The following publication H. -C. Ng, S. Liu, I. Coleman, R. S. W. Chu, M. -C. Yue and W. Luk, "Acceleration of Short Read Alignment with Runtime Reconfiguration," 2020 International Conference on Field-Programmable Technology (ICFPT), Maui, HI, USA, 2020, pp. 256-262 is available at https://doi.org/10.1109/ICFPT51103.2020.00044.

Acceleration of Short Read Alignment with Runtime Reconfiguration

Ho-Cheung Ng*, Shuanglong Liu†, Izaak Coleman‡, Ringo S.W. Chu§, Man-Chung Yue¶and Wayne Luk*

*Department of Computing, Imperial College London, London, United Kingdom {*h.ng16, w.luk*}*@imperial.ac.uk*
†School of Physics and Electronics, Hunan Normal University, Changsha, China *liu.shuanglong@hunnu.edu.cn*
‡ Department of Systems Biology, Columbia University, New York, USA *ic2465@cumc.columbia.edu*
§ Department of Computer Science, University College London, London, UK *ringo.chu.16@ucl.ac.uk*
¶ Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong *manchung.yue@polyu.edu.hk*

***Abstract*—Recent advancements in the throughput of next-generation sequencing machines pose a huge computational challenge in analyzing the massive quantities of sequenced data produced. A critical initial step of genomic data analysis is short read alignment, which is a bottleneck in the analysis workflow. This paper explores the use of a reconfigurable architecture to accelerate this process, based on the seed-and-extend model of Bowtie2. In the proposed approach, complete information available in sequencing data is integrated into an FPGA alignment pipeline for biologically accurate runtime acceleration. Experimental results show that our architecture achieves a similar alignment rate compared to Bowtie2, mapping reads around twice as fast. Particularly, the alignment time is reduced from 50 minutes to 26 minutes when processing 300M reads.**

I. Introduction

The significant advancement and decrease in the cost of next-generation sequencing (NGS) have driven a dramatic increase in genomic data. The NGS machines are now capable of generating millions or even billions of short DNA fragments from the sampled cells within hours [1]. These fragments, *i.e.* reads, are produced by randomly segmenting the sampled DNA strand. As a consequence of this action, the orientation and position information of the reads with respect to the original DNA is lost. Therefore, a crucial initial step of genomic analysis is short read alignment (mapping), where the short reads generated by the NGS machine are mapped to a reference genome.

However, the improvement of NGS technology has been far exceeding Moore's Law for a decade. Read alignment using software becomes unworkable because of its prolonged execution time [2]. This hinders the medical applications of NGS, such as prenatal diagnostics and monitoring [3], [4] where individuals' DNA and RNA should be analyzed quickly at a low cost. Therefore, it is imperative to accelerate short read alignment, so as to bridge the gap between alignment research and practice.

FPGA has shown to be a promising candidate to accelerate short read mapping because of its highly-parallel bit-wised architecture [5]. As the sequenced alphabet produced by the NGS machines can be abstracted into {A, C, G, T, N} which is represented using 3 bits, mapping the DNA nucleotides to the reference genome is inherently a bitwise operation. Different algorithms, for example, the FM-index [6] and Smith-Waterman [7], have been implemented and accelerated on FPGA, as they are frequently used in popular software such as Bowtie2 [8] and BWA-MEM [9].

Despite the success, FPGA-accelerated alignment is still rarely adopted in genomic research and clinical applications. The reason is two-fold:

- Most accelerators have failed to utilize the complete information available in NGS data. For hardware simplicity, they utilize only the Watson-Crick alphabets {A, G, C, T}, discarding the quality metric information and ambiguous characters (N characters) that are commonly present in NGS data. This can result in incorrect alignment and generate biologically invalid results, which are catastrophic for applications such as prenatal diagnostics and monitoring.
- Many FPGA researchers select and accelerate alignment algorithms that are in favor of hardware implementations. As a result, the alignment workflow can be inconsistent with state-of-the-art software. For example, Arram *et al.* [10] propose an FPGA aligner that performs exact string matching (ESM) based on the FM-index, followed by approximate string matching (ASM) based on the seed-and-extend model. Only reads unmapped by the ESM are directed to the ASM for processing to improve hardware performance. However, this methodology is inconsistent with the alignment model in popular software such as BWA [11], Bowtie [12] and Bowtie2, which in turn limits the biological validity and reproducibility of the alignment outputs from hardware.

