FPGA Implementation of a Short Read Mapping Accelerator

Mostafa Morshedi and Hamid Noori(✉)

Faculty of Engineering, Electrical Engineering Department,
Ferdowsi University of Mashhad, Mashhad, Iran
morstafa@yahoo.com, hnoori@um.ac.ir

Abstract. Recently, due to drastically reducing costs of sequencing a human DNA molecule, the demands for next generation DNA sequencing (NGS) has increased significantly. DNA sequencers deliver millions of small fragments (short reads) from random positions of a very large DNA stream. To align these short-reads such that the original DNA sequence is determined, various software tools called short read mappers, such as Burrows BWA, are available. Analyzing the massive quantities of sequenced data produced using these software tools, requires a very long run-time on general-purpose computing systems due to a great computational power it needs. This work proposes some methods to accelerate short read alignment being prototyped on an FPGA. We use a seed and compare architecture based on FM-index method. Also pre-calculated data are used for more performance improvement. A multi-core accelerator based on the proposed methods is implemented on a Xilinx Virtex-6. Our design performs alignment of short reads with length of 75 and up to two mismatches. The proposed parallel architecture performs the short-read mapping up to 41 and 19 times faster than parallel programmed BWA run on eight-core AMD FX9590 and 6-cores Intel Extreme Core i7-5820 k CPUs using 8 and 12 threads.

1 Introduction

Recently, processing massive data generated by NGS (Next Generation Sequencing) methods [1] has become the main bottleneck in genetic researches. Based on the moore's law [2] the available data that needs to be processed in genetic researches, massively exceeds the computational power of the modern processors.

Using the NGS methods, millions of small DNA fragments of length 20 to 100 base pairs (bp) named *short read*, are generated in each run. In short read mapping, short reads have to be aligned according to a larger DNA stream, named *reference genome*. Older alignment approaches such as Smith-Waterman (SW) [3] and BLAST [4] are not suitable for short read mapping due to searching the whole reference for each short read. Recent methods like BWT [5] and soap3 [6], make short read mapping a lot more faster compared to the previous approaches, by generating an index from the reference genome before alignment. These approaches mainly use two major methods including FM-index [7] and Hash-table [8]. Between these two methods, FM-index is more popular due to lower memory footprint and being independent of the length of reference genome during the search operation.

DOI: 10.1007/978-3-319-56258-2_25

Despite all the progress and improvements, due to massive amount of data that need to be processed, still short read mapping is a time consuming process on modern computers. To solve this problem many recent works try to accelerate short read mapping on other platforms like FPGAs due to the high parallelism and customization they provide. Researches such as [8–13] accelerate short read mapping using FPGAs. In this work an FPGA-based fully pipelined accelerator for short read mapping is proposed. The proposed hardware supports up to two mismatches in short read with 75 base-pairs (up to 100 bp). Our design uses the FM-index and the seed and compare methods. The main concepts of our design are:

- Pre-calculated data along using one memory controller for top and bot pointers.
- Extracting three identical and non overlapping seeds from each short read in the inexact match unit and comparing them with the reference.
- Through smart implementation, searching one of the three extracted seeds from each short read is done in the exact match unit.
- A multi-core system is presented to maximize the efficiency of the design.

2 Related Works

In the following subsection we briefly discuss the FM-index approach and after that review some recent short read mapping accelerators.

2.1 FM-index

To use FM-index method, the borrows-wheeler transform (BWT) [14] has to be generated from the reference genome (Fig. 1a). The suffix array (SA) values show the position of each suffix in the original reference stream (Fig. 1b). Using BWT stream, the occurrence array O(x,i) and the characters count C(x) are generated from the BWT (Fig. 1c). Then, searching any short read in the reference genome is done using Eqs. 1 and 2 with *n* steps, where *n* is the length of the short read.

