

## Molecular Fingerprinting on the SIMD Parallel Processor Kestrel

Eric Rice and Richard Hughey  
Department of Computer Engineering  
University of California, Santa Cruz, CA 95064  
{elrice,rph}@cse.ucsc.edu

### Abstract

In combinatorial library design and use, the conformation space of molecules can be represented using three-dimensional (3-D) pharmacophores. For large libraries of flexible molecules, the calculation of these 3-D pharmacophoric fingerprints can require examination of trillions of pharmacophores, presenting a significant practical challenge. Here we describe the mapping of this problem to the UCSC Kestrel parallel processor, a single-instruction multiple-data (SIMD) processor. Data parallelism is achieved by simultaneous processing of multiple conformations and by careful representation of the fingerprint structure in the array. The resulting application achieved a 35+ speedup over an SGI 2000 processor on the prototype Kestrel board.

### 1 Introduction

The large number of potential drugs that can now be synthesized and evaluated has led to the creation of immense synthetic combinatorial libraries of candidate molecules. This high-throughput combinatorial chemistry requires methods to evaluate molecular diversity within database libraries and to efficiently screen libraries for molecules likely to exhibit desired biological activity.

Both of these tasks require a method of abstracting useful information about molecules. One common strategy is to create molecular fingerprints by classifying the atoms of a molecule according to their electro-chemical properties, and then creating a bit-vector for the molecule based on whether specific configurations of labeled atoms ('pharmacophores') can occur in the conformational space of the molecule.<sup>1,2,3,4</sup> Such fingerprints, which are typically based on 2-, 3- or 4-point pharmacophores, can be used to evaluate diversity within libraries using a metric such as the Tanimoto coefficient or other method.<sup>1,2,4</sup> They also allow effective prediction of molecular behavior of library members by establishing relationships between specific pharmacophores (fingerprint bits) and targeted biological activity.<sup>5,4</sup>

A potential problem of the method is the extensive calculation required for fingerprinting combinatorial libraries. For 3-point pharmacophoric fingerprinting, every possible combination of three atoms (or in some cases, atom groups)

Table I: Pharmacophoric types

Label	Pharmacophoric definition
A	Hydrogen bond acceptor
D	Hydrogen bond donor
H	Hydrophobic
R	Aromatic
N	Negatively charged
P	Positively charged
-	All other atoms

Table II: Distance bin values.

Bin	Distances (angstroms)	
-	$d$	$< 2$
0	2	$\leq d < 4.5$
1	4.5	$\leq d < 7$
2	7	$\leq d < 10$
3	10	$\leq d < 14$
4	14	$\leq d < 19$
5	19	$\leq d < 24$
-	24	$\leq d$

of every conformation are considered for each molecule. For libraries containing hundreds of thousands of molecules with thousands of conformations each, this can involve trillions of atom triplet calculations, an impractical task on a serial processor.

This paper describes an attempt to address this computational bottleneck for the fingerprinting process of McGregor and Muskal<sup>4</sup> using the University of California at Santa Cruz Kestrel parallel processor, a single instruction multiple data (SIMD) 8-bit processor. While designed primarily for several computational biology algorithms, Kestrel was also designed to be as versatile as possible.<sup>6</sup> It has been shown to be effective for the targeted sequence matching algorithms (such as the Smith-Waterman algorithm), as well as for a number of other applications, such as Hidden Markov Models, and neural networks, where its performance approaches that of special purpose processors.<sup>7,8,9</sup>

While fingerprint generation represented a significant departure from Kestrel's target applications, the need for similar calculations on a large set of data made its SIMD architecture attractive. In the end, an efficient mapping was found requiring reorganizing the bits of the fingerprint for Kestrel and several other algorithm transformations. The application achieves a speedup of  $\sim 35$  on the prototype Kestrel board over an SGI Origin 2000 processor. A pending board redesign will provide additional speedup, and a second generation chip will provide further performance improvement.

