

A reconfigurable parallel disk system for filtering genomic banks

D. Lavenier, S. Guyetant, S. Derrien
IRISA/CNRS
Campus de Beaulieu
35042 Rennes cedex, France

S. Rubini
Brest University
20, avenue Le Gorgeu
29285 Brest cedex, France

Abstract *Scanning the genomic database is a common task, daily performed by thousands of researchers for extracting new knowledge from this huge amount of data. Today, this represents tens of Gigabytes. Tomorrow, according to the genomic data exponential growth, the volume of information to manipulate will reach Terabytes. In this paper we propose a parallel disk system whose originality is to attach reconfigurable computation capabilities near the disk for providing on-the-fly data filtering. Speed-up is provided both by accessing tens of disk in parallel and by hardwiring efficient filters. A 48 disk system is currently assembled for experimentation.*

Keywords: genomics, parallel disk, on-the-fly filtering, database search

1 Introduction

Genomic databases are growing exponentially: every year the number of genomic sequences (mostly DNA and RNA sequences) is doubling. The last GenBank and EMBL releases (dec. 2002) contain more than 20 millions of sequences, representing 28 billions of nucleotides [4] [1]. GenBank and EMBL are part of the International Nucleotide Sequence Database Collaboration, which also comprises the DNA DataBank of Japan (DDBJ). These three organizations exchange data on a daily basis and new releases are made every two months to include new data coming from worldwide research institutes.

These banks are routinely and daily scruti-

nized by thousands of researchers. Actually, a common task of the molecular biology is to try to assign a function to an unknown gene or an unknown protein. More precisely, proteins are synthesized within the cells of plants and animals. To be active, a protein must adopt a specific 3D shape related to its sequence of amino acids. The shape is important because it determines the function of the protein, and how it interacts with other molecules. It is assumed that two proteins with identical functions may have similar 3D structures, leading to a similar sequence of amino acids. Even if this hypothesis is not always verified, a great number of algorithms have been proposed to quickly extract significant alignments from the banks, i.e. sequences (or portion of sequences) having a high similarity with a query sequence.

BLAST [20] [15] has steadily become the reference software for exploring genomic banks. Large databases can be quickly and easily screened to detect similarity with a query sequence. The algorithm is fast: the main idea is that a statistically significant alignment is likely to contain a high-scoring pair of aligned words. Thus, the algorithm first detects small anchors of full similarity (identical words of W characters between the query and the sequences of the bank), then try to extend in either direction in an attempt to generate an alignment with a score greater than a threshold value. Larger the anchor, faster the algorithm, but smaller the sensitivity: sequences without (at least) a common word of W characters (anchor) are not reported.

This type of algorithm belongs to the string comparison algorithm class whose first implementation was using dynamic programming techniques [22]. The principal advantage comes from the speed-up: 2 or 3 orders of magnitude are gained. It allows a researcher to scan an entire bank in a very reasonable time, together with a correct sensitivity.

As mentioned above, the BLAST algorithm, and many other algorithms such as PATTERN-HUNTER [5] or CHAOS [3], proceed in two steps: first they seek for anchors, then they extend them into alignments. The load balancing between this two tasks depends on the quality of the anchors. Since the alignment extension can be time consuming, the goal is to limit the number of hits by providing anchors of good quality, i.e. reflecting a good probability of generating a significant alignment. Unfortunately, more complex the anchor detection, longer the computation time.

The central idea of our project is to hard-wire the first step into a reconfigurable system. More precisely, we directly filter the genomic data at the disk output, in order to provide the host computer with only relevant data. The challenge is to process data at the output rate of the disk and to forward only a low percentage of the database together with anchoring informations. The filter implemented on the reconfigurable system both depends of the data to process (DNA, protein) and the application (similarity search, motif search, etc.).

