Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 721 – 731

doi:10.3906/elk-1703-251

Highly accurate and sensitive short read aligner

Mehmet Yağmur GÖK[1], **Sezer GÖREN UĞURDAĞ**[2,*], **Cem ÜNSALAN**[3], **Mahmut Şamil SAĞIROĞLU**[1]

[1]TÜBİTAK, Kocaeli, Turkey

[2]Department of Computer Engineering, Faculty of Engineering, Yeditepe University, İstanbul, Turkey

[3]Department of Electrical & Electronics Engineering, Faculty of Engineering, Marmara University, İstanbul, Turkey

Received: 20.03.2017 • **Accepted/Published Online:** 27.11.2017 • **Final Version:** 30.03.2018

Abstract: Next-generation sequencing generates large numbers of short reads from DNA. This makes it difficult to process and store. Therefore, efficient sequence alignment and mapping techniques are needed in bioinformatics. Alignment and mapping are the basic steps involved in genetic data analysis. The Smith–Waterman (SW) algorithm, a well-known dynamic programming algorithm, is often used for this purpose. In this work, we propose to utilize Phred quality scores in Gotoh's affine gap model to increase the accuracy and sensitivity of the SW algorithm. Hardware platforms such as FPGAs and GPUs are commonly used to solve computationally expensive problems. In this work, a hybrid PC-FPGA system is built where the SW algorithm based on the affine gap model with Phred quality scores is implemented on the FPGA and a read compressor is implemented on the host PC. We compare our method with state-of-the-art systems such as Bowtie, BWA, and the Kim–Olson FPGA-based system in terms of sensitivity, accuracy, and speed. Based on extensive experiments, we observed that our proposed method is more sensitive and accurate as compared to other solutions.

Key words: Alignment, short read, FPGA, Smith–Waterman, genome, sensitivity, accuracy

1. Introduction

Interest in exploring the genetic data of organisms is growing every day, especially the data of mankind. Genetic data of humans are coded in a structure called DNA, which is a helical structure of two conjugate strands. One strand in a string of about three billion organic molecules named nucleotides. A string of nucleotides stores information about the organism. Although about 99% of DNA data are common for all humans, the complete picture is unique for an individual.

There are different ways of identifying the whole or part of DNA. Next-generation sequencing (NGS) is a popular technique that identifies DNA partially [1]. NGS reduces DNA sequencing costs by increasing throughput and makes a remarkable improvement. Today there are various fields in ecology, agriculture, biology, forensics, and drug studies where NGS has been utilized. NGS randomly fragments many copies of the genome of an organism and produces short DNA fragments with determined nucleotide sequences. These are called short reads. Short reads are usually of length between 20 and 300 base pairs. Short read production by NGS is performed in parallel and in the form of millions of short strings of genomic data.

The bottleneck in NGS data analysis is determining the location where each short read sequence aligns

*Correspondence: sgoren@cse.yeditepe.edu.tr

to the reference genome. Alignment and mapping are usually the first steps in the data analysis pipeline. Unfortunately, these are computationally expensive operations.

Performance of an aligner and mapper algorithm can be evaluated based on three metrics. The first one is sensitivity. Sensitivity is the capacity of the algorithm to map the reads that do not have an identical match in the reference genome due to technical problems or genetic variances. Mapping nonidentical reads is particularly important as the difference may point to a genetic disease. The second metric is accuracy. Accuracy is mapping reads to the best possible locations on the reference string. The third metric is the elapsed time to match all reads considering the amount of data to be mapped, which may be millions of short reads. Note that power consumption is also proportional to the elapsed time in the alignment and mapping processes.

The Smith–Waterman (SW) algorithm [2] is a well-known dynamic programming algorithm proposed for aligning and mapping reads to a reference genome. An improved SW algorithm based on an affine gap model was introduced by Gotoh [3]. In this work, our first contribution is combining Phred quality scores in Gotoh's affine gap model to improve the SW algorithm's accuracy and sensitivity. Our second contribution is the hardware implementation of the enhanced SW algorithm with Phred quality scores. To do so, we propose a hybrid PC-FPGA-based read aligner and compressor system. The proposed system aligns NGS short reads to the reference genome utilizing Phred quality scores to obtain better mapping accuracy and sensitivity.

2. Related work

There are two main short read aligner approaches in the literature. The first approach is based on the Burrows–Wheeler transform (BWT) [4]. This approach efficiently stores information in order to traverse a position tree of a given reference sequence [5] via the FM index. This approach can easily match some positions in a reference sequence with a smaller number of differences. BWT-based methods are less sensitive as compared to other approaches. Bowtie [6] and BWA [7] are based on BWT. The second approach enforces the fact that individual genomes are like each other with minor differences. This means that smaller subsequences, known as seeds, of a read map the reference sequence. Therefore, the first compilation of an index of the reference sequence maps each seed to the position where it occurs. All seeds in the read are searched in CALs [2] for aligning a short read. Then the short read is checked against the reference sequence at every CAL using the SW algorithm. The highest score location is selected as the alignment location for the corresponding short read. BFAST [8] is an example of this approach.

