International Journal of Electrical and Electronics Engineers
Vol. No.8 Issue 01, January-June 2016
www.arresearchpublication.com
IJEEE
ISSN 2321 - 2055

AN EFFICIENT VLSI BASED ERROR IDENTIFICATION AND CORRECTION OF TEXT DATA SEQUENCE TRANSMISSION

Ms. R. Niranjana Devi[1], Ms. K. Mahalakshmi[2], Ms. R. Gayathri[3]

[1,2,3]M.E-VLSI DESIGN ,Theni Kammavar Sangam College of Technology, (India).

ABSTRACT

The Bloom filter technique is used to create hash function transform in a given test data sets. It is a space-efficient probabilistic data structure. This proposed technique is used for many applications where the amount of source data would require an large hash area in memory if "conventional" error-free hashing techniques were applied. Existing system is to design a conventional bloom filter technique based parallel bloom filter architecture and this architecture is set hash value for every test data sequence. The proposed system is to design a Bloom filter with hamming distance based on fast error detection and correction methodology and this work is to improve the secure text sequence transmission. The proposed system is to identify the error bit location using the redundant bits add process and to correct the error for Ex-or based distance calculation process. The Bloom filter architecture is used to set the hash value for allocated transmitted text data sequence and to improve the fault identification methodology. The proposed system is to increase the text data transmission system and reception system in error checking performance level to improve the secure transmitting process and the system quality level.

Index terms: Bloom filter (BF), Basic logic alignment search tool(BLAST),Counting bloom filter(CBFs), Error correction control(ECC),Xilinx platform studio(XPS).

I. INTRODUCTION

Bloom filter (BF) with *k* hashes, *m* bits in the filter, and *n* elements that have been inserted. A Bloom filter is a space-efficient probabilistic data_structure, that is used to test whether an element is a member of a set. False positive matches are possible, but false_negatives are not, thus a Bloom filter has a 100% recall rate. In other words, a query returns either "possibly in set" or "definitely not in set". Elements can be added to the set, but not removed (though this can be addressed with a "counting" filter). The more elements that are added to the set, the larger the probability of false positives.

A Bloom filter is a data structure designed to tell you, rapidly and memory-efficiently, whether an element is present in a set.

International Journal of Electrical and Electronics Engineers
Vol. No.8 Issue 01, January-June 2016
www.arresearchpublication.com
IJEEE
ISSN 2321 - 2055

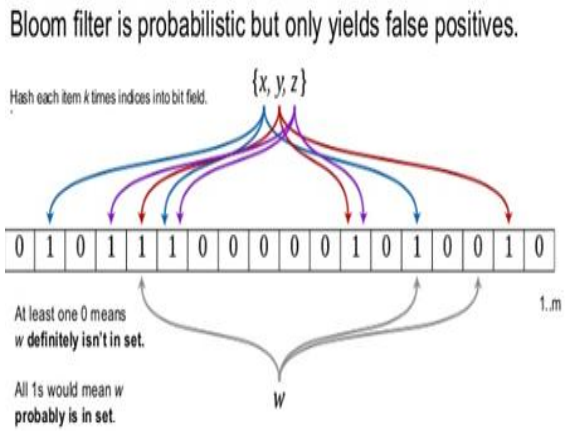

Fig 1.Membership Bloom Filter

A hash function is any function that can be used tomap data of arbitrary size to data of fixed size. The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes. One use is a data structure called a hash table, widely used in computer software for rapid data lookup. Hash functions accelerate table or database lookup by detecting duplicated records in a large file.

The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other.

$D_H = \sum | x_i - y_i |$ (i=1 to k)

$x = y \Rightarrow D = 0$

$x \neq y \Rightarrow D = 1$

Hamming distance equation

Counting filters provide a way to implement a delete operation on a Bloom filter without recreating the filter afresh BF can be considered as counting filters with a bucket size of one bit .CBFs to additionally implement error detection and correction in the elements of the set associated with the CBF is presented.

