An FPGA-Based Web Server for High Performance Biological Sequence Alignment

Ying Liu[1], Khaled Benkrid[1], AbdSamad Benkrid[2] and Server Kasap[1]
[1]*Institute for Integrated Micro and Nano Systems, Joint Research Institute for Integrated Systems, School of Engineering, The University of Edinburgh, The King's Buildings, Mayfield Road, Edinburgh, EH9 3J, UK*
[2] *The Queen's University of Belfast, School of Electronics, Electrical Engineering and Computer Science, University Road, Belfast, BT7 1NN, Northern Ireland, UK*
(y.liu,k.benkrid)@ed.ac.uk

Abstract

This paper presents the design and implementation of the FPGA-based web server for biological sequence alignment. Central to this web-server is a set of highly parameterisable, scalable, and platform-independent FPGA cores for biological sequence alignment. The web server consists of an HTML–based interface, a MySQL database which holds user queries and results, a set of biological databases, a library of FPGA configurations, a host application servicing user requests, and an FPGA coprocessor for the acceleration of the sequence alignment operation. The paper presents a real implementation of this server on an HP ProLiant DL145 server with a Celoxica RCHTX FPGA board. Compared to an optimized pure software implementation, our FPGA-based web server achieved a two order of magnitude speed-up for a pairwise protein sequence alignment application based on the Smith-Waterman algorithm. The FPGA-based implementation has the added advantage of being over 100x more energy-efficient.

1. Introduction

Scanning genome and protein sequence databases is an essential task in molecular biology. Biologists find out the structural and functional similarities between a query sequence and a subject database sequence by scanning the existing genome or protein database sequences, with real world applications in disease diagnosis, drug engineering, bio-material engineering and genetic engineering of plants and animals. There are numbers of biological sequence alignment algorithms with various execution speed/accuracy tradeoffs. Among these, we cite dynamic programming based algorithms [2, 3], heuristic-based algorithms [4, 5], and HMM-based algorithms [6].

The most accurate algorithms for pairwise sequence alignment are exhaustive search dynamic programming (DP)-based algorithms such as the Needleman-Wunsch algorithm [2], and the Smith-Waterman algorithm [3]. The latter is the most commonly used DP algorithm which finds the most similar pair of sub-segments in a pair of biological sequences. However, given that the computation complexity of such algorithms is quadratic with respect to the sequence lengths, heuristics, tailored to general purpose processors, are often introduced to reduce the computation complexity and speed-up bio-sequence database searching. The most commonly used heuristic algorithm for pairwise sequence alignment, for instance, is the BLAST algorithm [4]. In general, however, the quicker the heuristic method is, the worse is the result accuracy. Hence, accurate and fast alignment algorithms need faster computer technologies to keep up with the exponential increase in the sizes of biological databases [1].

Field Programmable Gate Arrays (FPGAs) have been proposed as a candidate technology to solve this problem as they promise the high performance and low power of a dedicated hardware solution while being reprogrammable. Few commercial players are offering real customer solutions for high performance FPGA-based sequence analysis, the most prominent of which are TimeLogic, Progeniq [7, 8] and Mitrionics [9]. TimeLogic, for instance, offers FPGA-based desktop and server solutions for biological sequence analysis applications. Progeniq on the other hand offer mostly small FPGA-based acceleration cards for workstations. Mitrionics offer an FPGA-based server for the BLAST algorithm. Speed-up figures reported by these companies are in the range x20-x80. Nonetheless, these solutions are specific to the hardware and software of choice, and hence do not offer users the flexibility to migrate to other platforms. Moreover, to the best of our knowledge, there is not any academic-based FPGA server solution for biological sequence analysis.

In this paper, we propose a flexible multi-process FPGA-based web server for efficient biological sequence analysis. An FPGA-based web server for pairwise sequence alignment has been realised to demonstrate the

978-0-7695-3714-6/09 $25.00 © 2009 IEEE
DOI 10.1109/AHS.2009.59

benefits of our approach. Central to this server is a highly parameterisable FPGA skeleton for pairwise bio-sequence alignment using dynamic programming algorithms [10].

