SAMBA : Systolic Accelerator for Molecular Biological Applications

Dominique Lavenier

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

SAMBA
Systolic Accelerator for Molecular Biological Applications

Dominique LAVENIER

N 2845

March 1996

THÈME 1

ISSN 0249-6399

SAMBA
Systolic Accelerator for Molecular Biological Applications

Dominique LAVENIER

Thème 1 — Réseaux et systèmes

Projet API

Rapport de recherche n° 2845 — March 1996 — 22 pages

Abstract: SAMBA is a full custom systolic array dedicated to the comparison of biological sequences. This hardware accelerator implements a parameterized version of the Smith and Waterman algorithm allowing the computation of local or global alignments with or without gap penalty. The speed-up provided by SAMBA over standard workstations ranges from 50 to 500, depending on the application.

Key-words: biological sequence comparison, Smith and Waterman algorithm, dedicated hardware, linear systolic array

(Résumé : tsvp)

This work was partially funded by the French Researh Group GREG (Groupement de Recherches et d'Etudes sur les Génomes) and the French Coordinated Research Program ANM (Architectures Nouvelles de Machines)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

SAMBA
Accélérateur Systolique pour Applications en Biologie Moléculaire

Résumé : Ce document présente SAMBA: un réseau systolique linéaire dédié à la comparaison de séquences biologiques. Cet accélérateur matériel implémente une version paramétrisée de l'algorithme de Smith et Waterman, autorisant ainsi l'obtention d'alignements locaux ou globaux avec ou sans coût d'insertion/omission. L'accélération des calculs par rapport à une station de travail standard se situe entre 50 et 500, suivant les applications.

Mots-clé : comparaison de séquences biologiques, algorithme de Smith et Waterman, matériel spécialisé, réseau systolique linéaire

1 Introduction

SAMBA[1] is a hardware accelerator designed for speeding up the algorithms involved in biological sequence comparison. Computations which require several hours on standard workstations are performed in a few tens of seconds on SAMBA.

SAMBA implements a parameterized Smith and Waterman algorithm [14] [7]. By setting differently a few parameters, local or global comparisons can be performed, with or without gap penalty. Thus, a variety of software, such as BLAST [1], FASTA [12] or SSEARCH [13], may be implemented on that accelerator.

The complete SAMBA system comprises a workstation, a systolic array of 128 full custom hardwired 12-bit processors, and a FPGA-based interface (see Fig. 1) The FPGA interface is the PeRLe-1 board developed by Vuillemin et al. [3]; it acts as a hardware programmable driver for the systolic array.

SAMBA may be compared with two other hardware systems – BISP [4] and BIOSCAN [17] – also designed to speed-up biological sequence comparison with full-custom chips. Both systems include a systolic array made out of a linear arrangement of dedicated processors. The BISP and BIOSCAN prototypes contain respectively 256 16-bit processors and 12 000 1-bit processors.

The architecture of BISP and SAMBA are similar. However, they differ on the way the array is supplied with data: the BISP array is driven with a programmable processor (Motorola 68020) while SAMBA uses FPGA technology. This last approach is simpler and ensures that the high data throughput required by the systolic array is sustained.

BIOSCAN does not support dynamic programing algorithms. Like BLAST, it has been designed to detect similar segments of identical length. Hence, the algorithm is simpler and this enables a very high density of processors (812 per chips) to fit on silicon.

