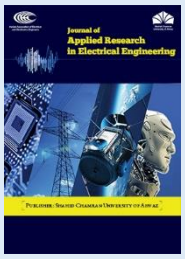
Shahid Chamran University of Ahvaz

Iranian Association of Electrical and Electronics Engineers

Journal of Applied Research in Electrical Engineering

E-ISSN: 2783-2864
P-ISSN: 2717-414X
Homepage: https://jaree.scu.ac.ir/



Research Article

A High-Performance MEMRISTOR-Based Smith-Waterman DNA Sequence Alignment Using FPNI Structure

Mahdi Taheri [1], Hamed Zandevakili [2], and Ali Mahani [2,*]

[1] *Department of Information and Communication Technology, Tallinn University of Technology, Tallinn 19086, Estonia*

[2] *Reliable and Smart Systems Lab (RSS), Shahid Bahonar University of Kerman, Kerman 7616913439, Iran*

* *Corresponding Author: amahani@uk.ac.ir*

Abstract: It is crucial to detect potential overlaps between any pair of the input reads and a reference genome in genome sequencing, but it takes an excessive amount of time, especially for ultra-long reads. Even though lots of acceleration designs are proposed for different sequencing methods, several crucial drawbacks impact these methods. One of these difficulties stems from the difference in read lengths that may take place as input data. In this work, we propose a new Race-logic implementation of the seed extension kernel of the BWA-MEM alignment algorithm. The first proposed method does not need reconfiguration to execute the seed extension kernel for different read lengths. We use MEMRISTORs instead of the conventional, complementary metal-oxide-semiconductor (CMOS), which leads to lower area overhead and power consumption. Also, we benefit from Field-Programmable Nanowire Interconnect Architecture as our matrix output resulting in a flexible output that bypasses the reconfiguration procedure of the system for reads with different lengths. Considering the power, area, and delay efficiency, we gain better results than other state-of-the-art implementations. Consequently, we gain up to 22x speedup compared to the state-of-the-art systolic arrays, 600x speed up considering different seed lengths of the previous state-of-the-art proposed methods, at least 10x improvements in area overhead, and 105x improvements in power.

Keywords: Bioinformatics, BWA-MEM, memristor, race logic.

Article history
Received 31 December 2020; Revised 27 April 2021; Accepted 27 May 2021; Published online 28 June 2021.

How to cite this article
M. Taheri, H. Zandevakili, and A. Mahani, "A high-performance MEMRISTOR-based smith-waterman DNA sequence alignment using FPNI structure," *J. Appl. Res. Electr. Eng.*, vol. 1, no. 1, pp. 59-68, 2022.
DOI: 10.22055/jaree.2021.36117.1016

Check for updates

1. INTRODUCTION

DNA sequencing is a laboratory technique used to determine the exact sequence of bases (A, C, G, and T) in a DNA molecule. The DNA base sequence carries the information a cell needs to assemble protein and RNA molecules. DNA sequence information is essential to scientists investigating the functions of genes [1]. Based on the recent research on genomic sequence alignment, various algorithms and specific designs improve the sequencing aligners' performance and energy consumption. We can put genomics in the category of big data science, and by growing technology, it is getting much bigger. The volume of produced data by genomics can be compared with three main big data generators [2]:

I. astronomy: Over these decades, Astronomy is placed in the group of Big Data challenges.

II. YouTube: There is a huge number of sharing stuff and videos that are created and shared on YouTube

III. Twitter: Makes a lot of opportunities for new insights by mining more than 400 million messages that are sent every day.

Table 1 [3] compares these four groups of data generators, showing how genomics is increasingly overcoming in the case of demanding data acquisition, storage, distribution, and analysis.

The first step for most genomics applications is sequence alignment. Many reads of DNA strands have to be aligned against the reference genome, and the best alignment for each read is produced as output. There are a variety of sequence alignment tools such as:

1. Bowtie [4]

Table 1: Comparison of four groups of Big Data in 2025 are shown in this Table [3].

Data Phase	Astronomy	Twitter	YouTube	Genomics
Acquisition	25 zetta-bytes/year	0.5-15 billion tweets/year	500-900 million hours/year	1 zetta-bases/year
Storage	1 EB/year	1-17 PB/year	1-2 EB/year	2-40 EB/year
Analysis	In situ data reduction	Topic and sentiment mining	Limited requirement	Heterogeneous data and analysis
	Real-time processing	Metadata analysis		Variant calling, ~2 trillion central processing unit (CPU) hours
	Massive volumes			All-pairs genome alignments, ~10,000 trillion CPU hours
Distribution	Dedicated lines from antennae to server (600 TB/s)	Small units of distribution	Major component of modern user's bandwidth (10 MB/s)	Many small (10 MB/s) and fewer massive (10 TB/s) data movement

2. BWA [5]
3. MAQ [6]
4. SOAP [7]
5. BWA-MEM [8]

Consider the state that we want to find all local alignment using a dynamic programming approach as an example of the alignment algorithms. If we choose the Smith and Waterman algorithm [9], which uses O (nm) time for aligning a read of length n against a reference of length m, it can be concluded that the approach is too slow.

For example, as the fastest sequencing, NGS takes about hours with a lot of memory usage to sequence an entire human DNA. Based on the [10] experimental results, aligning 1000 characters as a read against the human genome will take more than 15 hours.

In the case of actual application, we work with genes or chromosomes that are about a few thousand to a few hundred million lengths. If we align the whole human genome with the SW method, it will last for about some days to weeks.

There are other algorithms like BLAST [11], which are heuristic methods. They are used to find local alignments very efficiently. Using BLAST takes 10-20 seconds to align a read of 1000 bp against the human genome [11].

