A FPGA-Based Hardware Implementation of Generalized Profile Search Using Online Arithmetic

Emeka Mosanya
Emeka.Mosanya@epfl.ch

Eduardo Sanchez
Eduardo.Sanchez@epfl.ch

Logic Systems Laboratory,
Swiss Federal Institute of Technology,
INN Ecublens, CH-1015 Lausanne, Switzerland.

Abstract

This paper describes the hardware implementation of the Generalized Profile Search algorithm using online arithmetic and redundant data representation. This is part of the GenStorm project, aimed at providing a dedicated computer for biological sequence processing based on reconfigurable hardware using FPGAs. The serial evaluation of the result made possible by a redundant data representation leads to a significant increase of data throughput in comparison with standard non redundant data coding.

1 Introduction

In the context of the various genome sequencing projects, we are concerned with the hardware implementation of specific algorithms which intend to find structural or control units of elementary physiological processes in gene and protein sequences. The method used consists of two distinguishable but operationally interdependent components. The first one is a descriptor of such units of elementary physiological processes which is called motif; the second is a search method used to locate instances of a already defined motif in a particular sequence. In a typical application, a new sequence of unknown function is compared against a database of many known motifs. There are several methods used to model a particular motif. They are based, for example, on weight-matrices [1], on regular expressions [13] or on Hidden Markov Models [3].

This article will focus on the Generalized Profile defined by the Bioinformatic Group of the Swiss Institute for Experimental Cancer Research (ISREC) [4][5]. This technique has several advantages: it is sufficiently general to combine the majority of motif techniques developed earlier, the method has been mathematically and unambiguously defined and, software tools accessible by network are available and are used by the scientific community on a daily routine. A Generalized Profile is described using a well defined syntax [6] and the process to evaluate if a given sequence contains a subsequence corresponding to a given profile has also been defined. The result is the pfscan utility freely available by ftp [1] and the ProfileScan WWW Server [2].

We propose an hardware implementation of the pfscan utility in order to improve the response time offered. This implementation target the GenStorm machine [7], a FPGA-based accelerator for biological sequence processing. GenStorm is far to be the first hardware accelerator designed this kind of application. Several machines have been developed. Some systems are using custom non-reprogrammable VLSI chip design enabling some degree of parameterization like the BioSCAN [8], the P-NAC machines [9], or the machine proposed by D. Lavenier [10], or the Fast Data Finder from Paracel[3]. Other systems are using FPGAs as main processing units, for example, the SPLASH machine [11] and the SPLASH2 machine [12], or the Bioccelerator developed by Compugen[4]. But none of them implements algorithm to process sequences against Generalized Profiles. Furthermore, the complexity of the algorithm they implement is lower than the complexity of the Generalized Profile Search algorithm. To obtain significative performance improvement, we use serial operators and a redundant data format.

In the first part of this paper, we will briefly describe the Generalized Profiles and how to apply them on a query sequence. Then we will deal with the mapping of the problem in a systolic architecture and we will present an implementation of the algorithm using classical data representation and processing. Next, we introduce the online arithmetic and the redundant data representation. We present a new implementation using them and compare the two solutions.

[1] http://www.isrec.isb-sib.ch/ftp-server/pftools
[2] http://www.isrec.isb-sib.ch/software/PFSCAN_form.html
[3] http://www.paracel.com
[4] http://www.compugen.co.il

2 Generalized Profile

2.1 Definition

The definition of a Generalized Profile and the complete mathematical description of the process of aligning such profile with a query sequence is given in [5]. A General Profile is intended to model a family of sequences or subsequences (Figure 1). In a biological context, the characters involved are called residues and are nucleotides or amino-acids. As suggested by the figure, the structure of a Generalized Profile consists of an alternating sequence of match (M/D) and insert (I) nodes, called the profiles positions, starting and ending with an insert position. The number of node depends directly of the sequences modeled by the profile. Several parameters, called scores, are associated to each position. Each match position is associated to the following score values:

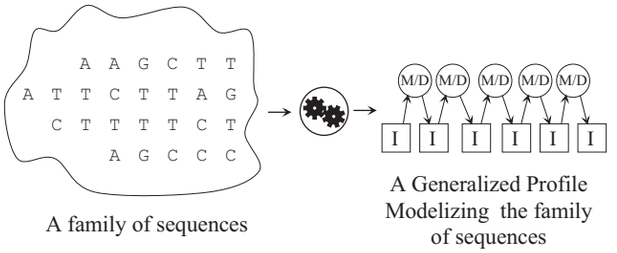

Figure 1: A Generalized Profile as a model of several sequences.

- $m(a)$: Match score for character a,
- d: Delete extension score,

and each insertion position is associated to the following score values :

- $i(a)$: Insert score for residue a,
- b: Initiation score at position y,
- e: Termination score at position y,
- $t_{1..16}$: 16 Transition scores.

where a is a residue of the query sequence and y is its position.

The value of these parameters are computed during a modeling phase where the profile is tuned to represent a given family of sequences the best as possible. This article does not deal with evaluation of the parameters themselves and it is supposed that such profile is already available. Actually, the known protein profiles are listed in the PROSITE database [13].

2.2 Alignment between a Generalized Profile and a sequence

We can verify if a query sequence or one of its subsequences corresponds to the family modeled by a given Generalized Profile. This can be achieved by computing the best alignment between the profile and the sequence. The method involved uses the same principles that the alignment between two sequences using the Smith and Waterman algorithm [14]. The evaluation of the best alignment between a Generalized Profile and a sequence will use a dynamic programming optimization process to maximize the value of the score associate to each possible alignment. The Figure 2 gives an example of such alignment. An insert position is placed between each match position.

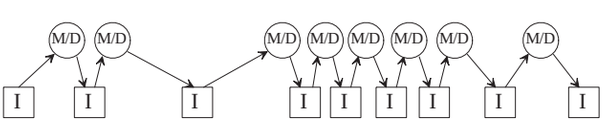

Figure 2: Alignment of a Generalized Profile against a sequence.