[1] SAMBA stands for Systolic Accelerator for Molecular Biological Applications

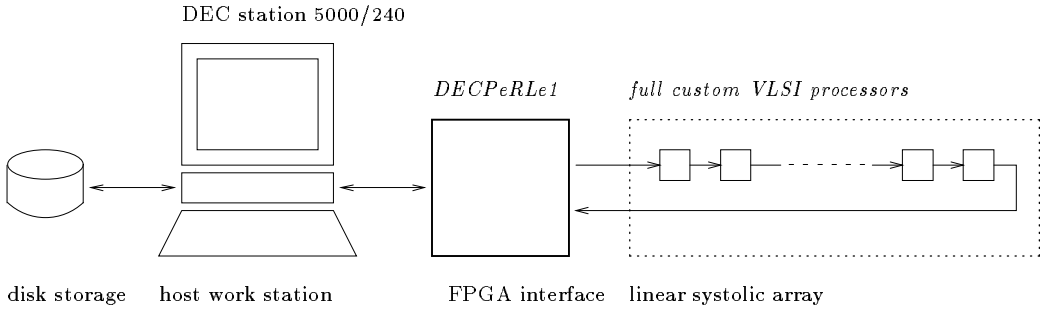

Figure 1: *SAMBA comprises a workstation (with its local disk), a systolic array made out of 128 VLSI full custom processors and a FPGA interface which fills the gap between a complete hardwired array of processors and a programmable Von Neuman machine*

The performance of SAMBA depend greatly of the application. The best ones are achieved for bank to bank comparison. In that case, the systolic array is continuously supplied with data, leading to a speed-up better than two orders of magnitude over standard workstations.

The scan of biological banks (one query sequence against a bank) provides also very interesting performances which, however, are limited by the disk access and the length of the query sequence. As we shall explain later, the longer the query sequence, the better the performances. SAMBA is thus best suited to intensive comparison tasks than to the scan of biological banks, even if this last task provides a noticeable speed-up.

This report is organized as follows: section 2 presents the class of algorithms supported by SAMBA. Sections 3 and 4 explain briefly how these algorithms are implemented on a linear systolic array. Section 5 gives more details on the hardware, particularly on the FPGA-based interface and the ASIC developed

for the array. Finally, section 6 gives some performance measured on the prototype.

2 SAMBA Algorithm

The algorithm implemented by SAMBA belongs to the dynamic programming class. This model allows biologists to compare biological sequences with two basic *edit operations*:

- the *substitution* of characters;
- the *insertion* or *omission* of characters; this operation is called a *gap*.

By using series of such edit operations, any sequence may be transformed into any other sequence. The smallest number of edit operations required to change one sequence into another is thus a measure of the distance between them. The computation of the edit distance is achieved by dynamic programming, which consists of computing recursively an $N \times N$ matrix, where N is the length of the sequences.

The recurrence equation computed by SAMBA comes from the well-known Smith and Waterman algorithm[2]. However, it has been adapted to cover a larger field. A similarity matrix H is calculated recursively using the following equation:

$$H(i,j) = Max \begin{cases} delta \\ E(i,j) \\ F(i,j) \\ H(i-1,j-1) + \mathbf{Sbt}\ (S1_i, S2_j) \end{cases} \quad (1)$$

with:

[2]A description of the Smith and Waterman algorithm can be found in appendix

$$E(i,j) = Max \begin{cases} H(i,j-1) - alpha \\ E(i,j-1) - beta \end{cases} \qquad F(i,j) = Max \begin{cases} H(i-1,j) - alpha \\ F(i-1,j) - beta \end{cases}$$

and the initializations:

$$H(i,0) = E(i,0) = hi(i) \qquad H(0,j) = F(0,j) = vi(j)$$

Sbt is a character substitution cost table. *alpha*, *beta*, *delta*, *vi* and *hi* are parameters used for selecting different variation of the Smith and Waterman algorithm. The two next sections present two examples of such variations, and indicate how the parameters must be set.

Exemple 1: alignment of two sequences

Consider two sequences `S1` and `S2` of respective length $l1$ and $l2$. The problem is to find the alignment of `S1` and `S2` which maximizes a similarity measure with simple gap cost. The similarity measure is given by $H(l1, l2)$ calculated from the above equation with the parameters set to:

- $delta = -\infty$
- $alpha = beta = g$ (gap cost)
- $hi(i) = vi(i) = -g \times i$

Setting *delta* to $-\infty$ invalidates the first term of the equation (1) while setting identically *alpha* and *beta* simplifies respectively $E(i,j)$ to $H(i,j-1) - g$ and and $F(i,j)$ to $H(i-1,j) - g$. The resulting equation becomes:

$$H(i,j) = Max \begin{cases} H(i,j-1) - g \\ H(i-1,j) - g \\ H(i-1,j-1) + \mathbf{Sbt}\ (S1_i, S2_j) \end{cases} \qquad (2)$$

with the initializations:

$$H(i,0) = E(i,0) = -g \times i$$

This is the equation introduced by Needleman and Wunsch [11]: the cost of k multiple gaps g_k is a simple function of the cost of one gap : $g_k = g \times k$

