Acceleration of a Production Rigid Molecule Docking Code[*†]

Bharat Sukhwani Martin C. Herbordt

Computer Architecture and Automated Design Laboratory
Department of Electrical and Computer Engineering
Boston University; Boston, MA 02215

Abstract: Modeling the interactions of biological molecules, or *docking* is critical to both understanding basic life processes and to designing new drugs. Here we describe the FPGA-based acceleration of a recently developed, complex, production docking code. We find that it is necessary to extend our previous 3D correlation structure in several ways, most significantly to support simultaneous computation of several correlation functions. The result is a hundred-fold speed-up of a section of the code that represents over 92% of the original run-time. An additional 4% is accelerated through a previously described method, yielding a total acceleration of almost $25\times$ for typical protein-ligand combinations.

1 Introduction

A fundamental operations in biochemistry is the interaction of molecules through non-covalent bonding (see Figure 1). Modeling this process of molecular *docking* is critical both to evaluating the effectiveness of pharmaceuticals, and to developing an understanding of life itself. In the former, millions of drug candidates may need to be evaluated for each molecule of medical importance. Computational experiments are therefore likely to be preferred over chemical.

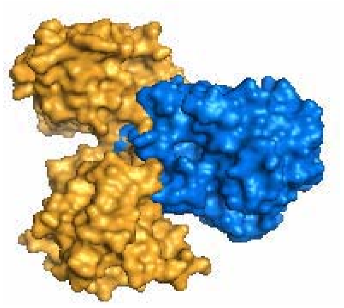

Figure 1: Docked complex of two proteins generated using Pymol [5].

*This work was supported in part by the NIH through awards #R21-RR020209-01 and and #R01-RR023168-01A1, and facilitated by donations from XtremeData, Inc., SGI, and Xilinx Corporation. Web: http://www.bu.edu/caadlab.

†EMail: {herbordt|bharats}@bu.edu

The basic computational task for docking is to find the relative offset and rotation (pose) between a pair of molecules that gives the strongest interaction. There are several issues. First, biomolecules can flex and rotate around chemical bonds. Second, certain molecule pairs interact only when one or both flex in a process known as induced fit. Third, modeling induced fit in some cases requires dynamic modeling, e.g., based on molecular dynamics. Fourth, the best docking pose for many molecule pairs can be determined via simple computations, but that of other pairs may be difficult even with the most sophisticated. As a result, hierarchical methods are often used: (i) an initial phase where candidate poses are determined, and (ii) an evaluation phase where the quality of the highest scoring candidates is rigorously evaluated.

This work describes the acceleration of PIPER, a state-of-the art code that performs the first of these tasks. PIPER minimizes the number of candidates needing detailed scoring with only modest added complexity [7].

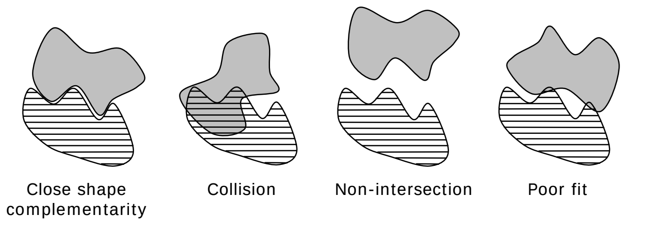

Figure 2: Shape complementarity, collisions, misses, and poor matches (from [10]).

Many docking applications including PIPER assume, at least initially, a rigid structure (see Figure 2). This still allows modeling of various force laws that govern the interaction between molecules, inluding geometric, electrostatic, atomic contact potential, and others. A standard technique maps the molecules' characteristics to three dimensional grids. The most energetically favorable relative position is determined by summing the voxel-voxel interaction values for each modeled force at all positions to generate a score, and then repeating this for all possible translations and rotations.

The resulting computational complexity is large.

With typical grid sizes of $N = 128$ in each dimension and the total number of angles $10,000$, 10^{10} relative positions are evaluated for a single molecule pair. Typically, the outer loop consists of the rotations while the translations are handled with a 3D correlation. Since the latter require $O(N^6)$ operations, this type of exhaustive search was long thought to be computationally infeasible [8]. The introduction of the FFT to docking [6] reduced the complexity of each 3D correlation to $O(N^3 \log N)$ for steric (shape only) models; further work expanded the method to electrostatic [3] and solvation contributions [1].

