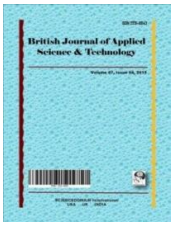

British Journal of Applied Science & Technology
10(5): 1-8, 2015, Article no.BJAST.19154
ISSN: 2231-0843

SCIENCEDOMAIN *international*
www.sciencedomain.org

A CAM (Content Addressable Memory) Architecture for Codon Matching in DNA Sequences

Parag K. Lala[1*]

[1]*Electrical Engineering, Texas A&M University-Texarkana, Texas, USA.*

Author's contribution

The sole author designed, analyzed and interpreted and prepared the manuscript.

Article Information

DOI: 10.9734/BJAST/2015/19154
Editor(s):
(1) Nan Wu, Department of Mechanical and Manufacturing Engineering, University of Manitoba, Winnipeg, Canada.
Reviewers:
(1) Anonymous, Czech Republic.
(2) Anonymous, São Paulo State University, Brazil.
(3) Anonymous, India.
Complete Peer review History: http://sciencedomain.org/review-history/10133

Original Research Article

Received 27th May 2015
Accepted 2nd July 2015
Published 10th July 2015

ABSTRACT

DNA sequences are long strands of four letters – A,T.C and G, that represent the amino-acid building components of proteins . A triplet sequence of adjacent letters on a DNA sequence is known as a codon. Multiple codons represent one of the 20 possible amino acids. DNA sequence matching is used to determine the similarity between an unidentified DNA sequence with the database of other sequences with known characteristics. Those sequences displaying high levels of similarity tend to be similar in nature and thus the matching can be a useful tool in determining the nature of the new genetic sample. This paper presents the conceptual architecture of a content addressable memory that can be used to provide simultaneous comparison of a query DNA sequence with 16 stored sequences, and identifies the ones with the highest number of codon matches with the query sequence.

Keywords: Amino acid; codon; protein; CAM; alignment.

1. INTRODUCTION

The DNA (Deoxyribonucleic acid) is composed of four constituent bases: adenine (A), thymine (T), cytosine (C), and guanine (G). When expressed as a part of a genetic sequence, the bases are grouped into triplets called *codons*; each codon produces a particular amino acid [1].

**Corresponding author: E-mail: plala@tamut.edu;*

Since there are 4 characters in the DNA alphabet, the triplet could encode 64 distinct values. However, there are only 20 amino acids. As a result, multiple codons can be used to code the same amino acid. Arginine, for example, is coded for by CGT, CGC, CGA, CGG, AGA, and AGG while Tryptophan is coded only by TGG.

Additionally, some codons, called *start* and *stop* codons, do not produce any amino acid at all and instead they are used to regulate the reading frame of a protein sequence. The fact that multiple codons can represent the same amino acid means that, in spite of mutations, similar DNA sequences represent proteins of similar function. This means that by comparing unknown DNA sequences to similar known sequences it is possible to determine the structure and function of their corresponding proteins. These projections of form and function, as well as the study of molecular evolution, are possible only by using tools of molecular sequence alignment.

2. DNA SEQUENCE MATCHING

DNA sequence matching is used for identification, analysis, and evolutionary placement of an unknown sequence [2].Typical applications involve the comparison of a single unknown DNA sequence against extremely large and growing databases of the known sequences. It is a computationally intensive task [3]. Existing algorithms for sequence matching are largely software-based with specially designed hardware implementations capturing the high-end market [4-6]. Improvements in the performance are typically provided by heuristic approaches. Heuristics are useful because they allow the databases to be quickly searched for potential matches. In doing so, a fair number of potential matches are overlooked and accuracy is thereby sacrificed.

Distributed computing speeds up the computation process by dividing the workload. It may take the form of specialized parallel hardware implementations or, in the realm of software, clusters of computers working jointly [7]. Though there are definite improvements in search speed, the same sequential comparison algorithms with their associated deficiencies are utilized in each distributed instance.