The idea of attaching computation capabilities near the disk for providing on-the-fly data filtering is not new. The SmartDisk project [10], the Active Disk project [11, 13] or the IDISK project [14] are examples of such investigation. All of them are motivated by a major trend: hard disk controllers are designed with an increasing amount of general purpose processing power and on-chip memory. Today, most of the processing power is devoted to disk scheduling and other duties, but the next generation of controllers will contain powerful embedded processors, able to perform extra tasks [12]. Thus, filtering the data by pushing computation closer to the storage system has be-

coming an attractive solution for providing reduction in data movement through the I/O system.

Compared to these projects, we differ by the type of processing power we attach to the hard disk. Instead of an embedded processor we propose to connect a reconfigurable system based on a low cost FPGA component. The main advantage is that the anchoring-search algorithm can be highly parallelized on simple hardware structures [18], allowing on-the-fly filtering of the genomic data.

Another point to consider is the time for accessing the genomic data. The goal is to design efficient filters ; therefor, the quantity of data transmitted to the processor is expected to be low. In that case, the processor is likely to be in a starving situation, with no data to process. The computation time is thus bounded by the time to access and filter the data coming from the disk. To reach a good balancing between the post-processing and the filtering process, several disks are attached to the processor, and a reconfigurable processing system is joined to each disk.

The complete system is thus made of a front-end computer connected to a bunch of hard disks coupled to reconfigurable processing and interconnected through a local network – in our case an Ethernet network. Depending of the type of the query, an adequate hardware filter is first downloaded to the FPGA component before scanning the banks. The filtering occurs locally and results are send back to the front-end computer for further post-processing.

A prototype board has been designed and successfully tested. We are currently assembling a reconfigurable parallel disk system of 48 boards. Genomic applications range from similarity search (as explain above) to complex motif extractions based on regular expression, or context free grammar.

Commercial reconfigurable machines exist to perform various genomic tasks, especially sequence comparison. The DeCypher engine from TimeLogic ¹, for example, implements

¹<http://www.timelogic.com>

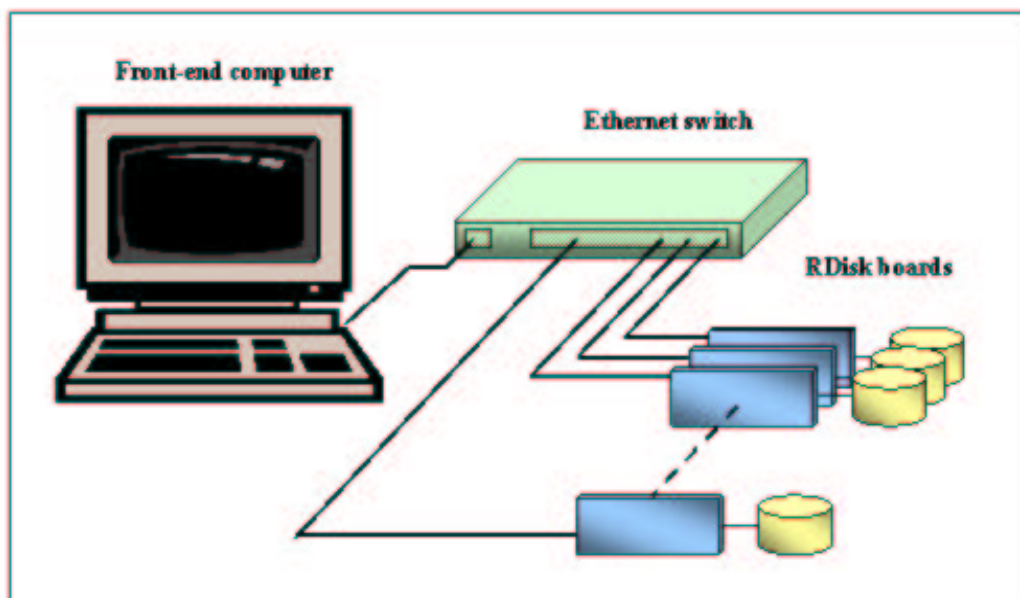


figure I: The Reconfigurable Parallel Disk System

the BLAST software. The purpose of this accelerators is to efficiently manage huge time-consuming computation such as bank to bank comparison, or genome to genome comparison. It is not particularly suited for speeding-up the scan of large databases. The BioXL/H system from Compugen² is another hardware product dedicated to genomic computations ; it also uses the reconfigurable technology. Again, this system focuses on speeding-up time-consuming algorithms, such as the Smith & Waterman algorithm [22], rather than providing a fast database search.

