SAMBA: hardware accelerator for biological sequence comparison

P. Guerdoux-Jamet and D. Lavenier

IRISA, Campus de Beaulieu, 35042 Rennes cedex, France

Received on May 14, 1997, revised on July 9, 1997, accepted on July 18, 1997

Abstract

Motivation: SAMBA (Systolic Accelerator for Molecular Biological Applications) is a 128 processor hardware accelerator for speeding up the sequence comparison process. The short-term objective is to provide a low-cost board to boost PC or workstation performance on this class of applications. This paper places SAMBA amongst other existing systems and highlights the original features.

Results: Real performance obtained from the prototype is demonstrated. For example, a sequence of 300 amino acids is scanned against SWISS-PROT-34 (21 210 389 residues) in 30 s using the Smith and Waterman algorithm. More time-consuming applications, like the bank-to-bank comparison, are computed in a few hours instead of days on standard workstations. Technology allows the prototype to fit onto a single PCI board for plugging into any PC or workstation.

Availability: SAMBA can be tested on the WEB server at URL http://www.irisa.fr/SAMBA/

Contact: E-mail: lavenier@irisa.fr

Introduction

The sequence comparison process is a common and often repeated task in molecular biology. It occurs at many levels, from the shotgun operation to phylogeny treatment or database scanning. Depending both on the volume of data to be processed and the accuracy of the comparisons, the computation can be time consuming and even a performance bottleneck.

Usually, this constraint is met by different techniques: software heuristics, parallelization on massively parallel computers, distribution on a network of workstations, or dedicated hardware. Each of these techniques has advantages and drawbacks, as summarized below.

Software heuristics can drastically reduce the time of computation: instead of systematically computing all the possible alignments between two sequences, the search starts from areas where strong similarities (identical words) have been found. The length of the words determines the speed of the process: the longer the word, the faster the comparison. At the same time, the longer the word, the better the chances are of missing some significant alignments. It is the responsibility of the biologist to find the best compromise between speed and sensitivity. Blast (Altschul et al., 1990) and Fasta (Pearson, 1991) (two commonly used software packages) include such heuristics.

Massively parallel computers are not available in biology laboratories and access must be performed through the Internet. The computation is fast, but the remote access restricts the interactivity: results are returned by mail and the delay depends on the load of the server.

Networks of workstations aim to use local computing power. This idea is attractive (and works), but is practically difficult to implement in the laboratories: it often requires a homogeneous park of machines, a distributed software system allowing different operating systems and typically, a system engineer for installing the software.

Dedicated machines (or hardware accelerators) are small hardware units connected to a PC or a workstation. They can only support a limited class of computations, but perform very rapidly.

This paper deals with the last category and introduces the SAMBA (Systolic Accelerator for Molecular Biological Applications) machine, a hardware accelerator dedicated to biological sequence comparison. SAMBA is not the first piece of hardware for speeding up the sequence comparison. The difference between our approach and that of the other existing hardware solutions is explained.

The next section presents the state of the art in current research and ongoing projects in that domain. Subsequent sections highlight the originalities of SAMBA compared with other existing systems, give performances measured on the prototype and briefly present a few applications implemented on SAMBA. The final section concludes with some perspectives.

State-of-the-art

This section surveys some dedicated machines designed specifically for speeding up sequence comparison. It describes...
similar projects to SAMBA, i.e. machines which are restricted to this class of applications. Reconfigurable systems are not considered, i.e. SPLASH-2 (Hoang, 1993), PeRLe-1 (Lemoine et al., 1994; Guerdoux-Jamet and Lavenier, 1995), or massively programmable parallel machines like RAPID-2 (Archambaud et al., 1995), Kestrel (Hirschberg et al., 1996) or MGAP (Borah et al., 1994). These machines can support a wide range of applications and cannot be ranked in the category of dedicated hardware for sequence comparison.

The machines are classified into two categories: the ASIC systems and the FPGA systems. The next two subsections detail each category and illustrate them with current realizations.

