HIGH LEVEL SYNTHESIS OF SMITH-WATERMAN DATAFLOW IMPLEMENTATIONS

S. Casale-Brunet[1*], *E. Bezati*[1*], *M. Mattavelli*[2]

[1]Swiss Institute of Bioinformatics, Lausanne, Switzerland
[2]École Polytechnique Fédérale de Lausanne, Switzerland

ABSTRACT

The paper presents the results of design explorations for the implementation of the Smith-Waterman (S-W) algorithm executing DNA and protein sequences alignment. Both design explorations studies and the corresponding FPGA implementations are obtained by writing a dynamic dataflow program implementing the algorithm and by direct high-level synthesis (HLS) to FPGA HDL. The main feature of the obtained implementation is a low-latency, pipelinable multistage processing element (PE), providing a substantial decrease in the resource utilization and an increase in the computation throughput when compared to state of the art solutions. The implementation solution is also fully scalable and can be efficiently reconfigured according to the DNA sequence sizes and to system performance requirements. The FPGA design presented in the paper can efficiently scale up to 250 MHz obtaining 14746 Alignments/s using a single S-W core with 4 PEs, and up to 31.8 Mega-Alignments/min using 36 S-W cores on the same FPGA for sequences of 160x100 nucleotides.

1. INTRODUCTION

Sequence alignment is an extensively used operation in bioinformatic data processing. Sequences of DNA, RNA, or proteins are aligned to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Among others, one important application is in the field of cancer identification. Cancer diseases are the results of changes occurred in the DNA sequence of the cells genomes [1]. These changes (also called aberrations) can be described as nucleotide substitutions, short insertions and deletions (indels), rearrangements and copy-number variations. The Smith-Waterman (S-W) algorithm [2] is a generalization of string matching algorithms that includes the computation of matching scores taking into account the insertions or deletion of parts of the string. The S-W algorithm can be used for identifying these aberrations since it is quite sensitive to identify most complex aberrations which instead result unrecognizable by using alternative heuristics provided by faster algorithms [3]. The essential drawback of the S-W algorithm when dealing with large amounts of genomic data versus reduced search heuristics is that it is much more demanding in terms of computations. The S-W algorithm performs local sequence alignment by executing two separate stages. The first is the computation of an optimal scoring matrix that presents an algorithmic complexity of $O(n \times m)$, where n and m are the lengths of the two sequences. The second stage is a traceback of the matrix which has an algorithmic complexity of $O(2 \times max(n, m))$. For the alignment of data yield by a modern sequencing machine of a single human biological sample, the entire S-W need to be executed several billion of times. Different implementation approaches have been proposed in literature to speed-up the entire analysis process. Among them, systolic arrays (SA) are a particular class of parallel processing architectures that can guarantee the best performance for embarrassingly-parallel applications [4]. For this reason, FPGA S-W implementations [5, 6, 7, 8, 9] generally outperform CPU and GPU implementations [10, 11, 12] in terms of cell updates per second (CUP). However, not all results appeared in literature provide a complete implementation of the two S-W stages, in almost all cases they only provide the evaluation of the scoring matrix leaving the traceback stage to a secondary processing platform. As consequence, the data transfer bandwidth between the two platforms might constitute the bottleneck of the final implementation strongly reducing the overall system throughput. In this paper the results of a complete FPGA S-W implementation including both the two computational stages is presented. The implementation solution has been obtained by describing the SA S-W algorithm using an high-level and dynamic dataflow program, analysing and exploring the system design space and successively synthesizing the FPGA solution by using an high level synthesis (HLS) framework developed by the authors. The results of the study are very interesting when compared to state of the art solutions since it is possible to reach 31.8 Mega-Alignments/min for sequences of 160x100 nucleotides on a single medium size FPGA. The paper is structured as follows. In Section 2 the S-W algorithm is introduced. Section 3 provides the description of the dataflow implementation that is successively synthesized using the HLS framework developed by authors. The results of the implementation are then discussed in Section 4. Section 5 concludes the papers summarizing the future works addressing an integrated solution

∗ This work has been carried on when the authors were at the SCI-STI-MM of École Polytechnique Fédérale de Lausanne, Switzerland.

usable in a complete genome analysis pipeline.

