IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 23, 2021, accepted May 22, 2021, date of publication June 3, 2021, date of current version June 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3085997

MapReduce Model Using FPGA Acceleration for Chromosome Y Sequence Mapping

ASMAA G. SELIEM[1], HESHAM F. A. HAMED[2,3], AND WAEL ABOUELWAFA[4]

[1]Faculty of Engineering, Nahda University, Beni Suef 62764, Egypt
[2]Electrical Engineering Department, Faculty of Engineering, Minia University, Minya 61519, Egypt
[3]Telecommunication Engineering Department, Egyptian Russian University, Badr 11829, Egypt
[4]Bio-Medical Engineering Department, Faculty of Engineering, Minia University, Minya 61519, Egypt

Corresponding authors: Asmaa G. Seliem (asmaa.seliem@s-mu.edu.eg) and Wael Abouelwafa (wael.wafa@minia.edu.eg)

ABSTRACT Genome assemblies sequenced by a Whole Genome Shotgun (WGS) project predict an organism's function and history. Sequence alignment is the foundation of bioinformatics by a computational search through large genome sequence databases, which generally requires enormous amounts of memory and takes a long execution time. In this paper, an Optimized Smith-Waterman algorithm based on the Gotoh algorithm with an affine gap for accuracy alignment, the divide and conquer technique, and the MapReduce framework implemented to establish a parallel process. This model was implemented on Virtex 7 field-programmable gate arrays (FPGAs). These techniques provide a better performance, reduce the hardware requirements, improve the accuracy, increase the computational throughput, and accelerate the alignment process for big data available in a complete Y chromosome. The hardware proposed system can achieve high performance, low time consumption 1.699 ns, and decrease FPGA utilization for big data alignments Y chromosome is used as an example.

INDEX TERMS MapReduce, PHSW-DC, Gotoh, smith-waterman, Y chromosome.

I. INTRODUCTION

Bioinformatics is an emerging field focusing on developing computational methods (hardware and software) to collect, handle, and analyze biological data for DNA sequence mapping. Genome sequencing is used to determine the order of DNA nucleotides adenine (A), cytosine(C), guanine (G), and thymine (T), which form the genetic code for storing biological information [1]. The human genome (reference genome) comprises more than 3 billion of these nucleotides [2]. Human DNA samples are slashed into billions of small fragments, called reads, and a sequencer decides each read's nucleotide order.

An entire genome is sequenced by next-generation sequencing (NGS) machines that determine 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 [1].

Figure (1) shows the mapping process that makes up each strand to obtain an individual organism's genetic code.

The associate editor coordinating the review of this manuscript and approving it for publication was Alessandra Bertoldo.

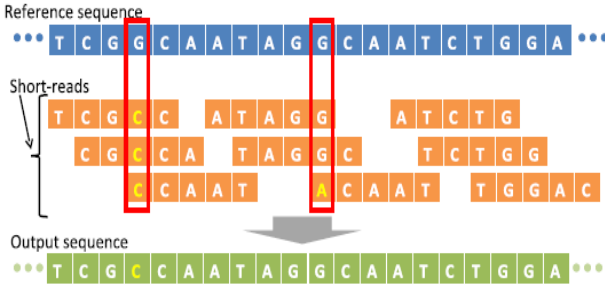

FIGURE 1. Mapping process after NGS.

The mapping process is challenging because of the large size of a given genome [1] and short DNA fragments from NGS by determines the location in the reference sequence to each read maps best.

There are two problems associated with such mapping; first, the mapping speed is low because of the big data. Second, the algorithm's accuracy for successfully mapping [3].

Scientists use two methods to solve the mapping problem-: heuristic methods and exact methods. Heuristic methods solve the mapping problem more rapidly than accurate

methods do. However, such methods, BLAST [4] and FASTA [5], suffer from accuracy. Therefore, bioinformatics researchers use dynamic programming, which is more efficient; the Smith-Waterman algorithm is a well-known sequence alignment algorithm for finding relationships between two sequences of all possible lengths.

Several methods have been utilized to accelerate the mapping process. Some of these methods approach this problem via software by using efficient algorithms. In contrast, the other uses special hardware or a hybrid approach, which combines both software and hardware approaches [1].

Software methods such as Maq [6], BFAST [7], Bowtie [8], and BWA [9] require days or weeks to map a whole genome. One major problem in software applications is low memory access. In CPUs incorporating hardware methods, the data paths are fixed, and the cache memory does not help much when the amount of data is immense. Thus, various computing architectures, such as GPUs and FPGAs, are used to exploit the traditional type of parallelism of each of these architectures [10]. CPU implementation is the slowest of these approaches because CPUs have limited pipelining and consecutive execution capabilities. GPU performance has increased with advancing chip technology, reflected in the much higher performance achieved with much lower power. In direct comparison with GPU implementation, FPGA implementation shows a much better GCUPS per Watt ratio.

Past endeavors incorporating short-read mapping utilizing FPGAs have yielded performance levels different from software tools [3].

SWPS3 [11] accelerates the Smith-Waterman algorithm through multi-threading and SIMD vector instructions in Intel's x86 or IBM's Cell architecture. CUDASW++ [12] is a CUDA implementation of the algorithm, and Altera provides an FPGA- implementation for its XD1000 platform [13]. Milik and Pulka reduced the processing time by optimizing the hardware architecture and benefitting FPGAs' properties [14].