**ASIC systems**

An ASIC (Application Specific Integrated Circuit) component is a chip devoted to a single function (or a restricted class of functions). Once designed and fabricated, it cannot be modified, and thus cannot develop without a major investment.

In dedicated ASIC systems for sequence comparison, the computation is usually performed by a linear array of identical ASIC processors. The peak performance of these machines is impressive since all the processors (a few hundred) work concurrently and synchronously. The BioSCAN machine (White et al., 1991) and the BISP machine (Chow et al., 1991) belong to this category.

BioScan accelerates the identification of similar segments for DNA or protein sequences without allowing gaps. It has been designed at the University of North Carolina. A chip contains 812 1-bit processors, and a system with 16 chips is currently working, leading to a total of 12 992 processors. In that configuration, scanning a database limits the query sequence to 12 992 characters. Currently, this is the most powerful system for detecting similar segments of identical length, but as far as we know, no commercial version is yet available.

BISP (Biological Information Signal Processor) provides a high-speed and linearly extensible system that can locate similar subsequences of two DNA or protein sequences. It implements a modified version of the Smith and Waterman algorithm and allows many parameters to be set. The machine has been designed at the California Institute of Technology. A chip contains 16 processors and a prototype machine of 16 chips (256 processors) has been validated, making possible the computation of unlimited sequence length. Again, this machine does not have a commercial version.

The computational power of these machines depends directly on the clock speed and the number of processors.

**FPGA systems**

The FPGA (Field Programmable Gate Array) components are based on recent technology. Their advantages, compared to the ASIC approach, are 2-fold: no specific chips need to be fabricated, and they can be reconfigured indefinitely (correction as well as evolution of the hardware is possible). However, they are slower and not so optimized as ASIC components. The Bioaccelerator (http://www.compugen-us.com/index.html) (Compugen, 1993) and the DeCypher II (http://www.timelogic.com) (Time Logic, 1996) systems are designed with this technology.

The Bioaccelerator system is a Unix workstation peripheral, attached to the sequence database server through a SCSI connection. Currently implemented algorithms include: frame search, profile search, Smith–Waterman local alignment, and Needleman–Wunsch global and overlap alignment programs. Translated search programs, as well as multiple alignment and clustering programs, are also implemented. This machine is marketed by Compugen Ltd.

The DeCypher II system has been mainly designed for speeding up database searching with the Smith and Waterman algorithm (Smith and Waterman, 1981). It is based on FPGA accelerator cards plugged into a personal workstation. The performance depends on the number of cards: Time Logic Inc. products range from small systems (1–3 cards) to high-performance cluster systems (up to 120 cards).

As opposed to the ASIC systems, these two machines are available commercially and can run a larger variety of algorithms. On the other hand, this technology is slower and less dense than the ASIC technology, and requires a large number of components to reach similar performances.

**SAMBA**

The SAMBA system belongs to the ASIC category since the heart of the system is a dedicated VLSI processor array, but the complete system contains an FPGA–Memory interface. Figure 1 shows the general architecture: the array is connected to the host workstation through an FPGA–Memory board which acts as an array controller and a high-speed mechanism for correctly feeding the array and filtering results on the fly.

The array of the SAMBA prototype is composed of 32 full-custom identical chips which each house four processors, leading to a 128 processor array. The chip has been designed at IRISA and provides a computational power of 400 million operations per second. Hence, the array is able to reach a peak performance of 12.8 billion operations per second. The prototype is currently split into three boards and connected to a Dec 5000/240 workstation.

The structure of the array, a systolic linear array, is very well suited for supporting string-processing parallelization.
and has been widely described in the literature. Hence, the general architecture of SAMBA cannot be considered as a revolutionary one. Nevertheless, it includes many features, not present in other dedicated systems, which make this project unique. The next subsection highlights these different aspects and compares them to other existing systems.

