

The UCSC Kestrel General Purpose Parallel Processor*

David Dahle, Leslie Grate, Eric Rice, Richard Hughey
Department of Computer Engineering, Jack Baskin School of Engineering
University of California, Santa Cruz, CA 95064
<http://www.cse.ucsc.edu/research/kestrel>
email: rph@cse.ucsc.edu

Abstract *The UCSC Kestrel Project has built a single-board programmable parallel processor with 512 processing elements (PEs). The fully operational system features a dense 1.4 million transistor chip composed of 64 byte-wide PEs, each with its own memory. The system architecture was designed specifically to support a large variety of computational biology algorithms, for which programmability was essential, but large memories and arbitrary communication networks were not. The Kestrel system performs Smith & Waterman and hidden Markov model biological sequence searches 20 times faster than a 433-MHz DEC Alpha, and is 35 times faster than one processor of an SGI Origin 2000 on a computational chemistry application.*

Keywords: SIMD, parallel processing, sequence analysis

1 Architecture

The UCSC Kestrel Project has built a single-board programmable parallel processor with 512 processing elements (PEs). The fully operational system features a dense 1.4 million transistor chip composed of 64 byte-wide PEs, each with its own memory. The system architecture was designed specifically to support a large variety of computational biology algorithms, for which programmability was essential, but large memories and arbitrary communication networks were not.

For programmability, we chose a SIMD design with broadcast instructions. This design provides a simpler, and in many ways more flexible, programming model than reconfigurable machines while avoiding overwhelming the small PEs with control circuitry as a MIMD design would.

For memory, a moderate amount of locally-addressed memory was required for the sequence analysis applications. Almost half of the Kestrel PE area is devoted to a 256-byte, locally addressed, single-port SRAM, while 8% of the PE area is devoted to a higher-bandwidth 32-byte, 3-ported (2 read, 1 write) register bank.

For communications, the linear array is well suited for the computational biology applications of interest. PEs communicate with each other via shared register banks [12]. For systolic algorithms, in which data flows through the array as computation is performed, the shared registers provide a seamless integration of communication and computation. In a sorting algorithm, for example, a single 1-cycle instruction can compare two numbers in the left register bank and store the minimum in the right register bank, making it available to the next processing element during the next cycle.

The core datapath of the an individual Kestrel PE (Figure 1) includes an ALU with an integrated minimizer to enable a family of add-and-min instructions, a signed and unsigned 8-bit multiplier, and a shift register and mask flag to support bit packing and unpacking as well as nested conditionals. The multiplier includes a four-operand multiply-accumulate-

*Supported in part by UCSC, NSF grants MIP-9423985 and DBI-9808007, and the Affymax Research Institute.

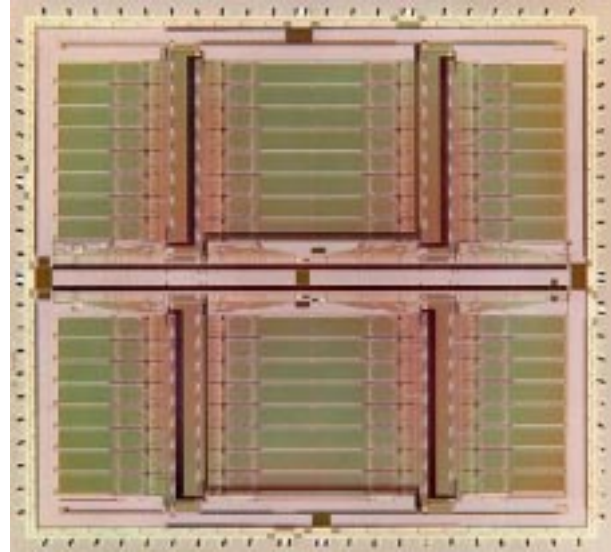
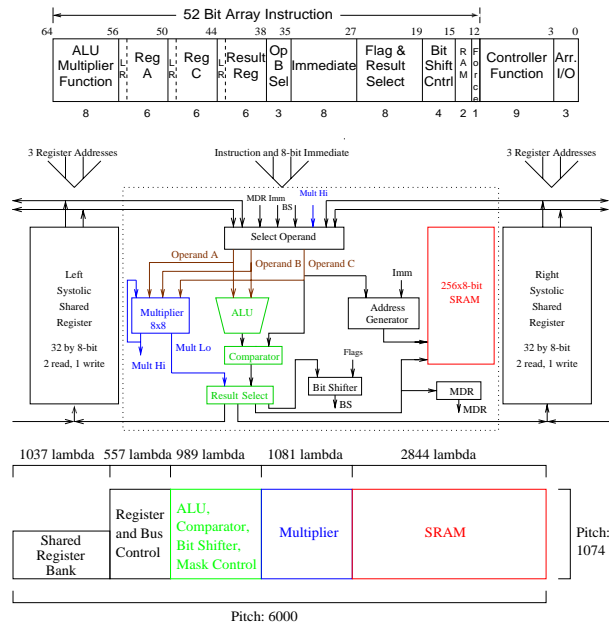


