Chapter 6
Problem Solving on One-Bit-Communication Cellular Automata

Hiroshi Umeo

6.1 Introduction

In recent years, interest in cellular automata (CA) has been increasing in the field of modeling real phenomena that occur in biology, chemistry, ecology, economy, geology, mechanical engineering, medicine, physics, sociology, and public transportation. Cellular automata are considered to provide a good model of complex systems in which an infinite array of finite state machines (cells) updates itself in a synchronous manner according to a uniform local rule. In the present paper, we study a problem solving on a special subclass of cellular automata: one-bit inter-cell communication cellular automaton. The problems dealt with are a firing squad synchronization problem, an integer sequence generation problem, an early bird problem, and a connectivity recognition problem for two-dimensional binary images, all of which are classical, fundamental problems that have been studied extensively on O(1)-bit communication models of cellular automata. The O(1)-bit communication model is a conventional CA in which the number of communication bits exchanged in one step between neighboring cells is assumed to be O(1) bits. However, such bit information exchanged between inter-cells is hidden behind the definition of conventional automata-theoretic finite state descriptions. On the other hand, the 1-bit inter-cell communication model studied in the present paper is a new subclass of CAs, in which inter-cell communication is restricted to 1-bit communication. We refer to this model as the 1-bit CA and denote the model as $CA_{1\text{-bit}}$. The number of internal states of the $CA_{1\text{-bit}}$ is assumed to be finite as in a usual sense. The next state of each cell is determined based on the present state of the cell and two binary 1-bit inputs from its left and right neighbor cells. Thus, the $CA_{1\text{-bit}}$ is one of the weakest and simplest models among the variants of the CAs. A main question in this paper is whether the $CA_{1\text{-bit}}$ can solve problems solved by conventional cellular automata without any overhead in time complexities.

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

A.G. Hoekstra et al. (eds.), *Simulating Complex Systems by Cellular Automata*,
Understanding Complex Systems, DOI 10.1007/978-3-642-12203-3_6,

In Sect. 6.2, we define the 1-bit communication cellular automaton and review a computational relation between the conventional O(1)-bit-communication CA and the $CA_{1\text{-bit}}$. In Sect. 6.3, a firing squad synchronization problem is studied and several state-efficient 1-bit implementations of synchronization algorithms for one-dimensional cellular arrays are presented. In Sect. 6.4 we consider an integer sequence generation problem on the $CA_{1\text{-bit}}$ and present a real-time prime generator with 34 states. In Sect. 6.5, we study an early bird problem and its 37-state implementation operating in twice real-time will be given. In Sects. 6.6 and 6.7, a two-dimensional (2-D) version of the $CA_{1\text{-bit}}$ is introduced and the firing squad synchronization problem is studied again on the 2-D $CA_{1\text{-bit}}$. In Sect. 6.7, a connectivity recognition algorithm for two-dimensional binary images will be presented.

6.2 One-Bit-Communication Cellular Automata

A one-dimensional 1-bit inter-cell communication cellular automaton ($CA_{1\text{-bit}}$) consists of a finite array of identical finite state automata, each located at a positive integer point. Each automaton is referred to as a cell. The cell at point i is denoted by C_i where $i \geq 1$. Each C_i, except for C_1 and C_n, is connected with its left and right neighbor cells via a left or right one-way communication link, where those communication links are indicated by right- and left-going arrows, respectively, as shown in Fig. 6.1. Each one-way communication link can transmit only one bit at each step in each direction.

A cellular automaton with 1-bit inter-cell communication (abbreviated as $CA_{1\text{-bit}}$) consists of a finite array of finite state automaton $A = (Q, \delta)$, where

1. Q is a finite set of internal states.
2. δ is a function that defines the next state of any cell and its binary outputs to its left and right neighbor cells such that δ: $Q \times \{0, 1\} \times \{0, 1\} \rightarrow Q \times \{0, 1\} \times \{0, 1\}$ where $\delta(p, x, y) = (q, x', y')$, $p, q \in Q$, $x, x', y, y' \in \{0, 1\}$, has the following meaning: We assume that, at step t, the cell C_i is in state p and receives binary inputs x and y from its left and right communication links, respectively. Then, at the next step t+1, C_i takes a state q and outputs x' and y' to its left and right communication links, respectively. Note that binary inputs to C_i at step t are also outputs of C_{i-1} and C_{i+1} at step t. A quiescent state $q \in Q$ has a property such that $\delta(q, 0, 0) = (q, 0, 0)$.

Thus, the $CA_{1\text{-bit}}$ is a special subclass of *normal* (i.e., *conventional*) cellular automata. Let N be any normal cellular automaton with a set of states Q and a transition function $\delta : Q^3 \rightarrow Q$. The state of each cell on N depends on the

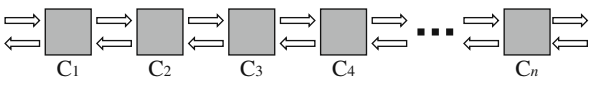

Fig. 6.1 One-dimensional cellular automaton connected with 1-bit inter-cell communication links

cell's previous state and states on its nearest neighbor cells. This means that the total information exchanged per step between neighboring cells is O(1) bits. Each state in Q can be encoded with a binary sequence of length $\lceil \log_2 |Q| \rceil$ and then sending the binary sequences sequentially bit-by-bit in each direction via each one-way communication link. The sequences are then received bit-by-bit and decoded into their corresponding states in Q. Thus, the $\mathrm{CA}_{1\text{-bit}}$ can simulate one step of N in $\lceil \log_2 |Q| \rceil$ steps. This observation gives the following computational relation between the normal CA and $\mathrm{CA}_{1\text{-bit}}$.

Theorem 1 (Mazoyer [21], Umeo and Kamikawa [37]) *Let N be any normal cellular automaton operating in $T(n)$ steps with internal state set Q. Then, there exists a $CA_{1\text{-bit}}$ that can simulate N in $kT(n)$ steps, where k is a positive constant integer such that $k = \lceil \log_2 |Q| \rceil$.*

A question is whether the $\mathrm{CA}_{1\text{-bit}}$ can solve problems solved by conventional cellular automata without any overhead in time complexities. In some cases, the answer is *yes*.

6.3 Firing Squad Synchronization Problem

Section 6.3 studies the firing squad synchronization problem (FSSP) on $\mathrm{CA}_{1\text{-bit}}$, the solution of which yields a finite-state protocol for large-scale synchronization of cellular automata. This problem was originally proposed by J. Myhill in Moore [23] to synchronize all parts of self-reproducing cellular automata. The firing squad synchronization problem has been studied extensively for more than 50 years. Recent developments in the FSSP algorithms are given in Umeo [33] and Umeo et al. [35]. An optimum-time (i.e., $(2n - 2)$-step for n cells) synchronization algorithm for one-dimensional array was devised first by Goto [10]. The algorithm needed many thousands of internal states for its realization. Afterwards, Waksman [45], Balzer [4], Gerken [9] and Mazoyer [19] developed an optimum-time algorithm and reduced the number of states realizing the algorithm, each with 16, 8, 7 and 6 states on the conventional O(1)-bit communication model.

The FSSP is defined as follows: At time $t = 0$, the left end cell C_1 is in the *fire-when-ready* state, which is the initiation signal for the array. The FSSP 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 set of states and the next-state function must be independent of n.

6.3.1 FSSP with a General at One End

Here we briefly sketch the design scheme for the firing squad synchronization algorithm according to Waksman [45] in which the first transition rule set was presented. It is quoted from Waksman [45].

The code book of the state transitions of machines is so arranged to cause the array to progressively divide itself into 2^k equal parts, where k is an integer and an increasing function of time. The end machines in each partition assume a special state so that when the last partition occurs, all the machines have for both neighbors machines at this state. This is made the *only* condition for any machine to assume terminal state.

Figure 6.2 (left) is a space-time diagram for the Waksman's optimum-step firing squad synchronization algorithm. The general at time $t = 0$ emits 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. 6.2 (left). 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.

Most of the implementations for the optimum-time synchronization algorithms developed so far on $CA_{1\text{-bit}}$ are based on the space-time diagram shown in Fig. 6.2 (left). Mazoyer [21] developed an optimum-time synchronization algorithm for the $CA_{1\text{-bit}}$ based on Balzer [4]. Each cell of the constructed $CA_{1\text{-bit}}$ had 58 internal states. The original set of transition rules constructed in Mazoyer [21] included a small error. Here we show a reconstructed version in Table 6.1. Figure 6.3 shows some snapshots of the synchronization processes on 21 cells, each for Balzer's

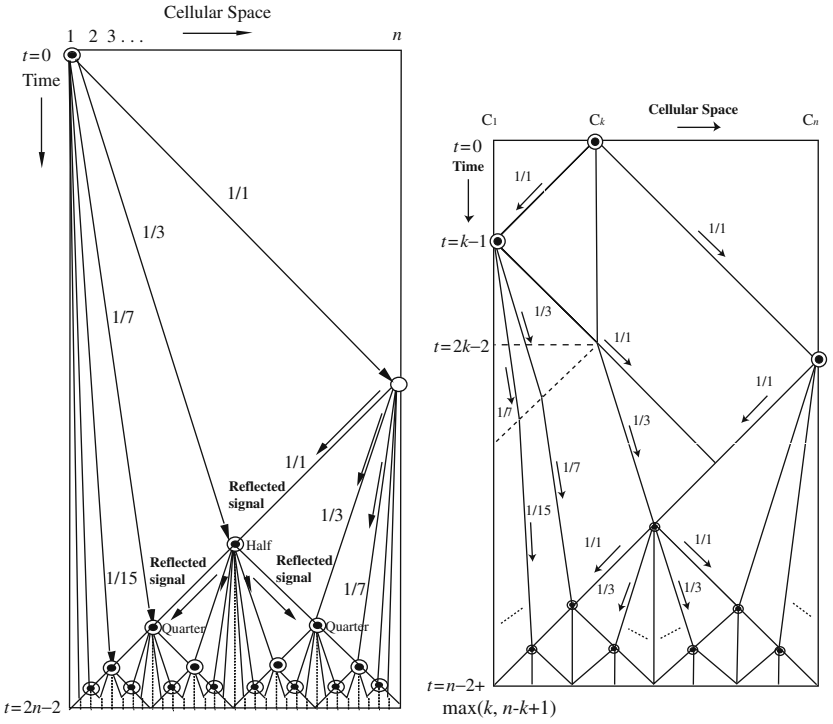

Fig. 6.2 Space-time diagram for optimum-time synchronization algorithms with a general at the left end (*left*) and a generalized case where a general at an arbitrary point (*right*)

Table 6.1 Reconstructed transition table for Mazoyer's 1-bit implementation (Mazoyer [21])

1

0	R = 0	R = 1
L = 0	(0,0,0)	--
L = 1	(ir,0,1)	--

2

ir	R = 0	R = 1
L = 0	(kr,1,0)	(kr,1,0)
L = 1	(2ar,0,0)	(2ar,0,0)

3

F	R = 0	R = 1
L = 0	--	--
L = 1	--	--

4

kr	R = 0	R = 1
L = 0	(Exl,0,1)	(il,1,0)
L = 1	(Oddr,0,0)	(Ex!r,1,0)

5

pF	R = 0	R = 1
L = 0	(pF,0,0)	(F,0,0)
L = 1	(F,0,0)	(F,0,0)

6

2ar	R = 0	R = 1
L = 0	(2b,0,1)	(pF,1,1)
L = 1	--	--

7

2b	R = 0	R = 1
L = 0	--	(2c,0,1)
L = 1	(2c,1,0)	--

8

2c	R = 0	R = 1
L = 0	(2c,0,0)	(pF,1,1)
L = 1	(pF,1,1)	--

9

Oddr	R = 0	R = 1
L = 0	--	(Oor,1,0)
L = 1	--	(3m,0,1)

10

Oor	R = 0	R = 1
L = 0	(Oo*r,0,0)	(il,1,0)
L = 1	--	--

11

Oo*r	R = 0	R = 1
L = 0	(Oor,0,0)	(Oe,1,0)
L = 1	(Op,0,0)	(Orl,1,0)

12

Oe	R = 0	R = 1
L = 0	(Oe*,0,0)	(Exr,1,0)
L = 1	(Exl,0,1)	--

13

Oe*	R = 0	R = 1
L = 0	(Oe,0,0)	(Oor,1,0)
L = 1	(Ool,0,1)	--

14

Op	R = 0	R = 1
L = 0	(Op*,0,0)	(il,1,0)
L = 1	(ir,0,1)	--

15

Op*	R = 0	R = 1
L = 0	(Op,0,0)	(Orl,1,0)
L = 1	(Orr,0,1)	--

16

Orr	R = 0	R = 1
L = 0	(Or*r,0,0)	(Rr,1,0)
L = 1	(pLl,0,1)	--

17

Or*r	R = 0	R = 1
L = 0	(Orr,0,0)	(Oe,1,0)
L = 1	(Om,1,1)	--

18

Om	R = 0	R = 1
L = 0	(Om*,0,0)	(M,1,1)
L = 1	(M,1,1)	--

19

Om*	R = 0	R = 1
L = 0	(Om,0,0)	(Orr,0,0)
L = 1	(Orl,0,0)	--

20

Exr	R = 0	R = 1
L = 0	(il,0,0)	(Ex!l,0,0)
L = 1	(Ee,0,1)	(Err,0,1)

