Online journal ISSN - 1676-5680
GMR Genetics and Molecular Research
EVOLUTION AND TECHNOLOGY

Reconfigurable systems for sequence alignment and for general dynamic programming

Ricardo P. Jacobi[1], Mauricio Ayala-Rincón[2], Luis G.A. Carvalho[1], Carlos H. Llanos[3] and Reiner W. Hartenstein[4]

[1]Departamento de Ciência da Computação, Universidade de Brasília, Brasília, DF, Brasil
[2]Departamento de Matemática, Universidade de Brasília, Brasília, DF, Brasil
[3]Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, Brasil
[4]Fachbereich Informatik, Technische Universität Kaiserslautern, Germany
Corresponing author: R.P. Jacobi
E-mail: rjacobi@cic.unb.br

Genet. Mol. Res. 4 (3): 543-552 (2005)
Received May 20, 2005
Accepted July 8, 2005
Published September 30, 2005

ABSTRACT. Reconfigurable systolic arrays can be adapted to efficiently resolve a wide spectrum of computational problems; parallelism is naturally explored in systolic arrays and reconfigurability allows for redefinition of the interconnections and operations even during run time (dynamically). We present a reconfigurable systolic architecture that can be applied for the efficient treatment of several dynamic programming methods for resolving well-known problems, such as global and local sequence alignment, approximate string matching and longest common subsequence. The dynamicity of the reconfigurability was found to be useful for practical applications in the construction of sequence alignments. A VHDL (VHSIC hardware description language) version of this new architecture was implemented on an APEX FPGA (Field programmable gate array). It would be several magnitudes faster than the software algorithm alternatives.

Key words: Sequence alignment, Dynamic programming, Hardware, Reconfigurable architectures

INTRODUCTION

A lot of effort has been made by the scientific community in recent years in order to better understand the Human Genome. This effort was aided by advances in computer algorithms and hardware, which has allowed the identification of the almost 40,000 human genes. The first results were published in Nature (Lander et al., 2001) and Science (Venter et al., 2001) in February 2001. A huge amount of biological data continue to be generated and stored in databases. For instance, GenBank, one of the main public genome databases has been fed at an exponential rate during the last years.

Due to the huge size of DNA sequences, purely software-based implementations of the Smith-Waterman algorithm (Smith and Waterman, 1981), whose space and running time complexity are O(mn) for sequences of size *m* and *n*, do not compete with such highly sensitive linear approximate solutions as the ones implemented in the well-known systems FASTA and BLAST. On the other hand, using dedicated hardware, matching can be processed in parallel, reducing the order to O(m + n). But most of the dedicated hardware solutions are both expensive and they lack the flexibility to be adapted to different problems. Solutions based on reconfigurable devices, such as Field Programmable Gate Arrays (FPGAs) may provide for those needs. The former solutions can be classified as purely *soft*ware approaches oriented to the exploration of parallel *hard*ware architectures, as in Single Instruction Multiple-Data and MultiMedia eXtensions available in Intel microprocessors (Rognes and Seeberg, 2000). The latter solutions can be classified as dedicated or *configware/morphware* approaches, where the threshold between what is *hard* and what is *soft* is flexible, allowing for sophisticated "algorithmic" solutions embedding reconfiguration and execution instructions (Becker and Hartenstein, 2003).

We present a prototype of a reconfigurable systolic architecture adequate for treating problems that are solvable by general dynamic programming algorithms. The architecture was modeled by a rewriting-logic-based methodology using ELAN (Ayala-Rincón et al., 2004), following the original lines of design also applied for a space/time efficient implementation of the Fast Fourier Transform (Ayala-Rincón et al., 2003). Its hardware design and prototyping is presented, which includes the architectural description, the specification with VHDL and simulation and synthesis using the Altera - Quartus II system (Altera, 2004). While most hardware solutions are limited to sequence comparison, which gives an estimation of the sequence similarity, we targeted sequence alignment, effectively producing the sequence matches.

