GENOMIC PCR SIMULATION WITH HARDWARE-ACCELERATED APPROXIMATE SEQUENCE MATCHING

Matej Lexa
Masaryk University Brno
Faculty of Informatics
Botanická 68, Brno 60200, Czech R
Email:lexa@fi.muni.cz

Tomáš Martínek, Patrik Beck, Otto Fučík
Brno University of Technology
Faculty of Information Technology
Božetěchova 2, Brno 61266, Czech R
Email:martinto, beck, fucik@fit.vutbr.cz

Giorgio Valle, Ivano Zara
CRIBI Biotechological Centre
University of Padova
U. Bassi 58, Padova 35133, Italy
Email:valle, zarivan@cribi.unipd.it

ABSTRACT

We report the latest details on improvements made or planned for the VPCR simulation software currently accessible on the Internet (installed on servers in Padova and Brno). We describe the inner workings of the dynamic amplification simulation model, concentrating mostly on the time- and sensitivity-critical step of sequence matching. Replacement of BLAST by the more sensitive and faster PRIMEX similarity search program resulted in a marked improvement of PCR product predictions. Further speed improvements were acheived using hardware acceleration of approximate sequence matching. We report our theorethical computation time estimates and results of the first tests using the Arabidopis and human genome sequences as PCR templates.

KEYWORDS

PCR, simulation, PRIMEX, approximate matching, similarity search, oligonucleotide melting, hardware, acceleration, FPGA

INTRODUCTION

Amplification of DNA by various flavors of the Polymerase Chain Reaction (PCR) is now a well-established practice in molecular biology. It has found interesting applications outside mainstream biological research (e.g. in medicine, detective work or food testing for genetically modified organisms) for its ability to amplify short regions of even small traces of DNA to quantities that can be inspected and used by other molecular methods. In PCR and RAPD reactions that use genomic templates, oligonucleotide primers anneal to target positions on a genomic sequence. This annealing is influenced most by the sequences involved, temperature and salt concentration. While certain primers may be highly specific for a given region of the studied genomic DNA and the outcome of the corresponding reaction easily predictable, other set-ups will have less certain outcomes. This is also the case of multiplex PCR, RAPD or any reactions using untested primers. Computer programs capable of predicting the outcome of such reactions could save users from running hopeless experiments. Such programs have been around for some time now (Rubin and Levy, 1996). As their precision and speed increases, they could become useful practical tools in PCR and RAPD design, primer evaluation, microarray design or teaching.

PCR SIMULATION PROGRAMS

The simplest program predicting amplification products from genomic templates (usually not recognized as such) is the UNIX *grep* command. Given a primer sequence *p* and a complete genomic sequence stored in a textfile *file*, one can simply issue the command

$$egrep\ p. + rev(comp(p))|comp(p). + rev(p)\ \ file$$

where rev() and comp() stand for the reversed and complementary sequences respectively. However, this simplicity comes with a price. Exact matching is a bad estimator of primer binding, which is why it will predict only a fraction of the possible amplification products. If one assumes a 20bp primer and allows single mismatches for binding to still occur, there will be 61 = 3*20+1 functional variants for each primer, i.e. 3721 different combinations for two of them. Of these combinations, only one would be predicted by precise matching. This would (on average and assuming a random sequence) result in an underestimation of the number of possible PCR products by about 99.97%. In other words, majority of products would go unnoticed. This estimate is, of course, grossly invalid for well-designed primers that have no one-mismatch neigbours in the genome, for which the grep method actually works. A program by (Lexa et al., 2001) is one of the early attempts to handle less-behaved primers. The use of BLAST for approximate matching improved the primer annealing predictions, which in turn resulted in better predictions of the resulting amplification products. Shortly after VPCR 1.0 had been announced a number of other groups that pursued similar ideas published their software tools (Bikandi et al., 2004)

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

(Boutros and Okey, 2004). These tools were often superior to the BLAST-based version of VPCR. Our goal was to eliminate several errors and inefficiencies in VPCR 1.0 and design a PCR simulation tool that would offer physically and chemically sound predictions at interactive speeds.