Obviously, with these time-consuming calculations, general-purpose processors are not a good solution for executing these bioinformatics workloads. Thus, we need more parallel and specific hardware such as GPU or FPGA dedicated to massively accelerating the intensive computations and leading to large speedups.

In this work, we accelerate the Smith-waterman-like algorithm with a race logic strategy based on memristor elements to speed up the execution time. The rest of this paper is organized as follows: We provide related works in section 2. Our design contributions and details of our MEMRISTOR-based design are discussed in Section 3. Section 4 evaluates the results, and finally, Section 5 concludes this article.

2. Related Works

We are experiencing exponential growth of experimental data and information in biology called data explosion [12]. One of the most valuable operations in Bioinformatics is DNA sequencing. Four nucleotides (A, C, G, T) make the foundation of the DNA sequences. Swapping these nucleotides causes alternate biochemical functions and products within the DNA. One of the most Severe computational parts of Bioinformatics is finding similarities between two DNA sequences called pairwise alignment. Different methods accomplish this for Biologists, which leads to different time consumption. The Smith-Waterman (SW) is one of the most accurate algorithms with high sensitivity degree but high computational time and high hardware resource usage. Consider that the complexity of SW is of quadratic order. The BLAST [13] and FASTA [14] are derivative methods of SW which do not lead to optimal solutions because of sensitivity loss but are significantly faster. Another dynamic programming method for comparing two macromolecules is the Needleman-Wunsch algorithm (NW) [15], which calculates the alignment score between two sequences based on the Levenshtein distance. There are different other efforts to reduce the computational time of different parts of the pairwise alignment algorithms. A custom ASIC implementation of a BioSCAN is introduced in [16], in which heuristic and very high-density implementation caused the high performance. A new method of information representation was proposed in [17] that performs computation by setting up logical race conditions in a circuit on ASIC platform and they achieved about 3x higher throughput at 5x lower power density. The authors in [18] evaluate SWIFOLD: A SW parallel implementation for long DNA sequences implemented on Intel core with OpenCL. They claim that their method increases better performance with higher resource consumption. In another work, in [19], a ReRAM-based process-in-memory architecture is designed to improve short read alignment throughput per Watt by 13×. Several techniques have been proposed to accelerate the SW inexact alignment algorithm. However, the seed extension

step of this algorithm makes it inherently a slow design. The authors provided a new 2-D technique regarding SW inexact alignment algorithm in which they have used fixed numbers for Match, Mismatch, and gap penalty [20]. The authors in [21] propose a new hardware accelerator in which the most incorrect candidate locations fill out with 130-fold speedup than software. There is a faster implementation of SW in [22] which achieves 2to8× performance improvement compared to other SIMD-based SW implementations. Also, intrinsic delay of the circuits edit-distance computation elements as in [23] was utilized to propose the ASAP accelerator based on the RACE-logic hardware acceleration presented in [17] for accelerating SW and NW algorithms on an ASIC platform. Their work leads to 200× speedup than an equivalent SW-C implementation. Some other works accelerated BWA-MEM genomic mapping algorithm on different platforms such as GPU and FPGA. BWA-MEM is a widely used algorithm to map genomic sequences onto a reference genome. This algorithm is composed of three main computational kernels [8]:

I. SMEM Generation: This kernel is used to find seeds (sub-strings of the reads) that are likely mapping the read against the reference genome. There is a chance of generating several seeds with the variable length for each read [5]. This step is an exact-match-finding phase that uses the Burrows-Wheeler transform. For this work, seeds are at least nineteen characters and a maximum of 131.

II. Seed Extension: This step is an inexact-matching step that executes chaining and extending seeds in two directions using an SW-like algorithm [9]. This part of the BWA-MEM algorithm finds the optimal local alignment by using a scoring system.

III. Output Generation: In this step, the best alignment (i.e., with the highest score) is finalized and provided as the output in SAM format, if necessary.

Note that the seed extension kernel used in BWA-MEM is different from the SW algorithm in two substantial ways (Table 2) [24]: (1) Non-zero initial values: The initial values in the first column and the first row depending on the alignment score of the seed found by the SMEM Generation kernel. (2) Additional output generation: Other than the local and global alignment scores, the exact location inside the similarity matrix and a maximum offset (indicating the distance from the diagonal at which a maximum score has been found) are also generated.

The first accelerated implementation of BWA-MEM is presented in [24] with evaluating several FPGA-based systolic array architectures. Their implementation is 3× faster than the software-only execution. Another hardware acceleration of the BWA-MEM genomics short read mapping for longer read length is stated in [25]. The authors discussed accelerating the seed extension kernel of the BWA-MEM algorithm on a GPU accelerator and achieved up to 1.6× improvement compared to application-level execution time [26]. Power efficiency analysis of accelerated BWA-MEM implementations on heterogeneous computing platform against the software-only baseline system is studied in [27] by offloading the seed extension phase on an accelerator.

Table 2: Profiling the BWA-MEM algorithm [24].

Kernel	Execution time (%)	Bound
SMEM generation	56	Memory
Seed extension	32	Computational
Output generation	9	Memory
Other	3	I/O

A high-performance FPGA-based Seed Extension IP core is designed [28] for BWA-MEM DNA alignment that achieves 350× speedup than an Intel Core i5 general-purpose processor. Authors gain up to 14.5× speedup than the SW algorithm by :(a) Applying heuristics; (b) Processing MEMs, and (c) Extracting MEMs by using a bit-level parallel method [29]. It is considered that after all these works, the problem of memory accessory, area overhead, time, and power consumption of the alignment algorithms methods and implementations are still extremely problematic. Thus, we aimed these problems in our work, and by our suggested methods, improved all of the problems mentioned above.