SYSTOLIC ARRAYS AND RECONFIGURABLE SYSTEMS

The term systolic array was coined by H.T. Kung around 1979 (Kung and Leiserson, 1979). A systolic array is a mesh-connected pipe network of datapath units, using only nearest neighbor interconnect. Datapath functional units usually operate synchronously, processing streams of data that traverse the network (an asynchronous mode of operation is also possible, where sometimes the term wavefront array is also used, instead of systolic array). Systolic arrays provide a large amount of parallelism and are well adapted to a restricted set of computational problems: those that present strictly regular data dependencies. Some typical structures are shown in Figure 1.

Systolic array restrictions may be circumvented by using reconfigurable circuits, since the same system may be reconfigured in order to deal with different tasks.

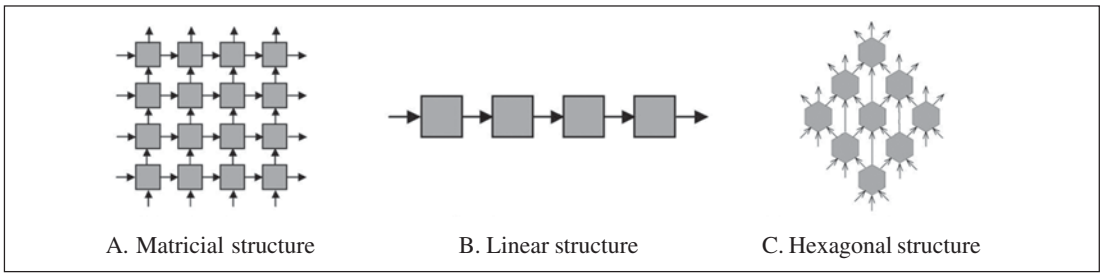

Figure 1. Some systolic structures.

RELATED WORK

Lipton and Lopresti (1985) showed that the parallelism in the Smith-Waterman algorithm can be mapped into a linear bidirectional systolic architecture. Each processing element in that structure computes one of the diagonals of the similarity matrix (Figure 2).

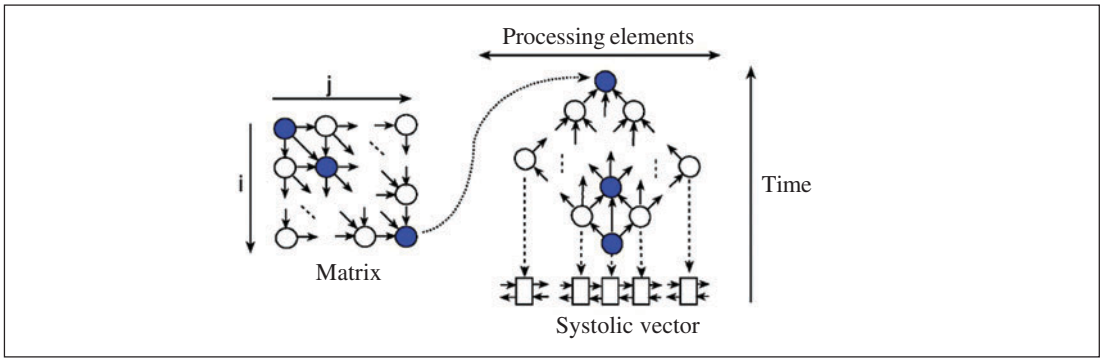

Figure 2. Mapping the matrix to a systolic vector.

Sequences to be compared should be input to the vector from opposite sides and are shifted at each clock cycle to cross the vector. If the two sequences has sizes *m* and *n* then the vector should have *n + m - 1* elements. The result provided by this architecture is a value that indicates the degree of similarity between the sequences.

