DATA MAPPINGS IN LARGE PARALLEL COMPUTERS

J. van Leeuwen

H.A.G. Wijshoff

RUU-CS-83-11

July 1983

Rijksuniversiteit Utrecht

Vakgroep informatica

Postbus 80.012
3508 TA Utrecht
Telefoon 030–53 1454
The Netherlands

DATA MAPPINGS IN LARGE PARALLEL COMPUTERS

J. van Leeuwen
H.A.G. Wijshoff

Technical Report RUU-CS-83-11
July 1983

Department of Computer Science
University of Utrecht
P.O. Box 80.012
3508 TA Utrecht
the Netherlands

DATA MAPPINGS IN LARGE PARALLEL COMPUTERS *

(Invited Paper)

J. van Leeuwen and H.A.G. Wijshoff

Department of Computer Science, University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht, the Netherlands

Abstract. Parallel computers (such as vector machines and array-processors) feature the availability of many, highly pipelined processing units and many memory banks that can be accessed independently in parallel at great speed. Aside from needing adequately parallelized ("vectorized") algorithms, their application requires general storage mappings for distributing vector data and retrieving it from memory at low cost. Data mappings of this kind, also known as "skewing schemes", were first considered during the design of the ILLIAC IV in the late nineteen sixties. We survey known results and recent advances of the remarkable theory of skewing schemes.

1. Introduction. Ever since computers are being built, researchers and manufacturers have looked for ways of designing faster machines. In the early nineteen fifties and up to the present day efforts focused on improving the technology and speed of individual components, with the increasing use of hardware overlap and pipelining (see e.g. Lorin [16]). In the nineteen sixties the insight came that the von Neumann-type computer architecture itself would have to be changed to achieve further speed-ups in execution. Schwartz [19] wrote in 1965: "The approach of present computers to speeds at which the velocity of light becomes a significant design factor, and the continued fall in the price of computer components have directed attention to the use of parallelism as a device for increasing computational power." The development that led to vector machines and other "super computers" is described in Hockney & Jesshope [11].

The improvements in computer speed were brought about both by improved technology and by radically different approaches to computer architecture, which led away from the "sequential" von Neumann-type architecture and gave rise to parallel (tightly coupled) and distributed (loosely coupled) machine designs. The effectiveness of parallel machines was doubted in their early stages of development (see e.g. Amdahl [1] and Thurber & Patton [25]) but it appears that the machines are now viable in most domains of scientific computing. The impact on e.g. numerical methods will be significant, although the investments for redesigning ("vectorizing") existing software will be high. New techniques of algorithm design for parallel and "systolic" computa-

---

* To be presented at the GI Jahrestagung, Hamburg, Oct. 4-6, 1983.

tion are only beginning to emerge (see Kung [13], Haynes et.al.[8], van Leeuwen [26]).

Underlying each computer architecture is a model of computation, i.e., a notion of how computations are to proceed and of how components in an architecture interact. A summary of the correspondences for present day architectures is given in figure 1 (from Böhm [2]). The following five broad categories of parallel computers are often

Model of Computation	Corresponding Computer Architecture
A. Sequential control on scalar data	A1. von Neumann-type computer A2. Multifunction CPU A3. Pipelined computer
B. Sequential control on vector data	B1. Vector computers B2. Array processors
C. Independent, communicating processes	C1. Shared memory multiprocessors C2. Ultra computers C3. Networks of small machines
D. Functional and data-driven computation	D1. Reduction machines D2. Dataflow machines

Figure 1. Computer architectures and their underlying computational model

distinguished: pipelined processors (including the CRAY-1/1S and Cyber 203/205), SIMD machines (including multiprocessor designs such as the ILLIAC IV and the Burroughs BSP), array processors (a distinguished class of SIMD machines including e.g. the AP-120B and the ICL-DAP), MIMD machines (distributed processor arrangements) and shared memory computers (a class of MIMD machines including e.g. the CRAY-XMP). The distinction between SIMD and MIMD machines was originally proposed by Flynn [6] (see also Stone [23]). All parallel computers fit the global form suggested in figure 2, with many differences in the way the various "sections" are realized. For example, the transport section is a highly pipelined data channel in some computers and a single-stage or multi-stage processor/memory interconnection network in other designs (see e.g. Thurber [24]). Memory is invariably partitioned, and is distributed as local storage over the individual processors or is divided into M memory banks for global access. In some machines M is a suitable power of two (M=8 for the Cyber 205, M=16 for the CRAY-1) whereas in other designs M was specifically chosen to be a prime number (M=17 in the Burroughs BSP, see also Lawrie & Vora [15]).