To increase the utility of short read mapping accelerators in real applications, we propose a two-stage alignment architecture that is similar to the seed-and-extend model adopted by Bowtie2. By exploiting the ***runtime reconfigurability*** of FPGA, we decouple the seeding and extension stage into two separate FPGA configurations to achieve a balanced, fully optimized ***alignment pipeline***. Finally, we consider complete NGS data including quality metrics (Phred quality) and ambiguous characters (N characters) during the alignment runtime to ensure accuracy. The main contributions of this work are as follows:

- A novel alignment architecture that is composed of a two-

stage configuration alignment pipeline. It exploits the reconfigurability of FPGA to achieve highly efficient implementation for each configuration and a fully optimized alignment pipeline.
- A hardware implementation of the reconfigurable architecture targeting a single FPGA. The seeding stage is based on an FM-index implementation and the extension stage is based on a Smith-Waterman implementation with affine-gap model.
- An evaluation of the optimal hardware architecture based on the target platform Xilinx VU9P, together with comparisons against the state-of-the-art software Bowtie2 on multi-core processor and some of the existing FPGA solutions.

II. Background and Related Work

This section provides an overview of the algorithms used in our two-stage architecture: the FM-index and Smith-Waterman with affine gap model. These two algorithms are also used in Bowtie2 to realize the seed-and-extend model, which is essentially the workflow of our accelerator.

A. Alignment Algorithms

FM-index — The FM-index [6] is a suffix-trie method that combines the properties of suffix array with the Burrows-Wheeler transform (BWT) [13]. It provides an efficient mechanism to exactly align a pattern P to the reference genome R, with a time complexity linear to $|P|$. This enables a rapid search of the pattern in the reference and quickly narrows the list of candidate alignment locations.

To compute the BWT of R, denoted by $BWT(R)$, we append a terminating character '$' to R, which is lexicographically the smallest value. We then generate and sort all the rotations of R correspondingly, forming a sorted rotation list. The suffix array can be obtained by considering the characters before '$' in each entry of the sorted rotation list. $BWT(R)$ can also be formed by selecting and concatenating the last characters of all the entries on the sorted list.

To generate the FM-index, we need to extend R and $BWT(R)$ to form two functions: $i(x)$ and $c(n, x)$. For each character x, $i(x)$ is the number of characters in R that is lexicographically smaller than x, while $c(n, x)$ stores the number of occurrences of x in $BWT(R)$ from position 0 to position $n-1$. Table Ia demonstrates an example of deriving the BWT of an example reference genome $R =$ CACGT. The strings preceding the '$' sign in the sorted rotation list form the suffix array (SA), which indicates the position of each possible suffix in the original reference. Table Ib illustrates the $i(x)$ and $c(n, x)$ functions for the reference R.

Alignment of reads using the FM-index operates on the functions $i(x)$ and $c(n, x)$ recursively. Two pointers top and $bottom$ are defined to perform the search. top refers to an index of the SA element where a specific pattern is firstly located, and $bottom$ is the location where the pattern can be lastly found. If $bottom$ points to an index that is less than or equal to the index pointed by the top, the pattern does not occur on the text.

TABLE I: (a) Example of deriving the suffix array and BWT of reference genome R. (b) $i(x)$ and $c(n, x)$ functions for R.

(a)

$R =$ CACGT$		
Index n	SA	Sorted Rotations
0	5	$CACGT
1	1	**ACGT**$C
2	0	**CACGT**$
3	2	**CGT**$CA
4	3	**GT**$CAC
5	4	**T**$CACG
$BWT(R) =$ TC$ACG		

(b)

$c(n, x)$

Index n	A	C	G	T
0	0	0	0	0
1	0	0	0	1
2	0	1	0	1
3	0	1	0	1
4	1	1	0	1
5	1	2	0	1
6	1	2	1	1

$i(x)$ $\{1, 2, 4, 5\}$

To search for a specific pattern P with the FM-index, one character is processed at a time, starting with the last character of P. The top and $bottom$ are first initialized with the first and last indices of the $c(n, x)$ function respectively. Then both pointers are updated according to the following equations:

$$\begin{aligned} top_{new} &= c(top_{current}, x) + i(x) \\ bottom_{new} &= c(bottom_{current}, x) + i(x) \end{aligned} \tag{1}$$

The final results of top and $bottom$ are the range of indices in SA that contains P as the prefix.