In the literature, there are FPGA- and GPU-based solutions [9–11] to speed up read alignment. FPGAs have become popular for their energy-efficiency and flexibility in hardware customization. Hence, in this paper, we also offer an FPGA-based aligner. There are two main approaches in FPGA-based aligners. The first one [9] is streaming the reference genome through a mapping system, which may demonstrate better sensitivity to genetic variations in short reads compared to software competitors such as Bowtie [6] and MAQ [9]. In this scheme, mapping time is as same as that of Bowtie. However, the scheme is dependent on the length of the reference genome and available hardware resources. The second approach [10] relies on finding CALs on the reference for short reads to map. Then parts of the reference genome at CALs are fed to a mapping system. Indexing is performed to identify locations on the reference string where reads are most likely to match. Once these locations are determined, a dynamic programming algorithm runs on them. Both approaches are based on the SW algorithm. Systolic arrays are often used to implement the SW algorithm in hardware. Processing elements (PEs) are connected in series [12] to construct a systolic array.

3. Proposed system

In this work, we propose a hybrid PC-FPGA-based read aligner and compressor system. This system is composed of a host PC, which is responsible for the hosting of reads, passing them to the SW aligner on the FPGA, and compressing reads that are mapped to the reference genome. The SW aligner on the FPGA maps the reads to the reference sequence by incorporating the quality scores of the reads. This provides better mapping performance by means of offering better precision in mapping positions. The proposed read aligner and compressor system architecture is given in Figure 1.

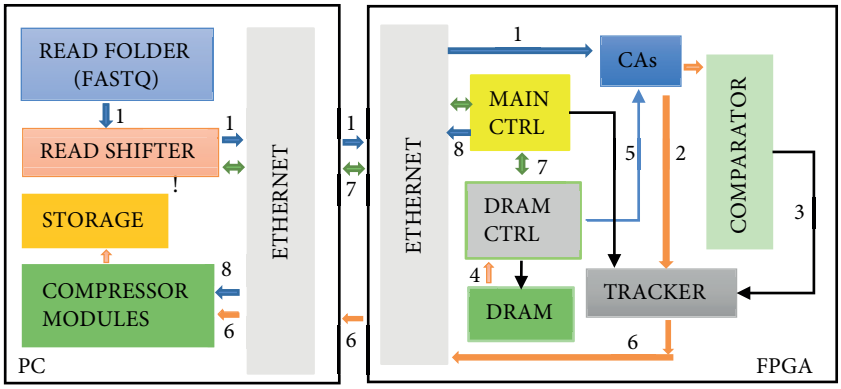

Figure 1. Proposed short read aligner.

The FPGA subsystem consists of computation arrays (CAs); a main controller for signaling data flow over the system; a DRAM controller to access DRAM, which holds the reference genome; and an Ethernet IP core to communicate with the host PC. There is a data bus to load the reference genome to the on-chip memory of the FPGA (BRAM) sequentially, a broadcast bus to drive reads to CAs, and a result bus to deliver results from CAs to the PC via an Ethernet interface. The main controller is responsible for fetching reads from the PC and sending them to CAs. These run the SW algorithm on the reads of a hundred base pairs, fill similarity matrices, and obtain alignment scores. The results are then compared with a predetermined threshold to eliminate reads with very low mapping scores. Finally, mapping positions and scores are passed back to the PC together with the read IDs and an indicator showing whether the read passes the threshold or not.

CAs form the similarity matrix of the SW algorithm shown in Figure 2. CAs are read-length long systolic arrays of PEs as shown in Figure 3. Gotoh [3] improved the level of the similarities between two sequences by means of an affine gap model. In Gotoh's model, a gap denotes either an insertion or a deletion. The gap can be a multiple of null characters. Gap-open is the penalty score of the first gap in the affine gap model. The rest of the subsequent gaps are called the gap-extend. The penalty for gap-extend is lower than the penalty for gap-open. The affine gap model is given as follows.

$$H(i,0) = 0,\ 0 \le i \le n \tag{1}$$

$$H(0,j) = 0,\ 0 \le j \le m \tag{2}$$

$$H(i,j) = \max\{H(i-1,j-1) + \sigma(S[i],T[j])\, E(i,j)\, F(i,j) \tag{3}$$

$$E(i,j) = \max\{H(i,j-1) - \alpha,\ E(i,j-1) - \beta\},\ 1 \le i \le n; 1 \le j \le m \tag{4}$$

$$F(i,j) = \max\{H(i-1,j) - \alpha, F(i-1,j) - \beta\},\ 1 \le i \le n; 1 \le j \le m \tag{5}$$