**Parameterized algorithm**

The algorithm implemented on the processors belongs to the dynamic programming class. The recurrence equation computed by SAMBA comes from the well-known Smith and Waterman algorithm (Smith and Waterman, 1981). However, it has been adapted—parameterized—to cover a larger scope. A similarity matrix, $H$, is calculated recursively using the following equation:

$$H(i,j) = \begin{cases} 
\text{delta} & 
\\ 
E(i,j) & 
\\ 
F(i,j) & 
\\ 
H(i-1, j-1) + \text{Sbt}(S1, S2) & 
\end{cases}$$

with:

$$E(i,j) = \text{Max}\left\{ H(i,j-1) - \alpha, E(i,j-1) - \beta \right\}$$

$$F(i,j) = \text{Max}\left\{ H(i-1,j) - \alpha, F(i-1,j) - \beta \right\}$$

and the initializations:

$$H(0,0) = E(0,0) = F(0,0) = 0$$

$\text{Sbt}$ is a character substitution cost table. $\alpha$, $\beta$, $\text{delta}$, $\text{vi}$ and $\text{vt}$ are the parameters used for selecting different comparison algorithms.

For instance, computing the global alignment between two sequences requires the parameters to be set as follows: $\text{delta} = -\infty$, $\alpha = \beta = g$ (gap cost), $\text{vi}(i) = \text{vt}(i) = -g \times i$. We find again the equation introduced by Needleman and Wunsch (1970): the cost of $k$ multiple gaps, $g_k$, is a simple function of the cost of one gap: $g_k = g \times k$.

Another example is the identification of similar segments. Setting $\text{delta}$ to 0 in equation (1) leads directly to the Smith and Waterman (1981) equation (Gotoh, 1982). Parameters must be set to: $\text{delta} = 0$, $\alpha = \beta = 0$, $\text{vi}(i) = \text{vt}(i) = 0$. In this case, the best score must be stored at each step to provide the final result. In other words, equation (1) must be complemented with an extra maximization.

Multiple gap costs are taken into consideration as follows: $\alpha$ is the cost of the first gap; $\beta$ 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 by setting $\alpha = \beta$.

Software tools, such as Blast (Altschul et al., 1990) for example, find local alignments without gap. This type of alignment can be computed on Samba with the parameters set to: $\text{delta} = 0$, $\alpha = \beta = -\infty$, $\text{vi}(i) = \text{vt}(i) = 0$. This is the equation computed by the BioScan machine (White et al., 1991) for locating similar segments of identical length.

Each processor is also provided with a local memory for storing independent substitution costs, allowing profile search of sequences.

Compared with the other systems introduced earlier, SAMBA is more flexible than BioScan (which is only used for Blast search), it has possibilities similar to BISP, although it cannot reach the FPGA versatility. However, it must be pointed out that the objective of SAMBA was, first and foremost, to speed up the usual sequence comparison algorithms, not to provide an experimental system for testing new algorithms.
FPGA–Memory Interface

The FPGA–Memory Interface is connected between the workstation and the array. It is an important element which greatly influences the performance of SAMBA. It has the job of managing three major tasks: partitioning the computations, providing local storage and filtering data on the fly. The remainder of this section explains these different actions.

The process of comparing a query sequence against a bank consists of first loading the query sequence into the array (one character per processor) and then sending the sequences of the bank through the array from the left to the right, in a pipeline manner.

This scheme assumes that we have an array whose size is equal to the length of the query sequence. This never happens: in practice, the query sequence is too long (>128 characters) and requires the sequence comparison to be split into several passes. The partitioning operates as follows: the 128 first characters of the query sequences are loaded in the array. Then the entire bank crosses the array, while all the data output by the last processor are memorized. In the next step, the 128 following characters of the query sequence are loaded. The data previously stored are mixed with the bank and sent again to the array. The process is iterated until the end of the query sequence is reached.

To maintain reasonable efficiency, the partitioning must be performed at the clock rate of the array. Unfortunately, this rate cannot be sustained by standard microprocessors: every 100 ns the array must be supplied with new data, and the results filtered. Hence, a hardwired solution (dedicated hardware) is essential. On the other hand, the data which are sent to the array depend on the algorithm being executed, and change according to the way the parameters (inside the processors) have been set. Hence, a fixed hardware mechanism for supplying the array with data cannot be implemented definitively. Consequently, the FPGA technology happens to be a better alternative to control the array, since it combines speed and flexibility.

