2019

Using Homing, Synchronizing and Distinguishing Input Sequences for the Analysis of Reversible Finite State Machines

Martin Lukac
Nazarbayev University

Michitaka Kameyama
Tohoku University

Marek A. Perkowski
Portland State University, marek.perkowski@pdx.edu

Pawel Kerntopf
Warsaw University of Technology

Citation Details

Lukac, M., Kameyama, M., Perkowski, M., & Kerntopf, P. (2019). USING HOMING, SYNCHRONIZING AND DISTINGUISHING INPUT SEQUENCES FOR THE ANALYSIS OF REVERSIBLE FINITE STATE MACHINES. Facta Universitatis, Series: Electronics and Energetics, 32(3), 417-438.

FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 32, No 3, September 2019, pp. 417 - 438
https://doi.org/10.2298/FUEE1903417L

USING HOMING, SYNCHRONIZING AND DISTINGUISHING INPUT SEQUENCES FOR THE ANALYSIS OF REVERSIBLE FINITE STATE MACHINES

Martin Lukac[1], Michitaka Kameyama[2], Marek Perkowski[3], Pawel Kerntopf[4]

[1]Nazarbayev University, Astana, Kazakhstan
[2]Ishinomaki Senshu University, Ishinomaki, Japan
[3]Portland State university, Portland, Oregon, USA
[4]University of Lodz, Lodz Poland

Abstract. *A digital device is called reversible if it realizes a reversible mapping, i.e., the one for which there exist a unique inverse. The field of reversible computing is devoted to studying all aspects of using and designing reversible devices. During last 15 years this field has been developing very intensively due to its applications in quantum computing, nanotechnology and reducing power consumption of digital devices. We present an analysis of the Reversible Finite State Machines (RFSM) with respect to three well known sequences used in the testability analysis of the classical Finite State Machines (FSM). The homing, distinguishing and synchronizing sequences are applied to two types of reversible FSMs: the converging FSM (CRFSM) and the non-converging FSM (NCRFSM) and the effect is studied and analyzed. We show that while only certain classical FSMs possess all three sequences, CRFSMs and NCRF-SMs have properties allowing to directly determine what type of sequences these ma-chines possess.*

Key words: *Reversible Logic, Finite State Machines, Testing*

1 INTRODUCTION

One of the problems when designing sequential logic is the ability to efficiently generate tests, apply them to the circuit under test and design an easily testable circuit (Design for test - DFT). In classical Finite State Machines (FSMs) this issue

Received August 28, 2018; received in revised form April 23, 2019
Corresponding author: Martin Lukac
53 Kabanbay Batyr Ave, Block 7, Office 7237, Nur-Sultan city, Republic of Kazakhstan, 010000
(E-mail: martin.lukac@nu.edu.kz)

is well studied and various techniques exist to determine FSM's testability [1–11]. Among the desired characteristics of a testable FSM is the identification of an unknown current state and the ability to bring the FSM to a known state. These properties are verified by the homing, synchronizing and distinguishing input sequences. Using these sequences it is possible to establish whether a classical FSM

- possesses a transition path from the initial to the final state,
- possesses a transition path from any state to any another state and
- allows by only observing the machine's output sequence, to reach an arbitrary state from a different arbitrary state.

The motivation for the construction of automata possessing a particular set of sequences can be appreciated by observing the advantages of each of the sequences separately. The homing sequences have been successfully used in hardware fault detection [12] and in machine learning [13, 14]. The synchronizing sequence has been successfully used in various designs where the existence of the synchronizing sequence allows to simplify the circuit implementation and testing. Finally, the distinguishing sequence is used to build a checking sequence used to verify if an implementation of an FSM is consistent with its specification [2, 10]. Consequently, an FSM that possesses the desired sequences would be highly testable and thus both practical in industrial applications and useful for theoretical research.

The general area of sequential reversible circuits and automata has been explored several traditional approaches. In [15, 16], optimized D and JK latches have been proposed. In [17] proposed to build reversible components for sequential circuits based on Toffoli reversible gates and in [18] the elements of sequential reversible circuits were implemented using conservative Fredkin logic gates. In [19] the reversible T-flip-flop was proposed to be built using custom gates. In [20] the authors proposed to build a memory cell from two Toffoli gates in standard CMOS. In [21] a reversible double-edge triggered flip-flop was build on FPGA. Testable sequential devices based on Quantum Cellular Automata have been explored in [22, 23]. Reversible sequential circuits have also been designed in multiple-valued logic such as in [24]. In the construction of Reversible FSMs (RFSMs), the reversible computation imposes the reversibility constraints and thus limits the construction [25, 26]. This makes the RFSMs harder to construct and more expensive because ancilla bits and additional logic are required to preserve or achieve the reversibility [27–30].

Consequently, applying methods based on the established classical FSMs to RFSMs, can determine if the established classical FSMs methods are sufficient,

need to be improved or if unique new methods must be designed when dealing with the reversible computational paradigm.

RFSMs have been studied for their properties of universality [25] and furthermore extensive studies have been conducted in the area of reversible cellular automata (RCA) [31–35]. An RCA can be seen as a RFSMs with spatial constraints. An RCA maps input states to next states using local rules. A local rule is a function of k spatially closest inputs that is repeatedly applied to all n inputs. While RCA have been studied for their effective implementation and powerful computation abilities [36] no study on the testability of RCA has been done.

Similarly, up to now there have been no serious efforts to explore the general testability of RFSMs from the point of view of the well established testing techniques such as those used in testing of classical irreversible FSMs. Specifically, the impact of the three above introduced sequences has not been studied at all for RFSMs and for the FSMs embedded in RFSMs. Some studies into testability have been performed from the classical point of view such as in [37].

In this paper the constraints of the reversible-permutative and unitary matrices used to specify FSMs are studied with respect to the three above introduced sequences. We assume that an RFSM is specified by a permutation matrix and our analysis is limited to such permutative and discrete RFSMs. We apply the three types of sequences to RFSMs and determine their power when used on reversible sequential devices. The main contributions of this paper are:

1. the analysis of the homing, synchronizing and distinguishing sequences for RFSMs,

2. criteria for RFSMs in order to have the homing, distinguishing, or synchronizing sequences.