The search operation uses two pointers named top and bot (bottom). These pointers needs to be updated *n* times. To find the location of a short read in the reference genome, top pointer is used as the address to read the SA values (Fig. 1d is an example of searching GA in ACTGA). This is very important to note that finding SA values using the top and bot values is not done in the FPGA accelerators and it is assumed that this step is done in software. Also to reduce the memory size required to store the O(x, i), the rows of O(x,i) are sampled with a factor of (d) and the rest of values (d−1 values) are calculated online using the sampled values and the BWT.

$$top_{new} = O(x, top_{old}) + C(x) \tag{1}$$

$$bot_{new} = O(x, bot_{old}) + C(x) \tag{2}$$

(a)

Reference Genome: ACTGA	
Rotations:	Sorted Rotations:
ACTGA$	$ACTG**A**
CTGA$A	A$ACT**G**
TGA$AC	ACTGA**$**
GA$ACT	CTGA$**A**
A$ACTG	GA$AC**T**
$ACTGA	TGA$A**C**
Borrows Wheeler transform: AG$ATC	

(b)

Reference Genome: ACTGA		
i	Suffixes:	SA
0	$	5
1	A$	4
2	ACTGA$	0
3	CTGA$	1
4	GA$	3
5	TGA$	2

(c)

Occurrence Array: O(x,i)					
i	BWT	A	C	G	T
0	A	0	0	0	0
1	G	1	0	0	0
2	$	1	0	1	0
3	A	1	0	1	0
4	T	2	0	1	0
5	C	2	0	1	1
6	total	2	1	1	1

Character Count: C(x)			
A	C	G	T
1	3	4	5

(d) Short Read = GA Initial values: top = 0, Bot = 6		

1st step (A)
Top = 0 + 1 = 1
Bot = 2 + 1 = 3
	2nd step (G)
	Top = 0 + 4 = 4
	Bot = 1 + 4 = 5
		Result
		i = 4
		SA = 3

Fig. 1. An example of generating the BWT, SA values, O(x,i) and C(x) from a reference genome and finding GA in the reference.

2.2 Recent FPGA Accelerators

While searching a short read in the reference genome (A = 00, C = 01, G = 10, T = 00) two cases can happen including: (1) exact match and (2) inexact match. Also it is known that more than 70% of the short reads can be exactly matched to the reference genome [12]. Among the FPGA implementations using FM-index, [9] is the first implementation which only supports exact matching. Actually, the FM-index method can only support exact matching which is the drawback of this method.

To support inexact matching with FM-index, software tools such as BWA [5] and FPGA implementations such as [11–13] mainly use the backtrack version of FM-index to support mismatches. Another method to support mismatches is the seed and extend method. The original seed and extend method was presented by [4] and its combination with FM-index was implemented on FPGA by [10]. Also, there is another version of BWA [5] which uses this method. In [10] smaller streams named seeds are extracted from each short read. These seeds are searched in the reference genome using FM-index and the SA values extracted from the seeds present the candidate locations. In the final step the short read is compared to the reference genome (in the candidate location) using SW algorithm and the results are streamed to output. In [12] a seed and compare module is used which directly compares some short reads to the reference genome.

3 Proposed Architecture

In this section the proposed architecture to implement short read mapping on FPGA is discussed in details. The fully pipelined design consists of two main modules: the exact match unit and the inexact match unit. Short reads enter the exact match unit and the short reads that cannot be aligned in the exact match unit, are transferred to the inexact match unit. The proposed inexact match unit does not use backtrack version of FM-index. Instead, it extracts seeds from each short read and searches them in the reference genome using simple FM-index. This section consists of four major sub-sections: (1) Exact match unit architecture. (2) Pre-calculated values. (3) Inexact match unit architecture. (4) Multi-core implementation of the design.

3.1 Exact Match Unit

To begin short read mapping operation, short reads are streamed to the exact match unit and searched in the reference genome using FM-index (Fig. 2a). The current top and bot for a short read are used as addresses to read O(x,i) values. The sampled O(x,i) is stored in BRAMs with d = 64 and C(x) values are stored in FPGA registers.

