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

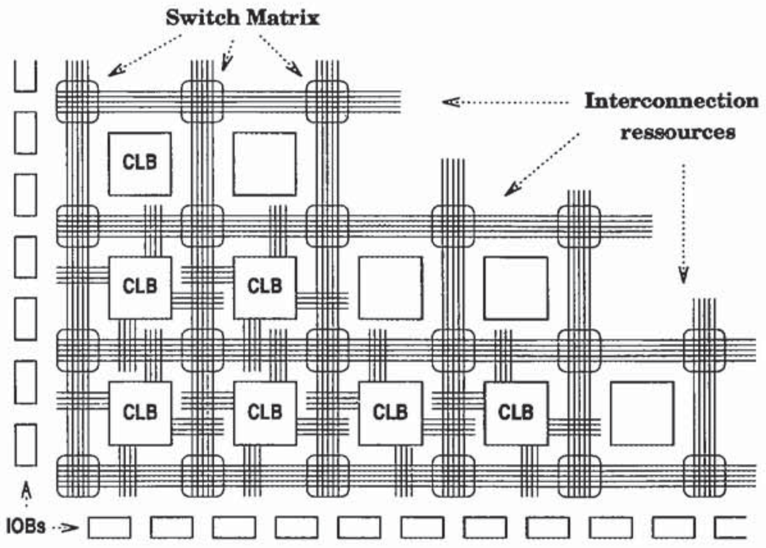

Figure 1: View of part of Xilinx FPGA architecture

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.

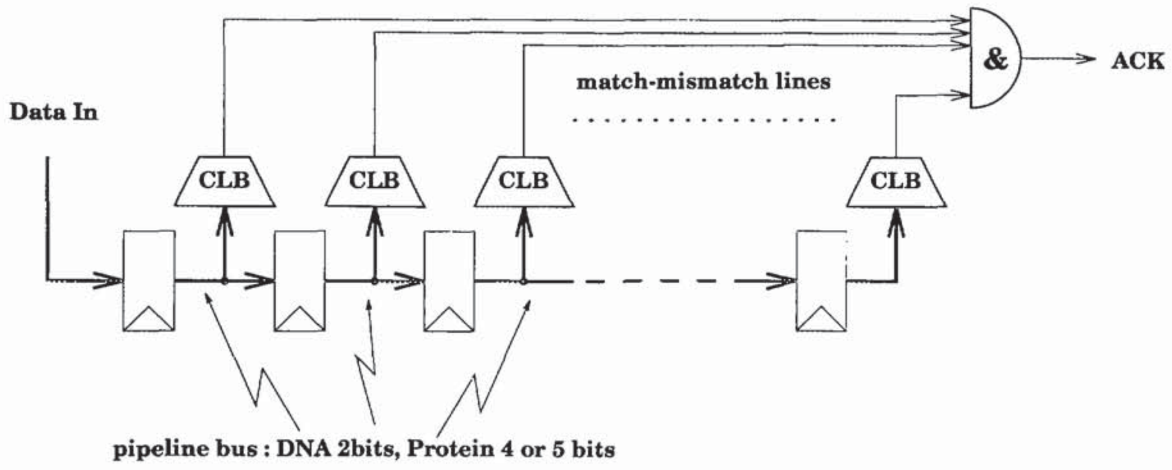

Figure 2: Scheme of a single motif detector

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.

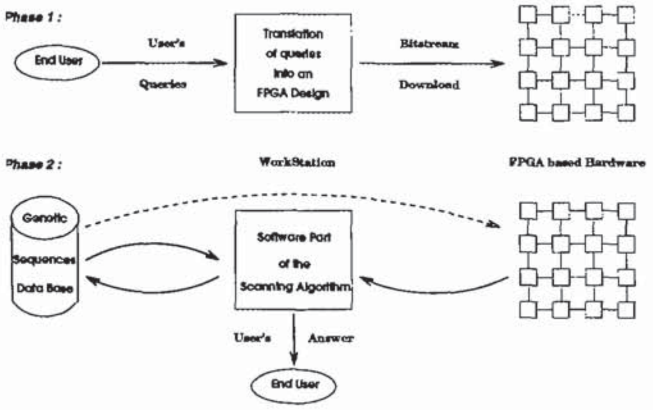
Phase 1 :
End User
User's
Queries
Translation of queries into an FPGA Design
Bitstream
Download
Phase 2 :
WorkStation
FPGA based Hardware
Genetic Sequences Data Base
Software Part of the Scanning Algorithm
User's
Answer
End User

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

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.