Design And Analysis Of Sequence Generator Module Using Eulerian Path Algorithm For DNA Fragment Assembly

M. M. Subri and A. K. Halim
Centre for Electronic Studies, Faculty of Electrical Engineering
University Teknologi MARA, 40450
Shah Alam, Selangor, Malaysia
Email: mus8989@yahoo.com.my

Abstract—**This project is to design and analyze of sequence generator module using Eulerian Path Algorithm for DNA fragment assembly. Mostly, "overlap – layout - consensus" method is used to assemble DNA, but a new system need to be introduced to overcome problems in assembling long DNA sequence in this method. Euler Path Algorithm is used since it has better assembling ability especially in assembling long sequence. The objective of this project is to design the DNA sequence generator modules using the Eulerian Path Algorithm. This project is designed based on speed optimization. DNA fragment assembly is a process reassembling fragment of DNA like a puzzle into several other sequence. DNA fragment assembly consists of two process, assembling and alignment. For FPGA design, synthesis and simulation is done using the Xilinx Vivado software to obtain the RTL schematic as well as the waveform of the module. In ASIC design, Synopsys tools is used for analysis the project. Tools used are VCS for re verifying the design for further process in ASIC design, DC for resynthesize and remodeling the design with constraints and lastly static timing analysis using PT for advanced static timing analysis. The average area for normal-compile using DC is 413,309.1um^2 while for ultra-compile is 83,096.06um^2. The average dynamic power and leakage power for normal compile is 159.0887uW and 1.8045mW, while for "compile ultra" is 99.09uW and 263.54uW. From comparison with timing analysis in DC and STA in PT, this system can be run on minimum 600ns period. Based on the result obtained, this project has been successfully designed and simulated on FPGA and ASIC design flow.**

Keywords-Sequence Generator Module; Eulerian Path Algorithm; DNA Fragment Assembly; DNASGM

I. INTRODUCTION

DNA fragment assembly is a process to assemble the DNA genome and reconstruct it. The demands on this process nowadays not only focus on medical field, it also covers in investigation, inheritance, food and many more fields. Few system had already been designed for DNA fragment assembly. Most of this systems take a lot times to process the DNA sequence and some even take days. A good system need to be cost effective, high reliability, high processing speed as well as low power consumption. Hence, the DNA sequence generator module (DNASGM) will be designed focusing on speed optimization.

Every living beings has Deoxyribonucleic acid (DNA) inside their body. This large macromolecule forms the genes. The DNA form is long and double helix. The sequence in DNA contains information of the DNA carrier, and throughout of the carrier body nearly all the DNA sequence is same [1]. The human DNA contains a nitrogenous organic bases, called nucleotides. In DNA, there are four types of nucleotides, each can be represented with a single letter. The nucleotides are Adenine (A), Cytosine (C), Guanine (G) and Thymine (T) [2]. A single DNA strand will contain thousands of these nucleotides. In the DNA, the helix is complimentary of each other [1]. Each nucleotides has it owns pair, adenine (A) with thymine (T) and cytosine (C) with guanine (G). By using this four letter, fragment assembly can be conducted faster compared by assembling using the 20 types of amino acid [2].

There are various techniques to be used for DNA sequencing. The first one is sequencing by Hybridization. Hybridization is a technique which the short sequences of nucleotides DNA is contacted with target DNA sequence [3]. This method used synthetic fragment of DNA in probe which contain 8 to 30 nucleotides. The target DNA sequence will hybridize to the probe when it is complement to each other [4]. The second technique for sequencing DNA is shotgun sequencing. Using this technique, sequence is produced from composite of overlaps small fragment sequence [5]. For this technique, two approach can be used. For the first approach, the DNA sequence is break into segments with random length [9]. All this fragment is then assembled by overlapping [6]. The fragments obtain from this approach can be assembled using Euler Path Algorithm, de Brujin graph as well as overlap-layout-consensus [6]. Another approach of Shotgun sequencing is by breaking the DNA sequence into longer length fragment [6]. From this long fragment, it will be break into smaller length fragment [6]. This smaller fragment is then assembled using various technique. This approach has better speed processing compared to previous approach [6]. Asides from Hybridization and Shotgun sequencing technique, another method is presented recently, which is High-throughput sequencing technologies [7].

There are many types of DNA fragment assembly algorithm. A studies had already been conducted to compare between the methods [8]. One of the DNA fragment assembly technique is "overlap-layout-consensus" [8]. This technique consists of three

phase. The first phase is Overlap Phase [8]. This phase objectives is to find the overlapping fragments. In this phase, the best or longest path of the sequence will be determined. The second phase is Layout Phase [8]. The order of fragment is determined by overlapping the fragment. This is the phase which combination of the alignment will be obtained. Lastly, the third phase is Consensus Phase [8]. This phase will determine the sequence from layout obtain in Layout Phase. Another method for assembling is using de Brujin graph [9] [10]. This graph consist of vertices and edges. It is dependent to $l - 1$. The l is the length of the edges. Euler path algorithm is another method in assembling the DNA fragment [11] [12] [13]. This algorithm is a better approach compare to "overlap-layout-consensus" approach [11] [12] [13]. This project module is design using Verilog language based on Euler path algorithm. The design is analyzed and simulated first before continuing into ASIC design flow.

