Design and Analysis of DNA Sequence Alignment Module Using Smith-Waterman Scoring Patterns.

W.A.Q.M. Salleh and A.K. Halim
Department of Electronic, Faculty of Electrical Engineering
University Teknologi MARA, 40450
Shah Alam, Selangor, Malaysia
Email : wanabdulqayyum@gmail.com

Abstract— **This project is to design and analyze the DNA sequence alignment module using Smith-Waterman scoring patterns. The main objective is to create SW pattern in matrix using Excel based equation. The second objective is to construct the module for SW sequence alignment using the matrix pattern. One of the major problems rose from DNA sequence alignment is the speed and accuracy. The computational demand needed to explore for analyze the data faster and accurate in these databases is quickly becoming a great concern. This system will be using Verilog language and optimize it using pattern recognition in order to overcome the accuracy issues. The system optimizes the aligning DNA fragment using Smith-Waterman algorithm with a pattern recognition algorithm. FPGA design, synthesis and simulation are done using the Xilinx ISE software and obtain the RTL schematic as well as the waveform of the module. Synopsys tools are used for analysis the project in ASIC design. VCS for verification the design for further process in ASIC design tools are used. Design Compiler is for resynthesize and remodelling the design with setup constraints. Static timing analysis using Prime Time is for advance timing analysis. Based on the result obtained, the minimum frequency that is suitable for this project is 50 MHz. The average power consumption for normal compile high compile and ultra-compile are 53 uW, 52.8 uW and 49.8 uW. The average Cell area for normal compile high compile and ultra-compile are 18700 um^2, 18280 um^2 and 9532 um^2.**

I. INTRODUCTION

Nowadays, demands for sensitive and high performance DNA sequence alignment is increase from year to year and it is supported by and proven where the size of genomic database is double in every 16 months [1]. From time to times, DNA sequence alignment experience complexity due to exponentially increase of DNA sequence data. Besides, in pharmaceutical world may have been corrupted or failure due to high chemical contained in a pharmaceutical product. This is why the DNA sequence alignment tools for applications are important, in order to protect the consumer from being cheated and prove evidence of the unethical pharmaceutical company.

DNA is the hereditary material in humans and not only that but almost all other organisms. Nearly every cell in a person's and any other organism body have the same DNA. DNA mostly located in the cell nucleus (where it is called nuclear DNA and some small amount can be found in mitochondria). The information in DNA is stored and can be represented by four chemical bases: *adenine* (A), *guanine* (G), *cytosine* (C), and *thymine* (T) [2]. Human DNA consists of about 3 billion bases, and more than 99 per cent of those bases are the same in all people's DNA. The order, or sequence, of these DNA bases determines the information available for building and maintaining an organism, which are similar to the way in which letters of the alphabet appear in a certain order to form words and sentences.

There are several types of DNA sequence alignments that have been implemented and develop throughout the years. Needleman-Wunsh [3] and Smith-Waterman [4] both are the famous global and local alignment algorithm. They are also is a dynamic operating based but suffer in terms of time and space complexity especially running on the microprocessor platform. There's also pair-wise alignment such as dot-plot method, dynamic programming (N-W and S-W) and word method (FASTA and BLAST). Smith-Waterman is more reliable in terms of accuracy than Needleman-Wunsh in a dynamic programing. In this paper, the SW algorithms were used. Smith–Waterman also is a local search and there are also other local search such as FASTA (Fast Alignment Search Tools-All) and BLAST (Basic Local Alignment Search Tool). This module is design using verilog language based. The design is analysed and simulated first before continuing into ASIC design flow. Previously, tools such as FASTA [5] and BLAST [6] are used for DNA sequence alignment. Recently, FPGA is also used for DNA sequence alignment [7].

II. SMITH-WATERMAN ALGORITHM

Needleman and Wunsch [8] and Sellers [9] were the first introducer for using technique of global alignment based on dynamic programming. Smith and Waterman continued it with the algorithm to identify the common molecular subsequence but the technique is on local alignment [10]. After that, some modifications by considering fine gap penalties and still be used until today for calculate the score alignment when involve with Smith-Waterman algorithm were introduced by Gotoh [11]. This algorithm were modified further in order to reduce the space required by introducing the quadratic time and linear space algorithm for optimization of Gotoh's modification by Myres and Miller [12].

The Smith-Waterman algorithm is used to compute the optimal local alignment of two sequences. The procedure consists of three steps:

1) Fill in the dynamic programming matrix.
2) Find the maximal value (score) and trace back the patch that leads to maximal score to find the optimal local alignment.
3) Detect the Max Score

The basis of a Smith-Waterman search is comparison of the two DNA sequences. It used individual pair-wise comparison between characters such equation (1):

$H(i, 0) = 0,\ 0 \le i \le m$ ——— (1)

$H(0, j) = 0,\ 0 \le j \le n$ ——— (2)

If $a_i = b_j$ then $w(a_i, b_j) = w$ (match) or ——— (3)

If $a_i \neq b_j$ then $w(a_i, b_j) = w$ (mismatch) ——— (4)

$$H(i.j) = \max \begin{cases} 0 \\ H(i-1, j-1) + w(a_i, b_j) & \text{Match/Mismatch} \\ H(i-1, j) + w(a_i, -) & \text{deletion} \\ H(i, j-1) + w(-, b_j) & \text{Insertion} \end{cases}$$

III. METHODOLOGY

A. *Macro Programming*

The equation (5) is created by understand how SW scoring works and implement it on excel to gain a pattern for 4 based pair of DNA sequence.