3. PROPOSED DESIGN

This section describes the proposed method for filling the similarity matrix of the SW-based algorithm and shows how it can speed up time and reduce power consumption compared to state-of-the-art architectures. Besides, our method uses an unfixed length strategy that can leads to higher speedup due to it does not need to be reconfigured for different reads lengths.

There is a new data representation that is used for a broad class of optimization problems which is called "race logic." This method can be used for the kind of problems that use dynamic programming algorithms to be solved. There are different implementations of race logic, such as synchronous and asynchronous, which we focus on synchronous type for our design. Race logic idea is based on the race conditions in a circuit to optimize computation in case of time.

We designed an SW similarity matrix with the idea of the race logic design. Also, we use MEMRISTOR instead of conventional, complementary metal-oxide-semiconductor (CMOS), which leads to better performance. In addition, we considered Field-Programmable Nanowire Interconnect [30] Architecture as our matrix output. Significantly, we achieve lower power consumption and area overhead due to using a MEMRISTOR structure compared to the previous CMOS, ASIC, and FPGA structures mentioned in the results. Moreover, we gain lower delay as a result of

I. Using MEMRISTOR structure that is using race logic strategy which leads to lower circuit delay.

II. Utilization of FPNI as a flexible output that bypasses the system's reconfiguration procedure for reads with different lengths.

3.1. Algorithm Description

First, we describe the main idea of our design and show how it can lead to a proper answer to the SW-like matrix with performance improvement. As we know, the SW algorithm is a dynamic programming algorithm that can compute the alignment score (Levenshtein distance) of two reads and

partial-reference genome string with the Q, R length, respectively. For calculating the scoring alignment of these two strings, the algorithm constructs a matrix S that is a lattice of size IQ × IR (IQ, IR are the length of two strings). With the recursive equation, it can calculate the minimum edit distance between two strings. Notice that in the BWA-MEM algorithm, which is in our consideration for implementing our proposed design, the length of two strings is as same as each other, and we have a Square matrix in each solution. But its dimension may be different based on the length of the reads. We solve this problem by using FPNI as a flexible output of the circuit which helps us earn all the outputs of different matrix dimensions without any problem to change the circuit of any reconfigurations.

$$DP_{(i,i)} = MIN \begin{Bmatrix} DP_{(i-1,i-1)} + T_{(Match,Miss-match)} \\ DP_{(i-1,i)} + T_{(Gap)} \\ DP_{(i,i-1)} + T_{(Gap)} \end{Bmatrix} \quad (1)$$

where DP denotes the similarity matrix, $T_{(Match,Miss-match)}$ is the assigned score for when a match or a mismatch occurs (usually 0 for a match and a 2 for a mismatch [23]), and $T_{(Gap)}$ is the gap penalty with the usual one value [23]. It is worth mentioning that Match is for a situation where two corresponding nucleotides are the same as each other, and Miss-match states that they are not the same. Notice that we can choose these parameters to optimize the accuracy of the alignment based on the structure of the sequences compared [31-33]. Besides, we use fixed penalties for the gap between nucleotides with the more commonly used value [33]. The above equation, which is representative of the SW similarity matrix local alignment, leads to finding the largest sub-string of R, which is mapped with string Q with the lowest Levenshtein distance (LD) (See [34,35] for more information). However, this method is accurate and yields optimal alignment with high computational complexity. To overcome this problem, we can replace the LD values in (1) with their equivalent propagation delays and use the delay-based approach for addition and minimization. Accordingly, these two operations (addition and minimization) are necessary for recursive (1).

We give some examples of how the addition and minimization operations can be modeled by the race logic strategy for more clearance (Fig. 1).

Suppose that we have two signals (M and N) set to logic value '1' (inject a high signal) at different times. This time delay is representing the different values of these two signals. For example, consider that the signal M is set to '1' with a specific time delay (time delay = D1) that means the value of M is "D1" and the second signal is set after D2 second-time delay (time delay = D2) that mean N value is "D2".

I. If we want to add these two values with each other, we can combine the circuit elements of M and N in series. That means the total propagation delay of the output results from adding "D1" with "D2".

II. If we connect these two circuit elements to an OR gate, the signal that arrives first to OR gate emerges out of that. This structure is a Viewer of the minimization operator. Because both signals have the '1' value and the signal which have less amount of delay, will arrive first to OR gate and make the output of this gate '1' earlier.

III. For calculating the output value, we can place a counter at the end of our race logic design that serves as a decoder [23].

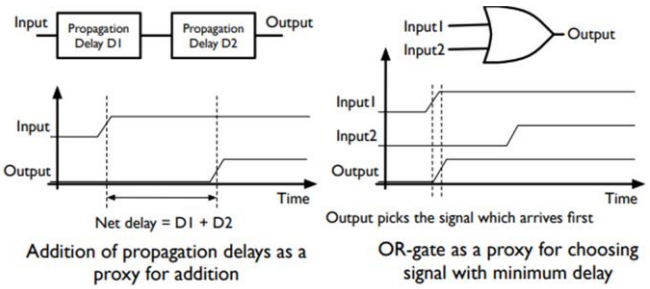


Fig. 1: Computing with propagation delays: Delay-based proxy for the addition operator is a series connection, and the proxy for the min operator is the OR gate [23].

We can apply these delay-based computations to SW similarity matrix of LD calculation. So, the delay between the rising edge of the input signal in the lattice and its emergence at any element on the last row is the minimum score of the local alignment.

3.2. Proposed Architecture

Fig. 2 demonstrates our accelerated architecture. It includes some basic cells to easily implement the desired functionality and a routing network to access some predefined basic cells' output. More details about the different parts of our proposed architecture will be presented in the following:

3.2.1. MEMRISTOR-element

Memristors [36] are new two-terminal logical and scientific basis and fourth classical circuit elements like resistors, inductors, and capacitors.

