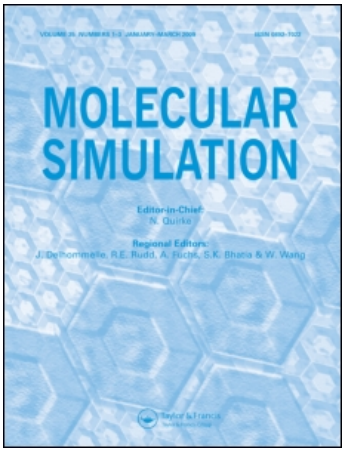

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

Concurrent Molecular Dynamics Simulation of ST2 Water on a Transputer Array

F. Brugè[a]; V. Martorana[a]; S. L. Fornili[b]
[a] Physics Department, University of Palermo Via Archirafi 36, Palermo, Italy [b] C.N.R.-I.A.I.F. Via Archirafi 36, Palermo, Italy

To cite this Article Brugè, F. , Martorana, V. and Fornili, S. L.(1988) 'Concurrent Molecular Dynamics Simulation of ST2 Water on a Transputer Array', Molecular Simulation, 1: 5, 309 — 320
To link to this Article: DOI: 10.1080/08927028808080952
URL: http://dx.doi.org/10.1080/08927028808080952

Molecular Simulation, 1988, Vol. 1, pp. 309–320
Photocopying permitted by license only
Reprints available directly from the publisher

CONCURRENT MOLECULAR DYNAMICS SIMULATION OF ST2 WATER ON A TRANSPUTER ARRAY

F. BRUGÈ,* V. MARTORANA* and S.L. FORNILI*+

**Physics Department, University of Palermo and +C.N.R.-I.A.I.F. Via Archirafi 36, I-90123 Palermo, Italy*

(*Received November 1987, in final form January 1988*)

A concurrent implementation of a Molecular Dynamics program for ST2 water molecules is presented, which exploits the great potentialities of the Transputer arrays for statistical mechanical calculations. High load-balance efficiency is obtained using a new task decomposition algorithm which evenly distributes particles and interaction calculations among the processors. This approach can also help to solve efficiently the more general problem of task distribution in parallel computing of symmetric pairwise system properties.

KEY WORDS: Transputer, MIMD, multiprocessor system, processor array, concurrent water simulation, parallelism.

INTRODUCTION

Simulation is one of the most powerful, widespread and expanding computer based activities in both science and engineering. However, it is in general highly demanding in terms of CPU-time, the more becoming so, the more realistic are the simulated systems. Thus, while it is already rather expensive to perform e.g. 100-ps long molecular dynamics simulations for systems consisting of a few hundred water molecules, it is still prohibitively costly to simulate realistically the interaction of water with biomolecules like proteins, which would require simulation times longer by several orders of magnitude [1]. Then, there is an urge to exploit new parallel programming techniques and computers which would appear to be more effective than traditional approaches [2–5].

High program concurrency is achievable with the so called M.I.M.D. (Multiple Instructions Multiple Data) architectures of multiprocessor computers [2–4]. The Transputer (T) family [6] includes VLSI electronic components which are particularly suited to implement M.I.M.D. systems. T-based arrays are efficiently programmable in Occam, a language that fully exploits T features [7].

The potential of T-based multiprocessor computers for statistical simulation calculations has already been shown [8, 9]. In particular, different parallelism schemes were adopted [9] to implement concurrent MD programs for Lennard–Jones (LJ) particles to assay their effectiveness as compared with that of a sequential version of the same program running on a VAX-11/750 computer. Indeed, it is difficult to design a reliable test for a configurable multiprocessor computer system, whose performance can be markedly affected both by the algorithm structure and the array complexity

and topology. We thought it more useful to test a simple T-based parallel computer system by running on it real simulation programs rather than benchmarks.

A new and efficient method, that will be outlined in the next section, was also designed and implemented [9] to evenly distribute particles and LJ interaction calculations among the processors. The resulting load-balance efficiency is equal to one for particular values of the particle to processor number ratio. The same method has been adopted to write the concurrent molecular dynamics program for ST2 water described in the present paper. This program, which is considerably more complex than the analogous program for LJ particles, will help us to extend our statistical simulation activity [10].