Hoang (1992) proposed a similar solution based on SPLASH (Gokhale, 1990) architecture, which is a matrix of programmable logic devices developed by the Supercomputer Research Center, using 32 XC30990 FPGAs from Xilinx (Xilinx Inc., 2004). They allowed the user to recover at least one alignment, using Lipton and Lopresti grading scheme. An improvement was later proposed, based on a new version of SPLASH. Lavenier (1996) developed SAMBA (Systolic Accelerator for Molecular Biological Applications) another systolic alternative for computing sequence comparisons. Later, in 2002, a new version of the Hoang approach was presented, based on Virtex FPGAs from Xilinx (Guccione and Keller, 2002). The Hoang solution also inspired HokieGene (Puttegowda et al., 2003), a reconfigurable system implemented with the Osiris card (Institute for Information Sciences, 2002). An architecture that is similar to this work was presented by Yamaguchi et al. (2002). It uses a PCI card with a XCV2000E FPGA

from Xilinx, which contains 43,200 logic cells that can hold 144 processor nodes, but each processor takes four clock cycles to compare two bases. Another solution, also based on Hoang, was proposed by Yu et al. (2003). It uses Xilinx XCV1000E FPGAs, with 27,648 logic cells. Some ASICs were also developed, including BioScan (White et al., 1991), KESTREL (Hirschberg, 1996) and the Proclets of Yang (2002). Commercial products in this area are DeCypher (TimeLogic, 2002) from TimeLogic, based on FPGAs and GeneMatcher2 (GeneMatcher2, 2004), which employs a dedicated ASIC.

There were some limitations in these projects. Almost all of them only computed the comparison and did not produce the alignments between sequences and subsequences. Moreover, the cost function was modified in order to simplify the hardware. This may produce inaccurate results according to biology researchers. In our system, the cost function followed the biological grading system.

THE RECONFIGURABLE SYSTOLIC ARRAY

Conception of the systolic array

The conceived architecture was reconfigured for the treatment of problems such as local and global sequence alignment (LSA and GSA) between two sequences: *s, t*; longest common subsequence (LCS) between two strings: *s, t*, and *k*-approximate string matching (ASM) of a pattern in a string: *s, t*.

All these problems are similarly solved by dynamic programming algorithms that build a table V of size $m + 1 \times n + 1$, where m and n are the length of the two input sequences or strings ($|s| = m$, $|t| = n$). The computation of the i, j-th components of all these tables are based on the values of the previous components in the same row ($i,j-1$), column ($i-1,j$) and diagonal ($i-1,j-1$). Components of these tables are denoted by $V[i, j]$ and symbols of the sequences by $s[i]$, $t[j]$. The computations are given for these problems by the following recurrence relations:

- *LSA:* $V[i,j] = max(V[i,j - 1] - 2, V[i - 1,j] - 2, V[i - 1,j - 1] + p, 0)$, where if $s[i] = t[j]$ then $p = 1$ else $p = -1$ and $V[i,0] = 0,$ for $i = 0..m$ and $V[0,j] = 0,$ for $j = 0..n$.
- *GSA:* $V[i,j] = max(V[i,j - 1] - 2, V[i - 1,j] - 2, V[i - 1,j - 1] + p)$, where if $s[i] = t[j]$ then $p = 1$ else $p = -1$ and $V[i,0] = -2 \times i,$ for $i = 0..m$ and $V[0,j] = -2 \times j,$ for $j = 0..n.$
- *LCS:* $V[i,j] = max(V[i,j - 1], V[i - 1,j], V[i - 1,j - 1] + p)$, where if $s[i] = t[j]$ then $p = 1$ else $p = 0$ and $V[i,0] = 0,$ for $i = 0..m$ and $V[0,j] = 0,$ for $j = 0..n.$
- *ASM:* $V[i,j] = min(V[i,j - 1] + 1, V[i - 1,j] + 1, V[i - 1,j - 1] + p)$, where if $s[i] = t[j]$ then $p = 0$ else $p = 1$ and $V[i,0] = i,$ for $i = 0..m$ and $V[0,j] = j,$ for $j = 0..n.$

Components of these dynamic programming tables are sequentially computed from left to right and top to bottom, but parallelization is possible by computing all components in one (minor) diagonal in a sole step, starting from the first diagonal ($i + j = 2$) and finishing at the last diagonal ($i + j = n + m$). In order to compute values in the diagonal k ($i + j = k$), it is necessary to maintain the values of the previous two diagonals $k - 1$ (since, $i - 1 + j = i + j - 1 = k - 1$) and $k - 2$ (since, $i - 1 + j - 1 = k - 2$) (Figure 3).

