ICPS

Informatique et
Calcul Parallèle
de Strasbourg

Publication 94-15

An efficient Allocation Strategy for Mapping affine Recurrences into space and time-optimal regular Processor Arrays

Ph. Clauss

Published in Parcella'94, September 21-23, 1994, Potzdam (Germany)

ICPS - Université Louis Pasteur
Pôle API, Boulevard Sébastien Brant, F-67400 Illkirch

An Efficient Allocation Strategy for Mapping Affine Recurrences into Space and Time Optimal Regular Processor Arrays

Philippe CLAUSS
IECS, University Robert Schuman
67082 Strasbourg FRANCE
E-mail: clauss@caseb.u-strasbg.fr

Abstract

This paper adresses the problem of efficient mappings of nested loops, and more generally of systems of affine recurrence equations, into regular arrays. The presented technique is based on the transformation of an initial systolic mapping. By studying the processor element (PE) activity, a nearly space-optimal mapping is designed by serializing the computations of several initial PEs into a single one. A new approach of this technique is presented. It allows to map into regular arrays which may be nonplanar. Moreover, the solutions can be linearly time-optimal since this technique does not affect the initial schedule. The methodology is illustrated by the LL^T Cholesky factorization.

1 Introduction

Distributed memory multiprocessors are attractive to their scalability, flexibility and performance but suffer from a lack of efficient methods for mapping algorithms into such parallel architectures. A large number of algorithms in signal and data processing applications can be written in the form of nested loops [1] and more generally in the form of systems of affine recurrence equations [2]. The problem of the concurrent execution of recurrence equations was first adressed by Karp et al. [3] and Lamport [4]. Later on, many works [5, 6, 7, 8] on the design of systolic arrays, proposed a linear transformation of the iteration space into a $space \times time$ processor space.

One of the major drawbacks of these methods is the low processing element (PE) utilization occurring in the solutions. Some further works focused on this problem in order to increase the efficiency of the solutions. But these techniques either apply only on systolic arrays whose PEs are not active at any instant between their first and last calculations, i.e. slower than the global clock (clustering [9, 10, 11]), or give in the general case irregular solutions with non-local and crossing connections (serializing [9, 12]).

The technique proposed in this paper allows to map recurrence equations into regular processor arrays which may be nonplanar such as cylindrical, orbital, and more generally toroidal. These arrays have regular topologies in the sense that any PE communicates only with its neighbours, except in the case of toroidal arrays where some boundary PEs may communicate with PEs of some opposite boundary in a regular manner : all these connections between boundary PEs are constant for any PEs of the same boundary. Such arrays have been proven to be superior to planar arrays in many cases : they offer better PE utilization, increased speed and reduced I/O bandwith requirements [13, 12]. The designed solutions are space-optimal in many cases or nearly space-optimal. Moreover they can be linearly time-optimal since our mapping technique does not affect the initial linear schedule.

Our method must be distinguished from some recent works focusing on the mapping of recurrences or nested loops into regular arrays. These studies correctly map algorithms into a mesh connected array. A good utilization of the PEs can be obtained but generally by empirical ways. For example, Peir and Cytron in [2], D'Hollander in [14], propose a partitioning into independent groups of variable instances and then an allocation of each group into a processor space of the same dimension. Unfortunately many recurrences (such as Cholesky factorization, LU decomposition, Gauss elimination, etc.), yield only one partition. The resulting array is then a mesh connected cube containing the whole index set. In this case our method applies and leads to a processor space of dimension $n-1$ for a problem of dimension n. Last, some other works as those of Barada and El-Amawy [15] propose a methodology for mapping algorithms into nonplanar time-optimal VLSI arrays, but these arrays have the same number of PEs as a classic systolic architecture designed by any well-known synthesis method, and therefore are not space-optimal.

2 Systems of Affine Recurrence Equations

Recurrence equations can be considered as abstract forms of nested loops, in which the only dependencies are flow-dependencies, since such statements apply the single assignment rule. We assume that dependencies are affine functions of the iteration vector and the iteration space is a convex polytope of Z^n (n is the depth of the nest).