We can calculate the score associated to an alignment. It depends of the score values associated with each position of the profile and the query sequence. To evaluate the alignment score, we simply run through the profile, from the left to the right, and add every significant position score as following:

- Each match position is placed in front of one residue of the query sequence or in front of one dash, respectively to account for the correspondence between one character of the sequence with one expected by the profile at the current position or to signify that the query sequence has a deleted character compared to what is expected by the profile. In the first case, the match score $m(a)$ depending of the character a is added to the profile score and in the other, the delete extension score d is added.
- An insert position is placed between each match position. If it is the first position of the alignment the initiation score b is added to the profile score. If it is the last position of the alignment, the ending score e is added to the profile score. An insertion score $i(a)$ is added for each character inserted between the two characters associated to the adjacent match positions. In addition, a transition score $t_{1..16}$ is added to the calculated profile score to quantify the transition between matching, inserting or deleting phases as well as the beginning and the ending step in the alignment process.

This transition score will be defined more precisely below.

An alignment can be represented by a path through a table as on Figure 3. A valid path begins at any dot and is composed of dots whose coordinates are equal or greater than coordinates of the preceding. The Figure 4 gives a closer view of such dot. It includes 8 nodes, 4 inputs and 4 outputs. Six nodes (M, I, D, M^+, I^+D^+) are associated with the path through the table between the beginning and terminating node. The nodes B and E^+ are respectively the node associated with the initiation and the ending of a path. In other words, we get in a path through the node B and get out through the node E^+. All edges are weighted with the score value associated with the corresponding profile M/D or I node.

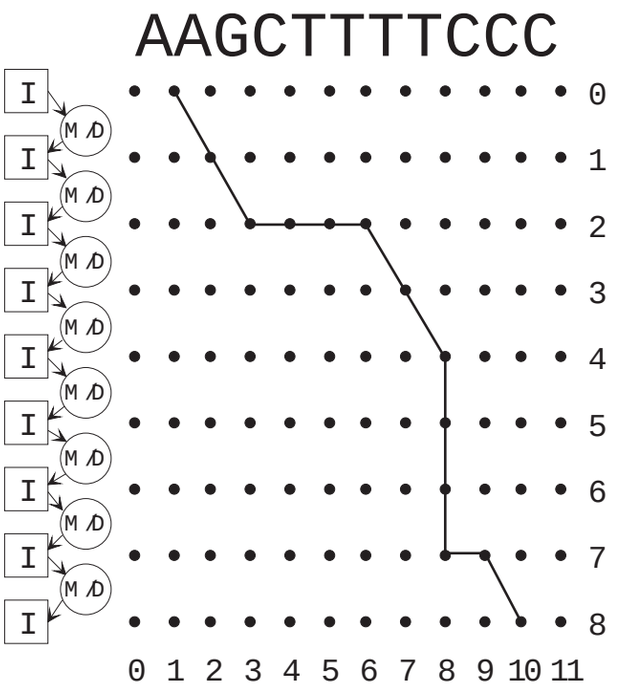

Figure 3: Alignment as a path through a table.

We can compute the score for an alignment between a Generalized Profile and a sequence by following one path in the table of Figure 3 and adding up the scores corresponding to the edge weights: the initiation score, the match score or the delete score for each match position, a transition score and, if necessary, an insert score for each insert position, and the termination score.

To compute the optimal alignment score, we use a dynamic programming method, derived from Smith and Waterman [14] and described in [5], to evaluate exhaustively the score associated with all possible paths. For a profile of length n and a query sequence of length m, we evaluate the score of all paths and we chose the one resulting in the highest value. If we define $s(V)$

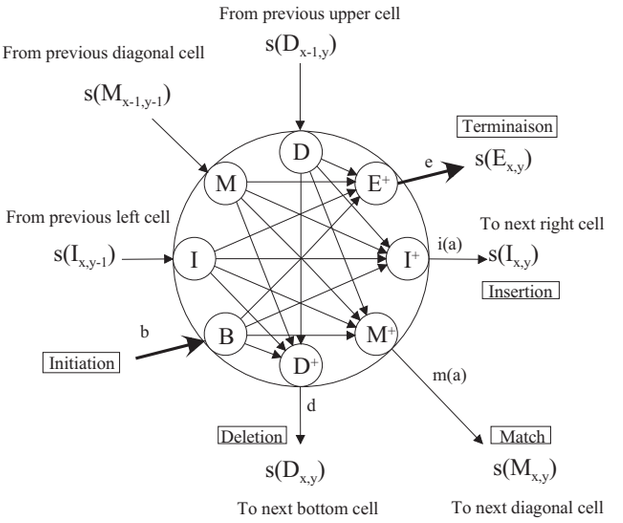

Figure 4: A close view of a insert and a match positions and the related scores.

as the best score from the initiation of the path to a vertex V, $V \in \{B, M, I, D, E^+, M^+, I^+, D^+\}$, then the highest score is given by

$$S_{max} = \max_{0 \leqslant x < n, 0 \leqslant y < m} [s(E_{x,y})] \quad (1)$$

.

The local score $s(V_{x,y})$ with $V \in \{B, M, I, D, E\}$ defined in [5] can be represented with the dataflow diagram of Figure 5 if we consider the following:

- $V \in \{B, M, I, D, E\}$,
- $v_x(a_x, y) \in \{e_x(y), m_x(a_y), i_x(a_y), d_x\}$,
- a_y and y respectively as the current residue and its position in the query sequence,
- $b_{0,t_{B,V}} = b^e_x + t_{B \to V,x}$,
- $b_{y,t_{B,V}} = b^i_x + t_{B \to V,x}$,
- $s(M_{x,y}) = -\infty$ if $x = 0$ or $y = 0$,
  $s(I_{x,y}) = -\infty$ if $x = 0$,
  $s(D_{x,y}) = -\infty$ if $y = 0$.

This completely defines the process of finding the best alignment between a profile and a query sequence. We are therefore ready to describe its hardware implementation.

