VOL. 10, NO. 18, OCTOBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

AN EFFICIENT CORE ARCHITECTURE FOR PROTEIN SEQUENCE ALIGNMENT

Mohd Nazrin, Md Isa, Sohiful Anuar Zainol Murad, Rizalafande Che Ismail and Asral Bahari Jambek
School of Microelectronic Engineering, Pauh Putra Campus, Universiti Malaysia Perlis, Arau Perlis, Malaysia
E-Mail: nazrin@unimap.edu.my

ABSTRACT

This paper presents efficient biological sequence alignment core architecture to reduce execution time of the well-known dynamic programming-based (DP) pairwise sequence alignment algorithms i.e. the Smith Waterman algorithm. The PE was prototyped in the Xilinx Virtex 5 FPGA with further improvements have been done in the scheduling strategy of alignment matrix computation and substitution coefficients' pre-loading onto the processing elements in folded systolic arrays. Implementation rresults showed that the new hardware architecture for the protein sequence alignment achieved over 100x speed-up performance, compared to the well-known SSEARCH 35 'software-only implementation' and more than ~1.6x speed up improvement against our previously implemented core in Virtex 5 FPGA.

Keywords: smith waterman, FPGA, processing element (PE), systolic array, protein sequence alignment.

INTRODUCTION

The well-known Smith-Waterman (S-W) algorithm are examples of pairwise Dynamic Programming (DP)-based sequence alignment algorithms. It is widely used to align pairs of biological sequences as it produces optimal alignment [1]. However, such algorithm has a quadratic complexity in computation time and memory space. Therefore, instead of performing sequence alignments in software such as using the SSEARCH35, the optimal algorithm can also be implemented in hardware. Typically, linear systolic array-based processors known as PEs are designed and prototyped in FPGA. Early implementation of the systolic array-based processors in hardware was known as SPLASH [2]. SPLASH was implemented in Xilinx XC3090 device and it was the first off-the-shelf Field Programmable Gate Arrays (FPGAs) used to perform sequence alignments. However, during that time, FPGA technology was as not as competitive as today's FPGAs and consequently, no extensive works were carried out after SPLASH1 and SPLASH2. Beginning from the last two decades, FPGAs have becoming the main platform for implementing systolic array-based processors. The works as reported in [3],[4],[5],[6],[7] and [8] for instance, focused on sequencing of either nucleotides (DNA sequence alignment) or amino-acid (protein sequence alignment).

Implementing sequence alignment in hardware requires a query sequence to be stored in the PE and a subject sequence from the database is streamed in into linear systolic processors. Each PE which implements the Smith-Waterman algorithmn calculates score of each query-subject sequence characters. Then score of each calculation is passed from one PE to the adjacent PE and ultimately, the final score is collected at the last PE. However, aligning DNA or protein sequences often involve hundreds, if not thousands-residue long. This is due to the fact that, each PE holds one residue at a time, thus considerable logic resources are required and this is a challenge even in modern FPGAs. Then, S-W algorithm is partitioned into smaller alignment steps and the same systolic array is re-used for multiple times, the so-called folding technique. The reported works in [3] and [6] for instance, generate n configuration elements (CE) in the PE. Another method which are reported in [7] and [9] use run time reconfiguration (RTR) technique to re-use PEs on the fly between computational passes. Due to the limited reconfiguration bandwidth and extra logic resources required for PE reconfiguration unit, the resulting overhead as reported in [7] and [9] are still considerable. In this paper, a new core architecture is proposed and a simultaneous computation and configuration (SCC)-based scheduling technique is introduced. Among its other advantages are as follows:

- **Reduced space complexity:** It alternately uses a fixed number of CEs (equal to 2) to compute any length of sequences. This will optimize logic resources instead of replicating CEs in the PE to support longer query sequences.
- **Reduced time complexity:** Optimizes the total execution time by virtually removing the con-figuration time overhead through the overlapping of alignment matrix calculation and CE configuration.