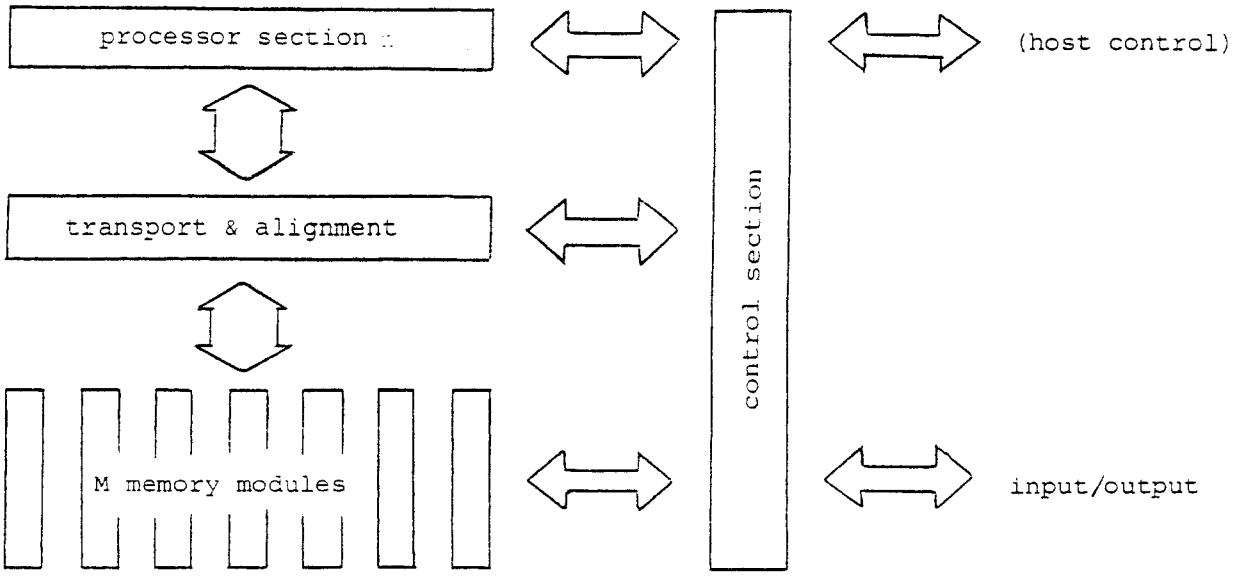

Figure 2. Global block diagram of a parallel computer

The effectiveness of (SIMD-) machines derives from the fact that in one cycle a complete vector of M data items can be fetched by simultaneous access to each of the M memory banks, which can subsequently be piped to the processor section through the data channel or alignment network. Large vectors must be broken up in chunks of size ≤M and are retrieved by multiple "parallel fetches". Kuck [12] (see also Budnik & Kuck [3]) has shown that the optimal benefit from "parallel memories" requires non-trivial data-mappings or storage schemes, in order that vectors and blocks of data that are needed in the course of an algorithm are indeed available from distinct banks. For example, storing an N×N matrix (N≤M) with one column in every bank allows fully parallel access to every row and diagonal in one cycle but forces sequential access for retrieving the elements of every column. On the other hand, a "skewed" organization as shown in figure 3 (with M=5) alleviates these difficulties at least for rows,