Figure 1: Kestrel microcode format, processing element, PE floorplan, and 64-PE chip.

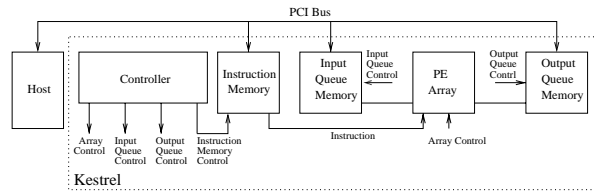


Figure 2: Kestrel board architecture.

accumulate instruction that greatly speeds multiprecision multiplication (requiring 71 cycles for a 64×64 -bit multiply with 128-bit product rather than 121 cycles with multiply-accumulate or 247 cycles with just multiply). We discovered an elegant means of compacting the ALU function encoding that may be useful to other designers of VLSI ALUs [5].

The Kestrel PCI board (Figures 2 and 3) has 512 processing elements in 8 chips, as well as instruction memory, a controller, and data queues. Kestrel uses some of the principles of long instruction word computers. Each instruction can specify several actions in the Kestrel PE, such as performing an add-min instruction that stores the selector bit and result while loading the memory data register (MDR) with a value from memory, along with several independent controller functions, such as copy-

ing data to and from the queues and performing a branch. Thus, jumps and branches have no cost because they are performed simultaneously with an array instruction.

Kestrel programs are currently written in an assembly language called Kasm. The Kestrel runtime environment is a uniform front end to the two ways of running a Kestrel program, a software simulator and the Kestrel PCI board. The environment enables instruction downloading, data transfer, and control and examination of the real or simulated controller and board. Both command line and graphical program debugging facilities are available. The Kestrel PCI board is installed in a PC running Windows NT. The system allows for both off-host and on-host data (data residing on the host allows the fastest operation of the Kestrel board). Kestrel became operational September 1998: we currently have two running 512 PE boards.

2 Applications

Two primary target applications are the Smith & Waterman (SW) [24] and hidden Markov model (HMM) [14] methods of sequence anal-

