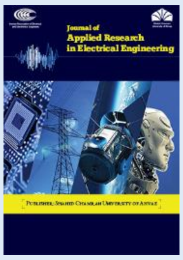
Shahid Chamran University of Ahvaz

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

Novel 2-D BWA-MEM FPGA Accelerator for Short-Read Mapping of the Whole Human Genome

Mahdi Taheri, and Ali Mahani*

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

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

Abstract: The mapping of DNA subsequences to a known reference genome, referred to as “short-read mapping”, is essential for next-generation sequencing. Hundreds of millions of short reads need to be aligned to a tremendously long reference sequence, making short-read mapping very time-consuming. Day by day progress in Next-Generation Sequencing (NGS) is enabling the generation of DNA sequence data at ever faster rates and lower cost, which means a dramatic increase in the amounts of data being sequenced. Nowadays, sequencing of nearly 20 billion reads (short DNA fragments) costs about 1000 dollars per human genome, and sequencers can generate 6 Terabases of data in less than two days. This article considers the seed extension kernel of the Burrows-Wheeler Alignment (BWA) genomic mapping algorithm for accelerating with FPGA devices. We propose an FPGA-based accelerated implementation for the seed extension kernel. The Smith-Waterman algorithm is used during the seed extension to find the optimum alignment between two sequences. The state-of-the-art architectures use 1D-systolic arrays to fill a similarity matrix. Based on the best score out of all match combinations, mismatches and gaps are computed. The cells on the same anti-diagonal are calculated in parallel in these architectures. We propose a novel 2-dimensional architecture. Our new modified algorithm is based on two editing and calculating phases. In each step of the calculation, some errors may occur in which all the cells on the same row and same column are computed in parallel, which significantly speeds up the process. Needless to say, these probable errors will be omitted before the next step of the calculation begins. Our simulation results show that the proposed architecture can work with up to 312 MHz frequency in Synopsys Design-Compiler for 180-nm CMOS technology and be up to 570x and 1.4x faster than the software execution and 1D-systolic arrays, respectively.

Keywords: Bioinformatics, FPGA, smith-waterman.

Article history
Received 20 December 2020; Revised 27 March 2021; Accepted 15 August 2021; Published online 11 April 2022.

How to cite this article
M. Taheri, and A. Mahani, "Novel 2-D BWA-MEM FPGA accelerator for short-read mapping of the whole human genome," *J. Appl. Res. Electr. Eng.*, vol. 1, no. 2, pp. 203-210, 2022. DOI: 10.22055/jaree.2021.36116.1015

Check for updates

1. Introduction

As a result of advances in next generation sequencing (NGS) techniques, the amount of genomic data is accumulating extremely rapidly [1]. Some projects will grow on an order of 1018 bytes per year by the next decade [2, 3]. Naturally, data of such scales impose a significant processing load in many aspects of bioinformatics. The new NGS techniques reduce the cost of generating a whole human genome, which, in return, further increases the amount of data [4]. Typically, NGS techniques produce millions of short reads, usually less than 200 bases in length [5]. In many cases, the short reads must be aligned to a reference sequence to obtain useful information about the sequenced molecule [6]. There are several different alignment algorithms to choose from, among which dynamic programming solutions, such as Bowtie [7], Burrows-Wheeler Alignment (BWA) [8], and SOAP2 [9], are shown to provide a better trade-off between the accuracy and speed. The most well-known, the BWA-MEM algorithm, which is popular for its high accuracy, is considered in this article as the target for FPGA acceleration.

The BWA-MEM algorithm, introduced in [10], manages superior alignment to a number of its contemporaries, but at the cost of a higher computational load. The three main operations in the BWA- MEM algorithm are (1) generating the super-maximal exact matches (SMEMs), (2) extending the seeds, and (3) generating the final outputs. The three kernels that perform these operations are listed in Table 1 along with their main bottlenecks [11]. As shown in Table 1, the step with the most severe computational bound is the seed extension kernel, which accounts for almost 33% of the execution time. A similarity matrix is dynamically filled during the seed extension operation. To compute the value of the cell (*i*, *j*) in the similarity matrix, the values of the following cells are required: (1) the west cell at (i, j-1), (2) the north cell at (i-1, j), and (3) the north-west cell at (i-1, j-1). State-of-the-art architectures aimed ataccelerating this problem use 1D-systolic arrays to fill the similarity matrix [8]. Specifically, for such computations, 1D-systolic arrays use a 1-dimensional array of *n* processing elements (PEs) where *n* is the number of cells on the main diagonal [11]. Note that increasing the number of PEs in such an architecture does not reduce execution time.

This article proposes a novel FPGA-based architecture to speed up the process of filling the similarity matrix, the bottleneck of the seed extension kernel. To the best of our knowledge, this is the first 2-dimensional architecture that, unlike 1D-systolic arrays, achieves higher throughput by increasing the number of PEs. The proposed architecture can compute the cells on the same row and on the same column in parallel as opposed to existing architectures that can only compute the cells on the same anti-diagonal in parallel. Novel PEs are proposed that compute cells in two phases: (1) the calculation phase that roughly approximates the cells and (2) the error compensation phase that fixes the potentially introduced errors during the first phase. The two phases are described in detail in Section 3.