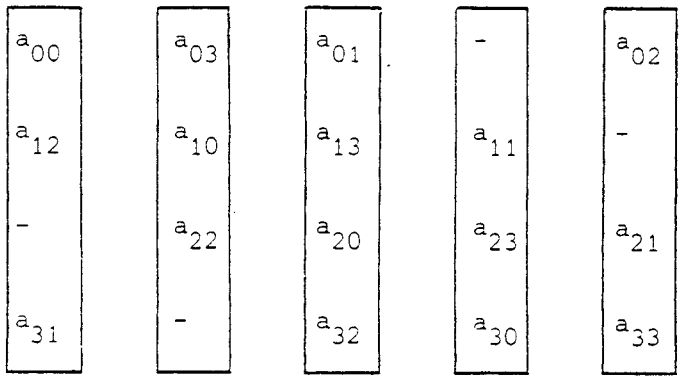

Figure 3. Storing a 4×4 matrix into 5 memory modules

columns and forward and backward diagonals.

Whenever a set of M data items must be retrieved but is not available from distinct banks and (hence) cannot be retrieved as an M-vector in one cycle we speak of a "conflict". Following the terminology of [12], any storage scheme s that maps the elements of an N×N matrix to M memory banks and provides for the conflict-free access to various vectors of interest is called a "skewing scheme". We often assume that $M \geq N$ (an obvious condition for conflictfreeness) and number the memory banks from 0 to M-1. In this paper we shall survey known results and recent advances in the remarkable theory of skewing schemes. For most proofs we refer to the published literature and to Wijshoff & van Leeuwen [28], [29], [30] and [31].

Even though M is "small" in current parallel computers (but designs exist with M up to 521) the theory will be presented in general terms. Also different interpretations make the general theory applicable in other domains of parallel computing e.g. the design of (parallel) raster graphics processors. The assignment of pixels to processor chips with the condition that e.g. 8×8 boxes of them can always be updated rapidly (thus, with parallel processing of the pixel information) as studied in Chor, Leiserson & Rivest [4] is just another instance of the theory of skewing schemes. Combined with assumptions of "unbounded parallelism" (see [26]) in processing power, the use of concrete data mappings makes the analyses of parallel algorithms more realistic (see e.g. Montoye & Lawrie [17]).

2. Data mappings. The theory of skewing schemes was initiated in Budnik & Kuck [3] and Lawrie [14], and developed further by Shapiro [20]. A skewing scheme s is a mapping from [0..N-1]×[0..N-1] into [0..M-1] assigning the "cells" of any N×N matrix to the available memory banks. Shapiro [20] argued that if a valid skewing scheme with some hereditary property is to exist for every N large enough then we may as well assume that there is a single, valid skewing scheme defined on the entire $\mathbb{Z}^2$ (as the ultimate matrix domain). Validity almost always refers to the conflictfreeness of s for retrieving any set of elements arranged according to some specified data template T.

The simplest and most commonly used skewing schemes are the "linear skewing schemes" defined by

$$s(i,j) = ai+bj \pmod M$$

for suitable a and b. The theory of such schemes will be developed in section 3. The use of linear skewing schemes explains why M is preferred to be prime: it makes $\mathbb{Z}_M$ a field, and desirable properties of s hold almost regardless the values of a and b. (There are algorithmic reasons as well, see e.g. Hockney & Jesshope [11] ch. 5.) Traditionally linear skewing schemes are introduced because, when the parameters are set right, they easily provide conflict-free access to common vectors of interest

such as rows, columns and diagonals of a matrix with almost no computational (or hardware!) costs. An additional reason is that "linearly skewed" vectors can be unscrambled, i.e., brought back to non-permuted order quite easily as well. If processor(s) and memories are connected by a switching network, then it may be harder to compute the required control vectors. Lawrie [14] has shown that this can usually be done for the Omega network (of order $M=2^m$), the log M-iterate of the common shuffle-exchange network.

Definition. A d-ordered k-vector is a vector of k elements whose i^{th} logical element ($0 \leq i < k$) is stored in memory bank $c+di \pmod M$, for some constant c.

Theorem 2.1([14], [30]). A d-ordered k-vector can be accessed conflict-free if and only if $M \geq k.\gcd(d,M)$.

Theorem 2.2([30]). A d-ordered k-vector can be accessed in r conflict-free fetches if and only if $M \geq (\lfloor \frac{k-1}{r} \rfloor + 1)\gcd(d,M)$.

