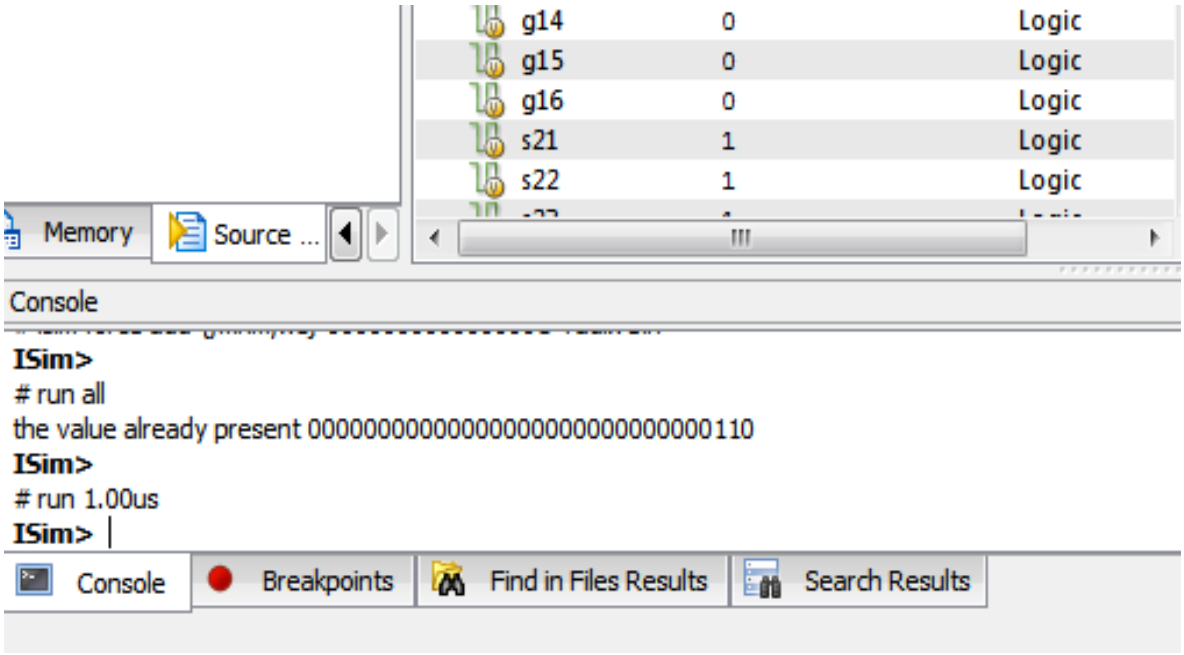


Figure:Data Rejection

5.CONCLUSIONS:

In this brief, a new application of Bloom Filters has been proposed. The idea is to use high-speed low-power comparator in Bloom Filters to compare element set. In particular omparator structured as parallel prefix trees with repeated cells in the form of simple stages that are one gate level deep with a maximum fan-in of five and fan out of four, independent of the input bit width. simulation results shows our proposed bloom filter has improved performance in terms of both comparison time and memory protection.

REFERENCES:

[1] 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.
[2] 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.
[3] A.Broder and M. Mitzenmacher, "Network applications of bloom filters:A survey," in Proc. 40th Annu. Allerton Conf., Oct. 2002, pp. 636–646.
[4] M. Mitzenmacher, "Compressed bloom filters," in Proc. 12th Annu. ACM Symp. PODC, 2001, pp. 144–150.
[5]C.Fay et al., "Bigtable: A distributed storage system for structured data,"ACM TOCS, vol. 26, no. 2, pp. 1–4, 2008.
[6] B.Bloom, "Space/time tradeoffs in hash coding with allowable errors,"Commun. ACM, vol. 13, no. 7, pp. 1159