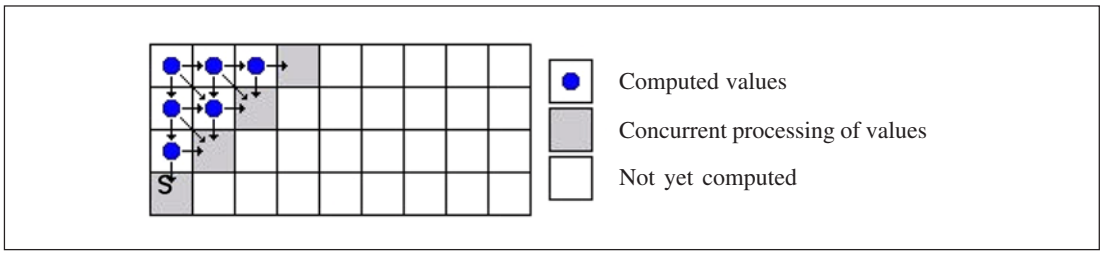

Figure 3. Parallelism in the similarity matrix.

The basic processing element is depicted in Figure 4A. The relative position of the neighbor values is indicated for the computation of value *w*. Note that *y* is the previous outcome of the same cell and is stored as the upper value. *z* was computed by the left neighbor in the previous step and is stored in an internal register for computing *w*. *x* was stored in the left neighbor as the upper value in the previous step, and is transferred at the same time as *z* to the cell computing *w*. Figure 4B gives an idea of the processing steps. The three processing elements store the "G C T" subsequence. At time *t* they are computing the values of the first dashed diagonal and at time *t + 1* they are computing the values of the second dashed diagonal. For simplicity of representation, the left value was stored with the base being processed (as 5 in C5). This illustrates the data flow in the systolic array. Each cell produces, beyond the comparison value, a 3 bit relative pointer that indicates from where the alignment that produced that result came from. This information is used by the host, in which the FPGA is connected, to recover the alignments for the similarity matrix.

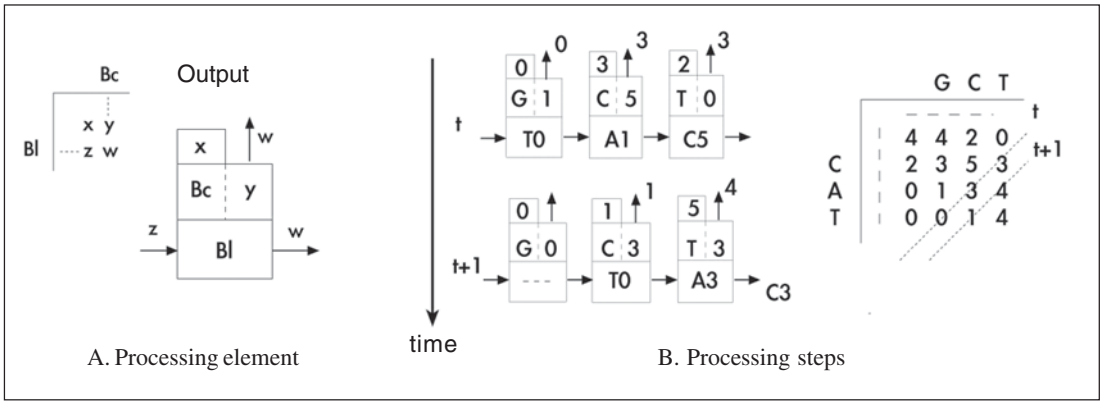

Figure 4. **A.** The processing element and **B.** two steps of the flow of computation.

Application of dynamic reconfiguration

Dynamic reconfiguration is useful for practical applications concerning molecular data. Once the systolic array proposed in the previous section is reconfigured for LSA, detection of the end positions of high scoring alignments between two sequences is possible without writing out all the components of the dynamic matrix. This is necessary for real molecular data because

of the enormous length of the usually treated sequences and the space complexity of the algorithm, which is *O(mn)* for sequences of length *m* and *n.* For example, a practical solution for constructing the alignments of interest between two sequences *s* and *t,* consists in alternating the execution of the following two phases of reconfiguration and execution:

- Reconfigure the systolic array for executing LSA moving the sequence *t* from left to right, without constructing the dynamic matrix, and maintain only the current scores in each diagonal of the table and selecting the good ones.