After a description of the chemical fingerprinting problem, we describe the UCSC Kestrel system. Next, we present the implementation and evaluate its performance. Finally, we discuss the strengths and weaknesses of the approach.

## 2 Problem Overview

The fingerprint (FP) structure is based on six pharmacophoric labels (Table I) and six valid distance bins (Table II). Each bit of the FP represents a particular

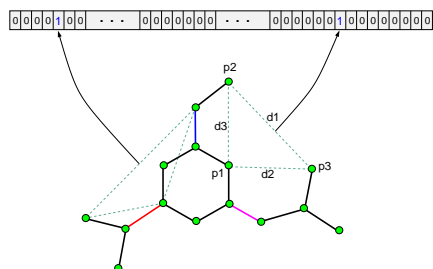


Figure 1: Fingerprint generation. Each group of three atoms maps to a particular bit (or bits) in the fingerprint, based on pharmacophoric labels  $p_i$  and atom-pair distances  $d_i$ . (A triplet can map to more than one bit when one of its atoms has more than one pharmacophoric label.)

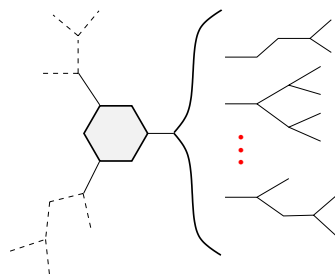


Figure 2: Hyperstructure molecule formation. Molecules are formed by choosing among multiple fragments for each of three positions on a central scaffold. Though not shown, each fragment will have many conformations.

arrangement of types and distance bins for a group of three atoms or atom groups. (For simplicity, we will refer to all pharmacophoric labels as belonging to ‘atoms’, though in fact they can represent a hydrophobic or aromatic group of atoms. Similarly, atom counts for molecules and sub-molecules will include these ‘pseudo-atoms’.) Distance bins that violate the triangle inequality are not represented. All permutations of three atoms are mapped to the same bit in the FP. The resulting fingerprint structure requires 6726 bits.

Generation of a fingerprint begins by initializing a bit-vector to all ‘0’s. Every possible set of three atoms of every molecular conformation is then examined. After atom types and distance bins have been determined for a particular set of three atoms, the appropriate bit in the FP is set to ‘1’ (Figure 1). This process is repeated until the finished FP represents the entire conformational space of 3-point pharmacophores for the molecule.

In the present study, molecules are formed using a ‘hyperstructure’ representation, consisting of a set of central scaffold atoms onto which three molecular fragments are attached (Figure 2). Each fragment is chosen from a set of candidates for that site. By pre-calculating atom coordinates for all conformations of each fragment choice, a large number of molecules can be generated at minimal cost (for  $f$  fragment choices at each position, each with  $k$  conformations,  $f^3$  molecules can be generated with  $k^3$  conformations each). The method also minimizes library storage requirements, since the partial structures can be stored separately and combined into complete molecules only when used.

The hyperstructure method requires checking that a set of specific fragment conformations is compatible—that it does not contain two atoms that are

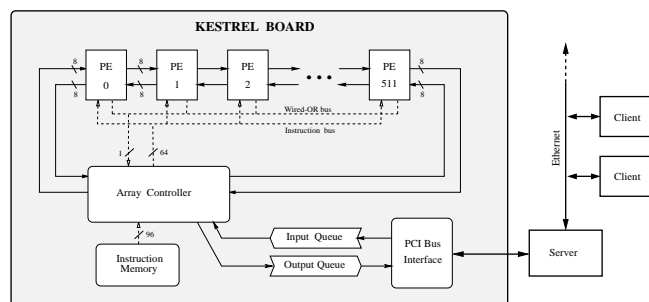


Figure 3: The Kestrel system high-level structure.

so close as to ‘bump’. This must be done before inserting any of the 3-point pharmacophores of the molecular conformation into the fingerprint. Bumps within a fragment and between a fragment and the scaffold can be screened during library formation, but conflicts between specific conformations of two different fragments must be determined for each possible pair.

