SALVATORE LA TORRE
MARGHERITA NAPOLI
MIMMO PARENTE

A compositional approach to synchronize two dimensional networks of processors

Informatique théorique et applications, tome 34, nº 6 (2000), p. 549-564
<http://www.numdam.org/item?id=ITA_2000__34_6_549_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **34** (2000) 549–564

A COMPOSITIONAL APPROACH TO SYNCHRONIZE TWO DIMENSIONAL NETWORKS OF PROCESSORS*

SALVATORE LA TORRE[1, 2], MARGHERITA NAPOLI[2] AND MIMMO PARENTE[2]

Abstract. The problem of synchronizing a network of identical processors that work synchronously at discrete steps is studied. Processors are arranged as an array of m rows and n columns and can exchange each other only one bit of information. We give algorithms which synchronize square arrays of $(n \times n)$ processors and give some general constructions to synchronize arrays of $(m \times n)$ processors. Algorithms are given to synchronize in time n^2, $n\lceil\log n\rceil$, $n\lceil\sqrt{n}\,\rceil$ and 2^n a square array of $(n \times n)$ processors. Our approach is a modular description of synchronizing algorithms in terms of "fragments" of cellular automata that are called signals. Compositional rules to obtain new signals (and new synchronization times) starting from known ones are given for an $(m \times n)$ array. Using these compositional rules we construct synchronizations in any "feasible" linear time and in any time expressed by a polynomial with nonnegative coefficients.

Mathematics Subject Classification. 37B15.

1. INTRODUCTION

We consider the problem of synchronizing a network of identical processors that work synchronously at discrete steps. Processors are arranged as an array of m rows and n columns. Initially a distinguished state starts computing and all others are in a quiescent state. At each step any processor sends/receives to/from

Keywords and phrases: Cellular automata, synchronization.

* *Work partially supported by Università di Salerno grant* "Metodi Formali ed Algoritmi per la Verifica di Sistemi Distribuiti".

[1] Department of Computer and Information Science, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104-638, U.S.A.

[2] Dipartimento di Informatica ed Applicazioni, Università degli Studi di Salerno "Renato M. Capocelli", 84081 Baronissi (SA), Italy.

its neighbours one bit of information. The network is modeled as a 2-dimensional Cellular Automaton (CA).

This synchronization problem is also known as the *Firing Squad Synchronization Problem* (FSSP) and it was introduced by Moore in 1964 [12] as the problem of synchronizing a line of processors (linear CA) where each processor at each step can transmit its current state to its two adjacent processors in the line. (Its name is due to the fact that the line of cells can be seen as a line of soldiers that have to fire simultaneously.) In literature many solutions to the original problem and to some variations of it have been given. The early results all focused on the synchronization in minimal time of a linear CA. Minsky showed that a solution to the FSSP requires at least $(2n-1)$ time, where n is the number of cells in the line [11]. Waksman in [16] gave the first solution in this minimal time, and Mazoyer in [9] constructed a minimal time solution with the least number of states to date: six (in [1] it has been shown that five states are always necessary).

A significant amount of papers have also dealt with some variations of the FSSP. These variations concerned both the geometry of the network and some computational constraints. In the following we briefly recall some of them. The FSSP has been studied on a (one-way) ring of n processors [2, 7], on arrays of two and three-dimensions [5, 6, 15], and on graphs [13, 14]. Some constrained variations of the FSSP have concerned solutions on reversible CA (*i.e.*, backward deterministic CA) [3] and CA with a number-conserving property (*i.e.*, a state is a tuple of positive integers whose sum is constant during the computation) [4]. Other kinds of constraints have concerned the amount of information which is allowed to exchange between any pair of adjacent processors: unidirectional flow of the information [2,7], 1-bit of information exchanged in both directions between any pairs [8,10].

Besides minimal-time solutions to the FSSP, also solutions at a predetermined (non minimal) time have been considered. This is an interesting and challenging theoretical problem, which is also directly connected to the sequential composition of cellular automata. Given two cellular automata A_1 and A_2 computing respectively the functions $f_1(x)$ and $f_2(x)$, the sequential composition of A_1 followed by A_2 is the cellular automaton obtained in the following way: first A_1 starts on a standard initial configuration and when it has done with its computation, A_2 starts using the final configuration of A_1 as initial configuration. The resulting automaton clearly computes $f_2(f_1(x))$. In order to compose these two automata, it is necessary to synchronize all the cells that will be used by A_2 at the time A_1 computes $f_1(x)$ for a given input x. Solutions to the FSSP at a given time have been studied on rings and toroidal square arrays with unidirectional flows of information [7], and on lines of cells exchanging only one bit at each time step [8].

In this paper we study non minimal time solutions to the FSSP in two dimensions and with the 1-bit constraint on the exchanged amount of information. We consider a *1-bit 2-dimensional CA*, that is a grid of $(m \times n)$ identical finite-state processors (*cells*) which exchange one bit of information each other. A synchronization in time $t(m,n)$ of an array of $(m \times n)$ cells is a 1-bit 2-dimensional cellular automaton such that, starting from a standard configuration, all cells enter for the first time the Firing state at time $t(m, n)$. We obtain new synchronization times in a compositional way: we first describe basic synchronizing algorithms, then we give general rules to compose synchronizations. Basic synchronizations are obtained by composing elementary signals, which can be seen as fragments of Cellular Automata. A synchronization is thus a special signal obtained as a composition of many simpler signals. Compositional rules for both signals and synchronizations include *parallel* composition, *sequential* composition, and *iterated* composition. We also give sufficient conditions to the applicability of these compositions. In the parallel composition we start many synchronizations or signals, all at the same time. Sequential composition appends a synchronization or a signal to the end of another signal, possibly with a constant time offset. This way we are able to construct a synchronization in time $t_1(m,n)+t_2(m,n)+d$, for $d \geq 0$, if there exist synchronizations in time $t_1(m,n)$ and $t_2(m,n)$. If we are given two synchronizations respectively in time $t_1(m,n)$ and $t_2(m,n)$, the iterated composition consists of iterating $t_2(m,n)$ times the synchronization in time $t_1(m,n)$, thus obtaining a new synchronization in time $t_1(m,n) \cdot t_2(m,n)$.

