A High Performance FPGA-based Implementation of Position Specific Iterated BLAST

Server Kasap
The University of Edinburgh
(+44) 131 650 5592
s.kasap@ed.ac.uk

Khaled Benkrid
The University of Edinburgh
(+44) 131 650 5592
k.benkrid@ed.ac.uk

Ying Liu
The University of Edinburgh
(+44) 131 650 5592
y.liu@ed.ac.uk

School of Electronics and Engineering, Mayfield Road, Edinburgh EH9 3JL, Scotland, UK

ABSTRACT
We present in this paper the first reported FPGA implementation of the Position Specific Iterated BLAST (PSI-BLAST) algorithm. The latter is a heuristic biological sequence alignment algorithm that is widely used in the bioinformatics and computational biology world in order to detect weak homologs. The architecture of our FPGA implementation is parameterized in terms of sequence lengths, scoring matrix, gap penalties and cut-off and threshold values. It is composed of various blmocks each of which performs one step of the algorithm in parallel. This results in high performance implementations, which easily outperform equivalent software implementations by one order of magnitude or more. Furthermore, the core was captured in an FPGA-platform-independent language, namely the Handel-C language, to which no specific resource inference or placement constraints were applied. This makes our core portable across different FPGA families and architectures.

Categories and Subject Descriptors
B.6.0 [**Logic Design**]: General.B.6.1 [**Logic Design**]: Design Styles – *Sequential circuits, Logic arrays, Parallel circuits, Combinational circuits.*

General Terms
Algorithms, Design

1. INTRODUCTION
In Bioinformatics and Computational biology (BCB), biological sequence alignment is a very common task where subject sequences from a large database are aligned to a query sequence to find similarities between the query sequence and the database sequences [1]. A major application of sequence alignment is to infer biological information about a newly discovered sequence from a set of previously annotated sequences. For instance, if a new sequence is found to be similar to a known cancerous sequence, then information regarding the functionality of the new sequence can be inferred, something which is extremely useful in early disease diagnosis and drug engineering. Furthermore, the study of evolutionary development and history of species is essentially based on biological sequence alignment [1] [2].
However, sequence alignment is a computationally intensive operation. Hence, utilization of fast computing platforms is mandatory. Field Programmable Gate Arrays (FPGAs) have been recently proposed as an efficacious and efficient implementation platform for sequence alignment algorithms, thanks to their flexible computing and memory architecture which gives them ASIC-like performance with the added programmability feature [3] [4] [5].

There are various biological sequence alignment algorithms. In this paper, we concentrate on Basic Local Alignment Search Tool (BLAST) [6] which is a local alignment algorithm. It is much faster than ordinary exhaustive dynamic programming algorithms since it is heuristic. Essential background information on the general BLAST algorithm can be found in our other paper [7].Hardware implementation of a variant of BLAST named Position Specific Iterated BLAST (PSI-BLAST) [8] is presented in this paper.

The remainder of this paper will first elaborate on PSI-BLAST algorithm. Then, the design and implementation of our FPGA core for PSI-BLAST will be detailed. Following this, implementation results are presented and then evaluated comparatively with the performance of equivalent software implementations running on a desktop computer. Finally, conclusions are laid out with plans for future work.

2. Position Specific Iterated BLAST (PSI-BLAST)
PSI-BLAST is a profile (or motif) based search method which is more sensitive than Gapped BLAST [8] at detecting distant relationships among query and database sequences. It can identify additional related database sequences that might otherwise be missed by Gapped BLAST. In essence, PSI-BLAST is iterative Gapped BLAST. It consists of following main steps:

1. A database search is conducted with Gapped BLAST using a query sequence and a scoring matrix (BLOSUM 50).
2. All of the subject sequences with local alignment score higher than a specific threshold value are identified and then a multiple alignment of the segments of these high scoring subject sequences and the query sequence is performed. This multiple sequence alignment is detailed to some extent in subsection 2.1.
3. A profile called PSSM (Position Specific Scoring Matrix) is abstracted from the aforementioned multiple sequence alignment. A PSSM is a matrix with *n* rows and *m* columns

FPGA'09, February 22–24, 2009, Monterey, California, USA.

where *n* is the size of the alphabet (*n*=20 for protein sequences) and *m* is the length of the query sequence. More information regarding PSSM and its construction from multiple sequence alignment is presented in subsection 2.2.