2. SMITH-WATERMAN ALGORITHM

Smith-Waterman algorithm (S-W) performs a local alignment of two sequences of RNA, DNA or protein. The first string $A = \{a_1, a_2, \ldots, a_n\}$ is generally referred to as the reference, and the second one $B = \{b_1, b_2, \ldots, b_m\}$ as the read (or query). The S-W is composed of two computational stages.

2.1. Scoring matrix evaluation

The first stage consists on arranging the two sequences in a matrix $H \in \mathbb{R}^{m+1,n+1}$ in which n and m are the sizes of the two sequences. For each $1 \leq i \leq n$ and $1 \leq j \leq m$ the values of H are evaluated as:

$$H[i][j] = max \begin{cases} H[i-1][j-1] + s(i,j) \\ H[i-1][j] + \delta \\ H[i][j-1] + \delta \\ 0 \end{cases} \quad (1)$$

where $s(i,j)$ is a similarity function of the alphabet Σ of the two sequences and δ is the gap cost. The four lines of Equation (1) can de described as follow: the first represents a match or mismatch, the second a deletion, the third an insertion. It should be noted that the fourth imposes that the H matrix can not contain negative values: this fact makes S-W a local alignment algorithm [2]. Boundaries of H are generally:

$$\begin{array}{rcl} H[0][j] & = & 0, \;\; 0 \leq j \leq n \\ H[i][0] & = & 0, \;\; 0 \leq i \leq m \end{array} \quad (2)$$

2.2. Traceback Procedure

The second stage is finding the best local alignment between the two sequences by a traceback procedure along the scoring matrix. Starting from the highest value of $H[i][j]$, referred to as *score*, the matrix is traversed choosing at each time the neighbor of the current matrix element used to construct the matrix (i.e. the one that maximizes Equation 1). The traceback procedure stops when a zero value is reached.

2.3. Systolic Array Configuration

From Equation (1) it can be seen that the values of each element $H[i][j]$ depends only on its left, up and diagonal neighborhoods. As consequence, the evaluation of the entire scoring matrix can be efficiently performed diagonal by diagonal, since each diagonal depends only on the result of the previous one. Using a systolic array (SA) configuration [4], a set of n processing elements (PE) is sufficient to fully parallelize this computation.Each character a_i is assigned to the PE_i processing element. Conversely, each character b_j is assigned to all the PEs during the j-th execution of each PE. If the evaluation of each element $H[i][j]$ requires p clock-cycles (CLKs), then the entire evaluation of H is performed in $p \times max(n,m)$ CLKs instead of $p \times n \times m$ CLKs as required by a non-SA implementation.

3. DATAFLOW IMPLEMENTATION OF THE S-W ALGORITHM

A dataflow program can be represented as a directed graph where each node represents an execution kernel, called actor, and each directed edge represents a first-in first-out (FIFO) queue, called buffer. Actors encapsulate their own state which cannot be shared among the other actors of the graph. Atomic data objects, called tokens, are stored in each buffer and used to exchange information among actors: tokens represent the only possible form of synchronization and data exchange between actors. The execution and activation of actors are governed by a given set of rules which is usually referred to as the program model of computation (MoC). The Dataflow Process Networks (DPN) [13] is a dynamic MoC frequently used to specify a wide variety of signal processing applications [14, 15, 16]. Each DPN actor evolves by performing sequences of discrete computational steps. The standard dataflow programming language [17] called RVC-CAL has been used to implement the S-W algorithm. RVC-CAL is a dataflow language which fully captures the behavioral features of DPN model of computation adding the notion of discrete and atomic firings. Each firing function is called *action*. Each firing of an action is enabled by conditions on the actor internal variables, on the availability of input tokens and on the actual values of the tokens. For a complete description of the language the interested reader can refer to [17, 18]. In the following, two RVC-CAL dataflow implementations of the SA S-W are reported. The first is referred as *static* implementation in which the sequence sizes are known at-priori. The second is a *dynamic* implementation in which the sizes are not know and can vary from one alignment to another. Since the sequence sizes are not generally known at-priori, only the synthesis results of the dynamic configuration are reported in the rest of the paper.