The rest of the paper is organized as follows: section 2 gives an overview of the system and a short description of the RDisk board we have developed. Section 3 details the reconfigurable SoC implemented in the FPGA component. Section 4 estimates the performance of the reconfigurable parallel disk system and section 5 concludes this paper.

²<http://www.cgen.com>

2 The reconfigurable parallel disk system

2.1 Overview

Figure I gives an overview of the reconfigurable parallel disk system. It is composed of a set of reconfigurable boards interconnected through an Ethernet network. All the boards are linked to a single Ethernet switch, also connected to the host system. The number of reconfigurable boards is not yet frozen: experimentations need to be carried out to tune the system to an optimal number of elements according to the selectivity of the filters. It is also expected to connect this parallel disk system to a grid dedicated to genomic applications [2]. In that case, a 48 disk system can be re-organized, for example, into 3 nodes of 16 disks, each node having its own front-end computer acting as a specific node of the grid.

The choice of Ethernet relies on its easiness of implementation and its low cost. In addition, an efficient filtering should highly limit the quantity of information coming from the disks. In that case, a high speed network is not required. A light communication protocol

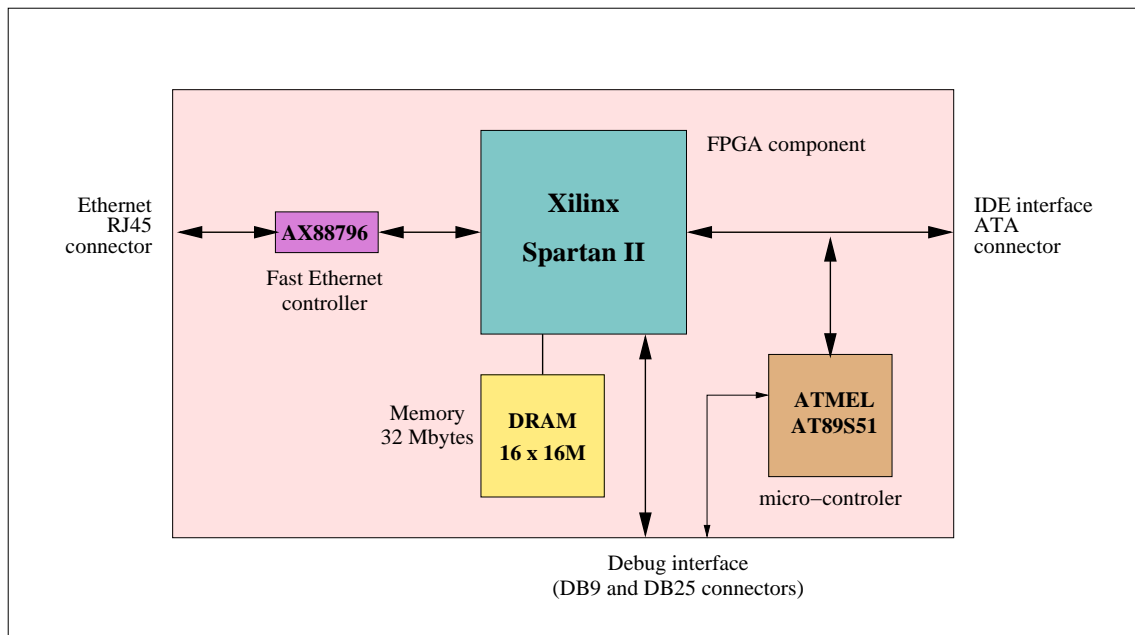


figure II: The Reconfigurable Board (RDisk board)

is currently developed to support efficient and fast exchange of messages. In our communication scheme, communications between two reconfigurable boards are not allowed: the boards only communicate with the front-end computer.