Smith-Waterman Algorithm with Affine Gap Model — The Smith-Waterman is a dynamic programming (DP) technique to perform approximate string matching. It uses a scoring matrix V to reveal the optimal local alignment between two sequences P and R, where $|P| = m$ and $|R| = n$. Every entry in V is calculated recursively according to the following equation:

$$V(i,j) = \max \begin{cases} 0 \\ V(i-1, j-1) + \sigma(P[i], R[j]) & Match/Mismatch \\ s_{i,j,\rightarrow} & Deletion \\ s_{i,j,\downarrow} & Insertion \end{cases} \tag{2}$$
$$\text{for} \quad 1 \le i \le m, 1 \le j \le n$$

The function $\sigma(x, y)$ determines the relative weighting of match or mismatch between characters x and y. Equation (3) explains the affine gap functions $s_{i,j,\rightarrow}$ and $s_{i,j,\downarrow}$. The $\rightarrow$ and $\downarrow$ denote a character deletion and a gap insertion in P respectively.

$$s_{i,j,\rightarrow} = \max \begin{cases} V(i-1, j) - \alpha \\ s_{i-1,j,\rightarrow} - \beta \end{cases} \quad s_{i,j,\downarrow} = \max \begin{cases} V(i, j-1) - \alpha \\ s_{i,j-1,\rightarrow} - \beta \end{cases} \tag{3}$$

α is the cost for opening-gap penalty while β is the cost for continuous-gap penalty, with $\beta < \alpha$. Accordingly, the penalty for initiating the gap is more expensive than the gap extension. Compared to the basic Smith-Waterman algorithm with a standardized σ function for all match/mismatch, insertion and deletion, the affine gap model provides a more realistic computation since a mutation normally causes insertion/deletion of a large block.

Seed-and-Extend Model — Despite the Smith-Waterman algorithm provides an optimal alignment for alignment with match/mismatch, insertion and deletion, its time complexity is

$O(mn)$ which is prohibitively expensive given the reference genome is three billion base pair (bp) in length. Therefore, the general strategy for approximate alignment in Bowtie2 or BWA-MEM is based on the seed-and-extend model. The short read is first partitioned into fixed-length subsequence, *i.e.* seeds. To achieve high speed, algorithm such as the FM-index is used to identify the possible matching locations by exactly aligning seeds to the reference. Finally, the short read is then mapped to the reference using the Smith-Waterman algorithm at each matching location. This model substantially improves the alignment efficiency with a negligible accuracy loss.

B. Related Work

Comprehensive surveys regarding the FPGA acceleration of short read alignment are provided in [2], [14]. Basically, many of the existing accelerators are based on the suffix-trie method, or the Smith-Waterman algorithm, or both. For example, Fernandez *et al.* [15] and Draghicescu *et al.* [16] introduce the use of FM-index and BWT to accelerate exact and approximate string matching respectively. In [10], Arram *et al.* propose a reconfigurable aligner that performs exact alignment with the FM-index, and approximate alignment with the seed-and-extend model. Recently, Fei *et al.* [17] propose FPGASW that accelerates the Smith-Waterman algorithm with affine gap model for short read mapping. It uses techniques such as overlapping memory access latency and data dependency elimination to achieve acceleration. Despite the significant speed-up given by these works, their underlying algorithms are not consistent with the ones in popular software. Complete NGS information is also not considered in these accelerators.

Processing the complete NGS data undoubtedly complicates the FPGA implementation. Some researchers work around this problem by proposing a software and hardware co-design based on the *defacto* software framework. For example, the authors in [16] develop a BWT accelerator that ties into existing software BWA. Chang *et al.* [18] propose a similar approach where the seeding stage of BWA-MEM [9] is accelerated on FPGA while retaining the expansion stage on CPU. The works [19], [20] implement a similar approach. However, these designs still suffer from the software bottleneck and communication overhead, unless the implementation is based on a multi-FPGA platform as in [16].

This paper presents an alignment pipeline that is similar to the algorithmic workflow of Bowtie2, with a substantial extension to the works [21], [22]. With the consideration of quality metrics and N characters, these previous works accelerate the Smith-Waterman algorithm with affine-gap model. However, they only provide a sub-optimal speed-up since seeding still relies on the processor. Koliogeorgi *et al.* [22] reported only a $35\,\%$ performance gain when Bowtie2 is required to locate the seeds.

III. Reconfigurable Architecture

State-of-the-art software usually applies a multi-stage approach to achieve short read mapping. In Bowtie2, it uses the seed-and-extend model where seed locations are calculated using the FM-index in the first stage. The seeds, together with all possible candidate locations, are extended in the second stage with the Smith-Waterman algorithm. This multi-stage approach is mainly a performance consideration, as it partitions the alignment workflow into smaller stages. Different techniques such as SIMD can be applied to each stage so as to fully utilize the modern CPU hardware.

