# Parallelization of the Gaussian Elimination Algorithm on Systolic Arrays

Jean-Claude Bermond, Claudine Peyrat, I. Sakho, Maurice Tchuenté

# Parallelization of the Gaussian Elimination Algorithm on Systolic Arrays

J-C. BERMOND,*[1] C. PEYRAT,*[2] I. SAKHO,†[3] AND M. TCHUENTÉ‡

*I3S, Bât. ESSI, Route des Colles, BP 145, 06903 Sophia Antipolis Cedex, France; †Ecole des Mines, Centre SIMADE Laboratoire Systèmes Coopératifs, 158 Cours Fauriel, 42023 Saint-Etienne Cedex 2, France; and ‡Faculté des Sciences, Laboratorie d'Informatique, BP 812 Yaoundé, Cameroon

**We study the parallel implementation of the Gaussian elimination scheme on systolic arrays. We first show that the time (resp. area) complexity of the algorithm is $T = 3n - 1$ (resp. $S = (n^2/4) + O(n)$), where $n$ is the size of the linear system. Then we exhibit three algorithms. The two first ones are optimal in time. The first one corresponds to an orthogonally connected array of size $3(n^2/8) + O(n)$. The second network is smaller, $S = 3(n^2/10) + O(n)$, but two layers are necessary in order to obtain a regular layout with local communications. The third one is hexagonally connected, has size $(n^2/3) + O(n)$, and is almost optimal in time.**

## 1. INTRODUCTION

### 1.1. Elimination Scheme

The well known Gaussian elimination scheme is, for numerical stability reasons, one of the best methods to solve the linear system $Ax = b$ where $A = [a_{i,j}]$, $1 \le i \le n$, $1 \le j \le n$ and $b = [a_{i,n+1}]$, $1 \le i \le n$.

To do so, we want to triangularize the $n \times (n + 1)$ matrix $[A, b]$ by the Gaussian elimination scheme.

We intend to parallelize the following sequential algorithm: For each column $j$, eliminate all $a_{i,j}$'s for $i > j$ by a combination of the $i$th and the $j$th rows of the $n \times n + 1$ matrix $[A, b]$. Suppose the $k - 1$ first columns are treated and that we have obtained the $n \times (n + 1)$ matrix $A^{k-1}$:

$$\begin{pmatrix} a_{1,1}^{k-1} & a_{1,2}^{k-1} & \cdot & a_{1,k-1}^{k-1} & a_{1,k}^{k-1} & \cdot & a_{1,n}^{k-1} & a_{1,n+1}^{k-1} \\ 0 & a_{2,2}^{k-1} & \cdot & a_{2,k-1}^{k-1} & a_{2,k}^{k-1} & \cdot & \cdot & a_{2,n+1}^{k-1} \\ \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & a_{k-1,k-1}^{k-1} & a_{k-1,k}^{k-1} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 & a_{k,k}^{k-1} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 0 & a_{n,k}^{k-1} & \cdot & a_{n,n}^{k-1} & a_{n,n+1}^{k-1} \end{pmatrix}$$

[1] E-mail: bermond@unice.fr.
[2] E-mail: claudine@essi.fr.
[3] E-mail: sakho@emse.fr.

We now proceed in two steps:

- Step 1: normalization of the $k$th row.
- Step 2: combination for elimination of the $a_{i,k}$'s for $i > k$.

Therefore the algorithm is the following (the $a_{k,k}$'s are supposed nonzero):

```
begin
for i := 1 to n do
        for j := 1 to n + 1 do a^0_{i,j} := a_{i,j};
for k := 1 to n do
        begin
        for j := k to n + 1 do a^k_{k,j} := a^{k-1}_{k,j} / a^{k-1}_{k,k};
        for i := k + 1 to n do
                        for j := k to n + 1
                        do a^k_{i,j} := a^{k-1}_{i,j} - a^{k-1}_{i,k} * a^k_{k,j}
        end
end
```

### 1.2. Problem Complexities

Following the methodology of Miranker and Winkler [6, 7], Moldovan [8], Moreno and Lang [9], and Quinton [10], we start by associating to these recurrence relations a directed graph $G$, called "the space–time diagram," whose set of points is a domain $D^n$ of $N^3$. $D^n$ contains the points $(i, j, k)$ associated to the computation of the $a_{i,j}^k$'s. Therefore $D^n = \{(i, j, k), 1 \le k \le n, k \le i \le n, k \le j \le n + 1\}$.

There is an arc in $G$ from point $x$ to point $y$ if the computation in $y$ uses a variable computed in $x$. The graph $G$ not only represents the computation executed during the elimination scheme but also the paths followed by the variables. Afterward, we determine an optimal time function which gives the minimum date at which the computation corresponding to each point of the domain can be done.