Because physical constraints prevent the frequency scaling of CPUs and power consumption is becoming a critical problem, parallel processing has turned into the prevailing worldview for extensive scale figuring applications; thus, FPGAs have been widely explored for various high-performance computing applications in recent years. Compared with other computing platforms in parallel, such as GPGPUs and multicores clusters, the FPGA has advantages as follows: i) FPGAs are reconfigurable, and it is easy to change functionalities without changing the platform; ii) logic elements in FPGA work in an ordinarily fine-grained parallel manner with high flexibility similar to those in software approaches; and iii) FPGAs represents the best hardware devices that can follow Moore's law persistently [15].

The MapReduce model has been investigated in most parallel computing platforms in the past few years. For example, in its clusters, Google introduced the first MapReduce system [16].

There are numerous tools based on Burrows-Wheeler that use Hadoop MapReduce to boost BWA efficiency [17]. In the meantime, the high power consumption of multi-core chips represents a wall for the massive usage of such chips. For the general use of GPU platforms, the MapReduce framework has also been explored [18].

A parallel process was applied to align DNA sequences reducing computational time with a custom architecture implemented on the FPGA to provide a more robust solution to take advantage of the algorithms in complete parallel while retaining a comparatively low power profile [19].

However, a MapReduce framework was implemented on Cell clusters. Yeung *et al.* [20] adopted both an FPGA and GPU to implement a MapReduce framework.

This paper focuses on a scalable MapReduce framework based on FPGA to reduce the need for development cycles of FPGA-based computing for big data analysis. In this framework, multi-level parallelism is utilized, ranging from the bit-level to the task level. Demonstrating the proposed framework's practicability, we implement the enhanced "Smith-Waterman (SW) algorithm" using Gotoh with an affine gap and a parallel platform by the divide and conquer technique [15] implemented on FPGA-based architecture to accelerate the mapping process. The study's significance lies in improving performance and optimizing the amount of power used with reasonable accuracy.

II. MATERIALS AND METHODS

Mapping accuracy is a bioinformatics problem solved in this paper by optimizing the Smith-Waterman algorithm using the Gotoh affine gap with parallel hardware implementation. Big data is another problem that is solved by three different techniques divide and conquer, MapReduce model, and FPGA implementation. These methods are used for the human Y chromosome, the sex-determining chromosome in many species; males have one Y chromosome and one X chromosome, while females have two X chromosomes.

The human Y chromosome is specially presented to high mutation rates due to the environment in which it is housed. It is gone solely through sperm, which undergoes multiple cell divisions during gametogenesis. Each cellular division gives a further chance to aggregate base-pair mutations. Additionally, sperm are secured in the uncommonly oxidative condition of the testis, which empowers further transformation. However, her extraordinary reference gets this number for the relative change rates in male and female germlines for human genealogy.

A significant strata forming mechanism is an inversion of the Y chromosome, which suppresses X - Y recombination in males in the inversion zone. The non-recombining regions on the X and Y chromosomes develop and diverge independently after each inversion. Also, in the absence of male recombination, non-recombining regions accumulate DNA elements, such as transposable or repetitive elements, and sequences with GC content shifts [21]

The Y chromosome genes count estimation according to NCBI are (73) Protein-coding genes & (122) Non-coding RNA genes and (400) Pseudogenes (NCBI Reference Sequence: NC_000024.10) with a length of (57,227,415 bps). According to The human reference genome GRCh38, which was released from the Genome Reference Consortium, all these techniques will be described.

A. OPTIMIZATION OF SMITH-WATERMAN ALGORITHM BY GOTOH ALGORITHM

The Smith-Waterman algorithm is used widely due to its capability to ensure high accuracy [22]. On the other hand, the Gotoh algorithm is considered the more restricted case of affine gap costs. Therefore, the Gotoh and Smith-Waterman algorithms are combined into one approach to computes the local affine alignment of two sequences using affine gap scoring.

Gotoh's algorithm's significant advantage is finding the minimum cost in O(MN) steps to align two sequences. Gotoh's algorithm attempts to discover just a single (rather than all) of the ideal alignments. However, Taylor described a modification of Gotoh's algorithm that consistently finds at least one optimal alignment [23]. The Gotoh and Smith-Waterman algorithms are combined as follows:

- Initialize matri
- Make a calculation matrix and a high score.
- Traceback alignment (TB process).

The first step, M+1 by N+1 matrix, is generated by a dynamic programming approach, where M is the length of the reference sequence complete Y chromosome, and N is the length of the query sequence. The first column and row are filled with zeros according to equation (1); the matrix cells score calculated and loaded using the local alignment score D(i,j) beginning from the top-left corner where S(i,j) denotes the substitution score value obtained by aligning character Reference Ri against character Query Qj according to equation (2) and E, F represents the gap penalty cost g and β (the cost of aligning a character to space, also known as gap insertion, deletions, or replacement) to give more flexibility and accuracy in alignment; these factors can be calculated by equations (3,4) for the rows and columns to get the final result as indicated by equation (5).