4. Gapped BLAST is iterated using the obtained PSSM instead of the query sequence itself and the substitution matrix with the aim of identifying a higher number of related database sequences. The way PSSM is utilized in Gapped BLAST is explained in section 3 below.
5. After the second iteration, PSSM is updated by taking newly discovered distant relative database sequences into account through steps 2 and 3.This new PSSM is utilized in subsequent Gapped BLAST iteration.
6. Iterations of Gapped BLAST continue until no more new related database sequences are discovered.

2.1 Multiple Sequence Alignment

After each iteration of Gapped BLAST, the high scoring segments of subject sequences and the query sequence are multiply aligned. The query sequence is used as a template for constructing the multiple alignments. That is to say that each subject sequence segment is first pairwisely and globally aligned to the query sequence and then all these obtained alignments are compiled to form a multiple alignment M.

Columns of M that involve gap characters inserted into the query sequence are ignored so that M has the same length as the query sequence. The PSSM matrix is constructed from the trimmed multiple alignment M as will be explained in the next subsection.

2.2 Construction of Position Specific Scoring Matrix (PSSM)

A PSSM is a motif descriptor which includes a weight (score, probability) for each residue occurring at each position along the motif. It is a 20 by *m* matrix for protein sequences where *m* is the length of the motif. The 20 rows of each column specify the probability of finding each of the 20 amino acids at that position in the motif. The M_{jk} element of the PSSM is the score for the j^{th} amino acid at the k^{th} position of the motif.

The PSSM can be constructed from the multiple alignment M described in subsection 2.1 above. The first step of PSSM matrix construction is the reduction of each column of M to form the columns of matrix M_C (motif under consideration). To construct each column C of M_C, the set R of sequences that contribute a residue in column C of M are identified.

We use the data-dependent pseudo-count method proposed in [8] to calculate the values of PSSM elements from M_C. In it, the PSSM score for the j^{th} amino acid at the k^{th} position (M_{jk}) is computed as shown in equation 1, where P_{jk} is the frequency of residue j at the k^{th} position of the M_C matrix and P_j is the background frequency of residue j. Background frequencies of residues are derived from large and carefully selected sets of alignments [9].

$$M_{jk} = \log\left(\frac{P_{jk}}{P_j}\right) \quad (1)$$

The following equation is used to compute P_{jk}:

$$P_{jk} = \frac{\alpha * f_{jk} + \beta * g_{jk}}{\alpha + \beta} \quad (2)$$

where f_{jk} and g_{jk} are the observed frequency and pseudo-count frequency of residue j at position k of M_C, respectively. α and β are the relative weights given to the observed and pseudo-count frequency residues, respectively. In equation 2, α is equal to N_C-1, where N_C is the total number of different residue types, including gaps, observed in the columns of M_C, whereas β is set to the default value of 7. The value of g_{jk} in equation 2 is set to depend on the observed residue frequencies via a scoring matrix S_{ij} (see equation 3) where λ is a natural scale for S_{ij} [8].

$$g_{jk} = p_j \sum_{i=1}^{20} f_{ik} e^{\lambda S_{ij}} \quad (3)$$

As stated above, in PSI-BLAST, it is this obtained PSSM matrix that is used instead of the query sequence and original substitution matrix (e.g. BLOSUM50) in subsequent database search iterations, with the aim of identifying a higher number of related database sequences. The process of database search and PSSM generation is iterated until no more new related database sequences are discovered.

