Reconfigurable Filtered Acceleration of Short Read Alignment

James Arram* and Wayne Luk* and Peiyong Jiang†
*Department of Computing, Imperial College London
†Department of Chemical Pathology, The Chinese University of Hong Kong

Abstract—**Recent trends in the cost and demand of next generation DNA sequencing (NGS) has revealed a great computational challenge in analysing the massive quantities of sequenced data produced. Given that the projected increase in sequenced data far outstrips Moore's Law, the current technologies used to handle the data are likely to become insufficient. This paper explores the use of reconfigurable hardware in accelerating short read alignment. In this application, the positions of millions of short DNA sequences (called *reads*) are located in a known reference genome. This work proposes a new general approach for accelerating suffix-trie based short read alignment methods using reconfigurable hardware. In the proposed approach, specialised filters are designed to align short reads to a reference genome with a specific edit distance. The filters are arranged in a pipeline according to increasing edit distance, where short reads unable to be aligned by a given filter are forwarded to the next filter in the pipeline for further processing. Run-time reconfiguration is used to fully populate an accelerator device with each filter in the pipeline in turn. In our implementation a single FPGA is populated with specialised filters based on a novel bidirectional backtracking version of the FM-index, and it is found that in this particular implementation the alignment time can be up to 14.7 and 18.1 times faster than SOAP2 and BWA run on dual Intel X5650 CPUs.**

I. Introduction

The downtrend in cost of DNA sequencing has increased the demand for sequenced data in a broad range of research areas. As a result, current sequence analysis applications are quickly becoming unable to handle the massive quantities of data within a satisfactory time frame and accuracy. One such sequence analysis application is short read alignment, where the short DNA sequences generated by an NGS machine are mapped to a known reference genome. This is essentially a pattern matching problem, where the positions in which a short string occurs in a much larger string are computed. Due to possible sequencing errors and genetic diversity, cases where the reads approximately map to the reference sequence must also be considered. This, the size of the reference genome (>billion characters), and the large quantity of short reads generated by the NGS machines (>Gb per run), make short read alignment computationally intensive.

Short read aligners are typically run in software on high performance CPUs. Current software implementations can take multiple CPU-days depending on the problem size, creating a bottleneck point in sequence analysis. Since exciting medical prospects such as pre-natal diagnostics are hinged on the fact that an individuals DNA can be analysed quickly and at a low cost, this application is an excellent candidate for acceleration.

In this paper we explore the use of reconfigurable hardware in accelerating short read alignment. We propose a new general approach for accelerating suffix-trie based short read alignment methods. The novel features of this approach (as compared to previous related work) is summarised in the following two statements:

1) specialised circuits (*filters*) are built to align short reads to a reference genome with a specific edit distance: for example, exact match, one mismatch, two mismatch, etc. The filters are arranged in a pipeline according to increasing edit distance, where reads unable to be mapped by a given filter are forwarded to the next filter in the pipeline for further processing.
2) The accelerator device is fully populated with each filter in the pipleline *in turn* using run-time reconfiguration, such that the population of each type of filter is maximised for the respective alignment stage.

The alignment performance is benefited by this approach as: a) specialised alignment heuristics can be applied to each filter to greatly reduce the process time, and b) the hardware configuration is optimised according to the intended work flow, short read data properties, and available resources.

II. Background and Related Work

The most commonly used algorithms for short read alignment can be categorised into two methods: suffix-trie methods, and seed-and-extend methods. For short reads of length $\sim 10^{1-2}$ bases (such as those produced by Illumina sequencing devices), it is found that suffix-trie methods can outperform seed-and-extend methods in terms of alignment time. In this paper we propose a new general method for accelerating suffix-trie based short read alignment methods, using the FM-index in our implementation.

A. FM-index

The FM-index [1] is a space-efficient data structure which supports substring matching. This data structure has been used to perform exact and approximate matching in several software short read aligners, such as SOAP2 [2] and BWA-backtrack [3].

The FM-index combines the properties of suffix array with the Burrows-Wheeler transform (BWT) [4], to provide a compact and efficient method for finding all occurrences of a pattern in a reference sequence (exact matching). In this approach each character in the pattern is matched against a suffix-trie of the reference sequence generated using the BWT.

978-1-4799-2198-0/13/$31.00 ©2013 IEEE

Suffix-trie methods can be extended with backtracking to allow for approximate matching between a short read and reference genome. In this approach, when a mismatch is detected, a stack is used to store the current search state. A new character in the reference alphabet is then attempted for matching to the reference genome. When the number of mismatches exceeds the permitted value, the state is restored from the stack and an untested character is attempted for matching to the reference genome.

