Design and Analysis of 4x4 Smith Waterman's Based DNA Sequences Alignment Accelerator Using Multicore Architecture

Nur Fariza binti Mohd Najib
Faculty of Electrical Engineering
Universiti Teknologi MARA Malaysia
40450 Shah Alam, Selangor, Malaysia
nurfarizamohdnajib@yahoo.com

Abstract- **This paper presents the design and analysis of 4x4 Smith Waterman's based DNA sequences alignment accelerator using multicore architecture. The first objective of this paper is to design 4 x 4 matrix filling module of DNA sequence alignment accelerator. The second objective is to combine matrix filling module with traceback and reconstruction module design by previous student in order to create single core. The third objective is to construct modules that consist of 2, 4 and 8 cores. The fourth objective is to simulate and verify the functionality of the new matrix filing, single and multicore modules on Xilinx FPGA design flow. The last objective is to verify, optimize and analyze the multicore modules using VCS, DC, ICC and PT. This paper focuses on the timing analysis, power and area of multicores architecture using Smith-Waterman algorithm. To achieve a higher performance with low latency and high throughput data has become a serious concern for today's DNA laboratory as the increase of number of DNA database all around the world. Researchers may have done different kind of architecture such as pipeline, vector, multicycle to boost up the speed performance of DNA sequences alignment. All design is written in Verilog language and the result is verified on Synopsys EDA tool which are Verilog Compiler Simulator (VCS), Design Compiler (DC) and ICCompiler (ICC).**

Keywords: ASIC, multicore architecture analysis, Synopsys EDA tools and Smith Waterman algorithm.

I. INTRODUCTION

DNA is a self-replicating material present in nearly all living organisms as the main constituent of chromosomes. It is the carrier of genetic information for all cells. Scientists realized that DNA molecules have a vertebra backbone unit consists of sugar (deoxyribose) and one phosphate. Each vertebra unit is attached by nitrogenous bases that are adenine (A), guanine (G), cytosine (C) and thymine (T) [1].

There are many algorithms for DNA sequence alignment. For example FASTA and BLAST [2]. FASTA (Fast Alignment Search Tool) and BLAST (Basic Local Alignment Tool) are most commonly used algorithm to obtain high performance system but both have low sensitivity to obtain the correct result compared to Smith-Waterman algorithm. FASTA and BLAST may have the speed but in term of accuracy of DNA sequencing, Smith Waterman is more preferable. However, the implementation of this algorithm is quite complicated and challenging due to limited space memory and speed for long DNA sequence.

Smith-Waterman algorithm is one of the most basic and well known algorithm that performs local sequence alignment for DNA [3]. The algorithm was first proposed by Temple Smith and Michael Waterman in 1981. Like the Needleman-Wunsch algorithm, of which it is a variation, Smith-Waterman is a dynamic programming algorithm [4]. This algorithm used comparison technique of two DNA (A,C,G,T) sequences functions based on local alignment in order to find the optimal local alignment of that two sequences [5].

In order to improve the efficiency of computer platforms, multithreading processor is introduced as industry-standard servers and the overwhelming majority of network applications can take advantage of the additional processors, multiple software threads, and multitasked computing environments [6]. All these advantages have enabled organizations to scale network applications for a greater performance. The next logical step for multiprocessing advancement is expected to come in the form of multiple logical processing units, or processor cores, within a single chip. Multicore processors are good because they have multiple memory, I/O, and storage. A multicore processor system consists of multiple processors and a method for communication between the core processors. The multicore processor is the separation of multiple cores on the same chip. This architecture is used to run more tasks simultaneously by dividing the tasks among the cores [7]. The increase in frequency on a single core demands faster switching transistors with higher operating voltages hence increase power dissipation.

II. SMITH-WATERMAN ALGORITHM

There are three modules on finding the optimal sequence in Smith-Waterman algorithm and the first module is to fill in the dynamic programming matrix, the second is to find the maximum score from that matrix and the last is to trace back the maximum path from the maximum score to find the optimal local alignment [3]. Consider the two comparison of DNA sequences are S for sample and T for target. The dynamic programming for matrix filling module will be based on this sample and target scoring according to this equation:

$$\begin{aligned} &\text{For } 0 \le i \le M,\ 0 \le j \le N, \\ &\quad Di0 = D0j = 0 \\ &\text{For } 1 \le i \le M \text{ and } 1 \le j \le N \\ &\quad Dij = 0 \text{ or} \end{aligned} \tag{1}$$

$$D_{ij} = \max \begin{cases} D_{(i-1),(j-1)} + Sbt_{(i,j)} \\ D_{(i-1),(j)} + d \\ D_{(i),(j-1)} + d \end{cases} \tag{2}$$