II. EULERIAN PATH ALGORITHM

The Euler path algorithm involves constructing the DNA sequences into de Brujin graph [14] first before applying the Euler path [15].

This algorithm originates from Leonard Euler, a Swiss Mathematician, in 1736. This algorithm is related to the problems of seven bridge of Konigsberg [16]. In this problem, there are seven bridge connecting between islands at the City of Konigsberg. A route need to be determine which the road only pass through each bridge once. [17].

There are two types of Euler, Euler path and Euler circuit. Both of this types uses the edges only once. The different is Euler path starts and ends at different vertices. While for Euler circuit, it starts and ends at the same vertices. Since the DNA sequence starting and ending point is already determined, Euler path is used for DNA sequencing [18].

Before constructing the de Brujin graph, the k value need to first identified. For this paper, $k = 4$ is used in the module. Once the k is identified, the DNA sequence is separated to k-1 sized vertices. All the vertices is connected with k sized edges. All the vertices will be aligned into a node graph with the edges respectively. Eulerian cycle is then applied to the graph [18]. All possible sequence is then generated.

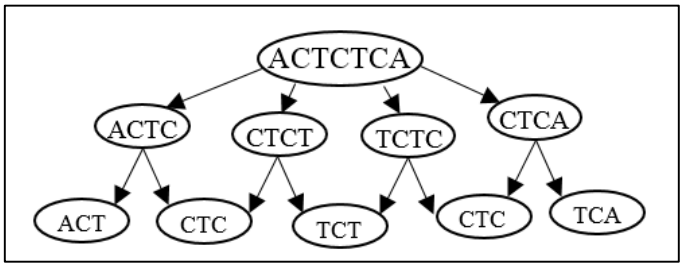

Figure 1 DNA sequence divided to edges and vertices diagram.

In Figure 1, a DNA sequence of 7 letter width is used to apply to de Brujin graph. First, k-mers is set to 4. Edges is 4 letter width and the vertices is 3 letter width. In a single edge, it consist of 2 vertices. 2 edges share one common vertice.

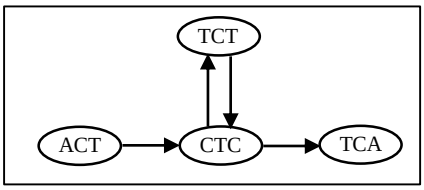

Figure 2 De Brujin graph with Euler path.

In Figure 2, the vertices obtain from Figure 1 is rearranged to de Brujin graph. The common vertices is tied together, leaving only 4 vertices in the graph. Two sequence can be obtained from Figure 2, which is ACTCTCTA and ACTCTA.

III. METHODOLOGY

The DNASGM will be go through process of FPGA and ASIC designing. Before designing, the specification of this module need to be identify and illustrate into block diagram. By using block diagram, all the important features of this modules can be review effectively before continuing into the FPGA and ASIC design flow.

A. Proposed Technique

To construct the DNASGM, the Assembly module will implement the hybrid technique of de Brujin graph and Euler Path Algorithm. The inputs need to be converted to edges and vertices. To avoid producing repetition output, a module is needed to check common between the edges. In this project, the Assembly module does not implement the Euler algorithm completely. Instead, all possible types of de Brujin graph with Euler path applied to the graph is first identified. From the identified graph, all possible output can be produced. This produced output is placed inside the coding. By doing this method, the outputs of the Assembly module will be many. A selector module will be added to select only the possible outputs for the input sequence to be outputted out of DNASGM.

B. DNASGM Block Diagram

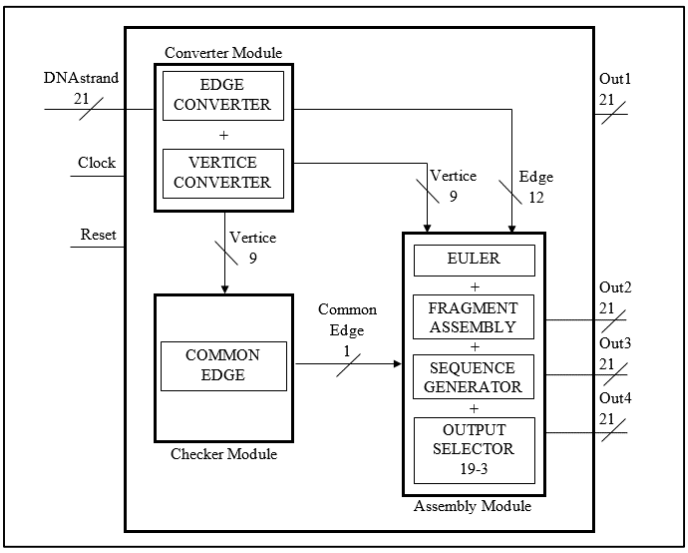

Figure 3 Block diagram of DNASGM

Figure 3 shows the block diagram of DNASGM. This system has three inputs and four outputs. The inputs consist of clock,

reset and port for inputting DNA sequence. For this system, the DNAstrand input can only take seven letters of DNA sequence. For one letter, it is represented with three bits binary number, which total up to 21 bits binary. Table I shows the representation of bit for each letter. The outputs consist of out1, out2, out3 and out4. The port out1 will output the original DNA sequence for checking purpose. The ports out2, out3 and out4 are used to output possible DNA sequence from Euler algorithm. All the outputs is 21 bits binary. Since some of possible DNA sequence from Euler algorithm might contain less than seven letter, this blank letter is replaced with ignore representation.