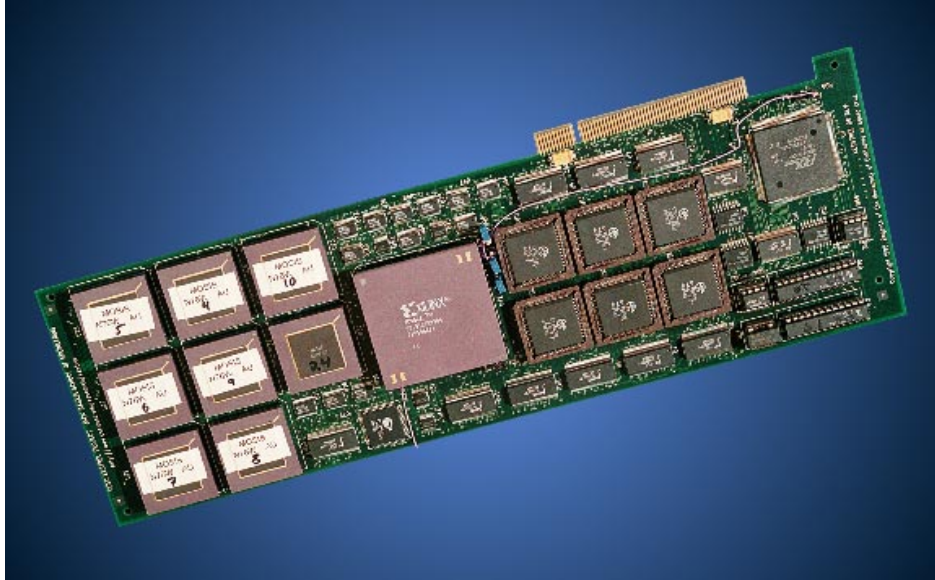


Figure 3: The 512-PE Kestrel Parallel Processor.

ysis. Many sequence analysis techniques, including HMMs, rely on aligning sequences in the database to a model or to other sequences, often with a variant of the following affine gap computation. Given two sequences, a and b , the total cost to transform one sequence into the other is calculated with three basic operations: deletion, insertion, and mutation of a character. The following dynamic programming recurrence computes the Smith & Waterman score:

$$I_{i,j} = \max \begin{cases} I_{i-1,j} + c \\ M_{i-1,j} + g \end{cases} \quad D_{i,j} = \max \begin{cases} D_{i,j-1} + c \\ M_{i,j-1} + g \end{cases}$$

$$M_{i,j} = \max \begin{cases} I_{i-1,j-1} + d(a_i, b_j) \\ D_{i-1,j-1} + d(a_i, b_j) \\ M_{i-1,j-1} + d(a_i, b_j) \\ 0 \end{cases}$$

where $d(a_i, b_j)$ is the cost of matching a_i to b_j , g is the cost of starting a gap, and c is the cost of continuing a gap.

The dynamic programming calculation easily maps to a linear array of processing elements [10], and was the main reason for much of Kestrel's design. A common mapping is to assign one PE to each character of the query string, and then to shift the database through the linear chain of PEs. Typical query

Table 1: Wall time in seconds to search a 10Mbase database on the 20 MHz Kestrel.

Query Size	Kestrel	433 MHz DEC alpha		
	≤ 512	32	128	512
SW	12	20	73	282
HMM (one path)	41	80	232	862
HMM (all paths)	236	155	541	2120

strings are hundreds to thousands of characters, matching Kestrel's 512-PEs, though longer queries require storing several adjacent characters in each PE's local memory.

Architecturally, sequence comparison can be aided by variable precision arithmetic (the d values can be at a lower precision than the c values), minimization, and addition. The twin problem of sequence alignment, finding the minimizing correspondence between two sequences, requires the saving of the selector bits of the minimizations and recirculation of sequence data [8].