3 Mapping of the problem to a systolic architecture

We intend to implement the optimal score evaluation process with a systolic architecture. A systolic architecture is a parallel architecture using several similar

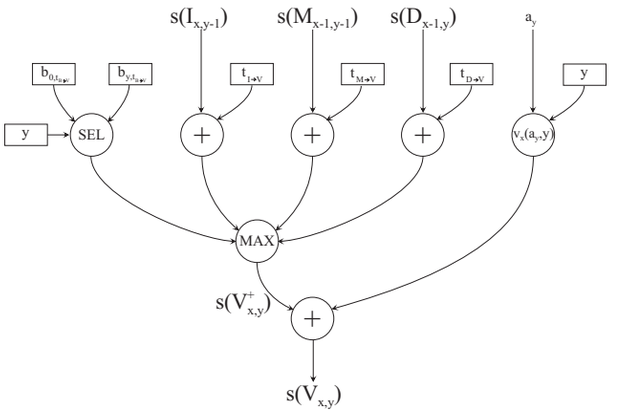

Figure 5: Dataflow diagram of local score computation.

processing elements (PEs) locally connected and exchanging their data synchronously. This architecture is well suited for this kind of dynamic programming based pattern matching problem [15] [16].

To obtain PEs with identical functionalities and with a minimal number of interconnections, they have to include the vertices $B, M, I, D, E^+, M^+, I^+, D^+$. We will also need to include the computation associated with the outgoing edges of these vertices. Therefore, the data circulating between the PEs correspond to the values $s(I)$, $s(M)$, $s(D)$ and the partial evaluation of $s(E)$. Once the smallest PE has been defined, we can, if need be, group several of them in a larger one. The two possible values of $s(B_{x,y})$ are locally memorized and only depend on the position of the PE. The S_{max} value is a global maximum which is the result of local maximum computation inside each PE. The result of this computation, the local maximum, is transmitted through the local connection between PEs.

Therefore, the implementation of the calculation of optimal alignment score between a sequence and a Generalized Profile has the same structure than the simpler problem of aligning two sequences [11]. We can use the uni-dimensional architecture described in [12] which uses a linear network of processing elements, each assigned to the evaluation of a profile position. If the number of PEs is smaller than the profile length, the sequence has to flow several time through the PEs which are successively initialized with the next set of profile position parameters. The resulting line-architecture adapted to the Generalized Profile Search evaluation is given in Figure 6.

As explained above, the value $s(I)$, $s(M)$, $s(D)$ and the partial evaluation of $s(E)$ are transmitted between the PEs, along with the current residue and some control signals.

- Since the $s(I_{x,y})$ value flows horizontally, its value is kept inside the same PE, and, after a delay cycle, is used as input for the I vertex.
- Since $s(D_{x,y})$ value flows vertically, its value is sent directly to the next PE as input to the D vertex and is used for the next cycle.
- Since $s(M_{x,y})$ value flows diagonally, its value is sent to the next PE and is used as input to the M vertex two cycles later.
- The maximum operator computes partial values of the S_{max} global maximal score.

The controller responsible of the initialization of the PEs, the transfer of the input data, the output of resulting values, and the segmentation of sequences if the problem is too large for the number of processors, is not described here.

4 Architecture of the GenStorm machine

The GenStorm architecture is composed by several cards linked together by a VME bus: a SPARC VME card, a dedicated disk controller, and one or several computing card(s)[7]. The SPARC card, running UNIX operating system, is the main controller of the machine. It configures the computing cards, controls the data content of the disks and their formatting, starts a calculation and reads the result, and allows network clients access to the calculator. This network capability also allows the client to remotely reconfigure the GenStorm machine.

The architecture of the calculation cards, called Genome, is shown in the Figure 7. Its general structure is similar to a lot of other existing cards since we did focus on the target application part of the problem and use a corresponding standard architecture. This card contains nine FPGAs of the Xilinx XC4000 family, eight processing units (PUs) and one controller unit (CU), a VME interface, and memory blocks. Each PU is connected to a local memory (512KB) and to its neighbours with a 30 bit-wide local connection. The CU is connected to two of the PU and to the memory blocks. It is responsible for the control of the memory blocks, i.e. for ensuring a correct sharing of the memory blocks between the Genome card and the VME bus and for the generation of interruption. The card is designed to carry out the subdivision of large query in hardware without communication with the host. That property is its main originality.

5 Parallel implementation of the score evaluation

In the rest of this document, we will focus on the implementation of the datapath of the processing elements and, in particular, the local score calculation. The

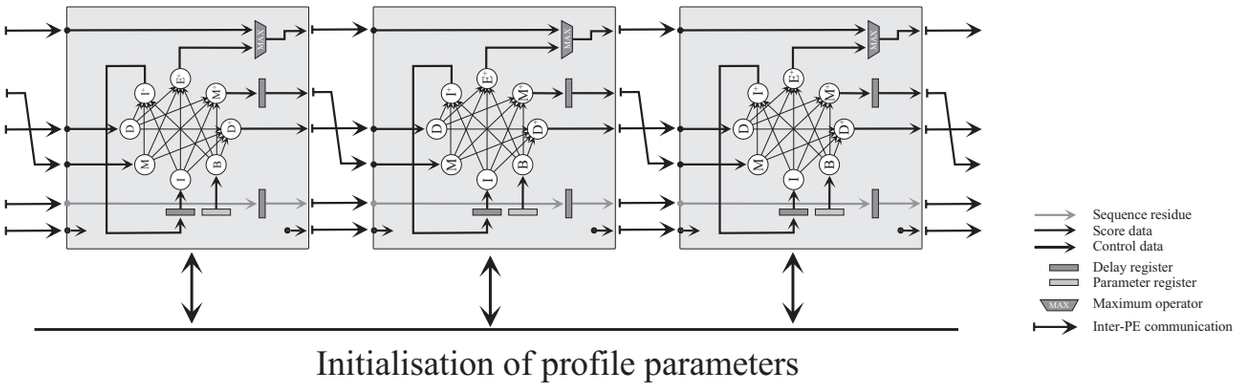

Figure 6: Line unidimensional architecture implementation of the sequence flow problem.