The rest of the paper is organized as follows. Section 2 provides general information on the BWA-MEM algorithm and related work. The mathematical description of the proposed architecture, the PE designs, and the final hardware implementation are discussed in Section 3. Section 4 evaluates the proposed architecture. Finally, Section 5 concludes the article.

2. Background

This section describes the BWA-MEM algorithm and related previous works dedicated to its acceleration. DNA sequences are chains comprised of the four nucleotide monomers (A, C, G, T). Alternative permutations of these nucleotides typically encode alternate biochemical functions and products within the DNA. A 2-bit representation is used in our implementation for each of these four nucleotides. Although even simple organisms possess contiguous DNA exceeding a million nucleotides in length, DNA measurements (i.e. the products of sequencing machines) rarely produce sequences this long due to a combination of technical limits at present. In fact, it is common for leading sequencers to produce outputs, ‘reads’, on an order of 100 nucleotides in length, obtained from unknown regions of the nativeDNA originally fed into the sequencer.

As a result, pairwise local alignment, searching for similarities between a new read (the query) and anticipated DNA pattern (reference genome), is one of the first steps in bioinformatics algorithms [12]. The main workload of the

Table 1: Profiling the BWA-MEM algorithm [11].

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

BWA-MEM algorithm is aligning millions of short DNA reads against a reference genome (usually the human genome) [13]. The authors introduced a new low-power and high-speed bioinformatic engine, a hardware-accelerated base caller, for mobile sequencing applications [14]. There are also several other works providing a generalized methodology and insights for efficient implementation of the DNA sequencing algorithms [15–19].

Several techniques have been proposed to accelerate the Smith-Waterman inexact alignment algorithm. However, the seed extension step of this algorithm makes it inherently a slow design. Authors have provided a new 2-D technique regarding the Smith-Waterman inexact alignment algorithm in which they have used fix numbers for the match, mismatch, and gap penalty [20]. An FPNI structure is proposed in [21] that uses race logic and CMOS-gate representation and leads to compromising results in the case of flexible input read length and omitting unnecessary latency. This is a new re-configuration sequencing method for difference of read lengths that may take place as input data in which crucial drawbacks impact DNA sequencing methods. A hardware acceleration of the BWA-MEM genomics short read mapping for longer read length is implemented in [22]. This design is based on an architecture previously proposed in [11] where an FPGA-based 1D-systolic array is used to accelerate the BWA-MEM genomics mapping algorithm. The main idea is to insert some exit points between the PEs of the 1D-systolic array to avoid unnecessary calculations for shorter reads. By doing so, shorter reads do not have to go through all of the PEs and can exit the array once they get to the first exit point.

3. Proposed Architecture

This section describes the methodology proposed for filling the similarity matrix and introduces the new architecture for implementing the seed extension kernel of the BWA-MEM algorithm.

Unlike the 1D-systolic array-based architectures, we propose a 2-dimensional architecture in which two different strings of PEs are assigned to all of the cells in the same rows and on the same columns. By doing so, the Smith-Waterman algorithm can be executed faster compared to the 1D-systolic array-based architectures. Although the proposed architecture is more resource-hungry than 1D-systolicarrays, it runs significantly faster. Note that the 1D-systolic arrays cannot operate faster even with more PEs. Each PE computes the values of its assigned cells (fills the similarity matrix) in two phases that, in total, take three clock cycles. During the first phase, the calculation phase, the value of each cell is roughly approximated by thecorresponding PEs. Then, the $Error_{flag}$ signal is asserted and the second phase, i.e. the error editing phase, starts. The main advantageof our structure is that the rows and columns calculation of the similarity matrix are performed independently. In some rare cases, the parallel calculations lead to approximation in the obtained scores. So, the editing phase (second phase) is needed to correct the approximated scores of the first phase.

Fig. 1 shows how PEs are assigned to cells in the similarity matrix in the proposed method and in the previously reported 1D-systolic array-based architectures. The cells with the same numbers in Fig. 1a are computed simultaneously and the cells with the same colors in Fig. 1b can be computed in parallel in the proposed architecture. It can be clearly seen in Fig. 1 that the similarity matrix can be filled in fewer steps by using the proposed architecture (three versus five).

Note that the 1D-systolic array needs only three PEs to fill the similarity matrix, while the proposed architecture uses five PEs. However, the main advantage of the proposed architecture is that more PEs are processed in parallel, thereby speeding up the process. Taking advantage of our two-phase architecture lets us better exploit the parallelism available in the process. The timing diagram of the proposed architecture is given in Fig. 2. As shown in this figure, when the first phase – the calculation phase – is completed, the PE may or may not perform the second phase. The error signal $Error_{flag}$ is updated after the end of the calculation phase and, if it is set, then the 2nd phase (the editing phase) needs to be performed.

The two phases of the proposed architecture and the details of its hardware implementation are thoroughly explained below.

3.1. Calculation Phase

During the calculation phase, an approximation of each cell is calculated. Depending on the position of the cell in the similarity matrix, this approximation follows different rules, as described below.

