ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

EFFICIENT ERROR DETECTION AND CORRECTION BY COMBINED BLOOM FILTER

PRIYA.M

P.G Scholar, ECE, AVS Engineering College, Salem, Tamil Nadu, India

Mpriya.pretty3@gmail.com

ABSTRACT-Bloom filter is effective, space-efficient data structure for concisely representing a data set and supporting approximate membership queries and provides a fast way to check whether a given element belongs to a set. In this brief, it is shown that BFs can be used to detect and correct errors in their associated data set. This allows a synergetic reuse of existing BFs to also detect and correct errors. This is illustrated through an example of a counting BF used for IP traffic classification.

1. INTRODUCTION

Bloom filters (BFs) provide a simple and effective way to check whether an element belongs to a set . They are used in many networking applications as well in computer architectures . The BFs are also used in large databases (e. g. , Google Bigtable uses it to reduce the disk lookups).The basic structure of BFs has also been extended over the years. For example, counting BFs (CBFs) were introduced to allowremoval of elements fromthe BF. To optimize the transmission over the network, another extension known as compressed Bloom filters was proposed. Recently Bloom filter (Biff) codes that are based on BFs have been proposed to perform error correction in large data sets .In most cases, BFs are implemented using electronic circuits . The contents of a BF are commonly stored in a

high speed memory and required processing is done in a processor or in dedicated circuitry. The set used to construct the BF is also commonly stored in a lower speed memory[1]-[3].

The reliability of electronic circuits is becoming a challenge as technology scales. Errors caused by interferences, radiation, and other effects become more common. Therefore, mitigation techniques are used at different levels to ensure that the circuits continue to operate reliably. For BFimplementation, memories are a critical element.For memories, permanent errors and defects are commonly corrected using spare rows and columns . However, soft errors caused for example by radiation can affect any memory cell changing its value during circuit operation. Soft errors do not produce damage to the

memory device that continues to operate correctly but has the wrong value in the affected cell . To deal with soft errors, the use of a per word parity bit or more advanced error correction codes (ECCs) has been common in memories for many years .

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

1.1 ERROR CORRECTION:

In information theory and coding theory with applications in computer science and telecommunication, error detection and correction or error control are techniques that enable reliable delivery of digitaldata , unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data in many cases. Christo Ananth et al. [2] discussed about Improved Particle Swarm Optimization. The fuzzy filter based on particle swarm optimization is used to remove the high density image impulse noise, which occur during the transmission, data acquisition and processing. The proposed system has a fuzzy filter which has the parallel fuzzy inference mechanism, fuzzy mean process, and a fuzzy composition process. In particular, by using no-reference Q metric, the particle swarm optimization learning is sufficient to optimize the parameter necessitated by the particle swarm optimization based fuzzy filter, therefore the proposed fuzzy filter can cope with particle situation where the assumption of existence of "ground-truth" reference does not hold. The merging of the particle swarm optimization with the fuzzy filter helps to build an auto tuning mechanism for the fuzzy filter without any prior knowledge regarding the noise and the true image. Thus the reference measures are not need for removing the noise and in restoring the image. The final output image (Restored image) confirm that the fuzzy filter based on particle swarm optimization attain the excellent quality of restored images in term of peak signal-to-noise ratio, mean absolute error and mean square error even when the noise rate is above 0.5 and without having any reference measures.If only error detection is required, a receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits; [4],[5],[6]if the values do not match, an error has occurred at some point during the transmission. In a system that uses a non-systematic code, the original message is transformed into an encoded message that has at least as many bits as the original message.

Good error control performance requires the scheme to be selected based on the characteristics of the communication channel. Common channel models include memory-less models where errors occur randomly and with a certain probability, and dynamic models where errors occur primarily in bursts. Consequently, error detecting and correcting codes can be generally distinguished between random error detecting / correcting and burst error detecting/correcting. Some codes can also be suitable for a mixture of random errors and burst errors.

Error detection is most commonly realized using a suitable hash function (or checksum algorithm). A hash function adds a fixed-length tag to a message, which enables receivers to verify the delivered message by recomputing the tag and comparing it with the one provided. There exists a vast variety of different hash function designs. However, some are of particularly widespread use because of either their simplicity or their suitability for detecting certain kinds of errors (e.g., the cyclic redundancy check's performance in detecting burst errors).