The remainder of this paper is organised as follows. First, important background information on pairwise bio-sequence alignment algorithms is briefly introduced in section 2. After that, the design of our FPGA-based web server is detailed in section 3. Section 4 then presents a real hardware implementation of a generic DP-based pairwise sequence alignment algorithm on an HP ProLiant DL145 server with a Celoxica RCHTX FPGA board, with detailed implementation results. Finally, conclusions and plans for future work are laid out in section 5.

2. Background

Biological sequences (e.g. DNA or protein sequences) evolve through a process of mutation, selection, and random genetic drift [11]. Mutation, in particular, manifests itself through three main processes, namely: *substitution* of residues (i.e. a residue A in the sequence is substituted by another residue B), *insertion* of new residues, and *deletion* of existing residues. Insertion and deletion are referred to as *gaps*. The gap character "-" is introduced to present the alignment between sequences. There are four ways to indicate the alignment between two sequence *s* and *t* as shown below:

$$(a,a)\ denotes\ a\ match\ (no\ change\ from\ s\ to\ t),$$
$$(a,-)\ denotes\ deletion\ of\ character\ a\ (in\ s),$$
$$(a,b)\ denotes\ replacement\ of\ a\ (in\ s)\ by\ b\ (in\ t),$$
$$(-,b)\ denotes\ insertion\ of\ chatacter\ b\ (in\ s).$$

For example, an alignment of two sequences *s* and *t* (Figure 1) is an arrangement of *s* and *t* by position, where *s* and *t* can be padded with gap symbols to achieve the same length:

$$s: A\ \ G\ \ C\ \ A\ \ C\ \ A\ \ C\ \ -\ \ C$$
$$t: A\ \ -\ \ C\ \ A\ \ C\ \ A\ \ C\ \ T\ \ A$$

Figure 1. Denotations of the alignment between sequences *s* and *t*

(A,A) indicates a match, $(\mathrm{G},-)$ indicates the deletion of G, $(-,\mathrm{T})$ indicates the insertion of T, and (C,A) indicates the replacement of C by A. Gaps should be taken into account when aligning biological sequences.

The most basic pairwise sequence analysis task is to ask if two sequences are related or not, and by how much. It is usually done by first aligning the sequences (or part of sequences) and then deciding whether the alignment is more likely to have occurred because the sequences are related or just by chance. The key issues of the methods are listed below [12]:

- What sorts of alignment should be considered;
- The scoring system used to rank alignments;
- The algorithm used to find optimal (or good) scoring alignments;
- The statistical methods used to evaluate the significance of an alignment score.

The degree of similarity between pairs of biological sequences is measured by a score, which is a summation of odd-log score between pairwise residues in addition to gap penalties. The odd-log scores are based on the statistical likelihood of any possible alignment of pairwise residues, and is often summarised in a substitution matrix (e.g. BLOSUM50, BLOSUM62, PAM). Figure 2 presents a 20 by 20 substitution matrix called BLOSUM50 for amino-acid residues, used for protein sequence alignments.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5

Figure 2. The Blosum50 substitution matrix

The gap penalty depends on the length of gaps and is often assumed independent of the gap residues. There are two types of gap penalties, known as linear gaps and affine gaps. The linear gap is a simple model with constant gap penalty, denoted as:

$$Penalty(g) = -g*d,$$

where g is the length of gaps and d is the constant penalty for each single gap. Affine gaps consist of opening gap penalties and extension gap penalties. The constant penalty value *d* for opening a gap is normally bigger than the penalty value *e* of extending a gap. Affine gaps are formulated as:

$$Penalty(g) = -d-(g-1)*e$$

Since it is often the case that a few gaps are as frequent as a single gap, the affine gap model is much more realistic than the linear gap model. The following however presents dynamic programming algorithms in the case of

linear gaps for the sake of clarity. The extension to the case of affine gaps is straightforward [12].