Once a good score is selected, say finishing at positions *i* and *j* of the sequences *s* and *t,* respectively, the systolic array is reconfigured to execute the GSA operation in reverse order.

IMPLEMENTATION

The hardware implementation has a size limit. Since biological sequences may have thousands of elements, often they will not match the size of the array. In this case, sequence partitioning is done by software. The architecture modeled in ELAN was refined to a structural VHDL description and synthesized and simulated in an Altera Quartus II design environment.

Design in VHDL

The basic architecture of a systolic node to compute the dynamic algorithms includes registers to hold the neighbor values and a reference sequence character, adders to compute the cost, and comparators to check for equality. It runs synchronously, performing one comparison for each clock cycle. The node 'netlist' is presented in Figure 5. The accumulated sequence matching cost grows from left to right, such that the number of bits needed to store the cost can be different for each node. The VHDL description of the nodes keep these values generic, such that they are defined when instantiating the cells.

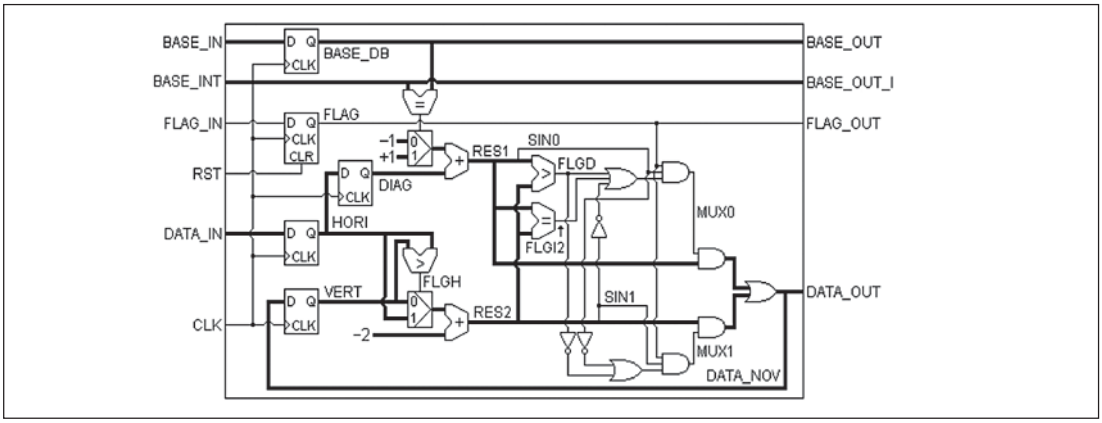

Figure 5. Systolic node structure.

The circuit for one node is simple. It compares the running base obtained from a data base with the base stored in the cell and thereby produces the diagonal cost, which is the

accumulated cost coming from column *j-1* and line *i-1*, added to +1 or -1, as described in the previous section.

Simulation and results

To verify the systolic array, implementation simulations were performed in the Quartus II environment. Figure 6 presents the similarity matrix obtained for the comparison of two pairs of sequences, CATAG and ATAGC and CATAG and CATGA, using the LSA algorithm. This algorithm looks for the best matches between subsequences of the strings. The main difference compared to global matching is that it does not accumulate negative values, allowing local matches along the sequences. The arrows in the figure indicate several alternative matches and the encircled elements are those that need to be stored in a sparse matrix to recover the sequences, which is done by software.