We also give two main techniques to obtain synchronizations of bidimensional arrays starting from synchronizations of linear arrays. The first one relies on the fact that an $(m \times n)$ array of processors can be seen as many lines of $(m+n-1)$ processors (each of them having as endpoints cells $(1,1)$ and (m,n)) where the same synchronization can be executed simultaneously. This way we obtain a synchronization on an $(m \times n)$ array in time $t(m+n-1)$ provided that there exists an algorithm for a linear array of k processors in time $t(k)$. The second technique consists of synchronizing a row (respectively, a column) of an $(m \times n)$ array and then starts a same synchronization in parallel (and at the same time) on all columns (respectively, all rows).

We apply all these results to obtain some interesting families of new synchronizations. By the above techniques, we give basic synchronizations of an $(n \times n)$ square array in time n^2, $n\lceil\log n\rceil$, $n\lceil\sqrt{n}\,\rceil$ and 2^n. The compositional rules are used to determine synchronizations in any "feasible" linear time and in any time expressed by a polynomial with nonnegative coefficients. These constructions use as building blocks the synchronizations in minimal time and in time n^2.

The remainder of this paper is organized as follows. In Section 2 we give the definitions and introduce the notation we will use in the rest of the paper. In Section 3 solutions to the FSSP on an $(n \times n)$-array in time n^2, $n\lceil\log n\rceil$, $n\lceil\sqrt{n}\,\rceil$ and 2^n are given. In Section 4 we discuss several ways to obtain new solutions to the FSSP on $(m \times n)$-arrays from known ones. In Section 5 we give our conclusions.

2. Preliminaries

In this section we give basic definitions and some preliminaries.

Basic Definitions. A one bit 1-dimensional cellular automaton (shortly 1-CA) is a line of n finite-state machines, called *cells*, which are identical except for those at the two endpoints. In a 1-CA, the i-th cell is connected to the $(i-1)$-th and

$(i+1)$-th cells, for all $i = 2, \ldots, n-1$. The first and the last cells are connected, respectively, to the second and the $(n-1)$-th cell, see Figure 1. The 2-dimensional case is a natural generalization of the 1-CA. Omitting minor details, a one bit 2-dimensional cellular automaton (shortly 2-CA) is an array of $(m \times n)$ finite-state machines (*cells*) which are identical except for the boundary ones, and where cell (i,j) is connected to cells $(i-1,j)$, $(i+1,j)$, $(i,j-1)$ and $(i,j+1)$, see Figure 1. The cells operate synchronously at discrete time steps. At each step each cell exchanges one bit of information with its adjacent cells and modifies its state depending on its current state and the bits sent by the adjacent cells at the previous step.

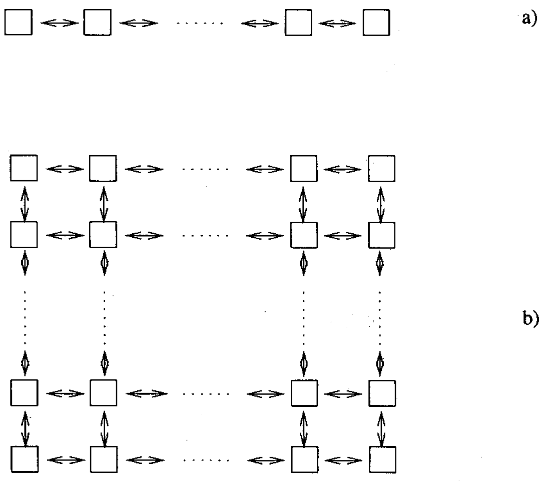


FIGURE 1. The 1-dimensional and 2-dimensional array.

In what follows, the symbol Q refers to the set of states of a given cellular automaton. A *configuration* of a 1-CA is a mapping $C : \{1, 2, \ldots, n\} \to \{0,1\} \times Q \times \{0,1\}$. A configuration at time t gives, for each cell i, the state entered and the two bits sent at this time. A starting configuration is a configuration at time 1. The definition of configuration can be easily extended to a 2-CA, by considering that each cell sends the bits to its four adjacent cells. In the following we often write "(A, C)" to denote a 1-CA, or a 2-CA, A starting on a configuration C. Within the state set there are three distinguished state: G the *General* state, L the *Latent* state, and F the *Firing* state. State L has the property that if a cell in state L receives all bits 0 from its neighbours, it remains in this state and sends bits 0 to its neighbours. A *standard configuration* is a configuration where each cell is in state L and sends bits 0, except for cell 1 (resp. cell $(1,1)$) which is in state G and sends bits 1 to each neighbour. A *synchronization in time* $t(n)$ of a linear array of n cells, is a 1-CA such that starting from a standard configuration all cells enter at time $t(n)$ for the first time state F. Analogously a *synchronization in time* $t(m,n)$ of a rectangular array of $(m \times n)$ cells is a 2-CA such that starting from a standard configuration all cells enter for the first time state F at time $t(m,n)$. When $m = n$, we speak about a synchronization of a square array in time $t(n)$. A synchronization of a linear array of n cells in time $(2n-1)$ is called a minimal time synchronization, since it can be easily proved that a synchronization is not possible in time less than $(2n-1)$.

Signals. In [8] the concept of signal was introduced as a mean to design a 1-CA. Informally a signal describes the information flow in the space-time description of a cellular automaton, allowing a modular description of the synchronization process. The scheme used to present some synchronization algorithms in time $t > 2n-1$ for a linear array of n processors is the following: some signals are designed and composed in order to obtain an overall signal that starts from the leftmost processor and comes back to it in exactly $(t-2n+1)$ time units; then a minimal time synchronization starts, thus synchronizing the n processors in time t. We consider the time unrolling of a 1-CA A starting on a configuration C, that is we reason about a space-time array. A pair (i,t) of this array, with $1 \leq i \leq n$ and $t \geq 1$, is called a *site*, the state of the cell i at time t is denoted *state*(i,t) and the bits sent to the adjacent cells are denoted by *left*(i,t) and *right*(i,t). A site (i,t) is said to be *active* if either sends/receives a bit 1 to/from its neighbours or changes its state. We denote by $\text{Cell}(A,C)$ the set of cells i such that site (i,t) is active for some t.

