Synthesis of Scalable Single Length Cycle, Single Attractor Cellular Automata in Linear Time

Bidesh Chakraborty
Department of Computer Science and Engineering
Haldia Institute of Technology, Haldia
Purba Medinipur, West Bengal 721657, India

Mamata Dalui
Department of Computer Science and Engineering
National Institute of Technology, Durgapur
Paschim Burdwan, West Bengal 713209, India

Biplab K. Sikdar
Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology, Shibpur
Howrah, West Bengal 711103, India

This paper proposes the synthesis of single length cycle, single attractor cellular automata (SACAs) for arbitrary length. The n-cell single length cycle, single attractor cellular automaton (SACA), synthesized in linear time $O(n)$, generates a pattern and finally settles to a point state called the single length cycle attractor state. An analytical framework is developed around the graph-based tool referred to as the next state transition diagram to explore the properties of SACA rules for three-neighborhood, one-dimensional cellular automata. This enables synthesis of an $(n + 1)$-cell SACA from the available configuration of an n-cell SACA in constant time and an $(n + m)$-cell SACA from the available configuration of n-cell and m-cell SACAs also in constant time.

Keywords: cellular automata; single length cycle, single attractor cellular automata; SACA; scalable test structure; NSRTD; linear time

1. Introduction

The cellular automaton (CA), an autonomous finite-state machine, consists of a number of cells organized in the form of a lattice. Over the years, it has proven its efficiency as a powerful modeling tool [1–6]. However, complete characterization of such a machine is still an open research area. In [7] Martinez reports several classifications of elementary cellular automata (ECAs), including Wolfram's classification, Wuensche's classification, and Li and Packard's classification, and proposes a new classification based on memory functions. The asymptotic behavior of one-dimensional cellular automata (CAs) is

https://doi.org/10.25088/ComplexSystems.30.3.415

explored in [8], where the complex behavior of CAs with an increasing number of symbols and neighborhood size has been illustrated. During characterization of the three-neighborhood, one-dimensional CA state space, the researchers identified a set of CA states called attractors, toward which neighboring states asymptotically approach in the course of their dynamic evolution [9]. The attractors of linear/additive CAs are identified and explored in [10–12].

A graph-based solution for null-boundary invertible CAs was proposed in [13] and that for periodic boundary CAs in [14]. However, to model a wide variety of physical systems that are nonhomogeneous in nature, a nonhomogeneous CA structure (also called a hybrid CA) is evolved as an alternative to the uniform structure [15, 10]. Characterizations of single length cycle, single attractor cellular automata (SACAs) over hybrid structures, especially beyond the linear domain, are yet to be explored. Such a hybrid CA (i.e., SACA) is of prime interest in devising schemes for different applications, especially in authentication, cache coherency verification and cryptography [10, 16–18].

A number of papers have reported on how to characterize the properties of SACAs based on the attractors. However, such schemes mostly rely on graphical analysis [19]. This paper provides complete characterization of SACA rules, employed for the design, to get better insight into their cascadable structure leading to the desired scalable design. With this target, we classify 256 three-neighborhood CA rules into nine groups and consider the relationship between two consecutive cell rules to enable analysis/synthesis of SACAs of any arbitrary length in linear time.

The precise contribution of this research is summarized below.

- A scheme is proposed for the synthesis of an n-cell scalable SACA structure in $O(n)$ time. The graph-based tool, referred to as the next state RMT transition diagram (NSRTD), is effectively employed for the synthesis of SACAs.
- A design methodology is proposed for constructing an $(n+1)$-cell SACA from an available n-cell SACA in constant time.
- A constant time solution is reported for constructing an $(n+m)$-cell SACA from the available n-cell and m-cell SACAs.
- A hardware structure is proposed to realize the synthesis of $(n+1)$-cell and $(n+m)$-cell SACAs from the n-cell and m-cell SACAs.

The next section provides CA preliminaries relevant for current work.

2. Cellular Automaton Preliminaries

A CA can be viewed as an autonomous finite-state machine. A CA cell has two states: 0 or 1, and the next state of the i^{th} CA cell is

$S_i^{t+1} = f_i\left(S_{i-1}^t, S_i^t, S_{i+1}^t\right)$, where S_{i-1}^t, S_i^t and S_{i+1}^t are the present states of the left neighbor, self and right neighbor of the i^{th} cell at time t. f_i is the next state function. The $S^t = \left(S_1^t, S_2^t, \ldots, S_n^t\right)$ at t is the present state of the CA.

The f_i can be expressed in the form of a truth table (Table 1). The decimal equivalent of the eight outputs is called "rule" R_i. In a two-state three-neighborhood CA, there can be 2^8 (256) rules. Three such rules, 80, 176 and 194, are illustrated in Table 1. The first row lists the possible 2^3 (8) combinations of the present states of cells $i-1$, i and $i+1$ at time t. The last three rows indicate the next states of the i^{th} cell at time $t+1$, forming the rules 80 ($\text{NS} = S_{i-1} \cdot S'_{i+1}$), 176 ($\text{NS} = S_{i-1} \cdot S'_i + S_{i-1} \cdot S_{i+1}$) and 194 ($\text{NS} = S'_{i-1} \cdot S'_i \cdot S_{i+1} + S_{i-1} S_i$).

Definitions 1–4 are relevant to make understanding the CA theory included in this paper easier.

Definition 1. A combination of present states $\left(S_{i-1}^t, S_i^t, S_{i+1}^t\right)$, shown in the first row of Table 1, is referred to as the rule min term (RMT).

For example, column 100 of Table 1 is the fourth RMT. The next states corresponding to this are 0 for rule 194 and 1 for rules 80 and 176. The RMT 0 (000) is 0 in rules 80, 176 and 192 (Table 1). An RMT is represented as T(m), $m = 0, 1, 2, 3, 4, 5, 6, 7$, that is, $\text{T}(m) = \{\text{T}\}$. The RMT for CA cell i at time t is denoted as $\text{T}_i^t(m)$. For example, $\text{T}_i^t(0)$ denotes cell 1 RMT $\text{T}(0)$ at time t. However, in the diagrams in this paper, an RMT is represented only by the corresponding decimal number m.

Parent States	111	110	101	100	011	010	001	000	Rule
RMT	(7)	(6)	(5)	(4)	(3)	(2)	(1)	(0)	
next state	0	1	0	1	0	0	0	0	80
next state	1	0	1	1	0	0	0	0	176
next state	1	1	0	0	0	0	1	0	194

Table 1. RMT of the CA rules.

Definition 2. An RMT x0y (x1y) in a CA rule is called passive (self-replicating) if the RMT x0y (x1y) is 0 (1). On the other hand, if an RMT x0y (x1y) is 1 (0), it is active (non-self-replicating). The RMT $\text{T}(0)$ (000) is 0 and RMT $\text{T}(6)$ (110) is 1 in rule 80 (Table 1). These two RMTs are passive. On the other hand, RMT $\text{T}(4)$ (100) in rule 80 is active, as it is 1 (Table 1).

Definition 3. An RMT string (RS) is defined as a sequence of consecutive RMTs that appear in a state of the CA. It is represented by the

sequence of RMTs $\langle T_1, T_2, \ldots, T_n \rangle$, where $T_i \in \{T\} = \{T(0), T(1), T(2), T(3), T(4), T(5), T(6), T(7)\}$. For example, the state 1101 of a four-cell null-boundary CA can be represented by the RS $\langle T_1, T_2, T_3, T_4 \rangle = \langle T(3), T(6), T(5), T(2) \rangle$.

Definition 4. A rule is balanced if it has an equal number of zeros and ones out of its eight RMTs; otherwise, the rule is unbalanced.

All the rules in Table 1 are unbalanced, whereas rule 170 (10101010) with an equal number of zeros and ones is a balanced rule.

The set $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_n \rangle$ of rules configures the cells of an n-cell CA. If all the $\mathcal{R}_i$ are the same, the CA is a uniform CA; otherwise, it is a nonuniform/hybrid CA. For the current paper, we consider null boundary CAs as in Figure 1, where the left neighbor of the leftmost cell as well as the right neighbor of the rightmost cell is 0.

A CA is reversible if its states form only cycles in the state transition diagram; otherwise, the CA is irreversible (Figure 2(a)). A set of states of a CA can form a loop (cycle) ($0 \to 0$, $8 \to 8$ and $2 \to 5 \to 2$ of Figure 2(a)). Such a loop is called an *attractor*. An attractor forms a basin with the states that lead to the attractor (0-basin of Figure 2(a) contains eight states, including the attractor state 0). The cycles $0 \to 0$ and $8 \to 8$ are single length cycle attractors, called the point state, which is of our current interest.