The Smith Waterman algorithmn

Biological sequences diverges from a common ancestor due to biological processes such as mutation, selection and random genetic drift [1]. Mutation for instance, involves substitution of residues, insertion of new residues and deletion of existing residues. The last two processes (i.e. insertion and deletion) are sometimes referred to as gaps in an alignment. Gaps are undesirable and thus they are penalized with certain value or known as gap cost. The cost of gaps depends on its length and generally, there are two different ways to penalize gaps. Equation (1) shows the Smith-Waterman algorithm with a linear gap penalty. It is introduced in 1981 by T. F. Smith and M. S. Waterman. Given a query sequence, X = x1,x2,x3..xi…xM (of length M) and Y = y1,y2,y3..yj…yN (of length N), this DP-based alignment algorithm searches

for the best alignment between sub-sequences of and using matrix F(i,j). This matrix calculates the largest score among the four alternatives as shown in (1) and it is built recursively. The s(xi,yj) is a substitution matrix score for residues xi and yj, while d is a constant penalty and it penalizes gaps of length g linearly.

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \quad (1)$$

In (2) a more realistic type gap penalty was introduced by GOTOH [10]. This algorithm is also known as the Smith-Waterman algorithm with affine gap penalty. In this type of gap penalty, a constant gap cost is given when opening a new gap (gap opening or *d*), while a linear and often smaller gap penalty is given for subsequent gap extensions (*e*).

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i, y_j) \\ I_x(i-1,j-1) + s(x_i, y_j) \\ I_y(i-1,j-1) + s(x_i, y_j) \end{cases} \quad (2)$$

$$I_x(i,j) = \max \begin{cases} F(i-1,j) - d \\ I_x(i-1,j) - e \end{cases}$$

In this paper, the GOTOH algorithm will be implemented and details of its corresponding hardware architecture will be discussed in the following sections.

The SCC scheduling technique

Typically, FPGA-based sequence alignment core architecture requires a PE to hold one character at a time during alignment matrix computation. Based on [6], average length of query sequence to be stored in the PEs is ~350 characters. However, typical FPGA such as Xilinx Virtex 5 FPGA, can implement only up to 160 PEs, after taking consideration of other logic units such controllers and other data path units [6]. This clearly shows that the PE systolic array must at least be re-used with two computational passes or 2-fold computations. Therefore, the SCC technique with new PE architecture as illustrated in Figure-1(a) is proposed. The PE has a fixed number of configuration elements (equal to two as shown in Figure-1(a), namely CE_0 and CE_1). With the SCC technique, alignment matrix computation uses one CE (CE_0) while configuring the content of another element (CE_1) for the subsequent pass (or fold), and vice-versa in the subsequent pass. Consider the example shown in Figure-1(b). A folding factor of 4 is assumed i.e. we want to align a query sequence of length $4N_{PE}$ and only N_{PE} can be implemented in hardware. The passes or folds are denoted as F_0, F_1, F_2 and F_3 in Figure-1(b).

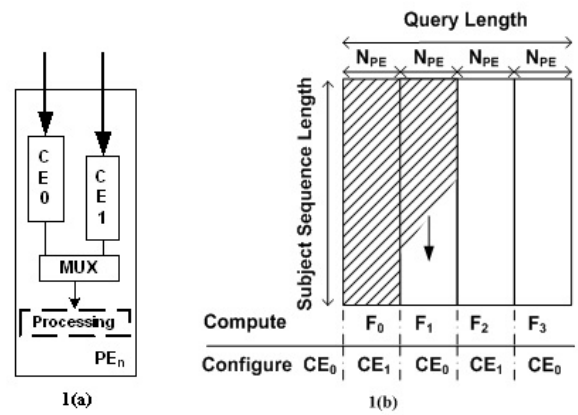

Figure-1. (a) Internal PE structure with fixed Configuration Elements (CEs) (b) Computation and configuration over the same systolic array.