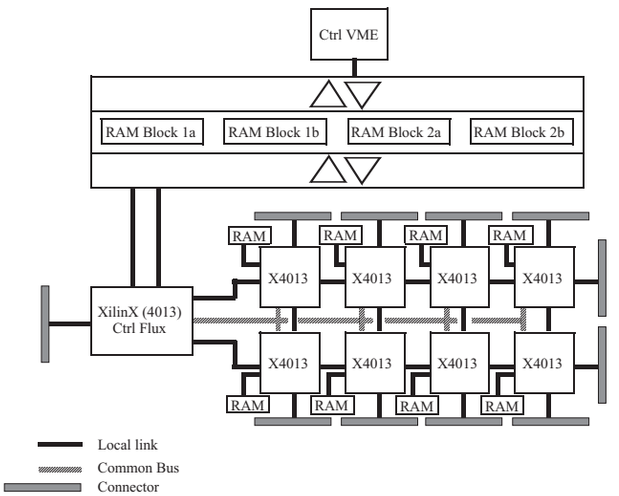

Figure 7: The architecture of the Genome card.

global score calculation consists simply of reading in the flow of local maxima and memorizing the current value if it is larger than the preceding one.

The proposed scheduling is mainly based on input port timing constraints. Since we aim at achieving the highest possible throughput, the scheduling does not take into consideration FPGAs resource constraints and we suppose we can use a device as big as we need.

The resulting scheduling is shown in Figure 8. In this figure, the grayed operators and registers correspond to the dataflow for the local score evaluation. More precisely, they correspond to the computation of the $s(E_{x,y})$,$s(I_{x,y})$,$s(M_{x,y})$, and $s(D_{x,y})$ values, as in Figure 5, where respectively $V \in \{E, M, I, D\}$ and $v_x(a_x, y) \in \{e_x(y), m_x(a_y), i_x(a_y), d_x\}$. This grey sub-dataflow corresponds therefore to four sub-dataflows giving four distinct results, each memorized in one of the striped registers.

We observe that the score calculation can be carried out in 6 clock cycles and uses the resources described in Table 1.

Resource	Qty	Input	Reuse	Desc.
Adder	4	2	4	Adder
Max2	4	2	3	Max
Sel2	4	2	1	Select

Table 1: Resources used during score calculation.

The Input and Reuse columns in Table 1 are used to calculate the quantity and the size of the multiplexer needed to select the input value of the corresponding operator. The resources needed to memorize the profile parameters in each PE are the following:

- twenty-six 8-bit registers for transfer, initialisation and termination scores,
- two 20x8-bit memory for match and insert scores,
- one 10-bit query sequence character counter,

The local score computation has been implemented in VHDL following the schedule presented above. It has been compiled with the Synopsys synthesizer with an input/output port size of 28 bits, a score size of 16 bits, a profile parameter size of 8 bits, and a sequence length size of 10 bits. All values are integers. This configuration is realistic and can be used in the everyday life of a biology laboratory. The target is the Xilinx XC4000 family, since they are used on the GenStorm machine. The results are given in Table 2.

We can observe that the score calculation is very resource intensive. This is a result of the parallelism used in the selected schedule. The four local score calculations are carried out in parallel, which involves the use of four adders and four maximum operators. This last operation used most of the resource and is very time consuming.

From the result of the synthesis, we can estimate the performance of our system. The longest datapath

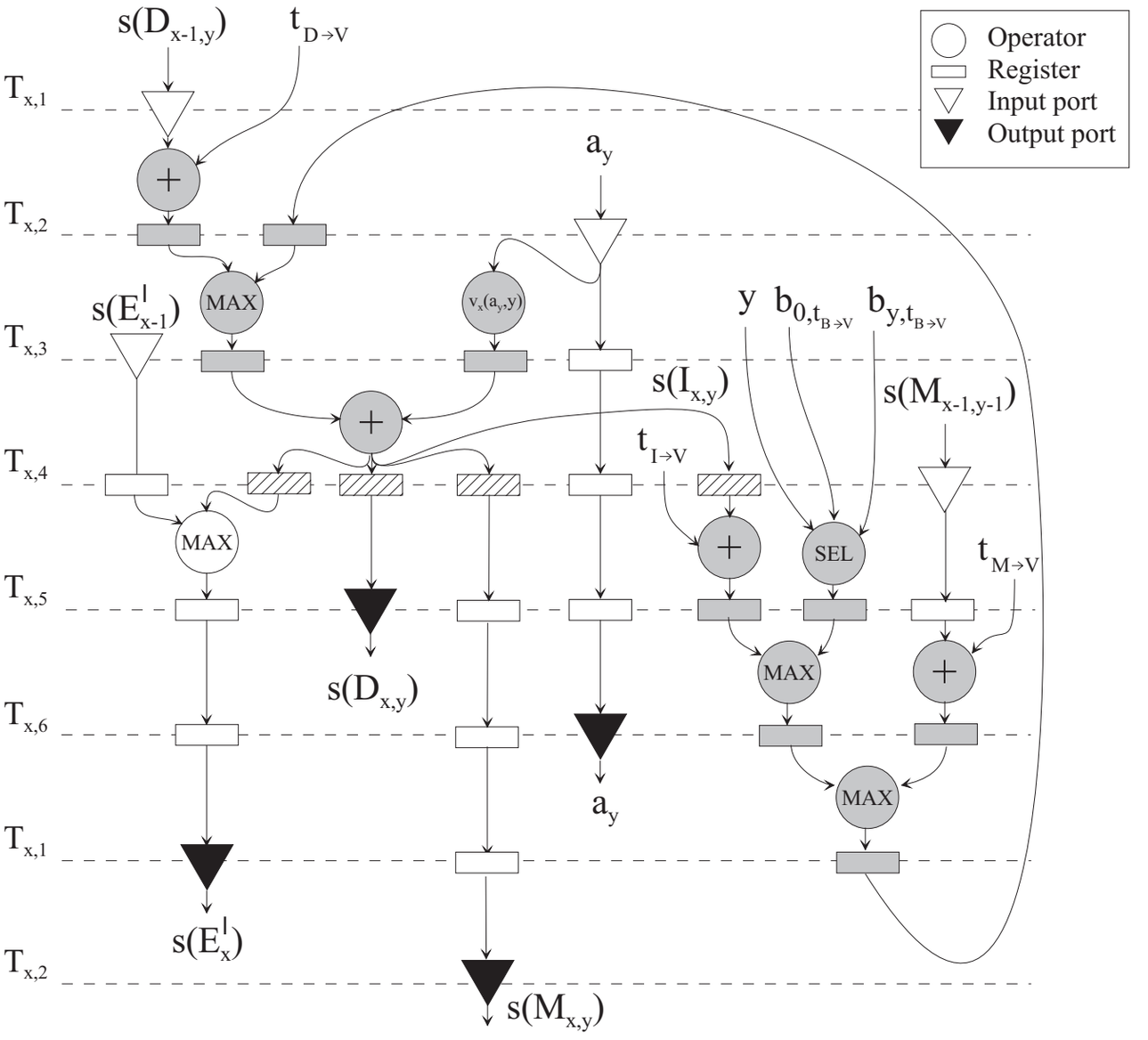

Figure 8: Scheduling diagram of score computation.

