JOURNAL OF COMPUTATIONAL PHYSICS 47, 165–178 (1982)

The Implementation of Lattice Calculations on the DAP

G. S. PAWLEY

Department of Physics, University of Edinburgh, Edinburgh EH9 3JZ, U.K.

AND

G. W. THOMAS

DAP Support Unit, Queen Mary College, London E1 4NS, U.K.

Received July 31, 1981; revised March 4, 1982

The applicability of the new DAP computer to problems involving lattices is assessed. As this computer uses a fixed number (4096) of parallel processing elements (PEs) this restricts the freedom of choice of sample shapes. It is shown, however, that there is usually no disadvantage in this restriction, and that in cases where a greater flexibility is needed, this can be achieved through user's software. The setting up of equitable lattices in three and four dimensions is detailed, using both skew and straight cyclic boundary conditions. The problems of lattices with a basis are considered, even those lattices which demand an odd number of elements. It would appear that there is no lattice in any dimension which cannot be implemented on the DAP, though sometimes the programming of logical masks is necessary. The efficient masking ability of the DAP lessens this disadvantage.

INTRODUCTION

Many scientific problems involve calculations or analysis of results over a large grid of points, and for this very reason ICL have developed a distributed array processor (DAP). This computer has a number of processing elements arranged in the topology of a square lattice, with optional plane or cyclic boundary conditions in both dimensions. It may seem at first sight that these boundary conditions restrict the usefulness of the machine, but this is not so. We endeavour to show here that the DAP is not restrictive, and that lattices of all symmetries and dimensions can be set up within its store, enabling many calculations to be done in parallel.

The particular examples which will be used for illustrative purposes herein are based on physics, though the applicability of the DAP in other fields is quite clear. In the solid state, molecules are found to occupy sites within a crystal or a plastic crystal or liquid crystal, and simple models for the behaviour of these systems can be set up if a suitable intermolecular potential can be found. For plastic crystals or liquid crystals, the intermolecular interaction is so complicated and the motion so anharmonic that computer simulation is the clear choice for making advances in understanding these phases. Such results can be obtained even with simple model

0021-9991/82/080165-14$02.00/0

Copyright © 1982 by Academic Press, Inc.
All rights of reproduction in any form reserved.

potentials, but as the computational time required is so large on a serial machine, progress has so far been limited to small samples of small molecules [1]. The DAP now allows us to set up samples of 4096 molecules, each of which can be quite extensive, but it is necessary to be able to set up any possible crystallographic packing, obviously in three dimensions. It will be shown how this can be done, and what restrictions there are on the reciprocal space that is used for certain correlation functions.

A four-dimensional example is then taken from quantum field theory. Gauge theory can be expressed on a lattice, where variables on lattice links or lattice sites interact, and recent Monte Carlo calculations [2] have involved lattices typically of size 8^4. Although this is the size of the DAP array, we show how to implement a sample of size 16^4.

Both of the examples just mentioned do not lose from the fact that, as we shall see, the cyclic boundary conditions involve a stacking which is somewhat skewed. It does, however, become important in some cases to be able to retain an unskewed cyclic condition. Accordingly, we end by outlining a packing in three and four dimensions involving straight cyclic boundaries, as are required for putting renormalisation group theory onto a computer [3].

The DAP Computer

The ICL Distributed Array Processor (DAP) consists of 4096 processing elements (PEs), each element being a 4 K RAM [4]. The total storage of the DAP is thus 2 Mbytes, which contains the program, working space, and variables. All the input and output facilities are controlled by a host computer in the ICL 2900 series, and this host also performs program compilation.

The PEs are arranged on a 64×64 lattice, each PE being hardwired to its four nearest neighbours. For those PEs on the boundaries of the 64×64 square, the nearest neighbours can be selected according to either cyclic or plane boundary conditions, as directed by the user's software. The PEs can be instructed to work on vectors of 64 elements in parallel, or on matrices of 64×64 elements also in parallel. The parallel operation of the PEs is a simultaneous execution of instructions in all the PEs, obeyed in such a fashion that allows some PEs to be switched off as required by the problem. An efficient program is thus one which has the least amount of loss due to idling PEs.