The core architecture

Figure-2 presents the proposed sequence alignment core architecture. The main element in the core is the linear systolic array processors in an array of PEs called *PE_BLOCK*. The PE performs elementary functions of the Smith-Waterman algorithm, and communicates with the adjacent PE using the chain interconnections as shown in Figure-2. All the substitution matrix scores are pre-stored in the Block RAM and during the CE configuration stage; the *PARALLEL LOADER* loads the CE will its corresponding probability scores (coefficients). Our previous work in [6] shifts all the CE-related coefficients into each PE synchronously. This results in proportional total configuration time as the number of CE increases. Alternatively, the circular buffers inside the loader are specially designed to constantly revolve the columns of the substitution matrix, presenting one complete row every cycle to the pipeline PEs. With this, all systolic array CEs are configured simultaneously with a worst case configuration time of 2x32 clock cycles regardless of any length of query sequences. In terms of hardware mapping, all of the circular buffers are efficiently implemented using shift registers based on the FPGA's Look-up Tables (LUTs), or referred to as SRL32 [11]. The *SUBJECT SEQUENCE FIFO* is used to transfer subject sequences from the host computer to the core for alignment matrix computation. Once subject sequences in the FIFO are enough to start the computation, the *MAIN CONTROLLER* triggers the core to align sequences. This process continues as the database sequence flows through the buffer and each PE until all subject sequences in the database are exhausted.

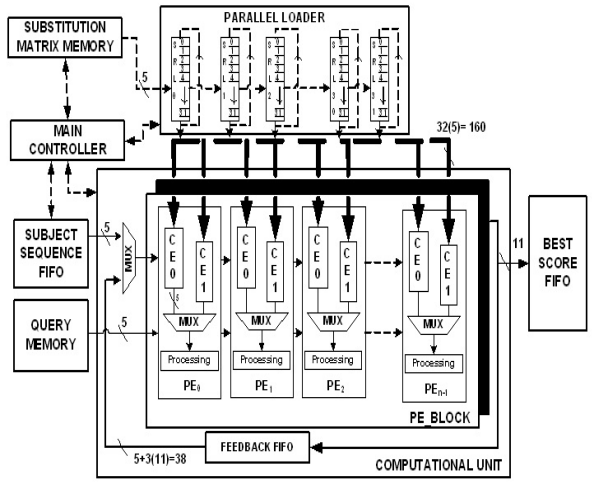

Figure-2. The core architecture with the proposed parallel loader and PE.

RESULTS

The core architecture was implemented on the Alpha Data ADM-XRC-5LX card with Virtex5VLX110 FPGA on it. A database sequence of 230,150 sequences with a total of 84,479,584 residues which is approximately 110 MB in size was stored in the host memory and transferred through the PCI bus with data transfer rate of 2,112 Mbps (i.e. 32 bit x 66MHz). As a sampel test, a query sequence of length 600 residues is tested on the SSEARCH software which ran on Intel(R) Quad Core 64-bit Q8300 (processor speed of 2.50 GHz and RAM of 4.00GB). The total scan time is 633 seconds. The same data is then tested on the proposed core, with folding factor of 6. The overall FPGA execution time reported is 5064 ms. Based on these tests, The same data then which is 125 time faster than This actual FPGA implementation is also compared against [9] which, it execution time is 6.34 seconds. The reported FPGA implementation searches the same query length against a database of 51,131,444 residues. The proposed core with the SCC scheduling technique is 25 percent faster with 65 percent greater size of database sequence. In addition, the proposed core with fixed CEs (two CEs) allows for execution of any length of query sequences by changing the fold factor at run time without reconfiguring new hardware.

Table-1. Execution time and speed up performance of the proposed core against [6].

Query Accession (length)	Q2IJ63 (128)	Q96B36 (256)	Q16515 (512)	Q96RT8 (1024)	Q8IYD8 (2048)
#Folds	1	2	4	8	16
Ref [6] (*us*)	59.64	120.71	242.81	486.99	975.36
Proposed (*us*)	40	78.38	155.1	300.24	600.04
Speed-up	1.49	1.54	1.57	1.62	1.63

Graph in Figure-3 summarizes the overall core performance against it corresponding software implementation. Various lengths of query sequences are tested, ranging from 200 residues up to 3200 residues. The SCC core used various factors (more than 20-fold) to evaluate performance of the proposed core architecture. A database sequence of 230,150 subject sequences is used for both hardware implementation and the SSEARCH. The total execution time includes communication overheads including PE configuration time and alignment matrix computation in the systolic arrays. Based on the measured time, their corresponding speed up is calculated for each query sequence by dividing the software execution time with the total execution time of the proposed core. From the graph, it is clearly shows that the speed up of the core with the Simultaneous Computation and Configuration (SCC) technique grows linearly as the fold factor increases with no additional logic resources for PE replications.