The banks are distributed over all the disks. A query is processed as follows: if the type of query differs from the previous one, then the boards are reconfigured. In that case, the front-end computer specifies the new configuration to download. The bitstreams are duplicated on the disks, and the boards manage themselves their own reconfiguration. Once this task is performed, the query is broadcasted to the boards, then locally processed. Results (i.e. relevant sequences) are sent back to the front-end computer before further treatments.

A configuration is attached to a filter. For example, running a default BLAST search first configures the RDisk boards with filters detecting anchors of 11 characters. If the next query requires a more sensitive search, then new filters detecting smaller anchors are downloaded. In this case, a small overhead due to the FPGA

reconfiguration time is introduced. In practice, this overhead is small compared to the scanning time and, thus, do not penalize the overall performances. Hundreds of often-used configurations are stored on the hard disk, but at any time, the host can broadcast new bitstreams on each disk.

2.2 The reconfigurable board

The heart of the reconfigurable parallel disk system is the reconfigurable disk board (called RDisk). Figure II presents its structure. It is architected around a low cost FPGA component: a Xilinx Spartan-II (XC2S200) having a capacity equivalent to 200,000 system gates [7] for a maximum frequency of 200 Mhz. Attached to this component is a SDRAM memory of 32 MBytes.

The board is driven by a standard 8-bit micro-controller (8051-compatible). The role of this device is first to initialize the system in a coherent state when powering on the board. Then it handles the configuration of the FPGA component directly from the hard disk, thus avoiding the need of the classical EEPROM

+ CPLD configuration solution. This configuration aims to provide the Spartan-II with a primary architecture allowing communications with the front-end computer. After the download is completed, the Spartan-II get the full control of the board and the micro-controller goes sleeping till the next reconfiguration: The FPGA wakes the CPU and tells him where to find on the drive its next bitstream.

The network connection is handled by the AX8896 Fast-Ethernet controller. This low price chip, NE2000 compatible, provides 10/100Mbps MAC, PHY and transceiver operations. We use a common ISA bus to interface it to the Spartan-II. The serial and parallel interfaces, available respectively through the D-B9 and DB25 connectors, are used for debug purpose and will be removed later. The clock that drives the FPGA is set by a programmable clock generator (not shown fig. II). The frequency can thus be set as of the maximum allowed by the synthesis step. For example, by the time the board was under design, the processor core was running at 25 MHz. Further improvements – once the board was assembled – gave us the possibility to run the core at 50 MHz; should we need a higher speed, we always may refine the design to shorten the critical path.

A prototype board has been designed (cf figure IV) and successfully tested. It includes a standard 40 GBytes disk embedded to a printed board designed to be plugged into a storage cabinet.

The *intelligence* of the board comes from the architecture programmed inside the Spartan-II component. Its role is to act as an adaptable filter according to the type of the request. But even if the requests may significantly differ, they still share a lot of common features that can advantageously be exploited into a single architecture framework. The next section presents the reconfigurable SoC (R-SoC) we are currently developing.

3 Reconfigurable SoC

The reconfigurable parallel disk system can be requested for various types of searches, leading to configure it with adapted filters specifically tuned to that search. The primary goal of the FPGA component is thus to efficiently implement the various filters. But another important task is to handle the Ethernet communication and control the IDE interface. These tasks are independent of the filtering process, and common to all search database applications.

The architecture we implement in the Spartan-II reflects this dichotomy: one part of the hardware resources is devoted to common services – and thus identical for each application – while the second one is fully dedicated to the application. The common part is build around an embedded 16-bit microprocessor, supporting a real-time kernel, and connected to an Ethernet, an IDE, and a filter interfaces. Special attention has been paid to minimize the hardware logic in order to keep the major resources for the filters.