3.1. Static Implementation

Using the dataflow formalism, the S-W algorithm with a SA configuration illustrated in Section 2 can be modeled as a graph of interconnected actors. Fig. 1 depicts a RVC-CAL static implementation with 4 PEs. As mentioned before, a static SA configuration can be used only if the sequence sizes are known at-priori. For this particular example, the reference can not be longer than 4 nucleotides. However, the design can be easily extended to support longer sequences by introducing additional PEs.So as, a S-W RVC-CAL design with n PEs is composed by $n + 2$ actors: n PEs actors, a con-

troller actor and an aligner actor. Due to the limited space of this paper, the RVC-CAL source code of the actors can not be provided. However, their main functionality is summarized in the following. Each **PE** actor can be seen as the main component used to evaluate the elements of the scoring matrix H. Equation (1) is solved each time that this actor is executed. The result is stored in a local variable and successively sent to the output port *V*. It should be noted that the values of $H[i-1][j]$ and $H[j-1][i-1]$ (i.e. up and diagonal neighbors, respectively) can be retrieved from the actor internal state since $H[i-1][j]$ is the value of *v* and $H[j-1][i-1]$ the value of the left neighbor obtained from the previous firing of the action. The actor internal state is reset when an end of sequence (EOS) character is received. The **Controller** actor parses the two input sequences that are then sent to the PEs. Each a_j is sent $m+1$ times to each PE_j, while each b_i is sent once to each PE. An extra EOS character is added at the end of each sequence B in order to reset all the PEs internal states, making possible to immediately start new alignments. The **Aligner** actor stores all the scoring matrix element values computed by the PEs. Furthermore, its computes the best local alignment of the two sequences by implementing the S-W traceback procedure. The score value and the corresponding matrix indexes are updated each time that a new PE value is received: this is done to avoid introducing extra latency in the overall algorithm.

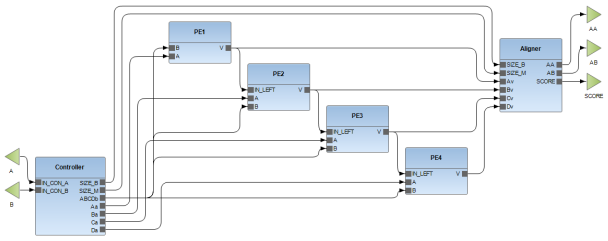

Fig. 1: RVC-CAL dataflow Systolic Array Smith-Waterman design: static configuration.

3.2. Dynamic Implementation

The *static* design described above requires to be modified to support the possibility of aligning sequences with variable and unknown sizes. The objective is to maintain the same functionality, but, at the same time, make possible to align sequences for which the reference string size is $n > PEs$ and both n and m can vary between two successive alignment. This result can be obtained by partitioning the scoring matrix H in $c\lceil\frac{PEs}{n}\rceil$ column sub-matrices, where $\lceil x\rceil = min\{n \in \mathbb{N} : n \geq x\}$ is the ceiling operator. The reference sequence A is divided in c sub-sequences, each of size PEs such as $A_1 = \{a_1, a_2, \ldots a_{PEs}\}$, $A_2 = \{a_{PEs+1}, a_{PEs+2}, \ldots a_{2PEs}\}$, $A_x = \{a_{xPEs+1}, a_{xPEs+2}, \ldots a_{xPEs}\}$. Each PE is then executed $m \times c$ times. Fig. 2 reports the graph of the S-W

	Available	Usage	
BRAM	3456	94	2.72%
FF	1075200	2468	0.23%
LUT	537600	3056	0.57%

(a) Overall resource utilization

Actor	BRAM		FF		LUT	
Aligner	66	1.91%	439	0.04%	591	0.11%
Controller	2	0.06%	220	0.02%	323	0.06%
LeftBuffer	0	-	68	0.01%	107	0.02%
PE1	0	-	188	0.02%	248	0.05%
PE2	0	-	188	0.02%	248	0.05%
PE3	0	-	188	0.02%	248	0.05%
PE4	0	-	188	0.02%	248	0.05%

