Design and Analysis of Low Powered DNA Sequence Alignment Accelerator Using ASIC Design Flow

RUDY AZMAN YUSOFF CHEAH, ABDUL KARIMI HALIM, SYED ABDUL MUTALIB AL JUNID, NORHAZLIN KHAIRUDIN

Faculty of Electrical Engineering
Universiti Teknologi MARA
40450 Shah Alam
Selangor, Malaysia
email: rudyazmanyc@gmail.com

Abstract - This paper presents the design and analysis of Low Powered DNA sequence alignment accelerator using ASIC design flow. The objective of this paper is to design and analyze DNA sequence alignment accelerator using clock cycle reduction and frequency scaling technique, which the power consumption can be optimize. The scope of this paper focuses on the minimization of power consumption for an ASIC DNA sequence alignment accelerator on the matrix filling module of the Smith-Waterman algorithm. Smith-Waterman algorithm is a sensitive algorithm used for procedure of DNA sequence alignment in computational molecular biology. As the number of DNA sequence database increases exponentially, it affects the performance of Smith-Waterman algorithm in general purpose computer. Therefore several techniques have been developed to optimize the performance of the algorithm by exploiting parallelism in the design. The low-powered part of the design focuses on the frequency scaling where it produces the minimal value of dynamic power. The design are described using Verilog HDL coding and compiled using Synopsys Tools. From the data obtained using Synopsys tools, the data is then manipulated to get the optimum combination of parameters to produce the most energy efficient IC. The design produces an ASIC that can work at 25ns-50ns clock period where it is in the high energy efficiency region. This range of frequencies produce dynamic power ranging from 224µW-89µW.

Keywords: Smith-Waterman algorithm, DNA Sequencing Alignment, ASIC,

I. INTRODUCTION

The demand for sensitive, accurate and low powered DNA sequence alignment tool for DNA sequences analysis study and investigation is dramatically increased from year to year. This pattern is supported and proved by the statement in [1]; where the size of genomic database is doubled in every 16 months. Genetic sequence searching is in general is highly computationally intensive. This sequence similarity searches are mathematical approaches to sequence comparisons. The most basic sequence alignment is to find the similarity between two sequences. This operation allows biologist to point out sequences sharing common subsequences. From a biological point of view, it leads to identifying similar functionality and determining how well to sequence align are useful for finding related genes. There are many existing tools for sequence alignment, among those are FASTA[2] and BLAST[3], these are software based application where the time complexity has been reduced through some heuristic algorithms. These heuristic algorithm obtain efficiency but degrades the sensitivity of the algorithm. As a result, a distantly related sequence may not be found using these algorithm. As a consequence the use of Smith-Waterman algorithm is becoming more popular, due to its guarantees to find optimal local alignment and return only one result per comparison. Therefore the Smith-Waterman algorithm [4] which is based on dynamic programming technique that provides high sensitivity should be used when getting the exact answer is more important than time.

This leads to another problem where such ASIC design would be used for long hours, therefore A low powered design is needed to tackle the problem of increasing database of DNA sequence.

II. SMITH-WATERMAN ALGORITHM

The algorithm was first proposed by Temple F. Smith and Michael S. Waterman in 1981[4]. Like the Needleman-Wunsch [9] algorithm, of which it is a variation, Smith-Waterman is a dynamic programming algorithm. As such, it has the desirable property that it is guaranteed to find the optimal local alignment with respect to the scoring system being used (which includes the substitution matrix and the gap-scoring scheme). The main difference to the Needleman-Wunsch algorithm is that negative
scoring matrix cells are set to zero, which renders the (thus positively scoring) local alignments visible. Backtracing starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment.

The Smith-Waterman algorithm is used to compute the optimal local alignment of two sequences. The procedure consists of two steps:
1) Fill in the dynamic programming matrix.
2) Find the maximal value (score) and trace back the path that leads to the maximal score to find the optimal local alignment.

\[
\text{score}_X = \max \begin{cases} 
\text{score}_{nw+2} & \text{if } S_i = T_j \\
\text{score}_{anw}-1 & \text{if } S_i \neq T_j \\
\text{score}_{n} - 1 & \text{(gap penalty)} \\
\text{score}_{w} - 1 & \text{(gap penalty)}
\end{cases}
\]

![Search Sequence (Si)](image)

In DNA sequence it consists only 4 characters which are adenine, cytosine, guanine and thymine which have been indicated as A, C, G and T.