Definition 1 *An affine recurrence equation is of the form* $X(\vec{z}) = f(..., Y(g(\vec{z})), ...)$, *where* $\vec{z} \in Dom(X) \subset Z^n$, *$g$ is an affine function from Z^n to Z^m and f has a constant computation time.* [1]

The vertices of the extended dependence graph associated with equations are the integral points inside the iteration space Dom, and the dependencies can be defined by dependence vectors $\vec{d} = \vec{z} - g(\vec{z})$ in Z^n. A space-time allocation consists in transforming problem statements by substituting some $S \times T$ space for Dom, where S is a

[1] In the following, Dom is the union of the $Dom(X)$ for all variables X.We assume that variables are fully indexed on Z^n and that Dom is a convex polytope depending on some size parameters.

$(n-1)$-dimensional space and T is a 1-dimensional space. The computation of S lies on a projection direction $\vec{\xi}$: the allocation is then characterized by a linear function $alloc(\vec{z}) = \sigma\,\vec{z}$, which maps the iteration space onto the processor space S. The allocation matrix σ is a $(n-1, n)$-matrix determined from a pivoting algorithm which computes an orthogonal basis containing $\vec{\xi}$. Concerning the schedule, an affine function defines, for any point of the iteration space associated with an iteration vector $\vec{z}$, its execution instant $t(\vec{z}) = \vec{\lambda}\cdot\vec{z} + \alpha$. In this paper we only consider practical cases where problems can be scheduled by a unique affine function, i.e. the schedule does not depend on the variable X. Vector $\vec{\lambda}$ is orthogonal to hyperplanes of Z^n, the *timing surfaces*, whose points have the same execution instant. So, the space $\times$ time transformation process is defined from linear functions. It is expressed as matrix $\begin{pmatrix} \vec{\lambda} \\ \sigma \end{pmatrix}$. The validity of such a transformation is proved by applying these matrices on the data dependencies expression : a schedule must satisfy $\vec{\lambda}\cdot\vec{d} > 0$ for any dependence vector $\vec{d}$, and the allocation is such that $\vec{\lambda}\cdot\vec{\xi} > 0$, since all the points of *Dom* aligned on $\vec{\xi}$ are executed into the same PE and, therefore, cannot be executed at the same instant.

Example *The Cholesky factorization.*
Let A be a $N \times N$ symmetric positive definite matrix. The Cholesky factorization calculates a lower triangular matrix L such that $A = LL^T$. It is defined by the following system of affine recurrence equations :

$$\begin{aligned}
L(i,j,k) &= L(i,j,k-1) - L(i,k+1,k) \times L(j,k+1,k) \\
&\qquad\qquad\qquad 1 \le i \le N, \quad 1 \le j \le i, \quad 0 \le k \le j-2 \\
L(i,j,k) &= L(i,j,k-1)/L(j,j,j-1) \\
&\qquad\qquad\qquad 1 \le i \le N, \quad 1 \le j \le i-1, \quad k = j-1 \\
L(i,j,k) &= L(i,j,k-1)^{1/2} \quad 1 \le i \le N, \quad j = i, \quad k = i-1
\end{aligned}$$

The input equation is $L(i,j,-1) = A_{ji},\ 1 \le i \le N, 1 \le j \le i$, and the iteration space is $\{(i,j,k) \in Z^3 \mid 1 \le i \le N,\ 1 \le j \le i,\ 0 \le k \le j-1\}$. Regarding dependencies a systolic solution can be synthesized. It is characterized by schedule $\vec{\lambda} = (1,1,1)$ and allocation direction $\vec{\xi} = (1,0,0)$ defining for example allocation matrix $\sigma = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. □

Such arrays present many qualities of regularity and locality of the connections. However, by studying PEs' activity through the geometrical modeling described in the next section, it is shown that the disadvantage of most systolic mappings is that they do not have an optimal parallelism rate. Moreover, the program computed by each PE may have a too much fine grain for a mapping onto a distributed memory computer. The performances and the memory storage capabilities of such array processors allow more sophisticated programs. That is the reason why, in order to keep their qualities but overcome these disadvantages we aim to map these solutions on more general distributed memory networks with the objective of minimizing the virtual processor count and keeping the regularity. For this purpose we present in the next section a geometrical modeling of the PE activity.

