4

Conservative Reversible Elementary Cellular Automata and their Quantum Computations

Anas N. Al-Rabadi
Computer Engineering Department, The University of Jordan
The Office of Graduate Studies and Research (OGSR), Portland State University

1. Introduction

Cellular automata (CA) are discrete spatially extended dynamical systems that are used as models of physical processes and as computational devices. An Elementary Cellular Automaton (ECA) is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules which are based on the states of neighboring cells and the previous state of the evolved cell. The next state of any cell in ECA depends only upon its present "neighborhood," which includes the state of the cell itself and those of its immediate neighbors to the left and right. The rules are then applied iteratively for as many time steps as desired. ECA model has been used as a "universal constructor" and were studied as one possible model for natural and engineered systems where comprehensive studies of CA have been performed in which a gigantic collection of results and discoveries concerning automata are presented [4,5,11,21,22,25,26,28,30,37,38,41,45,49,50,51,53,60,62,71,74,78,79,80,81,83,86,87,92,93,94,97,99, 100,106,107,108,109,110,111,112,113,114,115]. Several types of CA have been used extensively in several science and engineering applications such as: Image Processing [69,72], Finite State Machine (FSM) and VLSI circuit testing [2,13,17,19,20,29,30,32,46,48,56,75, 88,89,90,91,96,103], Encryption [39,59,61,85], Pattern Classification and Recognition [15,26,51,52,57,73, 77,95,104], Error Correcting Codes [16,68], Neural Networks [7,18,105], Fault Diagnosis [70,91], Fault Tolerance [65], Data Compression [39,69], Game Theory [37], Random Number Generator (RNG) [102], Number Sorting [64], and set-theoretic Reconstructability Analysis (RA) [117].

Motivations for pursuing the possibility of implementing ECA using Reversible Logic (RL) would include items such as [1,3,4,9,23,31,40,47,54,55,63,76,81,101]: (1) power: the fact that, theoretically, the internal computations in hardware RL-based systems consume no power. It was proven in [40], it is a necessary (but not sufficient) condition for not dissipating power in any physical circuit that all system circuits must be built using fully reversible logical components. For this reason, different technologies have been studied to implement reversible logic in hardware such as adiabatic CMOS and quantum [3,63,76]. Fully reversible discrete systems will greatly reduce the power consumption (theoretically eliminate) through three conditions: (a) logical reversibility: the vector of input states can always be uniquely reconstructed from the vector of output states, (b) physical reversibility: the physical switch operates backwards as well as forwards, and (c) the use of "ideal-like" switches that have no parasitic resistances; (2) size: since the newly emerging quantum

computing technology must be reversible [3,40,63], the current trends related to more dense hardware implementations are heading towards 1 Angstrom, at which quantum mechanical effects have to be accounted for; (3) speed: if the properties of superposition and entanglement of quantum mechanics can be usefully employed in the reversible ECA (RECA) context, significant computational speed enhancements can be expected; and (4) mapping: since RECA produces new constraint types of maps, it is interesting to explore the new dynamics and advantages of modeling discrete systems using such RL maps in contrast to the classical irreversible maps.

This research extends and implements several of the reversible and quantum computing concepts that were developed earlier [3] to the context of ECA and this includes a new method for modeling and processing via the reversibility property in the existence of noise. The main contribution of this research is the creation of a new algorithm that can be used in noisy discrete systems modeling using conservative reversible elementary cellular automata (CRECA) and the corresponding quantum modeling of such discrete systems. One of the advantages of the use of new families of CRECA is their potential utilization in reconfigurable low-power (adiabatic) VLSI circuit designs [76] in contrast to the role of classical ECA circuits in classical (non-adiabatic) VLSI systems. M-ary quantum representations in ECA of: (1) *m*-ary orthonormal computational basis states quantum decision trees (QDTs) and (2) *m*-ary orthonormal composite basis states QDTs are also introduced as possible quantum representations for the modeling and manipulation of the quantum ECA (QECA) dynamics.