It follows that a d-ordered k-vector can be accessed in precisely $\lfloor \frac{(k-1)\gcd(d,M)}{M} \rfloor + 1$ conflict-free fetches (where each fetch operation retrieves a d-ordered subvector), which is optimal. The simple results for d-ordered vectors are only of limited use because aside from rows, columns and diagonals only a few other classes of vectors can be molded into a "d-ordered" form for a given s.

Clearly more general skewing schemes are only practical if they can be described by a small amount of tabular information or by a simple formula. General skewing schemes that fit this description were called "periodic skewing schemes" by Shapiro [20]. The theory of such schemes will be developed in section 4 (from [29]).

Definition. A skewing scheme s is called regular if and only if the following property is satisfied for all cells $p,q \in \mathbb{Z}^2$: if $s(p)=s(q)$ then any pair of cells that are in the same relative position as p and q are mapped to equal banks.

Definition. A skewing scheme s is called periodic if there are M cells $a_1,\ldots,a_M$ and a lattice L such that (i) the "cosets" a_i+L ($1 \leq i \leq M$) are all disjoint but cover the entire $\mathbb{Z}^2$, and (ii) s maps all cells in a_i+L to memory bank i. (We shall call s periodic "with lattice L" and call L the lattice "of s".)

For the elementary notions concerning integral lattices (basis, fundamental parallellogram, determinant, reduction modulo a lattice) we refer to [7].

Theorem 2.3([29]). A skewing scheme is periodic if and only if it is regular.

In general skewing schemes are designed for obtaining conflict-free access to arrangements of cells that belong to a specified set of data templates $T_1,\ldots,T_k$. In case of a single template T the smallest number of memory banks needed obviously is M=|T| (the size of T) but more banks may be required. In general the problem arises of finding the smallest number of memory banks for skewing data so as to have conflict-free access for a set of templates of interest. The following result of Shapiro [20] links the theory of skewing schemes to the combinatorial theory of packing and covering (a simplified proof is given in [28]).

Theorem 2.4. There exists a valid skewing scheme for template T using exactly M=|T| memory banks if and only if T tessellates the plane.

(Here tessellations are assumed not to involve any rotations or reflections of the template.) Through many combinatorial interpretations data mappings that provide conflict-free access relate to several deep problems in mathematics.

The strict requirement of conflict-free access is a theoretical one because the number of memory banks, however large, is fixed and the vectors that arise in any practical algorithm will be much larger and vary in length. Thus conflict-free access is a requirement to guarantee a "minimum" number of foldings of a vector of interest, hence the number of parallel fetches required to assemble the full vector from memory. Clearly problems arise if a vector is "scattered" and the hardware (as in the CRAY) needs to transport data in blocks of fixed length to and from a register-file. In this way hardware-dependent considerations enter into the choice of a data mapping for a problem and consequently, in the design of the parallel algorithm. A data mapping and an algorithm that perform well on one machine may do badly on another machine because of differences in the handling of scrambled vectors by the hardware (and many other features of the architecture!). As noted by Lawrie & Vora [15] it is likely that future parallel computers will depend less on minimizing conflicts in array accesses and more on mere average time considerations. One possibility would be to map data by parallel hashing.

3. Linear skewing schemes. We consider linear skewing schemes s with $s(i,j) = ai+bj \pmod M$, some a and b, for $0\leq i,j<M$. We assume that s uses all memory banks, which is the case precisely when (a,b,M) = 1. Skewing schemes of this kind will be called "proper". Two linear skewing schemes s and s' may very well be equivalent, in the sense that one can be obtained from the other by a suitable renaming of the memory banks.

For example, when M is prime $s(i,j) = ai+bj \pmod M$ is equivalent to $s'(i,j) = i+a^{-1}bj \pmod M$ and there are only M essentially different linear skewing schemes. For M arbitrary this is no longer simple. Let $s(i,j) = ai+bj \pmod M$ and $s'(i,j) = a'i+b'j \pmod M$, and let

$$\Delta(s,s') = \begin{vmatrix} a & b \\ a' & b' \end{vmatrix}$$

Theorem 3.1([30]). Two proper linear skewing schemes are equivalent if and only if $\Delta(s,s') \equiv 0 \pmod M$.

