JAUES

Journal of Al-Azhar University Engineering Sector

Vol.15, No. 56, July, 2020, 798-809

ACCELERATE MULTIPLE SEQUENCE ALIGNMENT BY CLUSTER ALGORITHM USING RECONFIGURABLE HARDWARE

Asmaa G. Seliem*[1], Hesham F. A.Hamed [1], and Wael AbouElWafa[2]
[1]Electrical Engineering Department, Minia University, Minia Egypt,
[2] Bio-Medical Engineering Department, Minia University, Minia, Egypt
*Corresponding Author E-mail::asmaa.seliem@s-mu.edu.eg

ABSTRACT

Sequence alignment of the human genome is a foundation problem in molecular biology; bioinformatician uses this similarity comparison between DNA, RNA, or protein sequences, to find the relationship between organisms or species, and for personal identification. With the next-generation sequencer, the rate of data generated is exponentially increasing the rate at which it cannot be computationally processed.

Traditional sequence alignment based on PC software's alignment tools requires several hours on state of the art workstations which cannot fulfill the increasing demand for this daily repetitive task. A hardware-based multiple sequence alignment architecture is described in this manuscript, expresses a comprehensive blueprint of the hardware implementation of small sequence alignment, for pair-wise global alignment technique to achieve high-throughput processing in a far shorter time using reconfigurable hardware, which provides better performance compared to the other platforms. The experiment was conducted for the simulation study to examine a Parallel Hardware Smith-Waterman algorithm based on Divide and Extended technique (PHSW-DE) on different FPGAs, which changing the curve of Big O notation, leads to GCUPS multiplied by a factor of $10^{(M-P),}$ and about 690x faster than used software sequential algorithm. This work will conclude a solution and provide a reference to further accelerating sequence alignment on an FPGA-based architecture using a parallel algorithm. As a conclusion for this procedure, the whole human genome multiple sequencing alignment of K-Mer length can be done in less than one hour, to achieve a local hardware sequence alignment in every bioinformatics laboratory.

KEY WORDS: PHSW-DE, Multiple Sequence Alignments, FPGA, Hardware-Based Alignment, and Smith Waterman.

تسريع محاذاة التسلسل المتعدد بواسطة العنقودية الخوارزمية باستخدام أجهزة قابلة لإعادة التكوين

أسماء جمال سليم*[١]، و هشام فتحى حامد[١]، و وائل أبو الوفا[٢]
[١] قسم الهندسة الكهربية ، كلية الهندسة ، جامعة المنيا ، المنيا، مصر
[٢]قسم الهندسة الطبية، كلية الهندسة ، جامعة المنيا ، المنيا، مصر
*البريد الاليكتروني للباحث الرئيسي: E-mail : asmaa.seliem@s-mu.edu.eg

الملخص

المحاذاة التسلسلية للجينوم البشري هي مشكلة أساسية في علم البيولوجيا الجزيئية ؛ يستخدم أخصائي المعلوماتبه البيولوجية مقارنة التشابه بين تسلسلات الحمض النووي ، الحمض النووي الريبي ، أو البروتين ، لإيجاد العلاقة بين الكائنات الحية . وللتعرف على الهوية الشخصية. باستخدام جهاز التسلسل من الجيل الثاني ، يزيد معدل البيانات التي يتم تحليلها يوميا زيادة كبيرة حتى اصبح حجمها لا يمكن معالجته بطريقة حسابية او ان التحليل يستغرق أياما حتى يكتمل.