$$D_{(i,0)} = D_{(0,j)} = 0 \tag{1}$$

$$S(a_i, b_i) = \begin{cases} 5 & a_i = b_i \ \ match \\ -4 & a_i \neq b_i \ \ mismatch \end{cases} \tag{2}$$

$$E_{i,j} = max \begin{cases} D_{i-1,j} + g \\ E_{i-1,j} + \beta \end{cases} \tag{3}$$

$$F_{i,j} = max \begin{cases} D_{i,j-1} + g \\ F_{i,j-1} + \beta \end{cases} \tag{4}$$

$$D_{i,j} = max \begin{cases} D_{i-1,j-1} + S(a,b) \\ E_{i,j} \\ F_{i,j} \\ 0 \end{cases} \tag{5}$$

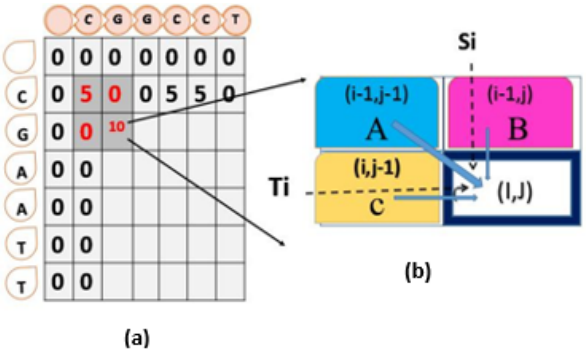

FIGURE 2. (a) Smith-waterman matrix fill, and (b) cell calculation procedure.

After matrix calculation, backtracing is performed from the highest score; several paths lead back to the original point, as shown in Figure (2).

The Smith-Waterman algorithm optimized by the Gotoh algorithm [24] has a running time of O(N2) and a memory requirement of O(N), where O refers to complexity, and N is the sequence size. Given the large size of genomic datasets, the computation time can be significantly decreased.

This paper's remainder is organized as follows; Section II describes the materials and methods used, including the Smith-Waterman algorithm optimized by the Gotoh algorithm, parallel hardware Smith-Waterman algorithm with the divide and conquer technique, the MapReduce pattern for big data problem, hardware Implementation on FPGAs, Section III describes the results of the paper. Finally, section IV provides a discussion of the findings.

B. DIVIDE AND CONQUER TECHNIQUE

Divide and conquer is a robust algorithm for solving conceptually complex problems: all the technique requires is to divide an issue into sub-issues, which decreases the size of the case to be solved.

The proposed algorithm uses D&C to divide the complete Y chromosome's length into (N* the size of the query sequence) and conquer each alignment process's results to get the final result. This division reduces the complexity of the main structure and reduces the amount of memory used.

Each subsequence will align with the Query in many processes at the same time. Then, the other subsequences are pipelined to the alignment in the same manner, as shown in figure (3).

The D&C technique is naturally adapted for execution in multi-processor Frameworks, especially shared-memory systems. The communication of data between processors should be arranged ahead of time, as certain sub-issues can be executed on the various process. Furthermore, the D&C technique naturally tends to make efficient use of memory caches. The reason is that once a sub-issue is sufficiently small, it can be illuminated inside the reserve without being susceptible to slower principle memory access.

Algorithm 1 Optimized Smith-Waterman Algorithm by Gotoh

```
##Inputs
RSequence ≫ Reference sequence of length m to be aligned
QSequence ≫ Query sequence of length n to be aligned
Gap plenty = 10
Match = 5 and mismatch = -4
load data sequence and generate the intial aligment Matrix
for i < m+1
  for j < n+1
    IF RSequence(i)= QSequence(j)
      cell(i,j) = cell(i-1,j-1) + match
    End
    IF RSequence (i)/= QSequence(j)
      cell(i,j) = cell(i-1,j-1) + Mismatch
    End
    for K1 < i-1
      loop for rolls over the row to get maximum value in the row begin from
    k_row := cell(above) –(GapPenalty + (GapExt ∗ k1)) value
    end
    for K2 < j-1
      loop for rolls over the column to get maximum value in the column begin from
      k_col := Row_vector(left) – (GapPenalty + (GapExt ∗ k2)) value
    end
  end
end
for i < m+1
    for j < n+1
      get the high Score of Matrix its position and
    End
End
 for i < position_row of highScore
   for j < position_column of highScore
     get the alignment sequence by trace back processing
   End
End
outputs
RSequence_output align reference sequence
QSequence_output align query sequence
```

C. MAPREDUCE PATTERN FOR BIG DATA PROBLEM

The proposed model gives two natives, map and reduce. The input data to a computing task are part of many <add, value> pairs and a map function processes them to create a set of intermediate <add, value> pairs. Intermediate pairs with similar intermediate keys are grouped and passed to the reduce function. Then the MapReduce runtime framework addresses parallel execution by issuing multiple maps and reducing tasks to computation nodes.