The paper is organized as follows. First, background on the classical and reversible FSMs is given in Section 2. Section 3 describes and defines the terms and concepts necessary for the understanding of the three considered sequences and Section 4 shows the application and analysis of the sequences related to both the converging and non-converging finite state machines CRFSMs/NCRFSMs. Section 5 concludes the paper by summarizing our results.

2 BACKGROUND

Let A and B be finite non-empty sets.

Definition 1 (Balanced Logic Function)**.** *A function $f : A \to B$ is balanced if*

for every value $b \in B$ there is the same number of combinations of input variable values in its domain A.

Definition 2 (Reversible Logic Function)**.** *A function $f : A \to B$ is reversible if it is one-to-one and onto. In other words, $f(X_a) = f(X_b) \implies X_a = X_b$ and $f(x) = y$ for $x \in X$ and $y \in Y$.*

Table 1: Example of (a) an irreversible logic function F, (b) a reversible logic function F_R.

AB	A'B'
00	01
01	11
10	00
11	11

F

(a)

AB	A'B'
00	00
01	11
10	01
11	10

F_R

(b)

Table 1a shows example of an irreversible logic function and Table 1b an example of a reversible function.

Definition 3 (Permutative Matrix)**.** *A permutative matrix for n input variables, is a sparse matrix with binary coefficients performing a reversible function $f : B^n \to B^n$.*

Reversible functions in this paper will be represented by truth tables and permutative matrices. Table 1a and 1b show examples of irreversible and reversible functions respectively. The corresponding matrices are shown in eq. (1) and (2).

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \quad (1) \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad (2)$$

Definition 4 (Indexing Variable)**.** *The indexing variable of a block diagonal matrix is the input variable that allows to separate a block diagonal matrix to independent reversible matrices.*

Definition 5 (Reversible Logic Gate)**.** *A reversible logic gate (or circuit) on n-variables realizes a $n \times n$ reversible function $F : I \to O$.*

Definition 6 (Controlled Reversible Logic Gate)**.** *A k-controlled reversible logic gate on $k+1$ variables performs a logic function on target variable i_{k+1} while leaving all control variables $i_1, \ldots, i_k$ unchanged such that*

$$F_{k+1}(i_1, \ldots, i_k, i_{k+1}) = \begin{cases} 1 \oplus i_{k+1} & \text{if for all } j = 1, \ldots, k, i_j = 1 \\ i_{k+1} & \text{otherwise} \end{cases} \quad (3)$$

That is, it realizes a one variable balanced function on a target variable if all k control variables have value 1.

A k-controlled reversible logic gate on $k+1$ variables uses k control variables and one target variable such that k variables remain unchanged after the application of the reversible logic gate on the target variable i_{k+1}.

Definition 7 (Positive and Negative Control)**.** *A positive (negative) control is a variable i that must be 1 (0) in order to activate the function on target variable.*

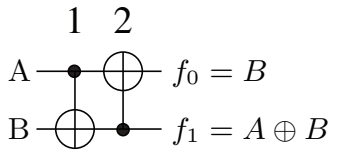

Fig. 1: Example of realization of a reversible function

Reversible circuits are built from reversible gates. For instance, consider the realization of an reversible function shown in Figure 1. Gates 1 and 2 are both two variable gates, called CNOT. A CNOT gate is a single variable positive controlled NOT gate implementing the function $f = A \oplus B$. Observe gate labelled 1 in circuit from Figure 1: the control variable A controls the NOT operation on variable B. The CNOT gate labelled 2, applies the NOT gate on variable A and is controlled by B. Therefore the CNOT gate 1 in Figure 1 implements function $f_1 = A \oplus B$ and CNOT gate 2 implements $f_0 = A \oplus B \oplus A = B$.

Definition 8 (Finite State Machine)**.** *A finite state machine (FSM) is a sequential device defined by a septuple $M = (I, O, S, S_i, S_f, F, G)$ where I is the input alphabet (a finite, non-empty ordered set of input symbols), O is the output alphabet (a finite, non-empty ordered set of output symbols), S is a finite, non-empty ordered set of states, $S_i \subset S$ is the set of initial states, $S_f \subset S$ is the set of final states, F is the next state function given by the mapping $F : I \times S \to S'$ and G is the output function given by the mapping $G : I \times S \to O$.*

Table 2: Example of an irreversible FSM

S	I	
	0	1
A	B/0	A/1
B	A/0	D/0
C	A/1	D/1
D	C/0	B/1

Table 2 shows an example of an FSM. In Table 2 first column shows in every row one possible current state of the FSM. The second and third columns show the the next sate and output assignments for input values $I = 0$ and $I = 1$ respectively.

Definition 9 (Reversible Finite State Machine (RFSM))**.** *A RFSM is a state machine* $M = (I, O, S, S_i, S_f, F, G)$ *with state transition function* $F : I \times S \rightarrow S'$ *and output function* $G : I \times S \rightarrow O$ *are balanced logic functions and such that* $I \times S \rightarrow O \times S'$ *is a reversible function.*

In this paper we will distinguish two types of RFSMs: the convergent RFSM (CRFSM) and its special case the non-convergent RFSM (NCRFSM). The difference is in the fact that for any NCRFSM and for a given input value, every next state assignment is unique. This difference is illustrated in Table 3; Table 3(a) shows an NCRFSM and 3(b) and CRFSM in the so called reversible specification table.

Table 3: Example of (a) non-converging Λ_N and (b) converging Λ_C specifications.

S	I	
	0	1
A	D/1	A/1
B	A/0	C/1
C	C/0	D/0
D	B/1	B/0

(a)

S	I	
	0	1
A	D/1	C/0
B	A/0	C/1
C	A/1	D/0
D	B/1	B/0

(b)

Definition 10 (NCRFSM Evolution)**.** *The input-state-output mapping* $\Lambda_N(F \times G) : I \times S \rightarrow S' \times O$*. The evolution function* Λ_N *is a bijection with constraint that when* I *is used as input variable(s), each output state occurs once at most:* $\forall i \in I, \forall s \in S,\ F(i, s) \neq F(i, s')$.