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

A random-error-correcting code based on minimum distance coding can provide a strict guarantee on the number of detectable errors, but it may not protect against a preimage attack. A repetition code, described in the section below, is a special case of error-correcting codes: although rather inefficient, a repetition code is suitable in some applications of error correction and detection due to its simplicity. Any error-correcting code can be used for error detection. A code with minimum Hamming distance, d, can detect up to d − 1 errors in a code word. Using minimum-distance-based error-correcting codes for error detection can be suitable if a strict limit on the minimum number of errors to be detected is desired. Codes with minimum[7],[8],[9],[10] Hamming distance d = 2 are degenerate cases of error-correcting codes, and can be used to detect single errors. The parity bit is an example of a single-error-detecting code.

An error-correcting code (ECC) or forward error correction (FEC) code is a process of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a backchannel is not required in forward error correction, and it is therefore suitable for simplex communicationsuch as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks, and RAM.

Error-correcting codes are usually distinguished between convolutional codes and block codes: Convolutional codes are processed on a bit-by-bit basis. They are particularly suitable for implementation in hardware, and the Viterbi decoder allows optimal decoding. Block codes are processed on a block-by-block basis. Early examples of block codes are repetition [11],[12] codes, Hamming codes and multidimensional parity-check codes. They were followed by a number of efficient codes, Reed–Solomon codes being the most notable due to their current widespread use. Turbo codes and low-density parity-check codes (LDPC) are relatively new constructions that can provide almost optimal efficiency.

1.2 SOFT ERRORS:

In electronics and computing, a soft error is a type of error where a signal or datum is wrong. Errors may be caused by a defect, usually understood either to be a mistake in design or construction, or a broken component. A soft error is also a signal or datum which is wrong, but is not assumed to imply such a mistake or breakage. After observing a soft error, there is no implication that the system is any less reliable than before. In the spacecraft industry this kind of error is called a single-event upset. In a computer's memory system, a soft error changes an instruction in a program or a data value. Soft errors typically can be remedied by cold booting the computer. A soft error will not damage a system's hardware; the only damage is to the data that is being processed.

There are two types of soft errors, chip-level soft error and system-level soft error.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

Chip-level soft errors occur when the radioactive atoms in the chip's material decay and release alpha particles into the chip. Because an alpha particle contains a positive charge and kinetic energy, the particle can hit a memory cell and cause the cell to change state to a different value. The atomic reaction is so tiny that it does not damage the actual structure of the chip. System-level soft errors occur when the data being processed is hit with a noise phenomenon, typically when the data is on a data bus. The computer tries to interpret the noise as a data bit, which can cause errors in addressing or processing program code. The bad data bit can even be saved in memory and cause problems at a later time.

1.3 OVERVIEWOF BFS

A BF is constructed using a set of k hash functions to access an array of m bits. The hash functions $h_1, h_2, \ldots, h_k$ map an input element x to one of the m bits. The following two operations are defined in a BF.

1) Insertion: To insert an element x in the BF, the bits in the array that correspond to the positions $h_1(x), h_2(x), \ldots, h_k(x)$ are set to one.
2) Query: To query for an element x in the BF, the bits in the array that correspond to the positions $h_1(x), h_2(x), \ldots, h_k(x)$ are read and if and only if all of them are one, the element is considered to be in the BF.

This operation guarantees that if an element has been added to the BF, it will be found when a query for it is done. However, a BF can produce false positives when a query for an element that has not been added to the BF is done. That is an element is incorrectly classified as being stored in the BF when in fact is not in the element set. This can occur if other elements have set to one the positions that correspond to the hash values of that element.The hash functions are uniformly distributed, after inserting n elements in the BF, the probability p0 (n) that a given bit in the array is zero can be approximated as

$$p_0(n) \cong \left(1 - \frac{1}{m}\right)^{kn} \cong e^{-\frac{k \cdot n}{m}}. \quad (1)$$

Therefore, the probability of a false positive can be approximated as

$$p_{fp}(n) \cong (1 - p_0(n))^k \cong \left(1 - e^{-\frac{k \cdot n}{m}}\right)^k. \quad (2)$$

It can be observed that pfp depends on (1 –p0 (n)) and k. The first expression gives the probability that an element in the CBF has a value different than zero and is commonly known as the load factor. [13]-[16]The load factor gives an indication of how many elements have been inserted in the CBF and also of the false positive probability. The load factor will be used in the experiments presented in this brief and is defined as

$$lf \cong (1 - p_0). \quad (3)$$

A problem with BFs is that elements cannot be easily removed. This is because a position with a one in the array can be shared by several elements and thus clearing the $h_1(x), h_2(x), \ldots, h_k(x)$ positions for an element x may also affect other elements

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

in the BF. To address this issue, CBFs which are a generalization of BFs were introduced. In a CBF, the array of m bits is replaced with an array of integers of b bits and the operations are defined as follows.

