Original article Biomath 1 (2012), 1209022, 1–6

Biomath Forum

BIOMATH

http://www.biomathforum.org/biomath/index.php/biomath/

A State-Efficient Zebra-Like Implementation of Synchronization Algorithms for 2D Rectangular Cellular Arrays

Hiroshi Umeo* and Akira Nomura*
* Univ. of Osaka Electro-Communication
Neyagawa-shi, Hatsu-cho, 18–8, Osaka, 572–8530, Japan
Emails: umeo@cyt.osakac.ac.jp, nomura@cyt.osakac.ac.jp

Received: 15 July 2012, accepted: 2 September 2012, published: 12 October 2012

***Abstract*—The firing squad synchronization problem on cellular automata has been studied extensively for more than forty years, and a rich variety of synchronization algorithms have been proposed for not only one-dimensional (1D) but two-dimensional (2D) arrays. In the present paper, we propose a simple and state-efficient mapping scheme: *zebra-like mapping* for implementing 2D synchronization algorithms for rectangular arrays. The zebra-like mapping we propose embeds two types of configurations alternately onto a 2D array like a zebra-like pattern, one configuration is a synchronization configuration of 1D arrays and the other is a stationary configuration which keeps its state unchanged until the final synchronization. It is shown that the mapping gives us a smallest, known at present, implementation of 2D FSSP algorithms for rectangular arrays. The implementation itself has a nice property that the correctness of the constructed transition rule set is clear and transparent. It is shown that there exists a nine-state 2D cellular automaton that can synchronize any (m x n) rectangle in (m+n+max(m,n)-3) steps.**

***Keywords*-cellular automaton; FSSP; firing squad synchronization problem**

I. Introduction

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

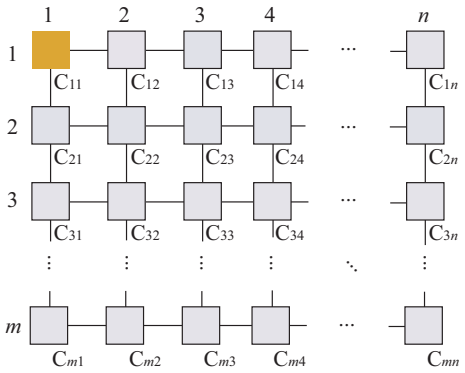

Fig. 1. A 2D rectangle cellular automaton.

In the present paper, we propose a simple and state-efficient mapping scheme: *zebra-like mapping* for implementing 2D synchronization algorithms. The zebra-like mapping we propose embeds two types of configurations alternately onto a 2D array like a zebra-like pattern, one configuration is a synchronization configuration of 1D arrays and the other is a stationary configuration which keeps its state unchanged until the final synchronization. The mapping gives us a smallest, known at present, implementation of 2D FSSP algorithms. Not only the number of states in the implementation is smaller, but the correctness of the constructed transition function with

Citation: H. Umeo, A. Nomura, A State-Efficient Zebra-Like Implementation of Synchronization Algorithms for 2D Rectangular Cellular Arrays, Biomath 1 (2012), 1209022, http://dx.doi.org/10.11145/j.biomath.2012.09.022

2561 rules is clear and transparent.

II. Firing Squad Synchronization Problem

A. FSSP on 2D Arrays

Figure 1 shows a finite two-dimensional (2D) rectangular array consisting of $m \times n$ cells. Each cell is an identical (except the border cells) finite-state automaton. The array operates in a 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_{11} 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. 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.

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 [1962], with each cell having several thousands of states. Mazoyer [1987] developed a six-state synchronization algorithm which, at present, is the algorithm having the fewest states for 1D arrays. On the other hand, a rich variety of synchronization algorithms for 2D rectangular arrays has been proposed.

The first optimum-time rectangle synchronization algorithm was proposed by Beyer [1969] and Shinahr [1974], independently. Concerning the time optimality of the 2D rectangle synchronization algorithms, the following theorems have been shown.

Theorem 1[Beyer [1969], Shinahr [1974]] There exists no cellular automaton that can synchronize any 2D rectangle array of size $m \times n$ in less than $m+m+\max(m,n)-3$ steps, where the general is located at one corner of the array.