Let A be a 1-CA and C be a configuration. Define the time $t_i^{\max} = \max\{t | (i,t) \text{ is active}\}$ and $t_i^{\min} = \min\{t | (i,t) \text{ is active}\}$. The set of sites $(i, t_i^{\min})$ for $i \in \text{Cell}(A,C)$ is called *rear* of (A,C) and the set of sites $(i, t_i^{\max})$ is the *front* of (A,C). Moreover we say that (A,C) is *tailed* if there exists a subset of Q, called *tail*(A,C) such that for all $i \in \{1,\dots,n\}$, *state*$(i,t) \in \text{tail}(A,C)$ if and only if (i,t) belongs to the front of (A,C). The states in $\text{tail}(A,C)$ are called *tail* states. In words, a tail state appears for the first time on the front of (A,C).

Two active sites $(i_1,t_1),(i_2,t_2)$ are *consecutive* if $t_2 = t_1 + 1$ and $i_2 \in \{i_1 - 1, i_1, i_1 + 1\}$. A *simple signal* of (A,C) is a subset S of consecutive sites with the property that if (A,C) is tailed, then $(i, t_i^{\max})$ belongs to S. The union of a finite number of simple signals of a given (A,C) is called *signal* of (A,C). A graphical representation of a simple signal S is obtained by drawing a line between:

(i) every pair of sites $(i,t) \in S$ and $(i,t+1) \in S$ and
(ii) every pair of sites $(i,t) \in S$ and $(i+1,t+1) \in S$ (resp. $(i-1,t+1) \in S$) if *right*$(i,t) = 1$ (resp. *left*$(i,t) = 1$).

A graphical representation of a signal is obtained by the graphical representation of its simple signals. The *length* of a signal S is $(t^{\max} - t^{\min} + 1)$ where $t^{\max} = \max\{t | (i,t) \in S, 1 \leq i \leq n\}$ and $t^{\min} = \min\{t | (i,t) \in S, 1 \leq i \leq n\}$. Sometimes, in the rest of the paper we refer to a signal without specifying a 1-CA and a starting configuration.

In the following examples we recall some signals introduced in [8].

Example 1. *Let $i \neq j$ and $\text{MAX}(i,j)$ be the set containing the sites $(i+h, h+1)$ if $i < j$, or sites $(i-h, h+1)$ otherwise, for $0 \leq h \leq |i-j|+1$. This set is a* *simple signal, with length $|i-j|+1$, of a tailed 1-CA that starts from a configuration having the states of cells i and j different from all the others.*

Example 2. *Given a positive constant $k < n$, the signal* MARK$(n-k)$ *is used to mark the cell $n-k$. The length of the signal* MARK *is $n+k$ (see Fig. 2). It can be easily seen that* MARK *is a signal of a tailed 1-CA.*

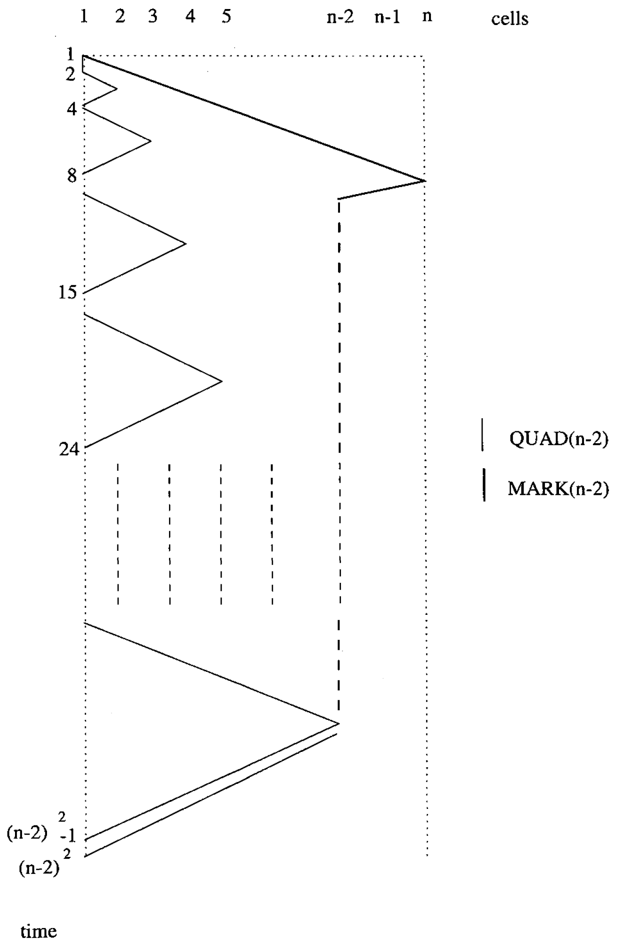


FIGURE 2. The signal $\mathrm{cat}_1(\text{QUAD}(n-2), \text{MARK}(n-2))$.

We recall now the signal composition. We say that a 1-CA A_2 on C_2 *can follow* a tailed 1-CA A_1 on C_1 if there exists a function h defined over tail(A_1, C_1) and such that $h(p) = C_2(i)$ if $p = state(i,t)$. Given two signals S_1 and S_2, we can define the concatenation $\mathrm{cat}_r(S_1, S_2)$ as the signal obtained by starting S_1 at time 1 and S_2 at time $r+1$, that is S_2 is delayed r time steps. More formally $\mathrm{cat}_r(S_1, S_2) = S_1 \cup \{(i, t+r) | (i,t) \in S_2\}$.

The following remark recalls some sufficient conditions for the existence of a tailed 1-CA for a signal $\mathrm{cat}_r(S_1, S_2)$.

Remark 1. [8] *Let S_1, S_2 be signals of tailed 1-CA's (A_1, C_1) and (A_2, C_2), respectively. The signal $S = cat_r(S_1, S_2)$ is the signal of a tailed 1-CA (A, C) if the following two conditions hold:*

1. *(A_2, C_2) can follow (A_1, C_1);*
2. *if the site (i, t) belongs to the front of (A_1, C_1) and (i, t') belongs to the rear of (A_2, C_2), then $t < t' + r$.*

3. Synchronization of a square array

In this section we introduce two signals of a 1-CA. The first has a quadratic length and the second has an exponential length in the number of cells. Then we design synchronizations of a square array using the following scheme: first we synchronize the first row of the square array and then we simultaneously synchronize all the columns. That is the rows and the columns are seen as 1-CA. This way we obtain synchronizations of a square array of $(n \times n)$ cells in n^2, 2^n, $n\lceil \log n \rceil$ and $n\lceil \sqrt{n}\, \rceil$.