The general aim of our work in parallel computing field is to gain know-how in programming techniques and in hardware integration of computing systems that will certainly have an increasing importance in the near future. Indeed, as the price of VLSI components like Transputers drops, powerful systems based on large numbers of processors will be available at a reasonable price. The aim of the present paper is to provide helpful information on the use of T-based systems for statistical simulation calculations.

COMPUTATION DETAILS

The parallel molecular dynamics for 216 water molecules was written in Occam2, debugged and run using the INMOS TDS700 Development System [11] for IBM-PC/AT/XT with hard disk and a 640-kbyte memory. It consists of a B004 board with one 32-bit 15-MHz T414 Transputer, 2-Mbyte memory and interface circuitry, and a software package including folding editor, compiler and configurer, which helps to load code onto the multiprocessor array. A major advantage of this development system is that it makes possible to emulate on the single-T B004 board concurrent processes which will actually run on a number of Transputers of the target multiprocessor computer. In our case, The MD program was developed on the B004 board and then loaded onto a two-board system consisting of the same B004 and one B003

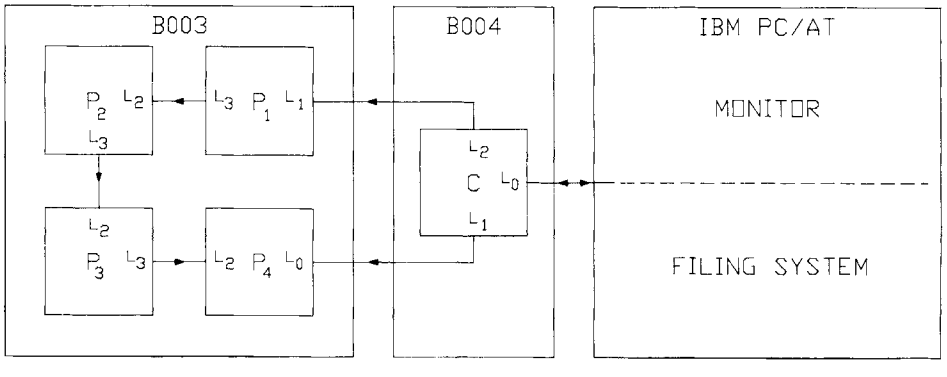

Figure 1 Block Schematics of the multi-Transputer system consisting an IBM-PC/AT equipped with an INMOS add-on B004 board connected to an INMOS B003 four-Transputer board. The B004 Processor C, which is connected only to the PC during the program development phase, is ring-connected to the B003 Transputers, P's, through the indicated links, L's, and communicates with the PC monitor and filing system. All processors are 15-MHz T414 Transputers.

board, whose four 15-MHz T414 Transputers were connected together and with the Transputer on the B004 board according to a simple ring topology (Figure 1). Duties of the B004 Transputer (controller) are to initialize the workers, to provide communication with the PC terminal and filing system and to route communication along the ring-connected processors.

In a typical molecular dynamics program the most time-consuming task concerns the calculation of the N(N-1)/2 pair interactions V(i, j) among the N particles, and the N forces F(i), the remaining time being spent to update the particular coordinates R(i) according to some integration method, to evaluate some relevent physical and statistical functions, and to output the "history" to external mass storage devices. In the present case Verlet method was used [12] together with the "SHAKE" algorithm [13] and reaction field evaluation [14], both contributing to make the molecular dynamics for ST2 water much more demanding than those for LJ particles in terms of double-precision floating-point computations. A major concern, while organizing a concurrent molecular dynamics program, is how to distribute the interaction calculations among the P processors, in order to reach a good load balance. However, if only this task is parallelized, the efficiency of a multiprocessor system quickly decreases with increasing the number of processors, P. Indeed, while the time spent to compute forces decreases as P increases, other tasks, like the integration of the dynamic equations, which must be executed after force evaluations might cause bottlenecks if it is run on one or few processors. Therefore, depending on the system size, task distribution and topology of the interprocessor connecting links should be carefully matched.