Theorem 2[Shinahr [1974]] There exists a 28-state cellular automaton that can synchronize any 2D rectangle array of size $m \times n$ at exactly $m+m+\max(m,n)-3$ optimum steps, where the general is located at one corner of the array.

B. Optimum-Time L-Shaped Mapping Algorithm

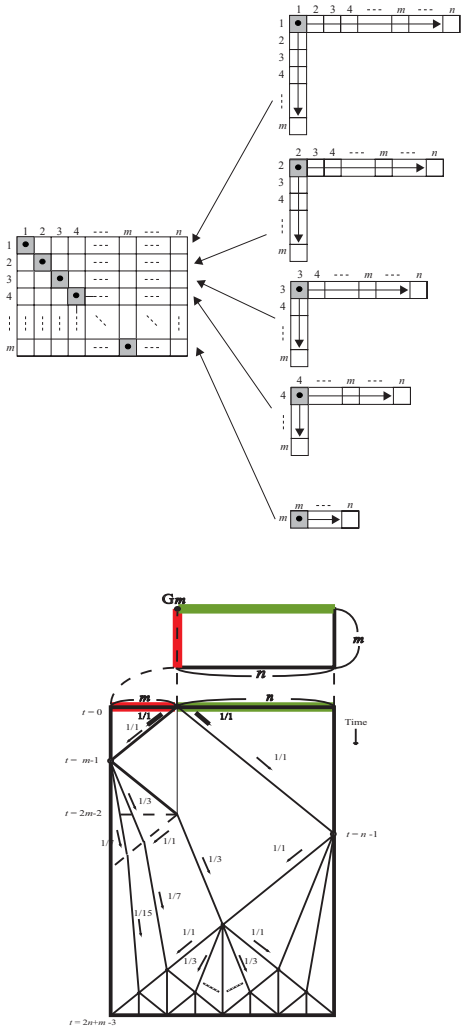

Fig. 2. L-shaped decomposition of an $m \times n$ rectangle cellular automaton and a space-time diagram for the L-shaped base synchronization algorithm. A black circle • in a shaded small square represents a general on each L_i and a wake-up signal for the synchronization generated by the general is indicated by a horizontal and vertical arrow.

The first optimum-time synchronization algorithm developed by Beyer [1969] and Shinahr [1974] for rectangle arrays operates as follows: We assume that an initial general is located on C_{11} on a rectangular array of size $m \times n$. By dividing the entire rectangle array of size $m \times n$ into $\min(m,n)$ *rotated L-shaped* 1D arrays, shown in Fig. 2, one treats the rectangle synchronization problem

as $\min(m,n)$ independent 1D generalized synchronizations with the general located at the bending cell of the L-shaped array. All cells on each L-shaped array fall into a pre-specified synchronization state, simultaneously. We denote the ith (from outside) L-shaped array by L_i and its horizontal and vertical segment is denoted by $\mathrm{L}_i^{\mathrm{h}}$ and $\mathrm{L}_i^{\mathrm{v}}$, $1 \leq i \leq \min(m,n)$, respectively. See Fig. 2. Concerning the synchronization of L_i, it can be easily seen that each general is generated at the cell C_{ii} at time $t = 3i-3$, and the general initiates the horizontal (row) and vertical (column) synchronizations on $\mathrm{L}_i^{\mathrm{h}}$ and $\mathrm{L}_i^{\mathrm{v}}$, via a 1D generalized optimum-time synchronization algorithm which can synchronize arrays of length ℓ with a general on kth cell from left end (from bottom in the L-shaped decomposition) in $\ell - 2 + \max(k, \ell - k + 1)$ optimum steps, where $1 \leq k \leq \ell$. Note that the length of L_i is $\ell_i = m + n - 2i + 1$ and the general is on $k_i = (m - i + 1)$th cell from left (bottom) end. For any $i, 1 \leq i \leq \min(m,n)$, all cells on L_i can be synchronized at time $t = 3i-3+\ell_i-2+\max(k_i, \ell_i-k_i+1) = m+n+i-4+\max(m-i+1, n-i+1) = m+n+\max(m,n)-3$. Thus, the rectangle array of size $m \times n$ can be synchronized at time $t = m + n + \max(m,n) - 3$ in optimum-steps. In Fig. 2 (top), 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.