To summarize, the Kestrel program inputs pre-calculated hyperstructure atom positions and pharmacophoric labels, identifies which molecular conformations are possible, and generates fingerprints based on these conformations.

### 3 Kestrel

The UCSC Kestrel Parallel Processor has 512 8-bit processing elements on a single PCI board (Figure 3).<sup>6,7</sup> The processing elements (PEs) are connected in a linear array. Data flows from the host to the board via 4 KB queues, and then can be passed through the array from one end to the other using 32-element shared register banks. Computation is performed according to the broadcast instructions. Each instruction controls the various functional units in each PE (local memory, multiplier, shift register, comparator, arithmetic logic unit) as well as operations within the array controller (input and output, data broadcast, branches, and loops). PEs can be masked for local conditional operation; nested conditionals are supported by the bit-shifter in each PE which can maintain nested values without overhead. A global wired-OR allows global branching based on events occurring anywhere in the array. The local memory (256 bytes) can be independently addressed in each PE. Because of Kestrel’s small word size, the add, compare, and multiply units have been designed for efficient multiprecision operations.

There are four running 20 MHz Kestrel boards at UCSC, one of which powers our Smith & Waterman WWW server. Two other boards are currently on loan, one to UCSF for large-scale Smith & Waterman searches for genome analysis, and the other to a pharmaceutical company for accelerating protein HMM searching.

The Kestrel group is designing a new board for the current Kestrel chips. The system will have 1024 PEs, a pipelined controller, and addressable, on-board memory. The on-board memory, added based on our computational chemistry experiences, will eliminate the need to use PEs for auxiliary memory, as discussed below. A higher clock speed, together with the larger number of PEs and on-board memory, will provide an additional factor of 4–5 performance improvement for the molecular fingerprinting application.

## 4 Implementation

SIMD machines broadcast the same control signals to all PEs, so program efficiency is largely determined by the extent to which multiple data requires the same manipulation. When fingerprinting large libraries containing flexible molecules, each molecule will have thousands of conformations. By parsing the Kestrel array into multiple blocks, each of which contains a different molecular conformation, instructions can be broadcast to process multiple conformations (or even multiple parts within each conformation) at the same time.

Given this overall approach, fingerprint generation has four main parts:

- Loading of conformations: Atom coordinates provided through the input stream must be loaded into blocks of PEs so that each contains a different molecular conformation.
- Calculation of distance bins: In each conformation-processing block, distance bins for all atom pairs need to be calculated. For efficiency, these distance bins are stored in the array for repeated access.
- Calculation of 3-point pharmacophores: For each atom triplet of the molecule, the three appropriate distance bins need to be accessed and combined with the pharmacophoric labels.
- Insertion into FP: For each 3-point pharmacophore, the appropriate FP bit needs to be set.

### 4.1 Conformation loading and distance bin calculation

Each atom coordinate is inputted and stored in the array using two bytes of SRAM, one integer and one fractional. This provides sufficient resolution given the coarse-grained nature of the fingerprinting method. The number of

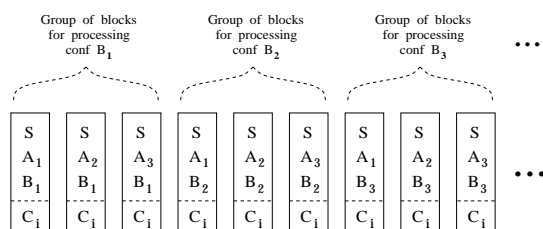


Figure 4: Conformation loading. Each box represents a conformation processing block. For molecule  $SABC$ , groups of blocks containing all conformations of  $A$  (3 in this case) are loaded with different conformations of  $B$ . A single conformation of  $C$  is then loaded into all blocks.

atoms that can be stored in a single PE's SRAM is  $\lfloor 256/6 \rfloor = 42$  atoms. This determines the limit on molecular size in the present program.