تتطلب محاذاة التسلسلات التقليدية المرتكزة على برامج الكمبيوتر عدة ساعات أو أيام وذلك على أحدث المعامل التي لا يمكنها تلبية الطلب المتزايد على هذه المهمة المتكررة يوميا. يوصف في هذه المقالة بنية محاذاة التسلسل المتعدد القائم على الأجهزة (ال اف يى جى ايه)، ويعرض المخطط الشامل لتنفيذ تلك الأجهزة من محاذاة تسلسل معلوم ، لتقنية المحاذاة االمعتمده عالميا لتحقيق معالجة دقيقه في وقت أقصر بكثير عن برامج الكمبيوتر ، والتي توفر أداء أفضل مقارنةً بالطرق الأخرى. أجريت التجربة من أجل دراسة المحاكاة لفح ص خوارزمية سميث ووترمان المتوازية لتلك الأجهزة على أساس تقنية التقسيم والتجميع على مصفوفات مبرمجة حقليا مختلفة الإمكانيات منها الصناعية ومنها العسكرية الاستخدام ، مما يؤدي إلى تغيير منحنى تدوين كبير، مما يؤدي إلى زيادة معدل العمليات الحسابية في الثانية الواحدة ضرب بعامل عشره اس (ام-بى)، وحوالي ٦٩٠مره أسرع من الخوارزمية التسلسلية للبرامج المستخدمة. يستنتج هذا العمل حلاً ويوفر مرجعا لمزيد من محاذاة التسلسل المتسارع على المصفوفات المبرمجة حقليا باستخدام خوارزمية متوازية. وكنتيجة لهذا الإجراء ، يمكن إجراء محاذاة التسلسل المتعدد للجينوم البشري بأكمله بطول كمير (تسلسلات بطول ك)في أقل من ساعة واحدة ، لتحقيق توافق تسلسلي للأجهزة المحلية في كل مختبر معلوماتية حيوية بتكلفه اقل مما تستخدم حاليا.

الكلمات المفتاحية: محاذاة التسلسل المتعدد ، المصفوفات المبرمجة حقليا ، المحاذاة علي الأجهزة، سميث وترمان، والجينوم البشري.

INTRODUCTION

Genome sequencing is figuring out the order of DNA nucleotides, in a genome-the order of as, Cs, Gs, and Ts, which make up an organism's gene. DNA sequencing is necessary for massive projects (1000 Genomes project and human Genomes project(HGP)), sequencing an entire genome is done by Next-generation sequencing (NGS) machines, which solve big data problem by determining the nucleotide sequence of short DNA fragments (short reads) which lowers the cost and increases the throughput of DNA sequencing, helping scientists find genes much more easily and quickly. the DNA sequence alignment is one of the areas that required a competent and high-performance solution after the first responsive algorithm implemented in 1981 [14].

Comparative analysis of DNA and amino acid sequences is an essential component of biological research applied to biology, agriculture, and drug design [3].

When performing a search, a computationally demanding 'alignment score' needs to be calculated between the query sequence and each sequence in the gene banks database in order to find the most similar sequences. Additionally, multiple sequence alignment is an important research topic of bioinformatics — the results of multiple sequence alignment used for more complicated genetic research.

Specific applications like aligning genomes are required to compute the optimum alignment position and similarity. Nevertheless, there are massive data sets to be processed in a gene bank (such as NCBI's GenBank has doubling every six months [15]) to get the exact alignment of genes or segments. Therefore, there is a need to develop an enhanced system that can accomplish such tasks [2].

Theoretically, multiple sequence alignment (MSA)is an alignment of more than two sequences, which can be solved by multidimensional dynamic programming.

- MAS is a non-deterministic polynomial (NP)-hard problem; therefore, there is no efficient algorithm to solve it.
- Heuristic algorithms have been proposed to align multiple sequences. They generally perform well when the sequences to be aligned are not too distantly related to one another.
- Most of these heuristic algorithms, such as the Clustal W and OMEGA algorithms, use a progressive alignment method.

Multiple sequence alignments used for many reasons:

(1) detecting of the variability regions or conservation in a family of proteins,

(2) providing stronger evidence for structural and functional inferences, than pairwise similarity

(3) the first step in phylogenetic reconstruction, in RNA secondary structure prediction, and building profiles (probabilistic models) for protein families or DNA signals.