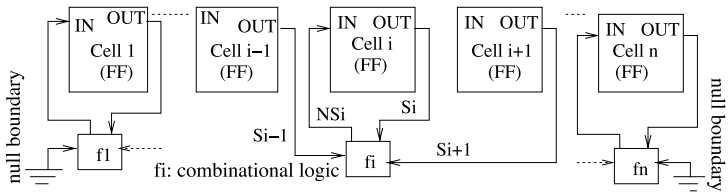

Figure 1. An n-cell null boundary CA.

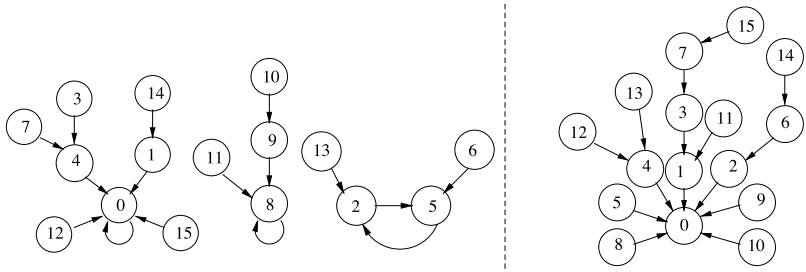

Figure 2. State transition diagrams of irreversible CAs.

CAs of Figure 2(a) have a single length cycle attractor (state 0/state 8) and can be referred to as a single length cycle attractor CA (SLCA). However, an SLCA producing a single graph and having only one single length cycle attractor in its state transitions is called a single length cycle, single attractor CA (SACA). An n-cell uniform CA with rule 192 forms a single graph with a single length cycle attractor in its state transition for all n, and therefore, it is an SACA (Figure 2(b)) and 192 is an SACA rule.

The class of SACA rules is widely used to devise test structures for on-chip testing [20–24]. In three-neighborhood null-boundary rules, 41 out of 256 CA rules are SACA rules [25] (Table 2). Hybridization of these 41 SACA rules can also generate SACAs. However, the time complexity to synthesize hybrid n-length SACAs is $O(41^n)$; that is, an NP-complete problem.

In the following sections, we report on the synthesis of a scalable SACA structure involving all 41 three-neighborhood SACA rules. The synthesis of an SACA is the outcome of an NSRT diagram of the CA, introduced in the next section.

Group	Rules for SACA
2	34, 48
3	2, 16, 32, 42, 56, 98, 112, 162, 176
4	0, 10, 15, 24, 40, 66, 80, 85, 96, 130, 144, 160, 170, 184, 226, 240, 255
5	8, 64, 128, 138, 143, 152, 168, 194, 208, 213, 224
6	136, 192

Table 2. CA rules for uniform SACAs.

3. Next State Rule Min Term Transition Diagram

The graph-based tool, referred to as the next state RMT transition diagram (NSRTD), is introduced to characterize the SLCA state space. It provides a methodology to determine the presence of single length as well as multilength cycle attractors in a nonuniform CA state transition diagram. The following definitions are essential for describing the NSRTD [19].

Definition 5. If RMT T_i^t is the i^{th}-cell RMT based on which cell i changes its state at time step t, then the next cell RMTs (NCRs) of T_i^t are the RMTs $T_{i+1}^t = \{(2 \times T_i \bmod 8), ((2 \times T_i + 1) \bmod 8)\}$ of the $(i+1)^{\text{th}}$-cell rule, based on which the $(i+1)^{\text{th}}$ cell can change its state at the t^{th} time step. If the i^{th}-cell RMT of a CA is $T(1)$, then its NCRs

https://doi.org/10.25088/ComplexSystems.30.3.415

are $T(2)$ $(2 \times T(1) \bmod 8)$ and $T(3)$ $((2 \times T(1) + 1) \bmod 8)$. The possible NCRs of the eight RMTs are shown in Table 3.

RMT T_i of the i^{th} Cell Rule	RMTs T_{i+1} of the $(i+1)^{\text{th}}$ Cell Rule
T(0)/T(4)	T(0), T(1)
T(1)/T(5)	T(2), T(3)
T(2)/T(6)	T(4), T(5)
T(3)/T(7)	T(6), T(7)

Table 3. Relationship between T_i and T_{i+1} (next cell RMT).

Definition 6. If T_i^t is the i^{th}-cell RMT corresponding to the present state S^t of a CA at time t, then the next state RMT (NSR) is the i^{th}-cell RMT (T_i^{t+1}) corresponding to the next state S^{t+1} of the CA at time $t+1$. The sequence of NSRs based on which the i^{th} CA cell changes its state during a CA run is called the next state RMT sequence (NSRS). That is, an NSRS for cell i (denoted as NSRS_i) is $T_i^0, T_i^1, \ldots, T_i^t, T_i^{t+1}, \ldots$, where T_i^{t+1} is the NSR of T_i^t for cell i. For the i^{th}-cell rule R_i of a CA, then all possible NSRSs can be represented by a directed graph G(V, E), called an NSRS graph, where all the $T_i^t \in V$ and $e(T_i^t, T_i^{t+1}) \in E$, where T_i^{t+1} is the NSR of T_i^t.

Definition 7. Two NSRSs defined for CA cells i and $i+1$ (NSRS_i and NSRS_{i+1}) are called compatible if the j^{th} NSRs of the two are related in the sense that the j^{th} NSR of NSRS_{i+1} is the NCR (Table 3) of the j^{th} NSR of NSRS_i.

NSRTD: The NCR, NSR, NSRS and NSRS graph of a CA enable constructing the set of directed graphs called NSRTD as a two-dimensional arrangement of nodes. Each node is an RMT and each NSRTD represents an attractor. The edge $(T_i^t, T_i^{t+1}) \in E$, such that T_i^{t+1} is the NSR of T_i^t and also the edge $(T_i^t, T_{i+1}^t) \in E$, such that T_{i+1}^t is the NCR of T_i^t. Further, the nodes $T_i^t(\forall\, t)$ form an NSRS for CA cell i.

If the i^{th}-cell RMT, based on which cell changes its state, is T_i xyz, and l and r are the present states of cell $i-2$ and cell $i+2$, respectively, then the NSR can be $d_l dd_r$, where the RMT xyz of the i^{th} CA cell rule R_i is d, d_l = RMT lxy of the $(i-1)^{\text{th}}$-cell rule R_{i-1}, and d_r = RMT yzr of the $(i+1)^{\text{th}}$ CA cell rule R_{i+1}. The derivation of NSR for the cells of a five-cell null-boundary CA $\langle 2, 194, 80, 176, 32 \rangle$ is shown in Figure 3. For the first cell, $d_l = x = 0$ and $y = z = r = 0/1$. That is, the NSR of RMT $T(0)$ $(xyz = 000)$ is

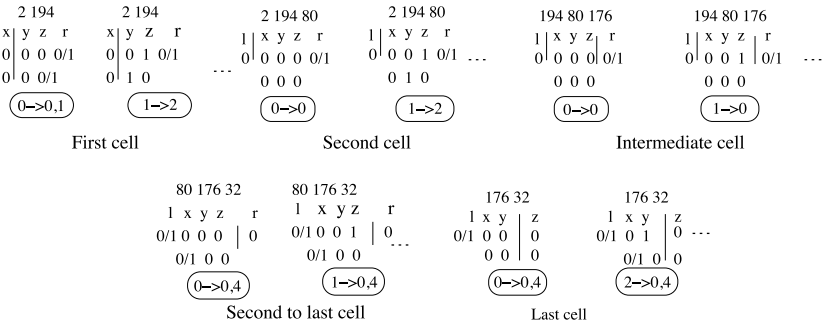

Figure 3. Finding NSRs for null-boundary CA $\langle 2, 194, 80, 176, 32\rangle$.

$d_l dd_r$ = '0'RMT xyz RMT yzr (where $d_l = 0$, d = RMT xyz, d_r = RMT yzr) = '0''RMT 000' 'RMT000' = 000 (RMT T(0) of rule 2 is 0 and RMT T(0) of rule 194 is 0)—that is, NSR of RMT T(0) is T(0) (000). Similarly, NSR for RMT T(1) is T(2). The RMTs of the first cell and corresponding NSRs for the CA $\langle 2, 194, 80, 176, 32\rangle$ are noted in columns 1 and 2 of Table 4(a), respectively. The NSRs of RMTs corresponding to the second cell and intermediate cell are noted in columns 4 and 6 of Table 4(a), respectively. The NSRs of the second to last cell and the last cell are noted in columns 2 and 4 of Table 4(b), respectively.

