Implementations of FSSP Algorithms on Fault-Tolerant Cellular Arrays

Hiroshi Umeo(✉), Naoki Kamikawa, Masashi Maeda, and Gen Fujita

University of Osaka Electro-Communication,
Hatsu-cho, 18-8, Neyagawa-shi, Osaka 572-8530, Japan
umeo@cyt.osakac.ac.jp

Abstract. The firing squad synchronization problem (FSSP, for short) on cellular automata has been studied extensively for more than fifty years, and a rich variety of FSSP algorithms has been proposed. Here we study the classical FSSP on a model of fault-tolerant cellular automata that might have possibly some defective cells and present the first state-efficient implementations of fault-tolerant FSSP algorithms for one-dimensional (1D) and two-dimensional (2D) arrays. It is shown that, under some constraints on the distribution and length of defective cells, any 1D cellular array of length n with p defective cell segments can be synchronized in $2n - 2 + p$ steps and the algorithm is realized on a 1D cellular automaton with 164 states and 4792 transition rules. In addition, we give a smaller implementation for the 2D FSSP that can synchronize any 2D rectangular array of size $m \times n$, including $O(mn)$ rectangle-shaped isolated defective zones, exactly in $2(m + n) - 4$ steps on a cellular automaton with only 6 states and 939 transition rules.

1 Introduction

Synchronization of large-scale networks is an important and fundamental computing primitive in parallel and distributed systems. The synchronization in ultra-fine grained parallel computational model of cellular automata, known as the firing squad synchronization problem (FSSP), has been studied extensively for more than fifty years [5,8], and a rich variety of synchronization algorithms has been proposed. In the present paper, we consider the FSSP from a viewpoint of fault tolerance. Reliable and fault-tolerant computation on a large-scale cellular automaton is a key issue to be studied so far. Gács [2] constructed reliable cellular automata from unreliable ones that make errors with some constant probability. Fault tolerance in FSSP has been studied by Kutrib and Vollmar [3], Umeo [6], Yunès [9], and recently by Dimitriadis, Kutrib, and Sirakoulis [1]. One of the major open questions on fault-tolerant FSSP is: how many states would be required in their realizations on a finite state automaton? No full implementations were given in the past. In this paper, we present two state-efficient implementations of fault-tolerant FSSP algorithms for one-dimensional (1D) and two-dimensional (2D) arrays. It is shown that, under some constraints on the

G. Mauri et al. (Eds.): ACRI 2018, LNCS 11115, pp. 274–285, 2018.
https://doi.org/10.1007/978-3-319-99813-8_25

distribution and length of defective cells, any 1D cellular array of length n with p defective cell segments can be synchronized in nearly minimum $2n-2+p$ steps and the algorithm is realized on a 1D cellular automaton with 164 states and 4792 transition rules. In addition, we give a smaller implementation for the 2D FSSP that can synchronize any 2D rectangular array of size $m \times n$, including $O(mn)$ rectangle-shaped isolated defective zones, exactly in $2(m+n)-4$ steps on a cellular automaton with only 6 states and 939 transition rules.

2 Fault-Tolerant FSSP Algorithm and Its Implementation on 1D Arrays

In this section we review a nearly minimum-time fault-tolerant FSSP algorithm in Umeo [6] and present an implementation of the algorithm on a 1D cellular automaton with 164 states and 4792 transition rules.

2.1 FSSP on Cellular Automata with Defective Cells

Consider a 1D array of cells, shown in Fig. 1, some of which are defective. Each cell has its own self-diagnosis circuit that diagnoses itself before its operation. The diagnosis result is stored as a flag in the special register augmented with each cell. We assume that new defections do not occur during the operational lifetime on any cell, thus the fault-tolerance we study is a static one. A consecutive defective (intact) cells are referred to as a *defective* (*intact*) segment, respectively. Figure 1 illustrates a 1D array with three defective and four intact segments. Any defective and intact cells can detect whether its neighbor cells are defective or not.

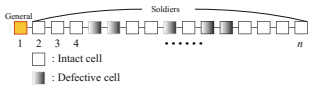

Fig. 1. A one-dimensional (1D) cellular array with three defective and four intact segments

We use the following notations. The array consists of p defective segments and $(p+1)$ intact segments, denoted by I_i and D_j, respectively and p be any positive integer, where $1 \leq p \leq n$. Let n_i and m_j be number of cells on the ith intact and jth defective segments, where $1 \leq i \leq p+1$ and $1 \leq j \leq p$. Let n be the length of the array such that $n = (n_1 + m_1) + (n_2 + m_2) +, ..., +(n_p + m_p) + n_{p+1}$.

