High Speed Pattern Matching in Genetic Data Base with Reconfigurable Hardware

Eric Lemoine and **Joel Quinqueton** and **Jean Sallantin**
Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
UMR 9928 Université Montpellier II/CNRS
161 rue ADA 34392 Montpellier Cedex 5, France
email : {el,jq,js}@lirmm.fr

Abstract

Homology detection in large data bases is probably the most time consuming operation in molecular genetic computing systems. Moreover, the progresses made all around the world concerning the mapping and sequencing of the genome of Homo Sapiens and other species have increased the size of data bases exponentially. Therefore even the best workstation would not be able to reach the scanning speed required. In order to answer this need we propose an algorithm, A^2R^2, and its implementation on a massively parallel system. Basically, two kinds of algorithms are used to search in molecular genetic data bases. The first kind is based on dynamic programming and the second on word processing, A^2R^2 belongs to the second kind. The structure of the motif (pattern) searched by A^2R^2 can support those from FAST, BLAST and FLASH algorithms. After a short presentation of the reconfigurable hardware concept and technology used in our massively parallel accelerator we present the A^2R^2 implementation. This parallel implementation outperforms any kind of previously published genetic data base scanning hardware or algorithms. We report up to 25 million nucleotides per scanning seconds as our best results.

Keywords : massively parallel computing, word processing, pattern matching, genetic sequences, large data base, reconfigurable hardware, dedicated system.

Introduction

Molecular biology requires both flexible and powerful tools to adapt itself to a rapidly expanding field. The processing of such an amount of data with a user-friendly response delay would be incompatible with the performance of even the most powerful workstation. The use of a supercalculator seems to be a solution, however, because of its financial cost, a supercalculator cannot be installed in each lab. Then it must be used in batch processing mode, which in(tro)duces a delayed access to the data bases information for the biologist and therefore breaking the interactivity between the computer and the end user. By computer we do not only mean the machine but also the huge amount of data in it and the algorithms needed to analyse these data. Besides, the supercalculator's hardware isn't at all suited to the biological issue as the data are symbolical and the floating point arithmetic is hardly used. (Fagin & Watt 1992; Lemoine 1993) Therefore a specialized hardware appears as a reasonable choice if we want the computer to become an everyday tool considered by the biologist as an *in silico* lab.

The advantages of an accelerator dedicated to molecular biology have already been demonstrated. The main argument is the exponential increase in size of the genomic sequence data bases (GSDBs). Indeed, the progresses made all around the world concerning the mapping and sequencing of the genome of Homo Sapiens and other species produce many new sequences each day at an incredible rate. Until now, all the dedicated systems only dealt with the comparison of whole sequences. Although, the need for computing power is universally recognized there is absolutely no consensus on which algorithms to use. See Waterman and Feng for a review on sequence comparaisons in biology. (Watermann 1984; Feng, Johnson, & Doolittle 1985) Molecular biology is heavily data driven, this implies a questioning at the arrival of each new sequence and a quick renewal of the algorithmic. This phenomenon is even more important as the mathematical significance of the results is well established and not the biological significance.

Two kinds of algorithms are currently used and illustrate the two extremes in sequence comparison ; either the search for short amino acids or nucleotide words, or the alignment of an entire sequence. The architectural concepts necessary for the implementation of these algorithms also differ. Indeed, in the case of the alignment of long sequences dynamic programming is used. Speeding up the computation is achieved by a parallelisation with a systolic array. Whereas in the case of the search for words, the most performant parallelisation is based on associative memories.

We introduce A^2R^2 [1] a pattern matching algorithm

[1] A^2R^2 stands for Associative Algorithm for Rapid seaRch, in French search and research have the same meaning.

dedicated to the scanning of GSDB. This algorithm runs on a new kind of massively parallel and low cost (ie the price of a workstation.) computer now available; the reconfigurable hardware systems. It is based on a bit level operation model that enables the implementor to fit the hardware to the problem rather than distorting the problem to fit the computer.

The main idea that has guided our work is : if we significantly increase the speed of an algorithm for molecular genetics we also increase the quality of the results produced by this algorithm. Indeed decreasing the computation time needed by any algorithm by two or three orders of magnitudes enables the statistical validation by Monte Carlo like methods, for instance.