First Cell		Second Cell		Intermediate Cell	
RMT	NSR	RMT	NSR	RMT	NSR
(1)	(2)	(3)	(4)	(5)	(6)
T(0)	T(0), T(1)	T(0)	T(0)	T(0)	T(0)
T(1)	T(2)	T(1)	T(2)	T(1)	T(0)
T(2)	T(0)	T(2)	T(4), T(5)	T(2)	T(5), T(1)
T(3)	T(1)	T(3)	T(4), T(5)	T(3)	T(0), T(1), T(4), T(5)
		T(4)	T(0)	T(4)	T(2), T(6)
		T(5)	T(0)	T(5)	T(0), T(4)
		T(6)	T(2), T(3)	T(6)	T(3), T(7)
		T(7)	T(2), T(3)	T(7)	T(0), T(1), T(4), T(5)

(a)

Table 4. (*continues*)

Second to Last Cell		Last Cell	
RMT	NSR	RMT	NSR
(7)	(8)	(9)	(10)
T(0)	T(0), T(4)	T(0)	T(0), T(4)
T(1)	T(0), T(4)	T(2)	T(0), T(4)
T(2)	T(0)	T(4)	T(0)
T(3)	T(0)	T(6)	T(0), T(4)
T(4)	T(2), T(6)		
T(5)	T(2), T(6)		
T(6)	T(0)		
T(7)	T(2)		

(b)

Table 4. NSRs for the cells of a null-boundary nonuniform CA $\langle 2, 194, 80, 176, 32\rangle$.

Property 1. NSR of an RMT T_i of the i^{th}-cell rule R_i will be T_i itself ($T_i^t = T_i^{t+1}$) if the RMTs T_{i-1} of R_{i-1}, T_i of R_i and T_{i+1} of R_{i+1} are passive.

Property 2. In an RS, if RMTs T_i and T_{i+1} are passive $\forall\ i = 1$ to $n-1$, where T_{i+1} is the NCR of T_i, the RMTs of that RS form compatibility classes of self-loops $\forall\ i = 1, 2, \ldots, n$, which creates a path consisting of self-loops in the NSRTD.

The NSRS graphs for the CA $\langle 2, 194, 80, 176, 32\rangle$ are shown in Figure 4. For the first cell of the CA $\langle 2, 194, 80, 176, 32\rangle$, the NSRs of RMT $T(0)$ are $T(0)$ and $T(1)$, the NSR of $T(1)$ is $T(2)$ and the NSR of $T(2)$ is $T(0)$. Further, the NSR of $T(3)$ for the first cell is $T(1)$ (column 2 of Table 4). These result in the NSRS graph of Figure 4(a) for the first cell (the RMT $T(m)$ is denoted as "m"). Figures 4(b) to (e) are the NSRS graphs for the second, intermediate, second to last and the last cells, respectively.

Figure 5 represents the NSRTD of the CA $\langle 2, 194, 80, 176, 32\rangle$. It defines that if the cell_1 changes its state at time t on RMT $T(0)$, then in the next time step $t+1$ it also changes its state on $T(0)$ (Figure 4(a)). Further, while cell_1 changes its state on $T(0)$, cell_2 can change its state on $T(0)$ or $T(1)$ at time t (NCR of cell_1, Table 3). In Figure 5, the $T(0)$ is chosen because $T(0)$ has a self-loop. The NCR of cell_2 is $T(0)$ where $T(0)$ has a self-loop. For the same reason, $T(0)$ is chosen from the third, fourth and last cells. It is composed of the only path of length n as shown in Figure 5. Such an n-length path can be

found for any arbitrary value of n. Since there is no such n-length path having a multilength cycle, the CA $\langle 2, 194, 80, 176, 32\rangle$ does not have a multilength cycle. However, the presence of an n-length path with only single length cycles indicates the presence of a single length cycle attractor. All cells, cell_1 through cell_5 of Figure 5, change their state on RMT 0 $(T(0))$, thus the CA $\langle 2, 194, 80, 176, 32\rangle$ is an SACA for any length n with an all-zeros attractor.

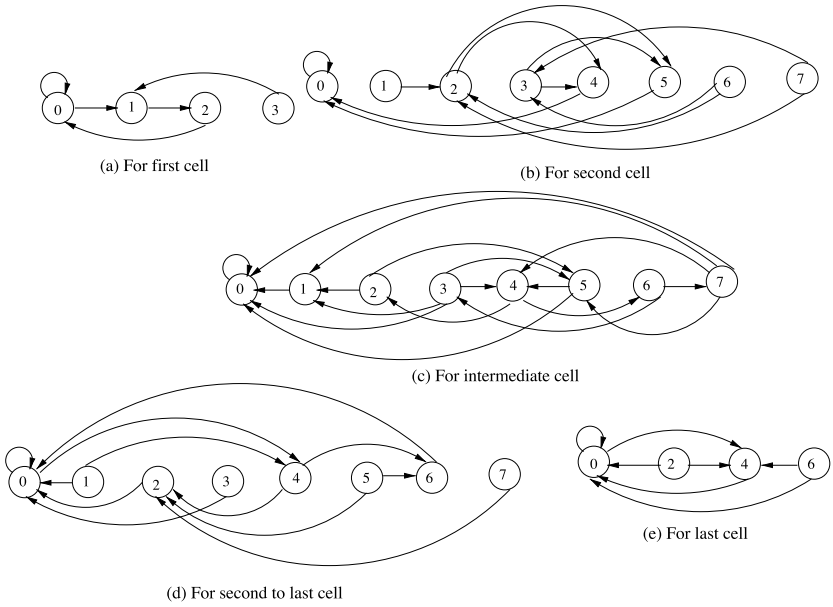

Figure 4. NSRS graphs for the CA $\langle 2, 194, 80, 176, 32\rangle$.

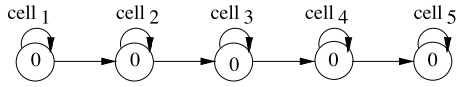

Figure 5. NSRT diagram for CA $\langle 2, 194, 80, 176, 32\rangle$.

In general, for an n-cell CA, an n-length path in the NSRTD consisting of only self-loops corresponds to a single length cycle attractor and an n-length path in an NSRTD with multilength cycles corresponds to a multilength cycle attractor. The number of such n-length paths denotes the number of attractors. The following properties of NSRS graphs and NSRTD guide the identification of SLCA rules for uniform CA as well as for nonuniform CA, as described in [19].

https://doi.org/10.25088/ComplexSystems.30.3.415

Property 3. If there is no multilength cycle in NSRS graphs of a CA, the CA cannot have a multilength cycle.

Property 4. If there is no multilength cycle in the NSRTD of a CA, the CA cannot have a multilength cycle.

Group	Rules for SACA
0	51
1	19, 35, 49, 50, 55, 59, 115, 179
2	3, 17, 18, 23, 27, 33, 34, 39, 43, 48, 53, 54, 57, 58, 63, 83, 99, 113, 114, 119, 123, 147, 163, 177, 178, 183, 187, 243
3	1, 2, 7, 11, 16, 21, 22, 25, 26, 31, 32, 37, 38, 41, 42, 47, 52, 56, 61, 62, 67, 81, 82, 87, 91, 97, 98, 103, 107, 112, 117, 118, 121, 122, 127, 131, 145, 146, 151, 155, 161, 162, 167, 171, 176, 181, 182, 185, 186, 191, 211, 227, 241, 242, 247, 251
4	0, 5, 6, 9, 10, 15, 20, 24, 29, 30, 36, 40, 45, 46, 60, 65, 66, 71, 75, 80, 85, 86, 89, 90, 95, 96, 101, 102, 105, 106, 111, 116, 120, 125, 126, 129, 130, 135, 139, 144, 149, 150, 153, 174, 158, 159, 160, 165, 166, 169, 170, 175, 180, 184, 189, 190, 195, 209, 210,215, 219, 225, 226, 231, 235, 240, 245, 246, 249, 250, 255
5	4, 8, 13, 14,28, 44, 64, 69, 70, 73, 74, 79, 84, 88, 93, 94, 100, 104, 109, 110, 124, 128, 133, 134, 137, 138, 143, 148, 152, 157, 164, 168, 173, 174, 188, 193, 194, 199, 203, 208, 213, 214, 217, 218, 223, 224, 229, 230, 233, 234, 239, 244, 248, 253, 254
6	12, 68, 72, 78, 92, 108, 132, 136, 141, 142, 156, 172, 192, 197, 198, 201, 202, 207, 212, 216, 222, 228, 232, 237, 238, 252
7	76, 140, 196, 200, 205, 206, 220, 236
8	204

Table 5. Classification of CA rules.

