circFA: a FPGA-based circular RNA aligner

Alberto Zeni, Francesco Peverelli, Enrico Cabri, Lorenzo Di Tucci, Luca Cerina, Marco D. Santambrogio
[1]Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Milano, Italy,
{*alberto.zeni, francesco1.peverelli, enrico.cabri*}@mail.polimi.it
{*lorenzo.ditucci, luca.cerina, marco.santambrogio*}@polimi.it

***Abstract*—Circular RNAs (CircRNAs) are a widespread form of Non-Coding RNAs (ncRNAs). Although circular RNAs have been known for a long time, it is only recently that their therapeutic implications are being investigated. Research has shown that they are suitable candidates to be genetic biomarkers for different types of cancer since they are implied in Carcinogenesis. However, their identification involves a computationally intensive and very long process. This represents a serious obstacle to their ability to provide a fast and accurate life-saving diagnosis. Here we present Circular RNA FPGA Aligner (circFa), a hardware implementation of an alignment algorithm designed for the circular RNAs identification protocol. Our solution aims to reduce diagnosis time and produces a pharmacological therapy tailored to the specific patient using his genetic information. CircFA implements the same scalable version of the Smith-Waterman (SW) algorithm with affine gaps employed by *Bowtie2*, a state of the art tool for sequence alignment, on Field Programmable Gate Array (FPGA). Our design shows a speedup over the software version of SW of 8.8x, and a speedup of 1.95x over a parallelized multicore version of the same software. Our architecture also shows a significant increase in power efficiency with an increase of 80.96x and 17.96x with respect to the single core and multicore software respectively.**

I. Introduction

Traditional cancer diagnoses establishments use a complex combination of clinical and histopathological data [1]. However, the retrieval of these data is long, time-consuming and in some instances, impossible. The latest technological advancements in RNA sequencing offer new paths to cancer screening such as biomarkers [2]. Biomarkers are biological molecules found in blood, other body fluids, and tissues that are a sign of a normal or abnormal process. These molecules may be used to see how the body responds to a treatment for a disease or to identify a pathological condition [3]. This approach to cancer diagnosis aims at detecting the pathology in its early stages and identifying potential metastasis and their location before a critical spreading. Recent discoveries have shown that Circular RNAs (CircRNAs) are suitable candidates to become genetic biomarkers [4, 5]. These biomolecules are involved in central nervous system diseases, such as Alzheimer's disease (AD) [6] and also in Carcinogenesis of various forms of malignancies [7, 8], showing a major role in Epithelial to Mesenchymal Transition (EMT) [9]. CircRNAs act as regulators of post-transcriptional products, behaving as sponges for microRNA (miRNA), crucial in fine-tuning gene expression, thus a possible dysregulation of the pathway can lead to a pathological event [10]. These crucial discoveries led to a growing interest in the identification of CircRNAs aimed at the detection of their position in the genome, and their effects in physiological and pathological conditions. This information would lead to a considerable reduction in diagnosis time, and also to the creation of drugs tailored to the genetic heritage of the patients, with less aggressive side effects and decreased resolution time. Latest CircRNAs identification pipelines, such as TopHat2 [11], count on a previous alignment step mostly performed by Bowtie2-based aligners. In these regards, Bowtie2 is employed to align reads together with a particular variation of the Smith-Waterman (SW) algorithm [12]. This alignment process represents the main bottleneck of the whole identification process, requiring long runtimes [11].

Field Programmable Gate Arrays (FPGAs) have already proven to be able to achieve outstanding results on these type of algorithms while requiring a relative low power consumption [13]. Our work proposes an acceleration on FPGA of the sequence alignment functionality of Bowtie2: the main bottleneck for the identification of CircRNA. Systolic-array-based architectures are the most popular solution for hardware accelerations of SW since they allow to exploit the maximum level of parallelism possible for the algorithm. The most notable drawback of this approach is that the number of query elements processed in parallel is fixed, thus it is impossible to process queries of arbitrary length. As a conclusion in this paper we propose the following contributions:

1) A flexible hardware implementation of the SW algorithm that allows input queries of arbitrary length
2) A hardware implementation of SW with affine gap penalties that is compliant with the standard alphabet and scoring system for Bowtie2
3) A software interface that allows to input reads and reference genome in standard FASTA format.

