Reconfigurable Computing for Accelerating Protein Folding Simulations*

Nilton B. Armstrong Jr.^{1,2}, Heitor S. Lopes^{1}, and Carlos R. Erig Lima^{1}

^{1} Bioinformatics Laboratory, Federal University of Technology – Paraná (UTFPR), Av. 7 de setembro, 3165 80230-901, Curitiba (PR), Brazil
^{2} Artificial Intelligence Division, Technology Institute of Paraná, Curitiba (PR), Brazil
narmstrong@tecpar.br, hslopes@pesquisador.cnpq.br, erig@utfpr.edu.br

Abstract. This paper presents a methodology for the design of a reconfigurable computing system applied to a complex problem in molecular Biology: the protein folding problem. An efficient hardware-based approach was devised to achieve a significant reduction of the search space of possible foldings. Several simulations were done to evaluate the performance of the system as well as the demand for FPGA's resources. Also a complete desktop-FPGA system was developed. This work is the base for future hardware implementations aimed at finding the optimal solution for protein folding problems using the 2D-HP model.

1 Introduction

Proteins are complex macromolecules that perform vital functions in all living beings. They are composed of a chain of amino acids, and their function is determined by the way they are folded into their specific tri-dimensional structure. This structure is called its native conformation. Understanding how proteins fold is of great significance for Biology and Biochemistry.

The structure of a protein is defined by its amino acid sequences. If we use a complete analytic model of a protein, the exhaustive search of the possible conformational space to find its native conformation is not possible, even for small proteins. To reduce the computational complexity of the analytic model, several simple lattice models have been proposed [4]. Even so, the problem is still very hard and intractable for most real-world instances [1]. The solution is either using heuristic methods that do not guarantee the optimal solution [7] or some scalable strategy capable of intelligently sweep the search space and find the optimal folding (that corresponds to the native conformation).

Reconfigurable computation is a methodology that has been sparsely explored in molecular Biology applications. For instance, [10] presented a new approach to compute multiple sequence alignments in far shorter time using FPGAs. In the same way, [11] describe the use of FPGA-based systems for the analysis of

* This work was partially supported by the Brazilian National Research Council – CNPq, under research grants no. 305720/2004-0 and 506479/2004-8.

DNA chains. A reconfigurable systolic architecture that implements a dynamic programming algorithm and can be used for sequence alignment was presented by [5]. Sequence alignment was also focused by [9], where a pipeline architecture was implemented using reconfigurable hardware. In addition, [8] presents a parallel hardware generator for the design and prototyping of dedicated systems to the analysis of biological sequences. However, there are still few research exploring the use of FPGAs in Bioinformatics-related problems.

On the other hand, recently, we have witnessed a pronounced growth of the hardware and software technologies for embedded systems, with many technological options arising every year. The use of open and reconfigurable structures is becoming more and more attractive, especially due to its robustness and flexibility, facilitating the adaptation to different project requirements.

The possibility of massive parallel processing makes reconfigurable computing (that is, systems based on reconfigurable hardware) an attractive technology to be applied to the protein folding prediction problem addressed here. Hence, the need for powerful processing of biological sequences, on one hand, and the appealing flexibility and performance of reconfigurable logic, on the other hand, are the primary motivations of this work.

The main goal of this project is to develop a methodology for sweeping all possible folding combinations of a protein, using the 2D-HP model [3], in order to find the conformation in which the number of hydrophobic contacts is maximized (as explained in section 2).

2 The 2D-HP Model for Protein Folding

The Hydrophobic-Polar (HP) model is the simplest and most studied discrete model for protein tertiary structure prediction. This model was proposed by Dill [3], who demonstrated that some behavioral properties of real-world proteins could be inferred by using a simple lattice model. The model is based on the concept that the major contribution to the free energy of a native conformation of a protein is due to interactions among hydrophobic (aversion to water) amino acids. Such amino acids tend to form a core in the protein structure while being surrounded by the polar or hydrophilic (affinity to water) amino acids, in such a way that the core is less susceptible to the influence of the solvent [6].