4. Characterization of Cellular Automata Rules for SACAs

This section reports on the characterization of rule $\mathcal{R}_i$ in respect of its contribution toward the state transition behavior of a CA ($\mathcal{R}$) that converges to a particular state (attractor). Since the next state of a single length cycle attractor is the attractor itself, there should be at least one RMT (Section 2) of each cell rule $\mathcal{R}_i$ of $\mathcal{R}$ (CA) for which the cell i does not change its state in the next time step. In rule $\mathcal{R}_i$, if the RMT $\mathrm{T}(0)$, $\mathrm{T}(1)$, $\mathrm{T}(4)$ or T(5) is 0, then the CA cell i, configured with $\mathcal{R}_i$, does not change its state. Similarly, if the RMT $\mathrm{T}(2)$, $\mathrm{T}(3)$, $\mathrm{T}(6)$ or T(7) is 1 in a rule R_i, a cell configured with $\mathcal{R}_i$ can stick to its current state in the next time step.

Property 5. [25] A rule $\mathcal{R}_i$ can contribute to the formation of single length cycle attractor(s) if at least one of the RMTs $T(0)$, $T(1)$, $T(4)$ or $T(5)$ is 0 (passive), and/or at least one of the RMTs $T(2)$, $T(3)$, $T(6)$ or $T(7)$ is 1, that is, passive.

A CA synthesized with arbitrary rules may result in one or more attractors with multilength cycles. To find the CA rules forming SACAs, the 256 rules are classified in nine groups (Table 5) based on Property 5 [25]). It is observed that to form an SACA, the CA rule should follow Property 5. However a rule (e.g., rule 204) that maintains Property 5 for all its RMTs cannot form an SACA. This leads to the following property.

Property 6. [25] To form a uniform SACA with rule $\mathcal{R}_i$, the $\mathcal{R}_i$ must deny Property 5 for some RMTs.

For example, rule 34 of group 2 denies Property 5 for six RMTs. On the other hand, rule 143 of group 5 denies Property 5 for three RMTs. Each of these rules is an SACA rule. Table 2 displays all the SACA rules.

Definition 8. A rule $\overline{\mathcal{R}_i}$ is the complement of $\mathcal{R}_i$ if each RMT of $\overline{\mathcal{R}_i}$ is the complement of the corresponding RMTs of $\mathcal{R}_i$.

Theorem 1. An SACA rule having a majority of ones in RMTs always forms a nonzero attractor.

Proof. If a majority of RMTs are 1 in an SACA rule, it implies that a majority of 0 RMTs $(T(0), T(1), T(4), T(5))$ are active. This implies that a majority of passive RMTs are 1. Now, a path of self-loops in NSRTD, which identifies an SACA, will consist of passive RMTs, some of which are essentially 1-RMTs. Hence, such a path in NSRTD will lead to a nonzero attractor. □

Figure 6(b) shows the NSRT diagram of rule 85. The nodes one, three and five in the NSRTD contain passive RMT 1, so it forms a nonzero attractor with value 10101.

Theorem 2. If the consecutive pair of RMTs of an SACA rule exhibits different next states, then the complement of the rule also forms an SACA.

Proof. A consecutive pair of RMTs of an SACA rule exhibiting a different next state indicates if RMT $T(0)$ is $0(1)$ then RMT $T(1)$ is $1(0)$. Similarly, if RMT $T(2)$ is $0(1)$ then RMT $T(3)$ is $1(0)$ and so on. Then the RMTs are 10101010 (01010101), of which RMTs $T(1)$, $T(2)$, $T(5)$, $T(6)$ $(T(0), T(3), T(4), T(7))$ deny Property 5. Now the RMT of the complement rule is 01010101 (10101010).

https://doi.org/10.25088/ComplexSystems.30.3.415

From the NSRT diagrams (Figures 6(a) and (b)) it can be seen that both rules $\mathcal{R}_i$ and $\overline{\mathcal{R}_i}$ have a single n-length path, where n is the number of CA cells. So we can say that both of the rules form an SACA. □

Theorem 3. If a rule $\mathcal{R}_i$ and its complement $\overline{\mathcal{R}_i}$ are SACA rules, then one of $\mathcal{R}_i$ or $\overline{\mathcal{R}_i}$ has an all-zeros attractor and the other has a nonzero attractor.

Proof. If rule $\mathcal{R}_i$ is an SACA with zero attractor, then the passive RMTs of $\mathcal{R}_i$ must be a subset of the set of 0-RMTs ($\{\mathrm{T}(0), \mathrm{T}(1), \mathrm{T}(4), \mathrm{T}(5)\}$). However, only the passive RMT $\mathrm{T}(0)$ can lead to the formation of a path with a self-loop on RMT $\mathrm{T}(0)$ that corresponds to a zero attractor. If RMT $\mathrm{T}(0)$ is a passive RMT in $\mathcal{R}_i$, then RMT $\mathrm{T}(1)$ and $\mathrm{T}(2)$ must be passive in $\overline{\mathcal{R}_i}$ (following Theorem 2). So a self-loop path for $\overline{\mathcal{R}_i}$ will start from either of the RMTs $\mathrm{T}(1)$ or $\mathrm{T}(2)$. Though $\mathrm{T}(1)$ is a 0-RMT, its NCR is $\{\mathrm{T}(2), \mathrm{T}(3)\}$ (Table 3), which are 1-RMTs, and any path of a self-loop in the NSRTD with one or more 1-RMTs corresponds to a nonzero attractor. Hence the proof. □

The rule $\mathcal{R}_i = 170$ (10101010) and its complement $\overline{\mathcal{R}_i} = 85$ (01010101) are both SACA rules where rule $\mathcal{R}_i$ forms an all-zeros attractor and $\overline{\mathcal{R}_i}$ forms a nonzero attractor.

The next section describes the synthesis of SACAs in linear time.

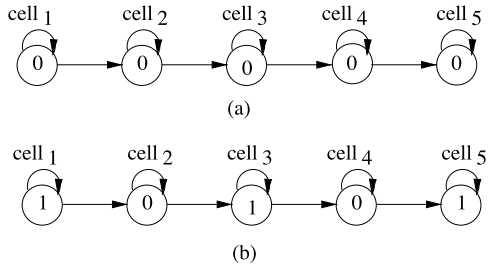

Figure 6. NSRT diagram of uniform 170 and 85.

5. Synthesis of SACAs

The 41 SACA rules are reported in Table 2. The design of a hybrid SACA of length n (for $n \geq 5$) from Table 2, in linear time, is our prime objective. Further, we want to devise a mechanism for synthesizing length-$(n+1)$ SACAs from a length-n SACA structure as well as

a length-$(n + m)$ SACA from the available structures of length-n and length-m SACAs.

To satisfy the given objective, the SACA rules are organized in three different tables: Tables 6 through 8. Table 6 lists the rules that can be selected as the first cell rule ($\mathcal{R}_1$) of the SACAs to be synthesized. The rules in Table 7 are for the intermediate cell rules ($\mathcal{R}_i$, $i = 2, 3, \ldots, n-1$). Table 8 defines the rules for the last cell ($\mathcal{R}_n$) of the SACAs. The logic behind the construction of these three tables is given in the following.

Rules for $\mathcal{R}_1$	Class of $\mathcal{R}_2$
2	IA, IB
8	IIA, IIB
15	III

Table 6. First rule table.

Class of $\mathcal{R}_i$	$\mathcal{R}_i$	Class of $\mathcal{R}_{i+1}$	Class of $\mathcal{R}_{n-1}$
IA	0, 2, 66, 128, 130, 194	IV	NA
IB	8, 16, 24, 40, 64, 192	V	NA
IIA	8, 24, 40, 56, 128, 136	VI	NA
IIB	0, 16, 32, 48	VII	NA
III	0, 2, 8, 10, 15, 128, 130, 136, 138	VIII	NA
IV	0, 8, 16, 24, 32, 40, 48, 56, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240	IV	IX
V	15, 85, 143, 213, 255	V	X
VI, VIII	0, 8, 16, 24, 32, 40, 48, 56, 128, 136, 144, 152, 160, 168, 176, 184	VI, VIII	XI
VII	0, 15, 85, 143, 213	VII	XII
IX, XI	0, 8, 16, 24, 32, 40, 48, 56, 128, 136, 144, 152, 160, 168, 176, 184	XIII	NA
X, XII	0, 66, 96, 98, 128, 136	XIV	NA

Table 7. Class relationship between $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$.