The rest of the paper is organized as follows: Section II presents an overview of SW hardware implementations; Section III details our implementation and provides a description of the optimizations implemented; Section IV reports the experimental results obtained and finally, Section V states the conclusions of our work and gives some insights on future improvements. The source code is publicly available[1].

II. State of the Art

In this section, we report a brief introduction on the most relevant approaches for the hardware implementation of the SW algorithm and other accelerations of Bowtie2.

The best implementation in software of the SW algorithm is the one described in [14], also used in Bowtie2. It leverages

[1]https://github.com/necst/circFAXOHW18public

978-1-7281-0848-3/19/$31.00 ©2019 IEEE

SIMD instructions and reaches performances of more than 11 and 20 GCUPS on four and eight cores respectively. The same software optimized to run on the PlayStation 3 and the IBM QS20 architecture [15] achieve performances of 15.5 and 11.6 GCUPS respectively. The work presented in [16] uses Nvidia GPUs instead of General Purpose Processors (GPPs), leveraging the parallelization opportunities of the algorithm and SIMD instructions to achieve 109.4 and 169.7 GCUPS on the GTX680 and GTX690 cards respectively. Although these results are very promising, they focus exclusively on short reads alignments. FPGA can offer more energy-efficient solutions with a relatively low power profile while still exploiting the full parallelism of the algorithms implemented [17, 18]. Li et al. work [19] accelerate the algorithm with a speedup of over 160x compared to a pure software implementation, using an FPGA with an Altera Nios II soft processor. However, this solution achieves performance comparable to existing optimized software implementations. *FPGASW* [20] also implements the backtracking phase of the algorithm, absent in the other implementations, achieving a speedup between 3.6x and 25.2x. The solution presented in Di Tucci et al. [13] obtains a speedup of 1.72x over the state-of-the-art FPGA implementation and is 8.49x more energy efficient than comparable GPU implementations. It uses a Xilinx Virtex 7 and Kintex Ultrascale platform. This implementation, however, it is currently limited to align queries and database segments that have a number of characters which cannot exceed the number of processing elements in the architecture. All the hardware implementations presented above accelerate the SW algorithm without considering the linearity of the gaps in the alignment, thus simplifying the execution removing a lot of data dependency. Also, both the query and the reference database are limited in their maximum length, thus making alignments of long reads impossible on those architectures.

III. Proposed Solution

In this Section, we provide a description of the proposed implementation. We start with a high-level overview of the architecture, followed by a description of its components and its memory organization. Finally, we describe the software interface we developed for the architecture.

A. Overview

Our architecture implements a scalable version of the SW algorithm accepting as inputs sequences of variable length. To achieve this, we split the computation of the alignment matrix of SW into smaller subsections. Figure 1 shows an overview of our architecture: a scoring kernel computes each subsection of the alignment matrix, while a scheduler module fetches data from memory and then feeds them to the scoring kernel accordingly. To support different inputs, data is stored both into BRAM, when longer sequences are aligned, and registers, when it needs to be rapidly accessed.

1) Scoring kernel: The scoring kernel performs the computation of the scoring matrix and the direction matrix used by the SW algorithm. Our implementation is based on a systolic

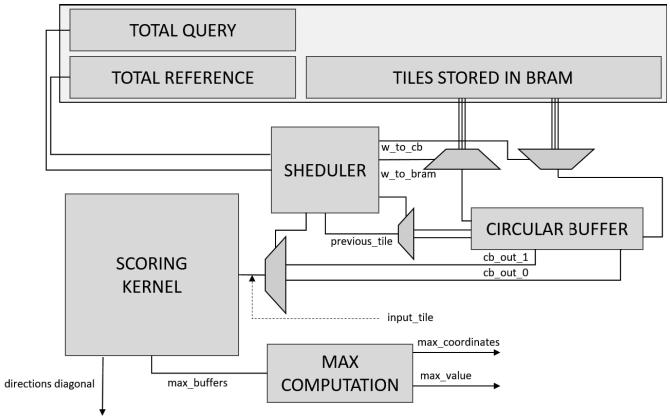

Fig. 1. High-level schema of the architecture

array architecture, a well-known architecture for hardware implementations of the SW algorithm. This architecture leverages the fact that all the cells of the scoring matrix, which lie on the same anti-diagonal, have no data dependencies among one another and can be computed in parallel (Figure 2).