Memristors are changeable resistors that can be used for memory. In this case, the resistance will be stored as data. We can also use Memristive devices [37] in other applications such as logic and analogue circuits.

We can refer to some points of using memristors instead of CMOS circuits in our race logic:

I. With these devices, we can read and write data faster than CMOS circuits [38].

II. They are typically small devices. Hence, the CMOS circuits are usually bigger than the memristive-based circuits.

III. Nonvolatility is the main feature of memristors and their compatibility with standard CMOS technology [39]. They are either ideal for FPGA-like applications.

From above, we can conclude that memristive devices provide nonvolatile, dense, fast, and power efficiency to solving many major problems of the semiconductor devices.

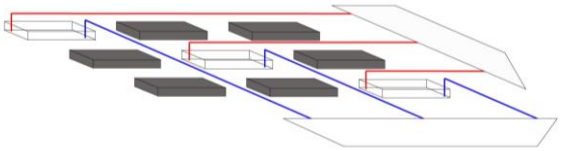


Fig. 2: Accelerated architecture.

Consider that we make a programmable design in which the user can set the corresponding delay of "match", "mismatch", and "gap" penalties. For example, when we know that the most nucleotide comparisons are Match, we can encode in how "match" delay has' 0' time delay, which ensures that large portions of our SW matrix are taken zero time to be explored. Different values for penalties help us to optimize the search time.

3.2.2. *Basic cells*

The schematic of our proposed cell is shown in Fig. 3. Accordingly, it includes three delay elements (DM, DI, DD) responsible for the mathematical operations of (1), respectively; a comparator/selector unit to compare the value of two nucleotides that are the inputs of each matrix cell and decides if Match or mismatch occurs, one local OR gate to implement the Min operation in (1), and one global OR gate to give us the flexibility of choosing output from different stages of the SW matrix.

3.2.3. *The comparator/selector unit*

This section includes several CMOS XNOR gates, and a memristor-based NAND gate to compare the "Ref" and Read" data. Also, the multiplexer controlled by the comparator stage's output defines the corresponding Match or mismatch penalty as its output. When the output value of the comparator becomes "0", this means the "Ref" data is equal to the "Read" data, and the proportional delay value for Match (which can be defined by the user in our design) goes out as output of the selector unit. The structure of our proposed comparator/selector unit is shown in Fig. 4.

3.2.4. *The delay element (DE)*

Delay elements are composed of:

I. Three input wavefront, which is the representation of the input signals and are the results of the preceding DEs in grid

II. Two corresponding nucleotides as input signals which have to be compared by the element

III. Three input signals representing the (Match, Mismatch, Gap penalty) values

IV. One output signal (global OR gate) which represent the output of the (1) ($DP_{(i,j)}$)

V. One output signal (local OR gate) which is designed to perform our desired flexible matrix output and used for local alignment.

The propagated output wavefront of each DE is a delay signal considering the corresponding match, mismatch, and gap delay penalties. When the other DE's outputs or signal

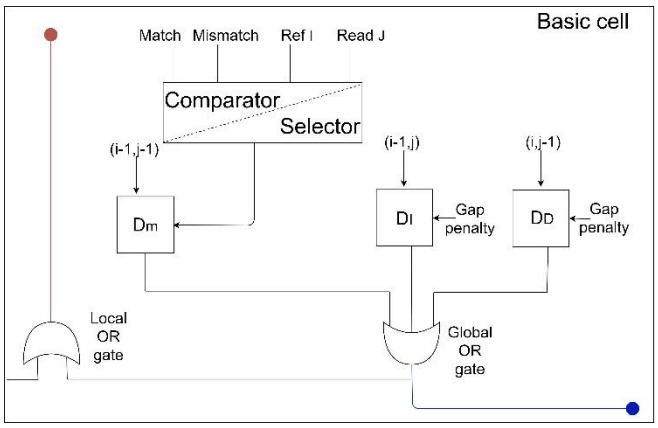


Fig. 3: Basic cell of our proposed design.

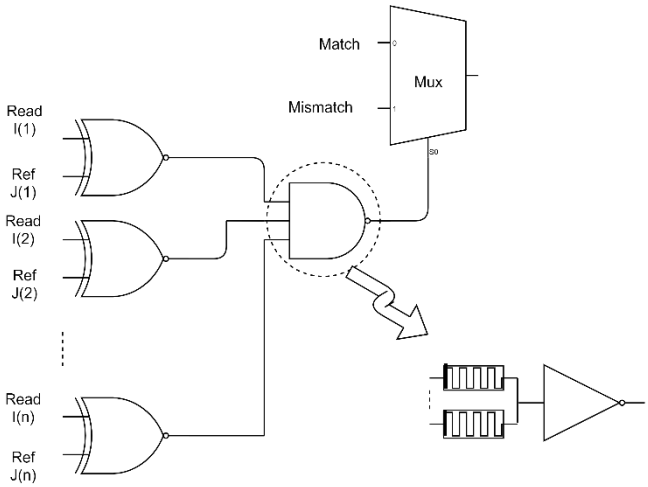


Fig. 4: Comparator/selector unit.

wavefront reaches an element, a delay is created based on the gap penalty specified for match/mismatch and gap penalty by propagating the signals through the memristors. The other advantage of our design is that it allows the user to program (i.e., dynamically set at runtime) the value of the Match, mismatch, and gap penalty based on the different applications and give the flexibility to use our approach in cases that merely require re-parameterization of the gap-penalties. The structure of our proposed delay element is shown in Fig. 5. It includes some delay elements to build different delays and a multiplexer to select the desired delay. As shown in Fig. 5, we have used memristors to implement the delay elements to reduce the area overhead.