Where
d = penalty
Sbt = substitution matrix
i = matrix cell row of search sequence
j = matrix cell column of target sequence
M = maximum length of search sequence
N = maximum length of target sequence
D_{ij} = dynamic matrix cell

The equation (2) above shows that, comparison match and mismatch at every horizontal and vertical move will be given penalty of -1. But for every match comparison in diagonal will be added +2 and for mismatch in diagonal will be given penalty of -1. There is no negative number in Smith-Waterman algorithm, so the smallest scoring in matrix filling will be converted into zero. Consider S = A-C-G-T and T = C-A-G-T. The comparison results for both sample and target are shown in Fig. 1.

		A	C	G	T
	0	0	0	0	0
C	0	0	2	1	0
A	0	2	1	1	0
G	0	1	1	3	2
T	0	0	0	2	5

Figure 1. Matrix filling scoring

The second module is to find the optimal and maximum path by finding the maximum score and trace it back until the zero value is obtained. There are three rules to consider in finding the maximum path in traceback module. The first rule is to find the highest value of scoring inside the matrix. For second rule, from that highest score value, the next instruction is to compare the vertical, horizontal and diagonal scoring. If the value for horizontal and vertical are equal, the path will follow the diagonal score. If all three values are equal, the path will follow the diagonal score. The last rule is, whenever the path meet a zero value as diagonal score, the path will be automatically stopped. Fig. 2 shows the correct maximum path sequences that consider all rules as mentioned before.

		A	C	G	T
	0	0	0	0	0
C	0	0	2	1	0
A	0	2	1	1	0
G	0	1	1	3	2
T	0	0	0	2	5

Figure 2. The optimal path from trace back module

After finding the maximum path from traceback module, the sequence of sample and target will be reconstructed back at the last module which is known as reconstruction. At reconstruction module, there will be an insertion of gap depending on match and mismatch of sample and target DNA along the maximum path. In Fig. 2, after reconstruction modules, the sequence will change into S = C-G-T and T = A-G-T.

III. METHODOLOGY

The first step of designing matrix filling module is to assign the first column and first row with zero. Then the row and column for each element in 4x4 matrix is represented by r1 to r16 as illustrated in Fig. 3. The value of scoring is based on comparison for each diagonal, vertical and horizontal in every element in the matrix. The input for sample and target is compared starting with r1 to r4, followed by r5 to r8, r9 to r12 and lastly r13 to r16. All comparison operations run continuously until every element is fulfilled by score.

		S[7:6]	S[5:4]	S[3:2]	S[1:0]
	0	0	0	0	0
T[7:6]	0	r1 →	r2 →	r3 →	r4
T[5:4]	0	r5 →	r6 →	r7 →	r8
T[3:2]	0	r9 →	r10 →	r11 →	r12
T[1:0]	0	r13 →	r14 →	r15 →	r16

Figure 3. Matrix filling representation

The output from matrix filling module is then transferred to traceback and recombination module that has been designed by previous student. At traceback and reconstruction module, the scoring is being compared again and checked for the longest path. That longest path will be reconstructed back with or without gap representation depending on the scoring produce by matrix filling. This combination of two modules

will become the single core in this project. Fig. 4 represents the single core block diagram.

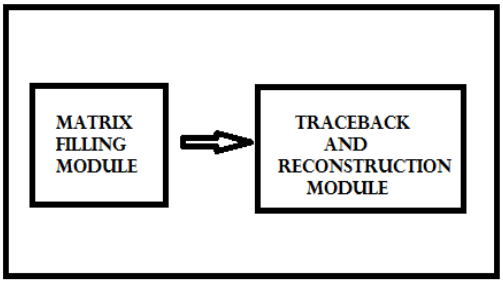

Figure 4. Single core block diagram

TABLE I. DNA SEQUENCE VARIABLES

Character	Input Data	Output Data
A	2'b 00	3'b 000
C	2'b 01	3'b 001
G	2'b 10	3'b 010
T	2'b 11	3'b 011
Gap	-	3'b 100

In matrix filling module, the comparison of DNA scoring is based on two bits binary number. In traceback and reconstruction module, the data must be converted into three bits binary number with the addition of gap insertion. The characters of each DNA sequence representative are tabulated in Table I. In order to design the multicore architecture, the single core block diagram is multiplied by the number of core. For this project, multicores are designed in two, four and eight cores. The clock remains synchronous all the time. Fig. 5 shows the block diagram for eight core of multicore.