TABLE IV. BINARY REPRESENTATION OF DNA LETTER.

DNA Letter	***Binary Representation***
A	000_2
C	001_2
G	010_2
T	011_2
Ignore	111_2

There are three major modules block in the system. The first module is Converter module. This module functions is to converts the input DNA strand into edges and vertices. The *k* is set to four. The edges will be four letter width and vertices will be three letter width. 12 bits and 9 bits are used for each edges and vertices respectively. The second module is Checker module. This module will check if there exist repetition or the edge path is used twice in the original sequence. If repetition exist in the sequence, this module will set the module output to 1 and vice versa. Last but not least, the Assembly module will transform the edges and vertices into de Brujin graph and apply Euler path on the graph. The output of this module is the possible DNA sequence after applying Euler path.

This system is clock driven. It has reset input for resetting purpose. The DNA sequence is first inserted into the system through the "DNAstrand" port. Vertices and edges are produced from the DNA sequence. The checker module will then check if there exist repetition in the DNA sequence. If repetition exist, the checker will disable the Euler block from further processing. If not, the vertices and edges will go through the Assembly module and the de Brujin is applied as well as the Euler path. If Euler path exist in the sequence, this module will sent the possible output data to the output port out2, out3 and / or out4. If there are no Euler path in the sequence, all the output port will produce ignore output.

C. Project Work Flow

This project design flow is separated into two, FPGA design flow and ASIC design flow. FPGA design flow will be conducted first before doing the ASIC design flow. The FPGA flow is designed using the Xilinx Vivado software. For the ASIC design flow, Synopsys tools are used.

1) FPGA Design Flow:

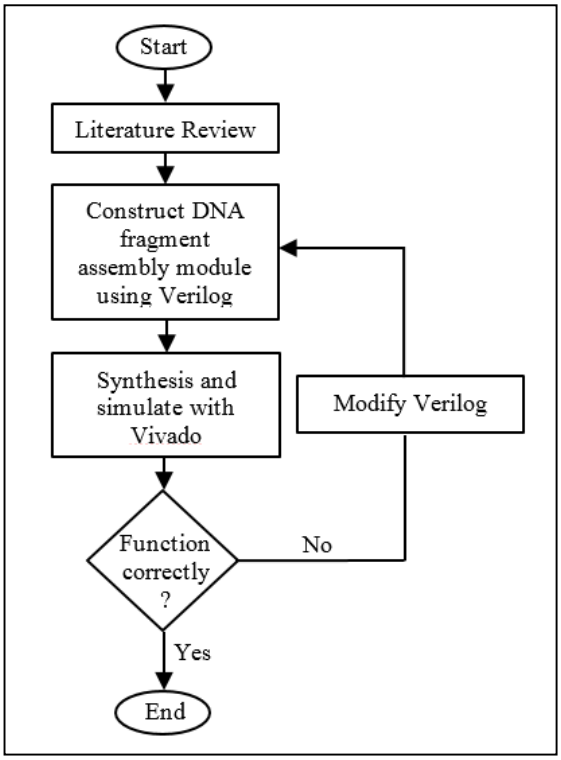

Figure 4 FPGA design flow

Figure 4 shows the design flow of DNASGM in FPGA design flow. Firstly, the theoretical of DNA is studied. Different types of algorithm for DNA fragment assembly is reviewed. All the reviewed material of algorithms for DNA fragment assembly are compared to create the working DNA fragment assembly module.

Once better understanding on the design is obtained, the module of the DNASGM module is modelled in Verilog language. This module is designed using Xilinx Vivado. Using this software, synthesis and test benching can be done to the design. If error occur during synthesis or test bench, the module is recheck for any syntax error.

2) ASIC Design Flow:

Once satisfied with the Verilog module, the coding is then simulated in Verilog Compiled Simulator (VCS). Figure 5 shows the work flow in designing the DNASGM in ASIC design flow. This process is to check the functionality of the design. If there exist error, the module need to be check for any error. The VCS is repeated until the simulation is as expected.

Then, the module is synthesized with constraints using Design Compiler (DC) of Synopsys. Various constraints are applied to the modules to check the modules timing and other parameters. If the result of compilation does not met the specification, the constraints are changed to other suitable value.

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 static timing.