The algorithm itself is very simple and now we are going to discuss its implementation in terms of a 2D cellular automaton.

The question is: how many states are required for its realization?

Let Q be a set of internal states for the 1D optimum-time generalized synchronization algorithm which is embedded onto a 2D array as a base algorithm. When we implement the algorithm on rectangle arrays based on the scheme above, we usually have to add a direction information to each state in order to simulate the embedded synchronization operations on each horizontal and vertical segment. Thus, approximately, $2 \mid Q \mid -1$ states are usually required for its independent row and column synchronization operations in order to avoid state mixing. Only a firing state is shared by the two areas. Shinahr [1974] gave a 28-state implementation based on the idea above.

C. Zebra-Like Mapping on Square Arrays

The proposed mapping for square arrays is basically based on the rotated L-shaped mapping scheme presented in the previous section, however, 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 fashion. The mapping is referred to as *zebra-like* mapping.

A key idea of the small-state implementation is:

- **Alternative mapping of two types of configurations:** A stationary layer separates two consecutive synchronization layers and it allows us to use the same state set for the vertical and horizontal synchronization on each layer, helping us to construct a small-state transition rule set for the synchronization layers.
- **A one-cell smaller synchronization configuration embedded:** A one-cell smaller synchronization configuration than the classical L-shaped mapping (Shinahr [1974]) is embedded, where we can save synchronization time by two steps.
- **A shared pre-firing state:** A single state `X` is shared between an initial general state of the square synchronizer, the stationary state in the stationary layer, and a pre-firing state of the embedded 1D synchronization algorithm used. The state `X` itself acts as a pre-firing state of the square synchronizer to be constructed.
- **A simple condition for final synchronization:** Any cell in state `X`, except $\mathrm{C}_{n,n}$, enters the final synchronization state at the next step if all its neighbors are in state X or the boundary state of the square. The cell $\mathrm{C}_{n,n}$ enters the synchronization state if and only if its north and west cells are in state `X` and its east and south cells are in the boundary state. A cell in state `X` that is adjacent to the cell $\mathrm{C}_{n,n}$ is also an exception. These are the only conditions that make cells fire.

In our construction we take the Mazoyer's 6-state 1D synchronization rule as an embedded synchronization algorithm. The set of the 6-states is {`G, Q, A, B, C, X`}, where `G` is a general, `Q` is a quiescent, and `X` is a firing state, respectively. The other three states `A`, `B` and `C` are auxiliary states, respectively.

The seven-state square synchronizer that we construct has the following state set: {`G, Q, A, B, C, X, F`}, where `F` is a newly introduced firing state, `X` is a

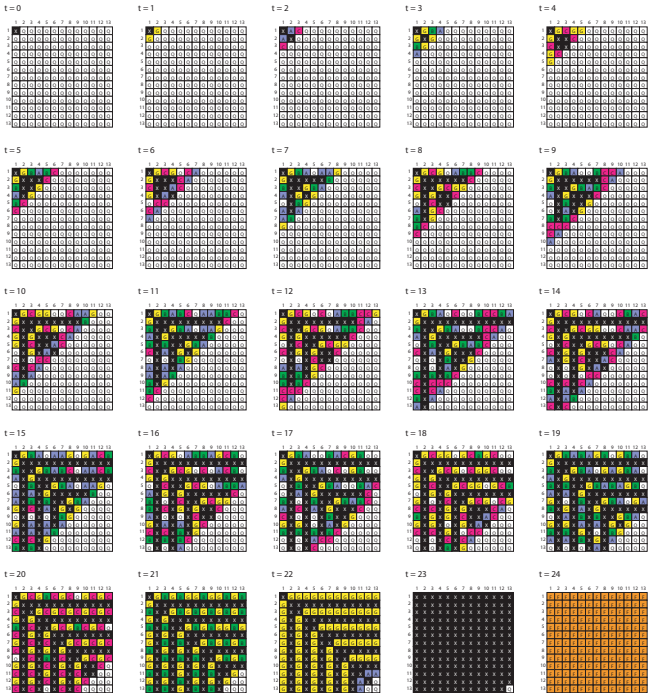

Fig. 3. Snapshots of the optimum-time synchronization process on a 13×13 square array.