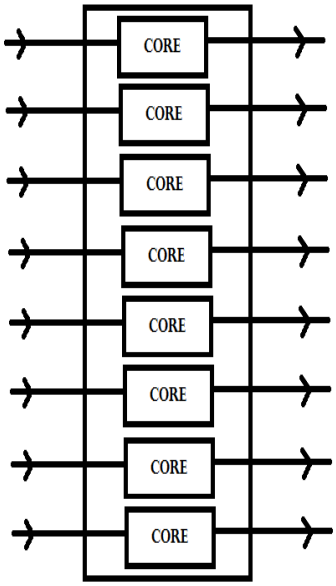

Figure 5. The block diagram for 8 cores

This project began with writing the verilog coding using Xilinx for matrix filling module and the coding is checked for any syntax error. After all syntax are clear from any error, the coding is then synthesized to view the RTL schematic circuit. Then, the verilog coding is examined by testbench to view the output waveform. This output waveform is checked to verify its consistency with the input. The finalize verilog coding is compiled using Verilog Compiler Simulator (VCS) for debugging and viewing the waveform. The RTL coding is simulated and verified the output with respect to its input. The next synthesis is the hardware synthesis using Synopsys Design Compiler (DC). At this stage, the RTL design is converted into an optimized Gate Level Netlist. It is the structural representation of standard cells based on the cells in the standard cell library. The RTL hardware description and a standard cell library are taken as the input to produce an output of gate-level netlist. The DC synthesis tools will attempt to meet the constraints specifications such as timing, area and power by calculating the cost of various implementation [8]. All the reports on timing, power and area analysis of current design are stated at the end of the DC analysis. The clock cycle in design constraint is changed according to the desired frequency. The timing analysis is used to find the operating frequency range that meet the timing requirement of the design module. The analysis procedure is continued by physical implementation in IC Compiler (ICC). There are three stages in this compiler which are floorplanning, placement and routing. Floorplanning is to map the design to the physical description by minimizing area and timing. In placement step, the standard cells are defined to a particular position in a row while the last step is routing that is used to build the connection between all the blocks and the nets. The last method is to optimize the design with the Prime Time (PT) analysis. This PT is the standard time off for gate level static timing analysis in the industry. Once PT analysis met the requirement, the design is capable to work if there is no other damage during fabrication. Fig. 6 shows the flowchart for ASIC design flow that illustrates the whole process of this project.

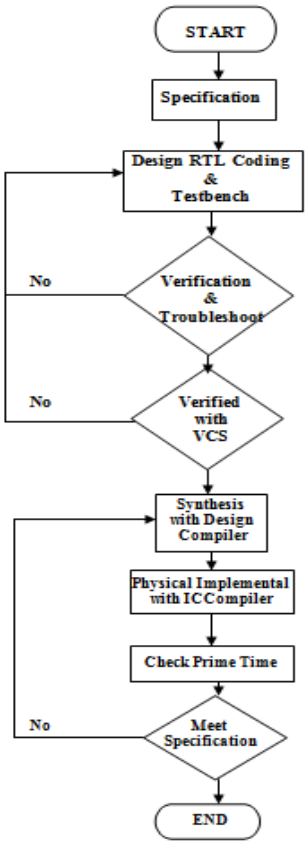

Figure 6. ASIC design flow

IV. Result and Discussion

All input and output for the simulation waveform for all single, two cores, four cores and eight cores will be based on Table II and all the waveforms are shown in appendices from Fig. 16 until Fig. 19. The score for each input on sample and target are tabulated in Table II. Fig. 7 shows the schematic of the top single core module generated by Xilinx. Fig. 8 to Fig. 11 shows the RTL schematic for single and multicore generated by Design Vision.

TABLE II: Matrix Filling Scoring Result

S		T		Score
Code	*Binary*	*Code*	*Binary*	
ACAC	00010001	AGCA	00100100	9
CGTA	01101100	CTAG	01110010	9
ATTA	00111100	ACTC	00011101	7
AGCG	00100110	GGCA	10100100	6
GGAA	10100000	CAAA	01000000	6
TGCA	11100100	TACA	11000100	11
CTAG	01110010	CTAA	01110000	12
TGCC	11100101	TAAG	11000010	2
ACAT	00010011	ACAT	00010011	20