3.2.5. *Local OR gate*

The local OR gate is used to make it possible to avoid unnecessary latency that is due to the variable input length. OR gate is a proxy for minimization operator, which emerges out the signal that arrives first at the gate. As shown in Fig. 6, to reduce the area overhead, we have used a memristor-based OR gate for this sake.

3.2.6. *Global OR gate*

The global OR gate is used to implement the minimization operation in (1). The structure of our proposed global OR gate is shown in Fig. 6. We have used a memristor-based OR gate for this sake to reduce the area overhead.

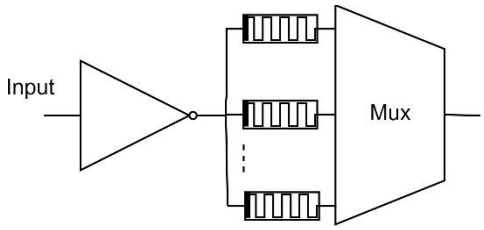


Fig. 5: Delay element unit that includes some delay elements to build different delays and a multiplexer to select the desired delay.

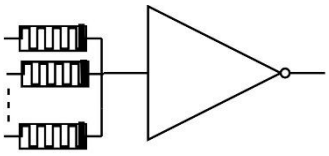


Fig. 6: Structure of memristor base OR gate in our design.

3.2.7. The routing network

Needleman and Wunsch [15] and Smith and Waterman [9] algorithms are well-known dynamic programming algorithms that lead to optimum global and local alignment of a read against the reference genome. A similarity matrix is filled in these approaches that have to find the local and global alignment score of reads against the corresponding reference sub-string [8]. Consider the practical scenario that read data has at most 150 base pairs (bp) for our comparison. Then we construct our similarity matrix with 131×131 dimension based on the BWA-MEM approach. We desire that the processing time of filling the similarity matrix kernel be independent of the read length but because of the fixed similarity matrix dimension, for shorter reads, we incur unnecessary latency.

To avoid this unnecessary latency, we have to contemplate a method that can be flexible with different read lengths and get output ready from the desired dimensions of the similarity matrix. Therefore, we can omit the unnecessary latency, which is the reason for not traveling through the entire elements irrespective of their length.

The original race logic design was demonstrated in simulation as an ASIC [14]. Even though this method has advantages in power consumption and substantial improvement in throughput in comparison of the state-of-the-art systolic implementations, but it suffers from the following problems:

I. The original race logic design uses conventional, complementary metal-oxide-semiconductor (CMOS) with size, power consumption, read and write time problems, and our approach.

II. Traveling through the entire elements irrespective of their length with the fixed similarity matrix dimension design that incurs unnecessary latency for shorter read size.

Our proposed accelerator is runtime-programmable for changing the input data size, which defines the size of the accelerator lattice. For this sake, we have used a nanowire-based routing network which is inspired by the FPNI technique [30]. Field-programmable nanowire interconnect (FPNI) is a new hybrid structure with advantages that are mentioned below:

I. high flexibility

II. low fabrication cost

By this technique, we can change the size of the accelerator lattice during the runtime according to the input data size. As shown in Fig.4, our proposed routing network includes some nanowires to access the output of some predefined basic cells and a selection unit controlled by the input data size to select the desired output. Each nanowire is connected through a "via" to the output of the local OR gate in the desired basic cell.

4. RESULTS

In this section, the simulation results of the proposed method will be compared with some well-known approaches. Performance of the mentioned methods is evaluated using several criteria such as area, delay, and power consumption. In Fig. 7, the numerical results of the proposed structure for delay parameter are compared with state-of-the-art systolic arrays and race logic design. In general, these are two of the best implementations of dynamic programming methods that achieve accuracy and speedup. Therefore, we compare our design to show the consummate performance of our work.

More details about each of the evaluation criteria will be presented in the following.

4.1. Area

To compute the occupied area of the mentioned methods, we have used the transistor counting technique in 65nm technology. According to the presented results in Table 3, the

Table 3: Occupied area of three methods in nm based on the transistor counting technique in 65nm technology

Read Length	Proposed	Systolic	Race logic
1	8.51E+02	7.34E+04	9.18E+03
2	3.40E+03	1.18E+05	2.09E+04
4	1.36E+04	2.34E+05	7.31E+04

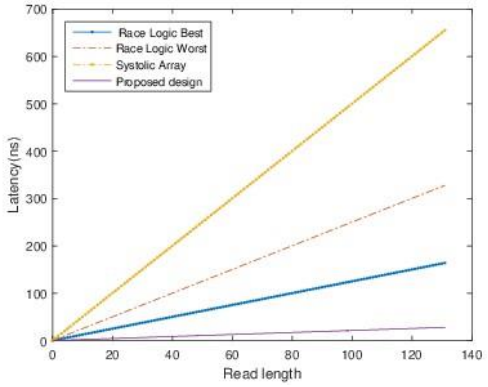


Fig. 7: Latency of the proposed method compared to the state-of-the-art systolic array and race logic designs.