Then, to construct a systolic array, we allocate to each processor some points of $D^n$ so that two points with the same value of the determined optimal time function are not allocated to the same processor.

We determine a lower bound on the minimum number

of processors needed to achieve the computation in optimal time (this is related to the area complexity). Furthermore, in a systolic array, the allocation of the processors must be done in such a way that the interconnection network of the processors has a drawing with all connections local. The interconnection network of the processors is the graph with the processors as vertices and so that there is an arc from processor $x$ to processor $y$ if processor $y$ uses a variable computed by processor $x$. This graph can also be obtained from the space–time diagram $G$ by forgetting the orientation and contracting all the points allocated to the same processor.

## 2. LOWER BOUND ON THE COMPLEXITIES

### 2.1. Construction of the Space–Time Diagram

Each point $(i, j, k)$ of $D^n$ corresponds to a computation executed at fixed $i$, $j$, and $k$, either in a normalization step or in a combinational one.

Note that $D^n = \{1 \le k \le n, k \le i \le n, k \le j \le n + 1\}$.

See Fig. 1 for an example of $G$ when $n = 5$. (Note that the directions on the drawing are not the usual ones and that the orientation of the arcs is missing as they are all directed in increasing order of coordinates.)

In the diagram we use the following representations:

- A white square, located at point $(i, j, k)$ with $i \ge j$ and $j = k$, represents a synchronization point. It receives the value of the variable $w = a_{i,k}^{k-1}$ in the direction (0, 0, 1). It delays it during one time unit before sending it in the direction (0, 1, 0).
- A circle, located at point $(i, j, k)$ with $i = k$ and $j > i$, represents a division point. It receives the values of the variables $w = a_{k,j}^{k-1}$ and $u = a_{k,k}^{k-1}$ respectively in the directions (0, 0, 1) and (0, 1, 0). After the computation of $v = a_{k,j}^{k} := a_{k,j}^{k-1}/a_{k,k}^{k-1}$, the values of $v = a_{k,j}^{k}$ and $u = a_{k,k}^{k-1}$ are sent respectively in the directions (1, 0, 0) and (0, 1, 0).
- A black square, located at point $(i, j, k)$, receives the values of the variables $w = a_{i,j}^{k-1}$, $u = a_{i,k}^{k-1}$ and $v = a_{k,j}^{k}$ respectively in the directions (0, 0, 1), (0, 1, 0), and (1, 0, 0). After the transformation $a_{i,j}^{k} := a_{i,j}^{k-1} - a_{i,k}^{k-1} * a_{k,j}^{k}$, the values of the variables $w' = a_{i,j}^{k}$, $u = a_{i,k}^{k-1}$ and $v = a_{k,j}^{k}$ are sent respectively in the directions (0, 0, 1), (0, 1, 0), and (1, 0, 0).

### 2.2. Construction of the Optimal Time Function

In order to simplify, we will assume in the following that each point of $D^n$ has the same computation time. In practice, this is not exact because the computations are not the same in all the nodes. However, it is possible to assume a synchronous execution by taking as the time unit the maximum of the execution time of the elementary transformations as defined in the previous section (synchronization, accumulation, and normalization)

Clearly, the sequence of computation located on a path of $G$ must be done sequentially. Therefore, the minimal

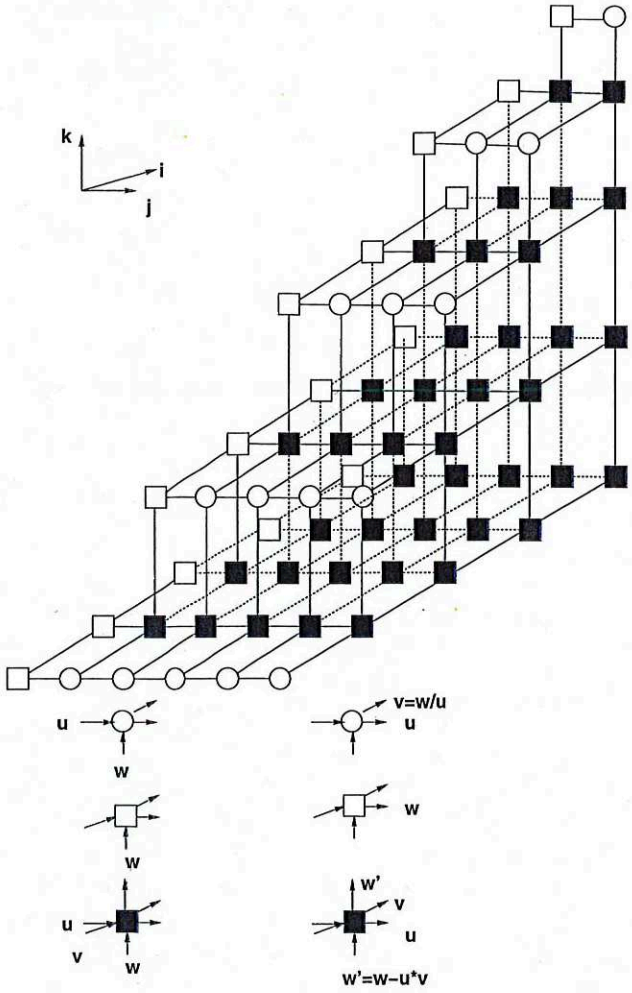


FIG. 1. Space–time diagram for $n = 5$.