general, and Q is a quiescent state, respectively. The state G is the general state of the embedded synchronization. Those states A, B and C are also auxiliary states, respectively. The transition rule set is constructed in such a way that: The initial general on $C_{1,1}$ in state X generates a new general in state G on the cell $C_{1,2}$ and $C_{2,1}$ at time $t = 1$. The general in state G initiates a synchronization for the following cells $\{C_{1,2}, C_{1,3}, ..., C_{1,n}\}$ and $\{C_{2,1}, C_{3,1}, ..., C_{n,1}\}$, each of length $n-1$. Note that the length of the array where optimum-time synchronization operations are embedded is shorter by one than the usual embedding in section 2. The cells on the segments are constructed to operate so that they simulate the Mazoyer's optimum-time synchronization operations. All cells on the two horizontal and vertical segments of length $n-1$ enter the pre-firing state X at time $t = 1 + 2(n-1) - 2 = 2n - 3$. In this way, the first L_1 acts as a synchronization layer. At time $t = 2$, the cell $C_{2,2}$ takes the state X and it extends an X-arm (a cell segment in state X) in the right and lower direction, respectively, towards the cells $\{C_{2,3}, C_{2,4}, ..., C_{2,n}\}$ and $\{C_{3,2}, C_{4,2}, ..., C_{n,2}\}$, respectively, each of length $n-2$. Every cell once entered in state X remains unchanged by the time before it meets a local condition for the synchronization given later. At time $t = 2 + n - 2 = n$, the filled-in operation with the stationary state X on the second layer is finished. In this way, the second L_2 acts as a stationary layer.

Concerning the embedding on the odd ith layer, the cell $C_{i,i}$ takes the stationary state X time $t = 2i - 2$ and generates a new general in state G on the cell $C_{i,i+1}$ and $C_{i+1,i}$ at time $t = 2i - 1$. The general in state G initiates a synchronization for the following cells $\{C_{i,i+1}, C_{i,i+2}, ..., C_{i,n}\}$ and $\{C_{i+1,i}, C_{i+2,i}, ..., C_{n,i}\}$, each of length $n-i$. All cells on the two horizontal and vertical segments of length $n-i$ enter the pre-firing state X at time $t = 2i - 1 + 2(n-i) - 2 = 2n - 3$. In this way, for odd i, the ith L_i acts as a synchronization layer. As for the even ith layer, at time $t = 2i - 2$, the cell $C_{i,i}$ takes the state X and it extends the X-arm in the right and lower direction, respectively, towards the cells $\{C_{i,i+1}, C_{i,i+2}, ..., C_{i,n}\}$ and $\{C_{i+1,i}, C_{i+2,i}, ..., C_{n,i}\}$, each of length $n-i$. Every cell once entered in state X remains unchanged by the time before synchronization. At time $t = 2i - 2 + n - i = n + i - 2$, the filled-in operation on the ith layer for even i is finished. At time $t = 2n - 3$, all of the cells, except $C_{n,n}$, on the square of size $n \times n$ enter the state X, which is a pre-firing state.

Thus we have seen:

Theorem 3 Umeo and Kubo [2010] There exists a seven-state 2D CA that can synchronize any $n \times n$ square array in $2n - 2$ steps.

Figure 3 shows some snapshots of the synchronization process operating in optimum-steps on a 13×13 square array.