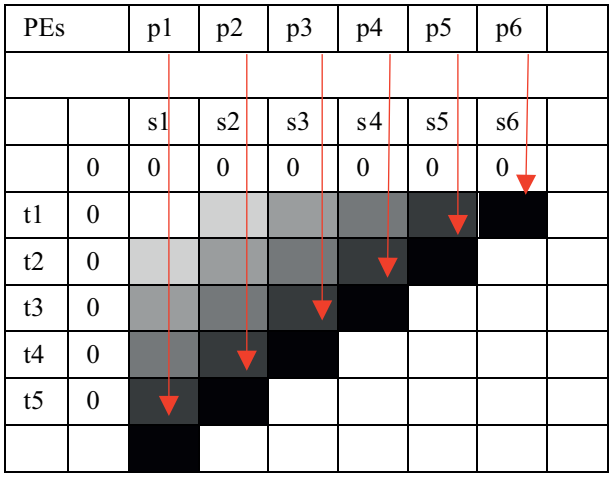

Figure 2. Mapping similarity matrix to PE array.

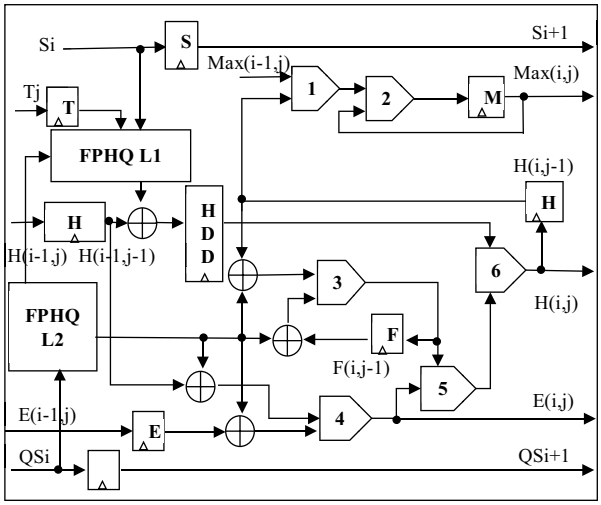

Figure 3. Proposed PE.

Note that $H(ij)$ is the similarity of the two sequences S and T ending at positions i and j. $E(ij)$ and $F(ij)$ represent the alignment between a character in one string and a gap in the other string, respectively. In Eqs. (3)–(5), σ stands for reward/penalty comparator, α denotes the gap opening, and β denotes gap extension. The CA (PE array) is mapped to the diagonal going from the lower left corner to the upper right corner (antidiagonal) of the similarity matrix shown in Figure 2, where every entry of the antidiagonal is computed once.

Quality scores were first introduced in the program Phred [1]. Phred quality scores show how reliable the base read from the sequencing technique is by characterizing the quality of DNA sequences. Due to this, Phred quality scores are widely accepted in order to compare different sequencing techniques. In this work, we exploit CAs, which incorporate Phred quality scores during the calculation of the similarity matrix. This scheme, which was proposed in our earlier work [13,14], provides a better alignment. In the proposed system, we do not directly use the quality score itself but rather a function of it. The quality score range, lying between 33 and 83 (represented in six bits), is divided into five levels. The function $FPhQ()$ simply quantizes the quality score value per those levels producing three bit values. Therefore, the Phred quality score-based implementation of the affine gap model given in Eqs. (1)–(5) can be reformulated as follows.

$$H(i,0) = 0,\ 0 \le i \le n \tag{6}$$

$$H(0,j) = 0,\ 0 \le j \le m \tag{7}$$

$$H(i,j) = \max\{H(i-1,j-1) + FPhQ(S[i])\,\sigma(S[i],T[j])\,E(i,j)\,F(i,j) \tag{8}$$

$$E(i,j) = \max\{H(i,j-1) - \alpha FPhQ(S[i]),\ E(i,j-1) - \beta FPhQ(S[i])\},\quad 1 \le i \le n; 1 \le j \le m \tag{9}$$

$$F(i,j) = \max\{H(i-1,j) - \alpha FPhQ(S[i]),\ F(i-1,j) - \beta FPhQ(S[i])\},\quad 1 \le i \le n; 1 \le j \le m \tag{10}$$

In the Phred quality score-based affine gap-model, the expression $FPhQ$ represents the quality score of the read. The architecture of our proposed SW approach with the new PEs based on quality scores was proposed in our earlier work [14], as shown in Figure 3. In Figure 2, elements of the same color in the similarity matrix on

the same antidiagonal are computed in parallel. Each entry in the similarity matrix is dependent on the upper left, left, and upper neighbor cell scores. Each base of the short read is scored against a base of the reference simultaneously with the help of CA. This improves the runtime of SW for an m-length reference and n-length short read from $O(mn)$ to $O(m+n)$.