Theorem 3.2([30]). The number of essentially different (i.e., non-equivalent) proper linear skewing schemes using M memory banks is $O(M \log\log M)$.

Given a linear skewing scheme $s(i,j) = ai+bj \pmod M$ that is used for storing an $N \times N$ matrix ($M \geq N$) it is easily seen that (i) rows are b-ordered N-vectors, (ii) columns are a-ordered N-vectors, (iii) non-circulant diagonals of length k are (a+b)-ordered k-vectors ($1 \leq k \leq N$) and (iv) non-circulant anti-diagonals of length k are (a-b)-ordered k-vectors ($1 \leq k \leq N$). Using theorem 2.1. we obtain simple conditions for s in order that it provides conflict-free access to rows, columns and non-circulant diagonals and anti-diagonals (cf. Lawrie [14]):

$$(*) \begin{cases} M \geq N.\gcd(a,M) \\ M \geq N.\gcd(b,M) \\ M \geq N.\gcd(a+b,M) \\ M \geq N.\gcd(a-b,M) \quad (a \neq b) \end{cases}$$

Conditions (*) are clearly satisfied when M is chosen as the smallest prime $\geq N$ (for $N>3$).

Theorem 3.3([30]). In order to have conflict-free access to rows, columns and non-circulant diagonals and anti-diagonals using a linear skewing scheme, the smallest number of memory banks required is

$$M = \begin{cases} N & \text{if } 2 \nmid N \text{ and } 3 \nmid N \\ N+1 & \text{if } 2 \mid N \text{ and } N \equiv 0,1 \pmod 3 \\ N+2 & \text{if } 2 \nmid N \text{ and } 3 \mid N \\ N+3 & \text{if } 2 \mid N \text{ and } N \equiv 2 \pmod 3 \end{cases}$$

Moreover, it is possible to achieve this in all cases using the scheme $s(i,j) = i+2j \pmod M$.

Note that theorem 3.3. extends the observation of Budnik & Kuck [3] (see also Lawrie [14]) that there is no linear skewing scheme to store an N×N matrix in N memory banks and have the desired types of conflict-free access when N is even.

Clearly different conditions will arise when the set of vectors of interest is changed. We shall study the case of obtaining conflict-free access to rows, columns and full circulant diagonals and anti-diagonals. (Note that the latter no longer are "d-ordered".) Shapiro [20] has taken an even more general approach.

Definition. An (x,y)-line in an N×N matrix a is any N-vector of elements $a_{c+yj \pmod N, d+xj \pmod N}$ with $0 \leq j < N$, some c and d.

Theorem 3.4([21]). There exists a linear skewing scheme using N memory banks that provides conflict-free access to every (x,y)-line with $(x,y) \in \{(x_i,y_i) \mid 1 \leq i \leq k\}$ if and only if there exist integers a and b such that for all $1 \leq i \leq k$: $\gcd(ax_i+by_i,N) = 1$.

It easily follows that no valid linear skewing scheme for conflict-free access to rows, columns, circulant diagonals and anti-diagonals with M=N exists when 2|N or 3|N. (We will see shortly that a scheme does exist in all other cases.) Of great interest is the following result of Shapiro [22] that requires a deep analysis.

Theorem 3.5([22]). If there exists an (arbitrary!) skewing scheme using N memory banks at all that provides conflict-free access to every (x,y)-line with (x,y) belonging to a given set L, then there exists a linear skewing scheme using N memory banks that is conflict-free on these lines.