time needed to achieve the parallel computation of the elimination scheme is greater than or equal to the number of vertices of a longest path in $G$.

A time function is an application $T$: $D^n \to N$ such that $T(x) < T(y)$ for each arc $(x, y)$ of $G$. This inequality constraint is natural, as it simply comes from the fact that the computation executed in $y$ must be done after the computation in $x$ because it uses variables computed in $x$. A time function is optimal if its maximum value is equal to the number of vertices of a longest path in $G$.

In the following, we are mainly interested in time optimal algorithms, that is, algorithms executed in a minimal time.

PROPOSITION 1. *The time function* $T(i, j, k) = i + j + k - 2$ *is optimal.*

*Proof.* As all the arcs of $G$ are either of the form $((i, j, k), (i + 1, j, k))$ or $((i, j, k), (i, j + 1, k))$ or $((i, j, k), (i, j, k + 1))$, $T$ is obviously a time function.

For every point $x$ of $D^n$, we have $T(x) \le T(n, n + 1, n) = 3n - 1$. Therefore, to show that $T$ is optimal, it suffices to exhibit a path of length $3n - 2$ in $G$. For example, starting from the vertex (1, 1, 1), we can do $n - 1$ steps

in the direction (1, 0, 0), reach the point $(n, 1, 1)$, then with $n$ steps in the direction (0, 1, 0) reach the point $(n, n + 1, 1)$, and finally with $n - 1$ steps in the direction (0, 0, 1) reach the point $(n, n + 1, n)$.

See Fig. 2 for an example with $n = 5$. ■

COROLLARY 2. *The minimum time required to achieve the Gaussian elimination scheme is* $3n - 1$.

We will now prove that the optimal time function is unique.

Let us call critical point each point of $D^n$ which belongs to at least one longest path in $G$.

LEMMA 1. *Let $x$ be a critical point and let $P = x_1, x_2, \ldots, x_r = x, \ldots, x_{3n-1} = y$ be a longest path in $G$ containing $x$ as its $r$th point, then $T(x) = r$.*

*Proof.*

- $T(x) \geq r$ for each time function $T$. Indeed the execution of $x$ must be preceded by the execution of all the $x_i$'s for $0 < i < r$.
- $3n - 1 \geq T(y) \geq T(x) + 3n - 1 - r$. Indeed for any optimal time function, the execution of $y$ must be preceded by the execution of all the $x_i's$, $r \leq i \leq 3n - 2$.

Therefore $T(x) = r$ for every optimal time function $T$. ■

PROPOSITION 3. *For the Gaussian elimination scheme defined in Section* 1, *there is a unique optimal time function.*

*Proof.* According to Lemma 1, we only need to show that each point belongs to at least a longest path in $G$. Let $x = (i, j, k), 1 \leq k \leq i \leq n, k \leq j \leq n + 1$ be a point of $D$.

Let $P$ be the following path: start from (1, 1, 1), then do $i - 1$ steps in direction (1, 0, 0), $j - 1$ steps in direction (0, 1, 0), $k - 1$ steps in direction (0, 0, 1) (at this point we are in $(i, j, k)$), then do $n - i$ steps in direction (1, 0, 0), $n + 1 - j$ steps in direction (0, 1, 0) and $n - k$ steps in direction (0, 0, 1). $P$ is a longest path of $G$ and $x$ belongs to $P$. ■

### 2.3. Complexity in Terms of Processors

We will now determine the minimum number of processors needed to execute the algorithm in optimal time.