Aiming to implement algorithms that are efficient for computers based on a small number of processors as well as for larger systems, with suitable changes of the array topology, we have previously [9] examined different approaches to parallelism, which incorporate features of both "processor farm" and "algorithmic distribution" schemes [2]. Since a ring topology of the T array was chosen in all cases, the computational power of the parallel molecular dynamics programs for LJ systems was compared using the following definitions of load-balance efficiency E and speed-up factor S:

$$E = I_{av}/I_{mx} = ((I_{to})/P)/I_{mx} \quad \text{and} \quad S = P^*E,$$

where $I_{to} = N(N\text{-}1)/2$ is the total number of pairwise interactions, and I_{av} and I_{mx} represent, respectively, the average and the maximum number of interactions computed by processors. This choice disregards the communication overhead and the random fluctuations of the number of interaction calculations performed by each processor due to the distance cut-off value used to evaluate the particle-particle interactions. This choice is then reasonable until the computation to communicaton time ratio is large. This condition is met in our case both for LJ program and even more for ST2 water program, as it will be shown below.

In the first algorithm considered [9] (designated with "PM" for particle mapping [15]), N/P particles are attributed to each processor. The interactions are first calculated among the particles belonging to each T and then among particles mapped onto different T's, which send through the interprocessor links the coordinates of their own particles. Thus, the number of interactions per T is given by 0.5*(N/P)*(N/P − 1) + (N/P)*(N − N/P), many calculations being duplicated over a pair of processors, since the interaction symmetry (i.e. V(i,j) = V(j,i)) is not fully exploited. The resulting load-balance efficiency decreases, rapidly approaching 0.5 as P is increased.

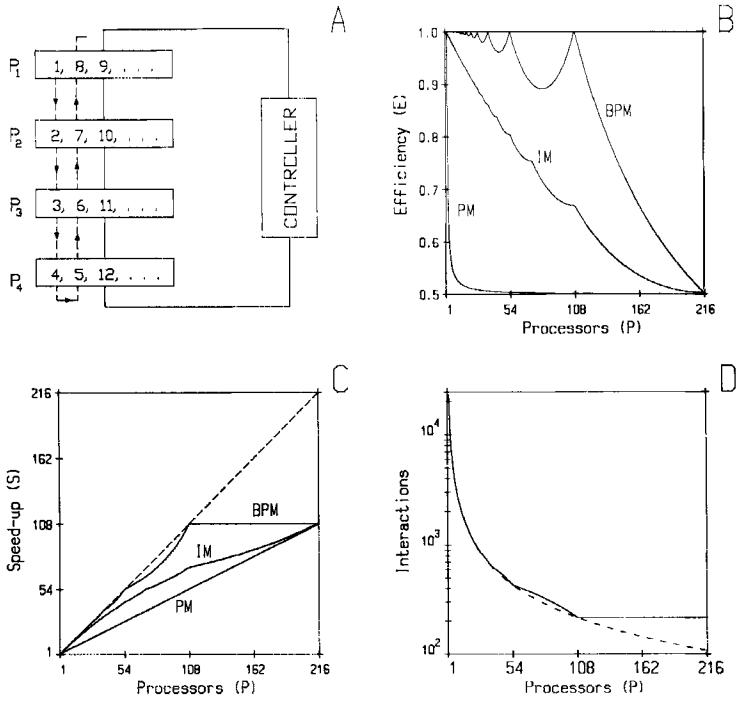

Figure 2 **(A)** Boustrophedonic Particle Mapping (BPM) onto Transputers of the B003 board; **(B)** load-balance efficiency E and **(C)** speed-up factor S for PM, IM and BPM schemes; **(D)** plots of the maximum (full line) and average (dashed line) interaction calculations per Transputer as a function of the number of processors P for BPM scheme. All the calculations refer to 216 particles.

Higher efficiency was obtained [9] by coding a molecular dynamics program according to the uniform subtask allocation strategy (here called "IM" for interaction mapping) designed by the Clementi's group for their loosely coupled array processors (lCAPs) [16]. In our implementation each worker processor fills a roughly equal part of the interaction matrix without any duplication of calculations, and the particle coordinates are updated by the master processor and rerouted to workers. A good load balance can be obtained if $P/N < 0.1$. However, if the coordinate updating is performed only by the controller, a bottleneck occurs at this task.

