**Original Research Article** **Bioinformatics**

**International Journal of Pharma and Bio Sciences**

**ISSN 0975-6299**

# A REAL-TIME SEARCHING AND SEQUENCING ASSEMBLY PLATFORM BASED ON AN FPGA IMPLEMENTATION FOR BIOINFORMATICS APPLICATIONS

***A.SURENDAR, AND ASHLINE GEORGE**

**[1]Assistant Professor, School of Electronics, Vignan's University, Guntur, India*

## ABSTRACT

DNA database is increasing day by day. The DNA sequence comparison is used for classification of species, finding heredity and disease detection. This similarity search for identifying similarities between subsequences of strings is called as sequence alignment. Gene sequence alignment is a tough task in bioinformatics due to huge data and high searching time. Researchers have used various bioinformatics algorithms for sequencing, but it takes an enormous amount of time for computations. The bloom filter which is commonly used for searching in Intrusion Detection System (IDS) is proposed with FPGA to overcome the space and time complexity problems in bioinformatics FPGAs have high-performance computing accelerators and their reprogramming ability allows different algorithms to be implemented using the same hardware. For realizing the biological mysteries to meet the necessary requirements such as time, memory and speed, an efficient searching method has been used and compared with FPGA.

**KEY WORDS**: ***Bioinformatics; Gene Sequence Alignment; Search Algorithms; Bloom Filter;*** **FPGA.**

**A.SURENDAR**
**Assistant Professor, School of Electronics, Vignan's University, Guntur, India**

***Corresponding Author**

## INTRODUCTION

In the world of growing set of biological species finding repetitive structures in genomes and Proteins are used to realize their biological functions and classification of species. DNA sequencing is the process of determining the precise order of nucleotides such as Adenine (A), Guanine (G), Cytosine (C) and Thymine (T) within the DNA molecule. It includes any method or algorithm that is used to determine the order of the four bases A, G, C, and T in a strand of DNA. The introduction of fast DNA sequencing methods has greatly enhanced biological and medical research and discovery. Sequence alignment is the operation trade with alignment of two or more sequences based on their fundamental elements.[2] Sequence alignment is used to study the evolution of the sequences from a common ancestor, such as protein sequences or DNA sequences. It is not very easy to compare the sequences of several hundred nucleotides and amino acids alignments by hand. There is a need of an algorithm which is also efficient. For an algorithm to work efficiently, a hardware implementation should be plotted into a platform which appreciates its inherent complexity and utilizes it completely. Biological algorithms have been often encountered with processes which require significant parallel architecture. Field Programmable Gate Arrays are well known for their efficiency and flexibility and plays a major role in completing the reliability.The Aho-Corasick algorithm adopted for execution in field programmable gate arrays (FPGA) in a manner that optimize space and performance [1]The traditional Aho-Corasick finite state machine (FSM) is split into smaller FSMs, operating in parallel,[2] each of which matches up to 20 peptides in the input translated genome. Each of the smaller FSMs further divided into five simpler FSMs such that each simple FSM operates on a single bit position in the input. The BFAST algorithm has proven to be an accurate, commonly-used software alignment program. BFAST [3] addresses the performance drawbacks of sequence alignment, dynamic programming algorithms by using a premade index to quickly identify candidate alignment locations (CALs). CALs are locations in the reference where the read is most likely to match.Aho Corasick is the Multi pattern matching algorithm which locates all the occurrence of set of patterns in a text of string. It first creates deterministic finite automata for all the predefined patterns and then by using automaton, it processes a text in a single pass. It Consists of constructing a finite state pattern matching automata from the patterns and then using the pattern matching automata to process the text string in a single pass[4] At a high level, the KMP algorithm is similar to the naive algorithm: it considers shifts in order from 1 to n−m, and determines if the pattern matches at that shift.[5] The difference is that the KMP algorithm uses information gleaned from partial matches of the pattern and text to skip over shifts that are guaranteed not to result in a match. Suppose that, starting with the pattern aligned underneath the text at the leftmost end, repeatedly “slide” the pattern to the right and attempt to match it with the text. A content-addressable memory (CAM) is a high speed matching unit because it has parallel matching capability. It speeds up the data searching and pattern matching[4]CAMs are storage devices that allow its contents to be accessible on the basis of a match between a specified key and the contents, a process called "content addressing".[6] CAM architectures fall between two extremes: the bit serial CAM and the fully parallel CAM. In the bit serial CAM, the matching logic is associated with one bit position, and shared among all the bits in a word, in effect matching one bit at-a-time simultaneously in all the CAM words. In the fully parallel CAM, each word has its own bit-parallel matching logic, allowing that match of all words to process.The global and semi-global Needleman-Wunsch, and Smith-Waterman algorithms with a backtracking procedure which is needed to construct the alignment.[16] The backtracking procedure of the sequence alignment algorithms may be designed to fit in with the GPU architecture. Algorithm is able to compute pairwise alignments. Moreover, the speed of their GPU-based algorithms can be almost linearly increased when using more than one graphics card.[7] The Burrows-Wheeler Transform and wavelet trees used for finding all the maximal repeats from massive texts in a time- and space- efficient manner.[8] For data sets the space usage is no more than three times the text size. For genomic sequences stored using one byte per base, the space usage is less than double the sequence size. This space-efficient method keeps the timing performance fast such that it find all the maximal repeats in the whole human genome in less than 17 hours.[9]A new seeding method called a spaced k-mer neighbor which provides a better tradeoff between the sensitivity and speed in protein sequence similarity search. With the method of spaced k-mer neighbors, for each spaced k-mer, a set of spaced k-mer is selected as its neighbors.[10] These pre-selected spaced k-mer neighbors are then used to detect hits between query sequence and database sequences. Bloom filters technique used where long linear data files[11] (i.e., flat files) contain multiple signatures that are to be found using a multiplicity of processors (parallel processor).[12,13] This used to find signatures in files residing in the nodes of parallel processors configured as trees, two dimensional meshes and hyper cubes[14]. Elegant expressions are found for average signature searching time and speedup, and graphical results are provided.