Algorithm 2 Parallel Pseudo-code

```
Step 1: Divide reference sequence to p Query sub-sequences
Step 2: Compute matrix and conduct alignment for p processes
Process (1)
Optimized smith waterman algorithm by Gotoh
Process (2)
Optimized smith waterman algorithm by Gotoh
⋮
Process (p)
Optimized smith waterman algorithm by Gotoh
Step 3:
Combine the result from each process
OptA _alignment A = Opt A1 & Opt A2 & Opt A3
Opt B_alignment = Opt B1 & Opt B2 & Opt B3
```

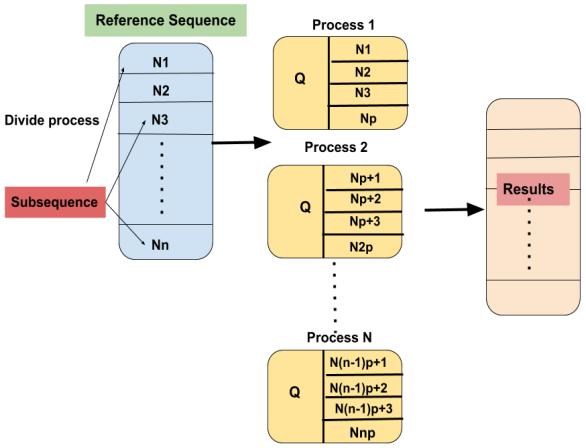

FIGURE 3. The proposed hardware, divide, and conquer technique with the parallel process for alignment where n is the number of subsequence reference, and p is the number of processes.

Proposed FPGA MapReduce (FPMR) Framework:

we present five different forms of the MapReduce framework. Furthermore, dedicated processors are designed for various applications under the FPMR framework for FPGA.

Five MapReduce Design Patterns

1. Input-Map-Reduce-Outpu
2. Input-Map-Output
3. Input-Multiple Maps-Reduce-Outpu
4. Input-Map-Combiner-Reduce-Output
5. Input-Multiple Maps-Multiple Reductions Output

The proposed model use Pattern (5) to take input from many files with different schema, different processes are carried out in the reducers, and the output is then combined in the end stage, which we use in our system as shown in Figure (4).

Proposed MapReduce Data Flow

The MapReduce data flow can be simplified as follows:

In the initial < add, the local memory in FPGA prepares value > pairs. Then the mappers and reducers perform operations on the FPGA.

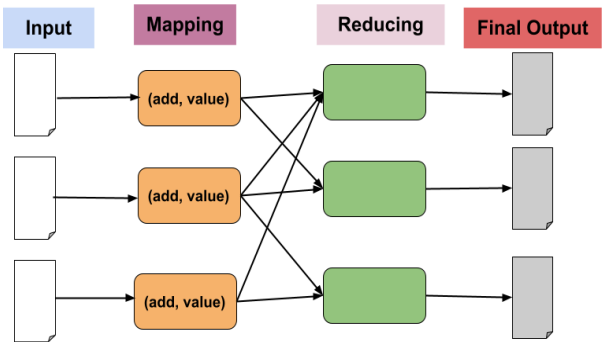

FIGURE 4. The five patterns of MapReduce.

The mappers process the initial input < add,value > pairs and create intermediate < add, value > pairs. The reducers then join the intermediate pairs to acquire the last results. In bioinformatics applications, reducers' outputs should be further processed to obtain a single result to resolve this issue using the D&C technique.

The data controller is responsible for communicating with the FPGA, transmitting data to the mappers, and receiving the reducers' data.

The basic workflow and scheduling policy are as follows:

1. Produce < add, value > pairs on the FPGA local memory.
2. Write the FPGA registry configuration parameters.
3. The processor scheduler assigns each mapper.
4. Mappers process the assigned < add, value > pairs and store the generated intermediate <add, value> pairs in the local memory controlled by a data controller.
5. When all the tasks are completed, the results are returned to the data controller's primary host memory.

Processor

There are two types of processors on a chip designed, mappers and reducers. Mappers and reducers are specifically designed according to the target chromosome Y. The processing times of mappers vary depending on the number of mappers, data size, and reducers assigned according to genes separated. Then the data are exchanged between mappers and reducers.

Storage Hierarchy and Data Controller

There are two levels of storage in our framework. The first level is the local memory divided into two parts; one stores the initial <add, value> pairs. The second part stores the intermediate <add, value> pairs and serves as the shared memory for mappers and reducer; the second level is the processor's registry file designed for temporary variables, configuration parameters, and results.

Local memory

Local memory is implemented in on-chip RAMs. The intermediate results acquire from a mapper are stored in the local memory, and the reducer will obtain the intermediate data from the local memory. Thus, multiple RAMs are implemented, and They can be reached through mappers and reducers simultaneously.

Register file

The register file stores the temporary variables. The framework's parameters and the results during processor operation.

Data controller

The data controller is responsible for the following three functions: 1) dispatching of requested data to mapper;

2) communicate data between mapper and reducer, and 3) store the output data from reducers.

D. HARDWARE IMPLEMENTATION ON FPGA

The proposed system is simulated on Virtex-7, which involves a TSMC 28 nm HPL process, a 40 nm V6, 6.8 billion transistors, 2 million logic cells. Technology Integrated 12-bit ADCs in 17 channels at 1 MSPS, low-power mode: 0.9 V.

The proposed solution assigns 2 bits for each DNA character for faster alignment. The design was synthesized from VHDL using Xilinx software tools. ISE 14.7 is used to write and simulate the VHDL code.

The symbols are coded as follows (A = 00, C = 01, G = 10, and T = 11). Therefore, the input for sequences N or M was implemented in the simulation based on a size of 2-bits; for example, for M = A T C G, the system can implement M = 00, 11, 01,10, Similarly, for N = T C G A, for example, the system can implement N = 11, 01, 10, 00.