3.1 R-SoC architecture

Figure III depicts the R-SoC architecture. The central component is the Xr16 CPU, a 16-bit RISC processor especially designed for the Xilinx family [8]. It is a 3-stage pipelined processor (fetch, decode, execute) holding sixteen 16-bit registers. This CPU was created by J. Gray from Gray Research LCC and is part of the XSOC project³. Our Spartan-II implementation gives a 50 Mhz frequency. The Xr16 is connected to the external 32 Mbytes SDRAM memory through a memory controller. In order to achieve good performances, we enhanced the open-source version with an instruction-cache memory. In addition, a small ROM memory, initialized through the bitstream, contains a boot program able to download a larger code from the disk.

The filter is the application specific part of the SoC and should be easily integrated or interchanged. Only its interface has been spec-

³<http://www.fpgacpu.org/xsoc>

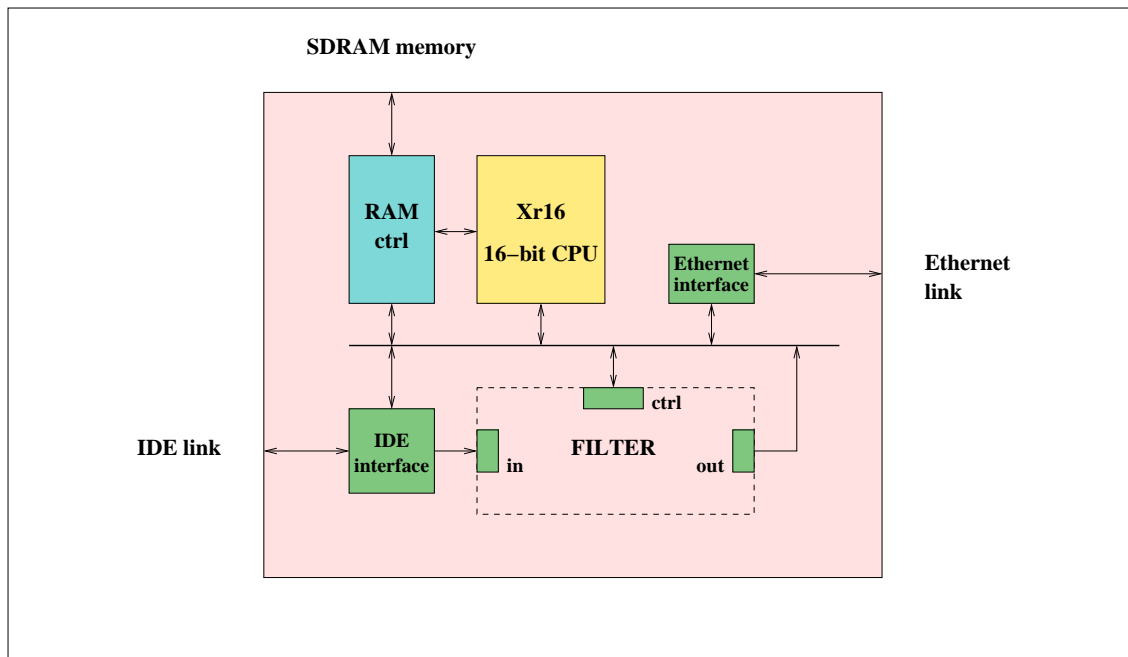


figure III: The Reconfigurable SoC

ified, and the internal logic varies greatly depending on the algorithm and the data structure. The Xr16 processor *sees* only three ports: an input port directly linked to the IDE channel, an output port connected to the system bus, and a control port. To achieve a high throughput out from the disk, the filter is fed by a direct link from the IDE controller. As our goal is to process data at the maximum sustained rate (which is roughly the product between the rotational speed and the density of data at the surface of disk platters; not to be confused with the maximum burst rate, used for advertisement), the quality of this connection is mandatory for the whole system performance. On the other hand, the output of the filter is expected to be the lowest fraction of incoming data so it can be attached to the system bus. The control interface accepts commands from the processor such as reset, register access (initialization, reading the internal state) or interruption of the data flow.

