FPGA-based Heterogeneous Architecture for Sequence Alignment

Xin Chang Fernando A. Escobar Carlos Valderrama
Service d'électronique et de Microélectronique
Faculté Polytechnique de Mons
Université de Mons,7000, Belgium

ABSTRACT

With the rapid development of genome sequencing technology, the cost of getting genome data is becoming decreasingly insignificant. However, the computational speed for analysis genome data remains same. The bioinformatics community is facing a serious challenge to deal with massive data. In this paper, we proposed a novel heterogeneous architecture for sequence alignment. As will be demonstrated, the speed of sequence alignment can be improved with reasonable resources utilization on programmable logic.

Keywords

Sequence Alignment,Smith-Waterman,Heterogeneous Architecture,FPGA,Systolic Array;

1. INTRODUCTION

Nowadays bioinformatics plays an essential role in processing genomic, medical and proteomic data generated by high-throughput technologies. Genome sequencing is, in particular, an emerging technique widely used. With the development or the novel technologies such as PCR[1] sequencing, the cost and time of genome sequencing have dropped dramatically over the last decade.

Sequence alignment analyzes similarities between DNA, or protein sequences, to assess the genetic relationship between organisms or species. It helps scientists to check pathogenic mutations, understand the evolution of creatures and predict the structure of organisms. It is widely used in the bioinformatics, drugs and medicine design among other related areas. However, the rate of genome data generation exceeds the speed at which it can be computationally processed. In addition, the databases of genome sequences are spreading and becoming large-scale. These are reasons why the acceleration of genome sequence alignment has become an emerging bioinformatics activity.

This paper is going to propose a novel FPGA-based heterogeneous architecture for sequences alignment. As will be demonstrated, the proposed architecture outperforms state-of-art approaches in terms of speed thanks to an optimized hardware and software partition. In the next section, current solutions for sequence alignment and background of Smith-Waterman algorithm will be introduced. Section III will analyse our optimization target, the Smith-Waterman algorithm, highlighting the bottlenecks of Smith-Waterman algorithm and optimization strategies. Section IV will define the proposed implementation architecture. Preliminary evaluation results will be shown in Section V. The last section will conclude and present the future works.

2. BACKGROUND

A broad set of genome sequence alignment algorithms are available. They can be classified according to the number of sequences that will be processed simultaneously in pairwise and in multiple sequence alignment. Probcons[2], T-Coffee[3] are known as the most commonly used algorithms for multiple sequence alignment. Nevertheless, most of multiple sequence approaches are based on an extended version of pairwise sequence alignment algorithms. Regarding pairwise sequence alignment algorithms, the Smith-Waterman, BLAST[4] and HMMER[5] are the most representative of this category. Due to its accuracy, The Smith-Waterman algorithm is one of the most widely used sequence alignment algorithm. Indeed, it is even used to refine the results of other less accurate. However, because it is also a computationally intensive algorithm; it requires more time compared to others. These are the reasons why we are particularly interested in the Smith-Waterman algorithm.

There exist several implementation architectures, from classic multi-cores to Special Application Specific Integrated Circuits (ASICs) [6], proposed to effectively accelerate the Smith-Waterman algorithm. The ASIC is able to provide a significantly low cost and low power consumption, but with limited scalability and not negligible design and fabrication times. As alternative, recent approaches focus on Graphic Processing Units (GPUs) [7] and Field Programmable Gate Array (FPGA) [8][9]. GPU-based solutions provide massive multi-threading for parallelize the Smith-Waterman algorithm. However, conflicting memory accesses, therefore sequential, become the bottleneck of the overall system. In 2012, K. Bankrid et al. explored the pros and cons of using FPGAs in bioinformatics [10]. Their results revealed that, compared to other platforms, FPGAs are generally a cost effective and energy efficient solution as it comes on top of both, performance criteria per dollar and per Watt. There are already some researches using FPGA-based platforms to accelerate the Smith-Waterman such as in [8], [9], and [11]. In [8] and [11], the whole Smith-Waterman algorithm (including trace-back) was implemented on an FPGA platform. However, parts of the algorithm spent a large amount of FPGA resources without a clear performance improvement. Moreover, in these designs, due to the sequential nature of certain tasks, such as the trace-back, the FPGA is not able to achieve the best performance of its inherent parallelism.

