A New Optimum-Time Firing Squad Synchronization Algorithm for Two-Dimensional Rectangle Arrays

- One-Sided Recursive Halving Based -

Hiroshi Umeo[1], Kinuo Nishide[1] and Takuya Yamawaki[1]

[1]Univ. of Osaka Electro-Communication,
Neyagawa-shi, Hastu-cho, 18-8, Osaka, 572-8530, Japan
Corresponding e-mail address: umeo@cyt.osakac.ac.jp

Abstract. The firing squad synchronization problem on cellular automata has been studied extensively for more than fifty years, and a rich variety of synchronization algorithms have been proposed for not only one-dimensional arrays but two-dimensional arrays. In the present paper, we propose a new optimum-time synchronization algorithm that can synchronize any two-dimensional rectangle arrays of size $m \times n$ with a general at one corner in $m+n+\max(m,n)-3$ steps. The algorithm is based on a simple recursive halving marking schema which helps synchronization operations on two-dimensional arrays. A proposed computer-assisted implementation of the algorithm gives a description of a two-dimensional cellular automaton in terms of a finite 384-state set and a local 112690-rule set.

Keywords. cellular automaton, firing squad synchronization problem, optimum-time rectangle synchronization algorithm

1 Introduction

We study a synchronization problem that gives a finite-state protocol for synchronizing a large scale of cellular automata. The synchronization in cellular automata has been known as a firing squad synchronization problem (FSSP) since its development, in which it was originally proposed by J. Myhill in Moore [1964] to synchronize all parts of self-reproducing cellular automata. The problem has been studied extensively for more than 50 years [1-18].

In the present paper, we propose a new time-optimum synchronization algorithm for rectangle cellular automata. The algorithm can synchronize any rectangle arrays of size $m \times n$ with a general at one corner in $m+n+\max(m,n)-3$ steps. The algorithm is based on a simple marking schema which prints a special mark in a given cellular space. We also give a computer-assisted implementation of the algorithm operating on a two-dimensional cellular automaton with 384-state and 112690-rule. Our computer simulation shows that the state and rule sets presented is valid for the synchronization on any rectangle arrays of size $m \times n$ such that $2 \leq m, n \leq 253$.

2 Firing Squad Synchronization Problem on Two-Dimensional Arrays

Figure 1 shows a finite two-dimensional (2-D) cellular array consisting of $m \times n$ cells, that is, m rows and n columns, each denoted by $\mathrm{C}_{ij}, 1 \leq i \leq m, 1 \leq j \leq n$. 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 well-known *von Neumann neighborhood*. 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_{11} is in the *fire-when-ready* state, which is the initiation signal of the synchronization for the array. The firing squad synchronization problem (FSSP) is to determine a description (a state set $\mathcal{Q}$ and a next-state function such that $\delta : \mathcal{Q}^5 \rightarrow \mathcal{Q}$) for cells that ensures all cells enter the *fire* state at exactly the same time and for the first time. The tricky part of the problem is that the same kind of soldier having a fixed number of states must be synchronized, regardless of the size $m \times n$ of the array. The set of states and next state function must be independent of m and n.

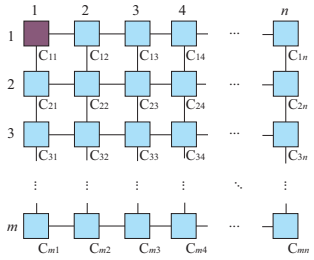

Fig. 1. A two-dimensional cellular automaton.

Several synchronization algorithms on 2-D arrays have been proposed by Beyer [1969], Grasselli [1975], Shinahr [1974], Szwerinski [1982], Umeo, Maeda, Hisaoka and Teraoka [2006], and Umeo and Uchino [2008]. It has been shown by Beyer [1969] and Shinahr [1974] independently that there exists no two-dimensional cellular automaton that can synchronize any 2-D array of size $m \times n$ in less than $m + n + \max(m, n) - 3$ steps. In addition they first proposed an optimum-time synchronization algorithm that can synchronize any 2-D array of size $m \times n$ in optimum $m+n+\max(m, n)-3$ steps. Shinahr [1974] gave a 28-state implementation. Umeo, Hisaoka and Akiguchi [2005] presented a new 12-state synchronization algorithm operating in optimum-step, realizing the smallest,

in the number of states, solution to the rectangle synchronization problem at present.