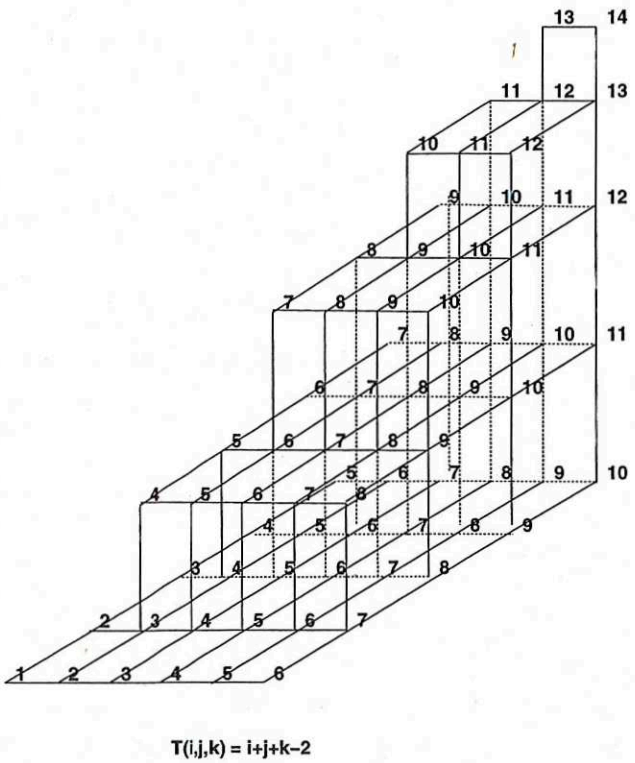


FIG. 2. The optimal time function for $n = 5$.

Let $D^n(t) = \{(i, j, k) \in D^n / i + j + k = t + 2\}$, $m^n(t) = |D^n(t)|$, $M^n = \max_{1 \leq t \leq 3n-1} m^n(t)$.

Let $D_k^n = \{(i, j, z) \in D^n / z = k\}$, $D_k^n(t) = \{(i, j, z) \in D^n(t) / z = k\}$ and $m_k^n(t) = |D_k^n(t)|$.

Clearly, $m^n(t) = \sum_{k=1}^{n} m_k^n(t)$.

Note that $m^n(t)$ is equal to the number of tasks executed at date $t$ with the optimal time function and for a system of size $n$, therefore it is also equal to the minimum number of processors requested at time $t$ in order to achieve the computation in optimal time. Consequently, each parallelization which corresponds to the optimal time function needs at least $M^n$ processors.

EXAMPLE 1. $n = 5$.

| $t$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $m_1^5(t)$ | 1 | 2 | 3 | 4 | 5 | 5 | 4 | 3 | 2 | 1 | | | | |
| $m_2^5(t)$ | | | | 1 | 2 | 3 | 4 | 4 | 3 | 2 | 1 | | | |
| $m_3^5(t)$ | | | | | | | 1 | 2 | 3 | 3 | 2 | 1 | | |
| $m_4^5(t)$ | | | | | | | | | | 1 | 2 | 2 | 1 | |
| $m_5^{5(t)}$ | | | | | | | | | | | | | 1 | 1 |
| $m^5(t)$ | 1 | 2 | 3 | 5 | 7 | 8 | 9 | 9 | 8 | 7 | 5 | 3 | 2 | 1 |

Therefore to execute the Gaussian elimination scheme with $n = 5$ and in time 14, at least nine processors are required.

Similarly, we can compute $m^6(t)$ and $m^7(t)$.

$m^6(t)$: 1 2 3 5 7 9 11 12 12 12 11 9 7 5 3 2 1

$m^7(t)$: 1 2 3 5 7 9 12 14 15 16 16 15 14 12 9 7 5 3 2 1

LEMMA 2. *The sequence $m_k^n(t)$ for $3k-2 \le t \le 2n+k-1$ is a palindrome consisting of the $n-k+1$ first integers $1, 2, \ldots, n-k, n-k+1, n-k+1, n-k, \ldots, 2, 1$. In other words, $m_k^n(t) = t-3k+3$ if $3k-2 \le t \le n+2k-2$, $m_k^n(t) = 2n+k-t$ if $n+2k-1 \le t \le 2n+k-1$,*

*Proof.* Indeed $D_k^n$ is a rectangle starting at point $(k, k, k)$ with $n-k+1$ vertices in the direction $(1, 0, 0)$ and $n-k+2$ vertices in the direction $(0, 1, 0)$.

As the points of $D_k^n(t)$ correspond to a diagonal of slope $-1$, the sequence of the $m_k^n(t)$, $3k-2 \le t \le 2n+k-1$, corresponds to the cardinality of the successive diagonals of $D_k^n$. ■

Figure 3 shows $D_2^5$.

LEMMA 3.

$$m^{n+1}(t) = m^n(t-3) + t \quad \text{for } 1 \le t \le n+1,$$
$$m^{n+1}(t) = m^n(t-3) + 2n + 3 - t \quad \text{for } n+2 \le t \le 2n+2,$$
$$m^{n+1}(t) = m^n(t-3) \quad \text{for } 2n+3 \le t \le 3n+2.$$

*Proof.* The domain $D^{n+1}$ is the union of $D_1^{n+1}$ and the translated $\mathrm{tr}(D^n)$ of the domain $D^n$ with $i \to i+1$, $j \to j+1$ and $k \to k+1$. Therefore $D^{n+1}(t) = D_1^{n+1}(t) \cup \mathrm{tr}(D^n(t-3))$.