Exemple 2: identification of similar segments

The challenge is to locate similar subsequences between two sequences. This is probably the most useful algorithm for current research, since two biological sequences which present a little overall similarity may share surprising relationships on short segments. Setting *delta* to 0 in equation (1) leads directly to the Smith and Waterman equation [14] [7]. Parameters must be set to:

- $delta = 0$
- $alpha = \alpha$
- $beta = \beta$
- $hi(i) = vi(i) = 0$

The similarity matrix H contains local maxima which may be interpreted as areas where similarities appear between two sequences. Multiple gap costs are taken into consideration as follows: (α) is the cost of the first gap; (β) is the cost of the following gaps. The total gap cost function $G(k)$ is given by: $G(k) = \alpha + \beta \times (k-1)$ with $(\beta \leq \alpha)$.

Note that the gap cost could be simpler (as in example 1) by setting identically the parameters *alpha* and *beta*.

Software tools, such as BLAST [1] for example, find local alignments without gap. This type of alignment can be computed on SAMBA with the parameters set to:

- $delta = 0$

- $alpha = beta = -\infty$
- $hi(i) = vi(i) = 0$

leading to the following recurrence equation:

$$H(i,j) = Max \begin{cases} 0 \\ H(i-1,j-1) + \textbf{Sbt}\ (S1_i, S2_j) \end{cases} \quad (3)$$

This is the equation computed by the BIOSCAN machine [17] for locating similar segments of identical length.

3 Parallelization

The parallelization of string comparison algorithms on linear systolic arrays has been abundantly described in the litterature. The readers interested by this design step may refer to [9] [8] [4] [10] [17].

Samba uses an architecture showned in Figure 2. It is composed of identical processors, linearly arranged, and which perform one step of the matrix computation described in the previous section. Data move in a unidirectional left-to-right direction. Typically, a parallel comparison of two sequences is performed as follows:

1. the array is initialized with one sequence (usually called the query sequence) at the rate of one character per processor;
2. the other sequence is flushed to the left of the array and progresses every systolic cycle;
3. the result is collected on the rightmost processor when the last character of the second sequence is output.

Thus, if l_q is the length of the query sequence and l_b is the length of a sequence coming from a bank, the computation is performed in $l_q + l_b - 1$ systolic steps, providing a speed-up S_1 over sequential machines given by:

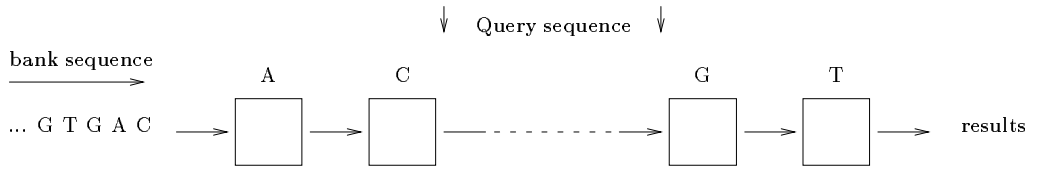

Figure 2: *Sequence comparison on a linear systolic array: one sequence (the query sequence) is stored into the array (one character per processor) and the other sequence flows from the left to right through the array. On each systolic step, one elementary matrix computation is performed on each processor. The result is available on the rightmost processor when the last character of the flowing sequence is output.*

$$S_1 = \frac{l_q \times l_b}{l_q + l_b - 1}$$

In the case of a comparison of one sequence against a bank of sequences, the speed-up may be increased by pipelining the sequences flowing throught the array: when the last character of a sequence enters the array, the first character of the following sequence can be input in the next systolic cycle. For k sequences, $l_q + k \times l_b - 1$ systolic steps are required. The speed-up (S_k) is then given by:

$$S_k = \frac{K \times l_q \times l_b}{l_q + K \times l_b - 1}$$

When scanning a biological database, the number k of sequences is generally high and the speed-up S_k can be approximated by l_q (the length of the query sequence). This last case represents the application domain where SAMBA excels.