21

Ex!r	R = 0	R = 1
L = 0	(2ar,0,0)	(Ex!l,0,0)
L = 1	--	(pF,1,1)

22

3m	R = 0	R = 1
L = 0	(3m*,0,0)	(M,1,1)
L = 1	(M,1,1)	--

23

3m*	R = 0	R = 1
L = 0	(3m,0,0)	(3r,0,0)
L = 1	(3r,0,0)	--

24

3l	R = 0	R = 1
L = 0	(3l*,0,0)	(pLr,1,0)
L = 1	(pLl,0,1)	--

25

3l*	R = 0	R = 1
L = 0	(3l,0,0)	(3m,0,1)
L = 1	(3m,1,0)	--

26

Err	R = 0	R = 1
L = 0	(Er*r,0,0)	(Rr,1,0)
L = 1	(pLl,0,1)	--

27

Er*r	R = 0	R = 1
L = 0	(Err,0,0)	(Eor,1,0)
L = 1	(Em,1,1)	--

28

Ep	R = 0	R = 1
L = 0	(Ep*,0,0)	(il,1,0)
L = 1	(ir,0,1)	--

29

Ep*	R = 0	R = 1
L = 0	(Ep,0,0)	(Erl,1,0)
L = 1	(Err,0,1)	--

30

Eor	R = 0	R = 1
L = 0	(Eo*r,0,0)	(il,1,0)
L = 1	--	--

31

Eo*r	R = 0	R = 1
L = 0	(Eor,0,0)	(Ee,1,0)
L = 1	(Ep,0,0)	(Erl,1,0)

32

Em	R = 0	R = 1
L = 0	(Em*,0,0)	(M,1,1)
L = 1	(M,1,1)	--

33

Em*	R = 0	R = 1
L = 0	(Em,0,0)	(Err,0,0)
L = 1	(Erl,0,0)	--

34

Rr	R = 0	R = 1
L = 0	--	(pF,1,0)
L = 1	--	--

35

M	R = 0	R = 1
L = 0	(pF,1,1)	--
L = 1	--	--

36

pLr	R = 0	R = 1
L = 0	(Lr,1,1)	(Lr,1,1)
L = 1	--	--

37

Lr	R = 0	R = 1
L = 0	(pF,0,1)	--
L = 1	(F,0,0)	--

38

irW	R = 0	R = 1
L = 0	(pF,0,1)	--
L = 1	--	--

39

0W	R = 0	R = 1
L = 0	(0W,0,0)	--
L = 1	(ilW,1,0)	--

40

3r	R = 0	R = 1
L = 0	(3r*,0,0)	(Rr,1,0)
L = 1	(Rl,0,1)	--

41

3r*	R = 0	R = 1
L = 0	(3r,0,0)	(3l,0,0)
L = 1	(3l,0,0)	--

42

Ee	R = 0	R = 1
L = 0	(Ee*,0,0)	(Exr,1,0)
L = 1	(Exl,0,1)	--

43

Ee*	R = 0	R = 1
L = 0	(Ee,0,0)	(Eor,1,0)
L = 1	(Eol,0,1)	--

44

ilW	R = 0	R = 1
L = 0	(pF,1,0)	--
L = 1	(F,0,0)	--

45

2al	R = 0	R = 1
L = 0	(2b,1,0)	--
L = 1	(pF,1,1)	--

46

Oddl	R = 0	R = 1
L = 0	--	--
L = 1	(Ool,0,1)	(3m,1,0)

47

Ool	R = 0	R = 1
L = 0	(Oo*l,0,0)	--
L = 1	(ir,0,1)	--

48

Oo*l	R = 0	R = 1
L = 0	(Ool,0,0)	(Op,0,0)
L = 1	(Oe,0,1)	(Orr,0,1)

49

Orl	R = 0	R = 1
L = 0	(Or*l,0,0)	(pLr,1,0)
L = 1	(Rl,0,1)	--

50

Or*l	R = 0	R = 1
L = 0	(Orl,0,0)	(Om,1,1)
L = 1	(Oe,0,1)	--

51

Exl	R = 0	R = 1
L = 0	(ir,0,0)	(Ee,1,0)
L = 1	(Ex!r,0,0)	(Erl,1,0)

52

Ex!l	R = 0	R = 1
L = 0	(2al,0,0)	--
L = 1	(Ex!r,0,0)	(pF,1,1)

53

Erl	R = 0	R = 1
L = 0	(Er*l,0,0)	(pLr,1,0)
L = 1	(Rl,0,1)	--

54

Er*l	R = 0	R = 1
L = 0	(Erl,0,0)	(Em,1,1)
L = 1	(Eol,0,1)	--

55

Eol	R = 0	R = 1
L = 0	(Eo*l,0,0)	--
L = 1	(ir,0,1)	--

56

Eo*l	R = 0	R = 1
L = 0	(Eol,0,0)	(Ep,0,0)
L = 1	(Ee,0,1)	(Err,0,1)

57

Rl	R = 0	R = 1
L = 0	--	--
L = 1	(pF,0,1)	--

58

pLl	R = 0	R = 1
L = 0	(Ll,1,1)	--
L = 1	(Ll,1,1)	--

59

Ll	R = 0	R = 1
L = 0	(pF,1,0)	(F,0,0)
L = 1	--	--

60

il	R = 0	R = 1
L = 0	(kl,0,1)	(2al,0,0)
L = 1	(kl,0,1)	(2al,0,0)

61

kl	R = 0	R = 1
L = 0	(Exr,1,0)	(Oddl,0,0)
L = 1	(ir,0,1)	(Ex!l,0,1)

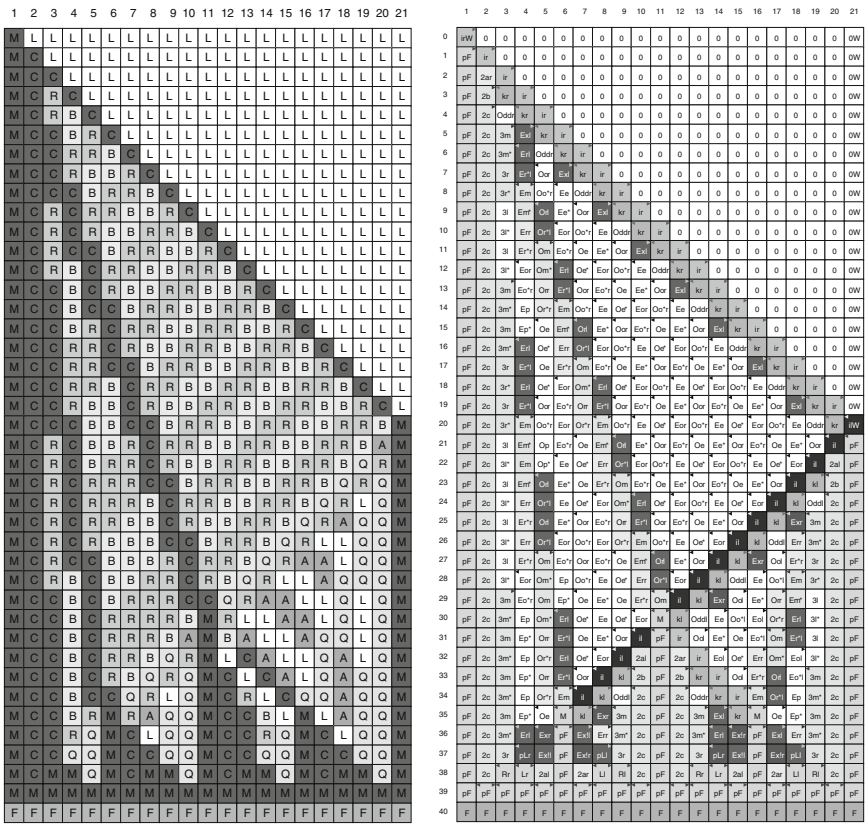

Fig. 6.3 Snapshots for synchronization processes on 21 cells, each for Balzer's algorithm [4] on the O(1)-bit-communication model (*left*) and the reconstructed 1-bit implementation on the $CA_{1\text{-bit}}$ (*right*)

algorithm [4] on the O(1)-bit-communication model (left) and the reconstructed 1-bit implementation on the $CA_{1\text{-bit}}$ (right). The small right- and left-facing black triangles, ► and ◄, in the figure, indicate a 1-bit signal transfer in the right or left direction between neighbor cells. The symbol in each cell shows its internal state. Nishimura et al. [25] also constructed an optimum-time synchronization algorithm (NSU algorithm for short) based on Waksman's algorithm [45]. Each cell had 78 internal states and 208 transition rules. Figure 6.4 shows snapshots for synchronization processes on 21 cells, each for Waksman's algorithm [45] on O(1)-bit-communication model (left) and NSU algorithm [25] on $CA_{1\text{-bit}}$ (right).

Theorem 2 (Mazoyer [21], Nishimura, Sogabe and Umeo [25]) *There exists a* $CA_{1\text{-bit}}$ *that can synchronize n cells with the general at a left end in* $2n - 2$ *steps.*

Umeo et al. [42] developed a non-optimum-step synchronization algorithm for $CA_{1\text{-bit}}$ based on Mazoyer's 6-state algorithm [19] for the O(1)-bit model. The

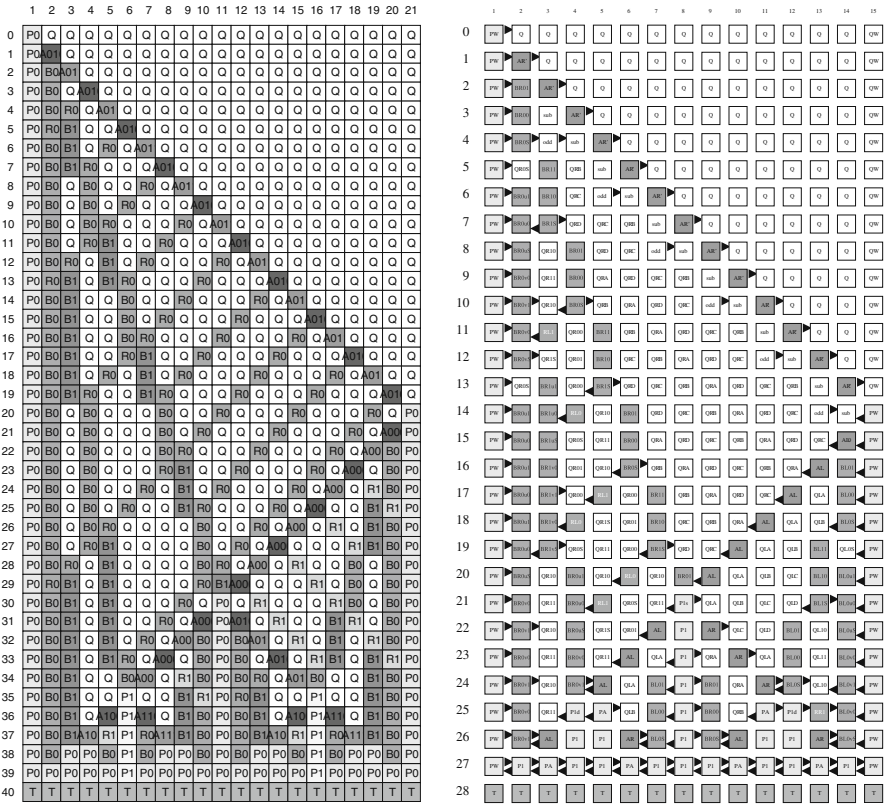

Fig. 6.4 Snapshots for synchronization processes on 21 cells, each for Waksman's algorithm [45] on O(1)-bit-communication model (*left*) and NSU implementation [25] on $CA_{1\text{-bit}}$ (*right*)

constructed $CA_{1\text{-bit}}$ synchronizes n cell in $2n - 1$ steps and each cell has 54 states and 207 transition rules.

Theorem 3 (Umeo, Yanagihara and Kanazawa [42]) *There exists a 54-state* $CA_{1\text{-bit}}$ *that can synchronize any n cells in* $2n - 1$ *non-optimum-step.*

Umeo and Yanagihara [41] also constructed a smaller optimum-time implementation based on Gerken's synchronization algorithm [9] on the O(1)-bit-communication model. The constructed $CA_{1\text{-bit}}$ has 35 internal states and 114 transition rules. Table 6.2 presents its transition rule set for the 35-state synchronization protocol and Figure 6.5 shows snapshots for synchronization processes on 17 cells, each for Gerken's algorithm [9] on O(1)-bit-communication model (left) and our 35-state algorithm [41] on the $CA_{1\text{-bit}}$ (right).

