# Preliminary Steps Towards Protein Folding Prediction Using Reconfigurable Computing

Nilton B. Armstrong Jr., Heitor S. Lopes, Carlos R. Erig Lima
*Bioinformatics Laboratory (CPGEI) & Electronics Department*
*Federal University of Technology – Paraná*
*Av. 7 de Setembro, 3165 802030-901 Curitiba (PR) – Brazil*
*nlt_br@yahoo.com.br, hslopes@pesquisador.cnpq.br, erig@cefetpr.br*

## Abstract

*This paper presents methodologies for the design of a system based on reconfigurable hardware applied to a complex problem in molecular Biology: the protein folding problem. Different approaches are explored and advantages and drawbacks are analyzed. Efficient strategies are devised to achieve a significant reduction of the search space of possible foldings. Several simulations are done to evaluate the performance of the system as well as the demand for FPGA's resources. This work is the base for future hardware implementations aiming at finding the optimal solution for protein folding problems using the 2D-HP model.*

## 1. Introduction

Proteins perform vital functions in all living beings. They are composed of a chain of amino acid residues, and their function is determined by the way they are folded into its specific tertiary structure. This structure is called its native conformation. Therefore, understanding how proteins fold is of great significance for Biology and Biochemistry.

The structure of a protein is defined by its amino acid sequences. The exhaustive search of the conformational space of a protein is not possible using a complete analytic model, even for small proteins. Therefore, simple lattice models have been proposed to decrease the complexity of the problem [5]. Even so, the problem is still very hard and intractable for most real-world instances [1]. The solution is either using heuristic methods that not guarantee the optimal solution [7] or some scalable strategy capable of intelligently swap the search space and finding the optimal folding.

Reconfigurable computation is a methodology that has been sparsely explored in molecular Biology applications. For instance, [9] presented new approach to compute multiple sequence alignments in far shorter time using FPGAs. In the same way, [10] and [2] describe the use of FPGA-based systems for the analysis of DNA chains. In addition, [8] present a parallel hardware generator for the design and prototyping of dedicated systems to the analysis of biological sequences. However, there are still few research groups exploring the use of FPGAs in bioinformatics 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 attractive, especially due to its robustness and flexibility for easy adaptation to different project requirements. The possibility of massive parallel processing makes reconfigurable computing (that is, systems based on reconfigurable hardware) a suitable technology to be applied to 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 main 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 [4], and find the conformation in which the number of hydrophobic contacts is maximized. This methodology

This work was supported by the Brazilian National Research Council (CNPq) under grants no. 506479/04-8, 501900/04-7 and 305720/04-0.

is focused on a future implementation in hardware using reconfigurable logic.

## 2. The 2D-HP model for protein folding

The Hydrophobic-Polar (HP) model is the most known and studied discrete model for protein tertiary structure prediction and it is due to [4]. It 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 amino acids. Such amino acids tend to form a core in the protein structure while being surrounded by the hydrophilic residues, in such a way that the core is less susceptible to the environmental influence [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 90º left or right, or 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 only one residue.

If two hydrophobic residues occupy adjacent grid points in the lattice but are not consecutive in the sequence, it is said that a non-local bond (or H-H contact) takes place. 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 [4]. Notwithstanding, from the computational point of view, the problem of finding the native structure using the 2D-HP model is proved to be NP-complete [3].

## 3. Methodology

### 3.1 Topology

Fig. 1 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 3.2). Conformations have to be converted to Cartesian coordinates (section 3.5) and then checked for validity (sections 3.3 and 3.6). 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 the solution for the problem.