Example of a NCRFSM is shown in Table 3(a).

Definition 11 (CRFSM Evolution)**.** *The input-state-output mapping* $\Lambda_C(F \times G) : I \times S \rightarrow S' \times O$*. The evolution function* Λ_C *is a bijection with the only constraint being that each pair* $\{S', O\}$ *occurs only once in* Λ_C*.*

Example of CRFSM is shown in Table 3(b). In this paper the reference RFSM will be used to address any of the CRFSM/NCRFSM unless specifically indicated.

Note two major differences between the definition of the RFSM and FSM:

1. A RFSM must preserve reversibility
2. it has to be specified for all the combinations of the input-state $\{I, S\}$ values
3. all state-outputs $\{S'O\}$ combinations must occur only once in the reversible specification table.

Definition 12 (State-Output Set θ)**.** *Given a RFSM* M_R*, the state-output set* θ *is the set of all combinations of* $s \in S$ *and* $o \in O$*. The size of* θ *is* $|S| * |O|$*, with element indices* $i = 1, \ldots, |S| * |O|$*. Let* $k = 0, \ldots, |S| - 1$ *and* $j = 0, \ldots, |O| - 1$ *then the indices of elements in* θ *are calculated as* $i = k + |O| * j$*.*

Definition 13 (Don't Care)**.** *The Don't Care* $*$ *is used to represent unknown or unconsidered values of states, output or input values.*

In the analysis of an RFSM the input sequence uncertainty is used to describe the knowledge about the current state of the RFSM. To understand the concept of input sequence uncertainty, first the concepts of partition and cover need to be defined and explained. The following concepts are adapted to the CRFSMs and NCRFSMs from the original ones defined in [11].

Definition 14 (Ordered State Partition)**.** *Given a CRFSM/NCRFSM* M*, the Ordered State Partition* π *of states* S *of* M*, is a collection of disjoint subsets of states whose set union is* S *and an ordering* $\prec_{i-1,i}$*. The order* $\prec_i$ *of states* $S \in \pi_i$ *is given by the order* $\prec_{i-1}$ *of* $S \in \pi_{i-1}$ *of the direct predecessor state.*

Let the CRFSM from Table 3(a), $\pi_1 = \{A/*, B/*, C/*, D/*\}$ and $I = 0$ then the ordered partition is $\pi_2 = \{D/1, A/0, C/0, B/1\}$. Using this notation it can be easily determined that the predecessor to $B/1$ was the state D.

Note that the original definition of state partition [38] breaks the states into groups such that each group contains states with same output. In the Definition 14 these groups are broken visually but still exists if grouped by variable values. Thus an unordered partition $\pi_2 = (AC)(BD)$.

Definition 15 (Ordered State Cover)**.** *An ordered cover is a collection ϕ of subsets of states S (with their associated outputs O) whose union is S, such that no subset is contained in another subset in the collection [11]. Additionally, the order of states in the cover is given $\prec_{i-1,i}$.*

Let the CRFSM from Table 3(b), $\pi = \{A/*, B/*, C/*, D/*\}$ and $I = 0$ then the ordered cover is $\phi = \{D/1, A/0, A/1, B/1\}$. Using this notation it can be easily determined that the predecessor to $B/1$ was the state $D/*$.

Definition 16 (Ordered Input Sequence Uncertainty)**.** *The uncertainty $\Delta(x)$ of an input sequence $x = x_1, \ldots, x_k$ of an RFSM is a cover ϕ of S where two distinct states $F(x_t, s_h) = s_i$ and $F(x_t, s_j) = s_k$ are ordered according to $\prec_{i-1,i}$.*

The input sequence uncertainty is highest when the machine is in an unknown state given by $\pi = \{A/*, B/*, C/*, D/*, \}$; all states are indistinguishable by their outputs. The lowest uncertainty is either when the cover of machine states is a singleton state or when each group of states is a singleton state such as $\pi = \{A/0, C/1\}$.

The uncertainty of input sequence can be complemented by the amount of information computed from various states present in the block.

Definition 17 (Information Content of Uncertainty)**.** *Information Content of Uncertainty is given by $E(\phi) = -\sum_i p_i log(p_i)$ where the p_i coefficient represents the multiplicity of the state s_i in the cover ϕ of the states resulting from an input value.*

For instance $E(\{A/*, B/*, C/*, D/*\}) = -\sum_4 0.25 * log(0.25) = 1.3863$ and $E(\{D/*, A/*, A/*, A/*\}) = -(0.25 * log(0.25) + \sum_3 0.75 * log(0.75)) = 0.9939$.

The difference between the information content and the uncertainty is: the uncertainty $\Delta(x)$ of input sequence x represents the partition/cover of states given their observable outputs and the information content $E(\phi)$ is given by the mixture of individual unique states (or states/output combinations) present in the partition or cover. For instance the lowest uncertainty $\Delta(x) = 0$ represents the fact that each output represents one unique state, independently of how many such states are in the block. The lowest possible information content $E(\phi) = 0$ is such ordered partition that contains exactly one state (i.e., $\{A/*, A/*, A/*, A/*\}$)

Let M_R be a RFSM (Definition 9) and $x = x_1, \ldots, x_j$, $s = s_1, \ldots, s_j$ and $o = o_1, \ldots, o_j$ be an input, state and output sequences, correspondingly, each of them of length j.

Definition 18 (Transition Cycle)**.** *A transition cycle $\tau_j = so|x$, is the sequence of length j of elements of state-output $so \in \theta$ combinations obtained as a result of applying the sequence x to a machine M_R in some initial state $\theta_1 = s_1$ and such that the $F_R(x_1 s_1) = F_R(x_j, s_j)$.*

The transition cycle defines the shortest sequence of input values x that would starting from arbitrary initial state s_i and ending in the same state s_i.

Definition 19 (Maximal Transition Cycle)**.** *If the transition cycle τ_i reaches all available states in S, then the transition cycle is called maximal and is denoted τ_{max}.*

For instance, the machine from Table 3(b) starting from $IS = 0/A$ has $\tau_4 = D/1B/1C/1A/1|0010$. Note that τ_4 is also a maximal transition cycle and thus for M_R, $\tau_4 = \tau_{max}$.