The development of VPCR 2.0 has a goal of acheiving very fast simulations for the largest genomes. While others were proposing to use grid computing to rapidly simulate PCR amplification (Anonymous, 2003), we implemented an efficient algorithm that supports PCR simulations on a single computer, even with the largest genomes (Lexa and Valle, 2003). Another important improvement was the introduction of a PCR simulation step. This allowed us to predict amplification yield for a given product. Finally, we are in the process of building hardware-accelerated string matching methods for VPCR. The latest details of this undertaking are reported in this paper. To name features that are desirable, but not available at the moment, the program still lacks the ability to handle nested and overlapping primers, or to consider the effects of DNA secondary structures on amplification rates.

In future, we envisage fully implementing procedures to handle these more complicated situations, as well as fully utilize the benefits of hardware acceleration, so that a single PCR reaction can be predicted at current or better quality in a fraction of a second on a single properly equipped personal computer.

PCR SIMULATION WITH VPCR 2.0

Simulation of PCR reactions in VPCR 2.0 is done in three main stages. The user is prompted for primer sequences and asked to choose a genome to use as a template in the simulation. In the first stage (sequence matching), we identify positions in the genome that are likely to be annealing sites for the given primers. In the second stage these positions are evaluated as to the melting temperature of the potential template-primer dimers. In the third stage (simulation), the melting temperature data is used to simulate individual PCR cycles, considering the dynamic equillibria between free and bound primers.

Approximate Sequence Matching Using PRIMEX

The search for candidate matches seems to be crucial for the whole PCR simulation. It must be sensitive enough to extract all the relevant candidate sequences. At the same time it should be relatively fast, because with large genomes it tends to be the time-limiting step in the simulation. Using BLAST in our first algorithms to predict PCR products (VPCR 1.0) (Lexa et al., 2001), we realized several shortfalls of the program for these purposes and searching for matches always used several orders more computation time than the simulation of the PCR reaction itself. Evaluating BLAST and other available software that could possibly carry out the job resulted in a decision to write a new program that would be better suited for our purpose. We called the new program PRIMEX (PRImer Match EXtractor)(Lexa and Valle, 2003).

In PRIMEX, fast approximate searches in large sequences are acheived by a hierarchical approach to searching, including a filtering step at the intermediate level. The program splits the query sequence into non overlapping words. For instance, if the word size is 10, then a 22-base query sequence will be split into two non-overlapping words. The search for nearly perfect word matches (typically with no more than one mismatch per word) of a pre-defined length can be extremely fast if appropriate data structures are used. We use a simple and straightforward array-based lookup table. In addition, we constructed the program to run in server mode to prevent loading the whole genome and creating the lookup table repeatedly before every query. For faster start-ups, the lookup table may be saved to disk or loaded from disk into the memory as needed.

The mechanism of translating lookup table entries into candidate matches and further into satisfactory query matches represents the central algorithm lying at the very heart of PRIMEX. After the first filtering step, we obtain a set of candidate sequences, each of which matches at least one word extracted from the query. If the number of mismatches allowed per word is *m1*, it can be shown that this approach will identify all sequences that differ by less than $n*(m1+1)$ nucleotides from the query, where n is the number of non-overlapping words extracted from the query.

This equation shows it is important to choose the right word-size and search depth (*m1*) for each oligonucleotide query to acheive a desired specificity. The searches become much slower with every increase of *m1*. We empirically determined word lengths of 8-10 as good values for evaluating PCR primers with PRIMEX. Using these values we obtained a good compromise between sensitivity and speed. In combination with $m1=0$ or $m1=1$ we can fully identify approximate oligonucleotide matches with 3-6 mismatches. Higher sensitivity is probably not needed, since melting temperature of oligonucleotides with too many mismatches would fall outside the range of temperatures relevant to a typical PCR reaction. Shorter word sizes than 8 could provide better sensitivity for queries shorter than 20, but in our experience they also make the program much slower, because of a high number of candidates that have to be evaluated further downstream. This may not be the case for the hardware-accelerated matching reported below, allowing for higher sensitivity and shorter primers when using FPGA support.

Melting Temperature Estimates

The previous stage selects a set of candidate annealing sites satisfying a maximal number of mismatches or a minimal Needleman-Wunsch score. However, the parameter relevant for PCR simulation is the binding energy which can be expressed as the melting temperature, not the number of mismatches or the score. The score-based filtering is important though, because the energy calculations are much slower and can only be done in reasonable time on a limited subset of sequences. We calculate melting temperature using the nearest-neighbour method of (SantaLucia, 1998).