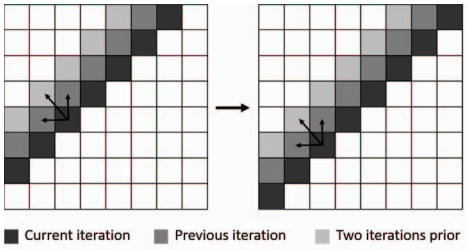

Fig. 2. Dependencies for a given cell of the alignment matrix at a given iteration.

The scoring kernel takes as inputs a query and a database sequence to be aligned, as well as four buffers. These buffers contain the values of the scoring matrices calculated from the two previous anti-diagonals. These values are required to compute each anti-diagonal of the scoring matrix correctly, as shown in Figure 2 The main loop calculates in parallel the values of each cell of the anti-diagonal of the scoring matrix, as well as the corresponding values of the directions matrix D and stores them in a temporary buffer. The directions matrix is used for the backtracking phase of the SW algorithm, and it is necessary to identify the position of the optimal alignment after the scoring matrix is computed. The maximum score value calculated for the current anti-diagonal and its position are also saved in a buffer by the scoring kernel. These values are necessary to calculate the overall maximum score and its position once the alignment scores of all the anti-diagonals have been calculated. A second loop then writes back the diagonal of the D matrix in a dedicated buffer in BRAM. We chose the dimension of the systolic array to maximize the hardware utilization, taking into account the variable length of the input queries. Our architecture implements a systolic array of 42 Processing Elements (PEs). This number aims to

balance the achievable level of parallelism for long queries and the minimization of unused hardware resources for short queries.

2) Scheduling: To calculate the complete scoring matrix and the corresponding directions matrix for any given input query and reference, we have to split the computation of the matrices.

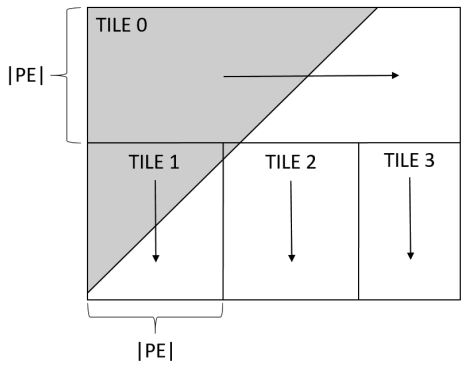

Fig. 3. Tile division of the alignment matrix

We cannot split the computation of the scoring matrix into smaller anti-diagonals, as there will always be some dependency between two consecutive anti-diagonals. For this reason, we split the scoring matrix in subsections, called *tiles* (Figure 3) To calculate a complete anti-diagonal of the scoring matrix we have to start from its bottom subsection, and then proceed to provide the appropriate data dependencies to the next subsection. Each tile stores the dependencies for a given sub-diagonal, and, at each iteration, computes the values for the next sub-diagonal. In our design, we have two types of tile: one which covers a horizontal subsection of the scoring matrix and several others which cover vertical subsections. When the execution starts, the horizontal tile starts computing the anti-diagonals. The characters of the reference sequence are given to the scoring kernel following a FIFO policy, where each of the characters is assigned to one PE. When the number of elements in the anti-diagonal exceeds the number of PEs in the systolic array, another tile starts to calculate the sub-diagonal immediately below. The horizontal tile reads the data dependencies written by this second tile when it completes its execution and then proceeds to calculate the remaining subsection of the anti-diagonal. The tile that started its execution will then continue to compute the sub-diagonals of a vertical section of the scoring matrix. A new vertical tile is scheduled to start executing every |PE| cycles. In this way, we are able to cover the whole scoring matrix as we increase the number of PEs together with the length of the anti-diagonal of the scoring matrix. The scheduler calls the scoring at every iteration with the appropriate input arguments of the tile that needs to be executed. The results coming from the scoring kernel are then written back into memory. The scheduler takes care of computing which tile needs to execute at the next iteration, setting the inputs for the next execution of the scoring kernel. Finally, when all the anti-diagonals have been calculated, a loop computes the overall maximum score value and its position.