In our model we assume that any cell in defective segment can only transmit a signal to its right or left neighbor depending on the direction in which it comes to the defective segment. The speed of the signal in any defective segment is

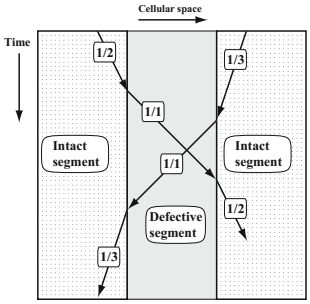

Fig. 2. In defective segments, any signal is transmitted at a constant speed 1/1

fixed to 1/1, that is, one cell per one step. In defective segments, both the information carried by the signal and the direction in which the signal is propagated are preserved without any modifications. Thus, we can see that any defective segment has two one-way pipelines that can transmit the state at 1/1 speed in either direction (Fig. 2).

The *fault-tolerant* FSSP for cellular automata with *defective* cells is to determine a description for cells that ensures all *intact* 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.

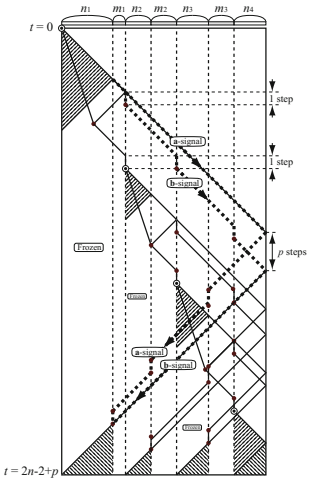

Fig. 3. A space-time diagram for the fault-tolerant FSSP algorithm operating on an array with three defective segments

2.2 Fault-Tolerant FSSP Algorithm and Its Implementation on 1D Arrays

First we introduce a *freezing-thawing* technique that yields a delayed synchronization developed in Umeo [6].

Theorem 1. Let t_1, t_2 and Δt be any integer such that $0 \leq t_1 \leq n-1$, $t_1 \leq t_2$ and $\Delta t = t_2 - t_1$. We assume that the right end cell of the array of length n receives a special signal from outside at time $t = t_1$ and t_2. Then, there exists a CA that can fire at time $t = 2n - 2 + \Delta t$.

We can freeze the entire configuration on the array during Δt steps and delay the synchronization on the array for Δt steps.

Fault-Tolerant FSSP Algorithm
Let p be any positive integer and M be any cellular array of length n with p defective segments, where $n_i \geq m_i$ and $n_i + m_i \geq p - i$, for any i such that $1 \leq i \leq p$. A space-time diagram of the fault-tolerant FSSP algorithm is illustrated in Fig. 3. The algorithm is based on the freezing-thawing technique in Theorem 1. In order to thaw the intact segment, special thawing signals: a- and b-signals, are used and the initiation of synchronization process is delayed for one step at each intact segment. Precisely, the synchronization for the i-th segment is initiated at time $t_i = 2\sum_{j=1}^{i-1}(n_j + m_j) + (i-1)$. Whenever the fast signal arrives at each right end of intact segment, it splits into two signals. One is the freezing signal and the other is the a- and b-signals which propagate toward the right end of the array at 1/1-speed. The b-signal stays for one step at the left end of each intact segment that it encounters. Both a- and b-signals reflect at the right end of the array and proceed to the left direction at 1/1-speed. This time the reflected a-signal stops for one step at the left end of each defective segment that it encounters. When the conditions given above are satisfied, two reflected a- and b-signals meet at the right end of right intact segment just where the original a- and b-signals have been generated. Now the thawing operation for the configuration of the intact segment is started.

Let t_i^a and t_i^b be time steps at which the a- and b-signals emitted by the i-th segment hit the right end of the array, respectively. We have:

$$t_i^a = t_i + \sum_{j=i}^{p}(n_j + m_j) + n_{p+1}, t_i^b = t_i^a + p - i + 1.$$

The freezing and thawing operations for I_i are started, respectively, at time $t_{i_1} = t_i + n_i - 1$ and $t_{i_2} = t_{i_1} + 2m_i + 2\sum_{j=i+1}^{p}(n_j + m_j) + 2n_{p+1} + p - i + 1$. The condition: $t_{i+1}^a \geq t_i^b$ for any i such that $1 \leq i \leq p$ is necessary and sufficient for the configuration on I_i to be thawed by the thawing signal emitted by the i-th segment. The condition is satisfied for any i such that $1 \leq i \leq p$, since $t_{i+1}^a - t_i^b = n_i + m_i - p + i \geq 0$.

