SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I8P105

Volume 12 Issue 8, 51-60, August 2025

Review Article

FPGA Based Acceleration of Biological Sequence Alignment Algorithms Based on a Dynamic Programming Approach: A Survey

Anita Wagh[1], Prachi Mukherji[2], Seema Rajput[3]

[1,2,3]*Department of Electronics & Telecommunication Engineering, MKSSS's Cummins College of Engineering for Women, Maharashtra, India.*

[1]*Corresponding Author : anita.wagh@cumminscollege.in*

Received: 03 June 2025 Revised: 04 July 2025 Accepted: 05 August 2025 Published: 30 August 2025

Abstract *- The rapid growth of biological sequence data necessitates efficient computational methods for sequence alignment, a fundamental task in bioinformatics. In this study, we provide a thorough overview of the literature on the topic of employing Dynamic Programming (DP) to speed up methods for biological sequence alignment using Field-Programmable Gate Arrays (FPGAs). Needleman-Wunsch and Smith-Waterman are two examples of DP algorithms that ensure optimum alignment; nonetheless, they are computationally demanding. FPGAs offer a promising platform for accelerating these algorithms by exploiting parallelism and hardware customization. This survey reviews existing research and methodologies employed to implement sequence alignment algorithms on FPGAs, comparing their performance, scalability, and energy efficiency against traditional CPU-based approaches. We also discuss the challenges and opportunities associated with FPGA-based acceleration, including data dependencies, reconfigurability, and optimization techniques. Hardware-software co-design and advancements in FPGA architectures can further enhance performance and scalability in biological sequence alignment tasks. Alignment accuracy may be improved by using Machine Learning (ML) models like Convolutional Neural Networks (CNNs) to extract features from raw sequence data. In order to facilitate future developments in this dynamic area, the study seeks to enlighten scholars and practitioners on the cutting-edge acceleration methods for biological sequence alignment that are based on FPGAs.*

Keywords *- Dynamic Programming, FPGA accelerator, Needleman-Wunsch Algorithm, Sequence alignment, Smith-Waterman Algorithm.*

1. Introduction

Sequence alignments serve as a robust method for assessing similarities between related DNA or protein sequences. These alignments serve to encapsulate diverse insights regarding the aligned sequences, such as shared evolutionary lineage or analogous structural roles. Aligning letters from two or more sequences implies the hypothesis of a shared ancestral origin.

DNA is like a twisted ladder made up of two long strings, and each string is composed of basic elements referred to as nucleotides. The four nucleotides are Adenine (A), Guanine (G), Thymine (T), and Cytosine (C). The two strings of DNA run in opposite directions, like a zipper. Each nucleotide in one string pairs up with a matching one in the other string. They stick together because of specific chemical attractions. A always pairs with T, and C with G. This pairing is important because the sequence of nucleotides in one string determines the sequence in the other. So usually, we only need to know the sequence of one string, and the other one can be figured out from it. The building blocks of proteins are amino acids, which chain together and interact to give proteins their unique properties. A single chain of amino acids is like a line, with each amino acid arranged in a specific order. There are twenty common amino acids, which we represent with letters like A, C, D, and so on. Each amino acid is defined by a triplet sequence of DNA building blocks known as nucleotides, termed a codon. For instance, the codon UGG means the amino acid Tryptophan, abbreviated as W. Some amino acids have just one codon, while others have up to six. RNA is a vital cellular molecule with many jobs, including building proteins and regulating which genes are turned on or off, as well as genetic expression. While RNA and DNA share a similar design, there are some important distinctions. Like DNA, RNA is made up of recurring building blocks, i.e. nucleotides. In addition to A, C, and G, RNA incorporates Uracil (U) instead of Thymine (T), which is found in DNA.

DNA sequences, and the proteins they make, change over time through mutation and natural selection. Mutations can

happen in different ways, but usually, they involve replacing one building block, i.e. a nucleotide, with another, or adding or removing one or more nearby building blocks. Similarly, in proteins, mutations often involve swapping one amino acid for another, or adding or removing one or more nearby amino acids. Unlike shuffling a deck of cards, there is no easy way to switch the order of building blocks in DNA or proteins. So, when we compare two DNA or protein sequences to see if they come from a common ancestor, the matching parts line up directly, without any mixing or swapping.

Basecalling is an essential computational stage in nanopore sequencing that converts complex electrical signals into a recognizable DNA base sequence or reads. Sequence alignment helps to identify similarities between sequences; therefore, it is possible to know the function of newly basecalled sequences by comparing them with sequences already present in the database. Additionally, it allows scientists to create phylogenetic trees, which are diagrams that depict the evolutionary relationships between the analysed sequences. Thus, it helps in determining evolutionary similarity, especially in closely related species. It becomes possible to predict the presence of more members within gene families due to sequence alignment. It allows researchers to predict the functions of the unknown sequences by identifying conserved sequence patterns and motifs. Thus, it helps identify regions within DNA/proteins that have structural and functional similarity.