Another important feature is the ability to have local storage. As mentioned above, the partitioning process requires the bank to cross the array several times. At first glance, one can envisage storing the entire bank in a fast local memory. A closer understanding of the partitioning mechanism may help to reconsider this solution. When comparing a query sequence against a bank, the SAMBA memory holds only a subset of a bank. While computation is performed on this subset, another one is loaded from the disk. The overlap between the computation and the data acquisition is efficient thanks to the partitioning operation which takes time and permits the data from the bank to be accessed at a reasonable rate.

The main advantage of this solution is the small size of memory required compared to the size of the banks: the minimum size of the memory is determined by the sum of the query sequence length and that of the longest sequence of the bank. Practically, the problem is formulated differently: the memory size is a fixed resource which limits the length of the sequences to be processed.

The last point deals with the recovery of the results. Communication with the array can be handled only through the left-most and the right-most processor. Then, the results (especially when local alignment is performed) must be routed to the extremity. SAMBA processors are provided with a hardware mechanism for propagating such information. More precisely, the processors carry partial results which need to be processed outside the array for recovering the final result. Once again, to be efficient this treatment must be done on the fly as results are output from the array. Furthermore, this treatment changes according to the comparison algorithm being executed: a global or a local search delivers results having different meanings and which must be handled differently. In that case, FPGA technology still represents the best trade-off between fixed hardware and a programmable solution.

Finally, we conclude this section with some hardware complexity considerations compared to other systems. The SAMBA FPGA–Memory interface can be fit into a single FPGA component of the current generation and requires a small amount of memory (our prototype uses only 2 Mbytes of static RAM). By comparison, the VME BioScan board houses ~250 components (only 16 dedicated to the array) and the BISP controller is a MC68020 microprocessor system with DMA capabilities.

Programming SAMBA

SAMBA can be considered as a co-processor which is accessed when intensive biological sequence comparison is needed. Typically, the operations which have to be accomplished when using SAMBA concern mainly the initialization of the board and loading the query and the subset bank sequences. As an example, scanning a bank can be summarized by the following simplified loop:

```c
InitSW(Matrix, GapL, GapN);
LoadQuerySequence(QS);
Load(SUBSET[0]);
for (i=1;i<NbSubSet;i++)
{
    LoadAndCompare(SUBSET[i], Score[i-1]);
}
Compare(Score[i-1]);
```

The InitSW function configures the FPGA–Memory interface according to the Smith and Waterman algorithm, but
other initialization functions are available, e.g. to perform a global search. In addition, this family of functions loads the substitution matrix and gap costs into the processors.

The LoadQuerySequence and the Load functions, respectively, load the query sequence (QS) and the first subset of the bank (SUBSET[0]) into the SAMBA memory. Then, the loop iterates on the LoadAndCompare function which loads the next subset of sequences while performing the computation on the previous subset.

From a programming point of view, the accelerator is then controlled by a few procedure or function calls inside a normal C-program, without forcing the user to have specific knowledge of the structure of the accelerator and how it works. A library of basic procedures has been developed to be rapidly understood by programmers, but these procedures stay close enough to the accelerator hardware to provide efficient speed-up.

This approach has been chosen to cover a large range of applications which require conventional sequence comparison treatments. By carefully choosing the basic library procedures, programmers will not encounter too many limitations.

The Bioccelerator adopts the GCG environment and speeds up the most time-consuming programs. The DeCypher II system provides a package for the most useful comparison algorithms. Both strategies have the advantage of being ready to use, but at the same time, and from the user's point of view, are restricted to the usual treatments. Adaptation is obviously possible, but requires the intervention of the manufacturers.

Compared with other systems, SAMBA must be seen as a hardware accelerator with a library of high-level comparison C-functions. This allows the choice of pre-defined programs, such as the classical programs already developed for bank scanning, or user-defined programs tuned to a specific application.