4 Getting back the results

In a systolic array, only the border processors communicate with the outside world. Hence, the results must necessarily be forwarded to the border to be output. In the present case, the computation consists of calculating a matrix in which the interesting values can appear anywhere: for example, the identification of similar segments requires the full matrix to be analyzed to detect high scores.

Forwarding values to the array extremities requires additional computing features. The SAMBA algorithm is completed with the computation of two terms:

- `MaxRow`(i,j), corresponding to the maximum score $H(i,j)$ computed on the ith row of the matrix H;
- `IdxCol`(i,j), an index specifying the column where `MaxRow` is modified.

These two terms are computed using the following two recurrence equations:

$$\texttt{MaxRow}(i,j) = Max \begin{cases} \texttt{MaxRow}(i,j-1) \\ H(i,j) \end{cases} \tag{4}$$

$$\begin{aligned} &\texttt{if } (\ \texttt{MaxRow}(i,j-1) > H(i,j)\ ) \\ &\quad \texttt{then IdxCol}(i,j) = \texttt{IdxCol}(i,j-1) \\ &\quad \texttt{else IdxCol}(i,j) = j \end{aligned}$$

Let N be the size of the array and i the systolic step number. During each systolic step, the rightmost processor of the array delivers `MaxRow`(i,N) and `IdxCol`(i,N). These informations enable the direct location on the matrix H of the coordinates of local maxima.

5 Hardware

As mentionned previously, SAMBA is composed of a workstation, a FPGA-interface and a full custom systolic array. This section describes in details each one of these elements.

Workstation and I/O

The DECstation 5000 models have a CPU based on the MIPS R3000A RISC architecture; the DS5000/240 model (used in SAMBA) integrates the 40 MHz R3400 version of this processor architecture. The workstation's CPU measured system performance is rated at 42.9 MIPS, 32.4 SPECmarks, and 10.8 MFLOPS [6]. This system has been introduced to the market in the early 1990s.

The I/O system is based on the TURBOchannel open interconnect developed by Digital and provides three slots for connecting optional peripherals. It runs on the 25 MHz memory system clock and has maximum peak bandwith of 100 MB/s. It is connected to a custom I/O controller which interfaces the TURBOchannel on one side to several different devices on the other side, mostly controllers for particular types of peripherals or data communication interfaces, such as serial ports, SCSI controller, Ethernet, etc.

The SCSI controller can perform asynchronous or synchronous data transfers at up to 5 MB/s through DMA access. The hard disk drive available on the system has a capacity of 426 MB with a maximum bus bandwith of 4 MB/s. This local disk is used to store the banks of biological sequences.

Host/array interface

The host/array interface uses the concept of PAM (Programmable Active Memories) introduced by Vuillemin et al. [3] [2]. As mentioned by the authors, the purpose of a PAM is to implement a virtual machine which can be dynamically configured as a large number of specific hardware devices.

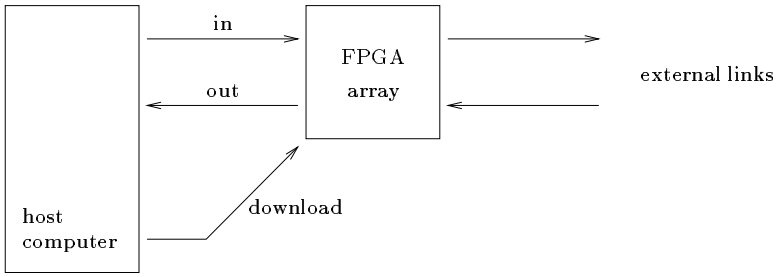

Figure 3: *Principle of a Programmable Active Memory:*

Figure 3 shows the structure of a PAM. It is connected, throught two links (*in* and *out*) to a host processor. A third connection allows a configuration bitstream to be downloaded into the PAM; after the configuration phase, the PAM behaves like an ASIC. The PAM may operate in different modes:

- stand-alone mode: the PAM is hooked to external systems through the external links;
- co-processor mode: the PAM is controlled by the host and specialized to speed-up some crucial computations;
- mixed mode: the PAM is both controlled by the host and connected to some specific hardware.