When using backtracking to extend suffix-trie methods for approximate matching, the performance is greatly affected by the mismatch position. For example, reads with mismatches at the end are matched several times slower than ones with mismatches at the beginning. This problem can be reduced by using a bi-directional search, where the read is matched in both a forward and backward direction. This is supported by the 2BWT [5] data structure, which supports searching in both directions and allows dynamic switching of search direction.

B. Related Work

There exist several efficient hardware designs for accelerating short read alignment using reconfigurable hardware. In [6], a multi-threaded design based on the FM-index is proposed for FPGA. For every n allowed mismatches, $n + 1$ exact string matchers (engines) from [7] statically populate an accelerator device. Short reads are first processed by engine 0. If a mismatch is detected, the read is copied and the mismatch character is replaced with different characters in the reference genome alphabet. The copies are then sent to the next engine in line for further exact matching. In [8], a similar design is proposed, where an FPGA is statically populated with exact string matchers based on the BWT. The host CPU sends reads to the FPGA for processing. If the FPGA reports that a read cannot be mapped to the reference sequence, the host modifies one or two characters, and sends the short read back to the FPGA for further processing. The host maintains a stack to keep track of the variations of a short read sent to the FPGA. In [9], an alignment processor based on a backtracking version of the FM-index is designed. Several optimisations are applied to improve the alignment performance, these include; interleaving the processing of multiple reads to hide external memory access latencies, reducing the external memory requirement, and improving the external memory bandwidth utilisation.

The main contribution of this work is a general method for accelerating suffix-trie short read alignment methods. The proposed method consists of a pipeline of specialised filters, where each filter is able to align short reads to a reference genome with a specific edit distance. Run-time reconfiguration is used to fully populate the accelerator device with each filter in the pipeline in turn. This differs from the previous work above which all consist of an accelerator device statically populated with a homogeneous array of string matchers.

III. Method Overview

In this section the two novel features of the proposed approach for accelerating suffix-trie short read alignment methods are presented, namely the pipeline of specialised filters and the use of run-time reconfiguration to populate the accelerator device. The advantages of these features over previous related work are discussed and a particular implementation for each feature is presented.

A. Specialised Filters

In software, suffix-trie methods are typically implemented as a filtered search. In this approach when a short read cannot be aligned to the reference genome within a given constraint (e.g. number of mismatches), it is processed again and the constraint is slackened. In previous work this functionality has been attempted by using two different approaches:

1) A homogeneous array of exact string matchers. In this approach when a short read cannot be aligned with a permitted number of mismatches, all the possible permutations of the short read with an increased number of mismatches are generated and reprocessed.
2) A homogeneous array of general backtracking processors. In this approach when a short read cannot be aligned with a permitted number of mismatches, the permitted number of mismatches is incremented in the processor and the short read is reprocessed.

For both approaches the search space increases exponentially with edit distance, and as a result the alignment performance scales poorly with the number of permitted mismatches.

In the proposed method, rather than using a homogeneous array of string matchers, specialised filters are built to align short reads to a reference genome with a specific edit distance: for example, exact match, one mismatch, two mismatch, etc. The filters are arranged in a pipeline according to increasing edit distance, where reads unable to be mapped by a given filter are forwarded to the next filter in the pipeline for further processing. The benefit of this approach is that it allows the incorporation of specialised alignment heuristics in each filter to greatly speed up the alignment time (especially for approximate matching).

For our particular implementation the backtracking FM-index design in [9] is transformed into a pipeline of specialised filters for aligning reads with up to two mismatches with respect to the reference genome (exact match, one mismatch and two mismatch filters). A specialised alignment heuristic is incorporated in each mismatch filter to greatly speed up the alignment time for approximate matching. For reads with one mismatch, the mismatch can either occur in the first half of the read (case A) or the second half of the read (case B). For reads with two mismatches, the mismatches can either occur in: the first two thirds of the read (case A), the last two thirds of the read (case B), or the first and last third of the read (case C). By constraining the mismatch position, long sections of the read can be exactly matched first. The resulting suffix array interval is usually very small, therefore each permutation of the short read can be tested for with only a few additional search steps. Note that this alignment heuristic would be impossible to implement in a homogeneous array of string matchers.

Since the alignment heuristic features both forward and backward searching, the 2BWT data structure presented in Section II is used. Each filter comprises of processing elements (PE's) which map the reads according to the cases in the alignment heuristic for the permitted number of mismatches. The packed short reads are streamed to each PE, where they are stored using on-chip BRAMs. The PE's contain a buffer which is able to store a small batch of short reads. This allows for the processing of reads whilst others are waiting for external memory (for accessing the 2BWT data structure). As a result the throughput for each PE is one short read symbol matched per cycle. BRAM is also used to store the search state for each mismatch permitted to allow backtracking when a given permutation cannot be aligned to the reference genome. A system of counters and multiplexers is used to control the backtracking functionality and alignment heuristic (when to exact match or test for mismatches) for each PE.