This paper is organized as follows. The first section introduces the types of algorithms used for scanning the GSDBs, then proposes a generic pattern including those from FAST, BLAST and FLASH. In the second section, after a brief introduction of the reconfigurable hardware concept, we show how an exhaustive search based on the pattern previously described can be efficiently implemented. Finally, comparative results between other molecular genetic dedicated systems and our own are presented and discussed.

Scanning Algorithmic

A great variety of algorithms have been written to compare or align genomic sequences. However, a small number of mechanisms are at the heart of most of these algorithms.

We have restricted our interest to the search for information in the genetic data base. To simplify, this comes to comparing a sequence or part of a sequence to all the sequences of the data base.

We have examined how to implement the common basis of these algorithmics on a reconfigurable hardware. As the biologists have associated a semantic to their current methods, one of the goals of our proposal is to speed up their methods without breaking up the semantics.

The two principles of sequence comparison are the search for consensus patterns and the alignment by dynamic programming. These methods consist in offering aligned sequences in such a way that the anchor points overlap each other. The anchoring points are set either by the biologist or by the pattern matching algorithm.

Dynamic Programming

Dynamic programming dedicated to molecular genetics was rediscovered at the begining of the Seventies by Needlman and Wunsh, then improved by Sellers, Gotoh, Smith and Waterman. (Needlmann & Wunsch 1970; Sellers 1974; Gotoh 1982; Watermann 1984)

Dynamic programming computation is extremely time consuming, therefore many dedicated machine projects have arisen. (Gokhale *et al.* 1991; Chow *et al.* 1991; White *et al.* 1991)

We have evaluated an implementation of the Needleman and Wunsch algorithm on our hardware. We find that a scanning speed of only 3 million nucleotides per second is possible. Moreover, the computation only takes place on a band stretching along the main diagonal of the matrix. This adds up to an under optimal computation heuristic on the whole of the matrix.

On its own, dynamic programming remains too computing power consuming to search efficiently in large data bases, even with a speed up from specialized systems. This is even truer as several rounds are necessary to validate the results. Therefore, another kind of algorithm that deals better with the trade offs between scanning speed, flexibility and cost (in development time and finance) must be used for a specialized machine. The pattern matching algorithms fulfills these requirements.

Pattern Matching

FAST, BLAST and FLASH are three examples of pattern matching algorithms, they are significant because of the extent to which they have been distributed. The algorithmic choice on which they were based were guided by computing time considerations, in terms of length of the data base, but also the nature of patterns, their number and length. They illustrate the state of the art on how to obtain the best performances on sequential machines. They establish a hierarchy in the structure of the words searched for. The differences between algorithms lie mainly in the way they look for a fragment, as well as the law of acceptance or rejection.

These three algorithms are based upon the splitting up of the pattern searched for into small segments. In the case of FLASH this is even done in the base.

To summarize the characteristics of these three algorithms they allow:

- A comparison of a sequence against a base of sequences.
- A splitting up of the sequence searched for into smaller pieces. They are continuous, of size two for FAST, four for BLAST and discontinuous for FLASH.
- A search based upon lookup tables.
- The filtering by the means of a minimum score of the sequences possessing the most fragments.

Califano and Rigoutsos's paper (Califano & Rigoutsos 1993) provides an excellent statistical evaluation of the BLAST and FAST pattern definition, compared to their own in FLASH. One of the interesting points displayed by Califano and Rigoutsos is that the increase in sensitivity of the method is related to the nature of the fragments, this respects our intuition.

Besides, it is better to use discontinuous segments to characterize the information that is contained in the sequence and thus optimize the search.

The term of segment has become inappropriate because inserting spaces in the fragments has made them

become more complicated, therefore we prefer to speak about a motif. The motifs are still very simple in the case of FLASH but we intend to complexify them.

The Generic Motif

In what follows we summarize the most important characteristics of a pattern matching algorithm.

- The scanning of several hundreds or even thousands of motifs. Moreover, a degree of flexibility in the description of the motif is required to stretch the algorithm's research field. This implies the motifs must not only be words but rather signatures of their belonging to a class of words.
- Sufficient velocity. This implies a few seconds to scan current bases, as in the near future the sequencing aims at entering whole genomes that would increase the volume to several billions of nucleotides (3.5 billion of nucleotides for the entire human genome).