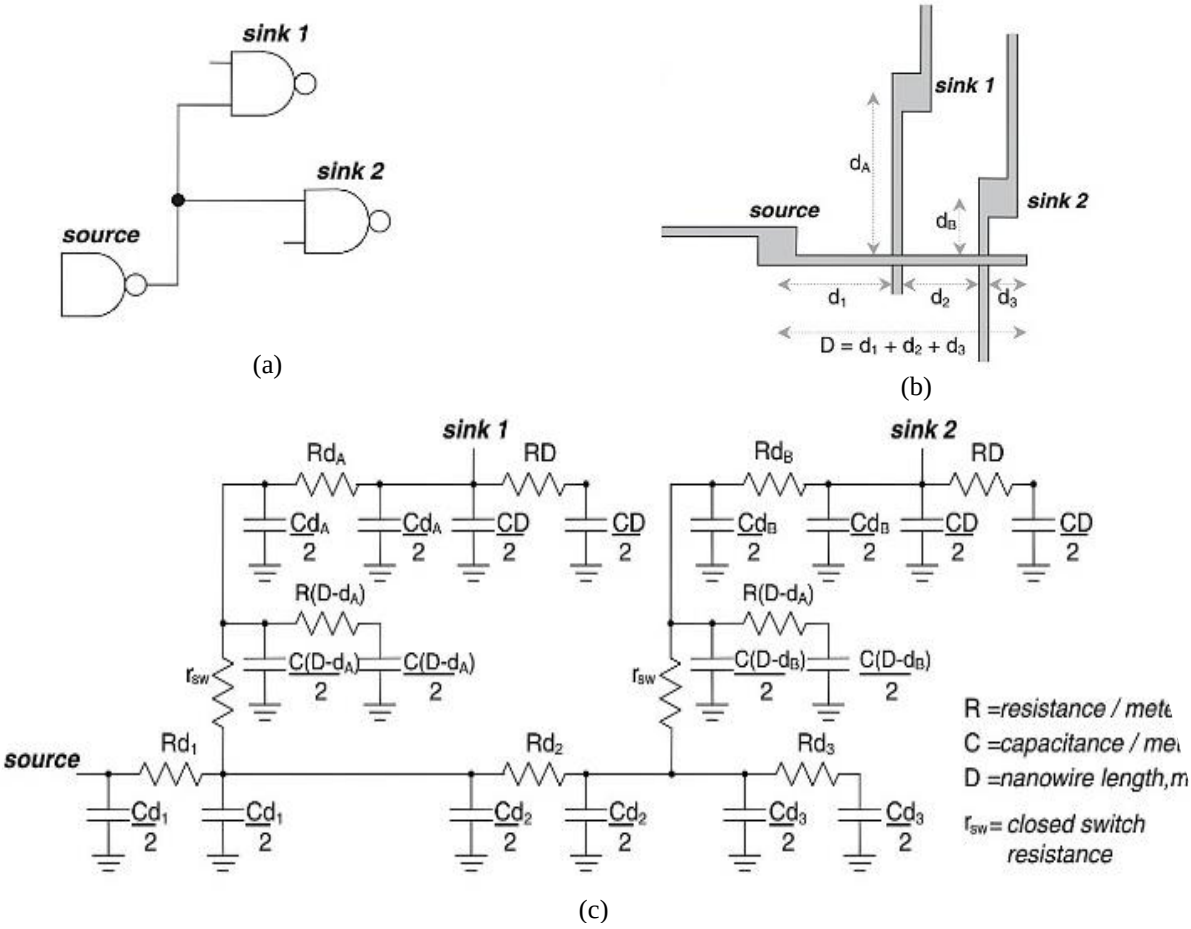


Fig. 8: (a) A signal with a fan-out of 2 (b) the implemented form by the nanowires (c) the electrical model [40].

4.2. Power Consumption

occupied area of the proposed method is compared with two other methods, and the results show that we achieve up to 10fold area improvement.

4.3. Delay

We need an electrical model of the nanowires, junctions, and CMOS components to calculate the delay of the proposed structure. For this sake, we have used the electrical model proposed in [40] for the FPNI structure. The electrical model for a simple circuit is shown in Fig. 8. Some of the model parameters such as closed junction resistance, the capacitance and resistance per unit length and geometry of the wires are also listed in Table 4 [40]. In this paper, we have used the HSpice tool to calculate the delay of the proposed structure. The presented results in Fig. 9 show how our design flaunts himself in case of fixed length matrix dimension implementation.

Power consumption of the proposed structure is evaluated using the formula presented in [40]:

$$Dynamic - power = \frac{1}{2} ANCV_{dd}^{2} f \qquad (2)$$

where A is the average 'activity' of a signal, N is the number of allocated nanowires, C is the capacitance of a single nanowire, V_{dd} is the supply voltage used by the CMOS, and f is the maximum clock frequency determined by timing analysis. To calculate the power consumption of the proposed

Table 4: Experimental parameters for FPNI architecture [40]

Parameter	Description	FPNI 30 nm
Pnano	Nanowire pitch	30 nm
Wnano	Nanowire width	15 nm
Wpin	Pin diameter	90 nm
Wpinvar	Pin size variation	20 nm
Walign	Alignment error	40 nm
Wsep	Pin/wire separation	15 nm
Rclosed	Closed junction resistance	24 K
p	On/off resistance ratio	>200
	Nanowire resistivity	8u cm
	Nanowire length	7115 nm
	Nanowire resistance	2.53 K

structure, we have used the HSpice tool. According to the presented results in Fig. 10, we compare our design with systolic arrays and the race logic approach. Results show those designs are power-hungry compared to our memristor-base design.

5. Conclusion

We present a new memristor-based SW matrix implementation that achieves more than six times speedup compared to the state-of-the-art race logic approach and 22

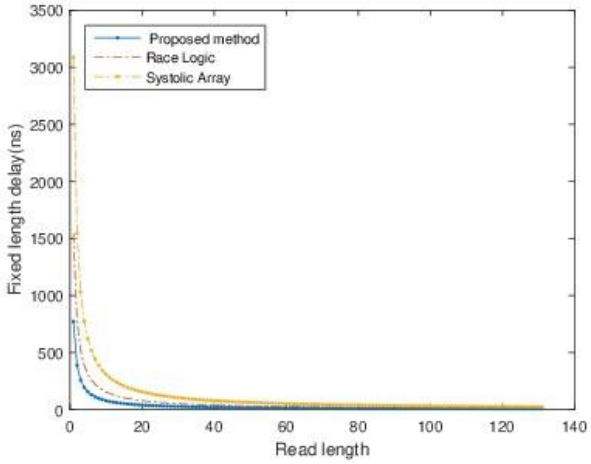


Fig. 9: Delay ratio of the proposed method, Systolic array, and race logic considering the fixed 131×131 SW matrix dimension in different read lengths.