Theorem 4 (Umeo and Yanagihara [41]) *There exists a 35-state* $CA_{1\text{-bit}}$ *that can synchronize n cells with the general on the left end in* $2n - 2$ *steps.*

Table 6.2 Transition table for a 35-state implementation of the optimum-time synchronization algorithm (Umeo and Yanagihara [41])

1 Q	R = 0	R = 1
L = 0	(Q,0,0)	(Q,0,0)
L = 1	(RA,0,1)	(LGW,1,1

2 RGW	R = 0	R = 1
L = 0	(RGW,0,1)	(F,0,0)
L = 1	(RGW,1,1)	(F,0,0)

3 RPW	R = 0	R = 1
L = 0	(RPW,0,1)	(LGW,1,1)
L = 1	--	--

4 RA	R = 0	R = 1
L = 0	(RQoS,0,0)	--
L = 1	(RP,0,0)	--

5 RQoS	R = 0	R = 1
L = 0	(RQ0A,0,1)	(LP',1,0)
L = 1	(RQeS,0,0)	(LP,1,0)

6 RQeS	R = 0	R = 1
L = 0	(RQ1B,1,0)	--
L = 1	(RG1,0,1)	--

7 RQ1A	R = 0	R = 1
L = 0	(RQ0A,0,0)	(LG,1,0)
L = 1	--	--

8 RQ0A	R = 0	R = 1
L = 0	(RQ1A,0,0)	(RQ1A,1,0)
L = 1	(RQ1A,0,0)	(RP1,1,1)

9 RQ1B	R = 0	R = 1
L = 0	(RQ0B,0,0)	(LP,1,0)
L = 1	--	--

10 RQ0B	R = 0	R = 1
L = 0	(RQ1B,0,0)	(RQ1B,1,0)
L = 1	(RQ1B,0,0)	(RG1,1,1)

11 RQ1C	R = 0	R = 1
L = 0	(LQ1B,0,0)	(LQ1C,0,0)
L = 1	(RQ0C,0,0)	(LP,1,0)

12 RQ0C	R = 0	R = 1
L = 0	(LQ1A,0,1)	--
L = 1	(RQ1C,0,0)	(RG1,0,1)

13 RG1	R = 0	R = 1
L = 0	(RG0,0,0)	(LPW,1,0
L = 1	(RG0,0,0)	(LPW,1,0

14 RG0	R = 0	R = 1
L = 0	(RG1,0,1)	(RQ1B,0,0
L = 1	(RG1,0,1)	(RQ1C,0,0)

15 RP1	R = 0	R = 1
L = 0	(RP0,0,0)	(LGW,1,1)
L = 1	--	--

16 RP0	R = 0	R = 1
L = 0	(RP1,0,1)	(RQ1A,1,0)
L = 1	--	--

17 RG	R = 0	R = 1
L = 0	(LQ1C,0,0)	(LPW,1,0
L = 1	(LQ1C,0,0)	(LPW,1,0

18 RP	R = 0	R = 1
L = 0	(LQ0C,0,0)	(LGW,1,1
L = 1	(RP,0,1)	(LGW,1,1

19 LGW	R = 0	R = 1
L = 0	(LGW,1,0)	(LGW,1,1)
L = 1	(F,0,0)	(F,0,0)

20 LPW	R = 0	R = 1
L = 0	(LPW,1,0)	--
L = 1	(RGW,1,1)	-

21 LQ1A	R = 0	R = 1
L = 0	(LQ0A,0,0)	(LQ1B,0,0
L = 1	(RG,0,1)	(LP1,1,0)

22 LQ1B	R = 0	R = 1
L = 0	(LQ0B,0,0)	(LQ1A,0,0
L = 1	(RP,0,1)	(RP,0,0)

23 LQ0A	R = 0	R = 1
L = 0	(LQ1A,0,0)	(LQ1A,0,0)
L = 1	(LQ1A,0,1)	(LP1,1,1)

24 LQ0B	R = 0	R = 1
L = 0	(LQ1B,0,0)	(LQ1B,0,0)
L = 1	(LQ1B,0,1)	(LG1,1,1)

25 LQ1C	R = 0	R = 1
L = 0	(RQ1B,0,0)	(LQ0C,0,0
L = 1	(RQ1C,0,0)	(RP,0,1)

26 LQ0C	R = 0	R = 1
L = 0	(RQ1A,1,0)	(LQ1C,0,0
L = 1	--	(LG1,1,0)

27 LG1	R = 0	R = 1
L = 0	(LG0,0,0)	(LG0,0,0)
L = 1	(RPW,0,1)	(RPW,0,1)

28 LG0	R = 0	R = 1
L = 0	(LG1,1,0)	(LG1,1,0)
L = 1	(LQ1B,0,0)	(LQ1C,0,0)

29 LP1	R = 0	R = 1
L = 0	(LP0,0,0)	(LP0,0,0)
L = 1	(RGW,1,1)	(RPW,0,0)

30 LP0	R = 0	R = 1
L = 0	(LP1,1,0)	(LP1,1,0)
L = 1	(LQ1A,0,1)	(LQ1B,0,0)

31 LG	R = 0	R = 1
L = 0	(RQ1C,0,0)	(RQ1C,0,0)
L = 1	(RPW,0,1)	(RPW,0,1)

32 LP	R = 0	R = 1
L = 0	(RQ0C,0,0)	(LP,1,0)
L = 1	(RGW,1,1)	(RGW,1,1)

33 LP'	R = 0	R = 1
L = 0	--	(LQ1A,0,0)
L = 1	--	(RPW,0,0)

34 F	R = 0	R = 1
L = 0	--	--
L = 1	--	--

35 QW	R = 0	R = 1
L = 0	(QW,0,0)	--
L = 1	(LGW,1,0)	--

6.3.2 Generalized FSSP with a General at an Arbitrary Point

Section 6.3.2 considers a *generalized* firing squad synchronization problem which allows the initial general to be located anywhere on the array. It has been shown to be impossible to synchronize any array of length n less than $n-2+\max(k, n-k+1)$ steps, where the general is located on C_k, $1 \leq k \leq n$. Moore and Langdon [24], Szwerinski [30] and Varshavsky et al. [43] developed a generalized optimum-time synchronization algorithm for O(1)-bit cellular automaton each with 17, 10 and 10 internal states, respectively, that can synchronize any array of length n at exactly $n-2+\max(k, n-k+1)$ steps. Recently, Settle and Simon [28] and Umeo et al. [34] have also proposed a 9-state generalized synchronization algorithm operating in optimum-step for the O(1)-bit model.

Umeo et al. [34] developed a generalized synchronization algorithm on the $CA_{1\text{-bit}}$ model operating in non-optimum steps. The implementation for the $CA_{1\text{-bit}}$

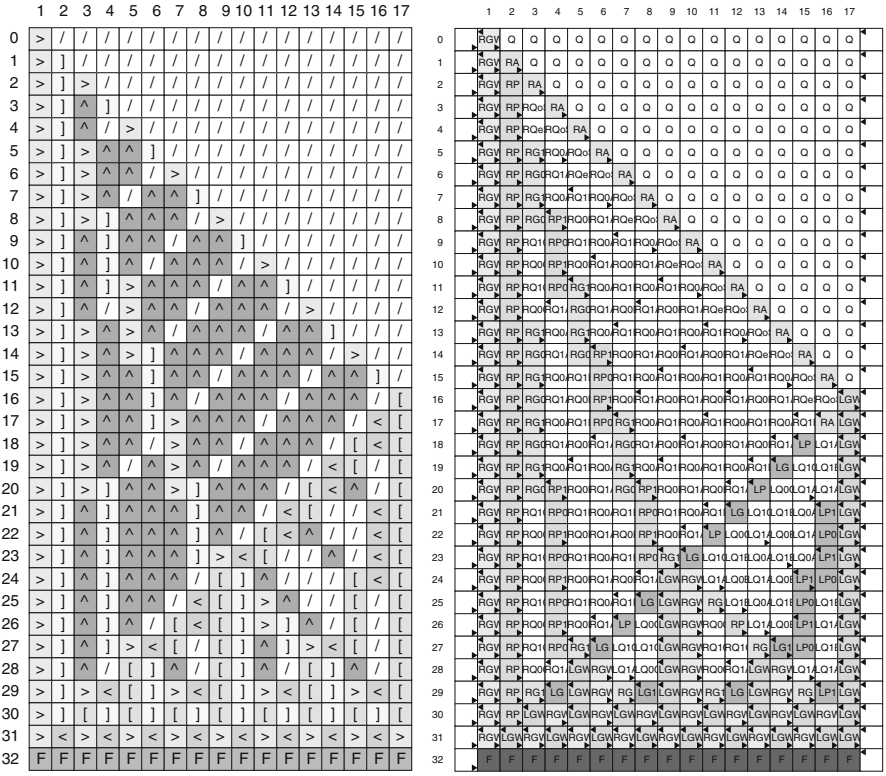

Fig. 6.5 Snapshots for synchronization processes on 17 cells, each for Gerken's algorithm [9] on O(1)-bit-communication model (*left*) and the 35-state implementation (Umeo and Yanagihara [41]) on $\mathrm{CA}_{\text{1-bit}}$ (*right*)

has 282-state and 721 transition rules. Kamikawa and Umeo [12] also developed a generalized synchronization algorithm on the $\mathrm{CA}_{\text{1-bit}}$ model operating in $n+\max(k, n-k+1)$ steps, which is one-step larger than optimum-step. The total numbers of internal states and transition rules of the constructed $\mathrm{CA}_{\text{1-bit}}$ are 219 and 488, respectively. Figure 6.2 (right) shows a space-time diagram for the optimum-time generalized firing squad synchronization algorithm. We also show some snapshots for the synchronization processes on 21 cells with a general at C_7 on $\mathrm{CA}_{\text{1-bit}}$ in Fig. 6.6. We present Table 6.3 that shows a quantitative comparison of synchronization algorithms and their implementations proposed so far with respect to the number of internal states of each finite state automaton, the number of transition rules realizing the synchronization and time complexity.

Theorem 5 (Kamikawa and Umeo [12]) *There exists a 219-state, 488-transition-rule* $CA_{\text{1-bit}}$ *that can synchronize n cells in* $n-1+max(k, n-k+1)$ *steps, where k is any integer such that* $1 \leq k \leq n$ *and a general is located on the kth cell from the left end of the array.*

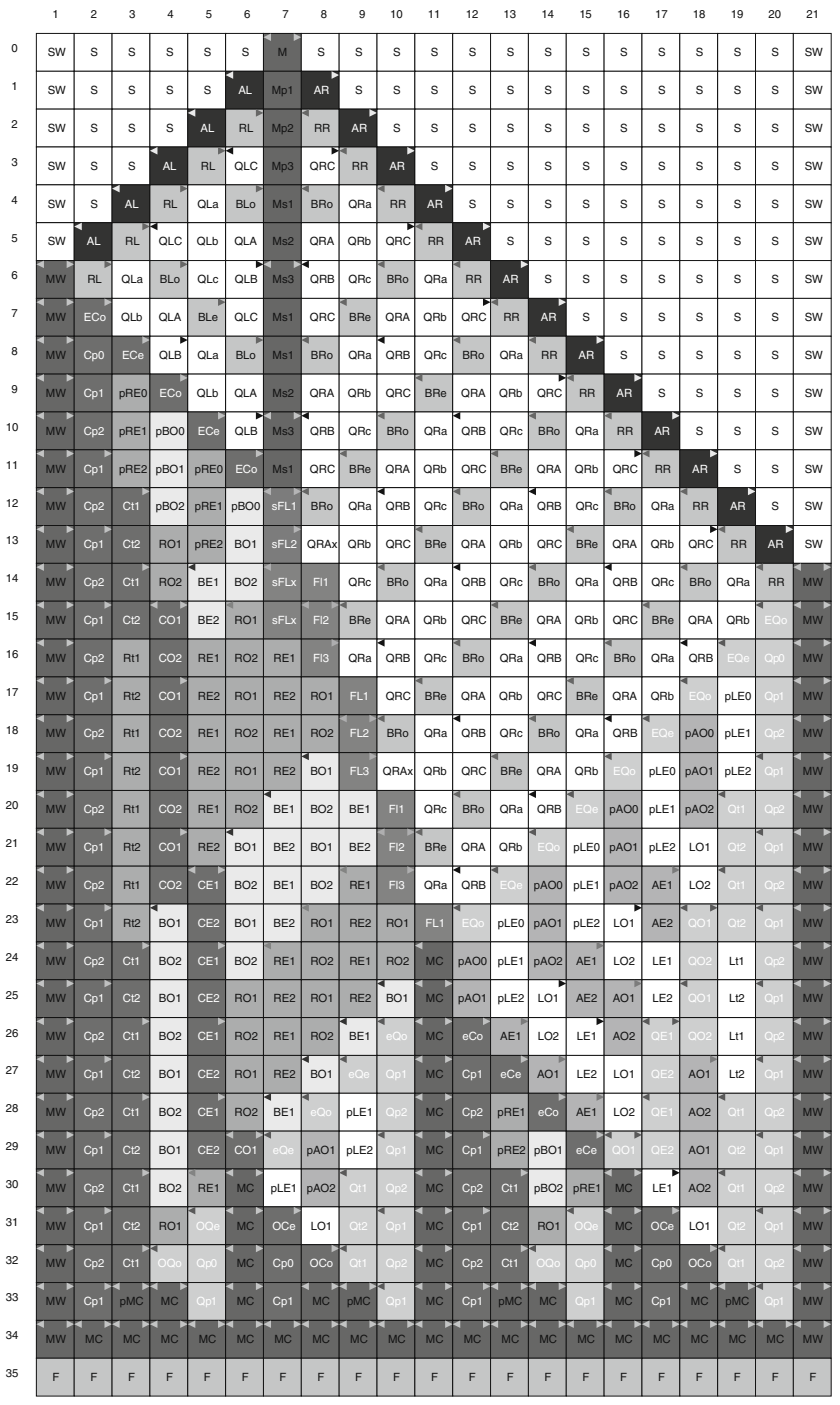

Fig. 6.6 Snapshots for the generalized synchronization processes on 21 cells with a general at C_7 on $CA_{1\text{-bit}}$