III. Zebra-Like Mapping on Rectangle Arrays

In this section we give three implementations for rectangle synchronization algorithms. As is shown in Fig. 2, a 1D generalized FSSP algorithm is mapped on an L-shaped 1D array, where the cells on the horizontal and vertical segments have to cooperate with each other. Thus, in contrast to the square implementation, two independent, small-size synchronization configurations cannot be implemented on the horizontal and vertical segment on a single synchronization layer in the rectangle case. All the implementations given are variants of the zebra-like mapping. The first ten-state implementation is a straightforward implementation of the zebra-like mapping, which yields a non-optimum algorithm. The second 11-state implementation is a variant of the zebra-like mapping where the first synchronization layer L_1 and the thereafter layers $L_i, i \geq 3$ take a different set of synchronization rule set. The third one is a nine-state implementation which regards the marking symbol

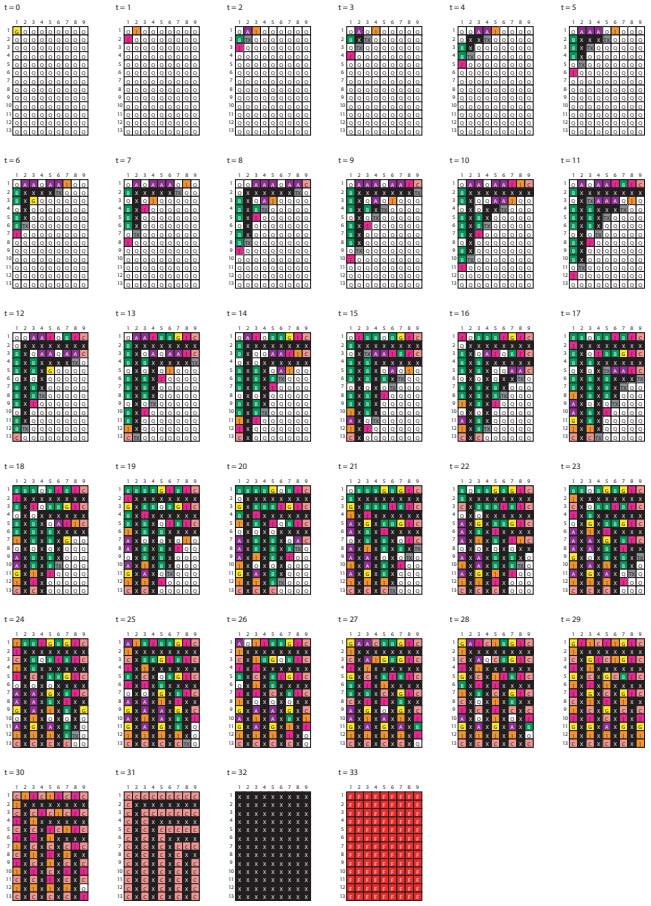

Fig. 4. Snapshots of the non-optimum-time ten-state synchronization process on a 13×9 rectangular array.

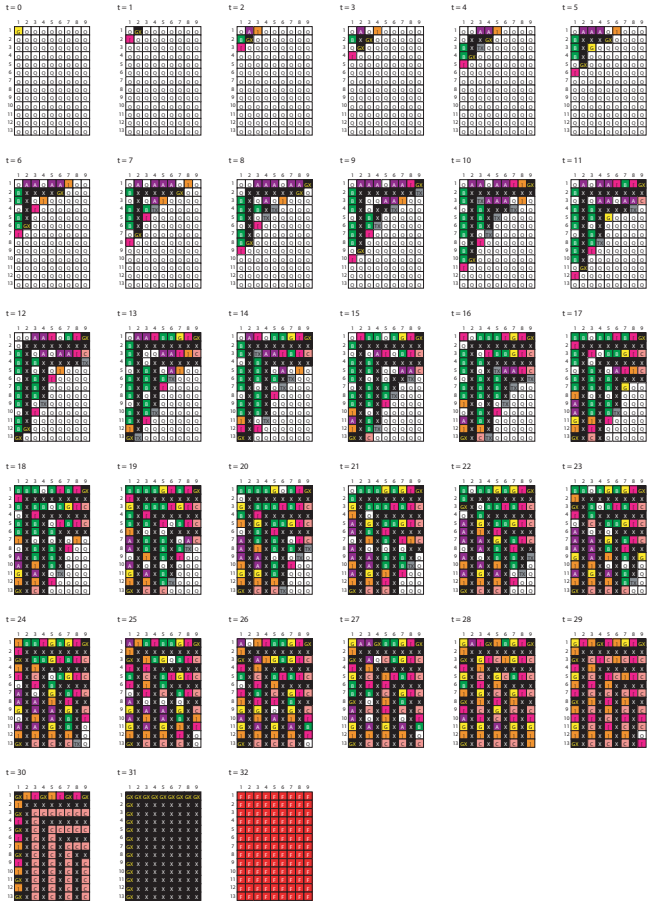

Fig. 5. Snapshots of the optimum-time eleven-state synchronization process on a 13×9 array.

used in the recursive division as the pre-firing state, making the algorithm work in optimum-steps. Those three implementations are stated in Theorems 4, 5 and 6. Their proofs are omitted due to the limited space available. Some snapshots of the synchronization processes in those three implementations are given in Figures 4, 5, and 6.