Thus $m^{n+1}(t) = m_1^{n+1}(t) + m^n(t-3)$.

Then we get the result by applying Lemma 2. ■

LEMMA 4.

$$m^{n+1}(t) = m^n(t) \quad \text{for } 1 \le t \le n,$$
$$m^{n+1}(t) = m^n(t) + t - n \quad \text{for } n \le t \le 2n+1,$$
$$m^{n+1}(t) = m^n(t) + 3n + 3 - t \quad \text{for } 2n+2 \le t \le 3n+2.$$

*Proof.* $D^{n+1}$ is also the union of $D^n$ plus the set of points $(i, j, k)$ such that at least one of the following conditions is satisfied: $i = n+1$ or $j = n+2$ or $k = n+1$.

The number of points of $D^{n+1}(t)$ with $j = n+2$ is equal to the number of couples $(i, k)$ such that $n+2+i+k = t+2$ with $k \le i \le n+1$, that is, $\lfloor (t-n)/2 \rfloor$ for $n+2 \le t \le 2n+1$ and $\lceil (3n+3-t)/2 \rceil$ for $2n+2 \le t \le 3n+2$.

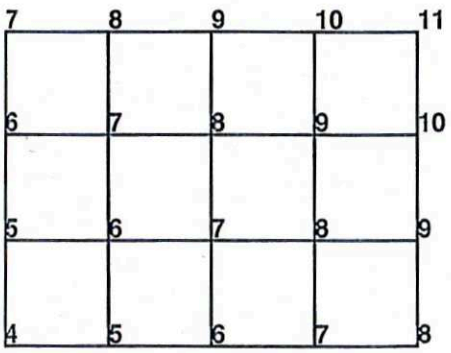


FIG. 3. $D_2^5$.

For example, if $n = 5$, we have the following couples $(i, k)$: $t = 7$ (1, 1); $t = 8$ (2, 1); $t = 9$ (3, 1), (2, 2); $t = 1$ (4, 1), (3, 2); $t = 11$ (5, 1), (4, 2), (3, 3); $t = 12$ (6, 1), (5, 2), (4, 3); $t = 13$ (6, 2), (5, 3), (4, 4); $t = 14$ (6, 3), (5, 4); $t = 15$ (6, 4), (5, 5); $t = 16$ (6, 5); $t = 17$ (6, 6).

The number of points such that $i = n+1$ $(j \ne n+2)$ is equal to the number of couples $(j, k)$ such that $n+1+j+k = t+2$ with $k \le j \le n+1$ that is $\lceil (t-n)/2 \rceil$ for $n+1 \le t \le 2n+1$ and $\lfloor (3n+3-t)/2 \rfloor$ for $2n+2 \le t \le 3n+2$.

If $k = n+1$, then necessarily $i = n+1$.

The lemma follows easily. ■

LEMMA 5. *The sequence $m^n(t)$ for $1 \le t \le 3n-1$ is a palindrome.*

*Proof.* The proof is by induction. If $n = 1$, $m^1(1) = m^1(2) = 1$. Suppose now that $m^n(t) = m^n(3n-t)$, for a given $n \ge 1$.

First, suppose $t \le n$. Then $m^{n+1}(3n+3-t) = m^n(3n-t)$ by Lemma 3 and $m^{n+1}(t) = m^n(t)$ by Lemma 4. Therefore, by the induction hypothesis $m^{n+1}(3n+3-t) = m^{n+1}(t)$.

Suppose now that $n+1 \le t \le \lfloor (3n+3)/2 \rfloor$.

Then $m^{n+1}(3n+3-t) = m^n(3n-t) + 2n + 3 - (3n+3-t) = m^n(3n-t) + t - n$ by Lemma 3.

$m^{n+1}(t) = m^n(t) + t - n$ by Lemma 4. Therefore, $m^{n+1}(3n+3-t) = m^{n+1}(t)$. ■

LEMMA 6. *The sequence $m^n(t)$ is strictly increasing for $t \le \lceil 3n/2 \rceil - 1$ and $m^n(\lceil 3n/2 \rceil - 2) = M^n - 1$, $m^n(\lceil 3n/2 \rceil - 1) = m^n(\lceil 3n/2 \rceil) = M^n$.*

Furthermore, $M^{n+1} = M^n + \lceil (n+1)/2 \rceil$.

*Proof.* The proof is by induction on $n$. If $n = 1$, $m^1(1) = m^1(2) = 1$ and $M^2 = 2$. Assume that Lemma 6 is true for a given $n \ge 1$.

