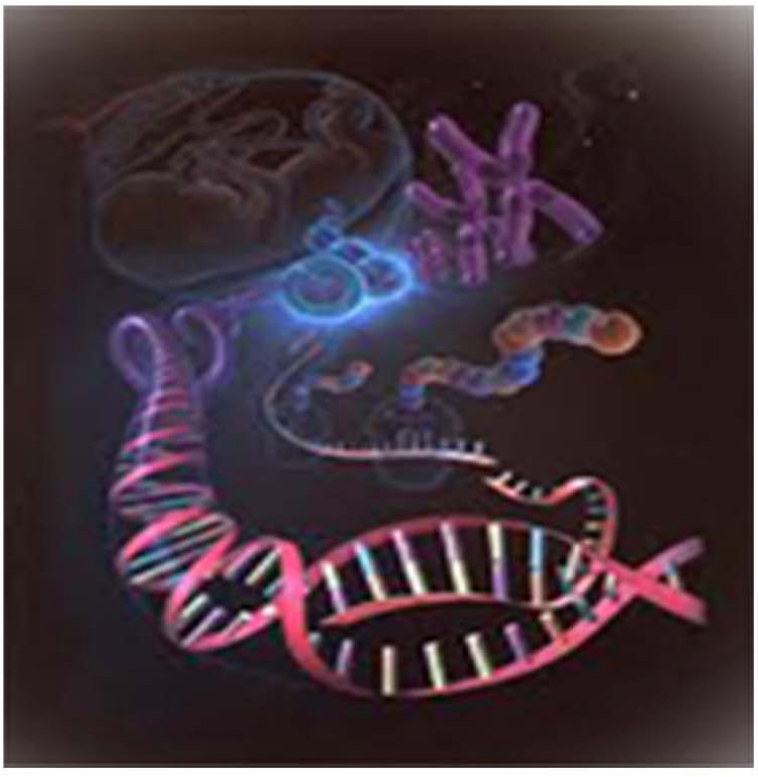
IJLBPR

ISSN 2250 - 3137
Vo1. 2, No. 4, October 2013

International Journal of

Life Sciences Biotechnology and Pharma Research



International Journal of Life Sciences Biotechnology and Pharma Research
Hyderabad, INDIA

www.ijlbpr.com

Email : editorijlbpr@gmail.com or editor@ijlbpr.com

International Journal of Life Sciences Biotechnology and Pharma Research

ISSN 2250-3137 www.ijlbpr.com
Vol. 2, No. 4, October 2013

Review Article

EVOLUTION OF RECONFIGURABLE BASED ALGORITHMS FOR BIOINFORMATICS APPLICATIONS: AN INVESTIGATION

A Surendar[1], M Arun[2] and C Bagavathi[3]

Corresponding Author: **A Surendar ✉ surendararavindhan@gmail.com*

In the world of expanding constellation of biological species for various technological and biological reasons, it requires high efficiency and memory to record the genetic database of all the species so far found. If a species is found and it is established that it was not classified before, it is essential for the researchers to find out the family of species where it had been evolved from. Thus researchers use various bioinformatics algorithms which deal with varying operations on biological data such as DNA analysis, Protein analysis, Genome information retrieval and Nucleotide applications. It is mandatory that the process of working on a DNA of a new species should be accurate, fast and energy efficient. It should ensure the correctness of the solution. The concept of classifying the biological information follows the biological evolution procedure. The evolution of bioinformatics algorithms started from Needleman-Wunsch algorithm in 1970 and still it is continuing towards better configuration and performance. For an algorithm to work efficiently, a hardware implementation should be mapped into a platform which appreciates its inherent complexity and utilizes it completely. Biological algorithms have been often encountered with processes which require substantial parallel architecture. The best candidate would be Field Programmable Gate Arrays (FPGA) for the process of solving the biological mysteries satisfying the necessary requirements. In this paper, prominent algorithms for bioinformatics applications are studied and their FPGA implementation challenges and opportunities are explored.