1) Insertion: To insert an element x in the CBF, the integers in the array that correspond to the positions $h_1(x)$, $h_2(x)$, …, $h_k(x)$ are incremented by one.

2) Query: To query for an element x in the CBF the integers in the array that correspond to the positions $h_1(x)$, $h_2(x)$, …, $h_k(x)$ are read and if and only if all of them are larger than zero the element is considered to be in the CBF.

3) Removal: To remove an element x from the CBF, the integers in the array that correspond to the positions $h_1(x)$,$h_2(x)$, …, $h_k(x)$ are decremented by one.

The use of integers instead of bits allows the removal of elements as now each position in the array stores the number of elements that share that position. The false positive rate of a properly dimensioned CBF is the same as that of a standard BF.

1.4 PROPOSED SCHEME

The proposed scheme is based on the observation that a CBF,in addition to a structure that allows fast membership check to an element set, is also in a way a redundant representation of the element set. Therefore, this redundancy could possibly be used for error detection and correction.

To explore this idea, a common implementation of CBFs where the elements of the set are stored in a slow memory and the CBF is stored in a faster memory is considered. In particular, it is assumed that the elements of the set are stored in DRAM while the CBF is stored in a cache . The reasoning behind this is that the CBF is accessed frequently and needs a fast access time to maximize performance, while the elements of the set are only accessed when elements are read, added or removed and therefore the access time is not an issue. It should also be noted that when the entire element set is stored in a slow memory, no incorrect deletions can occur as they would be detected when removing the element from the slow memory[17]-[19].

Typically, memories are protected with a per word parity bit or with a single bit error correction code . This is based on the observation that most errors affect a single bit or even if they affect multiple bits, the errors can be spread among different words by

the use of interleaving . In addition, soft errors are rare events so that the time between errors is typically large . The arrival rate for terrestrial applications is in the order of at least days or weeks and therefore, it is commonly assumed that errors are isolated. That is, by the time a soft error arrives any previous soft error has been corrected or detected. This is an assumption that is needed, for example, when single bit error correction codes are used.

In the following, one of these two most common protection options is used. In particular, it is assumed that both the DRAMand the cache are protected with a per word parity bit that can detect single errors. As when using single bit error correction codes, it is also assumed that errors are isolated.

The goal for this implementation is to achieve the correction of single bit errors

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

using the CBF. That is, the CBF would enable single bit error correction without incurring in the cost of adding an ECC to the memories.

The first step to achieve error correction is to detect errors. This is done by checking the parity bit when accessing either the DRAM or the cache. To ensure earlier detection of errors, the use of scrubbing to periodically read the memories could be considered . Once an error is detected, a correction procedure is triggered. If the error occurs in the CBF, it can be corrected by clearing the CBF and reconstructing it using the element set. If the error occurs in the element set, the procedure is more complex and can be divided in two phases that are described in the following sections. The idea is that the simpler and faster procedure is used first and only when it is unable to correct the error, the second more complex error correction procedure is used subsequently

A. Simple Procedure for the Correction of Errors in the Element Set

To present the simple correction procedure, let us assume that a single bit error affects element x and that it is detected using the parity bit. Therefore, x_e is read fromthe memory. The correct value x has to be xe if the error affected the parity bit. If the error affected the ith data bit, the correct value will be $x_{em}(i)$ where $x_{em}(i)$ is the value read (x_e) with the ith bit inverted. To determine which of those is in fact the correct value x, the candidates [x_e and all the $x_{em}(i)$] can be tested for membership to the CBF. If only one of the candidates is found in the CBF, then no false positives have occurred and the value found is the correct one. Instead, if more than one candidate is found, the procedure is unable to find the correct value due to the occurrence of false positives. . This simple and fast procedure requires only l + 1 queries to the CBF, where l is the number of bits in each element of the set. However, the correction rate that can be achieved depends on the false positive rate of the CBF. In particular, the probability that an error can be corrected using this procedure can be approximated as

$$P_{\text{correction}} \cong \left(1 - p_{fp}\right)^{l} \quad (4)$$

which is the probability that none of the l candidates that are not x return a false positive on a query. The above formula does not take into account that some elements on the set may only differ in one or two bits from another element in the set. In that case, the proposed correction procedure may fail as one of the candidates may also be a valid element and therefore, the advanced procedure must be used. This effect will be heavily dependent on the properties of the elements in the set and will therefore be application dependent. In any case, to account for it, the probability given by (4) should be used as an upper bound rather than an approximation.