C7=MAX (0, B6+IF (C5=A7, $B$1, $B$2), C6+$B$3, B7+$B$3) —(5)

Match score:	2	= B$1 ——— (6)
Mismatch score:	-1	= B$2 ——— (7)
Gap penalty:	-1	= B$3 ——— (8)

	A	B	C	D	E	F
5			A	A	A	T
6		0	0	0	0	0
7	A	0	2	2	2	1
8	A	0	2	4	4	3
9	A	0	2	4	6	5
10	A	0	2	4	6	5

Figure 1. Complete matrix table with a reference value

Firstly, the SW scoring parameter is setup as shows in equation (6), (7) and (8). The value '0' in the equation (5) indicates that there are no negative value will be generated after scoring been calculated. The equation will evaluate first whether the character is match or mismatch for the S and T. If equal, it will add to the matrix and at the same time it will compare with the horizontal matrix and vertical matrix which one is the largest so it will add according to the largest value.

If the result is less or is in negative value, it will give a result as '0'. The matrix filling will finish once the 4th character of S and T were compared. After the first block of matrix is completed, it will automatically generate the entire possible pattern for 4 based pair DNA sequence.

		A	A	A	C
	0	0	0	0	0
A	0	2	2	2	1
A	0				
A	0				
A	0				

		A	A	A	C
	0	0	0	0	0
A	0	2	2	2	1
A	0	2	4		
A	0	2	4		
A	0	2	4		

		A	A	A	C
	0	0	0	0	0
A	0	2	2	2	1
A	0	2	4	4	3
A	0	2	4	6	5
A	0	2	4	6	5

Figure 2. Flow of matrix filling using SW scoring pattern

An example of complete alignment of 4 based pair of two DNA Sequences using Smith-Waterman algorithm in Excel as shown in figure 3.

		A	A	A	A
	0	0	0	0	0
A	0	2	2	2	2
A	0	2	4	4	4
A	0	2	4	6	6
A	0	2	4	6	8

Figure 3. Example of a complete matrix table.

B. *Proposed Method*

For a 4 based pair DNA sequence, there are 66,536 matrix patterns from "AAAA" until "TTTT". From the total blocks of matrix, the same pattern will be detected and combined it together. However, in this paper only 4 modules were developed as proof of concept that the pattern recognition can be implemented on the FPGA and ASIC tools.

C. Block Diagram

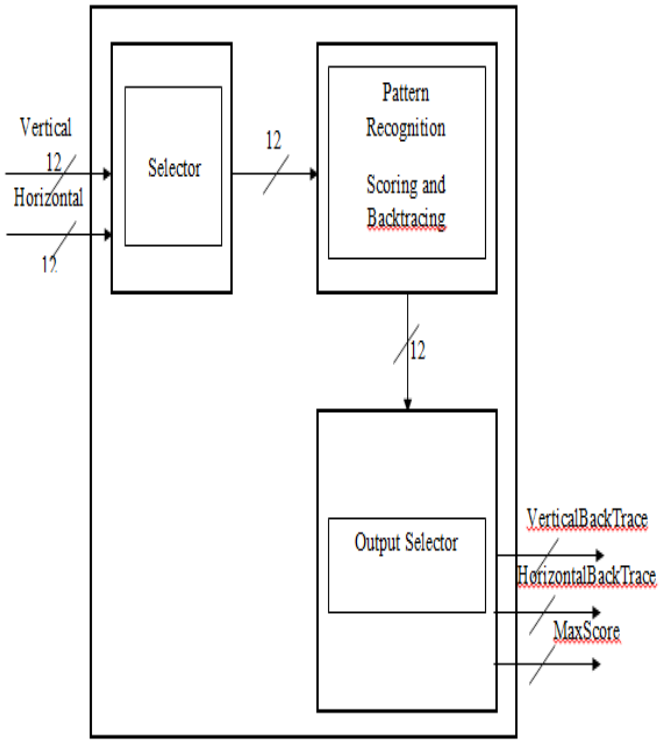

Figure 4. Block diagram for SW scoring patterns

The system consists of three blocks. The inputs for the system came from assembly sequence. The sequence will be in shot gun assembly, so every chunk of sequence will be fill in a matrix table to calculate its gap and penalty.

Table I show the representation of bit for each letter. The DNA character is assigned with three bit data representation where A is represented by "001" while C,G and T as "010","011" and"100". There is also bit "000" that carry out don't care value.

TABLE 1. BINARY REPRESENTATION OF DNA LETTER.