Although several approaches were introduced previously in dealing with reversible ECA [23,31,33,34,35, 36,47,54,55], none of these approaches considered the important modeling and processing case which uses Swap-based operations (primitives) to represent reversible ECA even in the presence of noise. Modeling using Swap-based circuits is important since many of the two-valued (binary) and many-valued (*m*-ary) quantum circuit implementations use two-valued and many-valued quantum *Swap*-based and *Not*-based gates. This can be important, since the Swap and Not gates are basic primitives in quantum computing, from which many other *m*-valued fundamental gates are built such as [3,63]: (1) *m*-valued Not gate, (2) *m*-valued Controlled-Not gate (C-Not gate or Feynman gate), (3) *m*-valued Controlled-Controlled-Not gate (C^2-Not gate or Toffoli gate), (4) *m*-valued Swap gate, and (5) *m*-valued Controlled-Swap gate (C-Swap gate or Fredkin gate).

The remainder of this research is organized as follows: basic backgrounds on RL and ECA are given in Sections 2 and 3. An algorithm for the synthesis of Swap-based CRECA is presented in Section 4. Quantum computation (QC) of CRECA is presented in Section 5. The extension to the *m*-ary (*m*-valued) case is introduced in Section 6. Conclusions and future work are presented in Section 7.

2. Fundamentals of reversible logic

A (*k*, *k*) reversible circuit is a circuit that has the same number of inputs (*k*) and outputs (*k*) and is a one-to-one mapping between the vectors of inputs and the vectors of outputs, thus the vector of input states can be always uniquely reconstructed from the vector of output states [3,9,40,63,101]. Thus, a (*k*, *k*) *reversible* map is a *bijective* function which is both injective ("one-to-one") and surjective ("onto"). The *auxiliary* outputs and inputs that are needed *only* for the purpose of reversibility are called "garbage" outputs and "garbage" inputs respectively. These are auxiliary outputs and inputs from which a reversible map is

constructed (cf. Table 1 in Example 1). If a circuit is composed of interconnecting reversible elements (primitives or gates) then the overall circuit is also reversible [3].
A (k, k) conservative circuit has the same number of inputs (k) and outputs (k) and has the same number of values in inputs and outputs (e.g., the same number of ones in inputs and outputs for binary, the same number of ones and twos in inputs and outputs for ternary, etc).
An algorithm called conservative reversible Boolean function (CRBF) that produces a conservative reversible Boolean map from its irreversible counterpart is as follows:

Algorithm CRBF

1. Augment the number of outputs to become equal the number of inputs: add a sufficient number of auxiliary output variables such that the number of outputs equals the number of inputs. Allocate a new column in the mapping table for each auxiliary variable.
2. For the construction of the first auxiliary output, assign a constant C_1 to half of the cells in the corresponding table column (e.g., zeros), and the second half as another constant C_2 (e.g., ones). For convenience, one may assign C_1 to the first half of the column, and C_2 to the second half of the column (cf. Table 1, column Y_1).
3. For the next auxiliary output, **If** non-reversibility still exists, **Then** assign for *identical* output tuples (irreversible map entries) values which are half zeros and half ones (cf. Table 1, bottom two entries of column Y_2), and then assign a constant for the remainder that are already reversible (cf. first two entries of Y_2).
4. **If** number of inputs i is now less than number of outputs j, **Then** add new auxiliary inputs ($j - i$) with constant column entries that will retain conservativeness (cf. Table 1, constant "0" as first two entries of column c and constant "1" as bottom two entries of column c).
5. **Goto** steps 3 and 4 until the total map entries are reversible and conservative.

