2004 International Symposium on Nonlinear
Theory and its Applications (NOLTA2004)
Fukuoka, Japan, Nov. 29 - Dec. 3, 2004

A Novel Design of One-Sided Recursive Firing Squad Synchronization Algorithms for One-Dimensional Cellular Arrays

Hayato YANASE†, Masaya HISAOKA† and Hiroshi UMEO†

†Univ. of Osaka Electro-Communication,
Faculty of Information Science and Technology,
Neyagawa-shi, Hatsu-cho, 18-8, Osaka, Japan

Abstract—The firing squad synchronization problem in cellular automata has been studied extensively for more that forty years, and a rich variety of synchronization algorithms have been proposed [1-14]. In the present paper, we introduce a new notion of one- and two-sided recursive properties to classify those synchronization algorithms proposed so far. We propose a new scheme for designing synchronization algorithms with the one-sided recursive property operating in optimum- and linear-time, respectively. We also give their implementations on a computer.

1. Introduction

In recent years cellular automata (CA) have been establishing increasing interests in the study of modeling non-linear phenomena occurring in biology, chemistry, ecology, economy, geology, mechanical engineering, medicine, physics, sociology, public traffic, etc. Cellular automata are considered to be a nice model of complex systems in which an infinite one-dimensional array of finite state machines (cells) updates itself in synchronous manner according to a uniform local rule. We study a synchronization problem which gives a finite-state protocol for synchronizing a large scale of cellular automata. The synchronization in cellular automata has been known as firing squad synchronization problem since its development, in which it was originally proposed by J. Myhill to synchronize all parts of self-reproducing cellular automata [8]. The firing squad synchronization problem has been studied extensively for more than forty years [1-14].

In this paper, we introduce a new notion of one- and two-sided recursive properties to classify those synchronization algorithms proposed so far. It is shown that optimum-time synchronization algorithms developed by Balzer [1], Gerken [3], and Waksman [13] are two-sided ones and an algorithm proposed by Mazoyer [6] is an only synchronization algorithm with the one-sided recursive property. We propose a new scheme for designing synchronization algorithms with the one-sided recursive property operating in optimum- and linear-time, respectively. We also give their implementations on a computer. Due to the space available, we omit the proofs of the theorems presented.

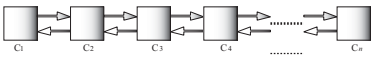

Figure 1: A one-dimensional cellular automaton.

2. Firing Squad Synchronization Problem for One-Dimensional Cellular Automata

The firing squad synchronization problem is formalized in terms of the model of cellular automata. Figure 1 shows a finite one-dimensional cellular array consisting of n cells, denoted by C_i, where $1 \leq i \leq n$. 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. All cells (*soldiers*), except the left end 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 left end cell (*general*) is in the *fire-when-ready* state, which is an initiation signal to the array. The firing squad synchronization problem is stated as follows. Given an array of n identical cellular automata, including a *general* on the left end which is activated at time $t = 0$, we want to give the description (state set and next-state function) of the automata so that, *at some future time*, all of the cells will *simultaneously* and, *for the first time*, enter a special *firing* state. The set of states must be independent of n. Without loss of generality, we assume $n \geq 2$. 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 length n of the array.

3. One-Sided vs. Two-Sided Recursive Synchronization Algorithms

Firing squad synchronization algorithms have been designed on the basis of parallel divide-and-conquer

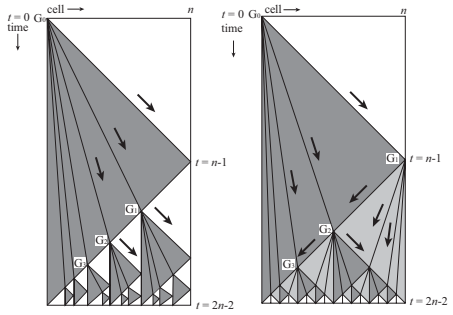

Figure 2: One-sided recursive synchronization scheme (left) and two-sided recursive synchronization scheme (right).