Name	***DNA Letter***	***Binary Representation***
adenine	A	001_2
cytosine	C	010_2
guanine	G	011_2
thymine	T	100_2
Don't care	DC	000_2

Once the input is inserted, it will be selected according to its character. It will only read the vertical characters. Then, it will delegate the input to the exact module to be recognised.

After the input passed the selector, it will now then go to pattern recognition module where all the data and scoring will be generated accordingly with the input given. In this module, it will give an output for back tracing and maximum score for 4 based pair DNA sequence.

Next module is the output selector where it will choose the sequence pattern that had been recognised. The output of this selector is a MaxScore, Horizontal and Vertical BackTrace where the value is already in storage data. This block diagram was design as minimal block so it will execute faster.

D. Project Work Flow

The flow chart in Figure 5 and Figure 6 shows the step in designing the DNA module as in figure 2 from FPGA Design Flow to ASIC Design Flow. For Verilog, Xilinx software is used while for ASIC design, a Synopsis tool is used.

1. FPGA Design Flow:

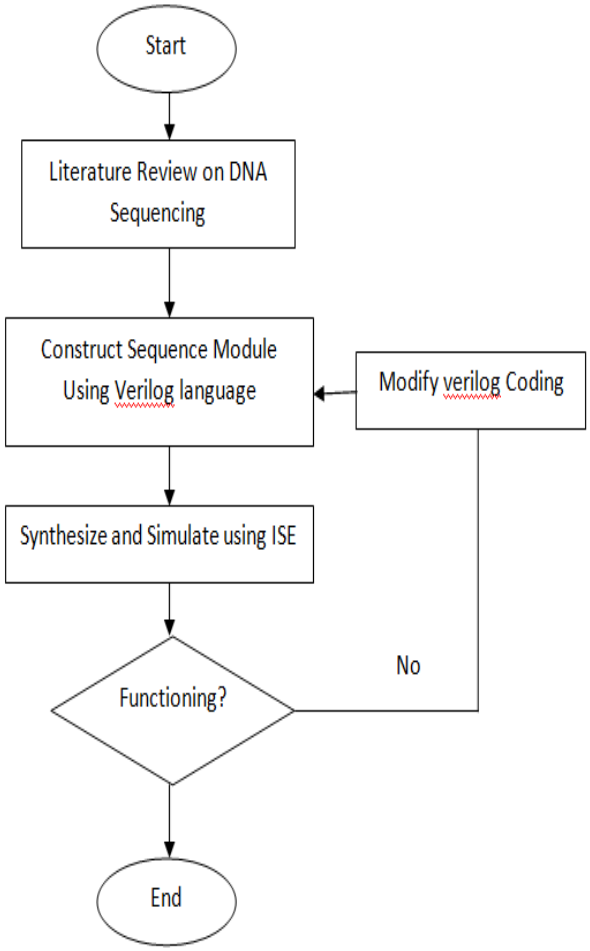

Figure 5 FPGA Design Flow

Figure 5 shows the design flow in FPGA design flow. After some paper about DNA and sequencing method had been reviewed, start constructs the module by using Verilog language in ISE software. Synthesize and test benching to get the result in terms of speed so after that it can be optimize using ASIC. Recheck the Verilog for any syntax errors.

After synthesized and test benching have been made, macro programming data will be compared. This is to check whether the pattern recognition algorithm results are similar with the macro programing. If not, then modify the Verilog and try to get the minimal line of coding to get minimal path when doing the RTL schematic. In this case, the *nested if* is used instead of *for loop* so the circuit will have minimum flip-flops.