The signal QUAD. Given a positive constant $k < n$, $\textsc{Quad}(n-k)$ is a signal of a 1-CA A which is described as follows:

- initially the cell 1 sends a bit 1 to the right; then if it receives a bit 1 from the right, it sends with a delay of one step (except for the first time, when there is no waiting), a bit 1 back to the right; the cell 1 eventually halts when it receives two consecutive bits 1;
- for $1 < h < (n-k)$, the cell h sends a bit 1 to the left when it receives for the first time a bit 1 from the left; then, if the cell h receives again a bit 1 from an adjacent cell, it sends a bit 1 to the other adjacent cell;
- the cell $(n-k)$ sends two consecutive bits 1 to the left when it receives a bit 1 from the left.

The 1-CA A can be further designed such that it is tailed by observing that the cells from 1 to $(n-k)$ can enter a tail state when they receive two consecutive bits 1. The length of the QUAD signal is $(n-k)^2 - 1$.

Clearly, for the implementation of this signal cell $(n-k)$ needs to be distinguished. In what follows we will use $\textsc{Quad}(n-2)$, thus we only need to distinguish cell $(n-2)$: this can be done by $\textsc{Mark}(n-2)$ and for all $n > 5$. Clearly for smaller n much easier ad hoc algorithms can be given (see Fig. 2).

The signal EXP. Given two positive constants k and c, we will define the signal $\textsc{Exp}(n-k, c)$.

An *idle* cell is a cell which never sends a bit 1 unless it receives a bit 1 from the left and in this case it sends two consecutive bits 1 to the left.

Initially the only idle cell is the cell $(n-k)$. $\textsc{Exp}(n-k, c)$ is a signal of a 1-CA which is described as follows:

- first cell 1 sends a bit 1 to the right; then, whenever cell 1 receives a bit 1 from the right, it immediately replies sending back a bit 1; finally, if cell 1 receives two consecutive bits 1 from the right, then it changes into an idle cell;
- for $1 < h < (n-k)$, we distinguish two cases:
  - if the bit is received from the left then it alternates the following two behaviours:
    1. it sends a bit 1 back to the left, (let us call these *peak cells*)[3]
    2. it sends a bit 1 to the right;

    each peak cell starts counting from 1 to $2^{i+1}-2$, for $1 < i \leq c$. When $2^{i+1}-2$ has been just counted, if the peak cell receives a bit 1 from the left at the next time unit, then it is the i-th cell in the line and is marked (see below for an explanation). This way it can be distinguished later.
  - if a bit 1 is received from the right, then it sends a bit 1 to the left. If at the next time unit cell h receives another bit 1 from its right neighbour, then two other subcases need to be considered:

    if $h > c$ then the cell switches into an idle cell;

    else, for $h \leq c$, the cell sends two consecutive bits 1 to the left. (Note that when this case occurs, cells $h \leq c$ have already been marked by step 2 above.)

From the algorithm we have just described, a proof by induction on $i \leq c$ can be given to show how a peak cell can be marked, in fact the following property holds: the length of the interval from the instant cell i is a peak cell for the first time and the instant it becomes a peak cell for the second time is $2^i + \sum_{j=1}^{i-1} 2^j(i-j)$ (see Fig. 3 where $c = 3$, cell 2 is marked at time 9 and cell 3 is marked at time 20).

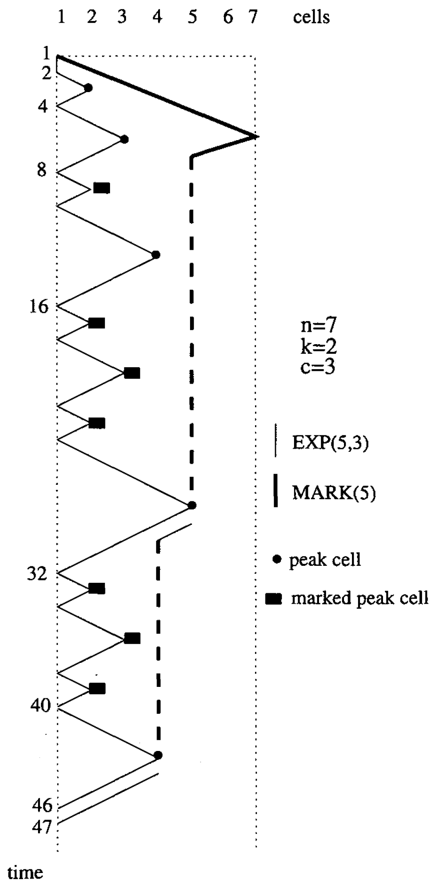
To implement a tailed 1-CA for $\textsc{Exp}(n-k, c)$ initially the cell $(n-k)$ must be distinguished. In what follows we will use the signals $\textsc{Exp}(n-2, 3)$ and $\textsc{Exp}(n-2, 1)$: the cell $n-2$ can be distinguished by using $\textsc{Mark}(n-2)$, for $n > 5$. Observe also that the cells from 1 to $(n-2)$ can enter a tail state after they received two consecutive bits 1. The length of $\textsc{Exp}(n-k, c)$ is $2^{n-k+1} - 2(n-k) - 2^{c+1} + 2(c+1)$ (see Fig. 3).

We can give now the synchronizing algorithms for the square array.

Theorem 1. *There is a synchronization of an $(n \times n)$ square array in time n^2.*

Proof. The algorithm is the following: first a signal $\text{cat}_1(\textsc{Mark}(n-2), \textsc{Quad}(n-2))$ is started on the first row, the length of this signal is $(n-2)^2$ since $\textsc{Quad}(n-2)$ is delayed one time step. This is a signal of a tailed 1-CA starting from a standard configuration (see Rem. 1). Thus after $(n-2)^2$ time units the cell $(1,1)$ enters a tail state, say G'. Considering G' as the General state, a minimal time synchronization on a linear array of n cells is executed on the first row and this takes other $(2n-2)$ time units. Once the Firing state F' is reached, we use F' as the General state of a minimal time synchronization that this time

[3] Actually, this is a property of the state entered by this cell.