SAMBA uses the last mode: the external links are connected to the systolic array. The major role of the PAM is to feed rapidly the array, and filter on the fly the data output by the array.

The PAM we use is the PeRLe-1 prototype board, a configurable co-processor organized around a central computational matrix made out of 16 Xilinx XC3090 FPGAs, surounded by 4 memory banks for local storage, and 7 other FPGAs to implement switch and control functions. PeRLe-1 is connected to the worksta-

tion through one of the TURBOchannel extension slots, providing a very fast communication channel between the PAM and the CPU.

This PAM has been designed to be a general purpose configurable platform and includes many features that have not been selected for our specific use.

Systolic array

The array is a linear systolic array composed of 32 full custom chips distributed over two printed boards. One chip (designed in 1 micron CMOS technology) integrates 4 processors, leading to a systolic array of 128 processors. The datapath of the chip is encoded on 12 bits; this may be insufficient for some applications but is enough for validation of the SAMBA architecture.

Each processor computes the recurrence equation presented in section 2 as well as the two auxiliary equations used for locating the results (cf section 4). A performance measure usually used is the number of *million cell operation per second* (MCOPS), that is, the completed computation of a matrix $H(i,j)$ score, including all comparisons, additions and maxima calculations that give a value at that node. SAMBA performes this in one systolic step (100 ns) leading to a 128 processor array peak performance P_p of:

$$P_p = Nb_{proc} \times freq = 128 \times 10^7 = 1280 \text{ MCOPS}$$

Futhermore, a processor contains a 32 word internal static memory for storing the substitution costs. This size is sufficient since a processor has to know a maximum of 20 substitution costs (they are 20 different animo acids) which refer to one character of the query sequence.

The systolic array is connected through the direct custom links available on the PeRLe-1 board.

6 Performances

This section gives some results obtained with SAMBA on a actual intensive protein bank to bank comparison example. The purpose was to compare a set of 815 proteins against the release 31 of SWISS-PROT using the rigorous Smith and Waterman algorithm with different substitution matrices and different gap penalties.

More precisely, the sequences to test belong to yeast protein sequences considered as orphan (i.e. with no similarities with other sequences) when compared to SWISS-PROT with software such as BLAST or FASTA. The idea was to evaluate the capability of the dynamic programming algorithm – which is renowned to be the most sensitive – to find *parents* for some orphan sequences.

Our purpose here is not to discuss the biological interest of such an experiment, but to focus on the performances of SAMBA for that particular task.

Let us first consider the execution of SSEARCH on a DEC-Alpha workstation with a 21064 150MHz micro-processor. SSEARCH is a software provided with the FASTA package; it finds local alignment according to the Smith and Waterman algorithm. Measurements indicate that the computation of a matrix cell is achieved in 0.25 μs when run as follows[3]:

```
ssearch -Q -b 20 -d 0 query_seq sprot31.fasta
```

The release 31 of SWISS-PROT contains exactly 43 470 sequences distributed on 15 335 248 amino acids. The 815 yeast sequences represent a total of 307 400 amino acids. The time t_{seq} for comparing this set of sequences against the bank is thus given by:

$$t_{seq} = 15\,335\,248 \times 307\,400 \times 0.25 \times 10^{-6} = 1\,178\,514 \ sec.$$

This is equivalent to 327 hours (or 13 days and 15 hours) of non-stop computation. Knowing that this task should be repeated, at least, with three different

[3]or an average performance of 4 MCOPS

substitution matrices and two different gap costs, this experiment would have required nearly three months of intensive computation.

The implementation of the dynamic programming algorithm on SAMBA is straightforward as explained in section 2. It just requires to handle properly – on the interface – the data supply of the array as well as the filtering of the results. The comparison of the 815 sequences against the bank is achieved in 1 hour and 45 minutes providing a speed-up of 190 for this particular application.

The time have been measured using the UNIX command `time`. Thus, this is the total elapsed time, as it directly affects the user; it includes time for reading the database from the disk and time for re-computing on the workstation the exact score which cannot be found by the array due to the 12-bit processor arithmetic (in that case the array indicates an overflow and the comparison is performed by the workstation).