3 Geometrical modeling of a systolic design

The aim of this section is to recall some notions about a geometrical modeling of PEs' activity. The reader can refer to [16, 9] for further investigations. The following notions come from well-known mathematical results [17] :

- Assuming the iteration space Dom is a convex polytope of Z^n defined by a non redundant set of integral inequations $\{\vec{z} \in Z^n \mid A\vec{z} \leq \vec{b}\}$, where A is totally unimodular, any line A_i of matrix A defines the normal vector to a facet, which is contained in the hyperplane which saturates the i^{th} constraint, i.e. defined by the equation $A_i \cdot \vec{z} - b_i = 0$,

- By projecting Dom along an allocation direction $\vec{\xi}$, the activity of any resulting PE is defined by the points belonging to the same *allocation segment* defined as $Dom_s = \{\vec{z} \in Dom \mid alloc(\vec{z}) = s\}$. Extremities of these segments characterize facets of Dom : they correspond either to the first or to the last computation on this PE. However it may happen that these facets belong to several parallel integer hyperplanes. In this situation they define *thick facets.* To simplify the presentation, we consider from now on that there are no thick facets.

Definition 2 *Assuming the schedule and the allocation are such that* $\vec{\lambda} \cdot \vec{\xi} > 0$*, we call* first facets *of the iteration space Dom, facets whose points* $\vec{z}$ *are such that* $\vec{z} - \vec{\xi} \notin Dom$*. We call* last facets *of the iteration space Dom, facets whose points* $\vec{z}$ *are such that* $\vec{z} + \vec{\xi} \notin Dom$*.*

The intersections between a timing surface and the first (respectively last) facets characterize all the PEs having their first (respectively last) activation at the same instant. These intersections are either faces of dimension $(n-2)$, or the facets themselves. In the following we consider the former case.

We denote as F_i (respectively L_j) a first (respectively last) facet defined by equation $A_i \cdot \vec{z} - b_i = 0$ (respectively $A_j \cdot \vec{z} - b_j = 0$), and call it the i^{th} first facet (respectively the j^{th} last facet). The intersection of this facet with the timing surface at instant t_0 is denoted as F_{i,t_0} (respectively L_{j,t_0}). Their union for all first and last facets are respectively denoted as F_{t_0} and L_{t_0}. Projections of these sets F_{t_0} and L_{t_0} along the allocation direction $\vec{\xi}$, on the initial architecture, define the activity period of every PE. More precisely we introduce the concept of activity cone.

Definition 3 *Hyperplanes* $alloc(F_{t_0+\vec{\lambda}\cdot\vec{\xi}})$ *and* $alloc(L_{t_0})$ *define the so called* activity cones *of the array. They delimit three areas of PEs at any instant* t_0 *: the dead PEs, the possibly active ones and the previously unused ones.*

This geometrical characterization of active PEs is going to be used to define a specific allocation technique whose objective is to map the systolic solution into regular arrays with minimization of the processor count.

