

Implementing Smith Waterman's Similarity Matrix Computations on Reconfigurable Logic Hardware

Farouq H. Humed*, Razali Jidin, Sri Noraima Othman*

Department of Electrical and Communication Engineering – College of Engineering

Universiti Tenaga Nasional

*Quantum Beez Sdn Bhd-Block D-2, UPM-MTDC

farouq.hatem@quantumbeez.com, razali@uniten.edu.my, aima.othman@quantumbeez.com

Abstract

Reconfigurable computing is a computing paradigm that combines the flexibility of software with high performance hardware. The reconfigurable computing platform such as field programmable logic arrays (FPGA) can be employed to efficiently solve a wide spectrum of computational problems especially the repetitive tasks. In this paper, we present a method to accelerate part of Smith Waterman (SW) algorithm on the FPGA. The SW is typically used to align two or more genomic sequences. Specifically the task that is to be performed by the FPGA is to compute the similarity matrix as it requires heavy CPU or processor time. A parallel computation of similarity matrix in the form of multi-stage processing element (PE) can execute in parallel to reduce processing time. Depending on the number of PE, multiple-fold speedup can be achieved over the typical processor's sequential implementation.

1. Introduction

With the objective to infer homology and subsequently gene function, Smith-Waterman algorithm (SW) is currently being used to search the optimal local alignment between two sequences. When searching sequence databases that contain very long genomic sequence that can be hundreds of millions of sequences, this algorithm will spend huge processor time. Other algorithms like FASTA [1] and BLAST [2] can execute at higher speed compared to SW, the result produced will not be as accurate as the one obtained by the SW.

On many occasions, accuracy of the result that can be obtained by the SW method is more desirable, thus the slow computation speed have be overcome with various methods. One of the methods is to realize the

SW algorithm into the reconfigurable hardware where the computation of similarity matrix can be done truly in parallel. In this paper, we propose a hardware design to leverage FPGA attributes in accelerating certain part of SW computation. Since the number of parallel processing that can be implemented within the FPGA device is limited by its resources such as configurable logic block, our design approach use other on-chip resources such as block memory. Also, the processing elements should able to handle the long sequences where multi-iteration computation is required with overlapping operations.

2. Description of the Smith Waterman Algorithm

The Smith-Waterman algorithm is presently being widely used to find local similarity between two biological or genomic sequences. In Smith-Waterman database search, dynamic programming method is used to compare the query sequence to every sequence that is available in the database, and then a score for each comparison is calculated.

The dynamic programming compares a character within a sequence with every possible character within another sequence. The two sequences arranged in a two dimension array in which every matrix cell represents the alignment score of a specific row and column of the two sequences (matrix). The value of each cell depends on the residues from the diagonal, left and upper neighbor cells. Examples of its application suppose that one wishes to compare to compare sequence S ("AGG") to another sequence T ("TTC"). The intermediate values a, b and c (shown in Figure 1) is then used to compute d according to the specified formula. The first operation will be started from the comparator between the two sequences, if they are similar, the value of d will come from a, otherwise the

value of minimum of b plus insertion factor and c plus deletion will be used for d value.

$$d = \min \begin{cases} a & \text{if } S_i = T_j \\ a & \text{if } S_i \neq T_j \\ b + \text{ins} \\ c + \text{del} \end{cases} \quad (1)$$

		S_0	S_1	S_2	S_3	
		$0a$	c^1a	c^2a	c^3a	c^4
T_0	$1b$	$d:b$	$d:b$	$d:b$	d	
	$1a$	$c:a$	$c:a$	$c:a$	c	
T_1	$2b$	$d:b$	$d:b$	$d:b$	d	
	$2a$	$c:a$	$c:a$	$c:a$	c	
T_2	$3b$	$d:b$	$d:b$	$d:b$	d	
	$3a$	$c:a$	$c:a$	$c:a$	c	
T_3	$4b$	$d:b$	$d:b$	$d:b$	d	
	$4a$	$c:a$	$c:a$	$c:a$	c	

Figure 1. Similarity matrix cell equation [4]

By application of equation 1, we can create two dimensions array ($H_{i,j}$), where the array will initialized with $H_{0,j}=0$ and $H_{i,0}=0$, for all i and j for leftmost row and top column. This is referred to as the initialization step. After initialization, the second step is to fill the array cell by using Equation 1, which will fill out all entries.

3. FPGA Implementation

The FPGA implementation of the Smith Waterman algorithm consists of four parts, the processing element, the state machine, the up/down counter, and data control. The processing element computes the array value for each character in the query string.

A state machine follows the processing element to convert the output of the processing elements into an up/down signal. The up/down signals connects to an up/down counter that computes the final edit distance value. The method for receiving and sending data through the system is the final part of the implementation. Figure 2 shows the overall system of the implementation.

An important goal in our work for the hardware was to put maximum numbers of processing elements that fit on an FPGA device. If query sequence length is long that cannot fit into systolic array, multi-iteration scheme have to be adopted.