Example 1. The standard two-variable Boolean OR: $Y = a + b$ is irreversible as in the following map:

a	b	Y
0	0	0
0	1	1
1	0	1
1	1	1

Following the above CRBF algorithm, the following is one possible reversible two-variable Boolean OR:

c	a	b	Y	Y_1	Y_2
0	0	0	0	**0**	**0**
0	0	1	1	**0**	**0**
1	1	0	1	**1**	**0**
1	1	1	1	**1**	**1**

Table 1. A (3, 3) conservative reversible map for the Boolean inclusive OR.

Using CRBF algorithm, the construction of the conservative reversible map in Table 1 is obtained as follows: Since Y is irreversible, assign garbage output Y_1 and assign the first half of its values as constant "0" and the second half as another constant "1". Since the new map is still irreversible, assign garbage output Y_2 and assign the 3rd cell value to constant "0" and the 4th cell value to constant "1". Since by now the new map is a reversible 2-input 3-output non-conservative map (i.e., Boolean (2, 3) non-conservative OR), add a new garbage input c that converts the map to be a Boolean (3, 3) conservative OR. One notes that the solution in Example 1 is not unique, i.e., several other conservative reversible maps can be obtained as well.

As data in real life situations can be exposed to noise sources: (1) environmental (external) noise, (2) structural (internal) noise, (3) interfacing (measurement; data acquisition) noise, (4) a system output state that will never occur (i.e., don't care state), or (5) unknown data (undecided), it is important to produce a conservative reversible mapping method that could incorporate noise in input or/and output data. This problem can be stated as: *given that certain cells in a map are acceptable to be noisy (i.e., can take values* "0" *or* "1"*), generate a conservative reversible map from the corresponding irreversible counterpart*.

An observation that can be noted is that if noise occurs, value "0" can turn to value "1" or value "1" can turn to value "0", and thus the reversible map obtained can become irreversible. One method to solve this problem is by adding *redundant* input/output variables that will maintain (1) reversibility and (2) conservativeness even with the existence of noise.

Example 2. Let us assume in a simplified noise situation that input data in Table 1 has been corrupted with noise as follows: $\{Y=1,Y_1=0,Y_2=0\}\rightarrow\{Y=1,Y_1=1,Y_2=0\}$. One can note that the map in Table 1 becomes irreversible since two input sets {c=0,a=0,b=1} and {c=1,a=1,b=0} has the same output $\{Y=1,Y_1=1,Y_2=0\}$. To obtain a reversible and conservative map while incorporating noise, one may add another *redundant* output Y_3 and input *k* (cf. CRBF):

k	**c**	**a**	**b**	**Y**	**Y_1**	**Y_2**	**Y_3**
1	0	0	0	0	0	0	**1**
1	0	0	1	1	**0→1**	0	**0**
1	1	1	0	1	1	0	**1**
1	1	1	1	1	1	1	**1**

The previously illustrated procedure can handle as many cell changes that have altered values due to noise. In the most straightforward case a redundant variable is added for each noisy cell as an upper extreme. Yet, if the number of noisy cells is *N* and the number of needed added cells (to maintain reversibility) is *A*, then in general one may need $A \leq N$ iff the bijective mapping is still maintained. To achieve this, a simple exhaustive search algorithm using the CRBF algorithm over groups of cells may be utilized to obtain the constraint {*min* (*A*) *for max* (*N*)}, and this depends on the *distribution (i.e., type) of noise* that affects the map.

Example 3. The following map demonstrates an example for the use of the CRBF algorithm for achieving reversibility in the simultaneous presence of three erroneous cells.

k	c	a	b	Y	Y_1	Y_2	Y_3
1	0	0	0	0	0	0	**1**
1	0	0	1	1	**0→1**	0	**0**
1	1	1	0	1	**1→0**	0	**1**
1	1	1	1	1	1	**1→0**	**1**

As mentioned previously, in general, the errors (i.e., changes) in the cells may follow a pattern of certain noise distribution or may not. The important issue of noise distribution, and its direct effect on the method of selecting the needed added auxiliary variables, is to be addressed in a future publication.