Theorem 4 There exists a ten-state 2D CA that can synchronize any $m \times n$ rectangle arrays in $m + n + \max(m, n) - 2$ non-optimum steps.

Theorem 5 There exists an eleven-state 2D CA that can synchronize any $m \times n$ rectangle arrays in $m + n + \max(m, n) - 3$ optimum steps.

Theorem 6 There exists a nine-state 2D CA that can synchronize any $m \times n$ rectangle arrays in $m + n + \max(m, n) - 3$ optimum steps.

TABLE I
A LIST OF IMPLEMENTATIONS FOR 2D RECTANGLE FSSP ALGORITHMS.

Implementations	# of states	# of rules	Time complexity	Notes
Beyer [1969]	—	—	optimum	rectangle
Shinahr [1974]	28	—	optimum	rectangle
Umeo, Maeda and Fujiwara [2002]	6	1718	non-optimum	rectangle
Umeo and Kubo [2010]	7	787	optimum	square
Theorem 4 (this paper)	10	1629	non-optimum	rectangle
Theorem 5 (this paper)	11	4044	optimum	rectangle
Theorem 6 (this paper)	9	2561	optimum	rectangle

IV. CONCLUSION

We have proposed a nine-state optimum-time synchronization algorithm that 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 new, simple zebra-like mapping scheme which embeds

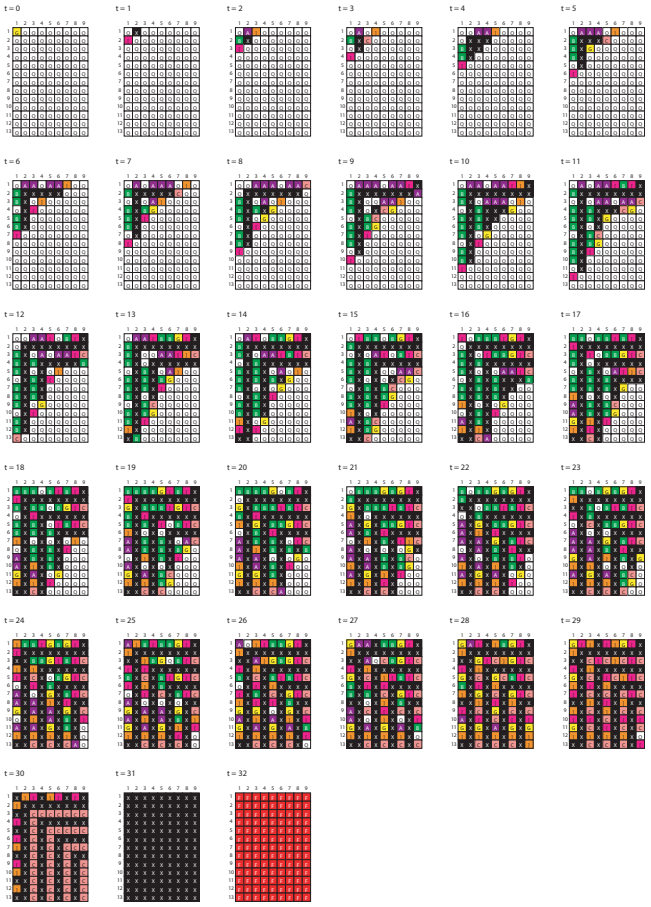

Fig. 6. Snapshots of the optimum-time nine-state synchronization process on a 13×9 array.