The HP model classifies the 20 standard amino acids in two types: either hydrophobic (H) or hydrophilic (P, for polar). Therefore, a protein is a string of characters defined over a binary alphabet {H,P}. Each amino acid in the chain is called a residue. In this model, the amino acids chain is embedded in a 2-dimensional square lattice. At each point of the lattice, the chain can turn 90^o left or right, or else, continue ahead. For a given conformation to be valid the adjacent residues in the sequence must be also adjacent in the lattice and each lattice point can be occupied by at most one residue. A very simple example is shown in Fig. 4.

If two hydrophobic residues occupy adjacent grid points in the lattice, not considering the diagonals, but are not consecutive in the sequence, it is said that a non-local bond (or H-H contact) occurs. The free energy of a conformation is inversely proportional to the number of H-H contacts. This yields two basic characteristics of real proteins: the protein fold must be compact and the hydrophobic residues are buried inside to form low-energy conformations [6]. The protein folding problem may be considered as the maximization of the hydrophobic non-local bonds, since this is the same as the minimization of the free energy of a conformation in this model.

Although simple, the folding process with the 2D-HP model has behavioral similarities with the real process of folding [3]. Notwithstanding, from the computational point of view, the problem of finding the native structure using the 2D-HP model was proved to be NP-complete [1,2]. Thus, many heuristic algorithms have been proposed to solve this problem [7].

3 Methodology

The structure of the developed system is composed of several hardware and software blocks, described in the next subsections. The system was designed to be capable of analyzing the folding of proteins using a FPGA as a sort of co-processor of a desktop computer. This approach takes the advantage of the FPGA's flexibility and processing power and is integrated with the desktop computer by a user-friendly interface. Also, intelligent strategies are proposed, leading to a dramatic reduction of the search space.