delay is 238 ns with a Xilinx speed grade of -6, which is a very slow device but currently used on the processing card. This path corresponds to the path through the maximum operator which is a critical component of the implementation. Nevertheless, even with a slow device, one computing card of the GenStorm machine using a parallel implementation of the score calculation is able to compare more than 5.6 millions profile positions by second on a GenStorm card which include 8 PEs. This is to compare to the average of 600 thousand profile positions by second evaluated by a SPARC 20 machine. The system has been designed to support 16 cards.

resource	Quantity
FG Function Generator	1132
Number of Flip Flops:	607
Total Number of CLBs:	620

Table 2: Resource used by the local score calculation after VHDL synthesis.

6 Online operators

Even if the speedup of the parallel implementation of the score calculation is promising, we were targeting better results. In order to increase the throughput, we needed to make a better use of pipelining. We took the radical decision to use serial operators which will give us a bit-level pipelining. This is possible thanks to the ability of these operators to output the first bit of the result before to have received all its inputs.

Online operators use algorithms which satisfy the "online" property [17]. This implies that to generate the j^{th} digit of the result, it is necessary and sufficient to have the operands available up to the $(j+\delta)^{th}$ digit, where the index difference δ is a small positive constant. It is necessary to accumulate δ initial digits of the operands in order to produce the first digit of the result. Subsequently, one digit of the result is produced upon receiving one digit of each operands. δ is called the online delay which is misleading since it corresponds to a latency. An online operator can also be characterized by τ, the clock period, i.e. the time needed by the signal to cross through the longest path of the circuit (Figure 9).

Serial data Transmission may be executed least significant digit first (LSDF) or most significant digit first (MSDF). Algorithms for LSDF transmission seem to be more natural because they correspond to classical "paper and pencil" methods [18]. However, this approach has several disadvantages. Operations such as division, square root, or simply the maximum operator produce outputs in MSDF form. Therefore a sequence involving these operations cannot be performed without large delays between successive operators to transform these outputs to LSDF form (Figure 10). When we use redundant number representation, all operators can produce their outputs in MSDF and formatting is not necessary anymore.

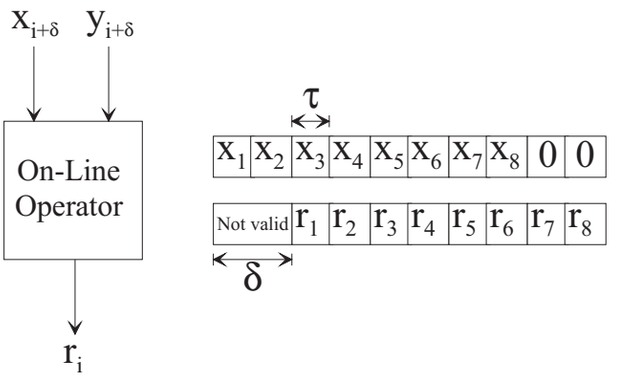

Figure 9: Online operator associated with its online delay δ.

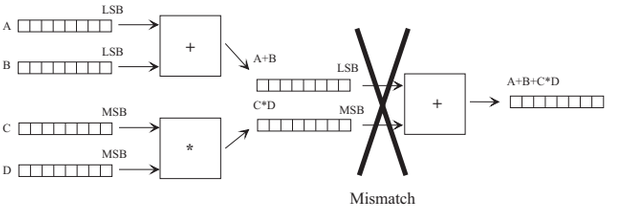

Figure 10: Mismatch of the operand orientation.

Since the δ value associated with each operator is small compared to the number of bit of their operands, it is possible to start the second of two successive operations before the first is completed. This property enables bit-level pipelining. For example, in Figure 11, the first bit of the result is evaluated after four clock cycles.

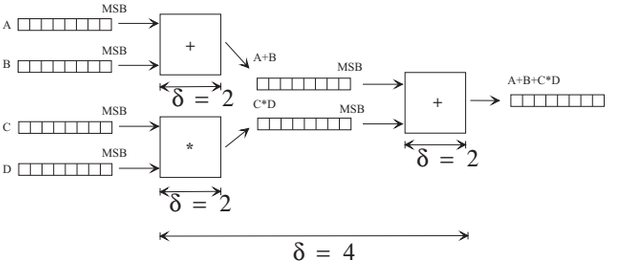

Figure 11: Composition of several online operator and resulting δ value.

6.1 Redundant representation

To perform an operation like the addition with the most significant digit first, we need a data representation which does not need carry propagation. A solution is to use a redundant number system. In a conventional number representation with an integer radix $r > 1$, each digit is allowed to assume exactly r values: $0, 1, ..., r-1$. In a redundant representation with the same radix r each digit is allowed to assume more than r values. Both positive and negative digit values are allowed for this purpose. Redundancy in the number representation allows, for example, a method of fast addition and subtraction in which each sum (or difference) digit is the function only of the digits in two or three adjacent digit positions of the operands. The time duration of the operation is independent of the length of the operands.