3.1.1. *Cells on the main diagonal*

For the cells on the main diagonal, we use the exact PE functionality, as given by:

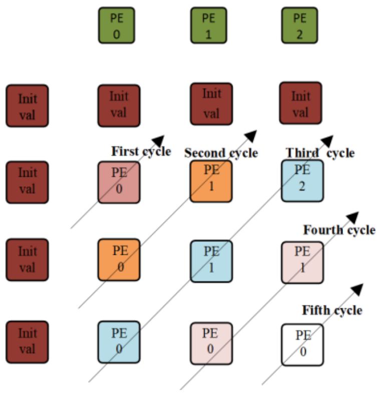
$$DP(i,i) = MAX \begin{cases} DP_{(i-1,i-1)} + T_{(Match,Miss-Match)} \\ DP_{(i-1,i)} + T_{(GAP)} \\ DP_{(i,i-1)} + T_{(GAP)} \\ 0 \end{cases} \quad (1)$$

where DP denotes the similarity matrix, $T_{(Match,\ Miss\text{-}Match)}$ is the assigned score for when a match or a mismatch occurs (usually +2 for a match and a -1 for a mismatch [11]), and $T_{(Gap)}$ is the gap penalty.

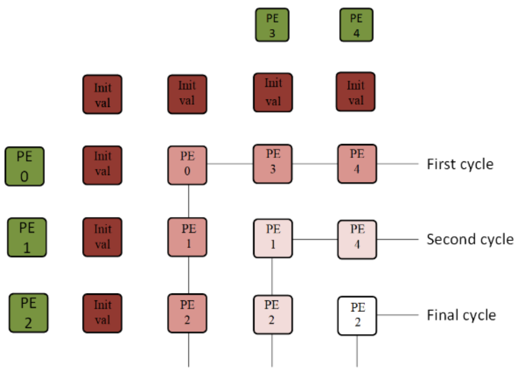
As shown, (1) accounts for all contributions neighboring cells (i, i) on the main diagonal and is thus 'exact'. However, since the adjacent cells, i.e., west, north, and north-west, have approximate values (see immediately following sub-section), the output of this equation would likely be an erroneous value.

3.1.2. *Other cells*

As implied above, the elements that are not placed on the main diagonal use only two of the three adjacent cells. Note that this approximation is a property of the underlying algorithm and not a specific hardware-related design choice. For the main diagonal of the cells below, we use the values of the west and the north-west cells as given by (2).



(a)



(b)

Fig. 1: The PE assignment in (a) 1D-Systolic array-based architecture and (b) the proposed design.

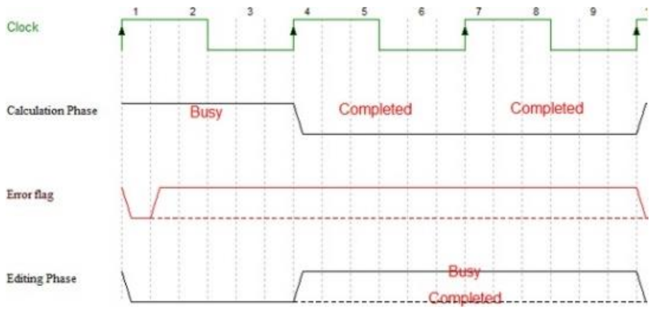


Fig. 2: Timing diagram of the proposed 2-dimensional architecture. The editing phase might or might not be required depending on the value of the $Error_{flag}$ signal.

$$DP(i,j) = MAX \begin{cases} DP_{(i-1,j-1)} + T_{(Match,Miss-Match)} \\ DP_{(i,j-1)} + T_{(GAP)} \\ 0 \end{cases} \quad (2)$$

$$DP(i,j) = MAX \begin{cases} DP_{(i-1,j-1)} + T_{(Match,Miss-Match)} \\ DP_{(i,j-1)} + T_{(GAP)} \\ 0 \end{cases}$$

For the cells on the top of the main diagonal, on the other hand, we use the values of the north and the north-west cells as given by:

$$DP(i,j) = MAX \begin{cases} DP_{(i-1,j-1)} + T_{(Match,Miss-Match)} \\ DP_{(i-1,j)} + T_{(GAP)} \\ 0 \end{cases} \quad (3)$$

Fig. 3 provides a graphical description of the three above-mentioned equations to make the proposed architecture clearer. As shown in Fig. 3, the cells on the main diagonal ((1, 1) and (2, 2)) use the values of the three adjacent cells. The cell below the main diagonal, i.e. (2, 1) uses the values of the west and the north-west adjacent cells. Finally, the cell on the top of the main diagonal, i.e., (1, 2) uses the values of the north and the north-west adjacent cells. Note that according to (1), (2), and (3), two types of PEs are required in the proposed architecture. The horizontal PEs are used to calculate the values of the cells on the same row, from the first column to the column on which the cell on the main diagonal is located, and the vertical PEs calculate the values of the cells on the same column, from the first row to the row just below the row on which the cell on the main diagonal is located. Note that the horizontal PEs are distinct from the vertical PEs as they use the values of different neighbors to calculate their outputs during the first phase. Unlike the existing architectures, including 1D-systolic arrays, the distinction between the horizontal and vertical PEs in this work allows us to perform the calculations independently and in parallel. Thus, the proposed architecture is notably faster than the existing designs, as will be discussed later in the simulation results section, Section IV. With the given approximations in (1), (2), and (3), cells would most likely have erroneous values. Thus, a second phase is required i.e., the editing phase to compensate for the introduced errors.