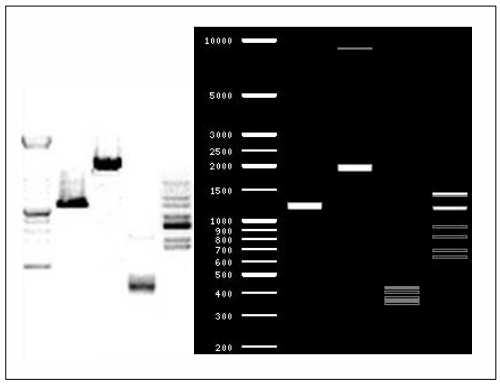

Fig. 1. Comparison of VPCR 2.0 simulation results with PCR data from (Lexa et al., 2001). The four reactions shown from let to right are primer sets ARR5, ARR7, ARR12 and ARR13 against the Arabidopsis thaliana genome.

Dynamic Simulation of PCR Cycles

Once melting temperatures are known, we can simulate the PCR reaction in time, estimating the binding of primers to template, following the amplification yield of each amplifiable primer pair through the complete set of PCR cycles. The present simulation model is relatively simple. It assumes 50% binding of primers at melting temperature and a sigmoidal response to temperature changes. Salt concentrations are fixed at 50mM for monovalent ions and 1.5mM for divalent ions, future releases of the program will allow the user to manipulate these values as well. The simulation also tracks the concentration of free nucleotides and the polymerase. If nucleotides are exhausted by too much priming, calculations stop in that cycle; if amplification is limited by the available polymerase, this is reported in the output.

Example PCR simulation results compared to experimental data can be seen on Fig 1.

NEEDLEMAN-WUNSCH HARDWARE ACCELERATION

The abovementioned approach to PCR simulation produces promising results, however further increase in speed is desireable for batch analysis of multiple reactions. Faster calculations may also allow shorter word-lengths needed to evaluate short primers. At present, most of the simulation computation time is spent comparing queries with candidate sequences using the Needleman-Wunsch algorithm inside PRIMEX. Consequently, we chose this step as an appropriate point for further acceleration.

The Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) represents a specific application of dynamic programming (DP) to solving the approximate string-matching (AM) task and was first used in molecular biology to align DNA sequences. Briefly, the algorithm splits the task of AM into elementary steps, which search for a locally-optimal extension of partially aligned subsequences. The computations are typically carried out in a DP matrix, which, upon being filled at completion of all the elementary steps, defines the best alignment(s) and their scores.

An example of comparing sequences `"GTA"` and `"GCT"` using Needleman-Wunsch algorithm can be seen in Figures 2a and 2b. The value of each item *d* in the DP matrix is computed from the three nearest neighbours *a*, *b* and *c* using the following formula:

$$d = min \begin{cases} a & if \quad U_i = V_j \\ a + sub & if \quad U_i \neq V_j \\ b + ins & \\ c + del & \end{cases}$$

Variables *sub*, *ins* and *del* represent substitution, insertion and deletion penalties that can be adjusted for different comparison requirements. For example, if the presence of redundant characters is less acceptable than difference in characters, the insertion penalty can be set higher than the penalty for substitution.

Typical software implementation needs n^2 computation steps and thus the algorithm time complexity is quadratic. However, several calculation steps of Needleman-Wunsch algorithm can be processed independently and in parallel fashion. In terms of the AM matrix, this means that the anti-diagonal values can be calculated concurrently, see Figure 2c.

This property is effectively used for algorithm acceleration using dedicated hardware (Hoang and Lopresti, 1992) (Lavenier, 1998) (Yu et al., 2003) (Court and Herbordt, 2004). Typical hardware architecture is based on a systolic array of Processing Elements (PE) (see Figure 2c). Every PE in the array calculates one column of AM matrix and sends the computed values to the next PE each clock cycle. Anti-diagonal values are calculated in parallel and thus the overall time complexity is reduced from quadratic to linear.