B. Run-time Reconfiguration

In previous work the accelerator device is statically populated with a homogeneous array of string matchers. For filtered approaches such as the one in [7], the short reads are internally forwarded to the next string matcher in the pipeline when a mismatch is detected. Given that the majority of the reads can be mapped with small edit distances, the filters at the later stages of the pipeline are mostly idle. As a result the hardware configuration is unoptimised according to the intended work flow, short read data properties, and available resources.

In the proposed method, rather than statically populating the accelerator device, run-time reconfiguration is used to fully populate the accelerator device with each filter in the pipeline *in turn*, such that the population of each type of filter is maximised for the respective alignment stage. This maximises the hardware efficiency as the entire population of filters on the accelerator device is active in each alignment stage. The overhead of this approach is the reconfiguration time of the accelerator device, however for typical workloads this is negligible compared to the total alignment time.

For our particular implementation the packed short reads are streamed from the host to the FPGA device, where they are processed by the first filter in the pipeline. The alignment results are streamed back to the host and parsed. The FPGA device is then reconfigured to become fully populated with the next filter in the pipeline. The unaligned short reads from the previous alignment stage are then re-streamed to the FPGA for further processing. This procedure is repeated until the unaligned short reads have been processed by the last filter in the pipeline. The overhead of this approach is transferring the reads via PCIe to and from the accelerator board, however since this application is memory bound, the impact is negligible.

IV. Performance Evaluation

In this section we evaluate the performance of the implementation in Section III on the MaxWorkstation provided by Maxeler Technologies. In this work, we use the Human Genome version 18 [10] and single-end 75 character reads sampled directly from the reference genome. Noise is inserted in random positions in the reads to simulate mismatches between the reads and reference genome.

In Table I the FPGA resource utilisation for each filter is presented. Note that the logic resources required to orchestrate the data flow between the filters and host application are included in the measurements. The resource utilisation values indicate that the filter size linearly increases with the number of mismatches. Based on this result, it makes sense to use run-time reconfiguration to populate the FPGA with each filter in the pipeline in turn, as in a static design most of the available hardware resources and power would be used for the mismatch filters which only process a small fraction of the short reads.

TABLE I
Resource utilisation comparison.

Filter	LUT	Flip-Flop	BRAM	approx. relative size
exact match	37366	58467	116	1x
one mismatch	55173	89085	249	2x
two mismatch	72980	119703	382	3x

Since different short read alignment implementations report their performance using different data sets and parameters, it is difficult to directly compare designs using raw results. To better assess the performance of various designs, we define *bases aligned per second* (bps) as a normalised performance merit. The bps value of a design is given by Equation 1.

$$bps = \frac{\text{read size} \times \text{read count}}{\text{process time}} \tag{1}$$

A limitation of implementing the proposed filter designs on the Maxeler Workstation is the MAX3 memory architecture. The fast DRAM is designed for accessing large contiguous chunks of data, rather than random access. Since memory access is completely random for FM-index, the performance is significantly lower than anticipated. Furthermore, for the MAX3 DFE, all the memory pins are used in a single channel, therefore one burst can be read per cycle at most. Since in each filter more than one burst is read per cycle, the performance does not improve when increasing the population of filters on the FPGA. We estimate the alignment performance without the platform-specific limitations of the MAX3 DFE by maximising the population of each type of filter according to the available resources, and assuming that there is sufficient bandwidth to process the reads concurrently. Given that accelerator platforms have multiple memory channels with random access speeds close to that for sequential access, the assumptions made here are reasonable.

In Table II, the alignment run-time of the implementation presented in Section III is compared to BWA and SOAP2. We include measurements for: a) a single filter design, where one filter populates the FPGA device in each alignment stage, and b) an upper bound estimate, where the filter population is maximised according to the available resources. The following commands are used to run the software aligners:

TABLE II
PERFORMANCE COMPARISON. NOTE THAT OUR RUN-TIME MEASUREMENTS TAKE INTO ACCOUNT ONLY THE ALIGNMENT TIME.