Thus the configuration on I_i is frozen during $\Delta t = t_{i_2} - t_{i_1} = 2m_i + 2\sum_{j=i+1}^{p}(n_j + m_j) + 2n_{p+1} + p - i + 1$ steps. Based on Theorem 1, the i-th intact segment I_i can be fired at time $t = t_i + 2n_i - 2 + \Delta t = 2n - 2 + p$. In

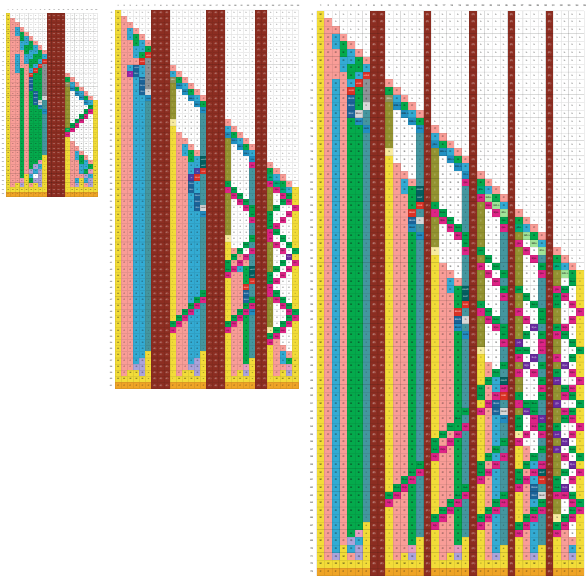

Fig. 4. Snapshots of the synchronization processes operating on a 1D array of length $n = 20$ with one defective segment (left), an array of length $n = 30$ with 3 defective segments (middle), and an array of length $n = 35$ with 5 defective segments (right), respectively

this way, the entire intact segments can be synchronized at time $t = 2n - 2 + p$. From the assumptions $n_i + m_i \geq p - i$, for any i, $1 \leq i \leq p$, it is seen that $p = O(\sqrt{n})$. Thus the time complexity of the algorithm is $2n + O(\sqrt{n})$. The algorithm is stated as follows.

Theorem 2. Let p be any positive integer and M be any cellular array of length n with p defective segments, where $n_i \geq m_i$ and $n_i + m_i \geq p - i$, for any i such that $1 \leq i \leq p$. Then, M can be synchronized in $2n - 2 + p$ steps.

We have implemented the algorithm on a 1D cellular automaton with 164 states and 4792 transition rules. In Fig. 4 we give several snapshots of the synchronization processes operating on a 1D array of length $n = 20$ with one defective segment (left), an array of length $n = 30$ with 3 defective segments (middle), and an array of length $n = 35$ with 5 defective segments (right), respectively.

3 Fault-Tolerant FSSP Algorithm and Its Implementation on 2D Arrays

A fault-tolerant FSSP on 2D arrays has never been discussed nor studied due to the difficulties in designing synchronization algorithms. Here we present a

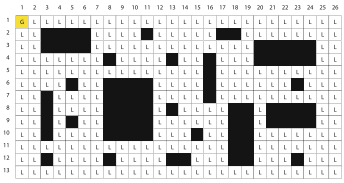

Fig. 5. A 2D rectangular array of size 13×26 with 20 isolated defective zones

6-state fault-tolerant FSSP algorithm on 2D arrays. The fault-tolerant model that we consider is slightly different from the 1D one in Sect. 2. Now we consider a 2D rectangular array of size $m \times n, m, n \geq 2$. Each cell is an identical (except the border and defective cells) finite-state automaton. The cell on the ith row, jth column is denoted by $\mathrm{C}_{i,j}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$. The array operates in lock-step mode in such a way that the next state of each cell (except border and defective cells) is determined by both its own present state and the present states of its north, south, east and west neighbors, thus assuming the von Neumann neighborhood. All cells (*soldiers*), except the general at the north-west corner and defective cells, 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 general on $\mathrm{C}_{1,1}$ is in the *fire-when-ready* state, which is the initiation signal for the array. The 2D rectangular array includes some defective regions, each consisting of defective cells that cannot transmit any information nor change their states. The defective regions can be regarded as obstacles or holes that cannot process any information in the array. We assume that no new defective cells appear after the initiation.