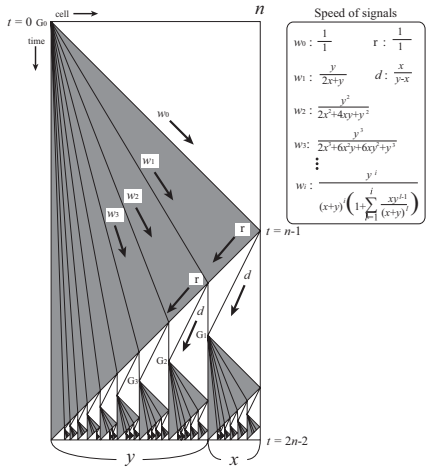

Figure 3: Time-space diagram for optimum-time one-sided recursive synchronization algorithm.

strategy that calls itself recursively in parallel. Those recursive calls are implemented by generating many *Generals* that are responsible for synchronizing divided small areas in the cellular space. Initially a *General* located at the left end is responsible for synchronizing the whole cellular space consisting of n cells. In Fig. 2 (left), the *General* G_i, $i = 2, 3, ...$, is responsible for synchronizing the cellular space between G_i and G_{i-1}, respectively. G_1 synchronizes the subspace between G_1 and the right end of the array. Thus, all of the *Generals* generated by G_0 are located at the left end of the divided cellular spaces to be synchronized by them independently. On the other hand, in Fig. 2 (right), the *General* G_0 generates *General* G_i, $i = 1, 2, 3, ...,$. Each G_i, $i = 1, 2, 3, ...$, synchronizes the divided space between G_i and G_{i+1}, respectively. In addition, G_i, $i = 2, 3, ...$, does the same operations as G_0. Thus, in Fig. 2 (right) we find *Generals* located at either end of the subspace for which they are responsible.

If all of the recursive calls are issued by *Generals* located at one (two) end(s) of partitioned cellular spaces for which the *General* is responsible, the algorithm is said to have *one-sided* (*two-sided*) recursive property. We call synchronization algorithm with one-sided (two-sided) recursive property as one-sided (two-sided) recursive synchronization algorithm. Figure 2 illustrates a time-space diagram for one-sided (Fig. 2 (left)) and two-sided (Fig. 2 (right)) recursive synchronization algorithms operating in optimum $2n - 2$ steps.

It is noted that optimum-time synchronization algorithms developed by Balzer [1], Gerken [3], and Waksman [13] are two-sided ones and an algorithm proposed by Mazoyer [6] is an only synchronization algorithm with the one-sided recursive property. In addition, it is also observed that all of the $3n$-step synchronization algorithms developed so far are in the class of two-sided algorithms.

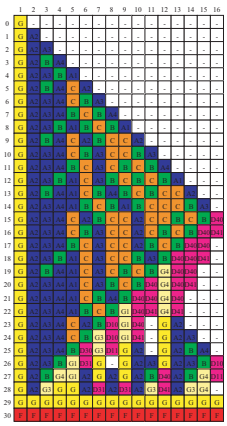

Figure 4: Configurations of a 22-state 498-rule implementation of one-sided recursive optimum-time synchronization algorithm on 16 cells.

[Observation 1] Optimum-time synchronization algorithms developed by Balzer [1], Gerken [3], and Waksman [13] are two-sided ones. The algorithm proposed by Mazoyer [6] is one-sided one.
[Observation 2] Synchronization algorithms operating in $3n \pm O(\log n) + O(1)$ steps developed by Minsky and MacCarthy [7], Fischer [4], and Yunès [14] are two-sided ones.

4. Design of Time-Optimum One-Sided Recursive Synchronization Algorithms

In this section we propose a design scheme for one-sided recursive optimum-time synchronization algorithms that can synchronize any n cells in $2n-2$ steps. Figure 3 is a time-space diagram for an optimum-time one-sided recursive synchronization algorithm. The *General* G_0 generates an infinite number of signals

w_0, w_1, w_2, .., to generate *Generals* G_1, G_2, .., by dividing the array recursively with the ratio x/y, where x, y is any positive integer such that $2x \leq y$. Propagation speed of the i-th signal w_i, $i \geq 1$ is as follows:

$$y^i/(x+y)^i(1+\sum_{l=1}^{i} xy^{l-1}/(x+y)^l)).$$

When the first signal w_0 hits the right end of the array, an r-signal is generated that propagates at speed $1/1$ in the left direction. At the same time, a d-signal propagating at speed $x/(y-x)$ in the left direction is generated. The w_1- and r-signals meets on cell C_m, $m = \lceil ny/(x+y) \rceil$, and a special mark is printed as a potential *General*. When the d-signal arrives at the cell C_m, a new *General* G_1 is generated. Its generation is delayed for $\lceil n(y-2x)/(x+y) \rceil$ steps. The G_1 does the same procedures as G_0 to the subspace between C_m and C_n.