3 One-Sided Recursive Halving Mark

In this section, we develop a marking schema for one-dimensional arrays, which is referred to as *one-sided recursive halving mark*. The schema prints a special mark on cells in a given cellular space, which is 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 $[i...j]$, where $j > i$. Let $|S|$ denote the number of cells in S, that is, $|S| = j - i + 1$. The center cell(s) C_x of S is defined by

$$x = \begin{cases} (i+j)/2 & |S|\text{: odd,} \\ (i+j-1)/2, (i+j+1)/2 & |S|\text{: even.} \end{cases} \tag{1}$$

Note that we have two center cells when $|S|$ is even. The one-sided recursive halving mark for a given cellular space $[1...n]$ is defined as follows:

One-sided recursive halving mark

```
begin
    S := [1...n];
    while |S| ≥ 2 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 call the schema one-sided recursive halving mark. For example, we consider a cellular space $S = [1...15]$ consisting of 15 cells. The first center cell is C_8, then the second one is C_{11} and C_{12}, and the last one is C_{13} and C_{14}, respectively. In case $S = [1...17]$, we get C_9, C_{13}, C_{15}, and C_{16} after four iterations.

The one-sided recursive halving marking schema can be realized on a cellular automaton. Figure 2 (left) shows a space-time diagram for the marking. At time $t = 0$, the leftmost cell C_1 emits two signals simultaneously, each propagating in the right direction at 1/1- and 1/3-speed. The 1/1-speed signal arrives at C_n at time $t = n - 1$. Then, the rightmost cell C_n generates an infinite set of signals

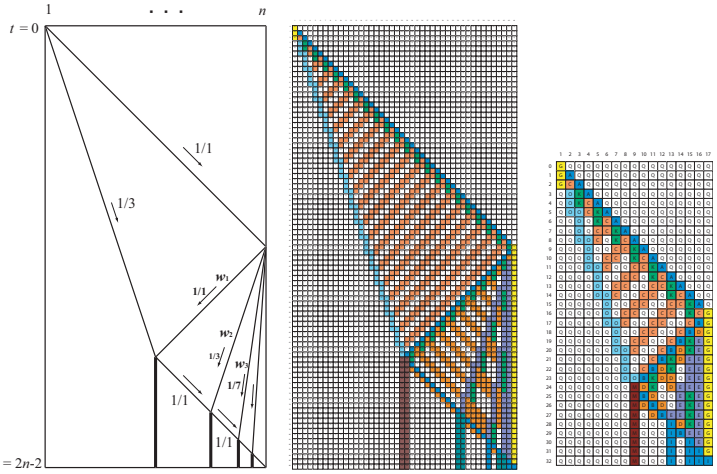

Fig. 2. Space-time diagram for one-sided recursive halving on one-dimensional array of length n (left) and some snapshots on 44 (middle) and 17 (right) cells, respectively, for the marking implemented on a 13-state cellular automaton.

$w_1, w_2, ..., w_k, ..$, each propagating in the left direction at $1/(2^k - 1)$ speed, where $k = 1, 2, 3, ...,$. The readers can find that each crossing, made by two signals of the first right-going 1/3-speed signal and with each $w_1, w_2, ..., w_k$, shown in Fig. 2 (left), enables the marking at middle points defined by the one-sided recursive halving. A finite state realization for generating the infinite set of signals above is a well-known technique used for the optimum-time synchronization algorithms on one-dimensional arrays in Balzer [1967], Gerken [1987], and Waksman [1966].

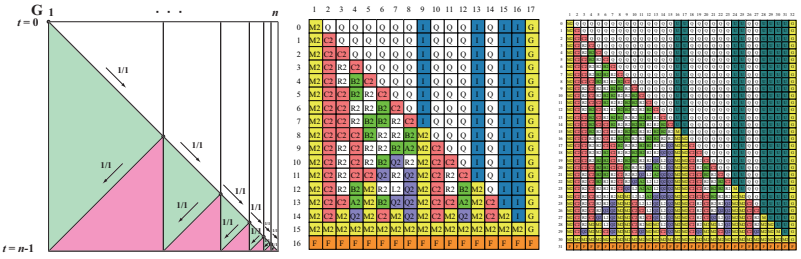

Fig. 3. Space-time diagram for synchronizing a cellular space with the one-sided recursive halving mark (left) and some snapshots for the synchronization on 17 (middle) and 32 (right) cells, respectively.

We have developed a simple implementation of the one-sided recursive halving marking on a 13-state cellular automaton. In Fig. 2 (middle and right) we present several snapshots for the marking on 44 and 17 cells, respectively. Thus we have:

Lemma 1 There exists a 13-state cellular automaton that can print the one-sided recursive halving mark in any cellular space of length n in $2n-2$ steps.