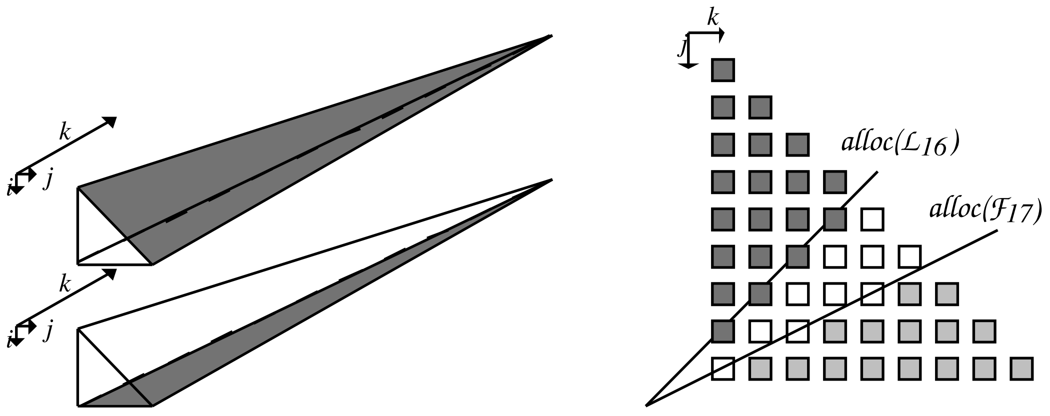

Figure 1: The Cholesky factorization

Definition 4 *For given schedule, we call* parallelism rate *the maximum number of simultaneous calculations* $pp = max_{1 \leq t_0 \leq \tau}\ p(t_0)$*, where* $p(t_0)$ *is the number of simultaneous calculations at instant* t_0 *and* τ *is the latency of the algorithm.*

The value $p(t_0)$ is determined from the first and last activity instant of every PE [16, 9].

Example *The Cholesky factorization (continued).*
The allocation direction $\vec{\xi} = (1, 0, 0)$ defines one first facet F and one last facet L (in this order for the considered schedule $\vec{\lambda} = (1, 1, 1)$) represented in Fig. 1. Inequations defining Dom characterize their normal vectors, respectively $(-1, 1, 0)$ and $(1, 0, 0)$. Sets F_{t_0} and L_{t_0} are lines generated by vectors $(1, 1, -2)$ and $(0, 1, -1)$. The resulting activity cone is then defined by its extremal vectors $(1, -2)$ and $(1, -1)$ and shifts along vector $(0, 1)$ at each time step. The parallelism rate, depending on N, can be determined from these results. For $N = 9$ we have $pp = 12$. $\square$

4 The serializing technique

The basic ideas of this technique have been presented in [9]. A more formal and constructive approach is developped in the following. Serializing consists in mapping into a single PE, the computations of PEs which never belong simultaneously to the set of active PEs, delimited by the cones on the initial processor space. This technique can systematically give space-optimal solutions [9]. An empirical serialization of the initial PEs often leads to arrays with non-local and crossing connections, as shown in Fig. 2 in a case of only one cone. Indeed the mapping generally results in a complete network and, if the serializing directions are not parallel or constant, it yields hard problems in the architecture drawing.

Conversely, our constructive approach of serializing comes from the observation that if the serializing directions are parallel and constant, then the resulting array is a

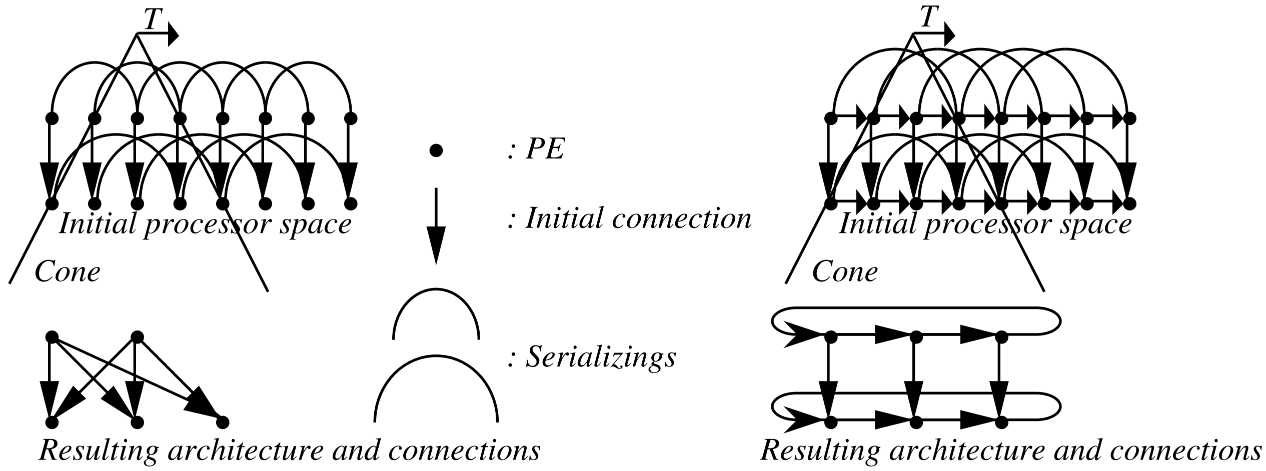

Figure 2: Empirical and constructive serializings

regular toroidal array. The toroidal connections are generated by serializing the initial connections occuring between PEs of the same serializing direction, as shown in Fig. 2. Notice that the number of PEs in the resulting architecture has to be increased : each constant serializing direction results in the same number of PEs. More formally, the constructive approach can be stated as follows :