3. HARDWARE IMPLEMENTATION

Figure 2 shows a hardware architecture which implements the PSI-BLAST algorithm. Each block in the architecture implements one step of the algorithm as described in the above sections, except for the pre-processing query sequence step and construction of the Position Specific Scoring Matrix (PSSM) which are implemented by high level application software running on the host computer. The architecture consists of 8 *HitFinderTwoHit* blocks, 2 *UngappedExtender* blocks and 1 *GappedExtender block* all of which are running in parallel. There are also 8 32K x 5 bits subject sequence memories each of which holds a number of subject sequences. Note that each subject sequence memory belongs to one *HitFinderTwoHit* block. Each *HitFinderTwoHit* block is composed of 5 *HitFinder* blocks and 1 *TwoHitMethod* block. Each *HitFinder* block scans its assigned subject sequence memory to find exact matches of the query words in the subject sequences. Each *TwoHitMethod* block performs the two-hit method procedure on hits coming from the 5 *HitFinder* blocks which are in the same *HitFinderTwoHit* block as the *TwoHitMethod* block. Besides these, each *UngappedExtender* block extends the two hits found by its 4 allocated *TwoHitMethod* blocks without allowing gaps, in order to obtain local ungapped alignments. Finally, a single *GappedExtender* block implements the modified Needleman-Wunsch algorithm to produce local gapped alignments from local ungapped alignments obtained in 2 *UngappedExtender* blocks.

The high level application software is explained in subsection 3.1 whereas all of the blocks which constitute the architecture shown in figure 2 are detailed in our other paper [7].

3.1 High Level Application Software

Figure 1 shows the organization of our PSI-BLAST FPGA implementation. Application software running on the host has many duties, the most important of which is the query sequence pre-processing. In brief, the application software finds 3 letter long query words which score at least threshold value T with a scoring matrix when aligned with words extracted from the query sequence. However, in case when there is a constructed PSSM, the application software finds 3 letter long query words which

score at least threshold value T when aligned with the PSSM. Then, the location address of each of these query words in the query sequence is placed at a vacant position in an upper word list and a lower word list pair depending on the 2 most significant letters and 2 least significant letters of the query word, respectively. Note that there are 5 upper word and lower word list pairs.

As it can be seen in figure 1, we produced various FPGA configuration bit files for different threshold and cut-off value parameters. The first task of the application software is to pick the proper bit file, depending on the user-supplied algorithm parameters, from a database of FPGA configurations and load it on to the FPGA chip. Afterwards, the application software runs the hardware configuration in 4 modes. In mode 1, the application software sends one of the 5 upper word and lower word list pairs to each of the 5 *HitFinder* blocks in every *HitFinderTwoHit* block. In mode 2, a number of subject sequences are sent to the 8 available subject sequence memories on FPGA, depending on the subject sequence lengths. In mode 3, the application software sends a query sequence to the FPGA to be stored in the query sequence memories within the 2 *UngappedExtender* blocks and the single *GappedExtender* block. Finally, the execution of the hardware configuration is launched in mode 4. After some time, the FPGA starts sending the high scoring subject sequences to host with their alignment scores. By repeating these steps several times for different subject sequences, we can align all subject sequences in a sequence database to the query sequence (or to PSSM when we have a constructed one).

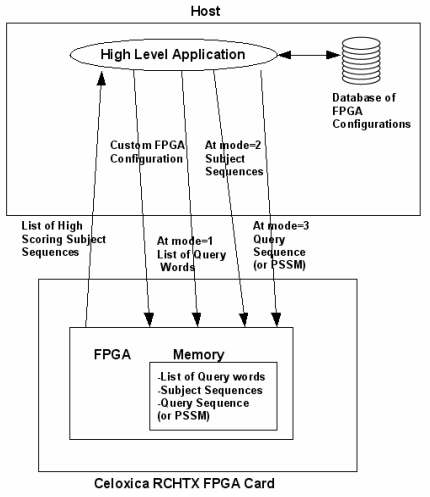

Figure 1. Organization of our Gapped BLAST system

When all subject sequences are aligned, segments of subject sequences which have a local alignment score higher than a specific threshold value are multiply aligned with the query sequence as explained in subsection 2.1 by the application software. Then, the application software constructs the PSSM matrix from the multiple sequence alignment as explained in subsection 2.2 above.