Field-Programmable Gate Arrays (FPGAs) are versatile, programmable hardware devices made up of configurable logic blocks such as Look-Up Tables (LUTs), memory, and DSPs, connected via a programmable routing fabric. FPGAs are widely used for prototyping and accelerating applications due to their reconfigurability and parallel processing capabilities. This makes them suitable for tasks like DNA sequence alignment, which involve irregular parallelism. By mapping alignment algorithms onto FPGAs, researchers achieve faster and more efficient processing compared to traditional CPU-based systems. Substantial advancements have been made in utilizing FPGAs for accelerating DNA sequence alignment algorithms based on Dynamic Programming, but most of the existing FPGA designs are optimized for specific sequence lengths or fixed scoring schemes. There is a lack of adaptive, scalable architectures that can efficiently handle varying read lengths and alignment configurations, which are common in real-world genomic datasets. When dealing with real-world datasets, FPGA implementations often suffer from inefficient memory management. Again, it is observed that there is limited exploration of hardware-aware optimization techniques to reduce LUT, BRAM, and power usage while maintaining alignment accuracy. Many FPGA-based implementations oversimplify alignment by neglecting biologically relevant features such as affine gap penalties, ambiguous bases, or multiple sequence alignment. This limits their applicability in comprehensive genomic analysis. To overcome these gaps in research, it is expected from researchers to design more scalable, biologically accurate, and resource-optimized FPGA architectures.

The aim of this paper is to provide a detailed and systematic review of how FPGAs have been utilised to accelerate biological sequence alignment algorithms based on dynamic programming techniques. It seeks to analyse and compare various FPGA architectures and implementation strategies, discuss design challenges and trade-offs, and identify potential areas for future research and improvement in the field of hardware-accelerated bioinformatics. The paper's structure is as follows: Section 2 focuses on the biological and computational materials used for sequence alignment and compares and contrasts the different methods used for this process. Section 3 describes the scoring system used for the dynamic programming approach of sequence alignment. Section 4 reviews studies focused on FPGA acceleration to improve Dynamic Programming (DP) approaches. In Section 5, the difficulties in employing FPGA acceleration for sequence alignment are discussed, followed by a brief overview of the entire work in Section 6.

2. Materials and Methods

The key biological material required for DNA sequence alignment is DNA extracted from biological samples, which can be sourced from humans, animals, plants, bacteria, viruses, or any organism under study. Technical materials include DNA extraction kits to isolate DNA, PCR reagents to amplify DNA segments before sequencing, and sequencing technologies such as Illumina, PacBio, and Oxford Nanopore. Illumina is a widely used sequencing technology, known for producing short reads with high accuracy and throughput. It utilizes a Sequencing-By-Synthesis (SBS) method, where nucleotides are added one by one, and fluorescently labeled nucleotides are detected as the DNA strand is synthesized. However, Illumina's read lengths (100–300 base pairs) limit its ability to resolve complex genomic regions, such as large structural variants and repetitive sequences. Pacific Biosciences (PacBio) offers Single Molecule Real-Time (SMRT) sequencing, a long-read technology that generates sequences thousands of base pairs long, which is advantageous for resolving complex genomic regions. However, PacBio has a lower throughput compared to Illumina, making it more expensive for large-scale projects. Oxford Nanopore Technologies (ONT) uses nanopore sequencing, which is capable of real-time, ultra-long reads by passing DNA through tiny pores. ONT stands out for its ability to generate real-time data and its portability, with devices like the MinION. However, ONT has a higher raw error rate (~5-15%) compared to Illumina and PacBio, though improvements in software have significantly enhanced accuracy. Each technology has specific strengths and weaknesses, and the choice of sequencing platform depends on the goals of the research project. Once DNA is extracted

and sequenced, the raw sequence data (comprising nucleotides A, T, C, G) is aligned. Computational materials for DNA sequence alignment include input sequences (query sequences) and reference genomes. DNA sequences are stored in formats like FASTA (which contains only the sequences) and FASTQ (which includes sequence quality scores). Alignment tools like BLAST, Burrows-Wheeler Aligner (BWA), Bowtie, and tools for multiple sequence alignment, such as MAFFT, ClustalW, and MUSCLE, are commonly used for sequence comparison and analysis. Aligning large datasets often requires significant computational power. Some modern sequence aligners leverage GPUs for faster processing, FPGAs to accelerate alignment, and cloud platforms like Amazon Web Services or Google Cloud to manage large-scale projects. Databases such as GenBank (NCBI), ENSEMBL, RefSeq, European Nucleotide Archive (ENA), and DNA Data Bank of Japan (DDBJ) store vast collections of nucleotide sequences and are frequently used as reference points for DNA sequence alignment.

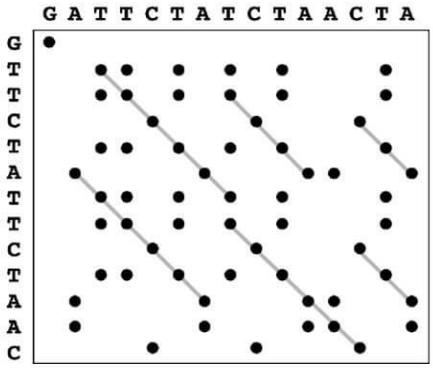

Fig. 1 Dot Plot Matrix[3]

Aligning sequences to find similarities can be done in different ways. Two common methods are Dynamic Programming (DP) and heuristics. The easiest way to check the similarity of two sequences is a dot plot [3]. This is a kind of grid (Matrix) where the sequences are placed along the sides. A dot is positioned at each matrix intersection where the respective elements from both sequences match (i.e., where rows and columns display the same letter). Diagonal lines of dots indicate areas of similarity between the two sequences, as illustrated in Figure 1. The time and space complexity of the Dot plot Matrix approach is $O(mn)$, where m and n are the lengths of the two sequences being compared. DP algorithms determine the best solution by considering all potential approaches to address a problem. This approach is used to determine the best solution for complex problems by dividing them into more manageable subproblems. This method is applicable to problems where these subproblems overlap. By merging the solutions of these smaller subproblems, we can derive the final solution. After solving a particular subproblem, its solution is stored in memory to avoid redundant computations of the same subproblem. DP algorithms discover the optimal solution by reviewing all potential methods to tackle a problem [13]. Some examples of DP algorithms used for sequence alignment include the Needleman-Wunsch (N-W) Algorithm, the Smith-Waterman (S-W) Algorithm, Hirschberg, and Miller-Myers.