In previous work [10, 11] we showed that, for FPGA-based coprocessors, the original direct correlation–rather than an FFT–is sometimes the preferred method for computing rigid molecule docking. Two reasons for this are the inherent efficiency with which FPGAs perform convolutions and the modest precision (2-7 bits) of the original voxel data. Note that this precision goes to 106 bits (double imaginary floating point) for the FFT. We also introduced a novel addressing technique for performing rotation that uses only a modest amount of logic, and whose latency can be entirely hidden. And finally, we presented an efficient filtering method that computes on-the-fly the biological importance of the poses and so minimizes the host-accelerator communication.

In this work we extend these methods to facilitate integration into PIPER. In particular, we have added support for (i) pairs of large molecules as necessary for modeling protein-protein interactions; previously we only supported protein interactions with small molecules, (ii) the efficient combining of a potentially large number of force models; previously we had flexibility in the force model, but required it to be simple, and (iii) handling charge reassignment after every rotation; previously we assumed that charge assignment was done only once.

The particular contributions are a modified structure to support these features and experiments that determine optimal configuration among several design parameters. The result is a hundred-fold speed-up of PIPER's correlation computation and a 25-fold speed up of the entire application. The overall significance is in reducing the typical running time from days to hours thereby dramatically increasing the throughput of computational docking experiments.

The rest of this paper is organized as follows. We next give a brief overview of PIPER. There follows the basic design for the 3D correlation. After that we give some implementation details and how they were determined. We conclude with results and discussion.

2 The PIPER Docking Program

2.1 Overview

A primary consideration in docking is preventing the loss of near-native solutions (false negatives); as a result, rigid molecule codes tend to retain a large number (thousands) of docked conformations for further analysis even though only a few hundred will turn out to be true hits. "Improving these methods remains the key to the success of the entire procedure that starts with rigid body docking [7]." PIPER addresses this issue by augmenting commonly used scoring functions (shape, electrostatics) with a desolvation computed from pairwise potentials; the rest of this section is based on the primary reference to that system [7].

Pairwise potentials represent interactions of atoms (or residues) on the interacting molecules. Different pairs of atoms have different values; these are empirically determined (and sometimes called *knowledge based*). For K atom types, there is a $K \times K$ interaction matrix; each column (or row) can be handled with a single correlation resulting in K forward and one reverse FFT. Since K is generally around 20 (and up to 160), and since the FFT dominates the computation, use of pairwise potentials could drastically increase run time. A fundamental innovation in PIPER is the finding that eigenvalue-eigenvector decomposition can substantially reduce this added complexity. In particular, "adequate accuracy can be achieved by restricting consideration to the eigenvectors corresponding to the P largest eigenvalues where $2 \leq P \leq 4$, and thus performing only 2 to 4 forward and one reverse FFT calculations." In practice, however, up to 18 terms are sometimes used.

PIPER's energy-like scoring function is computed for every rotation of the ligand (smaller molecule) with respect to the receptor (larger molecule). It is defined on a grid and is expressed as the sum of P correlation functions for all possible translations α, β, γ of the ligand relative to the receptor

$$E(\alpha, \beta, \gamma) = \sum_p \sum_{i,j,k} R_p(i,j,k) L_p(i+\alpha, j+\beta, k+\gamma) \quad (1)$$

where $R_p(i,j,k)$ and $L_p(i+\alpha, j+\beta, k+\gamma)$ are the components of the correlation function defined on the receptor and the ligand, respectively.

For every rotation, PIPER computes the ligand energy function L_p on the grid and performs repeated FFT correlations to compute the scores for different energy functions. These scores are then combined and an inverse FFT taken to obtain the total energy scores for the various 3-axis translations.

The performance profile is shown in Table 1. A po-

Table 1: PIPER run times for one rotation.

Phase	Run time (seconds)	% total
Rotation	0.00	0%
Charge assignment	0.17	3.4%
FFTs	4.7	92.5%
Filter top scores	0.21	4.1%

tential speed-up of $25\times$ is achievable by accelerating correlation (described here) and filtering (see [11]). Accelerating charge assignment is also possible using the methods developed in [4].