A single bit in one PE has a corresponding bit in every other PE, and thus the storage can be considered as *bit-planes*, each bit-plane being a 64×64 array of bits which can be used as a LOGICAL array. Such logical arrays, used as masks, are used to control the execution of the PEs. Arithmetic operations take place in three bit-planes only, and the consequences of using such a simple method are that the basic software algorithms are sometimes quite different from those used in serial machines. Thus for example we find that the REAL * 4 square root operation is quicker than a multiply and does not need to be avoided wherever possible as is the case for a serial machine.

ICL have plans to increase the PE storage to 16 K, and also have designs for a 128×128 DAP. A suggested improvement is the upgrading of the arithmetic unit to work on bytes rather than bits, and therefore with this, a 32 Mbyte DAP with approximately 30 times the capacity of the existing DAP is currently feasible. Comparisons between machines are hard to make, but it appears that on problems involving floating point arithmetic, the present DAP is roughly comparable with the CRAY-1 for problems which suit both machines. For problems involving logicals, for which the bit-processor DAP is ideally suited, it is best simply to record that 4096 logical comparisons can be made in 1.2 μsec.

Programming the DAP is done in the high-level language called DAP-FORTRAN [5]. This is an extension of FORTRAN to include LOGICAL, INTEGER $*m$ ($m = 1, 2,..., 8$) and REAL $*n$ ($n = 3, 4,..., 8$) arithmetic in scalar, vector, or matrix form. The DAP-FORTRAN commands which are pertinent to this paper will now be outlined. The square array of PEs must be considered to have four boundaries defined by the points of the compass. Let A and B be defined as 64×64 arrays, then the operation

$$A = \mathrm{SHWC}(B, N) \tag{1}$$

takes every element of B and shifts it westwards cyclically by N places and puts the result in A. Thus the members of A are replaced as in the following FORTRAN code, where $N \geqslant 0$:

```
      DO 1 I = 1, 64
      DO 1 J = 1, 64
      K = J + N
      IF(K.GT.64)K = K - 64
1     A(I, J) = B(I, K)
```

The cyclic nature of the shift, being a hardware feature, does not involve any computation and greatly facilitates the implementation of cyclic boundary conditions in many dimensions. Shifts east, north, and south are obtained by using E, N, and S instead of W.

There is another form of shifting operation which treats A and B not as 64×64 elements of arrays but as 4096 elements of vectors, known as long-vectors in order to distinguish from the ordinary vectors of 64 elements. These long-vectors can be shifted cyclically with the command

$$A = \mathrm{SHLC}(B, N) \tag{2}$$

which shifts B to the left by N places (SHRC would shift to the right) according to the normal FORTRAN instructions

```
      DO 1 I = 1, 4096
      J = I + N
      IF(J.GT.4096)J = J - 4096
    1 A(I) = B(J)
```

The third command required in this paper involves the use of logical masks. If L is defined as a logical array (one bit-plane) it can be formed in the usual ways of FORTRAN, for example

$$L = A.\text{GT}.B$$

In the FORTRAN context this would be

```
      DO 1 I = 1, 4096
      L(I) = .FALSE.
      IF(A(I).GT.B(I)) L(I) = .TRUE.
    1 CONTINUE
```

If we now wish to set the long-vector C to the larger value of the corresponding elements in A or B, this can be done by

$$C = \text{MERGE}(A, B, L), \tag{3}$$

where $C(I)$ is replaced by $A(I)$ if $L(I)$ is .TRUE. and by $B(I)$ if $L(I)$ is .FALSE. for all $I = 1, 4096$.

There are many more instructions in the DAP-FORTRAN language [5] necessary for the efficient construction of programs for parallel execution. The power of such commands is readily appreciated and they may well become accepted in a new international version of the FORTRAN language [6].

