DESIGN AND ANALYSIS OF HIGH PERFORMANCE AND LOW POWER MATRIX FILLING FOR DNA SEQUENCE ALIGNMENT ACCELERATOR USING ASIC DESIGN FLOW

Nurul Farhana Bt Abd Razak
Department of Electronic
Faculty of Electrical Engineering
Univesiti Teknologi MARA
40450 Shah Alam
Selangor, Malaysia
email: nurulfarhana_ar@yahoo.com

Abstract--- **Effiecient sequence alignment is one of the most important and challenging activities in bioinformatics. Many algorithms have been proposed to perform and accelerate sequence alignment activities. Among them Smith-Waterman (S-W) is the most sensitive (accurate) algorithm. This paper presents a novel approach and analysis of High Performance and Low Power Matrix Filling for DNA Sequence Alignment Accelerator by using ASIC design flow. The objective of this paper is to improve the performance of the DNA sequence alignment and to optimize power reduction of the existing technique by using Smith Waterman (SW) algorithm. The scope of study is by using the matrix filling method which is in parallel implementation of the Smith-Waterman algorithm. This method provides more efficient speedup compared to the traditional sequential implementation but at the same time maintaining the level of sensitivity. The methodology of this paper are using FPGA and Synopsis. This techniques is used to implement the massive parallelism. The design was developed in Verilog HDL coding and synthesized by using LINUX tools. Matrix Cells with a design area 8808.307mm^2 at 40ns clock period is the best design. Thus the power required at this clock period also smaller; dynamic power 111.1415uW and leakage power 212.9538Nw. This is a large improvement over existing designs and improves data throughput by using a ASIC design flow.**

Keywords: Smith-Waterman algorithm, DNA Sequencing, ASIC, FPGA, Matrix Filling.

I. INTRODUCTION

In the past decade there has been an explosive growth of biological data the rapid expansion in digitization of patient biological data [1]. This has led to the emergence of a new area of research: bioinformatics, where powerful computational techniques are used to store, analyze, simulate, and predict biological information. Bioinformatics refers to the storage, analysis, and simulation of biological information and the prediction of experimental outcomes.

The general problem statement was the genomic data at GenBank (the NIH genetic sequence database, an annotated collection of all publicly available DNA sequences) is doubling every six months. Presently, the problem is the proteomic and cellular imaging data is expected to grow even faster. Post-genomic-era bioinformatics will require high-performance computing power of the order of several hundred of teraflops or more. The significance of the study is to improve the performance and optimize the power consumption used. Besides, the bioinformatics sequence is to identify similarities between subsequences of strings as far as possible. The dissimilarity of two sequences of nucleotides or amino acids can be defined as the minimum change required in one to get the other one, among all possible alignments between them [1].

The main objective of this paper are to do the speed optimization, reduced power consumption, and study the effect of high performance and low power techniques on area. In recent few years, FPGAs have emerged as high-performance computing accelerators capable of implementing fine-grained, massively parallelized versions of computationally intensive algorithms. The reprogrammability of FPGAs enables algorithm specific computing architectures to be implemented using the same hardware resource across the range of algorithms.

Alignment of DNA sequences is a necessary step prior to comparison of sequence data. High-speed alignment is needed due to the large size of DNA databases. Smith-Waterman, a standard pattern recognition technique, can be used to perform alignment. Smith-Waterman can be performed by using dynamic programming techniques. Thus, dynamic programming offers the potential for highspeed processing of DNA sequence data.

The Smith-Waterman algorithm is more effective compared to the FASTA and BLAST. Nowadays, this algorithm is well-known algorithm to performe local sequence alignment. 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 overcame with various methods.

A. Smith-Waterman Algorithm

Smith-Waterman algorithm was first proposed by Temple F.Smith and Michael S.Waterman in 1981 [3]. The Needleman-Wunsch algorithm is like variation, while the Smith-Waterman is a dynamic programming algorithm. Such as, it has the desirable property that guaranteed to find the optimal local alignment with respect to the scoring system being used, and includes the substitution matrix and the gap-scoring scheme. The main difference to the Needleman-Wunsch algorithm is negative scoring matrix cells are set to zero, which renders the positively scoring local alignments visible. Backtracing starts at highest scoring matrix cell and it proceed until scoring zero encountered, yielding the highest scoring local alignment.

