FSSP Algorithms for Square and Rectangular Arrays

Hiroshi Umeo

Abstract The synchronization in cellular automata has been known as firing squad synchronization problem (FSSP) since its development. The firing squad synchronization problem on cellular automata has been studied extensively for more than fifty years, and a rich variety of synchronization algorithms has been proposed not only for one-dimensional arrays but also for two-dimensional arrays. In the present paper, we focus our attention to the two-dimensional array synchronizers that can synchronize any square/rectangle arrays and construct a survey on recent developments in their designs and implementations of optimum-time and non-optimum-time synchronization algorithms for two-dimensional arrays.

1 Introduction

Synchronization of large scale networks is an important and fundamental computing primitive in parallel and distributed systems. We study a synchronization problem that gives a finite-state protocol for synchronizing cellular automata. The synchronization in cellular automata has been known as firing squad synchronization problem (FSSP) since its development, in which it was originally proposed by J. Myhill in Moore [8] to synchronize all parts of self-reproducing cellular automata. The problem has been studied extensively for more than 50 years [1–28].

In the present paper, we focus our attention to two-dimensional (2D) array synchronizers that can synchronize square/rectangle arrays and construct a survey on recent developments in designs and implementations of optimum-time and non-optimum-time synchronization algorithms for the two-dimensional arrays. Specifically, we attempt to consider the following questions:

- Is there any new 2D FSSP algorithm other than classical ones?
- What is the smallest 2D synchronizer?

H. Umeo (✉)
University of Osaka Electro-Communication,
Neyagawa-shi, Hastu-cho, 18-8, Osaka, 572-8530, Japan
e-mail: umeo@cyt.osakac.ac.jp

H.G. Bock et al. (eds.), *Modeling, Simulation and Optimization of Complex Processes - HPSC 2012*, DOI 10.1007/978-3-319-09063-4_20

- How can we synchronize 2D arrays with the general at any position?
- How do the algorithms compare with each other?
- Can we extend the 2D synchronizers proposed so far to three-dimensional arrays?

Generally speaking, in the design of 2D synchronizers, configurations of a one-dimensional synchronization algorithm are mapped onto a 2D array through a mapping scheme so that all of the cells on the 2D array would fall into a final synchronization state simultaneously. The mapping schemes we consider include a rotated L-shaped mapping, a zebra mapping, a diagonal mapping, and a one-sided recursive-halving marking based mapping. All mappings will be employed efficiently in the design of 2D FSSP algorithms for square and rectangle arrays. Due to the space available we omit the details of those algorithms and their implementations.

2 Firing Squad Synchronization Problem

2.1 FSSP on 1D Arrays

The firing squad synchronization problem (FSSP, for short) is formalized in terms of the model of cellular automata. Consider a one-dimensional (1D) array of finite state automata. All cells (except the end cells) are identical finite state automata. The array operates in lock-step mode such that the next state of each cell (except the end cells) is determined by both its own present state and the present states of its right and left neighbors. Thus, we assume the nearest left and right neighbors. All cells (*soldiers*), except one *general* cell, are initially in the *quiescent* state at time $t = 0$ and have the property whereby the next state of a quiescent cell having quiescent neighbors is the quiescent state. At time $t = 0$ the *general* cell is in the *fire-when-ready* state, which is an initiation signal to the array.

The FSSP is stated as follows: Given a 1D array of n identical cellular automata, including a *general* at one end that is activated at time $t = 0$, we want to design the automata $M = (\mathcal{Q}, \delta)$ such that, *at some future time*, all the cells will *simultaneously* and, *for the first time*, enter a special *firing* state, where $\mathcal{Q}$ is a finite state set and $\delta : \mathcal{Q}^3 \rightarrow \mathcal{Q}$ is a next-state function. The tricky part of the problem is that the same kind of soldier having a fixed number of states must be synchronized, regardless of length n of the array. The set of states and next state function must be independent of n. Figure 1 is a space-time diagram for the optimum-step firing squad synchronization algorithm. The general at left end emits at time $t = 0$ an infinite number of signals which propagate at $1/(2^{k+1}-1)$ speed, where k is positive integer. These signals meet with a reflected signal at half point, quarter points, ..., etc., denoted by $\odot$ in Fig. 1. It is noted that these cells indicated by $\odot$ are synchronized. By increasing the number of synchronized cells exponentially, eventually all of the cells are synchronized.

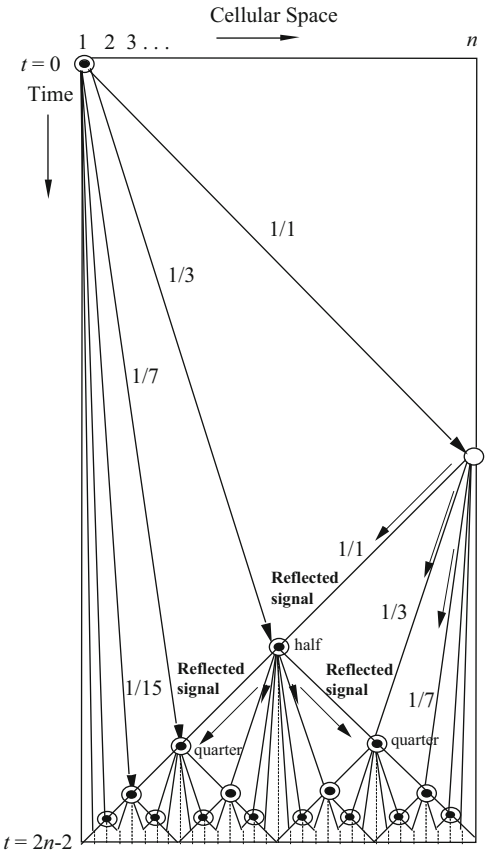

Fig. 1 Space-time diagram for optimum-time firing squad synchronization algorithm