3.2. Editing Phase

Similar to the calculation phase, the editing phase is also done differently for the cells on the main diagonal and the other cells.

3.2.1. Cells on the main diagonal

For the cells on the main diagonal, our calculation phase was performed based on (1). Consider Fig. 3 where a subset of the similarity matrix is shown. The cells (0, 0), (1, 1), and (2, 2) are explained here, and calculations for other cells on the main diagonal would be just like the cell (2, 2).

Cell (0, 0): The value of this cell is calculated according to (1) by using PE11, see Fig. 3, during the calculation phase. Since (1) is the exact equation and the three cells that are used to calculate the value of the cell (0, 0) have all exact values (i.e. the initial values), there is no calculation error. Hence, the editing phase is not performed for the cell (0,0).

Cell (1, 1): The value of the cell (1, 1) is calculated based on the values of the cells (1, 0) and (0, 1), which are computed according to (2) and (3), respectively, and the value of the cell (0, 0). There is no error in the value of the cell (0, 0) and, therefore, to edit the value of the cell (1,1), we only need to edit the values of the cells (1,0) and (0,1).

The value of the cell (1, 0) is initially computed by ignoring the value of the cell (0, 0). Hence, the first step in editing this cell would be figuring out if (0, 0) was effective in the value of the cell (1, 0), which can be checked with the following condition: $DP_{(0,0)} + T_{(gap)} > DP_{(1,0)}$. If this condition is met, it means that an error has occurred in the calculation of the value of the cell (1, 0) and, therefore, the $Error_{flag}$ of $PE_{(12)}$ is set. Similarly, the cell (0, 1) should be checked for possible errors that happen in some seldom cases. Finally, the value of the cell (1, 1) is recomputed based on (1) to avoid any possible error if at least one of the two $Error_{flag}$ signals for the two corresponding PEs is set.

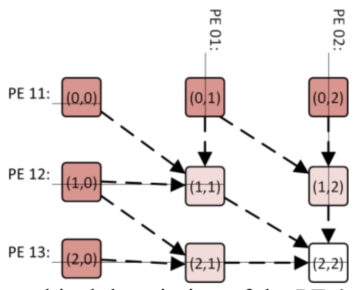


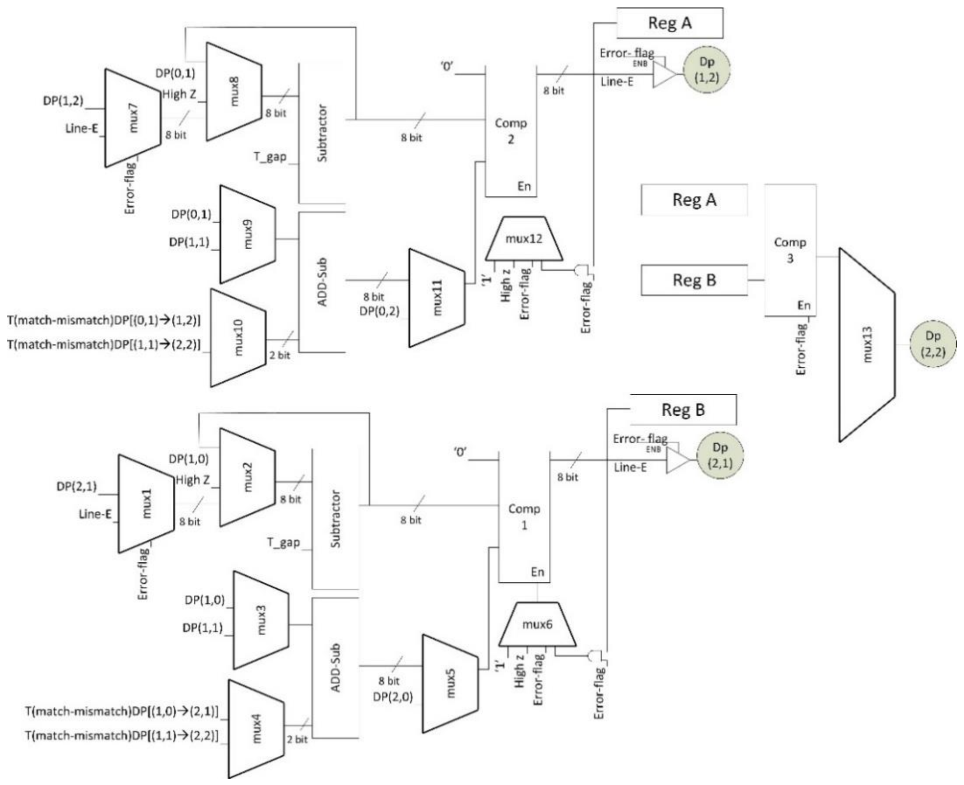
Fig. 3: The graphical description of the PEs' functionality during the calculation phase.