When $x = 1, y = 2$, the scheme given above coincides with the Mazoyer's algorithm [6]. It is noted that, in this case, we need no delay for the generation of *Generals*. We have implemented the scheme in the case where $x = 1, y = 3$ on a computer and got a 22-state 498-rule cellular automaton that realizes the one-sided recursive synchronization. Figure 4 is configurations of the 22-state implementation of one-sided recursive optimum-time synchronization algorithm on 16 cells.

[Theorem 3] There exists a one-sided 22-state 498-rule cellular automaton that can synchronize any n cells in $2n-2$ optimum steps.

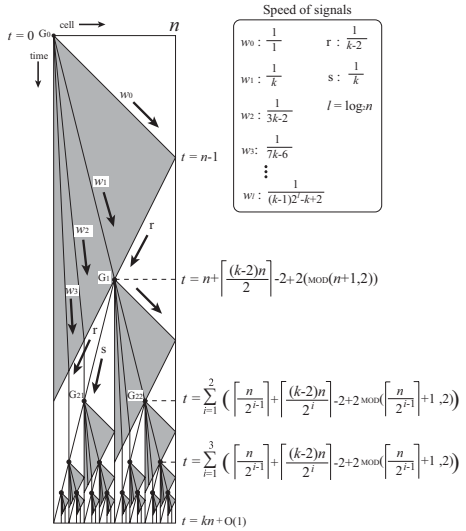

Figure 5: Time-space diagram for kn+O(1)-step one-sided recursive synchronization algorithm.

5. Design of Linear-Time One-Sided Recursive Synchronization Algorithms

In this section we study a class of synchronization algorithms operating in linear-time, that can synchronize any n cells in kn+O(1) steps, where k is any positive integer such that $k \geq 3$. The first synchronization algorithm developed by Minsky and MacCarthy [7] operates in $3n$+O(1) steps. Yunès [14] gave its 14- and 15-state implementations. Fischer [4] also presented a solution with 15 states that synchronizes n cells in $3n-4$ steps. In 1994 Yunès [14] proposed a seven-state solution operating in $3n \pm 2\theta_n \log n$+O(1) steps, where $0 \leq \theta_n \leq 1$. In addition, the $3n$-step synchronization algorithm is often used as a subroutine in the design of other optimum-time synchronization algorithms. Thus, a class of linear-time synchronization algorithms is interesting and important in its own right.

We propose an efficient way to cause a *General* cell to generate infinite signals $w_0, w_1, w_2, ..., w_\ell$ propagating at speeds of $1/1, 1/k, 1/(3k-2)$, .., $1/((k-1)2^\ell - k + 2)$, where $\ell = \lceil \log_2 n \rceil$. These signals play an important role in dividing the array into two, four, eight, ..., equal parts synchronously. The end cell in each partition takes a special prefiring state so that when the last partition occurs, where all cells have a left and right neighbor in this state.

Figure 5 is a time-space diagram for kn+O(1)-step one-sided recursive synchronization algorithm. At time $t = 0$, the initial *General* G_0 generates an infinite right-going signals mentioned above. The first w_0-signal arrives at the right end of the array at time $t = n-1$. Then, the right end cell sends out an reflected r-signal that proceeds towards the left end at speed $1/(k-2)$. The r-signal meets with the second w_1-signal at the center point of the array at time $t = n + \lceil (k-2)n/2 \rceil - 2 + 2(n+1 \bmod 2)$, and the second *General* G_1 is generated there. The *General* G_1 does the same operations as G_0 to the right half of the array. Simultaneously, G_1 generates a left-going s-signal that continues to propagates towards the left end of the array at speed $1/k$. The w_2- and r-signals meet at the quarter point of the array and a special mark, being kept until the s-signal arrives at the cell, is given as a potential General. The s-signal acts as a delay for the generation of *General* G_{21} so that both G_{21} and G_{22} can be generated simultaneously to synchronize three quarters of the array. The G_{21} is delayed for $\Delta t = n/2$ steps. The readers can see how these signals work and all of the *Generals* generated are located at the left end of the divided subspaces. Let $T(n)$ be time complexity for synchronizing n cells. Then, $T(n) = kn/2 + T(n/2) = kn$+O(1).