3. Basics of elementary cellular automata

A cellular automaton is a decentralized computing model that provides an excellent platform for performing complex computations with the help of only local information [14,20,44,48,59,62,93,99,111,112,113,115]. A CA consists of a spatial *lattice* of cells, each of which, at time *t*, can be in one *m* states.

The lattice starts out with some initial configuration of local states (where configuration means the pattern of states over the entire lattice) and at each time step, the states of all cells in the lattice are synchronously updated [84]. We denote the lattice size (or number of cells) as *d*. For a 2-valued (binary) finite lattices, there is only a finite number of 2^d of possible configurations. In one-dimensional CA, the neighborhood of a cell includes the cell itself and some number of neighbors on either side of the cell. The number of neighbors on either side of the center cell is referred to as the CA's radius *r*. For example, an elementary one-dimensional CA is of radius $r = 1$. The motion equations for a CA are often expressed in the form of a *rule table*, which is a look-up table listing each of the neighborhood patterns and the state to which the central cell in that neighborhood is mapped [113].

The communication in CA between constituent cells is limited to local interaction, and the overall structure can be viewed as a parallel processing device. CA exhibits the same dynamics behaviors (including fractals and chaos) which is seen in systems of continuous differential equations [25,42,49,66,98,106]. Elementary Cellular Automata (ECA) as a special type of CA has been proven to be a powerful computing paradigm for the following properties [62,113,115]: (1) universality: an ECA can model any discrete system, and thus it is a powerful logically *complete* system from which all functions can be obtained and can be used to model complex systems; (2) simplicity: an elementary cellular automaton is the building block of the elementary cellular automata, which is a simple structure that evolves over time using specific evolution rules, that leads to modeling complex discrete systems using such simple structures; and (3) regularity: the evolution of ECA to generate discrete time systems consists of geometrically evolving cellular grids. Set-theoretically, a regular ECA rule is a mapping of $(s_t(i-1) \otimes s_t(i) \otimes s_t(i+1))$ onto $s_{t+1}(i)$, where $\otimes$ is the Cartesian product.

An ECA consists of an array of cells in one dimension (1-D). In a Boolean ECA, each cell can take on one of two state {0, 1} where the binary string representing the array changes at discrete time steps (intervals). The Boolean ECA dynamics is commonly represented: (1) *spatially*: by a horizontal sequence of 0's and 1's or of white and black cells, and (2)

temporally: time, in successive rows, is the vertical axis. The next state of any cell in ECA depends only upon its present "neighborhood," which includes (a) the state of the cell itself and (b) those of its immediate neighbors to the left and right. That is, if $s_t(i)$ is the state of cell i at time t, the dynamic law governing the Boolean ECA is described by the Boolean mapping (function):

$$s_{t+1}(i) = f(s_t(i-1), s_t(i), s_t(i+1)) \quad (1)$$

Since there are $2^3 = 8$ possible neighborhoods and since each neighborhood can map into either of the two states of $s_t(i+1)$, then there are $2^8 = 256$ mappings (or ECA rules).
Example 4. Table 2 shows the binary string representation of ECA Rule #30.

$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i)$
0	0	0	0→Automaton1
0	0	1	1→Automaton2
0	1	0	1→Automaton3
0	1	1	1→Automaton4
1	0	0	1→Automaton5
1	0	1	0→Automaton6
1	1	0	0→Automaton7
1	1	1	0→Automaton8

Rule

Table 2. An illustrative example of ECA rule (Rule #30).

Figure 1 shows the dark and white cells as a second representation of ECA Rule #30.

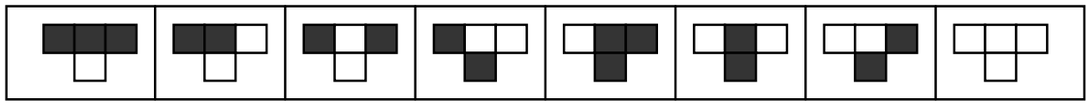


Fig. 1. A second ECA representation of Rule #30.