## PROBLEM DESCRIPTION

Most of the search algorithms use dynamic programming for solving the problem of optimization. Use of FPGA for highly computation time demanding algorithms have been found to be highly effective in order to find or match a particular sequence of biological information to a huge database of genetic information of millions of species. Hence, a hardware implementation of Bloom filter can do string matching at high speeds. Bloom filters use less memory space to store the compressed strings. The amount of memory depends on the number of strings being compressed and typically is few megabits.

## TECHNIQUE USED

A Bloom filter offers a perfect choice for string matching. It is a randomized technique to test membership of a string in a group of given strings. Using this technique, a

group of strings is compressed by calculating multiple hash functions carried over each string. Then, compressed set of strings is stored using fixed size of memory. This set can be enquired to find out if a given string belongs to it. The two important properties of a Bloom filter that make it a practicable solution for string matching are the following: Scalability: Bloom filter uses a fixed amount of memory to compress each string unrelatedly to the length of the original string. Thus, large strings can be easily stored with smaller memory space. This makes it highly scalable in terms of efficient memory space. Speed: The amount of computation involved in detecting a string using Bloom filter is constant. This computation is a calculation of hash functions and the corresponding memory lookups. Efficient hash functions can be implemented in hardware easily with little resource consumption. The block diagram of bloom filter is shown in figure 1.Given a string x, the Bloom filter computes k hash functions on it producing hash values ranging from 1 to m. It then sets k bits in a m bit long vector at the addresses corresponding to the k hash values. The same procedure is repeated for all the strings of the set. This process is called “programming” of the filter. The query process is similar to programming, where a string whose membership to be queried is given as input to the filter. The Bloom filter computes k hash values using the same hash functions which are used to program the filter. The bits in the m bit long vector at the locations corresponding to the k hash values are looked up. If at least one of these k bits is found not set then the string is informed to be a non-member of the set. If all the bits are found to be set then the string is said to belong to the set with a certain probability. This uncertainty in the membership is called as false positive comes from the fact that those k bits in the m bit vector can be set by any of the n members. The false positive rate increases with increasing members in the set. For better optimum result, the false positive rate is kept to be less. It can be calculated by using probability theory [8][14]. The probability that a random bit of the m-bit vector is set to 1 by a hash function is simply $\left(\frac{1}{m}\right)$. The probability that it is not set and set to 0 is 1-$\left(\frac{1}{m}\right)$. The probability that it is not set by any of the n members of x is $\left(1-\left(\frac{1}{m}\right)\right)^{n}$, since each of the bit-strings sets k bits in the vector, the probability becomes $\left(1-\left(\frac{1}{m}\right)\right)^{nk}$. The probability that this bit is 1 becomes 1-$\left(1-\left(\frac{1}{m}\right)\right)^{nk}$[13]. For a bit of string to be detected as a possible member of the set, all k bit locations generated by the hash function s need to be 1. The probability (f) that this happens is given by Equation (1).