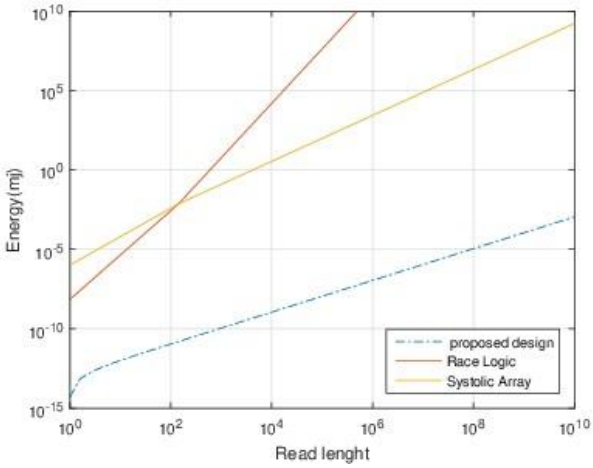


Fig. 10: Power consumption of our proposed design in comparison of Systolic arrays and race logic design.

times speedup than the systolic arrays implementation. We show how our design gives this flexibility to get the matrix output depending on the different input dimensions without unnecessary latency. Our implementation achieves up to 600x speedup with considering the fixed 131×131 SW matrix dimension by testing different read lengths. We also achieved at least 10x improvements in area overhead and also 10^5x improvements in power. Furthermore, our approach can be more practical and optimum in presenting programmable penalty matches, which gives the initiative to change them based on the biological application.

CRediT Authorship Contribution Statement

Mahdi Taheri: Conceptualization, Formal analysis, Methodology, Resources, Roles/Writing - original draft, Writing - review & editing. **Hamed Zandevakili:** Formal analysis, Software, Writing - review & editing. **Ali Mahani:** Project administration, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, redundancy, have been completely observed by the authors.

References

[1] M. Taheri, and A. Mahani, "Development and hardware acceleration of a novel 2-D BWA-MEM DNA sequencing alignment algorithm", in *The First Conference on Applied Research in Electrical Engineering*, Iran, 2021.

[2] J. Giles, "Computational social science: Making the links", *Nature*, vol. 488, no. 7412, pp. 448-450, 2012.

[3] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big data: astronomical or genomical?", *PLoS Biology*, vol. 13, no. 7, 2015.

[4] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. "Ultrafast and memory-efficient alignment of short dna sequences to the human genome", *Genome Biology*, vol. 10, no. 3, pp. 1-10, 2009.

[5] H. Li, and R. Durbin, "Fast and accurate short read alignment with burrows– wheeler transform", *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[6] H. Li, J. Ruan, and R. Durbin, "Mapping short dna sequencing reads and calling variants using mapping quality scores", *Genome Research*, vol. 18, no. 11, pp. 1851– 1858, 2008.

[7] R. Li, C. Yu, Y. Li, T. Lam, S. Yiu, K. Kristiansen, and J. Wang. "Soap2: an improved ultrafast tool for short read alignment", *Bioinformatics*, vol. 25, no. 15, pp. 1966–1967, 2009.

[8] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with bwa-mem", arXiv preprint arXiv:1303.3997, 2013.

[9] T. .F Smith, M. S. Waterman, "Identification of common molecular subsequences", *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[10] T. W Lam, W. Sung, S. Tam, C. Wong, and S. Yiu, "Compressed indexing and local alignment of dna", *Bioinformatics*, vol. 24, no. 6, pp. 791– 797, 2008.

[11] W. J. Kent, "Blat—the blast-like alignment tool", *Genome Research*, vol. 12, no. 4, pp. 656– 664, 2002.

[12] V. Marx, "Biology: The big challenges of big data", vol. 498, no. 7453, pp. 255-260, 2013.

[13] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs", *Nucleic Acids research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[14] W. R. Pearson, and D. J. Lipman, "Improved tools for biological sequence comparison", *Proceedings of the National Academy of Sciences*, vol. 85, no. 8, pp. 2444–2448, 1988.

[15] W. Wilbur, and D. J. Lipman, "The context dependent comparison of biological sequences", *SIAM Journal on Applied Mathematics*, vol. 44, no. 3, pp. 557–567, 1984.

[16] R. K. Singh, DL Hoffman, S. G. Tell, and C. Thomas White, "Bioscan: a network sharable computational resource for searching biosequence databases", Bioinformatics, vol. 12, no. 3, pp.191–196, 1996.

[17] A. Madhavan, T. Sherwood, and D. Strukov, "Race logic: A hardware acceleration for dynamic programming algorithms", in *2014 ACM/IEEE 41st International Symposium on Computer Architecture*, 2014, pp. 517–528.

[18] E. Rucci, C. Garcia, G. Botella, A. De Giusti, M. Naiouf, and M. Prieto-Matias, "Swifold: Smith-waterman implementation on fpga with opencl for long dna sequences", BMC systems biology, vol. 12, no. 5, pp. 96, 2018.

[19] F. Zokaee, H. R Zarandi, and L. Jiang, "Aligner: A process-in-memory architecture for short read alignment in rerams," *IEEE Computer Architecture Letters*, vol. 17, no. 2, pp. 237–240, 2018.