A new and more attractive approach was devised by modifying the "PM" algorithm into what we call "boustrophedonic [17] particle mapping" (BPM) scheme, where the N particles are distributed by scanning the P processors bi-directionally, as indicated in Figure 2A. The following recursive expressions are used to compute the m_i^k indices of the particles attributed to the processor k:

$$m_1^k = k; \quad m_i^k = m_{i-1}^k + \begin{cases} 2(P-k)+1 & (i \text{ even}) \\ 2k-1 & (i \text{ odd}) \end{cases} \quad \begin{pmatrix} k = 1,2,\ldots,P \\ i = 1,2,\ldots,M \end{pmatrix}, \quad (1)$$

where M equals N/P for all processors, if the total number of particles N is a multiple of the processor number P, as supposed in what follows unless otherwise stated. In

order to avoid the duplication of the interaction calculations, the additional rule is imposed that each processor calculates both "internal" (i.e. among its own particles) and "external" (i.e. among its own particles and those belonging to other processors) interactions V(i,j) only if $i < j$. In Figure 2B and C, the resulting efficiency E and speed-up factor S, respectively, are plotted *vs.* the processor number P for the 216-particle case.

For comparison, plots relative to PM and IM schemes are also reported. As one can see, the computational efficiency E is higher for BPM algorithm than for both IM and PM. Further, $E = 1$ when $N = 2kP$, where k is a positive integer. Indeed, at these points in Figure 2D, the curve (full line) representing the average number of interaction calculations per processor I_{av} coincides with the curve (dashed line) representing the maximum number of interaction calculations per processor I_{mx}.

To further clarify how BPM algorithm works and improves the PM scheme, in Figure 3 we report histograms representing the number of interactions calculated by each processor relatively to each particle attributed to it. Simple examples are considered where eight particles are distributed among two, three, four and five processors. In particular, one can observe that i) the BPM scheme helps to evenly distribute among the processors the particles with low index, which involve the

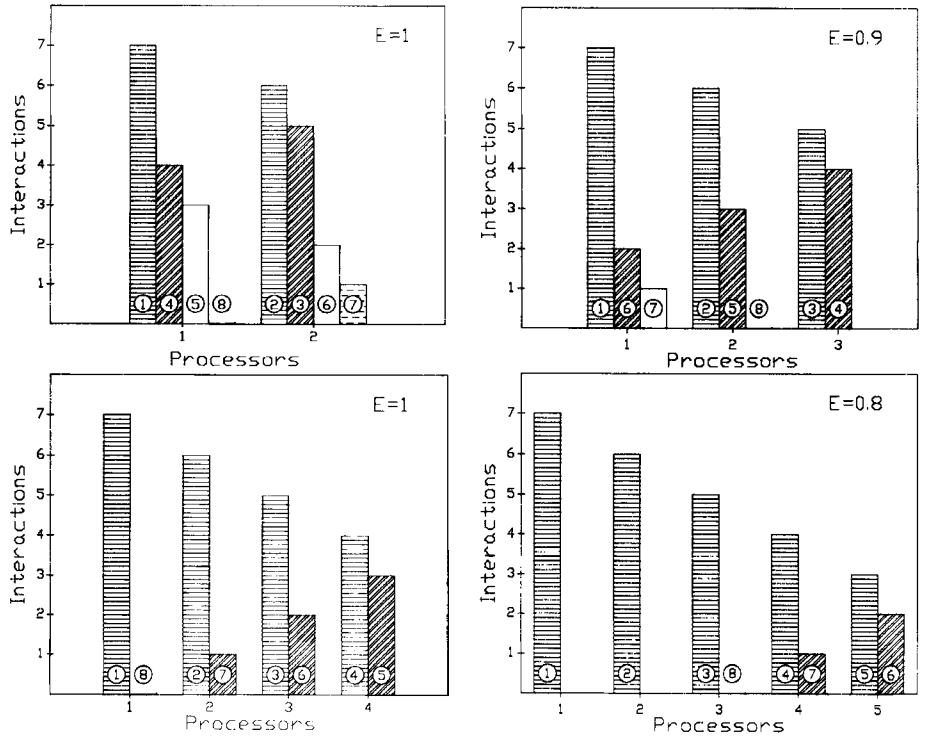

Figure 3 Examples of particle and interaction calculation distribution according to the BPM scheme for a number of particles $N = 8$ and processor number $P = 2, 3, 4$ and 5. Circled numbers represent particle indices and histograms the number of interaction calculations per processor relatively to the indicated particle.

heaviest computation; **ii)** complete load balance is achieved in two- and four-processor cases, where the average and the maximum interaction calculation numbers are the same for all processors.