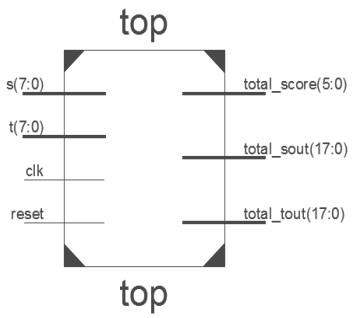

Figure 7. RTL schematic for single top core generated by Xilinx simulation

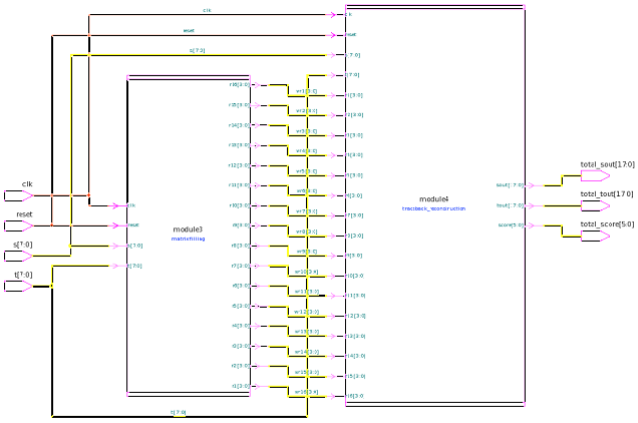

Figure 8. RTL schematic for single core generated by Design Vision

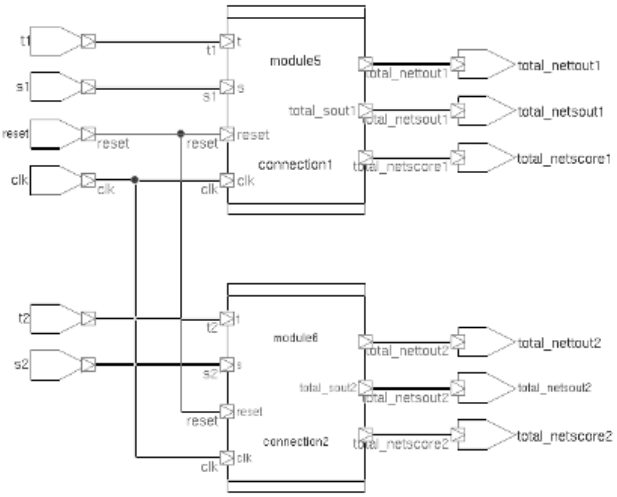

Figure 9. RTL schematic for 2 cores generated by Design Vision

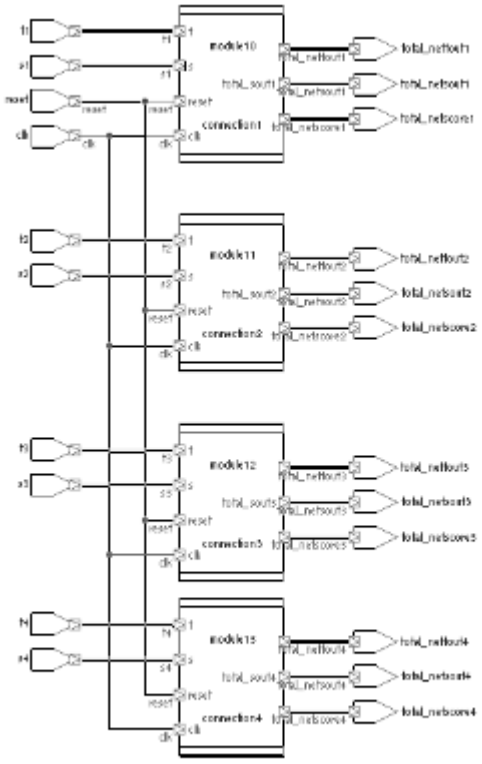

Figure 10. RTL schematic for 4 cores generated by Design Vision

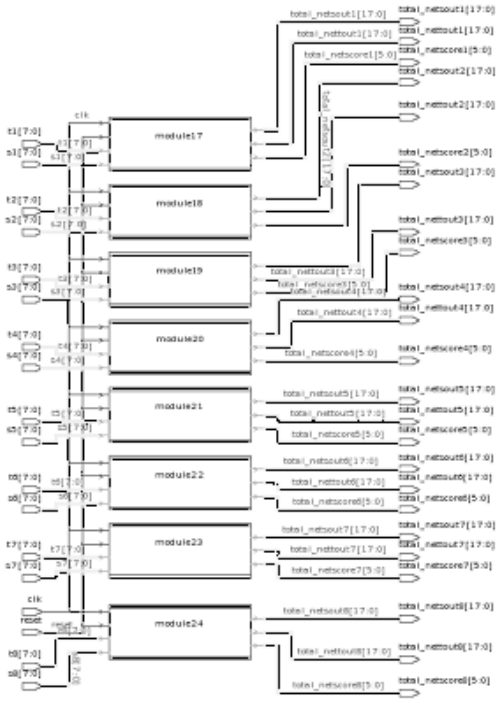

Figure 11. RTL schematic for 8 cores generated by Design Vision

All reports on power and timing analysis on DC analysis are based on wire load model but for ICC analysis the reports are based on actual device model. A wire load model allows the tool to estimate the effect of wire length, resistance, and capacitances, area of nets, wire delays, circuit speeds, area and timing information for each standard cell. DC uses this information to optimize the synthesis process [8]. For any design to work at a specific speed, timing analysis has to be performed to check whether the design is meeting the speed requirement. It is use to check the design for all possible timing violations for example, set up time and hold time.

TABLE III: SIMULATION RESULT ON TIMING ANALYSIS BY VARYING FREQUENCY FOR SINGLE CORE

Clk Cycle (ns)	DC (ns)		ICC (ns)		PT (ns)	
	max	*min*	*max*	*min*	*max*	*min*
5	0.00	0.00	-0.24	0.21	-0.04	0.10
10	0.86	0.00	0.76	0.23	1.05	0.16
15	1.99	0.00	2.02	0.25	1.80	0.16
20	6.60	0.00	4.16	0.27	4.39	0.16
25	11.60	0.00	9.02	0.25	9.33	0.16
50	36.61	0.01	34.00	0.25	34.33	0.16

Results in Table III show the timing analysis for ICC and PT at 5ns because it gives negative slack at maximum time. It shows the violation happened at 5ns. At 10ns to 50ns there is no time violation. PT analyzes the timing delays in the design and flags violation that must be corrected.

TABLE IV: SIMULATION RESULT ON TIMING ANALYSIS BY VARYING FREQUENCY FOR 2 CORES

Clk Cycle (ns)	DC (ns)		ICC (ns)		PT (ns)	