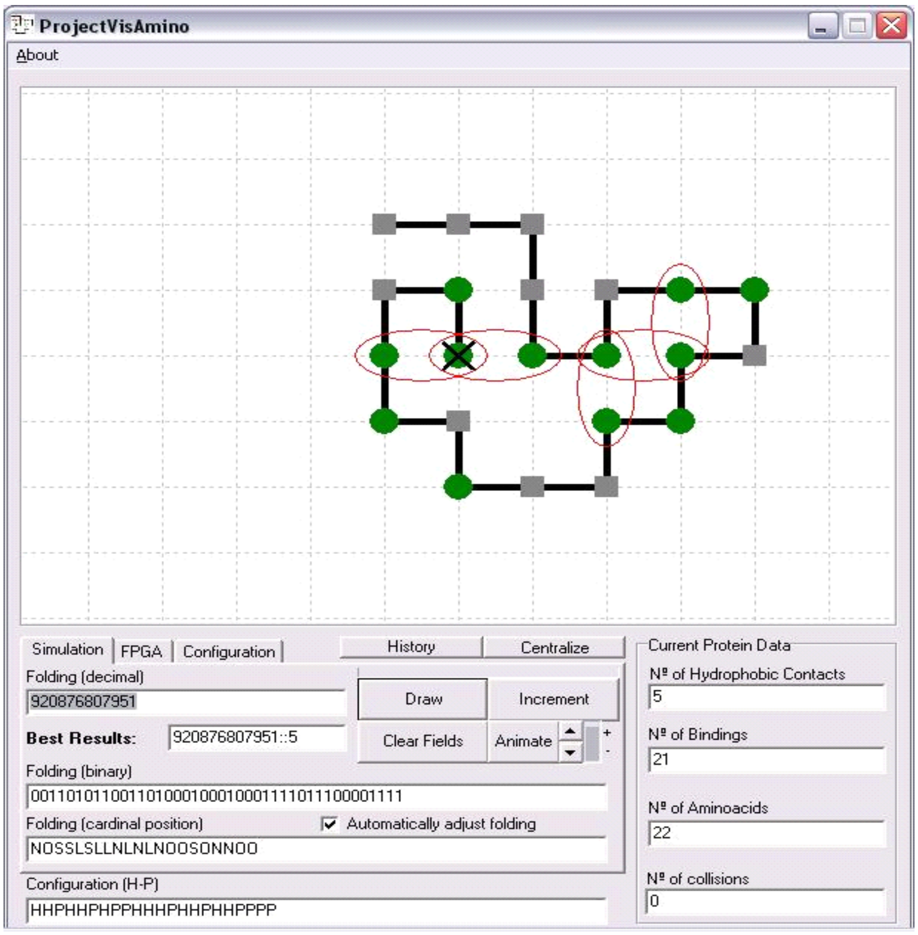
3.1 User Interface

The software developed to run in the desktop computer implements a user-friendly visual interface that enables anyone to easily understand the 2D-HP model. This software was developed in C++ language and it is composed by three basic modules.

The first module is aimed at enabling the user to have a fast visual reference of the protein being analyzed. It automatically displays, on the screen, important information to the researcher: the number of contacts (both graphically and numerically), the collision between amino acids (with graphical marks), and three ways of representing the folding itself: a decimal notation (that is easy to handle), an absolute positional notation (that is easy to understand), and the binary notation (that is the way the FPGA sees the folding). Fig. 1 presents a screen shot of the graphical user interface.

The second module of the software is aimed at providing an intuitive way of exploring the conformational search space. The exhaustive exploration of search space is strongly reduced by using intelligent strategies.

The third module is the communication interface with the FPGA device. As a matter of fact this module does not really communicate directly with the FPGA. It was designed to do the required handshake with a Motorola MCU (Microcontroller Unit) MC68HC908JB8. This MCU is especially suitable for USB (Universal Serial Bus) applications. Considering the small amount of data to be transferred between the desktop computer and the dedicated hardware in our specific



Folding (decimal): 920876807951

Best Results: 920876807951::5

Folding (binary): 0011010110011010001000100011110111000011 11

Folding (cardinal position): NOSSLSLLNLNLNOOSONNOO

[✓] Automatically adjust folding

Configuration (H-P): HHPHHPHPPHHHPHHPHHPPPP

Current Protein Data

Nº of Hydrophobic Contacts: 5

Nº of Bindings: 21

Nº of Aminoacids: 22

Nº of collisions: 0

Fig. 1. Screen shot of the graphical user interface

application, the implemented USB interface is adequate. The implementation of an USB interface external to the FPGA saves internal resources of the device and gives more flexibility to the project. Based on the MCU's architecture a small kit was built to support its operation and allow it to communicate, simultaneously, with the desktop computer and the FPGA. Therefore, a communication protocol has was developed to allow the desktop computer to use the MCU just as a data gateway to reach FPGA. The FPGA works as a slave of the MCU and so it does to the desktop computer. Therefore, the computer communicates to the MCU through the USB and the MCU uses its multi-purpose pins to communicate with the FPGA. When an analysis is to be done, the desktop computer sends the hydrophobicity data related to the protein to the MCU through the USB interface. Then, the MCU serializes this data and shift it into the FPGA input shift register. When the analysis is complete, the MCU acknowledges this completion