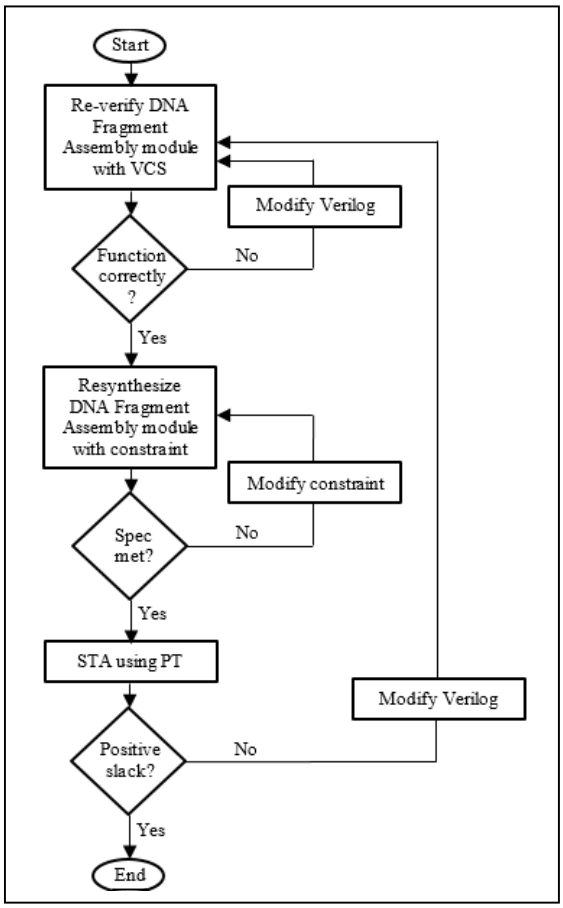

Figure 5 ASIC design flow

D. Construct Module Using Verilog

The DNASGM is designed with Verilog language using the Xilinx Vivado tool. Compared to Xilinx ISE, Vivado has better synthesis engine. Before writing all the coding, all the internal modules need to be specified first. Each of modules is designed based on Figure 5. The DNASGM RTL schematic generated from Vivado tools can be referred to Figure 6 in Appendixes.

During this process, few additional modules are added to the system to handle timing issues as well as short listing the Euler module from 19 outputs to only three outputs.

Pipeline is added in the modules to synchronize the data arrival from the converter module to assembling module. The first pipeline is added at the same level of common edge module, while the second pipeline is placed between the common edge and assembly module.

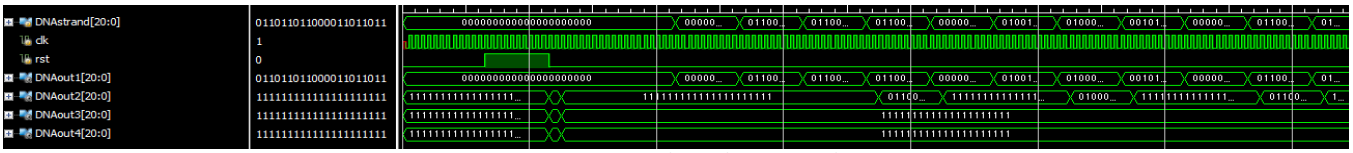

Figure 8 Simulation waveform of DNASGM from Vivado.

After the Sequence module, another module, Selector module, is placed before the output port. This module functions is to select only output pin with data to be connected to the output port. This is due to the Euler block inside the Sequence module have a total of 19 outputs. From all the 19 outputs, only three will have data in it. Before the output port, D flip-flop is added for each port.

IV. RESULTS AND DISCUSSION

A. Synthesize RTL

Vivado software is used to synthesis the Verilog file at RTL stage. The RTL schematic can be referred to Figure 6. As stated before, pipeline is added in the system to synchronize the data transmission timing between the edge converter, vertice converter and common vertice checker. Figure 7 in Appendixes shows the synthesize schematic generated from Vivado software.

B. RTL Simulations

Waveform in Figure 8 is obtained using the Vivado simulator. The inputs are DNAstrand, clock and reset. The outputs are DNAout1, DNAout2, DNAout3 and DNAout4. DNAout1 will generate outputs the same as in input sequence. In this simulation, the reset is asserted to 1 after 130ns. After 100ns, the reset is set to 0. It is noticed that after the reset change to 0, the output for Euler path has value in it. This is due to after reset, the common vertice module will produce output 0, which let the Euler module to apply the Eulerian path on the input sequence. After two clock cycles, the common vertice will sent signal 1 to Euler module to disable the module from applying the Euler path on the input sequence. For this simulation, the clock period is set to 10ns. The output will be produced after 2 clock cycle.

TABLE V. SIMULATION RESULT FOR DNASGM USING VIVADO.

DNA Sequence	*Euler Out 1 (DNAout2)*	*Euler Out 2 (DNAout3)*	*Euler Out 3 (DNAout4)*
AAAAAAA	-	-	-
TAAAAAC	-	-	-
TAGATAC	-	-	-
TATATGC	TATGC	-	-
AATATAT	-	-	-
GGGGGGG	-	-	-
GATGATT	GATT	-	-
CTTGTAT	-	-	-
AATTTTT	-	-	-
TAGAGAC	TAGAC	-	-
TTTATTT	-	-	-

In Table II, the input sequence applied to the DNASGM is in the DNA Sequence column. The three Euler out is the output port of the DNASGM, which all possible Euler path result is