Theorem 1. *Any RFSM that does not possess τ_{max} must have p transition cycles such that $\cup_{l=1}^{p} \tau_l = \theta$ and $\sum_{l=1}^{p} |\tau_l| = |S|$ and $\cap_{l=1}^{p} \tau_l = \emptyset$.*

Proof. Every RFSM is specified by a set of unique mappings $\{I, S\} \rightarrow \{S', O\}$. Additionally the evolution operator Λ_C is a one-to-one bijection and thus every single $\{S', O\} \in \theta$ can be reached from exactly one distinct $\{I, S\}$. This implies that each state can be reached from at maximum $|I|$ different cycles. Therefore, any existing $\{S', O\}$ is accessed by at maximum one existing cycle in Λ_C. Consequently, at maximum there are $|O \times S \times I|/2$ independent cycles. The content of each transition cycle implies that $\sum_{l=1}^{p} |\tau_l| = |S|$ and $\cap_{l=1}^{p} \tau_l = \emptyset$. □

Definition 20 (Reversible Successor Tree (RST))**.** *A reversible successor tree is a rooted DAG $RST = \{V, E\}$ where: (1) each edge E corresponds to a transition between two states given an input I value, and (2) node V represents a block of states with the associated output [11].*

The RST of the FSM shown in Table 3(a) is shown in Figure 2. Notice that unlike in classical successor tree [11], the RST preserves the order of the states between two successor nodes as shown in Figure 2. For instance, at node 0 the state cover is $\phi = \{A/*, B/*, C/*, D/*\}$ because it represents the current state, no output was yet generated and therefore the machine could be in any possible state. For the input value of 0, node 1 transforms the changed ordered cover $\phi = \{D/1, A/0, C/0, B/1\}$. Thus a state with output at j^{th} position in a parent node will generate the state with output at the j^{th} position in every child node.

Definition 21 (Impulse Response (IR))**.** *Impulse response of a machine M is the vector of output values $v_i = \{o^i_{s_1}, \ldots, o^i_{s_n}\}$ for all possible states $s_j | s_j \in S$ generated to an input value x_i.*

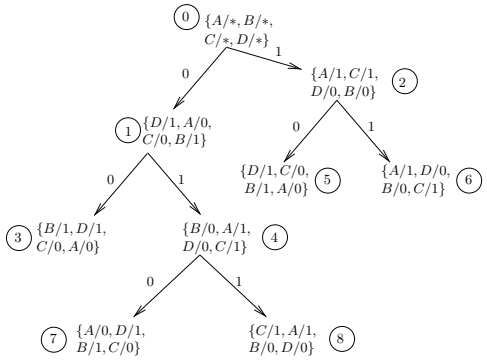

Fig. 2: Partially expanded successor tree of the NCRFSM defined in Table 3(a)

Definition 22 (Response Sequence (RS))**.** *For a CRFSM/NCRFSM M starting from a state s_i and an input sequence $x = x_1, \ldots, x_n$, the sequence of IRs $v_i^x = \{o_{s_i}^{x_1}, \ldots, o_{s_i}^{x_n}\}$ of obtained output values is called Response Sequence (RS).*

Definition 23 (Machine Signature)**.** *Combining IR and RS of a machine M into a matrix which columns are labeled by output values given a state $o_{S/O}$ and rows by input values from sequence $x = x_1, \ldots, x_n$ results in the so called machine signature ms.*

Example of machine signature ms of CRFSM from Table 3(b) obtained as a result of the input sequence $x = 00101$ is shown in Table 4. Columns in Table 4 represent, the index of the input variables, the input variable, the next state and output, and the impulse response respectively. For instance, in second row, $k = 0$ indicating the input variable is $x_0 = 0$. The third column indicates that given the state and output $A/*$, the next state output is $D/1$. The outputs generated for input x_0 are all gathered in $v_0 = 1011$.

Table 4: Response matrix of the CRFSM from Table 3(b)

k	x	$o_{A/*}$	$o_{B/*}$	$o_{C/*}$	$o_{D/*}$	ms
0	$x_0 = 0$	$D/1$	$A/0$	$A/1$	$B/1$	$v_0 = 1011$
1	$x_1 = 0$	$B/1$	$D/1$	$D/1$	$A/0$	$v_1 = 1110$
2	$x_2 = 1$	$C/1$	$B/0$	$B/0$	$C/0$	$v_2 = 1000$
3	$x_3 = 0$	$A/1$	$A/0$	$A/0$	$A/1$	$v_3 = 1001$
4	$x_4 = 1$	$C/0$	$C/0$	$C/0$	$C/0$	$v_4 = 0000$

3 SEQUENCES FOR THE ANALYSIS OF FSMS

In this section we present adapted definitions of testing sequences originally introduced for irreversible FSMs [38] to the models of NCRFSM and CRFSM introduced in this paper.

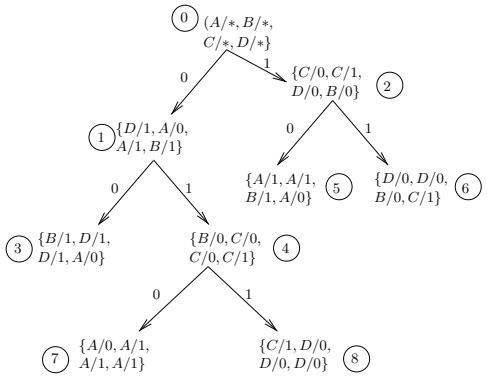

Fig. 3: Partially expanded successor tree of the CRFSM defined in Table 3(b)

3.1 Homing Sequence

Definition 24 (Homing Sequence)**.** *A homing sequence is a sequence of inputs x that independently on the initial state s_i and by observing the output sequence o allows to bring the machine to a distinct final state s_f.*

We look to Figure 2 for homing sequence. Starting from the initial cover ϕ (node 0) the input sequence 01 leads to B, A, D and C for $o = 10$, $o = 01$, $o = 00$ and $o = 11$ respectively.