The problem was first solved by J. McCarthy and M. Minsky who presented a $3n$-step algorithm for 1D cellular array of length n. In 1962, the first optimum-time, i.e. $(2n-2)$-step, synchronization algorithm was presented by Goto [4], with each cell having several thousands of states. Waksman [28] presented a 16-state optimum-time synchronization algorithm. Afterward, Balzer [1] and Gerken [3] developed an eight-state algorithm and a seven-state synchronization algorithm, respectively, thus decreasing the number of states required for the synchronization. Mazoyer [7] developed a six-state synchronization algorithm which, at present, is the algorithm having the fewest states for 1D arrays. In the sequel we use the following theorem as a base algorithm in the design of 2D array algorithms.

Theorem 1 (Goto [4], Waksman [28]). *There exists a cellular automaton that can synchronize any 1D array of length n in optimum* $2n-2$ *steps, where an initial general is located at a left or right end.*

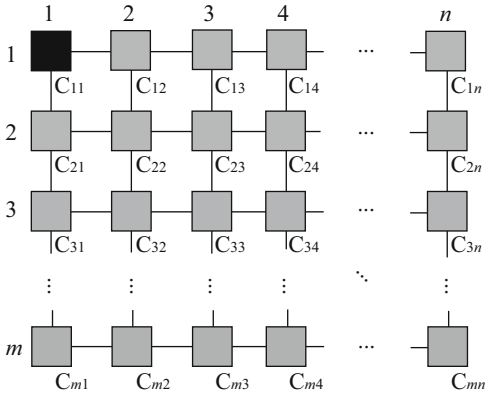

Fig. 2 A two-dimensional (2D) cellular automaton

2.2 FSSP on 2D Arrays

Figure 2 shows a finite 2D array consisting of $m \times n$ cells. Each cell is an identical (except the border cells) finite-state automaton. The array operates in lock-step mode in such a way that the next state of each cell (except border cells) is determined by both its own present state and the present states of its north, south, east and west neighbors. Thus, we assume the von Neumann-type four nearest neighbors. All cells (*soldiers*), except the north-west corner cell (*general*), are initially in the quiescent state at time $t = 0$ with the property that the next state of a quiescent cell with quiescent neighbors is the quiescent state again. At time $t = 0$, the north-west corner cell $C_{1,1}$ is in the *fire-when-ready* state, which is the initiation signal for synchronizing the array. The firing squad synchronization problem is to determine a description (state set and next-state function) for cells that ensures all cells enter the *fire* state at exactly the same time and for the first time.

A rich variety of synchronization algorithms for 2D arrays has been proposed. Concerning the rectangle synchronizers, see Beyer [2], Shinahr [10], Schmid [9], Szwerinski [11], Umeo [12], Umeo [13], and Umeo, Hisaoka, and Akiguchi [14]. As for square synchronization which is a special class of rectangles, several square synchronization algorithms have been proposed by Beyer [2], Shinahr [10], and Umeo, Maeda, and Fujiwara [20]. In recent years, Umeo and Kubo [18] developed a seven-state square synchronizer, which is a smallest implementation of the optimum-time square FSSP algorithm, known at present. One can easily see that it takes $2n - 2$ steps for any signal to travel from $C_{1,1}$ to $C_{n,n}$ due to the von Neumann neighborhood. Concerning the time optimality of the two-dimensional square synchronization algorithms, the following theorems have been established.

Theorem 2 (Beyer [2], Shinahr [10]). *There exists no 2D cellular automaton that can synchronize any square array of size $n \times n$ in less than $2n - 2$ steps, where the general is located at one corner of the array.*

Theorem 3 (Shinahr [10]). *There exists a 17-state cellular automaton that can synchronize any square array of size $n \times n$ at exactly $2n-2$ optimum steps.*

The lower bound of the time complexity for synchronizing rectangle arrays is as follows:

Theorem 4 (Beyer [2], Shinahr [10]). *There exists no cellular automaton that can synchronize any rectangle array of size $m \times n$ in less than $m+n+max(m,n)-3$ steps, where the general is located at one corner of the array.*

Theorem 5 (Beyer [2], Shinahr [10]). *There exists a cellular automaton that can synchronize any rectangle array of size $m \times n$ in exactly $m+n+max(m,n)-3$ steps, where the general is located at one corner of the array.*

3 Rotated L-Shaped Mapping Based Algorithm $\mathcal{A}_1$

The first 2D synchronization algorithm was developed independently by Beyer [2] and Shinahr [10]. It is based on a simple mapping which embeds a 1D optimum-time FSSP algorithm onto L-shaped sub-arrays composing a 2D array. We refer the embedding as *rotated L-shaped mapping*.

