JOURNAL OF COMPUTATIONAL PHYSICS 75, 138–150 (1988)

The Implementation of Fluid Phase Monte Carlo on the DAP

D. J. ADAMS

Department of Chemistry, The University, Southampton S09 5NH, United Kingdom

Received August 12, 1986; revised March 13, 1987

A scheme by which a distributed array processor (DAP) may be used efficiently for the Monte Carlo simulation of condensed matter is described. The approach gives a certain flexibility to the size of system simulated. For a 64×64 DAP one simulates systems each of $2^n \times 64$ particles or molecules simultaneously, where n is an integer in the range 0 to 6. The method of coping which systems with a hardsphere potential using either constant volume or constant pressure ensembles is described. © 1988 Academic Press, Inc.

1. INTRODUCTION

The ICL distributed array processor (DAP) is a working example of a single-instruction-multiple-data computer [1]. It is arranged as a square lattice of single-bit processing elements (PEs), each able to access its own data area and the data of its four nearest neighbours. It can be programmed with its own high-level language, DAP FORTRAN, which has the necessary extensions to exploit the highly parallel architecture of the machine, together with a library of functions and subroutines [2]. The presently available machine has 64×64 PEs, and this is the size that will generally be referred to in this paper, though a 32×32 LSI DAP is undergoing user trials and a 128–128 VLSI machine is being developed, sponsored by the U.K. Ministry of Defence. The attraction of the DAP is its speed and high cost efficiency for many large-scale computations. The present 64×64 machine can maintain 15 million floating point operations per second [1].

The methods of Monte Carlo and molecular dynamics for the simulation of condensed matter at the molecular level are large-scale problems that consume very large amounts of computer time and are ready candidates for parallel processing. When the system to be simulated has a lattice structure with only near-neighbour interactions, which occurs in a wide variety of problems from the Ising lattice to the plastic phase of SF_6, for example, then the DAP can be used with high efficiency. This is done by devoting each PE to a lattice site and using the DAP in matrix mode. In molecular dynamics the force on each particle can be computed simultaneously. In Monte Carlo many lattice sites can be updated independently

0021-9991/88 $3.00

(because only near-neighbours interact) and therefore simultaneously. With sufficient ingenuity it is often possible to achieve 100% use of the PEs. Pawley and Thomas have shown how a variety of 2, 3, and 4-dimensional lattices can be mapped onto the square lattice of the DAP [3].

When a fluid system, or indeed a solid with long-range forces, is simulated by molecular dynamics then the DAP is used in a slightly different way. Each PE is used to compute the forces between a *pair* of particles, again using matrix mode. In the simplest case there would be 64 particles and the DAP computes all 64×64 pair interactions simultaneously. The supplied function SUMC is used to find the net force on each particle. SUMC takes a 64×64 matrix as argument and returns a vector value, each component of which is the sum of all the components in the row of the matrix argument [2]. Larger systems are readily accommodated by working with multiples of 64 particles; the forces are calculated by simultaneously computing the interactions of all the particles on one block of 64 with all the particles in a second block. This is the method used by Fincham *et al.* [4] in the simulation of CO_2/C_2H_6 mixtures on the DAP.

In molecular dynamics every particle is moved at each step and the force on every particle has to be known at each step. However, in Metropolis Monte Carlo only one particle is moved at each step and only the energy of that one particle needs to be computed. When the particle interacts with many others, or the system is a fluid and near neighbours are not trivial to locate, only one Monte Carlo step can be made at a time. The only immediately obvious way of achieving high "efficiency" is then to have a system of $64^2 = 4096$ particles and calculate the interaction of the one moved particle with all others. While this is feasible, there are few cases where one would choose to do such a thing using a conventional scalar processor, and the apparent highly efficient use of the DAP would be illusory for most systems.

It is the purpose of this paper to demonstrate how a greater flexibility for the system size may be achieved in Monte Carlo while maintaining a highly efficient use of the DAP. Section 2 describes the basic scheme: the DAP is used to simulate 64 independent systems, each of 64 particles, simultaneously. Section 3 describes an extension to this scheme so that systems of size $2^n \times 64$ particles may be efficiently simulated. The fourth section deals with the special case, frequently encountered in Monte Carlo, where there is a hard (i.e., infinite) term in the pair potential.