The fault-tolerant FSSP is to determine a description (state set and next-state function) for the intact cells that ensures all intact cells enter the *fire* state at exactly the same time and for the first time. The set of states and its transition function must be independent of m and n. A typical 2D rectangular array of size 13×26 with 20 isolated holes (obstacles) is shown in Fig. 5. Each defective region may be a rectangle, but must be isolated from each other and from the boundary of a given array. The readers can see that the initial general in yellow is on $\mathrm{C}_{1,1}$, intact cells in white take the quiescent state L, and defective cells are illustrated as black cells in Fig. 5, respectively.

The fault-tolerant FSSP algorithm is based on a mapping developed in Umeo, Maeda, Hisaoka, and Teraoka [7], where any 1D FSSP algorithm can be embedded onto 2D arrays without introducing additional states. We consider a 2D array of size $m \times n$, where $m, n \geq 2$, shown in Fig. 6. The array is decomposed into $m + n - 1$ groups g_k, $1 \leq k \leq m + n - 1$, defined as follows.

$$g_k = \{\mathrm{C}_{i,j} | i + j = k + 1\}, \text{i.e.,}$$

$g_1 = \{\mathrm{C}_{1,1}\}$, $g_2 = \{\mathrm{C}_{1,2}, \mathrm{C}_{2,1}\}$, $g_3 = \{\mathrm{C}_{1,3}, \mathrm{C}_{2,2}, \mathrm{C}_{3,1}\}$, $\ldots$, $g_{m+n-1} = \{\mathrm{C}_{m,n}\}$. Figure 6 shows the decomposition of the 2D array of size $m \times n$ into $m + n - 1$ groups.

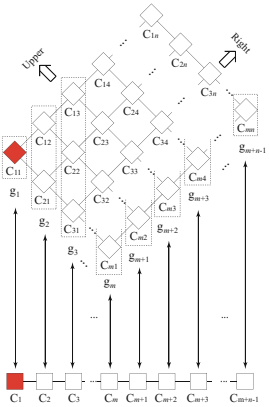

Fig. 6. Correspondence between 1D and 2D arrays

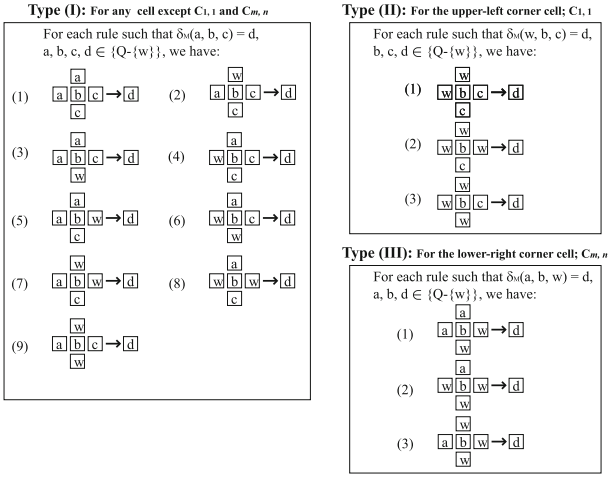

Fig. 7. Construction of transition rules for 2D fault-tolerant FSSP algorithm

Let $M = (Q, \delta_M, w)$ be any 1D array that fires ℓ cells in $T(\ell)$ steps, where Q is the finite state set of M, $\delta_M : Q^3 \to Q$ is the transition function, and $w \in Q$ is the state of the right and left ends. We assume that M has $m+n-1$ cells, denoted by C_i, $1 \leq i \leq m+n-1$. For convenience, we assume that M has a left and right end cells of the array, denoted by C_0 and C_{m+n}, respectively. Both end cells C_0 and C_{m+n} always take the state $w \in Q$. We consider a one-to-one correspondence between the ith group g_i and the ith cell C_i on M such that $g_i \leftrightarrow \mathrm{C}_i$, where $1 \leq i \leq m+n-1$ (see Fig. 6). We can construct a 2D array $N = (Q, \delta_N, w)$ such that each cell in g_i simulates the ith cell C_i in real-time and N can fire any 2D $m \times n$ array at time $t = T(m+n-1)$ if and only if M fires the 1D array of length $m+n-1$ at time $t = T(m+n-1)$, where $\delta_N : Q^5 \to Q$ is the

transition function, and $w \in Q$ is the border/defective state of the array. Note that the set of internal states of N is the same as M. The transition function δ_N is constructed as follows:

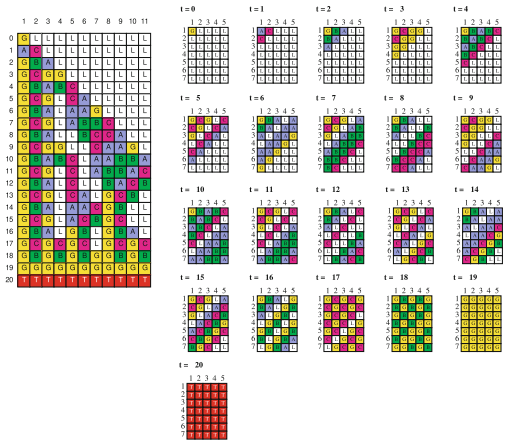

Fig. 8. Configurations of Mazoyer's 6-state FSSP algorithm on 11 cells (left) and snapshots of the synchronization processes on a 2D array of size 7×5 (right)

Let $\delta_M(a, b, c) = d$ be any transition rule of M, where $a, b, c, d \in \{Q - \{w\}\}$. Then, N has nine transition rules, as shown in Fig. 7, Type (I). The first rule (1) in Type (I) is used by an inner cell that does not include border/defective cells amongst its four neighbors. Rules (2)-(9) are used by an inner cell that has a border/defective cell as its upper, lower, left, right, lower left, and upper right neighbor, respectively. Here the terms *upper*, *right* etc. on the rectangular array are interpreted in a usual way, shown in Fig. 7, although the array is rotated by 45° in the counter-clockwise direction. When $a = w$, that is, $\delta_M(w, b, c) = d$, where $b, c, d \in \{Q - \{w\}\}$, then N has three rule, as shown in Type (II). These rules are used by the cell located in the upper left corner. When $c = w$, that is, $\delta_M(a, b, w) = d$, where $a, b, d \in \{Q - \{w\}\}$, then N has three rules, as shown in Type (III). These rules are used by the cell located in the lower right corner.

Now let M have $m + n - 1$ cells. We can show that the constructed 2D array N can generate the configuration of M in real-time. Specifically, for any i, $1 \leq i \leq m + n - 1$, the state of any cell in g_i at any step is the same and is identical to the state of C_i at the corresponding step. Let S_i^t, $S_{i,j}^t$ and $S_{g_i}^t$ denote the state of C_i, $C_{i,j}$ and the set of states of the cells in g_i at step t, respectively.

First we consider the case where a given 2D array of size $m \times n$ includes no defective zones. The following lemma holds.

Lemma 3. *Let i and t be any integer such that $1 \leq i \leq m + n - 1$, $0 \leq t \leq T(m + n - 1)$. Then, $S_{g_i}^t = \{S_i^t\}$.*