The algorithm for 2D square arrays operates as follows: By dividing an entire square array of size $n \times n$ into n rotated L-shaped 1D arrays, shown in Fig. 3 (left), in such a way that the length of the ith (from outside) L-shaped array is $2n-2i+1$ ($1 \leq i \leq n$). One treats the square synchronization as n independent 1D synchronizations with the general located at the bending point of the L-shaped array. We denote the ith L-shaped array by L_i and its horizontal and vertical segment is denoted by $\mathrm{L}_i^{\mathrm{h}}$ and $\mathrm{L}_i^{\mathrm{v}}$, respectively. Note that a cell at each bending point of the L-shaped array is shared for each synchronization by the two segments. See Fig. 3 (left). Concerning the synchronization of L_i, it can be easily seen that a general is generated by the cell $\mathrm{C}_{i,i}$ at time $t = 2i-2$ with the four nearest von-Neumann neighborhood communication, and the general initiates the horizontal (row) and vertical (column) synchronizations on $\mathrm{L}_i^{\mathrm{h}}$ and $\mathrm{L}_i^{\mathrm{v}}$, each of length $n-i+1$ using an optimum-time synchronization algorithm which can synchronize arrays of length ℓ in $2\ell-2$ steps (Theorem 1). For each i, $1 \leq i \leq n$, the ith L-shaped array L_i can be synchronized at time $t = 2i-2+2(n-i+1)-2 = 2n-2$. Thus the square array of size $n \times n$ can be synchronized at time $t = 2n-2$ in optimum-steps. In Fig. 3 (left), each general is represented by a black circle • in a shaded square and a wake-up signal for the synchronization generated by the general is indicated by a horizontal and vertical arrow. Shinahr [10] gave a 17-state implementation based on Balzer's eight-state synchronization algorithm (Balzer [1]). Later, it has been shown in Umeo, Maeda and Fujiwara [20] that nine states are sufficient for the optimum-time square synchronization:

Theorem 6 (Umeo, Maeda, and Fujiwara [20]). *There exists a nine-state 2D CA that can synchronize any $n \times n$ square array in $2n-2$ steps.*

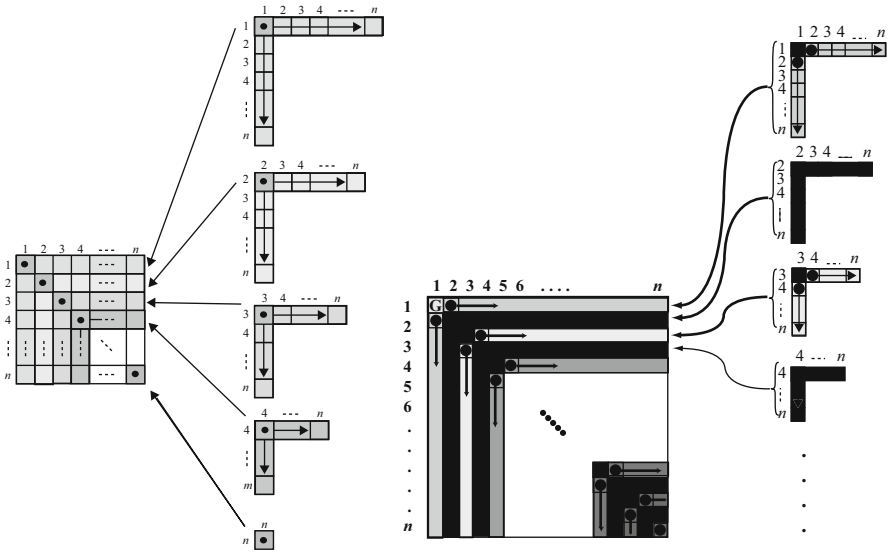

Fig. 3 A synchronization scheme based on *rotated L-shaped mapping* for $n \times n$ square cellular automaton (*left*) and *zebra mapping* for $n \times n$ square cellular automaton (*right*)

The first optimum-time rectangle synchronization algorithm was developed by Beyer [2] and Shinar [10] based on the rotated L-shaped mapping. The rectangular array of size $m \times n$ is regarded as $\min(m,n)$ rotated L-shaped 1D arrays, where they are synchronized independently using the *generalized* firing squad synchronization algorithm. The configurations of the generalized synchronization on 1D array are mapped onto the 2D array. Thus, an $m \times n$ array synchronization problem is reduced to independent $\min(m,n)$ 1D generalized synchronization problems such that $\mathcal{P}(m, m+n-1), \mathcal{P}(m-1, m+n-3), \ldots, \mathcal{P}(1, n-m+1)$ in the case $m \leq n$ and $\mathcal{P}(m, m+n-1), \mathcal{P}(m-1, m+n-3), \ldots, \mathcal{P}(m-n+1, m-n+1)$ in the case $m > n$, where $\mathcal{P}(k, \ell)$ means the 1D generalized synchronization problem for ℓ cells with a general on the kth cell from left end. Beyer [2] and Shinahr [10] presented an optimum-time synchronization scheme in order to synchronize any $m \times n$ arrays in $m + n + max(m,n) - 3$ steps. Shinahr [10] has given a 28-state implementation. Umeo, Ishida, Tachibana, and Kamikawa [16] gave a precise construction of the 28-state automaton having 12,849 rules.