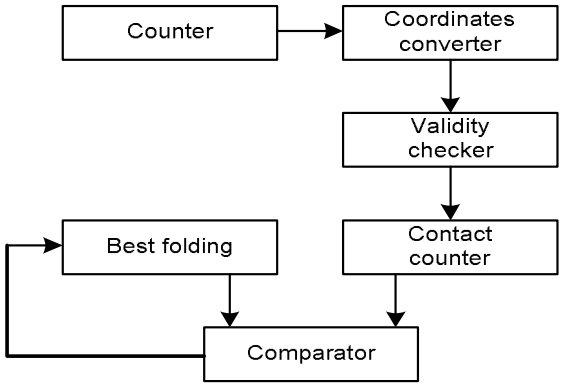


**Fig. 1.** Functional blocks of the proposed system.

### 3.2 Representation

To find the conformation with minimum energy, the whole search space has to be swapped. It will be shown later that, instead of a crude exhaustive search, we devised an intelligent search that avoids most of the invalid conformations. 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.

To solve the representation problem, a relative positional convention and hydrophobicity information was defined. Basically, only the relative positional information is 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 considers the folding backwards, from the last amino acid of the sequence to the first one. That is why the binary number generated does not looks like the actual structure itself, but its reverse image. Actually, this is just a convention to make the least significant pair of bits represent the initial amino acid of the folding. This

convention represents the bindings themselves and not the amino acids.

Another relevant information is the hydrophobicity data (HD), that is, a reverse single binary number representing which amino acids are Hydrophobic (bit 1) and which are Polar (bit 0). Therefore, an entire folded protein is represented by two binary numbers: its positional information and its HD configuration. According to this convention, Fig. 2 shows an example of a 6 amino acids-long protein, its representation and how this specific fold would be represented in the lattice. The filled out points represent hydrophobic amino acids. The "X" mark indicates the first amino acid of the chain.

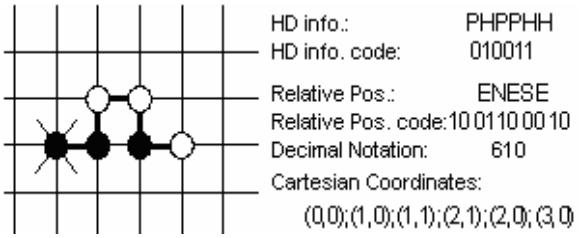


**Fig. 2.** Example of a folded protein and its representation.

### 3.3 Intelligent counter

The straightforward advantage of this binary representation, to the folding perspective, is that it is possible to generate a single step binary counter to generate every possible folding (described by a binary number) for a given amino acids chain. However, there is a serious drawback. For a *N* amino acids-long protein it is necessary a number of bits that is twice the number of bindings (*2*(*N-1*)), according to the proposed representation, thus resulting in $2^{2(N-1)}$ possible foldings. For instance, to fold a 50 amino acids protein there would be $2^{98}$ or $3.16913 \times 10^{29}$ possible combinations. Such a combination explosion could render the counter unlikely to sweep through all these combinations in a useful time, even considering a typical maximum clock of 500 MHz of modern FPGA devices.

However, checking closely the physical behavior of the folding nature, it can be noticed that the folding must follow a self-avoiding path in the lattice. That is, if the previous folding was to the North direction, the next folding cannot be to the South. The same applies to the West-East directions. These foldings are invalid since, according to the HP model, 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 adjacent invalid foldings. Consequently, we created an intelligent counter that generates only valid foldings (according to Rule2). It can be proved (not shown here) that using this type of counter a significant reduction of the search space is obtained, up to 97% or more. However, depending on the length of the protein, analyzing 3% of a huge search space is still too large be done.

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 are very difficult to be foreseen, as will be explained later.

Another feature addressed in this counter is the primary elimination of repetitive hydrophobic count information. As shown in Fig. 3, if all of the possible foldings of a protein are drawn, it can be seen that there is a pattern (shown in light gray) that repeats itself, rotated in the space. Since each occurrence of this pattern contains exactly the same set of foldings, ¾ of the possible foldings, already considering the Rule2 applied, can be discarded saving processing time.

With the proposed intelligent counter a dramatic reduction of the search space is achieved, that is, only 0.75% of the original search space has to be analyzed.