2.1 Dynamic Programming Algorithms

The Needleman-Wunsch (NW) and Smith-Waterman (SW) algorithms are two widely used dynamic programming algorithms for pairwise biological sequence alignment. Needleman-Wunsch is a global alignment algorithm, which is suitable for small sequences, as it aligns the sequences from the beginning to the end. In the case of longer sequences, Needleman-Wunsch introduces too much gap penalty noises that reduce the accuracy of the alignment. Hence, the Smith-Waterman algorithm is used to avoid this problem by looking for similar segments (or subsequences) in sequence pairs (the so-called local alignment problem). In both cases, however, an alignment matrix is computed by a recursion equation with different initial values (see Equation 1 below for the Needleman-Wunsch case, and Equation 2 in the case of the Smith-Waterman algorithm). Here, an alignment between two sequences X = $\{x_i\}$ and Y= $\{y_j\}$ is made.

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

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

From the recursion equations, the alignment score is obtained as the largest value of three alternatives:

- An alignment between x_i and y_j, in which case the new score is $F(i-1,j-1)+s(x_i,y_j)$ where $s(x_i,y_j)$ is the substitution matrix score or entry for residues x_i and y_j.
- An alignment between xi and a gap in Y, in which case the new score is $F(i-1,j)-d$, where d is the gap penalty.
- An alignment between y_j and a gap in X, in which case the new score is $F(i,j-1)-d$, where d is the gap penalty.

The dependency of each cell can be clearly shown in Figure 3. Here, each cell on the diagonal of the alignment matrix is independent of each other, which allows systolic architectures to be introduced to increase the parallelism and speed up the computation of dynamic programming algorithms.

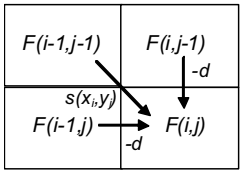

Figure 3. Data dependency of dynamic programming algorithms

As mentioned earlier, affine gap penalties provide a more realistic model of the biological phenomenon of residue insertions and deletions. The affine gap penalty is defined using two constants d and e as follows:

*Penalty(g)=-d-(g-1)*e, where g is the gap length.*

Multiple values of each pair of residue *(i,j)* need to be computed instead of just one in the affine gap case, with recursive equations similar to the ones for linear gaps, both for local and global alignment [12].

2.2 BLAST

The BLAST algorithm is a heuristic algorithm for pairwise sequence alignment, developed by the National Center for Biotechnology Information (NCBI). The basic idea of NCBI BLAST is to find positions in the subject sequences (the database) which are similar to certain query sequences segments, allowing for insertion, deletion and substitution. These positions are called High-Scoring Pairs (HSPs), which are defined as pairs of aligned segments of sequence pairs that generate an alignment score above a certain threshold T. Starting with these HSPs instead of computing the whole score matrix for pairwise sequences result in substantial computational savings, which makes BLAST searches faster than Smith-Waterman, for instance, on general purpose processors. In general, NCBI BLAST consists of three steps as illustrated in Figure 4 below:

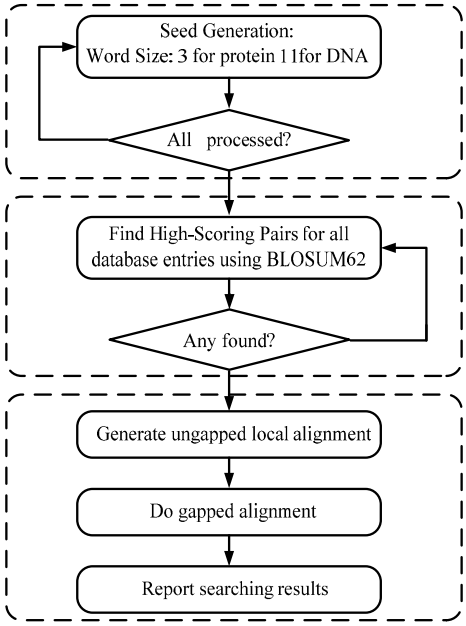

Figure 4 Steps employed by BLAST

Seed generation: A seed is a fragment of fixed length W (W = 3 for protein sequences and W = 11 for DNA sequences) from the query sequence. For a query sequence of length M, the number of seeds generated is M-W+1 as shown in Figure 5.