The motif isn't described any more from an alphabet of nucleotides or amino acids even extended with joker characters, but from a list of authorized letters at a position. A motif becomes an array of lists of letters. The motifs must also be rather short, but this isn't critical. Indeed, Califano and Rigoutsos have demonstrated that the motif must have a length around 10 for the search on DNA and 5 for the proteins, to obtain a scanning with a good sensitivity.

In order to take into account these considerations, the motifs are looked for with vectors of binary substitution. The motifs are formed by a set of Binary Substitution Vectors (BSVs). A BSV is simply the list of authorized letters at the position of the vector. A motif of length N on an alphabet of L letters is made up by a list of N BSV of L bits.

Alphabet	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
A	1	0	0	0	1	0	1	1
T	0	0	1	0	1	1	0	0
G	0	1	0	1	1	0	0	0
C	1	0	0	0	1	0	0	1

This motif of length 8 on the DNA alphabet can recognize words like AGTGCTAA or CGTGATAC and so on...

This motif structure has several advantages. On one hand it encompasses the FAST, BLAST and FLASH motifs and at the same time it maintains a certain simplicity and a direct access to the underlying semantic. This makes the production of motifs by a program quite easy. On the other hand, these motifs can be used as anchor points for multiple alignment (Gracy, Chiche, & Sallantin 1992), searching by profile, or for producing examples for learning systems. The latter exploits the possibilities of systematic exploring with motifs of the size of a data base to find consensus fields in a set of sequences. (Mephu Nguifo & Sallantin 1992)

To adress the problem of sensitivity of the search, let us first notice that it depends only of the discriminant power of the motif. We have set the maximum length of a motif to be 16 BSVs. Indeed the most specific motif (ie with only one letter at each position) will be found, in average, once in a random sequence of length $4^{16} \approx 4.10^9$, which is greater than the size of the human genome. Thus the selectivity of the motif can be tuned from a probability of one (ie the motif match with any sequence in the data base) to a probability less than (data base length)$^{-1}$. Furthermore the selectivity can be tuned independentely of the efficiency of the search.

A^2R^2 Algorithm and Its Implementation

The A^2R^2 algorithm aims at searching exhaustively on the whole of a GSDB, for the situation of words or short sequences. The motifs looked for, described in the previous section, can be established either by the user, or automatically by a software.