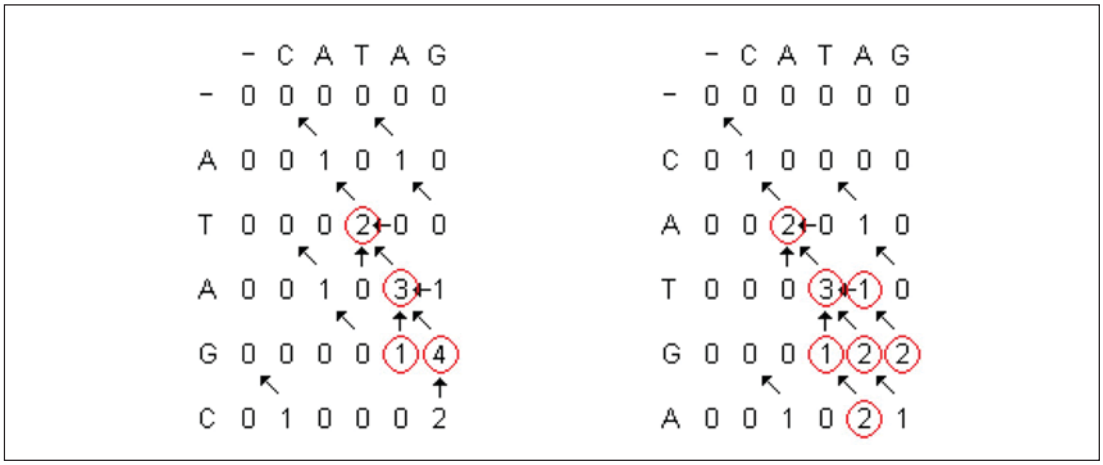

Figure 6. Local sequence alignment for sequences CATAG, ATAGC and CATGA.

The simulation of the comparison between CATAC and ATAGC sequences is shown in Figure 7. The symbols are coded as follows: A = 00, T = 01, C = 10, and G = 11. Signal VAL shows the best score obtained in the comparison, which was four in this example. Signals MEM2, MEM3, MEM4, and MEM5 are the data outputs of the systolic array. CONTA is an auxiliary signal provided to help build the similarity matrix from the data produced by the array. Several arrays were generated and synthesized in order to produce a profile of synthesis results. The results are summarized in Figure 8. The increasing curve (y-axis to the left) shows the number of logic elements required to implement arrays of varying sizes, indicated on the x-axis. For a systolic array of 50 nodes around 4,500 logic elements were required on an APEX device. Since the circuit size grows almost linearly, we can estimate the size of the vector for devices based on the same logic element. For instance, an APEX EP20K400, with 16,640 logic elements could hold 180 nodes. Late generation devices, like Stratix II or Virtex II, could hold thousands of nodes. The decreasing curve (y-axis to the right) corresponds to the frequency attained by the array. Initially the frequency decays strongly with the size of the vector and then stabilizes for vectors with 30 or more cells, at around 56 MHz.

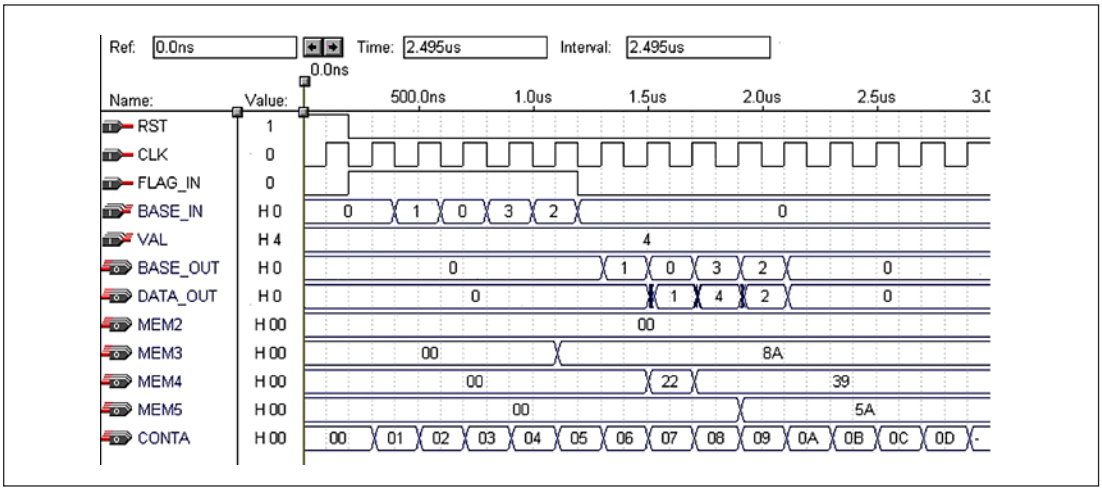

Figure 7. Local sequence alignment simulation of sequences CATAG and ATAGC.