Our accelerator architecture is based on a similar approach in Bowtie2 where we divide the entire alignment workflow into a seeding stage and an extension stage. Each stage corresponds to an individual FPGA implementation. We propose a novel ***runtime reconfigurable architecture*** where each implementation is executed ***in order*** on FPGA, forming an alignment pipeline. The candidate alignment locations computed by the first stage are temporarily buffered on the host and are redirected to the extension stage after runtime reconfiguration.

A. Motivations for Runtime Reconfiguration

In the various efforts that accelerate alignment with FPGA, the target device is ***statically*** configured with a circuit functionally equivalent to multiple alignment algorithms. A module within a circuit that implements a specific algorithm is interlinked with other modules, forming a complete alignment workflow. For example, the FPGA design in [15] is composed of interconnected modules that implement exact-match alignment with standard FM-index, one-mismatch, and two-mismatch alignment with backtracking FM-index. Despite a static configuration of these modules as a single implementation eliminates the time for reconfiguration, it suffers from several limitations that reduce the overall performance and usefulness of the design.

Significant Amount of Data Hazard — A typical alignment workflow is composed of multiple stages in which the execution of the next stage depends on the results from the current one. In Bowtie2, a read is only processed by the extension subroutine if the seeds can be identified and exactly aligned to the reference. This incurs numerous data hazards for a statically configurable circuit, as some of the seeds can fail to be exactly aligned. Since all the stages are mapped onto FPGA statically, data hazards result in idle cycles for some modules which reduce hardware efficiency.

Distinct Module Latency — Different alignment modules require a distinct number of cycles to process a read. Therefore, some modules are replicated more than others to maintain a balanced pipeline in a static configuration. For example, when the seed-and-extend model is mapped onto FPGA, the Smith-Waterman circuit must be replicated more to match the throughput of the FM-index module for seed location. This can be difficult and even impossible due to the limited resources available on FPGA.

Inflexible Parameters — Depending on the alignment requirements and experiments performed, parameters such as the maximum number of gap sizes or mismatches can be

different. For a statically configured design where the hardware is fixed, we have limited control over these parameters. Any major change on a particular stage in the alignment workflow requires a tremendous amount of redesign efforts, and it suffers from a prolonged re-placement and re-routing time. This substantially reduces the usefulness and limits the flexibility of an alignment architecture.

IV. Architecture Implementation

This section discusses the FPGA implementations of the two-stage alignment pipeline, targeting a single FPGA device. Particularly, the modules that handle the quality metrics and N characters are elaborated. The techniques used to improve the alignment performance are also discussed.

A. Seeding: FM-index Implementation

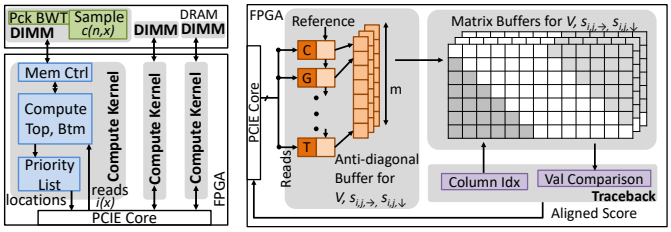
The FPGA configuration of the first stage implements the seed extraction and seed alignment using the FM-index, which is stored on each DIMM of the onboard memory. Basically, the seeding process begins with streaming the reads from the host. By following a similar seeding strategy in Bowtie2, 16 nucleotides are extracted as seeds for every 10 nucleotides in each read. Then every seed is exactly aligned to the reference using the FM-index with the pointers Top and $Bottom$. The pointers Top and $Bottom$ are updated based on the current character in the seed, and the correlated $i(x)$ and $c(n, x)$ values from the onboard memory. The final values for these pointers indicate the SA range for the seed. For each read, we maintain a priority list such that seeds with a smaller SA range are given higher priority in the extension stage. Figure 1a displays a simplified top-level diagram that implements seed extraction and seed alignment using the FM-index.

The implementation of FM-index features the following optimizations to improve the alignment efficiency:

2-bit Representation — Despite the DNA sequence produced by the NGS machines can be abstracted into {A, C, G, T, N}, statistically, only a small portion of reads contains N characters. Based on our observation on accession ERR194147 [23] generated by Illumina HiSeq 2000 sequencing machine, less than 4% of the reads consist of N characters. With the aim to optimize the hardware efficiency, each nucleotide is represented in 2 bits in the FM-index circuit. Reads containing N characters are not transferred to FPGA and are instead processed by CPU for seed extraction and seed alignment. Given that only an inconsiderable amount of reads contains the N character, the seeding on the processor can be completely overlapped by the execution of FPGA. Note that seeds containing N characters are considered unaligned.