Cyclic Boundary Conditions

Many problems in three or more dimensions are conventionally tackled by setting up a block with cyclic boundary conditions in all directions. For example, a $3 \times 3 \times 3$ block would be arranged and packed as follows. The first nine elements are arranged in two dimensions (Fig. 1) and two further layers are placed on top of this

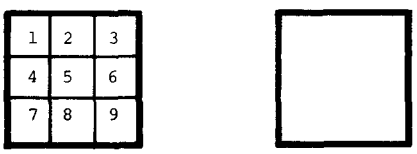

FIGURE 1

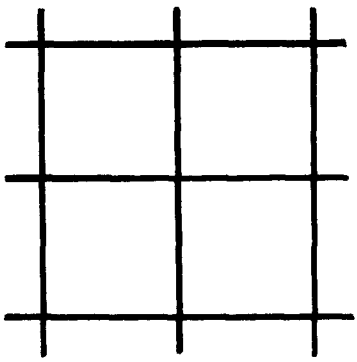

FIGURE 2

square, making the basic block. These blocks are packed as in Fig. 2, so that the nearest neighbours for element 1 in the plane of the diagram are elements 2–4 and 7, as shown in Fig. 3, and to these must be added 10 situated above 1, and 19 below 1. Second nearest neighbours are elements 5, 6, 8, and 9, (Fig. 4) with elements 11–13 and 16 above and 20–22 and 25 below. Third neighbours (cube corners) are elements 14, 15, 17, and 18 above and 23, 24, 26, and 27 below.

Such an arrangement, as described above, is not straightforward in the DAP, as the DAP is designed for the easy use of a two-dimensional cyclic boundary condition on a sample restricted to 64×64. However the DAP is endowed with the facility to use a vector, length 64, in a one-dimensional cyclic mode, or the whole array of 4096 elements as a long-vector, again in a one-dimensional cyclic mode. This facility can be used to set up arrays of any dimensionality with cyclic conditions in all dimensions, though it is not easy to set up the conventional form of packing as described in the previous paragraph.

First let us understand how to use the long-vector representation to give cyclic conditions in two dimensions. We take as an example a linear DAP with 12 PEs, though it must be remembered that this example is merely illustrative—it is not possible to use the actual DAP conveniently as a 12 PE linear DAP. The numbering of the PEs is as in Fig. 5. In Fig. 6, the whole vector of Fig. 5 has been copied a number of times, whereupon it is clear that all the twelve distinct elements appear

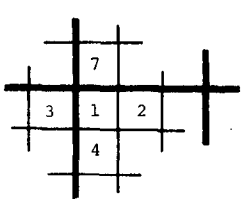

FIGURE 3

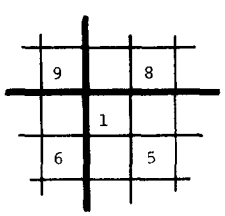

FIGURE 4

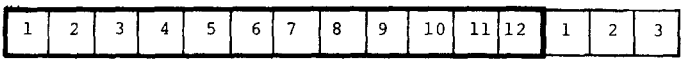

FIGURE 5

within a rectangle, and this rectangle, when repeated, completely covers the two-dimensional plane. If any particular element and all its copies are selected, they form a lattice with a unit cell which is not a rectangle but a parallelogram, as in Fig. 7.

The two-dimensional space has thus been covered, using a skew cyclic boundary condition implemented via a one-dimensional built-in condition. We shall see that the generalisation of this result is that any dimensionality can be modelled using a linear DAP, but it is important to show at this stage that the size of the DAP is not a restriction, as long as it is large enough. Thus if we had a 5 PE linear DAP, we could still cover two-dimensional space equitably (Fig. 8).

The meaning of equitable coverage must now be examined. If we return to the 12 PE linear DAP, we could cover two-dimensional space as in Fig. 9, but it is immediately obvious, as the double-headed arrow shows, that the distance between any site and its nearest copy is much shorter than in Fig. 6. In all interactive statistical problems which can be attempted on the DAP, it is important to find an arrangement where the closest approach of two copies is as large as possible. This is then an equitable arrangement. It must be remembered that the computer itself does not have stored copies of an element, but that the *copy* is in our mental picture of the sample; only one copy of any sample is ever needed.