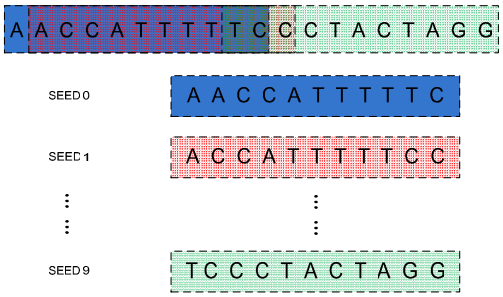

Figure 5. DNA Seed Generation

Hits finding: The second step in the BLAST algorithm is to find matches (potential HSPs) between seeds and the database stream. A substitution matrix is used for proteins in order to obtain the substitution scores for pairwise sequence segments. If the substitution score is above a threshold *T*, the system considers these segments to be High-Scoring Pairs (HSPs).

Hits extension: The final step is to extend the HSPs found in the second step in both directions to complete the alignment. The most widely used implementation of BLAST looks at ungapped alignments only. Nonetheless current versions of NCBI BLAST provide gapped alignments too. The extension process terminates either when it reaches the end of one sequence, or when the score decreases sufficiently to a falloff parameter *F* [13].

NCBI BLAST is able to perform five different similarity searches, namely BLASTn, BLASTp, BLASTx, tBLASTn, and tBLASTx (see Table 1 below).

Table 1. Various searches of BLAST

Search Name	Query Type	Database Type	Translation
BLASTn	Nucleotide	Nucleotide	None
BLASTp	Protein	Protein	None
BLASTx	Nucleotide	Protein	Query
tBLASTn	Protein	Nucleotide	Database
tBLASTx	Nucleotide	Nucleotide	Both

BLASTn is a member of the BLAST program package which searches DNA sequences against DNA database. Due to the high degree of conservation in DNA sequences, the High-Scoring Pairs in the second step of BLASTn are the exact matches between query seeds and database stream. BLASTp is the most widely used program for aligning a Protein sequence against a Protein database. Instead of finding exact matches, it looks for non-identical pairs that generate high similarity scores by using similarity substitution matrices, such as PAM and BLOSUM62. One HSP stands for a pair of similar fragments with a score above the threshold *T*.

The other three members of BLAST searches require translation. A nucleotide sequence can be translated into protein sequences in 6 different frames. Searches are processed against all 6 frames to get the final BLAST result.

3. Our Proposed FPGA-based Web Server for Efficient Biological Sequence Alignment

Figure 6 presents our FPGA-based web server for biological sequence analysis. The web server consists of an HTML based web interface, a MySQL database for storing the queries and results, a list of biological sequence database, a database of FPGA configuration, a host application that services user requests, and FPGA co-processor(s) that accelerate the sequence alignment tasks. The web interface takes all the parameters needed for one unknown query as following:

- *sequence symbol type* i.e. DNA, RNA, or Protein sequences
- *Alignment task* e.g. Smith-Waterman, Needleman-Wunsch, BLASTn, BLASTp.
- *Query sequence*: Here the query sequence length and the alignment task dictate the configuration(s) to be downloaded to the FPGA(s), if need be.
- *match score* i.e. the score attributed to a symbol match. This is supplied in the form of a substitution matrix e.g. BLOSUM matrix.
- *gap penalties*: This could be either linear or affine. The gap penalty is loaded to the FPGA co-processor at run time.
- *match score thresholds*: e.g. the HSPs threshold *T* and the fall-off parameter *F* in the case of the BLAST algorithm.