The Kasm assembly language code for SW (Figure 4) mirrors the orthogonality in the Kestrel PEs and controller. Each line of Kasm code specifies one or more opcodes and modifiers, and an appropriate number of operands, to specify array functions.

```

;; Insert <-- max (Insert+continue,MGC)          ;; Add the gap cost to the Match cost for
add      R$Inslo, R$Inslo, $CONTL              ;; Delete and Insert calculations
smaxc add mp R$Inshi, R$Inshi, $CONTH, R$MGChi add R$MGClo, L$TMPlo, $GAP_LO
smaxc cmp R$Inslo, R$Inslo, R$MGClo           add R$MGChi, L$TMPPhi, $GAP_HI

;; Delete <-- max (Delete+continue,MGC)        ;; Store the best score so
add      R$Dello, L$Dello, $CONTL              smaxc L$score_hi, L$score_hi, L$TMPPhi
smaxc add mp R$Delhi, L$Delhi, $CONTH, R$MGChi smaxc cmp L$score_lo, L$score_lo, L$TMPlo
smaxc cmp R$Dello, R$Dello, R$MGClo

;; Shift the sliding sequence;
;; read a new value from the queue;
;; Lookup the character cost in SRAM
move R$Seq, L$Seq, qtoarr, read(L$Seq)

;; Match <-- max(MDI+charcost,0)
;; Zero-threshold for local scoring
;; of Smith & Waterman
add      L$TMPlo, L$MDIlo, mdr
smaxc add mp L$TMPPhi, L$MDIhi, smdr, #0
smaxc cmp L$TMPlo, L$TMPlo, #0

;; MDI <-- max (Match, Insert, Delete)
;; for future Match calculation.
;; Branch to start of nested loop
smaxc R$MDIhi, L$TMPPhi, R$Inshi
smaxc cmp R$MDIlo, L$TMPlo, R$Inslo
smaxc R$MDIhi, R$MDIhi, R$Delhi
smaxc cmp R$MDIlo, R$MDIlo, R$Dello, endLoop

;; mp = multiprecision ALU op
;; cmp = topdown multiprecision comparator op
;; smdr = sign extension of the 1-byte MDR
;; smaxc= signed maximum with operand C

```

Figure 4: The core kasm code for Smith and Waterman dynamic programming.

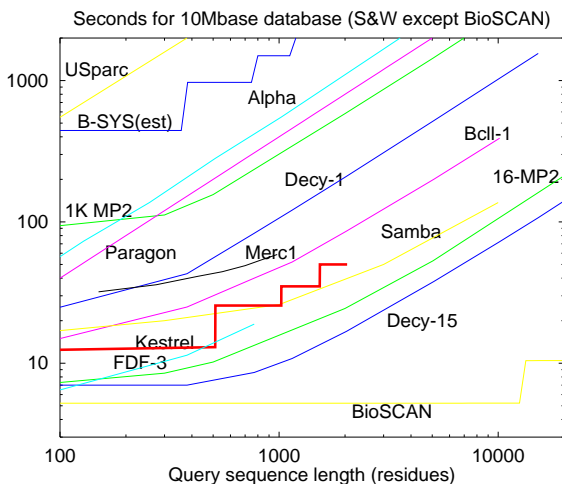


Figure 5: Database search time as a function of protein query length.

On SW, a single 20 MHz Kestrel board is 20 times faster than the 433 MHz DEC Alpha, and a board redesign would make this a factor of 40 (Figure 5). Kestrel is slightly slower than the much larger 16 384-processor MasPar and a 15-board FPGA-based system, and comparable to a 3-board single-purpose FDF machine. The stepping in Kestrel's performance occurs at multiples of the 512 array length. Kestrel's programability makes it far more flexible than the other specialized systems.

Machines: BioSCAN (single purpose, not Smith & Waterman) [23], 15-board Timelogic Decypher-II (FPGA) [27], 1-board Kestrel, 5-board Paracel FDF (single purpose) [17], 16,536-PE MasPar [16], 2-board SAMBA (single purpose) [9], 1-board Mercury (single purpose) [3], Biocellator (FPGA) [4], 1-board Decypher-II (FPGA), 32-node Paragon [20], 1024-PE MasPar, 433 MHz DEC Alpha, B-SYS [12], and Sun Ultrasparc 140. Since data collection in 1995-6, TimeLogic, Paracel, and Compugen have announced multiboard systems with configurations for about \$120,000 for unverified performance between the 16K MP-2 and the 15-board Decypher II. See related surveys [10, 26].

2.1 HMM Sequence Analysis

Linear hidden Markov models (HMMs, Figure 6) are a powerful extension of standard sequence analysis methods [14, 11, 7, 6]. Instead of comparing or aligning one sequence against another, one compares or aligns one sequence against a statistical model of the family of sequences of interest.