For example, with the scheme described in this paper the author is able to perform Monte Carlo calculations for the dipolar hard-sphere fluid, including the calculation of several distribution functions, at a rate of ~ 50 thousand steps per hour for each of eight systems of 512 dipoles. For a less complicated potential with a hard-sphere term, the restricted primitive model of electrolyte solutions using a simple isotropic approximation to the Ewald sum [5], a rate of ~ 200 thousand steps per system per hour is achieved.

Chapman and Quirke [6] have considered an alternative approach to Monte Carlo on a parallel processor in which all, or a large fraction, of particles are moved at each step. This interesting suggestion does seem practical for small systems.

However, it is most unlikely to compete with the present scheme for systems larger than 64 particles at most. Only in its most favourable case, a system of 32 particles on a 32×32 DAP is it a serious contender to the scheme described here.

2. The Basic Scheme

The DAP can be thought of as a square matrix. In the basic scheme each row of the matrix stores and processes a system of particles completely independently of those in other rows. With a 64×64 DAP there will therefore be 64 systems of 64 particles which will be simulated simultaneously. The user has complete freedom to set the density and temperature (or pressure and temperature with the NpT ensemble) of each individual system. It is even possible to vary the parameters of the potential between systems, though the form of the potential must be the same throughout. When 64 different state points are not required it is possible to set two or more systems to the same state point. This has the advantage that simple error estimates can be obtained because the results from different systems are truly independent, once properly equilibrated.

The DAP Support Unit (at Queen Mary College, London) provides random number generators for the DAP which produce at each call a 64^2 matrix of random numbers, evenly distributed between 0 and 1 and independent both of each other and previous random numbers in the sequence [7]. A call to one one of these would look like

$$\text{RAND}(\,,\,) = \text{GO5\_NAG\_REAL4}(0.0) \tag{1}$$

where RAND is a real matrix variable and GO5_NAG_REAL4 is a real matrix function with a dummy scalar argument. One call provides more than sufficient numbers for the entire Monte Carlo step. The construction RAND(,) refers to the entire range of the matrix RAND.

The first stage in a Monte Carlo step is to choose at random the particle to be

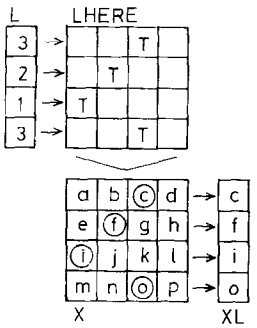

Figure 1

given a trial displacement. The first column of RAND can be used to choose a particle for each system:

$$L(\,) = \mathrm{FIX}(1. + \mathrm{RAND}(\,, 1) * 64.) \tag{2}$$

where L is an integer vector variable and FIX is a supplied function. The construction $L(\,)$ refers to the entire range of the vector L.

It is necessary to set up a logical mask to identify which particle on each row has been chosen; this is the logical matrix LHERE. Vectors of the chosen particle coordinates are then extracted from the matrices of all particle coordinates:

$$\begin{aligned} &\mathrm{LHERE}(\,,\,) = \mathrm{COL}(L) \\ &XL(\,) = X(\,, \mathrm{LHERE}) \\ &YL(\,) = Y(\,, \mathrm{LHERE}) \\ &ZL(\,) = Z(\,, \mathrm{LHERE}) \end{aligned} \tag{3}$$

COL is a supplied function [2]. Figure (1) illustrates how (3) operates.

The trial displacement of the chosen particles and the correction for the periodic boundary conditions is done by using the DAP in vector mode. This is ~ 4 times as fast as matrix mode [1], but only 64 calculations are made simultaneously instead of 64^2. A vector of trial positions is produced by

$$TXL(\,) = XL + \mathrm{DMAX} * (\mathrm{RAND}(\,, 2) - 0.5) \tag{4}$$

Note that now the second column of RAND is used, the third and fourth will be used for the Y- and Z-coordinates. DMAX is a real vector variable, each element corresponding to the maximum allowed displacement for that system. As 64 values in DMAX have to be set, it is advisable to do this by automatic adjustment during the equilibration part of the calculation to give the usual $\sim 50\%$ acceptance rate for each system. For simple cubic periodic boundary conditions (PBC) with a unit cell length over the range -0.5 to $+0.5$ the correction for the PBC takes two lines for each dimension:

$$\begin{aligned} &TXL(TXL\ .LT.\ -0.5) = TXL + 1.0 \\ &TXL(TXL\ .GT.\ 0.5) = TXL - 1.0. \end{aligned} \tag{5}$$