A schematic fold structure and a flow-chart of the iterative part of the ST2-water molecular dynamics program that includes the above features are shown in Appendix A and Figure 4, respectively. Two parallel processes with different priority are mapped onto the controller, the low-priority process being dedicated to exchange data with the PC monitor and filing system. This function of the controller is thus decoupled from its mainly routing activity within the ring connected Transputer array. On the other hand, one and the same process is replicated on each worker. Occam channel communication among concurrent processes are represented in Figure 4 by dashed lines.

During the initialization phase (not shown in Figure 4) the controller receives from the PC filing system two sets of coordinates R(i) to initialize particle position and speed. Coordinates and indices, computed according to (1), are passed unidirectionally by the controller to workers *via* the closed communication pipe formed by the Transputer links shown in Figure 1. The transfer of small rather than large data blocks was found more effective in spite of a larger cumulative setup time of the links, since it favors an early parallel communication activity along the pipe. Also, the energy control parameter, which is needed by the workers at a later time, when they update the particle coordinates, is communicated early in the iterative cycle in order to minimize the number of syncronizing events among the parallel processes.

As soon as each processor has received the full complement of its particle coordinates, it starts calculating the pairwise "internal" interactions V(i,j) with $j < i$, and the force contributions $\mathbf{f}(i,j) = -\nabla V(i,j)$, always adding to the force matrix for both

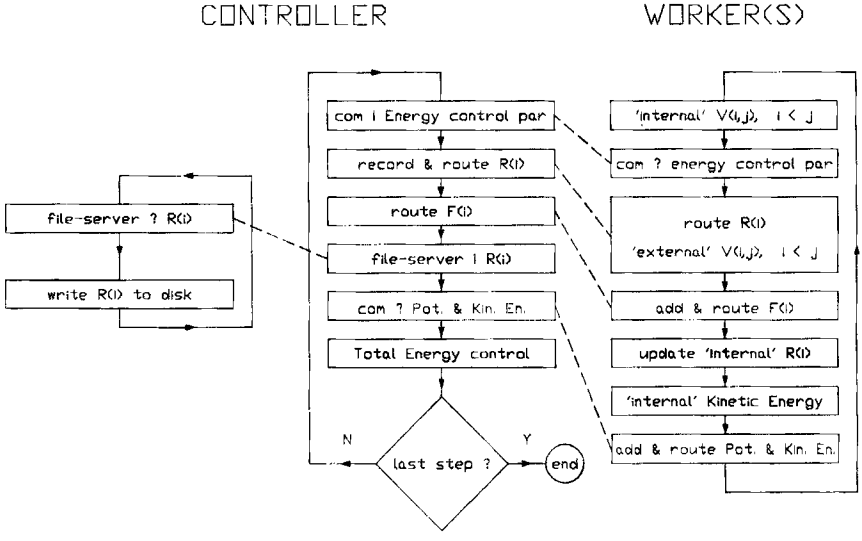

Figure 4 Schematic flow-chart of the iterative part of the MD program according to the BPM approach. Two concurrent processes are mapped onto the controller, one of which, at a lower priority, communicates with the PC monitor and filing system. One and the same process is mapped onto each worker. Dashed lines represent communications among concurrent processes. The symbols ! and ? represent output and input processes, respectively.

i and j elements (i.e. $\mathbf{F}(i) \leftarrow \mathbf{F}(i) + \mathbf{f}(i,j)$ and $\mathbf{F}(j) \leftarrow \mathbf{F}(j) - \mathbf{f}(i,j)$). Then, each worker routes systolically the coordinates of its own particles to the next processor and successively computes for every received particle with $j > i$ the "external" interactions V(i,j) and the $\mathbf{f}(i,j)$ contributions to $\mathbf{F}(i)$ as well as to $\mathbf{F}(j)$. The force matrix is completed during the next systolic transfer cycle, when each processor, while routing to next processor the partial force matrix, adds to its $\mathbf{F}(i)$ values the corresponding values integrated and passed on by the preceding processors in the ring. As soon as each worker has got his complete copy of force matrix, it starts updating the coordinates $\mathbf{R}(i)$ of its own particles and computes the "internal" kinetic energy. Then, kinetic and potential energy values are added up as they are routed along the pipeline towards the controller, which uses the resulting values to calculate the energy control parameter. Meanwhile, the workers start computing the "internal" interactions for the next integration step.