After the construction of PSSM, application software iterates all the aforementioned steps to perform a new Gapped BLAST operation. However, in subsequent iteration, the application software sends the PSSM constructed matrix instead of the query sequence to the FPGA, in mode 3, to be stored in the PSSM memories within the 2 *UngappedExtender* blocks and the single *GappedExtender* block. These Gapped BLAST iterations continue until no more new high scoring subject sequences are found.

4. Results

Our PSI-BLAST design was captured in the Handel C language to which no specific resource inference or placement constraints were applied. Hence, it can be directly targeted to a variety of FPGA platforms. The resulting core was compiled into EDIF by Agility's DK5 SP2 suite from which FPGA bitstreams were generated using Xilinx ISE9.2 tool.

A real hardware implementation of the core was achieved on a Celoxica RCHTX FPGA board [10] which has a Xilinx Virtex 4 (xc4vlx160ff1148-11) FPGA and off-chip memory fitted on it. In our implementation, however, the off-chip memory was not used. The operation of the core was tested on the Swiss-Prot protein sequence database [11] with various query protein sequences.

We have also implemented the PSI-BLAST algorithm in C in order compare our hardware implementation with a pure software implementation. Table 1 presents timing performance figures of both hardware and software implementations for one Gapped BLAST iteration for 9 random query protein sequences of various lengths. Note that all Gapped BLAST iterations take approximately same amount of time. The FPGA hardware was clocked at 15 MHz only and the software implementation was executed on an Intel Centrino Duo 2.2 GHz PC with 2 GB RAM. Furthermore, the same threshold and cut-off values were used in both hardware and software implementations at every step of the algorithm

As it can be seen from table 1, our FPGA implementation result in substantial speed-up compared to software, ranging from 20x to 44x (the speed-up figure depends on the query sequence). The reason behind this high speed-up figure of the FPGA implementation, despite the huge difference in clock frequency, is due to the high level of process parallelism on FPGA.

5. CONCLUSION

In this paper, the detailed FPGA implementation of the PSI-BLAST algorithm has been presented. To our knowledge this is the first FPGA implementation of this algorithm ever reported in the literature. The hardware architecture is composed of various blocks each of which performs a specific step of the algorithm in parallel. Moreover, the FPGA core is parameterized in terms of the sequence lengths, scoring matrix, gap penalties and cut-off and threshold values. The resulting implementation outperforms an equivalent desktop-based software implementation by at least one order-of magnitude. Furthermore, it was designed in the Handel-C language which makes it FPGA-platform-independent. As a result, the same core can be ported to other FPGA architectures from different vendors.

The work presented in this paper is part of a bigger project which seeks to harness the computational performance and re-configurability features of FPGAs in the field of bioinformatics and computational biology. Future work includes a multi-threaded implementation of various flavours of BLAST (including the PSI-BLAST algorithm) and other sequence analysis algorithms with a web interface that allows users to submit queries remotely to an FPGA-based server.