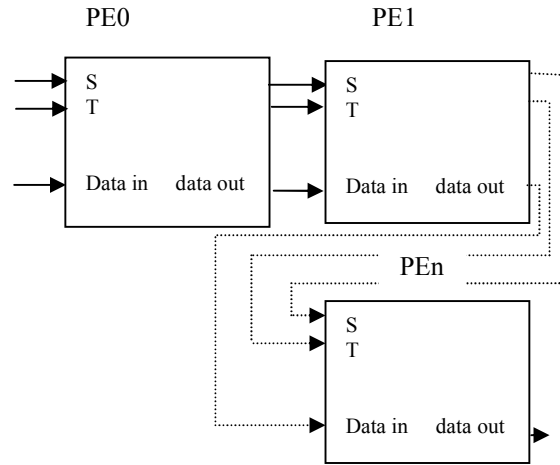


Figure 2. Block of three Processing Elements

Table 1. A, C, T, and G character encoding

A	T	G	C
00	01	10	11

As for the character of the sequence, its nucleotides are represented by a two-bit encodes as given in table1.

Insertions or deletions have a penalty of one and substitutions have a penalty of two. Using these penalty values reduces the amount of the data that needs to be stored in each PE and that to be forwarded to the next PE. Because the insertions and deletion penalty is one, the value in the array cell differs from values in cells b and c by exactly one. Thus the d cell differs from the cell by either zero or two.

Intermediate edit distance values are computed and stored in each PE at every clock cycle. Because adjacent cells vary by exactly one, the least significant bit can be inferred. The intermediate edit distances are computed modulo four, reducing storage space required for the array value in the processing elements. Because successive edit distances vary by exactly one, no information is lost if the values are stored in modulo four.

4. Processing Element (PE) Description

The hardware implementation of the Smith Waterman algorithm has processing elements that

compute the values in each column of the edit distance array. The query character that is used compute the column is folded into the hardware logic. Each gene matching PE fits within three, or less than one Virtex4 CLB.

This implementation of the sequence comparison algorithm uses only local signals. Each processing element requires only the edit distance from the previous column. All of the components in this design are reset synchronously. The database sequence is fed into the PE, resulting in a signal array computation each clock cycle per PE.

Details of a processing element where equation 1 (given in Figure 1) being implemented into the hardware circuit is shown in the Figure 3.

When the transfer signal is high, the sequence S is shifted through the PEs. All the PEs will connect to process diagonal elements in the array in parallel scheme.

Each PE has local memory to store a, b and c values. The PE holds a column of the array in its LUT. The score is calculated, saved and pass to the next PE in the array.

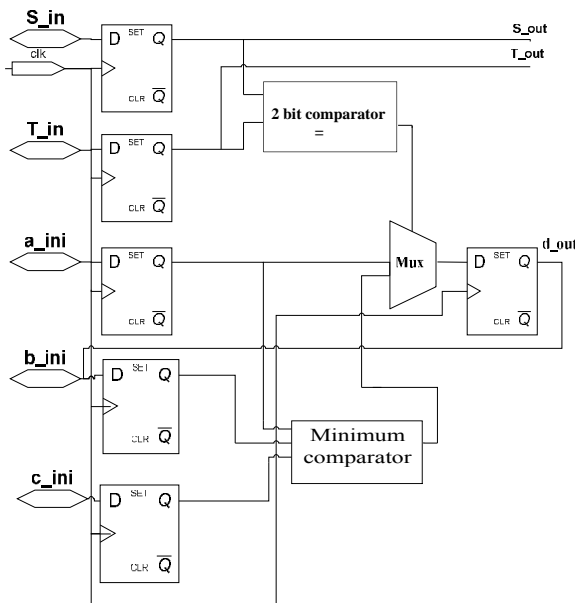


Figure 3. The Smith Waterman processing element (PE)

Our PE design consists of :-

- 1- Flip flops (DFF): - we used 6 DFFs in every PE, the benefits of use it to store the data, two DFFs for storing the two sequence, Source sequence (S_in) and Target sequence (T_in) as input and the another four DFFs used for store

the intermediates data (a_in, b_in and c_in as input and d_out as output).

- 2- Comparator: two comparators, one of them used to compare between the S_in and T_in and the result of this comparator must be either 1 or 0, if the two sequence are equal then the selector equal 1 else equal 0, and another comparator used to compare between all the intermediates values (a_ini+sub, b_ini+ins and c_ini+del) and the output of this comparator will be equal to the minimum of the three input..
- 3- Multiplexer: we used one multiplexer in our design. And with two input for this multiplexer, the first input is a_ini and the second input is the result of the minimum comparator, the selector of this multiplexer is the result of similarity comparator. If the selector equal 1 then the output of this multiplexer equal the value of a_ini else the value of the multiplexer equal to the value of minimum comparator.

5. Simulation and results

The design was synthesized from VHDL using Xilinx software tools. We used ISE 10.1 to write and simulation the VHDL code. The simulation of the comparison between the two sequences (the source and the target) is shown in Figure 4.