Additional code is required if truncated octahedral PBC [8] are used, with two additional logical vector variables POS and CORNER:

$$\begin{aligned} &\mathrm{CORNER}(\,) = ABS(TXL) + ABS(TYL) + ABS(TZL)\ .GT.\ 0.75 \\ &\mathrm{POS}(\,) = TXL\ .GT.\ 0.0 \\ &TXL(\mathrm{CORNER}\ .\mathrm{AND.}\ \mathrm{POS}) = TXL - 0.5 \\ &TXL(\mathrm{CORNER}\ .\mathrm{AND.}\ .\mathrm{NOT.}\ \mathrm{POS}) = TXL + 0.5 \end{aligned} \tag{6}$$

and similarly for *TYL* and *TZL*. The assignments in (5) and (6) are only made for those elements whose corresponding element in the left-hand-side argument of *TXL* is true [2]. If the cell coordinates go from -1 to $+1$ then the correction for PBC can be speeded up by using short-length integer arithmetic [9]. Rhombic dodecahedral PBC would be similarly programmed.

The matrix of x-distances between the trial particles and all the others is constructed so

$$DX(\ ,\ ) = X - \text{MATC}(TXL) \tag{7}$$

and similarly for the real matrix variables *DY* and *DZ*. The supplied function MATC returns a matrix in which every element in is *i*th row is equal to the *i*th component of the function's vector argument [2]. The matrices of pair separations can be corrected for the nearest image distance in the manner of (5) and (6).

However, Fincham [9] has described a superior method relying on a feaure of the ICL DAP that when a single-bit logical matrix is declared EQUIVALENT to a real matrix variable then each element in the logical matrix is mapped onto the sign bit of the corresponding real element, with true for positive and false for negative.

Having constructed the matrices of all modified interparticle distances, the potential energy between every pair of particles can then be calculated simultaneously using matrix mode. The result is a matrix of energy terms. For the Lennard–Jones potential the code would look something like

$$\begin{aligned} R2(\ ,\ ) &= DX{**}2 + DY{**}2 + DZ{**}2 \\ R2(\text{LHERE}) &= 1.0 \\ \text{TERM}(\ ,\ ) &= (\text{MATC}(\text{SIGMA}){**}2/R2){**}3 \\ U(\ ,\ ) &= 4.0 * \text{MATC}(\text{EPSILON}) * \text{TERM} * (\text{TERM} - 1.0) \\ U(\text{LHERE}) &= 0.0 \end{aligned} \tag{8}$$

where TERM is a real matrix variable and EPSILON and SIGMA are real vector variables, each component corresponding to one of the 64 systems. The line

$$R2(\text{LHERE}) = 1.0 \tag{9}$$

prevents division by zero on the following line and

$$U(\text{LHERE}) = 0.0 \tag{10}$$

removes the resulting terms for the interaction of each trial particle with itself. The above code (8) is illustrative only and would be optimised in practice. The important point is that all but 64 of the 64^2 PEs are doing useful calculations. Having, in the usual way, computed the energy of the chosen particles in both their old and trial positions the net changes in energy can then be calculated using the SUMC function already described:

$$DU(\ ) = \text{SUMC}(U\_\text{TRIAL\_POSITION} - U\_\text{OLD\_POSITION}) \tag{11}$$

A logical vector variable ACCEPT is given by

$$\text{ACCEPT}(\,) = \text{EXP}(-DU/KT)\,.GT.\,\text{RAND}(,5) \tag{12}$$

where the real vector variable KT contains the Boltzmann's constant times temperature for each system. ACCEPT is used as a mask to update only those systems which pass the Boltzmann weighting test, modifying the coordinates

$$X(\text{LHERE}\,.\text{AND}.\,\text{ACCEPT}) = \text{MATC}(TXL) \tag{13}$$

etc., and updating the total energy

$$\text{UTOTAL(ACCEPT)} = \text{UTOTAL} + DU, \tag{14}$$

for example. The various additions are made to the (vector) running sums and Monte Carlo step is complete.