Returning to Fig. 6, it is obvious that the shortest vector between copies is given by the top side of Fig. 7, the unit cell of this sample. The fact that this vector is not straight across the page does not usually matter, but for certain purposes it is necessary to establish what the unit vectors defining Fig. 7 are. Let us take axes X

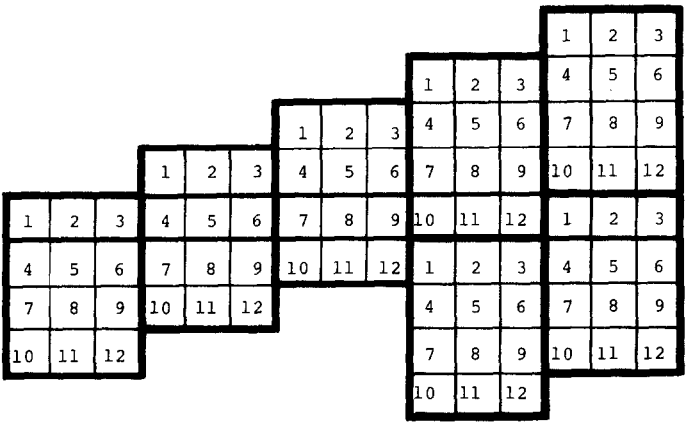

FIGURE 6

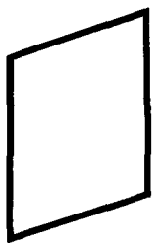

FIGURE 7

across the page and Y downwards and then it is clear that the unit vectors for this case are

$$(X, Y) = (3, -1) \qquad \text{and} \qquad (0, 4), \tag{4}$$

giving a minimum lattice spacing of $\sqrt{3^2 + 1^2}$ which is not very much less than the optimum $\sqrt{12}$ for the DAP of this example.

With this same example, let us continue to build up a sample in three dimensions, using exactly the same procedure as before. A copy of the infinite two-dimensional space of Fig. 6 is laid on top of the first, but displaced in such a way as to make an equitable lattice. There is clearly a considerable flexibility in this choice, the decision made here not being unique. Let us take the relative arrangement such that element 6 of the copy lies above element 1 in the original, and define the Z axis to be in the

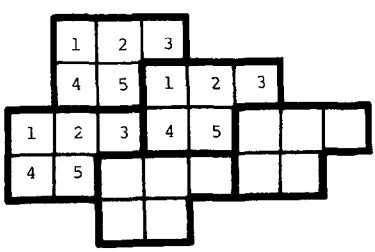

FIGURE 8

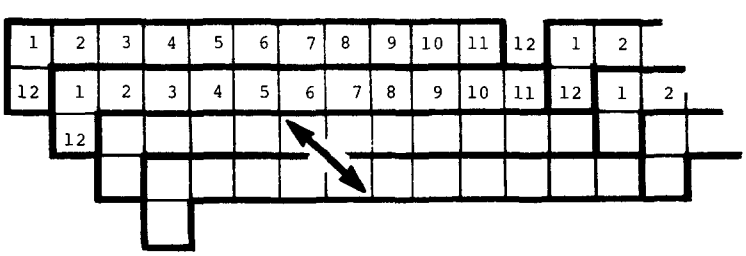

FIGURE 9

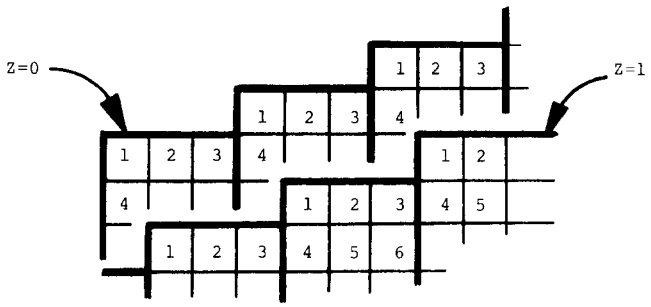

FIGURE 10

direction from element 1 to the element 6 above. In Fig. 10, part of the layer at $Z = 1$ is drawn out over the layer at $Z = 0$. The unit vectors which now define the unit cell are as in Fig. 9, with $Z = 0$ plus a third vector involving Z. The following can be chosen

$$\begin{aligned}(X, Y, Z) &= (3, -1, 0), \qquad (0, 4, 0), \qquad (1, 2, 1) \\ &= \quad \mathbf{a}, \qquad\qquad \mathbf{b}, \qquad\qquad \mathbf{c}.\end{aligned} \tag{5}$$