To obtain a Homing sequence, start from the top node with index 0 and expand the node into successor nodes by analyzing the state change from each state in the current node for input value 1 and 0. The expansion stops when each state is a singleton or if all states in a node are the same.

3.2 Distinguishing Sequence

Definition 25 (Distinguishing Sequence)**.** *A sequence x of inputs that creates a unique sequence of outputs o starting from any unknown initial state s_j of the state machine. Such output sequence permits to determine the unknown initial state of the machine.*

Let's again use the successor tree from Fig. 2. The sequence $x = 01$ shows that for each of the possible current state a unique output sequence is be generated. Starting from node 0, the output sequences $o = 10$, $o = 01$, $o = 00$, and $o = 11$ for the initial states A, B, C and D respectively.

3.3 Synchronizing Sequence

Definition 26 (Synchronizing Sequence)**.** *The sequence x of inputs to create a path from any initial state s_i to the same specific final state s_f independently of the output sequence o. The synchronizing sequence is in fact a more powerful type of the homing sequence and thus if a sequence is synchronizing it is also a homing sequence.*

Let's consider the successor tree from Figure 3. Similarly to the homing sequence, starting from the root node (with index 0) and sequentially feeding the state machine a sequence of input values the machine will end up in the same state. For instance, the input sequence 010 leads the machine through the nodes 1, 4, and 5 and in node 5 all states are $A/0$. Thus independently of the output and of the initial state, the machine will be end up in the state A.

4 Analyzing Reversible Finite State Machines

CRFSMs and NCRFSMs used here are all reduced and thus do not contain any compatible states. Consequently, most of trivial models of state devices are not discussed.

4.1 NCRFSM

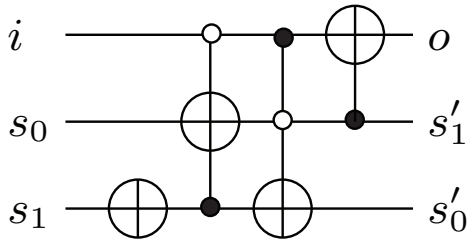

Fig. 4: Circuit realization of the NCRFSM defined in Table 3(a).

For the analysis of the NCRFSM we use the NCRFSM from Table 3(a) that has the RST shown in Figure 2. Figure 4 shows one possible realization of the

NCRFSM defined by Table 3(a) using the encoding of the states $A = (s_0 = 0, s_1 = 0)$, $B = (s_0 = 0, s_1 = 1)$, $C = (s_0 = 1, s_1 = 0)$ and $D = (s_0 = 1, s_1 = 1)$.

Table 5: Encoding of the NCRFSM from Table 3(a)

S		I					
		0			1		
s_0	s_1	s_0	s_1	o	s_0	s_1	o
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	1
1	0	1	0	0	1	1	0
1	1	0	1	1	0	1	0

Table 5 shows the individual bits for nest state and output assignments. Table 5 allows us to generate a set of equations describing the individual variable assignment:

$$s'_0 = \bar{s}_1 \oplus i\bar{s}_0 = \bar{s}_1 \oplus i\bar{s}'_1 \tag{4}$$

$$s'_1 = s_0 \oplus \bar{i}\bar{s}_1 \tag{5}$$

$$o = \bar{s}_0 \oplus \bar{i}s_1 = i \oplus s'_1 \tag{6}$$

However because we are dealing with reversible circuits, we cannot simply assign, but rather we have to use the equations 4∼6 to change one of the state, input or ancilla bit variables. Here we decide to use the following variable mapping: $i \to o$, $s_0 \to s'_1$ and $s_1 \to s'_0$ and we use one ancilla bit.

The permutative matrix representing Λ_N of the NCRFSM from Table 3(a) is shown in eq. (7). The variables indicated at the top of the matrix are the input I, the state S and the output O variable respectively.

$$\Lambda_N = \begin{array}{c} \\ A/0 \\ B/0 \\ C/0 \\ D/0 \\ A/1 \\ B/1 \\ C/1 \\ D/1 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0/A & 0/B & 0/C & 0/D & 1/A & 1/B & 1/C & 1/D \end{array} \\ \left(\begin{array}{cccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array} \tag{7}$$

Lemma 1. *An function Λ_N does not changes the information content of the input uncertainty in an NCRFSM.*

Proof. The information content of the input uncertainty $\Delta(x)$ is changed only when for a given input value x_j the number of distinct successor states is increased or decreased with respect to the number of distinct states in the predecessor state partition. The evolution function $\Lambda_N \equiv F \times G$ (Definition 10) preserves for each input value i the number of unique states. Therefore, starting from an initial partition $\pi = \{s_1/i_1, \ldots, s_1/i_k, \ldots, s_n/i_1, \ldots, s_n/i_k\}$ each vertex of the successor tree will lead to a partition that contains exactly the same states permuted according to function Λ_N. Consequently, there is no input sequence that would modify the information content $E(\phi)$ in state partitions of the successor tree of an NCRFSM. □

For instance, observe that every node in the RST shown in Figure 2 contains all states in every node of the tree.

Theorem 2. *An NCRFSM always possesses a homing sequence.*

Proof. The assignment of states and output values for each input using Λ_N means that all states of the NCRFSM appear at every node of the successor tree (lemma 1). Additionally, Λ_N for an altering sequence of input values such as $0, 1, 0, etc.$ at every node all available states will be having output 1 or 0. Finally, NCRFSM always contains a τ_{max}: there exists at least one sequence of inputs that will traverse the τ_{max} and thus generating a unique sequence for each initial state and consequently identifying the final state distinctively. □

Theorem 3. *An NCRFSM always possesses a distinguishing sequence.*

Proof. This is a direct consequence of Theorem 2. NCRFSM can always identify a final state by a unique output sequence. Λ_N is specified by a reversible matrix and Λ_N is a bijection. Starting from an arbitrary final state with an associated sequence of outputs will lead backward to a unique and distinctive initial state.