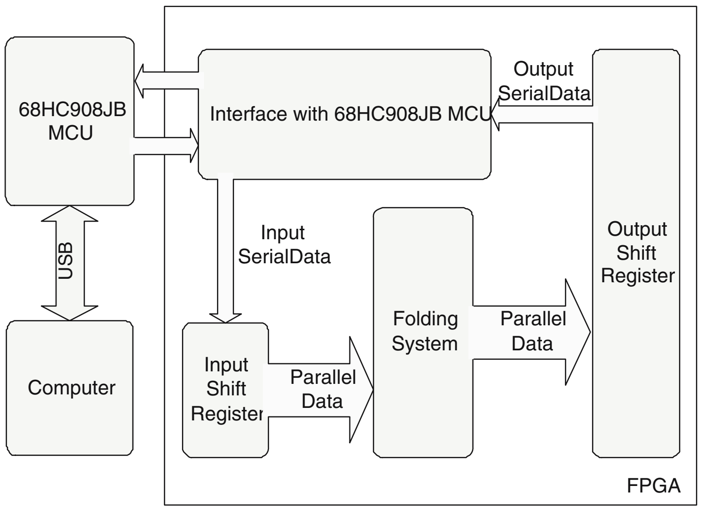


Fig. 2. FPGA solution block diagram

and sends a message to the desktop computer. Next, the computer requests the MCU to retrieve the results and it assembles the data clocked out from the FPGA's output shift register. This FPGA's internal structure is shown in Fig. 2, which we called of FPGA Solution. This architecture based on shift registers to enable a serial communication between the FPGA and the MCU. This serial protocol was used in order to avoid the problems related to the variable width of a parallel bus. With this serial approach, the only thing that changes with the change of the number of amino acids is the number of clock pulses that have to be issued to send the input data and receive back the results.

3.2 Topology of the Folding System

Fig. 3 shows a functional block diagram of a hardware-based system for finding the optimum conformation (folding) of a protein. This system uses the primary structure of a protein and is based on the 2D-HP model. Basically, a counter will swap all possible conformations, according to a given encoding (section 3.3). Conformations have to be converted to Cartesian coordinates (section 3.5) and then checked for validity (sections 3.4 and 3.6). This validity checker reduces the search space analyzed by the next block. After, the number of H-H contacts is counted for the valid conformations found. The conformation with the largest number of contacts is kept and this is one of the solutions for the problem.

3.3 Representation

In order to efficiently sweep the conformational space, the central issues to be addressed is how to represent a protein chain folded in the 2D-HP model using reconfigurable logic, and how to browse the search space.

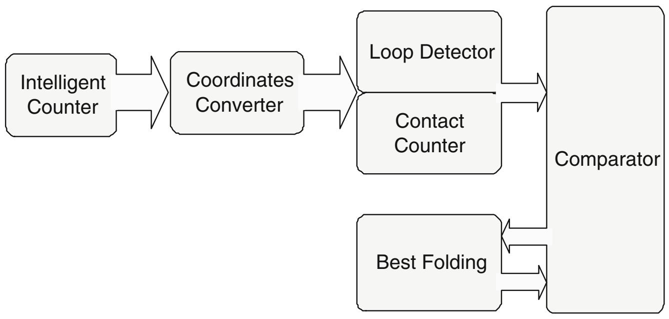


Fig. 3. Functional blocks of the proposed folding system

To solve the representation problem, an absolute positional convention and hydrophobicity information was defined. Basically, only the relative positional information was stored, saving the system from storing the set of Cartesian coordinates of the amino acids in the lattice. This convention is simple and comprises the four possible relative folding directions: North (N), South (S), East (E) and West (W), encoded with two bits, respectively, 00, 01, 10 and 11, and stands for the bindings between the amino acids. Therefore, a complete fold of N amino acids has $(N-1)$ bindings and is represented by a $2(N-1)$ long binary number. It is important to note that this representation, by itself, does not avoid any collisions among the amino acids, it needs a validity check, as will be explained later in section 3.6.

Another relevant information is the hydrophobicity data (HD), that is, a single binary number representing which amino acids are Hydrophobic (bit 1) and which are Polar (bit 0). Therefore, an entire protein can be represented by two binary numbers: its positional information and its HD configuration. According to this convention, Fig. 4 shows an example of a hypothetical 6 amino acids-long protein fragment, its representation and how this specific folding would be represented in the lattice. Black dots represent hydrophobic amino acids, and white dots, the polar ones. The square dot indicates the first amino acid of the chain. However, as our model uses an absolute coordinate system, it is necessary to define an initial point as it would result in different folding, yet belonging to the same set of possible foldings for that protein. It is important to notice that both information are read from the left to the right, meaning that the leftmost amino acid is represented by the leftmost letter in the HD info and the leftmost pair of bits in the absolute positional code.