Since every atom pair is constituent to multiple triplets, it is more efficient to pre-calculate and store distance bins in the array than to re-calculate them for every triplet. In addition to avoiding redundant calculations, this allows bumps between fragment conformations to be identified before processing any triplets of the conformation, avoiding the necessity to pre-calculate and input conflict information.

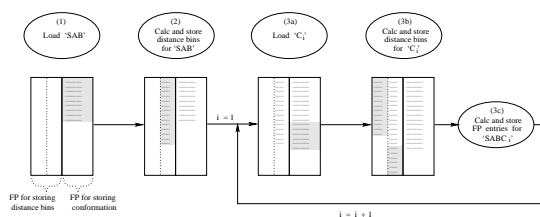
The size of conformation-processing blocks is determined by the number of PEs required for storing distance bins for a particular molecule. Since each distance bin can be specified by 3 bits (6 valid distance ranges plus 1 to represent invalid distances), two bin values can be stored in each byte of SRAM. A single PE thus stores up to  $2 \times 256 = 512$  bins, sufficient for molecules containing up to 32 atoms (requiring  $32 \times 31/2 = 496$  bins). For larger molecules, two PEs are sufficient for 45 atoms, more than the 42-atom limit based on storage of atom coordinates. Conformation processing blocks in the present program thus contain either 2 or 3 PEs, depending on molecular size.

Taking advantage of the large number of conformations each fragment is expected to have, each block is loaded with a different conformation of the sub-molecule  $SAB$  (the scaffold  $S$  and two of the three side fragments). The same conformation of fragment  $C$  can then be broadcast to all blocks in the array, minimizing loading costs in each round of calculations (Figure 4). This arrangement reduces the cost of calculating distance bins. By pre-calculating and storing distance bins for the  $SAB$  sub-molecule, after each conformation of  $C$  is subsequently loaded, distance bins for the molecule can be completed by considering only atom-pairs containing at least one atom in  $C$  (Figure 5).

#### 4.2 Calculating 3-point pharmacophores

After distance bins are calculated and stored, atom triplets must be looped through, accessing the correct three distance bins for each. The preprocessor

Figure 5: Conformation loading and distance bin calculations in a block of 2 PEs during a round of fragment  $B$  conformations. Shaded areas show newly written data. Each  $C$  conformation is looped through before reloading  $SAB$ .



groups triplets of each molecule so that all triplets containing the same three pharmacophoric labels are examined in succession. By keeping track of where distance values for each atom-pair will be stored in the conformation blocks, distance bin access information can be sent to Kestrel via the input file. Since all blocks access the same three SRAM locations for the distance bins of a given set of three atoms, this data is broadcast to all blocks.

There are two advantages of using the preprocessor to manage this process. First, the pharmacophoric labels and atom-pair bin locations do not need to be stored in the array, where memory is at a premium. Second, the overhead of accessing distance bin values is minimized by providing exact location information to the array.

A drawback of the approach is the large amount of data that must be broadcast to the array (potentially gigabytes). To reduce the amount of data needing to be sent to the Kestrel board, a block of PEs in the Kestrel array is partitioned as auxiliary memory for storing this triplet-processing information. After being loaded from the input stream for each molecule, this data can be broadcast repeatedly to the array in each round of calculations needed for the molecule. There can be dozens of rounds for large flexible molecules, where all conformations of  $C$  must be sequentially examined in each of several rounds of  $B$  conformations.

#### 4.3 Inserting 3-point pharmacophores into fingerprints

As described above, the Kestrel program produces multiple 3-point pharmacophores with the same pharmacophoric labels in each round. The FP structure used in the Kestrel array is designed for efficient insertion of such a set of entries. While different than the target FP structure, the resulting FPs can be efficiently converted to the target structure during post-processing.

The most important feature of the design is that each combination of three pharmacophoric types is mapped to a particular bit-column of SRAM (Figure 6). Multiple 3-point pharmacophores produced simultaneously then need placement in the same bit-column of the same PE.