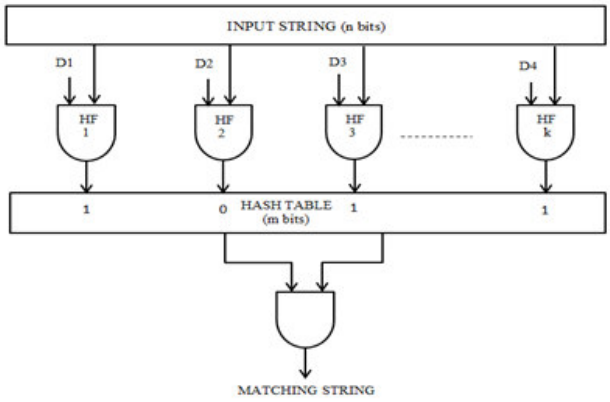



**Figure 1**
***Functional diagram of bloom filter.***

$$f = \left[1 - \left[1 - \left(\frac{1}{m}\right)\right]^{nk}\right]^{k} \quad (1)$$

Where n-number of members in the set, m-hash vector range from 0 to m, k-number of hash functions, f- false positive probability rate. For the large values of m the above equation reduces to Equation (2).

$$f \approx \left[1 - e^{\frac{-nk}{m}}\right]^{k} \quad (2)$$

This probability is independent of the input bit-string and is termed the false positive probability. The false positive probability can be reduced by choosing suitable values for m and k for a given size of the member set, n. In the optimal case, when false positive probability is minimized with respect to k, the following relationship is obtained and given by equation (3).

$$k = \frac{m}{n} \ln 2 \quad (3)$$

The false positive probability at this optimal point is given by Equation (4).

$$f = \left(\frac{1}{2}\right)^k \qquad (4)$$

It should be noted that if the false positive probability is to be fixed, then the size of the filter, m, needs to scale linearly with the size of the bit-string set, n. In the optimally configured Bloom filter, the probability of finding a bit set is 0.5.

**Table.1**
***Results of bloom filter***

| HASH TABLE | | MATCHING STRING |
|---|---|---|
| 0 | 0 | NEGATIVE |
| 0 | 1 | NEGATIVE |
| 1 | 0 | NEGATIVE |
| 1 | 1 | POSITIVE |
| 1 | 1 | FALSE POSITIVE |

The pseudo-code for bloom filter is given in the table 2.

**Table 2**
***Pseudo-code for bloom filter***

i. Clear hash table with zeros
ii. Insert input(s)
iii. Do hash $h_k(s) \leftarrow 1$
iv. Insert query(q)
v. Do hash $h_k(q) \leftarrow 1$
vi. If ($h_k(s)$ AND $h_k(q)=0$)return negative
vii. Else check positive or false positive

Querying the Bloom filter for set membership of a given bit-string, k hash values are created using the same hash functions used to program the filter. The bits in the m-bit vector at the locations corresponding to the k hash values are checked. If at least one of the k bits is 0, then the bit-string is declared to be a non-member of the set. If all the bits are found to be 1, then the bit-string is said to belong to the set with a certain probability. If all the k bits are found to be set and x is not a member, then it is said to be a false positive. The results of the bloom filter in the process of searching a string is given in the following table 1.