When the user submits a query, a unique ID is given for checking the result. A MySQL database is used to store all query information with the unique ID into a query list. The host application manages communications between the MySQL database, sequence databases, and FPGA configurations on the one hand, and the FPGA coprocessor(s) on the other hand.

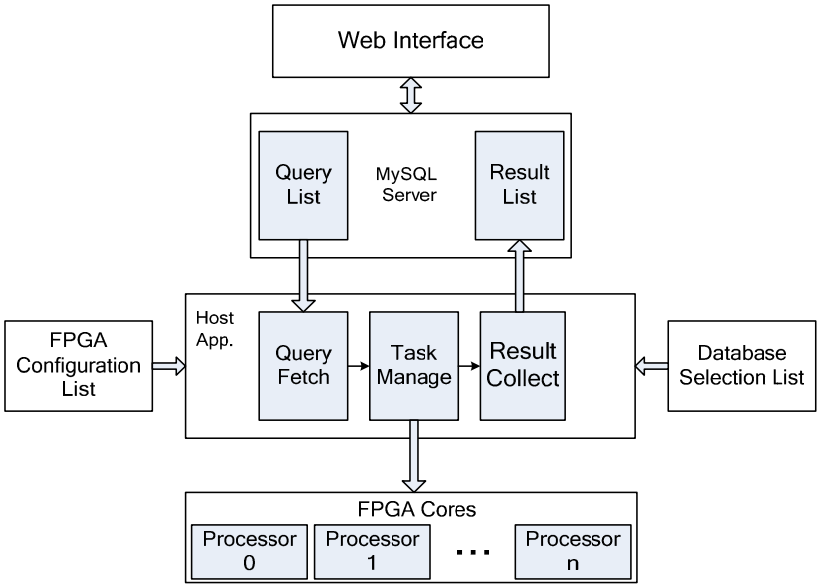

Figure 6. Proposed FPGA-based Web Server for Efficient Biological Sequence Alignment

The FPGA coprocessors are initially configured with the most commonly used configurations by the host application. The subject sequence databases are loaded into host memory. When a query comes, the host application will check if FPGA coprocessors need to be reconfigured. Since many users might be requesting the server at the same time, several processes would run concurrently on the same FPGA chip, and/or across many FPGA chips. Efficient partitioning and scheduling algorithms need to be employed to minimize the average user waiting time. After the configuration of the FPGA co-processor(s), the host application processes the incoming query set before sending it to FPGA coprocessor, and configures the FPGA with query coefficients according to the query sequence submitted. The selected database sequences are sent from the host memory to the FPGA processor(s) for the alignment executions. At the end of processing, alignment results are collected by the host application and stored into a result list in the MySQL database. The users can obtain the results through their unique ID via a web interface to the MySQL database.

4. Real Implementation: an FPGA-based Web Server for DP-based Pairwise Sequence Alignment

In order to demonstrate our proposed FPGA-based web server, we implemented a prototype web server on an HP ProLiant DL145 server with a Celoxica RCHTX FPGA board, and run a generic FPGA skeleton for DP-based pairwise sequence alignment on it. The following first describes our generic skeleton, before implementation results are presented.

4.1 A Generic FPGA Skeleton for DP-based Pairwise Sequence Alignment

Figure 7 presents a generic systolic architecture for DP-based pairwise sequence alignment [14]. Each processing array (PE) in the array consists of control logic, coefficient RAM and computation logic.