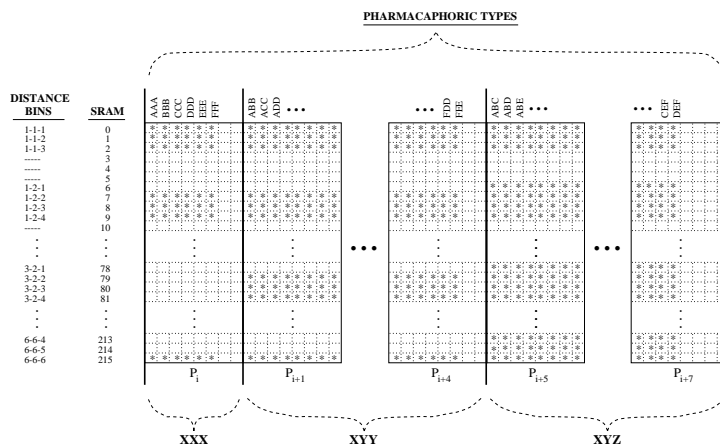


Figure 6: Fingerprint representation in the Kestrel array. Eight PEs are used to represent each fingerprint. Each bit-column of each PE stores information about a particular combination of three pharmacophoric types, and each SRAM between 0 and 215 represents a particular combination of three distance bins (some of which are impossible). Combinations of three pharmacophoric labels are grouped according to whether they allow sorting by distance bin values (XXX, XYY, or XYZ) to increase fingerprint compactness.

The set of three distance bins for each entry is mapped to an SRAM address of the appropriate bit-column by calculating  $(36d_1) + (6d_2) + (d_3)$ , with  $d_1, d_2, d_3 \in \{0, 1, 2, 3, 4, 5\}$ .

Insertion of a set of 3-point pharmacophores into the structure is very efficient, requiring only 2 cycles per entry. This efficiency more than makes up for the increased size of the redesigned FP, which requires 8 instead of 4 PEs per fingerprint. Since some of the bytes of SRAM in these 8 PEs do not contain data, a fingerprint can be output from the array in 963 bytes (compared to a minimum possible 841 bytes).

Another important strategy for reducing the cost of inserting FP entries is to maintain multiple fingerprints equally spaced throughout the Kestrel array, allowing parallel insertions. After processing all molecular conformations, these 'partial' fingerprints are logically Ored together to produce a finished fingerprint for a molecule. Because each FP copy displaces conformation processing blocks, there is a tradeoff between insertion efficiency and triplet-processing efficiency. A super-block size of 32 PEs was found to be efficient for the range of molecular sizes used in the present program. In addition to an 8-PE fingerprint, the super-blocks contain either 8 blocks of 3 PEs, or 12 blocks of 2 PEs,

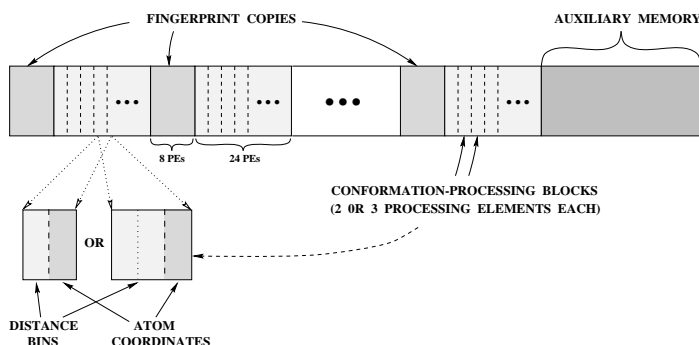


Figure 7: Parsing of Kestrel array for fingerprint generation. Sizes of conformation-processing blocks and auxiliary memory are determined by molecular size.

depending on the size of conformation processing blocks. The final parsing of the Kestrel array is shown in Figure 7.