In [9], the authors propose a heterogeneous FPGA-based solution providing hardware acceleration to the algorithm running on a PC. However, the communication between the FPGA and the PC (through a bus, PCIe or USB) needs an additional time, which reduces the overall performance. However, thanks to the recent technological evolution, it is possible to have processor and FPGA together on the same chip. It combines the software programmability of a processor (also called *Processing System*) with the hardware programmability of an FPGA (also called *Programmable Logic*), resulting in unrivalled levels of system performance, flexibility and scalability, while providing system benefits in terms of power reduction, lower cost and fast time to market. It also greatly improves the communication efficiency between the FPGA and the host reducing the loss of time during data transfer.

In this paper, we propose a heterogeneous architecture for sequence alignment based on the Smith-Waterman algorithm. Our effort was oriented to optimize the original algorithm in several directions, size and number or processing resources, memory management, power consumption and processing speed.

3. THE SMITH-WATERMAN ALGORITHM

In the pairwise sequence alignment, two sequences, *Query* and *Reference* on Figure 1(a), are used to build a score matrix (Figure 1(b)) based on the following equation 1:

$$H_{i,j} = \max \begin{cases} 0 \\ H_{i-1,j-1} + w(\text{match/mismatch}), \\ H_{i,j-1} - d \\ H_{i-1,j} - d \end{cases} \tag{1}$$

Hi,j indicates the value at the cell position (*i,j*), *w* stands for the match/mismatch score, and *d* means the penalty for the gap. For instance, the value of *w* is 2 if two nucleotides are equal (*match*) and *d* is 1 if a gap (*mismatch*) occurs. According to the Smith-Waterman algorithm, the first column and row of the score matrix are filled with zeros, and other cells in the matrix are computed by the equation 1.

After computing the score matrix, tracing back starting from the highest value in the score matrix to the origin point, the gap or mutation will be exposed and the alignment result will come out. There might be several paths back to the origin; an alternative path appears whenever we find two directions with the same score (paths green and yellow on Figure 1(c)).

The generation of results indicates for each trace-back path an outgoing pair (*Q,R*) of aligned sequences (Figure 1(d)). A mutation or gap, indicated by âĂŸ-âĂŹ, appears on the sequence any time the trace-back is not following the right-to-left diagonal; it happens on the *Q* when the trace-back is going up and on *R* when going to the left (*Best* alignment pair on Figure 1(d)).

Figure 2 shows the execution times of the different parts of the algorithm when aligning two sequences with 200 nucleotides each. Among the steps performed by the algorithm, the most time consuming operation is the matrix calculation. Furthermore, by looking at the FPGA-based implementation, the *Back-tracking* and *Reverse sequences* operations implied a large amount of resources and execution time due to sequential memory accesses [8][11].

We evaluated the performance of the Smith-Waterman algorithm in a dual-core i5 processor according to the different lengths of the sequences. Figure 3 shows that when the sequences length increases, the time consumed by the calculation of the matrix increases exponentially, whereas other parts remain almost identical. This increase on execution time becomes critical even for a regular number of nucleotidesâĂŹ sequences. Thus, acceleration of the matrix calculation is the key point of the processing speed. With an FPGA-based implementation, we can optimize the use of memory resources, parallelize operations, and finally to achieve a better performance.