3) Memory organization: Since our kernel accepts inputs of variable length, we cannot know at compile time the maximum number of tiles that will be executing concurrently. For this reason, we instruct the architecture on where to write the results of the scoring kernel at each iteration, and where to read the appropriate data dependencies for the next execution. A buffer holds all the inputs necessary for the next call to the scoring kernel, as well as results of the previous iteration. The data that will be required in the following clock cycles is stored in BRAM. When a single tile is calculated all the data is stored in the buffer, as the anti-diagonal is not divided into different sections. Moreover, when two active tiles are calculated, we can read the necessary dependencies, as well as write the results of the computation, by alternating the positions from which we write and read from the buffer. If we have more than two tiles, when the results of an iteration are written in the buffer, the results of the iteration preceding it are written to BRAM. In the same way, when a set of inputs is read from the buffer, the set of inputs for the iteration that will follow are written from BRAM into the buffer. In this way, we can store a variable number of results from different tiles in BRAM, processing queries of variable length.

B. Software Interface

The software application is a C++ host for the hardware kernel, which is implemented with the OpenCL paradigm through the interface provided by SDAccel. It implements a simple interface in which the user can provide as input arguments two FASTA files, containing the read sequences and the corresponding reference genome fragments. The software processes each sequence, transforming the original ASCII encoding into a custom three-bit encoding. A straight-forward two-bit encoding was not possible since, in addition to the four DNA bases, Bowtie2 uses an additional character N, which represents an ambiguity occurred in the sequencing phase and is treated as a separate case in the scoring process. Once the sequences have been encoded, the application sets up the necessary infrastructure to call the kernel. The kernel is then called for each pair of input read and reference, and it outputs the maximum score for the pair, its position in the dynamic-programming matrix and a matrix of directions for the following phase.

IV. Experimental results

In this section, we provide a description of the experimental settings to test our design as well a discussion on the results obtained.

A. Experimental setup

The design of the architecture has been described using C++ and Xilinx Vivado HLS. Xilinx SDAccel has been used to perform the bitstream generation step. We benchmarked the architecure using the OpenCL events that provide an easy to use API to profile the code that runs on the FPGA device.

TABLE I
EXECUTION TIMES WITH AN INPUT QUERY AND REFERENCE PAIR OF 3024 BP

Platform	No. cores	Execution Time (ms)	Speedup w.r.t Smith-Waterman	Speedup w.r.t Vectorized	PowerEff w.r.t Smith-Waterman	PowerEff w.r.t Vectorized
CPU	1	631.66	-	-	-	-
CPU	40	140.12	4.5x	-	4.5x	-
FPGA	1	71.78	8.8x	1.95x	80.96x	17.96x

To validate our architecture, we have compared it with two optimized software implementations of the SW algorithm. Both software versions have been benchmarked a server with two Intel Xeon E5-2680 v2 CPU with a peak frequency of 3.6 GHz, 380 GB of RAM and a TDP of 230W, compiled with gcc 4.8.5 with -O3 optimization. The second implementation has been parallelized, to vectorize its computation and to use all the 40 cores available, using OpenMP. The hardware-accelerated components were developed using both Vivado HLS 2017.2 and SDAccel 2017.1.op, and have been tested on a Amazon F1 instance equipped with a Virtex Ultrascale+ VU9P FPGA, with a TDP of 25W. The design was synthesized with a clock frequency of 250 MHz. As we can see in Table I, our design outperforms both software implementation, showing a speed-up of 1.95x over the vectorized version and 8.8x with respect to the single core one. Furthermore, we observe a significant increase in power efficiency, showing an 80.96x increase over the single core and a 17.96x with respect to the multicore design.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we have presented circFA a novel FPGA-based sequence aligner designed for CircRNA identification. The proposed architecture is able to obtain a speedup of 8.8x over the simple implementation of the Smith-Waterman algorithm, 1.95x over the vectorized version of the latter and an increase in power efficiency of 80.96x and 17.96x with respect to the same software. These results already show a significant speedup with respect to present software alternatives, that could led to a singificant reduction of the clinical times required by precision medicine therapies. Future implementations of our work will explore different ways to schedule the computation of the system and also introduce a multi-core implementation in order to achieve even better performances. Moreover, the final aim of this project is to be integrated with the state-of-the-art alignment software, in particular, Bowtie2, to create a software/hardware pipeline for the identification of circular RNAs.

REFERENCES

[1] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov *et al.*, "Multiclass cancer diagnosis using tumor gene expression signatures," *Proceedings of the National Academy of Sciences*, vol. 98, no. 26, pp. 15 149–15 154, 2001.