	max	*min*	*max*	*min*	*max*	*min*
5	0.00	-0.08	-0.11	0.19	0.03	0.06
10	0.85	0.00	0.17	0.23	0.36	0.03
15	1.99	0.00	0.15	0.23	0.71	0.14
20	6.56	0.00	3.66	0.24	3.91	0.12
25	11.55	0.00	8.46	0.24	8.98	0.14
50	36.55	0.00	33.33	0.24	34.01	0.13

Table IV shows the violation occurred at 5ns because it gives negative slack at minimum DC analysis and maximum ICC analysis. There are no time violations at 10ns to 50ns. Although it passes PT both max and min which are positive slack, it is not advisable to work because the positive output is very marginal.

TABLE V. SIMULATION RESULT ON TIMING ANALYSIS BY VARYING FREQUENCY FOR 4 CORES

Clk Cycle (ns)	DC (ns)		ICC (ns)		PT (ns)	
	max	*min*	*max*	*min*	*max*	*min*
5	0.00	-0.27	-0.11	0.19	-0.15	-0.05
10	0.85	0.00	0.33	0.24	0.80	0.04
15	1.25	0.00	1.01	0.24	0.90	0.04
20	6.55	0.00	3.86	0.24	4.38	0.04
25	11.55	0.00	8.97	0.24	9.35	0.03
50	36.53	0.00	33.61	0.24	34.26	0.03

Results in Table V show the violation timing analysis happened at 5ns for ICC and PT since it gives negative slack at maximum time. It shows the device could not work at 5ns. At 10ns to 50ns there is no time violation since the timing analysis gives the positive slack.

TABLE VI. SIMULATION RESULT ON TIMING ANALYSIS BY VARYING FREQUENCY FOR 8 CORES

Clk Cycle (ns)	DC (ns)		ICC (ns)		PT (ns)	
	max	*min*	*max*	*min*	*max*	*min*
5	0.00	-0.29	-0.27	0.19	-0.13	-0.07
10	0.85	0.00	-0.07	0.24	0.36	0.03
15	1.99	0.00	0.04	0.24	0.46	0.02
20	6.55	0.00	3.86	0.24	3.91	0.12
25	11.55	0.00	8.53	0.25	9.03	0.02
50	36.53	0.00	33.84	0.24	37.68	0.04

Timing violation happened at both 5ns and 10ns in Table VI because it illustrated negative slack in DC, ICC and PT analysis for 5ns and maximum ICC analysis for 10ns. Larger negative slack values mean that the design is missing the desired clock frequency by a greater amount. It shows the possibility that many paths are too slow. But if the negative slack is a small negative number, it indicates that only a few paths are too slow.