There are two parts to this process: training (building and refining) an HMM, and then using it. Taking the simpler part first, using

Squares represent match states, in which a position in the model corresponds to a position in the sequence. Circles represent delete states with no corresponding sequence character, and diamonds represent insertions of one or more characters. All transitions have associated probabilities. Dynamic programming is used to align the two sequences according to their best possible paths through the HMM.

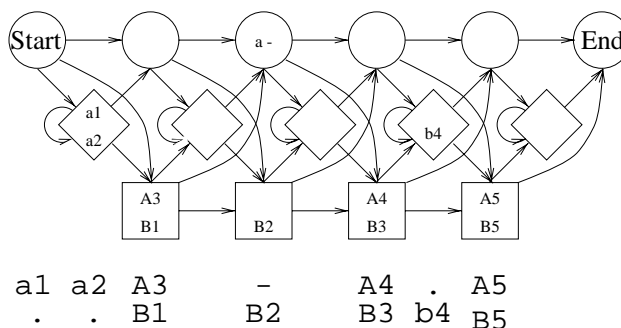


Figure 6: A linear hidden Markov model.

an HMM for sequence discrimination is essentially the same as other methods of dynamic programming for sequence comparison except that all character and transition probabilities are local to each position in the model. Thus, when using the HMM, these probabilities are first loaded into the Kestrel array, and then the database sequences are slid through the array in the course of performing the dynamic programming. As with simple sequence analysis, if an alignment rather than just a score is required, information must be saved and. Searching a 100 MB non-redundant protein database using a 400-node HMM requires 16 hours on a 433 MHz DEC Alpha but only a few minutes on Kestrel.

The most time consuming part of the HMM methodology is, however, training the probabilities from the sequence(s) of interest. This process consists of repeating the $O(n^2)$ dynamic programming calculation for perhaps 50 times for each sequence in the training set.

With both sequence-against-sequence and HMM methods, far more effective searches can be performed by using the sequence databases as well as the query sequence. In the sequence-sequence case [1, 19, 18], the target sequence is used to search the database for closely related sequences, and each of these is in turn used to search the database. A sequence is categorized as related to the target if it is found, according to some threshold, by either the target or one of the closely related sequences. This iterative method results in far better sequence discrimination [18]. UCSC's SAM-T98 iter-

ative HMM search procedure extends hidden Markov model methodology to iterative search, extending the capabilities of remote homology search beyond previous art [18, 13]

The higher computation requirements of HMMs (Table 1) provide significantly better database searching [13]. The most sensitive HMM method (all paths) can only be performed on programmable machines.

2.2 Computational Chemistry

We are investigating computational chemistry algorithms on Kestrel, in which a combinatorial library of small molecule alternatives is searched to find a desired attribute, such as having a likelihood of interacting with a certain protein's active site [2, 21]. Each of tens of thousands of molecules in the library can have tens of thousands of different configurations (conformations) to analyze for feasibility (Figure 7). The computational power of a million-processor machine would be invaluable in this endeavor.

Analyzing this massive number of 3-dimensional possibilities can be performed by first calculating distances between triples of atoms across a set of conformations of a given molecule. The atoms and distances are binned, and a bit-vector is computed based on the presence or absence of each possible set of six atom types and 6 distances for each side of the triangle. The resulting bit vector (~ 6700 bits) can then be used for structure comparison and database search [15].