(b) Resource utilization of each RVC-CAL actor

Table 1: Synthesis result for the dynamic SA S-W dataflow program depicted in Fig. 2.

dataflow program with a dynamic SA configuration that supports the possibility of size-varying sequences. Compared to the static design described above, an additional actor called *LeftBuffer* has been added. The actor is responsible of sending to PE_1 the scoring values evaluated by PE_{PEs}. This buffered-stream oriented implementation does not require that PE_1 have to directly access the scoring matrix to retrieve the sub-matrices border values. In other words, sub-matrices border values are streamed by a buffer without breaking the PEs execution pipeline. It should be noted that also the semantic of the *Controller* actor had to be accordingly modified to handle the splitting of the reference sequence into multiple sub-sequences.

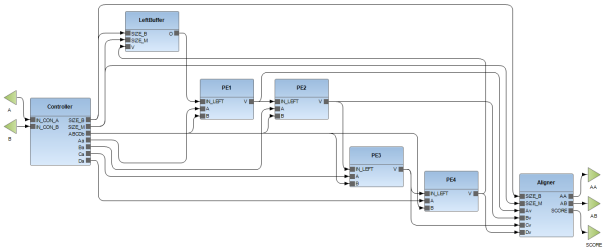

Fig. 2: RVC-CAL dataflow Systolic Array Smith-Waterman design: dynamic configuration.

Design	Traceback	HLS	Max. PEs	Freq. (MHz)	CUPS
[5]	no	no	174	324.1	56×10^9
[6]	no	no	384	66.7	25.6×10^9
[7]	no	no	1778	45	80×10^9
[8]	no	no	252	55	13.9×10^9
[9]	no	yes	398	83	33×10^6
Dynamic RVC-CAL	yes	yes	144	250	36×10^9
Static RVC-CAL	no	yes	1200	250	300×10^9

Table 2: Design performance in terms of GCUPS compared to other state of the art implementations.

4. EXPERIMENTAL RESULTS

The RVC-CAL dynamic S-W SA algorithm illustrated in Fig. 2 has been synthesized in a Virtex Ultrascale (xcvu095) FPGA. The Xronos HLS framework [19, 20, 21] has been used to generate a synthesizable HDL code of the dataflow program consisting of a single dynamic SA S-W core containing 4 PEs, as the one depicted in Fig. 2. Among others possible designs employing more PEs, such configuration has