2.2 PIPER Scoring Functions

The scoring function used in PIPER is based on three criteria: shape complementarity, electrostatic energy, and desolvation energy (through pairwise potentials). Each of these is expressed as a 3D correlation sum, and the total energy function is expressed as a weighted sum of these correlation scores:

$$E = E_{shape} + w_2 E_{elec} + w_3 E_{pair} \tag{2}$$

Shape complementarity refers to how well the two proteins fit geometrically (see Figure 2) and here is computed as a weighted sum of attractive and repulsive van der Waals (Pauli exclusion) terms, the latter accounting for atomic overlaps: $E_{shape} = E_{attr} + w_1 E_{rep}$.

Electrostatic interaction between the two proteins is represented in terms of a simplified Generalized Born (GB) equation [1]. The electrostatic energy is obtained as a correlation between the charge on the ligand grid and the potential field on the receptor grid. Unlike in our previous work, charge distribution is recomputed for every rotation.

Desolvation is a measure of change in free energy when a protein-atom/water contact is replaced by a protein-atom/protein-atom contact. In PIPER, it is represented using pairwise interaction potentials, as previously discussed, through P correlation functions.

3 Correlation Structure

Figure 3 shows the systolic 3D correlation array progressively formed starting from a 1D correlation array [11]. This structure is an extension of the 2D correlation array described in [9]. The systolic array performs direct correlation at streaming rate. In our original implementation, voxels for the ligand grid are stored in the compute cells on the FPGA and the voxels of the receptor grid are streamed through it, generating one correlation score per cycle. In this work we have extended the design to support two large molecules (see Section 4.3).

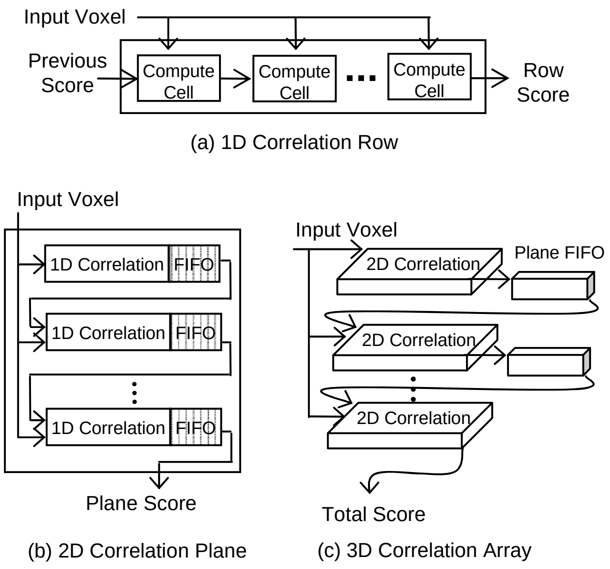

Figure 3: Structures to compute 3D correlations.

As shown in Figure 3a, the 1D correlation structure consists of pipelined compute cells. The input voxel is broadcast to every cell and the partial scores computed by each cell are passed to the next cell, with the last compute cell generating the total row score. The operation of the compute cells can be written as $Score_{out} = Score_{in} + F(Voxel_L, Voxel_R)$ where $Score_{in}$ is the score from the previous compute cell and $F(Voxel_L, Voxel_R)$ is the function between the two voxels. For correlation, $F(Voxel_L, Voxel_R)$ translates to a product between the two voxels. An advantage of direct correlation using this method is that it can handle linear or non-linear functions between the voxels, as opposed to the FFT method which can only handle linear functions.

In order to connect multiple 1D correlation rows to form a 2D correlation plane, the 1D correlation scores need to be delayed. The number of scores generated by each 1D correlation is $N_x + M_x - 1$, where N_x and M_x are the sizes of the two grids along the x-axis. A delay of N_x is inherently provided by the compute cells. To delay the scores by the remaining $M_x - 1$ cycles, 1D line FIFOs are used. Similarly, to connect multiple 2D correlation planes to form a 3D space requires plane FIFOs of size $(M_x + N_x - 1) \times (N_y - 1)$.