## SIMULATION DESCRIPTION

The results are obtained using ModelSim 6.5e. The ModelSim has in-built capabilities for Verilog, VHDL, and System C for ASIC and FPGA design. ModelSim is the process of finding design defects by debugging and simulation. The corrected VHDL code is used to implement the bloom filter in Altium Nanoboard 3000-Xilinx Spartan kit.Here the biological data is DNA sequences taken from NCBI (National Center for Biotechnology information) gene bank. Here baker's yeast sequences are taken. This is given as input to the Bloom Filter. The Bloom Filter has 2 hash functions. Each hash functions applied on each set of input strings to produce fixed length of 6 bit malicious strings. Initially the hash table is filled with zeros. The hash values of all malicious strings are stored in 16 bit hash table. So here the large size of input data is stored in fixed size of hash table. The 6 bit query sequence is entered to the bloom filter. The same operation is done on query strings by hash functions. The hash values of query strings are compared with previous hash table by simple AND operation. This comparison gives three types of results as positive, negative & false positive. The bloom filter algorithm is developed using VHDL language for implementation in reconfigurable platform. The same input is given to various search algorithms that written in VHDL code. The synthesis report are generated for each algorithms and the corresponding parameters are compared to choose the optimum search method.

## SIMULATION RESULTS AND DISCUSSION

The bioinformatics search algorithms and Bloom filter algorithm are implemented in reconfigurable platform of Xilinx Spartan 3E kit. Those performances are compared by the following parameters as real time, CPU time and total memory used by the algorithms.The figure 2 shows the comparison of following parameters number of input/output, cell usage, FF/Latches, IO buffers obtained by Xilinx Spartan 3E kit from the synthesis report. The figure 3 shows the result of time comparison between various search algorithms. The bloom filter takes less number of IOs and cell usage than the Aho corasick algorithm. This results Bloom filter has very less execution time and the needed fast response. Similarly the figure 4 shows the memory space comparison and clearly reports the Bloom filter is best for sequence searching.

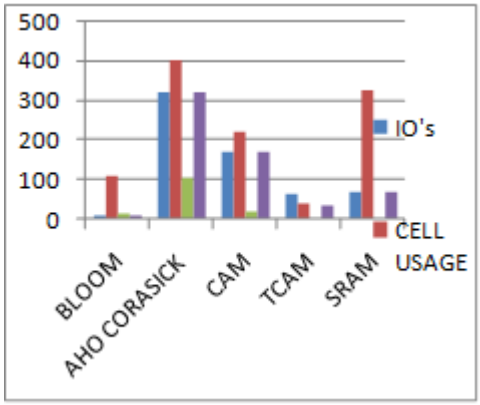
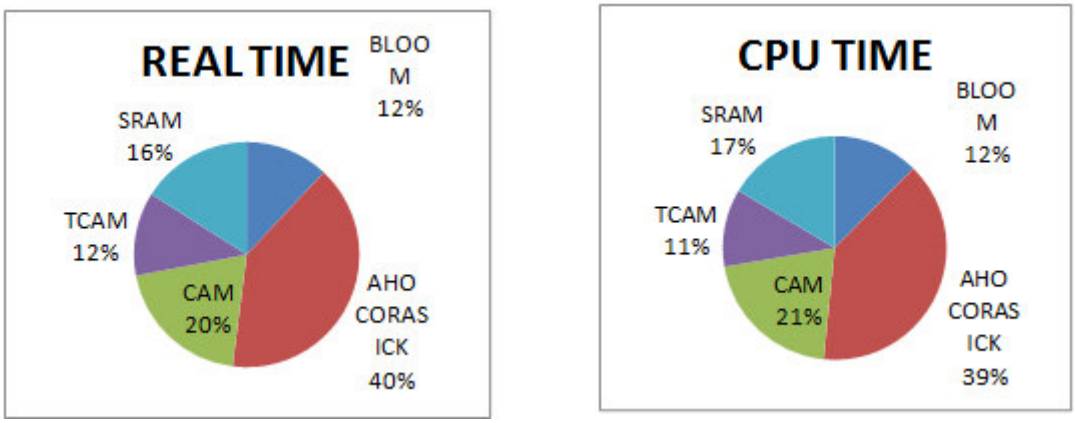