The symbols are coded as follows (A=00, T=01, C=10 and G=11). So our input for sequence S or T were implement in the simulation by 2 bit, for example S= A T C we can implement that by S=00, 01, 10 and etc. the same way to implement the T for example we have T=T C G, we can implement that by T= 01,10,11. Our result was implement in the value of the d_out, each PE in our design was implement 1 column in the similarity array.

The time needed to compute a sequence comparison is given by time the running sequence takes to travels the PE [5]. Thus, if the reference sequence has n elements and the data base sequence has m elements, then we need n + m clock cycles to compute the comparison. To obtain a rough estimate of the gain compared to a software solution, suppose that the frequency remains around 50 MHz for larger devices. Considering that we can chain FPGA in order to implement larger sequences, an estimation of the increase in speed provided by the systolic vector compared to the time required by a cluster of workstations.

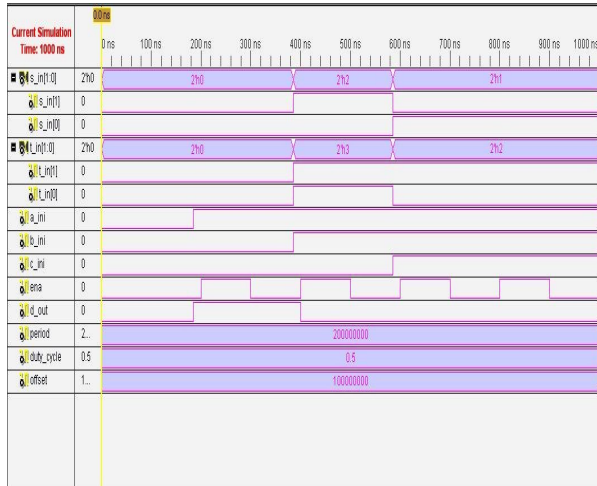


Figure 4. Local sequence alignment simulation of sequences S and T.

The actual increase in speed should be less than this value because it does not include the communication among FPGAs. Even in this case, the gain in speed is can be several orders of magnitude. These values do not take into account the time to recover the alignments or back-tracking.

The result of executing SW algorithm to perform local alignment between two short sequences is given in Figure 5. The figure provides result of similarity matrix cell value (of two sequences: CAGCGTTG and AGGTAC) and also the two sequence being aligned after back-tracking start from cell value of 6 (maximum value of the similarity matrix cell).

6. Conclusion

In this paper, we describe the processing elements that execute concurrently to compute similarity matrix of the Smith Waterman algorithm. The behavior of PE was verified by simulation that each PE can complete its computation in short cycles. The longer the length of the array of PEs, more acceleration can be achieved as more PE can participate in concurrently computing the similarity cell. Our future works are to include multi-iteration capability that will able to handle long sequences. Also to test the delivery of sequences from other computers or network nodes as well as performing backtracking on the similarity matrix cell upon completion of PE processing.

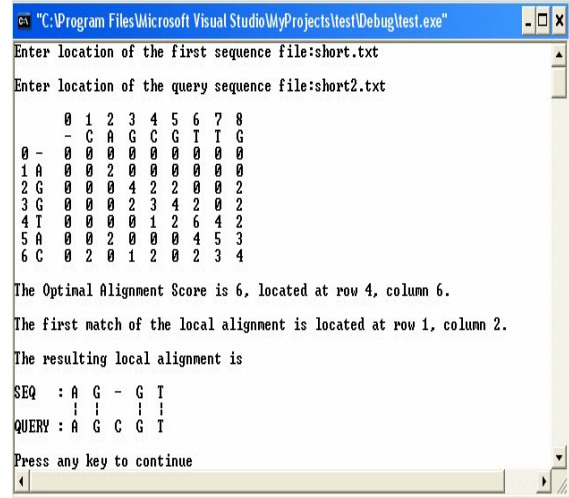


Figure 5. Two sequence aligned by the SW method

7. References

- [1] European Bioinformatics Institute, <http://www.ebi.ac.uk/fasta33>
- [2] National Center for Biotechnology Information, <http://www.ncbi.nlm.nih.gov/blast>
- [3] T.F Smith, M.S. Waterman. "Identification of common molecular subsequence", Journal of Molecular Biology, 147:196-197, 1981.
- [4] P. Leong, M. Leong, O. Cheung, T.Tung, C. Kowk, M. Wong and K.H Lee, "Pilchard – A Reconfigurable Computing platform with memory slot interface". IEEE FCCM, April 2001.
- [5] Richard J. Lipton, Daniel Lopresti, "A systolic array for rapid string comparison, Proceeding of Chapel Hill Conference on VLSI", 363-367, 1985
- [6] P. Zhang, G. Tan, G.R. Gao, "Implementation of the Smith-Waterman Algorithm on a Reconfigurable Supercomputing Platform", ALTERA White Paper, Sept 2007

8. Acknowledgement

Authors wish to thank Malaysian Genome Institute (MGI) for financial assistance provided to support this project.