3. A More Flexible Scheme

Sixty-four particles is just acceptable as sufficient for serious work, and the use of truncated octahedral PBC makes low numbers of particles more acceptable [4, 9]. However, calculations with 64 systems of only 64 particles are really suitable only for preliminary work and larger systems are always desirable and often necessary. A simple way of getting this is to increase the total number of particles so that each PE is used for two or more particles. This is the natural approach for molecular fluids: each PE is used for one molecule of (perhaps) several particles. However, 64 independent systems, each of (say) 512 particles, is a formidably large undertaking, particularly if one does not *want* 64 independent systems. A better scheme is to keep the total number of particles (or molecules) at 64^2 and have, for example, only 8 systems of 512 particles. This approach is described in this section. The extra overheads incurred are not time consuming and the simulation of 8 systems of 512 particles is only marginally slower than 64 systems of 64 particles.

The basic idea is very simple. Instead of one row of the matrix per system, each system occupies two or more sequential rows, all systems being the same size. If all rows of PEs are to be used this restricts us to $64/2^n$ systems of $2^n \times 64$ particles, $0 < n < 6$. The matrix-mode calculation of the interaction energies between the chosen particles and the others is unchanged. However, the setting up of the chosen particles and the summing up of the pair energies and, indeed, all the vector-mode parts become more involved.

At the start of each step choosing particles and making trial displacements is carried out exactly as for 64 systems. This means that one particle per row has been selected and moved and only one particle per system is required. A single number is

chosen at random and is used to pick out the single trial particle to be used in each system by identifying the row it occupies:

$$NS = \text{FIX}(\text{Rand}(1, 6) * \text{FLOAT}(\text{NSIZE})) \tag{15}$$

where NSIZE is the number of rows in each system. Think of the DAP matrix split into horizontal bands each of NSIZE rows. The rows of each system are numbered from the bottom up, with the bottom row or *base row* numbered zero. A preset logical vector, BASE, and a preset logical matrix, BASE_ROW, point to the base rows; that is, they take the value TRUE only on base rows. The logical matrix LINE is set TRUE for the PEs of the chosen row of each system by

$$\text{LINE}(\ ,\ ) = \text{SHNP}(\text{BASE\_ROW}, NS) \tag{16}$$

where SHNP is a supplied function, the name being an anacronym for *shift north planar* [2, 3]. The logical matrix LHERE is revised so as to mask only one chosen particle per system by

$$\text{LHERE}(\ ,\ ) = \text{LHERE .AND. LINE} \tag{17}$$

The vectors *XL*, *TXL*, etc., contain the coordinates of a different particle in each row, but now have to contain the coordinates of the one chosen particle in each row of a system. This is set up for all the systems simultaneously. First, the coordinates of the chosen rows are shifted down to the base rows, for example,

$$XL(\ ) = \text{SHRP}(XL, NS) \tag{18}$$

where the supplied function SHRP (*shift right planar*) moves every element in vector *XL* down by *NS* places, with zeros inserted at the top. Second, the values in the base positions are copied into the rows above. Logical vector variable *B* and real vector variable *A* are used as intermediaries:

$$\begin{aligned} &NBB = \text{NSIZE} - 1 \\ &A(\ ) = XL \\ &B(\ ) = \text{BASE} \\ &\text{DO } 100\ I = 1, NBB \\ &B(\ ) = B(+) \\ &A(\ ) = A(+) \\ &100\ XL(B) = A \end{aligned} \tag{19}$$

The line

$$B(\ ) = B(+) \tag{20}$$

in (19) shifts the contents of all the elements in B up the vector by one, and is equivalent to, but faster than, using SHLP [8, 1]. Figure (2) is an attempt to illustrate the operation of (19); in practice one would not use NSIZE = 3.

Similar code is used to find the change in energy of each system. After the call of SUMC the subtotals of each row are summed into the base position of each system:

$$\begin{aligned} &\text{SUM}(\ ) = DU \\ &\text{DO } 100\ I = 1, NBB \\ &\quad DU(\ ) = DU(-) \\ &100\ \text{SUM}(\ ) = \text{SUM} + DU \\ &\quad DU(\ ) = 0.0 \\ &\quad DU(\text{BASE}) = \text{SUM} \end{aligned} \tag{21}$$

Only base positions are used to store the total energy, etc., of each system. Code similar to (19) and (21) is used to update the coordinates in systems where the trial move is accepted and generally to compute and to update the properties of each system.

