A CAM(Content Addressable Memory)-based architecture for molecular sequence matching

P.K. Lala[1] **and J.P. Parkerson**[2]
[1]Department Electrical Engineering, Texas A&M University, Texarkana, Texas, USA
[2]Department Computer Science & Computer Engineering, University of Arkansas, Fayetteville, Arkansas, USA

Abstract - *The development of elegant matching procedures has significantly improved the search time of a query sequence in a database of molecular sequences. However the architectural limitations of conventional computers allow only one sequence access/retrieval from the database at a time. This paper presents a digital hardware-based solution for fast comparison of molecular sequences.*

Keywords: CAM, sequence alignment, codon, dynamic programming , barrel shifter

1 Introduction

The derivation of functional information from genomic sequences has many applications in molecular biology. Since similar sequences behave in a similar manner, the characteristics of a new sequence can be predicted by comparing it with known sequences. Sequence comparison provides an indication of which parts of comparing sequences are similar and which parts are different. It is employed to identify sequences similar to a given sequence from a database [1-3].Two sequences are said to be *homologus* if they evolved from the same ancestor sequence, and share many common features.

Sequence similarity in many ways is synonymous with the concept of sequence alignment, which identifies similarities and differences between sequences. An alignment is a correspondence between the sequences, in which each symbol in a sequence is assigned to at most one of the symbols in the other sequence while maintaining the order of the symbols in the sequences. The main objective of the alignment is to have matching symbols at maximum number of positions.

Sequences can be compared either by *global* or *local* alignment. Global alignment spans the whole length of comparing sequences, and is appropriate if the sequences are likely to share substantial similarity. The alignment attempts to match them to each other from end to end, even though parts of the sequences may not match. Local alignment is used to identify common sub-regions of similarity between long sequences. There is no attempt to force entire sequences into an alignment, just those parts that appear to have good similarity. Thus local alignment is therefore particularly useful for comparison of long DNA sequences where only small subsequences may be related.

The number of possible alignments for two sequences of length *m* and *n* is extremely large. Therefore the obvious solution of enumerating all alignments and then choosing the one with the smallest or the highest score (depending on the scoring scheme used) is computationally impractical. An efficient alignment process needs to employ a completely automatic method e.g. *dynamic programming algorithms*, which is usually used for solving optimization problems. Needleman and Wunsch [4] were the first to propose such a method. Their motivation for developing the method was to maximize similarities between amino acid sequences. It allows global comparison of an entire query sequence with all sequences in a database. One drawback of this global alignment algorithm is that highly similar shorter subsequences with meaningful similarities may be ignored because of the overall objective of matching largest number of residues of one sequence with another sequence.

Smith and Waterman [5] proposed an algorithm, perhaps the most widely used local similarity algorithm for biological sequence comparison, based on the concept of dynamic programming. It identifies pairs of subsequences of all possible lengths in the query and the database sequence that have maximum degree of similarity. Two other dynamic programming algorithms BLAST [1] and FASTA[2] have been developed to identify possible homologues for a query sequence in a database of all other known sequences.

2 Hardware-based matching

In recent years there has been some academic and industrial efforts on the use of FPGAs (Field Programmable Gate Arrays) for enhancing the speed of sequence matching [6-8]. However all the techniques developed so far for accelerating this task sequentially compare a query sequence with sequences stored in a conventional memory system. Although this strategy results in improvement in the matching speed compared to traditional software-based algorithms, the comparison of the query sequence with the stored sequences still needs to be done one sequence at a time because of the sequential nature of information retrieval from the memory system.

The only way significant improvement in the matching speed can be achieved if a query sequence can be compared with all the stored sequences in parallel, and if all the matched sequences can be accessed simultaneously. This paper presents a digital hardware-based sequence matching procedure based on this principle. It is assumed that a query sequence is a subsequence of one or more sequences stored in the computer memory system. The bases and the *gap* in a sequence are represented by binary patterns as shown below:

$$A = 000,\quad C = 010,\quad G = 100,\quad T = 110,\quad -(dash) = xx1$$

where x is a *don't care*. Thus, a 1 in the least significant bit of a 3-bit binary representation indicates a *dash*.