The CA keeps track of the maximum score to find the best position and score of sequences. It also holds the maximum score and position until the entire reference has shifted through the CA. Once the end of reference sequence arrives, the best score and location of that alignment is sent to output. This is through an output FIFO, which runs in a round-robin fashion. Here we did not use a multiple-output FIFO because of its area overhead.

The FPGA subsystem forces every read to map to a location in the reference string. This means that even reads that are produced as a result of sequencing error map to a location on the reference string. Also, reads may map to locations with a very low mapping score due to low Phred quality score or contamination. Besides, a mapped read that differs much from the reference that it is mapped to can be considered as insignificant. Therefore, we put that threshold as 15% of the read length, which is 15 base pairs in our system.

Regarding the above cases, these nonuseful mappings should be eliminated and marked as unmapped reads. The comparator module serves for this purpose. It simply compares the maximum similarity score of the read to a predetermined threshold, which is a function of the 15% threshold and acceptable Phred quality score. Reads with mapping scores below this threshold are marked as unmapped. Besides, the comparator in parallel compares mapping scores to another predetermined value. This second value is the all-mapping threshold, meaning a perfect match. Mappings passing the second comparison are marked as perfect matches. Therefore, for every read a mapped/unmapped and perfect/imperfect flag is set. Then they are reported to the host PC.

The main controller is responsible for the flow control of read sequences from the host PC and the streaming of results to the PC. It also signals the reference string that streams from DRAM. The main controller handles multiple CAs and FIFOs to CAs. It signals the read shifter module to pass read data to CAs and controls result data flow to the PC. Multiple CAs are loaded with read sequences in a round-robin fashion with a counter. Short read sequences are read from PC memory serially. As the proposed system relies on shifting the entire reference over short read sequences (which takes at least three billion clock cycles), the cost of loading reads to the CAs in 100 clock cycles is negligible. Moreover, it eliminates the need to implement a read serializer in hardware. Only the read FIFO is implemented in hardware, which is controlled by the controller block. The main controller hosts two FIFOs. The output FIFO collects the mapping score, mapping position, read ID from CAs, and the comparator module. It then streams these to the PC. Finally, the reference FIFO is used to stream the reference data over CAs.

Reference length of about three billion base pairs (each needing at least two bits) requires at least 715.26 MB of storage area. This is not achievable in the FPGA's block RAM. Therefore, an onboard DRAM is used to keep the necessary reference sequence string. The reference is stored in DRAM in 256 bit boundaries. Our FPGA board has DDR3 SODIMMs, which return 256 bits of data per read burst.

CAs receive the stream of reference sequence base pairs from the DRAM. A DRAM interface block is necessary to convert the DRAM output into a stream of two bit values. A DRAM controller typically has a wide data bus, but the CA requires a stream of two-bit reference base pair values. Therefore, the DRAM controller block in our system acts as an interface to the DRAM as well as a serializer. The DRAM interface serializes parallel reference string data from the DRAM and feeds to CA blocks in parallel. Since the streaming of the reference sequence to CAs is a critical operation, the RAM interface and serializer use a double-buffering

scheme to ensure a steady stream of data. Once a word is read from the DRAM, it is immediately written to the first port of a dual-port BRAM.

A clock divider is used to produce a pulse, which in turn feeds a counter that cycles through all possible addresses on the second BRAM port. The next word from the DRAM is requested and written to the dual-port BRAM as soon as possible.

This scheme operates as follows. Initially, the PE is set to zero to represent Eqs. (6) and (7). S and T are the D flip-flops (DFFs) to register the string sequence elements $S[i]$ and $T[j]$ to be aligned and shifted to the next PE at every clock cycle. Input of DFF HD comes from the previous PE as $H(i-1,j)$. It is delayed for one cycle before in order to be the value of the upper left neighbor, $H(i-1,j-1)$. The sixth MAX block selects the maximum of $H(i-1,j-1)$ and the output of the fifth MAX block, which is the maximum of F and E in Eq. (8). DFF F input represents the value from the top neighbor element and is used to hold $F(ij)$ as $F(i-1,j)$. These are compared to select the maximum of gap-extend or gap-open by the third MAX block as in Eq. (10). Similarly, DFF E input is the output of the former PE and it is used to latch $E(ij)$. It represents the value of the left neighbor element. Its gap-open and gap-extend score comparison is done by the fourth MAX block as in Eq. (9). Look-up table (LUT) FPhQ L1 is the LUT implementation of the quality score including comparison of $S[i]$ and $T[j]$ for reward/penalty. FPhQ L2 is the LUT implementation of the quality score of $S[i]$.