The bank to bank comparison, as illustrated by this example, fits completely the SAMBA fonctionalities. Performances are thus excellent. In addition, other applications such as the scan of databases (comparison of one query sequence agaisnt one databases) may also be performed very quickly.

Table 1 indicates the SSEARCH scan time (in second) on different workstations (bank : SWISS-PROT - release 31) with proteins of different length. It indicates also the speed-up when SAMBA is used instead.

The longer the query sequences, the better the speed-up. This is mainly due to the restricted bandwith of the I/O disk system which prevents the array from being fed at its maximum rate: a short sequence does not require, on the array, the computation to be split into several passes. Consequently, the array is fed at the disk access rate, which is generally much slower than the array feeding rate. On the other hand, the comparison of a long query sequence requires the computation to be split into many passes which re-use the data coming from the bank. The array average feeding rate is then substantially decreased and does not become a limitation factor.

Query sequence length	10	30	100	300	1000	3000	10000
SAMBA	25	25	26	30	40	77	210
DEC-Alpha - 150 MHz CPU	57	120	350	1041	3468	11510	38450
speed-up	*2.3*	*4.8*	*13.5*	*34.7*	*86.7*	*150*	*183*
SUN-SPARC 5 - 110 MHz CPU	95	239	746	2215	7300	24269	80300
speed-up	*3.8*	*9.5*	*28.6*	*74*	*183*	*315*	*382*
DEC 5000/250 - 40 MHz CPU	182	548	1407	4054	12920	41169	131193
speed-up	*7.3*	*22*	*54*	*135*	*323*	*534*	*625*

Table 1: SSEARCH *scan time (in second) of* SWISS-PROT 31 *for various length of the query sequence on SAMBA and different workstations. The speed-up compared to SAMBA is also reported. The longer the query sequence, the better the speed-up.*

7 Conclusion

The SAMBA prototype designed at IRISA has demonstrated that a dedicated full custom systolic array is very well suited for biological sequence comparison. This is a low cost solution compared to the implementations on massively parallel machines (MPSEARCH [15]) or commercial available systems (BIOCCELERATOR [5]).

Performances are coming from the association of full custom components and FPGA devices: the 128 processors of the systolic array deliver a high computational power, while the FPGA components of the interface board manage efficiently the host/array communications, the computation partitioning, the array data supply and the result filtering process.

SAMBA is well suited to sequence analyse that require a large amount of computation, such as the bank to bank comparison or the bank query with long sequences. The exponential growth of the biological banks associated with the sequencing of complete genome make SAMBA an interesting solution for the future.

The SAMBA prototype, designed at IRISA, is currently split into three distinct printed boards: the FPGA-interface board and two 64-processor boards. Today's integration density makes possible to fit SAMBA onto a single printed board. The chip could easily contains twice as many processors running at double speed, while the interface could be reduced to a few state of the art FPGA components associated with a few Mbytes of local memory.

We estimate that the cost of SAMBA would roughly the same as the cost of a middle range worsktation. In other words, computational power becomes accessible to every laboratory which has to face a daily need of power for sequence analysis.

Thanks

I would like to thanks Charles Wagner, the designer of the chip, and Pascale Guerdoux-Jamet, the first biologist who has used (and debugged ?) SAMBA. Without their help, the SAMBA project wouldn't be nearly as succesful.

References

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Biol. Mol*, 215:403–410, 1990.

[2] P. Bertin. *Mémoires actives programmables : conception, réalisation et programmation.* PhD thesis, Université Paris 7, juin 1993.

[3] P. Bertin, D. Roncin, and J. Vuillemin. Programmable active memories: a performance assessment. In F. Meyer, B. Monien, and A.L. Rosenberg, editors, *Parallel Architectures and their efficient use*, pages 119–130. LNCS, Springer-Verlag, oct 1992.

[4] E. Chow, T. Hunkapiller, and J. Peterson. Biological Information Signal Processor. In *ASAP*, pages 144–160, sep 1991.