	read size (base)	read count (million)	platform	clock freq. (MHz)	device(s)	cores per device	run time (s)	bps (million)	bps per slice
SOAP2 [2]	75	18	Intel X5650	2670	2	6	204	6.6	-
BWA [3]	75	18	Intel X5650	2670	2	6	252	5.4	-
single filter	75	18	Xilinx Virtex-6 SX475T	150	1	1:1:1	60.4	**20.8**	280
upper bound	75	18	Xilinx Virtex-6 SX475T	150	1	7:3:2	13.8	**97.5**	1310
SOAP3 [11]	100	70.7	NVIDIA GTX 580	900	1	512	1839	3.84	-
Design in [12]	76	50	Xilinx Virtex-6 LX240T	250	8	8	34	112	372
Design in [6]	101	18	Xilinx Virtex-5 LX330	150	4	3	138	13.1	63.2
Design in [13]	76	1	Xilinx Virtex-5 LX330	-	1	4	21	3.6	17.3
Design in [9]	76	1	Xilinx Virtex-6 SX475T	150	1	3	5.6	13.5	181

- ./soap -D Hg18.fa.index -a reads.fq -M 2 -v 2 -p 12 -o soap2.out
- ./bwa aln -k 2 -n 2 -t 12 Hg18.fa.index reads.fq > bwa.out

The *bps* values in Table II indicate that a single filter implementation is 3.1 and 3.8 times faster than SOAP2 and BWA run on 12 cores respectively. It is found that $\sim 66\%$ of the alignment time is spent on exact matching the reads, therefore the performance is significantly limited by the small population of exact match filters. The total reconfiguration time is 1.6 seconds, which is less than 3% of the alignment time. In the upper bound estimate, the FPGA can be populated with 7 exact match filters, 3 one mismatch filters, or 2 two mismatch filters ("7:3:2" in Table II). The performance is estimated to be 14.7 and 18.1 times faster than SOAP2 and BWA respectively. The significant speed up is due to the higher population of filters in each alignment stage.

In Table II, the alignment run-time of previous hardware and GPU-based implementations are listed. It is difficult to compare results as the reporting methods and short read data sets differ. Furthermore previous work such as [14] only provides upper bound estimates with no real measurements. By comparing the *bps* per slice in Table II, to take into account for each design the number of FPGAs and the number of slices in each FPGA, we predict that this implementation is the fastest suffix-trie alignment method implementation, and among the fastest overall.

V. CONCLUSION

In this work, we demonstrate that by designing highly specialised alignment filters, and using run-time reconfiguration to configure an FPGA device with each type of filter in turn, high throughput short read alignment can be achieved. Future research includes; designing more complex filters, further optimising the external memory accessing, and fully integrating our designs into an existing software aligner.

ACKNOWLEDGEMENT

We thank Dennis Lo and Rossa Chiu for their advice and encouragement. This work was supported in part by Maxeler University Programme, Xilinx, UK EPSRC, the European Union Seventh Framework Programme under grant agreement number 257906, 287804 and 318521, and the HiPEAC NoE.

REFERENCES

[1] P. Ferragina and G. Manzini, "An experimental study of an opportunistic index," in *Proc. SODA*, 2001, pp. 269–278.
[2] R. Li *et al.*, "SOAP2: an improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, no. 15, pp. 1966–1967, August 2009.
[3] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.
[4] M. Burrows and D. Wheeler, "A block-sorting lossless data compression algorithm," Digital Equipment Corporation, Tech. Rep., 1994.
[5] T. Lam, R. Li, A. Tam, S. Wong, E. Wu, and S. Yiu, "High throughput short read alignment via bi-directional BWT," in *Proc. BIBM*, nov. 2009, pp. 31 –36.
[6] E. Fernandez, W. Najjar, S. Lonardi, and J. Villarreal, "Multithreaded FPGA acceleration of DNA sequence mapping," in *Proc. HPEC*, 2012.
[7] E. Fernandez, W. Najjar, and S. Lonardi, "String matching in hardware using the FM-index," in *Proc. FCCM*, 2011, pp. 218–225.
[8] P. Draghicescu, G. Edvenson, and C. Olson, "Inexact search acceleration on FPGAs using the burrows-wheeler transform," 2012.
[9] J. Arram, K. H. Tsoi, W. Luk, and P. Jiang, "Hardware acceleration of genetic sequence alignment," in *ARC*, 2013, pp. 13–24.
[10] [Online]. Available: http://hgdownload.cse.ucsc.edu/goldenpath/hg18/chromosomes/
[11] C. Liu *et al.*, "SOAP3: Ultra-fast GPU-based parallel alignment tool for short reads," *Bioinformatics*, vol. 28, no. 6, pp. 878–879, 2012.
[12] C. B. Olson *et al.*, "Hardware acceleration of short read mapping," in *Proc. FCCM*, 2012, pp. 161–168.
[13] Y. Chen, B. Schmidt, and D. Maksell, "An FPGA aligner for short read mapping," in *Proc. FPL*, 2012, pp. 511–514.
[14] J. Arram, K. H. Tsoi, W. Luk, and P. Jiang, "Reconfigurable acceleration of short read mapping," in *Proc.FCCM*, 2013.