3.4 Intelligent Counter

The straightforward advantage of the binary representation mentioned before, from the folding perspective, is that it enables the creation of a single step binary counter to generate every possible folding (described by a binary number) for a given amino acids chain. In other words, each count would represent a different folding. However, there is a serious drawback. For a N amino acids-long protein

HD info: H H P P H P
HD info code: 1 1 0 0 1 0
Absolute position: E N E S E
Absolute position code: 1000100110
Decimal notation: 550
Cartesian coordinates:
(0,0);(1,0);(1,1);(2,1);(2,0);(3,0)

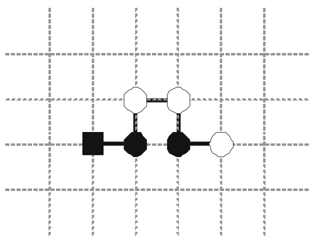


Fig. 4. Example of a folded protein fragment and its 2D-HP representation

it is necessary a number of bits that is almost twice the number of bindings $(2(N-1))$, according to the proposed representation, it would result in $2^{2(N-1)}$ possible foldings. For instance, to analyze a 50 amino acids protein there would be 2^{98} (or $\approx 3.16913 \times 10^{29}$) possible combinations. Such a combinational explosion could render the counter unlikely to sweep through all these combinations in an useful time, even considering a typical maximum clock of 500 MHz of modern FPGA devices.

If it was possible to analyze a valid folding in a single clock time, with a 500MHz clock, it would result in a analysis time of 6.34×10^{20} seconds or 2.01×10^{13} years for 50 amino acids proteins. Besides, such a high clock speed is very difficult to achieve in physical implementations, thus increasing even more the processing time.

However, checking closely the physical behavior of the folding representation, it can be noticed that the folding must follow a self-avoiding path in the lattice. That is, if the previous fold was to the North direction, the next fold cannot be to the South. The same applies to the West-East directions. According to the HP model, these foldings are invalid. In a valid protein conformation a point in the lattice can have at most a single amino acid. Thus, there is no reason to consider any folding that violate this rule, leading to the need of preventing the system of analyzing them, as they are previously known to be invalid. These violations were named of Rule2 violations, for being related to consecutive and adjacent invalid foldings. This counter approach attempts to foresee an invalid folding, according to the Rule2, and skips all binary numbers that could contain that invalid folding at that position.

Consequently, we created an intelligent counter that generates only Rule2 compliant foldings. It can be proved mathematically (not shown here) that using this type of counter a significant reduction of the combinations in the search space is obtained.

Notice that Rule2 does not prevent violations caused by the overlapping of distant amino acids in the chain, as a consequence of a loop in the folding. Although these loop violations are desirable to be eliminated from the analysis, they were not removed from the counting as they are very difficult to be previewed, as will be explained later. As a matter of fact, we already developed a single-clock parallel method to solve this issue. This new approach not only detects any collision in the folding but also counts the number of H-H contacts. However, it is being tested but already represents a significant enhancement to this algorithm.

Another important feature addressed in this counter are two other search space reductions which, even not being as huge as the Rule2 elimination, they also contribute to enhance the processing time. The first one is related to the elimination of repetitive foldings. As shown in Fig. 5, for a protein fragment with 3 amino acids, if all of the possible foldings of this fragment were drawn, it can be seen that there is a pattern (shown in light gray) that repeats itself, rotated in the plane. Since each of the four occurrences of this pattern contains exactly the same set of foldings, 3/4 of the possible foldings can be discarded saving processing time. This strategy adds no complexity to the design of the system. That is, only the most significant pair of bits, which encodes the binding between the first and the second amino acid, would not be part of the counter having its value fixed in "00" or North (see Fig. 5). In other words, the binary count is simply limited to 1/4 of its whole range. Applying this feature to the Rule2, the search space is divided by 4, reducing yet further the processing time.