There is several different type of redundant data representation as for example the borrow-save representation described in [19] or the signed-digit representation described by A. Avizienis in [20]. In our current implementation, we use the last one which consists in a positional, constant radix representations of algebraic values

$$Z = \sum_{i=1}^{n} z_i r^{n-i}, \tag{2}$$

in which the digits z_i assume one of the following set of values

$$-a, -(a-1), ..., -1, 0, 1, ..., a-1, a \tag{3}$$

$$\frac{1}{2}(r_o + 1) \leqslant a \leqslant r_o - 1 \text{ or } \frac{1}{2}r_e + 1 \leqslant a \leqslant r_e - 1 \tag{4}$$

where r_o is an odd integer $r_o \geqslant 3$, and r_e is an even integer $r_e \geqslant 4$. The redundancy of a signed-digit representation is minimal when $a = \frac{1}{2}(r_o+1)$ or $a = \frac{1}{2}r_e+1$, and the redundancy is maximal when $a = r_o - 1$ or $a = r_e - 1$. For our first implementation, we have chosen $r = 4$ and $a = 3$.

From this representation, we can define the addition and the maximum operators we need for the implementation of General Profile Search. Other operators such the multiplication and the division can be found in [17].

6.2 Signed-digit adder

The arithmetic operations of totally-parallel addition and subtraction of two digits two numbers X and Y

$$S = \sum_{i=1}^{n} s_i r^{n-i}, \tag{5}$$

$$X = \sum_{i=1}^{n} x_i r^{n-i}, \qquad Y = \sum_{i=1}^{n} y_i r^{n-i}, \tag{6}$$

where $s_i, x_i, y_i \in \{-r+1, ...-1, 0, 1, ..., r-1\}$, n is the number of digits, is defined as following:

$$s_i = t_i + w_i, \tag{7}$$

with t_i and w_i defined as following:

$$z_i + y_i = r t_{i-1} + w_i. \tag{8}$$

The Figure 12 shows the organization of the adder in its parallel form. We can observe the absence of carry propagation. The Figure 13 is the corresponding online adder. As said earlier, the operands flow through the operator with the MSB first. The online delay δ is equal to 1.

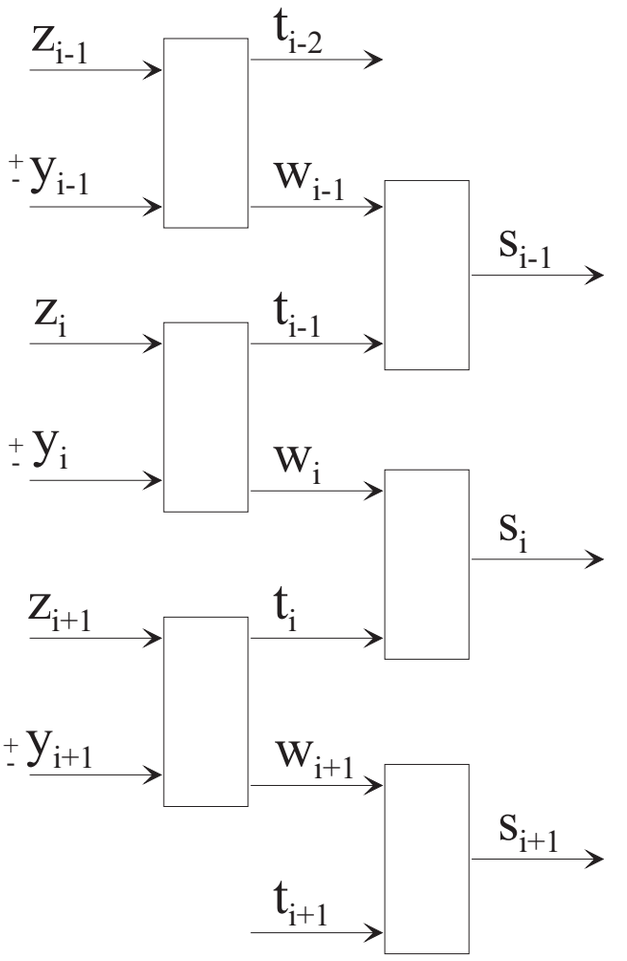

Figure 12: Parallel version of the signed-digit adder.

The value t_i is called transfer digit by Avizienis as opposed to carry since it can assume both positive and negative values and it is never propagated past the first adder position on the left. The reader can find the

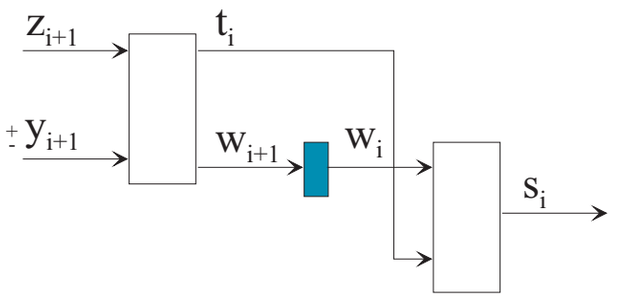

Figure 13: Online version of the signed-digit adder.

complete discussion in Avizienis' article [20] as well as conversion procedures between conventional (sign and magnitude) representation to a signed-digit representation of the same radix $r > 2$.

6.3 Online maximum operator

We want to evaluate the following function of two input values A and B :

$$Y = MAX(A, B) \tag{9}$$

where A,B, and Y are integer values represented in signed-digit number system:

$$Y = \sum_{i=1}^{n} y_i r^{n-i}, \tag{10}$$

$$A = \sum_{i=1}^{n} a_i r^{n-i}, \qquad B = \sum_{i=1}^{n} b_i r^{n-i}, \tag{11}$$

where $y_i, a_i, b_i \in \{-r+1, \ldots -1, 0, 1, \ldots, r-1\}$, n is the number of digits. Let us denote A_i, B_i, Y_i, the numbers constituted by the most i significant digits of A, B, Y :

$$Y_i = \sum_{j=1}^{i} y_j r^{n-j}, \tag{12}$$

$$A_i = \sum_{j=1}^{i} a_j r^{n-j}, \qquad B_i = \sum_{j=1}^{i} b_j r^{n-j}, \tag{13}$$

and let us require that after the i-th step (for any $i = 1, 2, \ldots, n$) :

$$Y_i = MAX(A_i, B_i) \tag{14}$$

This last condition allows us to evaluate i digits of Y knowing i digits of A and B. Let us define R_i as

$$R_i = rR_{i-1} + a_i - b_i, \tag{15}$$

then

$$y_{i+1} = \begin{cases} a_{i+1} & \text{if } R_{i+1} \geqslant 0 \text{ and } R_i \geqslant 0 \\ a_{i+1} + rR_i & \text{if } R_{i+1} > 0 \text{ and } R_i < 0 \\ b_{i+1} - rR_i & \text{if } R_{i+1} < 0 \text{ and } R_i > 0 \\ b_{i+1} & \text{if } R_{i+1} \leqslant 0 \text{ and } R_i \leqslant 0 \end{cases}, \tag{16}$$

By defining $R_0 = 0$, $a_0 = 0$ and $b_0 = 0$, Equations 15 and 16 give the recursion to evaluate all the digits of $Y = MAX(A, B)$. The Figure 14 gives us a representation of the corresponding hardware implementation. The online delay δ is equal to 0.