[5] Compugen. The bioccelerator machine. Israel, 1993.

[6] DEC. Decstation and decsystem 50000 model 240. Technical Report EC-N0194-51, Digital Equipment Corporation, Palo Alto, California, February 1992.

[7] O. Gotoh. An Improved Algorithm for Matching Biological Sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[8] P. Guerdoux-Jamet and D. Lavenier. Systolic filter for fast dna similarity search. In *ASAP'95*, Strasbourg, July 1995.

[9] D. Lavenier. An Integrated 2D Systolic Array for Spelling Correction. *Integration : the VLSI journal*, 15:97–111, Aug 1993.

[10] D. Lopresty and al. Building and using a highly parallel programmable logic array. *Computers*, pages 81–89, jan 1991.

[11] S.B. Needleman and C.D. Wunsh. A General Method Applicable to the Search of Similarities in the Amino Acid Sequence of Two Proteins. *J. Mol. Biol*, 48:443–453, 1970.

[12] W. R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci.*, 85:3244–3248, 1988.

[13] W.R. Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the smith and waterman and fasta algorithms. *Genomics*, 11:635–650, 1991.

[14] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol*, 147:195–197, 1981.

[15] S.S. Sturrock and J.F. Collins. Mpsearch version 1.3. Technical report, University of Edinburgh, Biocomputing Research Unit, 1993.

[16] M.S. Waterman. *Mathematical Methods for DNA Sequences*. CRC Press, Inc, 1989.

[17] C.T. White, R.K. Singh, P.B. Reintjes, J. Lampe, B.W. Erickson, W.D. Dettloff, V.L. Chi, and S.F. Altschul. BioSCAN: A VLSI-Based System for Biosequence Analysis. In *IEEE Int. Conf on Computer Design: VSLI in Computer and Processors*, pages 504–509. IEEE Computer Society Press, oct 1991.

Appendix

The Smith and Waterman algorithm

The Smith and Waterman algorithm generally refers to the identification of similar segments using the dynamic programming method. It has been introduced by T.F. Smith and M.S. Waterman [14] in 1981 and extended by Gotoh [7] in 1982.

Consider the two following sequences $S1$ and $S2$:

```
S1 = AGTCCGAGGGCTACTCTACTGAAC
S2 = CCAATCTACTACTGCTTGCAGTAC
```

One may found that the segment `CTACTCTACT` of $S1$ is similar to the segment `CCAATCTACT` of $S2$:

```
S1   AGTCCGAGGG CTACTCTACT GAAC
                |:|:||||||
S2              CCAATCTACT ACTGCTTGCAGTAC
```

or that the same segment `CTACTCTACT` of $S1$ is also similar to another segment `CTACTACTGCT` of $S2$:

```
S1   AGTCCGAGGG CTACT.CTACT GAAC
                ||||| ||:||
S2        CCAAT CTACTACTGCT TGCAGTAC
```

To identify common subsequences, the Smith and Waterman algorithm finds the similarity of two segments ending at position $S1_i$ and $S2_j$ of the two sequences $S1$ and $S2$ (respectively of length $l1$ and $l2$).

The basic recurrent equation is given by:

$$H(i,j) = Max \begin{cases} 0 & 0 < i \leq l1 \quad 0 < j \leq l2 \\ E(i,j) \\ F(i,j) \\ H(i-1,j-1) + \mathbf{Sbt}(S1_i, S2_j) \end{cases} \tag{5}$$

with:

$$E(i,j) = Max \begin{cases} H(i,j-1) - \alpha & 0 < i \leq l1 \quad 0 < j \leq l2 \\ E(i,j-1) - \beta \end{cases}$$

$$F(i,j) = Max \begin{cases} H(i-1,j) - \alpha & 0 < i \leq l1 \quad 0 < j \leq l2 \\ F(i-1,j) - \beta \end{cases}$$

The initializations are given by:

- $H(i,0) = 0 \qquad E(i,0) = 0 \qquad 0 \leq i \leq l1$
- $H(0,j) = 0 \qquad F(0,j) = 0 \qquad 0 \leq j \leq l2$

Sbt is a character substitution cost table. Multiple gap costs are taken into consideration as follows: (α) is the cost of the first gap; (β) is the cost of the following gaps. The total gap cost function $G(k)$ is given by: $G(k) = \alpha + \beta \times (k-1)$ with $(\beta \leq \alpha)$.