2. ASIC Design Flow:

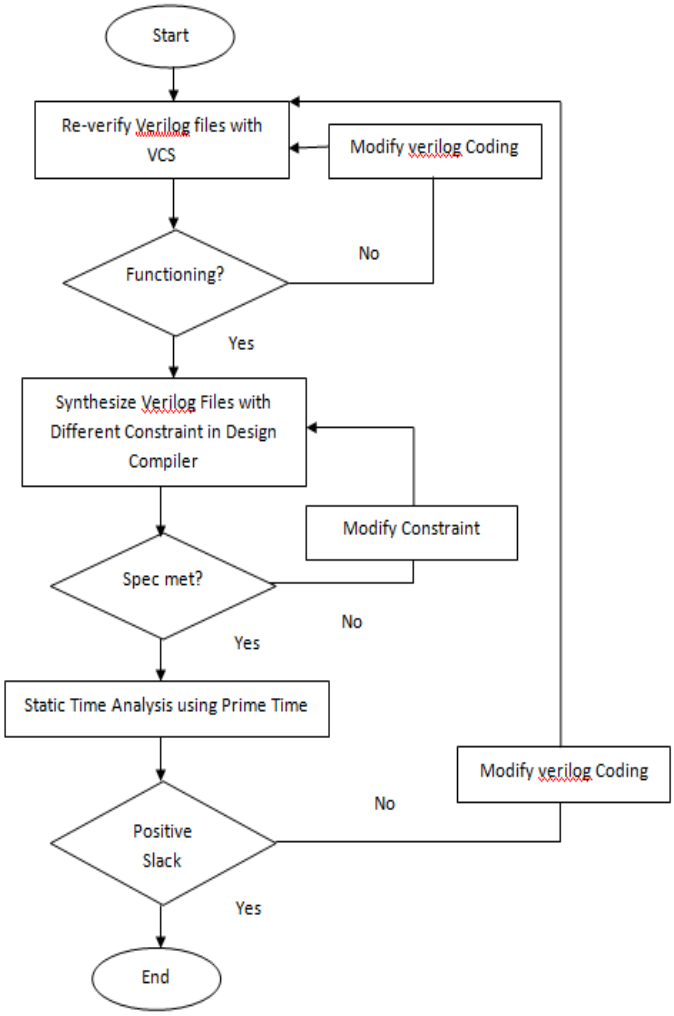

Figure 6 ASIC design flows

Once satisfied with the Verilog file, implement it into Verilog Compiled Simulator (VCS), Design Compiler is done next. This process is to check the functionality of the design. If there error occurs, the Verilog file needs to be check for any error. The DC is repeated until the simulation is as expected.

The parameter of constraints such as timing period, transition time, and delay are varied so the slack and the speed will meet the requirement value. If the slack value gets negative value, timing period, transition time, and delay will be varied.

Lastly, the PT is conducted to optimize the timing of design. The PT process is quite similar to DC process. The only difference is PT will perform more advance timing analysis. It gives better picture of the design timing.

IV. RESULT AND DISCUSSION

A. RTL schematic in ISE

The DNA Sequence Module is designed with Verilog language using the Xilinx ISE tool. Compared to Xilinx ISE, Vivado has better synthesis engine but due to the synthesized time did not take a very long time so ISE is used. Before writing all the coding, all the internal modules need to be specified first. Each of modules is designed based on Figure 4. The DNA Sequence Module RTL schematic generated from ISE tools can be referred to Figure 7.

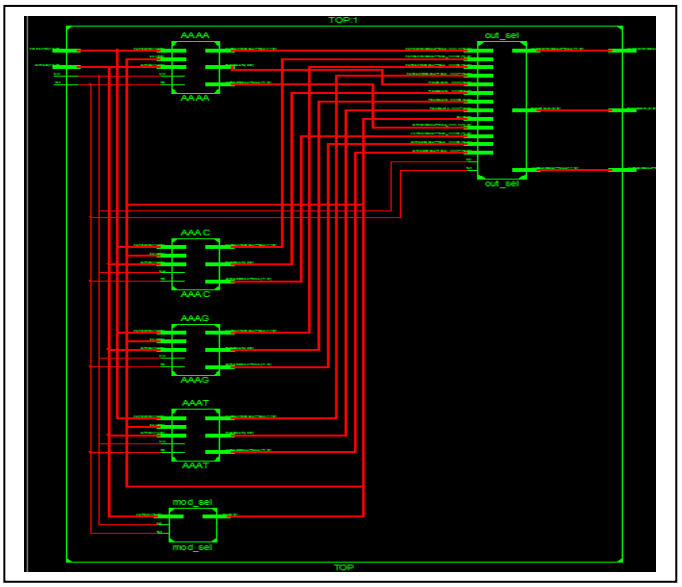

Figure 7 RTL Schematic of DNA Sequence Module.

At first, the idea was designing only one block and overcome the generated sequence module and implements the pattern recognition module. In this figure, the pattern recognition module has to be divided into several modules because the line of coding in every each of modules is too many. Several modules have been design and it may make the DNA sequence module less speed but in term of pattern recognition wise still functioning accordingly.

Another benefit of creating several modules is it's easier to debug and much less time to synthesized every one of the module.

B. Simulation Using ISE

Waveform in Figure 8 is obtained using the ISE simulator.In this simulation, the reset is asserted to 1 after 150ns. After 200ns, the reset is set to 0. Noticed that after the reset change to 0, the output for this sequence started to produce. This module was set that, the reset is active high so when the reset is in high state the DNA sequence module will be in off mode. The output will be generated after 9 clock cycles. The clock is set in 10 ns every cycle.

FIGURE 8 TEST BENCH USING ISE

TABLE 2. SIMULATION RESULT FOR DNA SEQUENCE.

Horizontal	*Vertical*	*Horizontalout*	*verticalout*	*MaxScore*
TTTT	TTTT	000000000000	000000000000	8
AAAG	AAAA	000000000100	000000000100	6
AAAA	AAAA	000000000000	000000000000	8

In Table 2, output for "TTTT" and "AAAA" is completely the same. It is because they have the similar input between horizontal and vertical. For the "AAAG", the value is different due to some mismatch at the input.

C. Verilog Compiler Simulator (VCS)

In ASIC design flow, the module need to be first re-verified using the VCS. This tool will simulate the Verilog module and produce waveform. Test benching is done during this process. The output waveform is observed to determine whether the module functions correctly or not. VCS is more powerful engine because the time needed to compile and synthesized the DNA module is 75% faster than ISE. The design code is successfully can be read by the VCS and it conclude that the codes have no problems in syntax or by running it.