Problem-solving through a heuristic approach relies on past experiences, observations, and insights. While this method offers a solution, it does not guarantee optimal results as it makes assumptions about where or how to find the best solution. However, this approach can expedite the original problem-solving process. Unlike DP algorithms that explore all potential solutions, which can be time-intensive, focusing on the most likely methods to solve a problem can significantly reduce computation time. Examples of a heuristic approach for sequence alignment are the Basic Local Alignment Tool (BLAST) [4], FASTA, etc.

Sequence alignment techniques are grouped into global and local categories based on the scope of the alignment (entire sequences vs. subsequences) and the specific objectives or criteria used to evaluate and optimize the alignment (maximizing overall similarity vs. identifying local similarities). One goal of global approaches is to align the reference and the search/query sequences, or as many characters as feasible, from the beginning to the end of both sequences. The Dot plot, the N-W algorithm, and the Hirschberg method are all examples of global alignment techniques. Dot plot uses a basic search algorithm, but Hirschberg and N-W use dynamic programming [3]. Aiming to identify brief portions of similarity between two sequences, specifically the query and database sequences, local approaches differ from global methods. S-W and Miller-Myers are two examples of DP approaches; FASTA and BLAST are two examples of heuristic-based approximation procedures.

Multiple Sequence Alignment (MSA) and Pairwise Sequence Alignment (PSA) are two other ways that sequence alignment techniques are categorized according to the number of sequences that need to be aligned, the particular goals that need to be achieved, and the applications that need to be implemented. PSA compares two biological sequences (like proteins or DNA) to find matching regions. These matches might mean the sequences are related in function, structure, or how they evolved.

The dot-plot method, DP methods like S-W and N-W, and heuristic methods like FASTA and BLAST are examples of PSA. MSA involves aligning three or more biological sequences of comparable length. By analyzing the results from MSA applications, one can infer homology and examine the evolutionary connections between the sequences. MSA can be conducted using either exhaustive or heuristic methods. Exhaustive alignment entails evaluating all potential alignments simultaneously.

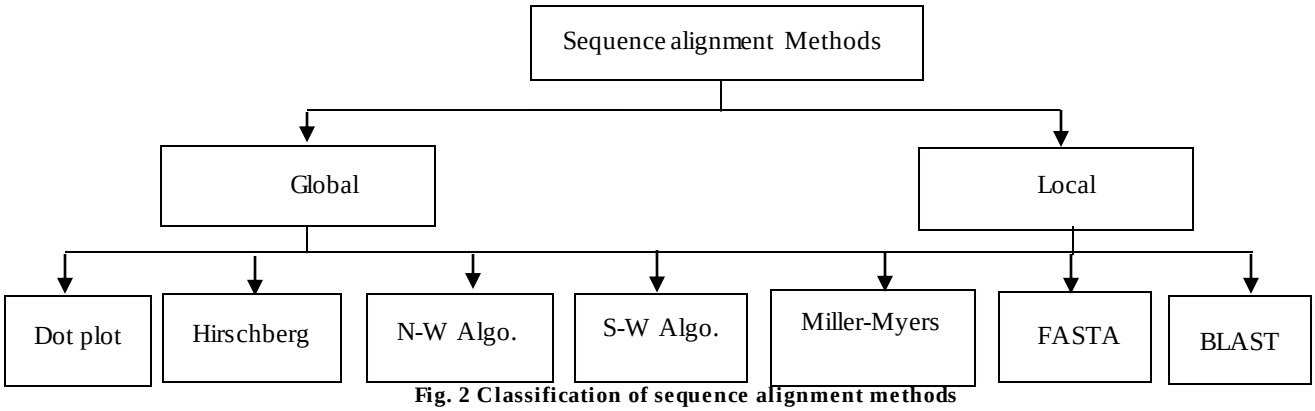

Fig. 2 Classification of sequence alignment methods

A multidimensional search matrix, resembling the two-dimensional matrix utilized in dynamic programming for pairwise alignment, is necessary for executing multiple sequence alignment with the exhaustive algorithm. The progressive technique constructs multiple alignments gradually, using pairwise similarity as a foundation. It is termed "progressive" because it aligns sequences in a sequential, step-by-step fashion. The classification is reflected in Figure 2.

3. Scoring System for DP Alignment Process

By assessing all potential alignments and selecting the DP matrix route that has the highest total alignment score, the DP alignment technique seeks to identify the ideal alignment. The scoring system is crucial for determining the optimal alignment by assigning scores to matches, mismatches, and gaps based on the similarity or dissimilarity between aligned residues or bases. The scoring matrix and scoring calculation methods guide the DP algorithm in efficiently exploring the solution space and identifying the best alignment between sequences.