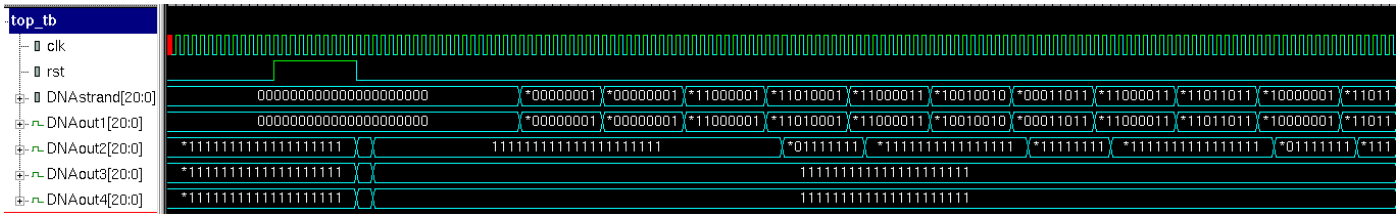

Figure 9 Waveform generated using the VCS

shown here. Whenever there is no Euler path on input sequence, the DNASGM will produce "Ignore" (refer Table I). This "Ignore" is represented with "-" in Table II.

C. Re-verify Design Using 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.

The waveform data obtained in Figure 9 is similar to the waveform generated from Vivado simulation. The output data is referred to Table II.

D. Resynthesize Design Using Design Compiler (DS)

This step involves synthesize, with additional constraints applied to the Verilog module. The schematic circuit before any process done is in Figure 10 in Appendix. The constraints applied to the design is 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 DNASGM.

Figure 11, 12 and 13 in Appendix show the schematic circuit of the DNASGM after normal compile, normal compile with high effort in area and map, and "compile ultra". Five types reading are taken using DC. The readings varies in the time period. The periods use are 500ns, 600ns, 700ns, 800ns and 900ns. QOR, power, timing for setup time and hold time reports are generated for each type of periods.

Figure 14 shows the relationship between cell area and timing period (T_P). The data in the graph is taken from Table III, IV and V. For normal compile, the design area is big for T_P = 500ns, but decrease when T_P increase to 600ns. The area increase after 700ns. The graph then change to 350000um^2 at 800ns. For normal compile with high effort in area and map, the design area is big for T_P = 500ns. When synthesize with T_P = 600ns, the area decrease. But, the area increases when T_P = 700ns. The area is then linearly decrease after T_P = 800ns. For "compile ultra", the area is saturated around 80000um^2.

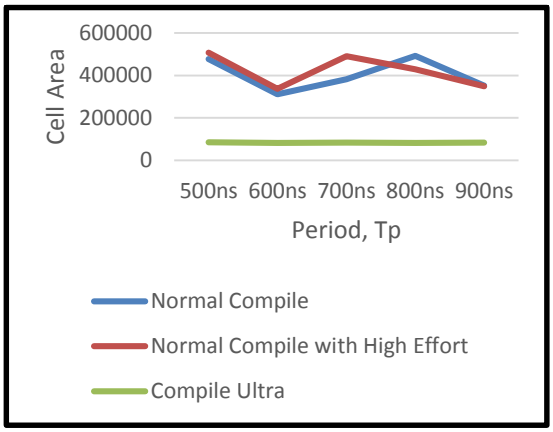

Figure 14 Graph of design area versus timing period (T_P).

Figure 15 shows the relationship between dynamic power and timing period (T_P). The data in the graph is taken from Table III, IV and V. For normal compile, the power is around 200uW at T_P = 500ns. It decrease around 50uW at T_P = 600ns. The graph increase to around 200uW at T_P = 700ns. The graph is the saturated around 180uW. For normal compile with high effort, the power is around 200uW at T_P = 500ns. It decrease to around 50uW at T_P = 600ns. The graph is then increased and saturated from T_P = 700ns at around 180uW. For "compile ultra", the graph is saturated at around 100uW.

Figure 16 shows the relationship between leakage power and timing period (T_P). The data in graph is referred to Table III, IV and V. For normal compile, the power is around 2.2mW at T_P = 500ns. At T_P = 600ns, the power decrease at 1.2mW. The graph increase from T_P 700ns to 800ns. The graph finally decrease to around 1.5mW at T_P = 900ns. For normal compile with high effort, at T_P = 500ns, the power is around 2.2mW. At T_P = 600ns, the power decrease to around 1.2mW. The power increased to around 2.2mW at T_P = 700ns. The graph is then linearly decrease after T_P = 800ns. The power is saturated at 0.2mW for "compile ultra".

TABLE III. DC REPORTS FOR NORMAL COMPILE.

T_P	Total Area		Dynamic Power	Leakage Power	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
	Cell Area	Design Area				
500ns	497910.5298	547561.5491	210.3703 uW	2.5161 mW	-89.76 ns	3.21 ns
600ns	341192.1564	398405.9106	62.3526 uW	1.2639 mW	0.00 ns	1.75 ns
700ns	382728.3902	434138.4808	177.8454 uW	1.4191 mW	0.00 ns	3.94 ns
800ns	492,859.0864	545,906.7130	180.7882 uW	2.5142 mW	0.00 ns	4.31 ns
900ns	351855.2144	406360.9006	164.0872 uW	1.3094 mW	0.00 ns	4.67 ns

TABLE IV. DC REPORTS FOR NORMAL COMPILE WITH HIGH EFFORT FOR AREA AND MAP.