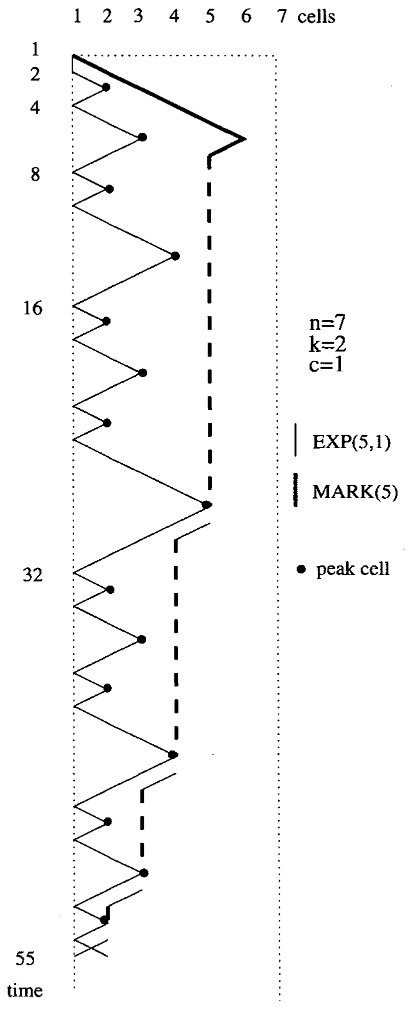


FIGURE 3. The signals $\mathrm{cat}_1(\text{EXP}(5,3), \text{MARK}(5))$ and $\mathrm{cat}_1(\text{EXP}(5,1), \text{MARK}(5))$.

runs on each column, thus taking another $(2n-2)$ time units, which adds up to a total time of n^2. □

Theorem 2. *There is a synchronization of an $(n \times n)$ square array in time 2^n.*

Proof. First a signal $\mathrm{cat}_1(\text{EXP}(n-2,3), \text{MARK}(n-2))$ is started on the first row, see Figure 3. After $(2^{n-1} - 2n - 3)$ time units the cell $(1,1)$ enters a tail state, say H. This is a signal of a tailed 1-CA starting from a standard configuration (see Rem. 1). Now the cell $(1,1)$ enters a state G' and a minimal time synchronization on the first row is accomplished, using G' as the General state, thus taking other $(2n-1)$ time units. Once the Firing state F' is reached, each cell of the first row enters a state G'', and launches the signals MARK$(n-2)$ and EXP$(n-2,1)$ on each column, using G'' as the General state. This takes another $(2^{n-1} - 2n + 5)$ time units, which sums up to time $(2^n - 2n + 1)$. Finally, a minimal time synchronization on each column is accomplished, thus reaching time 2^n. □

Theorem 3. *There is a synchronization of a $(n \times n)$ square array in time $n\lceil \log n \rceil$ and in time $n\lceil \sqrt{n} \rceil$.*

Proof. The algorithms resemble those used to synchronize a line of n cells at the same times shown in [8]. Therefore here we only outline the main idea. For the synchronization in time $n\lceil \log n \rceil$, we use a signal to synchronize the first row in time $(n \log n - 2n)$ and then we apply a synchronization to each column in time $2n$ (just a minimal time synchronization for a linear array with one more time unit).

Let us informally describe the synchronization of the first row. Initially the cells numbered $(1,5)$, $(1,\lceil n/2 \rceil)$, $(1,\lfloor n/2 \rfloor + 1)$ and $(1, n-4)$ are marked: this can be easily accomplished in time $2n$. This way the row can be seen as split in two halves and for each half a symmetric computation is done, therefore we will describe only the left half. A phase is iterated $(\lceil \log n \rceil - 5)$ times: each iteration starts at time $((i+1)n+1)$, $1 \leq i \leq (\log n - 5)$, and has length n. During the i-th iteration, the test $(i+5) \geq \lceil \log n \rceil$, is performed in the following way: a signal of length $2^{(i+5)}$ on the linear array consisting of the first $(i+5)$ cells and a signal MAX of length n, which is composed of MAX$(1, \lceil n/2 \rceil)$ and MAX$(\lceil n/2 \rceil, 1)$, are performed (see Fig. 4). We compose the two signals to give MAX a higher priority, thus if the exponential signal reaches a cell after the MAX signal, it is aborted. In this case the MAX signal finishes earlier than or at the same time as the exponential signal, and this means that $(i+5) \geq \log n$ and thus this is the last iteration. Otherwise (that is MAX finishes later) cell $(i+1)$ is marked and a new iteration starts (see Fig. 4). Omitting minor details, at the end of the last iteration all cells are forced in tail states, so determining a standard configuration for a synchronization of a linear array of $\lceil n/2 \rceil$ cells in time n. The synchronization in time $n\lceil \sqrt{n} \rceil$ can be obtained in a very similar way by considering a quadratic signal, instead of an exponential one, to synchronize the first row in time $(n\sqrt{n} - 2n)$. □

4. HOW TO OBTAIN NEW SYNCHRONIZATIONS OF A RECTANGULAR ARRAY

In this section we discuss how to obtain new synchronizations of $(m \times n)$ arrays using known algorithms to synchronize linear arrays. We start describing synchronizations of an $(m \times n)$ array in time $t(m+n-1)$, given a synchronization of a line of k processors in time $t(k)$. Then, we give some compositional rules on synchronizations of $(m \times n)$ arrays. We conclude this section showing how to construct synchronizations of a square array of processors in any arbitrary "feasible" linear time and in any time expressed by polynomials with nonnegative integer coefficients.