B. Advanced Procedure for the Correction of Errors in the Element Set

The correction process starts by making a copy of the CBF in DRAM memory. Then, all the elements in the set except for the erroneous one are removed fromthe CBF. This will leave a CBF with only the values that correspond to the original value of the element x . Once that is done, the candidates [x_e and all the $x_{em}(i)$] can be queried over the CBF that has only x as an entry. As in the previous procedure, if only one of the

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

candidates matches the CBF, that is the correct value. If more than one candidate matches the CBF then the error cannot be corrected. The probability that a given value x and another value y produce exactly the same values of the hash functions $h_1, h_2, \ldots, h_k$

can be approximated as

$$P_{\mathrm{CBF}(x)=\mathrm{CBF}(y)} \cong \frac{k!}{m^k}.$$

Therefore, the correction probability for this advanced procedure can be approximated as

$$P_{\mathrm{correction}} \cong \left(1 - \frac{k!}{m^k}\right)^l$$

which will be very close to 100% in many practical scenarios as m is typically large.The increased correction rate comes at the cost of a more complex correction procedure that needs the replication of the CBF, the removal of all the entries except the erroneous one (n −1), and finally the query for the l + 1 candidates. However, as soft errors are rare events, and the procedure is only needed when the simple procedure presented before cannot correct an error, the scheme can be useful in real applications[19]-[21].

3 SYSTEM ANALYSIS

Enhancement Block:

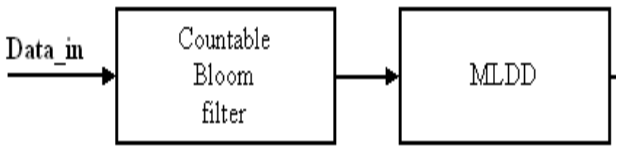

Fig 1.bloom filter

Bloom Filter Algorithm:

3.1 Existing system:

In most cases, BFs are implemented using electronic circuits. The contents of a BF are commonly stored in a high speed memory and required processing is done in a processor or in dedicated circuitry. The set used to construct the BF is also commonly stored in a lower speed memory. The reliability of electronic circuits is becoming a challenge as technology scales. Errors caused by interferences, radiation, and other effects become more common.

Therefore, mitigation techniques are used at different levels to ensure that the circuits continue to operate reliably. For BF implementation, memories are a critical element. For memories, permanent errors and defects are commonly corrected using spare rows and columns. However, soft errors caused for example by radiation can affect any memory cell changing its value during circuit operation. Soft errors do not produce damage to the memory device that continues to operate correctly but has the wrong value in the affected cell.

To deal with soft errors, the use of a per word parity bit or more advanced error correction codes (ECCs). The BFs have also been proposed to mitigate errors in electronic circuits. Use of a CBF is proposed to detect and correct

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

errors in content addressable memories (CAMs). In this case, the CBF is used in parallel with a CAM and the objective is to detect errors in the CAM entries. This is done by checking the results of the CAM and the CBF to ensure that they are consistent.

Once an error is detected, a correction procedure is initiated to restore the correct value in the affected CAM entry using an external copy of its contents.

Disadvantage:

- A problem with BFs is that elements cannot be easily removed.
- Soft errors occur
- lower speed memory

A Bloom filter is a space-efficient data structure used to test whether an element exists in a given set. This algorithm is composed of different hash functions and a long vector of bits. Initially, all bits are set to 0 at the preprocessing stage.

To add an element, the Bloom filter hashes the element by these hash functions and gets positions of its vector. The Bloom filter then sets the bits at these positions to 1. The value of a vector that only contains an element is called the signature of an element. To check the membership of a particular element, the Bloom filter hashes this element by the same hash functions at run time, and it also generates k positions of the vector.

If all of these k bits are set to 1, this query is claimed to be positive, otherwise it is claimed to be negative. The output of the Bloom filter can be a false positive but never a false negative. Therefore, some pattern matching algorithms based on the Bloom filter must operate with an extra exact-matching algorithm.