The R-SoC architecture – without the filter – takes about 1/3 of the logic resources of the Spartan-II FPGA component. The majority of

the resources is thus left to implement dedicated filters.

3.2 R-SoC programming

A real-time kernel, microC OS-II kernel [9], is currently being adapted to the R-SoC architecture. This open-source operating system is a pre-emptive multitasking system using some standard OS methods like tasks, semaphores and pre-emptive scheduling. It provides all the basic system primitives we need to support the RDisk functionalities. Furthermore, the kernel code is compact and requires less than 20 K-Bytes.

The Xr16 microprocessor is provided with a C compiler. This is a port of Hanson and Fraser's LCC [17], a retargetable compiler for ANSI C. In our case, having such a tool is essential for re-using or adapting existing C codes already written for IDE or Ethernet drivers.

Programming the RDisk board for a specific application requires to write (1) a C program and (2) a VHDL specification of a filter. The C program is the master process and control-

s all the transactions. Typically, it manages the Ethernet communication, controls the disk transfers, drives the filter, collects the results (data transfer from the filter output to the memory), etc. A C-library of high level primitives (functions) allows the programmer to efficiently steer the different elements of the R-SoC system.

The VHDL specification describes the hardware structure of the filter. As already mentioned, the filter architecture have to take into account the interface constraints: the input port is dedicated to access data from the disk, while the output port (cf figure III) is expected to be only used for transferring a small amount of data to the main memory. The designer have to kept in mind these few features to optimally design a filter fitting well into the R-SoC framework.

4 Performance

This section estimates the performance of the reconfigurable parallel disk system based on a previous study, and adapted to the hardware resource of the Spartan-II FPGA component. The idea of pre-processing the genomic data before running an advanced string comparison algorithm has been deeply investigating by the authors [18] and tested on a reconfigurable accelerator, the PeRLe-1 board [16]. The data were reading from the disk through the standard I/O bus system of the machine, stored in the main memory, then pushed to the accelerator whose task was to detect potential hits. These hits were reported to the processor, and treated for an attempt to generate an alignment. In this experiment, the Fasta software [21] had been used for post-processing the data, and the source code has been slightly modified to substitute the hit search step by a connection to the hardware filter.

A nearly identical filter can be implemented in our system. Differences rely on the new hardware capabilities of the Spartan-II component compared to the Xilinx XC3000 family used in the PeRLe-1 board. A re-design of this

filter, together with some optimizations, shows that a query sequence of 300 nucleotides can be filtered at the transfer rate of the disk. If we suppose an average disk bandwidth of 15 MBytes/second (in PIO access mode), we are able to filter 60×10^6 nucleotides per second (nt/sec): 4 nucleotides (4-letter alphabet) can be encoded on a single byte. Thus, a parallel system made of 48 disks would be able to sustain a rate of 2.88 Giga nt/sec.

We now have to consider the quality of the filter to estimate the Ethernet traffic. Several tests on the GenBank EST (Expressed Sequence Tag) division have been carried out to quantify its selectivity. An expressed sequence tag [19] is a small part of the active part of a gene, made from cDNA, which can be used to fish the rest of the gene out of the chromosome, by matching base pairs with part of the gene. The GenBank EST division contains more than 15 millions of such sequences, representing 7.8 billions of nucleotides. We run the filter with various query sequences and we measured the number of hits, i.e. the location on the EST sequences, where a potential alignment may exist.