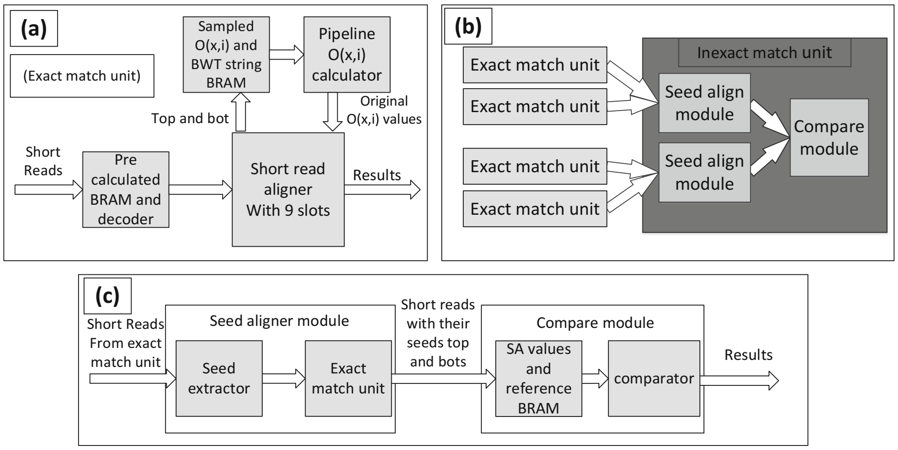

Fig. 2. Top view of exact match unit, inexact match unit and the quad-core design

Our design needs nine clock cycles to update a single top and bot (due to memory latency, generating O(x,i) values and adding O(x,i) to C(x)). To hide the nine clock cycles latency [10], we search nine short reads concurrently in the exact match unit. While other short reads are waiting for new data (top_{new} & bot_{new}), new short reads generate and send their requests for their corresponding tops and bots. As a result, searching any short read with the length of n can be done in n clock cycles in average.

Another important consideration is the memory interface. Basically, to implement Eqs. 1 and 2, two connections to two separate memories are needed, one for top (Eq. 1) and one for bot (Eq. 2), respectively. If only one memory is used for both top and bot, two memory accesses are needed to read $O(x, top_{old})$ (Eq. 1) and $O(x, bot_{old})$ (Eq. 2) values (and their BWTs). Therefore, the required memory size is decreased to the half but the delay for reading O(x,i) becomes doubled and the speedup decreases by half. In FM-index, top and bot have the maximum distance at the beginning. The distance decreases in each search step. Hence, in many cases the top and bot will hit at the same sampled O(x,i) and the original O(x,i) can be calculated for both top and bot, in one clock cycle by reading one memory address.

Our design uses one memory for both top and bot instead of two to reduce the number of memory interfaces. Through doing experiments for 10 K short reads we learn that after first seven steps (in average) the top and bot can be calculated using the same sampled O(x,i) which requires one memory access (around 13 steps when the reference is the whole human genome [12]). With this method (using one memory for top and bot instead of two), the number of memory controllers is reduced to one for each exact match unit. However, the speedup for each exact match unit with one memory controller decreases at most by 0.15x for short reads with the length of 75 compared to the exact match unit with two memory controllers.

3.2 Pre-calculated Data

Every DNA string, similar to short reads of length n, has 4^n different combinations. Therefore, if the top and bot for all combinations of length m (where m < n) are pre-calculated before the short read mapping operation, the m initial search steps can be skipped for all of the short reads by replacing this data with the initial values for top and bot. Also, when only one memory connection is used for both top and bot, the initial steps need two memory accesses to read $O(x, top_{old})$ (Eq. 1) and $O(x, bot_{old})$ (Eq. 2), because their distance is more than d = 64. Therefore, the speedup obtained by using pre-calculated data would be more, when one memory connection is used.