[20] M. Taheri, M. S. Ansari, S. Magierowski, and A. Mahani, "Hardware acceleration of the novel two dimensional burrows-wheeler aligner algorithm with maximal exact matches seed extension kernel", *IET Circuits, Devices and Systems*, vol. 15, no. 2, pp. 94-103, 2021.

[21] M. Alser, H. Hassan, H. Xin, O. Ergin, O. Mutlu, and C. Alkan, "Gatekeeper: a new hardware architecture for accelerating pre-alignment in dna short read mapping", *Bioinformatics*, vol. 33, no. 21, pp. 3355–3363, 2017.

[22] M. Farrar, "Striped smith–waterman speeds database searches six times over other simd implementations," *Bioinformatics*, vol. 23, no. 2, pp. 156–161, 2006.

[23] S. S. Banerjee, M. El-Hadedy, J. Bin Lim, Z. T. Kalbarczyk, D. Chen, S. S Lumetta, and R K Iyer, "Asap: accelerated short-read alignment on programmable hardware", *IEEE Transactions on Computers*, vol. 68, no. 3, pp. 331–346, 2018.

[24] E. J. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "An fpga-based systolic array to accelerate the bwa-mem genomic mapping algorithm", in *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2015, pp. 221–227.

[25] E. J. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "Hardware acceleration of bwa-mem genomic short read mapping for longer read lengths", *Computational biology and chemistry*, vol. 75, pp. 54–64, 2018.

[26] E. J. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "Gpu accelerated bwa-mem genomic mapping algorithm using adaptive load balancing", in *International Conference on Architecture of Computing Systems*, 2016, pp. 130– 142.

[27] E. J. Houtgast, V. Sima, G. Marchiori, K. Bertels, and Z. Al-Ars, "Power-efficiency analysis of accelerated bwa-mem implementations on heterogeneous computing platforms", in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2016, pp. 1–8.

[28] C. Pham-Quoc, B. Kieu-Do, and T. Ngoc Thinh, "A high-performance fpga-based bwa-mem dna sequence alignment", *Concurrency and Computation: Practice and Experience*, vol. 33, no. 2, e5328.

[29] A. Bayat, B. Gaëta, A. Ignjatovic, and S. Parameswaran, "Pairwise alignment of nucleotide sequences using maximal exact matches", *BMC bioinformatics*, vol. 20, no. 1, pp. 1-15, 2019.

[30] H. Zandevakili and A. Mahani, "A new asic structure with self-repair capability using field-programmable nanowire interconnect architecture", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 11, pp. 2268–2278, 2018.

[31] C. Wang, R. Yan, Xi. Wang, Jing-Na Si, and Ziding Zhang, "Comparison of linear gap penalties and profile-based variable gap penalties in profile–profile alignments", *Computational biology and chemistry*, vol. 35, no. 5, pp. 308–318, 2011.

[32] S. Henikoff and J. G Henikoff, "Amino acid substitution matrices from protein blocks", *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, pp. 10915– 10919, 1992.

[33] W. Sung, "*Algorithms in bioinformatics: A practical introduction*," CRC Press, 2009.

[34] G. Navarro, "A guided tour to approximate string matching", *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[35] V. I Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals", in *Soviet physics doklady*, vol. 10, no.8, pp. 707–710, 1966.

[36] L. Chua, "Memristor-the missing circuit element", *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[37] L. O. Chua, and S. Mo Kang, "Memristive devices and systems", *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976.

[38] A. C Torrezan, J. Paul Strachan, G. Medeiros-Ribeiro, and R S. Williams, "Sub-nanosecond switching of a tantalum oxide memristor", *Nanotechnology*, vol. 22, no. 48, pp. 1-7, 2011.

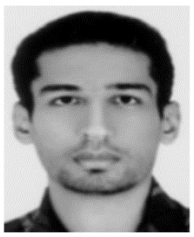
[39] J. Borghetti, Z. Li, J. Straznicky, X. Li, D. AA Ohlberg, W. Wu, D. R. Stewart, and R. S. Williams, "A hybrid nanomemristor/transistor logic circuit capable of self-programming", *Proceedings of the National Academy of Sciences*, vol. 106, no. 6, pp. 1699–1703, 2009.

[40] G. S. Snider, and R. S. Williams, "Nano/cmos architectures using a fieldprogrammable nanowire

interconnect", *Nanotechnology*, vol. 18, no. 3, 035204, 2007.

BIOGRAPHY

Mahdi Taheri received the B. Sc. degree in electronic engineering from Khaje Nasir University of Technology (KNTU), Tehran, Iran, in 2017 and The M.Sc. degree in Electronic Engineering from Shahid Bahonar University of Kerman, Kerman, Iran, in 2020. Since then, he has been with the RSS Lab at shahid bahonar university of Kerman for one year, and now, he is studying his Ph.D. at Tallinn University of Technology (TalTech). His research interests focus on Hardware assessment and reliability of neural networks, Fault-tolerant design, and FPGA-based accelerators.



Hamed Zandevakili received an M.S. degree in electronic engineering from Shahid Bahonar University, Kerman, Iran, in 2013. Since September 2014, he has been a Ph.D. student in the Department of Electrical Engineering at Shahid Bahonar University of Kerman, Iran. His research interests include reliable computing, VLSI testing, and reconfigurable computing.

Ali Mahani received the B. Sc. degree in electronic engineering from Shahid Bahonar University of Kerman, Iran, in 2001, The M.Sc. and Ph.D. degrees both in Electronic engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2003 and 2009 respectively. Since then he has been with the electrical engineering department of shahid bahonar university of kerman, where he is currently an associate professor. His research interests focus on Fault tolerant design, FPGA-based accelrators, approximate digital circuits, stochastic computing and Networked Systems.