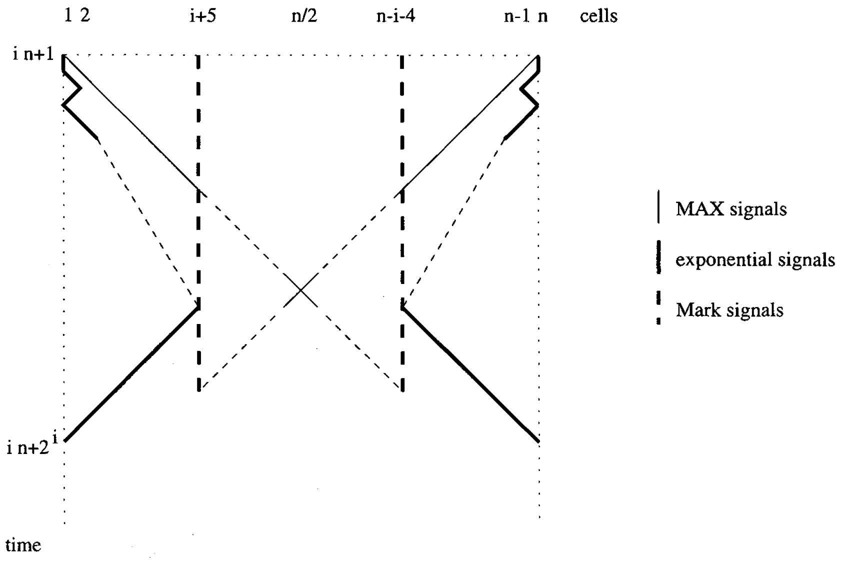


FIGURE 4. The phase in the i-th iteration, $i > 1$ and n odd of the synchronization in time $n \log n$.

Theorem 4. *Given a synchronization of a line of k processors in time $t(k)$, there exists a synchronization of an $(m \times n)$ array in time $t(m + n - 1)$.*

Proof. An $(m \times n)$ array can be seen as many lines of $(m + n - 1)$ cells, each of them having as endpoints cells $(1,1)$ and (m,n). Each of these lines corresponds to a "path" from cell $(1,1)$ to cell (m,n) going through $(m + n - 3)$ other cells. Each cell (i,j) of these paths has as left neighbour either cell $(i - 1, j)$ or cell $(i, j - 1)$ and as right neighbour either cell $(i + 1, j)$ or cell $(i, j + 1)$.

Notice that the cell (i,j) is the $(i + j - 1)$-th cell from the left in all the lines it belongs to. This property allows us to execute simultaneously on all these lines a synchronization in time $t(k)$ for a line of k cells. Since the length of each line is $(m+n-1)$, we have a synchronization of the $(m \times n)$ array in time $t(m+n-1)$. □

In [8] synchronizations for a linear array of n cells have been given in the following times: n^2, 2^n, $n\lceil \log n \rceil$, and $n\lceil \sqrt{n} \rceil$. Using these results and the above theorem we can give the following corollary.

Corollary 1. *Given an $m \times n$ array and $K = m + n - 1$, then:*

- *there are synchronizations of the $(m \times n)$ array in time K^2, 2^K, $K \lceil \log K \rceil$, and $K\lceil \sqrt{K} \rceil$;*
- *let a and b be two integer numbers, if $aK + b \geq 2K - 1$ then there is a synchronization of the $(m \times n)$ array in time $(aK + b)$;*
- *let $h \geq 2$ be an integer number and $a_0, \ldots, a_h$ natural numbers with $a_h \geq 1$, then there is a synchronization of the $(m \times n)$ array in time $a_h K^h + \ldots + a_1 K + a_0$.*

Given a 2-CA, we can consider its time unrolling and easily extend the definitions given in Section 2 to signals for 2-CA. For a 2-CA (A, C), we call $\text{Links}(A, C)$ the set of communication links effectively used by (A, C), that is all the ordered pairs of adjacent cells x, y such that there is a bit 1 sent from x to y at some time t. We give now some results on signal composition. The first lemma says that when two signals have disjoint sets of active communication links then it is possible to obtain a new signal which is their parallel composition. The second lemma generalizes Remark 1 to the 2 dimensional case and establishes when it is possible to design a 2-CA to concatenate two signals, thus obtaining their sequential composition.

Lemma 1. *Given an $(m \times n)$ 2-CA A, let S_1 and S_2 be two signals of A on configurations C_1 and C_2, respectively. If $\mathit{Links}(A, C_1) \cap \mathit{Links}(A, C_2) = \emptyset$ then there exist an $(m \times n)$ 2-CA A' and a configuration C' such that $S_1 \cup S_2$ is a signal of (A', C'). Moreover, if (A, C_1) and (A, C_2) are tailed then also (A', C') is tailed.*

Lemma 2. *Let S_1, S_2 be signals of two 2-CA's (A_1, C_1) and (A_2, C_2), respectively. The signal $\mathit{cat}_r(S_1, S_2)$ is the signal of a 2-CA (A, C) if the following two conditions hold:*

1. *(A_1, C_1) is tailed and (A_2, C_2) can follow (A_1, C_1);*
2. *if the site (i, j, t) belongs to the front of (A_1, C_1) and (i, j, t') belongs to the rear of (A_2, C_2), then $t < t' + r$.*

Moreover if (A_2, C_2) is tailed and $\mathit{Cell}(A_1, C_1) \subseteq \mathit{Cell}(A_2, C_2)$, then (A, C) is tailed too.

We can now give the sequential and iterated compositions of synchronizations of $(m \times n)$ arrays. In the following, if A_i is a synchronization, then G_i, L_i, and F_i are the General, Latent, and Firing states of A_i, respectively.