The PC hosts the read depot, read shifter, compressor, and storage modules. Read depot and storage modules are simply hard disk sectors. The read depot is the storage area for short read sequences, represented in the FASTQ file format, from the NGS sequencer. The reads shifter block is responsible for shifting reads to the FPGA using the Ethernet bus. The read shifter module fetches the indexed reads from the read depot. The read shifter expects "ready for load" signals from the FPGA controller module. It then sends the reads to the CAs of the FPGA module under supervision of the controller module hosted by the FPGA system.

4. Results

Performance of the system is evaluated with different aspects. We first examine the resource utilization of the system. A very important aspect is the correctness of the system, i.e. to see alignment algorithm on FPGA matches with the same algorithm running on the CPU. We then present the sensitivity and accuracy of the system. Alignment time is another aspect that is evaluated. The proposed system is compared to the state-of-the art aligners. While doing this, our system is configured to use 100 base pair short read sequences. Data used to evaluate the system are obtained from the Illumina sequencer operating at TÜBİTAK BİLGEM, specifically run by our study group.

The proposed short read aligner system is built on a Xilinx Virtex4 XC4VLX80 FPGA board with DDR3 SODIMMs. The Xilinx ISE 11.1 tool chain is used for system implementation and source evaluation. The FPGA platform operates in cooperation with a PC with two quadcore Intel CPUs (with a clock speed of 3.2 GHz and 16 GB memory). The host PC communicates with the FPGA board via the Ethernet. Logic utilization of the entire design is summarized in Table 1.

In the Virtex4 XC4VLX80, three CAs of 100 PEs can be implemented at most. This means that three short read sequences are mapped simultaneously to the reference genome. CAs run at 220 MHz with data path delay of 7.673 ns.

To test the correctness of the proposed system, we followed the same method defined in [15]. To do so, we loaded a *Drosophila melanogaster* reference sequence of one hundred million base pairs on the DRAM of the

Table 1. Resource utilization

Module	Instance count	LUT count	LUT util. (%)
Ethernet	1	1476	2.07
Computation array	3	93,966	90.00
Comparator	1	284	0.40
DRAM controller	1	376	0.53
Main controller	1	932	1.30
Total design	-	65,442	94.93

FPGA board. The FPGA was then configured to align 31 base pair length reads. We carried out alignment against a thousand different short read sequences of length 31. Comparing the results with a conventional CPU implementation written in C++, we observed that the FPGA and CPU results exactly matched.

Sensitivity comparison of the proposed system was conducted with the BOWTIE-2.0-beta5 CPU-based aligner and the Kim–Olson's FPGA-based solution [10]. Bowtie is configured and tested using eight threads. We applied such a strategy since we also compared the proposed system with Bowtie in terms of alignment time, for which an eight thread scheme leads to saturation of the memory bandwidth of the system, giving optimum mapping performance for mapping. For the comparison, 2 million short read sequences of 100 base pairs obtained from the MASON [16] read simulator were used. The reads contain SNPs and indels, as well as sequencing errors distributed per typical nonuniform error profiles of the Illumina sequencer.

The proposed system is configured to operate 100 CAs. At the end of the experiment, we observed that our proposed system mapped 94.3% of 1 million reads. Bowtie managed to align 87.6% using the best option mode. Regarding the comparison with the Kim–Olson system [10], the PC part is obtained from the authors and the FPGA part is implemented in VHDL and targeted for the same FPGA. On the same dataset, the mapping obtained with the Kim–Olson system is 91.7%. This is still lower than the sensitivity value obtained by our system. Figure 4 summarizes the sensitivity results. The main reason for sensitivity is clearly the search of all possibilities both on the streaming reference and SW score matrix computations with quality scores. Both methods first search for candidate locations and then run the SW algorithm on those candidate areas. The Kim–Olson method does not consider base quality scores, but mainly focuses on reducing the total alignment time.