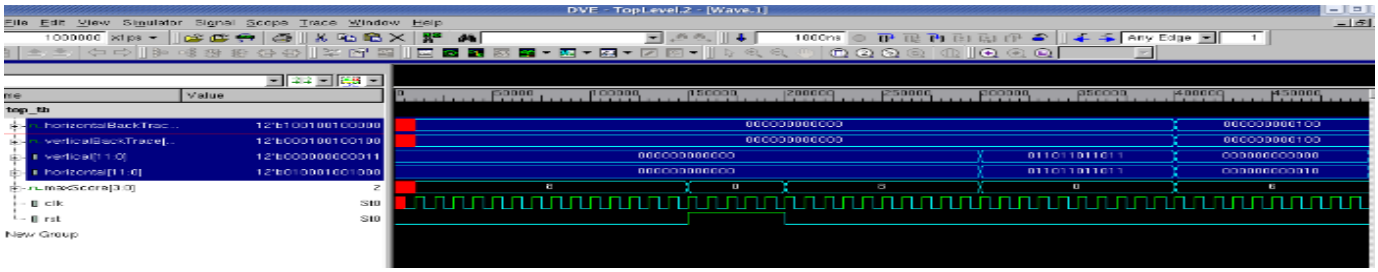

Figure 9 Waveform generated using the VCS.

The waveform data obtained is similar to the waveform generated from ISE simulation. The output data is referred to Table 2.

D. Design Compiler (DC)

This step involves synthesize, with additional constraints applied to the Verilog module. The constraints applied to the design are mostly timing constraints. Once constraints are applied to the design, the module is compiled. Various type of compiling are used to observe the difference in speed processing of the DNA sequence Module. The result is tabulated in Table 3 until Table 15. Figure 14 until Figure 17 in appendix shows the schematic circuit of the DNA Sequence after normal-compile and ultra-compile.

DNA sequence Module compile using 6 different timing period which is 30ns, 40ns, 50ns, 60ns, 70ns, and 80ns. Result for high-compile, ultra-compile and normal compile is taken in term of power consumption, QOR, area and timing for setup and hold.

TABLE 3. 'NORMAL COMPILE' IN DC

TP	*Dynamic Power*	*Leakage Power*	*Cell Area*
30ns	74.1602 uW	70.8040 uW	20017.337880
40ns	60.5241 uW	65.4434 uW	18669.607866
50ns	52.3816 uW	63.8371 uW	18516.746856
60ns	47.0020 uW	63.7054 uW	18371.447854
70ns	43.1254 uW	62.4209 uW	18256.124850
80ns	40.2480 uW	62.9583 uW	18376.175849

From Table 3, dynamic power for 'normal compile' at T_p= 80ns is around 40uW and it continue increasing as the time period decrease until at T_p=30ns where it around 74uW. Leakage power values from T_p= 80ns until T_p= 30ns are decreasing but not much different. For cell area, at T_p= 80ns are around 18376um^2 and it decreasing until 18256um^2 at T_p= 70ns. The area start to increase back at T_p= 60ns and continue to increase until T_p= 30ns.

TABLE 4. 'HIGH COMPILE' IN DC

TP	*Dynamic Power*	*Leakage Power*	*Cell Area*
30ns	74.1593 uW	66.9142 uW	18749.342848
40ns	60.5858 uW	65.2254 uW	18437.708838
50ns	52.4695 uW	63.9739 uW	18223.636831
60ns	47.0233 uW	63.4322 uW	18117.070824
70ns	43.1430 uW	62.2214 uW	18033.917825
80ns	40.2662 uW	62.7120 uW	18131.720822

From Table 4, dynamic power for 'high compile' at T_p= 80ns is around 40uW and it continue increasing as the time period decrease until at T_p=30ns where it around 74uW. Leakage power values from T_p= 80ns until T_p= 30ns are decreasing but not much different. For cell area, at T_p= 80ns are around 18131um^2 and it decreasing until 18033um^2 at T_p= 70ns. The area start to increase back at T_p= 60ns and continue to increase until T_p= 30ns.

TABLE 5. 'ULTRA COMPILE' IN DC

TP	*Dynamic Power*	*Leakage Power*	*Cell Area*
30ns	70.3711 uW	29.7866 uW	9586.581938
40ns	57.6832 uW	29.8094 uW	9578.481945
50ns	49.7305 uW	28.9091 uW	9532.343932
60ns	44.7164 uW	28.5201 uW	9433.450936
70ns	41.0704 uW	27.8698 uW	9328.162931
80ns	38.4407 uW	28.2933 uW	9332.535928

From Table 5, dynamic power for 'ultra-compile' at T_p= 80ns is around 38uW and it continue increasing as the time period decrease until at T_p=30ns where it around 70uW. Leakage power values from T_p= 80ns until T_p= 30ns are decreasing but not much different. For cell area, at T_p= 80ns are around 9332um^2 and it continue to increase until T_p= 30ns.