□

The successor tree shown in Figure 2 shows that one of the available homing sequences for this machine is 10, with output sequences $o = 11$, $o = 10$, $o = 01$ and $o = 00$, the resulting states are D, C, B and A respectively. The distinguishing sequence can be directly seen in the tree from Figure 2 because the input sequence 11 generates unique output sequences and thus confirms Theorems 2 and 3 (for the outputs refer to the state-output mapping shown in Table 3(b)).

Theorem 4. *An NCRFSM cannot possess a synchronizing sequence*

Proof. This is a natural consequence of Lemma 1: if an NCRFSM generates a balanced distribution of states and outputs and does not modify the information

content of input sequence uncertainty, it cannot converge to a single unique state. □

4.2 CRFSM

Now that we showed properties of the special case of the reversible NCRFSM, we extend these results to the general model of CRFSM. The CRFSM is a relaxed type of NCRFSM and can be obtained from NCRFSM by simply changing output states between different columns of the state transition function specified in a state-transition table. For instance the Table 3(b) shows a state transition function of a CRFSM that is obtained by changing the state assignment from the Table 3(a).

The matrix corresponding Λ_C to Table 3(b) is shown in Eq. (8).

$$\Lambda_C = \begin{array}{c} \\ A/0 \\ B/0 \\ C/0 \\ D/0 \\ A/1 \\ B/1 \\ C/1 \\ D/1 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0/A & 0/B & 0/C & 0/D & 1/A & 1/B & 1/C & 1/D \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{8}$$

For illustration the circuit realizing the CRFSM from Table 3(b) is shown in Figure 5. The encoding used for this realization is the same as in the case of the NCRFSM shown in Figure 4, which is $A = 0$ and $B = 1$. For the analysis of the

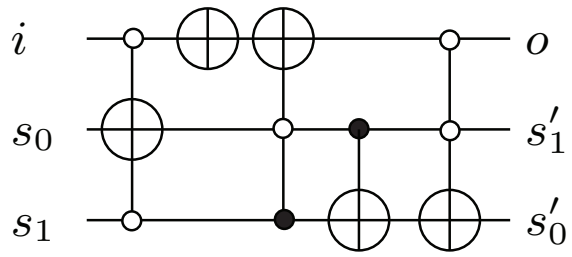

Fig. 5: Compact circuit realization of the CRFSM defined in Table 3(b).

CRFSM we use the CRFSM from Table 3(a) that has the RST shown in Figure 2. Figure 4 shows one possible realization of the CRFSM defined by Table 3(a) using the encoding of the states $A = (s_0 = 0, s_1 = 0)$, $B = (s_0 = 0, s_1 = 1)$, $C = (s_0 = 1, s_1 = 0)$ and $D = (s_0 = 1, s_1 = 1)$.

Again Table 6 shows the individual bits encoding. From Table 6 we can

Table 6: Encoding of the CRFSM from Table 3(b)

S		I					
		0			1		
s_0	s_1	s_0	s_1	o	s_0	s_1	o
0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	1
1	0	0	0	1	1	1	0
1	1	0	1	1	0	1	0

generate a set of equations describing the individual variable assignment:

$$s'_0 = i\bar{s}_0 \oplus i\bar{s}_1 \oplus \bar{s}_0\bar{s}_1 = s_1 \oplus s'_1 \oplus \bar{o}\bar{s}'_1 \tag{9}$$

$$s'_1 = s_0 \oplus \bar{i}\bar{s}_1 \tag{10}$$

$$o = \bar{i} \oplus s_0\bar{s}_1 = \bar{i} \oplus s_1\bar{s}_2 \tag{11}$$

Lemma 2. *Let Λ_C be defined by a reversible matrix (Definition 2), then Λ_C reduces the information content of input uncertainty in an CRFSM.*

Proof. A CRFSM defined by reversible Λ_C with the only restriction that for at least one $i_j s_k$ combination of the input and state values, the $\Lambda_C(i_j s_k)$ results in $s_n o_p$ such that $s_k = s_n$. This implies that for each step resulting by the application of Λ_C, at least one state will be assigned twice. This has for consequence that information content $E(\phi)$ reduces (Definition 17). □

Theorem 5. *A CRFSM always possesses a homing sequence*

Proof. The CRFSM specified by Λ_C is not guaranteed to possess τ_{max} and multiple cycles τ_j may exist. Because Λ_C is reversible and Λ_C reduces the information content, there is at least one input sequence x that leads to either a single final state (due to information reduction in the input uncertainty) or to a partition ϕ with distinct output sequences (due to reversibility of Λ_C). □

Before proceeding to the next step we introduce two sub-categories of CRFSMs: non-restricting CRFMS (NRCRFSM) and restricting CRFSM (RCRFSM). Consider the two CRFSMs shown in Table 7. The NRCRFSM from Table 7(a) is an example of machine that reduces the information content of the state partition. That is for a particular input value the $E(\pi_j) \leq E(\pi_{j+1})$. The RCRFSM from Table 7(b) is also a CRFSM because it reduces the information content of the initial unknown partition only once and then $E(pi_j) = E(pi_{j+1})$. Note that the RCRFSM from Table 7(b) is halfway between a CRFSM and NCRFSM: NCRFSM

Table 7: Example of (a) the NRCRFSM and (b) the RCRFSM

S	I	
	0	1
A	A/1	D/0
B	C/0	B/0
C	C/1	A/0
D	B/1	D/1

(a)

S	I	
	0	1
A	A/1	D/0
B	A/0	B/0
C	C/0	B/1
D	C/1	D/1

(b)

does not reduces $E(\pi_i)$ while RCRFSM reduces only $E(\pi_0)$ and then it behaves as NCRFSM.

Lemma 3. *A CRFSM always possess a distinguishing sequence.*

Proof. The proof is separated into two special cases:

1. if the CRFSM does not reduce the input sequence uncertainty information content then it is a NCRFSM.

2. if the CRFSM is RCRFSM, then once it reduced $E(\pi_0)$ is behaves like NCRFSM and therefore will also posses the distinguishing sequence

3. if the CRFSM is NRCRFSM, it reduces the input sequence uncertainty information. In order not to have a distinguishing sequence, for at least two initial states a and b any response sequences v_a^x and v_b^x must be exactly the same given any input sequence x. This is only possible (a) if two states lead to a same successor state in which case it is not an RFSM, (b) if $v_a^x = v_b^x$ for different state sequences s_a and s_b and for any input sequence x in such case the CRFSM is not properly reduced as it will contain redundant states.

