BIOINFORMATICS APPLICATIONS NOTE Vol. 19 no. 14 2003, pages 1739–1740
DOI: 10.1093/bioinformatics/btg228

Designing hardware for protein sequence analysis

Alessandro Marongiu[1], Paolo Palazzari[2,*] and Vittorio Rosato[2]

[1]IPITEC s.r.l.—Rome, Italy and [2]ENEA, Ente per le Nuove Tecnologie, l'Energia e l'Ambiente, Computing and Modeling Unit, Rome, Italy

Received on November 4, 2002; revised on February 13, 2003; accepted on March 28, 2003

ABSTRACT

We present the architecture of PROSIDIS, a special purpose co-processor designed to search for the occurrence of substrings similar to a given 'template string' within a proteome. Actual tests show speed up figures ranging from 5 to 50 with respect to conventional general-purpose processors.

Availability: the PROSIDIS configuration file and the c code are available at http://www.enea.it/hpcn/php/rosato/

Contact: palazzari@casaccia.enea.it

INTRODUCTION

Analysis of biological sequences could benefit from the adoption of devices purposely designed to accomplish this task. Unfortunately, the design of a dedicated device requires both long development time and good skills in electronic system design, thus discouraging bioinformatics researchers from the implementation of specialized architectures. In order to avoid these drawbacks, we have developed an automatic tool which allows the fast designing and prototyping of dedicated systems: the parallel hardware generator (PHG) (Marongiu and Palazzari, 2001). It allows short design times starting from the high level description of the computation, fast prototyping and high HW re-use resulting from the adoption of the field programmable gate array (FPGA) reconfigurable technology. Use of FPGA for biological computations is also described in Yamaguchi *et al*. (2002) and in the references reported therein.

PHG implements, in a fully automatic way, a design flow which, starting from an abstract representation of the algorithm to be implemented, gives the skeleton of a program written in a hardware description language (VHDL) which completely describes the highly parallel architecture of the system implementing the algorithm. At this stage, design only requires the definition of the VHDL code describing the combinatorial behavior of the functional units. The VHDL code is then processed by standard CAD tools to obtain the FPGA configuration bit-stream.

*To whom correspondence should be addressed.

METHODS

We report the results achieved in the implementation of a dedicated hardware device for 'proteomic computation': the PROSIDIS device (PROtein SImilarity DIScovery) designed to find similarity regions within a proteome. It could be used as a building block for more complex protein analysis algorithms—far beyond the scope of this application note. The reason compelling the development of PROSIDIS is that protein analysis (as well as DNA analysis) is a time consuming task not efficiently supported by conventional processor architectures.

Even if technology is growing fast, to date FPGAs are still clocked at frequency significantly lower than the clock frequency of general-purpose processors. For this reason the algorithm parallelism has to be efficiently exploited to make the FPGA-based implementation more effective than that implemented on the general-purpose processor. The PHG tool has been designed to automatically extract the parallelism from the algorithm and to generate the skeleton of the VHDL code describing the system.

The PHG design flow is composed of the following steps:

- the algorithm to be implemented is coded through the SIMPLE (Sare IMPLEmentation) language (Marongiu *et al*., 2000). Such a language allows to specify the algorithm by means of a System of Affine Recurrent Equations (SARE) (Loechner and Mongenet, 1996).
- the algorithm is automatically parallelized through the 'allocation and scheduling' step (Marongiu and Palazzari, 2000) which produces the system architecture which is composed by a data path managed by a finite state machine (data path controller);
- the VHDL programs for data path and data path controller are automatically generated;
- designer fills previous skeleton with the VHDL code implementing the combinatorial behavior of the functional units.

The final VHDL program is synthesized through standard design automation tools producing the FPGA configuration file.

PROSIDIS faces the problem of finding the 'degree of similarity' between a short sequence $\boldsymbol{s}$ of m amino acids called peptide and a long sequence $\boldsymbol{p}$ of n amino acids representing a proteome. Such an operation is the basis of many alignment algorithms.