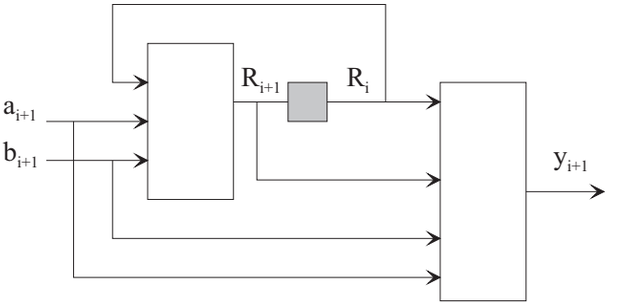

Figure 14: Online version of the maximum operator.

6.4 Normalization

In the general case, if we add two n-digit numbers A and B, the result S will be $(n+1)$-digit long. Each addition/subtraction can add one digit, not only as the result of possible overflow or underflow but also because of the redundant representation itself [21]. For example, using a radix 4 signed-digit, the number 22 can be represented by the number $1\overline{2}2$, adding one digit. This property can become an issue if we successively evaluate many additions, as it is the case in the Generalized Profile Search algorithm.

We have to ensure that any result has the same number of bits than the operands thanks to a normalization operator. If a value can not be represented with the available number of digit as a result of underflow or overflow, this operator will respectively output the smallest or the largest defined value.

Let us consider a radix-r signed-digit representation and a number of digits equal to n. Let us consider the $n+1$-digit number $A = \sum_{i=1}^{n+1} a_i r^{n+1-i}$ to be normalized. The resulting value $Y = NORM(A) = \sum_{i=1}^{n} y_i r^{n-i}$ is a n-digit number defined by

$$Y = \begin{cases} -r^n + 1 & \text{if } -r^n \geqslant A \\ A & \text{if } -r^n < A < r^n \\ r^n - 1 & \text{if } r^n \leqslant A, \end{cases} \tag{17}$$

where y_i and $a_i \in \{-r+1, \ldots -1, 0, 1, \ldots, r-1\}$.

Clearly, the trivial case where $a_1 = 0$ results in $a_i = y_{i+1}$ for $i = 1, 2, \ldots, n$. If $a_1 < 0$ then $A < 0$ and we

have to ensure that A is not smaller than the smallest representable n-digit number. Similarly, if $a_1 > 0$ then $A > 0$ and we have to ensure that A is not greater than the largest representable n-digit number. We can write these conditions as following :

$$Y = \begin{cases} MAX(-r^n + 1, A) & \text{if } a_1 < 0 \\ A & \text{if } a_1 = 0 \\ MIN(r^n - 1, A) & \text{if } a_1 > 0 \end{cases} \quad (18)$$

This normalization process can be implemented with a maximum and a minimum operator. Only two cases need conversion and have to be considered based on the value of the most significant bit a_1 of the number A. If the most significant bit is negative, then A is negative and the result of the normalization process is the largest value between A and the smallest representable value. If the most significant bit is positive then A is positive and the result is the smallest value between A and the largest representable value (Equation 18).

7 Serial implementation of the score calculation

We have all the elements to build an signed-digit online implementation of the score evaluation. The Figure 15 is an attempt to represent such implementation. A register is placed between each operator to reduce cycle length. The height of the operator is proportional to their online delay plus a unit value corresponding to the added registers. The online delay δ of the score evaluation corresponds to the delay to obtain the first digit of the $s(D_{x,y})$ value and is equal to 5. The implementation uses a signed-digit representation with a radix $r = 4$. In order to work with the same score value range than the parallel implementation which uses binary representation, we have to use scores with minimum 8 digits in base 4.

After implementation in VHDL and synthesis, we can also estimate the performance of the online version of the score evaluation. The longest path delay is about 23 ns with the same Xilinx device, that is an order of magnitude faster than the parallel implementation. Because of the serial flow of data, 9 clock cycles are needed compared to 6 previously. Nevertheless, the online implementation is able to compare more than 38 millions profile positions by second on a GenStorm card which include 8 PEs, that is a speed-up of about 7 on the parallel implementation.

The online implementation uses 10 percent less CLBs than the parallel one. This is due to the high number of FIFOs which have been synthesized without taking advantage of the optimized Xilinx library.

8 Conclusion

In this paper, we have described the Generalized Profile Search and we have shown that it can be implemented with a systolic architecture on GenStorm, a FPGA-based computer dedicated to sequence processing. We have proposed two implementations, the first one using classical parallel operators and the second one using serial operators and a different data representation called signed-digit number.

We have demonstrated that the online solution has a speed up of 7 on the parallel implementation with some gain in area consumption. This result shows that the serial operation enabled by the redundant number representation is an advantageous strategy to improve the performance of datapaths on a FPGAs. The serial evaluation of the result, which seems counter productive at first view, leads to spectacular performance improvement thanks to bit-level pipelining and the reduction of routing complexity. This method is very promising as an online implementation exists for every usual operators.

References

[1] M. Gribskov, A.D. McLachlan, and D. Eisenberg. Profile analysis: Detection of distantly related proteins. Proceedings of the National Academy of Science of USA, 84, 1987.

[2] Amos Bairoch. The prosite dictionary of sites and patterns in proteins, it curent status. Nucleic Acids Research, 21, 1993.