The S-W algorithm is a sequence alignment technique used to sequence either DNA or protein sequences. This is for determining similar regions between two nucleotide or protien sequences. Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity. The algorithm is based primarily on the class of algorithms which is known as dynamic programming. In Smith-Waterman database search, dynamic programming (DP) 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. When obtaining the local alignment, a matrix *Hi;j* is used to keep track of the degree of similarity between the two sequences to be aligned (*Ai* and *Bj*). Each element of the matrix *Hi;j* is calculated according to the following equation:

$$H_{i,j} = \max \begin{cases} 0 \\ H_{i-1,j-1} + S_{i,j} \\ H_{i-1,j} - d \\ H_{i,j-1} - d \end{cases} \qquad (1)$$

where $S_{i;j}$ is the similarity score of comparing sequence A_i to sequence B_j and d is the penalty for a mismatch. The whole algorithm is divided into three steps:

1. Initialization step
2. Matrix fill step
3. Trace back step

The matrix is first initialized with $H_{0j} = 0$ and $H_{i;0} = 0$, for all i and j. This is referred to as the initialization step. After the initialization, a matrix fill step is carried out using Equation (1), which fills out all entries in the matrix. The final step is the trace back step, where the scores in the matrix are traced back to inspect for optimal local alignment. As an example, the S-W algorithm, is used to compute the optimal local alignment of two sequences (i.e., A = a g g t a c and B = c a g c g t t g). Assume that

$$S_{i,j} = \begin{cases} +2 \ if(A_i = B_j) \\ -1 \ else \end{cases}$$

and $d = 2$. (2)

Table 1 illustrates the calculation of the Dynamic Programming matrix H and the trace back path which is shown in bold digits. The best score found in the matrix is 6, and the corresponding optimal local alignment is

A: a g - g t
B: a g c g t

TABLE 1. THE DYNAMIC PROGRAMMING MATRIX

		c	a	g	c	g	t	t	g
	0	**0**	0	0	0	0	0	0	0
a	0	0	**2**	0	0	0	0	0	0
g	0	0	0	**4**	**2**	2	0	0	2
g	0	0	0	2	3	**4**	2	0	2
t	0	0	0	0	1	2	**6**	4	2
a	0	0	2	0	0	0	4	5	3
c	0	2	0	1	2	0	2	3	4

B. Power Design

Power is drawn from a voltage source attached to the V_{DD} pin(s) of a chip.

Instantaneous Power:

$$P(t) = i_{DD}(t)V_{DD} \qquad (3)$$

Energy:

$$E = \int_0^T P(t)dt = \int_0^T i_{DD}(t)V_{DD}dt \qquad (4)$$

Average Power:

$$P_{avg} = \frac{E}{T} = \frac{1}{T}\int_0^T i_{DD}(t)V_{DD}dt \qquad (5)$$

1) Dynamic Power

Dynamic power is required to charge and discharge load capacitances when transistors switch. In one cycle they are involves a rising and falling output. Thus, during rising output, charge $Q = CV_{DD}$ is required. While, on falling output, charge is dumped to GND. This repeats Tf_{sw} times over an interval of T.

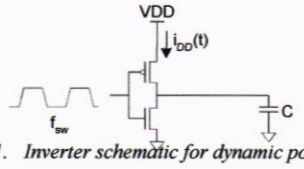

Figure 1. Inverter schematic for dynamic power analysis.

To derive the equation of the dynamic power:

$$P_{\text{dynamic}} = \frac{1}{T}\int_0^T i_{DD}(t)V_{DD}dt \qquad (6)$$

$$= \frac{V_{DD}}{T}\int_0^T i_{DD}(t)dt$$

$$= \frac{V_{DD}}{T}\left[Tf_{\text{sw}}CV_{DD}\right]$$

$$= CV_{DD}{}^2 f_{\text{sw}}$$

Thus, from the equation (6), in order to reduce the dynamic power, the capacitance, frequency or VDD must be reduced varied. Due of the technique to reduce the dynamic power is by varying the clock frequency.

2) Static power

Static power is consumed even when chip is quiescent. The ratioed circuit will dissipate power between ON transistors due to data contention.