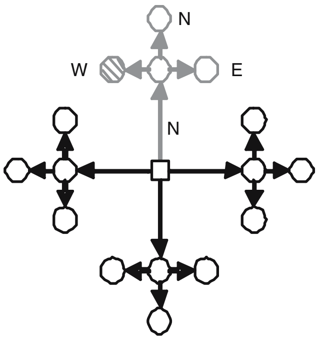


Fig. 5. Sketch of all possible foldings for a hypothetic protein fragment with 3 amino acids. The square dot is the initial amino acid.

The second reduction to the search space can be also seen in Fig. 5 (in light gray). One more duplicate folding can removed, which is related to the mirroring of the foldings from the right to the left of the central vertical axis. This reduction is achieved by making the counter skip every number composed of a folding to the West preceded purely by North folding. In Fig. 5, the eliminated folding would be the NW folding (which is hatched), leaving only two foldings left to be analyzed out of the original search space of 16 foldings. Notice that such elimination of repeated foldings still leaves a set of foldings that represents all the important folding information. Considering the use of Rule2, overall the search space is exponentially reduced. It can be demonstrated that effective search space behaves as: $y = 0.2223.e^{-0.2877.x}$, where x is the size of the amino acid chain and y the effective percent of the search space to be swept by using Rule2.

3.5 Coordinates Converter

The output of the counter, representing a given conformation, has to be converted into Cartesian coordinates (see Fig. 4) so as to effectively embed the amino acid chain in the lattice. Notice that the absolute position encodes only the bindings between the amino acids. However, the Cartesian coordinates represents the amino acids themselves.

Using the first amino acid as reference, the coordinates are generated by a combinational circuit, in real-time, for the whole protein. When a new count, or folding, is generated, based on the absolute reference of the bindings of the protein, the position occupied by each amino acid is promptly computed. This process is done in such a way that the system generates for the current folding being analyzed all the Cartesian Coordinates of its amino acids, all in a parallel circuit. These coordinates need not to be stored as they will be stable for as long as the output of the counter is stable. Therefore, in order for an analysis to be carried out, the counter must be frozen until the Loop Detector and Contact Counter enables it to generate the next folding.

3.6 Loop Detector and Contact Counter

The detection of loops is done at the same time the number of contacts of a valid folding is computed. This block checks for valid conformations, in which there are no overlapped aminoacids. For this purpose, we used the absolute positional information generated by the intelligent counter (section 3.4), based on the Cartesian coordinates explained in section 3.5.

The model intends to validate the current folding as a collision-free chain (i.e. a self-avoiding path). This module is composed basically by a finite state machine (FSM) that has the purpose of accomplishing this validation by building a circuit capable of detecting any pair of identical Cartesian coordinates. The FSM does sequential comparisons, starting at coordinates $(0, 0)$, until the last pair. For each new coordinate pair read, a comparison is carried out with all the pairs yet to be analyzed, to check for collisions of any distant amino acids with the current one. Simultaneously, if the amino acid is hydrophobic, its neighborhood is checked for non adjacent hydrophobic amino acids. As new H-H contacts are found, a contacts counter is incremented. Therefore, this block performs two functions at the same time: detects loops (invalid foldings) and counts H-H contacts (for the valid foldings). If a loop is found, invalidating the folding, the process is aborted and the intelligent counter is requested to generate the next folding.

Currently, this approach stores the absolute position code of the first occurrence of the highest contact count and also its hydrophobic count. Any subsequent occurrence of a folding with the same number of contacts is discarded. To date, there is no known method for predicting how many occurrences of the optimum folding will appear for a given HD configuration.