[3] Kevin Karplus. Using markov models and hidden markov models to find repetitive extragenic palindromic sequences in escherichia coli. Technical Report UCSC-CRL-94-24, University of California, Santa Cruz, 1994.

[4] Philipp Bucher and Amos Bairoch. A generalized profile syntax for biomolecular sequence motifs and its function in automatic sequence interpretation. In Proceedings of the 2nd ISMD Conference. AAAI Press, 1994.

[5] Philipp Bucher, Kevin Karplus, Nicolas Moeri, and Kay Hofmann. A flexible search technique based on generalized profiles. Computers and Chemistry, 20, 1996.

[6] Philipp Bucher. A generalised profile syntax for protein and nucleic acid sequence motifs. Technical report, Swiss Institute for Experimental Cancer Research, 1066 Epalinges s/Lausanne, 1997.

[7] Jean-Michel Puiatti. Genome : documentation de la carte. Technical report, EPFL-Logic Systems Laboratory, INN Ecublens, 1015 Lausanne, 1995.

[8] C. Thomas White, Raj K. Singh, et al. Bioscan: A vlsi-based sytem for biosequence analysis. In IEEE International Conference on Computer Design: VLSI in Compters and Processors. IEEE Computer Society Press, 1991.

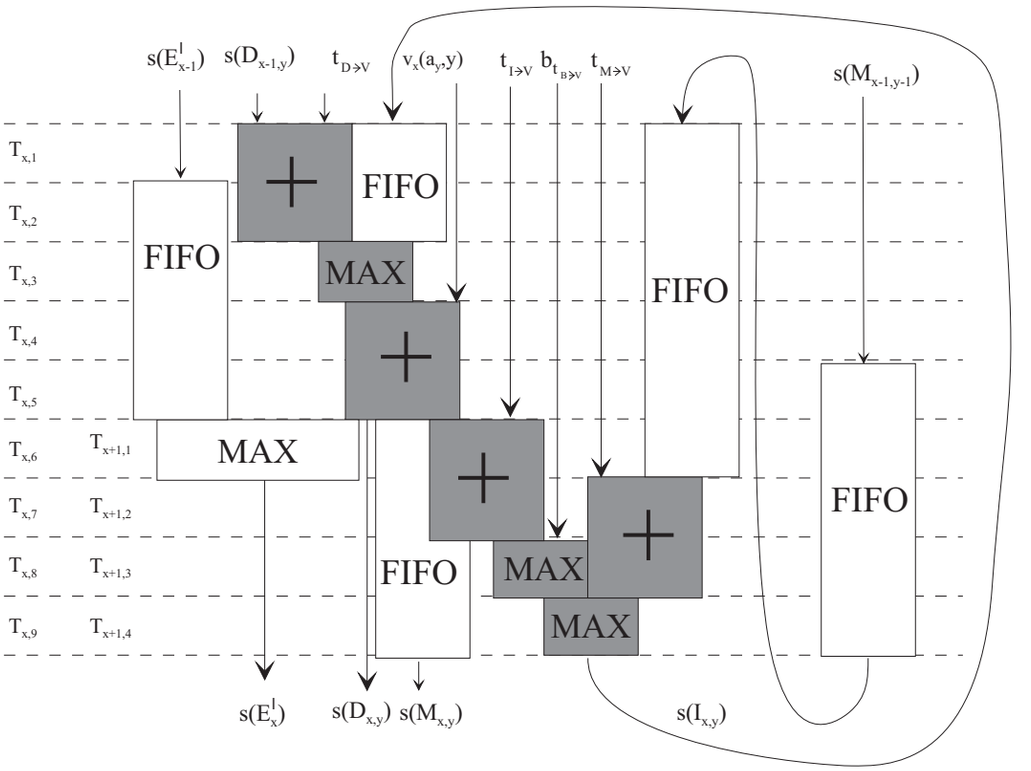

Figure 15: Online version of the score evaluation.

[9] Daniel P. Lopresti. P-nac: A systolic array for comparing nucleic acid sequences. IEEE Computer, July 1987.

[10] P. Guerdoux-Jamet and D. Lavenier. Systolic filter for fast dna similarity search. In International Conference on Application Specific Array Processor, Strasbourg, France, July 1995.

[11] Daniel Philip Lopresti. Rapid implementation of a genetic sequence comparator using field-programmable logic arrays. In Advanced Research in VLSI 1991, UC Santa Cruz, 1991.

[12] Dzung T. Hoang. Searching genetic databases on splash 2. In IEEE Workshop on FPGAs for custom computin machines, Napa, California, pages 185–191. IEEE Computer Society Press, Apr 1993.

[13] Amos Bairoch, Philipp Bucher, and Kay Hofmann. The prosite database, its status in 1997. Nucleic Acids Research, 24, 1997.

[14] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. Journal of Molecular Biology, 147, 1981.

[15] Patrice Quinton et Yves Robert. Algorithmes et architectures systoliques. Masson, 1989.

[16] Daniel Philip Lopresti. Discounts for dynamic programming with applications in VLSI processor arrays. PhD thesis, Faculty of Princeton University, 1987.

[17] Kishor S. Trivedi and Milos D. Ercegovac. On-line algorithms for division and multiplication. IEEE Transaction on Computers, C-26(7), July 1977.

[18] Richard Hartley and Peter Corbett. Digit-serial processing techniques. IEEE Transaction on Circuits and Systems, 37, 1990.

[19] J.C. Bajard, J. Duprat, S. Kla, and J.M. Muller. Some operators for on-line radix-2 computations. Journal of Parallel and Distributed Computing, 22, 1994.

[20] A. Avizienis. Signed-digit number representation for fast parallel arithmetic. IRE Transactions on Electronic Computers, 10, 1961.

[21] Arnaud Tisserand and Martin Dimmler. Fpga implementation of real-time digital controllers using on-line arithmetic. In Wayne Luk, Peter Y.K. Cheung, and Manfred Glesner, editors, Field-Programmable Logic and Applications. Proceedings of the 7th International Workshop, FPL'97,London, number 1304 in Lecture note in computer science, pages 472–481. Springer, Sep 1997.