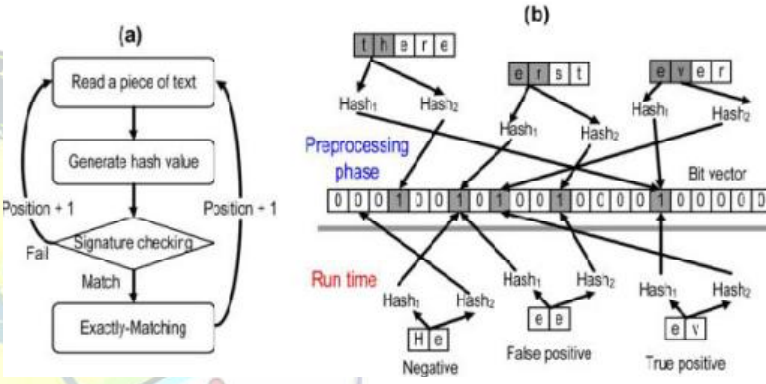

Fig 2. Bloom fliter algorithm

This algorithm fetches the prefix of a pattern from the text and hashes it to generate a signature.

Then, this algorithm checks whether the signature exists in the bit vector.

If the answer is yes, it shifts the search window to the right by one character fXZor each comparison and repeats the above step to filter out safe data until it finds a candidate position and launches exact-matching.

Fig. (b) Shows how a Bloom filter builds its bit vector for a pattern set {erst, ever, there} for two given hash functions. The filter only hashes all of

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

the pattern prefixes at the preprocessing stage. Multiple patterns setting the same position of the bit vector are allowed.

Fig. (c) Shows an example of the matching process. The arrows indicate the candidate positions. The gray bars represent the search window that the Bloom filter actually fetches for comparison. Both the candidate position and search window are aligned together. In step 1, the filter hashes "He" and mismatches the signature with the bit vector. The filter then shifts right 1 character and finds the next candidate position. For the search window "ee", the Bloom filter matches the signature and then causes a false alarm to perform an exact-matching in steps 2 and 3. The filter then returns to the filtering stage and shifts one character to the right in step 4, which launches a true alarm for the pattern "ever".

3.2 MLDD algorithm:

Proposed system:

In this brief, a scheme to exploit existing CBFs to additionally implement error detection and correction in the elements of the set associated with the CBF is presented. The approach is based on the concept of algorithmic-based fault tolerance (ABFT), which proposes to reuse existing properties or elements of the system to implement fault tolerance. In the line of ABFT, the proposed scheme enables a synergetic reuse of existing CBFs for error detection and correction. The scheme assumes that the elements of the set are stored in a memory protected with a per word parity bit and the CBF is used to implement the correction of single bit errors. The effectiveness of the scheme is illustrated using a traffic classification case study. The basic ideas behind the proposed technique can also be applied when the elements of the set are stored in a memory protected with more advanced ECCs.

Advantage :

- Elements can be easily remove
- Low cost
- High speed
- Single error correction

The proposed figure -3 fault-detection method significantly reduces memory access time when there is no error in the data read. The technique uses the majority logic decoder itself to detect failures, which makes the area overhead minimal and keeps the extra power consumption low. The ML detector/decoder (MLDD) has been implemented using the difference-set cyclic codes (DSCCs).

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

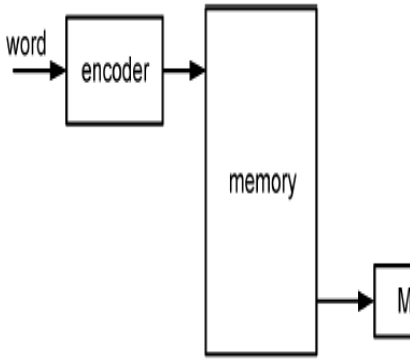

Fig 3. Memory system schematic of an MLDD.

This code is part of the LDPC [**low-density parity-check**]codes, and, based on their attributes, they have the following properties:

- ability to correct large number of errors;
- sparse encoding, decoding and checking circuits synthesizable into simple hardware;
- modular encoder and decoder blocks that allow an efficient hardware implementation;
- systematic code structure for clean partition of information and code bits in the memory.