The 256 possible mappings are indexed by the binary number defined by $s_{t+1}(i)$ for the set of all neighborhoods, where the lowest order bit of this index is f (0,0,0) and the highest order bit is f (1,1,1). For example, by using binary number expansion over base 2, the mapping of Table 2 is indexed by the number $(00011110)_2$ and is Rule #30.
A discrete dynamics using Rule #30 can be shown by evolving specific length grid l for n steps starting from an initial condition. An example can be shown from [113] as in Figure 2a where the total evolution occurs after 15 steps starting from a centered single black cell. The temporal behavior in a classical ECA, as shown in Figure 2a, is generally performed using *overlapping* neighborhoods for obtaining the automata evolution, and (as stated previously) from the algebraic mathematical modeling perspective, an overlapping-based ECA evolution using Equation (1) is a lattice.
As shown in [111,112,113], the 256 mappings are divided into 88 equivalence classes, given that one considers maps to be equivalent if they are related: (1) by *reflection* , i.e., by left-right inversion of their arguments (which, if the dynamics were shown on transparency, would

merely involve turning the transparency over), (2) by *complementing*, i.e., negating the arguments and the function (which merely produces a photographic negative reversal), or (3) by *both reflection and complementing*. In general, an equivalence class will have 4 members, but *f* may generate itself under reflection and/or complementing, so that an equivalence class may have 1, 2, or 4 members. A representative rule that is chosen consistently is used to label the CA classes.

Other ECA *complexities* of discrete dynamics also have been reported, and efforts have been undertaken to characterize the CA rule space, which is trying to understand and characterize the global dynamics from the local CA rules. CA evolution can lead to various dynamics that exhibits emergent behaviors such as fractals (Sierpinski triangle or Sierpinski gasket in Figure 2c), chaos (Figure 2a), randomness, complexity, and particles [113]. Additive CA (Figures 2b, 2c, 2d, 2f, and 2g) and linear CA gained popularity in the VLSI field [6,12,14,20,48,58,59] due to local interaction of simple cells, each having two states "0" or "1" which are the elements of the second radix Galois field GF(2) [3,67]. Also, it has been shown that Rule #110 (as shown in Figure 2e), like the game of life, which is an extremely simple one-dimensional system and one which is difficult to engineer to perform specific behavior, is universal [113]. From applications point of view, for instance, Rule #30 which generates randomness has been proposed to be used for pseudo-random number generator (PRNG) and as a possible stream cipher for the use in cryptography [39,59,61,85].

One can note that the characteristic features of an ECA are: (1) size: (1a) spatial size *l* and (1b) temporal size *n*, (2) initial condition, (3) evolution rule, (4) number of cell's states (colors), (5) number of neighbors, and (6) evolution strategy (i.e., the way one conducts evolution such as top-to-down and left-to-right, top-to-down and center-to-sides, etc). For the ECA evolution using overlapping neighbors, the same evolution result is obtained when one uses the same rule and the same initial condition, regardless of the evolution strategy used. Thus, one can note that for example for the same evolution rule and different initial conditions one would obtain distinct automata evolutions, and equivalently for different evolution rules and same initial conditions one would obtain distinct automata evolutions.

The CA local rules applied to each cell can be either identical or different. These two different possibilities are termed as uniform and hybrid CA, respectively [12,32,58,92]. While the next state function (rule) in general is deterministic, there are variations in which the rule sets are probabilistic [8,10,27] or fuzzy [24]. CA rules can be time-independent rules or time-dependent rules in which alternate rules at different time steps are used.