Theorem 7 (Shinahr [10]). *There exists a 28-state cellular automaton that can synchronize any $m \times n$ rectangular arrays in optimum-time $m + n + max(m,n) - 3$ steps.*

4 Zebra Mapping Based Algorithm $\mathcal{A}_2$

In this section we first consider a state-efficient optimum-time square synchronization algorithm $\mathcal{A}_2$ proposed in Umeo and Kubo [18]. The algorithm is a variant of the L-shaped mapping. We show that seven states are sufficient for the optimum-time square synchronization. The proposed algorithm is basically based on the rotated L-shaped mapping scheme presented in the previous section. However, it is quite different from it in the following points. The mapping onto square arrays consists of two types of configurations: one is a one-cell smaller synchronized configuration and the other is a filled-in configuration with a stationary state. The stationary state remains unchanged once filled-in by the time before the final synchronization. Each configuration is mapped alternatively onto an L-shaped array *in a zebra-like fashion*. The mapping is referred to as *zebra mapping*. Figure 3 (right) illustrates the zebra mapping which consists of an embedded synchronization layer and a filled-in layer. In our construction we take the Mazoyer's 6-state synchronization rule as an embedded synchronization algorithm. See Mazoyer [7] for the six-state transition rule set. Figure 4 shows some snapshots of the synchronization process operating in optimum-steps on a 13×13 square array. The readers can see how those two types of configurations are mapped in the zebra-like fashion. The constructed seven-state cellular automaton has 787 transition rules, which can be found in Umeo and Kubo [18].

Theorem 8 (Umeo and Kubo [18]). *The seven-state square synchronization algorithm $\mathcal{A}_2$ can synchronize any $n \times n$ square array in optimum $2n-2$ steps.*

As for the rectangular arrays, Umeo and Nomura [23] constructed a ten-state 1629-rule 2D cellular automaton that can synchronize any $m \times n$ rectangle arrays in $m+n+\max(m,n)-2$ steps. See Fig. 5 for its snapshots. Note that the time complexity is one step larger than optimum.

Theorem 9 (Umeo and Nomura [23]). *There exists a ten-state 2D CA that can synchronize any $m \times n$ rectangle arrays in $m+n+\max(m,n)-2$ steps.*

5 Diagonal Mapping Based Algorithm $\mathcal{A}_3$

In this section we study a synchronization algorithm based on diagonal mapping. With the diagonal mapping, configurations of 1D cellular array can be embedded onto a square/rectangle array divided along the principal diagonal. Here we give a decomposition of a square array. We divide n^2 cells on a square array of size $n \times n$ into $2n-1$ groups g_k, $-(n-1) \leq k \leq n-1$ along the principal diagonal such that

$$g_k = \{C_{i,j} \mid j - i = k\},\ -(n-1) \leq k \leq n-1.$$

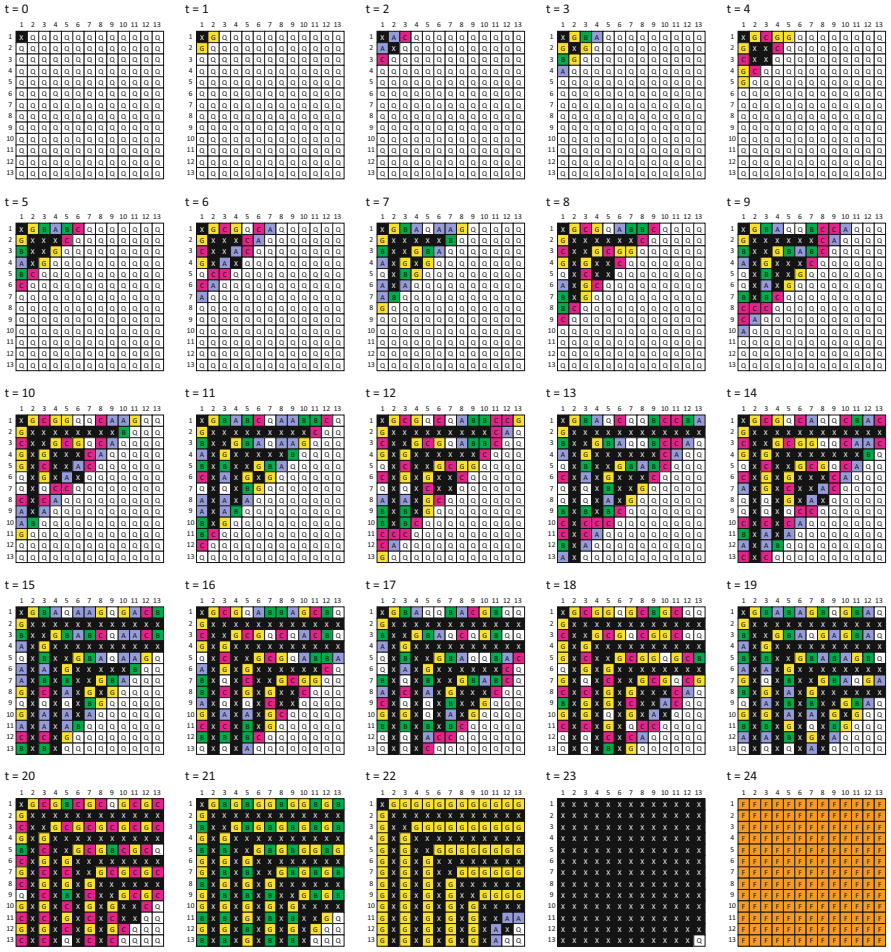

Fig. 4 Snapshots of the seven-state zebra-type square synchronizer on a 13×13 array