been found to be the best trade-off between FPGA resource utilization and final throughput. In fact, with a complete implementation of the S-W such as the one described here, increasing the number of PEs can accelerate the evaluation of the scoring matrix H, but the system bottleneck becomes the aligner which can sensibly reduce the overall throughput (i.e. see [22, 23]). Consequently, the design choice has been to evaluate the best trade-off between resource utilization of a single S-W core (i.e. a single implementation of the dataflow network depicted in Fig. 2) and the number of parallel S-W cores that can be placed in a single FPGA. The direction matrix has been stored together with the scoring matrix H to make possible the evaluation of the traceback procedure. Each matrix element $H[i][j]$ has been stored in 16 bits: 13 bits are reserved for the scoring value, 2 bits are used to store the direction, and 1 bit is used to mark a perfect match between a_j and b_i. The maximum nucleotides sequence size for both the reference and the read string supported by the described design has been set to 256. Such choice is motivated by the decision of storing the matrix directly on the FPGA, without using an external DDR which would have implied to reduce the overall throughput in terms of Alignments/s. It should be observed that this is only a dataflow program configuration parameter that could be easily modified before HDL synthesis to support longer sequence sizes. Finally, the synthesized implementation has been validated by a post-synthesis RTL behavioral simulation. Resource utilization of a single S-W core with 4 PEs synthesized with a clock frequency of 250MHz are summarized in Table 2. The set of nucleotides sequences have been encoded using 3 bits for each character, using as alphabet $\Sigma = \{A, C, G, T, -, EOS\}$, where $-$ and EOS are two special characters. The first is used for marking an unknown nucleotide in the reference/read sequence or an indel (insertion/deletion), the second is used to divide two nucleotide sequences when streamed to/from the board. Using this design solution, is has been possible to achieve the result of 14746 Alignments/s for a set of standard sequences of 160x100 nucleotides. According to Table 2, the maximal number of S-W cores that could be placed in the selected FPGA platform is 36 cores with 4 PEs each. With such implementation, it is possible to achieve a peak of 31.8 Mega-Alignments/min. In literature, when dealing with the SA S-W FPGA implementation, the general trend is to provide a solution only for the first stage of the S-W algorithm, completely neglecting the traceback procedure. The scoring matrix is sent through IO interfaces to an external platform (generally a CPU or a GPU). However, for long nucleotide sequences, this approach can compromise the overall throughput performance given that the IO interface bandwidth can easily become the system bottleneck. Thus, the paper provides a solution including both the two S-W stages (i.e. scoring matrix evaluation and traceback as described in Section 2). For this reason, the comparison with other state-of-art implementations has been done in terms of cell updates per second (CUPS). CUPS are a performance measure commonly used in computational biology that represents the time for a complete computation of one element of the scoring matrix H (see Equation (1)). CUPS performance can be measured as $CUPS = f \times PEs$, where f is the clock frequency of the design. Table 2 compares the performance in terms of CUPS of some state-of-art implementations with the one reported in the paper (denoted as RVC-CAL). As it can be seen, only the one implemented in this paper implements the S-W traceback procedure. In this case, the inputs of the system are the two unaligned sequences (i.e. the reference and the read) and the outputs are the two aligned sequences and the highest score value of H. Results for two different configurations of the RVC-CAL SA S-W design are reported in Table 2: the first is the dynamic SA S-W with 36 cores with 4 PEs each, and the second is the dynamic SA S-W with 1 core containing 1200 PEs. With these two configurations, it is possible to achieve 36 GCUPS and 300 GCUPS respectively. However, as mentioned before when augmenting the number od PEs in a single S-W core the bottleneck becomes the aligner. As a consequence, the throughput in terms of Alignment/s of the first configuration is 31.8 Mega-Alignments/min, conversely with the second is around 1500 Alignments/s which is very similar to the throughput of a single S-W core with 4 PEs.

5. CONCLUSION

A complete FPGA SA S-W design obtained by describing the algorithm with an high-level dataflow representation and then by directly synthesizing the FPGA implementation is reported. The dataflow program has been analyzed, system design options have been explored and the final solution synthesized by using an HLS framework developed by the authors. Results of the final implementation outperform state of the art solutions since it is possible to achieve 31.8 Mega-Alignments/min for standard sequences of 160x100 nucleotide. Moreover, the current design provides a complete implementation of both the two stages composing the S-W algorithm: the evaluation of the score matrix and the traceback procedure. The output of the design are the aligned sequences, not only the score matrix as usually provided in literature. The main benefit of the new design solution is the possibility of integrating into a single FPGA chip the entire alignment process, without dealing with IO interface bandwidth limitations faced by implementation in which the traceback procedure is performed by an external platform (generally CPU or GPU). Further work will focus on integrating the S-W cores with the IO interfaces to provide a final on-chip implementation. An IO bandwidth that guarantees a throughput of 31.8 Mega-Alignments/min can be estimated at about 0.39 Gb/s using 3 bits for representing each base or 1.03 Gb/s using 8 bits for representing each nucleotide base, solution which may lead to consider higher end FPGA platforms supporting even much higher throughput for the final system designs.

6. REFERENCES

[1] M. Stratton, P. Campbell, and P. Futreal, "The cancer genome," *Nature*, vol. 458, no. 7239, pp. 719–724, 2009.

[2] T. Smith and M. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[3] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing," *Briefings in bioinformatics*, vol. 11, no. 5, pp. 473–483, 2010.

[4] H. Kung, "Why systolic architectures?," *Computer*, vol. 15, no. 1, pp. 37–46, Jan 1982.