On Kestrel, each conformation of a molecule

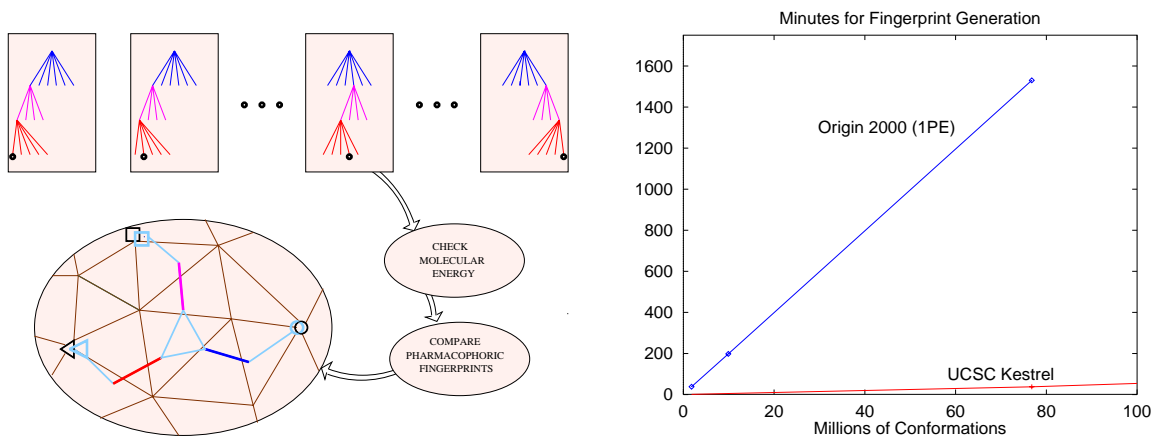


Figure 7: Conformation analysis on the Kestrel array.

is mapped to a small group of processing elements. Pairwise distances are then calculated for all conformations, and the bit-vector is formed by considering triples of atoms from all groups of PEs. The Kestrel implementation is approximately 35 times faster than one processor of an SGI Origin 2000 (Figure 7).

3 Looking to the Future: The Million Processor Machine

The decline of feature size is having a profound affect on computer architecture. The 64-PE chip is a relatively small 7.2 mm by 8.3 mm in a 0.5 μm process. A currently readily available 0.25 μm process would enable 256 PEs in a similar size. Getting down to 0.10 μm will enable a Kestrel-on-a-chip: 1024 processing elements in a 64 \times 64 array, a 16K \times 64 on-chip instruction memory implemented in high-speed scalable CMOS DRAM [25, 22], 32 KBytes of staging memory, a controller, and 128-element queues for synchronization with the outside world. Extrapolating current performance numbers to such a system indicate that whole genomes could be searched with complex models in seconds with this plug-in supercomputer board costing only \$15 to \$25 thousand dollars.

Moving to a larger system (and power supply), 32 or 64 of these chips could be usefully

placed on a board, enabling the construction of a relatively small million-processor machine in 16 or 32 boards within five years. With the chips running conservatively at 250 MHz, such a machine would provide 250×10^{12} 8-bit operations per second, 60 TOPS of 32-bit integer performance, 13 TOPS of 32-bit integer multiplication, and ~ 1 TFLOPS (64-bit addition; division would be ~ 1.5 TFLOPS and multiplication ~ 2.3 TFLOPS), assuming our current Kestrel architecture. Considering that each Kestrel PE can perform an addition, memory access, comparison, selection, and shift during every instruction, this could be considered a PETAOP machine.

Of course, although it is an exciting design point, there is no intrinsic need to build a million-processor machine unless there are applications, or mixes of applications, that can make use of such an architecture and there are methods of programming the machine.

A million-processor machine could certainly make an effective computational biology, text, or html search engine, but these are tasks that could also be solved with a cluster of small Kestrel systems. On the other hand, the computational intensity of combinatorial chemistry and protein structure evaluation will be able to make full use of a tightly coupled, massively parallel machine.