The main drawback of this approach is that each coordinate pair has to be compared with almost all of the pairs yet to be analyzed. This yields a number of comparisons (nc) computable by the expression: $nc = \Sigma_{i=1}^{na-1} i$, where na is the number of amino acids of the chain. This expression represents the sum of a linear progression. Therefore, the number of comparisons grows quadractically as the number of amino acids increases.

4 Results

In order to evaluate the efficiency of the proposed approach, two sets of experiments were done: simulations in software and hardware implementation. The results of these simulations are grouped in a single table (Table 1).

Several simulations were done using the Quartus II environment from Altera (*http://www.altera.com*). The motivations for these simulations are as follows:

– Check if the system can really identify the first occurrence of the optimum folding, compared to the known value of a benchmark.
– Determine the required processing time for foldings with a given number of amino acids and, further estimate the time required to process larger proteins.
– Estimate the FPGA's resources usage growth with the increment of the size of the amino acids chain.

Table 1. (Left)Comparison of software simulation and hardware. (Right) Resources usage and maximum clock.

na	t_{opt}	t_{sim}	t_{hard}	t_{pc}
6	2.08E-6	1.21E-5	-	-
7	2.93E-6	4.84E-5	-	-
8	1.32E-5	2.27E-4	-	2.00E-2
9	1.52E-4	9.63E-4	-	5.00E-2
10	6.98E-5	3.68E-3	4.00E-3	1.30E-1
11	7.26E-4	1.37E-2	1.50E-2	4.01E-1
12	-	-	6.20E-2	6.00E+1
13	-	-	1.87E-1	3.00E+1
14	-	-	9.90E-1	1.20E+2
15	-	-	2.03E+0	3.00E+2

na	$ALUT's$	Clk_{max}
4	106	274.88
5	173	228.26
6	243	217.78
7	268	235.18
8	442	205.47
9	520	186.39
10	604	184.37
11	687	177.12
30	3241	106.40
50	7611	73.42

For the hardware experiments we used an Altera Stratix II EP2S60F672C5ES device. Each on-board simulation was done considering that the system will supply results to a desktop computer, by reading directly the FPGA's internal memory. Every experiment respected the clock restrictions of the whole system, which is known to decrease as the internal logic is increased. The hardware experiments were carried out using the complete software solution, described earlier.

Table 1(left) shows the processing time needed to find the optimum folding (t_{opt}) and the total processing time necessary to sweep the search space of possible foldings for the simulation, the software (t_{sim}) and the hardware (t_{hard}) implementations. It is also presented the processing time using a desktop computer

(t_{pc}) with Pentium 4 processor at 2.8 GHz. These results are merely illustrative, since the timer resolution of a PC is 1 millisecond, and the algorithm (implemented in C language) is not exactly the same as the one simulated in hardware.

Table 1(right) shows the resources usage of the FPGA device for a growing number of amino acids chains. It is important to note that these values are specific to the FPGA device chosen and may be different for other chips. Column Clk_{max} is the speed the system is able to run in MHz, according to each amino acid chain simulated. The term $ALUT's$ (Adaptive Look-Up Table) is an Altera specific naming used to represent the amount of internal functional logic blocks within the chip. Actually, the chosen FPGA has 48,352 ALUTs (corresponding to 60,440 logic elements).

5 Conclusions

Results of the simulations showed that the proposed algorithm is efficient for finding the optimal folding. The number of H-H contacts found by the system for each simulation did match the expected value of the benchmark. Therefore, the proposed methodology for solving the protein folding problem gives correct answers and is reliable to run such analysis.

The integration between the software and hardware indeed performed well, as expected. Certainly the graphical user interface contributed to the development and the validation of the proposed algorithm. Moreover, the USB interface, using the Motorola MCU, also added flexibility to the project, allowing a transparent communication between the desktop and the FPGA. This USB-based communication protocol allows further development of the systems, especially with other improved protein-related analysis. Future development will focus on a USB 2.0 interface, allowing faster data transfers.