### III. LOW POWER DESIGN

There are two common sources of power consumption that engineers must attend to: dynamic and static power. Dynamic power, considered the "active" component of power consumption, is consumed when the device is changing state, and includes both switching power and short circuit power. Short circuit power is dissipated in CMOS logic when both complementary transistors are briefly turned on as the switch changes state, and is generally negligible. Switching power is dissipated by the charging and discharging of capacitors in logic and interconnect as circuits switch. Increasing transistor densities and clock frequencies generate more power dissipation in the circuit. While increasing dynamic power is problematic, engineers can control it by reducing unnecessary switching using techniques such as clock reduction and frequency scaling.

\[ P(t) = i_{DD}(t)\cdot V_{DD} \]

Instantaneous power

\[ E = \int_0^T \int_0^T i_{DD}(t)\cdot V_{DD} \, dt \]

Average power,

\[ P_{\text{avg}} = \frac{E}{T} = \frac{1}{T} \int_0^T i_{DD}(t)\cdot V_{DD} \, dt \]

\[ P_{\text{dynamic}} = \frac{1}{T} \int_0^T i_{DD}(t)\cdot V_{DD} \, dt \]

\[ = \frac{V_{DD}}{T} \int_0^T i_{DD}(t) \, dt \]

\[ = \frac{V_{DD}}{T} \left[ f_{sw} \cdot CV_{DD} \right] \]

\[ = CV_{DD}^2 \cdot f_{sw} \]

Let \( f_{sw} = \alpha f \), where \( \alpha \) is activity factor

\[ P_{\text{dynamic}} = \alpha CV_{DD}^2 \cdot f \]

The dynamic power (switching power) dissipated by a chip is \( C \cdot V^2 \cdot f \), where \( C \) is the capacitance being switched per clock cycle, \( V \) is voltage, and \( f \) is the switching frequency [8]. As frequency changes, the dynamic power will change linearly with it.

\[ I_d = I_{d0} e^{\frac{V_v - V_t}{\eta V_t}} \left[ 1 - e^{-\frac{\eta V_t}{V_v}} \right] \]

\[ V = V_0 - \eta V_{ds} + \gamma \left( \sqrt{\phi_i} + \sqrt{\phi_v} - \sqrt{\phi_d} \right) \]

Static (or leakage) power, commonly thought of as the "wasted" steady-state power, is consumed when the device is in steady state (not changing) and to some degree, also when it’s switching. Static power includes gate leakage and sub-threshold leakage. The adoption of smaller process geometries and lower supply voltages has resulted in an exponential growth in static power [7].

### IV. METHODOLOGY

First the specifications is set then the algorithm was coded using Verilog language and targeted to FPGA (Field-Programmable Gate Array) Spartan 3E XC3S100E board. Using ISE11.1 Isim, the design is synthesized and simulated to verify the outputs of the design and the search for minimal clock cycle is also done in this stage.

Then the design is targeted to ASIC design flow. Using Synopsys tools the design is once again
verified using VCS tools (Verilog Compiler Simulator), after this is done, the design constraints such as the clock period and input/output delay is varied to get the range of power consumptions for the design using DC tools (Design Compiler), also using DC tools the design is optimize and mapped according to the foundry base logic gates. Then the VLSI layout of the design is generated using ICC (Integrated Circuit Compiler). Finally the GDSII file for the design is generated through the ICC tools.

The first design was a hierarchical design where the main modules are consist with several smaller module which is Matrix_driver, Matching and operation_compare module as in figure 2.

The Matching module is to identify which is the base pair that are identical to each other, the input of this module is the sample and target which is both 8bit number that represent 4 base pair in the DNA strand and it produces a 16bit number which represent the 16 matrixes cell in the exact order. Logic ‘0’ for not match and logic ‘1’ for match.

The second module is the operation_compare module which initiates the operation in the matrix and compares all three possible answers from the operation on input of n, w and nw. the output of this module is Xout, which represent the largest answer from the operation.

The last module are the Matrix_Driver, which controls all the data routing for the circuit. Here is where all the matrixes cell are fill with answers from operation_compare module. From the input of the 16bit number from matching module, the driver will initiate operation_compare module and fill the matrix one by one until it finishes.