In this experimental local memory of FPGA is divided into two parts. Every part of local memory has two mappers, and each one goes to four reducers used for calculation processing.

The proposed algorithm comprises three main hardware components: memories, a processing unit, and a comparator—the allocated local memories store reference sequences and query sequences involving less utilization and higher speed than other systems. The memory size bottleneck is a significant problem when the forward process is implemented on an FPGA.

The processing unit computes the array value for each character by an optimized smith waterman algorithm. Values are computed for every combination of deletions, insertions, and matches. Affine gap penalties make the alignments more biologically relevant, as the matrix fill stage takes most of the overall processing time.

Finally, the comparator performs the traceback step to compare the query and reference sequences and obtains the alignment results. The design does depend on a clock, thus making it a synchronized system.

At the beginning of the optimized S-W flow chart, the reference sequence A(i) and query sequence B(j) are loaded and then compared to determine whether they match, and the gap penalty (gap constant) is subtracted. Figure (5) displays the HPSW-DC algorithm design, which is implemented on an FPGA. The figure shows how two sequences are aligned where $_{Mi}$, Ni is an input on the comparator, and the result goes to the first adder and is then added to the value of a diagonal cell. The first adder output is input to a maximum value selection step with the left cell's value added to the gap. Next, the maximum value output is input to another

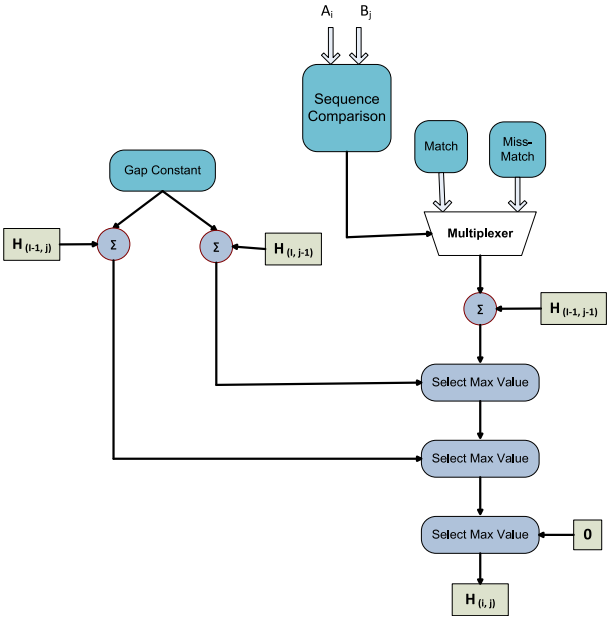

FIGURE 5. Flow chart of the optimized S-W algorithm.

maximum value selection step with the value of the upper cell, which is added to the gap. Finally, the second maximum value selection step's output is input to a final maximum value selection step with zero to obtain the final result $H_{i,}j$.

III. RESULTS

In this experiment, we select chromosome Y, the most mutated chromosome in the human genome, which is very useful for bioinformatics research; to increase the accuracy for alignment, we optimize the smith waterman algorithm by Gotoh affine gap, which increases the accuracy for the mapping process. Furthermore, adding the divide and conquer technique with parallel processing decreases the bio O notation from N2 to PN. Then the last Stage to solve the big data problem chromosome Y this experiment using MapReduce pattern (four reducers for each mapper) to handle this data on FPGA and transferring all this information to VHDL code implemented on Virtex 7. We are achieving by the end 1.69 ns for chromosome Y (57 Mbps).

The schematic diagrams of the complete proposed system shown in figure (6) show Local memories that store chromosome files and the counter that shares the address and Smith-Waterman block for processing, and figure (7) demonstrate the UART block that obtains output from the Smith-Waterman block and outputs the result to the PC second stage. First, the Y chromosome is converted to a FASTA file with 0 or 1 bit (.coe) files to FPGA memory. Each file has 571000 lines with 48bits per line; a Python program performs this transformation from the original file. Next, this file is loaded into two different memories (the size of each is 3.5MB) on the FPGA as a part of the mapping process using a counter to assign different addresses for every line in these