Keywords: Bioinformatics, Pattern matching, Sequence alignment, FPGA implementation

INTRODUCTION

DNA is the indispensable part of species' identity which distinguishes from other species. Understanding DNA is a very basic, imperative problem in Bioinformatics as it is often used to identify evolutionary relationships among the

[1] Faculty of Electrical Engineering, Anna University, Chennai-600025, India.
[2] School of Electronics Engineering, VIT University, Vellore-632014, India.
[3] Faculty of ECE, K.S.R College of Engineering, Tiruchengode-637215, India.

organisms and predict secondary or tertiary structure. They are various problems considering DNA analysis such as DNA Computation, DNA Detection, DNA Extraction, DNA Fragmentation, DNA Genetics, DNA Isolation, DNA Sequencing, and Protein Analysis such as Protein Sequence Alignment, Protein Identification, Structure Analysis, Enzyme Mapping and Nucleotide applications like Information Retrieval and Sequence Matching and Identification. Sequence alignment is the operation dealing with alignment of two or more sequences based on their constituent elements. Pairwise sequence alignment is used to uncover homologues of a gene from a database of known patterns. Sequence alignment is used to study the evolution of the sequences from a common ancestor such as protein sequences or DNA sequences. The mismatches in the alignment correspond to mutations, and gaps correspond to insertions or deletions. Sequence alignment refers to the process of putting together significant alignments in a database of potentially unrelated sequences.

Given the large DNA sequences that some researchers wish to study, certain parameters should be considered before choosing a bioinformatics sequence alignment algorithm. The space and time complexity, hardware utilization efficiency, reusability of design, reliability on the obtained result and power consumption becomes increasingly important. Most of the algorithms use dynamic programming for solving the problem of optimization. Field Programmable Gate Arrays are well known for their efficiency and flexibility and plays a major role in finalizing the reliability and characteristics of a design before its final tape-out. Use of FPGA for highly computation time demanding algorithms. Christopher Ma *et al.* (2012) have been found to be highly effective. Any bioinformatics algorithm falls under the category of heavy processor demanding algorithm as the algorithm has to find or match a particular sequence of biological information to a huge database of genetic information of millions of species. Such a process can be achieved by employing FPGA and analysis would help the designer to modify the algorithm and improve the quality. Any alterations on the algorithm such as increase in speed, reduction in complexity, and increase in the accuracy of the search process can be done through FPGA configurations (Grigorios Chrysos *et al.,* 2012). It would also give the specifications of hardware to be implemented. In this paper, we have made it an effort to deliver information concerning the hardware implementation possibilities of prominent algorithms and architectures for bioinformatics applications.

Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm performs a global alignment on two sequences (called A and B here). It is commonly used in bioinformatics to align protein or nucleotide sequences. The algorithm was published in 1970 by Saul B Needleman and Christian D Wunsch

The Needleman-Wunsch (NW) algorithm performs an optimal global alignment of two sequences based on certain constraints. The algorithm aligns the sequences based on maximum number of matches in amino acid and minimum number of gaps required to align the sequences. The algorithm was designed by Saul B Needleman and Christian D Wunsch. The Needleman-Wunsch algorithm is the first application of dynamic programming in biological sequence comparison. It is sometimes referred to as the Optimal Matching algorithm, because

the Needleman-Wunsch algorithm finds the optimal alignment of the entire sequence of both proteins, it is a global alignment technique. Consider two strings of gene characteristics *s* and *t* where *s* is ATTGCTCTG and *t* is ATGCCG. In these sequences of varying lengths as given in Table 1, it can be found by introducing few gaps; the maximum alignment of the sequences can be maximized. The score of alignment of two elements is 1. If there is a mismatch or gap in the resultant of the algorithm, the score is –1. The cost of entire alignment would be the sum of all scores for the alignment. For this example, the cost of the entire alignment is 3.

Table 1: Process of Score Calculation in NW Algorithm

S	A	T	T	G	C	T	C	T	G
t	A	T	-	G	C	-	C	-	G
Score	1	1	-1	1	1	-1	1	-1	1

The optimal alignment would maximize the cost of entire alignment. This would require a procedure to align the sequences in all possible combination and the cost of alignment should be ensured to be the highest. The FPGA used in Sotiriades *et al.* (2006) for implementing Needleman-Wuncsh algorithm is a Xilinx Virtex-II Pro, which is based on a 130 nm process, clocked at 100 MHz. It had employed IP cores for floating point operations. The issues in this implementation were related to precision of floating point operations in FPGA, require significant die-area and needs deep pipelining to get acceptable performance. Compared to GPU and FPGA utilized in Sotiriades *et al.* (2006), it has been found that in implementing Needleman-Wuncsh algorithm, FPGA requires lowest overhead in accessing memory compared to GPU.