In order to design low power, these criteria must be taken into consideration.

$$P_{\text{static}} = I_{\text{static}}V_{DD} \qquad (7)$$

In order to reduce dynamic power:

i. a: clock gating, sleep mode

ii. C: small transistors (esp. on clock), short wires

iii. V_{DD}: lowest suitable voltage

iv. f: lowest suitable frequency

In order to reduce static power:

i. Selectively use ratioed circuits

ii. Selectively use low V_t devices

iii. Leakage reduction (stacked devices, body bias, and low temperature).

II. METHODOLOGY

There are 2 difference method in implementing the design. The first design is coded and simulated on Xilinx FPGA design. Secondly, the design is reverified synthesis and implemented on Synopsis ASIC Design flow.

A. FPGA design implementation

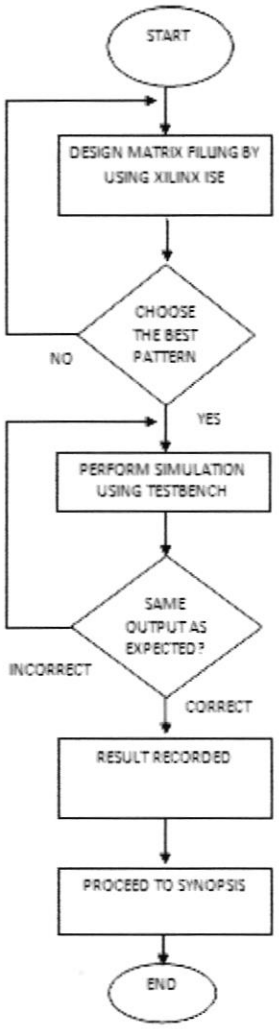

Figure 2. Flow chart for designing matrix filling on FPGA

The design implementation of the Smith Waterman algorithm consists of four parts which are the initialization, processing unit, comparator and memory. The processing unit computes the array value for each character in the query string. The processing unit will be compared between query and subject and stored the result into memory. Figure 3 shows the block diagram of matrix filling module.

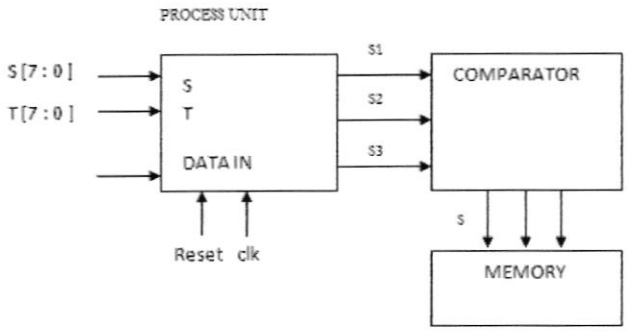

Figure 3. Block diagram of matrix filling module.

TABLE 2. A, C, T, and G character encoding

A	T	G	C
00	01	10	11

Table 2 shows the two bit codes that represent the nucleotides.

B. ASIC design implementation (synopsis)

There are several steps in ASIC design implementation. Firstly, the code is verified in verilog compiler simulator (VCS) in order to check the error. Then, synthesized the code using Design Compiler and proceed to physical implementation using ICC. Lastly, the result is generated by statis timing analysis.

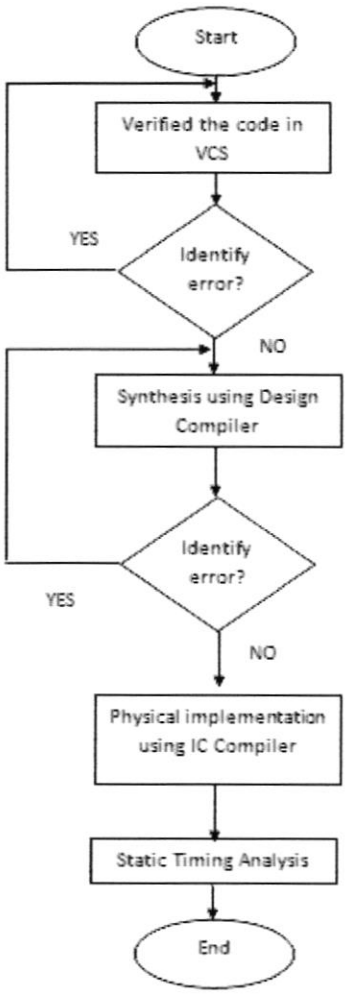

Figure 4. Flowchart for designing matrix filling on ASIC.