Introduction to Reconfigurable Hardware

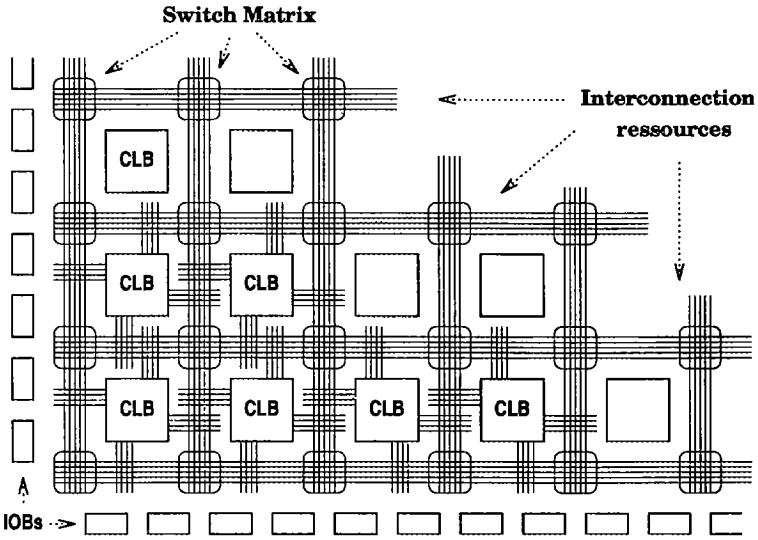

Figure 1: View of part of Xilinx FPGA architecture

The reconfigurable hardware concept, as well as its implementation with Field Programmable Gate Arrays (FPGAs) was introduced by different teams at the end of the Eighties. Vuillemin, Lopresti and Kean described various systems based on this concept. (Bertin, Roncin, & Vuillemin 1989; Gokhale *et al.* 1991; Gray & Kean 1989) In fact, this concept had been latent in the literature since the Seventies. One of the first to have expressed it was Schaffner (Schaffner 1978) in 1972. However, it is only with the arrival of the SRAM based FPGA in 1985 by Xilinx that it became possible to implement this concept. The characteristics of reconfigurable hardware are, a great development facility, together with a flexibility that is only encountered in programmable systems. The level of performances reached by a reconfigurable hardware are those of a dedicated system. The FPGA is mainly used because of its aptitude for rapid prototyping.

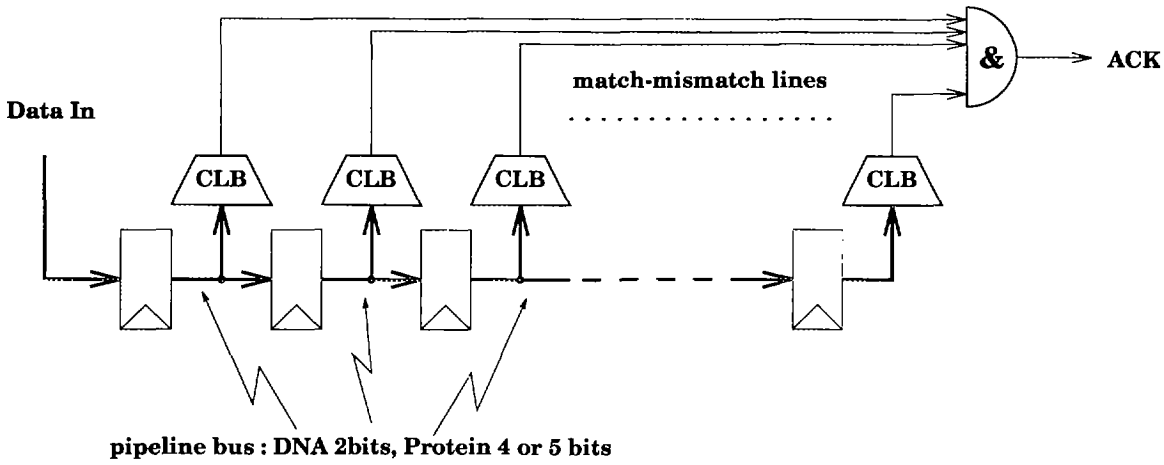

Figure 2: Scheme of a single motif detector

Basically a FPGA is a grid of small memories 16x1 bits interconnected by a network across the whole grid. These small memories are lookup tables that can implement any logic functions of four variables. In the FPGA that we used, the lookup tables are grouped by two with two additional 1 bit registers and form a CLB (Configurable Logic Block) the basic element of the Xilinx FPGA. The reader must refer to (Fagin & Watt 1992) for a comparison between supercomputer, ASIC[2] and FPGA based hardware.

The Experimental Platform

The experimental platform is constitued by a DEC workstation and a co-processing card attached to it. This card contains 22 Xilinx FPGAs and 4 MBytes of SRAM. 16 FPGAs are connected in a 2D array with local interconnection and memory buses. Therefore 5120 [3] binary fonctions, equivalent to 1 bit arithmetic and logic unit, can be evaluated, at each cycle, in the array. The maximum bandwidth between this array and the memory is 320 MBytes/s with an access time from SRAM to FPGAs of 50ns. The hardware is connected to a DecStation 5000/240 by a bus TurboChannel at 100 MBytes/s.

On this UNIX workstation the design is developed in C++ and translated through Xilinx tools in a bitstream configuration ready to be loaded on the FPGAs. The bitstream configuration can be considered as a unique nanoinstruction of 1.4 Megabits width that loads itself in less than 50 ms on the card.(Touati 1992)

The implementation on a Reconfigurable Hardware

Let us now examine how the detection of a given motif is translated into the hardware. The key point is to fit a Binary Substitution Vector with a lookup table. Then a 4x1 (20x1) bit lookup table implements a nucleotide (amino acid) BSV. This is due to the number of bits necessary to code the nucleotide or amino acid alphabet. As we have already pointed out one CLB contains two 16x1 bit lookup tables. One CLB can implement two nucleotide BSV or one amino acid BSV by bringing together each of their lookup tables of 16 bits. A motif is made up of a certain number of BSV therefore a motif detector is made up of several CLBs whose outputs are combined by an AND gate. See figure 2. In fact many optimisations can be made in order to increase either the speed or the number of motif detectors. But we don't intend to bother the reader with too many electronics details of the implementation.