The problem of this design were the clock cycles taken for all the matrixes cells to be filled. Theoretically, all the calculation of the matrix can be done in 16 clock cycle, but due to the various module that is used in this design, the actual clock cycle were 21 clock cycle.

The second design is also a hierarchical design with major alteration from the first design, instead of filling the matrix one by one, the second design is to fill the matrix diagonally. The improvement were done in the matrix_driver and the operation compare module. The matrix driver are now controlling from 1
matrices cell to a maximum of 4, which is the maximum number of matrix cells in the 4x4 matrix diagonally. In the operation compare module, the calculation is also altered to do a maximum of 4 calculation at a time.

Even with an accelerated process, the second design also faces the same problem which is the amount of clock cycles taken for the matrixes to be filled. Theoretically it should take 7 clock cycle, but it takes longer than that, due to the time taken for the signals to jump from a module to another module.

The final design are a flatten design as in figure 10, where all commands are executed in parallel. This design has only one main module, which can produce all the answer for 16 matrixes cells in 3 clock cycles. The coding style is kept at a minimum to insure that the rtl schematic circuit is also to a minimum. Because larger schematic circuit consumes greater amount of wattage to operate.

The propose architectural design is a flatten design, where all the verilog coding is executed in parallel order. This will reduce the clock cycle taken by the design to process data from the beginning till the end of the algorithm. The matrix filling module is a single module, where all the calculation is done.

![Figure 3: Block diagram of the design](image)

**C. DESIGN VERIFICATION**

The functional verification of the design is done in two difference stage, the first one is by simulating the design, using Isim simulator provided by the ISE 11.1 software. Verification is done by examining the output waveform from Isim and comparing it with the expected results from S-W algorithm.

The second one is verifying it with VCS tool from Synopsys where the steps of verifying is identical to the first stage.

If there is error in any of this verification process, the debugging will be done at the design.

**V. RESULTS AND DISCUSSION**

**A. Design Specifications**

The character been assigned in 2bit word in order to optimize the speed of alignment. All of the initial characters from ASCII been converted into 2 bit size to make it smaller and faster during alignment process [6].

<table>
<thead>
<tr>
<th>Name</th>
<th>Character</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adenine</td>
<td>A</td>
<td>00</td>
</tr>
<tr>
<td>Cytosine</td>
<td>C</td>
<td>01</td>
</tr>
<tr>
<td>Guanine</td>
<td>G</td>
<td>10</td>
</tr>
<tr>
<td>Thymine</td>
<td>T</td>
<td>11</td>
</tr>
</tbody>
</table>

Table1: DNA sequence character with reduction data assignment

<table>
<thead>
<tr>
<th>Target Sequence</th>
<th>Search Sequence</th>
<th>S-W Matrix Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>

Table2: Number of S-W matrix cell depending on the search and target sequence.

From table 2, we can see that S-W matrix cell is proportional to the multiplication of DNA search sequence and target sequence. As the number of DNA sequence basepair increases, the number of matrix cell required to align them also increases. This relation supports the significant of this design.

**B. Architectural Design**

The final design is a flatten design where all the calculation of the algorithm is done in three clock cycle. This is a major improvement in of term performance. The tree clock cycles represent the three layer of matrixes cell that build the 4x4 matrices.

By using the RTL schematic function in ISE11.1, the schematic diagram of the design is generated. Where we can see the difference in schematic diagram generated for 2x2 matrixes, 3x3 matrixes and 4x4 matrixes.

After the design is synthesized, we can obtain the value of slice used and flipflop.

<table>
<thead>
<tr>
<th>Matrixes Cell</th>
<th>Number of Slice</th>
<th>Number of Flipflop</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>49</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>213</td>
<td>43</td>
</tr>
</tbody>
</table>

Table3: Effect to the number of Slice and Flipflop depending on the number of matrixes cell.
From the graph in figure 4, we can estimate the number of Slice and Flipflop used if we were to extend the number matrixes cell to a 8x8 or 10x10 matrixes. For a 8x8 (64cells) matrixes which is an increased of 400% from 4x4 matrixes the estimated Slice is to be around 925 and the number of flip flop to be 88.