To evaluate the accuracy or mapping quality, we conducted a two-step experiment. In the first step,

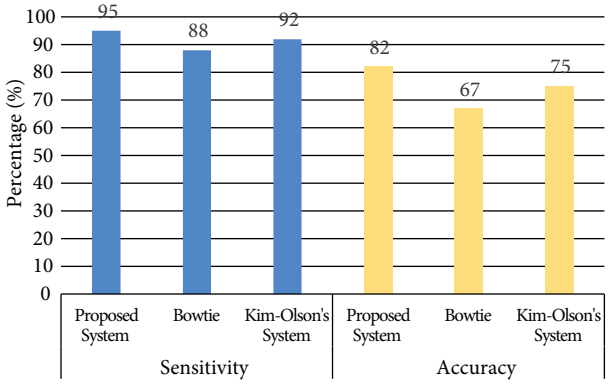

Figure 4. Sensitivity and accuracy.

we created 100 simulated reads as described in [14] and followed the same method. The test configuration was for reads of 100 base pairs in length. We tested the alignment using Bowtie, the Kim–Olson system, and our system. Our system could map 83 of the reads to correct positions. This dropped down to 72 with the Kim–Olson system. Bowtie could map only 59 reads. These results are also presented in Figure 4. The second step in testing accuracy is accomplished by using several read data, which is more concordant to the nature of the process. Here, 2 million Illumina single-end reads of length of 100 base pairs from chromosome 20 belonging to the Hg19 human genome are simulated using MASON [16] with a haplotype SNP rate of 0.1%. The choice of MUTATRIX (https://github.com/ekg/mutatrix) is made because it features position-specific error rates and base quality values. The human genome is aligned to the reads aligned using BOWTIE-2.0-beta5, the BWA-0.5.9 aligner, and our proposed system. Then we calculated the positive predictive value (PPV = TP/(TP + FP)) of each aligner. The location of a correctly aligned read should justify the output of the MASON [16] simulator. This method is like the one followed by another aligner named MOSAIK [15]. There, a correct alignment is considered as the one staying in a tolerant window of 20 base pairs. Figure 5 shows the PPV of the tested aligners. The mapping quality cutoff is the lower bound for the Phred quality score for the reads used in a particular run. For example, with a cutoff value of 20, short reads of mapping quality value 30 and over were used in [17]. The reason for using simulated data is that it is possible to estimate both discarded reads and wrongly aligned reads of alignment. With real data, in a shotgun sample, it is not possible to know where the reads come from. Therefore, it is not practical to calculate wrongly aligned reads as we do not know what the correct alignment is. Therefore, it is impossible to assess read alignment accuracy on real data. Hence, it is not appropriate to put effort into measuring the accuracy of the alignment via real sequencing data.

The reliable detection of genomic variation for variants larger than a few base pairs is a challenge. Though they are difficult to align, read sequences crossing boundaries of structural variation have potential for their identification. The proposed system solely relies on the SW algorithm, which is the appropriate solution to align gapped sequences with short indels relying on seeking all possible frames of alignments with all possible gaps. Effects of indels on the alignment sensitivity and a comparison with other aligners are tested on simulated Illumina single-end reads of 100 bp in length containing indel events of 1–10 bp. The simulation software was the MUTATRIX [17] genome simulator. This was again selected to keep the comparison like that in [15]. About 100 events with approximately 1000 spanning reads are simulated for each indel length. Sensitivity is expressed as the ratio of correct alignments to total reads versus different indel lengths. In Figure 6, correct alignment is mapping to the correct position and matching variant within the alignment. The proposed system expresses better sensitivity compared to Bowtie and BWA in terms of deletion handling. In terms of insertion handling, the performance of the proposed system is between that of Bowtie and BWA. Figure 7 presents the sensitivity of the proposed system with the correct alignment criterion, which is set to mapping the position of the read correctly.

As another conducted test, we simulated 1000 single nucleotide polymorphisms (SNPs) on the human genome chromosome 20 using MUTATRIX [17]. Next, MASON [16] was used to generate one million 100 bp reads from the mutated chromosome carrying SNPs. BWA, Bowtie, and the proposed system aligned them back to the human reference genome. After alignment, SNP calling was performed via GATK. Figure 5 shows the variant callers' sensitivity to SNPs versus the false discovery rate (FDR) taking SNP calls with variant quality scores of a minimum value. Sensitivity of the SNP caller to the data aligned by different aligners is given in Figure 8. As can be seen from Figure 5, the SNP call performance of the proposed system is better than that of the other two aligners.

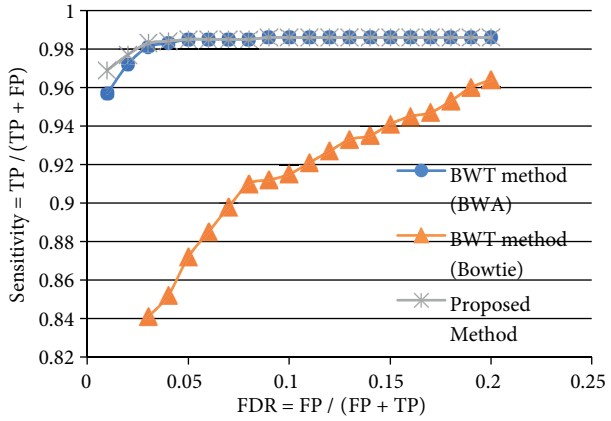

Figure 5. PPV vs. mapping quality cutoff. TP: True positive, FP: false positive.