The data base is injected nucleotide by nucleotide sequentially and is pipelined in the FPGA in such a way that a new word can be presented at each cycle to the motif detector. The pipeline can be seen as a window that moves along the sequence. Once a motif has been recognized, a signal is sent to the host station. The number associated with the motif and its position in the data base are the relevant informations sent to the host station.

The whole design can be considered as a sieve in which each hole fits a motif. The number of motifs that can be looked for simultaneously on the card is 512. In the case of amino acids the motifs are up to 8 BSV long, whereas the nucleotide motifs are up to 16 BSV long. The maximum speed in this case is constraint by the FPGA's intrinsic performance and is around 50ns per cycle.

One must point out that the motifs are coded in hard in the FPGA design. This is not a handicap since only the lookup tables must be changed. From a functional point of view this comes to breaking up the whole motif detector into elementary functions. For a given set of motifs that need to be matched, the only changes involved are those of the functions that code for the substitution vectors. The overall function decompo-

[2] Application Specific Integrated Circuit

[3] 16×320 : 16 FPGA, 320 CLBs per FPGA

sition is not changed. This can be done by writing straight into the bitstream configuration of the FPGAs as the motifs are generated. The figure 3 presents the two phases of this process as it can be seen by the end user.

If the search does not require more than 256 motifs the design can run at the highest speed supported by the I/O. A new character from the GSDB can be injected every 40 ns into it. A flow of 25 million nucleotides per second is achieved. The parallelism in this case is massive, 4k comparisons per cycle which represents 100 Giga operation.s^{-1}. To compare these results with a sequential machine, a program obtaining exactly the same results has been implemented on a DEC station. The most favorable situation for the sequential machine has been considered, that is the comparison where there is no matching at any position. This reduces the parallelism really useful to 250 comparaisons per cycles. Eight instructions are carried out on the workstation to make a comparison and 2000 instructions are necessary to execute what the hardware does in one cycle. Therefore, under the optimistic assumption of one instruction per cycle, a minimum speed up of 2000 times is reached. In fact, on the basis of realistic data (ie not only mismatches) the speed of the reconfigurable hardware is more than 5000 times that of the workstation.

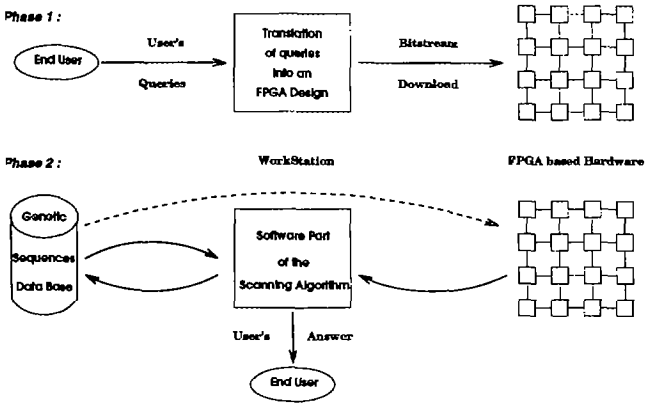

Figure 3: The two phases of a data base scanning

Related Study and Discussion

We present our dedicated hardware and software as well as four other specialized systems. Two were realized with full custom ASIC and the two others with FPGA. For a fair comparison the reader has to take into account that the SPLASH 2 project uses FPGA of a more recent technology than the 3 others and than our hardware. In fact, we cannot compare these different systems because each of them uses a particular algorithm and there is no common quality benchmark available for these 4 systems. The four systems implement different algorithms with the same goal; searching for a sequence in a data base.