Rule Class for $\mathcal{R}_n$	Rule Set for $\mathcal{R}_n$
XIII	16, 32, 48
XIV	16, 48

Table 8. Last rule table.

https://doi.org/10.25088/ComplexSystems.30.3.415

Table 6: Since the current design is concerned with the null boundary CA, there are 16 effective rules for the leftmost ($\mathcal{R}_1$) cell. The RMTs T(4), T(5), T(6) and T(7) are to be treated as do not care for $\mathcal{R}_1$ as the present state of the left neighbor of the first cell is always 0 (Figure 1). So only four RMTs, T(0), T(1), T(2) and T(3), are effective for $\mathcal{R}_1$. Therefore, the rules 2 (00000010) and 34 (00100010) have the same effect when selected as $\mathcal{R}_1$. The second column of Table 6 shows the class of the second cell rule $\mathcal{R}_2$ when $\mathcal{R}_1$ is selected as per column 1. The rule $\mathcal{R}_2$, selected from Table 7, must be compatible with $\mathcal{R}_1$. That is, self-loop NSRS (NSRS of length 1) of rule $\mathcal{R}_2$ must be the NCR of $\mathcal{R}_1$ (Section 3). All such rules can be classified in five classes: IA, IB, IIA, IIB and III. For example, if rule 2 is selected as $\mathcal{R}_1$ from Table 6, then the second cell rule ($\mathcal{R}_2$) can be selected from class IA or IB of Table 7. Like all other rules of class IA or IB, rule 194 is also compatible with rule 2 as self-loop NSRS on RMT T(0) of rule 194 is the NCR (self-loop NSRS) of RMT T(0) of rule 2. The details of class identification of rules follow in the subsequent discussions.

Table 7: All intermediate rules ($\mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_{n-1}$) for a CA can be selected from Table 7. Since the left and right neighbors of an intermediate cell are not null, all the RMTs T(0), T(1), ..., T(7) are effective for an intermediate cell rule. The relation between $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$ is such that the self-loop NSRS of $\mathcal{R}_{i+1}$ must be the NCR of self-loop NSRS of rule $\mathcal{R}_i$. For example, if the second cell rule ($\mathcal{R}_2$) is selected as 194 from class IA, then the next cell rule $\mathcal{R}_3$ ($\mathcal{R}_{i+1}$) must belong to class IV following Table 7. Similarly, if rule 80 (following row six of Table 7) is chosen as the third cell rule ($\mathcal{R}_3$), then the next cell rule $\mathcal{R}_4$ ($\mathcal{R}_{n-1}$) belongs to class IX (column 4 of Table 7) for getting an SACA of length $n = 5$. Here, we select rule 176 following row 10 of Table 7 as the fourth cell rule. The class of the last cell rule, which is associated with the fourth cell rule ($\mathcal{R}_4$), is XIII (last column of row 10 of Table 7). The fifth (last) cell rule $\mathcal{R}_5$ is to be selected from Table 8. For obtaining an SACA of length $n > 5$, the rule selection for $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ is the same as that of the case for $n = 5$ (as discussed earlier). The next successive cell rules $\mathcal{R}_4, \mathcal{R}_5, \ldots, \mathcal{R}_{n-2}$ can be selected from class IV (same class as that of $\mathcal{R}_3$). For example, for $n = 7$, we have selected $\mathcal{R}_1 = 2$, $\mathcal{R}_2 = 194$, $\mathcal{R}_3 = 80$. Now, $\mathcal{R}_4$ can be chosen from the class of $\mathcal{R}_3$ (class IV). Say, $\mathcal{R}_4 = 224$. Similarly, $\mathcal{R}_5 = 240$ has been chosen from class IV. Next, $\mathcal{R}_6$ ($\mathcal{R}_{n-1}$) should be chosen from class IX (column 4 of Table 7). Here, we have chosen $\mathcal{R}_6 = 176$ ($\mathcal{R}_{n-1}$) from Table 7 and $\mathcal{R}_7$ ($\mathcal{R}_n$) should be chosen consulting Table 8.

Note that some rows in Table 7 for the fourth column are marked with NA. This is to indicate that these rows do not correspond to classes of intermediate cell rules (cl_i) that are immediately followed by the class of the $n-1$ cell rule cl_{n-1}.

Table 8: Like the first cell, there are 16 effective rules for the last cell of an SACA. Here, the RMTs T(1), T(3), T(5), T(7) are do not care and T(0), T(2), T(4), T(6) are effective, as the present state of the right neighbor of cell n ($\mathcal{R}_n$) is always 0. Based on the class defined in Table 7, the last cell rule is chosen from Table 8. For the example design of a five-cell SACA, the class of the last cell rule is defined as XIII. So any one of the rules 16/32/48 (row 1 of Table 8) can be chosen for $\mathcal{R}_5$. Let it be rule 32. Then the synthesized five-cell SACA is $\langle 2, 194, 80, 176, 32\rangle$. Similarly, for a seven-cell SACA, the $\mathcal{R}_7$ should be chosen from the last rule table (Table 8) and the synthesized seven-cell SACA is $\langle 2, 194, 80, 224, 240, 176, 32\rangle$.

Algorithm 1 formalizes the steps for synthesis of an n-cell SACA.

Algorithm 1. SACA synthesis.

Input: n (length of SACA), Tables 6 through 8

Output: The SACA $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\rangle$

Step 1. Select $\mathcal{R}_1$ randomly from Table 6

Step 2. Find cl_2 (class of $\mathcal{R}_2$) from the corresponding row of Table 6

Step 3. Select $\mathcal{R}_2$ from the corresponding row of cl_2 of Table 7

Step 4. Find cl_3 (class of $\mathcal{R}_3$) from the corresponding row of cl_2 of Table 7

Step 5. Select $\mathcal{R}_3$ from the corresponding row of cl_3 of Table 7

Step 6. If ($n > 5$) then,

Step 7. for $i = 3$ to $(n-3)$

Step 8. $cl_{i+1} = cl_i$

Step 9. Select $\mathcal{R}_{i+1}$ from the corresponding row of cl_{i+1} of Table 7

Step 10. End for

Step 11. Find cl_{n-1} from the corresponding row of cl_{i+1} (cl_{n-2}) of Table 7

Step 12. Select $\mathcal{R}_{n-1}$ from the corresponding row of cl_{n-1} of Table 5

Step 13. Find cl_n (class of $\mathcal{R}_n$) from the corresponding row of cl_{n-1} of Table 7

Step 14. Select $\mathcal{R}_n$ from the corresponding row of cl_n of Table 8

https://doi.org/10.25088/ComplexSystems.30.3.415

Step 15. End if

Step 16. If ($n = = 5$)

Step 17. Find cl_4 (class of $\mathcal{R}_4$) from the corresponding row of cl_3 (as found in Step 4) of Table 7

Step 18. Select $\mathcal{R}_4$ from the corresponding row of cl_4 of Table 5

Step 19. Find cl_5 (class of $\mathcal{R}_5$) from the corresponding row of Table 5

Step 20.Select $\mathcal{R}_5$ ($\mathcal{R}_n$) from the corresponding row of cl_5 (cl_n) of Table 5

Step 21. End if

Step 22. Output $\langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n \rangle$

Step 23. End

Complexity: In Algorithm 1, the first ($\mathcal{R}_1$) and last ($\mathcal{R}_n$) cell rules can be selected in $O(1)$ time from Tables 6 and 8, respectively. The remaining rules ($\mathcal{R}_2$ to $\mathcal{R}_{n-1}$) can be set from Table 7, which requires $n-2$ iterations each of constant time. So the time complexity of Algorithm 1 is $O(n)$.

Example 1. Let us consider the synthesis of a five-cell SACA following Algorithm 1. At first, rule 2 is selected randomly as $\mathcal{R}_1$ from Table 6 (Step 1 of Algorithm 1). Therefore, the class (obtained from Table 6) for the second cell rule is either IA or IB (Step 2 of Algorithm 1). From class IB of Table 7, say rule 192 is selected randomly as $\mathcal{R}_2$ (Step 3 of Algorithm 1). Say rule 15 as $\mathcal{R}_3$ has been chosen randomly from class V (Step 5 of Algorithm 1). Now, the condition in Step 16 is satisfied. So rule 66 as $\mathcal{R}_4$ belonging to class X can be chosen randomly. At Step 19 of Algorithm 1, the cl_5 is XIV (class of $\mathcal{R}_5$). Therefore, the last cell rule $\mathcal{R}_5$ can be selected randomly as 16 from class XIV of Table 8. So the five-cell SACA thus constructed is $\langle 2, 192, 15, 66, 16 \rangle$.