	A	C	G	T
A	1	-1	-1	-1
C	-1	1	-1	-1
G	-1	-1	1	-1
T	-1	-1	-1	1

Fig. 3 Similarity matrix

3.1. Similarity Matrix

When two aligned sequences of nucleotides or amino acids are identical, a Match Score is given. Similar residues or bases are more likely to align when the match score is positive. A mismatch score is given if there is a difference between two matched sequences of nucleotides or amino acids. Mismatched residues or bases are penalized when their mismatch score is negative. A similarity matrix is utilized to capture all probable letter pairings along with their respective scores. The similarity matrix for the most basic system (for which the scores are match = {1}, mismatch = {-1}, indel = {-1}) is represented as shown in Figure 3. There are many substitution matrices, like Blocks Amino Acid Substitution Matrices (BLOSUM (50, 62)) or Point Accepted Mutation (PAM (80, 250)), available to compare sequences during alignment.

3.2. Gap Penalties

During sequence alignment, gaps, also known as indels (insertions or deletions), frequently occur. Occasionally, these gaps can be quite extensive. From a biological perspective, it is more plausible for a significant gap to result from a single large deletion rather than several individual deletions. Therefore, scoring two small indels should be penalized more harshly than scoring one large indel. A commonly employed method to address this is to assign a higher score for initiating a new gap (gap-opening penalty) and a lower score for each subsequent nucleotide or amino acid added to extend the existing gap (gap-extension penalty). In a system with a linear gap penalty, the points for creating and maintaining a gap are the same.

$$W_k = kW_1 \quad (1)$$

Here in Equation 1, W_k represents the cost of a single gap, with k denoting the gap length. An affine gap penalty distinguishes between gap opening and gap extension.

$$W_k = uk + v \quad (2)$$

Here in Equation 2, $v > 0$ serves as the penalty for gap opening, while $u > 0$ represents the penalty for gap extension. Compared to linear gap penalty, affine gap functions enhance the applicability of sequence alignment algorithms to biological sequences [23]. In the context of biological sequences, it's logical to assign a larger penalty for initiating

gaps while keeping a lesser penalty for extending them, i.e. $v > u$

3.3. Scoring Matrix

The score matrix H is ordered as $(n+1) \times (m+1)$, where m is the length of the reference sequence and n is the length of the query sequence. The elements of the scoring matrix, Hi, j (where i is the row index and j is the column index) of the S-W method, taking into account the linear gap penalty, are determined by Equation 3.

$$H_{i,j=} max \begin{cases} 0 \\ H_{i-1,j-1} + s(a_i, b_j) \\ H_{i-1,j} + W(1) \\ H_{i,j-1} + W(1) \end{cases} \quad (3)$$

Similarly, the elements of the scoring matrix of the N-W method, considering the linear gap penalty, are calculated using Equation 4.

$$H_{i,j=} max \begin{cases} H_{i-1,j-1} + s(a_i, b_j) \\ H_{i-1,j} + W(1) \\ H_{i,j-1} + W(1) \end{cases} \quad (4)$$

Where $s(a_i, b_j)$ is the similarity score of comparing a_i with b_j and $W(1)$ is the penalty for a mismatch.

The three steps of sequence alignment using the DP approach are matrix initialization, matrix filling(scoring) and traceback.

3.3.1. Matrix Initialization

The two-dimensional matrix, with dimensions $(m+1) \times (n+1)$, is initially populated with zeros. This is followed by the first row and column of the matrix being penalized for gaps. The top-left cell (0,0) will typically be initialized with a score of 0.

3.3.2. Fill the DP Matrix

Cell values are calculated using the formula given by equations (3) or (4) for S-W or N-W respectively. The process is iterated through the DP matrix cell by cell, calculating the alignment scores based on the recurrence relations using the scoring scheme. The completely filled DP matrix for alignment using the S-W algorithm is shown in Table 1.

3.3.3. Traceback

To find the optimal alignment path, begin with the cell or cells with the highest score in the DP matrix and use traceback to move across the matrix. Based on the traceback path, construct the aligned sequences by inserting gaps where necessary to align the sequences optimally. As S-W is a local alignment (interested in short patches of similarity), the traceback process starts with the cell having the highest score and stops when zero is encountered [1], whereas N-W is a global alignment method, so the traceback starts from the bottom rightmost cell and stops when zero is encountered [2].

Table 1. DP Matrix for S-W Algorithm

	-	A	G	C	G	A
-	0	0	0	0	0	0
A	0	1	**0**	0	0	1
C	0	0	0	**1**	0	0
G	0	0	1	0	**2**	0
A	0	1	0	0	0	**3**
A	0	1	0	0	0	1

Table 1 shows the results of calculating the dynamic programming matrix H and the tracing back route, which is shown in bold. The scoring scheme defined here is match = 1, mismatch -1 and gap = -1. Table 2 lists the methods for sequence alignment along with their respective time and space complexity. Assuming a Reference sequence length of m and a Query sequence length of n, the temporal complexity of each of these methods is $O(mn)$.

4. FPGA Acceleration of DNA Sequence Alignment Methods

The DP method calculates items inside the scoring matrix for about 98.6% of its execution time when executed on a Central Processing Unit (CPU) [3]. So, to get better results than what can be achieved with software on a regular CPU. It is necessary to speed things up using dedicated hardware. Separate from the CPU, a hardware accelerator is a piece of hardware that is purpose-built to increase this acceleration.

Deploying it on multiple platforms, including Graphical Processing Units (GPUs), CPUs, and FPGAs, has been the focus of efforts to speed up the method or its computationally heavy components. FPGAs are reconfigurable computing devices where algorithms are mapped directly onto fundamental processing logic components, such as NAND gates. While CPUs and GPUs are versatile and commonly utilized for a range of computing functions.