Each cell in g_k on the 2D square simulates the state of its corresponding cell C_k in the 1D array of length $2n-1$, $-(n-1) \leq k \leq n-1$. It has been shown in Umeo, Hisaoka, and Akiguchi [14] that any 1D generalized FSSP algorithm with an Inner-Independent Property $\mathcal{Z}$ (below) can be easily embedded onto 2D rectangle arrays *without introducing additional states*. The statement can also be applied to square arrays.

Inner-Independent Property $\mathcal{Z}$: Let S_i^t denote the state of C_i at step t. We say that an FSSP algorithm has *Inner-Independent Property $\mathcal{Z}$*, where any state S_i^t appearing *in the area $\mathcal{Z}$* can be computed from its left and right neighbor states S_{i-1}^{t-1} and S_{i+1}^{t-1} but it never depends on its own previous state S_i^{t-1}.

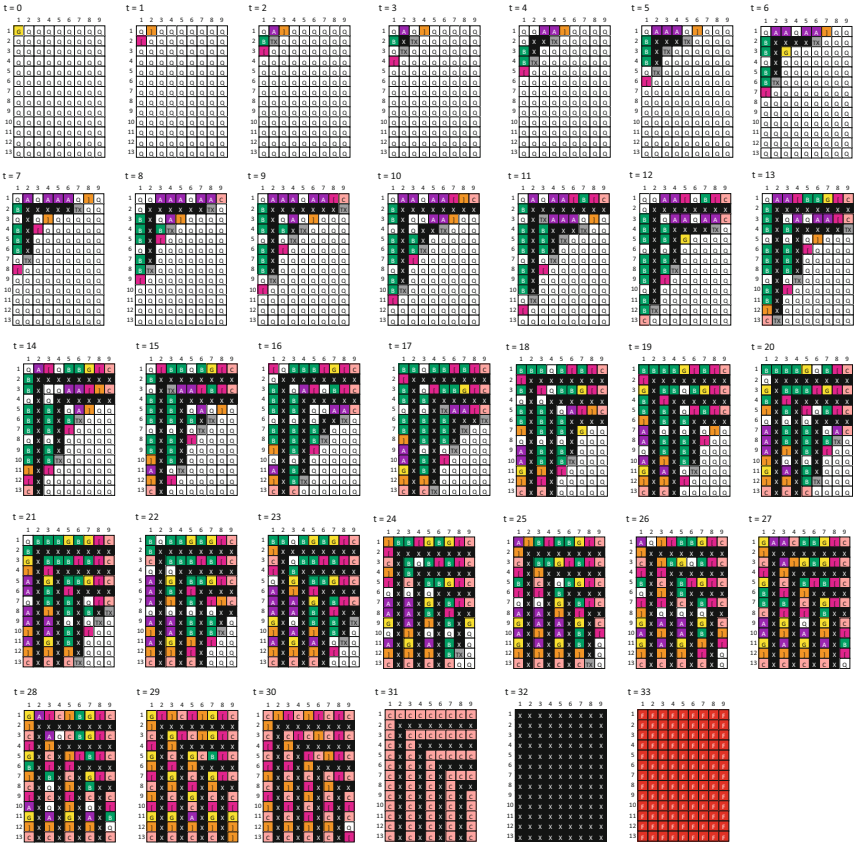

Fig. 5 Snapshots of the ten-state zebra-type rectangle synchronizer on a 13 × 9 array

A special 15-state generalized FSSP algorithm with the *Inner-Independent Property* $\mathcal{Z}$ can be realized in Ishii, Yanase, Maeda, and Umeo [6]. The 15-state algorithm with the property $\mathcal{Z}$ can be embedded on any square arrays without introducing additional states, yielding a 15-state optimum-time square synchronization algorithm.

Theorem 10 (Ishii, Yanase, Maeda, and Umeo [6]). *There exists a 15-state cellular automaton that can synchronize any* $n \times n$ *square array in optimum* $2n - 2$ *steps.*

As for the rectangle case, Umeo, Hisaoka, and Akiguchi [14] constructed a 12-state optimum-time rectangle synchronizer. See Umeo, Hisaoka, and Akiguchi [14] for details.

Theorem 11 (Umeo, Hisaoka, and Akiguchi [14]). *There exists a 12-state cellular automaton that can synchronize any $m \times n$ square array in optimum $m + n + \max(m, n) - 3$ steps.*

6 One-Sided Recursive-Halving Marking Based Algorithm $\mathcal{A}_4$

In this section we present an optimum-time synchronization algorithm $\mathcal{A}_4$ which is based on a marking called *one-sided recursive-halving marking*. The marking scheme prints a special mark on cells in a cellular space defined by one-sided recursive-halving. The marking itself is based on a well-known optimum-time one-dimensional synchronization algorithm. Let S be a one-dimensional cellular space consisting of cells C_i, C_{i+1}, ..., C_j, denoted by $[i \ldots j]$, where $j > i$. Let $|S|$ denote the number of cells in S, that is $|S| = j - i + 1$ for $S = [i \ldots j]$. A cell $C_{(i+j)/2}$ in S is a center cell of S, if $|S|$ is odd. Otherwise, two cells $C_{(i+j-1)/2}$ and $C_{(i+j+1)/2}$ are center cells of S.