□

Table 8: Example of the distinguishing sequence of the CRFSM from Table 3(b)

x	$A/*$	$B/*$	$C/*$	$D/*$	v_k
1	$C/0$	$C/1$	$D/0$	$B/0$	$v_k = 0100$
0	$A/1$	$A/1$	$B/1$	$A/0$	$v_k = 1110$
0	$D/1$	$D/1$	$A/0$	$D/1$	$v_k = 1101$
1	$B/0$	$B/0$	$C/0$	$B/0$	$v_k = 0000$

Lemma 4. *For a function Λ_C defined by a permutative matrix (Definition 2) that reduces the information content of input uncertainty for τ_j (NRCRFSM), there is at least one specific input sequence x_{red} that leads to a single final state s_f in τ_j.*

Proof. In order to prove Lemma 4 it is enough to show that starting from an arbitrary cover ϕ there is a sequence x_{red} that reduces the information content of the input sequence. In order to prove this, we assume that all states $s_j \in \pi$ can be reached within τ_j and Λ_C must either reduce or preserve the information content of π. Then the proof is shown using the following steps:

1. Let $\pi = \{\tau_0, \ldots, \tau_j\}$ be the initial ordered partition π and the initial input sequence uncertainty.
2. Let $i_j \in I$ be such that Λ_C reduces the information content of the input sequence uncertainty by performing mapping $M_R : \pi \xrightarrow{i_j} \phi$. This means $\exists s_o, s_p \in \phi | s_o = s_p$ (two states are equal in ϕ after applying Λ_C with i_j to π).
3. For any two states $s_j \neq s_k \in \phi$ there must be a sequence of inputs $x_{red,j,k} = \{i_0, \ldots, i_r\}$ such that $\Lambda_C(i_r, s_j, *) = \Lambda_C(i_j, s_k, *)$.

□

Theorem 6. *A RFSM possesses a synchronizing sequence if it reduces the information content of uncertainty and possesses maximal sequence τ_{max}.*

Proof. If an RFSM with evolution function Λ_C reduces the information of the input uncertainty, it can with finite sequence reach a common final state (Lemma 4). Additionally if the machine M_R contains τ_{max} this common final state is reachable from arbitrary initial state: If a CRFSM does not have τ_{max} then there are at least two smaller τ_j cycles (Theorem 1) that under any input sequence will end up in at least two different states. If a CRFSM is RCRFSM it can still have τ_{max} but it will never converge to a single state. □

Table 10(a) and 10(b) show an example of two RFSMs without and with synchronizing sequences respectively.

Note that one can observe the existence of the synchronizing sequence very quickly. The CRFSM from Table 10(a) is an RCRFSM and thus from an initial partition $\pi_1 = \{A, B, C, D, E, F\}$ it reduces the partition information content to $-\sum_{i=1}^{6} \frac{1}{3} log(\frac{1}{3})$. Then any next step will preserve the information content the same. Therefore, the RCRFSM cannot have a synchronizing sequence.

Table 9: Example of (a) RCRFSM without synchronizing and (b) NRCRFSM with a synchronizing sequence 001011 leading to a single state B.

S	I	
	0	1
A	A/0	E/1
B	F/0	B/1
C	C/1	D/1
D	A/1	D/0
E	C/0	E/0
F	F/1	B/0

(a)

S	I	
	0	1
A	A/0	F/1
B	F/0	C/0
C	D/1	B/0
D	E/0	C/1
E	D/0	E/1
F	A/1	B/1

(b)

CRFSM from Table 10(b) has a duplicate state in both columns; for $I = 0$ $\{A\}$, $\{F\}$ and $\{C\}$, $\{E\}$ lead to same state A and D respectively. For $I = 1$ $\{B\}$, $\{D\}$ and $\{C\}$, $\{F\}$ provide the information content reduction necessary for possessing a synchronizing sequence (Theorem 6).

Table 10: Example of a synchronizing input sequence in CRFSM from Table 3(b). Notice that every column ends with the same state and thus shows the existence of the synchronizing sequence 1101.

x	$A/*$	$B/*$	$C/*$	$D/*$
1	C	C	D	B
1	D	D	B	C
0	B	B	A	A
1	C	C	C	C

Conjecture 1. *If a CRFSM possesses a distinguishing (synchronizing) sequence that contains all possible input values it cannot at the same time possess the synchronizing (distinguishing) sequence as well.*

Proof. If an CRFSM possesses a distinguishing sequence it means that in each column of the RST there exists only unique combinations of $S \times O$. If that would not be the case, some of the columns in the RST of the RFSM would be reducing the input uncertainty. This also means that there exist at least two output sequences that have identical outputs and thus a distinguishing sequence cannot exist. However, if the distinguishing sequence exists, it means that each column does not reduce the state uncertainty and each element of $S \times O$ is present only once in each column. In such a case the RFSM cannot have the synchronizing sequence. □

5 CONCLUSIONS

In this paper we presented an analysis of the reversible finite state machines and proved a set of exact results concerning RFSMs being tested using the three testing sequences. We proved that because of the reversible requirement the NCRFSMs cannot have the synchronizing sequence but have a homing and distinguishing sequences. We also proved that the CRFSM allows to have all three sequences but at a cost of longer input sequences and we formulated precise conditions under which an CRFSM can have all three studied sequences. The obtained results show that there is a hierarchy of testability between the NCRFSM and CRFSM and the used methods can be used to study other FSM.

In the future works, we extend this work into a complete method for transforming irreversible FSM to RFSM and give exact cost of ancilla bits required to have RFSM with all sequences.

REFERENCES

[1] E. F. Moore, "Gedanken-experiments on sequential machines," vol. 34, pp. 129–153.

[2] A. Gill, *Introduction to the Theory of Finite-state Machines*. New York: McGraw-HillQ.