Theorem 5. *If A_i for $i = 1, 2$ are two synchronizations of an $(m \times n)$ array in time $t_i(m, n)$ and $d \geq 0$, then there is a synchronization in time $t_1(m, n) + t_2(m, n) + d$.*

Proof. Let S_i be the signal of (A_i, C_0), where C_0 is a standard configuration. From Lemma 2, if $r = t_1(m, n) + d$, then there exists A such that $\text{cat}_r(S_1, S_2)$ is a signal of (A, C_0). Moreover, $\text{cat}_r(S_1, S_2)$ is a synchronization in time $t(m, n) = t_1(m, n) + t_2(m, n) + d$. □

Theorem 6. *If A_i for $i = 1, 2$ are two synchronizations of an $(m \times n)$ array in time $t_i(m, n)$, then there is a synchronization in time $t_1(m, n)\, t_2(m, n)$.*

Proof. We define a synchronization A consisting of an Iterative phase with length $t_1(m, n)$ which is executed $t_2(m, n)$ times. The set of states of A is $Q_1 \times Q_2 \times \{0, 1\}^4$, the General state is $(G_1, G_2, 0, 1, 0, 0)$, the Latent state is $(L_1, L_2, 0, 0, 0, 0)$ and the Firing state is $(F_1, F_2, 0, 0, 0, 0)$. In the Iterative phase, the synchronization A modifies the first component of its state according to the transition functions of A_1, until this component is F_1. At the end of this phase A executes a transition step modifying the second component of the state according to the transition functions of A_2. Outputs of A_2 transition functions are saved in the last four components according to the order left, right, up, and down. Moreover, in this same step, A replaces F_1 with either G_1 or L_1 (depending on whether the cell is the one triggering in the initial configuration the firing signal of A_1) in the first component. So the Iterative phase can start again, until the Firing state is entered by all the cells. Thus, the synchronization A_1 is iterated exactly $t_2(m,n)$ times and A takes time $t_1(m,n) \cdot t_2(m,n)$. □

Let A be a synchronization in time $t(m,n)$ and $X \times Y \subseteq \{1,\dots,m\} \times \{1,\dots,n\}$, we say that A is $(X \times Y)$-*detectable* if the set of states *state*$(i,j,t(m,n)-1)$, $\forall (i,j) \in X \times Y$, is disjoint from the set of states *state*$(i',j',t(m,n)-1)$, for all $(i',j') \notin X \times Y$. Furthermore, we say that A has the *parity property* with respect to m (respectively, n), if for $i = 1,\dots,m$ and $j = 1,\dots,n$:

- the set of states containing *state*$(1,j,t(m,n)-1)$ when m is even (respectively, *state*$(i,1,t(m,n)-1)$ when n is even) is disjoint from the set containing *state*$(1,j,t(m,n)-1)$ when m is odd (respectively, *state*$(i,1,t(m,n)-1)$ when n is odd);
- the set of states containing *state*$(m,j,t(m,n)-1)$ when m is even (respectively, *state*$(i,n,t(m,n)-1)$ when n is even) is disjoint from the set containing *state*$(m,j,t(m,n)-1)$ when m is odd (respectively, *state*$(i,n,t(m,n)-1)$ when n is odd).

We recall that if we consider a line of n cells with an initial configuration *state*$(1,1)$ $=$ *state*$(n,1) = G$ and *state*$(i,1) = L$ for $i \neq 1,n$, we can synchronize the line in time n, if n is odd, and time $n-1$ otherwise. We call such a synchronization a *two-end synchronization.* Mainly, in this synchronization the linear array is split in two halves and, starting at the same time, on both the halves a minimal time synchronization is executed (the composition of the corresponding signals is possible by Lem. 1).

Lemma 3. *Let $d \geq 0$ and $m \geq d$ (respectively, $n \geq d$). Let A be a synchronization of an $(m \times n)$ array in time $t(m,n)$ with the parity property with respect to m (respectively, n) and $(X \times Y)$-detectable for a set $X \times Y = (X' \times \{1,\dots,n\})$ (respectively, $X \times Y = (\{1,\dots,m\} \times Y')$), where:*

- *$X' = \{1,\dots,d\} \cup \{m-d+1,\dots,m\} \cup \{m/2, m/2+1\}$ (respectively, $Y' = \{1,\dots,d\} \cup \{n-d+1,\dots,n\} \cup \{n/2, n/2+1\}$), if m (respectively, n) is even and*
- *$X' = \{1,\dots,d\} \cup \{m-d+1,\dots,m\} \cup \{\lceil m/2 \rceil\}$ (respectively, $Y' = \{1,\dots,d\} \cup \{n-d+1,\dots,n\} \cup \{\lceil n/2 \rceil\}$), otherwise.*

Then there exists a synchronization of an $(m \times n)$ array in time $t(m,n) + m - d$ (respectively, $t(m,n) + n - d$).

Proof. We consider only the case that A is a synchronization of an $(m \times n)$ array in time $t(m,n)$ with the parity property with respect to m, the other case is analogous. For $d = 0$, a synchronization in time $t(m,n) + m$ consists of two phases. The initial phase is the synchronization A itself and the second phase is a two-end synchronization on $(1,i),\dots,(m,i)$ for $i = 1,\dots,n$. By hypothesis at the end of the first phase the middle cells (or the shared middle cell, when m is odd) of each vertical line are marked, so they can behave as the first and the last in the line. Moreover, the first and the last cells are aware of the parity of m, so that they can be set in suitable General states, in such a way that the total time of the two-end synchronization is m in both cases. Thus from Lemma 2, a synchronization in time $t(m,n)+m$ exists. A synchronization A' in time $t(m,n)+m-d$ can be obtained by modifying the previous synchronization in such a way that A' jumps from the $(t(m,n)-1)$-th configuration of A exactly to the d-th configuration of the two-end synchronization. Notice that in the d-th configuration of the two-end synchronization only the first d and the last d cells of each line are in states which are different from the Latent state. Thus, by the $(X \times Y)$-detectability of A, A' is properly defined and is a synchronization in time $t(m,n)+m-d$. □

Lemma 4. *Let A be a synchronization of an $(m \times n)$ array in time $t(m,n)$ with the parity property with respect to m (respectively, n) and $(X \times Y)$-detectable for a set $X \times Y = (X' \times \{1,\dots,n\})$ (respectively, $X \times Y = (\{1,\dots,m\} \times Y')$) where:*

- *$X' = \{m/2, m/2+1\}$ (respectively, $Y' = \{n/2, n/2+1\}$) if m (respectively, n) is even, and*
- *$X' = \{\lceil m/2 \rceil\}$ (respectively, $Y' = \{\lceil n/2 \rceil\}$), otherwise.*

Then there is a synchronization of an $(m \times n)$ array in time $m\,t(m,n)$ (respectively, $n\,t(m,n)$).