- **BISP** : The BISP system consists of a Motorola 68020, SRAM and EEPROM memories and 48 BISP chips in the full configuration. The BISP chips integrate 400000 transistors, 75% is implemented with full-custom logic, and runs at 12.5 Mhz for an 1 μm CMOS process. It implements 16 cells of systolic array conceived for dynamic programing. (Chow *et al.* 1991)
- **BioSCAN** : The BioSCAN chips, 650000 transistors, integrates 800 cells of a systolic array. It runs at 32 Mhz but needs 16 cycles to execute one step of the systolic algorithm. A 1.2 μm CMOS process is used for the design technology. Only one chip is required for the scanning. (White *et al.* 1991)
- **SPLASH 1** : is a ring of 32 XC3090 interconnected with a hoststation through two FIFOs. 744 cells are implemented in a one dimensional mesh that runs at 1 Mhz. (Gokhale *et al.* 1991)
- **SPLASH 2** : This new version of SPLASH looks like an SIMD supercomputer and is based on the new generation of FPGA from Xilinx. The performance indicated in the table 1 are for one board; 17 FPGAs interconnected to their nearest neighbour in a ring fashion or through a crossbar. The algorithms are the same as SPLASH 1; a kind of dynamic programmation. (Hoang 1993)

In order to take into account these differences we compared the five systems on the same basis. That is the time it would take each system to search for a 500 nucleotide DNA sequence in a data base. In table 1 one can see the number of chips required to implement the heart of the algorithms, the scanning speed of a data base in mega nucleotides per seconds, and the speedup over a workstation claimed by the authors.

From this table it stands out that the reconfigurable hardware concept associated with our generic motif reaches and even over-takes the performances of a custom ASIC. Moreover the comparison between A^2R^2 and SPLASH shows that our implementation makes a better deal with the trade off between area efficiency and speed.

Let us now examine the performances of FLASH in terms of scanning speed. Califano and Rigoutsos (Califano & Rigoutsos 1993) report they require 24 seconds to match a 100 nucleotides DNA sequence against the Genbank. For a similar search with BLAST they report a time of 5 minutes. Our system does the same job in only 4 seconds.

To compare our result to FLASH we have to take into consideration another parameter, the additional cost of a dedicated hardware to the cost of a workstation versus the cost of a workstation alone. However FLASH compiles the data base in a huge lookup table. For example in Genbank, 10^8 nucleotides are compiled

System name	Nb chips required	Scanning speed	Speed up over a WS
BISP	32 ASIC	12M nuc.s^{-1}	4500 (i)
BioSCAN	1 ASIC	2M nuc.s^{-1}	1000 (ii)
SPLASH 1	32 XC3090	1M nuc.s^{-1}	290 (i)
SPLASH 2	16 XC4010	12M nuc.s^{-1}	2500 (iii)
A^{2}R^2 250 motifs	**16 XC3090**	**25M nuc.s^{-1}**	**5000** (vi)
A^{2}R^2 500 motifs	**16 XC3090**	**20M nuc.s^{-1}**	**10000** (vi)

i) SparcStation I: 12.5 Mips, ii) Sun 4/370: 8 Mips, iii) Sparc 10/30GX: 60 Mips vi) DECstation 5000/240: 40 Mips

Table 1: Performances of Systems Dedicated to Molecular Genetics

in a 9 Gigabytes lookup table that does not fit in the main memory of a workstation. The user of FLASH then needs an extra 9 Gigabytes of external storage and would require even more in the future. This is because the lookup table for the whole human genome will require around 0.5 Terabytes of disk storage. The price of that external storage is higher than the price for our hardware (equivalent to the price of a workstation). Moreover the performances of integrated circuits continue to improve dramatically and the performance of the I/O subsystem has not kept pace. (Patt 1994)

Algorithmic versus Dedicated Hardware One question remains : how far can hardware solutions go without new advances in the computational complexity of algorithms ?

In (Vuillemin 1993) Jean Vuillemin introduced the main equations one should bear in mind.

> $P = G \times F$ where G is the number of gates which one can effectively fit within a unit area by the end of the year y, F is the frequency at which one can reliably operate such gates and P the computing power of the resultant system. By the end of the next technology year $y' = y + 1$, the computing power increases according to $G' = \alpha_g^2 G$ and $F' = \alpha_f F$. The last 25 years the overall increasing factor of the computing power has remained equal to two and by choosing $\alpha = \alpha_g = \alpha_f$, a good approximation, we find $\alpha = \sqrt[3]{2}$.