There are many tools to perform MSA based on different heuristics, as Progressive Alignment and Genetic Algorithms (GA) [1].
In [4], the metropolis criterion of simulated annealing is exploited as an initial step to generate a good population for the particle swarm optimization algorithm. The employment of this hybridization manner allows finding a good solution for the multiple sequence alignment problem.
In [7] present PMFastR, an algorithm which iteratively uses a sequencestructure alignment procedure to build a structure-based RNA multiple alignment from one sequence with known structure and a database of sequences from the same family. PMFastR also has low memory consumption allowing for the alignment of large sequences such as 16S and 23S rRNA. The algorithm also provides a method to utilize a multicore environment
In [5], Chellapilla et al. presented a method for multiple sequence alignment using evolutionary programming techniques. In [19], Isokawa et al. presented a method of multiple sequence alignment using a genetic algorithm (GA). In [18], Thompson et al. presented a method for improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positioning specific gap penalties. In [6], Notredame et al. presented a method called SAGA for sequence alignment by genetic algorithms.
Sandeep Hosangadi and Subhash Kak have multiple approaches from which to select the right one based on distance minimization or other considerations. For other alignment algorithms, statistical analysis relating to alignment solutions can be analyzed in a manner similar to others [13].
Solution based on dynamic programming algorithms has the complexity of O(LN), where L is the length of each sequence and, N is the number of sequences, thus making such solutions difficult for a large number of sequences. These accurate optimization methods are also costly in terms of time and memory. Which made multiple sequence alignments techniques rely on heuristic algorithms, most popular being CLUSTALW [21], T-Coffee [17], MUSCLE [9-10], and ProbCons [8]. For example, CLUSTALW [21] is estimated to take one year to align 5000 sequences of an average length of 350. MUSCLE is the fastest and somewhat most accurate multiple alignment tool to date. It claims to align 5000 synthetic sequences of an average length of 350 in 7 minutes on a desktop computer [9].
A parallel process was implemented to align DNA sequences reducing computational time. with a custom architecture implemented on the FPGA to provide a more robust solution to exploit the algorithms in full parallel while retaining a relatively low power profile [22].
We explain the design and application of a hardware implementation tool for multiple sequence alignment that has an accelerated performance and generates an optimal alignment result using the Zynq-7000 FPGA and PHSW-DE algorithm we invented in this paper for multiple sequence alignment.

Method

The application of dynamic programming to the alignment of n sequences involves a fixed number of operations for each cell of an n-dimensional matrix. The number of cells is the product of the lengths of the sequences to be aligned.

A. Basic Algorithm

Smith-Waterman (S-W) algorithm [20] is a common alternative to performing local alignment analysis between two biological sequences. The extensive usage is due to its capability to ensure higher accuracy levels for heuristic algorithms such as BLAST [11].
Furthermore, the accuracy of the S-W algorithm is ensured by a precise evaluation of the different events occurring when aligning the sequences. These events can be summarized in:
Match that occurs if two amino acids or nucleotides are equal in the compared sequences;
Mismatch, identified if two amino acids or nucleotides are different in the two compared sequences;
Insertion, if one or more amino acids or nucleotides can be observed in the compared sequence for the reference one;
Deletion, if one or more is missing amino acids or nucleotides in the compared sequence for the reference.

S-W allows finding the optimal local alignment between the two compared sequences, respectively named Query of length N and Reference of length M. Smith-Waterman algorithm finds the 'highest cost' alignment between two DNA sequences via dynamic programming. For an N-long query and M-long database sequence, they calculate an m x n matrix of score values. Positive matches are rewarded, while mismatches and gaps are penalized. Additionally, an affine gap-penalty model used in order to provide more flexibility and accuracy in alignment, S-W equations (1), and (2) where Si,j the elements of matrix, Gi,j is the similarity score of the comparison between query sequence and reference sequence and W is penalty of mismatch.. The detected alignment with the highest score is reconstructed in the last phase of the algorithm, starting from the maximum alignment Score to a traceback procedure. A variant of the S-W algorithm more suitable to sequences similarity analysis and called the S-W Affine Gap method proposed by Gotoh [12].

$$S_{(i,j)} = max\begin{Bmatrix} S_{(i-1,j-1)} + G_{(a_i b_j)} \\ S_{(i-1,j)} - W \\ S_{(i,j-1)} - W \\ 0 \end{Bmatrix} \quad (1)$$

$$G(a_i, b_i)\begin{cases} 5 & a_i = b_i \quad Match \\ -4 & a_i \neq b_i \quad Mismatch \end{cases} \quad (2)$$

with the initial condition $S_{(i,0)} = S_{(0,j)} = 0$.

B. Pseudo code for Optimizing Smith-Waterman Algorithm by Gotoh Algorithm

Algorithm1:

INPT:

Striang A: Query sequence of length n to be aligned
Striang B: Reference sequence of length m to be aligned
Gap extension =8, Gap insertion penalty =10
Match =5, MisMatch = -4
SWArray: n*m internal alignment matrix

Process:

Step 1: Initialization & score matrix

```
SW Array:= (OTHERS => "00")
Loop_I : for I =1:lenA
Loop_J:for j=1:lenB
  IF  strA(i)= strB(j)
    SWArray(I,j)= SWArray(i-1,j-1) + match
End
IF strA (i)/= strB(j)
    SWAraay(I,j)=SWArray(i-1,j-1)+MisMatch
End
row loop
        k_row := col_vector(above) - (GapPenalty + (GapExt * k1));
end loop
```

```
column loop
   k_col := Row_vector(k2) - (GapPenalty + (GapExt * k2));
end loop
End loop
End loop
// find highest score in matrix
For i=1:lenA
  For j= 1:lenB
    IF (SWArray(I,j) >Hiscore)
      Hiscore = SWArray(I,j)
      HIpos(1)=i
      Hipos(2)=j
    End
  End
   End
```