In Table I we report the measured execution time values per molecular dynamics time-step obtained using the above mentioned ring-connected T-array and the BPM concurrent algorithm for both LJ and water particles. For comparison analogous results are shown which refer to sequential implementations of the same MD programs running on a VAX-11/750 computer equipped with 5-Mbytes central memory and Floating Point Accelerator, and on one of the Four Transputers of the B003 board. One can make the following observations: **i)** simulation is faster (*ca.* eleven times on Transputers and seven times on VAX-11/750 with FPA) for LJ particles than for ST2-water. This should be expected, since the ST2 model involves LJ as well as electrostatic interactions among point charges (four per molecule). Increased computation is required in the ST2 case by the SHAKE routine [13] and by the reaction field evaluation [14]. To explain the difference in simulation speed between VAX and T414-based systems, it is worth noting that in the latter both single and double precision floating point calculations are carried out by software. A markedly higher computational performance is expected for systems based on T800 Transputers rather than T414. Indeed, T800 features a 64-bit floating point unit working in parallel to the 32-bit CPU. However, due to the larger communication to calculation time ratio, the ring connection topology will, in T800 case, presumably cause a loss in the overall efficiency of the system as the processor number is increased. So, different interprocessor connection schemes will have to be explored. **ii)** The measured values of the overall efficiency E_m (i.e. the ratio between the execution times corresponding to 1-T and 4-T systems, reported in Table 1) and speed-up factor $S_m = P^*E_m$ show that for a four-T414 system the deviations from linearity are *ca.* 3% and 4% in simulating LJ particles and ST2-water molecules, respectively.

Table I Execution time T (in s) per MD time-step (2^*10^{-15} s), efficiency E_m, speed-up factor S_m and memory occupation (in kbytes) for 216 LJ particles [9] and ST2-water (WT) programs running on sequential and parallel (BPM algorithm) computer systems.

	Type	*Computer*	*T*	E_m	S_m	*memory*
	seq.	VAX-11/750[a]	7.0			
LJ	seq.	1 Transp[b]	33			47
	par.	4 Transp[b]	8.6	0.96	3.8	35/Trans.
	seq.	VAX-11/750[a]	47			
WT	seq.	1 Transp[b]	369			183
	par.	4 Transp[b]	95	0.97	3.9	92/Trans.

[a]Equipped with 5-Mbyte memory and Floating Point Accelerator.
[b]The Transputer on the B004 board is used mainly for communication exchange.

CONCLUSIONS

From our experimentation on Transputer based computer systems — to design, implement and run concurrent Occam versions of MD programs for LJ particles [9] and ST2 water molecules — we can conclude that **i)** powerful Transputer-based systems can be easily assembled with commercially available boards, whose floating point computational power will dramatically increase as soon as T800 are easily available; **ii)** rewriting in Occam existing programs is not too painful; FORTRAN compilers are also available; **iii)** a novel approach has been devised to evenly distribute particles and interactions among the processors, which results in a calculation-load balance efficiency equal to one for particular particle to processor number ratios. This procedure together with a careful choice of array topologies suitable to reduce the communication overhead, can help to approach an efficient use of M.I.M.D. systems consisting of larger numbers of Transputers. Further, the application of this method can be extended to more general cases of task distribution in parallel computing of symmetric pairwise properties of the simulated system.

In conclcusion, we think that T-based arrays show great potential for statistical simulation calculations. The outlook has further improved since boards started to appear which interface T-based systems with high capacity hard disks. Indeed, this helps to solve a problem of major concern for statistical simulations — the heavy mass storage requirement.