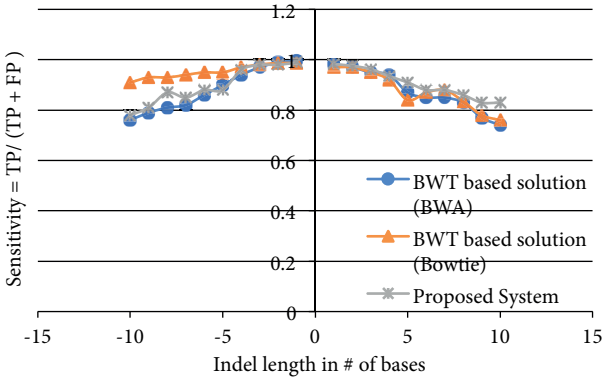

Figure 6. Sensitivity of simulated reads spanning indels with variant information.

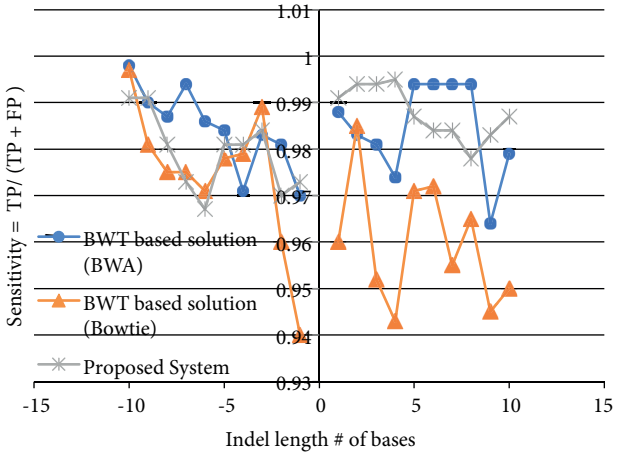

Figure 7. Sensitivity of simulated reads spanning indels without variant information.

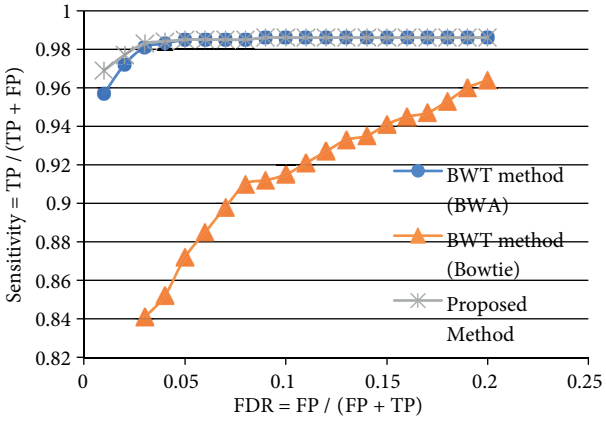

Figure 8. SNP calling sensitivity vs. false discovery rate.

In order to evaluate the proposed system's alignment speed, the proposed alignment method is coded in the C programming language to run on Intel 3.2 GHz Quadcore CPU with 16 GB RAM. Simulated short read sequences from datasets with various read lengths are aligned to the chromosome 20 of the human Hg19 reference genome. Each dataset contains 10,000 short read sequences simulated from the chromosome 20 with variants including SNPs and indel events. Alignments are performed with a single core on the CPU and a single CA on the FPGA of the proposed system. Alignment speed performance comparison in terms of the total execution time of alignment is presented in Table 2. The dataset is given over the total number of base pairs. As read lengths increase, there will be fewer reads, which is the situation for sequencers outputting longer reads (10K–50K). Our system's compute time for a read does not change with the read length. However, it directly depends on the reference length. Hence, our proposed system will make more sense with new sequencers outputting longer reads.

As can be seen from Table 2, the hardware implementation of our algorithm runs much faster than its CPU implementation. In addition, as reads get longer, the performance gap between the FPGA and CPU increases. This is a very beneficial property as NGS machines producing short reads of a few kilobytes of bases

Table 2. Speed evaluation of proposed algorithm on CPU vs. FPGA.

Read length	CPU runtime / FPGA runtime
26	1.312
50	2.402
76	3.392
100	4.618
300	14.528

are already on the market and read lengths are continuously increasing. However, for short read lengths of a hundred base pairs, the proposed system is clearly slower compared to the other two systems tested. The main reason for this is to achieve higher sensitivity and accuracy in the mapping, where the whole reference is scanned and searched for the best match. Speed comparison with the Kim–Olson method and Bowtie over total alignment time is given in Figure 9. Our solution offers higher sensitivity and accuracy with a penalty in alignment time as compared to the state-of-the-art. However, it must also be noted that as reads get longer the gap in alignment time between our solution and the state-of-the-art drops.