A minor modification of Algorithm 1 can allow controlling the synthesis of an SACA with an all-zeros attractor or with a nonzero attractor. Algorithm 2 describes the procedure to synthesize such an SACA with the desired attractor state. It reads the attr (desired attractor) as input and then synthesizes an SACA having the attractor attr.

Algorithm 2. Scalable SACA synthesis.

Input: n (length of SACA), Tables 6 through 8, attr

Output: The SACA $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n \rangle$ with specified attractor

Step 1. Read attr (attr = 0 for zero attractor and attr = 1 for nonzero attractor)

Step 2. If attr == 0, select $\mathcal{R}_1$ such that $\mathcal{R}_1 \in \{2, 8\}$ from Table 6

Step 3. If $(n >= 5)$ then,

Step 4. If $(\mathcal{R}_1 == 2)$ then,

Step 5. The next successive cell rule classes are IA, $(IV)^i$ and IX from Table 7 where $i = 1$ to $n - 4$ and XIII from Table 8

Step 6. End if

Step 7. If $(\mathcal{R}_1 == 8)$ then,

Step 8. The next successive cell rule classes are IIA, $(VI)^i$ and XI from Table 7 where $i = 1$ to $n - 4$ and XIII from Table 8

Step 9. End if

Step 10. End if

Step 11. End if

Step 12. If attr == 1, select $\mathcal{R}_1$ such that $\mathcal{R}_1 \in \{2, 8, 15\}$ from Table 6

Step 13. If $(n >= 5)$ then,

Step 14. If $(\mathcal{R}_1 == 2)$ then,

Step 15. The successive cell rule classes are IB, $(V)^i$ and X from Table 7 where $i = 1$ to $n - 4$ and XIV from Table 8

Step 16. End if

Step 17. If $(\mathcal{R}_1 == 8)$ then,

Step 18. The successive cell rule classes are IIB, $(VII)^i$ and XII from Table 7 where $i = 1$ to $n - 4$ and XIV from Table 8

Step 19. End if

Step 20. If $(\mathcal{R}_1 == 15)$ then,

Step 21. The successive cell rule classes are III, $(VIII)^i$ and XI from Table 7 where $i = 1$ to $n - 4$ and XIII from Table 8

Step 22. End if

Step 23. End if

Step 24. End if

Step 25. End

Complexity: The computation of Algorithm 2 is dominated by the time taken to select the CA cell rules one by one from the respective classes in steps 5, 8, 15, 18 and 21, except for the first cell rule. Each

of the steps is of linear time. So the time complexity of Algorithm 2 is $O(n)$.

Example 2. Let us consider the case for $n = 5$ and attr $= 0$. Then Algorithm 2 outputs a five-cell SACA with an all-zeros attractor. The first rule $\mathcal{R}_1$ can be set as either 2 or 8 from Table 6 (Step 2). Consider that $\mathcal{R}_1 = 2$ has been selected from Table 6. The successive cell rules are chosen one by one from the classes {IA, IV, IX, XIII}, respectively. So the resulting SACA is $\langle 2, 194, 80, 176, 32\rangle$ and it has an all-zeros (00000) attractor.

5.1 Synthesis of $(n+1)$-Cell SACAs

The target of this synthesis scheme is to configure an $(n+1)$-cell SACA from an available SACA structure of length n. Consider that $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_n\rangle$ is an SACA of length n where the first and last rules belong to Tables 6 and 8, respectively. All other rules (second, intermediate, second to last) belong to Table 7. Now, the task is to construct an $(n+1)$-cell SACA reusing the structure of the n-cell SACA. We propose to realize it by adding an intermediate rule $\mathcal{R}_{\text{add}}$ from Table 7. The new CA configured $\langle \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{\text{add}}, \mathcal{R}_{n-1}, \mathcal{R}_n\rangle$ is an SACA if and only if the rule $\mathcal{R}_{\text{add}}$ is compatible with its left and right neighbor cell rules. Algorithm 3 describes the synthesis scheme.

Algorithm 3. Scalable SACA synthesis.

Input: Table 7, n (length of SACA),
$\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_n\rangle$
Output: $(n+1)$-length SACA
$\mathcal{R}' = \langle \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{\text{add}}, \mathcal{R}_{n-1}, \mathcal{R}_n\rangle$

Step 1. Find $\mathcal{R}_1$ and $\mathcal{R}_2$ from $\mathcal{R}$

Step 2. Find cl_2, that is, the class of $\mathcal{R}_2$ (from the row of Table 6)

Step 3. If $cl_2 =$ IA then

Step 4. $cl_3 =$ IV

Step 5. If $cl_2 =$ IB then

Step 6. $cl_3 =$ V

Step 7. If $cl_2 =$ IIA then

Step 8. $cl_3 =$ VI

Step 9. If $cl_2 =$ IIB then

Step 10. cl_3 = VII

Step 11. If cl_2 = III then

Step 12. cl_3 = VIII

Step 13. Select $\mathcal{R}_{\text{add}}$ from the class cl_3 from Table 7

Step 14. Add $\mathcal{R}_{\text{add}}$ before the second to last rule ($\mathcal{R}_{n-1}$) to get the SACA of length $(n+1)$

Step 15. Return the $(n+1)$-length SACA
$\mathcal{R}' = \langle \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{\text{add}}, \mathcal{R}_{n-1}, \mathcal{R}_n \rangle$

Step 15. End

Complexity: Finding the class of rule $\mathcal{R}_{\text{add}}$ (cl_3) takes $O(1)$ time and then the selection of rule $\mathcal{R}_{\text{add}}$ can be made randomly from the class cl_3, which also takes $O(1)$ time. So Algorithm 3 takes $O(1)$ time.

Example 3. Let us consider the five-cell SACA $\mathcal{R} = \langle 2, 194, 80, 176, 32 \rangle$ constructed from Tables 6 through 8. The class cl_2 IA of the second-cell rule (194) can be determined from the first-cell rule. Similarly, the class cl_3 of the third or intermediate rule is class IV, as the class of second-cell rule cl_2 is IA. To design a six-cell SACA reusing the structure of the five-cell SACA $\mathcal{R}$, the synthesis tool has to select $\mathcal{R}_{\text{add}}$ as the intermediate rule from the class IV of Table 7. Let us assume the selected candidate for $\mathcal{R}_{\text{add}}$ is rule 192 from class IV. So the resulting six-cell SACA is $\mathcal{R} = \langle 2, 194, 80, 192, 176, 32 \rangle$. Similarly, $\mathcal{R} = \langle 2, 194, 80, 192, 208, 176, 32 \rangle$ is also a seven-length SACA synthesized by adding rule 208 from the same class (IV) of Table 7. Figure 7 depicts the synthesis process for $(n+1)$-cell SACA from an n-cell SACA.

5.2 Synthesis of $(n+m)$-Cell SACAs

In this subsection, we propose the synthesis of an $(n+m)$-cell SACA utilizing the available structure of an n-cell and m-cell SACA. Consider the SACA $\text{CA}_1 = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_n \rangle$ and $\text{CA}_2 = \langle \mathcal{R}'_1, \mathcal{R}'_2, \ldots, \mathcal{R}'_i, \ldots, \mathcal{R}'_{m-1}, \mathcal{R}'_m \rangle$ of length n and m, respectively. The requirement is that concatenating the CA_1 and CA_2 structures should form an SACA of length $(n+m)$. Algorithm 4 describes the logical steps of such a synthesis scheme.

https://doi.org/10.25088/ComplexSystems.30.3.415

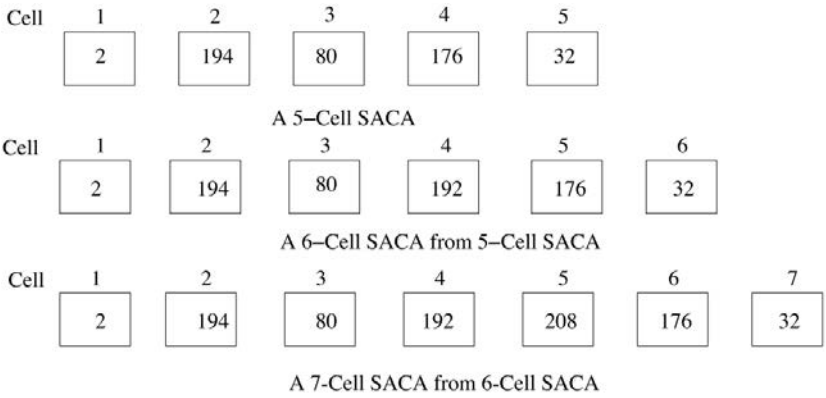

Figure 7. Scalable SACA design.