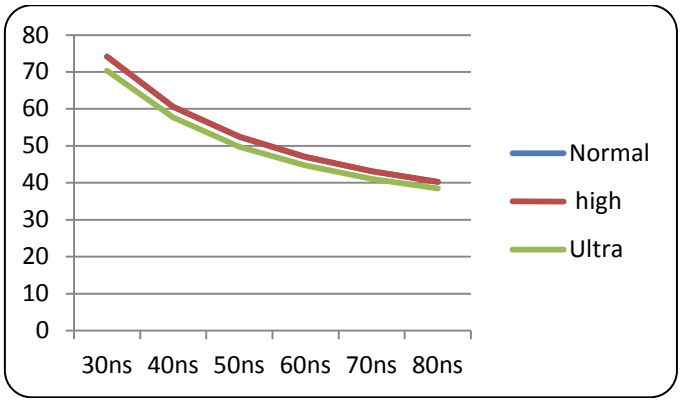

Figure 10. Graph of dynamic power versus timing period

Figure 10 shows a graph of dynamic power on different types of compile with variable time. Power consumption in Ultra compile is much less compare with normal and high compile, whereas the normal and high compile dynamic power are close and have very small different value comparison.

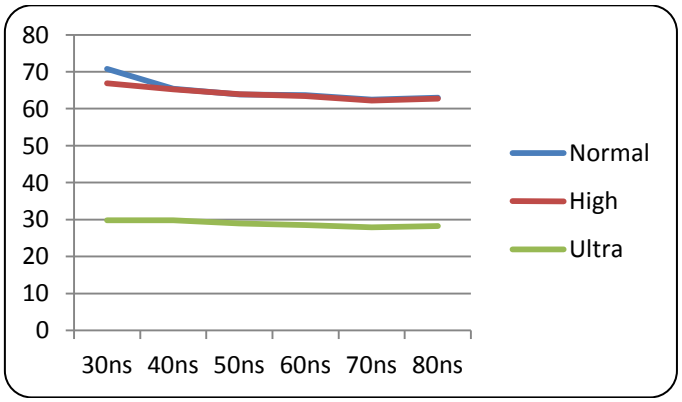

Figure 11. Graph of leakage power versus timing period

Figure 11 shows a graph of leakage power on different types of compile with variable time. Power leakage in Ultra compile is much less compare with normal and high compile, whereas the normal and high compile leakage power are close and have very small different value comparison .thus tis will conclude that the best compile in terms of power is Ultra compile.

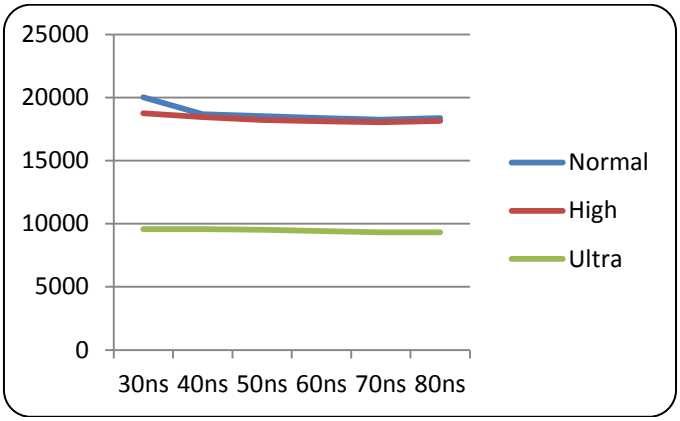

Figure 12. Graph of Cell Area versus timing period

Figure 12 shows a graph of cell area on different types of compile with variable time. Cell area in Ultra compile is much less compare with normal and high compile, whereas the normal and high compile cell area are close and have very small different value comparison .thus tis will conclude that the best compile in terms of area is Ultra compile.

TABLE 6. TIMING FOR' NORMAL COMPILE' IN DC

TP	*T max(T setup)*	*T min (T hold)*
20ns	-1.92	10.25
30ns	0.00	13.27
40ns	0.00	17.30
50ns	0.02	21.10
60ns	0.06	24.93
70ns	0.10	28.75
80ns	0.06	32.49

TABLE 7. TIMING FOR 'HIGH COMPILE' IN DC

TP	*T max(T setup)*	*T min (T hold)*
20ns	-1.88	10.10
30ns	0.00	13.47
40ns	0.04	17.30
50ns	0.00	21.10
60ns	0.39	24.93
70ns	0.76	28.75
80ns	0.06	32.49

TABLE 8. TIMING FOR 'ULTRA COMPILE' IN DC

TP	*T max(T setup)*	*T min (T hold)*
10ns	-1.71	5.03
20ns	0.00	9.24
30ns	0.02	13.27
40ns	0.00	17.09
50ns	0.18	21.86
60ns	1.76	26.07
70ns	0.02	30.28
80ns	0.66	33.06

From Table 6 until Table 8, the minimum value for T max is at T_p = 20ns using ultra-compile. It can be conclude that this module can be operated at 50MHz.