III. RESULT AND DISCUSSION

A. FPGA result

The design was synthesized from Verilog using Xilinx software tools. In this paper, the ISE 11.1 has been used to write and simulate the Verilog code. The simulation of the compassion between two sequences (the source and the target) is shown in Figure 5.

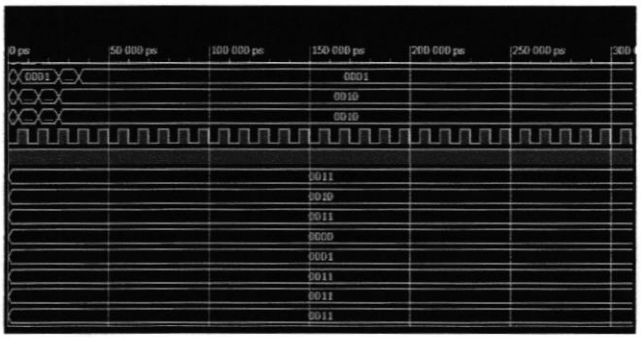

Figure 5. Local sequence alignment simulation of sequences S and T.

The symbols are coded as follows (A=00, T=01, C=10, and G=11). So the input for sequences S or T were implement in the simulation by 2 bit, for example S= A T C, it can implement that by S=00, 01, 10 and etc. The same way to implement the T for example T=T C G, it can implement that by T= 01,10,11.

The time needed to compute a sequence comparison is given by time the running sequence takes to travels the Processing unit[2]. Thus, if the reference sequence has n elements comparison and the data base sequence has m elements, then it needs 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.

The number of slices and flip-flop are depending on the number of matrix cells. Thus, Table 3 shows the relationship between the number of slices and flip-flop.

TABLE 3. Effect number of Slice and Flip-flop on number of matrix cell.

Matrix	Matrixes Cell	Number of Logic	Number of Flip-flop
2x2	4	29	8
3x3	9	297	21
4x4	16	723	40

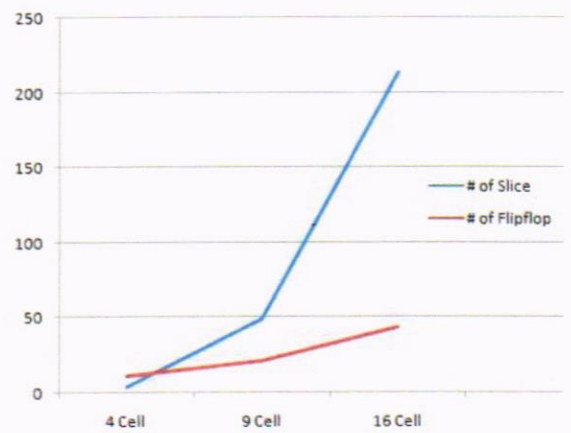

Figure 6: Graph number of slice and flip-flop for matrix filling module.

B. *Synopsis result*

The Design Compiler (DC) is a synthesis tool that takes a RTL [Register Transfer Logic] hardware description [design written in Verilog], and standard cell library as input and the resulting output would be a technology dependent gate-level-netlist.

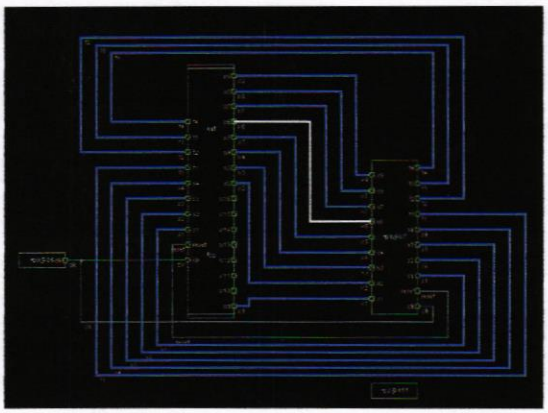

Figure 7. Schematic view of synthesized top level of the design

Figure 7 shows the schematic view after synthesized the code in Verilog Compiler Simulator. It shows the connection of input and output pins of the design. The input pins are clock (clk), reset, sample (S1,S2,S3,S4) and target(T1,T2,T3,T4). While output has 16 pins which are X1 until X16.