Algorithm 4. Synthesis SACA $(n + m)$.

Input: Tables 6 and 7, n-length SACA

$CA_1 = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_n \rangle$, m-length SACA
$CA_2 = \langle \mathcal{R}'_1, \mathcal{R}'_2, \ldots, \mathcal{R}'_i, \ldots, \mathcal{R}'_{m-1}, \mathcal{R}'_m \rangle$

Output: $(n + m)$ length SACA $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{m+n} \rangle$

Step 1. Find $\mathcal{R}_1$ and $\mathcal{R}_2$ from the n-length SACA CA_1

Step 2. Find cl_2; that is, class of $\mathcal{R}_2$, from the row of Table 6 corresponding to $\mathcal{R}_1$

Step 3. Find cl_3; that is, class of $\mathcal{R}_3$, from the row of Table 7 corresponding to $\mathcal{R}_2$

Step 4. If ($\{\mathcal{R}_{n-1}, \mathcal{R}_n\}$ of $CA_1 \in cl_3$ of CA_1 and $\{\mathcal{R}'_1, \mathcal{R}'_2, \mathcal{R}'_3\}$ of $CA_2 \in cl_3$ of CA_1), then

Step 5. Concatenate CA_1 and CA_2

Step 6. Return $(n + m)$ length SACA $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_{m+n} \rangle$

Step 7. Else

Step 8. Output, cannot generate $(n + m)$ length of SACA

Step 9. End

Complexity: Determining the classes of $\mathcal{R}_2$, $\mathcal{R}_3$ from the row of Tables 6 and 7 corresponding to $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively, takes constant time. Step 3 of Algorithm 4 checks whether the condition is satisfied or not. These checks are independent of the length of CA_1 (n) and CA_2 (m). Hence, Algorithm 4 requires constant time.

Example 4. Consider that $CA_1 = \langle 2, 8, 24, 32, 16\rangle$ and $CA_2 = \langle 8, 48, 136, 32, 48\rangle$ are two SACAs each of length five. Rule 24 is the intermediate cell rule of CA_1 whereas rules 32 and 16 are the second to last and last cell rules, respectively. Similarly, the first three rules of CA_2 are 8, 48 and 136. Now, for getting an SACA structure of length 10 utilizing the structure of CA_1 and CA_2, the rules 32, and 16 from CA_1 and the rules 8, 48 and 136 from CA_2 should belong to the same class as rule $\mathcal{R}_3$ of CA_1, that is, IV (shown in Table 7). The second to last cell rule (32) and last cell rule (48) of CA_2 should be the second to last and last cell rules of the resulting CA. So concatenating CA_1 and CA_2 forms an SACA $\langle 2, 8, 24, 32, 16, 8, 48, 136, 32, 48\rangle$ of length 10.

6. Hardware Implementation of SACAs

Designing an n-cell SACA requires n flip-flops (FFs) and combinational logic (AND/OR/XOR/NOT gates) to implement CA rules. Figure 8 depicts the hardware implementation of a five-cell SACA (cell 1 to cell 5), where the cells are configured with rules 2, 194, 80, 176 and 32, respectively. The next state logic of rules 2, 194, 80, 176 and 32 is $NS = S_i' \cdot S_{i+1}$, $NS = S_{i-1}' \cdot S_i' \cdot S_{i+1} + S_{i-1}S_i$, $NS = S_{i-1} \cdot S_{i+1}'$, $NS = S_{i-1} \cdot S_i' + S_{i-1} \cdot S_{i+1}$ and $NS = S_{i-1} \cdot S_i' \cdot S_{i+1}$, respectively.

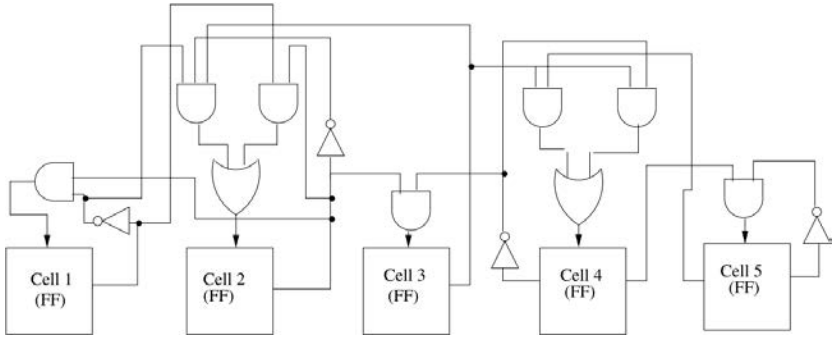

Figure 8. Hardware design for the synthesis of a five-cell SACA $\langle 2, 194, 80, 176, 32\rangle$.

6.1 Design of $(n+1)$-Cell SACAs Utilizing n-Cell SACA Structure

This targets the design of an $(n+1)$-cell SACA by sharing the hardware design for n-cell SACAs. Figure 9 illustrates the generic architecture of such a scalable SACA structure. For ease of understanding, the

https://doi.org/10.25088/ComplexSystems.30.3.415

combinational logic of each rule is hidden in the functional block (F). The design shown in Figure 9 can act as the n-cell SACA as well as an $(n+1)$-cell SACA. Before the second to last cell (cell 4 in Figure 9) of the existing design, two 2:1 multiplexers are added. While considering it to realize the n-cell SACA, the two multiplexers are reset to 0. For implementing the $(n+1)$-cell SACA, the multiplexers are set to 1, so that the next state logic of cell 3 and cell 4 also depends on the present state of the newly added cell (cell_{add}). Consequently, the next state of cell_{add} will also depend on the present state of cell 3, cell_{add} and cell 4.

6.2 Design of an $(n+m)$-Cell SACA Utilizing n-Cell and m-Cell Structure

The objective of this design is to construct an $(n+m)$-cell SACA from two different available SACAs of lengths n and m, respectively. While reusing the structure of an n-cell SACA, its rightmost cell should act as the n^{th} cell of the $(n+m)$-cell SACA. Further, the first cell of the m-cell SACA is selected in such a way that it can also act as the $(n+1)^{\text{th}}$ cell of the $(n+m)$-cell SACA.

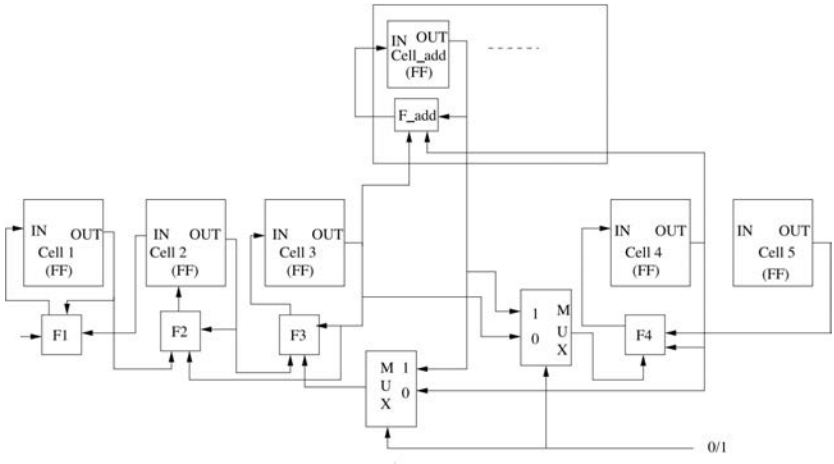

Figure 9. Hardware design for synthesis of $(n+1)$-cell SACAs from an n-cell SACA.

Figure 10 shows the architecture of such a design. Two switches FF_1 and FF_2 facilitate the SACA structures. Whenever both D_1 of FF_1 and D_2 of FF_2 are set to 0, the design acts as the n-cell SACA and m-cell SACA separately. To activate the $(n+m)$-cell SACA, both D_1 of FF_1 and D_2 of FF_2 are to be set to 1.