T_P	Total Area		Dynamic Power	Leakage Power	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
	Cell Area	Design Area				
500ns	507198.3388	555567.9978	205.4998 uW	2.4109 mW	-22.73 ns	3.21 ns
600ns	337672.9053	394136.4007	62.3634 uW	1.2377 mW	0.04ns	1.75 ns
700ns	491481.2238	543795.4263	186.8733 uW	2.3549 mW	0.01 ns	3.94 ns
800ns	428325.2779	485582.4985	176.2419 uW	2.0748 mW	0.01 ns	4.31 ns
900ns	349510.2654	403378.3786	163.4858 uW	1.2792 mW	0.03 ns	4.67 ns

TABLE V. DC REPORTS FOR "COMPILE ULTRA".

T_P	Total Area		Dynamic Power	Leakage Power	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
	Cell Area	Design Area				
500ns	85204.4271	93918.6687	105.6925 uW	278.7065 uW	0.06 ns	3.38 ns
600ns	82535.7720	90760.1180	97.8348 uW	256.0657 uW	0.01 ns	4.47 ns
700ns	82803.2030	90888.0817	99.6926 uW	263.4631 uW	0.32 ns	4.11 ns
800ns	82118.2870	90373.9433	97.8254 uW	257.4460 uW	0.01 ns	4.47 ns
900ns	82818.6010	91020.4440	94.4256 uW	262.0069 uW	0.02 ns	4.84 ns

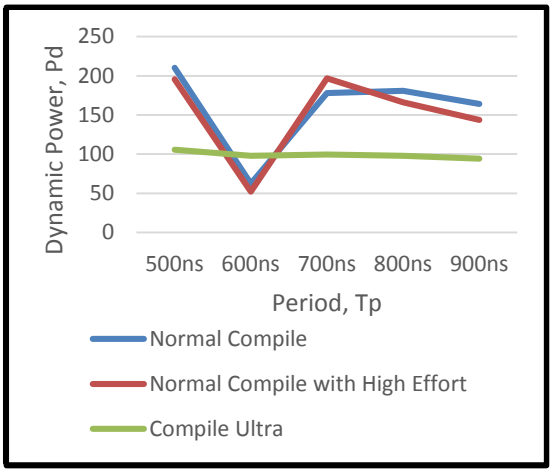

Figure 15 Graph of dynamic power versus timing period (T_P)

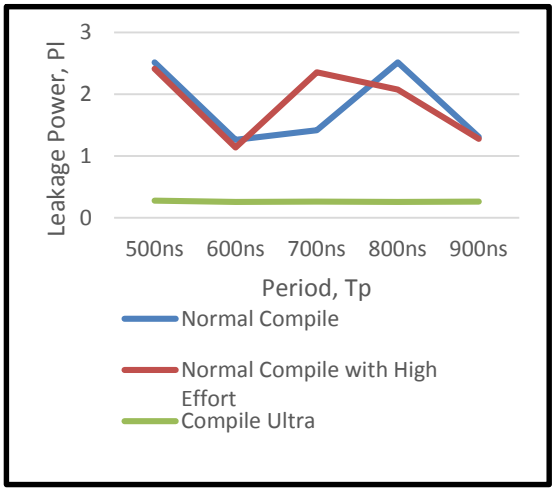

Figure 16 Graph of leakage power versus timing period (T_P).

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

This step is done after performing DC. In this step, more advance timing analysis is performed on the module. In this step, 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. It is noticed that the slack improves when doing STA. This due to STA process involves more advance timing compiling. This project stop at this process. This is due to lack of time and resource to continue the step to ICC and floor planning.

Data in Table VI, VII and VIII consist of all result from performing STA on DNASGM. The setup time and hold time reports are generated from this process. The data from both of this report are recorded in Table IV. Since T_{setup} calculation occurs between two flops, most of the readings has big value of slack. The data recorded shows that the DNASGM can be implement with period minimum of 600ns.

TABLE VI. STA RESULT OF DNASGM FROM DC NORMAL COMPILE.

T_P	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
500ns	393.4361 ns	3.1769 ns
600ns	493.1912 ns	3.5425 ns
700ns	593.0621 ns	3.9078 ns
800ns	692.8196 ns	4.2733 ns
900ns	692.8772 ns	4.2733 ns

TABLE VII. STA RESULT OF DNASGM FROM DC NORMAL COMPILE WITH HIGH EFFORT IN AREA AND MAP.

T_P	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
500ns	393.0659 ns	3.1769 ns
600ns	493.0516 ns	3.5425 ns
700ns	592.9235 ns	3.9078 ns
800ns	692.7366 ns	4.2733 ns
900ns	692.7369 ns	4.2733 ns

TABLE VIII. STA RESULT OF DNASGM FROM DC "COMPILE ULTRA".

T_P	T_{MAX} (T_{SETUP})	T_{MIN} (T_{HOLD})
500ns	13.2272 ns	3.3390 ns
600ns	112.7365 ns	3.7045 ns
700ns	211.2993 ns	4.0700 ns
800ns	310.3354 ns	4.4355 ns
900ns	231.6400 ns	4.4355 ns

V. CONCLUSION

In conclusion, this module is successfully designed using the Euler path Algorithm. This module can be applied to FPGA and ASIC design flow. The timing analysis from DC and STA show the module can be operated in 1.67MHz operation. All of the objectives of this project has successfully achieved.