[3] F. Hennie, "Fault detecting experiments for sequential circuits," in *In Proceedings of the Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, pp. 95–110.

[4] I. Kohavi and Z. Z. Kohavi, "Variable-length distinguishing sequences and their application to the design of fault-detection experiments," vol. C-17, pp. 792–795.

[5] A. D. Friedman and P. R. Menon, *Fault Detection in Digital Circuits*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

[6] E. P. Hsieh, "Checking experiments for sequential machines," vol. C-20, no. 10, pp. 1152–1166.

[7] M. N. Sokolovskii, "Diagnostic experiments with automata," vol. 6, pp. 44–49.

[8] S. M. Gobershtein, "Check words for the states of a finite automaton," vol. 1, pp. 46–49.

[9] M. Chen, Y. Choi, and A. Kershenbaum, "Approaches utilizing segment overlap to minimize test sequences," in *In Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification*, pp. 67–84.

[10] T.-F. Lee, A. C.-H. Wu, and Y.-L. Lin, "A transformation-based method for loop folding," vol. 13, no. 4, pp. 439–.

[11] Z. Kohavi and N. Jha, *Switching and Finite Automata Theory, 3rd edition*. Cambridge University Press.

[12] I. Pomeranz and S. Reddy, "Application of homing sequences to synchronous sequential circuit testing," in *Test Symposium, 1993., Proceedings of the Second Asian*, pp. 324–329.

[13] R. L. Rivest and R. E. Schapire, "Inference of finite automata using homing sequences," in *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '89. New York, NY, USA: ACM, pp. 411–420.

[14] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," vol. 55, no. 1, pp. 119 – 139. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002200009791504X

[15] M.-l. Chuang and C.-y. Wang, "Synthesis of Reversible Sequential Elements *," pp. 420–425.

[16] M. Ueda, "Optimization of reversible sequential circuits," vol. 2, no. 6, pp. 208–214, 2010.

[17] V. K. Siva Kumar Sastry Hari, Shyam Shroff, Sk. Noor Mahammad and E. Group, "Efficient Building Blocks for Reversible Sequential Circuit Design," in *49th IEEE International Midwest Symposium on Circuits and Systems*, 2006, pp. 437–441.

[18] S. Dhaarinee and N. Rajeswaran, "Implementation of Reversible Sequential Circuits Using Conservative Logic Gates," vol. 3, no. 5, pp. 500–505, 2014.

[19] D. S. Shubham Gupta, Vishal Pareek, "Low Cost Design of Sequential Reversible Counters," *INTERNATIONAL JOURNAL OF SCIENTIFIC & ENGINEERING RESEARCH,*, vol. 4, no. 11, pp. 1234–1240, 2013.

[20] M. Lukac, M. Perkowski, and M. Kameyama, "Quantum finite state machines - a circuit based approach," *International Journal of Unconvetional Computing*, vol. 9, no. 3-4, pp. 267–301.

[21] V. Singh and A. Sharma, "Implementation of Sequential Circuit using Reversible Fredkin gate on FPGA," *International Research Journal of Engineering and Technology (IRJET)*, vol. 03, no. 08, pp. 1873–1878, 2016.

[22] H. Thapliyal, S. Member, and N. Ranganathan, "Design of Testable Reversible Sequential Circuits," vol. 21, no. 7, pp. 1201–1209, 2013.

[23] N. Kumar, S. Wairya, and B. Sen, "Design of conservative , reversible sequential logic for cost efficient emerging nano circuits with enhanced testability," *Ain Shams Engineering Journal*, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.asej.2017.02.005

[24] M. Mohammadi, M. Eshghi, and M. Haghparast, "On Design of Multiple-Valued Sequential Reversible Circuits for Nanotechnology Based Systems," in *TENCON*, 2008, pp. 1–6.

[25] J. Pin, "On the languages accepted by finite reversible automata," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, T. Ottmann, Ed. Springer Berlin Heidelberg, vol. 267, pp. 237–249.

[26] J.-E. Pin, "On reversible automata," in *LATIN '92*, ser. Lecture Notes in Computer Science, I. Simon, Ed. Springer Berlin Heidelberg, vol. 583, pp. 401–416.

[27] R. Ali, M. Gooding, T. Szilagyi, B. Vojnovic, M. Christlieb, and M. Brady, "Automatic segmentation of adherent biological cell boundaries and nuclei from brightfield microscopy images," vol. 23, no. 4, pp. 607–621, 2011.

[28] M. Soeken, R. Wille, C. Otterstedt, and R. Drechsler, "A Synthesis Flow for Sequential Reversible Circuits," in *2012 IEEE 42nd International Symposium on Multiple-Valued Logic*, may 2012, pp. 299–304.

[29] M. H. A. Khan, "Design of Reversible Synchronous Sequential Circuits Using Pseudo Reed-Muller Expressions," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 11, pp. 2278–2286, nov 2014.

[30] S. Gupta, "Synthesis of Sequential Reversible Circuits through Finite State Machine," *CoRR*, vol. abs/1410.2, 2014. [Online]. Available: http://arxiv.org/abs/1410.2370

[31] N. Margolus, "Physics-like models of computation," vol. 10, no. 1-2, pp. 81 – 95. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167278984902525

[32] G. Y. Vichniac, "Simulating physics with cellular automata," vol. 10, no. 112, pp. 96 – 116. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167278984902537

[33] K. Morita, *Handbook of Natural Computing*. Springer Berlin Heilderberg, ch. Reversible Cellular Automata, pp. 231–257.

[34] ——, "Reversible computing and cellular automata a review," vol. 395, pp. 101–131.

[35] ——, "Reversible computing systems, logic circuits and cellular automata," in *Proceedings of the 3rd International Conference on Networking and Computing*, pp. 1–8.

[36] S. Wolfram, *A New Kind of Science*. Wolfram Media Inc.

[37] B. Anuradha and S. Sivakumar, "A Fault Analysis in Reversible Sequential Circuits," *IOSR Journal of VLSI and Signal Processing*, vol. 4, no. 2, pp. 36–42, 2014.

[38] Z. Kohavi, *Switching and Finite Automata Theory*. Mc Graw-Hill.