E. Static Timing Analysis (STA) Using Prime Time (PT)

In this process, more advance timing analysis is performed on the module. The result will determine whether the circuit can be proceed to ICC phase. If the STA fails in this step, the module need to re verify until the STA is succeed. Slack improves when doing STA. This due to STA process involves more advance timing compiling.

TABLE 9.STA RESULT FOR 'NORMAL COMPILE'

T_P	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
20ns	-0.2356	1.44
30ns	1.0895	1.56
40ns	3.0895	2.04
50ns	5.0895	3.00
60ns	7.0895	3.39
70ns	9.0895	4.76
80ns	11.0896	5.96

TABLE 10.STA RESULT FOR 'HIGH COMPILE'

T_P	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
20ns	-0.0251	0.532
30ns	3.6453	1.86
40ns	5.7563	2.74
50ns	7.0912	3.89
60ns	8.8711	3.98
70ns	10.8763	4.76
80ns	11.2361	6.96

TABLE 11.STA RESULT FOR 'ULTRA COMPILE'

T_P	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
10ns	-1.0025	0.25
20ns	1.2347	0.44
30ns	4.9094	0.56
40ns	5.0864	2.04
50ns	8.5463	3.00
60ns	11.0841	3.39
70ns	13.8354	4.76
80ns	19.9896	5.96

From the result shows that STA optimized the timing from the design compiler. The modules still operate in 50 MHz which is T_p= 20ns. If it lower than 20ns such as 10ns, it will violate the timing period.

V. CONCLUSION

In a nutshell, the objectives for this paper are successfully achieved. Excel based equation have been design and tabulated for its pattern. The design also can be implemented and executed in FPGA and ASIC tools. This project can be operated at 50 MHz. The average power consumption for normal compile high compile and ultra-compile are 53uW, 52.8uW and 49.8uW. Average Cell area for normal compile high compile and ultra-compile are 18700um^2, 1828um^2 and 56789um^2

ACKNOWLEDGMENT

The author would like to thank Mr. Abdul Karimi Halim for supervising this project. With his guidance, as well as technical feedback, this project managed to come to fruition. Author would also like to appreciate the Faculty of Electronic Engineering in Universiti Teknologi MARA Shah Alam for providing suitable workplace for the author to conduct the project and finish it successfully.

REFERENCES

[1] D.A Benson, I. Karsch-Mizrachi, D.J Lipman, J. Ostell, D.L. Wheeler, "GenBank, Nucleic Acids Res"", Jan 1;33(Database issue):D334-8, 2005.

[2] J. M. Claverie, "Bioinformatic For Dummies", 2nd ed., Indiana: Wiley Publishing Inc, pp.17–721, 2007.

[3] DNAlignTT: Pairwise DNA Alignment with Sequence Specific Transition-Tranversion Ratio, Ankit Agrawal, Xiaoqiu Huang, Dpeartment of Computer Science, Iowa State University, Ames, IA 50011, USA

[4] Vamsi K Kundeti,Sanguthevar Rajasekaran,Hieu Dinh,Matthew Vaughn,Vishal Thapar,"Efficient Parallel and out of core algorithm for constructing large bi-directed de Bruijin graph", BMC Bioinformatic,2010

[5] D. J. Lipman and W. R. Pearson, "Rapid and Sensitive Protein Similarity Searches", Science, vol. 227, pp. 1435-41, 1985.

[6] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool", J Mol Biol, vol. 215, pp. 403-410, 1990.

[7] S. A. M. Al Junid, Z. A. Majid and A. K. Halim, "High Speed DNA Sequencing Accelerator Using FPGA", In Proceeding of 2008 International Conference on Electronic Design, Penang, Malaysia, pp 1,4, Dec 2008.

[8] S.B Needleman, C.D wunsh, " A general method applicable to search the similarities in the amino acid sequence of two sequences" J. of Molecular Biology, 48 (3): 443-453,1970.

[9] P.H Sellers, "On the theory and computational of evolutionary distances", SIAM J.of Apllied Mathematics, 26 : 787-793, 1974.

[10] T.F Smith,M.S Waterman, "Identification of common molecullar subsequences" J. of Molecular Biology, 147 (1): 195-197,1981.

[11] O. Gotoh, " An improved algorithm for matching biological sequences," J. of Molecular Biology, 162 (3): 705-708,1982.

[12] E.W. Myres, W.Miller "Optimal aligments in linear space," Copm.App. in the Bioscineces, 4 (1): 11-17, 1988.