As mentioned previously, the ECA is divided into 88 classes where the dynamics of these classes are governed by different attractors [113]. Several approaches have been investigated for the automatic classification of CA [116]. In [113], four attractor types are identified: (1) homogeneous (where the dynamics settles (relaxes) to a fixed configuration which is uniform that consists of all 1's or 0's), (2) Fixed point or periodic (but not uniform), (3) chaotic, or (4) complex. An alternate classification has been also provided [43]: (a) null, (b) fixed point, (c) periodic, (d) locally chaotic (chaotic in some parts of the cell array but regular in other parts), and (e) chaotic. Several approaches were proposed to characterize the average behavior of rules passing from one CA regime (class) to another (e.g., fixed point $\rightarrow$ periodic $\rightarrow$ complex $\rightarrow$ chaotic) [42].

CA are also studied as computational devices, both as theoretical tools and as practical highly efficient parallel machines. Computing in the CA context may mean: (a) CA does a useful computational task where the rule is interpreted as a "program", the initial configuration is the "input", and the CA runs for specific number of time steps or until it

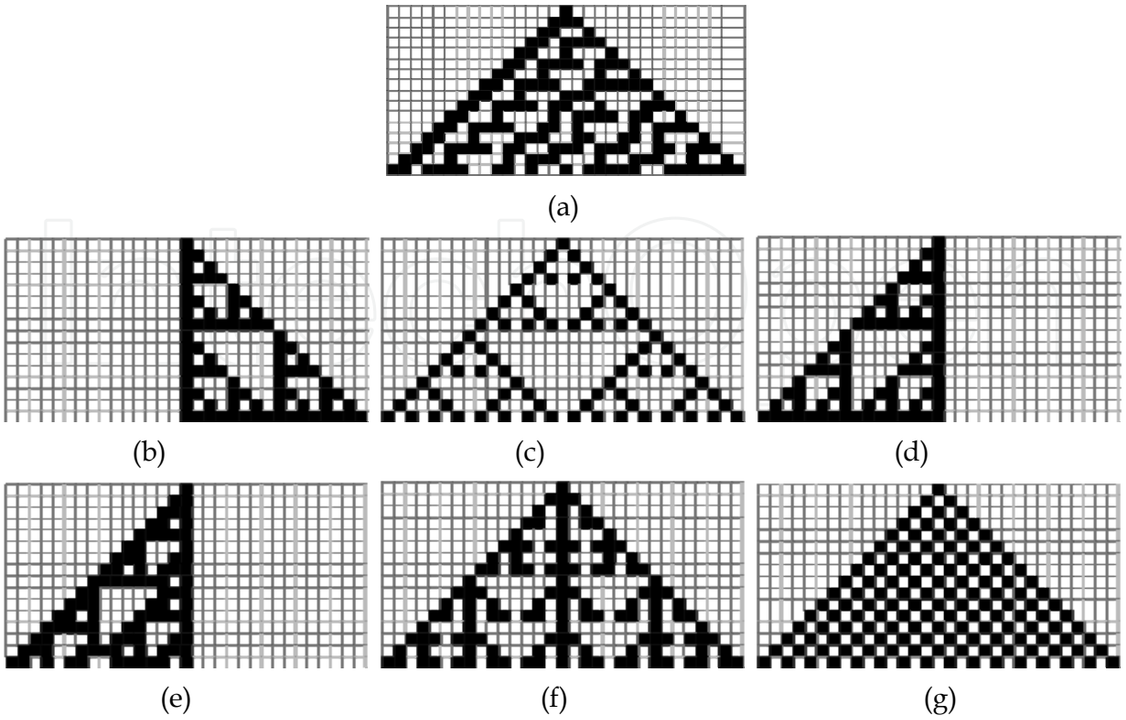


Fig. 2. Space-time diagram of the dynamical ECA which is temporally discrete and spatially discrete that can possess the following evolution dynamics where each row is an evolved Automata: (a) Rule #30, (b) Rule #60, (c) Rule #90, (d) Rule #102, (e) Rule #110, (f) Rule #150, and (g) Rule #250. All of the shown evolutions have started from an initial condition of a single black cell.

reaches a final goal "output" pattern (e.g., CA that performs image processing tasks [69,72]), or (b) a CA is capable of universal computing given specific initial configuration, which means (given the correct initial configuration) a CA can simulate a programmable computer, complete with logic gates, timing devices, etc (e.g., Conway's Game of Life, and the speculation that all Class 4 rules have the capacity for universal computing [113]).