Fig.1 shows the block diagram of the proposed matching system. It consists of a dedicated Content Addressable Memory (CAM) block and a bi-directional barrel shift register. A CAM unlike a traditional RAM (Random Access Memory) is addressed by the desired content, and an address that stores the content is obtained as the output of the CAM [9]. It is assumed that a CAM-based memory system will store a large number of sequences, and the content of a stored sequence has to have a few common codons i.e. all three bases identical, for it to be accessible. However, in order to avoid a large number of "hits" in the CAM, the number of codons considered for matching has to be properly selected to keep the number of accessed sequences to a realistic number. For the sake of simplifying the explanation of our approach we consider matching of only one codon for accessing relevant sequences in a CAM.

While comparing a query sequence with the stored sequences in a CAM, a better match may be obtained by shifting the query sequence left (or right) by one or more bases at a time, and then comparing it with the stored sequences. Traditionally the alignment process starts after some partially-matched sequences have been retrieved from the memory block. Then by proper placement of *dashes* in the query sequence, matching of bases in the comparing pair is maximized.

In the proposed hardware-based sequence matching approach, one of the goals is to retrieve sequences with a high degree of similarity whenever possible. The possibility of better matching is explored by shifting the query sequence to the left (or right) before accessing the stored locations. The shifting of a single base in the query sequence requires simultaneous shifting of three bits The barrel shifter in Fig.1 is used for shifting pre-selected multiple data bits. It is used to store the query sequence, and has a length of 3b where b is the number of bits in the sequence. It has a shift left and shift right capability, and can be used to shift three bits i.e. one base at a time.

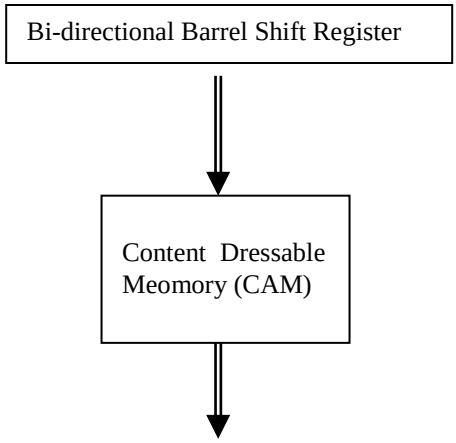

Fig. 1 Hardware based system for sequence match

Fig.2 shows the structure of an n-bit barrel shifter; the control logic determines the direction of shifting (i.e. left or right). It is assumed that that the shift-in data is 001 i.e. a *dash*. Although the proposed system is designed for local sequence alignment, it could also be used for global sequence alignment.

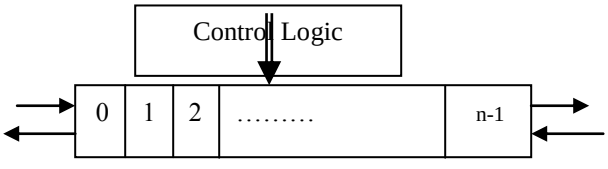

Fig. 2 Bi-directional Barrel Shift Register

A dedicated CAM-based architecture has been designed for simultaneous comparison of a pre-defined number of codons in a query sequence with the same number of codons in stored sequences in the CAM. Fig.3 shows the proposed architecture. If a majority (pre-selected) of the number of codons match with the corresponding codons in the stored sequences, then the addresses of these stored sequences are retrieved; they identify the locations of sequences that partially match the query sequence.The address corresponding to a matched location is activated via the output of an *k-out-of-n* detector. A *k*-out-of-*n* detector produces an output of 1 if at least *k* out of *n* inputs is at 1.

For example, if the first five codons of a query sequence are compared with a stored sequence and a match is assumed if any three of the five codons are similar, then a 3-out-of-5 detector is used to identify the address of the stored location. To illustrate let us assume that the first five codons in the following six codon query sequence