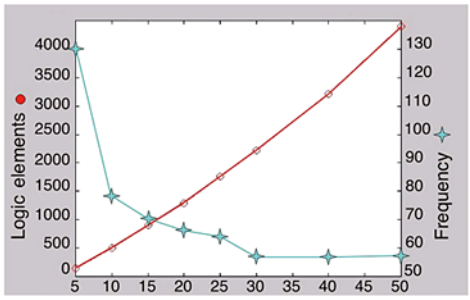

Figure 8. Size (logic elements) and frequency (MHz) versus vector length.

The time needed to compute a sequence comparison is given by the time the running sequence takes to traverse the systolic array. Thus, if the reference sequence has n elements and the data base sequence has m elements, then we need $n + m$ clock cycles to compute the comparison. To obtain a rough estimate of the gain compared to a software solution, suppose that the frequency remains around 50 MHz for larger devices. Considering that we can chain FPGA in order to implement larger sequences, an estimation of the increase in speed provided by the systolic vector compared to the time required by a cluster of workstations (Melo et al., 2003) is given in Table 1. The actual increase in speed should be less than this value because it does not include the communication among FPGAs. Even in this case, the gain in speed is several orders of magnitude. These values do not take into account the time to recover the alignments.

CONCLUSIONS AND FUTURE WORK

This systolic array can speed up string comparison (string pattern matching and sequence alignment) algorithms by software by several orders of magnitude. Previous publications on string comparison in hardware focused on sequence comparison for biological problems

Table 1. Processing time for a cluster of workstations, compared to the systolic array.

Seq. size	1 Proc.	2 Proc.	4 Proc.	8 Proc.	Systolic array
15 K x 15 K	296 s	283.18 s	202.18 s	181.29 s	0.000614 s
50 K x 50 K	3461 s	2884.15 s	1669.53 s	1107.02 s	0.002048 s
80 K x 80 K	7967 s	6094.19 s	3370.40 s	2162.82 s	0.003277 s

Seq. = sequence; Proc. = processor.

using different approaches, but if they were implemented with current technologies, they should provide similar speed-up. Our main contribution is the computation of the sequence alignment instead of sequence comparison. The systolic architecture generates alignments through relative one-bit pointers that allow the host to recover the proper alignments by software in a post-processing step. By using rewriting-logic, we could easily extend the range of problems covered by our systolic architecture through design exploration and simulation. The reconfigurability of the systolic architecture plays a fundamental role, allowing the designer to switch from one algorithm to another. Current work addresses the integration of this architecture in a cluster of reconfigurable workstations, where the FPGAs work in parallel and a distributed operating system controls the process.

ACKNOWLEDGMENTS

Research partially supported by CAPES-DFG, Brazilian-German international cooperation.

REFERENCES

Altera® Corporation (2004). Quartus II User Guide. Available at http://www.altera.com. Accessed February 2004.

Ayala-Rincón, M., Llanos, C.Q., Jacobi, R.P. and **Hartenstein, R.W.** (2003). Modeling a Reconfigurable System for Computing the FFT in Place via Rewriting-Logic. *Proceedings of the SBCCI'03*. IEEE Computer Society Press, São Paulo, SP, Brazil, pp. 205-210.

Ayala-Rincón, M., Jacobi, R.P., Carvalho, L.G., Llanos, C.H. and **Hartenstein, R.W.** (2004). Modeling and Prototyping Dynamically Reconfigurable Systems for Efficient Computation of Dynamic Programming Methods by Rewriting-Logic. *Proceedings of the SBCCI'04*. ACM Press, Porto de Galinhas, PE, Brazil, pp. 248-253.

Becker, J. and **Hartenstein, R.W.** (2003). Configware and morphware going mainstream. *J. Systems Architecture 49*: 127-142.