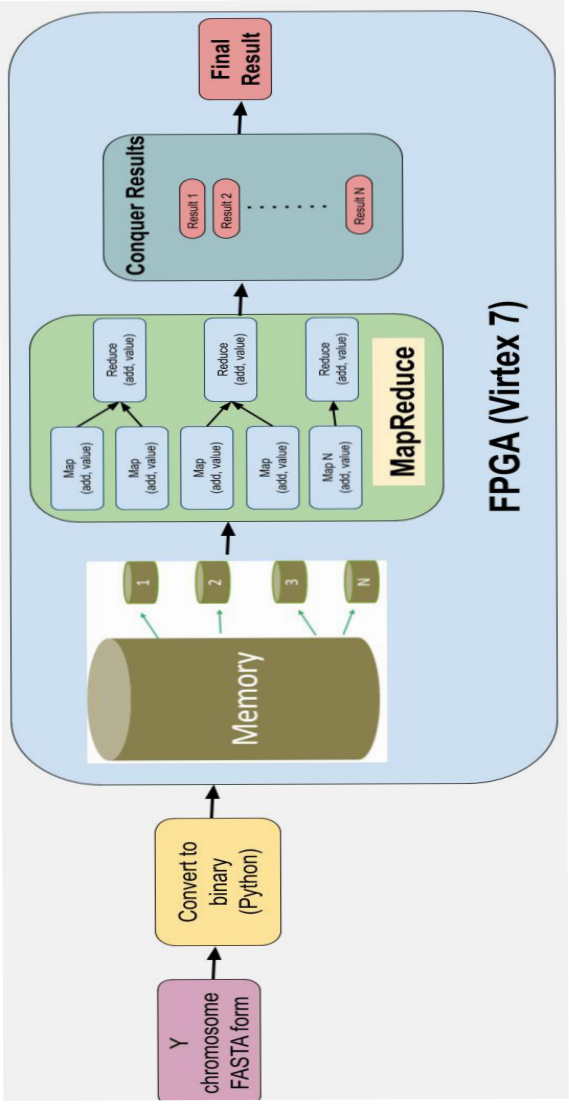

FIGURE 6. The MapReduce used pattern.

files with the value stored in the memory; the data are then transferred to the output of this memory to the two reducers (SW_Gquery6_gene24) which execute an optimization algorithm to calculate the final alignment via four parallel processes. Finally, the divided results are then transferred to obtain the Y chromosome's final alignment using the counter to map the previous process's address and values.

Table 1 shows an up to date comparison between this paper, and other implementations CloudSW has outstanding performance as a cloud software (0.1224 sec.) and achieves up to 621 times speedup over SparkSW(1 min. 11sec.) with an Execution time taken by SPARK-MSNA for datasets with different similarity. Datasets were of equal size (3.75MB) [30].

OpenCL-based FPGA predicts performance and track optimizations on different platforms used in GCUPS as

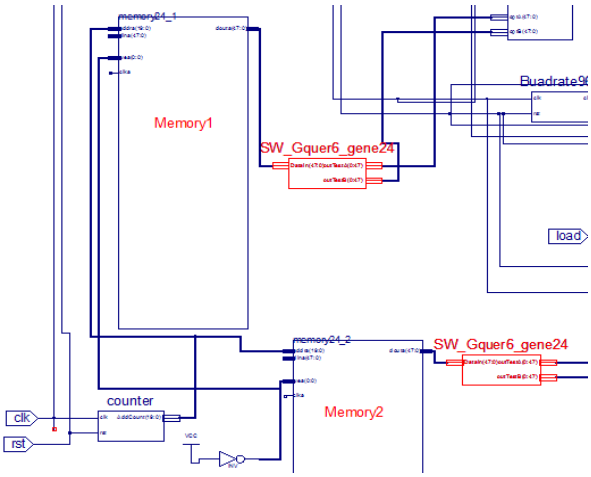

FIGURE 7. First stage.

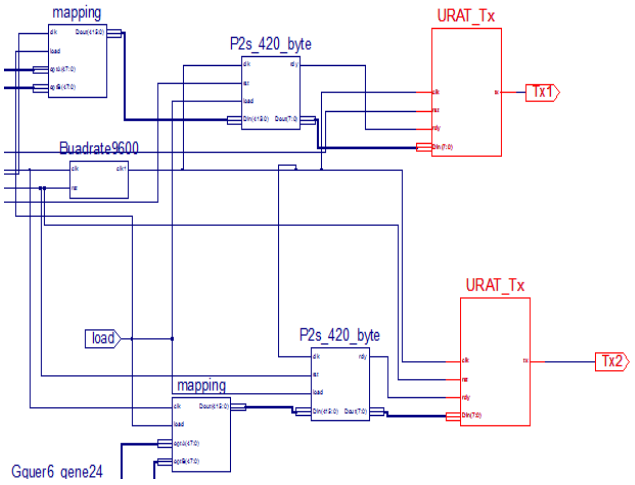

FIGURE 8. Second Stage.

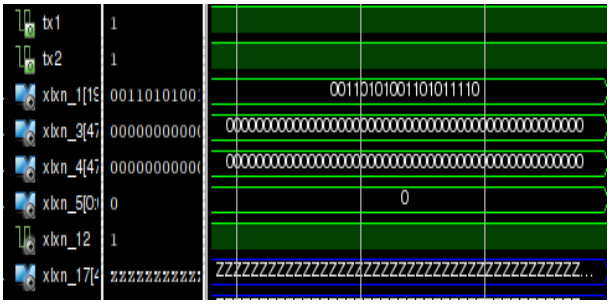

FIGURE 9. Simulation results were obtained for the alignment of chromosome Y by four parallel processes.

a performance indicator, which is not accurate as a separate one because the size of reference and Query is the most critical factor for comparison.

In the last column, an innovative reconfigurable supercomputing platform – XD1000 is developed by Xtreme Data Inc. to exploit FPGA technology's rapid progress and the high-performance of Hyper-Transport. The protocols used in this paper are more accurate with big data sequences and fast alignment.

TABLE 1. compares PHSW-DC and the other methods for sequence alignment with time, size, frequency, and types of kit.