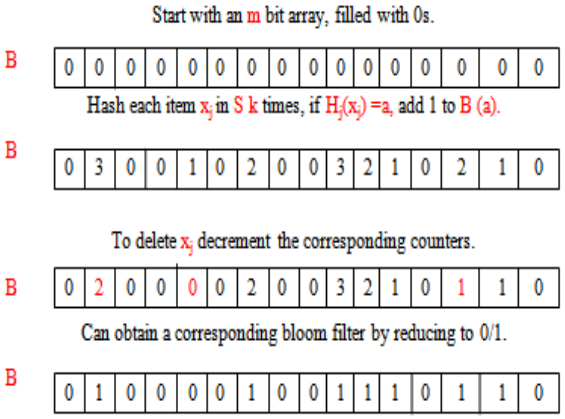

Fig 2.Counting bloom filter process

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. In addition, a simplified version of 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.

II. OVERVIEW OF BFS

Existing system is to design a BLASTN technique based parallel bloom filter architecture and this architecture is set hash value for every test data sequence. The existing system is used to identify the fault test data sequence transmission and reception based on hash function mismatch operation. The existing system is to improve the error detection calculation between the data base expected sequence and received transmission sequence.

The existing BLASTN, a streaming implementation of BLASTN, the BLAST family algorithm for text data sequence comparison. Unlike most previous streaming accelerators for bio sequence comparison, Mercury BLASTN uses FPGA hardware, specifically the Mercury platform for high-speed stream processing, to accelerate the performance-critical early stages of seeded alignment.

It relies on the widely used BLASTN software implementation produced by the National Centre for Biological Information (NCBI) for later non- bottleneck stages . On large text sequence comparisons, Mercury BLASTN on previous generation FPGA hardware runs 5 to 11 times faster than NCBI BLASTN software on a current generation general-purpose CPU, while delivering results 99% identical to those from the software. On FPGA hardware available today, we anticipate a further eight-fold speedup.

To create specialized processors, the existing systems turn to field- programmable gate arrays, which are a fast, inexpensive option for implementing specialized architectures. A computing architecture can be specified using standard hardware description languages, then compiled to run on a particular family of FPGAs, without the need for expensive circuit fabrication. Moreover, a single FPGA, like a general-purpose CPU, can be reused for different computations by simply reprogramming it with new architectures.

The existing system is to show how to modify the bottleneck stages of the BLASTN algorithm to exploit the power of an FPGA, while producing results substantially identical to those of the standard NCBI BLASTN software. In particular, the system shown how to speed up the critical seed matching stage, which does not use dynamic programming and so is not helped by techniques previously used to accelerate Smith-Waterman. The system proposed how to re -implement BLASTN's own ungapped extension algorithm as an efficient hardware filter that prevents most fruitless extensions from reaching the software

Existing system is used to detect the error sequence, but doesn't correct the error in fault sequence.The existing system is to improve the system performance level. The existing system is to consume more time for error identification process.

III. PROPOSED SCHEME

The proposed is to design a Bloom filter with hamming distance based fast error detection and correction methodology and this work is to improve the secure text sequence transmission.

The proposed system is to identify the error bit location using the redundant bits add process and to correct the error for xor based distance calculation process.

The proposed Bloom filter architecture is used to set the hash value for allocated transmitted text data sequence and to improve the fault identification methodology.

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

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.

The error detection process is to check the bloom hash sequence value and to detect the error in single bit values This process is to add the redundant bits in different bit position in overall transmission bit sequence, Then to send transmission bit sequence and to get the receiver side and to check the parity value.

Error correction to check the parity values to the receiver sequence level and to identify the error accrued bit position. Then to invert the error bit and to correct the overall received bit sequence. This process is mainly used to secured error detection and error correction in received text sample sequence.

The text sample sequence is to check whether the sequence is correct or wrong format presentation. So we apply the bloom filter function and to matching the selected text sequence. Finally we simulate and synthesis result about our proposed error detection and correction architecture.