Step 2 : TB process

OUTPUT:

OptA: aligned Reference sequence

OptB: aligned Query sequence

C. Algorithm 2: Parallel Pseudo-code

Step 1: Divide reference sequence to p sub-sequences each equal to Query

Step 2: Calculate matrix and conduct alignment for p processes

Process (1)

Sequential SW steps (1), (2)

Process (2)

Sequential SW steps (1), (2)

⋮ ⋮

Process (p)

Sequential SW steps (1), (2)

Step 3:

Combine the result from each process

Opt A = Opt A1 & Opt A2 & Opt A3

Opt B = Opt B1 & Opt B2 & Opt B3

D. PHSW-DE algorithm for multiple sequence alignment

In order to explore our algorithm and parallelism of multiple sequence alignment on FPGA, we start by Assuming for simplicity that our R sequences R1, R2, … , Rk all have length N. If we were aligning two such sequences, we would need an array of size [n + 1] [n + 1] to account for all possible solutions. Similarly, for R sequences, we need an array of size [n + 1] k, or a k-dimensional array.

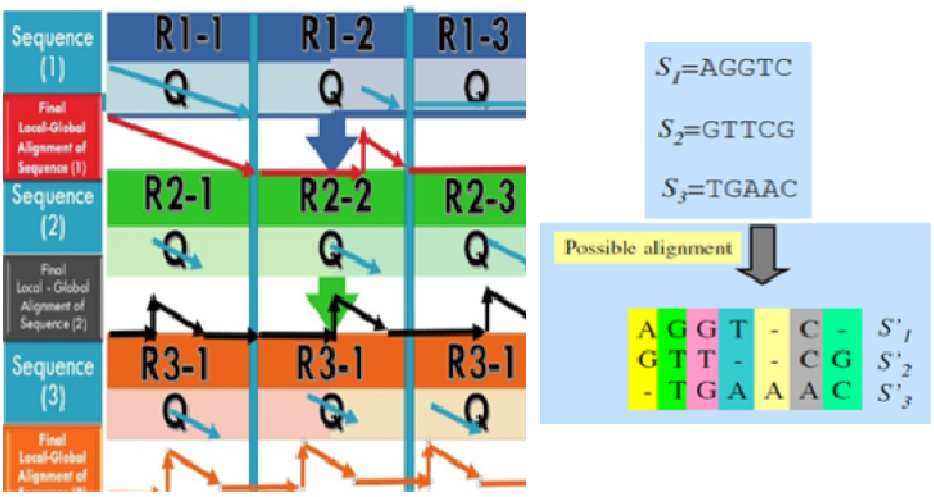

Figure 1) PHSW-DE algorithm block diagram and Example.

We construct The PHSW-DE algorithm as a local-global algorithm for multiple sequence alignments, for performance acceleration and more optimal solution of the alignment so simplicity, if we have three reference sequences from gene banks with a short read query sequence Q, we divided all reference sequences to sub-sequences equal to query sequence in length

R1: R1-1, R1-2, R1-3, R2: R2-1, R2-2, R2-3, and R3: R3-1, R3-2, R3-3

as shown in figure (1), then we start to compute Smith-Waterman matrix in parallel for every sub-sequence with the short read query in same time, then compute the traceback step in matrices in parallel finally we combine the final results for all nine alignments in three final combinations to get local-global final alignment to the short read query. An example for alignment multi chromosome with query show in figure (2).