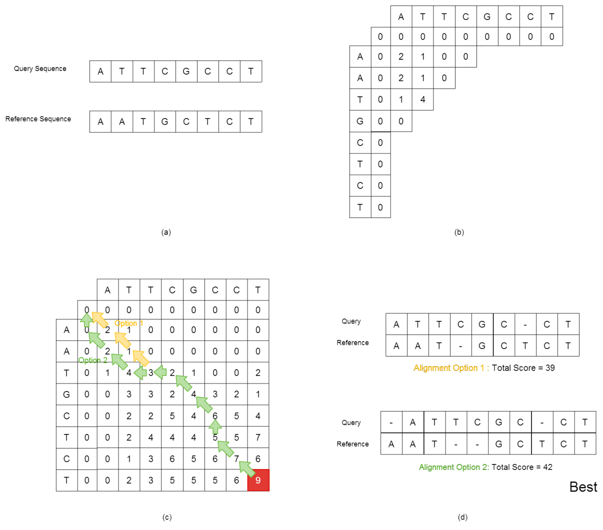

Figure 1: Illustration of Smith-Waterman Algorithm

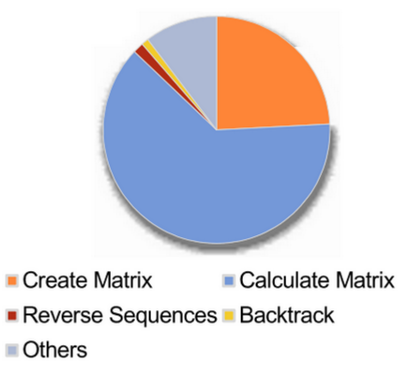

Figure 2: Percentage of Time-Consumption by Different Parts of Smith-Waterman on PC

4. ARCHITECTURE DESIGN

According to the profiling of the Smith-Waterman algorithm, the whole system is divided into two parts (see Figure 4): the first part is implemented in the *Processing system* (*PS*) or software region, including the *Initialization*, *Backtrack*, *Reverse* and *Result generation*; the most computational-intensive part, *Matrix calculation*, is implemented in the *Programmable logic* (*PL*) or hardware region. The blocks in the *PS* are written in C, while the *Matrix calculation* is in VHDL.

The *Initialization* block handles user requests. According

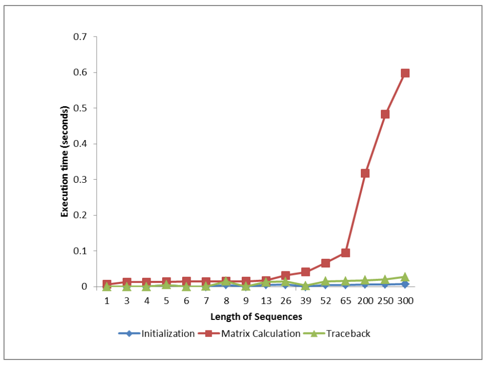

Figure 3: Execution time of Smith-Waterman's each step with different sequence length

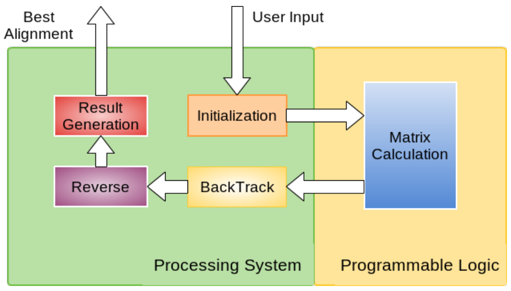

Figure 4: Block Diagram of Smith-Waterman on Heterogeneous Platform

to the length of sequences, it creates the memory spaces of the score matrix. In addition, it transforms the sequences from alphabet into binary code, the format which the *PL* region can use directly and efficiently. In this design, every nucleotide is represented in a 4-bit format, e.g. the letter '*A*' is mapped to '0000'. It also sends the query and reference sequences to the *Matrix calculation* block, responsible for building and performing the parallel computation of the score matrix. Each value in the matrix will be sent back to the *PS* region as soon as its calculation is completed. The *Backtrack* block finds out the possible alignment based on the data sent back, and picks out the one with the highest score. The *Reverse* block is dedicated to put the sequences in the right order. The *Result generation* block is used to generate the final results, such as the visualization of the aligned sequences, execution time, etc.

We paid attention to the implementation of the *Programmable logic*, since, for the overall system, it is the main contributor to the improved performance. The *Matrix calculation*, shown on Figure 5, uses a *FIFO* as *PS-PL* interface data exchange and Smith-Waterman processing elements (*PEs array*) managed by the *PE Controller*. The *FIFO* receives the sequences to be aligned, while the *PE Controller* distributes the workload to the Smith-Waterman *PE*s. The latter organizes the array data processing and workload distribution, according to the commands received from the *PS*. The array is a particular data processing organization, whose number of elements can be changed depending on the scale of sequences and FPGA resources available. A *PE* can treat incoming data in a few clock cycles.