The case of M=N that we consider here has interesting connections to other mathematical problems. In the statistical analysis of experiments any assignment of the numbers 1 to N to the cells of an N×N matrix such that (in our terminology) conflict-free access is provided to rows, columns and all circulant diagonals and anti-diagonals, is called a Knut Vik design (after Vik [27]). In 1973 Hedayat & Federer [10] showed that no Knut Vik designs exist for N even, and in 1975 Hedayat [9] completed the analysis and proved that Knut Vik designs of order N exist if and only if $2 \nmid N$ and $3 \nmid N$. Shapiro [22], on the other hand, observed that the condition of conflict-free access to rows, columns and all circulant diagonals and anti-diagonals can be translated in terms of the positioning of (non-attacking) "superqueens" on a chessboard. Superqueens were introduced as early as 1918 by Polya [18] and defined to have the normal attacking moves of a chess queen extended by "wrap around". Shapiro [22] proved essentially the same result as Hedayat's (but independently of it) in his "Polya Superqueen Theorem": N non-attacking superqueens can be placed on an N×N chessboard if and only if $2 \nmid N$ and

$3 \not| N$. By Theorem 3.5. it follows that in this case there will even be a linear skewing scheme. Interestingly enough the construction of Hedayat [9] did exactly provide such a scheme, using some observations due to Euler [5].

Theorem 3.6([9], [22]). There exists a (proper) linear skewing scheme using M=N memory banks that provides conflict-free access to rows, columns and all circulant diagonals and anti-diagonals if and only if $2 \not| N$ and $3 \not| N$.

For all other values of N we will need a number of memory banks strictly larger than N to have the same effect. Wijshoff & van Leeuwen [30] show that M can be bounded by the smallest prime >2N+1 (when $3 \not| N$ then the smallest prime $\geq 2N+1$ will do).

The more practical case when M<N and vectors must retrieved by multiple (conflict-free) fetches has been studied in [30].

Theorem 3.7([30]). Given N and M, there exists a linear skewing scheme to store an N×N matrix into M memory banks such that every rookwise connected template of t cells can be retrieved by means of at most $\lfloor t/\sqrt{M} \rfloor + 1$ conflict-free fetches of vectors from the M memory banks.

4. Periodic skewing schemes. Originally Shapiro [20] defined "periodic skewing schemes" through the requirement that they be determined by some K×K table to which all arguments (i,j) are reduced modulo K. We now know that the proper approach is through the theory of integral lattices. Let s be determined by the lattice L generated by the vectors $u = (u_1,u_2)$ and $v = (v_1,v_2)$, and let

$$\Delta(L) = \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}$$

be the (absolute value of) the determinant of L. A fundamental relationship is expressed in the following result.

Theorem 4.1([29]). The number of memory banks used by a periodic skewing scheme is equal to the determinant of its underlying lattice, i.e., $M=\Delta(L)$.

We shall see in a moment how the theory of integral lattices is applied further.

Theorem 4.2([31]). Every linear skewing scheme is periodic.

There are several indications for the central role of periodic skewing schemes in the general theory. If one wants to achieve conflict-free access to (x,y)-lines by some skewing scheme, then theorem 3.5. (due to Shapiro [22]) expresses that it must be possible using a linear, hence a periodic skewing scheme. It appears that also for many other templates that can be skewed there exists in fact a valid periodic skewing scheme. Define a polyomino to be any rookwise connected arrangement (template) of cells with no "holes". By a deep analysis Wijshoff & van Leeuwen [28] were able to confirm the following conjecture of Shapiro [20].

Theorem 4.3([28]). Let T be a polyomino. There exist a valid (arbitrary) skewing scheme for T using |T| memory banks if and only if there exists a valid periodic skewing scheme for it using |T| memory banks.

Combined with theorem 2.4. it follows that whenever a polyomino T tessellates the plane it can tessellate the plane periodically, i.e., with the instances of T arranged according to a lattice. This fact leads to an interesting corollary.

Theorem 4.4([28]). There exists a polynomial time algorithm to determine whether a polyomino tessellates the plane or not. (Recall that no rotations or reflections of the polyomino are allowed.)

Using elementary lattice theory it is possible to prove a kind of converse to theorem 4.3. and to link tessellations and periodic skewing schemes in a strong sense.

Theorem 4.5([29]). For every periodic skewing scheme s using M memory banks there exists a data template T of size M such that s is valid for T (or: T tessellates the plane periodically with the underlying lattice of s). T can in fact be chosen to be a rectangle.