Typical lengths of proteome $\boldsymbol{p}$ are between 10^6 and 10^7 characters. Given a peptide of m amino acids, the PROSIDIS problem can be stated as the computation of the $n-m$ values

$$M(i) = \sum_{j=0}^{m-1} DM[p(i+j), s(j)] \quad i = 0, \ldots, n-m-1 \tag{1}$$

which give the 'similarity' between the peptide $\boldsymbol{s}$ and a segment of $\boldsymbol{p}$. $DM(a, b)$ is a weighting matrix which, for any pair of amino acids a and b, returns a 4 bit integer number representing the degree of similarity between them [we used the BLOSUM62 weighting matrix (Henikoff and Henikoff, 1992)]. During the cumulation process, the partial value of $M(i)$ is set to 0 whenever it becomes negative.

The whole design process, from the SIMPLE algorithm implementing (1) to the first working prototype, has been carried out in less than one week.

The test bed we used to implement the PROSIDIS architecture is a prototyping board equipped with a PCI interface (33 MHz), 8 MB of SRAM memory and one Xilinx Virtex XV1000 FPGA; the cost of the board is approximatively 3K$. The board is hosted by a standard PIII@550 MHz PC. Due to the large size of the FPGA device and in order to increase the computing power of the PROSIDIS architecture, we choose to implement four times the system implementing (1) hence the PROSIDIS architecture is able to perform the simultaneous comparison of a single peptide of length $m = 24$ against four proteomes of length $n = 524\,000$. The clock cycles needed to carry out the whole computation are $m + n = 524\,024$. Constrained at the speed grade of the FPGA we used (-4), the synthesized design is clocked at a frequency $fck = 30$ MHz, corresponding to a sustained computing rate of 2.88×10^9 operations per second (one operation is composed by the access to the weighting matrix and by the sum operation). Referring to the Protein Similarity Discovery problem, the core of the computation is demanded to the PROSIDIS processor while the global control flow (loading/reading data to/from board memory, starting the FPGA computation) is demanded to the PIII host computer. In order to test the advantages attainable by using the PROSIDIS dedicated processor as booster for a conventional sequential system, we implemented optimized algorithms to solve the same problem on different general purpose platforms. Results have been compared with those obtained on the test bed architecture.

The architectures used in the tests are reported in Table 1. Table 2 summarizes the results reporting, for each machine configuration, the time spent to execute the searching for a sequence with $m = 24$ amino acids on a portion of proteome with length $n = 2\,096\,000$ amino acids.

Table 1. Architectures used in tests

Processor	Clock frequency	L2 cache size	OS	Compiler
Pentium III	1 GHz	512 KB	Win2000	MS Visual Studio 6.0
Alpha EV67	667 MHz	4 MB	Linux (kernel 2.3.14)	ccc 6.2
R12000	300 MHz	4 MB	Irix 6.5	MIPS C
Power3	200 MHz	4 MB	AIX 4 OS	AIX C 4.4
UltraSparc	450 MHz	4 MB	Solaris 2.7	Sun WorkShop C 5.0

Table 2. Test results

Architecture	Exec time (ms)	FPGA speed up
FPGA + PIII@550 MHz	57	1
PIII@1000 MHz	290	5.1
Alpha EV6.7@667 MHz	387	6.8
SGI R12000	533	9.4
IBM Power 3	1152	20.2
UltraSparc	2892	50.7

REFERENCES

Marongiu,A. and Palazzari,P. (2001) Automatic implementation of affine iterative algorithms: design flow and communication synthesis. *Comp. Phys. Comm.*, **139**.

Yamaguchi,Y., *et al*. (2002) High speed homology search with FPGAs. *Pacific Symposium on Biocomputing*.

Marongiu,A., Palazzari,P., Cinque,L. and Mastronardo,F. (2000) High level software synthesis of affine iterative algorithms onto parallel architectures. *Proc. of HPCN Europe* 8–10, May 2000, Amsterdam.

Marongiu,A. and Palazzari,P. (2000) Automatic mapping of system of affine recurrence equations (SARE) onto distributed memory parallel systems. *IEEE Trans. Soft. Eng.*, **26**, 262.

Loechner,V. and Mongenet,C. (1996) OPERA: a toolbox for loop parallelization, *International Workshop on Software Engineering for Parallel and Distributed Systems*.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, **89**, 10915–10519.