Index Compression — When the human reference genome is converted into FM-index, the resulting table $c(n, x)$ is around $51\,\mathrm{GB}$. This is often far larger than the capacity of onboard memory. To compress the index size so that multiple copies of the index can be associated with multiple kernels, we only store a subset of $c(n, x)$. The remaining entries are substituted with a portion of the original BWT. Intrinsically, we sample every d entry of $c(n, x)$ and pack the BWT in the range of every d and $d-1$ alongside, forming a bucket. Normally we set the bucket being a multiple of the burst size to fully utilize the memory bandwidth. During a character search, the missing entries can be recalculated on-the-fly using the packed BWT. The required memory storage is significantly reduced to:

$$\begin{aligned}&\text{Sampled } c(n,x) \text{ Size} + \text{Packed } BWT \text{ Size}\\ =&\frac{3.2G \times 32bit}{d} \times 4 + 3.2G \times 2bit\end{aligned} \tag{4}$$



(a) (b)

Fig. 1: Simplified top-level diagram for (a) the FM-index implementation for the seeding stage, (b) the Smith-Waterman implementation for the extension stage. For readability, some data and control paths are omitted in both graphs.

Concurrent Processing of Multiple Reads — Based on Equation II-A, the new pointers for Top and $Bottom$ are calculated based on the values from $c(n, x)$, which in turn depends on the pointers' values from existing iteration. This incurs a large amount of stall in between memory access. We negate the latency of memory access by interleaving the processing of multiple reads. In this approach, a BRAM buffer is used to store a few reads during alignment. A read is selected from the buffer every cycle and the next symbol is processed, making a corresponding memory request. This enables the processing of reads while others are waiting for the memory data. As a result, the FM-index circuit is fully utilized as almost one nucleotide can be processed in each cycle.

B. Extension: Smith-Waterman Implementation with Affine-Gap Model

Figure 1b displays the top-level architecture of the Smith-Waterman implementation. It is composed of three major parts: systolic arrays that fill up $s_{i,j,\rightarrow}$ and $s_{i,j,\downarrow}$ and V, three matrix buffers, and a traceback unit. Our design draws inspiration from the work in [22], however, we rely on our FM-index circuit to locate the seed instead of using the software Bowtie2. Basically, accelerating both the seeding and extension steps provide a more superior speed-up, as the acceleration of only one step brings limited improvement based on Amdahl's law.

The systolic array consists of a pipeline of cells where each of them computes one score for the matrices. The cells are parallel-loaded with one base of the short read and its Phred quality, while the reference extracted in the proximity of the matching position is shifted through the array. This allows the calculation of the anti-diagonal of the matrix in parallel, which

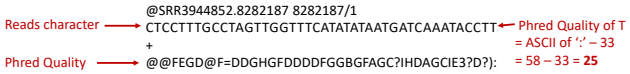


Fig. 2: Explanation on Phred Quality using an example read.

decreases the time complexity from $O(mn)$ to $O(m+n)$. The result of the current anti-diagonal is buffered with registers and is reused in the computation of the next anti-diagonal.

In the second part, we use block RAM to create matrix buffers in order to store up every score for $s_{i,j,\rightarrow}$ and $s_{i,j,\downarrow}$ and V during the anti-diagonal calculation. Despite the calculation does not require buffering every value in the matrices, the complete storage reduces the efforts of the traceback unit. It can reconstruct the alignment path by traversing the matrix in reverse order, instead of recalculating the values and refilling the matrices.

Consideration of all NGS Data — A major component of this work is the capability of recognizing the complete NGS data in the alignment process. One important information is the quality metrics, *i.e.* Phred quality [24], a metric that is usually neglected in previous works. Its values are ranged from 0 to 42 and are presented in ASCII character (with addition of 33). Figure 2 displays an example snippet of a read file in which the second line is the read and the fourth line is Phred quality. Each DNA character is associated with a quality score q in the same position and a larger value represents a better quality.

The quality metrics and N characters are processed completely in the Smith-Waterman implementation. The reference and the reads are represented in 3 bits. The Phred quality, which is transferred alongside the reads, is loaded in the systolic array and used in $\sigma(P[i], R[j])$ for mismatch calculation. Despite the increase in the data transfer, the entire operation is highly computationally intensive and is therefore bound by the matrix computation. Note that based on Bowtie2, the σ function is given by: $2 + floor(4 \times min(q, 40.0)/40)$.