Confirmation that these vectors define a correct unit cell is given by calculating the cell volume, as this must be equal to the size of the DAP. In this example

$$\mathbf{a} \cdot \mathbf{b} \times \mathbf{c} = 12. \tag{6}$$

Certain simulation studies require the calculation of correlation functions, some of which are expressed in the reciprocal space related to the sample unit cell. This space has axes $\mathbf{a}^*, \mathbf{b}^*, \mathbf{c}^*$, where

$$\mathbf{a}^* = (\mathbf{b} \times \mathbf{c})/(\mathbf{a} \cdot \mathbf{b} \times \mathbf{c}), \tag{7}$$

etc., and, using the vectors of (5) and the result (6) we find

$$\mathbf{a}^* = (\tfrac{1}{3}, 0, -\tfrac{1}{3}), \qquad \mathbf{b}^* = (\tfrac{1}{12}, \tfrac{3}{12}, -\tfrac{7}{12}), \qquad \mathbf{c}^* = (0, 0, 1). \tag{8}$$

These vectors define the reciprocal lattice, and the appropriate correlation function is only defined at the points of this lattice. Like the direct lattice, the reciprocal lattice is oblique which, although perhaps inconvenient in practice, does not detract from the validity of the results which are obtained.

Some molecular dynamics simulations involve Coulomb interactions, and where it is desired to calculate these accurately, the Ewald transformation [7] can be calculated just as with a serial computer. The reciprocal space contribution to the Coulomb interaction will then be calculated on the reciprocal lattice just defined without any loss of generality. In the very large samples afforded by the DAP, it is

probably more realistic to use the screened version of the Ewald summation [8]; this should present no more of a problem on a parallel processor than on a serial machine.

A Nonprimitive Lattice

The discussion so far has involved a primitive cubic lattice in direct space, but such an arrangement is not common in solid state systems. A more typical real example is afforded by the plastic crystalline phase of SF_6 [9], in which the roughly spherical molecules are arranged on a body-centred cubic lattice. The number of body-centred cubes which can be represented on the full-size DAP is half of 4096, as each molecule needs its own PE, and therefore the radius of equitability is

$$\sqrt[3]{4096/2} \simeq 13. \tag{9}$$

Packing therefore begins as before in two dimensions (Fig. 11) until all 4096 elements are used and an infinite plane is built up. A copy of this plane is now placed with the element 170 $(=13^2+1)$ not directly above element 1 at the point (0, 0, 1), but at $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, this being the basic vector for the body centre. The same principles are used in this construction, however, and the reader will be able to calculate that an equitable packing has been achieved. The unit cell and reciprocal cell vectors are found to be

$$\mathbf{a} = (13, -1, 0), \qquad \mathbf{b} = (-\tfrac{1}{2}, 12\tfrac{1}{2}, -\tfrac{1}{2}), \qquad \mathbf{c} = (\tfrac{1}{2}, 3\tfrac{1}{2}, 12\tfrac{1}{2});$$

$$v\mathbf{a}^* = (316, 12, -16), \qquad v\mathbf{b}^* = (25, 325, -92), \qquad v\mathbf{c}^* = (1, 13, 324);$$

where $v = 4096 = \mathbf{a} \cdot \mathbf{b} \times \mathbf{c}$.

The body-centred lattice just described is typical of all crystallographic lattices in that no special character of the lattice is invoked in order to simulate it in the DAP. No further proof is required to show that any lattice without a basis can be simulated on the DAP, this statement being true in higher dimensions than three as lattices in higher dimensions can be constructed by repeating the procedure used for going from one to two and then from two to three dimensions. Difficulties are encountered, however, when attempting to simulate lattices with a basis, as we shall see in the next section.