In an effort to offer an alternative, the use of a dedicated CAM (Content Addressable Memory) architecture is proposed in this paper; this allows simultaneous comparison of a single query sequence with a number of stored DNA sequences. The use of CAM for faster sequence matching was first proposed in Ref. [8,9]; these papers, however, did not present a specific CAM architecture for this purpose.

Sequence alignment is the process by which two or more sequences are arranged alongside one another to maximize the similarity observed in a pair-wise comparison of their members. Gaps ('—') are often used to pad sequences in order to achieve a better match, and typically represent insertion or deletion type mutations.

```
ATCGTACGT      A T C G T A C G - T
|        |     |   |  | | | | |  |
ACGTACGCT      A - C G T A C G C T
```

Fig. 1. Improving alignment with null characters

While the first alignment (on the left) shown above in Fig. 1 is perfectly valid, the second represents an optimal alignment of the two sequences. An alignment is said to be *optimal* if it features the maximum number of matching positions. Finding an optimal alignment is not a computationally trivial task as the number of potential placement combinations is enormous. Algorithms to find these optimal alignments must maximize the number of matching bases while minimizing null character utilization.

The Smith-Waterman algorithm [10] is a popular choice for finding the alignment between two sequences. The algorithm, however, requires the construction of a matrix where memory and time requirements are quite large and increase with the sizes of the sequences being compared. When comparisons are needed with more than one sequence, such as when searching through a large database to find and score near matches, faster implementations are required.

Progress has been made in both the realm of hardware- and software-based sequence alignment techniques; these can be divided into two categories. The first is the reduction of the search set [11]. Software solutions, such as BLAST [12] and FASTA [13] work by using heuristics to eliminate, in part or whole, some of the database entries and then execute the Smith-Waterman algorithm on the remainder. By eliminating sequences that appear to be unlikely matches, comparison time can be reduced dramatically. The time savings comes at a cost of accuracy as some potentially good alignments

can be discarded. The second group attempts to exploit of parallelism to speed up the search process. For example, the Smith-Waterman algorithm's distance matrix lends itself to cellular organization and a degree of pipelining. Additionally, by breaking the database of sequences to be searched into smaller pieces, comparisons, in either hardware or software, can be distributed and executed concurrently.

The fastest way to enhance the alignment speed is through parallel computation, for example, by distributing *n* sequences across *n* machines with each machine running a comparison between the two sequences. By changing the nature of the comparison, this is possible in a single machine by using a content addressable memory.

3. CONTENT ADDRESSABLE MEMORY

Random access memory (RAM) works by accepting an address and then returning the value stored at the memory module referenced by that address. Content-addressable memory (CAM)], in contrast, works in the opposite fashion [14]. A query value is supplied to the CAM and the addresses of the modules that contain matching stored values are returned. The ability to simultaneously compare a single query value with multiple stored values makes CAMs very useful as a basis for hardware-based searching. Of the commonly available CAM implementations, the binary CAM is the most straightforward. Sequence composition is limited to 1s and 0s and a match is returned only if the query and stored values are identical as determined by a bitwise comparison.

This paper presents a dedicated binary-CAM architecture for comparing DNA sequences; it is designed to report matches on similar rather than identical sequences. It features a three-level hierarchal organization that allows varying match criteria to be used across the three different levels of abstraction: the base, the codon, and the DNA sequence. The lowest level of the hierarchy is a grouping of three bits into a group representing a single molecular base (A,C,G or T). Three of these bases are grouped to form a codon, and sixteen codons are joined into a DNA sequence; a sequence of 16 bases is chosen arbitrarily in this paper to simplify the presentation of the main concept of the design. Thus the CAM module presented in this paper is composed of 16 words, each composed of 124 bits as shown in Fig. 2. The objective is to simultaneously compare the query sequence with 16 sequences each stored in a 124-bit word, and to identify among these the ones with the highest number of matches.