Table 6.3 A list of firing squad synchronization algorithms for $CA_{1\text{-bit}}$. A symbol "$*$" indicates the reconstructed rule set given in Table 6.1

Implementations	No. of states	No. of rules	Time complexity	Prototype algorithms
Mazoyer [21]	61*(58)	167*	$2n-2$	Balzer [4]
Nishimura et al. [25]	78	208	$2n-2$	Waksmann [45]
Umeo et al. [42]	54	207	$2n-1$	Mazoyer [19]
Umeo and Yanagihara [41]	35	114	$2n-2$	Gerken [9]
Umeo et al. [34]	282	721	$n+\max(k, n-k+1)$	–
Kamikawa and Umeo [12]	219	488	$n-1+\max(k, n-k+1)$	–

6.4 Prime Sequence Generation Problem

Sequence generation is an important, fundamental problem in cellular automata. Arisawa [3], Fischer [8], Korec [4] and Mazoyer and Terrier [20] have considered the sequence generation problem on the conventional O(1)-bit cellular automata model. Fischer [8] showed that the prime sequence can be generated in real-time on the O(1)-bit cellular automata with 11 states for C_1 and 37 states for $C_i (i \geq 2)$. Arisawa [3] also developed a real-time prime generator and decreased the number of states of each cell to 22. Korec [14] reported a real-time prime generator having 11 states on the same model.

Here we study a real-time prime generator on $CA_{1\text{-bit}}$. The sequence generation problem on $CA_{1\text{-bit}}$ can be defined as follows: Let M be a $CA_{1\text{-bit}}$, and $\{t_n | n = 1, 2, 3, \ldots\}$ be an infinite monotonically increasing positive integer sequence defined on natural numbers such that $t_n \geq n$ for any $n \geq 1$. We then have a semi-infinite array of cells, and all cells, except for C_1, are in the quiescent state at time $t = 0$. The communication cell C_1 assumes a special state r in Q and outputs 1 to its right communication link at time $t = 0$ for initiation of the sequence generator. We say that M generates a sequence $\{t_n | n = 1, 2, 3, \ldots\}$ in k *linear-time* if and only if the leftmost end cell of M falls into a special state in $F \subseteq Q$ and outputs 1 to its leftmost communication link at time $t = kt_n$, where k is a positive integer. We call M a *real-time generator* when $k = 1$.

In this section, we present a real-time prime generation algorithm on $CA_{1\text{-bit}}$. The algorithm is implemented on a $CA_{1\text{-bit}}$ using 34 internal states and 71 transition rules. Our real-time prime generation algorithm is based on the well-known sieve of Eratosthenes. Details can be found in Umeo and Kamikawa [37]. Figure 6.7 is a space-time diagram for the real-time prime generation algorithm. We have implemented the algorithm on a computer. Each cell has 34 internal states and 71 transition rules. The transition rule set is given in Table 6.4. We have tested the validity of the rule set from $t = 0$ to $t = 20000$ steps. In Fig. 6.8, we show a number of snapshots of the configuration from $t = 0$ to 40. The readers can see that the first 11 primes can be generated in real-time by the left end cell. Now we have:

Theorem 6 (Umeo and Kamikawa [37]) *Prime sequence can be generated by a $CA_{1\text{-bit}}$ in real-time.*

Table 6.5 is a list of typical non-regular sequences generated by $CA_{1\text{-bit}}$ in real-time.

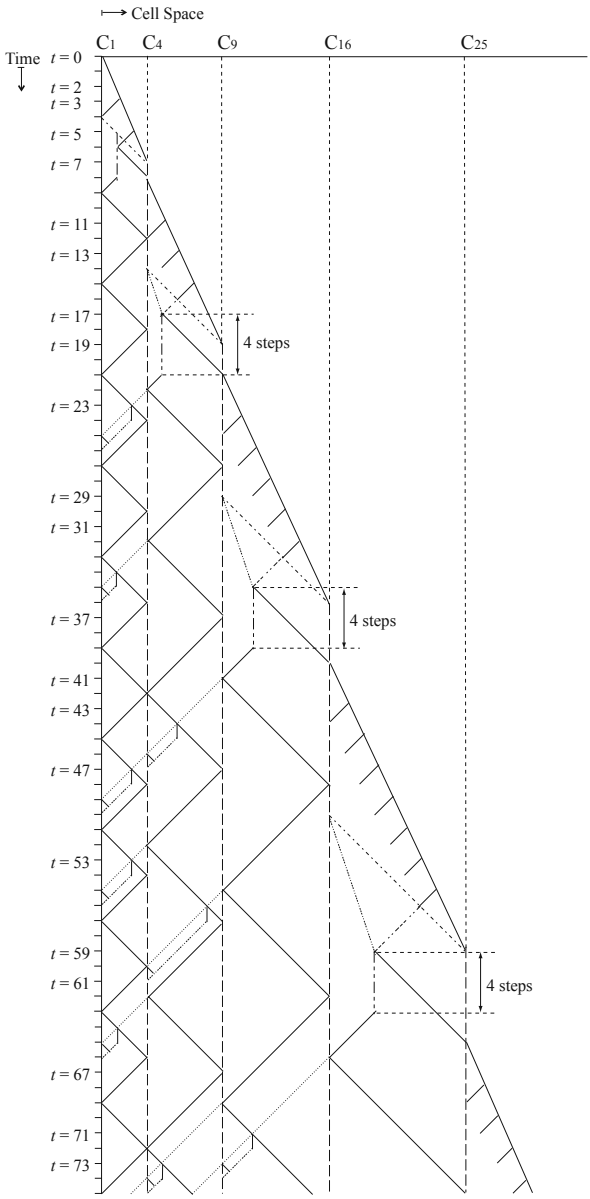

Fig. 6.7 Space-time diagram for real-time prime generation

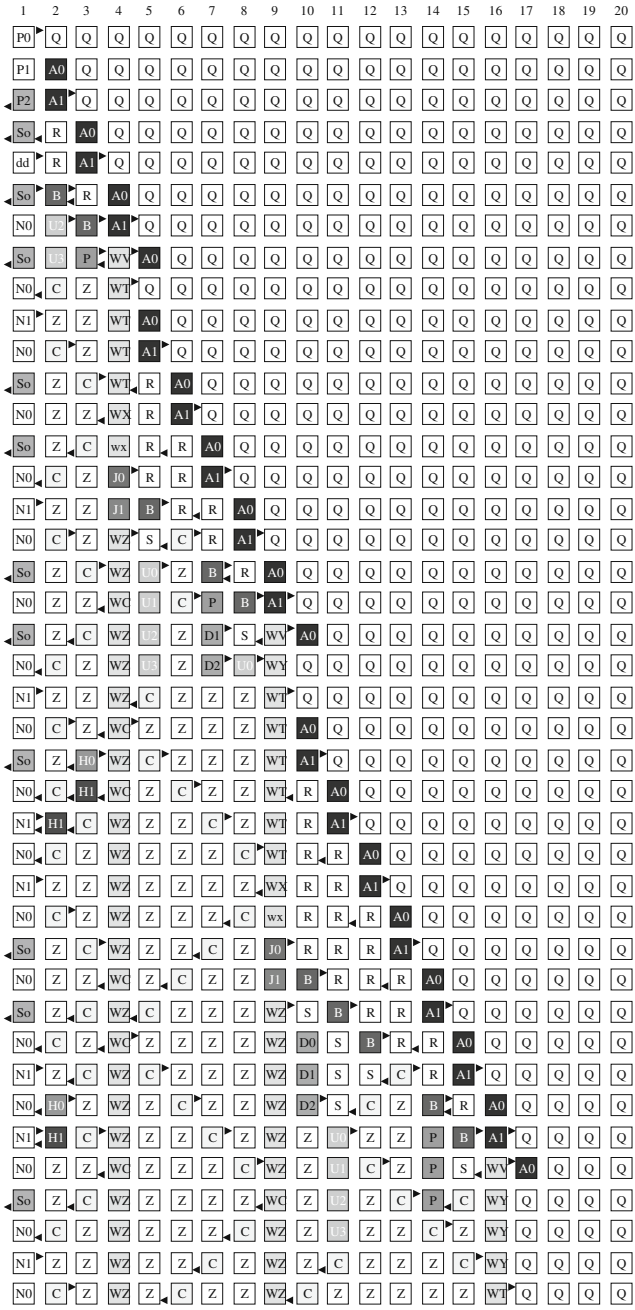

Fig. 6.8 A configuration of real-time generation of prime sequences on the $CA_{1\text{-bit}}$ with 34 states

Table 6.4 Transition rule set for real-time prime generator

Internal states : {Q, P0, P1, P2, R, S, Z, So, N0, N1, A0, A1, B, C, D0, D1, D2, U0, U1, U2, U3, WV, WY, WX, wx, WT, WZ, WC, dd, P, J0, J1, H0, H1}

Current state \ Input from left link	Input from right link (next state, left output, right output)

1

Q	R = 0	R = 1
L = 0	(Q,0,0)	--
L = 1	(A0,0,0)	--

2

P0	R = 0	R = 1
L = 0	(P1,0,0)	--
L = 1	--	--

3

P1	R = 0	R = 1
L = 0	(P2,1,0)	--
L = 1	--	--

4

P2	R = 0	R = 1
L = 0	(So,1,0)	--
L = 1	--	--

5

R	R = 0	R = 1
L = 0	(R,0,0)	(R,0,0)
L = 1	(B,0,1)	(C,1,1)

6

S	R = 0	R = 1
L = 0	(S,0,0)	(C,1,0)
L = 1	(D0,0,0)	(U0,0,1)

7

Z	R = 0	R = 1
L = 0	(Z,0,0)	(C,1,0)
L = 1	(C,0,1)	(H0,1,1)

8

So	R = 0	R = 1
L = 0	(N0,0,0)	(dd,0,1)
L = 1	--	--

9

N0	R = 0	R = 1
L = 0	(So,1,0)	(N1,0,1)
L = 1	--	--

10

N1	R = 0	R = 1
L = 0	(N0,0,0)	(N0,0,0)
L = 1	--	--

11

A0	R = 0	R = 1
L = 0	(A1,0,1)	--
L = 1	(Q,0,0)	--

12

A1	R = 0	R = 1
L = 0	(R,1,0)	--
L = 1	(WV,1,1)	--

13

B	R = 0	R = 1
L = 0	(S,0,0)	(P,0,0)
L = 1	(P,0,1)	(U2,0,1)

14

C	R = 0	R = 1
L = 0	(Z,0,0)	(H1,1,0)
L = 1	--	--

15

D0	R = 0	R = 1
L = 0	(D1,0,0)	--
L = 1	--	--

16

D1	R = 0	R = 1
L = 0	(D2,0,1)	--
L = 1	--	--

17

D2	R = 0	R = 1
L = 0	(Z,0,0)	--
L = 1	--	--

18

U0	R = 0	R = 1
L = 0	(U1,0,0)	--
L = 1	(Z,0,0)	--

19

U1	R = 0	R = 1
L = 0	(U2,0,0)	--
L = 1	--	--

20

U2	R = 0	R = 1
L = 0	(U3,0,0)	--
L = 1	--	--

21

U3	R = 0	R = 1
L = 0	(C,1,0)	--
L = 1	--	--

22

WV	R = 0	R = 1
L = 0	(WY,0,0)	--
L = 1	(WT,0,1)	--

23

WY	R = 0	R = 1
L = 0	(WY,0,0)	(WY,0,0)
L = 1	(WT,0,1)	(WX,0,0)