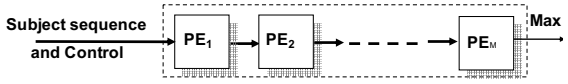

Figure 7. Linear Processor Elements array architecture of pairwise biological sequence alignment with single pass

Given the DNA sequences comprise only four nucleotides, whereas protein sequences comprise fivefold larger variety of sequence characters, it is much easier to detect patterns of sequence similarity between protein sequences than between DNA sequences [13]. Pearson [14, 15, 16] has proven that searches with a protein sequence encoded by a DNA sequence against a DNA

sequence database yield far fewer significant matches than searches using the corresponding protein sequence. Hence, we conducted the web server implementation on Smith-Waterman algorithm for protein sequence and extended it to all the variances of DP based bio-sequence alignment. Figure 8 illustrates the PE architecture/behaviour for a DP pairwise sequence alignment algorithm with linear gap penalty. Control logic separates the different database subject sequence calculations. Coefficient RAMs are run-time reconfigurable according to the query sequence in hand (one column of substitution matrix coefficients are stored in one RAM). Given that each coefficient is 2-bit wide for DNA and 5-bit wide for proteins, and that there are 21 elements in one column, distributed memory on FPGAs is used to implement the coefficient RAMs. The computation logic is tailored to the parameters of dynamic programming algorithms in hand, i.e. sequence type, alignment type, gap penalty, etc.

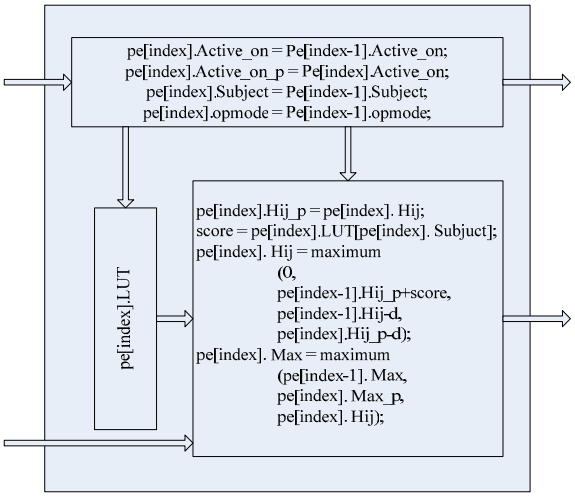

Figure 8. Processor Element for pairwise biological sequence alignment using the Smith-Waterman algorithm

As each processing element copes with one residue from the query sequence, the computation complexity of a single pass linear array of processing elements is reduced from quadratic to linear, as denoted below:

$$O(m, n) = m + n - 1 \quad (4)$$

where $O(m, n)$ represents the number of clock cycles spent on each query sequence with length n aligned to a subject sequence database of m residues.

In the case of longer sequences, due to the possible resource limitation of the FPGA chip in hand, our design can be tailored to cope with this by folding the systolic array and using several alignment passes instead of just one, at the expense of longer processing time (Figure 9). The complexity of the design with multiple passes is denoted as:

$$O(m, n) = m * passNum + n - 1 \quad (5)$$

where passNum presents the amount of passes that one query needs. passNum is calculate as follows:

$$passNum = \left\lceil \frac{querylength}{PENum} \right\rceil \quad (6)$$

where $querylength$ is the length of the query sequence, and $PENum$ is the maximum number of PEs that can be fitted into the FPGA chip in hand. Hence, the single pass design can be considered as a special case of multiple pass design with $passNum = 1$, which gives the best throughput.

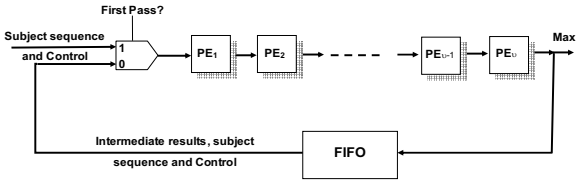

Figure 9. Linear Processor Element (PE) array architecture for pairwise biological sequence alignment with multiple passes

The above described skeleton has been captured in Handel-C, which makes it FPGA-platform independent. Indeed, since our Handel-C description did not use any specific FPGA resource or placement constraints, it can be easily retargeted to a variety of FPGA platforms e.g. from Xilinx and Altera.