The basic component, and base of the CAM hierarchy, is the 3-bit block shown in Fig.3. These blocks are capable of storing values representing any of four DNA bases or one of two conditions: *don't care* or *off*. A don't care serves as a *wildcard* and will signal a match with any base. *Off* simulates a '—'in the sequence that indicates the base is not intended for comparison. This is helpful when comparing sequences of different sizes as the ends of smaller length DNA sequences can be padded with *offs*. A match is indicated by the block when the query and stored bases are identical, when

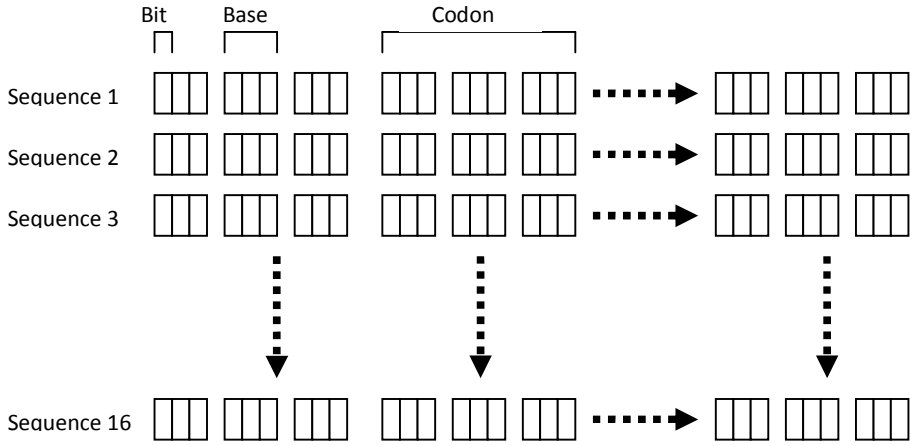

Fig. 2. A 16-word CAM organization

one of the two is a base and the other a don't care. Additionally if one is *off*, no match should occur. For example:

query	*stored*	*Match*
A	C	0
G	G	1
T	*don't care*	1
C	*off*	0
don't care	*C*	1
off	*G*	0

Bases and special conditions are encoded using three bits. The leftmost bit of the sequence functions as a flag that indicates whether a base (0) or a special condition (1), is being represented. Assignments within the base category are not subject to any design constraints and are made as follows:

000 = A 001 = C 010 = G 011 = T

The two special conditions, *don't care* and *off*, are assigned as follows:

10X = *don't care* 11X = *off*

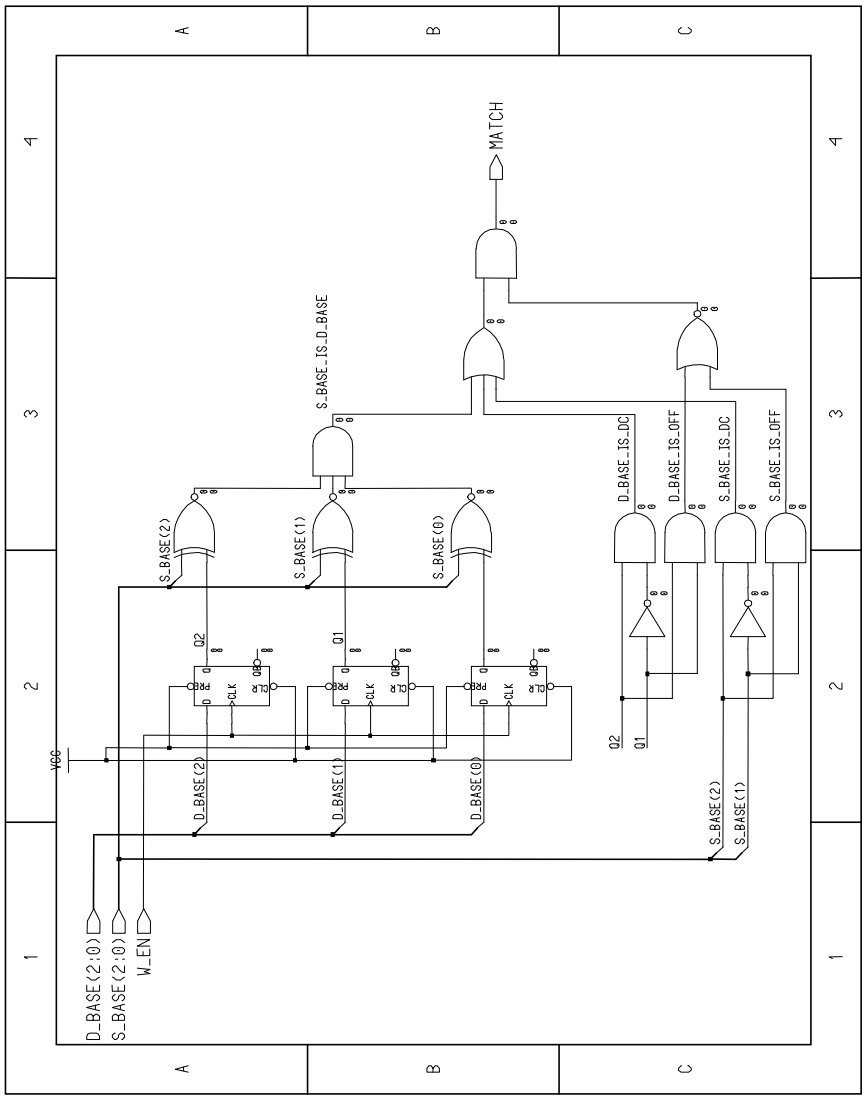

Fig. 3. A 3-bit CAM block

The data storage in the CAM is realized using D flip-flops; EX-NOR gates are utilized to compare the stored data sequence in the flip-flops with the query sequence (Fig.4). An output of 1 from an EX-NOR gate indicates a match of a single bit. If a match is signaled for all three bits, then a preliminary match is assumed for the two bases being compared.

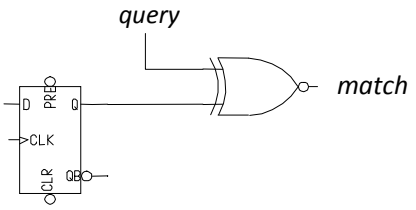

Fig. 4 Comparison logic for 1-bit

While the query and stored sequences are being compared for exact bit matches, simultaneous checks for don't care and off values are also performed. Values of 1 and 0 are returned by the don't care (DC) check and off (OFF) check logic respectively, if their corresponding conditions are detected (Fig. 3). A logical OR operation is performed on the results of the query don't care check, the stored don't care check and the preliminary match results. In this way, the presence of a don't care can overrule an exact base mismatch. The final stage in generating the match signal takes into account the possibility that one of the bases was off.

The preliminary match results are allowed to pass through if the logical AND of the OR output and the off check indicates that both sequences are on. Otherwise, the base-match signal is forced to 0 and all other considerations are overruled. The emerging signal is a binary match/no-match and is passed up the hierarchy to the codon-level for further handling.

4. MATCHING OF CODONS

Sequence comparison at the codon level is shown in Fig. 5. A comparison of the query and stored codons results in a two-bit match value. The same three bases, in the same order, are termed a complete match. When only two of the three bases are found to match in corresponding positions, a partial match has occurred as indicated below:

query	***stored***	***match***
ATG	CGT	*No*
ATG	ACG	*Partial*
ATG	ATG	*Complete*

The ability to recognize a partial match is useful in detecting related sequences. Because multiple codons can identify a particular amino acid, and those codons typically vary by a single base, being able to detect two of three matching bases allows for a match on codons that may have equivalent values.