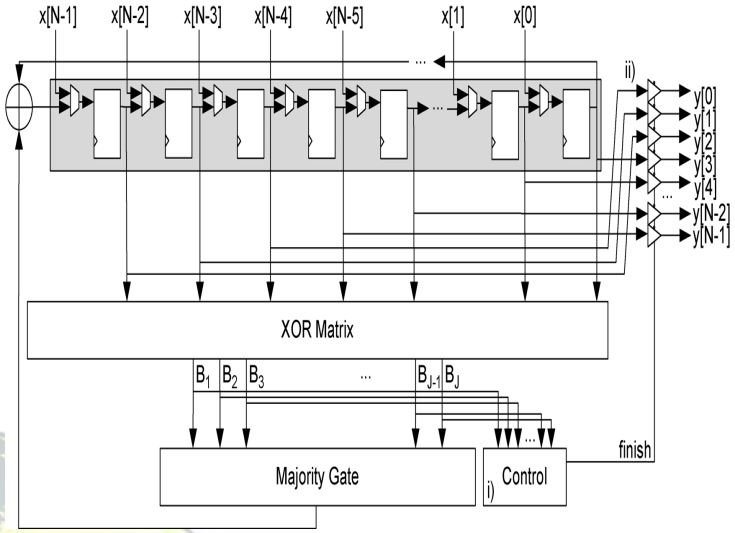

Fig . 4 tap shift register

The figure 4 shows the basic ML decoder with an -tap shift register, an XOR array to calculate the orthogonal parity check sums and a majority gate for deciding if the current bit under decoding needs to be inverted. The nhardware to perform the error detection is illustrated as: i) the control unit which triggers a finish flag when no errors are detected after the third cycle and ii) the output tristate buffers. The output tristate buffers are always in high impedance unless the control unit sends the finish signal so that the current values of the shift register are forwarded to the output.

CONCLUSION:

The proposed approach can also be used for traditional BFs but in that case, the percentage of errors that can be corrected is much lower.To overcome this problem our enhancement process can be include

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

with DMC (Decimal matrix code) process.In this brief, a new application of BFs has been proposed. The idea is to use the BFs in existing applications to also detect and correct errors in their associated element set. In particular, it is shown that CBFs can be used to correct errors in the associated element set. This enables a cost efficient solution to mitigate soft errors in applications which use CBFs.The configuration considered in this brief is that of a memory protected with a per word parity bit for which it is demonstrated that the CBF can be used to achieve single bit error correction. This shows how existing CBFs can be used to achieve error correction in addition to perform their traditional membership checking function.The general idea can also be used when the memory is protected with more advanced codes. For example, if an SEC-DED code is used, the CBF could be used to correct double errors. In addition, the simplest part of the error correction scheme can also be applied to traditional BFs to achieve 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] Christo Ananth, Vivek.T, Selvakumar.S., Sakthi Kannan.S., Sankara Narayanan.D, "Impulse Noise Removal using Improved Particle Swarm Optimization", International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Volume 3, Issue 4, April 2014,pp 366-370

[3] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary, "Jetty: Filtering snoops for reduced energy consumption in SMP servers," in *Proc. Annu.Int. Conf. High-Perform. Comput. Archit.*, Feb. 2001, pp. 85–96.

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

[5] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese,"An improved construction for counting bloom filters," in *Proc. 14thAnnu. ESA*, 2006, pp. 1–12.

[6] M. Mitzenmacher, "Compressed bloom filters," in *Proc. 12th Annu. ACMSymp. PODC*, 2001, pp. 144–150.

[7] 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.

[8] 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.

[9] T. Kocak and I. Kaya, "Low-power bloom filter architecture for deep packet inspection," *IEEE Commun.*

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 22, April 2016

Lett., vol. 10, no. 3, pp. 210–212, Mar. 2006.

[10] 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.

[11] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, *Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances*. New York, NY, USA: Springer-Verlag,2010.

[12] D. Bhavsar, "An algorithm for row-column self-repair of RAMs andits implementation in the alpha 21264," in *Proc. Int. Test Conf.*, 1999,pp. 311–318.

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

[14] 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.

[15] 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.

[16] 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.

[17] A. Reddy and P. Banarjee, "Algorithm-based fault detection for signalprocessing applications," *IEEETrans. Comput.*, vol. 39, no. 10, pp. 1304–1308, Oct. 1990.

[18] D. Guo, Y. Liu, X. Li, and P. Yang, "False negative problem of counting bloom filter," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 651–664, May 2010.

[19] 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.

[20] 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.

[21] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area Web cache sharing protocol," in *Proc. ACM SIGCOMM*, Sep. 1998, pp. 254–265.