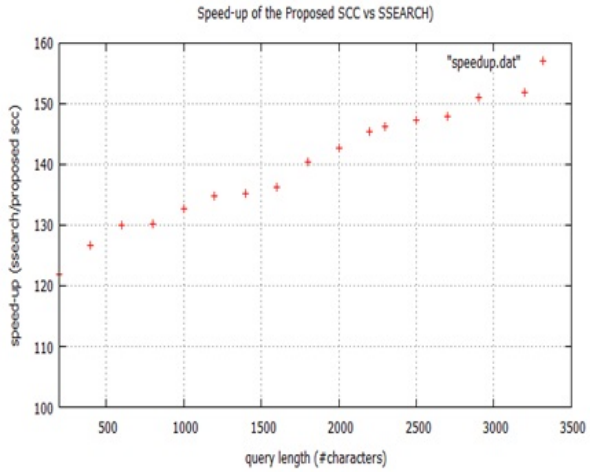

Figure-3. Total execution time and speed-up of the proposed core against the SSEARCH.

CONCLUSIONS

The proposed core architecture with the Simultaneous Computation and Configuration (SCC) scheduling technique has successfully optimized the execution time of the Smith-Waterman algorithmn in hardware. Results showed that the biological sequence alignment core with the efficient scheduling of alignment matrix computation and substitution matrix pre-loading for subsequent computations, achieved over 100x speed-up compared to the 'software only implementation' and more than 1.6x speed-up against our previously implemented core.

REFERENCES

[1] R. Durbin, Eddy S., Krogh A. and Mitchison G. 1998. Biological Sequence Analysis: Probabilistic Models for Proteins and Nucleic Acids: Cambridge University Press, Cambridge UK.

[2] D. T. Hoang. 1992. FPGA Implementation of Systolic Sequence Alignment, in International Workshop on Field Programmable Logic and Applications. Vienna, Austria.

[3] K. Benkrid, L. Ying and A. Benkrid. 2009. A Highly Parameterized and Efficient FPGA-Based Skeleton for Pairwise Biological Sequence Alignment, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 17, pp. 561-570.

[4] P. Zhang, G. Tan and G. R. Gao. 2007. Implementation of the Smith-Waterman Algorithm on A Re configurable Supercomputing Platform, Altera Corporation.

[5] T. Van Court and M. C. Herbordt. 2007. Families of FPGA-based accelerators for approximate string matching, Microprocessors and Microsystems, Vol. 31, pp. 135-145.

[6] M. N. Isa, K. Benkrid, T. Clayton, C. Ling and A. T. Erdogan. 2011. An FPGA-based parameterised and scalable optimal solutions for pairwise biological sequence analysis, presented at 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS).

[7] Y. Yoshiki, M. Yosuke, M. Tsutomu and K. Akihiko. 2002. High Speed Homology Search Using Run-Time Reconfiguration. In: Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications: Springer-Verlag.

[8] Y. Yamaguchi *et al.* 2011. FPGA-Based Smith-Waterman Algorithm: Analysis and Novel Design Reconfigurable Computing: Architectures, Tools and Applications, vol. 6578, Lecture Notes in Computer Science: Springer Berlin, Heidelberg, pp. 181-192.

[9] T. F. Oliver, B. Schmidt and D. L. Maskell. 2005. Reconfigurable architectures for bio-sequence database scanning on FPGAs, IEEE Transactions on Circuits and Systems II, Vol. 52, pp. 851-855.

[10] G. Osamu, An improved algorithm for matching biological sequences, Journal of Molecular Biology, Vol. 162, pp. 705-708.

[11] "Virtex-5 Family User Guide," Xilinx, Inc., San Jose, CA 2009.