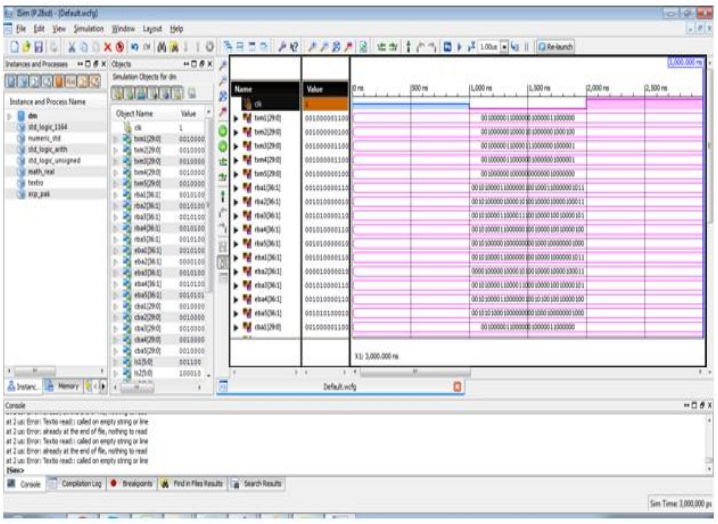

Fig 3.Simulation result of error identification

International Journal of Electrical and Electronics Engineers
Vol. No.8 Issue 01, January-June 2016
www.arresearchpublication.com
IJEEE
ISSN 2321 - 2055

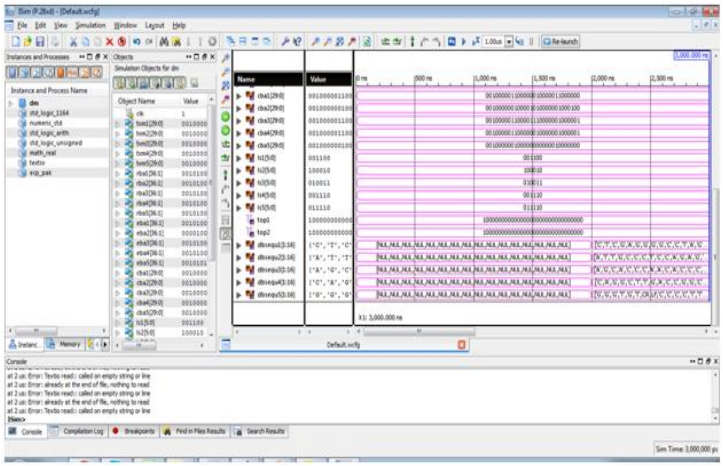

Fig 4.Simulation result of error correction

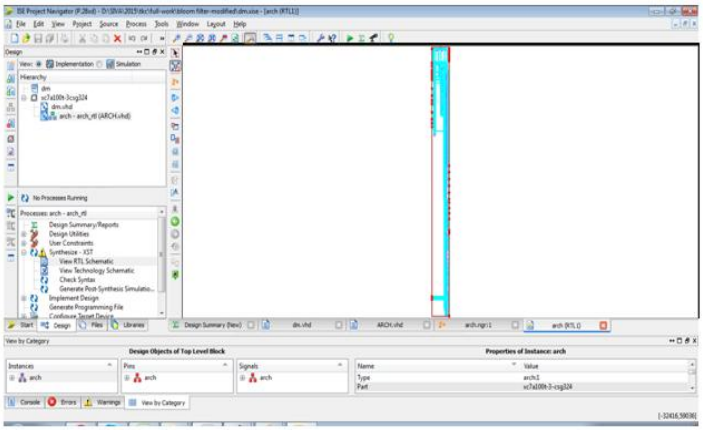

Fig 5. RTL schematic structure

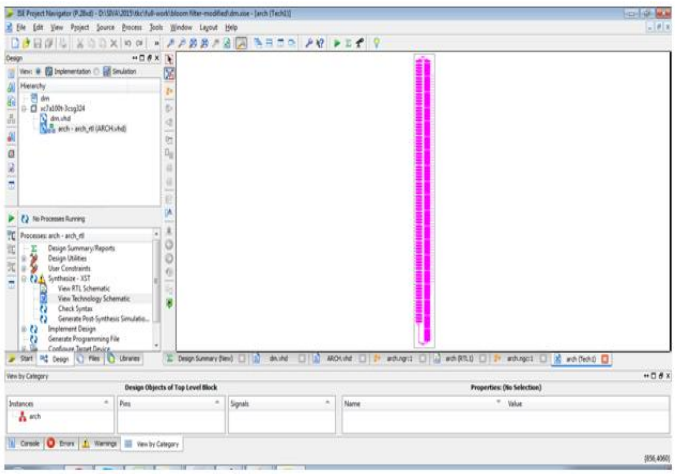

Fig 6.Technology schematic structure

International Journal of Electrical and Electronics Engineers
Vol. No.8 Issue 01, January-June 2016
www.arresearchpublication.com
IJEEE
ISSN 2321 - 2055