SYSTEM ARCHITECTURE

The system architecture for acceleration of PRIMEX is composed of two basic parts: (1) PRIMEX application

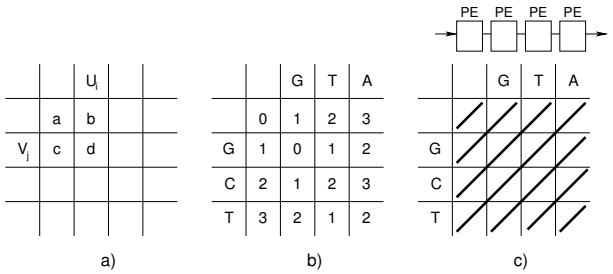

Fig. 2. a) Evaluation of one table cell. The data from *a*, *b* and *c* are used to compute content of the cell *d*. b) The entire table is evaluated for two strings "GTA" and "GTC". c) The data dependence for parallel processing optimization. All cells highlighted by the wide line can be computed at the same time.

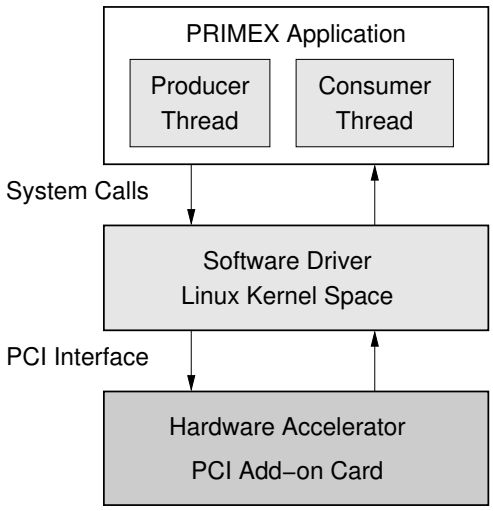

Fig. 3. System Architecture

software and (2) an FPGA-based hardware accelerator connected to the PCI system bus. For effective use of the acceleration hardware, the software application is split into two threads. The first thread follows the basic algorithm of PRIMEX to the point where it generates a list of candidates for sequence similarity with the query. It sends the list to the hardware. The second thread waits for the results from the hardware and continues the execution of PRIMEX to the end. The rest of the PRIMEX algorithm has been described already and it does not change in the accelerated version.

Communication between software and the hardware accelerator is mediated by an operating system driver (see 3). The driver processes the requests for string comparisons, prepares the data for the hardware and manages the execution of all necessary operations. The strings to be compared are kept in RAM. For higher efficiency, the data is gradually transferred from RAM to the acceleration card via DMA operations controlled directly by the FPGA. In the same manner, calculation results are automatically moved to the host RAM memory. The driver only prepares a list of requests in the

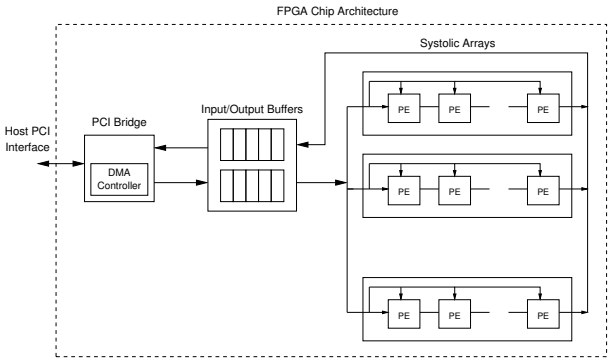

Fig. 4. FPGA Chip Architecture

form of memory pointers and forwards this list to the accelerator.

Hardware Architecture

The block diagram of FPGA architecture is shown on Figure 4. It is composed of a PCI bridge component providing communication via the PCI interface; Input/output buffers for storage of incoming and outgoing data; and multiple instances of systolic arrays, which perform the string comparisons. The process of string comparison operates as follows:

1) The input strings are downloaded using DMA operation from host RAM memory via PCI bus into the FPGA internal input buffers.
2) The appropriate systolic arrays read the data from input buffers and perform the score computations.
3) The resulting scores are forwarded into the output buffer and then via DMA operation transferred back from the FPGA chip into the host RAM memory.
4) After a number of transfers is finished, the FPGA adaptor generates an interrupt to signalize the software driver, that computed scores are available for further processing. Alternatively, the software driver can pool the operation status register placed in the FPGA chip.

All DMA transfers between RAM and the FPGA chip are controlled via the DMA controller, which is part of the PCI bridge.