The one-sided recursive-halving marking for a given 1D cellular space $[1 \ldots n]$ is defined as follows:

One-Sided Recursive-Halving Marking

```
begin
  S := [1...n];
  while |S| > 1 do
    if |S| is odd then
      mark a center cell C_x in S
      S := [x...n];
    else
      mark center cells C_x and C_{x+1} in S
      S := [x + 1...n];
end
```

We have developed a simple implementation of the one-sided recursive halving marking on a 13-state cellular automaton. It can be easily seen that any 1D cellular space of length n with the one-sided recursive-halving marking initially can be synchronized in optimum $n - 1$ steps.

Now we consider a square array of size $n \times n$ with an initial general G on $C_{1,1}$. The square is regarded as consisting of two triangles: upper and lower halves separated by a diagonal, shown in Fig. 6 (left). Each upper and lower half triangle consists of n columns and n rows, each denoted by c_k and r_k, $1 \leq k \leq n$, such that:

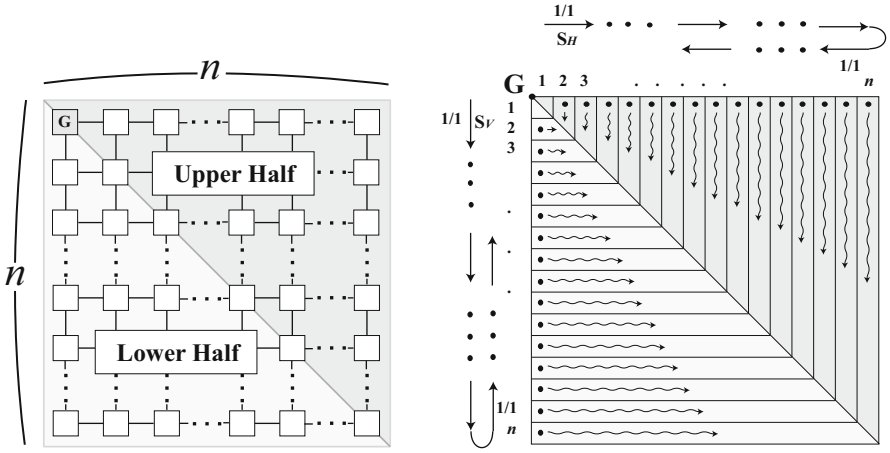

Fig. 6 A square array is decomposed into an upper and lower triangle (*left*) and an illustration of the synchronization scheme in each triangle (*right*)

$$c_k = \{C_{i,k} \mid 1 \leq i \leq k\}, \qquad r_k = \{C_{k,j} \mid 1 \leq j \leq k\}.$$

Note that the length of c_k and r_k is k for $1 \leq k \leq n$. An overview of the algorithm $\mathcal{A}_4$ is:

- Each upper and lower half triangle is synchronized independently.
- At time $t = 0$ the array begins to prepare printing the one-sided recursive halving mark on each column and row, each starting from top of each column and a left end of each row, respectively, in the triangles. The marking operation will be finished before the arrival of the first wake-up signal for the synchronization.
- Simultaneously, the general generates two signals $\mathbf{s}_H$ and $\mathbf{s}_V$ at time $t = 0$. Their operations are as follows:
  - **Signal $\mathbf{s}_H$**: The s_H-signal travels along the first row at 1/1-speed and reaches $C_{1,n}$ at time $t = n - 1$. Then it reflects there and returns the same route at 1/1-speed, and reaches $C_{1,1}$ again at time $t = 2n - 2$. On the return way, it generates a general on $C_{1,i}$ at time $t = n - 1 + n - (i - 1) = 2n - i$, at every visit of $C_{1,i}$, where $1 \leq i \leq n$. See Fig. 6 (right). The general is denoted by a black circle •. A ripple-like line, staring from the symbol •, shown in Fig. 6 (right), illustrates the initiation of the synchronization process initiated by the general. The general initiates a synchronization for the ith column, and yields a successful synchronization at time $t = 2n - 2$. Note that the length of the ith column is i and the synchronization is started at time $t = 2n - i$, for any $1 \leq i \leq n$. In this way, the upper half triangle can be synchronized in $2n - 2$ steps.
  - **Signal $\mathbf{s}_V$**: The s_V-signal travels along the 1st column at 1/1-speed and reaches $C_{n,1}$ at time $t = n - 1$. Then it reflects there and returns the same