Compared to previously reported FPGA-based biological sequence alignment accelerators [18-22], our FPGA-based web server solution has been designed to be platform-independent with a service-based model of operation in mind. Detailed comparison between our implementation and previously reported FPGA-based biological sequence alignment accelerators is presented in [24].

4.2 Implementation Results

As mentioned above, our FPGA-based web server has been targeted to an HP ProLiant DL145, which has an AMD 64bit processor and a Celoxica RCHTX FPGA board. The latter has a Xilinx Virtex 4 LX160-11 FPGA chip on it. All data transfer between the host application and FPGA coprocessor pass through the Hyper-Transport interface and is supported by the DSM library in Handel-C. Celoxica DK5 suite and Xilinx ISE 9.1i were used to compile our Handel-C code into FPGA configurations (bitstreams).

The performance of the resulting FPGA-based web server is illustrated with a single core implementation of the Smith-Waterman algorithm on the Virtex-4 FPGA,

using the Swiss-Prot Protein database as a sequence database. A single processing element of a systolic array implementing the Smith-Waterman algorithm with linear gap penalty and a processing word length of 16 bits, consumes ~110 slices. Consequently, we were able to fit in ~500 PEs on a Xilinx Virtex 4 LX160 -11 FPGA.

We also compared our FPGA implementation with equivalent optimised software implementation running on the Dual-Core AMD Opteron™ processor 2218, and captured in C++. A set of 100 queries of 250 residues is chosen to align against the latest Swiss-Prot database [23] as a test bench. The processing time of a single core FPGA implementation was 188.4 seconds (i.e. 3min 8sec) while it was 28840 seconds (i.e. 8 hours 36sec) for the software implementation. This means that our FPGA implementation outperforms an equivalent software implementation by 150x.

We also performed power consumption measurements for both hardware and software implementations, using a power meter. Factoring the execution time, the total energy consumed by our FPGA-based web server implementation was 2.09 Wh (i.e. 7542 Joule), whereas the software implementation consumed 360.5 Wh (i.e. 1297800 Joule), for the same set of queries. This means that our FPGA-based web server implementation is 172x more energy efficient than the software implementation. This shows that FPGAs offer a high performance and low power platform for biological sequence alignment applications.

It is worth noting that since the computation complexity of the software implementation grows quadratically with the sequence sizes, while it grows linearly in the case of FPGA implementation, the more PEs we can fit on FPGAs, the better the speed up figure we get.

5. Conclusion

In this paper, we have presented the design and implementation of an FPGA-based web server for biological sequence alignment, where FPGA coprocessors are used for the acceleration of sequence alignment tasks. A demonstrator FPGA-based web server was implemented on an HP ProLiant DL145 server with a Celoxica RCHTX FPGA board containing one Xilinx Virtex 4 LX160-11 FPGA chip. Using a highly parameterisable FPGA skeleton for pairwise sequence alignment, our FPGA-based web server implementation outperformed an equivalent optimised software implementation of the Smith-Waterman algorithm by 150x, while consuming 172x less energy. This shows FPGAs to be an efficient and efficacious computing platform for biological sequence alignment applications.

The work presented in this paper is part of a bigger effort by the authors which aims to harness the computational performance and reprogrammability features of FPGAs in the field of Bioinformatics and Computational Biology. Future work includes the extension of the library of biological sequence analysis algorithms implemented on the server including profile HMM searches, various BLAST algorithm variations, and phylogenetic tree construction algorithms.