Smith-Waterman Algorithm

The algorithm was first proposed by Temple F Smith and Michael S Waterman in 1981. The Smith-Waterman algorithm performs local sequence alignment; that is, for determining similar regions between two strings or nucleotide or protein sequences. Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure

The Smith-Waterman algorithm, dynamic programming algorithm, is for determining analogous regions between two nucleotide or protein sequences. It is a non-heuristic algorithm that guarantees to find the optimal local alignment with respect to the scoring system being used in Needleman Wunsch Algorithm. Smith-Waterman algorithm reduces the number of counterfeit positives. However, the Smith-Waterman algorithm produces optimal results by demanding of time and memory resources and by sacrificing speed. To confirm a solution to be accurate and optimal, Smith-Waterman is required to be more computations costing its reputation to taken the position of a fast algorithm. As a result, it has been reinstated in by its successor, BLAST algorithm with an option of approximate solution.

Smith-Waterman algorithm had been known for its well-known memory requirement. By using FPGA, the parallel nature of the algorithm can be exploited to increase the speed of the algorithm. Without the pipelining, the direct implementation of the algorithm resulted in 40 MHz frequency. The optimal alignment score can be computed in FPGA through parallel processing. Xilinx Vertex 2 Pro series (Euripides Sotiriades *et al.,* 2005) has dual port Block RAMs which can be utilized for parallel reading and writing of data in two

channels simultaneously. It can provide excellent memory throughput. The design was successfully verified in Memec Design Spartan II FPGA board.

When there is a procedure to be repeated on different data sets at different scheduled time instants, the systolic architecture would be the best choice of design methodology. Systolic arrays in FPGA have also been implemented for Smith-Waterman algorithm. Processing Elements for the analysis DNA and Protein sequences based on XD1000 reconfigurable supercomputing platform have been utilized in Timothy Oliver *et al.* (2005). The use of systolic elements in FPGA achieved a speedup of 250 times compared to direct implementation of the algorithm.

BLAST Algorithm

The BLAST algorithm and the computer program that implements it were developed by Stephen Altschul, Warren Gish, and David Lipman at the US National Center for Biotechnology Information (NCBI), Webb Miller at the Pennsylvania State University, and Gene Myers at the University of Arizona.

Basic Local Alignment Search Tool (BLAST) is known for its wide use in Bioinformatics. BLAST is used to find similarities between genetic sequences (queries) and sequence databases. It follows a heuristic approach based on Smith Waterman algorithm. It locates best of possible local alignments. It is well known for its statistical significance.

The inputs of the algorithm are the genetic sequence database and a query which has to be found in the database. The outputs of the algorithm are the positions of the areas of these two strings that have similarity, as well the score of these similarities. The quality of each pair-wise alignment is represented as a score and the scores are ranked. Scoring matrices are used to calculate the score of the alignment base by base (DNA) or amino acid by amino acid (protein). The alignment score will be the sum of the scores for each position. The significance of each alignment is computed as E-value. The lower the E-value, the more significant is the score and the sequences are homologues for low E values. Each of these pairs, comprising of a database area and a query area, is called a High Score Pair (HSP). The score has significant value for biologist because it is used to compute several variables, of which the E-value is the most important.

Depending on the query and database data types, each BLAST implementation can be classified into many types. Some types are given in Table 2.

Table 2: Variations of BLAST algorithm

S. No.	Algorithm	Query	Database
1.	BLASTp	Amino acid	Protein
2.	BLASTn	Nucleotide	Nucleotide
3.	nBLASTp	Nucleotide translated	protein
4.	tBLASTn	Amino acid	Nucleotide translated
5.	tBLASTx	Nucleotide translated	Nucleotide translated

The database and query are separated in small substrings known as words. After the word list generation, the database sequences are searched for an exact match between any words of the word list found in the database is called hit. When these words are separately pattern matched among database and query, the patterns searching is extended in both directions with an aim of maximizing the alignment score S. The BLAST algorithm extends the initial word hit to a High scoring Segment Pair (HSP). The BLAST algorithm was designed by balancing speed and increased sensitivity for distant sequence relationships. BLAST emphasizes regions of local alignment to discover relationships among sequences which share only remote regions of similarity.