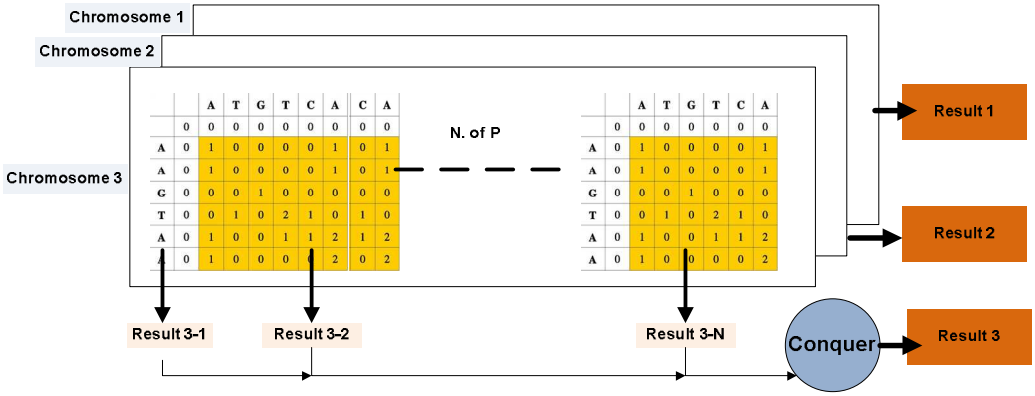

figure 2 Multiple sequence alignment illustrative example.

E. PHSW-DE Hardware Design Block Diagram

The hardware design for VHDL code inside Zynq-7000 figure (2), consists of five blocks each block has individual VHDL code and connected by its inputs and outputs. The first block is a buadrate9600, which adjusts system clock (100 MHz) to 9600 b/s, which the same rate of transmitted UART. Its inputs are clk and rst, and its output is clk1; it is the input clock for the Tx-UART block.

The second block is a Tx-UART this block responsible for transmitted data from Zynq-7000 to the PC; its inputs are clk1 come from Buadrate9600, rst, rdy, and Din where it is the data which will be transmitted and its output tx which carry the data to PC.The third block is a

p2s_128_byte this block convert data from parallel state to serial state so data compatible for transmitter UART, its inputs are clk come from Buadrate9600, rst, load and Din this the data come from the result of alignment (this the letter results) and its outputs are rdy, Dout it is the data go to Tx-UART.
The fourth block is a mapping block this block converts the result data (output alignment) from digital number (0, 1) to letters can help human read and send this letters to p2s block to become compatible for Tx-UART, its inputs are CLK system clock (100MHz), load, optA, and optB which the result data from alignment and its output is Dout which is data go to p2s block. The last block is a SWalign it is the leading block in our design which it does the SW algorithm, it takes the sequences, and out the alignment results, its input is switch load the inputs, and its outputs are optA and optB this is the result of alignment represented by (0, 1) go to mapping then go to p2s then go to Tx-UART and finial read the result which transmitted from Zynq-7000 to PC.

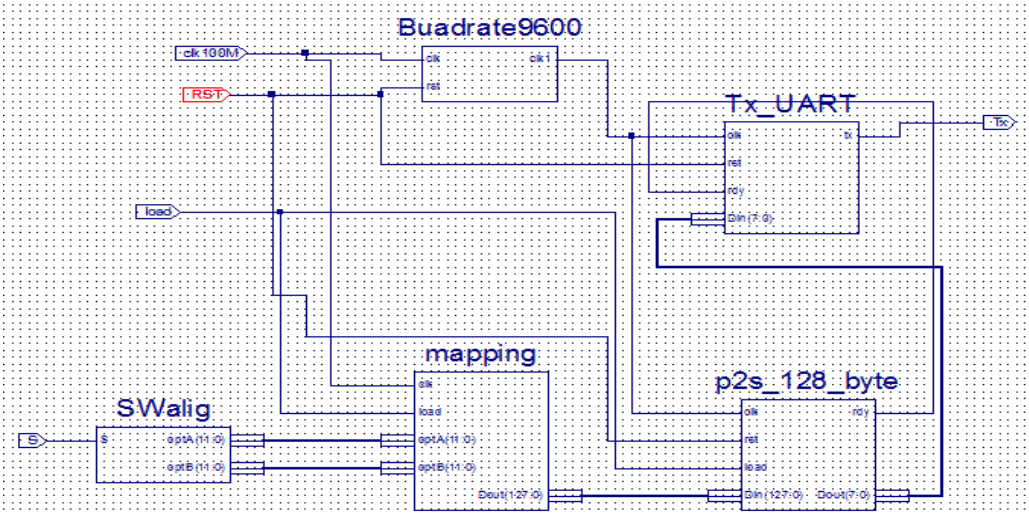

Figure 3 PHSW-DE algorithm hardware design.

F. Verification and Testing

several short alignment examples were executed both on a hardware simulator and by hand, checking to ensure the validity of the results. Which performed a local alignment mode. Calculating the score matrix by hand becomes tedious as the matrices grow substantial, so manual checking was only performed on matrices up to 6 x 6 nucleotides. Then double-check the output results by software package in C++.