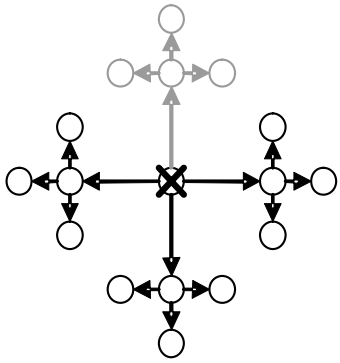

Fig. 3. Out-of-scale sketch of all possible foldings of a 3-amino acids protein. The "X" marks the initial amino acid.

### 3.4 Alternative implementations

The intelligent counter was designed in such a way to generate only the numbers that do not violate Rule2. The main challenge is to build a counter that implements Rule2 in real-time (for instance, without

the addition of any machine state). The counter cannot simply stop updating its output when a Rule2 violation is found. Actually, it has to be intelligent enough to avoid these violations by avoiding the counting, and to do this spending the minimal number of clocks as possible.

For instance, the actual count, for a 3 amino acids counter could be 0-2-3-5-6-7-8-9-10-12-13-15, with no clock spent on conformations 1-4-11-14, as they contain Rule2 violations. Notice that the objective is devise a counter that behaves like this example, with no clocks wasted between the valid counts. However, it turned out to be unfeasible. It is possible to find a mathematical way to detect Rule2 violations, but loops in any point of the chain cannot be predicted with formulas, requiring a further test converting a folding to the coordinates space (see validity checker block in Fig. 1). This kind of counting is required to preserve the real-time property, such that the output of the counter should be stable for no more than 1 clock cycle. This problem revealed itself to be considerably difficult, as there are no formulas or methods for predicting such violations.

The first attempt tried to implement a full combinational logic counter, specific to each number of amino acids. In this circuit, the actual output is analyzed at the moment of its generation, and evaluated to be valid or not. A truth table is created for each possible, valid and invalid, counting, which already incorporates the required analysis. The advantages of this architecture are its speed and the warranty that the output of the circuit would contain only valid, Rule2 violation free countings, at each clock cycle. However, since this circuit must be specific for a given number of amino acids, it tends to become hard to be implemented, when the number of amino acids increases, demanding large amount of FPGA's resources. Due to an exponential growth of resources used with this approach, it is unlike to be feasible for chains larger than 9.

The second approach is to create a general solution that can be applied to any number of amino acids. This attempt is purely combinatorial, and uses sophisticated capabilities of the VHDL language to implement this counter, such as parallel/sequential blocks of code and check the validity of a counting before it is generated. This general solution turned out to be much more efficient, regarding resources (memory and logic elements), but did not have the same speed as the first approach. On the other hand, it can be used with any number of amino acids desired. It implements a series of comparisons between the pairs of bits which define the positional information. This approach looks for Rule2 violations, that is, the four combinations: 00-01, 01-00, 10-11, 11-10. If any of them is detected, the counter is automatically incremented, generating a new folding. This solution is particularly useful when the violation occurs in the most significant pair of bits. For instance, this approach jumps from 00-00-11-11 (which has a Rule2 violation between pairs two and three) directly to 00-10-00-00. This prevents the system from counting 16 useless combinations (every number which started with 00-01-…). This is done within a single clock cycle in most of the cases, except when the new number is also invalid, thus justifying the check to be done in the following number.

### 3.5 Coordinates converter

The output of the counter, representing a given conformation, has to be converted to Cartesian coordinates (see Fig. 2) so as to effectively embed the amino acid chain in the lattice, for further loop detection and contact counting. 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 is generated, the system checks the position relative to the first pair of bits, and goes on checking until the whole protein is completely analyzed. This process is done in such a manner that the current pair of bits is analyzed. Then, according to a comparison with the previous coordinates, it increments or decrements conveniently to generate the next pair of Cartesian coordinates. This process is also done by a combinational circuit and does not depend on clock changes.