Proof. We consider only the case that A is a synchronization of an $(m \times n)$ array in time $t(m,n)$ with the parity property with respect to m, the other case is analogous. We define a synchronization A' consisting of an iterative phase, with length $t(m,n)$, executed m times. The iterative phase consists of the synchronization A. The states of A' are tuples whose first component is a state of A and the second component is a state of the two-end synchronization applied to each row of the array. In the iterative phase, A' modifies the first component of a state according to the transition functions of A. At each iteration, just a step of a two-end synchronization is performed. Thus A' is a synchronization in time $m\,t(m,n)$. □

In the rest of the section we present the polynomial time synchronizations of a square array of $(n \times n)$ processors. By the minimal time synchronization presented in [10], we obtain the following result.

Remark 2. *There exists a minimal time synchronization of an $(n \times n)$ square array in time $t(n) = 3n-2$ with the parity property and $(X \times Y)$-detectable for a set $X \times Y = (X' \times \{1,\dots,n\})$, where $X' = \{n/2, n/2+1\}$, if n is even, and $X' = \{\lceil n/2 \rceil\}$, otherwise.*

Thus we have the following theorems:

Theorem 7. *Let a and b be integers. There is a synchronization of an $(n \times n)$ square array in time $3n-2+a(n-2)+b$.*

Proof. Directly by Remark 2, Lemma 3 (for $d=2$), and Theorem 5. □

Theorem 1 shows the existence of a synchronization in time n^2, which includes a minimal time synchronization, thus the next remark follows:

Remark 3. *There exists a synchronization of an $(n \times n)$ square in time n^2 with the parity property and $(X \times Y)$-detectable for a set $X \times Y = (X' \times \{1, \dots, n\})$, where:*

- $X' = \{n/2, n/2+1\}$, *if n is even, and*
- $X' = \{\lceil n/2 \rceil\}$, *otherwise.*

Finally we present synchronizations for any feasible polynomial time.

Theorem 8. *Let $h \geq 2$ be an integer number and $a_0, \dots, a_h$ natural numbers with $a_h \geq 1$. There is a synchronization of an $(n \times n)$ array in time $a_h n^h + \ldots + a_1 n^1 + a_0$.*

Proof. From Remark 3 and Lemma 4, a synchronization in time n^b can be obtained for every $b \geq 2$. Using Theorem 5 to compose these times, the theorem follows. □

5. Conclusions

We have considered the problem of synchronizing a network of identical processors that work synchronously at discrete steps and are arranged as an array of m rows and n columns. We assume that each processor can exchange only one bit of information with its neighbours. We have given algorithms which synchronize square arrays of $(n \times n)$ processors and some general constructions to synchronize arrays of $(m \times n)$ processors. Our algorithms are obtained using a compositional approach based on the concept of signal. In this perspective we have used some compositional rules to obtain new signals (and thus new synchronizations) starting from known ones. In particular we have signals and synchronizations composition to construct synchronizations of an $(n \times n)$ square in time n^2, $n\lceil \log n \rceil$, $n\lceil \sqrt{n} \rceil$, 2^n, and polynomial. We observe that all these synchronizations can be extended to the general case of an $(m \times n)$ array, considering the time of the synchronization as a function of either m or n. We have also presented a result that relates synchronizations of lines of processors to synchronizations of $(m \times n)$ arrays. In particular, we have proved that an $(m \times n)$ array of processors can be seen as many lines of $(m+n-1)$ processors where a synchronization can be executed simultaneously. This gives an algorithm synchronizing an $(m \times n)$ array in time $t(m+n-1)$, provided that an algorithm for a linear array of k processors in time $t(k)$ is already given.

An interesting future direction of research is the case when half duplex links are used to connect adjacent processors instead of full duplex connections. We think that one can obtain the same synchronization times as those presented in this paper. Moreover it maybe interesting to investigate also the minimal time synchronization in this framework.

References

[1] R. Balzer, An 8-states minimal time solution to the firing squad synchronization problem. *Inform. and Control* **10** (1967) 22-42.

[2] K. Culik, Variations of the firing squad problem and applications. *Inform. Process. Lett.* **30** (1989) 153-157.

[3] K. Imai and K. Morita, Firing squad synchronization problem in reversible cellular automata. *Theoret. Comput. Sci.* **165** (1996) 475-482.

[4] K. Imai, K. Morita and K. Sako, Firing squad synchronization problem in number-conserving cellular automata, in *Proc. of the IFIP Workshop on Cellular Automata.* Santiago, Chile (1998).

[5] K. Kobayashy, The Firing Squad Synchronization Problem for Two Dimensional Arrays. *Inform. and Control* **34** (1977) 153-157.

[6] K. Kobayashy, *On Time Optimal Solutions of the Two-Dimensional Firing Squad Synchronization Problem*, MFCS Workshop On Cellular Automata (1998).

[7] S. La Torre, M. Napoli and D. Parente, Synchronization of One-Way Connected Processors. *Complex Systems* **10** (1996) 239-255.

[8] S. La Torre, M. Napoli and D. Parente, Synchronization of a Line of Identical Processors at a Given Time. *Fund. Inform.* **34** (1998) 103-128.

[9] J. Mazoyer, A six states minimal time solution to the firing squad synchronization problem. *Theoret. Comput. Sci.* **50** (1987) 183-238.

[10] J. Mazoyer, On optimal solutions to the firing squad synchronization problem. *Theoret. Comput. Sci.* **168** (1996) 367-404.

[11] F. Minsky, *Computation: Finite and Infinite Machines.* Prentice-Hall (1967).

[12] E.F. Moore, *Sequential Machines, Selected Papers.* Addison-Wesley, Reading, Mass (1964).

[13] Y. Nishitani and N. Honda, The firing squad synchronization problem for graphs. *Theoret. Comput. Sci.* **14** (1981) 39-61.

[14] Z. Roka, The Firing Squad Synchronization Problem on Caley Graphs, in *Proc. of MFCS'95.* Prague, Czech Republic (1995). *Lecture Notes in Comput. Sci.* **969** (1995) 402-411.

[15] I. Shinair, Two and Three-Dimensional Firing Squad Synchronization Problems. *Inform. and Control* **24** (1974) 163-180.

[16] A. Waksman, An optimum solution to the firing squad synchronization problem. *Inform. and Control* **9** (1966) 66-78.

Communicated by J. Gruska.
Received August 10, 2000. Accepted February 16, 2001.