An average of 1 sequence over 120 were exhibited to likely contain a significant alignment, that is 125,000 sequences over 15 millions. The average length of an EST being equal to 500 nt, the amount of data to move to the front-end computer represents 62.5 millions of nucleotides, or approximatively 16 Mbytes. If we suppose that the scan of the 7.8 billions of nucleotides can be performed in 3 seconds – from the above estimation – the transfer rate over the Ethernet network will be equal to 5.3 MBytes/sec. This is half of the peak transfer of a 100 Mbps Ethernet network.

We also measured the time for searching the EST database on a standard parallel system. We run the parallel version of BLAST on a SUN-FIRE 6800 multiprocessors (16 processors, 16 GBytes of memory) using different sensitivities. With a standard sensitivity (anchor of 11 nucleotides), the execution time is equal to 8 seconds. For a higher sensitivity (anchor of 7 nucleotides), the execution time jumps to 40



figure IV: The RDisk board

seconds. The time corresponds to the user time in the best condition: the 16 processors are only devoted to this task and are not disturbed by other processes.

The speed-up compared to our system can be calculated as the ratio of the execution time of the SUN-FIRE machine over the execution time of the parallel disk system. This second time is actually the scan time as estimated above (3 sec) since the post-processing can easily overlap the filtering step. A speed-up ranging from 2.5 to 13 is achieved knowing that a few sequences are not reported by BLAST when a 11-nt anchor is selected. Note that the filter does not miss any sequences, so that it provides both speed and high sensitivity.

5 Conclusion and perspectives

The reconfigurable parallel disk system is an ongoing project. A basic element – the RDisk board - has been designed and successfully tested. A complete system with 48 RDisk boards is currently assembled for real test.

One of the first applications supported by the system will be the fast scan of genomic databases for extracting similar sequences. The expected speed-up will range from 2 to 15

compared to the BLAST software running on a parallel system typically used in bioinformatics centers. If this kind of database search is intensively and daily performed, it is not the only application requiring a systematic scan of large genomic databases. We are currently investigating the search of motifs or the detection of signatures over complete genome. The query is not a sequence, but a description specifying, for example, a family of genes. This description can be formulated as a regular expression, or as a set of grammar rules. In both cases, the genome can be efficiently filtered using the reconfigurable hardware resources of the RDisk board to report only the interesting zones.

More generally, this type of system can be extended to other application domains demanding a systematic scan of large databases. For example, the content-based image retrieval perfectly suits: the purpose is to find, from a query image, similar images – or part of images – among millions of them stored in an image bank [6]. A descriptor is associated to each image and encode various features. Typically, this is a vector of real numbers defining *points of interest* in a high dimensional space. To search a base, the first step extracts a set of descriptors from the query image. Then,

this set is *compared* to all the descriptors of the database. Images from the database having closed descriptors are supposed to share some similarities with the query image. In that application, the filter computes distances between descriptors. Only the descriptors having a low distance with the query are taken into consideration for further processing.

Today, searching a one million image database takes hours of computation. Because the vector distance computation can be highly parallelized, one might expect to drastically reduce the execution time from hours to seconds. The implementation of this application to our reconfigurable parallel disk system is currently investigated by another group, here, at IRISA.

Finally, our system have to be compared with a PC cluster approach, as it is becoming a rather cheap solution to connect hundreds of processing units together. The advantages of the PC cluster is (1) that many companies propose ready to-use systems, and (2) that the programming effort for parallelization is *weak* according to the applications we consider. As a matter of fact, the database scanning belongs to the *embarrassingly parallel class* problem: a search can be split into a large number of independent tasks working on separate data, with a final step consisting in a simple merge process. Parallelization is thus straightforward and highly efficient.

However, a minimal node only made of a powerful microprocessor, a few MBytes of memory, and a hard disk drive will always be much more expensive than a RDisk board based on low cost FPGA components. The next generation of low cost reconfigurable devices will certainly enhance, bringing to the market, for example, the equivalent of the Xilinx Virtex-II Pro component for a few Euros. Then, the entire **RDisk** board will fit into a single component.