Pre-calculated values are stored in FPGA embedded RAMs (BRAMs). Obviously, there is a limitation for available BRAM memories in any FPGA. According to the limitation of BRAM modules in Virtex 6 LX240T FPGA, we assume m = 9 for our design. Pre-calculated data are compressed more than 5 times in our design. As a result pre-calculated data with m = 9 can are used in our FPGA but another pipeline stage is needed to obtain the original pre-calculated data from compressed data. The speedup for searching short reads with 75 base pairs using pre-calculated data for m = 9 is 1.28x.

3.3 Inexact Match Unit

The proposed inexact match unit (Fig. 2c) is fed by the exact match unit. In this unit three identical seeds (25 bp each) are extracted from each short read and they are searched in the reference genome by an exact match unit. If a seed successfully aligns to the reference genome, the SA values are obtained which specify the locations where the seed exactly matches to the reference genome. These locations are called candidate locations. After reading the candidate locations from the reference genome these locations are compared to the short read which the corresponding seed was extracted from. This comparison is performed using a pipelined comparator and the outputs are the number and the locations of mismatches between the short read and the candidate location.

The pre-calculated data is also used in this module. Because of the smaller length of seeds (25) comparing to the short reads (75) the effect of using pre-calculated data is much more than its effect in the exact match unit in terms of speedup. For the seeds with 25 base pairs, the speed-up while using pre-calculated data is 2x (m = 9).

Another enhancement used in our design, is that the seed aligner needs to search only two seeds instead of three seeds. In our design, additional counters and registers are added to the exact match unit so that when the search steps reach to the one third of the short read, the related top and bot are stored in a register. These data is sent to the inexact match unit. The seed aligner in inexact match unit reuses these data and therefore, does not need to search one of the seeds again. Using this technique and pre-calculated data, searching the seeds in the seed aligner module becomes 3x faster, which is always the slowest module in the inexact match unit pipeline stages.

3.4 Multi-core Version

In order to achieve higher performance, a multi-core accelerator is designed and implemented on the FPGA (Fig. 2b). By considering two changes and applying them to the design we could fit a quad-core design on the target FPGA. (1) Due to lower percentage of short reads with mismatch, two exact match units are connected to a single seed aligner. (2) The seed aligner is the slowest part in our design therefore two seed aligners are connected to a single compare module in the inexact match unit and design works exactly like two separated simple inexact match units. As a result the final design works exactly like four separated simple cores.

4 Implementation and Experimental Results

4.1 Experimental Setup

The aim of this section is to evaluate the proposed methods discussed in Sect. 3. A reference genome with the length of 128 k base pairs is used as the reference genome which is extracted from chromosome 22 (available in [15]). Our design is implemented on the ML605 development board including a Xilinx Virtex-6 LX240T FPGA. Each module is modeled and developed in VHDL language in ISE design suite. The BWT, SA values, O(x,i) and pre-calculated data are generated offline from the reference genome. In our experiment, 10 K short reads with 75 base pairs are extracted directly from the reference genome and 1–3 mismatches are injected randomly into 30% of short reads. Our experiments are done for one million short reads by saving these 10 K short reads in the BRAMs and processing them 100 times.

4.2 Evaluating the Performance

Our design speed is limited to the exact match units. The information about area, number of BRAMs and the run time for processing one million short reads is reported in Table 1. The quality of alignment in FPGA is exactly similar to software and both versions of BWA are tested and the faster result is chosen to be compared with FPGA results.

Table 1. Area and BRAM usage and the run time for searching one million short reads.

	LUT	Register	32 Kb BRAM	Run time (sec)
Quad core design	*29554 (19%)*	31091 (10%)	361 (87%)	0.095

According to the recent works discussed in Sect. 2, searching the smaller percentage of the short reads which contains mismatches is the most time consuming part in short read mapping. Using the optimization techniques proposed in our design, searching the short reads with mismatches has become much faster than searching the short reads in the exact match unit.