Table 2. Comparisons of sequence alignment algorithms

Method	Type	Search method	Time complexity	Space complexity
Dot matrix	global	Basic	*O(mn)*	*O(mn)*
Needleman Wunsch	global	DP	*O(mn)*	*O(mn)*
Hirschberg	global	DP	*O(mn)*	*O(m+n)*
Smith Waterman	local	DP	*O(mn)*	*O(mn)*
Miller-Myers	local	DP	*O(mn)*	*O(m+n)*
Fasta	Local	Heuristic	*O(mn)*	*O(mn)*
Blast	Local	Heuristic	*O(mn)*	$O(20^w+mn)$.

FPGAs provide a distinct blend of features, including reconfigurability, parallelism, low latency, high throughput, energy efficiency, scalability, and real-time processing, making them exceptionally suitable for enhancing sequence alignment algorithms' speed. By harnessing FPGAs' inherent parallelism and customization capabilities, substantial performance enhancements and computational efficiency can be achieved compared to conventional CPU and GPU-centric methods for sequence alignment [5]. To reduce the *O(mn)* complexity usually associated with the matrix filling step, it is helpful to compute several elements of the *H* matrix simultaneously. Nevertheless, this method becomes more complicated due to data dependencies. Each *Hi, j* value relies on the values of three adjacent entries *Hi*−1,−1, *Hi*−1,*j* , and *Hi,j*−1. In addition, there are three nearby values that are dependent on each of these neighboring entries. This pattern of reliance is essentially applicable to all other entries in the area. Due to their placement outside of each other's data dependence zones, all items inside each anti-diagonal may be computed concurrently. Figure 4 displays an example of an *H* matrix for two sequences, where the cells along the anti-diagonal bands (highlighted in the same color) represent elements that can be calculated concurrently. The greatest number of elements that can be calculated simultaneously is determined by the length of the longest anti-diagonal. The bold diagonal arrow signifies the direction of computation advancement. With a maximum of 5 cells that may be calculated simultaneously, this calculation requires at least 9 cycles due to the 9 anti-diagonals [6].

	-	A	G	C	G	A
-	0	0	0	0	0	0
A	0	1	**0**	0	0	1
C	0	0	0	**1**	0	0
G	0	0	1	0	**2**	0
A	0	1	0	0	0	**3**
A	0	1	0	0	0	1

Fig. 4 Sample H matrix where the elements in the antidiagonal band are computed in parallel

The authors of [7] looked at how using bespoke instructions on an FPGA board may improve the computational processing time of the SW method. For the sake of comparison, the S-W algorithm is first implemented in the C language. Based on the current letters being compared, the scores and gaps from nearby cells, and the assessment model in the C code, each cell in the S-W matrix is given a score. The most computationally intensive part of the program, the evaluation module, is replaced by an FPGA Custom Instructions (CI) created in Verilog by the FPGA accelerator. The Nios II microprocessor, known as Altera, is used to instantiate these CI on the FPGA. After that, the hardware-accelerated version's runtime was evaluated and compared to that of the pure software version to find out how much faster the processing was. The results showed that the average processing time was cut by 287% using the hardware-accelerated approach. Thus, it seems that using FPGA-specific instructions might be a great way to further genomic sequence searching research.

An approach to fast sequence alignment using Run Time Reconfiguration (RTR) is presented in [8]. RTR is the capacity to change the hardware configuration of an FPGA while the system is running. Unlike traditional hardware design, where the logic and connections are fixed after the design is synthesized and implemented, RTR allows for dynamic changes to the FPGA's configuration. In an RTR-enabled FPGA system, the design is partitioned into multiple configurations or "tiles." Each tile represents a specific function or operation within the application or algorithm. During system operation, the FPGA can switch between these configurations based on the requirements of the task at hand. According to [9], using run-time reconfiguration significantly enhanced the performance of the S-W algorithm. The overall time it took to run the method went down from 6,461 seconds to just over 100 seconds, and the time allotted for calculating the *Hi,j* matrix elements was down by about a third. This signifies a speed enhancement of around 64 times compared to the implementation relying solely on software.

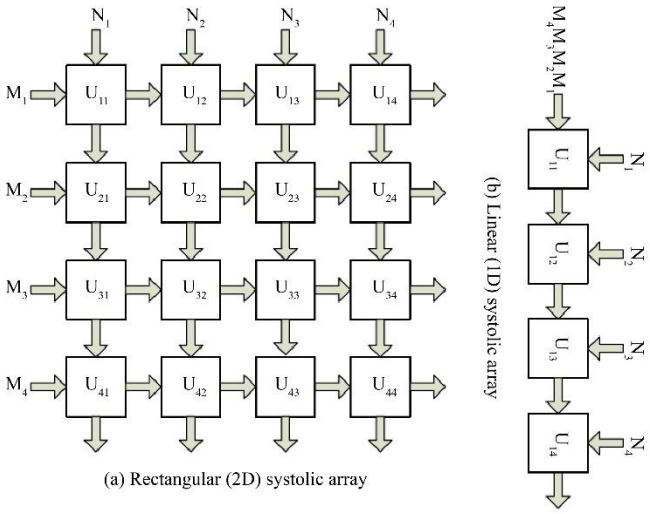

Fig. 5 Systolic array [6]