There are many methods to map sequences to *PE*s; Figure 6 shows the conventional one. The first row and column of the score matrix are assigned zeroes. Due to the data-dependency, the diagonal marked numbers means that the values can be computed at the same time. In this figure, when a PE ends its calculation (yellow cells), it moves along the diagonal to compute the next neighboring cells. As long as there are *PE*s available, it will continue operating automatically with the cells all along the next diagonal (such as orange and purple cells show). In this way, we can exploit the full set of PEs all the time.

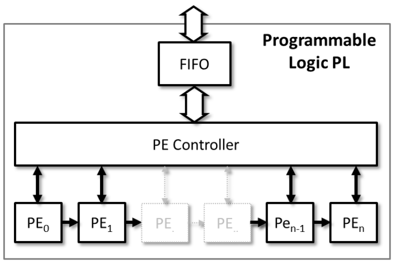

Figure 5: Block Diagram of Matrix Calculation in Programmable Logic

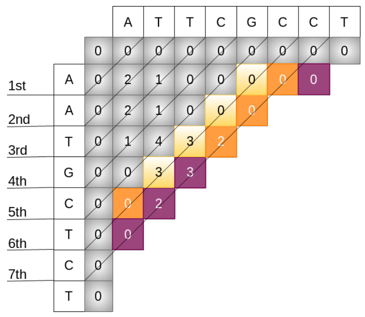

Figure 6: Conventional method to deploy Smith-Waterman PEs

The Smith-Waterman *PE*, whose block diagram is represented in Figure 7, computes the value of the score matrix by using just three adders and comparators (two are inside the *MAX* block). This operation takes only three clock cycles. The *PE controller* designates the pair of values to compute, and the result destination; back to the controller or to the next *PE* for further processing.

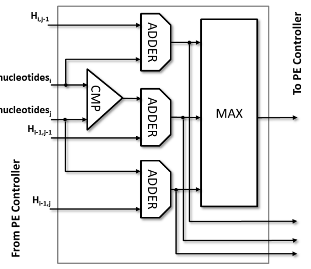

Figure 7: Block Diagram of Smith-Waterman Processing Element

5. PRELIMINARY RESULT

We implemented the sequence alignment system on a Zedboard. The ZedBoard contains a Zynq SoC (System-on-Chip) from Xilinx, which combines a dual-core ARM Cortex A9 processor (*PS*) with an FPGA (*PL*). For convenience, we used the Xillybus hardware/software interface to establish communication between the dual-core and the FPGA using two of the Xillybus's *FIFOs*. On that system, data payload bandwidth can easily reach 200MB/s on each *FIFO*. The *Matrix Calculation* block used for the test was composed by an array of 100 *PE*s.

Figure 8 shows the performance improvement when running two sequences with 5000 nucleotides each by applying the proposed FPGA-based heterogeneous architecture compared to a pure software implementation running on a dual-core i5 PC.

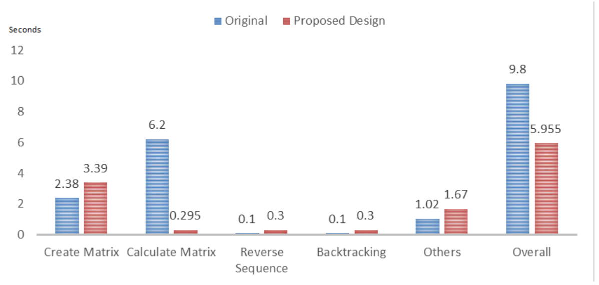

Figure 8: Performance of our design and conventional design based on dual-core

The execution time of the *Matrix Calculation* was significantly reduced from 6.2 to 0.295 seconds. However, since the operating frequency of the dual-core ARM A9 is lower than that of the Core i5 and the *PS-PL* communication time is included, the rest of the algorithm took longer than the Core i5. Nevertheless, the overall performance is improved. The execution time of the proposed architecture was reduced by 39.23%.