Acknowledgements

A fellowship from C.R.R.N.S.M. to one of us (F.B.) and general indirect support from Italian MPI-60% and C.R.R.N.S.M. are acknowledged. We thank I.A.I.F.-C.N.R. for providing technical facilities, and Mr. M. Lapis and Mr. G. La Gattuta for skilful technical assistance. We also thank Prof. M.U. Palma and Prof. M.B. Palma Vittorelli for useful discussions and Prof. A. Geiger for the FORTRAN VAX-version of the ST2-water molecular dynamics program.

APPENDIX

In what follows we show schematically the structure of the molecular dynamics program taking advantage of the fold notation that characterizes the INMOS TDS editor [11]. The most general view of the Occam2 (beta release B) program, which includes a "ring" EXE, a "proring" Program, and I/O and monitor files, is:

```
{{{F RING TOP
{{{
...   EXE ring
...   PROGRAM proring
{{{   i/o files
...F  inp.tsr
...F  out.tsr
...F  mon.tsr
}}}
}}}
}}}
}}}
```

The "ring" EXE, which runs on the controller, consists of two concurrent processes with different priority: a) the higher priority "master" which connects the controller

to the workers according to the ring topology of Figure 1; b) the lower priority "buffer" provides communication between controller and PC monitor and filing system.

```
{{{    EXE ring
{{{F   ring.tsr
PROC ring (CHAN keyboard, screen, [4]CHAN from, to)
  ... file read procs
  ... declarations
  ... master proc
  ... buffer proc
  PRI PAR
    master (to.buffer)
    buffer (to.buffer)
:
}}}
}}}
}}}
{{{    master proc
PROC master (CHAN to.buffer)
  ... external links
  ... definitions and declarations
  SEQ
    ... initialization
    WHILE n < num. cycles
      ... main loop
    ... input and write statistical functions
    ... close file
    write.full.string (screen, "*c*nEND OF RUN")
    keyboard? any
:
}}}
{{{    buffer proc
PROC buffer (CHAN to. buffer)
  ... variables
  SEQ
    listen : = TRUE
    WHILE listen
      SEQ
        to.buffer ? command
        IF
          command = open.the.file
            ... open the file
          command = corr.exc.write.coordinates
            ... correct, exchange and write coordinates
          command = write.statistics
            ... write statistical functions
          command = stop
            ... close file and stop
:
}}}
```

The "proring" PROGRAM includes the three separate compilable (SC) procedures "first", "last" and "ith", which are mapped onto the first, the last and the middle workers of the ring, respectively (see Figure 1). These procedures have the main body consisting of the "worker" procedure, the only difference being the communication channel usage.

```
{{{  PROGRAM proring
{{{F proring. tsr
...   SC first
...   SC last
...   SC ith
...   Channel declarations
...   link addresses
PLACED PAR
  PROCESSOR 0 T4
    place link[0] AT link3out:
    first (link[0])
  PROCESSOR 1 T4
    PLACE link[0] AT link2in:
    PLACE link[1] AT link3out:
    ith (link[0], link[1], 1)
  PROCESSOR 2 T4
    PLACE link[1] AT link2in:
    PLACE link[1] AT link3out:
    ith (link[1], link[2], 2)
  PROCESSOR 3 T4
    Place link [2] AT link2in:
    last (link[2], 3)
}}}
{{{F  first.tsr
PROC first (CHAN out)
  ... link definition
  ... proc worker
  worker (in, out, 0)
:
}}}
{{{F  last.tsr
PROC last (CHAN in, VAL INT i)
  ... link definition
  ... proc worker
  worker (in, out, i)
:
}}}
{{{F  ith.tsr
PROC ith (CHAN in, out, VAL INT i)
  ... proc worker
  ... worker (in, out, i
:
}}}
```

```
{{{   proc worker
PROC worker (CHAN in, out, VAL INT i)
  {{{ main body
  ... declarations
  ... SC force. tsr
  SEQ
    ... initialization
    alfa : = one
    cont : = TRUE
    ncont : = 0
    WHILE cont
      SEQ
        ncont : = ncont + 1
        ... reset forces, potentials, etc.
        ... calc. "internal" interactions
        ... input alfa and flag controls
        ... route coord. and calc. "external" interactions
        ... route and add force matrix
        ... calc new coordinates
        ... calc and send kin. and pot. energy
        ... exchange old with new coordinates
      ... output statistical functions
:
}}}
```


References

[1] S.H. Koenig, "The dynamics of water-protein interactions", in *Water in Polymers*, S.P. Rowland, ed., *A.C.S. Symp. Ser. 127,* Am. Chem. Soc., Washington, D.C., 1980 pg. 157.