#### 4.4 Implementation Summary

In summary, the present fingerprinting system includes three programs:

1. Pre-processing program:
  - Determine array parsing.
  - Insert into the Kestrel input file the arrays of PE labels needed for selective masking of blocks within the array.
  - Insert into the Kestrel input file the triplet access information to be stored in the auxiliary memory.
  - Insert into the Kestrel input file the atom coordinates of fragment conformations.
2. Kestrel program:
  - Load masking arrays into Kestrel to establish block structure.
  - Load triplet processing data into auxiliary memory block.
  - Load conformations as shown in Figure 4.
  - Loop through all triplets of all molecular conformations as shown in Figure 5, where step 3c is performed in each round using the distance bin locations stored in the auxiliary memory block. (These values are shifted out of the right side of the array to the controller and then broadcast via the instruction 'immediate' field to all conformation blocks.) Once calculated, each 3-point pharmacophore is stored in the nearest fingerprint to its left.

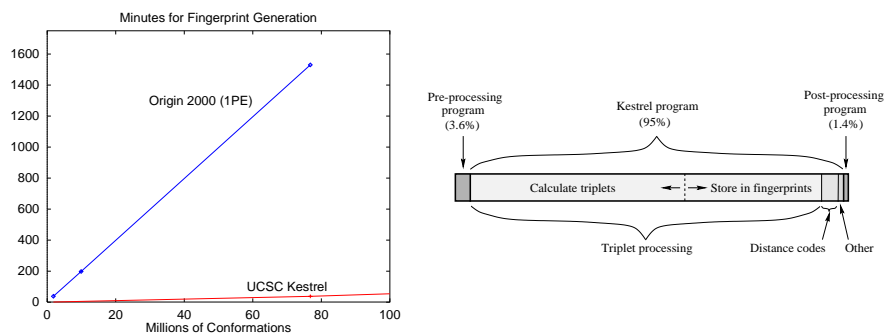


Figure 8: Performance on the Kestrel system and an SGI Origin 2000 serial processor (left) and relative execution time for the different parts of the Kestrel program (right).

- After all triplets of all conformations have been processed, logically OR all fingerprint copies together and output from the array.
3. Post-processing program:
    - Convert the outputted FPs to the desired FP format and store.

## 5 Results

The Kestrel chemical fingerprinting program was evaluated on several subsets of a large hyperstructure library. The library consisted of 20 possible fragments for each of the three scaffold positions, with an average of 21.3 conformations per fragment. The average size of each fragment was  $\sim 9$  atoms (or in some cases 'pseudo-atoms'). Together with a scaffold containing 8 atoms, this led to hyperstructure molecules averaging  $\sim 35$  atoms with nearly 10,000 conformations each.

The results are compared to those of a Silicon Graphics Origin 2000 serial program in Figure 8, where a speedup of over 35 was achieved by Kestrel for all tested subsets. The most significant problem with the program was the 42-atom limit on molecular size. Almost 7% of the hyperstructure molecules were too large for the program, with the largest molecule containing 53 atoms. To compensate for the time saved on these large molecules (which were skipped over by the present program), extra conformations of other molecules were added to make the results reflect the number of triplets which would have needed to be calculated.

A study of the distribution of labor shows that triplet processing dominates the Kestrel program (Figure 8). This cost depends on molecular size and ranges from  $\sim 0.4$  to  $\sim 0.9$  cycles per fingerprint entry. The most costly single process

is calculating 3-point pharmacophores (accessing the correct distance bins and combining them into a value between 0 and 215). This cost increases with block size, since more distances need to be accessed to fill up all the PEs of a block with different triplet information prior to parallel calculations and insertions.

Inserting values into the FP represents the other significant cost, and is nearly constant for all molecular constraints. The remainder of the program, representing less than 10%, consists mostly of distance bin calculations, preparation of Kestrel's input file, and conformation loading.