1D synchronization operations onto rectangle arrays. The nine-state implementation described in terms of state transition table is a smallest, known at present, realization of time-optimum rectangle synchronizer. The embedding scheme developed in this paper would be useful for state-efficient implementation of multi-dimensional synchronization algorithms.

REFERENCES

[1] R. Balzer, "An 8-state minimal time solution to the firing squad synchronization problem", *Information and Control* 10, 22–42 (1967). http://dx.doi.org/10.1016/S0019-9958(67)90032-0

[2] W. T. Beyer, "Recognition of topological invariants by iterative arrays", Ph.D. Thesis, MIT, pp. 144 (1969).

[3] H. D. Gerken, "Über Synchronisations — Probleme bei Zellularautomaten", *Diplomarbeit*, Institut für Theoretische Informatik, Technische Universität Braunschweig, pp. 50 (1987).

[4] E. Goto, "A minimal time solution of the firing squad problem", Dittoed course notes for Applied Mathematics 298, Harvard University, 52–59 (1962).

[5] J. Mazoyer, "A six-state minimal time solution to the firing squad synchronization problem", *Theoretical Computer Science* 50, 183–238 (1987). http://dx.doi.org/10.1016/0304-3975(87)90124-1

[6] E. F. Moore, "The firing squad synchronization problem", in *Sequential Machines, Selected Papers*, (E. F. Moore, ed.), Addison-Wesley, Reading MA, 213–214 (1964).

[7] H. Schmid, "Synchronisationsprobleme für zelluläre Automaten mit mehreren Generälen", Diplomarbeit, Universität Karsruhe, (2003).

[8] H. Schmid and T. Worsch, "The firing squad synchronization problem with many generals for one-dimensional CA", *Proc. of IFIP World Congress*, 111–124 (2004).

[9] I. Shinahr, "Two- and three-dimensional firing squad synchronization problems", *Information and Control* 24, 163–180 (1974). http://dx.doi.org/10.1016/S0019-9958(74)80055-0

[10] 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* 19, 305–320 (1982). http://dx.doi.org/10.1016/0304-3975(82)90040-8

[11] H. Umeo, "Firing squad synchronization algorithms for two-dimensional cellular automata", *Journal of Cellular Automata* 4, 1–20 (2008).

[12] H. Umeo, "Firing squad synchronization problem in cellular automata", In: *Encyclopedia of Complexity and System Science*, R. A. Meyers (Ed.), Springer, Vol.4, 3537–3574 (2009).

[13] 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*, 214–223 (2005).

[14] H. Umeo, N. Kamikawa, K. Nishioka, and S. Akiguchi, "Generalized firing squad synchronization protocols for one-dimensional cellular automata — a survey", *Acta Physica Polonica B, Proceedings Supplement* 3, 267–289 (2010).

[15] Umeo and Kubo, "A seven-state time-optimum square synchronizer.", *Proc. of the 9th International Conference on Cellular Automata for Research and Industry*, LNCS 6350, Springer-Verlag, 219–230 (2010).

[16] H. Umeo, M. Maeda and N. Fujiwara, "An efficient mapping scheme for embedding any one-dimensional firing squad synchronization algorithm onto two-dimensional arrays", *Proc. of the 5th International Conference on Cellular Automata for Research and Industry*, LNCS 2493, Springer-Verlag, 69–81 (2002).

[17] H. Umeo, M. Maeda, M. Hisaoka and M. Teraoka, "A state-efficient mapping scheme for designing two-dimensional firing squad synchronization algorithms", *Fundamenta Informaticae*, 74 No. 4, 603–623 (2006).

[18] H. Umeo and H. Uchino, "A new time-optimum synchronization algorithm for two-dimensional cellular arrays", *Proc. of Intern. Conf. on Computer Aided Systems Theory, EUROCAST 2007*, (A. Quesada-Arenciba, J. C. Rodriguez, R. Moreno-Diaz Jr., R. Moreno-Diaz (Eds.)), 213–216, (2007).

[19] H. Umeo, T. Yamawaki, N. Shimizu and H. Uchino, "Modeling and simulation of global synchronization processes for large-scale-of two-dimensional cellular arrays", *Proc. of Intern. Conf. on Modeling and Simulation, AMS 2007*, 139–144 (2007).