FPGAs use custom hardware building blocks named Processing Elements (PEs). Each PE can compare a pair of sequence elements in a single clock cycle. By connecting these PEs in a special grid or line (systolic arrays), FPGAs can perform many comparisons in parallel, making sequence alignment much faster. In each stage, data is received by one or more nearby elements (e.g., North and West), processed, and then sent in the opposite direction (e.g., South and East) by each Processing Element (PE). Matrix multiplication and

other systolic array operations work by taking one matrix at a time and sending it down the array row by row. Another matrix is simultaneously added to the array, starting from the left side and working its way to the right, column by column. Each processor processes a whole row and column before moving on to the next set of fake data. After this is done, the array is used to hold the result of the multiplication, which allows it to process the array row by row or column by column, depending on the user's preference. Figure 5 shows the fundamental configuration of a systolic array. M and N are the two input vector arrays here. In most cases, a preset procedure is used to extract the value Uij from inside the processing cells. Giga Cell Updates Per Second (GCUPS) is a common metric measuring DP algorithm performance. Here, a "cell" typically refers to an element in a DP matrix used to score the alignment. Each cell update involves calculating a new value based on the alignment score of neighboring cells, considering possible matches, mismatches, and gaps. In sequence alignment using a systolic array, GCUPS measures the system's performance and efficiency. This proves that the systolic array can update the dynamic programming matrix with one billion cells per second.

The formula to get GCUPS is $GCUPS = (n \times m /t) \times 10^9$, where n and m are the sequence lengths and t is the calculation time. When working with huge biological datasets, this parameter becomes even more important for determining the system's computing capability. High-throughput processing is vital in these cases. The systolic array might be fed several data sets all at once, according to the authors of [10]. A high-speed linear systolic array was used to implement the S-W algorithm. The goal was to find a way around the limitations caused by the I/O bus. The scoring matrix, which is the edit-distance matrix, was divided into four-element clusters so that each systolic cell could analyze two data points at the same time. As a result, the data is sent over the bus with two nucleotides packed together, which makes the processing speed twice as fast.

An improved linear systolic array is used to construct the S-W method in [11]. The method of generating PE included merging two processing units into a compact cell. To determine the edit distance between the reference and test sequences, the compact cell that is so generated is used. This cell utilizes 3 Xilinx Vertex slices. It enables loading both sequences into the system, avoiding the need for the runtime configuration. Successful verification of the implementation was achieved with the assistance of the Pilchard platform, which offers a memory-mapped bus that is 64 bits in width and operates at 133 MHz.

A new approach to meeting the need for high-throughput processing with minimal FPGA board resources is detailed in [12]. Alignment Pointer Generation Unit (APGU), Accurate Memory Address Generator (AMAG), Weight Matrix (WM), and Alignment Matrix (AM) are the components of the hardware circuitry. In order to make the method more compact, we modify the NW technique and use a dynamic WM. After the WM is built, the AMAG will allocate the direction pointers to the AM at a certain location. Ultimately, alignment is performed utilizing the values stored in the AM cells. Simulation outcomes have demonstrated that this method utilizes FPGA resources quite conservatively.

Parallel implementations on GPU and FPGA platforms compared to a sequential CPU-based version of the NW algorithm in terms of execution times, in [14]. Focusing on tree-based transformation on CUDA-enabled GPUs, the study aimed to parallelize the N-W technique. A tree-based approach is used to sort out dependencies and get everything in sync. This method eliminates interdependence by coordinating various tasks. If the computer needs to do the same thing multiple times, instead of doing it over and over again, it does it once and then uses that result for all the similar tasks at the same time. This way, tasks with different needs work together better because they communicate more efficiently. The implementation of this method emerges as highly efficient, employing block synchronization with a lock-free approach, enabling the aggregation of numerous threads compared to conventional sequential algorithms. Despite this, data movement presented challenges for GPUs, prompting the adoption of FPGA-based implementation. Data movement presents no concerns in FPGA setups, as all components, including RAM and Flash cells, are linked via horizontal and vertical channels on the same board, unlike in CPUs. Both methodologies demonstrated notable advancements over the sequential CPU-based implementation. Specifically, the FPGA implementation showcased consistent execution times for small sequences, whereas for larger sequences, there was a considerable improvement.

The S-W algorithm, outlined in [15], is renowned for its computational demands in large database sequencing tasks. To address this, the algorithm is accelerated by implementing it on an FPGA board using 2D systolic arrays. However, due to the limited hardware resources on the FPGA, only a constrained number of processing elements (PE) can be deployed. Consequently, the similarity/score matrix is partitioned into sub-matrices for computation. The PE array calculates a single sub-matrix during each cycle and keeps the intermediate results in memory for the next cycle. With a top performance of 78 GCUPS, this method considerably improves performance, reaching a speedup of up to 625xwhen compared to a software-only solution.

In [16], the banded Smith-Waterman method is implemented on an FPGA using a two-stage pipeline topology. This structure enables scoring and backtracking to be performed simultaneously. A lookahead calculation technique is employed for the scoring matrix, resulting in a reduction in LUT consumption and improved throughput. Instead of using larger bit-widths to represent increasing

scores, the scoring matrix is replaced by a direction matrix, which uses only two bits to represent match, mismatch, insertion, and deletion, effectively compressing data and conserving resources. An error-counting mechanism, executed concurrently with scoring, filters out reads with excessive errors, thereby skipping the backtracking step and saving computational time. The backtracking module is fully implemented in hardware to address potential bandwidth issues encountered when sending the scoring matrix to a CPU for backtracking. Reading the direction matrix buffer is initiated by this module upon score completion. When the clock cycles, the backtracking module receives 2-bit direction information at each position, starting with the last entry and working its way to the first. The alignment route is then updated with these bits, and the final alignment information is generated by combining the alignment path with the read and reference sequences. However, a limitation of this technique is that it can only backtrack along a single alignment path.