Applications and performance

This section aims to highlight the computational power of SAMBA and its flexibility for specific applications. The first point is shown on bank scanning, a classical application for which SAMBA is well suited. The second point is presented through three applications for which specific software has been developed.

Bank scanning

Traditionally, the most typical use of the hardware accelerators is in scanning databases. SAMBA was first evaluated with that application. Table 1 reports the average time for scanning SWISS-PROT (Version 34: 59 021 sequences, 21 210 389 amino acids) using the Smith and Waterman algorithm for different lengths of a protein query sequence.

The first line gives the scanning time (in minutes:seconds) for SAMBA. The three following pairs of lines give, respectively, the time for scanning SWISS-PROT with a 200 MHz Pentium Pro, a 300 MHz Dec Alpha and a 167 MHz VIS Ultra Sparc. These computation times have been estimated from the best sequential implementations of the Smith and Waterman algorithm, as reported in Wozniak (1997). Note that the longer the query sequence, the better the speed-up. This is mainly due to the restricted bandwidth of the I/O disk system which prevents the array from being fed at its maximum rate; a short query sequence does not require the computation to be split into several passes. Consequently, the array is fed at the disk rate, which is generally much slower than the array throughput.

In all cases, the reported time is the total elapsed time, as it directly affects the user; it includes time for reading the databases from the disk, as well as time for post-processing. Hence, we measure the performance of the complete system and not the peak performance as it is often reported in the literature. The speed of a system is always dictated by the speed of the slowest element!

<table>
<thead>
<tr>
<th>Query sequence length (amino acids)</th>
<th>100</th>
<th>300</th>
<th>1000</th>
<th>3000</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMBA</td>
<td>0:22</td>
<td>0:32</td>
<td>0:55</td>
<td>1:40</td>
</tr>
<tr>
<td>200 MHz Pentium Pro</td>
<td>5:20</td>
<td>16:00</td>
<td>53:20</td>
<td>160:00</td>
</tr>
<tr>
<td>Speed-up</td>
<td>14</td>
<td>30</td>
<td>58</td>
<td>96</td>
</tr>
<tr>
<td>300 MHz Dec Alpha</td>
<td>2:47</td>
<td>8:20</td>
<td>27:50</td>
<td>83:30</td>
</tr>
<tr>
<td>Speed-up</td>
<td>7</td>
<td>15</td>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>167 MHz VIS Ultra Sparc</td>
<td>1:56</td>
<td>5:48</td>
<td>19:20</td>
<td>58:00</td>
</tr>
<tr>
<td>Speed-up</td>
<td>5</td>
<td>10</td>
<td>21</td>
<td>35</td>
</tr>
</tbody>
</table>

Tests on long sequences have also been performed. As an example, the portion 3 of the *Haemophilus influenzae* genome [12 268 nucleotides (nt)] requires a computation time of 15 min when compared with the viral subpart of GenBank (Version 96, gbvrl, 36 246 305 nt). The same computation performed on the 150 Mz Dec Alpha workstation with Ssearch takes 35 h, giving a speed-up of 140 for that particular comparison. Other applications well suited to SAMBA are probably of a similar style, since the exponential growth of the genome banks associated with the systematic sequencing of complete genomes provides a large amount of data which need huge computational resources for detailed analysis.

Note that the speed-up measurement is unfavorable to SAMBA; the chips have been designed with a technology which is 5 years older than that of the workstation. Considering that the computing power per unit doubles every year...
(Noyce’s thesis) (Vuillemin, 1993), an up-to-date SAMBA machine (more processors per chip and quicker clock frequency) will push performance an order of magnitude higher than other machines.

Finally, SAMBA’s performance on bank scanning may be compared with other VLSI (BISP) or FPGA (DeCypher II) dedicated systems. If only peak performance is considered, BISP and SAMBA are (more or less) equivalent: they implement the same algorithm and the processors run at nearly the same speed. The major difference is 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 latter approach is simpler and ensures that the high data throughput required by the systolic array is sustained. The DeCypher II system offers a reconfigurable platform in which the usual S&W bank scanning algorithm is implemented. Roughly speaking, SAMBA and one of the biggest DeCypher II TS-series Tower Servers configuration have equivalent peak performance.