6. References

[1] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A. and Wheeler, D.L. '*Genbank*'. *Nucleic Acids Res.*, **28**, 15-18. 2000

[2] Needleman, S. and Wunsch, C. '*A general method applicable to the search for similarities in the amino acid sequence of two sequences*', J. Mol. Biol., 48(3), (1970), 443-453

[3] Smith, T.F. and Waterman, M.S. '*Identification of common molecular subsequences*'. Journal of Molecular Biology, 147, 195-197, 1981

[4] Altschul, S. F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. 'Basic Local Alignment Search Tool', Journal of Molecular Biology, 215, pp. 403-410, 1990

[5] Pearson, W.R. and Lipman, D.J. '*FASTA: Improved tools for biological sequence omparison'*, Proceedings of the National Academy of Sciences, USA 85, (1988), pp. 2444-2448.

[6] HMMER user's guide: biological sequence analysis using pro hidden Markov models. http://hmmer.wustl.edu (1998)

[7] Gollery, M., Rector, D., Lindelien, J, *'TLFAM-Pro: A New Prokaryotic Protein Family Database'*, http://www.timelogic.com/ **,** TimeLogic Corporation,2009

[8] '*BioBoost for Bioinformatics'*, http://www.progeniq.com/**,** progeniq Private Limited, 2009

[9] *'Mitrion-C Open bio-project'*, http://*www.mitrionics.com*, mitrionics, 2009

[10] Benkrid, K., Liu, Y., and Benkrid A., '*Design and Implementation of a Highly Parameterised FPGA-Based Skeleton for Pairwise Biological Sequence Alignment'.* FCCM'07, pp. 275-278, 2007

[11] Harrison G. A., Tanner, J. M., Pilbeam D. R., and Baker, P. T. '*Human Biology: An introduction to human evolution, variation, growth, and adaptability*', Oxford Science Publications, 1988

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

[13] Kasap, S., Benkrid, K., Liu, Y., *'High performance FPGA-based core for BLAST sequence alignment with the two-hit method',* BioInformatics and BioEngineering 2008, pp. 1-7, 2008.

[14] Mount D.W., *'Bioinformatics: Sequence and Genome Analysis'*, Cold Spring Harbour Laboratory Press, Cold Spring Harbour, New York, USA, 2001

[15] Pearson W.R., *'Comparison of methods for searching protein sequence databases'*, 1995. *Protein Science e.* **4:** 1150–1160.

[16] Pearson W.R., *'Effective protein sequence comparison'*, 1996, *Methods Enzymol*. **266:** 227–258.

[17] Pearson W.R., *'Flexible sequence similarity searching with the FASTA3 program package'*, 2000, *Methods Mol. Biol.* **132:** 185–219

[18] Yamaguchi, Y., Maruyama, T., and Konagaya, A. *'High Speed Homology Search with FPGAs'*, Proceedings of the Pacific Symposium on Biocomputing, pp.271-282, 2002.

[19] VanCourt, T. and Herbordt, M. C. *'Families of FPGA-Based Algorithms for Approximate String Matching',* Proceedings of Application-Specific Systems, Architectures, and Processors, ASAP'04, pp. 354-364, 2004

[20] Oliver, T., Schmidt, B. and Maskell, D. *'Hyper customized processors for bio-sequence database scanning on FPGAs'*, Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays

[21] Bojanic, S., Caffarena, G., Pedreira., C., and Nieto-Taladriz, O., *'High Speed Circuits for Genetics Applications'*, Proceedings of the 24th International Conference on Microelectronics (MIEL 2004), Vol. 12, pp. 517-524, 2004

[22] Puttegowda, K., Worek, W., Pappas, N., Dandapani, A., and Athanas, P., *'A Run-Time Reconfigurable System for Gene-Sequence Searching'*, Proceedings of the 16th International Conference on VLSI Design VLSI'03, pp. 561–566,2006.

[23] http://www.uniprot.org/, Uniprot, 2008

[24] K. Benkrid, Y. Liu, A. Benkrid, A Highly Parameterised and Efficient FPGA-Based Skeleton for Pairwise Biological Sequence Alignment, IEEE Transactions on Very Large Scale Integration Systems, v 17, p 561-570, No. 4, 2009