Reconfigurable computing was used for speeding up of BLAST elements. NCBI databank contains millions of sequences. The hardware implementation should match a sequence with NCBI database, which grows rapidly in size (Tan and Sherwood , 2005). The input output (I/O) operations of FPGA have been found to be a bottleneck in implementing the BLAST algorithm in previous versions of FPGA configurations due to enormous amount of input data to be analyzed. Recent FPGAs provide embedded blocks of RAM which offers flexibility in design and faster memory access time. Virtex-II Pro and Virtex Pro has a transceiver named ROCKET I/O transceiver which can allow transfer rates of 10.3125 Gb/s. It has been found to work efficiently under smaller transfer rates of 8 Gb/s (Stefan Dydel and Piotr Bala, 2004).

Aho-Corasick Algorithm (ACA)

The Aho-Corasick string matching algorithm is a string searching algorithm invented by Alfred V. Aho and Margaret J Corasick. It is a kind of dictionary-matching algorithm that locates elements of a finite set of strings (the “dictionary”) within an input text. It matches all patterns simultaneously.

Aho-Corasick String Matching algorithm, developed by Alfred V Aho and Margaret J Corasick belongs to a class of dictionary-matching algorithm that finds elements of a finite set of strings within an input text. It matches all patterns simultaneously. The algorithm defines a finite state machine resembling a digital tree with essential links between the various internal nodes. These internal links allow fast transitions between failed pattern matches to other branches of the tree that share a common prefix. It specializes in locating all occurrences of any of a finite number of keywords in a string of text. It consists of constructing a finite state pattern matching machine from the keywords and then using the pattern matching machine to process the text string in a single pass.

When there is a search for cat in a tree that only contains cart, the search would be a failure when it reaches a node with prefix value ca. Hence the search allows the automaton to transit between pattern matches without the need for backtracking. The time complexity of the algorithm is linear with the length of the patterns (Lp), the length of the searched text (Ls) and the number of output matches (Lo).

Time complexity = Lp + Ls + Lo

Aho-Corasick algorithm was implemented in Virtex IV f x 100 with speed grade-12. The FX series device offered better RAM/logic ratio compared to the other devices in the Virtex IV series as the architecture is constrained only by

the amount of block RAM and not the logic (West *et al.,* 2003).

Parsimony-based Phylogeny-Aware Short Read Alignment Algorithm

Significant advances in the methods for determining the arrangement of nucleotides in a DNA molecule, have led to a concept of short reads. The term short read refers to DNA sequence data that are produced by a new-generation sequencer. The read lengths typically vary between 30 and 450 nucleotides. To allow for an efficient and accurate phylogenetic analysis of such short read samples, novel maximum likelihood-based methods can be used. These approaches are known as phylogenetic placement algorithm which expertise in assigning short reads to a fixed, reference phylogeny.

PaPaRa is a popular likelihood-based phylogenetic inference algorithm following the principle analogous to that of Smith-Waterman algorithm, with affine gap penalties. However, the specific alignment kernel in PaPaRa is used to align a sequence against an ancestral state vector that is derived from varying branches in the phylogenetic reference tree. Compared to the Smith-Waterman algorithm, PaPaRa implements a unique alignment kernel and scoring scheme.

The algorithm aligns Query Sequences (QS) against ancestral state vectors derived from the reference multiple sequence alignment (RA) and the corresponding phylogenetic Reference Tree (RT) that has been inferred using the RA. In a phylogenetic tree, known sequences of living species (taxa) are assigned to the leaves of the tree. The internal nodes correspond to hypothetical common ancestors of the sub-trees they define. Because the real sequences at the ancestral nodes are not known, different methods for representing the inherent uncertainty of ancestral states (ancestral sequences) have been introduced in the context of functions for scoring alternative phylogenetic trees. The hardware/software implementation for boosting performance of a novel short read alignment method, that simultaneously aligns reads to reference multiple sequence alignments and corresponding phylogenetic trees. When mapped to a Virtex 6 FPGA, our reconfigurable architecture can compute 133.4 billion cell updates per second for the novel, tree-based alignment kernel of PaPaRa. Compared to PaPaRa, running on a 3.2 GHz Intel Core i5 CPU, we obtain speedups for the alignment kernel that range between 170 and 471. The main advantage of this algorithm is that the tree structure is also incorporated in the alignment process of phylogenetic elements.