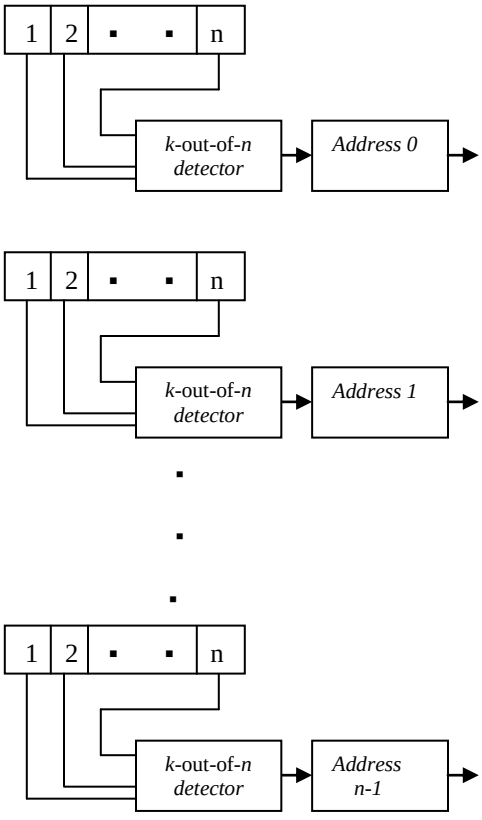

Fig. 3. Proposed $m \times n$ CAM Architecture

For example, if the first five codons of a query sequence are compared with a stored sequence and a match is assumed if any three of the five codons are similar, then a 3-out-of-5 detector is used to identify the address of the stored location. To illustrate let us assume that the first five codons in the following six codon query sequence

ACG AT- CGT –GA TCG ATG

are compared with a stored sequence

ACG CAG CGT TTC TCG AC- C-T ATC

Since three codons in the comparing sequences match, a 3-out-of-5 detector circuit will produce an output of 1. On the other hand if four codons have to be similar for a match, then a 4-out-of-5 detector has to be used; this detector will produce an output 0 in this case. Thus, a programmable detector that allows pre-selection of the *k* value in the *k*-out-of-*n* detectors will enable comparison of pre-selected parts of the query sequence to stored sequences.

As shown in Fig. 3 the CAM architecture has *m* rows (addresses) and *n* columns (codons). Each row consists of *n* comparators, *n* two-input AND gates, and a *k*-out-of-*n* programmable detector. One of the inputs to a two-input AND gate is the output of a comparator, the other input is programmable. The programmable input to the AND gate is the output of a comparator, the other input is programmable. The programmable input to the AND gate is set at 1 if a codon in the query sequence is being matched with the codon at the identical position in the stored sequence.

A matching circuit is used for comparing a codon in the query sequence with that in a stored sequence/. Fig. 4 shows the circuit for matching of two codons where *lmn* and *pqr* are the codons in the query sequence and a stored sequence respectively. Note that this circuit includes a 2-out-of-3 detector. . Thus as long as two bases in the comparing codons match, the codons are assumed to be matched. The output the 2-out-of-3 detector is connected to the input of the programmable detector associated with the stored sequence address decoder via a two-input AND gate.

To illustrate the application of the system of Fig.1 for sequence comparison let us assume the following query sequence in the barrel shifter:

ACT –GAT-CGAA

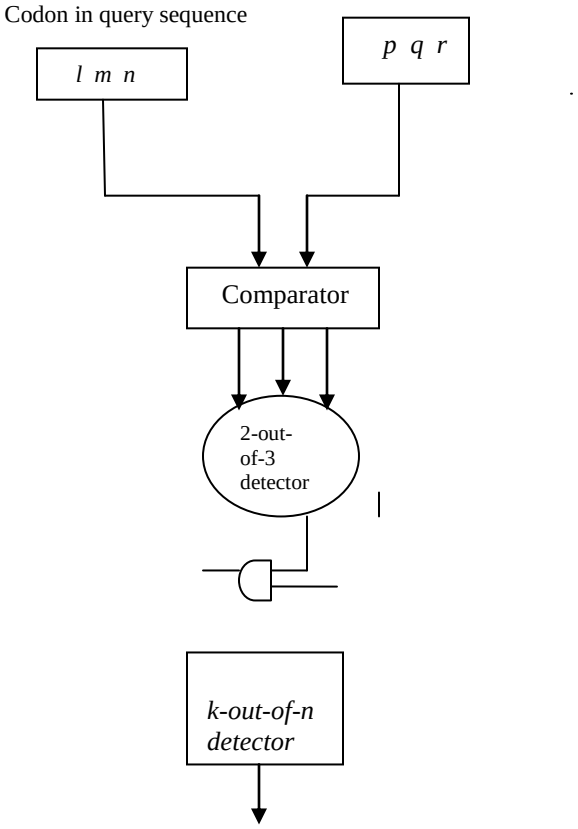

Fig . 4 Matching circuit

:Suppose the contents of the CAM (assuming it has four locations) are as follows

Content	Address
A–CT–GTCGCGA	00
ACTG–CT– GGAA	01
– CTATGT – CACT	10
AC– – G – – T– –AG	11

If the programmable detectors driving the address identifiers in the CAM are considered to be 3-out-of-4 (n=4 in this example), then only address 01 will be identified as the CAM location that contains the sequence with most matched codons. However if k=2 (i.e. 2-out-of-4 detector) is used instead, both addresses 01 and 10 will be identified.