24

WX	R = 0	R = 1
L = 0	(wx,0,0)	--
L = 1	--	--

25

wx	R = 0	R = 1
L = 0	(J0,0,1)	--
L = 1	--	--

26

WT	R = 0	R = 1
L = 0	(WT,0,0)	(WT,0,0)
L = 1	(WX,1,0)	(WX,1,0)

27

WZ	R = 0	R = 1
L = 0	(WZ,0,0)	(WC,1,1)
L = 1	(WC,1,0)	(WC,1,1)

28

WC	R = 0	R = 1
L = 0	(WZ,0,0)	(WZ,0,0)
L = 1	--	--

29

dd	R = 0	R = 1
L = 0	(So,1,1)	--
L = 1	--	--

30

P	R = 0	R = 1
L = 0	(P,0,0)	(Z,0,0)
L = 1	(D1,0,1)	(C,0,1)

31

J0	R = 0	R = 1
L = 0	(J1,0,0)	--
L = 1	--	--

32

J1	R = 0	R = 1
L = 0	(WZ,0,1)	--
L = 1	--	--

33

H0	R = 0	R = 1
L = 0	(H1,1,0)	--
L = 1	--	--

34

H1	R = 0	R = 1
L = 0	(Z,0,0)	(C,1,0)
L = 1	(Z,0,0)	(C,1,0)

Table 6.5 A list of non-regular sequences generated by $CA_{1\text{-bit}}$ in real-time

Sequences	No. of states	No. of rules	Time complexity	References
$\{2^n \mid n = 1, 2, 3, \ldots\}$	4	12	Real-time	Umeo and Kamikawa [36]
$\{n^2 \mid n = 1, 2, 3, \ldots\}$	3	7	Real-time	Umeo and Kamikawa [36]
Fibonacci	9	26	Real-time	Umeo and Kamikawa [36]
Prime	34	71	Real-time	Umeo and Kamikawa [37]

6.5 Early Bird Problem

In this section, we study an early bird problem on $CA_{1\text{-bit}}$. Consider a one-dimensional $CA_{1\text{-bit}}$ consisting of n cells in which any cell initially in a quiescent state may be excited from outside world. The problem is to describe the automata

(state set and next state function) so that the first excitation(s) can be distinguished from the later excitations. This problem was originally devised by Rosenstiehl et al. [27] to design some graph-theoretic algorithms operating on networks of finite state automata with O(1)-bit communication. Rosenstiehl et al. [27] presented a $2n$-step solution on a condition that at most one excitation occurs at each step. Vollmar [44] extended the problem allowing more than one cell to be excited at a given step. Legendi and Katona [16] gave a 5-state solution with multiple excitations operating in $3n+O(1)$ steps on a conventional CA of length n. Kleine-Büning [13] showed that the 5-state solution developed by Legendi and Katona [16] is the optimal solution with regard to the number of internal states of each cell on the O(1)-bit communication model.

Based on the 5-state solution given by Legendi and Katona [16], Umeo et al. [40] have given a 37-state implementation on $CA_{1\text{-bit}}$ of size n operating in $6n+O(1)$ steps. In our implementation, multiple birds are allowed to appear at only even steps. Two steps are required for the simulation of each one step of Legendi and Katona's solution. Thus the time complexity for the implemented algorithm is twice. An improvement in the time complexity seems to be difficult. Figure 6.9 shows some snapshots of the 37-state implementation. An appearance of the early bird is repre-

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	QW	Q	B	Q	Q	Q	Q	Q	Q	Q	Q	B	Q	Q	Q	Q	B	Q	Q	QW
1	QW	q	B_	q	Q	Q	Q	Q	Q	Q	q	B_	q	Q	Q	q	B_	q	Q	QW
2	QW	La	B_	Ra	Q	Q	Q	Q	B	Q	La	B_	Ra	Q	Q	La	B_	Ra	Q	QW
3	QW_	Lb	B_	Rb	q	Q	Q	q	B_	q	Lb	B_	Rb	q	q	Lb	B_	Rb	q	QW
4	QW_	L3a	B_	R3a	Ra	B	Q	La	B_	q	L3a	B_	R3a	Ra	La	L3a	B_	R3a	Ra	QW
5	LRW	L3b	B_	R3b	Rb'	B10	q	Lb	B_	N	L3b	B_	R3b	Rb'	Lb'	L3b	B_	R3b	Rb	QW_
6	Nl	L3a	B_	R3a	Nr	R4a	q	L3a	B01	L2a	Nl	B_	R3a	Nr	Nl	L3a	B_	R3a	R2a	QW_
7	Nlb	L3b	B_	R3b	Nrb	R3b	N	L3b	B01	L2b	Nlb	B_	R3b	Nrb	Nlb	L3b	B_	R3b	R2b	LRW
8	L2a	Nl	B_	Nr	R4a	Nr	N	Nl	L4a	Nla	N	B10	Nr	R2a	L2a	Nl	B_	R3a	R2a	Nr
9	L2b	Nlb	B_	Nrb	R3b	Nrb	N	Nlb	L3b	Nlb	N	B10	Nrb	R2b	L2b	Nlb	B_	R3b	R2b	Nrb
10	Nla	N	B11	Nr2	Nr	R2a	N	L2a	Nl	Nl2	N	B11	N	Nra	Nla	N	B10	R3a	Nr	R2a
11	Nlb	N	B11	Nr2	Nrb	R2b	N	L2b	Nlb	Nl2	N	B11	N	Nrb	Nlb	N	B10	R3b	Nrb	R2b
12	N	N	B11	N	N	Nra	N	Nla	N	N	N	B11	N	N	N	N	B10	Nr	R2a	Nra
13	N	N	B11	N	N	Nrb	N	Nlb	N	N	N	B11	N	N	N	N	B10	Nrb	R2b	Nrb
14	N	N	B11	N	N	N	N	N	N	N	N	B11	N	N	N	N	B11	N	Nra	R2a
15	N	N	B11	N	N	N	N	N	N	N	N	B11	N	N	N	N	B11	N	Nrb	R2b
16	N	N	B11	N	N	N	N	N	N	N	N	B11	N	N	N	N	B11	N	N	Nra
17	N	N	B11	N	N	N	N	N	N	N	N	B11	N	N	N	N	B11	N	N	Nrb
18	N	N	B11	N	N	N	N	N	N	N	N	B11	N	N	N	N	B11	N	N	N

Fig. 6.9 Snapshots of a 37-state implementation of the early bird problem on $CA_{1\text{-bit}}$

sented by an internal state "B" with output signal "1" to both neighbors. In Fig. 6.9, three birds appear on C_3, C_{12} and C_{17} at time $t = 0$ and one bird appear on C_9 at time $t = 2$ and on C_6 at time $t = 4$, respectively. The first three birds can survive and the last two birds will be *killed*. The transition rule set is given in Table 6.6.

Theorem 7 (Umeo, Michisaka, Kamikawa, and Kanazawa [40]) *There exists a 37-state $CA_{1\text{-bit}}$ that can solve the early bird problem in $6n+O(1)$ steps under an assumption such that multiple excitations are allowed to appear at only even steps.*

Table 6.6 Transition rule set for early bird generator

Internal state : {Q, q, QW, QW, B, B_, BW, BW, B01, B10, B11, Ra, Rb, R2a, R2b, R3a, R3b, R4a, La, Lb, L2a, L2b, L3a, L3b, L4a, LRW, Nr, Nra, Nrb, Nr2, Nl, Nla, Nlb, Nl2, N, Lb', Rb'}

Current state	Input from right link
Input from left link	(next state , left output , right output)

1

Q	R=0	R=1
L= 0	(Q,0,0)	(q,0,0)
L= 1	(q,0,0)	(q,0,0)

2

q	R=0	R=1
L= 0	(N,1,1)	(La,1,0)
L= 1	(Ra,0,1)	(q,0,0)

3

QW	R=0	R=1
L= 0	(QW,0,0)	(QW_,0,0)
L= 1	(QW_,0,0)	--

4

QW_	R=0	R=1
L= 0	(LRW,0,0)	(QW_,0,0)
L= 1	(QW_,0,0)	--

5

B	R=0	R=1
L= 0	(B_,1,1)	(B01,1,0)
L= 1	(B10,0,1)	(B11,0,0)

6

B_	R=0	R=1
L= 0	(B_,0,0)	(B01,0,0)
L= 1	(B10,0,0)	(B11,0,0)

7

BW	R=0	R=1
L= 0	(BW_,1,1)	(B11,0,0)
L= 1	(B11,0,0)	--

8

BW_	R=0	R=1
L= 0	(BW_,0,0)	(B11,0,0)
L= 1	(B11,0,0)	--

9

B01	R=0	R=1
L= 0	(B01,0,0)	(L4a,0,0)
L= 1	(B11,0,0)	(L2a,0,0)

10

B10	R=0	R=1
L= 0	(B10,0,0)	(B11,0,0)
L= 1	(R4a,0,0)	(R2a,0,0)

11

B11	R=0	R=1
L= 0	(B11,0,0)	(L2a,0,0)
L= 1	(R2a,0,0)	(N,0,0)

12

Ra	R=0	R=1
L= 0	(Rb,0,1)	(Rb',0,1)
L= 1	--	--

13

Rb	R=0	R=1
L= 0	(R3a,0,0)	(Nr,0,0)
L= 1	(R2a,0,0)	(Nr,0,0)

14

R2a	R=0	R=1
L= 0	(R2b,0,1)	--
L= 1	--	--

15

R2b	R=0	R=1
L= 0	(Nra,0,0)	(Nra,0,0)
L= 1	(R2a,0,0)	(Nr,0,0)

16

R3a	R=0	R=1
L= 0	(R3b,0,1)	--
L= 1	--	--

17

R3b	R=0	R=1
L= 0	(R3a,0,0)	(Nr,0,0)
L= 1	(R2a,0,0)	(Nr,0,0)

18

R4a	R=0	R=1
L= 0	(R3b,1,1)	--
L= 1	--	--

19

La	R=0	R=1
L= 0	(Lb,1,0)	--
L= 1	(Lb',1,0)	--

20

Lb	R=0	R=1
L= 0	(L3a,0,0)	(L2a,0,0)
L= 1	(Nl,0,0)	(Nl,0,0)

21

L2a	R=0	R=1
L= 0	(L2b,1,0)	--
L= 1	--	--

22

L2b	R=0	R=1
L= 0	(Nla,0,0)	(L2a,0,0)
L= 1	(Nla,0,0)	(Nl,0,0)

23

L3a	R=0	R=1
L= 0	(L3b,1,0)	--
L= 1	--	--

24

L3b	R=0	R=1
L= 0	(L3a,0,0)	(L3a,0,0)
L= 1	(Nl,0,0)	(Nl,0,0)

25

L4a	R=0	R=1
L= 0	(L3b,1,1)	--
L= 1	--	--

26

LRW	R=0	R=1
L= 0	(q,0,0)	(Nl,0,0)
L= 1	(Nr,0,0)	--

27

Nr	R=0	R=1
L= 0	(Nrb,1,0)	--
L= 1	--	--

28

Nra	R=0	R=1
L= 0	(Nrb,0,0)	--
L= 1	--	--

29

Nrb	R=0	R=1
L= 0	(N,0,0)	(Nr2,0,0)
L= 1	(R2a,0,0)	(R4a,0,0)

30

Nr2	R=0	R=1
L= 0	(Nr2,0,0)	(N,0,0)
L= 1	(R4a,0,0)	(R2a,0,0)

31

Nl	R=0	R=1
L= 0	(Nlb,0,1)	--
L= 1	--	--

32

Nla	R=0	R=1
L= 0	(Nlb,0,0)	--
L= 1	--	--

33

Nlb	R=0	R=1
L= 0	(N,0,0)	(L2a,0,0)
L= 1	(Nl2,0,0)	(L4a,0,0)

34

Nl2	R=0	R=1
L= 0	(Nl2,0,0)	(L4a,0,0)
L= 1	(N,0,0)	(L2a,0,0)

35

N	R=0	R=1
L= 0	(N,0,0)	(L2a,0,0)
L= 1	(R2a,0,0)	(N,0,0)

36

Lb'	R=0	R=1
L= 0	(Nl,0,0)	(Nl,0,0)
L= 1	(Nl,0,0)	(Nl,0,0)

37

Rb'	R=0	R=1
L= 0	(Nr,0,0)	(Nr,0,0)
L= 1	(Nr,0,0)	(Nr,0,0)

6.6 Firing Squad Synchronization Problem on 2-D $CA_{1\text{-bit}}$

Here we consider the FSSP again on two-dimensional arrays. The FSSP on 2-D arrays for O(1)-bit communication model is studied in Umeo et al. [38]. Figure 6.10 shows a finite two-dimensional (2-D) cellular array consisting of $m \times n$ cells. A cell on (i, j) is denoted by $C_{i,j}$. 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 binary inputs from its north, south, east and west neighbors. The cell also outputs four binary values to its north, west, south and east neighbors, depending on both its own present state and the present binary inputs from its north, south, east and west neighbors. Thus we assume a von Neumann-like neighborhood with the 1-bit communication. All cells except for the general cell are initially in the quiescent state and have a property such that the next state of a quiescent cell with four 0 inputs is the quiescent state and outputs 0 to its four neighbors.