### 3.6 Loop detector and contact counter

The next step before actually counting the number of contacts of a valid folding is the loop detection. This block checks for valid conformations, in which there are no overlapped amino acids. For this purpose, two approaches were studied, both using the data generated from the relative positional information from the intelligent counter (section 3.3) and are based on the Cartesian coordinates explained in section 3.5.

The first model intends to predict overlaps in a single clock and the second one uses a finite state machine (FSM). The objective is to devise a circuit for detecting any pair of identical coordinates. If two pairs of XY coordinates are identical, there is an overlap in the chain, therefore, the folding is invalid. To predict an overlap in a single clock cycle, the circuit must be purely combinatorial. To do this, every XY pair is connected to $(N-1)$ comparators, where $N$ is the number of coordinates. However, although it performs well, the number of FPGA's resources used tends to grow exponentially as the length of amino acids chain

increase. This approach is feasible only for very small amino acids chains.

The second approach does sequential comparisons, using a FSM that reads the coordinates (starting in (0,0)) sequentially, 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 overlapping of any distant amino acids with the current one. Also, if the amino acid is hydrophobic, its neighborhood is checked for non adjacent hydrophobic amino acids. As new H-H contacts are found, they are summed up. 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 stops and the next folding is analyzed.

Currently, this approach stores the relative 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.

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. When the number of amino acids increases, the analysis can demand a significant processing time. Another alternative for these comparisons, currently under study, is somehow to abstract a matrix within the FPGA and try to draw the protein into this matrix. This process indeed would take no longer than $N$ clock cycles to check the occurrence of overlapping, with a protein of $N$ amino acids, but the process of writing into a logic cell matrix and erasing it seems to request a lot of FPGA's resources.

## 4. Results

In order to evaluate the efficiency of the proposed approach before implementing it in hardware, several simulations were done using Altera's Quartus II software. There are three motivations for these simulations, as follows:

- Check if the system really can identify the first occurrence of the optimum folding, according to its hydrophobicity, compared to the known value.
- Determine the required processing time for foldings with a given set of amino acids and, then, estimate the time required for processing proteins with higher number of amino acids.
- Estimate the FPGA's resources usage growth with the increment of the size of the amino acids chain.

Simulations were done using an Altera Stratix II EP2S15F484C3 device. Each simulation was done considering that the system will supply results to a desktop computer, by reading directly the FPGA's internal memory. Every simulation respected the clock restrictions of the whole system, which is known to decrease as the internal logic is increased.

Table 1 shows the results obtained regarding the number of contacts for a set of hypothetical amino acids chains. All of these proteins are purely hydrophobic, in order to count not only the real H-H contacts, but also to be sure that the system is not considering contacts on adjacent and consecutive amino acids. The column "Best folding number" in the table is the positional information of the best folding converted to a decimal number, for better understanding. Column "Desktop computer" presents the processing time of the same amino acids chains using a desktop computer with Pentium 4 processor with 800 MHz clock. These results are merely illustrative since the resolution of the PC timer is 1 millisecond and the algorithm (implemented in C language) is not the same as the one simulated in hardware.

**Table 1.** Results of simulations regarding folding number and contact count for purely hydrophobic amino acids chains.

| Number of amino acids | Best folding number | Number of contacts |
|---|---|---|
| 4 | 9 | 1 |
| 5 | 9 | 1 |
| 6 | 37 | 2 |
| 7 | 37 | 2 |
| 8 | 149 | 3 |
| 9 | 2400 | 4 |
| 10 | 597 | 4 |
| 11 | 9568 | 5 |

Table 2 shows the processing time needed to find the optimum folding and the total processing time necessary to sweep the whole search space of possible foldings.

Finally, Table 3 shows the resources usage of the FPGA versus the several amino acids chain sizes. It is important to note that these values are specific to the FPGA device chosen and may be different for other chips. The "Maximum clock" column is the speed the system is able to run, according to each amino acid chain simulated.