One particular advantage of the proposed strategy is that it simplifies the identification of shorter subsequences that may be common among several stored sequences. To illustrate let us determine whether – – TAAG is part of the following stored sequences:

Address	Contents
00	ATCT–A–CAGCG
01	–TAAGC–CAGAG
10	–TGCTAAGCTGA

First – – TAAG is transferred to the barrel shift register as the query sequence. As shown below, the subsequence occupies the first six most significant positions in the barrel shifter, the remaining positions are filled with *dashes*.

– – TAAG – – – – – –

This results in matching with the contents of address 01.

A shift to the right by two bases

– – – – TAAG – – – –

will result in matching with the contents of address 10.

In certain cases a desired subsequence may appear in a stored sequence in bits and pieces. For example, it is not immediately obvious that the above subsequence is part of the sequence at location 00. However, a number of shift operations of the query sequence as shown below verifies its presence:

– – – TAAG – – – – –
– – – TA –AG – – – –
– – – T–A–AG – – – –
– – – T–A– –AG – –

Certain shift operations in this case required shifting of individual bases in the query sequence. Similar situation will arise when a query sequence has certain similarity with a stored sequence. Once the stored sequence has been retrieved a number of shift operations of the individual bases or subsequences in the query sequence may be needed to increase the number of positions with matching symbols. Thus a major objective will be how to design the barrel shifter such that one or more bases can be shifted left or right without necessarily shifting the whole query sequence,

3 Conclusions

Currently available tools for molecular sequence matching are in general software-based. The computation time needed for matching is dependent on search sensitivity. A low sensitivity search requires short computation time,

whereas for a high sensitivity search the computation time is very long. The conventional i.e. Von Neumann architecture based computers can process information only serially, thus limiting their speed of computation. This paper presented a digital hardware-based sequence matching technique that allows simultaneous comparison of a query sequence with all the stored sequences in a database, thereby achieving significant improvement in the matching speed compared to software-based techniques.

4 References

[1] S. F. Altschul, W.Gish, W. Miller, E. W. Myers and D. J. Lipman, "Basic Local alignment Search Tool", Jour. Molecular Biology, vol.215(3), pp.403-410, 1990

[2] D. J. Lipman and W.R.Pearson, "Rapid and sensitive protein similarity searches", Science 227 (4693), pp. 1435–41, 1985

.

[3]. W.R. Pearson , "Using the FASTA program to search protein and DNA sequence data base", Methods in Molecular Biology, 25, pp.365--389, 1994

[4]. S.D. Needleman and C.D. Wunsch., " A general method applicable to the search for similarities in the amino acid sequences of two proteins", Jour. Mol. Biol., vol.48, pp.443-453, 1970.

[5]. T. F.Smith and M.S.Waterman, "Identification of common molecular subsequences", Adv. Appl. Math., vol.2, pp.482- 489, 1981.

[6]. E. Sotiriades, C. Kozanitis and A. Dollas, "FPGA based Architecture for DNA Sequence Comparison and Database Search", Proc.20th IEEE International Parallel & Distributed Processing Symposium, pp 186-, 2006

[7] T. Oliver, B. Schmidt, D. Maskell, D. Nathan, and R. Clemens, " High-speed Multiple Sequence Alignment on a reconfigurable platform", International Journal of Bioinformatics Research and Applications, Vol. 2, No.4, pp.394 – 406, 2006

[8] S. Lloyd and Q.O. Snell, "Hardware Accelerated Sequence Alignment with Traceback", International Journal of Reconfigurable Computing,Vol. 2009, Article ID 762362, , 10 pages, 2009

[9] S.M. Jalaeledine and L.G. Johnson, " Associative IC memories with relational search and nearest match capabilities", IEEE Jour. Solid. State Circuits, vol.27, no.6, pp. 892-900,1992.

Acknowledgement

This work was supported in part by the National Science Foundation, USA under Grant 0925080

)