G. PHSW-DE Simulation and Results:

Table 1 shows the several FPGA's implementations of new (PHSW-DE) For multiple sequence alignment (4*4 on 8 process) based on Divide and Extended technique which achieves better improvement; even the number of PEs is not as much as traditional smith waterman algorithm. Compared to the traditional algorithm (16*4), (PHSW-DE) also has more potential to be a speedup. Also, the performances of the implementations of different FPGA's are proportional to the higher frequency and more significant number of LUTs.

Table 1 The several FPGA implementations of Sequential Smith-Waterman Algorithm (16*4).

FPGA	LUT	Time (ns)	Memory (kilobyte)	Utilization	Frequency (MHz)
Kintex 7	92239	93.9	1370000	224%	36.7
Artix 7	92367	127	1370000	145%	27
Defense Artix 7	92412	136	1770000	145%	25.5
Defense Kintex 7	92701	105	1940000	45%	34
Defense Virtex 7	92701	105	1990000	45%	34
Defense Zynq	92559	-------	1770000	173%	34
Spartan 6	92772	215	1330000	3655%	19.6
Virtex 6	92316	114	1350000	198%	32
Virtex 7	92670	93.56	1370000	45%	36.8
Zynq-7000	92239	93.9	1370000	173%	36.7

Table 2 The several FPGA implementations of new (PHSW-DE) (4*4 on 8 process)

FPGA	LUT	Time (ns)	Memory (kilobytes)	Utilization	Frequency (MHz)
Kintex 7	71037	606.7	7508676	34%	2.5
Artix 7	67855	711.7	6832408	107%	2.2
Defense Artix 7	67768	771.8	7184500	106%	2
Defense Kintex 7	71037	606.7	7510316	34%	2.5
Defense Virtex 7	71037	606.7	7562484	34%	2.5
Defense Zynq	67892	598.7	7196124	127%	2.5
Spartan 6	67687	1189.3	6780588	2820%	1.25
Virtex 6	67820	646.4	6816484	145%	2.37
Virtex 7	91531	524.3	7072596	44%	2.8
Zynq-7000	67929	537.9	6848532	127%	2.8

Two main reasons limit the maximum number of alignment: The first one is the data dependence of the algorithm calculation itself. For instance, the maximum number of LUTs is constrained by the query and reference sequences' lengths. The other reason is that the available resource of FPGA is limited. Which changed from model to other.

The design goals and strategies provide a balanced optimization of performance results vs. runtime. The default property values correspond to the default values of each of the underlying implementation tools. This strategy keeps all properties in an unlocked state.

ISim debug the HDL code to verify that the design is running as expected. Debugging is accomplished through controlled execution of the source code to determine where problems. simulation of the six-sequence parallel processes shows in figure (3), and figure (4).

Furthermore, Zynq-7000 FPGA (our hardware design) result shows that thousands of folds' acceleration of the PHSW-DC algorithm than pure software. And almost same results as defense FPGA's. We performed the same input for each implementation and measured the time to complete the computations, our FPGA result shows that is about 690x faster than used software sequential algorithm [16].

Table 3 PURE SOFTWARE, 1XSCM AND 64X SCM COMPUTATION TIME

Number of cells	Pure software (ms)	1x scm (ms)	64x scm (ms)	PHSW-DC algorithm (ns)
١٦	0.120	0.0568	0.0124	17.45
٦٤	0.400	٠.٢٠٤	٠.٠١٢٨	69.8

Performance measurement for Smith-Waterman algorithm implementations is the number of billions of cell updates per second (GCUPS), which is calculated with the equation (3).

$$GCUPS = \frac{MN}{T \times 10^{-9}} \quad (3)$$

Where M and N are the sizes of both sequences R and Q respectively, and T is the runtime in seconds, Table (1) presents the execution times, Memory, LUT used which enplaning the utilization and efficiency of the FPGA in different parallel implementations on Xilinx.