Cell (2, 2): The value of the cell (2, 2) is calculated based on the values of the cells (1, 1), (2, 1), and (1, 2). The details for this error correction are provided in Algorithm 1. The editing phase for the cell (1, 1) is already explained and the cells (2, 1) and (1, 2) have similar error editing calculations. Thus, only the calculation for the cell (2, 1) is explained here.

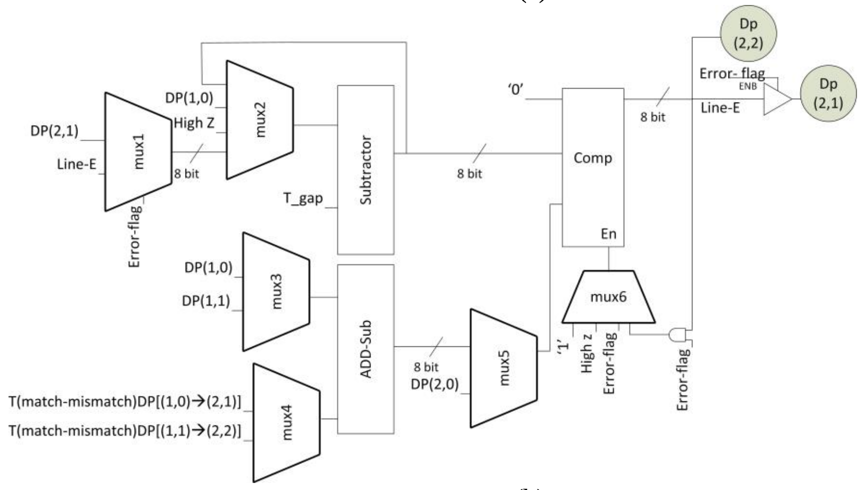
Cell (2, 1) might have an erroneous value caused by the values of the cells (1, 0) and (2, 0) and more importantly, by

Algorithm 1: Editing phase for the cells that are located on the maindiagonal.

```
1:  Editing by using horizontal PEs
2:  if $Error_{flag(PE_{12})} = 1$ then
3:      If : $DP_{(1,0)} + T(gap) > DP_{(2,0)}$ then
4:      $DP_{(2,0)} \leftarrow DP_{(1,0)} + T_{(gap)}$
5:       $DP_{(2,1)} = MAX \begin{cases} DP_{(1,0)} + T_{(Match,Miss-match)} \\ DP_{(2,0)} + T_{(gap)} \\ 0 \end{cases}$
6:      $Error_{flag(PE_{12})} \leftarrow 0$
7:      else
8:      $Error_{flag(PE_{12})} \leftarrow 0$
9:  end if
10: if $DP_{(1,1)} + T_{(gap)} > DP_{(2,1)}$ then
11:     $DP_{(2,1)} \leftarrow DP_{(1,1)} + T_{(gap)}$
12:     $Error_{flag(PE_{13})} \leftarrow 1$
13: else
14:     $Error_{flag(PE_{13})} \leftarrow 0$
15: end if
16:                                   ▷ Editing by using vertical PEs
17: if $Error_{flag(PE_{01})} = 1$ then
18:     If : $DP_{(0,1)} + T(gap) > DP_{(0,2)}$ then
19:     $DP_{(0,2)} \leftarrow DP_{(0,1)} + T_{(gap)}$
20:      $DP_{(1,2)} = MAX \begin{cases} DP_{(0,1)} + T_{(Match,Miss-match)} \\ DP_{(0,2)} + T_{(gap)} \\ 0 \end{cases}$
21:     $Error_{flag(PE_{01})} \leftarrow 0$
22:     else
23:     $Error_{flag(PE_{01})} \leftarrow 0$
24: end if
25: if $DP_{(1,1)} + T_{(gap)} > DP_{(1,2)}$ then
26:     $DP_{(1,2)} \leftarrow DP_{(1,1)} + T_{(gap)}$
27:     $Error_{flag(PE_{02})} \leftarrow 1$
28: else
29:     $Error_{flag(PE_{02})} \leftarrow 0$
30: end if
31:  $DP_{(2,2)} = MAX \begin{cases} DP_{(1,1)} + T_{(Match,Miss-match)} \\ DP_{(2,1)} + T_{(gap)} \\ DP_{(1,2)} + T_{(gap)} \end{cases}$
```



(a)



(b)

Fig. 4: The data path architecture of the proposed method: (a) horizontal PE and (b) vertical PE.

ignoring the value of the cell (1,1). Error editing for the cell (1, 0) is already explained, so the cell (2, 0) and the effects of ignoring the value of the cell (1, 1) need to be elaborated. According to Algorithm 1, the value of the cell (2, 0) is corrected by considering the gap penalty at the cell (1, 0) and the effect of ignoring the cell (1, 1) is addressed by reflecting the gap penalty at the cell (1, 1). Note that the entire calculation and edition phase are completed within three clock cycles and, therefore, a 3x3 subset as shown in Fig. 3 is large enough to explain the entire calculation. In fact, for larger matrices, the errors for the cells that are located further away will be completely edited within the three clock cycles and, therefore, we do not need to consider them in our calculations.