Example

Consider the two sequences:

```
S1 = AGTCCGAGGGCTACTCTACTGAAC
S2 = CCAATCTACTACTGCTTGCAGTAC
```

The gap costs such as $\alpha = 20$ and $\beta = 10$ and the *Sbt* function defines as:

$$Sbt(x,y) = \begin{cases} 10 & if\ (x = y) \\ -9 & otherwise \end{cases}$$

The follwing matrix is computed:

		C	C	A	A	T	C	T	A	C	T	A	C	T	G	C	T	T	G	C	A	G	T	A	C
A		0	0	10	10	0	0	0	10	0	0	10	0	0	0	0	0	0	0	0	10	0	0	10	0
G		0	0	0	1	1	0	0	0	1	0	0	1	0	10	0	0	0	10	0	0	20	0	0	1
T		0	0	0	0	11	0	10	0	0	11	0	0	11	0	1	10	10	0	1	0	0	30	10	0
C		10	10	0	0	0	21	1	1	10	0	2	10	0	2	10	0	1	1	10	0	0	10	21	20
C		10	20	1	0	0	10	12	0	11	1	0	12	1	0	12	1	0	0	11	1	0	0	1	31
G		0	1	11	0	0	0	1	3	0	2	0	0	3	11	0	3	0	10	0	2	11	0	0	11
A		0	0	11	21	1	0	0	11	0	0	12	0	0	0	2	0	0	0	1	10	0	2	10	1
G		0	0	0	2	12	0	0	0	2	0	0	3	0	10	0	0	0	10	0	0	20	0	0	1
G		0	0	0	0	0	3	0	0	0	0	0	0	0	10	1	0	0	10	1	0	10	11	0	0
G		0	0	0	0	0	0	0	0	0	0	0	0	0	10	1	0	0	10	1	0	10	1	2	0
C		.10	.10	0	0	0	10	0	0	10	0	0	10	0	0	20	0	0	0	20	0	0	1	0	12
T		0	.1.	1	0	10	0	20	0	0	20	0	0	20	0	0	30	10	0	0	11	0	10	0	0
A		0	0	.11	.11	0	1	0	30	10	0	30	10	0	11	0	10	21	1	0	10	2	0	20	0
C		10	10	0	.2.	2	10	0	10	40	20	10	40	20	10	21	1	1	12	11	0	1	0	0	30
T		0	1	1	0	.12.	0	20	0	20	50	30	20	50	30	20	31	11	1	3	2	0	11	0	10
C		10	10	0	0	0	.22.	2	11	10	30	41	40	30	41	40	20	22	2	11	0	0	0	2	10
T		0	1	1	0	10	2	.32	.12	2	20	21	32	50	30	32	50	30	20	10	2	0	10	0	0
A		0	0	11	11	0	1	12	.42	.22	12	30	12	30	41	21	30	41	21	11	20	0	0	20	0
C		10	10	0	2	2	10	2	22	.52	.32	22	40	20	21	51	31	21	32	31	11	11	0	0	30
T		0	1	1	0	12	0	20	12	32	.62	.42	32	50	30	31	61	41	31	23	22	2	21	1	10
G		0	0	0	0	0	3	0	11	22	42	53	33	30	60	40	41	52	51	31	21	32	12	12	0
A		0	0	10	10	0	0	0	10	12	32	52	44	24	40	51	31	32	43	42	41	21	23	22	3
A		0	0	10	20	1	0	0	10	2	22	42	43	35	30	31	42	22	23	34	52	32	22	33	13
C		10	10	0	1	11	11	0	0	20	12	22	52	34	26	40	22	33	13	33	32	43	23	13	43

For each position $(S1_i, S2_j)$ a similarity value is given. From this point, the two segments producing this score can be determined by a backtrack procedure [16]. In that example, the best score is 62 and the two segments detected as similar are `CTACTCTACT` and `CCAATCTACT`.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399