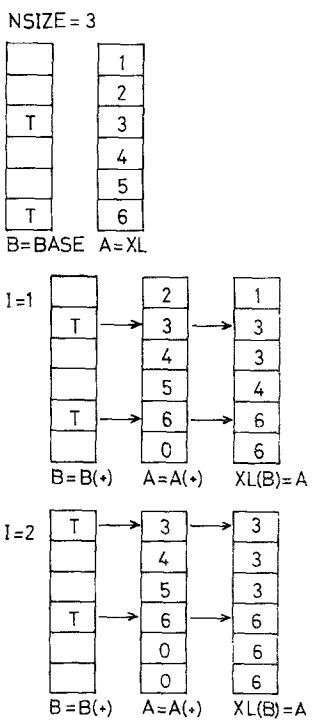

FIGURE 2

The whole scheme seems elaborate and time consuming. In practice this is not so. A few subroutines contain all the variations on (19) and (21) required and, because this sort of data manipulation on the DAP is very much quicker than numerical calculation, the time overheads are small. Note that the code for calculating the pair potential terms in matrix mode is unaltered from that in the basic scheme of one system per row.

4. Hard-Sphere Potential

The combination of a continuous potential with a discontinuity, such as the hard-sphere potential,

$$\phi_{\mathrm{HS}}(r) = \begin{cases} \infty, & r < \sigma \\ 0, & r > \sigma, \end{cases} \tag{22}$$

presents a particular difficulty for molecular dynamics simulation, and such potentials are very likely to be tackled by Monte Carlo. Using the canonical ensemble, for which the system volume remains constant, the presence of a hard-sphere term requires only a minor addition to a DAP Monte Carlo program. A logical matrix variable OVERLAP is introduced,

$$\mathrm{OVERLAP}(\ ,\ ) = R\,.LT.\ \mathrm{MATC(SIGMA)}\,.\mathrm{AND}..\mathrm{NOT}.\ \mathrm{LHERE} \tag{23}$$

where real matrix variable R contains all pair separations. The logical equivalent of SUMC in this context is the supplied function ORCOLS, and the logical vector OVER is set TRUE for any row with an overlap by

$$\mathrm{OVER} = \mathrm{ORCOLS(OVERLAP)}. \tag{24}$$

A logical-OR into the base positions is made by code similar to (19) and then (12) becomes

$$\mathrm{ACCEPT}(\ ) = \mathrm{BASE}\,.\mathrm{AND}.\ \mathrm{EXP}(-DU/KT)\,.GT.\ \mathrm{RAND}(,5)$$
$$.\mathrm{AND}..\mathrm{NOT}.\ \mathrm{OVER} \tag{25}$$

The constant NpT ensemble, introduced into Monte Carlo by Wood [10], is a valuable technique as it avoids the necessity of extrapolating the radial distribution function, $g(r)$, to contact in order to find the pressure. Radial distribution functions can be found efficiently on the DAP [11] but the extrapolation to contact inevitably limits the precision, particularly at high pressure when $g(r)$ is a steeply varying function. In fact the NpT ensemble has seen very little use for this purpose, possibly due to the more involved programming required. The method for a normal, serial processor is therefore outlined first and then the implementation on the DAP is described.

In addition to choosing a particle and moving it by a small random shift, a small random change is made to the linear dimensions of the periodic cell. It is necessary to test whether, as a result of this change in cell size, two or more particles overlap. To do this efficiently the program keeps a record of the two particles closest together, i and j, and their separation $r_{\min}$.