The presence of a match is indicated by the least significant bit of match result. This bit is the output of a majority circuit fed by the outgoing match lines from the codon's three constituent bases. Whether that match is partial or complete is revealed by the most significant bit, a logical AND of the three base match lines. The coding scheme is as follows:

00: no match 01: partial match 11: complete match.

An output of 10 indicates a contradiction, that the three bit sequence has both three 1s and a majority value of 0. This pattern should therefore never occur. The remaining valid match codes are passed upwards to the sequence level upon generation.

5. FULL SEQUENCE COMPARISON

At the sequence level, the CAM architecture begins taking user-specifiable values into account when generating match signals as shown in Fig. 5. The ability to customize the sensitivity of the CAM allows for the search criteria to be better suited to the makeup of the data. For example, in a comparison of highly similar sequences a high threshold is useful to prevent being overwhelmed with matches. It is likewise helpful to avoid a complete lack of matches in sequence comparisons involving less similar entries.

To accommodate this feature, the sequence level CAM module features additional logic to count the number of matches occurring in the 16 codon modules, and then compare that value with a user-configurable threshold value. The match counting logic works by taking as inputs the 16 two-bit values indicating the match types present in each of the codon level modules. A threshold select line controls whether *partial* or *complete* matches will be accepted. A value of 0 on the

threshold select line indicates that a *partial* match is sufficient. For a partial or a complete match, the output is a 1 on the match line. A value of 1 on the threshold select line indicates that only *complete* matches are acceptable; in this case, complete matches result in an output of 1 on the match line and a 0 output for either partial matches or non-matches. The logic that accomplishes this is shown in Fig. 6.