The FSSP on 2-D $CA_{1\text{-bit}}$ is defined as follows: Given an array of $m \times n$ identical cells, including a *General* on $C_{1,1}$ cell that is activated at time $t = 0$, we want to describe (state set and next-state function) the automata such 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 and transition rules must be independent of m and n. The difficult part of this problem is that the same types of cells with a fixed number of states must be synchronized, regardless of the size m and n of the array. The firing squad synchronization problem on 2-D 1-bit communication cellular automata has been studied by Torre et al. [31], Gruska et al. [11], and Umeo et al. [40]. This section presents two 1-bit implementations for square and rectangular arrays.

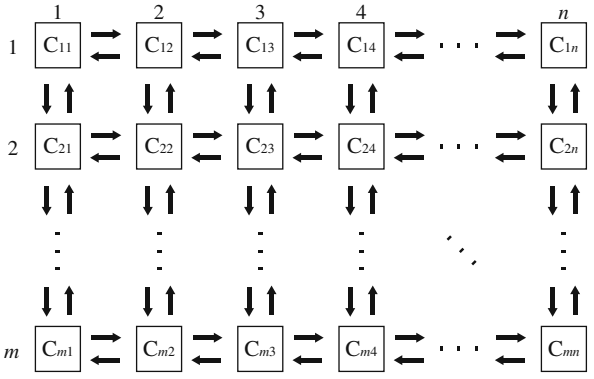

Fig. 6.10 Two-dimensional cellular automaton

6.6.1 Synchronization Algorithm on Square Arrays

The first one is for square arrays given in Umeo et al. [40]. It runs in $(2n - 1)$ steps on $n \times n$ square arrays. The proposed implementation is one step slower than

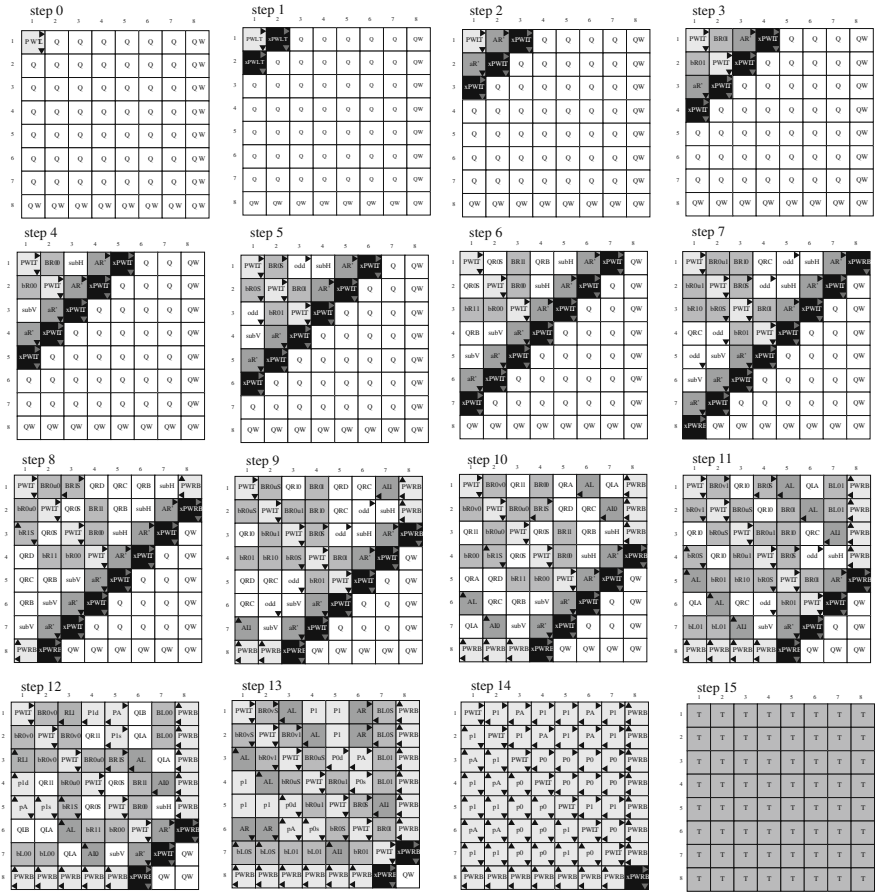

Fig. 6.11 Snapshots of the $(2n-1)$-step square synchronization algorithm with the general on the northwest corner

optimum-time for the O(1)-bit communication model. The total numbers of internal states and transition rules of the $\text{CA}_{\text{1-bit}}$ are 127 and 405, respectively. Figure 6.11 shows snapshots of configurations of the 127-state implementation running on a square of size 8×8. Gruska, Torre, and Parente [11] presented an optimum-time algorithm.

Theorem 8 (Gruska, Torre, and Parente [11]) *There exists a 2-D* $CA_{\text{1-bit}}$ *that can synchronize any* $n \times n$ *square arrays in* $2n-2$ *steps.*

6.6.2 Synchronization Algorithm on Rectangle Arrays

The generalized firing squad synchronization algorithm for 1-D arrays presented in Sect. 6.3.2 can be applied to the problem of synchronizing rectangular arrays with

the general at the northwest corner. The rectangular array is regarded as $\min(m, n)$ L-shaped 1-D arrays that are synchronized independently using the generalized firing squad synchronization algorithm. Configurations of the generalized synchronization on 1-D $CA_{1\text{-bit}}$ can be embedded on 2-D array. The original embedding scheme for O(1)-bit communication model was presented in Beyer [5] and Shinahr [29] in order to synchronize any $m \times n$ arrays in optimum $m + n + max(m, n) - 3$ steps. Umeo et al. [40] have implemented the rectangular synchronization algorithm for 2-D $CA_{1\text{-bit}}$. The total numbers of internal states and transition rules of the $CA_{1\text{-bit}}$ are 862 and 2217, respectively. Figure 6.12 shows snapshots of the synchronization process on a 5×8 rectangular array. Thus we have:

Theorem 9 (Umeo, Michisaka, Kamikawa, and Kanazawa [40]) *There exists a 2-D $CA_{1\text{-bit}}$ that can synchronize any $m \times n$ rectangular arrays in $m + n + max(m, n)$ steps.*

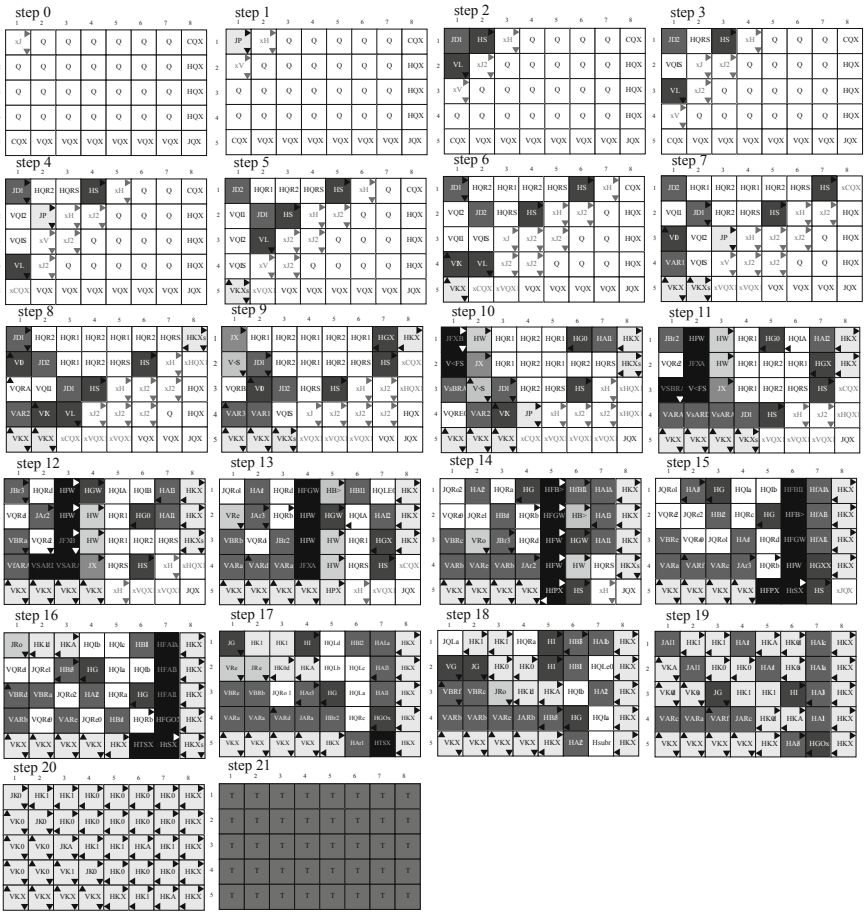

Fig. 6.12 Snapshots of the proposed rectangular firing squad synchronization algorithm with the general at the northwest corner

6.7 Connectivity Recognition Problem

Recognizing and labeling connected regions of images are important problems in image processing and machine vision, and many parallel algorithms for them have been developed on a rich variety of parallel architectures. See Alnuweiri and Prasanna [1, 2], Cypher et al. [6], Cypher and Sanz [7], Leighton [15], Manohar and Ramapriyan [18], and Miller and Stout [22]. In this section, we consider a connectivity recognition problem on a 2-D $CA_{1\text{-bit}}$. The connectivity recognition problem of binary images on cellular automata has been investigated by Beyer [5] and Levialdi [17]. We present a linear-time connectivity recognition algorithm for two-dimensional binary images. Precisely, it is shown that a set of two-dimensional connected binary images of size $m \times n$ can be recognized in $2(m+n) + O(1)$ steps by a 2-D $CA_{1\text{-bit}}$.

6.7.1 Connectivity

Before describing the connectivity recognition algorithm, we need some definitions of the connectivity for binary images. We assume that the given image is of size $m \times n$ where a pixel (i, j) denotes the pixel in row i and column j of the image for every $1 \leq i \leq m$ and $1 \leq j \leq n$. We put an input of size $m \times n$ on the 2-D $CA_{1\text{-bit}}$ of the same size in such a way that the cell (i, j) receives the pixel (i, j) as its initial input. We are concerned with black and white binary images where the black pixel has 1-value and white one has 0-value, respectively. We regard black components as objects and white ones as a background of the objects. Due to technical reasons, we attach a boundary consisting of white pixels to the input image. Note that those boundary pixels are not counted as the size of the image. Connectivity among pixels can be defined in terms of adjacency. Two black pixels (i_1, j_1) and (i_2, j_2) are *4-adjacent* and they are said to be in *4-neighbor*, if $\mid i_1 - i_2 \mid + \mid j_1 - j_2 \mid \leq 1$. Two black pixels (i_1, j_1) and (i_k, j_k) are said to be *4-connected*, if there exists a sequence of black pixels (i_p, j_p), $2 \leq p \leq k$ such that each pair of (i_{p-1}, j_{p-1}) and (i_p, j_p) are in 4-neighbor. A maximum connected region of black pixels is called a *4-connected component*. A 4-connected component is *isolated* if it consists of only one black pixel. A pattern is said to be *4-connected* if it has exactly one 4-connected component. Thus we employ the 4-connectivity for black pixels. The readers can define 8-connectivity, similarly. See Rosenfeld [26] and Umeo and Mauri [39] for details.

6.7.2 Parallel Shrinking Transformation

Beyer [5] proposed an interesting parallel shrinking transformation which trims all 4-connected components of binary images simultaneously, preserving the connectivity of binary images. The transformation was implemented on a conventional O(1)-bit communication model of cellular automaton. The recognition algorithm

we develop here is based on the parallel shrinking algorithm proposed by Beyer [5]. We first review the Beyer's algorithm. The algorithm is based on a connectivity-preserving operation which trims all connected components simultaneously in the diagonal (from south-east to north-west) direction of each component. Figure 6.13 shows the Beyer's connectivity-preserving operation consisting of two rules R_1 for black pixels and R_2 for white pixels. If a black pixel has a white pixel in its south and east neighbors, then the rule R_1 is applied to the black pixel and the pixel becomes white at the next step. If a white pixel has three black pixels in its south, east, and south-east (diagonal) neighbors, respectively, then the rule R_2 is applied to the white pixel and it becomes black at the next step. The symbols x and y denote any value in {white, black}. When we apply the above operations repeatedly to all pixels of an image simultaneously, we observe that each connected component of the image is reduced to one isolated black pixel and then vanishes after one application of the rule R_1. What is important is that, all the while, every distinct connected component remains distinct and either vanishes at each different position or in the same position at different time. This is the reason why the Beyer's shrinking rule is called connectivity preserving operation.