At each step, in addition to checking for overlap of the trial particle with the others, some extra checks may be necessary. If the trial particle is not i or j then the trial move is rejected if the volume has been reduced so that $r_{\min}$ is now too small. If the trial particle **is** i or j and the volume has been reduced sufficiently that $r_{\min}$ would have been too small it is necessary to check **all** pair separations in case there is another pair of particles too close together. If the trial move is accepted and the trial particle was i or j it is necessary to go through all pairs of particles anyway to find the new i, j, and $r_{\min}$. Otherwise it is only necessary to consider the revision of i, j, and $r_{\min}$ on the basis of the new distances between the trial particle and the others. The time required for the search over all pairs increases as N^2. However, the probability that the trial particle is i or j goes as N^{-1} and therefore the extra time required for these checks increases only linearly with the number of particles, N. There is an optimum order for the various stages in the Monte Carlo step. For example, the trial particle and trial volume change are chosen first and the old configuration is retained, skipping over the rest of the checks and changes, if $r_{\min}$ is now too small and the trial particle is not i or j.

A major programming difference with the DAP is that as several systems are being simulated simultaneously it is not possible to take short cuts: every stage of the checking procedure has to be gone through regardless of whether overlap has already been found. However, the search over all pairs in a system is only performed as necessary. The search over all pairs is done one system at a time. The rows of a system are taken in pairs, including the "pair" of a row with itself, and 64^2 pairs of particle separations computed at a time. A supplied function, MINP, finds the position of the smallest element in its matrix argument.

A logical matrix variable, MINHERE, is constructed in which elements are TRUE at the locations of the i and j of each system, and a real vector variable holds the $r_{\min}$.

When there is just one row per system it is easy to pick out the minimum distances between the trial positions of the chosen particles and the others for every system simultaneously. The logical matrix variable MINLOC is set to TRUE for the position of the minimum in each row by

$$\text{MINLOC}(\,,\,) = \text{COL}(\text{COLN}(XO5\_\text{EMINPC}(R))) \tag{26}$$

where $XO5$_EMINPC is a logical matrix function is the DAPSU library [12]. It returns the location of the minimum on each row. The inelegant construction with COL and COLN [13] is required to pick out the first location when $XO5$_EMINPC finds the same minimum value in two or more components on a row. (When MINP is used to find the minimum of the entire matrix then the sup-

plied function FRST can be used for the same purppose). The values of the minima are put into a vector variable by

$$\text{RMIN\_OF\_}L(\,) = R(,\ \text{MINLOC}) \tag{27}$$

The information on the minimum separations in the trial configurations is set up by

$$\begin{aligned}
&I\text{_OR_}J(\,) = \text{ORCOLS(LHERE .AND. MIN_HERE)}\\
&\text{UPDATE}(\,) = \text{RMIN_OF_}L\ .LT.\ \text{RMIN .AND.. NOT. (OVER .OR. }I\text{_OR_}J)\\
&\text{TRIAL_MIN_HERE}(\ ,\ ) = \text{MIN_HERE}\\
&\text{TRIAL_MIN_HERE(MATC(UPDATE))} = \text{LHERE .OR. MINLOC}\\
&\text{TRIAL_RMIN}(\,) = \text{RMIN}\\
&\text{TRIAL_RMIN(UPDATE)} = \text{RMIN_OF_}L
\end{aligned} \tag{28}$$

A search over all pairs in a system is made for those systems for which (I_OR_J .AND.. NOT. OVER) is TRUE.

When there are two rows per system a few extra lines are necessary to find which row has the lower minima:

$$\begin{aligned}
&I\text{_OR_}J(\,) = \text{ORCOLS(LHERE .AND. MIN_HERE)}\\
&I\text{_OR_}J\text{(BASE)} = I\text{_OR_}J\text{ .OR. }I\text{_OR_}J(-)\\
&BB\text{(BASE)} = \text{RMIN_OF_}L\ .LT.\ \text{RMIN_OF_}L(-)\\
&BB\text{(. NOT. BASE)} = \text{.NOT. }BB(+)\\
&\text{RMIN_OF_}L\text{(BASE)} = \text{MERGE(RMIN_OF_}L\text{, RMIN_OF_}L(-)\text{, }BB)\\
&\text{MINLOC}(\ ,\ ) = \text{MINLOC .AND. MATC(}BB\text{)}\\
&\text{UPDATE(BASE)} = \text{RMIN_OF_}L\ .LT.\ \text{RMIN}\\
&\qquad\qquad \text{.AND.. NOT. (OVER .OR. }I\text{_OR_}J\text{)}\\
&\text{UPDATE(.NOT. BASE)} = \text{UPDATE}(+)\\
&\text{TRIAL_MIN_HERE}(\ ,\ ) = \text{MIN_HERE}\\
&\text{TRIAL_MIN_HERE(MATC(UPDATE))} = \text{LHERE .OR. MINLOC}\\
&\text{TRIAL_RMIN}(\,) = \text{RMIN}\\
&\text{TRIAL_RMIN(UPDATE)} = \text{RMIN_OF_}L
\end{aligned} \tag{29}$$

The supplied function MERGE returns in each component the corresponding component of the first or second argument according to whether the corresponding component of the logical vector variable BB is TRUE of FALSE.