APPENDIX

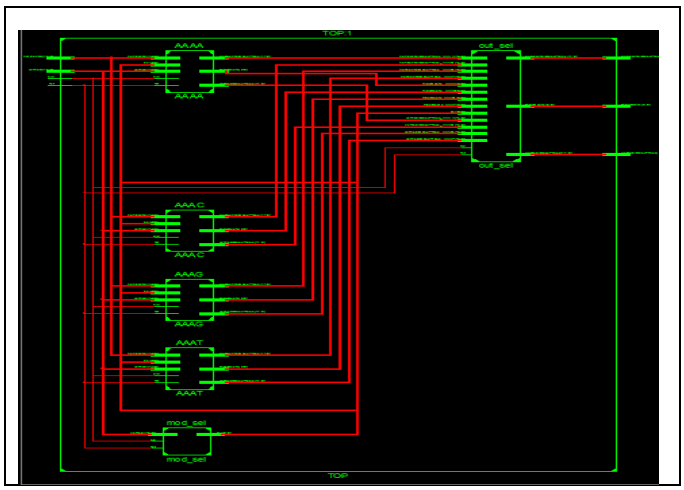

Figure 13 Synthesize schematic using ISE.

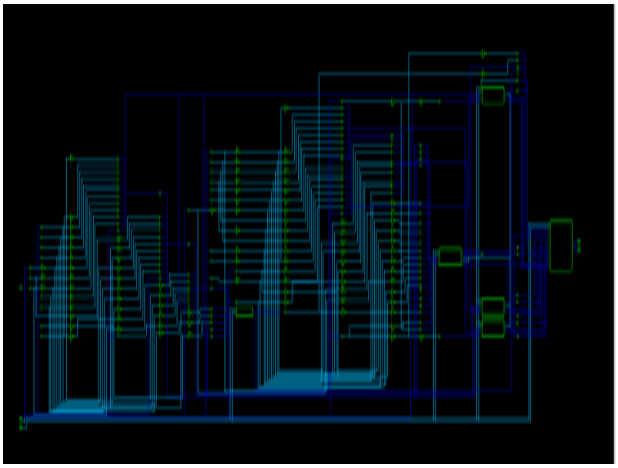

Figure 14 Schematic Circuit of DNA sequence medium compile.

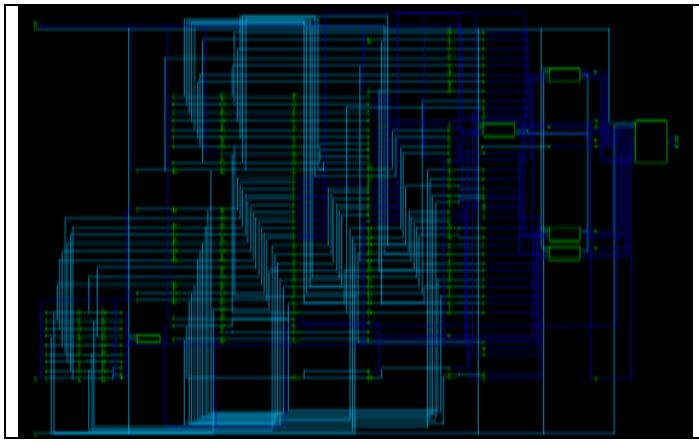

Figure 15 Schematic Circuit of DNA sequence high-medium compile.

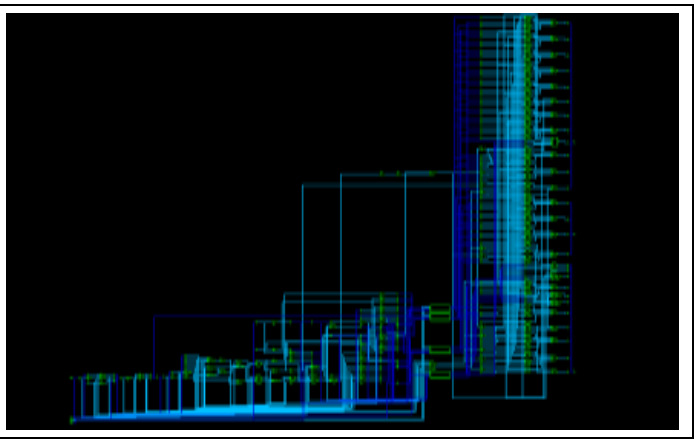

Figure 16 Schematic Circuit of DNA sequence ultra-compile.

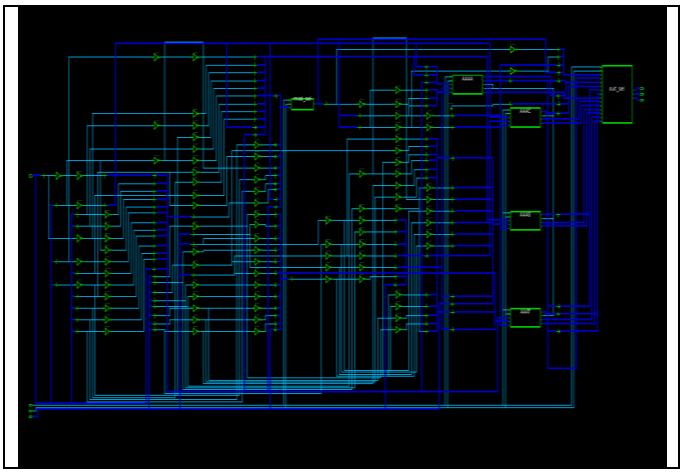

Figure 17 Schematic Circuit of DNA sequence high compile.