It can be seen that any cellular space of length n initially set up with a recursive halving mark and a general at left end can be synchronized in optimum $n-1$ steps. Figure 3 (left) shows the space-time diagram for synchronizing a cellular space with the one-sided recursive halving mark. Note that a general is located at left end of the cellular space with the marking. Several snapshots for the synchronization on 17 and 32 cells are shown in Fig. 3 (middle and right), respectively.

Lemma 2 Any cellular space of length n with the one-sided recursive halving mark can be synchronized in $n-1$ steps.

4 A New Two-Dimensional Optimum-Time Synchronization Algorithm

4.1 Overview of the Algorithm

In this section we give an overview of the synchronization algorithm $\mathcal{A}$ proposed for two-dimensional arrays. First we consider the algorithm which can synchronize any rectangle arrays wider than long, that is, $m \leq n$. A more generalized case will be discussed later.

An overview of the algorithm is as follows:

1. At time $t = 0$ an initial general on the north west corner emits a wake-up signal along the first row to synchronize the first row with a tentative firing state. The first row will be synchronized at time $t = 2n - 2$. The wake-up signal on the first row also acts as an activation signal for the one-sided recursive halving marking which prints special marks on each column. The wake-up signal reaches the ith column at time $t = i - 1$ and it begins the marking operation for the column. The recursive-halving making for the ith column will be finished at time $t = i - 1 + 2m - 2$. The computed time is due to Lemma 1.
2. Once the first row could be synchronized with the tentative firing state, then the cell C_{1i} in the tentative firing state initiates the synchronization for the ith column for each i such that $1 \leq i \leq n$. Using Lemma 2, the ith column will be synchronized at time $t = (2n-2) + (m-1) = m + 2n - 3 = m + n + \max(m, n) - 3$ in the case where $m \leq n$.

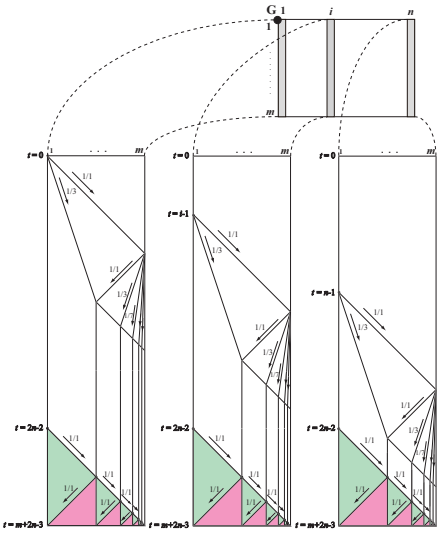

Fig. 4. Space-time diagrams for the wider-than-long column synchronization algorithm on the 1st, ith, and nth columns, respectively.

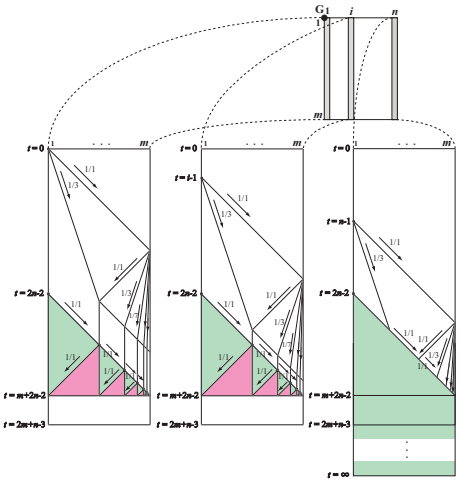

Fig. 5. Space-time diagram of the wider-than-long column synchronization algorithm, being applied to a rectangle longer than wide.

For any i such that $1 \leq i \leq n$, the first center cell(s) of the ith column is marked at time $t_1 = i-2+\lceil 3m/2 \rceil$ and the wake-up signal for the synchronization arrives at the center cell(s) of the ith column at time $t_2 = 2n - 2 + \lceil m/2 \rceil$. The first center cell(s) has been marked before the arrival of the wake-up signal in the case where $m \leq n$, since $t_2 - t_1 = 2n - m - i \geq 0$, for any i such that $1 \leq i \leq n$. Note that the signal propagation for the halving and the wake-up signal for the column synchronization are made at the same 1/1 speed. Thus, the synchronization can be performed successfully for each column and the array can be synchronized in optimum steps. We call the synchronization *wider-than-long column synchronization*. Figure 4 illustrates a space-time diagram for the wider-than-long column synchronization algorithm on the 1st, ith, and nth column, respectively. One can see that each marking operation has been finished before the arrival of the first synchronization signal on each column.