The value of the DAP has already been demonstrated [9] in the study of the

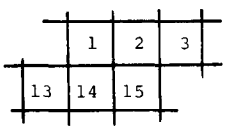

Figure 11

condensed phases of SF_6. The phase below the melting point is plastic, and the molecular dynamics calculations show the occasional molecular reorientations characteristic of this phase. On cooling, the sample shrinks and undergoes a transition to a triclinic crystalline phase—this is made possible because the sample is sufficiently large that a number of differently oriented crystallites nucleate and grow. The region between the crystallites becomes disordered, thus allowing relief from all possible shear strains.

Lattices with a Basis

A lattice *without* a basis consists of a set of points which are related to each other by the basic vectors, which are equal in number to the dimensionality of the space. In our long-vector mode of using the DAP, this means that if one element has a neighbour three places to the right, then all elements have corresponding neighbours three places to their right. This set of neighbours can be selected in DAP FORTRAN from the array of molecules *MOL* by shifting the whole array by three places to the left cyclically, SHLC(*MOL*, 3), bringing the neighbour set into position coincidence with the molecule set originally considered.

This procedure does not work so simply when there is a basis. In such a case, there are two or more molecules in the unit cell, each positioned in the cell by a basis vector from the cell origin. Many cases in crystallography are such that when the basis vectors are considered without regard to the fact that the molecules at the positions they define are in various different orientations, then the result is a true lattice. Such a lattice can be represented on the DAP as before as long as the number of basis vectors is 2^n. Difficulties therefore arise in cases such as exemplified by a trigonal lattice with three basis vectors.

Such an arrangement can sometimes be represented by a sequence of molecules $A, B, C, A, B, C, \ldots$, where A, B, and C have different orientations. Following this sequence around the long-vector we find that there is one element too many, the remainder of dividing 4096 by 3, and one must be chosen to be discarded. This is not impossible as we can see by taking an example where we wish to discard the fifth member (Fig. 12). We bring the neighbour from three to the right as before (Fig. 13).

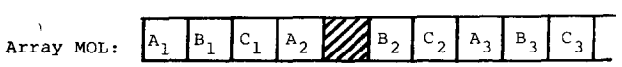

FIGURE 12

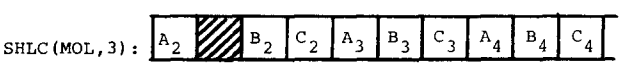

FIGURE 13

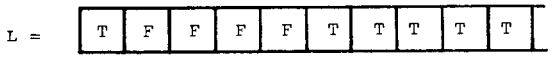

FIGURE 14

The coincidence is true (T) for most of the pairings, but is false (F) in four places (Fig. 14). In these four places SHLC(MOL, 4) would make matters correct. This can be brought about by setting up a logical matrix L as shown in Fig. 14, and using the MERGE operation:

$$\text{MERGE}(\text{SHLC}(MOL, 3), \text{SHLC}(MOL, 4), L) \tag{10}$$

The logical matrix permits the first entry inside MERGE to be used for those values which are .TRUE. and the second for those which are .FALSE..

The example just given shows how some problems of a lattice with a basis can be tackled. A much easier way is sometimes open if there is no problem imposed by the store size of the DAP. Each PE has the capacity for 128 four-byte words, and it is therefore quite possible that the information for more than one molecule can be stored at one PE. Thus if each PE kept information on one A, one B, and one C molecule of the example just considered, there would be no need for the double shift of (10). The sample is then much larger, but as the needs of many simulations are such that larger samples give equivalent results with shorter computation times, then there is no loss. Indeed there are advantages in having a larger sample as sample size artefacts are reduced, but this is balanced by the longer calculation usually required to develop a sample from approximate original conditions to a stable state.

The analysis so far has not been complete as there will be cases where there is insufficient depth in the DAP for multiple storage and where it is not possible to arrange the molecules of the basis in a sequence like $A_1, B_1, C_1, A_2, \ldots$, such that the relationship between A_1 and B_1 is the same as between B_1 and C_1 and as between C_1 and A_2. In such cases, masking techniques have to be used, for which the DAP is ideally suited. The examples to be discussed in the remainder of the paper make use of masks to solve other problems which arise in certain classes of lattice calculations.