V. Evaluation and Discussion

A. Experimental Parameters

In this section, we evaluate the alignment pipeline on Maxeler's MAX5C DFE provided by Maxeler Technologies [25]. The MAX5C DFE is equipped with Xilinx Virtex UltraScale+ VU9P and three DIMMs of 16 GB onboard DRAM. MaxCompiler 2018.2 and Vivado 2017.4 are used for synthesis and implementation. The FPGA runs at 200 MHz while the host runs with Intel Xeon CPU E5-2643 and 64 GB DDR4-2400 memory. PCI-e 2.0 $\times 8$ is used to transfer the data between the FPGA and the host, which runs with Centos 7.0. As the onboard DRAM in the FM-index implementation and the matrix computations in the Smith-Waterman circuit are the major alignment bottlenecks, PCIe 2.0 is already sufficient.

Our compressed FM-index is constructed with the parameter value $d = 192$. Consequently, the index size becomes 1.06 GB and can be loaded in each DIMM to maximize parallelism of FM-index implementation. Moreover, the Smith-Waterman algorithm supports a read length of $m = 104$ and reference length of $n = 160$. We need a larger segment from the reference to enable gapped alignment at the start and the end of a read.

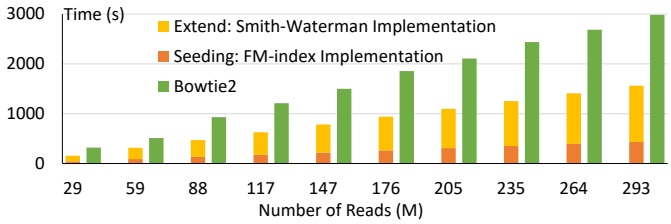


Fig. 3: The alignment runtime of the proposed architecture versus Bowtie2 based on different sizes of the simulated dataset.

Given the relatively small workload (1 % on average of a full alignment workload), the reconfiguration time (10-12 s per configuration) for VU9P does not pose a huge impact on the overall performance. Moreover, since different datasets and alignment parameters are used in different works, we define a normalized metric, base pairs aligned per second per device (bps^{-1}), to allow a fair comparison.

B. Alignment Runtime

We use ART sequencing read simulator [26] to simulate single-end, Illumina-like reads based on the Human Genome Build 38. Parameter qs is set to 5 while others are left as default. We vary the size of the read dataset by changing the parameter f in the simulator. We ran the simulated dataset upon the proposed architecture and also Bowtie2 to record the execution time.

Figure 3 presents the alignment time for different sizes of the read dataset. Also displayed are the results for the software Bowtie2 running on Xeon Silver 4110 using 8 threads and 192 GB RAM with the command:

- Bowtie2 -x grch38_index grch38_art.fq −−local -p 8 -L 10 -R 0 -S out.sam

Compared to the Bowtie2, the proposed architecture can achieve around 2 $\times$ speedup with the Smith-Waterman circuit taking up more than 75 % of the overall computation. For a statically configured circuit, this requires three-time replications for the Smith-Waterman module compared to the FM-index, so as to maintain a balanced pipeline. Given the limited resources available on FPGA, our runtime reconfigurable architecture can provide a more optimized and balanced alignment pipeline.

Note that the runtime of our Smith-Waterman circuit shares similar results as illustrated in [22]. While for a large dataset with around 300M reads the proposed architecture substantially decreases the overall runtime from around 50 min to 26 min, it can still be further improved to achieve better acceleration. For example, instead of successively walking through the candidate locations from each entry in the priority list, Bowtie2 heuristically selects candidate locations to perform extension. This provides better performance because it finds the alignment sooner and avoids unnecessary extensions.

TABLE II: Performance comparison with previous hardware accelerators that are based on similar algorithms. The speed-up is based on the software platform chosen by the authors in each respective work.

Year	Work	Algorithm & Method	Platform	Device	Read Count (Million)	Read Length (Base)	Speed-up	Mbps^{-1}
2013	[4]	FM-index	Maxeler MAX3	Virtex-6 SX475T	18	75	4.2×	20.8
2015	[27]	FM-index	Maxeler MPC-X1000	Stratix-V×8	10	75	14.9×	8.3
2016	[19] [20]	Smith-Waterman	Alpha Data ADM-PCIE-7V3	Virtex-7 XC7VX690T-2	8	150	2×	4.1
2018	[17]	Smith-Waterman	-	Virtex-7 XC7VX485T	-	128	25.2×	-
2019	[28]	SNAP [29] + DP	Alpha-Data ADM-PCIE-7V3	Virtex-7 XC7VX690T	100	128	1.86×	2.5
2019	[22]	Bowtie2 + Smith-Waterman	Maxeler MAX5C	Virtex Ultrascale+ VU9P	60	100	35%	1.03
2020	This work	FM-index + Smith-Waterman	Maxeler MAX5C	Virtex Ultrascale+ VU9P	293	100	1.9×	18.79

Future work includes further analysis on Bowtie2 so as to employ the same heuristic on FPGA for added acceleration.