3.2.2. *Other cells*

Editing the values of the cells that are not located on the main diagonal is simpler than that of those on the main diagonal and can be easily concluded from Algorithm 1.

3.3. Hardware Implementation

This section describes the hardware implementation of the proposed horizontal and vertical PEs and describes how they calculate the roughly approximated values for each cell and also the error editing phase. Fig. 5 shows a simplified block diagram of the proposed architecture. As shown in this figure, the control unit manages the control signals on the data path. Additionally, the query and reference symbols should be stored in shared memory and the score arrays will be sent to the local memory for the editing and calculation phases.

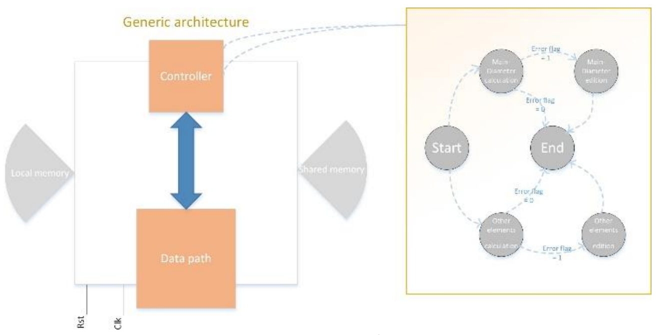


Fig. 5: The block diagram of the generic design.

Other than the control unit, an important part of the hardware implementation is the design of PEs. As mentioned earlier, two types of PEs are used in the proposed architecture i.e., vertical PEs and horizontal PEs. The detailed hardware implementation of these PEs is described below.

3 . 3 . 1 . Vertical PE

A vertical PE does not compute the cells on the main diagonal and, therefore, implements simpler equations. Hence, they are less complex and, of course, have faster designs than the horizontal PEs. Fig. 4b shows the functionality of the proposed architecture

3.3.2. Horizontal PE

Unlike vertical PEs, horizontal PEs are further used to compute the cells on the main diagonal. Hence, they are more complex and, consequently, have slower designs. The hardware

4. Simulation Results

This section is further divided into two sections. The first section provides the FPGA and ASIC implementation results of the proposed architecture, while the second section analyzes the performance of the proposed architecture in terms of the total execution time.

4.1. Synthesis Results

The proposed architecture is implemented in VHDL hardware description language and synthesized on (1) FPGA and (2) using an ASIC implementation.

Table 2 reports the resource utilization of the horizontal and vertical PEs when implemented on Virtex-7 and Spartan-7 FPGAs. Moreover, both designs are implemented by using Synopsys Design Compiler for 180-nm CMOS technology, and the results are provided in Table 3.

Our simulation results, specifically the results in Table 2, show that the proposed architecture can operate at 360*Mhz* on Virtex 6. Note that this is the worst-case scenario, i.e., all of the cells have erroneous values and the error editing phase needs to be performed for all of them.

4.2. Performance

In this section, we analyze the worst-case execution time of the proposed architecture. Given that each PE requires three clock cycles to accomplish its computations, the total execution time τ_{total} for an $(N+1)$x$(N+1)$ similarity matrix (a seed with the length of N) can be calculated as:

$$T_{total} = N \times 3 \times \frac{1}{clk_{horizontal}} \tag{4}$$

where $clk_{horizontal}$ is the maximum clock frequency of the horizontal PEs, i.e., the slowest part of the proposed architecture. Note that in an $(N+1)$ x $(N+1)$ similarity matrix with a seed length of N, the first row and the first column of the matrix are initial values, which are calculated by the SMEM kernel. We need three clock ticks for each PE in the worst case, and all of the matrix elements will be calculated in N cycles of the calculation, which is shown in Fig. 1.

Compared to the state-of-the-art 1D-systolic array-based architecture that computes an N x N matrix in 2N-1 clock cycles, the proposed architecture is faster as it only requires N-1 clock cycles. We compare our architecture performance with the original algorithm in terms of execution time on Intel(R) Core(TM) i7-4702 MQ CPU. The BWA-MEM 0.7.11 is used and run over the free NCBI dataset [23]. Since we are interested in comparing the execution time of the seed extension kernel, we only used the output of the SMEM kernel as the input to the proposed design. Note that the speedup in Table 4 is the average speedup over 1000 randomly selected inputs from the dataset. The comparison results are provided in Table 4 and Fig. 6. Fig. 6a provides a comparison of the execution time of our proposed method and 1-D systolic architecture based on different seed lengths. As indicated from this figure, our execution time decreases asymptotically with the growth of seed length, and our acceleration pace is fixed for longer seed lengths. This speed-up ratio is illustrated in Fig. 6b.

Area overhead of both 1D-Systolic arrays and our approach is enlightened herein. Based on [11], implementation results considering a 131*131 array on Xilinx Virtex-6 LX760 FPGA show 4% of total FGPA resources, while our approach utilizes almost 13.5% of Xilinx Virtex-6 LX760 FPGA resources that is about 3.3x more area overhead than the 1D-systolic. This increase in area overhead is due to the addition of the editing phase, which lets us for the first time eliminate data dependency in calculating the Smith-Waterman matrix in a 2D manner (Fig. 1). However, the area is not usually a bottleneck in the implementation of the BWA-MEM algorithm on FPGA, and the execution time is a more important concern. The main contribution of our work lies in the fact that we have eliminated the data dependency to devise a new 2D approach to speeding up the seed extension kernel. It is

Table 2: Result of FPGA implementation.