TABLE VII. SIMULATION RESULT ON POWER ANALYSIS BY VARYING FREQUENCY FOR SINGLE CORE

Clk Cycle (ns)	DC (W)		ICC (W)	
	Dynamic	*Static*	*Dynamic*	*Static*
5	6.8588m	3.3527µ	10.1462m	3.7274µ
10	3.2838m	2.9696µ	4.3016m	2.8445µ
15	2.2066m	2.9957µ	2.9396m	2.8602µ
20	1.6832m	2.9997µ	2.1964m	2.8634µ
25	1.3466m	2.9985µ	1.7824m	2.8667µ
50	673.2762µ	2.9963µ	890.3331µ	2.8680µ

TABLE VIII:. SIMULATION RESULT ON POWER ANALYSIS BY VARYING FREQUENCY FOR 2 CORES

Clk Cycle (ns)	DC (W)		ICC (W)	
	Dynamic	*Static*	*Dynamic*	*Static*
5	13.8569m	6.6823µ	20.4290m	7.3330µ
10	6.6047m	5.9396µ	8.7233m	5.6990µ
15	4.4487m	5.9994µ	5.8400m	5.7294µ
20	3.3886m	5.9980 µ	4.4742m	5.7375 µ
25	2.7116m	5.9998 µ	3.5380m	5.7315 µ
50	1.3556m	5.9933 µ	1.7637m	5.7344 µ

TABLE IX. SIMULATION RESULT ON POWER ANALYSIS BY VARYING FREQUENCY FOR 4 CORES

Clk Cycle (ns)	DC (W)		ICC (W)	
	Dynamic	*Static*	*Dynamic*	*Static*
5	27.3373m	13.3285µ	40.3304m	14.6303µ
10	13.1810m	11.8853µ	17.4706m	11.3902µ
15	10.3373m	11.9285µ	12.3241m	11.4303µ
20	6.7495m	11.9895µ	8.9328m	11.4653µ
25	5.4016m	11.9920µ	7.1275m	11.4769µ
50	2.6998m	11.9978µ	3.5644m	11.4788µ

TABLE X. SIMULATION RESULT ON POWER ANALYSIS BY VARYING FREQUENCY FOR 8 CORES

Clk Cycle (ns)	DC (W)		ICC (W)	
	Dynamic	*Static*	*Dynamic*	*Static*
5	56.2640m	27.0311µ	81.5181m	29.2874µ
10	26.3182m	23.7660µ	34.9675m	22.7760µ
15	17.7645m	24.0819µ	23.7891m	22.9694µ
20	13.4826m	23.9785µ	17.8261m	22.9431µ
25	10.7893m	23.9772µ	14.2614m	22.9429µ
50	5.3920m	23.9527µ	7.1172m	22.9550µ