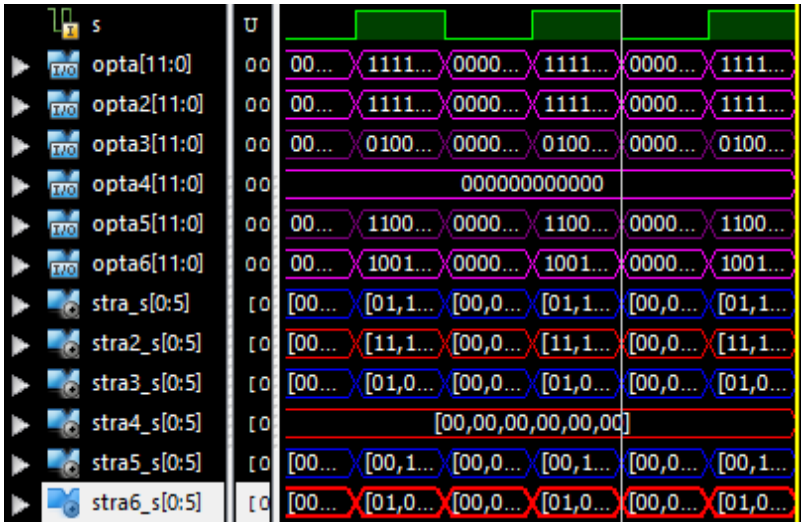

Figure 4 Multiple sequence alignment using PHSW-DE algorithm 6*6 six processes.

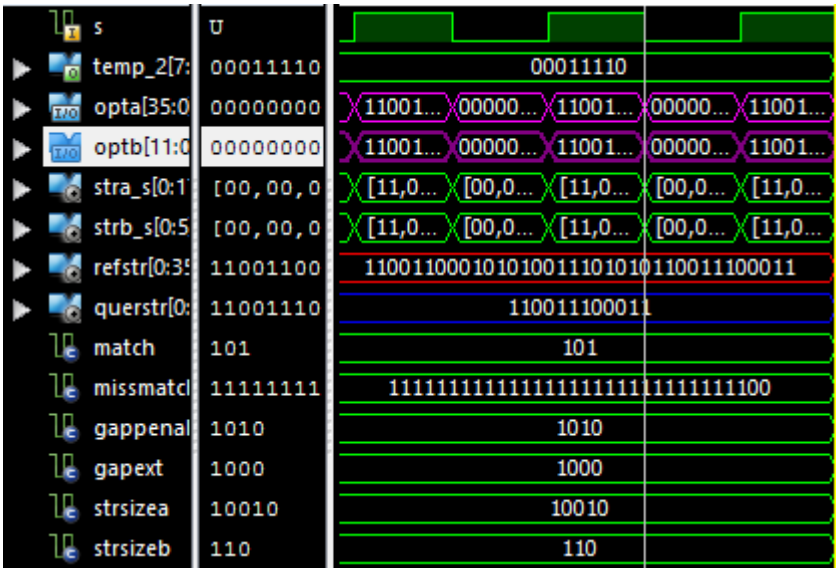

Figure 5 Traditional smith waterman algorithm 6*18 one process.

CONCLUSION

In this paper, we present a new complete parallel hardware Smith-Waterman algorithm for DNA sequences alignments by dividing the reference sequence into subsequences each subsequence equal to query sequence length in different processes and implement the algorithm in a Xilinx Zynq-7000 AP SoC XC7Z020-CLG484 FPGA

In the sequential SW algorithm, we need to store an (N+1) (M+1) matrix, but in our implementation, we divide the Reference in parallel processes by (N+1) ((M/p) +1).

the size of the matrix is more significant than the size of the total memory space and total FPGA LUT's, so the data cannot be handled according to the sequential algorithm. In comparison, our algorithm, each process needs to store (1/p) of the matrix, where p is the number of processes in the VHDL code. So, the data that were not be handled by existing algorithms can easily be handled using our cluster algorithm.

The experimental results were obtained with Intel(R) core™ i7-4700MQ CPU @ 2.40GHz with 8GB of memory with Xilinx simulator (ISim). Our parallel technique was able to obtain the alignment in maximum speedups of 69x then sequential algorithm in FPGA and hundreds of times faster than software, more efficiency and GCUPs when compared to a sequential smith-waterman algorithm.

The architecture exhibits high alignment throughput at optimum memory in contrast to existing alignment approaches considering the same sequence set while preserving the accuracy precision like an optimal alignment.

Furthermore, it increases performance, decreasing size and power consumption. Due to resource constraints of the ZYNQ-7000 FPGA platform.

The proposed algorithm is changing the curve of Big O notation, which leads to GCUPS multiplied by a factor of $10^{(M-P),}$ which is about 690x faster than used software sequential algorithm. As a conclusion for this procedure, the whole human genome multiple sequencing alignment can be done in less than an hour. which can be the future work as it needs more designs and complicated VHDL code, by using the same algorithm in clustering FPGA's to align the whole human genome in one shot.