ACKNOWLEDGMENT

The author would like to thank Mr. Karimi Abdul 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] "What Is DNA?", (Genetics Home Reference), [Online] 2013, http://ghr.nlm.nih.gov/handbook/basic/dna/ (Accessed: 31 May 2013).

[2] J. M. Claverie, "Bioinformatic For Dummies", 2nd ed., Indiana: Wiley Publishing Inc, pp.17–21, 2007.

[3] F. P. Preparata and E. Upfal, "Sequencing-By-Hybrization At The Information-Theory Bound: An Optimal Algorithm", Journal Computational Biology 2000, New York, USA, pp. 245-253, 2000.

[4] P. A. Pevzner, "Computational Molecular Biology: An Algorithm Approach", London, The MIT Press, Cambridge, pp. 67-70, 2000.

[5] S. Anderson, "Shotgun DNA Sequencing Using Cloned DNase I-Generated Fragments", In Nucleic Acid Research, pp. 3015-3027, July 1981.

[6] J. Commins, C. Toft and M. A. Fares, "Computational Biology Methods and Their Application to the Comparative Genomics of Endocellular Symbiotic Bacteria of Insects", In Biological Procedures Online Vol. 11, Dublin, Ireland, pp. 52-78, Dec 2009.

[7] D. MacLean, J. D. G. Jones and D. J. Studholme, "Application Of 'Next-Generation' Sequencing Technologies To Microbial Genetics", In Nature Review Microbial Vol.7, The Sainsbury Laboratory, UK, pp. 287-296, April 2009.

[8] L. Li and S. Khuri, "A Comparison Of DNA Fragment Assembly Algorithm", In Proceeding of 2004 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences, Las Vegas, pp. 329-335, 2004.

[9] J. Kaptcianos, "A Graph Theoritical Approach to DNA Fragment Assembly", In American Journal of Undergraduate Research Vol. 7 No 1, Vermont, USA, 2008.

[10] P. E. C. Compeau, P. A. Pevzner and G. Tesler, "How To Apply de Brujin Graphs To Genome Assembly", Online Publication on 8 November 2011, Nature Biotechnology 29, pp 987-991, 2011.

[11] P. A. Pevzner, H. Tang and M. S. Waterman, "An Eulerian Path Approach To DNA Fragment Assembly", Department of Computer Science and Engineering, University of California, CA, 2001.

[12] P. A. Pevzner, H. Tang and M. S. Waterman, "A New Approach To Fragment Assembly In DNA Sequencing", In Proceeding of 2001 Fifth Annual International Conference on Computational Biology, New York, pp 256-267, 2001.

[13] 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.

[14] H. Zhou, Z. Zhao and H. Wang, "A New Approach For Motif Discovery Based On The de Brujin Graph", In Proceeding of 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin, China, pp. 39-42, 2009.

[15] Y. Zhang and M. S. Waterman, "An Eulerian Path Approach To Global Multiple Alignment For DNA Sequences", Journal of Computational Biology, Vol. 10, Mary Ann Liebert, Inc, pp 803-819, 2003.

[16] G. E. Alexanderson, "About The Cover: Euler and Konigsberg's Bridges: A Historical View", In Journal of The American Mathematical Society Vol. 43, pp. 567-673, July 2006.

[17] L. B. H. Vector, "Eulerian Path and Circuit", [Online] 2013, http://www.informatika.bg/resources/Eulerianpathandcircuit.pdf (Accessed: 1 June 2013).

[18] J. L. Martin, "Euler Paths and Euler Circuits", [Online] 2013, http://www.math.ku.edu/~jmartin/course/math105-F11/Lectures/chapter5-part2.pdf (Accessed: 1 June 2013).