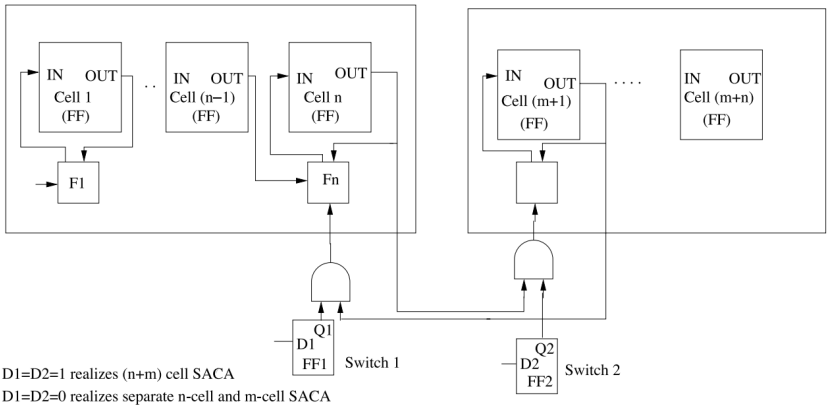

Figure 10. Hardware design for synthesis of $n + m$-cell SACAs.

1. Conclusion

This paper introduces the synthesis of single length cycle, single attractor cellular automata (SACAs) in $O(n)$ time, where n is the number of cells in the cellular automaton (CA). The cascadable structure of an n-cell single length cycle, single attractor cellular automaton (SACA) is utilized to construct the $(n+1)$-cell SACAs. This paper also proposes constructing $(n+m)$-cell SACAs from the available design of n-cell and m-cell SACAs. The proposed scalable SACA synthesis scheme is effectively employed for designing the test application for very large-scale integration (VLSI) design, memory testing and cache coherence in CMPs, and so on.

References

[1] E. F. Moore, "Machine Models of Self Reproduction," in *Proceedings of Symposia in Applied Mathematics*, Vol. 14, (R. E. Bellman, ed.), Providence, RI: American Mathematical Society, 1962 pp. 17–33.

[2] D. A. Rosenblueth and C. Gershenson, "A Model of City Traffic Based on Elementary Cellular Automata," *Complex Systems*, **19**(4), 2011 pp. 305–322. doi:10.25088/ComplexSystems.19.4.305.

[3] J. Thatcher, "Universality in von Neumann Cellular Model," University of Michigan, Technical Report 03105-30-T, ORA, 1964.

[4] N. Margolus and T. Toffoli, "Cellular Automata Machines," *Complex Systems*, **1**(5), 1987 pp. 967–993. complex-systems.com/pdf/01-5-5.pdf.

[5] L. Zaloudek and L. Sekanina, "Increasing Fault-Tolerance in Cellular Automata-Based Systems," in *Unconventional Computation (UC'11)*, Turku, Finland (C. S. Calude, J. Kari, I. Petre and G. Rozenberg, eds.), Berlin, Heidelberg: Springer-Verlag, 2011 pp. 234–245. doi:10.1007/978-3-642-21341-0_26.

[6] T. Yang, "Morphology Analysis of Urban Traffic Based on Multilevel Cellular Automata Networks Dynamics," in *Proceedings of 6th International Congress on Image and Signal Processing (CISP)*, Hangzhou, China, Piscataway, NJ: IEEE 2013 pp. 980–984. doi:10.1109/cisp.2013.6745307.

[7] G. J. Martinez, "A Note on Elementary Cellular Automata Classification," *Journal of Cellular Automata*, **8**(3), 2013 pp. 233–259.

[8] H. Zenil and E. Villarreal-Zapata, "Asymptotic Behavior and Ratios of Complexity in Cellular Automata," *International Journal of Bifurcation and Chaos*, **23**(9), 2013 1350159. doi:10.1142/S0218127413501599.

[9] A. Wuensche and M. Lesser, *The Global Dynamics of Cellular Automata*, Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA: Addison-Wesley, 1992.

[10] P. Pal Chaudhuri, D. R. Chowdhury, S. Nandi and S. Chattopadhyay, *Additive Cellular Automata: Theory and Applications*, Los Alamitos, CA: IEEE Computer Society Press, 1997.

[11] N. Ganguly, "Cellular Automata Evolution: Theory and Applications in Pattern Recognition and Classification," Ph.D. thesis, Bengal Engineering College (DU), 2004.

[12] P. Maji, "Cellular Automata Evolution for Pattern Recognition," Ph.D. thesis, Bengal Engineering College (DU), 2004.

[13] N. S. Maiti, S. Ghosh, B. K. Sikdar and P. Pal Chaudhuri, "Rule Vector Graph (RVG) to Design Linear Time Algorithm for Identifying the Invertibility of Periodic-Boundary Three Neighborhood Cellular Automata," *Journal of Cellular Automata*, **7**(4), 2012 pp. 335–362.

[14] N. S. Maiti, S. Ghosh, S. Munshi and P. Pal Chaudhuri, "Linear Time Algorithm for Identifying the Invertibility of Null-Boundary Three Neighborhood Cellular Automata," *Complex Systems*, **19**(1), 2010 pp. 89–113. doi:10.25088/ComplexSystems.19.1.89.

[15] K. Cattel and J. C. Muzio, "Synthesis of One-Dimensional Linear Hybrid Cellular Automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **15**(3), 1996 pp. 325–335. doi:10.1109/43.489103.

[16] P. Dasgupta, S. Chattopadhyay and I. Sengupta, "An ASIC for Cellular Automata Based Message Authentication," in *Proceedings of the 13th International Conference on VLSI Design (VLSID '00)*, Calcutta, India, Los Alamitos, CA: IEEE Computer Society, 1999 pp. 538–541. doi:10.1109/ICVD.2000.812663.

[17] H. Gutowitz, "Cryptography with Dynamical Systems," *Cellular Automata and Cooperative Systems* (N. Boccara, E. Goles, S. Martinez and P. Picco, eds.), Dordrecht: Springer, 1993, pp. 237–274. doi:10.1007/978-94-011-1691-6_21.

[18] M. Mukherjee, N. Ganguly, A. Chaudhuri and P. Pal Chaudhuri, "Cellular Automata Based Authentication (CAA)," in *Cellular Automata (ACRI 2002)*, Geneva, Switzerland (S. Bandini B. Chopard and M. Tomassini, eds.), Berlin, Heidelberg: Springer, 2002 pp. 259–269. doi:10.1007/3-540-45830-1_25.

[19] M. Dalui, "Theory and Applications of Cellular Automata for CMPs Cache System Protocol Design and Verification," Ph.D. thesis, IIEST Shibpur, West Bengal, India, 2014.

[20] B. Chakraborty, M. Dalui and B. K. Sikdar, "Design of Coherence Verification Unit for Heterogeneous CMPs Integrating Update and Invalidate Protocols," in *30th International Conference on VLSI Design and 16th International Conference on Embedded System (VLSID)*, Hyderabad, India, Los Alamitos, CA: IEEE Computer Society, 2017 pp. 115–120. doi:10.1109/VLSID.2017.30.

[21] B. Chakraborty, B. P. Singh, M. Chinnapureddy, M. Dalui and B. K. Sikdar, "Design of Coherence Verification Unit for Heterogeneous CMPs," in *19th International Symposium on VLSI Design and Test*, Ahmedabad, India, Los Alamitos, CA: IEEE Computer Society, 2015 pp. 1–6. doi:10.1109/ISVDAT.2015.7208098.

[22] B. Chakraborty, M. Dalui and B. K. Sikdar, "Design of Coherence Verification Unit for CMPs Realizing Dragon Protocol," in *20th International Symposium on VLSI Design and Test (VDAT 2016)*, Guwahiti, India, Los Alamitos, CA: IEEE Computer Society, 2016 pp. 1–6. doi:10.1109/ISVDAT.2016.8064850.

[23] B. Chakraborty, M. Dalui and B. K. Sikdar, "Cellular Automata Based Test Design for Coherence Verification in 3D Caches," *Journal of Circuits, Systems and Computers*, **28**(9), 2019 1950148. doi:10.1142/S0218126619501482.

[24] B. Chakraborty, M. Dalui and B. K. Sikdar, "Design of a Reliable Cache System for Heterogeneous CMPs," *Journal of Circuits, Systems and Computers*, **27**(14), 2018 1850219. doi:10.1142/S0218126618502195.

[25] S. Das, N. N. Naskar, S. Mukherjee, M. Dalui and B. K. Sikdar, "Characterization of CA Rules for SACA Targeting Detection of Faulty Nodes in WSN," in *Cellular Automata (ACRI 2010)*, Ascoli Piceno, Italy (S. Bandini, S. Manzoni, H. Umeo and G. Vizzari, eds.), Berlin, Heidelberg: Springer, 2010 pp. 300–311. doi:10.1007/978-3-642-15979-4_32.