By Lemma 4, $m^{n+1}(t+1) - m^{n+1}(t) \geq m^n(t+1) - m^n(t)$. Therefore $m^{n+1}(t)$ is strictly increasing for $t \leq \lceil 3n/2 \rceil - 1$. By Lemma 4, we also have

$$m^{n+1}\left(\left\lceil \frac{3(n+1)}{2}\right) - 2\right\rceil = M^n + \left\lceil \frac{3(n+1)}{2}\right\rceil$$

$$-2-n = M^n + \left\lceil \frac{n-1}{2}\right\rceil.$$

$$m^{n+1}\left(\left\lceil \frac{3(n+1)}{2}\right) - 1\right\rceil = M^n + \left\lceil \frac{3(n+1)}{2}\right\rceil$$

$$-1-n = M^n + \left\lceil \frac{n+1}{2}\right\rceil.$$

$$m^{n+1}\left(\left\lceil \frac{3(n+1)}{2}\right\rceil\right) = M^n - 1 + \left\lceil \frac{3(n+1)}{2}\right\rceil$$

$$-n = M^n + \left\lceil \frac{n+1}{2}\right\rceil. \quad \blacksquare$$

PROPOSITION 4. *The minimum number of processors $M^n$ of a systolic array designed for the triangularization of an $n \times (n+1)$ matrix $[A, b]$ by the Gaussian elimination scheme described above satisfies*

$$M^{2p} = p(p+1) \quad \text{and} \quad M^{2p+1} = (p+1)^2.$$

*Proof.* We use the relation $M^{n+1} = M^n + \lceil (n + 1/2 \rceil$ and $M^1 = 1$. ■

## 3. TIME-OPTIMAL NETWORKS

Knowing the minimal number of processors needed for the execution of the algorithm, we will deal with the problem of constructing networks with good performance both in terms of time and of number of processors. The networks proposed by Ahmed *et al.* [1] and Kung and Gentleman [5] are optimal in time as the algorithm runs in time $T = 3n - 1$ but needs $(n^2/2) + O(n)$ processors.

Here we first give an orthogonally connected network optimal in time using only $(3n^2/8) + O(n)$ processors. Then we give a hexagonally connected networks using $n^2/3$ processors and quasi-optimal in time. In the original version of this paper, we also gave a network optimal in time, using $3n^2/10$ processors whose drawing requires two planar layouts. We refer the reader to [3], but do not give it here. Indeed, during the "long refereeing process" of this paper, the problem of finding a simple network using $n^2/4$ processors has been solved by Benaini and Robert [2]. However, their solution does not contain only local connections. Sakho and Muntean [12] have constructed another network both optimal in time and space, using only local connections. They conjecture that there is no planar optimal systolic solution.

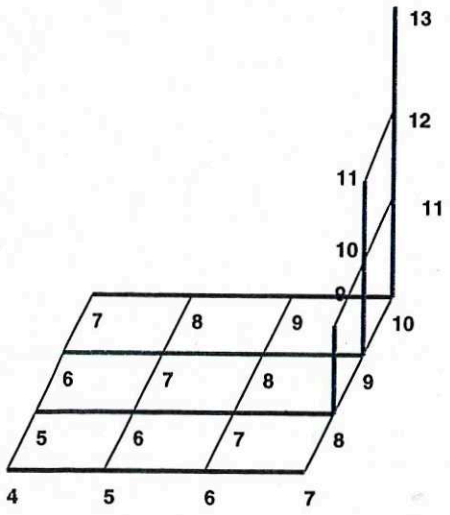


FIG. 4. All vertices on a bold line are allocated to the same processor.

### 3.1. *A Time-Optimal Network of Area* $(3n^2/8) + O(n)$

We will define the allocation function by allocating the points recursively to the processors.

Let $D'$ be the set of nonallocated points and let $D'_k = D' \cap D^n_k$. We allocate to the next processor the points of the first non empty horizontal line of the first nonempty $D'_k$ plus the corresponding vertical line in the right face of $D'$. See Fig. 4 for an example with $n = 5$ and $k = 2$. The bold lines represent the set of points allocated to a given processor.

More precisely, let $k_0$ be the least integer such that $D'_{k_0} \neq \emptyset$ let $i_0$ be the least integer such that $D'_{k_0} \cap \{(i, j, k_0)\} \neq \emptyset$, and let $j_1$ be the greatest integer such that $(i_0, j_1, k_0) \cap D'_{k_0} \neq \emptyset$. Then allocate to the next processor the set $\{(i_0, j, k_0), k_0 \leq j \leq j_1\}$ $\{(i_0, j_1, z)\ k_0 \leq z\}$. (In fact, $i_0$ varies from $k_0$ to $n$ and $j_1 = n + 2 - k$.)