References

[1] G. Stoesser and al., The EMBL Nucleotide Sequence Database : major new develop-

ments, Nucleic Acids Research, vol 31, no 1, 2003.

- [2] D. Lavenier, H. Leroy, M. Macwing, R. Andonov, M. Hurfin, P. Raipin-Parvedy, L. Mouchard, F. Guinand, GénoGRID: an experimental grid for genomic applications, HealthGrid 2003, January 16-17, 2003.
- [3] M. Brudno, B. Morgenstern Fast and sensitive alignment of large genomic sequences Proceedings of the IEEE Computer Society Bioinformatics Conference (CSB), 2002.
- [4] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, B.A. Rapp, D.L. Wheeler, GenBank, Nucleic Acids Research, vol 30, no 1, 2002.
- [5] B.Ma, J. Tromp, M. Li, PatternHunter: Faster And More Sensitive Homology Search, Bioinformatics, vol 18, no 3, 2002.
- [6] L. Amsaleg, P. Gros, Content-based retrieval using local descriptors: problems and issues from a database perspective, Pattern Analysis and Applications, vol 4, no 2/3, 2001.
- [7] Xilinx, Spartan-II 2.5V FPGA Family: functional Description, DS001-2(V2.1), March 5, 2001.
- [8] J. Gray, Building a RISC System in an FPGA, Circuit Cellular, issue 116, march 2000.
- [9] J.J. Labrosse, MicroC/OS-II - The real time Kernel, CMP Books, 2000.
- [10] G. Memik, M.T. Kandemir, A. Choudhary, Design and Evaluation of Smart Disk Architecture for DSS Commercial Workloads, In Proceedings of International Conference on Parallel Processing (ICPP), Toronto, Canada, 2000.
- [11] M. Uysal, A. Acharya, J. Saltz, Evaluation of Active Disks for Decision Support Databases, HPCA-2000, Toulouse, France, 2000

- [12] B. Steurich, G. Born, Siemens "System-on-Silicon" Integrated hard drive Hard-disk for Controller to Gigabyte HD Drives, Siemens Tools Partners, 2000.
- [13] A. Acharya, M. Uysal, J. Saltz, Active Disks: Programming Model, Algorithms and Evaluation, ASPLOS-VIII, San Jose, California, 1998.
- [14] K. Keeton, D. A. Patterson, J. M. Hellerstein, A Case for Intelligent Disks (IDISKs), SIGMOD Record, Vol. 27, No. 3, 1998.
- [15] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman, Gapped Blast and PSI-Blast: a new generation of protein database search programs, Nucleic Acids Research, vol 27, no 17, 1997.
- [16] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, P. Boucard, Programmable Active Memories: the Coming of Age, IEEE Trans. on VLSI, Vol. 4, No. 1, 1996.
- [17] C. Fraser, D. Hanson, Retargetable C Compiler: Design and Implementation, Benjamin/Cummings, Redwood City, CA, 1995.
- [18] P. Guerdoux-Jamet, D. Lavenier, Systolic Filter for fast DNA Similarity Search, AS-AP'95, International Conference on Application Specific Array Processors, Strasbourg, France, 1995.
- [19] MD. Adams, JM. Kelley, JD. Gocayne, M. Dubnick, MH. Polymeropoulos, H. Xiao, CR. Merrill, A. Wu, B. Olde, RF. Moreno, et al., Complementary DNA sequencing: expressed sequence tags and human genome project, Science 252(5013), 1991.
- [20] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, Basic local alignment search tool, J. Mol. Biol. no 215, 1990.
- [21] W.R. Pearson and D. Lipman, Improved tools for biological sequence comparison, Proc. Natl. Acad. Sci. USA, vol 85, 1988.
- [22] T.F. Smith, M.S. Waterman, Identification of common molecular subsequences, J. Mol. Biol., vol 147, no 1, 1981.