Specific applications

This section presents first two applications (named, respectively, Orphan Sequences and Multigenic Family) which have required the development of programs in which SAMBA speed ups the comparison sequence task. The results obtained would probably not have been found without the help of SAMBA. The last application, currently being developed, concerns the assembly of a complete genome from thousands of fragments issued from a single shotgun.

Orphan sequence application. This is an exhaustive study of the orphan sequences of the *Saccharomyces cerevisiae* genome. A set of 814 putative open reading frame sequences, for which no similarities exist when they are compared with Blast or Fasta against SWISS-PROT, constitute the first bank. The second bank is SWISS-PROT (Release 31). The idea of the experiment was to test whether a more sensitive algorithm (namely the Smith and Waterman algorithm) could detect new similarities.

A preliminary step was to determine the optimal parameters (matrix and gap costs) to perform the most sensitive search. Hence, the 814 orphan sequences were submitted to the bank several times using different matrices and gap costs. We finally retained the BLOSUM50 matrix and the 15/0.9 gap costs. With SAMBA, each submission (814 sequences against SWISS-PROT 31) took 1 h and 45 min, while the computation on a 150 MHz Dec Alpha was estimated to take >300 h (12.5 days).

The result of this study can be found in Guerdoux-Jamet and Risler (1996), but can be summarized as follows: the Smith and Waterman comparison detected similarities for 17 orphan sequences. These resemblances were not found by Blast or Fasta. The significance of the 17 alignments has been validated with a Monte-Carlo randomization.

Multigenic family application. Clustering sequences in a family according to their homology is an important technique in the investigation of genomes. For example, some functional categories of proteins, notably those involved in metabolite transport and transcription regulation, tend to form large clusters. The whole genome of *Escherichia coli* was utilized, and all the coding sequences compared using the Smith and Waterman algorithm (matrix = PAM250 and gap costs = 17/1). This represents 4285 sequences (1 355 128 amino acids). As we were only interested in clustering the sequences, SAMBA was programmed to report only the alignment with a score better than 120. Then we retained only the alignment having a Z-score better than 14. A total of 6440 pairs corresponding to these criteria (score > 120 and Z-score > 14) have been found.

With SAMBA, the pairwise comparison took 41 min, while the same treatment carried out on a recent workstation would have taken 127.5 h. From a biological point of view, the alignments have been used to generate 350 families having connex components. In other words, one sequence belonging to a family is similar, at least, to one other sequence in this family. The clusters can provide information on the existence of isozymes or putative functions for hypothetical proteins. These results have not yet been completely analysed and are a subject of ongoing research.

Genome assembly application. This application is under study. It concerns the assembly of long consensus sequences from thousands of small fragments coming from a single shotgun of a complete genome. The genomes we consider are fairly small (a few Mb), and the length of the fragments ranges from 2 to 4 kb. The assembly implies the pairwise comparison of all the fragments to detect overlap regions. This is a typical bank-to-bank comparison problem for which SAMBA is well suited and should provide significant speed-up.

Conclusion

The SAMBA prototype could be improved by using up-to-date technology. As far as we know, the evolution of the chip density can now provide a commercial version on a single PCI board for plugging into any PC or workstation. The cost of such a board would not exceed $10 000.

SAMBA has been working since the end of 1995. Its first application was concerned with bank scanning. It can now be tested on our Internet site (http://www.irisa.fr/SAMBA/). The flexibility due to the parameterized algorithm implemented in silicon and the FPGA-Memory interface make SAMBA a useful tool to develop specific applications which involve intensive sequence comparison. The C-function li-
library is an easy way to control SAMBA and constitutes an open programming environment which grows richer as new applications are implemented.

Further research will continue to investigate the benefits of SAMBA on applications which deal with intensive biological sequence comparisons. Such applications may come, for example, from multi-alignment on a large set of sequences, synteny analysis of genomes or chromosomes between different species, etc. All suggestions are welcome!

References