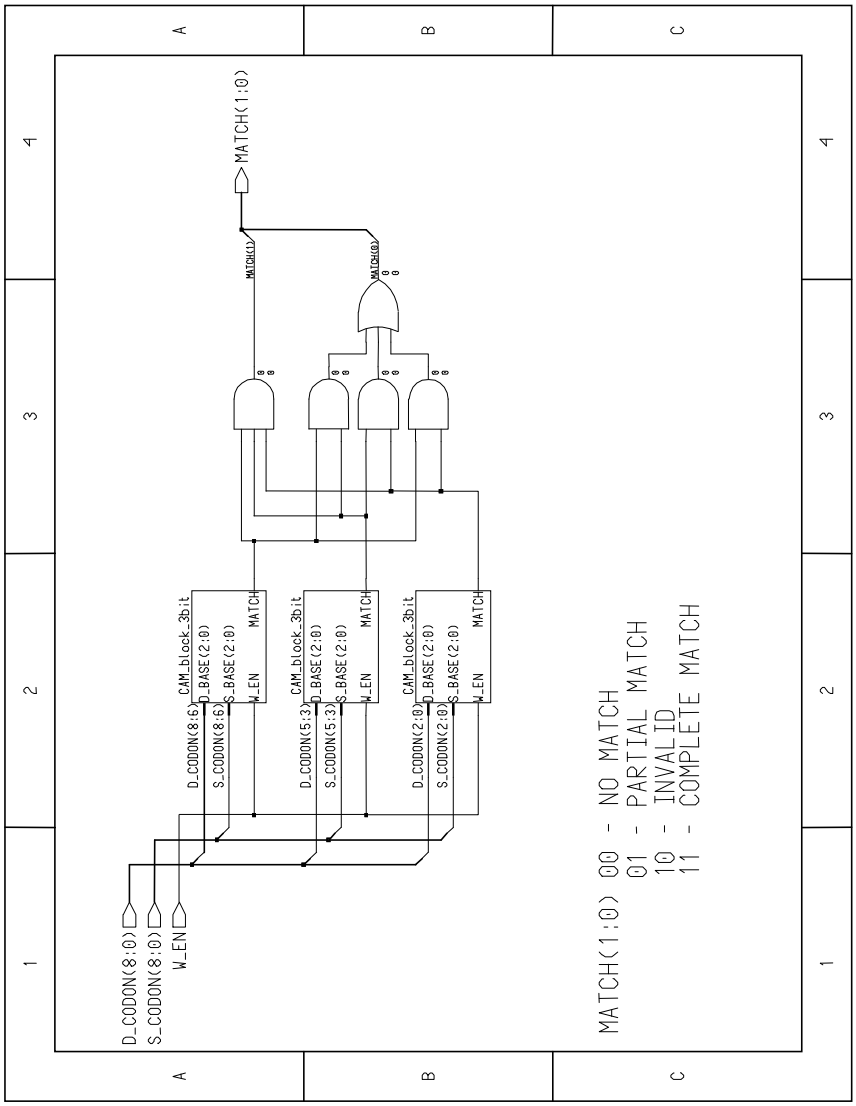

Fig. 5. Matching codons in CAM

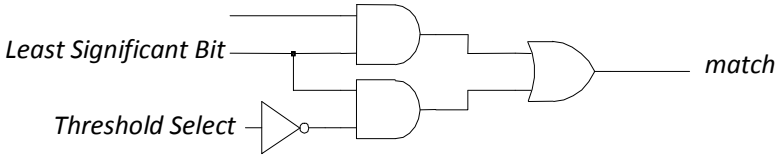

Fig. 6. Codon matching logic

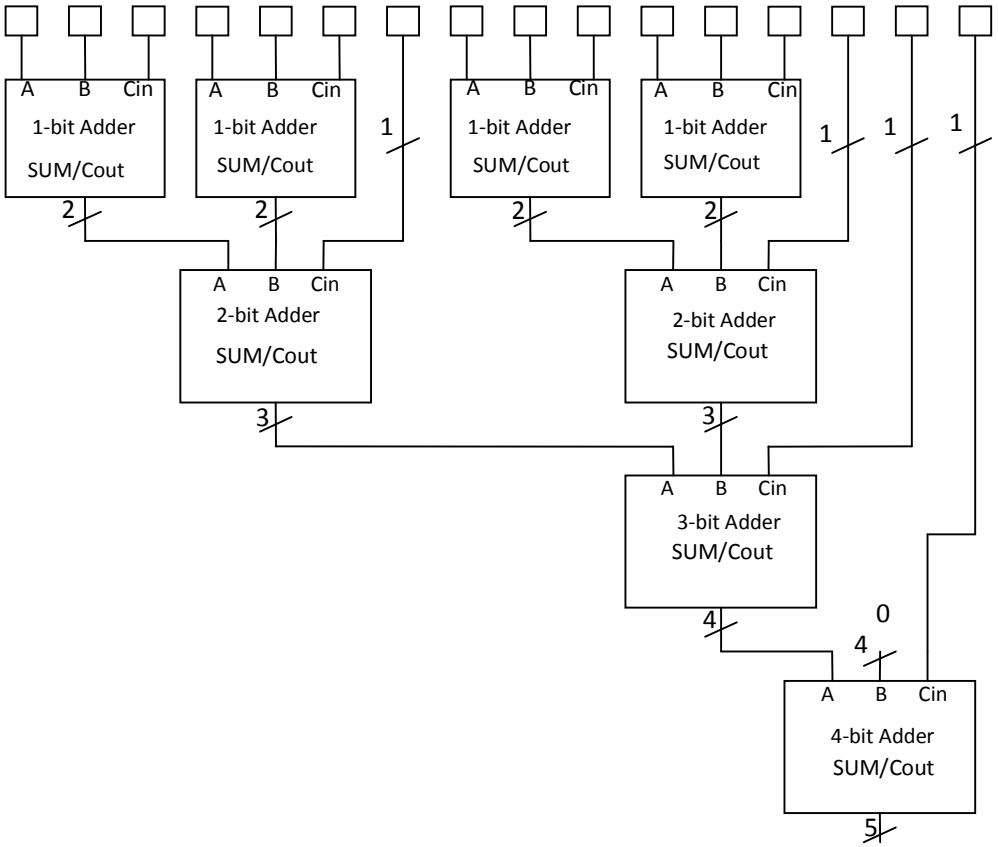

Fig. 7. Match Counter Organization

This logic is repeated in parallel for each of the 16 codon modules. Once the partial/complete match threshold has been taken into account, the match count logic uses a tree of adders as shown in Fig. 7 above to determine the number of match signals with a value of 1. The output lines from each codon matching logic block (Fig. 6) are fed into the A, B, and carry-In lines of 1-bit full adders. The 2-bit results are passed down to a row of 2- bit adders that also make use of their carry-in lines. This continues down through 3-bit adders and then 4 bit adders until the final 5-bit result holding the number of active match lines is available.

6. CONCLUSIONS

Sequence alignment and matching using currently available software-based techniques are computationally intensive and time-consuming. The CAM architecture proposed in this paper can speed up these tasks by making initial comparisons, and reducing the sequence set to be considered to a more manageable size. Thus a possible application of the proposed CAM-based methodology could be as a preliminary filter for a more exhaustive later comparison. The prescreening of sequences can take place as a global alignment in the case of very high similarity sequences, or as a local alignment between a subsequence of the query sequence and the stored sequences. The later alignment, between a query subsequence and multiple stored sequences, constitutes an important application in its own right. In order to take advantages of the speed and parallelism offered by the customized CAM implementation, new sequence alignment and matching algorithms need to be developed.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Griffiths AJF, Miller JH, Suzuki DT, Lewontin RC, Gelbart WM. Introduction to Genetic Analysis. 7th ed. New York: W. H. Freeman & Co; 1999.