Table 1 A list of FSSP algorithms for square arrays

Algorithms & implementations	# of states	# of rules	Time complexity	Communication model	Mapping
Beyer [2] Algorithm A_1	—	—	$2n-2$	O(1)-bit	L-shaped
Shinahr [10] Algorithm A_1	17	—	$2n-2$	O(1)-bit	L-shaped
Umeo, Maeda and Fujiwara [20] Algorithm A_1	9	1718	$2n-2$	O(1)-bit	L-shaped
Umeo and Kubo [18] Algorithm A_2	7	787	$2n-2$	O(1)-bit	Zebra
Umeo, Maeda, Hisaoka, and Teraoka [21] Algorithm A_3	6	942	$4n-4$	O(1)-bit	Diagonal
Ishii et al. [6] Algorithm A_3	15	1614	$2n-2$	O(1)-bit	Diagonal
Umeo, Uchino and Nomura [25] Algorithm A_4	37	3271	$2n-2$	O(1)-bit	Recursive-Halving
Gruska, Torre and Parente [5] Algorithm A_1	—	—	$2n-2$	1-bit	L-shaped
Umeo and Yanagihara [26] Algorithm A_1	49	237	$2n-2$	1-bit	L-shaped

route at 1/1-speed, and reaches $C_{1,1}$ again at time $t = 2n-2$. On the return way, it generates a general on $C_{i,1}$ at time $t = n-1+n-(i-1) = 2n-i$, at every visit of $C_{i,1}$, where $1 \leq i \leq n$. The general initiates a synchronization for the ith row. Note that the length of the ith row is i. The ith row can be synchronized at time $t = 2n-2$, for any i, $1 \leq i \leq n$. Thus, the lower half triangle can be synchronized in $2n-2$ steps.

We implemented the algorithm $\mathcal{A}_4$ on a 2D cellular automaton. The constructed cellular automaton has 37 internal states and 3,271 transition rules. Thus we have:

Theorem 12 (Umeo, Uchino, and Nomura [25]). *The synchronization algorithm $\mathcal{A}_4$ can synchronize any $n \times n$ square array in optimum $2n-2$ steps.*

As for the rectangle case, Umeo, Nishide, and Yamawaki [22] constructed a 2D cellular automaton with 384-states and 112,690-rules. See Umeo, Nishide, and Yamawaki [22] for details.

Table 2 A list of FSSP algorithms for rectangle arrays

Algorithms & implementations	# of states	# of rules	Time complexity	Communication model	Mapping
Beyer [2] Algorithm A_1	—	—	$m+n+max(m,n)-3$	O(1)-bit	L-shaped
Shinahr [10]	28	—	$m+n+max(m,n)-3$	O(1)-bit	L-shaped
Umeo et al. [16] Algorithm A_1	28	12849*			
Umeo and Nomura [23] Algorithm A_2	10	1629	$m+n+max(m,n)-2$	O(1)-bit	Zebra
Umeo, Hisaoka and Akiguchi [14] Algorithm A_3	12	1532	$m+n+max(m,n)-3$	O(1)-bit	Diagonal
Umeo, Maeda, Hisaoka, and Teraoka [21] Algorithm A_3	6	942	$2m+2n-4$	O(1)-bit	Diagonal
Umeo, Nishide and Yamawaki [22] Algorithm A_4	384	112690	$m+n+max(m,n)-3$	O(1)-bit	Recursive-Halving

Theorem 13 (Umeo, Nishide, and Yamawaki [22]). *The algorithm $\mathcal{A}_4$ can synchronize any $m \times n$ rectangular array in $m+n+max(m,n)-3$ optimum steps.*

7 Conclusions

In this paper, we have presented a survey on recent developments of optimum-time and non-optimum-time FSSP algorithms for 2D arrays. In Tables 1 and 2 we present a list of implementations of square/rectangle FSSP algorithms for cellular automata with O(1)-bit and 1-bit communications.

The O(1)-bit communication model, discussed in this paper, is a usual cellular automaton in which the amount of communication bits exchanged in one step between neighboring cells is assumed to be O(1) bits. The 1-bit communication model is a subclass of the O(1)-bit model, in which inter-cell communication is restricted to 1-bit communication.

For a long time only one FSSP algorithm based on the rotated L-shaped mapping proposed by Beyer [2] and Shinar [10] has been known. The readers can see how a rich variety of 2D FSSP algorithms exists. Some algorithms can be easily extended to 3D arrays. The embedding schemes developed in this paper would be useful for further implementations of multi-dimensional synchronization algorithms.

Acknowledgements A part of this work is supported by Grant-in-Aid for Scientific Research (C) 21500023.