On typical high-end FPGAs, these FIFOs can be implemented using block RAMs. Note that the size of the FIFO is proportional to the size of the larger grid M_x. In addition, since the FIFOs are used to delay the correlation score, the width of the FIFOs is dependent on the number of bits the correlation score requires. Although enough block RAMs are present to implement FIFOs for grids of quite large sizes, incorporating multiple correlations can pose a problem. This is discussed in the next section and a modified correlation pipeline proposed.

4 Supporting Multiple Energy Functions

4.1 Overview

There are two obvious ways to extend the structure of Section 3 to compute the multiple correlations required of PIPER: compute them singly or together. Neither is by itself preferred. The first method uses the same control structure as previously, but for each different correlation, the FPGA is reconfigured to the appropriate data types and energy model. The scores must be saved off-chip and combined. That is, the k FFTs are replaced with k correlations, plus the overhead of reconfiguration and combining. The second method involves expanding the structure to perform k different correlations simultaneously. This method requires only a single pass through the large grid, and generates k independent correlation scores per cycle. Recall from Section 2.2, however, that the energy functions are weighted so that for k functions

$$Score_{out} = Score_{in} + \sum_{i=1}^{k} w_i \times correlation\_score_i \quad (3)$$

Thus combining on-the-fly requires multiplications as well as additions, resulting in (perhaps) a substantially more complex compute structure. Combining can be done in three ways: within each compute cell, upon completion, or by integrating the weights into the scoring functions. These options are now examined.

Combining within the compute cells obviously requires that the weighted sums be computed within each one. The problem here is the number of multipliers that this requires: k times the number of cells, or between 256 and 2000 additional multipliers. This is problematic for current FPGAs and would end up drastically reducing the number of compute cells.

Combining on completion means that we must propagate k independent running scores through the line and plane FIFOs of Figure 3; the width of the FIFOs must then be increased by $k\times$. Even with average sized grids, the block RAM requirements to implement the FIFOs are way over the available block RAMs on present day FPGAs, making this approach impractical.

Integrating the weights into the grids requires significantly increasing the precision throughout the entire system. This reduces the number of compute cells and thus the throughput. While a plausible solution, it is still not preferred.

4.2 Augmented Structure

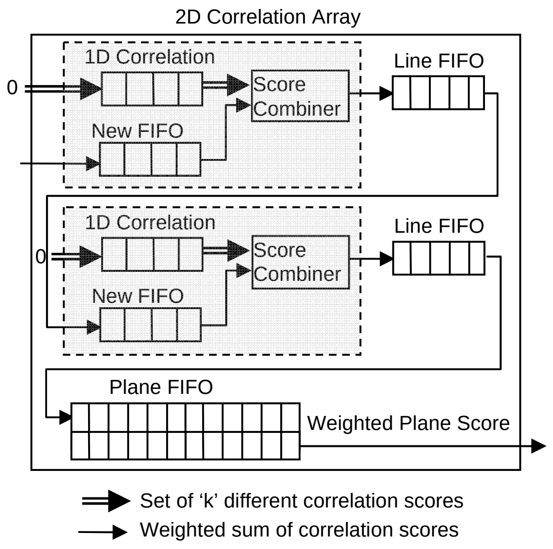

Figure 4: Modified 3D correlation pipeline.

The solution we use is a hybrid: we compute all the energy functions simultaneously and we combine the running scores once per row (see Figure 4). One non-obvious detail is that, now, the scores entering and leaving the FIFO are weighted sums of individual correlation scores, whereas scores computed by the compute nodes are still k individual correlation sums. Thus, compute cells cannot simply add the output of the FIFO to their current score. This requires a slight change in the existing pipeline and the compute cells. The compute rows no longer receive the partial correlation score from the FIFO. The weighted score from the FIFO is sent directly to the score combiner module at the end of next compute row. The score combiner computes the weighted sum of partial correlation scores from the current compute row and adds it to the weighted score from the FIFO of the previous row as shown in Equation 3. Note that a new FIFO is needed in order to align the weighted sum with the output of the 1D correlation array before it enters the score combiner.

The voxel data at every grid point must represent energy values for different energy functions. To implement PIPER energy functions, we modified the voxel data to contain the following five (or more) energy values: attractive van der Waals, repulsive van der Waals, Born electrostatics, Coulomb electrostatics, and P pairwise-potentials. In serial PIPER code, these

Table 2: Number of bits per voxel for computing various energy functions.