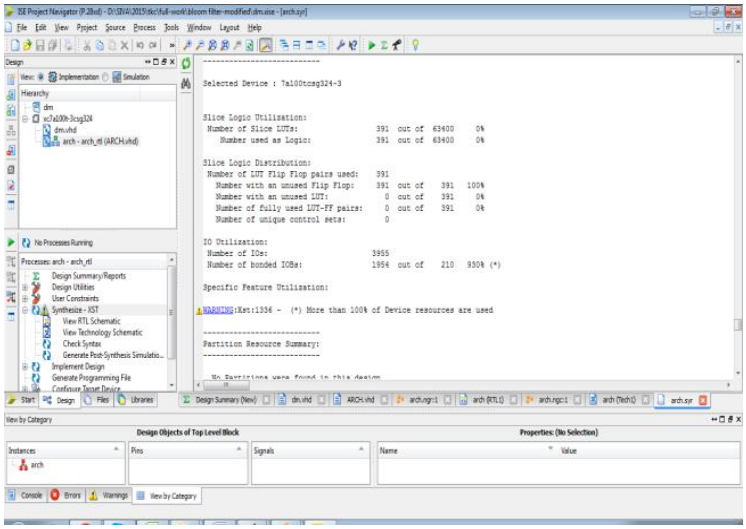

Fig 7.Synthesis report for bloom filter technique

Table1. Comparison result

Parameters	Existing System	Proposed system
SLICES COUNT	1116	391
LUT COUNT	1931	391
FLIP FLOP COUNT	605	391
DELAY TIME(ns)	15	3.933

IV. CONCLUSION

In this work a new application of bloom filter has proposed. This is used to detect and correct errors in the associated element set. The proposed scheme is based on the observation that a Bloom filter, 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.

The reasoning behind this is that the Bloom filter 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. 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. Finally we design a bloom filter with fast error detection and correction architecture and to

improve the error identification process for given selected text data sequence. The basic ideas behind the proposed technique can applied when the elements of the set are stored in a memory protected with more advanced ECCs.

In addition, a simplified version of 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. The exploration of the scheme extension to these cases is left for future work.

REFERENCES

[1] Andrei Broder and Michael Mitzenmacher, (2004), "Network Applications of Bloom Filters: A Survey", Internet mathematics Vol 1,NO. 4.

[2] Christian Esteve Rothenberg, Carlos A. B, (2010), "The delectable bloom filter: a new member of the bloom family",arXiv:1005.0352v1 [cs.DS].

[3] F Chang , J dean , S. Ghemawat ,WC Hsieh, (2014) "Bigtable: A distributed storage system for structured data" to appear in OSDI 2006.

[4] Haoyu Song, Sarang Dharmapurikar, (2005), "Fast Hash Table Lookup Using Extended Bloom Filter:An Aid to Network Processing", SIGSCOMM'05,ACM 1-59593-009-4/05/0008.

[5] Ion Stoica , Robert Morris , David Karger, (2001) , "Chord : a scalable peer-to-peer lookup service for internet applications " , SIGCOMM'01,August 27-31.

[6] Michael Mitzenmacher, (2002), "Compressed Bloom Filters" IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 10, NO. 5.

[7] Michael Paynter and Taskin Kocak, (2008), "Fully Pipelined Bloom Filter Architecture", IEEE COMMUNICATIONS LETTERS,VOL. 12,NO. 11.

[8] Sylvia Ratnasamy,(2001) "A scalable content-addressable network",SIGCOMM'01,august 27-31.

[9] Taskin Kocak and Ilhan Kaya,(2006) "Low-Power Bloom Filter Architecture for Deep Packet Inspection" IEEE COMMUNICATIONS LETTERS,VOL. 10,NO. 3.

[10] Yi Lu, Balaji Prabhakar, 2005," Bloom Filters: Design Innovations and Novel Applications".