The Adaptively Banded Smith-Waterman method (ABSW), developed by Y. Liao et al. [19], is designed to align long genomic sequences and is compatible with many types of hardware. Given the computational intensity of the Smith-Waterman algorithm, computing entire score matrices during read-to-reference genome alignment is unfeasible. Hence, the search space is minimized using seed-and-extend paradigms. In this initial phase, the algorithm identifies small, highly similar subsequences (called seeds) between the read and the reference genome. Seeds are short, exact or nearly exact matches that serve as anchor points for further alignment. Once the seeds are identified, the algorithm extends the alignment outwards from these anchor points. In order to determine the best local alignment, this extension step calculates the alignment score in the areas around each seed. To make sure the end result is correct and takes gaps and mismatches into consideration, the S-W algorithm is used to improve the alignment. However, because this calculation is limited to the seeds' immediate vicinity, it is more efficient. To address the significant time difference between the seeding and extension phases, an FPGA Accelerator is created. The Banded Smith-Waterman method with constant memory is used to align fixed-length subsequences during the extension phase, when the best alignment pathways of the subsequences are concentrated in diagonal bands. Heuristic techniques and dynamic overlapping are proposed to further improve accuracy by band overlapping in subsequences.

The author showed a digital version of the S-W and N-W algorithms that was optimized in terms of both software and hardware in [21]. For hardware implementation, a combinational circuit is built using Look-Up Tables (LUT) and then implemented on an FPGA platform. A Convolutional Neural Network (CNN) model that has been specifically tailored is used for the software implementation. The use of ML in a sequence alignment method has never been done before. With a minimum value of four, N denotes the length of each sequence, and this approach runs in O(N/4) computation steps. The N-W method achieves a 98.3% accuracy in its implementation. The author presented a new application of classical ML for global sequence Alignment using the N-W algorithm in [20]. Building a lookup table or dataset is important to this solution. The category of the alignment array is the goal, and the two DNA sequences expressed in binary or decimal form are the dataset's input (features or characteristics). With just a handful of possible output classes or alignment patterns, the N-W method is implemented using ML techniques. On two real-life, 4.1-million-nucleotide-long DNA sequences, an astonishing 99.7 percent accuracy was achieved by combining a multilayer perceptron with the ADAM optimizer, leading to up to 2912 CGUPS. However, the drawback of these two approaches is that the length of the two sequences to be compared must be the same and a multiple of four.

5. Challenges in FPGA-Accelerated Biological Sequence Alignment

The survey highlights the challenges encountered by researchers when utilizing FPGA accelerators for biological sequence alignment. The challenges are as follows: 1) Fitting resource-hungry dynamic programming algorithms onto FPGAs requires a careful balancing act due to limited logic, memory, and interconnect availability; 2) Limited resources of FPGAs create a bottleneck when dealing with massive datasets like whole genomes. 3) The inherently sequential nature of DP algorithms clashes with the parallel processing power of FPGAs, creating a challenge in optimizing resource usage and maximizing throughput. 4) Although FPGAs excel at parallel processing, fully utilizing this strength for sequence alignment algorithms requires careful optimization. The key challenge lies in partitioning the DP algorithm and mapping it onto the FPGA architecture in a way that minimizes communication between processing units and maximizes parallel execution. 5) Effectively transferring data between the FPGA and external memory, including RAM or storage devices, is essential for achieving optimal performance. The challenges lie in reducing data movement and enhancing data transfer speeds, especially when dealing with extensive datasets.

6. Conclusion

Within this paper, a classification scheme for diverse sequence alignment algorithms documented in existing literature has been outlined. DP algorithms offer accurate but computationally intensive solutions suitable for scenarios where precision is crucial, whereas heuristic approaches provide faster, approximate solutions that are more scalable for large datasets but may sacrifice some level of accuracy. Through this paper, the key challenges faced by researchers up to the present have been outlined. Therefore, the survey has shown that FPGA is the most appropriate hardware to accelerate the DP algorithms. Flexible and adaptable designs,

hardware-software co-design, and advanced FPGA architectures can be used further to enhance performance and scalability in biological sequence alignment tasks.

There is a need for future work to implement AI algorithms for the optimization of hardware and software. ML models like CNNs and RNNs can be used to identify important features from raw sequence data that help to improve alignment accuracy. Embedding techniques like Word2Vec, Doc2Vec, and others can be adapted to generate dense vector representations of biological sequences, which capture semantic similarities and can be used to improve alignment algorithms using ML.

References

[1] T.F. Smith, and M.S. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195-197, 1981. [CrossRef] [Google Scholar] [Publisher Link]

[2] Saul B. Needleman, and Christian D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443-453, 1970. [CrossRef] [Google Scholar] [Publisher Link]

[3] Robert Giegerich, "A Systematic Approach to Dynamic Programming in Bioinformatics," *Bioinformatics*, vol. 16, no. 8, pp. 665-677, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[4] Stephen F. Altschul et al., "Basic Local Alignment Search Tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403-410, 1990. [CrossRef] [Google Scholar] [Publisher Link]