The intriguing conclusion is that polyominos that tessellate the plane are always equivalent to a rectangle, modulo the proper lattice.

Linear skewing schemes have the desirable property that they are extremely easy to evaluate. (We have only shown the mapping to memory banks, but the assignment of addresses within the banks follows a simple algorithm as well.) For periodic skewing schemes this is not necessarily much harder. Let the lattice of s be generated by the vectors $u = (u_1,u_2)$ and $v = (v_1,v_2)$, and assume the lattice is strictly 2-dimensional (i.e., non-degenerate).

Theorem 4.6([31]). A periodic skewing scheme s is essentially linear (i.e., linear after a suitable renumbering of the memory banks) if and only if $(u_1,u_2,v_1,v_2) = 1$.

The proof involves some elementary number theory. For general periodic skewing schemes there exist fairly simple evaluation methods as well. The following result reconciles our notion of periodicity using geometric lattices with Shapiro's.

Theorem 4.7([29]). Every periodic skewing scheme s using M memory banks can be completely described by an M×M table a such that s(i,j) = a[i mod M,j mod M].

Thus a table of size M^2 is sufficient to "drive" any periodic skewing scheme. To do better, observe from theorem 4.5. that every lattice has a fundamental region that is a rectangle. It has the "natural" form of a table and by theorem 4.1. it must have size M. One can show that the rectangle can be chosen to be of size $\gcd(u_2,v_2)$ by $M/\gcd(u_2,v_2)$ using the earlier notation.

Theorem 4.8([29]). Every periodic skewing scheme s using M memory banks can be completely described by a table of size M and a simple look-up procedure.

The look-up procedure is given in [29] and essentially performs the required reduction modulo the lattice to the fundamental rectangle.

5. Conclusion. Parallel algorithms require that processors and memories communicate data at great speed. Depending on the architecture of a machine, data may be communicated over an interconnection network or through a data channel to and from the processor section. To achieve optimal simultaneity of the transports, parallel algorithms heavily depend on a suitable initial and runtime distribution of the data over the processors and/or the memories. In systolic algorithms the data are more or less rhythmically passed among the processors but in vector machines entire blocks (vectors) of data must be stored in and retrieved from a specified number of memory banks. Suitable segmentations and data mappings are crucial to the success of these machines, together with the availability of algorithms that operate efficiently on the parallelized data sets. As almost no two high-speed parallel computers are similar from the architectural viewpoint, non-trivial machine-dependencies characterize the domain of "vectorization". Some machines (such as the Cyber 205) need relatively large start-up times and thus become competitive with the larger "sequential" computers only when sufficiently large vectors are manipulated. In this paper we have shown some of the results that are emerging towards a general understanding of (static) parallel data structures.

6. References.

[1] Amdahl, G.M., Validity of a single processor approach to achieving large scale computing capabilities, Spring Joint Comp. Conf. 1967, AFIPS Conf. Proc. vol. 30, 1967, pp. 483-485.

[2] Böhm, A.P.W., Dataflow computation, Ph.D. Thesis (in progress), Dept. of Computer Science, University of Utrecht, Utrecht, 1983.

[3] Budnik, P. and D.J. Kuck, The organisation and use of parallel memories, IEEE Trans. Comput. C-20 (1971) 1566-1569.

[4] Chor, B., C.E. Leiserson and R.L. Rivest, An application of number theory to the organisation of raster-graphics memory, Conf. Rec. 23rd Ann. IEEE Symp. Found. Computer Sci., Chicago, 1982, pp. 92-99.

[5] Euler, L., Recherches sur une nouvelle espece des quarrés magiques, Verh. Zeeuwsch Gen. Wetensch. Vlissingen 9 (1782) 85-239.

[6] Flynn, M.J., Some computer organisations and their effectiveness, IEEE Trans. Comput. C-21 (1972) 948-960.

[7] Hardy, G.H. and E.M. Wright, An introduction to the theory of numbers, 5th ed., Clarendon Press, Oxford, 1979 (1st ed. 1938).