APPENDIX

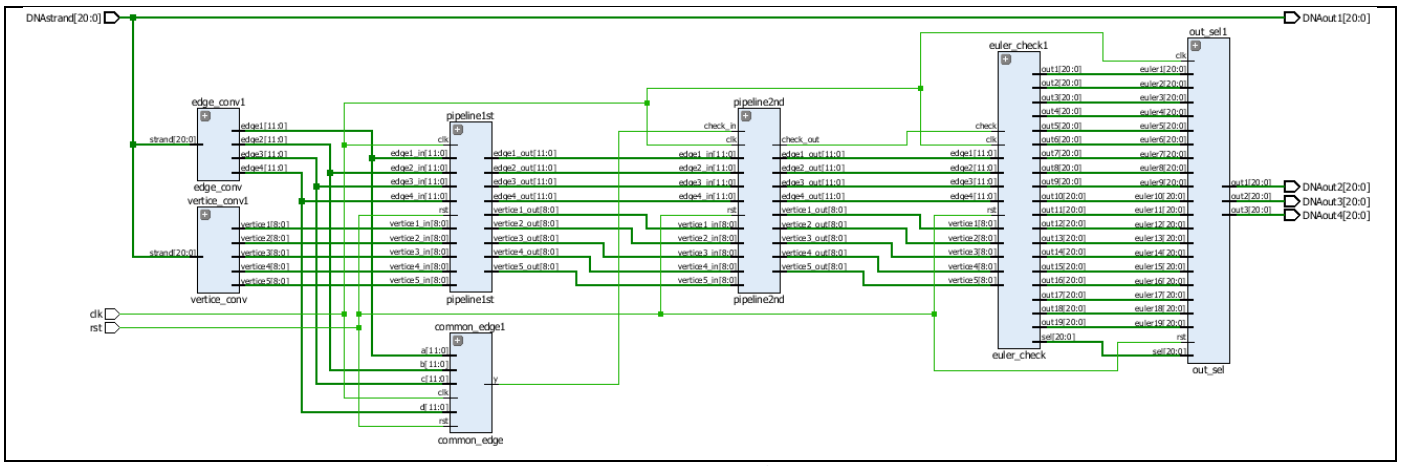

Figure 6 RTL Schematic of DNASGM

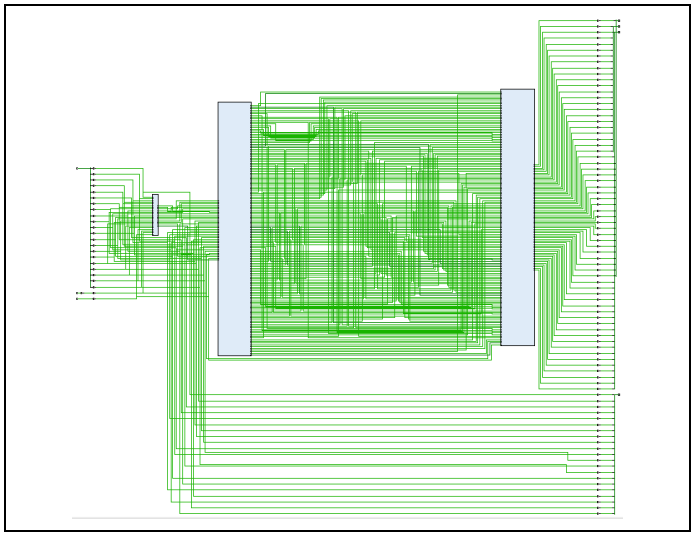

Figure 7 Synthesize schematic of DNASGM.

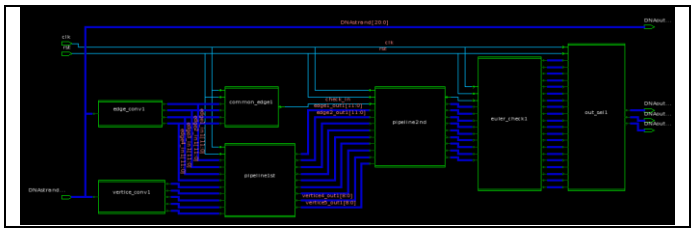

Figure 10 Schematic Circuit of DNASGM before DC.

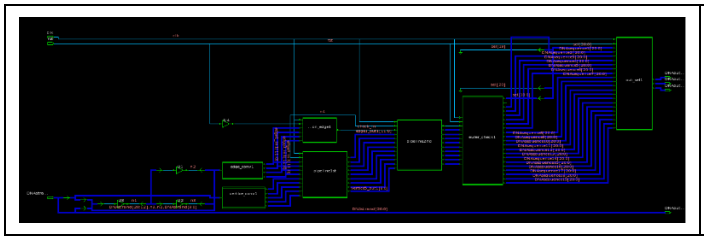

Figure 11 Schematic Circuit of DNASGM for normal-compile.

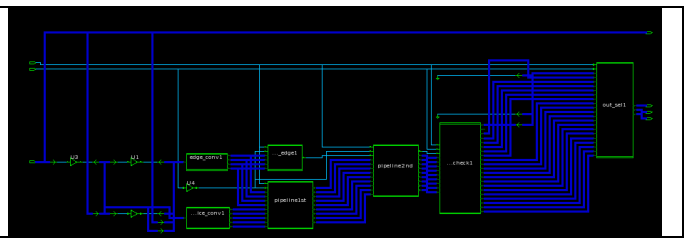

Figure 12 Schematic Circuit of DNASGM for normal compile with high map effort and high area effort.

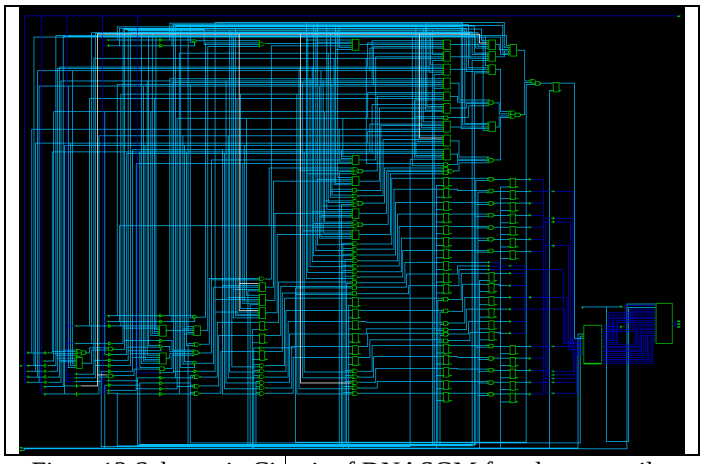

Figure13 Schematic Circuit of DNASGM for ultra-compile.