Thus, we have:

[Theorem 4] Let k be any positive integer such

that $k \geq 3$. The one-sided recursive synchronization scheme given above can synchronize any n cells in $kn + O(1)$ steps.

Figure 6 is a time-space diagram in the case of $k = 3$. We have implemented the scheme on a 12-state 236-rule cellular automaton. Configurations of a 12-state implementation of one-sided recursive $(3n - 3)$-step synchronization algorithm on 21 cells are shown in Fig. 7.

[Theorem 5] There exists a one-sided recursive 12-state 236-rule cellular automaton that can synchronize any n cells in $3n - 3$ steps.

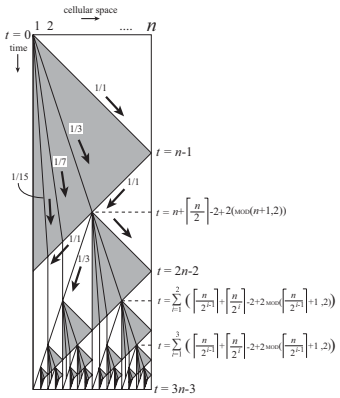

Figure 6: Time-space diagram for $(3n - 3)$-step one-sided recursive synchronization algorithm.

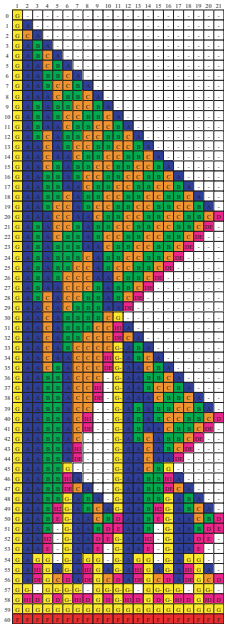

Figure 7: Configurations of a 12-state 236-rule implementation of one-sided recursive $(3n - 3)$-step synchronization algorithm on 21 cells.

6. Conclusions

We have introduced a new notion of one- and two-sided recursive properties in the synchronization algorithms for one-dimensional cellular automata and proposed a new scheme for designing synchronization algorithms with the one-sided recursive property operating in optimum- and linear-time, respectively. We have also given their 22- and 12-state implementations that realize those algorithms on a computer. We omitted the complete lists of transitions rules designed due to the space available. We have checked their validity for cellular spaces of size $n = 2$ to 10000 through our computer simulation. All of the synchronization algorithms proposed so far for two-dimensional arrays are designed based on two-sided recursive synchronization algorithms for one-dimensional arrays. It is an interesting question whether we can design a *real* two-dimensional synchronization algorithm with one-sided recursive properties.

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, pp. 144 (1969).

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

[4] P. C. Fischer: Generation of primes by a one-dimensional real-time iterative array. *J. of ACM*, vol. 12, No. 3, pp. 388-394, (1965).

[5] A. Grasselli: Synchronization of cellular arrays: The firing squad problem in two dimensions. *Information and Control*, 28, pp. 113-124 (1975).

[6] J. Mazoyer: A six-state minimal time solution to the firing squad synchronization problem. *Theoretical Computer Science*, vol. 50 (1987), pp. 183-238.

[7] M. Minsky: *Computation: Finite and infinite machines*. Prentice Hall, (1967), pp. 28-29.

[8] 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.

[9] I. Shinahr: Two- and three-dimensional firing squad synchronization problems. *Information and Control*, 24, pp. 163-180 (1974).

[10] 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, pp.69-81, (2002).

[11] H. Umeo, M. Hisaoka, and T. Sogabe: An investigation into transition rule sets for optimum-time firing squad synchronization algorithms on one-dimensional cellular automata. *Interdisciplinary Information Sciences*, Vol.8, No.2, pp.207-217, (2002).

[12] H. Umeo: A simple design of time-optimum firing squad synchronization algorithms with fault-tolerance. *IEICE Trans. on Information and Systems*, Vol.E87-D, No.3, (2004), pp.733-739.

[13] A. Waksman: An optimum solution to the firing squad synchronization problem. *Information and Control*, vol. 9 (1966), pp. 66-78.

[14] J. B. Yunes: Seven-state solution to the firing squad synchronization problem. *Theoretical Computer Science*, 127, pp.313-332, (1994).