Individual systolic arrays always contain the correct number of processing elements to be able to process even the longest strings in the request. If we assume that the software is able to sustain a continuous flow of data to the systolic arrays, then the acceleration rate of PRIMEX calculations depends mostly on the number of systolic arrays placed on the FPGA chip. The maximum number of systolic arrays is constrained by two conditions: (1) limited computational resources

on the chip and (2) limited input/output bandwidth of the chip, allowing only a certain number of systolic arrays to be supplied by data without interruption.
The method, which is able to verify both of these conditions and automatically map the systolic array to programmable chips was subject of our previous research (Martinek et al., 2006). From the results of a practical implementation (see Table I) the number of systolic arrays is limited by input/output throughput of the system. By derivation, it is possible to obtain common formula for the maximal number of systolic arrays:

$$N_{SA} \leq min(\frac{B_{IN}}{\frac{(L1+L2)*Ch_{Width}}{L1}.f}, \frac{B_{OUT}}{\frac{S_{Width}}{L1}.f}) \quad (1)$$

where $L1$ and $L2$ are lengths of input strings, Ch_{Width} is width of input characters in bits, S_{Width} is width of score in bits, f is FPGA accelerator clock frequency and B_{IN} and B_{OUT} are input and output throughputs of the communication bus, respectively. Based on the maximal number of systolic arrays N_{SA}, it is possible to realize the input task with Q queries in time:

$$T = \frac{L1}{f}.\lceil\frac{Q}{N_SA}\rceil \quad (2)$$

Alternatively, the performance of hardware accelerator can be expressed in billions of updates per second (BUps), where one update represents computation of one item of the Needleman-Wunsch matrix:

$$P = \frac{L1.L2.Q}{T} \quad (3)$$

RESULTS

The described system was implemented on a Pentium 4 3.2GHz computer with 2MB cache, 2GB RAM supplemented with acceleration card COMBO6X (see Figure 5) developed in the scope of the *Programmable Hardware* project and provided by the CESNET association (Liberouter, 2004). This card was initially developed for the purposes of network application acceleration, but with respect to the amount of computational power and connectivity to the PCI bus, it can be also utilized for purposes of bioinformatic algorithm acceleration. COMBO6X card contains the powerful FPGA gate array with Virtex II Pro (xc2vp50) technology, static synchronous SSRAM memories with capacity 8MB, additional associative CAM memory and four gigabit Ethernet interfaces. The card communicates with the host system via the PCI 64/66MHz interface.
We used the formulas above to calculate the predicted speed-up of the Needleman-Wunsch algorithm used within the PRIMEX program (please, see section III.A for detailed description) on our FPGA hardware. The

Fig. 5. COMBO6X Acceleration Card

TABLE I
EVALUATION OF PRIMEX USING COMBO6X AND COMBO6E ACCELERATION CARD

Card Type	COMBO6X	COMBO6E
Host interface	PCI 64/66	PCI-XP x4
FPGA chip	xcv2p50	xcv2p70
Bus throughput [Gbps]	4	10/10
Number of systolic arrays	8	22
FPGA resource utilization [%]	58.3	91.2
Number of queries	100k	100k
SW comput. time (software)	3.2 s	3.2 s
SW comput. time (hardware)	5.43 ms	1.88 ms
SW comput. speed up	589	1700
PRIMEX comput. time (software)	3.4 s	3.4 s
PRIMEX comput. time (hardware)	255 ms	240 ms)
PRIMEX comput. speed up	13	14

parameters needed for the estimation were set to maximal query length of 40 DNA bases evaluated against 100000 candidate sequences, allowing for 20Achieved results are listed in Table I. With respect to limited input/output throughput of PCI bus, FPGA chip contains 8 systolic arrays with 40 processing elements. The time for comparison of all strings with Needleman-Wunsch algorithm consumes 5.43 ms. In comparison with pure software implementation, the speed-up is approximately 589. Overall PRIMEX computation implemented in software using the same data takes an average of 3.4 s (the exact time and speed-up depends on the query in a nontrivial manner). Using the hardware accelerator, it is possible to reduce the computation time to 0.25 s and speed up the whole application 14 times. The PRIMEX algorithm contains a sorting step which at present is fully implemented in software and prevents us taking full advantage of hardware acceleration. Future work should optimize the two steps to achieve higher speed-ups.
The bottleneck of the system is limited capacity of communication bus. If the input/output throughput increases the number of systolic arrays placed in FPGA chip will

increase as well as the accelerator performance. In the second column of Table I are shown possible results for COMBO6E card with PCI Express x4 interface. Please note, that listed values are estimated because, the card is currently in the development stage and thus the PRIMEX method can not be verified directly.