REFERENCES

1. A. R. Amorim, J. M. V. Visotaky, A. D. G. Contessoto, L. A. Neves, R. C. G. De Souza, C. R. Valêncio ,&G. F. D. Zafalon. (2016). Performance Improvement of Genetic Algorithm for Multiple Sequence Alignment. Paper presented at the 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT).
2. G. Ananth Prabhu ,&G. Aithal. Automatic Parallelization for Parallel Architectures Using Smith Waterman Algorithm-Literature Review.
3. S. R. Bhalekar ,&P. Chilveri. (2015). A review: FPGA based word matching stage of BLASTN. Paper presented at the 2015 International Conference on Pervasive Computing (ICPC).
4. L. Chaabane. (2018). Resolving Multiple Sequence Alignment Problem Using an Intelligent Cooperative Algorithm. Paper presented at the 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS).
5. K. Chellapilla ,&G. B. Fogel. (1999). Multiple sequence alignment using evolutionary programming. Paper presented at the Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406).
6. S.-M. Chen, C.-H. Lin ,&S.-J. Chen. (2005). Multiple DNA sequence alignment based on genetic algorithms and divide-and-conquer techniques. International Journal of Applied Science and Engineering, 3(2), 89-100.
7. D. DeBlasio, J. Bruand ,&S. Zhang. (2011). A memory efficient method for structure-based RNA multiple alignment. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 9(1), 1-11.
8. C. B. Do, M. S. Mahabhashyam, M. Brudno ,&S. Batzoglou. (2005). ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome research, 15(2), 330-340.
9. R. C. Edgar. (2004). MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC bioinformatics, 5(1), 1.
10. R. C. Edgar. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic acids research, 32(5), 1792-1797.
11. W. Gish, W. Miller, E. Myers ,&D. J. Lipman. (1990). Basic local alignment search tool. J. Mol. Biol., 215, 403-410.
12. O. Gotoh. (1982). An improved algorithm for matching biological sequences. Journal of Molecular Biology, 162(3), 705-708. doi: https://doi.org/10.1016/0022-2836(82)90398-9
13. S. Hosangadi ,&S. Kak. (2012). An alignment algorithm for sequences. arXiv preprint arXiv:1210.8398.
14. S. A. M. A. Junid, M. F. M. Idros, A. H. A. Razak, F. N. Osman ,&N. M. Tahir. (2017, 10-12 March 2017). Parallel processing cell score design of linear gap penalty smith-waterman algorithm. Paper presented at the 2017 IEEE 13th International Colloquium on Signal Processing & its Applications (CSPA).
15. N. Khairudin, M. Haron, S. Al Junid, A. A. Halim, M. M. Idros ,&N. A. Razak. (2011). Design and Analysis of High Performance and Low Power Matrix Filling for DNA Sequence Alignment Accelerator Using ASIC Design Flow. Paper presented at the 2011 UKSim 5th European Symposium on Computer Modeling and Simulation.
16. I. T. Li, W. Shum ,&K. Truong. (2007). 160-fold acceleration of the Smith-Waterman algorithm using a field programmable gate array (FPGA). BMC bioinformatics, 8(1), 185.
17. C. Notredame, D. G. Higgins ,&J. Heringa. (2000). T-Coffee: A novel method for fast and accurate multiple sequence alignment. Journal of molecular biology, 302(1), 205-217.
18. J. C. Setubal, J. Meidanis ,&Setubal-Meidanis. (1997). Introduction to computational molecular biology: PWS Pub. Boston.
19. S. Siva Sathya, S. Kuppuswami ,&K. Syam Babu. (2008). Nomadic genetic algorithm for multiple sequence alignment (MSANGA). International Journal of Adaptive and Innovative Systems, 1(1), 44-59.

20. T. F. Smith ,&M. S. Waterman. (1981). Identification of common molecular subsequences. Journal of molecular biology, 147(1), 195-197.
21. J. D. Thompson, D. G. Higgins ,&T. J. Gibson. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic acids research, 22(22), 4673-4680.
22. A. Zeni, M. Crespi, L. D. Tucci ,&M. D. Santambrogio. (2019, 28 April-1 May 2019). An FPGA-Based Computing Infrastructure Tailored to Efficiently Scaffold Genome Sequences. Paper presented at the 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).