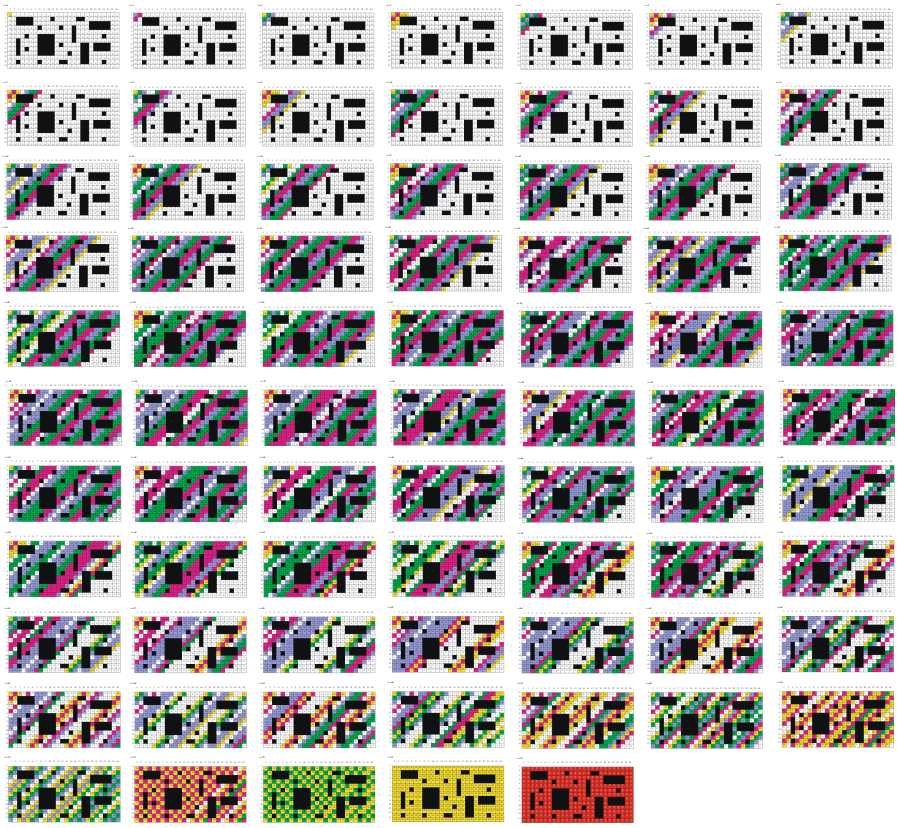

Fig. 9. Snapshots of the synchronization processes on a 2D array of size 13×26, containing 20 defective rectangle zones

We see that any configuration on a 1D array consisting of $m+n-1$ cells can be mapped onto a 2D array of size $m \times n$. Therefore, if the embedded 1D array fires $m+n-1$ cells in $T(m+n-1)$ steps, then the corresponding 2D array of size $m \times n$ can be synchronized in $T(m+n-1)$ steps. Thus, we can embed any 1D FSSP algorithm onto a 2D array without increasing the number of internal states. We complete the observation in the next theorem.

Theorem 4. Let M be any s-state FSSP algorithm operating in $T(\ell)$ steps on 1D arrays of length ℓ. Then, there exists a 2D s-state cellular automaton that can synchronize any rectangular array of size $m \times n$ in $T(m+n-1)$ steps.

Here we can embed a 1D 6-state minimum-time FSSP algorithm developed in Mazoyer [4], synchronizing ℓ cells in $2\ell-2$ steps. The next rectangle synchronization algorithm fires any $m \times n$ array in $2(m+n)-4$ steps, since $T(m+n-1) = 2(m+n-1)-2 = 2(m+n)-4$.

Fig. 10. Snapshots of the synchronization processes on a 2D array of size 10×10, including 4 defective zones

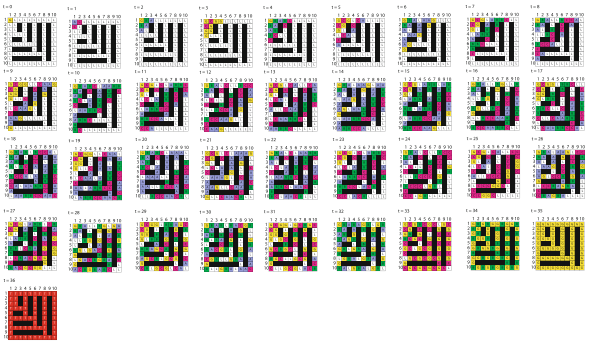

Fig. 11. Snapshots of the synchronization processes on a 2D array of size 10×10, including 7 defective rectangle zones

Theorem 5. There exists a 6-state 939-rule FSSP algorithm that can synchronize any $m \times n$ rectangular array in $2(m+n)-4$ steps.

Figure 8 (left) illustrates snapshots of Mazoyer's 6-state FSSP algorithm on 11 cells. These configurations are mapped on a 2D array of size 7×5, shown in Fig. 8 (right).

We now consider a class of 2D arrays $\mathcal{A}$ of size $m \times n$, initially including intact and defective cells, which satisfies the following conditions:

1. The initial general is on the north-west corner cell $C_{1,1}$.
2. Any intact cell, except $C_{1,1}$, takes a quiescent state initially.
3. Any intact cell $C_{i,j}, 1 \leq i \leq m, 1 \leq j \leq n$, except $C_{1,1}$ and $C_{m,n}$, must have at least one intact cell in $\{C_{i-1,j}, C_{i,j-1}\}$ and one intact cell in $\{C_{i+1,j}, C_{i,j+1}\}$ at time $t = 0$.
4. The defective cell, assuming the boundary state initially, keeps the state during operations.