[5] Licheng Guo et al., "Hardware Acceleration of Long Read Pairwise Overlapping in Genome Sequencing: A Race between FPGA and GPU," *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, San Diego, CA, USA, pp. 127-135, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Laiq Hasan, and Zaid Al-Ars, "An Overview of Hardware-Based Acceleration of Biological Sequence Alignment," *Computational Biology and Applied Bioinformatics*, pp. 187-202, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[7] Jason Chiang et al., "Hardware Accelerator for Genomic Sequence Alignment," *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, New York, NY, USA, pp. 5787-5789, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yoshiki Yamaguchi et al., "High Speed Homology Search using Run-Time Reconfiguration," *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, pp. 281-291, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[9] S. Margerms, "Reconfigurable Computing in Real-World Applications," *FPGA Structures and ASIC Journal*, vol. 10, no. 5, pp. 1-8, 2006. [Google Scholar]

[10] S. Bojanic et al., "High Speed Circuits for Genetics Applications," *2004 24th International Conference on Microelectronics*, Nis, Serbia, vol. 2, pp. 517-524, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[11] C.W. Yu et al., "A Smith-Waterman Systolic Cell," *Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL)*, Lisbon, Portugal, pp. 375-384, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[12] Ardhendu Sarkar, and Som Banerjee, "FPGA Implementation of DNA Sequence Alignment with Traceback," *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, pp. 47-52, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Yu-Cheng Li, and Yi-Chang Lu, "BLASTP-ACC: Parallel Architecture and Hardware Accelerator Design for BLAST-based Protein Sequence Alignment," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1771-1782, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[14] Chirag Kyal, Rishav Kumar, and Adil Zamal, "Performance-based Analogising of Needleman-Wunsch Algorithm to Align DNA Sequences using GPU and FPGA," *2020 IEEE 17th India Council International Conference (INDICON)*, New Delhi, India, pp. 1-5, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Anna Hakim et al., "Performance Analysis of DNA Sequencing Using Smith-Waterman Algorithm on FPGA," *Journal of VLSI Design Tools & Technology*, vol. 9, no. 2, pp. 9-15, 2019. [Google Scholar] [Publisher Link]

[16] Luyi Li, Jun Lin, and Zhongfeng Wang, "PipeBSW: A Two-Stage Pipeline Structure for Banded Smith-Waterman Algorithm on FPGA," *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Tampa, FL, USA, pp. 182-187, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Muhammad Irfan, Kizheppatt Vipin, and Rizwan Qureshi, "Accelerating DNA Sequence Analysis using Content-Addressable Memory in FPGAs," *2023 IEEE 8th International Conference on Smart Cloud (SmartCloud)*, Tokyo, Japan, pp. 69-72, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Behnam Khaleghi et al., "SALIENT: Ultra-Fast FPGA-based Short Read Alignment," *2022 International Conference on Field-Programmable Technology (ICFPT)*, Hong Kong, pp. 1-10, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Yi-Lun Liao et al., "Adaptively Banded Smith-Waterman Algorithm for Long Reads and its Hardware Accelerator," *2018 IEEE 29th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, Milan, Italy, pp. 1-9, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[20] Amr Ezz El-Din Rashed et al., "Sequence Alignment using Machine Learning-based Needleman–Wunsch Algorithm," *IEEE Access*, vol. 9, pp. 109522-109535, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Amr Ezz El-Din Rashed, Marwa Obaya, and Hossam El~Din Moustafa, "Accelerating DNA Pairwise Sequence Alignment using FPGA and a Customized Convolutional Neural Network," *Computers & Electrical Engineering*, vol. 92, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Osamu Gotoh, "An Improved Algorithm for Matching Biological Sequences," *Journal of Molecular Biology*, vol. 162, no. 3, pp. 705-708, 1982. [CrossRef] [Google Scholar] [Publisher Link]

[23] Stephen F. Altschul, and Bruce W. Erickson, "Optimal Sequence Alignment using Affine Gap Costs," *Bulletin of Mathematical Biology*, vol. 48, pp. 603-616, 1986. [CrossRef] [Google Scholar] [Publisher Link]

[24] T.K. Attwood, and D.J. Parry-Smith, *Introduction to Bioinformatics*, Pearson PLC, 2003. [Google Scholar] [Publisher Link]

[25] Peng Chen et al., "Accelerating the Next Generation Long Read Mapping with the FPGA-based System," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 5, pp. 840-852, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[26] Jeff Allred et al., "Smith-Waterman Implementation on a FSB-FPGA Module Using the Intel Accelerator Abstraction Layer," *2009 IEEE International Symposium on Parallel & Distributed Processing*, Rome, Italy, pp. 1-4, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[27] Enzo Rucci et al., "Accelerating Smith-Waterman Alignment of Long DNA Sequences with OpenCL on FPGA," *Proceedings of the 5th International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO)*, Granada, Spain, pp. 500-511, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[28] Hasitha Muthumala Waidyasooriya, Masanori Hariyama, and Michitaka Kameyama, "FPGA-Accelerator for DNA Sequence Alignment based on an Efficient Data-Dependent Memory Access Scheme," *Highly-Efficient Accelerators and Reconfigurable Technologies*, pp. 127-130, 2014. [Google Scholar] [Publisher Link]

[29] Mostafa Morshedi, and Hamid Noori, "FPGA Implementation of a Short Read Mapping Accelerator," *Proceedings of the 13th International Symposium on Applied Reconfigurable Computing (ARC)*, Delft, The Netherlands, pp. 289-296, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[30] Simone Casale-Brunet, Endri Bezati, and Marco Mattavelli, "Design Space Exploration of Dataflow-Based Smith-Waterman FPGA Implementations," *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, Lorient, France, pp. 1-6, 2017. [CrossRef] [Google Scholar] [Publisher Link]