Let us now apply the above equations to A^2R^2 and its implementation. The FPGA used are in $1.2\mu m$ 1990 CMOS technology and the board was realized and finished in 1992. On this experimental platform A^2R^2 operates at 25Mhz and uses 5000 CLBs. So by the year 2000 with a 1998 FPGA technology we will implement A^2R^2 on a $\alpha^{2\times 8} \times G \approx 2.10^5$ CLBs and operate at $\alpha^8 \times F \approx 150$ Mhz.

But all of the hardwares will not completely take advantage of this technology increase in performances. Jean Vuillemin demonstrated that reconfigurable hardware based system effectively grow by a factor two each year whereas general purpose microprocessors grow only by a factor of two each three years. This is because the peak performance of a microprocessor is ultimately limitated by the frequency of the clock no matter the amount of gates[4] that were used to make the microprocessor.

The question is now whether current algorithms can exploit the whole hardware's increase in computing power.

If we focus on data base scanning we can see two kinds of parallelism (Miller, Nadkarni, & Carriero 1991). The first one is obvious, since the data base is composed of a huge amount of independant[5] sequences we can divide the initial data base in many smaller data bases and search independantly in each of these data base. The other kind of parallelism can be found in the pattern matching algorithm itself. A^2R^2 is an example of highly parallelized pattern matching algorithm. If we add the flexibility of reconfigurable hardware we can balance between two ways of using the increase in gates by the year 2000 :

- Increasing only the number of motifs scanned : $250 \times \alpha^{16} \approx 10000$ motifs at a scanning speed of 150Mnuc.s^{-1}.
- Increasing only the scanning speed of the data base for the actual numbers of motifs (250). Therefore the data base is split in $\alpha^{16} \approx 40$ data bases resulting in a 40 times 150 Mnuc.$^{-1}$ (ie 6 GIGAnuc.s^{-1} or scanning the entire human genome in half a second)

To conclude, under the assumption that no breakthrough in integrate circuit technology will take place, a computing power improvement of two orders of magnitude must be gained by the year 2000, thanks to the reconfigurable hardware. Whereas only an optimistic 6 times improvement should be gained by general purpose workstation in the same time.

Conclusion and Future Work

We demonstrate that the couple formed by A^2R^2 and reconfigurable hardware system achieves very high performances. Indeed, a genomic sequence data base of 1 billion nucleotides is scanned in less than a minute.

The following statements summarize the advantages of both A^2R^2 and reconfigurable system:

- **Efficiency**, due to the massive parallelism achievable without any restriction on the quality of the results produced.

[4]The number of gates only control the ratio (average / peak) performances.

[5]for the pattern matching algorithm

- **Generality**, because the motifs structures include patterns previously used by FAST, BLAST and FLASH algorithms.
- **Simplicity**, thanks to clear semantics, the motif can be generated by hand or automatically by a program.
- **Scalability**, the increase by a factor two each three year in speed and a factor of four each three years in density of the new generation of FPGAs gives one matter to think about the future use of reconfigurable hardware.
- **Flexibility**, the motif detector can be seen as a building block for more sophisticated pattern matching algorithms.

One restriction remains, the quality of the result is directly connected to the motifs given to A^2R^2 and therefore brought back to the user the responsability of the quality of the search.

We think that the speed sustained by our systems opens a new way for high level genomic analysis computation. Indeed, Artificial Intelligence algorithms because of their nature need a huge amount of computing power. Usually the main drawback of such algorithms is the latency of data base request. With A^2R^2 we offer a solution to this problem as well as the possibility of exhaustive exploration of the current and future data bases. Moreover, the acceleration of the time consuming subroutine of pattern matching methods allows a better control of the result. Statistical methods, like Monte Carlo, can be used to validate knowledges obtained by learning methods in AI systems.

We are now concentrating our efforts on speeding up the main part of AI algorithms produced by our team. The first candidate is the Galois lattice algorithms used in the LEGAL system. (Mephu Nguifo & Sallantin 1992) And also implement in hardware quality analyse of results produces by A^2R^2 and other algorithms.