Energy Term	Number of bits	
	Receptor	Ligand
Attractive v. d. Waals	8	4
Repulsive v. d. Waals	4	4
Electrostatics	9	9
Pairwise potentials	9	9

energies are represented using single precision floating point numbers. In our FPGA implementation, we use the fixed point numbers shown in Table 2 with no loss of precision. This drastically increases the number of compute cells that can fit on the FPGA and thus the throughput.

The basic compute cell has been extended to compute multiple correlation functions per cycle. As can be seen in Table 2, computing the pair-wise potentials does not require a multiplier, but merely an AND. Since multipliers are the critical resource in this design, P (the number of pairwise potentials) can be quite large without affecting the number of cells that can fit on the FPGA.

4.3 Piecewise Ligand Docking

In order to support larger ligand grids with PIPER energy functions, we implemented a scheme to compute correlation scores in pieces. Note that the receptor size can still be large as previously. We call this piecewise correlation, as it basically involves loading different pieces of the grid into the FPGA correlation cores and storing the partial scores in the score memory. We added a new controller FSM that controls the loading of a grid-piece, generation of correlation scores, and generation of the address of score memory where the current partial score must be added. A new-score accumulator is also added that fetches the current score from the score memory, adds the new partial score to it, and stores it back to the same location in score RAM.

Table 3 compares logic utilization for correlation pipelines with and without support for piecewise correlation. For this example, a simple compute cell is used resulting in an $8 \times 8 \times 8$ on-chip array of cells. The designs in the first two rows both operate on an $8 \times 8 \times 8$ ligand; the difference is that the second has the overhead hardware for swapping. We see that the overhead for supporting piecewise correlation scheme is minimal. Also, the clock rate is virtually unchanged. The last two rows show the support required to operate on 16^3 and 32^3 ligands, respectively, keeping the number of hardware cells constant. Clearly, larger correlations can be supported without much increase in resources required.

Table 3: Resource utilization for piecewise correlation, Xilinx Virtex-II Pro VP70.

Design	Resource Utilization		
	slices	flip-flops	LUTs
No piecewise support $8 \times 8 \times 8$ ligand	31009	22110	33522
Piecewise support $8 \times 8 \times 8$ ligand	31457	22429	34295
Piecewise support $16 \times 16 \times 16$ ligand	31519	22448	34405
Piecewise support $32 \times 32 \times 32$ ligand	31630	22475	34627

5 Results

Our target system is a Xilinx Virtex-II Pro XC2VP70-5 FPGA on Wildstar-II coprocessor board from Annapolis Microsystems. The coprocessor board is plugged into one of the PCI slots of a Dell workstation running the Windows XP operating system. The API from Annapolis Microsystems is used to transfer the data between the host and the FPGA. The unaccelerated PIPER reference code is run on a similar system.

Before the correlation between the receptor and ligand grid is performed, they need to be assigned with charges corresponding to different energy functions. This is currently done on the host using the PIPER code. For the receptor, PIPER assigns the charges only once, since it stays fixed throughout the entire docking process. This grid is downloaded into the FPGA and stored in on-chip block RAMs. For every rotation, the PIPER program rotates the ligand and updates charges on the ligand grid. This grid is then downloaded into the correlation cells on the FPGA. In the case of piecewise correlation, the ligand grid is downloaded into off-chip SRAM, whence it is loaded as just described.

Once the ligand is downloaded, the FPGA correlation starts, generating one correlation score per cycle. These scores are passed to a data reduction filter, which selects a pre-specified number of top scoring positions and stores them on the on-chip block RAMs. Upon completing the correlations for one rotation, the host program is instructed to download the ligand grid for next rotation. As the grids can be downloaded during processing, the data transfer latency can be completely hidden.

Our test for validation was done in two parts. The first validated the FFT correlation of the original PIPER against direct correlation sums. This was done using MATLAB: the results for the two correlations were identical. The second part involved verifying the direct correlation results generated by our hardware accelerator against those generated by a serial correlation code. We simulated our FPGA correlation pipeline using ModelSim and verified the resulting correlation scores against those generated by direct correlation function in MATLAB.