It is easy to check that two points with the same value of the time function are not allocated to the same processor.

The number of processors needed to allocate all the points of $D^n_k$ is $n - k + 1$. Furthermore, once the points of the $D^n_k$'s for $1 \leq k \leq \lfloor (n+2)/2 \rfloor$ are allocated, all the points of $D^n$ are allocated.

Therefore, this allocation scheme gives rise to a network of size

$$\sum_{k=1}^{\lfloor (n+2)/2 \rfloor} n - k + 1 = \begin{cases} \dfrac{(3n+1)(n+1)}{8} & \text{if } n \text{ is odd,} \\ \dfrac{3n(n+2)}{8} & \text{if } n \text{ is even.} \end{cases}$$

All the connections are local; see Fig. 5 for an example with $n = 5$. The numbers in the squares give the time at which the processors begin and finish working.

Cell behavior can be divided into two phases:

- a first phase during which it executes the computation of the points in the direction (0, 1, 0);

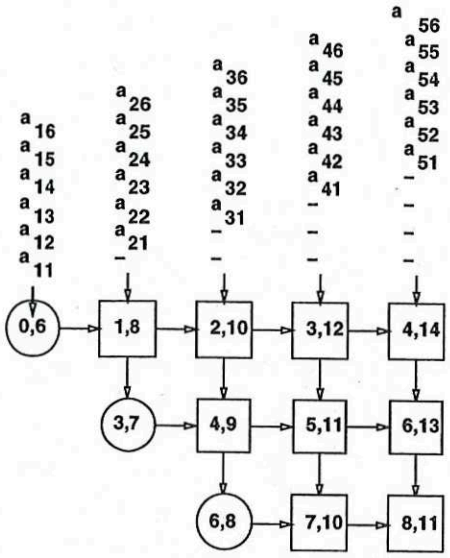


FIG. 5. The induced network for $n = 5$.

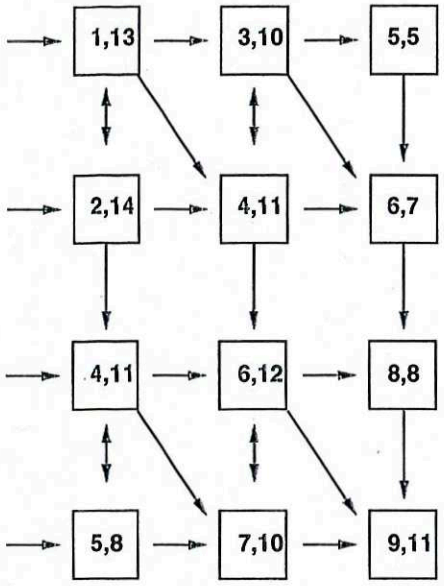


FIG. 7. Case $n = 5$.

• a second phase corresponding to the points in the direction (0, 0, 1).

As both phases are easy to realize, it suffices to define control signals which activate each cell first at time $t_1$ for the begining of phase 1, then at time $t_2$ for the beginning of phase 2. Sakho [11] has proved that such a control can be done with two Boolean signals.

The systolic array we obtain has the same performance as those exhibited by Cosnard *et al.* [4] or Benaini and Robert [2].

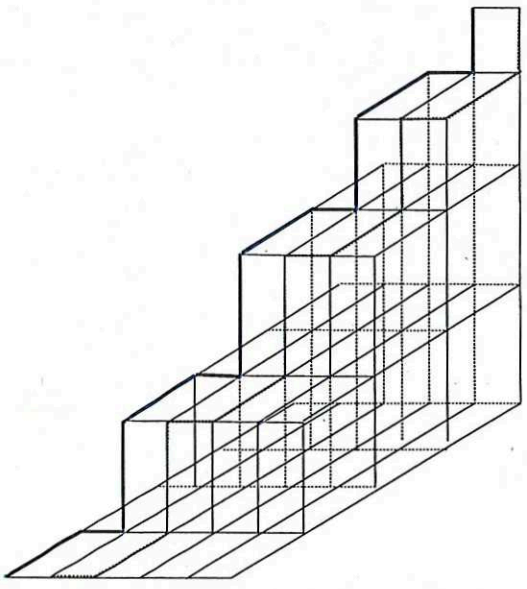

FIG. 6. Allocation to processors.

### 3.2. *A Hexagonally Connected Network of Area* $(n^2/3) + O(n)$ *and Time* $4n - 1$

Let us define another allocation scheme. In this scheme at most three points of $D_k^n$ are allocated to the same processor. We consider the two $(\lfloor n/2 \rfloor + 1)(\lfloor n/3 \rfloor + 1)$ processors $P_{i,j}$ and $P'_{i,j}$ for $0 \leq i \leq \lfloor n/2 \rfloor$ and $0 \leq j \leq \lfloor n/3 \rfloor$.