MONTE CARLO CALCULATIONS

To do Monte Carlo calculations for $U(1)$ gauge theory [2], a four-dimensional lattice is set up with variables associated with each link of the hypercubic lattice. Possible new values for these variables are generated by a random procedure and the choice as to whether to replace an old value by a new value depends on the configuration of neighbouring variables. In these problems it is a general rule that, in order to maintain detailed balance, the neighbouring variables should not be simultaneously subjected to the possibility of updating, and this causes a grouping of the elements of the lattice into sets of noninteracting variables which can be

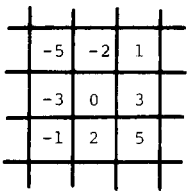

FIGURE 15

simultaneously updated. In the case of the four-dimensional lattice gauge calculation, the grouping is into two sets, one set being the even positions and the other set the odd positions in the long-vector. The choice of basic vectors is then made so that an even element has only odd neighbour elements and vice versa. Certain operations in the calculation require the distinction between the two sets to be made, and this is done by setting up a logical long-vector which alternates between .FALSE. and .TRUE., and then using this as a mask to select the appropriate calculation. The operation of such masks on the DAP is very easy and very swift. In the particular example of $U(1)$ gauge theory mentioned above, about 25,000 possible updates per second were analysed [2], the relative efficiency of the DAP being about 90%. The efficiency rises further when a larger sample is used, as the odd and even sample points can be arranged on different planes of the DAP.

A more complex organisational problem is afforded by an Ising model sample with $64 \times 64 \times 64$ spins which can interact with nearest and next-nearest neighbours. To do this, all interacting sites must be stored on different planes of the DAP. The sites on the same sample plane can be stored on planes given by the indices shown in Fig. 15, where the plane numbered 0 is the plane through an arbitrary point in the sample taken for consideration. Any actual planes indexed outside the range 1 to 64 are treated mod 64. The nearest and next-nearest neighbours for element 0 which lie in the planes of the sample directly above and below it are respectively stored on the planes shown in Figs. 16a and b. The regular pattern of the storage is easily seen to be consistent with the built-in cyclic boundary conditions, and as no neighbour lies on plane number 0, all the spins on this storage plane may be simultaneously updated. The index which serves to define the requisite storage plane may be obtained

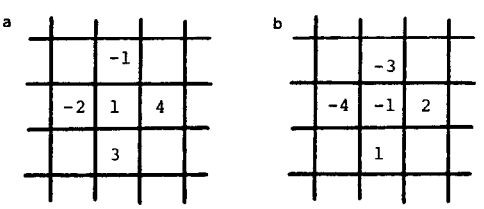

FIG. 16. The nearest and next-nearest neighbours for element 0 on the planes (a) above and (b) below.

through indirect addressing in a most convenient manner. Let *INDEX* be a 64 component vector, initialised so that $INDEX(I) = I$. The storage plane indices are taken from this vector, which can then be shifted after each plane is updated by

$$INDEX = \text{SHLC}(INDEX, 1),$$

thus presenting new indices consistent with the cyclic condition for the third dimension.

In certain problems, such as computerised renormalisation group theory [3], it is necessary to have straightforward sample cyclic boundary conditions, such as $16 \times 16 \times 16 \times 16$, so that this sample can be reduced to a topologically similar $8 \times 8 \times 8 \times 8$ sample and thence to a $4 \times 4 \times 4 \times 4$ arrangement. First we see how an $8 \times 8 \times 8 \times 8$ arrangement can be set up on the DAP.

To do this the geometry of the DAP to be used is the two-dimensional cyclic geometry. The array is divided into 8×8 blocks each of 8×8 elements so that the cyclic condition for two dimensions is automatic by blocks, but the arrangement for the other two dimensions must be got from within an 8×8 block. Let these two latter dimensions be y and z, while w and x are the two dimensions with automatic conditions. Thus the relation between a point with value w and one with value $w + 1$ is a movement laterally by 8 steps (1 block), and this can be done by shifting *west* cyclically, SHWC(*MOL*, 8). The relation between a point with value x and one with value $x + 1$ is likewise SHNC(*MOL*, 8), being a shift *north* cyclically. The relation between a point with value y and one with value $y + 1$ is SHWC(*MOL*, 1), except that this is incorrect at the edges of the 8×8 blocks. For any member so situated on the *western* edge of an 8×8 block the correct relationship is SHEC(*MOL*, 7), as this refers cyclically to the other (eastern) edge of the same block. If the logical matrix L is set .TRUE. except for the elements on the western sides of the blocks where the second shift is appropriate, then the neighbours for $y + 1$ are found by again using the MERGE facility:

$$\text{MERGE}(\text{SHWC}(MOL, 1), \text{SHEC}(MOL, 7), L). \tag{11}$$

A similar procedure is used for implementing the cyclic condition in the coordinate z, and the result is a straight-packed $8 \times 8 \times 8 \times 8$ arrangement.

Finally, we present a way of achieving a straight-packed $16 \times 16 \times 16 \times 16$. As this is 16 times the DAP area capacity, one of the dimensions has to be satisfied by using the depth of the DAP. The technique used for this needs no explanation as it is the same as would be used on a serial computer. Thus all that remains is to implement $16 \times 16 \times 16$ in the 64×64 array. Again coordinates w and x can be satisfied by blocking, this time into 16×16 square blocks, each of 4×4 elements. The coordinate y has to be implemented within each 4×4 array. The values of y within this array could be chosen to be as in Fig. 17, so that moving an element with value $y + 1$ to coincide with y would be given by SHWC(*MOL*, 1) for elements 2–4, 8, 14; SHEC(*MOL*, 1) for elements 6, 10–12, 16; SHNC(*MOL*, 1) for 5, 7, 9 and

1	2	3	4
16	15	6	5
13	14	7	8
12	11	10	9

FIGURE 17

SHSC(*MOL*, 1) for 1, 13, 15. Four masks are set up in order to select the particular shift required, and the result can be produced by multiple merging.

This last example may seem at first sight to be rather clumsy, but it should not therefore be assumed that its implementation is inefficient. It so happens that a shift by *one* place, as exemplified by SHWC(*MOL*, 1) is so much quicker than a cyclic long-vector shift such as SHLC(*MOL*, 4) that the operation of performing the four single shifts and masks for working on the arrangement in Fig. 17 takes no more time than the single long-vector shift.

CONCLUSIONS

There would seem to be no lattice and boundary condition that may be needed for computations on the DAP that cannot be implemented. It is shown that there is no disadvantage in the two-dimensional architecture of the DAP for the implementation of any lattice calculation. The algorithms developed for work on this particular computer will certainly be appropriate for any true large scale parallel processor that may be developed in the future, and thus such computers should greatly assist the progress of the appropriate simulations. For further details of the DAP-FORTRAN language the reader is referred to the references cited [4, 5].

REFERENCES

1. D. G. BOUNDS, M. L. KLEIN, AND G. N. PATEY, *J. Chem. Phys.* **72** (1980), 5348.
2. K. C. BOWLER, G. S. PAWLEY, B. J. PENDLETON, D. J. WALLACE, AND G. W. THOMAS, *Phys. Lett. A* **104B** (1981), 481.
3. J. TOBOCHNIK, S. SARKER, AND R. CORDERY, *Phys. Rev. Lett.* **46** (1981), 1417; D. P. LANDAU AND R. J. SWENDSEN, *Phys. Rev. Lett.* **46** (1981), 1437.
4. R. W. GOSTICK, *ICL Tech. J.* **1** (1979), 116; R. W. HOCKNEY AND C. R. JESSHOPE, "Parallel Computers," Hilger, Bristol, 1981.
5. "Introduction to FORTRAN programming," ICL Technical Publication No. 6755, 1979.
6. W. BRAINERD AND J. ADAMS, *in* "Information Processing '80" (S. H. Lavington, Ed.), p. 361, North-Holland, Amsterdam, 1980.
7. M. BORN AND K. HUANG, "Dynamical theory of crystal lattices," Clarendon, Oxford, 1954.
8. E. R. COWLEY, J. K. DARBY, AND G. S. PAWLEY, *J. Phys. C* **2** (1969), 1916.
9. G. S. PAWLEY AND G. W. THOMAS, *Phys. Rev. Lett.* **48** (1982), 410.