Regarding the growth of resources usage, the implementation behaved within satisfactory limits. Despite this growth is not linear with the increase of the amino acids chain, it does not increase exponentially. The maximum allowed clock cycle decreased slower than expected, and it still can be run in a fair speed even with 30 or 50 amino acids.

The main focus of this work was to devise a methodology for finding the optimal folding of a given protein sequence, in a feasible processing time. This objective was achieved thanks to Rule2 that allowed a dramatic reduction the search space. In some cases, less than 0.001% of the original search space has to be analyzed. These achievements were possible only due to the features of reconfigurable computing, especially the parallelism.

Since one of the main purposes was to find the first occurrence of the optimum folding, sweeping the search space as fast as possible, the system achieved this goal. Recall that the optimum folding is the one with the highest number of H-H contacts. The Table 1(left) shows that the time necessary to run the complete analysis is indeed small, when compared to the time needed by the software implementation. It also shows that the hardware implementation in fact yields a significant improvement on the total analysis time. However, the time required to accomplish the analysis still grows too fast, and can be a serious limitation for sequences with an increased number of amino acids.

A few points should yet be addressed by future versions of this system. Since the search space grows faster, the main priority would be devising strategies for reducing it even more, perhaps by predicting and avoiding any folding that could contain a collision not discarded by Rule2. The idea would be to build a "RuleN" algorithm, in which any collision could be previously detected, not only the adjacent collisions, as does Rule2, yet keeping the capability of counting the H-H contacts in a single step.

On the theoretical ground, further research is necessary to find a method for predicting the maximum number of contacts in a given HP configuration by using only de amino acids sequence. Such upper-bound would allow the system to stop when the first occurrence of the best folding was found. Since such upper-bound value is not known, a full sweep of the search space is required.

Overall, the use of reconfigurable computing for the folding problem using the 2D-HP model is very promising and, for short sequences, it allows obtaining the optimal folding in reasonable processing time.

References

1. Berger, B., Leight, T.: Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete, *J. Comput. Biol.* **5** (1998) 27–40.
2. Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., Yannakakis, M.: On the complexity of protein folding, *J. Comput. Biol.* **5** (1998) 423–466.
3. Dill, K.A., Bromberg, S., Yue, K., Fiebig, K.M., Yee, D.P., Thomas, P.D., Chan, H.S.: Principles of protein folding - a perspective from simple exact models, *Protein Sci.* **4** (1995) 561–602.
4. Dill, K.A.: Theory for the folding and stability of globular proteins, *Biochemistry* **24** (1985) 1501–1509.
5. Jacobi, R.P., Ayala-Rincón, M., Carvalho, L.G., Quintero, C.H.L., Hartenstein, R.W.: Reconfigurable systems for sequence alignment and for general dynamic programming. *Genetics and Molecular Research* **4** (2005) 543–552.
6. Lehninger, A.L., Nelson, D.L., Cox, M.M.: *Principles of Biochemistry*, 2nd ed. Worth Publishers, New York (1998).
7. Lopes, H.S., Scapin, M.P.: An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model. in *Proc. Artificial Evolution, LNCS* **3871** (2005) 238–246.
8. Marongiu, A., Palazzari, P., Rosato, V.: Designing hardware for protein sequence analysis. *Bioinformatics* **19** (2003) 1739–1740.
9. Moritz, G.L., Jory, C., Lopes, H.S., Erig Lima, C.R.: Implementation of a parallel algorithm for pairwise alignment using reconfigurable computing. *Proc. IEEE Int. Conf. on Reconfigurable Computing and FPGAs*, (2006) pp. 99-105.
10. Oliver, T., Schmidt, B., Nathan, D., Clemens, R., Maskell, D.: Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW. *Bioinformatics* **21** (2005) 3431–3432.
11. Yamaguchi, Y., Maruyama, T., Konagaya, A.: High speed homology search with FPGAs. in *Proc. Pacific Symposium on Biocomputing* (2002) 271–282.