4.2 Applying the wider-than-long column synchronization algorithm to rectangles longer than wide

Note that the above synchronization scheme doesn't work for the rectangle longer than wide, because the pre-synchronization on the first row is finished too early. Consider a longer-than-wide rectangle of size $2n \times n$. The synchronization on the first column is successfully done, however the last nth column fails to synchronize, since the wake-up signal for the synchronization continues to look for the markings.

In the case of rectangles longer than wide, each horizontal segment, that is a shorter one in this case, falls into a firing state at time $t = 2n + m - 3$, which is smaller than the optimum-time $t = 2m + n - 3$. Moreover at time $t = 2n + m - 3$ some cells on some vertical segments are still in quiescent state. Thus the algorithm presented above fails to synchronize rectangles longer than wide. See Fig. 5. The figure illustrates a space-time diagram for those unsuccessful synchronization operations on the 1st, ith, and nth column when applying the wider-than-long column synchronization algorithm to a rectangle longer than wide.

4.3 Synchronization of rectangle longer than wide

The synchronization for the rectangles longer than wide can be done by interchanging the roles of row and column operations described above. By the similar way, the rectangle can be synchronized at time $t = n + 2m - 3 = m + n + \max(m, n) - 3$ in the case where $m \geq n$. We call the synchronization *longer-than-wide row synchronization*. By symmetry, the longer-than-wide row synchronization algorithm doesn't work for the rectangle wider than long.

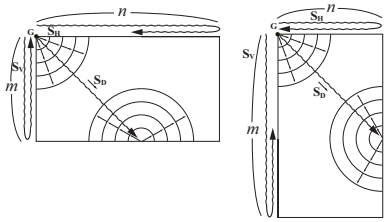

Fig. 6. A trajectory of inhibition signals generated by a general on a two-dimensional wider than long (left) and longer than wide (right) array of size $m \times n$. Three signals s_V, s_H, and s_D (denoted by bold arrows in the figure) for stopping undesirable synchronization operations are also illustrated.

4.4 Stopping of undesirable synchronization operations

The initial general has no a priori knowledge on the side length of rectangle arrays. To synchronize a given rectangle in optimum steps, the array, at time $t = 0$, begins to prepare both two synchronization operations, one for wider than long and the other for longer than wide rectangles. In order to check the type of the given rectangle, the array generates a zigzag signal s_D at time $t = 0$ which propagates at a unit speed along the principal diagonal. If the signal meets the lower (south) boundary, then the rectangle is wider than long and if it meets the right (east) boundary, then the rectangle is longer than wide. In addition, at time $t = 0$, the array also generates two signals, s_V and s_H, each propagates in the vertical and horizontal direction at 1/1-speed, respectively.