FASTA Algorithm

FASTA is a DNA and protein sequence alignment software package first described (as FASTP) by David J Lipman and William R Pearson in 1985. Its legacy is the FASTA format which is now ubiquitous in bioinformatics.

The FASTA algorithm was developed in 1985 by Lipman and Pearson. Unlike the Needlman-Wunsch and Smith-Waterman algorithms, FASTA approximates the optimal alignment by searching and matching *k-tuples*, or subsequences of length *k*. The algorithm assumes that related proteins will have regions of identity, and by searching with *k-tuples*, the FASTA algorithm allows small regions of local identity to be found quickly. For proteins, these k-tuples tend be of length two. FASTA creates a hash table of all possible *k-tuples* and goes

through the entire query protein of length *N* and inputs the location of all the *k-tuples* into the table. Each *k-tuple* in the database sequence can be looked-up in the hash table, and any matches will allow the algorithm to mark the matching cells in the matrix. This results in a matrix in which all points of local identity of length *k* are marked.

The FASTA algorithm then identifies the 10 highest scoring diagonal runs by identifying each marked point on the matrix, and adding a positive score for every other marked cell along a diagonal, and subtracting a penalty for unmarked cells between marked cells along the diagonal. These 10 highest scoring segments are kept, and all other segments of local alignment are discarded. The 10 diagonals are scored once again using an amino acid weight matrix (PAM or BLOSUM matrix) and any diagonals with scores below a threshold are discarding again. The highest scoring diagonal is termed *init1*. The algorithm then calculates the scores of joining every combination of diagonals, as long as the diagonals are downstream from one another. To calculate the score of a joined series of diagonals, the individual scores of each of the diagonals are summed, and a constant joining penalty is subtracted each time two subsequences are joined. The maximum of the joined alignments is termed *initn* and joined alignments which have a score below a threshold are discarded. The *initn* scores are used to rank each of the alignments with sequences from the database. The final step in creating the sequence alignment is to define a diagonal band of around 32 residues wide around the *init1* diagonal from the upper-left of the matrix to the lower-right. The FASTA algorithm assumes that the optimal alignment will include or be near the *init1* diagonal. A dynamic programming algorithm is then performed in this band to find the final optimal alignment, and it essentially merges the regions of local alignment into a single alignment. The FASTA algorithm is substantially faster than the Needleman-Wunsch or Smith-Waterman alignments and thus can be more easily used in database queries. However, the time complexity of this algorithm does not seem to suggest this concept.

FastLSA Algorithm

Dynamic programming sequence alignment algorithms like Needleman-Wunsch algorithm and Smith Waterman algorithm had been found to have high time and space complexity, which would prove these algorithms to be costly. Fast Linear Space Alignment (FastLSA) algorithm is a sequential, global optimal pairwise sequence alignment algorithm that becomes accustomed to the amount of space available by trafficking space for operations. FastLSA can effectively adapt to use either linear or quadratic space, depending on the amount of available memory. It has been found that due to memory caching effects, FastLSA is always fast or faster than many algorithms.

Knowing FastLSA's strong systematic and experimental characteristics with respect to space and time complexity, FastLSA is a good contestant for parallelization, while maintaining the strong complexity properties of the sequential algorithm.

AGREP Algorithm

AGREP is a proprietary approximate string matching program, developed by Udi Manber Sun Wu between 1988 and 1991, for use with the UNIX operating system. It was later ported to OS/2, DOS, and Windows. It selects the best-suited

algorithm for the current query from a variety of the known fastest (built-in) string searching algorithms, including Manber and Wu's bitap algorithm based on Levenshtein distances.

When concentrating on power consumption of a sequence matching algorithm for matching a sequence in very large database, it would be essential to settle the methods which are heuristic in nature. AGREP algorithm, expanded as approximate GREP, is a rapid text matching algorithm that allows approximate values. This string searching algorithm incorporates possible errors and the process is more flexible to approximate string matches. With a string query compared against a database, the algorithm can detect exact and approximate matches, which are those with insertions, deletions, and substitutions. The approximate matches are introduced to find all substrings with a measure of closeness relative to the query pattern. The hardware used was Opal Kelly XEM3010 which mainly performed the functions of communication with the host software through USB protocol. It was observed that the speedup of the algorithm increased ,as the size of the database and query length increased.