Finally, Table II presents the performance comparison between the proposed architecture and previous accelerators from recent years. We select the works that are mostly based on the FM-index, or the Smith-Waterman algorithm, or both for fair comparison. Since our performance results are based on the implemented hardware instead of the theoretical upper bound values, all the numbers obtained from previous works are based on this metric. The speed-up on the right is based on the software platform chosen by the authors in each respective work. This number provides an indication of the performance gain against the processor of the time.

The million bps^{-1} values in Table II indicate that our alignment pipeline performs similarly compared to previous accelerators. It showcases the advantages of our architecture where the addition of computation logic for processing quality metrics and N characters does not affect alignment time. However, we could not compare our results to [17] as they did not specify the size of their dataset in their experiments.

C. Alignment Reported and Resource Consumption

As mentioned in Section I, medical applications require strict accuracy constraints. Therefore, we evaluate the alignment pipeline with a sample of the real dataset: accession ERR194147 [23]. We select approximately 8M of reads and obtain the reported alignments as shown in Table III. To maximize the number of alignments reported in Bowtie2, we enable the reseeding mechanism in software by specifying the parameter $R = 2$.

Because of the reseeding mechanism, Bowtie2 can report $3\,\%$ more alignments compared to the proposed architecture. Basically, when a read is considered to have repetitive seeds, i.e. total number of seed hits $\div$ the number of seeds that aligned at least once > 300, Bowtie2 chooses a new set of reads with the same length at different offsets and searches for more alignments. This contributes to more valid alignment discoveries. Future work will incorporate this functionality in the FM-index circuit to improve the overall alignment results.

Finally, Table IV indicates the total resource utilization for each implementation on Xilinx VU9P. With an adequate area remaining for the FM-index circuit, more computational kernels can be populated on the FPGA if the number of DIMMs increases. Currently, the number of DIMMs determines the number of computational kernels in hardware, as each kernel computes and operates on one copy of the FM-index buffered in the onboard DRAM. The Smith-Waterman implementation, however, consumes almost half of the available resources. Further optimization to the hardware implementation is required to decrease the area cost by leveraging an automatic design analyzer and merger [30], so as to include more computational kernels and increase the parallelism.

TABLE III: Overall alignment rate of Bowtie2 and the proposed architecture. The number of reads $= 8240796$.

	Bowtie2	This work
Aligned 0 times (rate)	518646 (6.29%)	749399 (9.09%)
Aligned 1 time (rate)	4018761 (48.77%)	4799046 (58.24%)
Aligned $>$ 1 times (rate)	3703389 (44.94%)	2692351 (32.67%)
Overall alignment rate	93.71%	90.91%

TABLE IV: Resource utilization of different implementations on Xilinx VU9P. Percentage values are relative to the available resources on target FPGA.

	LUT	Register	BRAM	DSP
FM-index	16%	17%	45%	1%
Smith-Waterman	46%	41%	45%	12%

VI. Conclusion

A novel, two-stage reconfigurable architecture is proposed to accelerate the seed-and-extend model, similar to the ones in Bowtie2. With complete consideration of NGS data, the proposed alignment pipeline can achieve a comparable alignment rate and exhibit around 2 times speedup versus Bowtie2. Further research includes optimizing the seeding stage with the heuristic in Bowtie2, decreasing the resource utilization of the Smith-Waterman circuit, and supporting pair-end reads.

Acknowledgment

The support of the Lee Family Scholarship, the Great Britain-China Educational Trust, the Hong Kong Scholarship For Excellence Scheme, the UK EPSRC (EP/L016796/1, EP/N031768/1, EP/P010040/1 and EP/L00058X/1), the National Natural Science Foundation of China (62001165), Maxeler, Intel and Xilinx is gratefully acknowledged.

REFERENCES

[1] G. Lightbody *et al.*, "Review of applications of high-throughput sequencing in personalized medicine: barriers and facilitators of future progress in research and clinical application," *Briefings in Bioinformatics*, vol. 20, no. 5, pp. 1795–1811, 2019.