## 6 Discussion

Two modifications of the program would make it compatible with any molecular structure. The first is to remove the 42-atom limit by storing atom coordinates in more than one PE when needed (with a corresponding increase in the number of PEs storing distance bins). For molecules greater than  $\sim 70$  atoms, it becomes best to recalculate distance bins for each atom triplet, resulting in a (nearly worst-case) cost per triplet 2.5–3.5 times higher than in the present program. The second modification is to make conformation loading more flexible and molecule-specific. This would allow the program to accommodate molecules whose 3 fragments each contain more conformations than there are conformation blocks in the array, as well as making the program more efficient on molecules with fragments containing small numbers of conformations.

For larger fingerprint structures, such as would be needed for fingerprints based on 4-point pharmacophores<sup>3</sup>, storing complete FPs in the Kestrel array would not be practical due to the limited local memory. An alternative would be to calculate and store only part of the fingerprint at a time in the array. The partial FPs of each molecule could then be combined during post-processing.

The results of the present study show that the SIMD paradigm can be extremely efficient for problems requiring significant parallel computation. Arrays containing small, simple, processing elements such as Kestrel allow a large number of processors to fit on a single board. This avoids the overhead often associated with parallel processing, and provides significant computing power in a convenient and cost-effective format. While in some cases (as here), efficient mapping requires creative examination of a problem, the price/performance advantage is significant; over 35 SGI Origin 2000 processors would be needed to duplicate Kestrel's prototype single-board performance for chemical fingerprinting.

## Acknowledgments

The authors greatly thank Steven Muskal and Malcom McGregor for their willingness to explain and reexplain the application. We also thank the Kestrel team for many helpful discussions. Additional information on the UCSC Kestrel project can be found at <http://www.cse.ucsc.edu/research/kestrel>.

This work was supported in part by NSF grant EIA-9905322 and a grant from the Affymax Research Institute, a member of the Glaxo Wellcome group.

## References

1. D. D. Pickett, J. S. Mason, and I. M. McLay, "Diversity profiling using 3D pharmacophores: Pharmacophore-derived queries (PDQ)," *J. Chem. Inf. Comput. Sci.*, vol. 36, pp. 1214–1223, 1996.
2. S. Pickett, C. Luttmann, V. Guerin, A. Laoui, and E. James, "DIVSEL and COMPLIB—strategies for the design and comparison of combinatorial libraries using pharmacophoric descriptors," *Journal of Chem. Inf. Comput. Sci.*, vol. 38, pp. 144–150, 1998.
3. J. Mason and D. Cheney, "Ligand-receptor 3-D similarity studies using multiple 4-point pharmacophores," *Proceedings of the Pacific Symposium on Biocomputing*, vol. 4, pp. 456–467, 1999.
4. M. J. McGregor and S. M. Muskal, "Pharmacophore fingerprinting: Application to QSAR and focused library design," *J. Chem. Inf. Comput. Sci.*, vol. 39, no. 3, pp. 569–74, 1999.
5. X. Chen, A. Rusinko, and S. S. Young, "Recursive partitioning analysis of a large structure—activity data set using three-dimensional descriptors," *Journal of Chem. Inf. Comput. Sci.*, vol. 38, pp. 1054–1062, 1998.
6. J. D. Hirschberg, D. Dahle, K. Karplus, D. Speck, and R. Hughey, "Kestrel: A programmable array for sequence analysis," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 19, no. 2, pp. 115–126, 1998.
7. L. Grate, M. Diekhans, D. Dahle, and R. Hughey, "Sequence analysis with the kestrel simd parallel processor," in *Proceedings of the Pacific Symposium on Biocomputing*, 2001.
8. A. Di Blas and R. Hughey, "Explicit SIMD programming for asynchronous applications," in *Proc. Int. Conf. Application-Specific Systems, Architectures, and Processors*, pp. 258–267, 2000.
9. R. Hughey, "Parallel sequence comparison and alignment," *CABIOS*, vol. 12, no. 6, pp. 473–479, 1996.