[5] G. Causapruno, G. Urgese, M. Vacca, M. Graziano, and M. Zamboni, "Protein alignment systolic array throughput optimization," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 23, no. 1, pp. 68–77, 2015.

[6] P. Zhang, G. Tan, and G. Gao, "Implementation of the Smith-Waterman algorithm on a reconfigurable supercomputing platform," in *Proceedings of the 1st international workshop on High-performance reconfigurable computing technology and applications: held in conjunction with SC07*. ACM, 2007, pp. 39–48.

[7] L. Hasan, Y. Khawaja, and A. Bais, "A Systolic Array Architecture for the Smith-Waterman Algorithm with High Performance Cell Design," in *IADIS*, 2008.

[8] T. Oliver, B. Schmidt, and D. Maskell, "Hyper customized processors for bio-sequence database scanning on FPGAs," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. ACM, 2005, pp. 229–237.

[9] S. Singh and D. Greaves, "Kiwi: Synthesis of FPGA circuits from parallel programs," in *Field-Programmable Custom Computing Machines, 2008. FCCM'08. 16th International Symposium on*. IEEE, 2008, pp. 3–12.

[10] Q. Zhang, H. An, G. Liu, W. Han, P. Yao, M. Xu, and X. Li, "The optimization of parallel Smith-Waterman sequence alignment using on-chip memory of GPGPU," in *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*. IEEE, 2010, pp. 844–850.

[11] P. Pham, T. Duong, N. Ta, C. Tran, D. Nguyen, T. Nguyen, and H. Le, "Applying GPUs for Smith-Waterman Sequence Alignment Acceleration," *GSTF Journal on Computing (JoC)*, vol. 1, no. 2, 2014.

[12] J. Singh and I. Aruni, "Accelerating smith-waterman on heterogeneous cpu-gpu systems," in *Bioinformatics and Biomedical Engineering,(iCBBE) 2011 5th International Conference on*. IEEE, 2011, pp. 1–4.

[13] E. Lee and T. Parks, "Dataflow Process Networks," in *Proceedings of the IEEE*, 1995, pp. 773–799.

[14] M. Mattavelli, J. Janneck, and M. Raulet, "MPEG Reconfigurable Video Coding," in *Handbook of Signal Processing Systems*, S. Bhattacharyya, E. Deprettere, R. Leupers, and J. Takala, Eds., pp. 43–67. Springer US, 2010.

[15] M. Mattavelli, "MPEG reconfigurable video representation," in *The MPEG Representation of Digital Media*, L. Chiariglione, Ed., pp. 231–247. Springer New York, 2012.

[16] E. Jang, Mattavelli M., M. Preda, M. Raulet, and H. Sun, "Reconfigurable media coding: An overview," *Signal Processing: Image Communication*, vol. 28, no. 10, pp. 1215–1223, 2013.

[17] ISO/IEC 23001-4:2011, "Information technology - MPEG systems technologies - Part 4: Codec configuration representation," 2011.

[18] J. Eker and J. Janneck, "CAL language report: Specification of the CAL Actor Language," Technical Memo UCB/ERL M03/48, Electronics Research Laboratory, University of California at Berkeley, Dec. 2003.

[19] E. Bezati, *High-level synthesis of dataflow programs for heterogeneous platforms*, Ph.D. thesis, EPFL IEL STI, Lausanne, 2015.

[20] E. Bezati, S. Casale-Brunet, M. Mattavelli, and J. Janneck, "Synthesis and optimization of high-level stream programs," in *Electronic System Level Synthesis Conference (ESLsyn), 2013*, May 2013, pp. 1–6.

[21] "Xronos," `http://github.com/orcc/xronos`, Online; Accessed December. 2016.

[22] S. Casale-Brunet, *Analysis and optimization of dynamic dataflow programs*, Ph.D. thesis, EPFL IEL STI, Lausanne, 2015.

[23] A. Ab Rahman, R. Thavot, S. Casale-Brunet, E. Bezati, and M. Mattavelli, "Design space exploration strategies for FPGA implementation of signal processing systems using CAL dataflow program," in *Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on*. IEEE, 2012, pp. 1–8.