References

1. Balzer, R.: An 8-state minimal time solution to the firing squad synchronization problem. Inf. Control **10**, 22–42 (1967)
2. Beyer, W.T.: Recognition of topological invariants by iterative arrays. Ph.D. Thesis, MIT, pp. 144 (1969)
3. Gerken, H.D.: Über Synchronisations problem bei Zellularautomaten, Diplomarbeiten, Institut für Theoretische Informatik, Technische Universität Braunschweig, pp. 50, (1987)
4. Goto, E.: A minimal time solution of the firing squad problem. Dittoed Course Notes for Applied Mathematics 298, Harvard University, pp. 52–59 (1962)
5. Gruska, J., Torre, S.L., Parente, M.: The firing squad synchronization problem on squares, toruses and rings. Int. J. Found. Comput. Sci. **18**(3), 637–654 (2007)
6. Ishii, S., Yanase, H., Maeda, M., Umeo, H.: State-efficient implementations of time-optimum synchronization algorithms for square arrays. Technical Report of IEICE, Circuit and Systems, pp. 13–18 (2006)
7. Mazoyer, J.: A six-state minimal time solution to the firing squad synchronization problem. Theor. Comput. Sci. **50**, 183–238 (1987)
8. Moore, E.F.: The firing squad synchronization problem. In: Moore, E.F., (Ed.) Sequential Machines, Selected Papers, pp. 213–214. Addison-Wesley, Reading (1964)
9. Schmid, H.: Synchronisationsprobleme für zelluläre Automaten mit mehreren Generälen. Diplomarbeit, Universität Karsruhe, (2003)
10. Shinahr, I.: Two- and three-dimensional firing squad synchronization problems. Inf. Control **24**, 163–180 (1974)
11. Szwerinski, H.: Time-optimum solution of the firing-squad-synchronization-problem for n-dimensional rectangles with the general at an arbitrary position. Theor. Comput. Sci. **19**, 305–320 (1982)
12. Umeo, H.: Firing squad synchronization algorithms for two-dimensional cellular automata. J. Cell. Autom. **4**, 1–20, (2008)
13. Umeo, H.: Firing squad synchronization problem in cellular automata. In: Meyers, R.A. (Ed.) Encyclopedia of Complexity and System Science, vol. 4, pp. 3537–3574. Springer, Berlin, Heidelberg (2009)
14. Umeo, H., Hisaoka, M., Akiguchi, S.: Twelve-state optimum-time synchronization algorithm for two-dimensional rectangular cellular arrays. In: Proceedings of 4th International Conference on Unconventional Computing: UC 2005, Sevilla. LNCS 3699, pp. 214–223 (2005)
15. Umeo, H., Hisaoka, M., Teraoka, M., Maeda, M.: Several new generalized linear- and optimum-time synchronization algorithms for two-dimensional rectangular arrays. In: Margenstern, M. (Ed.) Proceedings of 4th International Conference on Machines, Computations and Universality: MCU 2004, Saint Petersburg. LNCS 3354, pp. 223–232 (2005)
16. Umeo, H., Ishida, K., Tachibana, K., Kamikawa, N.: A transition rule set for the first 2-D optimum-time synchronization algorithm. In: Proceedings of the 4th International Workshop on Natural Computing, PICT 2, Himeji, pp. 333–341. Springer (2009)
17. Umeo, H., Kamikawa, N., Nishioka, K., Akiguchi, S.: Generalized firing squad synchronization protocols for one-dimensional cellular automata – a survey. Acta Phys. Pol. B, Proc. Suppl. **3**, 267–289 (2010)
18. Umeo, H., Kubo, K.: A seven-state time-optimum square synchronizer. In: Proceedings of the 9th International Conference on Cellular Automata for Research and Industry, Ascoli Piceno. LNCS 6350, pp. 219–230. Springer (2010)

19. Umeo, H., Kubo, K.: Recent developments in constructing square synchronizers. In: Proceedings of the 10th International Conference on Cellular Automata for Research and Industry, Santorini. LNCS 7495, pp. 171–183. Springer (2012)
20. Umeo, H., Maeda, M., Fujiwara, N.: An efficient mapping scheme for embedding any one-dimensional firing squad synchronization algorithm onto two-dimensional arrays. In: Proceedings of the 5th International Conference on Cellular Automata for Research and Industry, Geneva. LNCS 2493, pp. 69–81. Springer (2002)
21. Umeo, H., Maeda, M., Hisaoka, M., Teraoka, M.: A state-efficient mapping scheme for designing two-dimensional firing squad synchronization algorithms. Fundam. Inform. **74**, 603–623 (2006)
22. Umeo, H., Nishide, K., Yamawaki, T.: A new optimum-time firing squad synchronization algorithm for two-dimensional rectangle arrays – one-sided recursive halving based. In: Löwe, B. et al. (Eds.) Proceedings of the International Conference on Models of Computation in Context, Computability in Europe 2011, CiE 2011, Sofia. LNCS 6735, pp. 290–299 (2011)
23. Umeo, H., Nomura, A.: Zebra-like mapping for state-efficient implementation of two-dimensional synchronization algorithms (2014, manuscript in preparation)
24. Umeo, H., Uchino, H.: A new time-optimum synchronization algorithm for rectangle arrays. Fundam. Inform. **87**(2), 155–164 (2008)
25. Umeo, H., Uchino, H., Nomura, A.: How to synchronize square arrays in optimum-time. In: Proceedings of the 2011 International Conference on High Performance Computing and Simulation (HPCS 2011), Istanbul, pp. 801–807. IEEE (2011)
26. Umeo, H., Yanagihara, T.: Smallest implementations of optimum-time firing squad synchronization algorithms for one-bit-communication cellular automata. In: Proceedings of the 2011 International Conference on Parallel Computing and Technology, PaCT 2011, Kazan. LNCS 6873, pp. 210–223 (2011)
27. Umeo, H., Yamawaki, T., Shimizu, N., Uchino, H.: Modeling and simulation of global synchronization processes for large-scale-of two-dimensional cellular arrays. In: Proceedings of International Conference on Modeling and Simulation, AMS 2007, Phuket, pp. 139–144 (2007)
28. Waksman, A.. An optimum solution to the firing squad synchronization problem. Inf. Control **9**, 66–78 (1966)