Homology Search Algorithm

Homology search, can achieve high performance using off-the-shelf FPGA boards. The performance is almost comparable with small to middle class dedicated hardware systems when we use one board with one of the latest FPGAs (Xilinx XCV2000E). The time for comparing a query sequence of 2048 elements with a database sequence of 64 million elements by the Smith-Waterman algorithm is about 34 s, which is about 330 times faster than a desktop computer with a 1 GHz Pentium III. We can also accelerate the performance of a laptop computer using one PC card with one FPGA (Xilinx XCV300). The time for comparing a query sequence (1024) with database sequence (64 million) is about 185 s, which is about 30 times faster than the desktop computer.

It can evaluate the performance for the translated nucleotides. When we need to translate the sequences during the comparison, the size of each unit on the FPGA becomes about 10% larger and the parallelism in the first phase will go down to 120 from 144 (about 20% performance down). Some parts of the programs for the homology search are still under development, and we also need to improve other parts and also developing software for parallel processing of the homology search with more number of pairs of FPGAs and host computers connected by Ethernet.

Bloom Filter

A Bloom filter, conceived by Burton Howard Bloom in 1970 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; i.e., a query returns either "inside set but may be wrong" or "definitely not in set". Elements can be added to the set, but not removed. The more elements that are added to the set, the larger the probability of false positives.

Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. This compact representation is the payoff for allowing a small rate of false positives in membership queries; that is, queries might incorrectly recognize an element

as member of the set which can be made negligible by the intensive design effort. Bloom filter was implemented on FPGAs for query and data retrieval applications of computer network security [5], etc. Counting *k*-mers (substrings of length *k* in DNA sequence data) is an essential component of many methods in bioinformatics, including for genome and transcriptome assembly, for metagenomic sequencing, and for error correction of sequence reads. Using a Bloom filter, a probabilistic data structure that stores all the observed *k*-mers implicitly reduced memory requirements [6].

Content Addressable Memory (CAM):

CAM is a special type of computer memory used in certain very high speed searching applications. It is also known as associative memory, associative storage, or associative array, although the last term is more often used for a programming data structure.

A CAM is a critical device for applications involving communication networks, local area network bridges/switches, databases, lookup tables, and tag directories, due to its high-speed data search capability. Bloom filter and CAM are methods which have more hardware compatibility and higher degree of parallelism (Arun and Krishnan, 2011). CAM based architectures were implemented for high data intensive search applications using Xilinx Virtex5LX85T (Arun and Krishnan, 2011). Hardware compatible architectures like Bloom filter and CAM are need to be explored further for FPGA implementation of bioinformatics applications.

DISCUSSION AND CONCLUSION

The volume of NCBI database increases in pace much faster than the speed of IC integration predicted by Moore's law. The species count has exceeded millions with intricate sequences of base pairs. In identifying a species, various methods have been used previously, which were tedious and inaccurate attributed to human err.

It has been found that use of various gene matching algorithms in FPGA utilizing the faster data access time, flexibility and configurability of FPGA ensures accuracy and high speed throughput of genome analysis.

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. Hardware are architectures (Arun and Krishnan, 2011) have gained a lot of attention recently due to the intrinsic speed advantage over software systems.

Prominent algorithms for bioinformatics have been listed along with hardware implementations in this paper. Algorithms and architectures can be improvised to attain higher degree of parallelism, configurability and scalability to implement in a FPGA and achieve the required target of power and data speed.

REFERENCES

1. Christopher Ma, Thomas K F Wong, Lam T W, Hon W K, Sadakane K and Yiu S M (2012), "An Efficient Alignment Algorithm for Searching Simple Pseudoknots over Long Genomic Sequence", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 9, No. 6, December.
2. Grigorios Chrysos, Agathoklis Papadopoulos and Geore Petihakis (2012), "Opportunities from the Use of FPGAs as

Platforms for Bioinformatics Algorithms", Twelfth IEEE International Conference on Bioinformatics and Bioengineering (BIBE), November 2012.

3. Alachiotis N, Berger S A and Stamatakis A (2011), "Accelerating Phylogeny-Aware Short DNA Read Alignment with FPGAs", 19th IEEE Annual International Symposium Field-Programmable Custom Computing Machines (FCCM).