The term ALUTs is an Altera specific naming used to be equivalent to Logic Cells, which are

functional logic blocks within the chip. Actually, the chosen FPGA has 12,480 ALUTs.

**Table 2.** Processing time.

| Number of amino acids | Time to find the optimal folding (s) | Total time (s) | Desktop computer (s) |
|---|---|---|---|
| 4 | 1.92E-07 | 3.44E-07 | - |
| 5 | 4.48E-07 | 2.30E-06 | - |
| 6 | 2.08E-06 | 1.21E-05 | - |
| 7 | 2.93E-06 | 4.84E-05 | - |
| 8 | 1.32E-05 | 2.27E-04 | 2.00E-02 |
| 9 | 1.52E-04 | 9.63E-04 | 5.00E-02 |
| 10 | 6.98E-05 | 3.68E-03 | 1.30E-01 |
| 11 | 7.26E-04 | 1.37E-02 | 4.01E-01 |

**Table 3.** Resources usage and maximum clock.

| Number of amino acids | Resources usage (ALUTs) | Maximum clock (MHz) |
|---|---|---|
| 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 |

## 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 with the expected value. Therefore, the proposed methodology for solving the protein folding problem gives correct answers.

Since the main objective is to find the first occurrence of the optimum folding 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. Table 2 shows that the time necessary to find the best folding is indeed small, when compared to the time needed for the software implementation.

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

The main focus in this work is to devise a methodology to reduce the huge search space to a limit in which it can be analyzed in an acceptable time. This feature has been achieved by reducing the search space, thanks to the Rule2 elimination and by discarding 75% of the redundant combinations. With the proposed intelligent counter a dramatic reduction of the search space is achieved, that is, only 0.75% of the original search space has to be analyzed.

We believe that the performance of the system could be greatly increased by using strategies for massive parallelization of the processing elements.

Another point to be explored so as to enhance the system is the substitution of the algorithm based on the comparison of the Cartesian coordinates by another algorithm based on a matrix of bits. In such approach, the entire chain could be drawn, enabling the system to check validity in parallel with the contact count.

## 6. References


[1] B. Berger and T. Leight, "Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete," J. *Comput. Biol*., vol. 5, pp. 27–40, 1998.

[2] M. Canella, F. Miglioli, A. Bogliolo, E. Petraglio and E. Sanchez, "Performing DNA comparison on a bio-inspired tissue of FPGAs," in *Proc. of IEEE International Parallel and Distributed Processing Symposium*, pp. 193–199, 2003.

[3] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis, "On the complexity of protein folding," *J. Comput. Biol.*, vol. 5, pp. 423–465, 1998.

[4] K.A. Dill, S. Bromberg, K. Yue, K.M. Fiebig, D.P. Yee, P.D. Thomas, and H.S. Chan, "Principles of protein folding – a perspective from simple exact models," *Protein Sci.*, vol. 4, pp. 561–602, 1995.

[5] K.A. Kill, "Theory for the folding and stability of globular proteins," *Biochemistry*, vol. 24, pp. 1501–1509, 1985.

[6] A.L. Lehninger, D.L. Nelson, and M.M. Cox, *Principles of Biochemistry*, 2nd ed. New York, USA: Worth Publishers, 1998.

[7] H.S. Lopes and M. Scapin, "An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model," in *Proc. Artificial Evolution, LNCS*, vol. 3871, pp.238-246, 2005.

[8] A. Marongiu, P. Palazzari, V. Rosato, V., "Designing hardware for protein sequence analysis, *Bioinformatics*, vol. 19, pp. 1739–1740, 2003.

[9] T. Oliver, B. Schmidt, D. Nathan, R. Clemens, and D. Maskell, "Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW," *Bioinformatics*, vol. 21, pp. 3431–3432, 2005.

[10] Y. Yamaguchi, T. Maruyama and A. Konagaya, "High speed homology search with FPGAs," in *Proc. Pacific Symposium on Biocomputing*, pp. 271–282, 2002.