4.3 Comparing the Results with Software

To compare the results with software implementation, short reads are searched in the reference genome by the BWA tool on the following two platforms: (1) AMD FX9590 (an eight cores processor) and (2) Intel Extreme Core i7-5820 k (a six cores processor that can handle 12 threads). In this experiment, 100 K short reads are processed by both software and FPGA (using the same reference). The results are compared to the parallel programmed version of the software that supports 8 and 12 threads against the four cores design on the FPGA. The results are shown in Table 2. For a fair comparison the run time for the BWA software is measured only for the align step which only calculates the top and bot values and the number of mismatches.

Table 2. Comparing software and FPGA run time for searching 100 thousand short reads.

	Number of threads	Clock freq. (MHZ)	Run time (sec)	Speed up
AMD FX9590	*8*	4600	0.39	41
Intel core-i7 5820 k	*12*	3300	0.18	19

4.4 Comparing with Other Designs

Our design is compared with [14] which also use a small reference genome. Similar to this paper, [14] uses FPGA memory to store values such as O(x,i), but it uses the backtrack version of FM-index. In [14] one million short reads with 72 bp are searched in the reference with one million base pair and our quad core design is 126 times faster than the six core design in [14] ([14] supports one open gap).

5 Conclusion

In this paper an FPGA implementation of an accelerator with parallel architecture is proposed to solve the long run-time of short read mapping algorithm. The accelerator has been designed based on FM-index algorithm and considers multiple optimizations to enhance the short read alignment speedup such as multi-core structure, multithreading, pipelining and using pre-calculated data. Our paper uses a modified seed and compare version of FM-index to align short reads with 75 bp (up to two mismatches) which does not use the backtrack version of FM-index which is more complex.

References

1. Mardis, E.R.: The impact of next-generation sequencing technology on genetics. Trends Genet. **24**(3), 133–141 (2008)
2. Wetterstrand, K.: DNA sequencing costs, data from the NHGRI Genome Sequencing Program (GSP) (2014). http://www.genome.gov/sequencingcosts
3. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1970)
4. Altschul, S.F., et al.: Basic local alignment search tool. J. Mol. Biol. **215**(3), 403–410 (1990)
5. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics **25**(14), 1754–1760 (2009)
6. Liu, C., et al.: SOAP3: ultra-fast GPU-based parallel alignment tool for short reads. Bioinformatics **28**(6), 878–879 (2012)
7. Ferrragina, P., Manzini, G.: An experimental study of an opportunistic index. In: Proceeding of 12th ACM-SIAM Symposium on Discrete Algorithms, pp. 269–278 (2001)
8. Olson, C.B., et al.: Hardware acceleration of short read mapping. In: 2012 IEEE 20th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 161–168. IEEE (2012)
9. Fernandez, E., Najjar, W., Lonardi, S.: String matching in hardware using FM-index. In: 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 218–225. IEEE (2011)
10. Arram, J., Tsoi, K.H., Luk, W., Jiang, P.: Reconfigurable acceleration of short read mapping. In: 2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 210–217. IEEE (2013)
11. Arram, J., Luk, W., Jiang, P.: Ramethy: reconfigurable acceleration of bisulfite sequence alignment. In: Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 250–259. ACM (2015)
12. Arram, J., et al.: Leveraging FPGAs for accelerating short read alignment. IEEE/ACM Trans. Comput. Biol. Bioinform. (2016). http://ieeexplore.ieee.org/document/7422003/
13. Xin, Y., et al.: Parallel architecture for DNA sequence inexact matching with Burrows-Wheeler Transform. Microelectron. J. **44**(8), 670–682 (2013)
14. Burrows, M., Wheeler, D.: A block-sorting lossless data compression algorithm. Digital Equipment Corporation. Technical report (1994)
15. UCSC Genome Bioinformatics. http://hgdownload.cse.ucsc.edu