Acknowledgements

A great thanks to P. Bertin, C. Boegner, P. Boucard, H. Touati and J. Vuillemin for helping us along the way, and to the anonymous referees for their valuable suggestions.

References

Bertin, P.; Roncin, D.; and Vuillemin, J. 1989. Introduction to programmable active memories. In *Systolic Array Processors*. J. McCanny et al. 300–309.

Califano, A., and Rigoutsos, I. 1993. FLASH: A fast look-up algorithm for string homology. In *Procc. of the First Int. Conf. on Intelligent Systems for Molecular Biology*, 56–64. AAAI.

Chow, E. T.; Hunkapiller, T.; Peterson, J. C.; and Zimmerman, B. A. 1991. Biological information signal processor. In *Proceedings of the Int. Conf. on Aplication Specific Array Processor*, 144–160.

Fagin, B., and Watt, J. G. 1992. FPGA and rapid prototyping technology use in a special purpose computer for molecular genetics. In *Procc. of the Int. Conf. on Computer Design*, 496–501. IEEE.

Feng, D.; Johnson, M.; and Doolittle, R. 1985. Aligning amino acid sequences: comparison of commonly used methods. *J. of Molecular Evolution* 21:112–125.

Gokhale, M.; Holmes, W.; Kosper, A.; Lucas, S.; Minnich, R.; Sweely, D.; and Lopresti, D. 1991. Building and using a highly parallel programmable logic array. *IEEE Computer* 24(1):81–89.

Gotoh, O. 1982. An improved algorithm for matching biological sequences. *J. of Molecular Biology* 162:705–708.

Gracy, J.; Chiche, L.; and Sallantin, J. 1992. A modular learning environnment for protein modeling. In *Procc. of the First Int. Conf. on Intelligent Systems for Molecular Biology*, 145–153. AAAI.

Gray, J. P., and Kean, T. A. 1989. Configurable hardware: A new paradigm for computation. In *Proceedings of the 1989 Decennial Caltech Conf.* C. L. Seitz. 279–295.

Hoang, D. T. 1993. Searching genetic database on splash 2. In *Proc. of the Workshop on FPGAs as Custom Computing Machines*, 185–191. IEEE.

Lemoine, E. 1993. Reconfigurable hardware for molecular biology computing systems. In *Procc. of Int. Conf. on Application-Specific Array Processors*, 184–187.

Mephu Nguifo, E., and Sallantin, J. 1992. Prediction of primate splice junction gene sequences with a cooperative knowledge acquisition system. In *Procc. of the First Int. Conf. on Intelligent Systems for Molecular Biology*, 292–300. AAAI.

Miller, P. L.; Nadkarni, P. M.; and Carriero, N. M. 1991. Parallel computation and FASTA: confronting the problem of parallel database search for a fast sequence comparison algorithm. *CABIOS* 7(5):71–78.

Needlmann, S., and Wunsch, C. 1970. A general method applicable to the search for simillarities in the amino acid sequence of two proteins. *J. of Molecular Biology* 48:443–453.

Patt, Y. N. 1994. The I/O subsystem : a candidate for improvements. *IEEE Computer* 27(3):15–16.

Schaffner, M. 1978. Processing by data and program blocks. *IEEE Transactions on Computers* 27(11):1015–1028.

Sellers, D. 1974. On the theory an computation of evolutionnary distances. *S.I.A.M. J. Appl. Math.* 26(4):787–793.

Touati, H. 1992. Perle1DC: a C++ library for the simulation and generation of DECPeRLe-1 designs. Technical Report TN4, Paris Research Laboratory.

Vuillemin, J. E. 1993. On computing power. *Lecture Notes on Computer Sciences* 782:69–86.

Watermann, M. 1984. General methods of sequence comparison. *Bull. Math. Biol.* 46:473–500.

White, C. T.; Singh, R. K.; Reintjes, P. B.; Lampe, J.; Erickson, B. W.; Dettloff, W. D.; Chi, V. L.; and Altschul, S. F. 1991. BioSCAN: A VLSI-based system for biosequence analysis. In *Proceedings of the IEEE Int. Conf. on Computer Design*.