[2] C.R. Jesshope, "Computational Physics and the need for parallelism" *Comp. Phys. Commun.*, **41**, 363 (1986)

[3] a) R. Asbury, S.G. Frison and T. Roth, "Concurrent computers ideal for inherently parallel problems", Comp. Design, 35 (Sep. 1985); J.V. Hornstein, "Parallel processing attacks real-time world", Mini-Micro Systems, 65 (Dec. 1986); c) N. Mokhoff, "Parallelism breeds a new class of supercomputers", Comp. Design 53 (Mar. 1986); d) J. Bond, "Parallel-processing concepts finally come together in real systems", Comp Design, 51 (Jun. 1987).

[4] D. Fincham, "Parallel computers and molecular simulation", *Mol. Simul.*, **1**, 1 (1987).

[5] A. Wallqvist and B.J. Berne, "Exploiting physical parallelism using supercomputers. Two examples from Chemical Physics", *IEEE Computer,* **20**, 9 (May 1987).

[6] *Transputer Reference Manual*, INMOS, Bristol, 1986.

[7] D. Pountain, *A Tutorial Introduction to Occam Programming*, INMOS, Bristol (1986).

[8] C.R.Askew, D.B.Carpenter, J.T. Chalker, A.J.G. Hey, D.A. Nicole and D.J. Pritchard, "Simulation of statistical mechanical systems on Transputer arrays", *Comp. Phys. Commun.* **42**, 21 (1986).

[9] V. Martorana, M. Migliore and S.L. Fornili, "Molecular Dynamics simulation of Lennard-Jones systems: parallel implementations on Transputer arrays", *Proceedings of the 7th Occam User Group and Intl. Workshop on Parallel Programming of Transputer based Machines,* Grenoble, 1987.

[10] a) S.L. Fornili, M. Migliore, D. Vercauteren and E. Clementi, "Monte Carlo simulation of water interaction with Gramicidin A transmembrane channel: hydrogen bond analysis", *J. Physique* **45**, C7-219 (1984); b) S.L. Fornili, M. Migliore and M.A. Palazzo, "Hydration of the Hydronium ion. Ab initio calculations and Monte Carlo simulation." *Chem. Phys. Lett.* **125**, 419 (1986); c) M. Migliore, S.L. Fornili, E. Spohr and K.Heinzinger, "Molecular Dynamics study of a KCl aqueous solution: dynamical results." *Z. Naturforsch.* **42a**, 227 (1987); d) R. Noto, M. Migliore, F. Sciortino and S.L. Fornili, "Solute-induced water structure: computer simulation on a model system", *Mol. Sim.*, **1**, 225 (1988).

[11] *IMS701B Transputer Development System*, INMOS, Bristol, 1986.
[12] L. Verlet, "Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules", *Phys. Rev.* **159**, 98 (1967).
[13] J.P. Ryckaert, G. Ciccotti and H.J.C. Berendson, "Numerical integration of the cartesian equations of motion of a system with constraints: Molecular Dynamics of n-alkanes", *J. Comput. Phys.* **23**, 327 (1977).
[14] O. Steinhauser, "Reaction field simulation of water", *Mol. Phys.* **45**, 335 (1982).
[15] G.C. Fox and S.W. Otto, "Algorithms for concurrent processes", *Physics Today*, **37**, 50 (May 1984).
[16] H.L. Nguyen, H. Khanmohammadbaigi, and E. Clementi, "A parallel Molecular Dynamics strategy", *J. Comput. Chem.* **6**, 634 (1985).
[17] Particles are attributed to the processors of the ring-connected multiprocessor system in successive bi-directional scans, each time turning direction as oxen do in plowing. So, this particle mapping is called boustrophedonic, from bous = ox and strephein = to turn. Webster's Third New International Dictionary, Encyclopaedia Britannica, Inc., G. & C. Merriam Co., London, 1966, p.261.