**Figure 2**
***Various parameters compared among different search algorithms.***




**(a)** **(b)**

**Figure 3**
***(a) and (b).Time comparison interms of real time and CPU time taken by different search algorithms.***

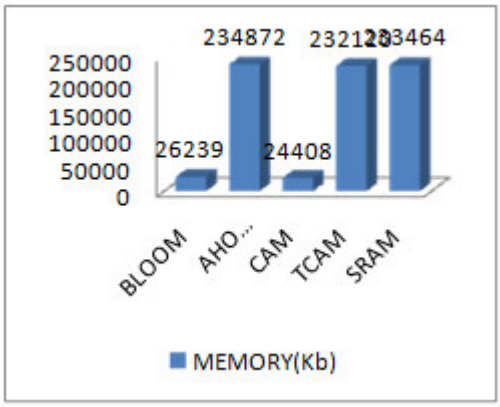



**Figure 4**
***Total memory occupied for executing the different search algorithms***

## CONCLUSION

A scalable and fast solution is needed to accommodate the largest bioinformatics data today and to sustain the real time processing. Software based algorithms are not scalable to high speeds. But the bloom filter is scalable and feasible with FPGA platforms. In this paper bloom filter is efficiently used for sequence alignment and searching in bioinformatics. Our proposed technique produces discovering of similar sequences and queue of similar sequences for searching process. The results are also shows that the bloom filter technique can improve the quality, memory and speedup metrics.

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## REFERENCES


1. Pagiamtzis K, Sheikholeslami A. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. IEEE Journal of Solid-State Circuits. 2006 Mar;41(3):712-27.
2. Beebe N, Dietrich G. A new process model for text string searching. InIFIP International Conference on Digital Forensics 2007 Jan 28 (pp. 179-191). Springer New York.
3. Aygün RS. S2S: structural-to-syntactic matching similar documents. Knowledge and information systems. 2008 Sep 1;16(3):303-29.
4. Surendar A, Arun M, Bagavathi C. Evolution of Reconfigurable Based Algorithms for Bioinformatics Applications: An Investigation. Int. J. Life Sci. Bt & Pharm. Res. 2013 Oct;2(4):17-27.
5. Cao P, Wu S. Parallel research on KMP algorithm. In2011 International Conference on Consumer Electronics, Communications and Networks (CECNet) 2011 Apr 16.
6. Rognes T. Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation. BMC bioinformatics. 2011 Jun 1;12(1):1.
7. Surendar A, Arun M, Bagavathi C. Evolution of Reconfigurable Based Algorithms for Bioinformatics Applications: An Investigation. Int. J. Life Sci. Bt & Pharm. Res. 2013 Oct;2(4):17-27.
8. Hao F, Kodialam M, Lakshman TV, Song H. Fast dynamic multiple-set membership testing using combinatorial bloom filters. IEEE/ACM Transactions on Networking (TON). 2012 Feb 1;20(1):295-304.
9. Kulekci MO, Vitter JS, Xu B. Efficient maximal repeat finding using the burrows-wheeler transform and wavelet tree. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB). 2012 Mar 1;9(2):421-9.
10. A. Surendar, Arun M, A.M. Basha, “Micro Sequence Identification of Bioinformatics Data Using Pattern Mining Techniques in FPGA Hardware Implementation”, Asian Journal of Information Technology, 2016, Mar 16(1):.76-81
11. Hasib S, Motwani M, Saxena A. Importance of Aho-Corasick String Matching Algorithm in Real World Applications. international journal of computer science and information technologies. 2013;4(3):467-9.
12. Surendar A, Arun M, Kavitha M. A reconfigurable approach for Dnasequencing and Searching methods. Asian Journal of Research in Social Sciences and Humanities. 2016;6(4):316-29.
13. Jiang P, Ji Y, Wang X, Zhu J, Cheng Y. Design of a multiple bloom filter for distributed navigation routing. IEEE Transactions On Systems, Man, And Cybernetics: Systems. 2014 Feb;44(2):254-60.
14. Li W, Ma B, Zhang K. Optimizing spaced k-mer neighbors for efficient filtration in protein similarity search. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB). 2014 Mar 1;11(2):398-406.