- Partition the systolic mapping into a set of np parallel lines π_q of PEs,
- For each line compute the maximum count $ppp(\pi_q)$ of simultaneously active PEs,
- Compute the maximum of these values, $ppp = MAX_q\ ppp(\pi_q)$,
- For each line, serializing of length ppp.

Since the systolic array results from the application of the allocation function *alloc* to the iteration space *Dom* of dimension n, this initial architecture is represented in a space of dimension $n-1$. Therefore, the PEs of any given line π_q result from the projection of a 2-dimensional hyperplane of points cutting *Dom*. This hyperplane is generated by vectors whose projection is the same $(n-1)$-dimensional vector generating the lines π_q. It intersects with the first and last facets of *Dom*. These intersections are line segments generated by n-dimensional vectors denoted respectively by $\vec{\pi}_{F_i}$ and $\vec{\pi}_{L_j}$. The application of the allocation function *alloc* to these vectors gives the vector generating the lines π_q, and denoted by $\vec{\pi} = alloc(\vec{\pi}_{F_i}) = alloc(\vec{\pi}_{L_j})$.

Therefore, the first step of our constructive approach consists in determining a best fitted vector $\vec{\pi}$ characterizing the partitioning of the initial systolic mapping. In order to compute $ppp(\pi_q)$, we use the following proposition allowing to determine the vectors $\vec{\pi}_{F_i}$ and $\vec{\pi}_{L_j}$ from any vector $\vec{\pi}$.

Proposition 1 *Vectors* $\vec{\pi}_{F_i}$, $\vec{\pi}_{L_j}$ *and* $\vec{\pi}$ *are such that*

$$\vec{\pi}_{F_i} = \begin{pmatrix} \sigma \\ A_i \end{pmatrix}^{-1} \cdot \begin{pmatrix} \vec{\pi} \\ 0 \end{pmatrix} \qquad \vec{\pi}_{L_j} = \begin{pmatrix} \sigma \\ A_j \end{pmatrix}^{-1} \cdot \begin{pmatrix} \vec{\pi} \\ 0 \end{pmatrix}$$

for any i^{th}*-first and any* j^{th}*-last facets.*

Definition 5 *We call sub-line* π_q^i *the segment of PEs of the line* π_q *resulting from the projection of points belonging to the first facet* F_i : $\pi_q = \bigcup_{1 \le i \le f} \pi_q^i$ *where* f *is the number of first facets. We define in the same way sub-lines* π_q^j *resulting from the projection of points belonging to the last facet* L_j.

Proposition 2 *The number of PEs of line* π_q *having their first activation before instant* t_0 *(included) is*

$$\begin{array}{ll} \forall\ t_0 \in [1 \ldots \vec{\lambda} \cdot \vec{z}_{max_0}[& nf_q(t_0) = 0 \\ \forall\ t_0 \in [\vec{\lambda} \cdot \vec{z}_{min_i} \ldots \vec{\lambda} \cdot \vec{z}_{max_i}[& nf_q(t_0) = (t_0 - \vec{\lambda} \cdot \vec{z}_{min_i}) \div \vec{\lambda} \cdot \vec{\pi}_{F_i} \\ & \qquad + nf_q(\vec{\lambda} \cdot \vec{z}_{max_{i-1}} - 1) + 1 \\ \forall\ t_0 \in [\vec{\lambda} \cdot \vec{z}_{max_f} \ldots \tau[& nf_q(t_0) = nf_q(\vec{\lambda} \cdot \vec{z}_{max_f} - 1) + 1 \end{array}$$

where $\forall\, i \in [1 \ldots f]$, $\vec{z}_{min_i}$ *and* $\vec{z}_{max_i}$ *are points in the first facet* F_i *allocated into the two extremal PEs* pe_{min_i} *and* pe_{max_i} *of sub-line* π_q^i, *i.e.*

$$\vec{z}_{min_i} = \begin{pmatrix} \sigma \\ A_i \end{pmatrix}^{-1} \cdot \begin{pmatrix} pe_{min_i} \\ b_i \end{pmatrix} \qquad \vec{z}_{max_i} = \begin{pmatrix} \sigma \\ A_i \end{pmatrix}^{-1} \cdot \begin{pmatrix} pe_{max_i} \\ b_i \end{pmatrix}$$