Precisely, the above statement is described as follows: Let c be a non-isolated connected component and $n(c)$, $w(c)$, and $se(c)$ be positive integers defined as follows:

$$
\begin{aligned}
n(c) &= \min\{i \mid \text{cell } C_{i,k} \text{ is in } c \text{ for some } k\},\\
w(c) &= \min\{k \mid \text{cell } C_{i,k} \text{ is in } c \text{ for some } i\},\\
se(c) &= \max\{i + k \mid \text{cell } C_{i,k} \text{ is in } c\}.
\end{aligned}
$$

The connected component c is within the triangle consisting of the row $n(c)$, column $w(c)$, and the 45° diagonal line containing the farthest cells from the cell $C_{n(c),w(c)}$, shown in Fig. 6.14. Let ψ denote the Beyer's transformation and c be an

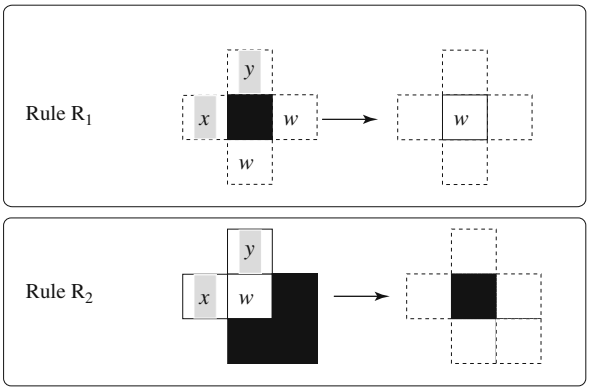

Fig. 6.13 Beyer's 4-connectivity preserving operation

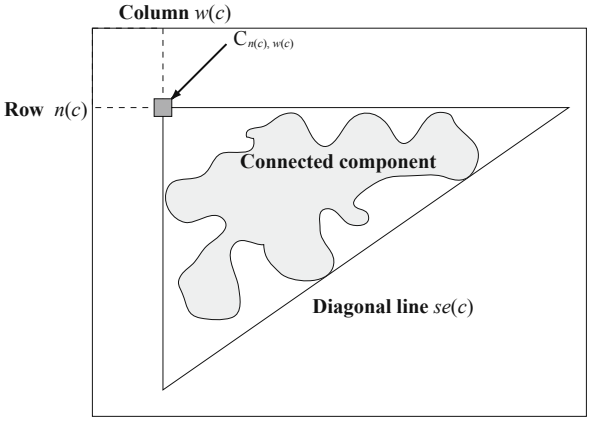

Fig. 6.14 Beyer's parallel shrinking transformation

image. A function $\psi^{t+1}(c)$ is defined as follows:

$$\psi^0(c) = c,$$
$$\psi^{t+1}(c) = \psi(\psi^t(c)), t \geq 0.$$

Then, the following lemmas are given in Beyer [5].

Lemma 10 (Beyer [5]) *For any non-isolated component c, we have* $n(\psi(c)) = N(c)$, $w(\psi(c)) = w(c)$, *and* $se(\psi(c)) = se(c) - 1$.

Lemma 11 (Beyer [5]) *For any non-isolated component c, let k be an integer such that* $k = se(c) - n(c) - w(c)$. *Then,* $\psi^k(c)$ *is an isolated component located at* $C_{n(c),w(c)}$.

Lemma 10 assures the exact shrinking to the north-west corner of the connected component. From Lemma 11, we can know the number of applications of the ψ operation necessary to shrink the original connected component to an isolated black pixel. It is shown that, for any image of size $m \times n$, all of the connected components will vanish within $(m+n-1)$-time applications of ψ. In Fig. 6.15, we show several snapshots obtained after consecutive applications of ψ to a binary image. Note that T means the application times.

6.7.3 One-Bit Implementation of Connectivity-Preserving Transformation

Here we show that the Beyer's connectivity-preserving transformation can be implemented on 2-D $\text{CA}_{\text{1-bit}}$. We construct a two-dimensional $\text{CA}_{\text{1-bit}}$ M that can simulate the Beyer's transformation in $2(m+n)-1$ steps for any given binary image x of size $m \times n$. Each cell has two auxiliary registers X and Y. The register X holds a binary pixel value during the transformation and Y acts as a temporary register for storing a pixel value in the south neighbor cell. Any operation of each cell at step

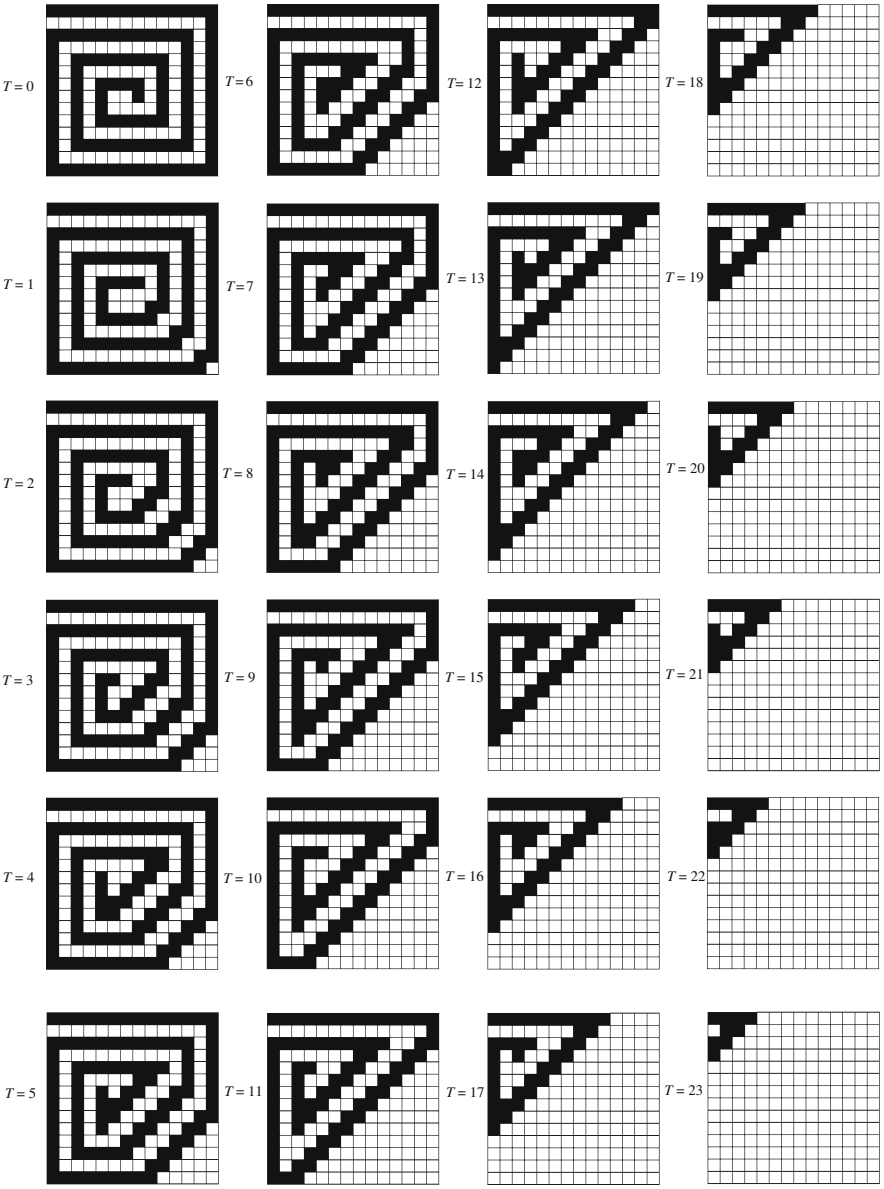

Fig. 6.15 Beyer's connectivity-preserving transformation on O(1)-bit communication model

t (≥ 1) is classified into two categories according to the parity of global clock step t such that $t \equiv 1 \pmod 2$ or $t \equiv 0 \pmod 2$. We refer to the former operation as A-phase operation and the latter B-phase operation, respectively. Each cell repeats an A- and B-phase operation alternatively, that is, it repeats two operations, one in A-phase followed by the other in B-phase.

The 1-bit implementation is as follows: At time $t = 1$, each register X in $C_{i,j}$ holds an initial pixel value (i, j) of x and the register Y has been set empty. At the beginning of the A-phase, each cell outputs a 0 or 1 signal to its north and west output communication links depending on the pixel value 0 or 1 in the X register, and the signals are received at that step by its north and west neighbor cells through their input communication links. In the shrinking transformation shown in Fig. 6.13, one step is sufficient for the execution of the Rule R_1, however, it takes two steps for the execution of the Rule R_2. To get the pixel value in the east and south neighbor cells, each cell uses its east and south input communication links in the A-phase operation. In order to get south-east (diagonal) pixel value, which is necessary for the execution of the rule R_2, each cell uses its east input communication link in the B-phase operation. For this purpose, the diagonal pixel value has been stored in the Y register in its east neighbor cell in the latest A-phase. Thus, with those two steps of the A- and B-phases, each cell can get all pixel values that are necessary to perform one application of the transformation. At odd step t of M where $t = 2k + 1$, for any k such that $0 \leq k \leq m + n - 1$, we can see the values of $\psi^0(x)$, $\psi^1(x)$, .., $\psi^k(x)$, .., $\psi^{m+n-1}(x)$ in the X registers of each cell on the array. It is observed that, in the construction above, both east-to-west horizontal

Table 6.7 Transition rule set for 1-bit shrinking transformation

Internal State : {B, W, B/B, R, BR, W/B-B}

Current State	Input from Right and Left Link
Input from Upper and Lower Link	(Next State, Left Output, Right Output, Upper Output, Lower Output)

1

B	R=0,L=0	R=1,L=0	R=0,L=1	R=1,L=1
U=0,D=0	(R,1,0,1,0)	(BR,1,1,0,0)	--	--
U=1,D=0	(R,1,0,1,0)	(BR,1,1,0,0)	--	--
U=0,D=1	(B/B,0,1,1,1)	(B/B,0,1,1,1)	--	--
U=1,D=1	(B/B,0,1,1,1)	(B/B,0,1,0,1)	--	--

2

W	R=0,L=0	R=1,L=0	R=0,L=1	R=1,L=1
U=0,D=0	(W,0,0,0,0)	(W,0,0,0,0)	(W,0,0,0,0)	(W,0,0,0,0)
U=1,D=0	(W,0,0,0,0)	- -	--	--
U=0,D=1	(W,0,0,0,0)	(W/B-B,0,0,0,0)	(W,0,0,0,0)	--
U=1,D=1	--	(W/B-B,0,0,0,0)	--	--

3

B/B	R=0,L=0	R=1,L=0	R=0,L=1	R=1,L=1
U=0,D=0	(B,1,0,1,0)	--	(B,1,0,1,0)	--
U=1,D=0	(B,1,0,1,0)	--	(B,1,0,1,0)	--
U=0,D=1	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)
U=1,D=1	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)

4

R	R=0,L=0	R=1,L=0	R=0,L=1	R=1,L=1
U=0,D=0	(W,0,0,0,0)	--	(W,0,0,0,0)	--
U=1,D=0	(W,0,0,0,0)	--	(W,0,0,0,0)	--
U=0,D=1	--	--	--	--
U=1,D=1	--	--	--	--

5

BR	R=0,L=0	R=1,L=0	R=0,L=1	R=1,L=1
U=0,D=0	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)
U=1,D=0	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)	(B,1,0,1,0)
U=0,D=1	--	--	--	--
U=1,D=1	--	--	--	--

6

W/B-B	R=0,L=0	R=1,L=0	R=0,L=1	R=1,L=1
U=0,D=0	(B,1,0,1,0)	--	(B,1,0,1,0)	--
U=1,D=0	--	--	--	--
U=0,D=1	(B,1,0,1,0)	(W,0,0,0,0)	(B,1,0,1,0)	(W,0,0,0,0)
U=1,D=1	--	--	--	--

and south-to-north vertical one-way communication links are utilized in the A-phase operation, however, in the B-phase operation, only an east-to-west horizontal link is used in each cell. Thus we have:

Theorem 12 (Umeo [32]) *For any binary image of size $m \times n$, the Beyer's connectivity-preserving transformation can be performed on a 2-D $CA_{1\text{-bit}}$ in $2(m+n)-1$ steps.*

We have implemented the 1-bit shrinking transformation algorithm on a 2-D $CA_{1\text{-bit}}$ with 6 internal states. Table 6.7 gives the transition rule set for the shrinking operation and Fig. 6.16 shows some snapshots of the shrinking process on a binary image of size 14×14.