Algorithm	Time consume	Chromosome size Mbp		Frequency (MHz)	Kit
PHSW-DC (FPGA)	1.699 ns	57,227,415 bps		582 MHz	Virtex 7
CloudSW [26]	621 times speedup over SparkSW (1 min.11sec.)	16384 million char		NA	Remote cluster in the Alibaba Aliyun Cloud
OpenCL-based design SDAccel [27]	1.72x speedup over FPGA	Query 256 char	Ref 1million char	NA	kintex
Intel FPGAs using OpenCL [28]	theoretical performance of 214 GCUPS	Query 131	Ref 100,000 pairwise	137 – 193 MHz	(BittWare A10PL4) & Intel A10 REF
XD1000 is developed by Xtreme Data [29]	0.1808 sec	65536 BP		66.7 MHz	FPGA

IV. CONCLUSION

Smith-Waterman is one of the most important and accurate algorithms for inference of DNA sequencing to predict an organism's function and history, but it requires intensive computation. Furthermore, the time to complete the alignment, to achieve short execution time, researchers need to have access to expensive high-performance computing clusters, which are usually not available; faster and more efficient sequencing hardware will continue making DNA sequencing cheaper and amenable for use in forensics, medical diagnosis, and even biological computing. Moreover, FPGA exemplifies one of the best attempts to handle large-scale bioinformatics problems by parallel computing systems. However, various types of parallelism characterize it. This paper solved this problem by significantly reducing the computing time using the divide and concur technique for algorithm processes in parallel sequence alignment that is implemented as well as data-level parallelism MapReduce framework for big data (Y chromosome) without losing the inference accuracy the design uses Smith-Waterman algorithm optimized by the Gotoh algorithm to obtain higher efficiency and high accuracy. Virtex XC7VX485T-3ffg1157 is used to implement the optimized algorithm for chromosome Y's alignment to demonstrate the effect of the MapReduce model in achieving low time consumption of 1.699 ns, decreasing FPGA utilization.

The design uses advanced parallel computing illustrated by FPGA exhibit not only new possibilities to accelerate bioinformatics algorithms.

REFERENCES

[1] S. R. Bhalekar and P. G. Chilveri, "A review: FPGA based word matching stage of BLASTN," in *Proc. Int. Conf. Pervasive Comput. (ICPC)*, Jan. 2015, pp. 1–4.

[2] Y.-T. Chen, J. Cong, J. Lei, and P. Wei, "A novel high-throughput acceleration engine for read alignment," in *Proc. IEEE 23rd Annu. Int. Symp. Field-Program. Custom Comput. Mach.*, May 2015, pp. 199–202.

[3] C. B. Olson, M. Kim, C. Clauson, B. Kogon, C. Ebeling, S. Hauck, and W. L. Ruzzo, "Hardware acceleration of short read mapping," in *Proc. IEEE 20th Int. Symp. Field-Program. Custom Comput. Mach.*, Apr. 2012, pp. 161–168.

[4] R. Dharayani, W. C. Wibowo, Y. Ruldeviyani, and A. Gandhi, "Genomic anomaly searching with BLAST algorithm using MapReduce framework in big data platform," in *Proc. Int. Workshop Big Data Inf. Secur. (IWBIS)*, Oct. 2019, pp. 27–32.

[5] M. Aledhari, M. Di Pierro, and F. Saeed, "A Fourier-based data minimization algorithm for fast and secure transfer of big genomic datasets," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, Jul. 2018, pp. 128–134.

[6] H. Li, J. Ruan, and R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," *Genome Res.*, vol. 18, pp. 1851–1858, Nov. 2008.

[7] N. Homer, B. Merriman, and S. F. Nelson, "BFAST: An alignment tool for large scale genome resequencing," *PLoS ONE*, vol. 4, no. 11, p. e7767, Nov. 2009.

[8] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biol.*, vol. 10, no. 3, p. R25, May 2009.

[9] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinformatics*, vol. 25, pp. 1754–1760, Jul. 2009.

[10] N. Cadenelli, Z. Jaksić, J. Polo, and D. Carrera, "Considerations in using OpenCL on GPUs and FPGAs for throughput-oriented genomics workloads," *Future Gener. Comput. Syst.*, vol. 94, pp. 148–159, May 2019.

[11] A. Szalkowski, C. Ledergerber, P. Krähenbühl, and C. Dessimoz, "SWPS3—Fast multi-threaded vectorized smith-waterman for IBM Cell/B.E. and ×86/SSE2," *BMC Res. Notes*, vol. 1, no. 1, p. 107, Oct. 2008.

[12] Y. Liu, A. Wirawan, and B. Schmidt, "CUDASW++ 3.0: Accelerating smith-waterman protein database search by coupling CPU and GPU SIMD instructions," *BMC Bioinf.*, vol. 14, no. 1, p. 117, Apr. 2013.

[13] P. Zhang, G. Tan, and G. R. Gao, "Implementation of the Smith-Waterman algorithm on a reconfigurable supercomputing platform," in *Proc. SC*, 2007, pp. 39–48.

[14] L. Di Tucci, K. O'Brien, M. Blott, and M. Santambrogio, "Architectural optimizations for high performance and energy efficient Smith-Waterman implementation on FPGAs using OpenCL," in *Proc. DATE*, 2017, pp. 716–721.

[15] Y. Shan, B. Wang, J. Yan, Y. Wang, N. Xu, and H. Yang, "FPMR: MapReduce framework on FPGA," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2010, pp. 93–102.

[16] S. Maitrey and C. K. Jha, "MapReduce: Simplified data analysis of big data," *Procedia Comput. Sci.*, vol. 57, pp. 563–571, 2015.