Proposition 3 *The maximum number of simultaneously active PEs in a line* π_q *is equal to* $ppp(\pi_q) = MAX_{1 \le t_0 \le \tau}\ (nf_q(t_0) - nl_q(t_0 - 1))$.

The third step of our constructive approach can now be achieved by computing for all the lines the maximum of these values, $ppp = MAX_q\ ppp(\pi_q)$. Notice that the number of resulting PEs is equal to $np \times ppp$. Therefore, direction $\vec{\pi}$ can be chosen in order to optimize this quantity. The determination of a direction $\vec{\pi}$ minimizing the number of lines can be made by resolving the following mixed linear integer programming problems :

$$min_{1 \le p \le n-2}\ \{MAX_{(\vec{z},\vec{z}') \in Dom(p)}\ (\vec{z} - \vec{z}') \cdot \vec{\pi}_p^{\perp} \mid \forall\ q < p, \vec{\pi}_p^{\perp} \cdot \vec{\pi}_q^{\perp} = 0 \mid \vec{\xi} \cdot \vec{\pi}_p^{\perp} \neq 0\} \qquad (1)$$

where $Dom(p)$ is defined by

$$\begin{cases} Dom(1) = Dom \\ Dom(p) = \{(\vec{z}, \vec{z}') \in Dom \mid \forall\ q \in [1 \ldots p-1], (\vec{z} - \vec{z}') \cdot \vec{\pi}_q^{\perp} = 0\} \end{cases}$$

More intuitively, $\vec{\pi}_q^{\perp}$ are the $(n-2)$ vectors orthogonal to any 2-dimensional hyperplane whose projection into the processor space results in a line π_q. Therefore, the whole problem of minimizing $np \times ppp$ consists in resolving the following non-linear integer programming problem :

$$min\{np \times ppp \mid (1) \mid ppp = MAX_q\ ppp(\pi_q)\} \qquad (2)$$

Notice that (1) consists in the composition of linear programs which can be transformed and optimized by using the duality theorem of linear programming [17].