Gokhale, M. (1990). Splash: A reconfigurable linear logic array. *Proceedings of the International Conference on Parallel Processing*, Urbana-Champaign, IL, USA, pp. 526-532.

Guccione, S.A. and **Keller, E.** (2002). Gene matching using jbits. *Proceedings of the 12th International Workshop on Field-Programmable Logic and Applications, FPL*. Springer-Verlag, Berlin, Germany, pp. 1168-1171.

Hirschberg, J.D., Hughey, R. and **Karplus, K.** (1996). Krestel: A programmable array for sequence analysis. In: *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors*. IEEE Computer Society Press, Chicago, IL, USA, 25: 34 .

Hoang, D.T. (1992). *A Systolic Array for the Sequence Alignment Problem*. Technical Report CS-92-22, Brown University, Providence, RI, USA.

Information Sciences Institute - East (2002). http://slaac.east.isi.edu/. Accessed October 2003.

Kung, H.T. and **Leiserson, C.E.** (1979). Systolic Arrays for VLSI Sparse Matrix. *Proceedings of the Society*

for Industrial and Applied Mathematics, Santa Monica, CA, USA, pp. 256-282.

Lander, E.S., Linton, L.M., Birren, B., Nusbaum, C. et al. (2001). Initial sequencing and analysis of the human genome. *Nature 409*: 860-921.

Lavenier, D. (1996). Dedicated Hardware for Biological Sequence Comparison. *J. Universal Computer Sci. 2*: 77-86.

Lipton, R.J. and **Lopresti, D.** (1985). A systolic array for rapid string comparison. *Chapel Hill Conference on VLSI*, Chapel Hill, NC, USA, pp. 363-376.

Melo, R.C.F., Walter, M.E.T., Melo, A.C.M.A., Batista, R., Nardelli, M., Martins, T. and **Fonseca, T.** (2003). Comparing Two Long Biological Sequences Using a DSM System. *Proceedings of the Euro-Par 2003 - Parallel Processing, LNCS*, *2790*: 517-524.

GeneMatcher2 (2004). The Genematcher2 System Datasheet. [http://www.design-beauty.com/pdfs/paracel/gm2.pdf]. Accessed July, 2002.

Puttegowda, K., Worek, W., Pappas, N., Dandapani, A. and **Athanas, P.** (2003). A run-time reconfigurable system for gene-sequence searching. *Proceedings of the International VLSI Design Conference*, New Delhi, India, pp. 561-566.

Rognes, T. and **Seeberg, E.** (2000). Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics 16*: 699-706. See also Sencel Bioinformatics http://www.sencel.com. Accessed July, 2002.

Smith, T.F. and **Waterman, M.S.** (1981). Identification of common molecular subsequences. *J. Mol. Biol. 147*: 195-197.

TimeLogic Corp. (2002). http://www.timelogic.com. Accessed July, 2002.

Venter, J.C., Adams, M.D., Myers, E.W., Li, P.W. et al. (2001). The sequence of the human genome. *Science 291*: 1304-1351.

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. *Proceedings of the International Conference on Computer Design - ICCD*. IEEE Computer Society Press, Cambridge, MA, USA, pp. 504-509.

Xilinx Inc. (2004). http://www.xilinx.com. Accessed March, 2004.

Yamaguchi, Y., Maruyama, T. and **Konagaya, A.** (2002). High Speed Homology Search Using Run-Time Reconfiguration. *12th International Conference on Field-Programmable Logic and Applications*. Springer-Verlag LNCS, Montpellier, France, *2438*: 281-291.

Yang, B.H.W. (2002). A Parallel Implementation of Smith-Waterman Sequence Comparison Algorithm. Technical Report ID: 4469409. Stanford University Press, Stanford, CA, USA.

Yu, C.W., Kwong, K.H., Lee, K.H. and **Leong, P.H.W.** (2003). A Smith-Waterman Systolic Cell. *13th International Conference on Field-Programmable Logic and Applications*. Springer-Verlag LNCS, Lisbon, Portugal, *2778*: 375-384.