2. Rosenberg MS. Sequence alignment: methods, models, concepts, and strategies, University of California Press; 2009.
3. Krane D, Raymer M, Fundamental concepts of bioinformatics. Benjamin Cummings; 2003.
4. Compeau P, Pevzner PA, Bioinfomatics algorithms, Active Learning Publishers; 2014.
5. Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing Briefings in Bioinformatics. 2010;11(5):473-483.
6. Delcher A, Phillippy A, Carlton J, Salzberg SL. Fast algorithms for large-scale genome alignment and comparison' Nucleic Acids Research. 2002;30(11): 2478-2483.
7. Sotiriades E, Kozanitis C, Dollas A, FPGA based architecture for DNA sequence comparison and database search, Proc. 20th IEEE International Parallel & Distributed Processing Symposium.2006; 25-29.
8. Lala PK. A Digital hardware-based approach for molecular sequence comparison. Information Engineering (IE). 2013;2(3):37-43.
9. Lala PK, Parkerson JP. A CAM (Content Addressable Memory)-based architecture for molecular sequence matching. Proc. Int. Conf. Bioinformatics & Computational Biology. 2011;252-256.
10. Smith TF, Waterman MS. Identification of common molecular subsequences. Jour. Mol. Biol. 1980;147:403-410.
11. Durbin R, Myers G. Invited Lecture – Accelerating Smith-Waterman Searches. Proc. Second Int. Workshop on Algorithms in Bioinformatics, Lecture Notes In Computer Science. 2002;2452:331–342.
12. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tools. Jour. Mol. Biol. 1990;215(3):403-10.
13. Lipman DJ, Pearson WR. Rapid and sensitive protein similarity searches. Science. 1985;227(4693):1435-41.
14. Chisvin L. Duckworth RJ. Content-addressable and associative memory: alternatives to the ubiquitous RAM. IEEE Computer. 1989;22:51–64.

Peer-review history:
The peer review history for this paper can be accessed here:
http://sciencedomain.org/review-history/10133