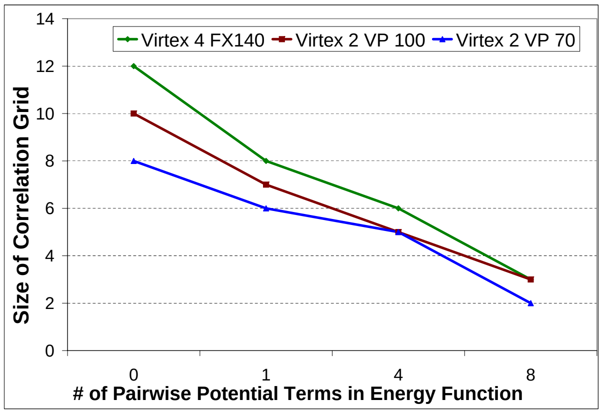

Figure 5: Complexity of correlation cell versus number of cells per dimension for various FPGA technologies.

A fundamental result here is the trade-off between the complexity of the energy function and the number of grid cells that can fit on the FPGA and thus the amount of parallelism in the design (see Figure 5, area determined through post place-and-route). For all cases, we assume the same four non-pairwise energy functions; the X-axis refers to the number of pairwise potentials computed in addition to those four. The Y-axis represents N, the size of one dimension of the 3D grid of computation cells. In the left-hand part of the graph, multipliers are the limiting resource. In the middle, the number of multipliers stays roughly the same; adding more pairwise correlations increases only slice utilization. In the right-hand part of the graph, the number of slices limits the grid size. Translating these values into performance, even the smallest FPGA results in a $100\times$ speed-up for typical protein-ligand combinations for the four (or fewer) terms generally used.

6 Discussion

We have presented an FPGA acceleration of a sophisticated, current, production docking code. In the process, we created a novel addition to our 3D correlation structure to enable effective computation of complex correlations. The result was a factor of hundred speed-up for 92.5% of the original computation for typical protein-ligand combinations. Acceleration of another 4% using an existing method brings the total acceleration up to $25\times$. Accelerating the remaining 3% is work in progress – using a previously developed method for charge-to-grid assignment [4] appears promising.

We believe that this is the only work in accelerating docking with FPGAs. Its significance is in its potential to drastically increase the pace of discovery in both basic science and in drug discovery. As PIPER gets integrated into the popular online ClusPRO system [2], the impact of this work should increase further.

Acknowledgments. We thank members of the Structural Bioinformatics group at Boston University for their help in understanding the PIPER code.

References

[1] Chen, R., and Weng, Z. A novel shape complementarity scoring function for protein-protein docking. *Proteins: Structure, Function, and Genetics 51* (2003), 397–408.

[2] Comeau, S., Gatchell, D., Vajda, S., and Camacho, C. ClusPro: an automated docking and discrimination method for the prediction of protein complexes. *Bioinformatics 20*, 1 (2004), 45–50.

[3] Gabb, H., Jackson, R., and Sternberg, M. Modelling protein docking using shape complementarity, electrostatics, and biochemical information. *Journal of Molecular Biology 272* (1997), 106–120.

[4] Gu, Y., and Herbordt, M. FPGA-based multigrid computations for molecular dynamics simulations. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines* (2007), pp. 117–126.

[5] http://pymol.sourceforge.net.

[6] Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A., Aflalo, C., and Vakser, I. Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques. *Proc. Nat. Acad. Sci. 89* (1992), 2195–2199.

[7] Kozakov, D., Brenke, R., Comeau, S., and Vajda, S. PIPER: an FFT-based protein docking program with pairwise potentials. *Proteins: Structure, Function, and Genetics 65* (2006), 392–406.

[8] Kuntz, I., Blaney, J., Oatley, S., Langridge, R., and Ferrin, T. A geometric approach to macromolecule-ligand interactions. *Journal of Molecular Biology 161* (1982), 269–288.

[9] Swartzlander, E. *Systolic Signal Processing Systems*. Marcel Dekker, Inc., 1987.

[10] VanCourt, T., Gu, Y., and Herbordt, M. FPGA acceleration of rigid molecule interactions. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications* (2004).

[11] VanCourt, T., and Herbordt, M. Rigid molecule docking: FPGA reconfiguration for alternative force laws. *Journal on Applied Signal Processing v2006* (2006), 1–10.