The general on C_{11} gets the side length information at time $t = 2m - 2$ or $t = 2n - 2$ by the first arrival of the return of the signal s_V or s_H. See Fig. 6. The array has to inhibit the operations of the column synchronization in the

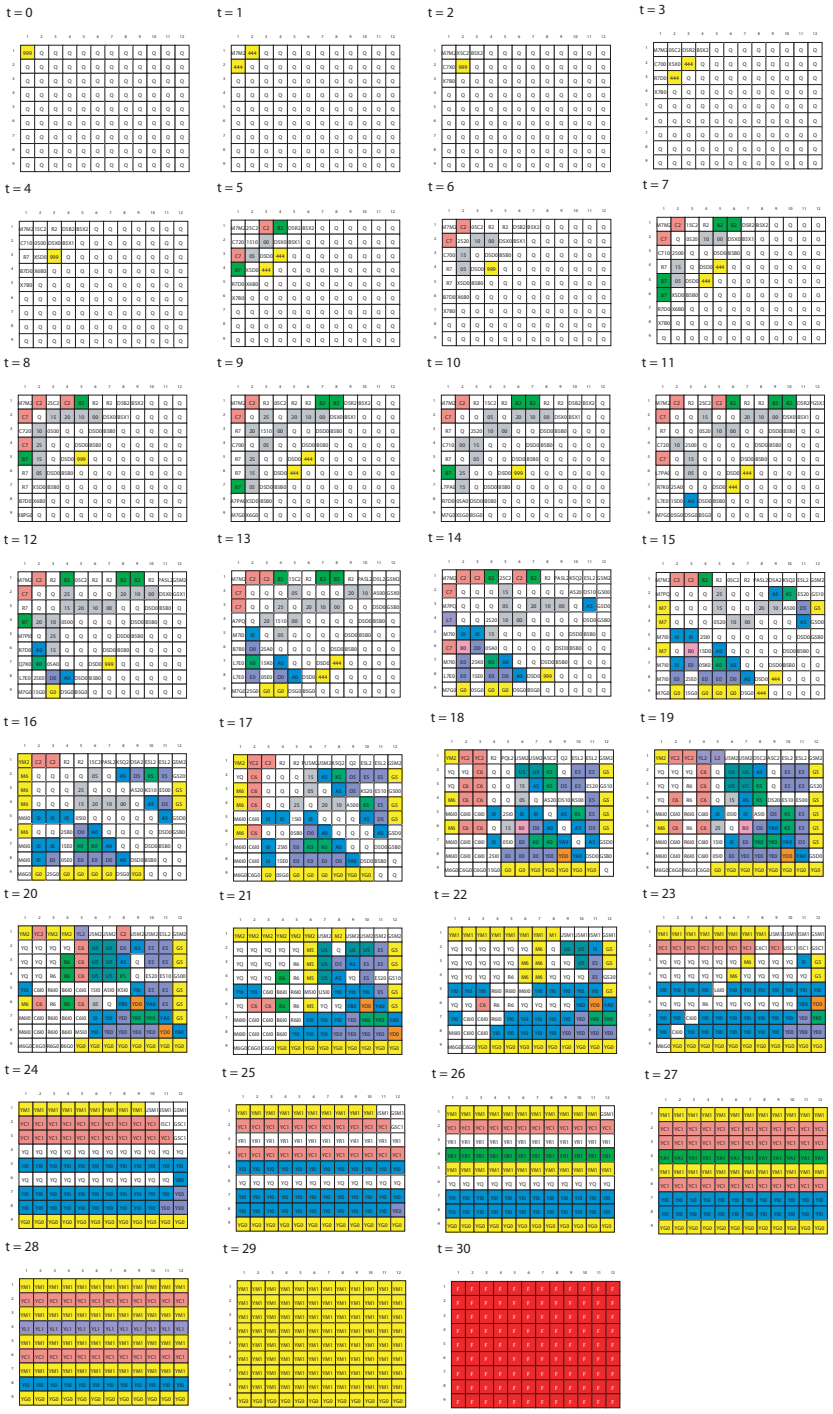

Fig. 7. Snapshots of the final synchronization algorithm on a 9×12 array.

case it is longer than wide and the row synchronization operations in the case wider than long. Otherwise some cells fall into a firing state before the final firing time. To inhibit the column synchronization, the diagonal signal generates a 1/1 speed signal when it hits the right boundary at time $t = 2n - 2$. The signal propagates horizontally to the left direction on the nth row and prints a special inhibition mark on each cell. The inhibition signal arrives at the left end at time $t = 3n - 3$. On the other hand, the wake-up signal for the column synchronization arrives at the nth row at time $t = 3n-3$, thus all of the wake-up signal can be stopped on that row and inhibit the column synchronization. As for the row synchronization, a similar technique can be employed.

An alternative way for the inhibition is to generate a signal at the cell C_{11} and C_{mm} (or C_{nn}), where the s_D-signal hits depending on the side length. The signal spreads at 1/1 speed in every direction like a circular wave, inhibiting undesirable synchronization operations. Figure 6 shows how the signal spreads over the array.

4.5 Final algorithm

Now we can establish the following theorem. We have also implemented the algorithm on a two-dimensional cellular automaton having 384 states and 112690 local rules. The rule set is generated by a computer program based on the algorithm $\mathcal{A}$. Our computer simulation shows that the rule set generated is valid for the synchronization on any rectangle of size $m \times n$ such that $2 \leq m, n \leq 253$. In Figure 7, we present some snapshots of the synchronization processes of the implementation, operating on a 9×12 array.

Theorem 3 The synchronization algorithm $\mathcal{A}$ can synchronize any $m \times n$ rectangular array in optimum $m + n + \max(m, n) - 3$ steps.

5 Conclusions

We have proposed a new optimum-time rectangle synchronization algorithm based on the one-sided recursive halving mark. It can synchronize any two-dimensional rectangle array of size $m \times n$ in optimum $m + n + \max(m, n) - 3$ steps. We have also implemented the algorithm on a two-dimensional cellular automaton having 384 states and 112690 local rules. A smaller-state implementation would not be so complicated.

Acknowledgements

The authors would like to express their thanks to reviewers for useful comments. A part of this work is supported by Grant-in-Aid for Scientific Research (C) 21500023.

References

1. R. Balzer: An 8-state minimal time solution to the firing squad synchronization problem. *Information and Control*, vol. 10 (1967), pp. 22-42.

2. W. T. Beyer: Recognition of topological invariants by iterative arrays. Ph.D. Thesis, MIT, (1969), pp. 144.