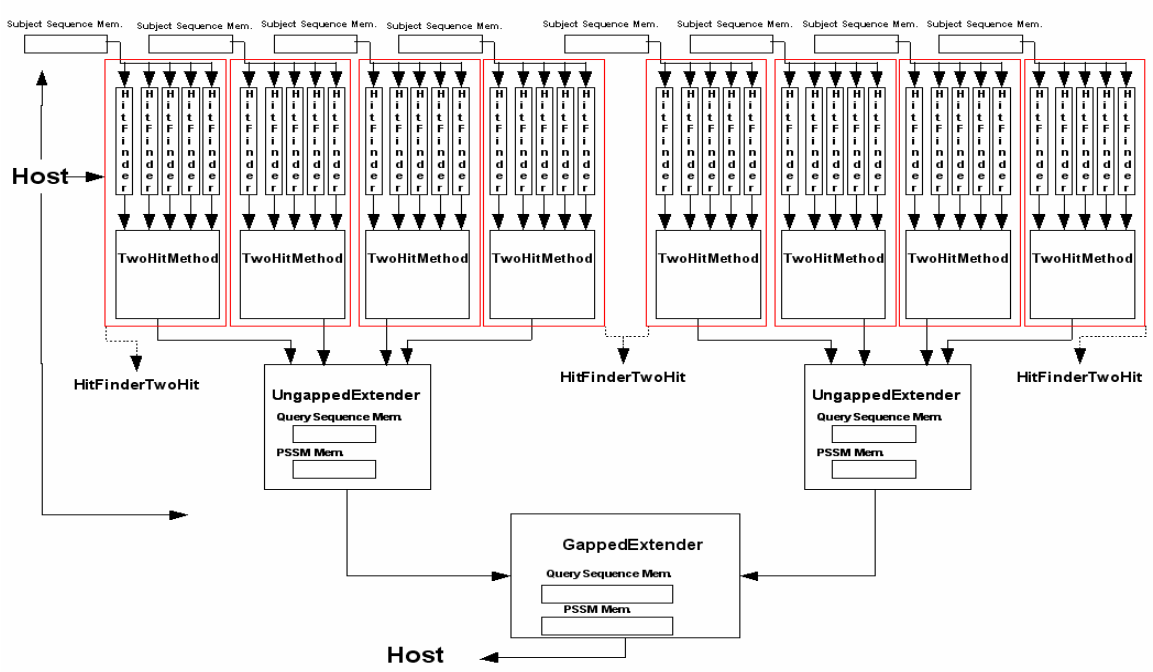

Figure 2. Hardware architecture for the PSI-BLAST algorithm

. **Table 1. Timing performance figures of hardware and software implementations for one Gapped BLAST iteration for 9 random protein sequences queried in Swiss-Prot protein sequence database**

	No of Residues in Query Sequence	No of Query words	FPGA Execution time (sec)	Software execution time (sec)	FPGA Speed-up
1. Query Sequence	111	116	4.45	91.56	20.58
2. Query Sequence	214	98	5.01	131.93	26.34
3. Query Sequence	368	136	4.32	137.42	31.81
4. Query Sequence	459	263	5.88	211.42	35.96
5. Query Sequence	565	137	5.73	181.48	31.67
6. Query Sequence	635	140	5.36	194.45	36.28
7. Query Sequence	746	117	6.83	233.25	34.15
8. Query Sequence	864	240	7.01	311.23	44.40
9. Query Sequence	985	53	5.33	194.12	36.42

.

6. REFERENCES

[1] 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.

[2] Hein, J. 'A New Methodology that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when a phylogeny is given'. Journal of Molecular Biology, 6, pp.649-668, 1989.

[3] Hoang, D.T. 'Searching genetic databases on Splash 2', in Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, pp. 185-191, 1993.

[4] Gokhale, M. et al. 'Processing in memory: The Terasys massively parallel PIM array', Computer, 28 (4), pp. 23-31, April 1995.

[5] TimeLogic Corporation, 'Decypher Scalable, High Performance Biocomputing Solutions', http://www.timelogic.com.

[6] 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.

[7] Kasap, S., Benkrid, K., Liu, Y.,' Design and Implementation of an FPGA-based Core for Gapped BLAST Sequence Alignment with the Two-Hit Method', Engineering Letters, Vol. 16, Issue: 3, pp. 443-452,2008.

[8] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. 'Gapped BLAST and PSI-BLAST: a new generation of protein database search programs', Nucleic Acid Research, Oxford Journals, 25(17), pp. 3389-3402, 1997.

[9] Yi-Kuo Yu, John C. Wootton, and Stephen F. Altschul, 'The compositional adjustment of amino acid substitution matrices', PNAS, Vol. 100,no 26,pp. 15688-15693, December, 2003.

[10] RCHTX FPGA Board Reference Manual, Celoxica Plc, http://www.celoxica.com.

[11] Boeckmann, B., et al., 'The SWISS-PROT protein knowledgebase and its supplement TrEMBL' in 2003 Nucleic Acids Research, Vol.31, pp. 365-370, 2003.