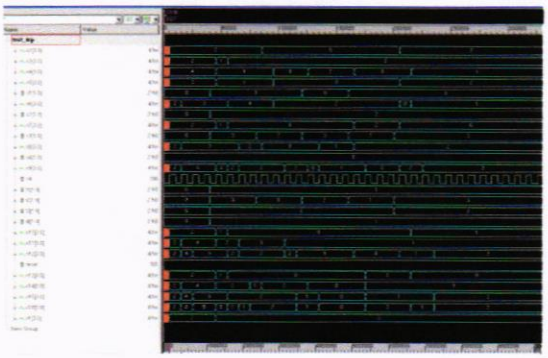

Figure 8. DVE simulation waveform

Figure 8 shows the simulation waveform simulated from design vision (DVE). The DVE provides a graphical user to debug the design. From figure 8, it is proved the synopsis output is equal to the FPGA's output.

TABLE 4. Effect of varying frequency on design area

Clock period (ns)	Design area (um^2)	Slack dc_timing_max (ns)
5	8977.954	34.46
10	8834.918	29.46
15	8818.286	24.63
20	8818.286	19.63
25	8818.286	14.63

30	8818.286	9.63
35	8808.307	4.49
40	8808.307	2.25
41	8808.307	-1.03

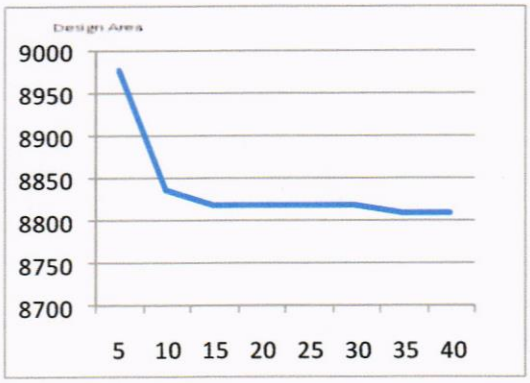

Figure 9. Graph of clock period versus design area.

As shown in Figure 9, the design area is inversely proportional to clock period. It shows that the larger area will reduce the compiling process time taken. In this case the best design is at 40ns clock period. This is because, during this time has a small area and have a reasonable time slack. Table 4 also shows at 41ns clock period, the results showed a negative slack time, and it violated. Realistic specification is important, because unrealistic constraints might result in excess area, increased power and degrading in timing.

TABLE 5: The relationship between dynamic power and clock period.

Clock period (ns)	Dynamic power (uW)	Leakage power (nW)
5	902.6910	243.3606
10	450.5993	240.2271
15	300.1035	236.8609
20	225.0776	236.8609
25	180.0621	236.8609
30	150.0517	236.8609
35	128.7286	235.1861
40	112.6376	235.1861

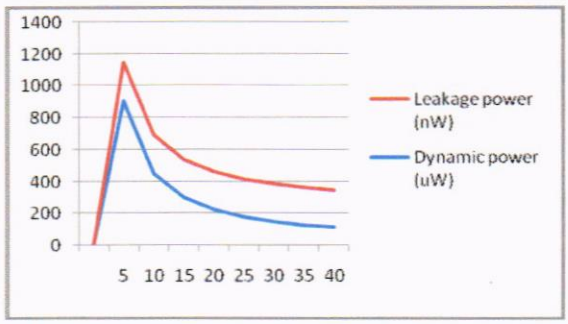

Figure 10. Graph of clock period versus dynamic power and leakage power.

Figure 10 shows, additional clock period will result in reduced power. This proved that the results follows dynamic equation(6) because the equation indicates that frequency is proportional to dynamic power. Leakage power is slightly decreased and remain constant at certain clock cycle.

C. IC Compiler (ICC) Report

IC Compiler is represented the abstract version of layout. Every desired logic function in the standard cell library will have both a layout and abstract view and also contain timing information about the function such as cell delay and input pin capacitance which is used to calculated output loads.