Table 1 shows the comparison of FPGA (*PL*) resources used by the different designs proposed in [9] and [12]. [9] implemented their architecture on two different FPGA platforms, and [12] only implemented on 2VP70 FPGA platform. As we can see in this table, our design, with 100 PEs, used the least amount of LUTs and registers. Especially, the registers' utilization only took 8.8% of all that the chip can provide. Thus, it is still possible to implement more PEs to achieve better performance.

Ref.	Device	PEs	Number of Slice LUTs(#)	Number of Registers(#)
[9]	2VP70-5	168	38,720(57.3%)	15.218(22.5%)
[12]	2VP70-5	138	65,964(99.7%)	no data
[9]	2V6000-4	168	31,790(57.3%)	12,488(18.9%)
Ours	XC7Z020	100	27,551(51.8%)	9,389(8.8%)

Table 1: Programmable Logic Resources Utilization Comparison

Our design is running at 66.7 MHz while [9] and [12] work under 39.2 MHz and 59.3 MHz, respectively. However, when comparing the performance of the alignment of two sequences with 5000 nucleotides each, the proposed design, even with less number of PEs, outperform [9] by 38.9% and [12] by 11.1%.

6. CONCLUSIONS AND FUTURE WORK

This paper proposed a novel FPGA-based heterogeneous platform for sequence alignment. Preliminary results show that it can provide better performance than a pure software multi-core solution, and with less resources utilization on the Programmable Logic compared to other approaches. However, there are still many issues to explore, for instance, power consumption including *PS-PL* communication, optimizing data transfer, to provide additional memory management operators, in particular supporting multiple users/requests/sequences and big data-bases, to finally come up with a fully flexible and powerful solution.

7. REFERENCES

[1] Polymerase Chain Reaction (PCR). `http://www.ncbi.nlm.nih.gov/genome/probe/doc/TechPCR.shtml`

[2] Probabilistic Consistency-based Multiple Alignment of Amino Acid Sequences. `http://probcons.stanford.edu/`

[3] T-Coffee Multiple Sequence Alignment Tools.

[4] The Basic Local Alignment Search Tool (BLAST) . `http://blast.ncbi.nlm.nih.gov/Blast.cgi`

[5] Biosequence analysis using profile hidden Markov models . `http://hmmer.janelia.org/`

[6] T. Han and S. Parameswaran, "SWASAD: an ASIC design for high speed DNA sequence matching," in Proceedings of ASP-DAC 7th Asia and South Pacific and the 15th International Conference on VLSI Design, pp.541-546, 2002

[7] L. Junjie, S. Ranka,S. Sahni,: "Pairwise sequence alignment for very long sequences on GPUs," in Proceedings of Computational Advances in Bio and Medical Sciences (ICCABS), pp.1-6, 23-25 Feb. 2012

[8] Scott Lloyd, Quinn O.Snell. Hardware Accelerated Sequence Alignment with Traceback. International Journal of Reconfigurable Computing, Volume 2009, January 2009 Article No. 9

[9] Yoshiki Yamaguchi, Kuen Hung Tsoi, Wayne Luk. FPGA-Based Smith-Waterman Algorithm: Analysis and Novel Design. 7th International Symposium, ARC 2011, Belfast, UK, March 23-25, 2011. Proceedings. Volume 6578 of Lecture Notes in Computer Science, pages 181-192, Springer, 2011.

[10] Khaled Benkrid, Ali Akoglu, Cheng Ling, Yang Song, Ying Liu, and Xiang Tian, âĂIJHigh Performance Biological Pairwise Sequence Alignment: FPGA versus GPU versus Cell BE versus GPP,âĂİ International Journal of Reconfigurable Computing, 2012.

[11] David Greaves,Satnam Singh, Sutirtha Sanyal, âĂIJSynthesis Of A Parallel Smith-Waterman Sequence Alignment Kernel Into FPGA Hardware,âĂİ Many-Core and Reconfigurable Supercomputing Conference, Berlin, Mar 2009

[12] Van Court, T.Herbordt and M.C, "Families of FPGA-based accelerators for approximate string matching" microprocessors & Microsystems 31, 135-145(2007).