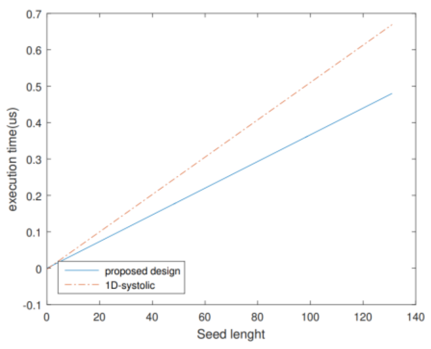
FPGA	PE	LUT			FF			IO		BUFG		Freq.
		Usage	Utilization	Available	Usage	Utilization	Available	Usage	Utilization	Usage	Utilization	(MHz)
xc7s50csga324-2	Vertical	89	0.2%	32600	18	0.03%	65200	72	32%	1	3.13%	189
xc7s50csga324-2	Horizontal	205	0.7%	32600	39	0.05%	65200	123	57.7%	1	3.13%	185
xc7v58tffg1157-3	Vertical	89	0.02%	364200	18	0.01%	728400	72	11.8%	1	3.13%	363
xc7v58tffg1157-3	Horizontal	205	0.05%	364200	39	0.01%	728400	123	20.4%	1	3.13%	360

Table 3: The ASIC design of the proposed architecture.

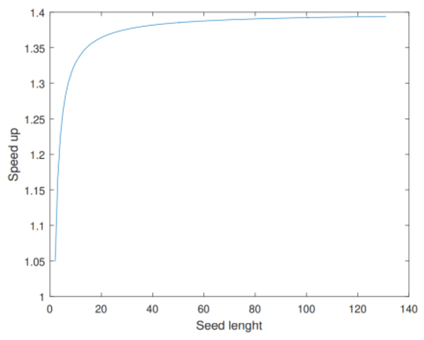
PE Leakage Type	Vertical	Horizontal
Critical path delay (ns)	3.2	3.22
Cell area (nm2)	10245.31	21584.18
Number of nets	446	830
Levels of logic	14	18
Dynamic power (nW)	2.23	4.4
Power (nW)	366.7	747.3

Table 4: Execution time comparison of CPU vs. the proposed architecture (*us*).

Matrix dimension	i7-4702	Proposed architecture	FPGA Utilization	Speedup
10 × 10	8.956	0.0366	0.95%	245 ×
25 × 25	43.178	0.091	2.45%	474 ×
50 × 50	105.24	0.183	4.95%	576 ×
100 × 100	198.35	0.366	9.95%	542 ×
131 × 131	244.21	0.48	13.5%	509 ×



(a)



(b)

Fig. 6: The proposed architecture with operating Freq. 360 MHz vs.1-D systolic arrays with operating Freq. 380 MHz (on the same platform Xilinx Vertix 6), (a) Execution time, and (b) Speedup.

obvious that our proposed architecture can benefit from more parallelism for computing different seeds in parallel, and we will not face the lack of FPGA resources.

5. Discussion

The proposed architecture can be implemented on a Virtex 6 FPGA device with a typical 131 x 131 matrix dimension and a minimum of 19 base-pairs seed length from a read containing 150 base-pairs, while using less than 15% of the FPGA available resources. This indicates that the proposed architecture can benefit from more parallelism for computing different seeds in parallel. Our simulation results show up to 570x and 1.4x acceleration and speedup in comparison to software execution and the 1D-systolic arrays, respectively in case of the 131 × 131 matrix dimension on Virtex 6 FPGA.

6. Conclusion

We have proposed a newly developed algorithm of the seed extension kernel for the well-known BWA-MEM sequence alignment algorithm. This novel 2-dimensional, high-throughput seed extension kernel is implemented on FPGA. We achieved up to 570x speedup with 360MHz operating frequency in comparison to the Intel(R) Core(TM) i7-4702 MQ CPU. Also, we achieved up to 1.4x speedup in comparison to the 1D-systolic arrays implemented on the same hardware as our architecture. Our simulation results show that the proposed architecture can work with up to 312 MHz frequency in Synopsys Design-Compiler for 180-nm CMOS technology. By introducing the calculation and error editing phase, we notably reduced the number of required clock cycles for filling an $N \times N$ similarity matrix. In fact, the state-of-art 1D-systolic array-based architectures fill the NxN similarity matrix in $2N$-1 cycles, while our proposed architecture only takes N-1 cycles to fill the same matrix.

Credit Authorship Contribution Statement

Mahdi Taheri: Conceptualization, Formal analysis, Methodology, Resources, Roles/Writing - original draft, Writing - review & editing. **Ali Mahani:** Formal analysis, Project administration, Software, 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, and redundancy, have been completely observed by the authors.