TABLE 6. Power report in IC Compiler

Clock period (ns)	Dynamic power (uW)	Leakage power (nW)
5	892.8958	214.8681
10	442.1232	215.6031
15	295.8026	213.3234
20	222.2299	215.2308
25	177.7766	214.6538
30	148.2143	214.6538
35	126.9898	212.9538
40	111.1415	212.9538

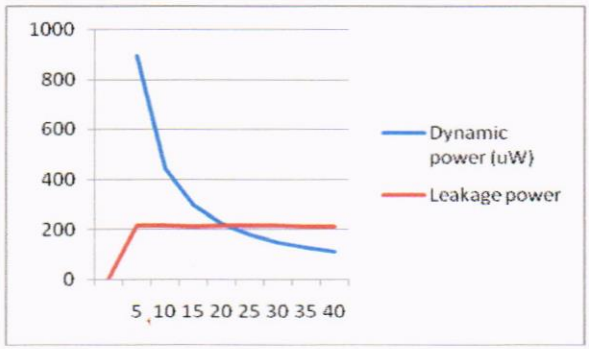

Figure 11. Graph of clock period versus dynamic power and leakage power in IC Compiler

Figure 12 shows dynamic power decreased when clock period increased while leakage power mostly remain the same value. In this case, it proved that the result follows dynamic power equation(6). The value of frequency is proportional to dynamic power.

TABLE 7. Timing analysis in DC, ICC nad Prime Time (PT)

Clock period (ns)	Tpd (ns)	Slack dc_timing_max (ns)	Slack icc_timing_max (ns)	Slack pt_timing_max (ns)
2	4.32	-0.13	-1.12	5.04
5	4.32	2.25	3.20	5.04
10	4.32	4.49	6.50	5.04
15	4.32	9.63	11.56	5.04
20	4.32	14.63	16.62	5.04
25	4.32	19.63	21.56	5.04
30	4.32	24.63	26.60	5.04
35	4.49	29.46	31.54	5.04
40	4.49	34.46	36.50	5.04

Tpd = critical path length

Table 7 shows the timing in Design Compiler, IC Compiler and Prime Time. The proporgation delay shows the values are remain constant until 35ns clock period. It indicates critical path is 4.49ns which is the slowest logic path between any two registers. There is also stated the maximum path depend on the clock cycle. Besides, the datapath also showed the negative values at 2ns clock period where it is violated. However, the final register file of the design has a IC Compiler timing max of 36.50 ns at the 40ns clock period.

IV. CONCLUSION

In conclusion, the objectives of this paper is fully achieved. The FPGA's result showed the reduction of number clock cycle from 16 clock cycles to 3 clock cycle. The synopsis results show there is an improvement of the design where design area of matrix cells is 8808.307um^2 at 40ns clock period. Thus it reduced the power consumption where dynamic power is 111.1415uW and leakage power is 212.9538nW. On top of that, reduction in power also causes the reduction of design area. This is a large improvement over existing designs and improves data throughput by using a ASIC design flow. The recommendation for future developement are reducing in number of clock cycle into 1 clock cycle and reducing power consumption of the design. On top of that, the area must be smaller than existing area.

ACKNOWLEDGEMENT

I would like to express my deep and sincere gratitude to my supervisor, Pn Norhazlin Kahirudin, co-supervisor En Abdul Karimi Halim and En Syed Abdul Mutalib Al Junid. Their wide knowledge and thier logical way of thinking have been great value for me. Their understanding, encouraging and personal guidance have provided a good basis for this technical paper. I owe my loving thanks to my parent. Without their encouragement and understanding it would have been impossible for me to finish this work.

REFERENCES

[1] Keith Regester, Jong-Ho Byun, Arindam Mukherjee, Arun Ravindran, "Implementing Bioinformatics Algorithms on Nallatech-Configurable Multi-FPGA Systems :1-4, 2005.

[2] Richard J. Lipton, Daniel Lopresti, "A systolic array for rapid string comparison, Proceeding of Chapel Hill Conference on VLSI", 363-367, 1985.

[3] Smith, Temple F.; and Waterman, Michael S. (1981). "Identification of Common Molecular Subsequences". *Journal of Molecular Biology* 147: 195 197. doi:10.1016/0022-2836(81)90087-5.

[4] Farouq H. Humed, Razali Jidin, Sri Noraima Othman,"Implementing Smith Waterman's Similarity Matrix Computations on Reconfigurable Logic Hardware". Department of Electrical and Communication Engineering – College of Engineering *Universiti Tenaga Nasional, Quantum Beez Sdn Bhd-Block D-2, UPM-MTDC.*

[5] David Sankoff, "The Early Introduction in Dynamic Programming in Computational Biology," Bioinformatics, vol. 16, pp. 41-47, Jan, 2000.