Since real-life data is in general many-valued, the general case of many-valued ECA is used to model natural systems and their dynamics [113]. The following example shows an example of a ternary ECA.

Example 5. Discrete dynamics using ternary (i.e., 3-valued) Rule #322 in Figure 3a can be shown by the overlapping-neighborhood evolution of the grid of length 9 for 5 steps starting from an initial condition as shown in Figure 3b, where color-based values are as follows: "white" represents value (state) "0", "grey" represents value (state) "1", and "black" represents value (state) "2".

Note that the number of colors (values) allowed for the input cells defines the input radix (m_i) and the number of colors (values) allowed in the output cell defines the output radix (m_o). For the previous cases in Examples 4 and 5, the input radix equals the output radix since they use the same number of colors (i.e., values), and for this case the number represented by the ECA rule can be expressed by the radix m number expansion $N = \sum_{n=0}^{k-1} c_n m^n$, where c_n is the expansion coefficient which takes the value (color) of the

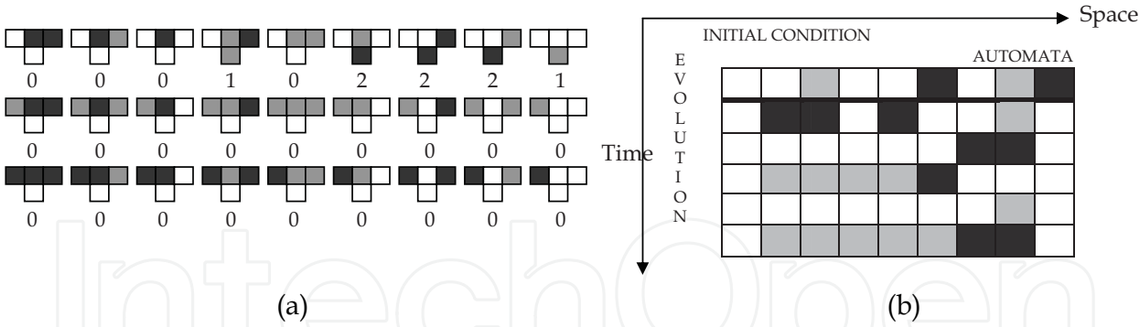


Fig. 3. Three-valued ECA evolution: (a) Rule #322 which is obtained as: $1\cdot3^0 + 2\cdot3^1 + 2\cdot3^2 + 2\cdot3^3 + 1\cdot3^5 = 322$, and (b) ECA evolution using Rule #322 for the given initial condition.

output of the automaton at index n for number of automata $k = m^3$, and the index n is computed using the number expansion over the input colors (values or states).

Other important types of CA rules were proposed such as the Gacs-Kurdyumov-Levin (GKL) CA in which the evolution rule is the "majority vote" rule with $m = 2$, $r = 3$ as follows:

$$\begin{cases} If \quad s_i(t) = 0, then \quad s_i(t+1) = majority[s_i(t), s_{i-1}(t), s_{i-3}(t)] \\ If \quad s_i(t) = 1, then \quad s_i(t+1) = majority[s_i(t), s_{i+1}(t), s_{i+3}(t)] \end{cases}$$

where $s_i(t)$ is the state of site i at time t. The GKL rule states that for each neighbor for seven adjacent cells (that is $r = 3$), if the state of the central cell is "0", then its new state is decided by a majority vote among itself, its left neighbor, and the cell two cells to the left away, else if the state of the central cell is "1", then its new state is decided by a majority vote among itself, its right neighbor, and the cell two cells to the right away.