When there are four or more rows per system, and therefore fewer systems, it is probably simplest to locate the minimum distance between each trial particle and the others, one system at a time, using the function MINP on matrix R with all but the rows of that system masked out [13].

5. Conclusion

Provided it can be used efficiently, which means keeping most of its PEs in useful employment, the DAP is an extremely cost-effective computer. This paper has shown how this may be achieved for the Monte Carlo study of condensed matter. Much use is made of the ability of the ICL DAP to function with moderate efficiency in vector-mode arithmetic where several PEs (32 with REAL*4 variables) handle each variable element. The calculation of the pair potential, the time consuming part with a serial machine, is done in matrix mode with all but 64 of the PEs active. The method is particularly attractive when the pair potential is relatively complicated, such as the Ewald potential expanded in Kubic Harmonics [5]. A DAP is not appropriate when the pair potential is a tabulated function, but as such functions as SQRT, LOG, EXP, and COS are all performed at about the same speed as a single multiplication [1], it is often possible to compute a pair potential in a way not acceptable on a serial machine.

The flexibility of the present scheme, allowing systems of size $m \times 2^n$ on an $m \times m$ DAP, can be further increased very easily at the expense of a few rows of permanently idle PEs. Thus on a 64×64 DAP one could have 21 systems of 192 molecules with only one row idle.

The present scheme should be of use with all sizes of DAP from 32×32 upwards. While $32^2 = 1024$ particles is not a large system by current standards, a system of 1024 molecules would be, and the alternative of say four systems each of 256 molecules is attractive. However, the larger the DAP the more important a flexibility in the system size becomes, and the prospective 128×128 DAP will need such a scheme if it is to be used for Monte Carlo of fluids.

Acknowledgments

The author in a SERC Advanced Fellow. The assistance of the staff of the DAP Support Unit at Queen Mary College is gratefully acknowledged.

References

1. R. W. Hockney and C. R. Jesshope, *Parallel Computers* (Adam Hilger, Bristol, 1981).
2. ICL Technical Publication No. 6755, 1978 (unpublished).
3. G. S. Pawley and G. W. Thomas, *J. Comput. Phys.* **47**, 165 (1982).
4. D. Fincham, N. Quirke, and D. J. Tildesley, *J. Chem. Phys.* **84**, 4535 (1986).

5. D. J. ADAMS AND G. S. DUBEY, *J. Comput. Phys.*, **72**, 153–173 (1987).
6. W. CHAPMAN AND N. QUIRKE, *Physica B* **131**, 34 (1985).
7. K. A. SMITH, S. F. REDDAWAY, AND D. M. SCOTT, *Comput. Phys. Commun.* **37**, 239 (1985).
8. D. J. ADAMS, in *Proceedings NRCC No. 9*, p. 13; Report LBL-10634, Lawrence Berkeley Laboratory, University of California, 1980 (unpublished).
9. D. FINCHAM, *CCP5 Newsletter* **12**, 43 (1984) (an informal publication available from the SERC Daresbury Laboratory, Warrington WA4 4AD, U. K.)
10. W. W. WOOD, "Monte Carlo Studies of Simple Liquid Models," in *Physics of Simple Liquids*, edited by H. N. V. Temperley, J. S. Rowlinson, and G. S. Rushbrooke (North-Holland, Amsterdam, 1968), ch. 5.
11. D. FINCHAM, *CCP5 Newsletter* **8**, 45 (1983).
12. H. M. LIDDELL AND G. S. J. BOWGEN, *Comput. Phys. Commun.* **26**, 311 (1982).
13. ICL Technical Publication No. 6918, 1979 (unpublished).