CONCLUSIONS AND FUTURE WORK

We have described a system for simulating complex PCR reactions with genomic templates on a single computer that can achieve interactive speeds and realistic predictions. Much of the computation speed is due to the use of a unique filtration method provided by the PRIMEX software combined with the possibility to accelerate the bottleneck operation of sequence alignment using FPGA chips. Future work will concentrate on improving the prediction power of the method by including DNA secondary structure evaluation and possibility for correct treatment of overlapping PCR products. The possibility of hardware acceleration could be extended to other time-demanding parts of the simulation, such as the PCR cycle simualtion itself.

REFERENCES

Anonymous (2003). On-demand rapid in-silico pcr. Proposal in the Grid Innovations and Applications Competition.

Bikandi, J., Millan, R. S., Rementeria, A., and Garaizar, J. (2004). In silico analysis of complete bacterial genomes: Pcr, aflp-pcr and endonuclease restriction. *Bioinformatics*, 20(5):798.

Boutros, P. C. and Okey, A. B. (2004). Puns: transcriptomic- and genomic-in silico pcr for enhanced primer design. *Bioinformatics*, 20(15):2399–2400.

Court, T. V. and Herbordt, M. C. (2004). Families of fpga-based algorithms for approximate string matching. In *IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP 2004)*, pages 354–364, Galveston, TX, USA.

Hoang, D. T. and Lopresti, D. P. (1992). FPGA implementation of systolic sequence alignment. In Grünbacher, H. and Hartenstein, R. W., editors, *Field-Programmable Gate Arrays: Architectures and Tools for Rapid Prototyping*, pages 183–191. Springer-Verlag, Berlin.

Lavenier, D. (1998). Speeding up genome computations with a systolic accelerator. *SIAM news*, 31(8).

Lexa, M., Horak, J., and Brzobohaty, B. (2001). Virtual pcr. *Bioinformatics*, 17(2):192–193.

Lexa, M. and Valle, G. (2003). Primex: rapid identification of oligonucleotide matches in whole genomes. *Bioinformatics*, 19(18):2486–2488. Application note.

Liberouter (2004). Liberouter Project WWW Page. http://www.liberouter.org.

Martinek, T., Lexa, M., Korenek, J., and Fucik, O. (2006). A flexible technique for the automatic design of approximate string matching architectures. In *Proc. of 2006 IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*, pages 83–84. IEEE Computer Society.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453.

Rubin, E. and Levy, A. A. (1996). A mathematical model and a computerized simulation of pcr using complex templates. *Nucleic Acid Research*, 24(18):3538–3545.

SantaLucia, J. (1998). A unified view of polymer, dumbbell, and oligonucleotide dna nearest-neighbor thermodynamics. *Proc. Natl. Acad. sci. USA*, 95:1460–1465.

Yu, C. W., Kwong, K. H., Lee, K.-H., and Leong, P. H. W. (2003). A smith-waterman systolic cell. In *Field Programmable Logic and Application (FPL 2003)*, pages 375–384, Lisbon, Portugal.

AUTHOR BIOGRAPHIES

MATEJ LEXA obtained a PhD in Plant Biology from the University of Illinois at Champaign-Urbana. His research combines biology with computer science. He has been with the Masaryk University in Brno since 1999, where he currently teaches bioinformatics. Homepage: `http://www.fi.muni.cz/ lexa/`

OTTO FUČÍK works at the Brno Technical University. He is interested in programmable hardware and hardware/software codesign using FPGA technology. Homepage: `http://www.fit.vutbr.cz/ fucik/`

TOMÁŠ MARTÍNEK is a doctoral student at the Brno Technical University interested in applications of programmable hardware in bioinformatics. Homepage: `http://www.fit.vutbr.cz/ fucik/`

GIORGIO VALLE is a leading figure on the Italian molecular biology and bioinformatics scene. His laboratory has actively participated in several multinational sequencing projects. He teaches bioinformatics at the University of Padova. Homepage: `http://grup.cribi.unipd.it/ valle/`