[8] Haynes, L.S., R.L. Lau, D.P. Siewiorek and D.W. Mizell, A survey of highly parallel computing, IEEE Computer Magaz. Januari 1982, pp. 9-24.

[9] Hedayat, A., A complete solution to the existence and non-existence of Knut Vik designs and orthogonal Knut Vik designs, J. Combin. Th., Ser. A, 22 (1977) 331-337.

[10] Hedayat, A. and W.T. Federer, On the non-existence of Knut Vik designs for all even orders, Ann. Stat. 3 (1975) 445-447.

[11] Hockney, R.W. and C.R. Jesshope, Parallel computers, Hilger, Bristol, 1981.

[12] Kuck, D.J., ILLIAC IV software and application programming, IEEE Trans. Comput. C-17 (1968) 758-770.

[13] Kung, H.T., The structure of parallel algorithms, Techn. Rep. CMU-CS-79-143, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, 1979.

[14] Lawrie, D.H., Access and alignment in an array processor, IEEE Trans. Comput. C-24 (1975) 1145-1155.

[15] Lawrie, D.H. and C.R. Vora, The prime memory system for array access, IEEE Trans. Comput. C-31 (1982) 435-442.

[16] Lorin, H., Parallelism in hardware and software: real and apparent concurrency, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1972.

[17] Montoye, R.K. and D.H. Lawrie, A practical algorithm for the solution of triangular systems on a parallel processing system, IEEE Trans. Comput. C-31 (1982) 1076-1082.

[18] Pólya, G., Über die 'doppelt-periodischen' Lösungen des n-Damenproblems, in: W. Ahrens (ed.), Mathematische Unterhaltungen und Spiele, Teubner, Leipzig, 1918, pp. 364-374.

[19] Schwartz, J., Large parallel computers, J. ACM 13 (1966) 25-32.

[20] Shapiro, H.D., Theoretical limitations on the efficient use of parallel memories, IEEE Trans. Comput. C-27 (1978) 421-428.

[21] Shapiro, H.D., Storage schemes in parallel memories, 1975 Sagamore Comp. Conf. Parall. Proc., 1975, pp. 159-166.

[22] Shapiro, H.D., Generalized Latin squares on the torus, Discr. Math. 24 (1978) 63-77.

[23] Stone, H.S., Parallel computers, in: H.S. Stone (ed.), Introduction to computer architecture, SRA Inc., Chicago, 1980 (2nd ed.), pp. 363-425.

[24] Thurber, K.J., Large scale computer architecture: parallel and associative processors, Hayden Book Comp., Rochelle Park, New Jersey, 1976.

[25] Thurber, K.J. and P.C. Patton, The future of parallel processing, IEEE Trans. Comput. C-22 (1973) 1140-1143.

[26] van Leeuwen, J., Distributed computing, in: J.W. de Bakker and J. van Leeuwen (eds.), Foundations of Computer Science IV, MC Tract 158 (part 1), Mathem. Centre, Amsterdam, 1982, pp. 1-34.

[27] Vik, K., Bedømmelse av feilen på forsoksfelter med of uten malestokk, Meldinger fra Norges Landbrukshøgskole 4 (1924) 129-181.

[28] Wijshoff, H.A.G. and J. van Leeuwen, Periodic versus arbitrary tessellations of the plane using polyominos of a single type, Techn. Rep. RUU-CS-82-11, Dept. of Computer Science, University of Utrecht, Utrecht, 1982 (revised version to appear in Inf. Control).

[29] Wijshoff, H.A.G. and J. van Leeuwen, Periodic storage schemes with a minimum number of memory banks, Techn. Rep. RUU-CS-83-4, Dept. of Computer Science, University of Utrecht, Utrecht, 1983.

[30] Wijshoff, H.A.G. and J. van Leeuwen, On linear skewing schemes and d-ordered vectors, Techn. Rep. RUU-CS-83-7, Dept. of Computer Science, University of Utrecht, Utrecht, 1983.

[31] Wijshoff, H.A.G. and J. van Leeuwen, A linearity condition for periodic skewing schemes, Techn. Rep. RUU-CS-83-10, Dept. of Computer Science, University of Utrecht, Utrecht, 1983.