C. Design Verification

The design verification is done using ISim simulator from ISE 11.1 and DVE tools from Synopsys, where the test bench is done for 4x4 matrixes with multiple inputs. The target input is fixed at ‘11111111’ while the sample input is varied to full match (11111111), half match (11110000), full mismatch (00000000) then another 3 sets of random input. See in figure 6 and 7.

<table>
<thead>
<tr>
<th>S</th>
<th>T</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>FF</td>
<td>FF</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>00</td>
<td>FF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CC</td>
<td>FF</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>00</td>
<td>FF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FF</td>
<td>FF</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4: The expected scoring output of S-W algorithm

Figure 6: Waveform simulation of the design generated by ISim simulator.

<table>
<thead>
<tr>
<th>Clock Cycle (ns)</th>
<th>Dynamic Power (µW)</th>
<th>Static Power (nW)</th>
<th>ICC Dynamic Power (µm)</th>
<th>ICC Static Power (nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>899.3296</td>
<td>360.4747</td>
<td>1083.6</td>
<td>385.4025</td>
</tr>
<tr>
<td>10</td>
<td>443.7505</td>
<td>351.4709</td>
<td>448.0261</td>
<td>319.2808</td>
</tr>
<tr>
<td>15</td>
<td>295.5526</td>
<td>352.6317</td>
<td>298.5992</td>
<td>320.8182</td>
</tr>
<tr>
<td>20</td>
<td>221.7549</td>
<td>353.1845</td>
<td>298.6017</td>
<td>320.8182</td>
</tr>
<tr>
<td>25</td>
<td>177.7261</td>
<td>352.4469</td>
<td>224.0973</td>
<td>322.9244</td>
</tr>
<tr>
<td>30</td>
<td>148.1051</td>
<td>352.4469</td>
<td>224.1171</td>
<td>322.9244</td>
</tr>
<tr>
<td>35</td>
<td>126.6696</td>
<td>352.6317</td>
<td>149.762</td>
<td>320.025</td>
</tr>
<tr>
<td>40</td>
<td>110.8004</td>
<td>351.454</td>
<td>127.9032</td>
<td>322.1726</td>
</tr>
<tr>
<td>45</td>
<td>98.4892</td>
<td>351.454</td>
<td>111.9881</td>
<td>321.978</td>
</tr>
<tr>
<td>50</td>
<td>88.6393</td>
<td>351.454</td>
<td>89.534</td>
<td>319.2664</td>
</tr>
<tr>
<td>55</td>
<td>80.5812</td>
<td>351.454</td>
<td>81.4004</td>
<td>319.2664</td>
</tr>
</tbody>
</table>

Table 5: Effect of frequency to the dynamic and static power

Figure 7: Waveform simulation of the design generated by DVE. Both simulations from figure 6 and figure 7 shows the same results as in table 4, this verifies that the design coding is correct where it is to replicate the Smith-Waterman algorithm.

D. Low Power Design

From the dynamic power formula, the frequency, $f$ plays a major role in determination of the power consumption, therefore by varying the clock cycle period of the design we can obtain the various value of power used by the design, this technique is called frequency scaling technique.
From figure 8, we can say that the IC design can work from clock period of 5ns to 55ns. This is because if the IC is applied to a clock period lower than 5ns, there will be negative value of slack at its critical path, this means that the data can’t get to its destination on time at the longest path of the circuit. If the IC is applied to greater clock period than 55ns, there will be too much slack, meaning that there is too much travel time wasted for the data to arrive at its destination pin.

Therefore to get optimal energy efficiency while considering some tradeoff to the timing, the IC must run in the range of 25ns-50ns (energy efficient region) clock period.

VI. CONCLUSION

This paper presents the design of a Low Powered DNA Sequence Alignment Accelerator. In this design, the power consumption are the main priority, it has been optimize by applying clock cycle reduction and frequency scaling technique to get the most energy efficient IC design and not neglecting the performance of the design. The design also does not affect the sensitivity of the Smith-Waterman algorithm itself. This design also have been developed using ASIC design flow until GDSII level which is the back end of the ASIC design flow.

REFERENCE

Figure 10: Schematic diagram generated by Design Vision (DV).

Figure 11: Physical layout of the design generated by Integrated Circuit Compiler (ICC).