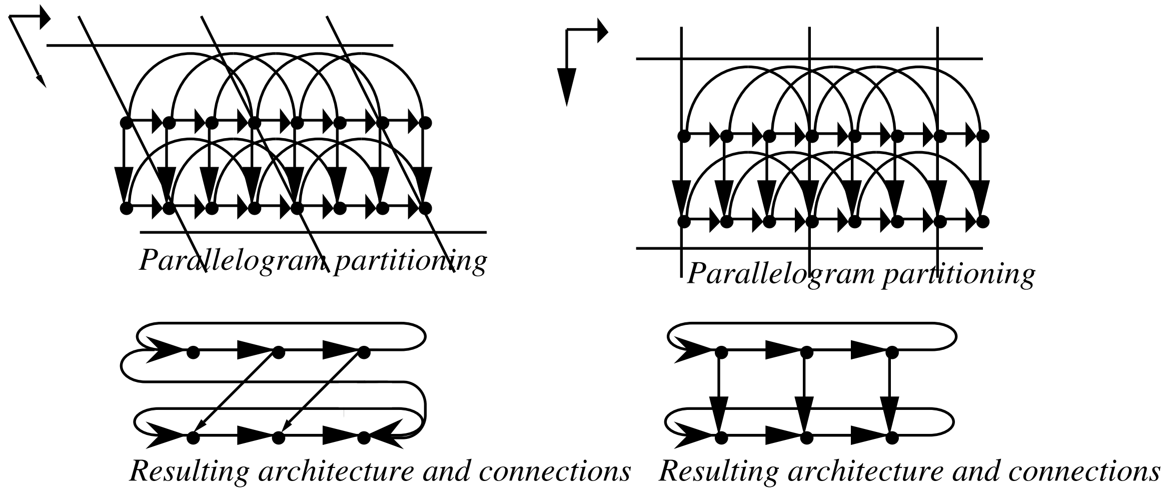

Figure 3: Maximization and minimization of the number of toroidal connections

Definition 6 *A polyhedron P of Z^n is a* parallelotope *if its $2 \times n$ facets constitute n couples of parallel facets characterized by the same edge vectors and by the same area. If $n = 3$ P is a parallelepiped, if $n = 2$ P is a parallelogram.*

The constructive serializing can be seen as a partitioning into equal parallelotopes of area $np \times ppp$. Some of their facets are characterized by the vector $\vec{\pi}$ and one parallelotope defines the architecture solution of the serializing. Concerning the connections between PEs in the final mapping, it has been shown at the beginning of this section that the serializing of initial connections between PEs of a same line π_q results in toroidal connections. The other connections are determined by the choice of the other vectors $\vec{\Theta}_i$ characterizing any parallelotope. In order to minimize the number of toroidal connections, these vectors are chosen equal to the directions of some initial connections as it is shown in Fig. 3 in the case $n = 3$.

Example *The Cholesky factorization (continued).*
The initial systolic mapping is partitioned following the direction $\vec{\pi} = (1, 1)$. The vectors $\vec{\pi}_F$ and $\vec{\pi}_L$ associated with $\vec{\pi}$ are determined :

$$\vec{\pi}_F = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \vec{\pi}_L = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

The functions nf_q and nl_q and the maximum number of simultaneously active PEs per line, $ppp(\pi_q)$ can be computed :