Fig. 12. Snapshots of the synchronization processes on a 2D array of size 10×10, including 4 defective zones

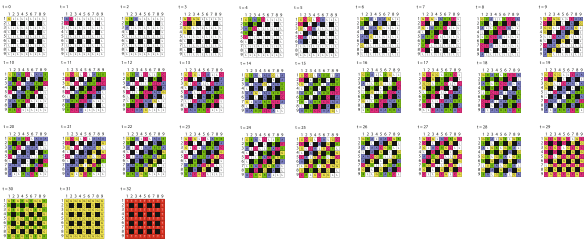

Fig. 13. Snapshots of the synchronization processes on a 2D array of size 9×9, including 16 defective cells

In the case where a given 2D array includes some defective zones satisfying the conditions above, the following lemma holds.

Lemma 6. *Let i and t be any integer such that $1 \leq i \leq m+n-1$, $0 \leq t \leq T(m+n-1)$. For any initial configuration in $\mathcal{A}$, we have:*

$$S_{g_i}^t = \begin{cases} \{S_i^t, w\}, & \text{if } g_i \text{ includes some defective cells,} \\ \{S_i^t\}, & \text{otherwise.} \end{cases} \tag{1}$$

The 6-state 2D FSSP algorithm stated in Theorem 5 can also synchronize any 2D array in $\mathcal{A}$. We have:

Theorem 7. There exists a 6-state 939-rule fault-tolerant FSSP algorithm that can synchronize any $m \times n$ rectangular array in $\mathcal{A}$ in $2(m+n)-4$ steps.

Several snapshots of the 6-state fault-tolerant FSSP algorithm running on a rectangular array of size 13×26 including 20 holes (Fig. 5) are shown in Fig. 9. Figures 10, 11, 12 and 13 illustrate similar snapshots for some different initial configurations.

4 Conclusions

It has been shown that, under some constraints on the distribution of defective cells, any 1D cellular array of length n with p defective cell segments can be synchronized in $2n-2+p$ steps and the algorithm has been realized on a finite state automaton having 164 states and 4792 rules. We have also given a smaller implementation for the 2D FSSP that can synchronize any 2D rectangular array of size $m \times n$, including $O(mn)$ rectangle-shaped isolated defective zones, exactly in $2(m+n)-4$ steps on a cellular automaton with only 6 states and 939 transition rules.

References

1. Dimitriadis, A., Kutrib, M., Sirakoulis, G.C.: Cutting the firing squad synchronization. In: El Yacoubi, S., Wąs, J., Bandini, S. (eds.) ACRI 2016. LNCS, vol. 9863, pp. 123–133. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44365-2_12
2. Gács, P.: Reliable computation with cellular automata. J. Comput. System Sci. **32**, 15–78 (1986)
3. Kutrib, M., Vollmar, R.: The firing squad synchronization problem in defective cellular automata. IEICE Trans. Inf. Syst. **E78–D**(7), 895–900 (1995)
4. Mazoyer, J.: A six-state minimal time solution to the firing squad synchronization problem. Theor. Comput. Sci. **50**, 183–238 (1987)
5. Moore, E.F.: The firing squad synchronization problem. In: Moore, E.F. (ed.) Sequential Machines, Selected Papers. Addison-Wesley, Reading, pp. 213–214 (1964)
6. Umeo, H.: A simple design of time-efficient firing squad synchronization algorithms with fault-tolerance. IEICE Trans. Inf. Syst. **E87–D**, 733–739 (2004)
7. Umeo, H., Maeda, M., Hisaoka, M., Teraoka, M.: A state-efficient mapping scheme for designing two-dimensional firing squad synchronization algorithms. Fundam. Inform. **74**, 603–623 (2006)
8. Umeo, H.: Firing squad synchronization problem in cellular automata. In: Meyers, R. (ed.) Encyclopedia of Complexity and System Science, vol. 4, pp. 3537–3574. Springer, New York (2009). https://doi.org/10.1007/978-0-387-30440-3
9. Yunès, J.-B.: Fault tolerant solutions to the firing squad synchronization problem in linear cellular automata. J. Cell. Autom. **1**(3), 253–268 (2006)