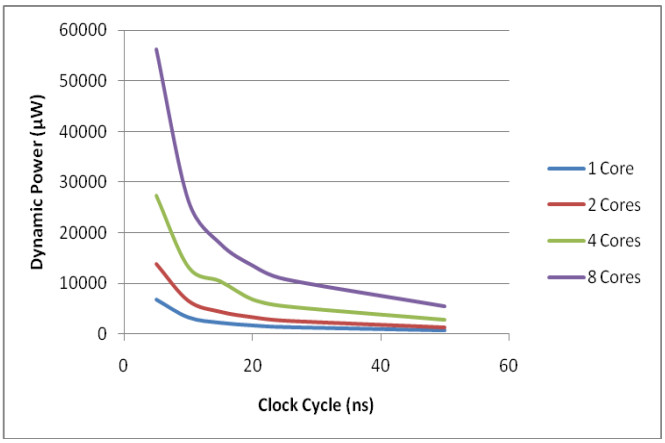

Figure 12. Dynamic power consumption in DC

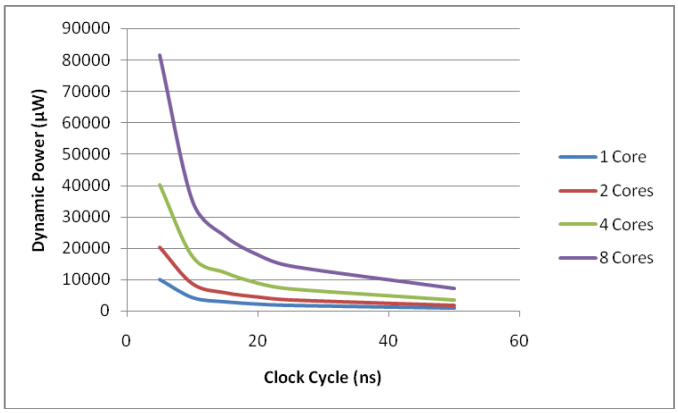

Figure 13. Dynamic power consumption in ICC

Results in Table VII, VIII, XI and X are illustrated by graph in Fig. 11 and Fig. 12. The graph for dynamic power in DC analysis shows that power consumption in 8 cores is higher than other cores. High number of cores represents high number of gates in chips. Dynamic powers for all cores are rapidly decreasing from 5ns to 10ns. The graph shows that at low frequency, the power consumption is also low. Power consumption at two cores is twice higher than at single core. The results on power consumption at four cores are twice higher of two cores and four times higher than power consumption at single core. The same results are obtained for eight cores power consumption. Eight cores design consumed twice higher power consumption of four cores, four times higher than two cores and eight times higher than single core. The increase number of cores will increase the number of logic gates. Increase number of switching activity will also increase the dynamic power. The value of dynamic power in DC is different from ICC analysis because in DC the power analysis is done on the wire load model that does not contain information about some type of delay.

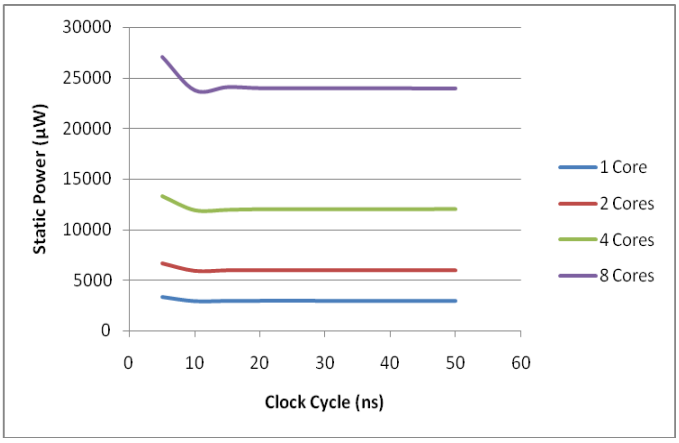

Figure 14. Static power in DC

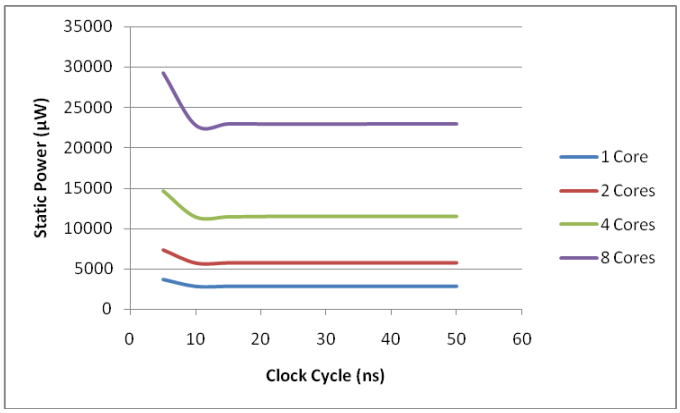

Figure 15. Static power in ICC

Static powers for both cores are barely constant at 10ns to 50ns. There are only small changes in static power in DC and ICC analysis. Since the technology is being used in 0.18µm^2 this static power is not considered as major power consumption. But in smaller size technology such as 19 nano and below, this static power will be the main concern in all design.

TABLE XI. SIMULATION RESULT ON DESIGN AREA FOR SINGLE CORE

Clk Cycle (ns)	DC (µm^2)	ICC (µm^2)
5	107572.451178	113500.096024
10	102928.796599	99236.492670
15	104725.052709	101119.235154
20	104705.094222	100833.164692
25	104715.073421	100912.998291
50	104658.524622	100879.734294

TABLE XII. SIMULATION RESULT ON DESIGN AREA FOR 2 CORES

Clk Cycle (ns)	DC (µm^2)	ICC (µm^2)
5	214556.129517	226122.022404
10	205681.293995	198546.166134
15	209177.340599	201955.726282
20	209230.562844	201639.718174
25	209240.542041	201589.822177
50	209207.278040	201623.086181

TABLE XIII:. SIMULATION RESULT ON DESIGN AREA FOR 4 CORES

Clk Cycle (ns)	DC (μm^2)	ICC (μm^2)
5	429840.740624	451073.151999
10	412297.306393	396403.767531
15	429840.740624	451073.151999
20	419289.399292	403309.374013
25	419056.551301	403163.012417
50	419169.648889	403389.207617