4. Gabriel F Villoente, Mark Oliver L Ouano, Mary Grace C Dy Jongco, and Emilyn B Escabarte (2011), "FPGA Based Agrep for DNA Microarray Sequence Searching", *International Conference on Computer Engineering and Applications,* IACSIT Press, Vol. 2.

5. Arun M and Krishnan A (2011), "Low Power Bloom Filter Architectures Using Multi Stage Lookup Techniques", *Australian Journal of Electrical & Electronics Engineering,* Australia, Vol. 8, No. 3, pp. 1-10.

6. Páll Melsted and Jonathan K Pritchard (2011), "Efficient counting of *k* -mers in DNA sequences using a bloom filter", *BMC Bioinformatics*, Vol. 12, p. 333.

7. Arun M and Krishnan A (2011), "Functional Verification of Signature Detection Architectures for High Speed Network Applications", *International Journal of Automation and Computing,* Springer, Vol. 9, No. 4, pp. 395-402.

8. Che S, Boyer M, Meng J, Tarjan D, Sheaffer J W, Lee S H and SkadronK (2009), "Rodinia: A Benchmark Suite for Heterogeneous Computing", In Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), pp. 44-54.

9. Yoginder S Dandass, Shane C Burgess, Mark Lawrence and Susan M Bridges (2008), "Accelerating String Set Matching in FPGA Hardware for Bioinformatics Research", *BMC Bioinformatics* 2008, 9:197 doi:10.1186/1471-2105-9-197, April 2008.

10. Lysaght P (2006), "FPGAs in the decade after Von Neuman Century", DATE06 Conference Proceedings, Munich, Germany.

11. Junid S A M (xxxx), "Dept. of Electron.", Univ. Teknol. MARA, Shah Alam, Majid Z A, Halim A K, "Development of DNA sequencing accelerator based on Smith Waterman algorithm with heuristic divide and conquer technique for FPGA implementation".

12. Sotiriades E, Kozanitis C and Dollas A (2006), "FPGA based architecture for DNA sequence comparison and database search", 20th International Symposium on Parallel and Distributed Processing.

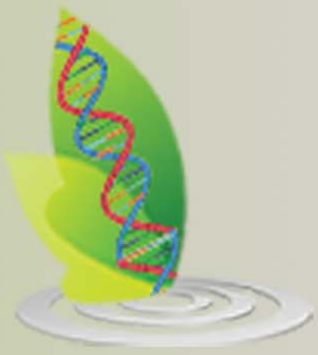
13. Euripides Sotiriades, Christos Kozanitis and Apostolos Dollas (2005), "Some Initial Results on Hardware BLAST acceleration with a reconfigurable architecture", IEEE, 1-4244-0054-06, 2006.

14. Timothy Oliver, Bertil Schmidt and Douglas Maskell (2005), "Hyper Customized Processors for Bio-Sequence Database Scanning on FPGAs," FPGA'05, Monterey, CA.

15. Tan L and Sherwood T (2005), "A High Throughput String Matching Architecture for Intrusion Detection and Prevention",

Proceedings of the 32nd Annual Intl. Symposium on Computer Architecture (ISCA 2005).

16. Stefan Dydel and Piotr Bala (2004), "Large Scale Protein Sequence Alignment Using FPGA Reprogrammable Logic Devices", Springer FPL, LNCS 3203, pp. 23-32, 2004.
17. West B, Chamberlain R D, Indeck R, and Zhang Q (2003), An FPGA-based Search Engine for Unstructured Database, *Proc. of 2nd Workshop on Application Specific Processors*, December.
18. Yoshiki Yamaguchi and Tsutomu Maruyama (2002), "High Speed Homology Search with FPGAs", *Pacific Symposium on Biocomputing*, Vol. 7, pp. 271-282.
19. Mukhopadhyay S, Changhong Tang, Huang J, Mulong Yu and Palakal M(2002), "A comparative study of genetic sequence classification algorithms", *Neural Networks for Signal Processing*, pp. 57-66.



International Journal of Life Sciences Biotechnology and Pharma Research
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlbpr@gmail.com or editor@ijlbpr.com
Website: www.ijlbpr.com

ISSN 2250-3137
9 772250 313001 04