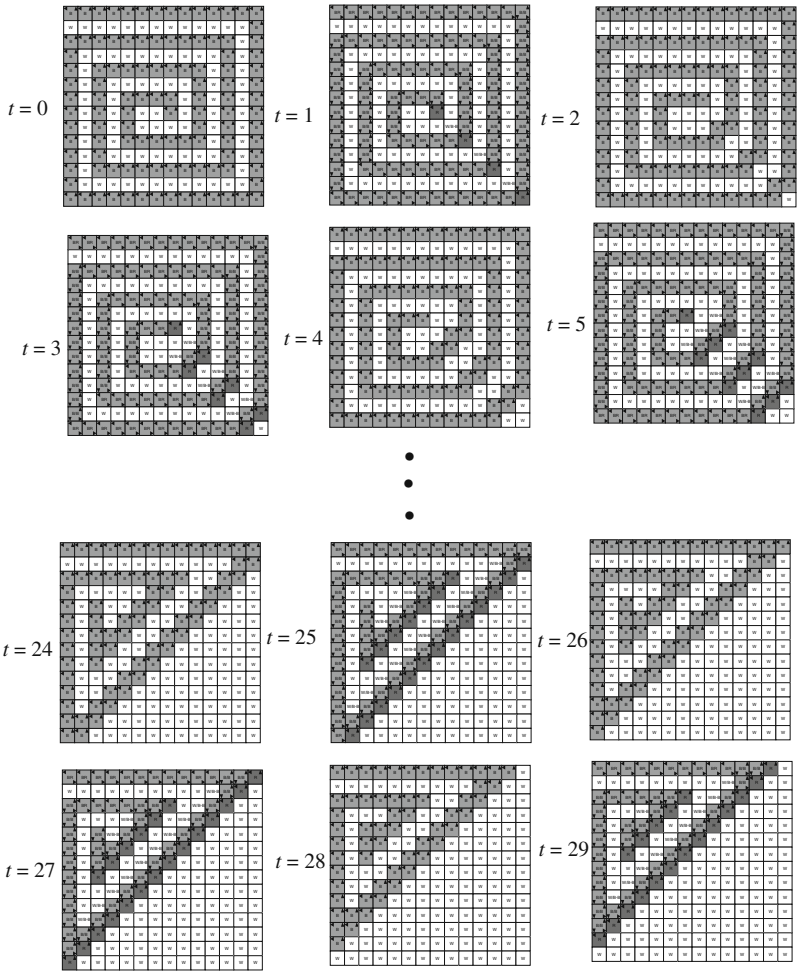

Fig. 6.16 Snapshots for connectivity-preserving shrinking transformation on 1-bit communication model

It is shown that the connectivity of any binary images can be also detected in linear-time by a 2-D $CA_{1\text{-bit}}$. The algorithm is based on our previous 1-bit implementation of the shrinking transformation. The detection of the connectivity of binary images can be done by counting up the number of vanished isolated black pixels by the accept cell located in the north-west corner of the array. The array accepts the input if and only if the count is exactly one. Every cell works not only for the transformation of images into isolated black pixels but also for the transmission of the vanished isolated black pixels toward the accept cell. Both of the operations can be simultaneously implemented on a $CA_{1\text{-bit}}$. See Umeo [32] for details.

Thus it has been shown that the $CA_{1\text{-bit}}$ can recognize the connectivity of any binary images of size $m \times n$ in $2(m + n) + O(1)$ steps. We have implemented our algorithm on a computer program, which simulates a $CA_{1\text{-bit}}$ with 61 states, recognizing 2-D connectivity. For typical binary images of size from 4×4 to 45×47, the program recognizes them correctly. Thus we have:

Theorem 13 (Umeo [32]) *There exists a 2-D $CA_{1\text{-bit}}$ that can recognize a set of 2-D 4-connected binary images of size $m \times n$ in $2(m + n) + O(1)$ steps.*

6.8 Summary and Further Works

A 1-bit inter-cell communication cellular automaton model ($CA_{1\text{-bit}}$) studied in this paper is a subclass of cellular automata (CA) whose inter-cell communication at one step is restricted to 1-bit. We have investigated a problem solving on the $CA_{1\text{-bit}}$. The problems treated are a firing squad synchronization problem, an integer sequence generation problem, a connectivity recognition problem for two-dimensional binary images, an early bird problem, and a connectivity recognition problem for two-dimensional binary images, all of which are known as the classical, fundamental problems in cellular automata. We presented several state-efficient implementations on the 1-bit inter-cell communication cellular automata for those classical cellular automata problems. Those implementations presented are not optimum ones in the number of states required. The class of $CA_{1\text{-bit}}$ is confirmed to be an interesting computational subclass of CAs that merits further study.

Acknowledgements A part of this work has been supported by the Kayamori Foundation of Informational Science Advancement. The author would like to express his thanks to N. Kamikawa, T. Yanagihara, M. Kanazawa, K. Michisaka, and T. Fujiwara who helped to develop several implementations on $CA_{1\text{-bit}}$.

References

1. H.M. Alnuweiri, V.K. Prasanna, Fast image labeling using local operators on mesh-connected computers. IEEE Trans. PAMI. **13**(2), 202–207 (1991)
2. H.M. Alnuweiri, V.K. Prasanna, Parallel architectures and algorithms for image component labeling. IEEE Trans. PAMI. **14**(10), 1014–1034 (1992)

3. M. Arisawa, On the generation of integer series by the one-dimensional iterative arrays of finite state machines (in Japanese). The Trans. IECE. 71/8 **54-C**(8), 759–766 (1971)
4. R. Balzer, An 8-state minimal time solution to the firing squad synchronization problem. Inf. Control, **10**, 22–42 (1967)
5. W.T. Beyer, *Recognition of Topological Invariants by Iterative Arrays*. Ph.D. Thesis, (MIT, Massachusetts, 1969)
6. R.E. Cypher, J.L.C. Sanz, L. Snyder, Algorithms for image component labeling on SIMD mesh-connected computers. IEEE Trans. Comput. **39**(2), 276–281 (1990)
7. R.E. Cypher, J.L.C. Sanz, *The SIMD Models of Parallel Computation*. (Springer-Verlag, Heidelberg, 1994)
8. P.C. Fischer, Generation of primes by a one-dimensional real-time iterative array. J. ACM. **12**(3), 388–394 (1965)
9. H.D. Gerken, Über Synchronisations – Probleme bei Zellularautomaten. *Diplomarbeit* (Institut für Theoretische Informatik, Technische Universität Braunschweig, Braunschweig, 1987)
10. E. Goto, A minimal time solution of the firing squad problem. Dittoed course notes for Applied Mathematics 298, Harvard University (1982)
11. J. Gruska, S.L. Torre, M. Parente, The firing squad synchronization problem on squares, toruses and rings. Intern. J. Found. Comput. Sci. **18**(3), 637–654 (2007)
12. N. Kamikawa, H. Umeo, A generalized FSSP algorithm on one-bit communication cellular automata. (draft version) (2008)
13. H. Kleine-Büning, The early bird problem is unsolvable in a one-dimensional cellular space with 4 states. Acta Cybernetica, **6**, 23–31 (1983)
14. I. Korec, Real-time generation of primes by a one-dimensional cellular automaton with 11 states. Proc. 22nd Int. Symp. MFCS '97. **LNCS 1295**, 358–367 (1997)
15. F.T. Leighton, Introduction to parallel algorithms and architectures: arrays, trees, hypercubes. (Morgan Kaufmann, San Fransisco, CA, 1992)
16. T. Legendi, E. Katona, A 5-state solution of the early bird problem in a one-dimensional cellular space. Acta Cybernetica. **5**(2), 173–179 (1981)
17. S. Levialdi, On shrinking binary picture patterns. Commun. ACM. **15**(1), 7–10 (1972)
18. M. Manohar, H.K. Ramapriyan, Connected component labeling of binary images on a mesh connected massively parallel processor. Comput. Visi. Graph. Image Process. **45**, 133–149 (1989)
19. J. Mazoyer, A six-state minimal time solution to the firing squad synchronization problem. Theore. Comput. Sci. **50**, 183–238 (1987)
20. J. Mazoyer, V. Terrier, Signals in one-dimensional cellular automata. Theor. Comput. Sci., **217**, 53–80 (1999)
21. J. Mazoyer, On optimal solutions to the firing squad synchronization problem. *Theor. Comput. Sci.* **168**, 367–404 (1996)
22. R. Miller, Q.F. Stout, Parallel algorithms for regular architectures: meshes and pyramids. (The MIT Press, Massachusetts, 1996)
23. E.F. Moore, The firing squad synchronization problem, ed. by E.F. Moore, *Sequential Machines, Selected Papers* (Addison-Wesley, Reading MA, 1964)
24. F.R. Moore, G.G. Langdon, A generalized firing squad problem. Information and Control. **12**, 212–220 (1968)
25. J. Nishimura, T. Sogabe, H. Umeo, A design of optimum-time firing squad synchronization algorithm on 1-bit cellular automaton. *Proceedings of The 8th International Symposium on Artificial Life and Robotics*, 381–386 (2003)
26. A. Rosenfeld, Connectivity in digital pictures. J. ACM. **17**(1), 146–160 (1970)
27. P. Rosenstiehl, J.R. Fiksel, A. Holliger, Intelligent graphs: Networks of finite automata capable of solving graph problems, ed. by R.C. Reed. *Graph Theory and Computing* (Academic, New York, NY, 1973)
28. A. Settle, J. Simon, Smaller solutions for the firing squad. Theoretical Computer Science. **276**, 83–109 (2002)

29. I. Shinahr, Two- and three-dimensional firing squad synchronization problems. Inf. Control. **24**, 163–180 (1974)
30. H. Szwerinski, 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)
31. S.L. Torre, M. Napoli, M. Parente, Firing squad synchronization problem on bidimensional cellular automata with communication constraints. Proc. MCU 2001. LNCS **2055**, 264–275 (2001)
32. H. Umeo, Linear-time recognition of connectivity of binary images on 1-bit inter-cell communication cellular automaton. Parallel Comput. **27**, 587–599 (2001)
33. H. Umeo, Firing squad synchronization problem in cellular automata. ed. by R.A. Meyers, *Encyclopedia of Complexity and Systems Science* (Springer, Heidelberg, 2009)
34. H. Umeo, M. Hisaoka, K. Michisaka, N. Nishioka, M. Maeda, Some new generalized synchronization algorithms and their implementations for large scale cellular automata. LNCS **2509**, 276–286 (2002)
35. H. Umeo, M. Hisaoka, T. Sogabe, A survey on optimum-time firing squad synchronization algorithms for one-dimensional cellular automata. Int. J. Unconventional Comput. **1**, 403–426 (2005)
36. H. Umeo, N. Kamikawa, A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. Fundamenta Inf. **52**, 255–275 (2002)
37. H. Umeo, N. Kamikawa, Real-time generation of primes by a 1-bit-communication cellular automaton. Fundamenta Inf. **58**(3, 4), 421–435 (2003)
38. H. Umeo, M. Maeda, M. Hisaoka, M. Teraoka, A state-efficient mapping scheme for designing two-dimensional firing squad synchronization algorithms. Fundamenta Inf. **74**(4), 603–623 (2006)
39. H. Umeo, G. Mauri, A duality in two topology-preserving parallel shrinking algorithms - Between Beyer's and Levialdi's algorithms –. Future Generation Comput. Syst. **18** 931–937 (2001)
40. H. Umeo, K. Michisaka, N. Kamikawa, M. Kanazawa, State-efficient one-bit communication solutions for some classical cellular automata problems. Fundamenta Inf. **78**(3), 449–465 (2007)
41. H. Umeo, T. Yanagihara, State-efficient optimum-time implementations of synchronization algorithms on $CA_{1\text{-bit}}$. (draft version) (2008)
42. H. Umeo, T. Yanagihara, M. Kanazawa, State-efficient firing squad synchronization protocols for communication-restricted cellular automata. LNCS 4173, 169–181 (2006)
43. V.I. Varshavsky, V.B. Marakhovsky, V.A. Peschansky, Synchronization of interacting automata. Mathematical Systems Theory. **4**(3), 212–230 (1970)
44. R. Vollmar, On two modified problems of synchronization in cellular automata. Acta Cybernetica. **3**(4), 293–300 (1978)
45. A. Waksman, An optimum solution to the firing squad synchronization problem. Inf. Control. **9**, 66–78 (1996)