[17] M. AlJame and I. Ahmad, "DNA short read alignment on apache spark," *Appl. Comput. Inform.*, 2020.

[18] S. C. Purbarani, H. R. Sanabila, A. Bowolaksono, and B. Wiweko, "A survey of whole genome alignment tools and frameworks based on Hadoop's MapReduce," in *Proc. Int. Workshop Big Data Inf. Secur. (IWBIS)*, Oct. 2016, pp. 65–70.

[19] A. Zeni, M. Crespi, L. Di Tucci, and M. D. Santambrogio, "An FPGA-based computing infrastructure tailored to efficiently scaffold genome sequences," in *Proc. IEEE 27th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2019, p. 333.

[20] J. H. C. Yeung, C. C. Tsang, K. H. Tsoi, B. S. H. Kwan, C. C. C. Cheung, A. P. C. Chan, and P. H. W. Leong, "Map-reduce as a programming model for custom computing machines," in *Proc. 16th Int. Symp. Field-Program. Custom Comput. Mach.*, 2008, pp. 149–159.

[21] R. S. Pandey, M. A. Wilson Sayres, and R. K. Azad, "Detecting evolutionary strata on the human X chromosome in the absence of gametologous Y-linked sequences," *Genome Biol. Evol.*, vol. 5, pp. 1863–1871, 2013.

[22] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.

[23] S. Altschul and B. Erickson, "Optimal sequence alignment using affine gap costs," *Bull. Math. Biol.*, vol. 48, nos. 5–6, pp. 603–616, 1986.

[24] O. Gotoh, "An improved algorithm for matching biological sequences," *J. Mol. Biol.*, vol. 162, no. 3, pp. 705–708, Dec. 1982.

[25] S. Pal, S. Mondal, G. Das, S. Khatua, and Z. Ghosh, "Big data in biology: The hope and present-day challenges in it," *Gene Rep.*, vol. 21, Dec. 2020, Art. no. 100869.

[26] B. Xu, C. Li, H. Zhuang, J. Wang, Q. Wang, and X. Zhou, "Efficient distributed smith-waterman algorithm based on apache spark," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 608–615.

[27] L. Di Tucci, K. O'Brien, M. Blott, and M. D. Santambrogio, "Architectural optimizations for high performance and energy efficient smith-waterman implementation on FPGAs using OpenCL," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 716–721.

[28] E. Houtgast, V.-M. Sima, and Z. Al-Ars, "High performance streaming smith-waterman implementation with implicit synchronization on intel FPGA using OpenCL," in *Proc. IEEE 17th Int. Conf. Bioinf. Bioeng. (BIBE)*, Oct. 2017, pp. 492–496.

[29] P. Zhang, G. Tan, and G. R. Gao, "Implementation of the smith-waterman algorithm on a reconfigurable supercomputing platform," in *Proc. 1st Int. Workshop High-Perform. Reconfigurable Comput. Technol. Appl. Held Conjunct (SC-HPRCTA)*, 2007, pp. 39–48.

[30] V. Vineetha, C. L. Biji, and A. S. Nair, "SPARK-MSNA: Efficient algorithm on apache spark for aligning multiple similar DNA/RNA sequences with supervised learning," *Sci. Rep.*, vol. 9, no. 1, pp. 1–11, Dec. 2019.

ASMAA G. SELIEM was born in Minia, Egypt, in 1990. She received the B.Sc. degree in electronics and communication engineering from Minia University, Minia, in 2012, the Diploma degree from the Information Technology Institute (ITI) Open Source Track, in June 2015, and the M.Sc. and Ph.D. degrees in electronics and communications engineering from Minia University, in 2016 and 2020, respectively. She was a Lecturer with the Department of Electrical Engineering, Nahda University, Beni Suef, Egypt, in 2016.

HESHAM F. A. HAMED was born in Giza, Egypt, in 1966. He received the B.Sc. degree in electrical engineering and the M.Sc. and Ph.D. degrees in electronics and communications engineering from Minia University, Minia, Egypt, in 1989, 1993, and 1997, respectively. He was a Teacher Assistant with the Department of Electrical Engineering, Minia University, from 1989 to 1993. From 1993 to 1995, he was a Visiting Scholar with Cairo University, Cairo, Egypt. From 1995 to 1997, he was a Visiting Scholar with the Group of VLSI, Texas A&M University, College Station, TX, USA. From 1997 to 2003 and from 2003 to 2005, he was an Assistant Professor and an Associate Professor with the Department of Electrical Engineering, Minia University. From 2005 to 2007, he was a Visiting Researcher with Ohio University, Athens, OH, USA. He is currently a Professor and the Vice Dean of Post-Graduate Studies and Researchers with the Faculty of Engineering, Minia University. He has authored over 65 articles and one book chapter. His current research interests include analog and mixed-mode circuit design, low-voltage low-power analog circuits, current-mode circuits, nanoscale analog, and digital integrated circuits design field-programmable gate arrays.

WAEL ABOUELWAFA received the Ph.D. degree in bio-medical engineering and systems from Cairo University, Cairo, Egypt, in 2012. He is currently an Assistant Professor with the Biomedical Engineering Department, Minia University. His current research interests include bioinformatics, machine learning, big data analytics, and field-programmable gate array.