[2] D. E. Henson, S. Srivastava, and B. S. Kramer, "Molecular and genetic targets in early detection," *Current opinion in oncology*, vol. 11, no. 5, p. 419, 1999.

[3] NIH National Cancer Institute., "Biomarker Definition." [Online]. Available: https://www.cancer.gov/publications/dictionaries/cancer-terms/def/biomarker

[4] S. Meng, H. Zhou, Z. Feng, Z. Xu, Y. Tang, P. Li, and M. Wu, "Circrna: functions and properties of a novel potential biomarker for cancer," *Molecular cancer*, vol. 16, no. 1, p. 94, 2017.

[5] D. Rong, H. Sun, Z. Li, S. Liu, C. Dong, K. Fu, W. Tang, and H. Cao, "An emerging function of circrna-mirnas-mrna axis in human diseases," *Oncotarget*, vol. 8, no. 42, p. 73271, 2017.

[6] W. J. Lukiw, "Circular rna (circrna) in alzheimers disease (ad)," *Front Genet*, vol. 4, no. 4, p. 307, 2013.

[7] T. L. H. Okholm, M. M. Nielsen, M. P. Hamilton, L.-L. Christensen, S. Vang, J. Hedegaard, T. B. Hansen, J. Kjems, L. Dyrskjøt, and J. S. Pedersen, "Circular rna expression is abundant and correlated to aggressiveness in early-stage bladder cancer," *NPJ genomic medicine*, vol. 2, no. 1, p. 36, 2017.

[8] Z. Kun-Peng, M. Xiao-Long, and Z. Chun-Lin, "Overexpressed circpvt1, a potential new circular rna biomarker, contributes to doxorubicin and cisplatin resistance of osteosarcoma cells by regulating abcb1," *International journal of biological sciences*, vol. 14, no. 3, p. 321, 2018.

[9] S. J. Conn, K. A. Pillman, J. Toubia, V. M. Conn, M. Salmanidis, C. A. Phillips, S. Roslan, A. W. Schreiber, P. A. Gregory, and G. J. Goodall, "The rna binding protein quaking regulates formation of circrnas," *Cell*, vol. 160, no. 6, pp. 1125–1134, 2015.

[10] T. B. Hansen, J. Kjems, and C. K. Damgaard, "Circular rna and mir-7 in cancer," *Cancer research*, vol. 73, no. 18, pp. 5609–5612, 2013.

[11] D. Kim, G. Pertea, C. Trapnell, H. Pimentel, R. Kelley, and S. L. Salzberg, "Tophat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions," *Genome biology*, vol. 14, no. 4, p. R36, 2013.

[12] S. TF and W. MS, "Identification of common molecular subsequence," *Mol. Biol*, vol. 147, pp. 195–197, 1981.

[13] L. Di Tucci, K. O'Brien, M. Blott, and M. D. Santambrogio, "Architectural optimizations for high performance and energy efficient smith-waterman implementation on fpgas using opencl," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 716–721.

[14] M. Farrar, "Striped smith–waterman speeds database searches six times over other simd implementations," *Bioinformatics*, vol. 23, no. 2, pp. 156–161, 2006.

[15] A. Szalkowski, C. Ledergerber, P. Krähenbühl, and C. Dessimoz, "Swps3–fast multi-threaded vectorized smith-waterman for ibm cell/be and× 86/sse2," *BMC research notes*, vol. 1, no. 1, p. 107, 2008.

[16] Y. Liu, D. L. Maskell, and B. Schmidt, "Cudasw++: optimizing smith-waterman sequence database searches for cuda-enabled graphics processing units," *BMC research notes*, vol. 2, no. 1, p. 73, 2009.

[17] J. P. Oliver, J. Curto, D. Bouvier, M. Ramos, and E. Boemo, "Clock gating and clock enable for fpga power reduction," in *Programmable Logic (SPL), 2012 VIII Southern Conference on*. IEEE, 2012, pp. 1–5.

[18] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient fpgas," in *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*. ACM, 2003, pp. 175–184.

[19] I. T. Li, W. Shum, and K. Truong, "160-fold acceleration of the smith-waterman algorithm using a field programmable gate array (fpga)," *BMC bioinformatics*, vol. 8, no. 1, p. 185, 2007.

[20] X. Fei, Z. Dan, L. Lina, M. Xin, and Z. Chunlei, "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.