TABLE XIV. SIMULATION RESULT ON DESIGN AREA FOR 8 CORES

Clk Cycle (ns)	DC (μm^2)	ICC (μm^2)
5	864538.025198	900975.410968
10	824657.814330	793033.730175
15	837308.114419	807217.500337
20	838642.000128	806595.463147
25	838585.451337	806565.525556
50	838459.048120	806771.762359

All results design area for each core are always changing at different frequency. In low frequency, simulation area for ICC is lower than DC in all single and multicore. This happen because optimization of area in DC is only like the first size estimation because it synthesizes the design into a gate-level netlist while still meeting the constraints. It uses virtual wiring routing to connect every block. In ICC, it uses real interconnection and real layout according to the standard cell library. This standard cell library is the representation of the physical shapes that will be fabricated. It also contains timing information about the function such as delay and input pin capacitance that are used to calculate output load. But the result in high frequency shows the area in ICC is larger than DC at all cores. This happen because in ICC synthesis, in order to meet the design specification, the standard cell library will use high drive cell that are able to operate in high frequency. High drive cells are needed because it has bigger width that will allow more electrons to flow and made it compatible for high frequency.

LATENCY (ns)	THROUGHPUT (MHz)			
	Single	2 cores	4 cores	8 cores
5	0.20	0.40	0.80	1.60
10	0.10	0.20	0.40	0.80
15	0.07	0.12	0.27	0.53
20	0.05	0.10	0.20	0.40
25	0.04	0.08	0.16	0.32
50	0.02	0.04	0.08	0.16

Figure 21. The Throughput data for all cores

Fig. 21 show the throughput data for single, 2, 4 and 8 cores. Throughput data on 8 cores is higher than other cores in every latency. It shows that the throughput in high number of cores is greater than the lower number of core.

V. CONCLUSION

All the design of matrix filling module and multicore architecture has been successfully constructed. The output for every core is exactly correct according to the sensitivity of Smith Waterman algorithm. The entire project's objectives have been successfully achieved. The timing analysis shows the best clock cycle that suitable for this design is from 15 ns and above since there is no violation occurs. Power consumption increases in high frequency because a faster switching activity is needed in every module in order to meet the specifications. The real area of the design is measured in ICC synthesis because it uses standard cell library to optimize every module. This design also has been developed using ASIC design flow from RTL coding until GDSII level. Finally, the overall IC layout is illustrated in Fig. 20.

REFERENCES

[1] E. Willett, *Genetics Demystified*. United States of America: McGraw Hill, 2006.

[2] S. A. M. Al Junid, M. A. Haron, Z. Abd Majid, F. N. Osman, H. Hashim, M. F. M. Idros, and M. R. Dohad, "Optimization of DNA Sequences Data to Accelerate DNA Sequence Alignment on FPGA," in *Mathematical/Analytical Modelling and Computer Simulation (AMS), 2010 Fourth Asia International Conference on*, pp. 231-236

[3] F. ZHANG, X.-Z. QIAO, and Z.-Y. LIU, "A Parallel Smith-Waterman Algorithm Based on Divide and Conquer," in *Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing*: IEEE, 2002, p. 8.

[4] S. A. M. Al Junid, M. A. Haron, Z. Abd Majid, A. K. Halim, F. N. Osman, and H. Hashim, "Development of Novel Data Compression Technique for Accelerate DNA Sequence Alignment Based on Smith–Waterman Algorithm," in *Computer Modeling and Simulation, 2009. EMS '09. Third UKSim European Symposium on*, 2009, pp. 181-186

[5] S. A. M. Junid, Z. A. Majid, and A. K. Halim, "Development of DNA sequencing accelerator based on Smith Waterman algorithm with heuristic divide and conquer technique for FPGA implementation," in *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, 2008, pp. 994-996

[6] D. Gohringer and J. Becker, "High performance reconfigurable multi-processor-based computing on FPGAs," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pp. 1-4.

[7] D. M. Harris and S. L. Harris, *Digital Design and Computer Architecture*: Morgan Kaufmann, 2007

[8] H. B. Kommuru and H. Mahmoodi, "ASIC Design Flow Tutorial Using Synopsys Tools," San Francisco, CA: San Francisco State University, 2009, p. 10.

[9] J. Li, G. Du, D. Zhang, Y. Song, L. Li, and H. Pan, "High throughput memory data-path design for multi-core architecture," in *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, pp. 28-31.

[10] I. Bolsens, "Programming customized parallel architectures in FPGA," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pp. 1-1