References

[1] H. Jung, C. Winefield, A. Bombarely, P. Prentis, and P. Waterhouse, "Tools and strategies for long-read sequencing and De novo assembly of plant genomes," *Trends in plant science,* vol. 24, no. 8, pp. 700 -724, 2019,

[2] C. Pham-Quoc, B. Kieu-Do, and T.N. Thinh, "A high-performance FPGA-based BWA-MEM DNA sequence alignment," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 18, pp. 5328, 2019.

[3] Li. Wu *et al.*, "FPGA accelerated INDEL realignment in the cloud," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 277–290.

[4] C. Pham-Quoc, B. Kieu-Do, and T. Ngoc Thinh," An FPGA-based seed extension IP core for BWA-MEN DNA alignment," in *2018 International Conference on Advanced Computing and Applications*, 2018, pp. 1–6.

[5] S. Canzar, and S. Salzberg, "Short read mapping: an algorithmic tour," *Proceedings of the IEEE*, vol. 105, no.3, pp. 436-458, 2015.

[6] D. Fujiki *et al.*, "Genax: a genome sequencing accelerator," in *45th Annual International Symposium on Computer Architecture*, 2018, pp. 69–82.

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

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

[9] R. Li *et al.*, "SOAP2: an improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, no, 15, pp. 1966–1967, 2009.

[10] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," *ArXiv Preprint ArXiv*, vol. 1303, pp. 3997, 2013.

[11] E. J. Houtgast, V. M. Sima, K. Bertels, and Z. Al-Ars, "An FGPA-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.

[12] J. Cohen, "Bioinformatics an introduction for computer scientists," *Association for Computing Machinery Computing Surveys*, vol. 36, no. 2, pp. 122–158, 2004.

[13] H. Cao *et al.*, "A short-read multiplex sequencing method for reliable, cost-effective and high-throughput genotyping in large-scale studies," *Human Mutation*, vol. 34, no. 12, pp. 1715–1720, 2013.

[14] K. Hammad, Z. Wu, E. Ghafar-Zadeh, and S. Magierowski, "A scalable hardware accelerator for mobile dna sequencing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 2, pp. 273–286, 2021.

[15] M. Taheri, and A. Mahani, "Development and hardware acceleration of a novel 2-D BWA-MEN DNA sequencing alignment algorithm," *1st Conference on Applied Research in Electrical Engineering (AREE)*, 2021.

[16] T. J. Ham *et al.*, "Genesis: A hardware acceleration frame- work for genomic data analysis," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 254–267.

[17] Y. L. Chen, B. Y. Chang, C. H. Yang, and T. D. Chiueh, "A high-throughput FPGA accelerator for short-read mapping of the whole human genome," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, pp. 6, pp. 1465–1478, 2021.

[18] Y. T. Chen, J. Cong, Z. Fang, Ji. Lei, and P. Wei, " When spark meets FPGA: A case study for next-generation DNA sequencing acceleration," in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, June 2016.

[19] P. Faes *et al.*, "Scalable hardware accelerator for comparing DNA and protein sequences," in *Proceedings of the 1st International Conference on Scalable Information Systems*, 2006, pp. 33–es.

[20] M. Taheri, S. Ansari, S. Magierowski, and A. Mahani, "Hardware acceleration of burrows-wheeler aligner algorithm with maximal exact matches seed extension kernel," *IET Circuits, Devices and Systems*, vol. 15, no. 3, pp. 1–10, 2020.

[21] M. Taheri, H. Zandevakili, and A. Mahani, "A high-performance memristor-based smith-waterman DNA sequence alignment using FPNI structure," *Iranian Association of Electrical and Electronics Engineers (IAEEE).* vol. 1, no. 1, pp. 59-68, 2020.

[22] E. J. Houtgast, V. M. Sima, K. Bertels, and Z. Al-Ars, "Hardware acceleration of BWA-MEN genomic short read mapping for longer read lengths," *Computational Biology and Chemistry*, vol. 75, pp. 54–64, 2018.

[23] D. Wheeler *et al.*, "Database resources of the national center for biotechnology information," *Nucleic Acids Research*, vol. 35, no. suppl_1, pp. D5–D12, 2006.

BIOGRAPHY

Mahdi Taheri received his B.Sc. degree in Electronic Engineering from the Khaje Nasir University of Technology (KNTU), Tehran, Iran in 2017 and his M.Sc. degree in Electronic Engineering from Shahid Bahonar University, Kerman, Iran in 2020. Since then, he was with the RSS Lab at Shahid Bahonar University for 1 year, and now, he is studying for his Ph.D. at the Tallinn University of Technology (TalTech). His research interests focus on hardware assessment and reliability of neural networks, fault- tolerant design, and FPGA-based accelerators.

Ali Mahani received his B.Sc. degree in Electronic Engineering from Shahid Bahonar University, Kerman, Iran in 2001 and his M.Sc. and Ph.D. degrees both in Electronic Engineering from the Iran University of Science and Technology, Tehran, Iran in 2003 and 2009, respectively. Since then, he has been with the Electrical Engineering Department of Shahid Bahonar University where he is currently an associate professor. His research interests focus on fault-tolerant design, FPGA-based accelerators, approximate digital circuits, stochastic computing, and networked systems.