To the processor $P_{i,j}$, we allocate the points $(2i + 1, 3j + 1, 1)$, $(2i + 2, 3j + 1, 1)$, $(2i, + 2, 3j + 2, 1)$, $(2i + 2, 3j + 2, 2)$, $(2i + 3, 3j + 2, 2)$, $(2i + 3, 3j + 3, 2)$, ... $(2i + k, 3j + k, k)$, $(2i + k + 1, 3j + k, k)$, $(2i + k + 1, 3j + k + 1, k)$ ..., as long as these points exist.

To the processor $P'_{i,j}$, we allocate the points $(2i + 1, 3j + 2, 1)$, $(2i + 1, 3j + 3, 1)$, $(2i + 2, 3j + 3, 1)$, $(2i + 2, 3j + 3, 2)$, $(2i + 2, 3j + 4, 2)$, $(2i + 3, 3j + 4, 2)$, ... $(2i + k, 3j + k + 1, k)$, $(2i + k + 1, 3j + k + 1, k)$, $(2i + k + 1, 3j + k + 2, k)$ ..., as long as these points exist.

Note that for the processor $P_{i,j}$ we start at the vertex $(2i + 1, 3j + 1, 1)$ and move alternatively in the directions (1, 0, 0), (0, 1, 0), and (0, 0, 1). For $P'_{i,j}$, we move alternatively in the directions (0, 1, 0), (1, 0, 0), and (0, 0, 1).

For example, in $D^5$, $P_{0,0}$ is the processor allocated to (1, 1, 1), (2, 1, 1), (2, 2, 1), (2, 2, 2), (3, 2, 2), (3, 3, 2), (3, 3, 3), (4, 3, 3), (4, 4, 3), (4, 4, 4), (5, 4, 4), (5, 5, 4), (5, 5, 5) (see Fig. 6).

The induced systolic array has only local connections and is hexagonally connected. However, all processors must get some data before processing, and at the end of the computation the result is shared by all processors. Therefore, we must add a first phase to load the network, and after the computation a third phase to unload is needed. During the loading and unloading phases, the data follow the horizontal lines. These two additional phases are not long, because they are only diffusion phases, and $n/2$ diffusions are needed. See Fig. 7 for the case $n = 5$.

### 3.3. A Time-Optimal Network of Area $3(n^2/10) + O(n)$

In the first version of this paper [3], we gave a descriptic of a time-optimal network of area $3(n^2/10) + O(n)$. Th network is nonplanar and requires two layouts.

## REFERENCES


1. H. M. Ahmed, J. M. Delosme, and M. Morf, Highly concurre computing structures for matrix arithmetic and signal processin *IEEE Comput. Mag.* **15** (1982).
2. A. Benaini and Y. Robert, Spacetime-minimal systolic arrays f gaussian elimination and the algebraic path problem. *Parallel Con put.* **15,** 211–225 (1990).
3. J-C. Bermond, C. Peyrat, I. Sakho, and M. Tchuente, Parallelizatio of Gauss elimination algorithm on systolic arrays. Internal Repo LRI 430, 1988.
4. M. Cosnard, Daoudi, J-M. Muller, and Y. Robert, On parallel an systolic givens factorizations of dense matrices. In *Parallel Algorithm and Architectures* (M. Cosnard, Y. Robert, P. Quinton, and M Tchuente, Eds.), pp. 245–258. North-Holland, Amsterdam, 1986.
5. H. T. Kung and W. M. Gentleman, Matrix triangularization by systoli arrays. *Proc. SPIE* **298,** 4 (1981).
6. W. L. Miranker and A. Winkler, Space-time representation of systoli arrays. IBM Technical Report, RC 9775, 1982.
7. W. L. Miranker and A. Winkler, Space-time representation of compu tational structures. *Computing* **32,** 93–114 (1984).
8. D. I. Moldovan, On the design of algorithms for vlsi systolic arrays *Proc. IEEE* **71,** 113–120 (1983).
9. J. H. Moreno and T. Lang, Graph-based partitioning for matrix algo rithms for systolic arrays: Application to transitive closure. *ICPP* **1** 28–31 (1988).
10. P. Quinton, The systematic design of systolic arrays. IRISA Researcl Report 193, 1983.
11. I. Sakho, Synthèse et simulation d'algorithmes systoliques. Thèse de troisième cycle, Grenoble, 1987.
12. I. Sakho and T. Muntean, Optimal systolic network for Gaussian elimination. Rapport de Recherche IMAG, RR837-I, 1991.


Received June 8, 1988; revised May 16, 1995; accepted May 28, 1995