q	1	2	3	4	5	6	7	8	9
$ppp(\pi_q)$	3	3	3	2	2	2	1	1	1

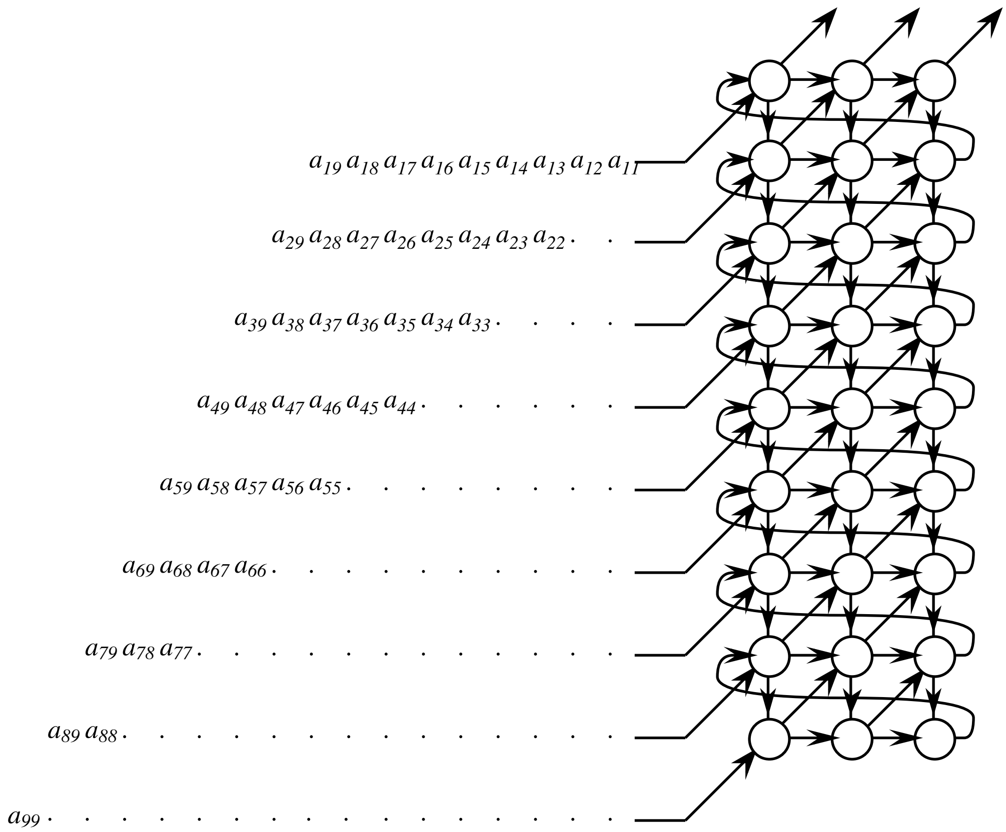

Figure 4: Resulting mapping by serializing the Cholesky factorization

Therefore, for each line, the serializing will be of length $ppp = MAX_q\ ppp(\pi_q) = 3$. The vector $\vec{\Theta}$ is chosen equal to the direction of connections (1,0) in order to minimize the number of toroidal connections. The resulting mapping is presented in Fig. 4. □

5 Conclusion

The mapping method presented in this paper allows to synthesize systematically compromise solutions between regularity and space and time optimality, unlike other nowadays methods which only ensure correct mappings [14, 2]. Moreover, our constructive approach of *serializing* offers a program analysis, suitable for a software tool and could be included in a data-parallel language compiler, for languages such as *High Performance Fortran* or C^*, for example : indeed, this technique allows to optimize the size of the virtual processor grids, mapped into SIMD architectures.

References

[1] S.K. Rao and Th. Kailath. Regular iterative algorithms and their implementation

on processor arrays. *Proc. of IEEE*, 76(4):259–269, 1988.

[2] J.K. Peir and R. Cytron. Minimum distance: a method for partitioning recurrences for multiprocessors. *IEEE Trans. on Computers*, 38(8):1203–1211, aug 1989.

[3] R.M. Karp, R.E. Miller, and S. Winograd. The organization of computations for uniform recurrence equations. *Journal of ACM*, 14(3):563–590, 1967.

[4] L. Lamport. The parallel execution of DO loops. *Comm. ACM*, 17(2):83–93, February 1974.

[5] J.A.B. Fortes and K.S. Fu. Systematic approaches to the design of algorithmically specified systolic arrays. *ICASSP'85*, pages 8.9.1–4, 1985.

[6] Gachet P., Joinnault B., and Quinton P. Synthesizing systolic arrays using diastol. *Int. Workshop on Systolic Arrays*, pages 25–36, 1986.

[7] S.Y. Kung. *VLSI array processors.* Prentice-Hall, 1988.

[8] C. Mongenet. *Une méthode de conception d'algorithmes systoliques : résultats théoriques et réalisation.* PhD thesis, INPL, 1985.

[9] Ph. Clauss, C. Mongenet, and G.-R. Perrin. Calculus of space-optimal mappings of systolic algorithms on processor arrays. *Int. Conf. ASAP, IEEE CS*, pages 591–602, 1990.

[10] J. Bu, E.F. Depreterre, and P. Dewilde. A design methodology for fixed-size systolic arrays. *Int. Conf. ASAP, IEEE CS*, pages 591–602, 1990.

[11] A. Darte. Regular partitioning for synthesizing fixed-size systolic arrays. *Integration, the VLSI Journal*, 12:293–304, 1991.

[12] Ph. Clauss, C. Mongenet, and G.-R. Perrin. Synthesis of size-optimal toroidal arrays for the algebraic path problem : A new contribution. *Parallel Computing*, 18:185–194, 1992.

[13] J.L. Aravena and W.A. Porter. Nonplanar switchable arrays. *Circuits Systems Signal Processing*, 7(1), 1988.

[14] E. D'Hollander. Partitioning and labelling of index sets in DO loops with constant dependence vectors. In *Int. Conf. on Parallel Processing, II*, pages 139–144, 1989.

[15] H. Barada and A. El-Amawy. A methodology for algorithm regularization and mapping into time-optimal vlsi arrays. *Parallel Computing*, 19:33–61, 1993.

[16] Ph. Clauss. *Synthèse d'algorithmes systoliques et implantation optimale en place sur réseaux de processeurs synchrones.* PhD thesis, Univ. de Franche-Comté, 1990.

[17] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.