References

- [1] S. Altschul *et al.*, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *NAR*, vol. 25, pp. 3899–3402, 1997.
- [2] G. W. Bemis and I. D. Kuntz, "A fast and efficient method for 2D and 3D molecular shape description," *J. Computer-Aided Molecular Design*, vol. 6, pp. 607–628, 1992.
- [3] D. Brutlag, J.-P. Deautricourt, and J. Griffin. Personal Communication, Oct. 1995.
- [4] Compugen Ltd., "Biocellator information package." Obtained from compugen@datasrv.co.il, 1994.
- [5] D. M. Dahle, J. D. Hirschberg, K. Karplus, H. Keller, E. Rice, D. Speck, D. H. Williams, and R. Hughey, "Kestrel: Design of an 8-bit SIMD parallel processor," in *Proc. 17th Conf. on Advanced Research in VLSI* (R. B. Brown and A. T. Ishii, eds.), pp. 145–162, IEEE CS, Sept. 1997.
- [6] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
- [7] S. Eddy, "Hidden Markov models," *Curr. Opin. Struct. Biol.*, vol. 6, no. 3, pp. 361–365, 1996.
- [8] J. A. Grice, R. Hughey, and D. Speck, "Reduced space sequence alignment," *CABIOS*, vol. 13, no. 1, pp. 45–53, 1997.
- [9] P. Guerdoux-Jamet and D. Lavenier, "SAMBA: hardware accelerator for biological sequence comparison," *CABIOS*, vol. 13, pp. 609–615, Dec. 1997.
- [10] R. Hughey, "Parallel sequence comparison and alignment," *CABIOS*, vol. 12, no. 6, pp. 473–479, 1996.
- [11] R. Hughey and A. Krogh, "Hidden Markov models for sequence analysis: Extension and analysis of the basic method," *CABIOS*, vol. 12, no. 2, pp. 95–107, 1996.
- [12] R. Hughey and D. P. Lopresti, "B-SYS: A 470-processor programmable systolic array," in *Proc. Int. Conf. Parallel Processing* (C. Wu, ed.), vol. 1, (Boca Raton, FL), pp. 580–583, CRC Press, Aug. 1991.
- [13] K. Karplus, C. Barrett, and R. Hughey, "Hidden Markov models for detecting remote protein homologies," *Bioinformatics*, vol. 15, no. 1, 1999.
- [14] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, "Hidden Markov models in computational biology: Applications to protein modeling," *JMB*, vol. 235, pp. 1501–1531, Feb. 1994.
- [15] M. J. McGregor and S. M. Muskal, "Pharmacophore fingerprinting: Application to QSAR and focused library design," *J. Chem. Inf. Comput. Sci.*, p. to appear, 1999.
- [16] J. R. Nickolls, "The design of the Maspar MP-1: A cost effective massively parallel computer," in *Proc. COMPCON Spring 1990*, (Los Alamitos, CA), pp. 25–28, IEEE Computer Society Press, Feb. 1990.
- [17] Paracel Inc, "FDF-3 product information." Pasadena, CA, <http://www.paracel.com>, 1996.
- [18] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia, "Sequence comparisons using multiple sequences detect twice as many remote homologues as pairwise methods," *J. Mol. Biol.*, vol. 284, no. 4, pp. 1201–1210, 1998.
- [19] J. Park, S. Teichmann, T. Hubbard, and C. Chothia, "Intermediate sequences increase the detection of homology between sequences," *JMB*, vol. 273, pp. 349–354, 1997.
- [20] W. R. Pearson, "Personal communication," 1995.
- [21] 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.
- [22] J. Poulton, "An embedded DRAM for CMOS ASICs," in *Proc. 17th Conf. on Advanced Research in VLSI*, pp. 288–302, IEEE CS, Sept. 1997.
- [23] R. K. Singh *et al.*, "BioSCAN: a network sharable computational resource for searching biosequence databases," *CABIOS*, vol. 12, no. 3, pp. 191–196, 1996.
- [24] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, pp. 195–197, 1981.
- [25] D. Speck, "The Mosaic fast 512K scalable CMOS dRAM," in *Proc. Conf. Adv. Research in VLSI* (C. Séquin, ed.), (Cambridge, MA), pp. 229–244, MIT Press, 1991.
- [26] T. A. Thanaraj and T. Flores, "Assessment of Smith-Waterman sequence search tools implemented in Biocellator, FDF, and MasPar," tech. rep., EBI, Cambridge, Feb. 1997. <http://industry.ebi.ac.uk/~thanaraj/seqassess/report.html>.
- [27] Time Logic Inc., "Decypher II product literature." Incline Village, NV, <http://www.timelogic.com>, 1996.