3. H. D. Gerken. (1987): Über Synchronisationsprobleme bei Zellularautomaten. *Diplomarbeit*, Institut für Theoretische Informatik, Technische Universität Braunschweig, pp. 50.
4. E. Goto: A minimal time solution of the firing squad problem. Dittoed course notes for Applied Mathematics 298, Harvard University, (1962), pp. 52-59.
5. J. Mazoyer: A six-state minimal time solution to the firing squad synchronization problem. *Theoretical Computer Science*, vol. 50 (1987), pp. 183-238.
6. E. F. Moore: The firing squad synchronization problem. in *Sequential Machines, Selected Papers* (E. F. Moore, ed.), Addison-Wesley, Reading MA.,(1964), pp. 213-214.
7. H. Schmid: Synchronisationsprobleme für zelluläre Automaten mit mehreren Generälen. Diplomarbeit, Universität Karsruhe, (2003).
8. I. Shinahr: Two- and three-dimensional firing squad synchronization problems. *Information and Control*, vol. 24(1974), pp. 163-180.
9. H. Szwerinski: Time-optimum solution of the firing-squad-synchronization-problem for n-dimensional rectangles with the general at an arbitrary position. *Theoretical Computer Science*, vol. 19(1982), pp. 305-320.
10. H. Umeo: A simple design of time-efficient firing squad synchronization algorithms with fault-tolerance. *IEICE Trans. on Information and Systems*, Vol. E87-D, No.3, 2004, pp.733-739.
11. H. Umeo: Firing squad synchronization problem in cellular automata. In *Encyclopedia of Complexity and System Science*, R. A. Meyers (Ed.), Springer, Vol.4(2009), pp.3537-3574.
12. H. Umeo, M. Hisaoka, and S. Akiguchi: Twelve-state optimum-time synchronization algorithm for two-dimensional rectangular cellular arrays. *Proc. of 4th International Conference on Unconventional Computing: UC 2005, LNCS 3699*, (2005), pp.214-223.
13. H. Umeo, M. Hisaoka and T. Sogabe: A survey on optimum-time firing squad synchronization algorithms for one-dimensional cellular automata. *Intern. J. of Unconventional Computing*, 2005, vol. 1, pp.403-426.
14. H. Umeo, M. Hisaoka, M. Teraoka and M. Maeda: Several new generalized linear- and optimum-time synchronization algorithms for two-dimensional rectangular arrays. *Proc. of 4th International Conference on Machines, Computations and Universality : MCU 2004, LNCS 3354* (M. Margenstern (Ed.), (2005), pp.223-232.
15. H. Umeo, M. Maeda, M. Hisaoka and M. Teraoka: A state-efficient mapping scheme for designing two-dimensional firing squad synchronization algorithms. *Fundamenta Informaticae*, Vol.74(2006), pp.603-623.
16. H. Umeo and H. Uchino: A new time-optimum synchronization algorithm for rectangle arrays. *Fundamenta Informaticae*, Vol.87, No.2, pp.155-164(2008).
17. H. Umeo, T. Yamawaki and K. Nishide: An optimum-time firing squad synchronization algorithm for two-dimensional rectangle arrays - freezing-thawing technique based -. *Proc. of the 2010 International Conference on High Performance Computing & Simulation (HPCS 2010)*, pp.575-581(2010).
18. A. Waksman: An optimum solution to the firing squad synchronization problem. *Information and Control*, vol. 9 (1966), pp. 66-78.