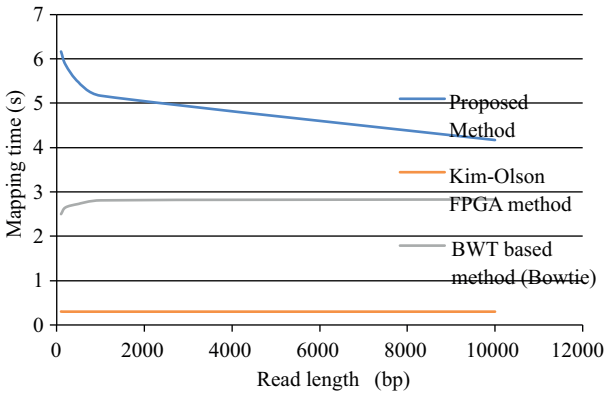

Figure 9. Alignment time vs. read length comparison.

5. Conclusion

In this work, we propose a hybrid PC-FPGA-based genome aligner and short read compressor system. This system is composed of a short read storage and read shifter units hosted on a PC, whereas the computation and control units are implemented on an FPGA platform. CAs in the proposed system are systolic arrays emulating our proposed SW alignment algorithm, which incorporates Phred quality scores of the read sequences. This approach has better sensitivity compared to other systems. It also gives a better mapping performance, which we call accuracy, in terms of approaching the best possible solution over the standard affine gap model. As reads get longer, like those of Oxford Nanopor or Pacific Biosciences PackBio, reads will offer more sequencing errors, such as 10%–15%. To accurately align those reads that are more than 100K long, the proposed system becomes a valuable tool.

References

[1] Ewing B, Hillier H, Wendl M, Green P. Base-calling of automated sequencer traces using Phred. I. Accuracy assessment. Genome Res 1998; 8: 175-185.

[2] Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol 1981; 147: 195-197.

[3] Gotoh O. An improved algorithm for matching biological sequences. J Mol Biol 1982; 162: 705-708.

[4] Burrows M, Wheeler DJ. A Block-Sorting Lossless Data Compression Algorithm. Palo Alto, CA, USA: Digital Equipment Corporation, 1994.

[5] Ferragina P, Manzini G. Opportunistic data structures with applications. In: Annual Symposium on Foundations of Computer Science; 12–14 November 2000; Washington, DC, USA. New York, NY, USA: IEEE. p. 309.

[6] Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 2009; 10: R25.

[7] Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 2009; 24: 1754-1760.

[8] Homer N, Merriman B, Nelson SF. BFAST: An alignment tool for large scale genome resequencing. PLoS One 2009; 4: e7767.

[9] Knodel O, Preusser TB, Spallek RG. Next generation massively parallel short-read mapping on FPGAs. In: International Conference on Application Specific Systems, Architectures and Processors; 11–14 September 2011; Santa Monica, CA, USA. New York, NY, USA: IEEE. pp. 195-201.

[10] Kim M, Olson CB. Hardware acceleration of short read mapping. In: International Symposium on Field-Programmable Custom Computing Machines; 29 April–1 May 2012; Ontario, Canada. New York, NY, USA: IEEE. pp. 161-168.

[11] Yongchao L, Bertil S, Douglas LM. CUSHAW: A CUDA compatible short read aligner to large genomes based on the Burrows–Wheeler transform. Bioinformatics 2012; 28: 1830-1837.

[12] Hoang DT. A Systolic Array for the Sequence Alignment Problem. Technical Report CS-92-22. Providence, RI, USA: Brown University, 1992.

[13] Gok M, Yilmaz C. Efficient cell designs for systolic Smith Waterman implementation. In: International Conference on Field Programmable Logic and Applications; 28–30 August 2006; Madrid, Spain. New York, NY, USA: IEEE. pp. 889-892.

[14] Gok M, Unsalan C, Gören S, Sagiroglu MS. Programmable hardware based short read aligner using Phred quality scores. In: International Conference on Social Computing, 8–14 September 2013; Washington, DC, USA. New York, NY, USA: IEEE. pp. 864-867.

[15] Lee WP, Stromberg MP, Ward A, Stewart C, Garrison EP, Marth GT. MOSAIK: A hash-based algorithm for accurate next generation sequencing short-read mapping. PLoS One 2014; 9: e90581.

[16] Holtgrewe M. Mason – A Read Simulator for Second Generation Sequencing Data. Technical Report. Berlin, German: Freie Universität Berlin, 2010.

[17] Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res 2008; 18: 1851-1858.