[2] H.-C. Ng *et al.*, "Reconfigurable Acceleration of Genetic Sequence Alignment: A Survey of Two Decades of Efforts," in *27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–8.

[3] N. B. Tsui *et al.*, "Maternal plasma RNA sequencing for genome-wide transcriptomic profiling and identification of pregnancy-associated transcripts," *Clinical Chemistry*, vol. 60, no. 7, pp. 954–962, 2014.

[4] J. Arram *et al.*, "Reconfigurable Filtered Acceleration of Short Read Alignment," in *2013 International Conference on Field-Programmable Technology (FPT)*, 2013, pp. 438–441.

[5] B. Schmidt and A. Hildebrandt, "Next-generation sequencing: big data meets high performance computing," *Drug Discovery Today*, vol. 22, no. 4, pp. 712–717, 2017.

[6] P. Ferragina and G. Manzini, "An Experimental Study of an Opportunistic Index," in *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2001, pp. 269–278.

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

[8] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with Bowtie 2," *Nature Methods*, vol. 9, pp. 357–359, 2012.

[9] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," *arXiv preprint arXiv:1303.3997v2*, 2013.

[10] J. Arram *et al.*, "Reconfigurable Acceleration of Short Read Mapping," in *IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, 2013, pp. 210–217.

[11] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinformatics*, vol. 25, no. 14, p. 1754–1760, 2009.

[12] B. Langmead *et al.*, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, vol. 10, no. R25, 2009.

[13] M. Burrows and D. Wheeler, "A Block-sorting Lossless Data Compression Algorithm," Digital Equipment Corporation, Tech. Rep., 1994.

[14] S. Salamat and T. Rosing, "FPGA Acceleration of Sequence Alignment: A Survey," *arXiv preprint arXiv:2002.02394v2*, 2020.

[15] E. Fernandez *et al.*, "Multithreaded FPGA acceleration of DNA Sequence Mapping," in *2012 IEEE Conference on High Performance Extreme Computing*, 2012, pp. 1–6.

[16] P. Draghicescu *et al.*, *Inexact Search Acceleration on FPGAs Using the Burrows-Wheeler Transform*. Pico Computing, 2012.

[17] X. Fei *et al.*, "FPGASW: Accelerating Large-Scale Smith-Waterman Sequence Alignment Application with Backtracking on FPGA Linear Systolic Array," *Interdisciplinary Sciences: Computational Life Sciences*, vol. 10, no. 1, pp. 176–188, 2018.

[18] M. C. F. Chang *et al.*, "The SMEM Seeding Acceleration for DNA Sequence Alignment," in *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2016, pp. 32–39.

[19] E. J. Houtgast *et al.*, "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.

[20] ——, "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.

[21] M. Y. Gök *et al.*, "Programmable Hardware Based Short Read Aligner Using Phred Quality Scores," in *2013 International Conference on Social Computing*, 2013, pp. 864–867.

[22] K. Koliogeorgi *et al.*, "Dataflow Acceleration of Smith-Waterman with Traceback for High Throughput Next Generation Sequencing," in *29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 74–80.

[23] M. A. Eberle *et al.*, "A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree," *Genome Research*, vol. 27, no. 1, pp. 157–164, 2017.

[24] B. Ewing *et al.*, "Base-Calling of Automated Sequencer Traces Using Phred. I. Accuracy Assessment," *Genome Research*, vol. 8, no. 3, p. 175–185, 1998.

[25] O. Pell *et al.*, "Maximum Performance Computing with Dataflow Engines," *High-Performance Computing Using FPGAs*, pp. 747–774, 2014.

[26] W. Huang *et al.*, "ART: a next-generation sequencing read simulator," *Bioinformatics*, vol. 28, no. 4, pp. 593–594, 2011.

[27] J. Arram *et al.*, "Ramethy: Reconfigurable Acceleration of Bisulfite Sequence Alignment," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 250–259.

[28] S. S. Banerjee *et al.*, "ASAP: Accelerated Short-Read Alignment on Programmable Hardware," *IEEE Transactions on Computers*, vol. 68, no. 3, p. 331–346, 2019.

[29] M. Zaharia *et al.*, "Faster and More Accurate Sequence Alignment with SNAP," *arXiv preprint arXiv:1111.5572v1*, 2011.

[30] H.-C. Ng *et al.*, "ADAM: Automated Design Analysis and Merging for Speeding up FPGA Development," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2018, pp. 189–198.