4. The synthesis of conservative reversible ECA

The evolution that results from Equation (1) is an irreversible evolution; the evolution of ECA according to Equation (1) in general leads to *irreversible dynamics*, i.e., result is not a one-to-one mappings between vectors of inputs and outputs, and thus the vector of input states cannot be always uniquely reconstructed from the vector of output states. To achieve reversible discrete dynamics, one may use the following alternative definition of ECA evolution [4]:

$$s_{cr,t+1}(i\text{-}1) = f_1(s_t(i\text{-}1), s_t(i), s_t(i+1)) \tag{2}$$

$$s_{cr,t+1}(i) = f_2(s_t(i\text{-}1), s_t(i), s_t(i+1)) \tag{3}$$

$$s_{cr,t+1}(i+1) = f_3(s_t(i\text{-}1), s_t(i), s_t(i+1)) \tag{4}$$

where subscript c means conservative and subscript r means reversible, such that the (3, 3) mappings from one step to the next are conservative and reversible and thus producing a CRECA.

One of the advantages of the use of new families of CRECA is their potential direct utilization in reconfigurable adiabatic (i.e., low-power) VLSI circuit designs [76] in contrast to the role of classical ECA circuits in classical (non-adiabatic) VLSI systems. This can be an

important application goal, especially that classical (irreversible) ECA have already proven their use in the irreversible VLSI applications such as in the testing of VLSI circuits and Finite State Machines (FSM) [2,13,17,19,20,29,30,32,46,48,56,75,88,89,90,91,96,103].

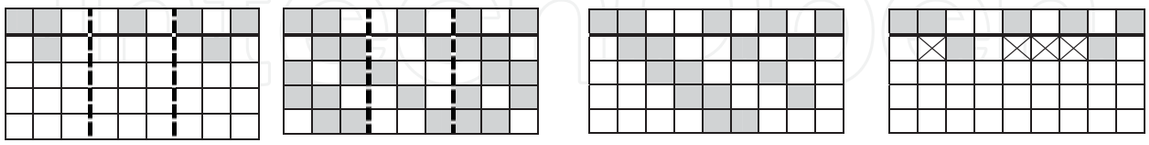
Example 6. Using the algorithm CRBF, Table 3 shows a (3, 3) conservative reversible map for Rule #170.

$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i-1)$	$s_{t+1}(i)$	$s_{t+1}(i+1)$
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	1	1

Table 3. An example of CRECA for $s_{t+1}(i-1)$ Rule #170.

The (3, 3) conservative reversible mapping in Table 3 for Rule #170 can be represented by the set of three binary strings as follows $\{10101010,11110000,11001100\}_2$, which produces the set of Rule #{170, 240, 204}.

Figure 4 shows an example of ECA discrete system dynamics for irreversible Rule #240 (Figure 4a) versus reversible Rule #{170, 240, 204} (cf. Table 3) using the same initial condition {110010101} (Figure 4b). While the evolution in Figure 4a is a non-overlapping neighbor evolution, the evolution in Figure 4c is an overlapping neighbor evolution. One notes that the irreversible overlapping-based neighbor ECA evolution for Rule #240 in Figure 4c leads to a more complex evolution than the irreversible non-overlapping-based neighbor ECA evolution for Rule #240 in Figure 4a. One can also observe that if one conducts a 3-block reversible evolution with overlapping neighbor (Figure 4d) then several cell(s) will result with conflict values inside it, and this case will be forbidden (avoided) since the value and its "opposite" cannot possess the same spatial location (address) at the same time. Therefore, 3-block reversible non-overlapping neighbor ECA evolutions (such as in Figure 4b) will be only used.



(a) (b) (c) (d)

Fig. 4. Discrete system dynamics for: (a) *irreversible* non-overlapping neighbor ECA evolution for Rule #240, (b) *reversible* non-overlapping neighbor ECA evolution for Rule #{170,240,204}, (c) *irreversible* overlapping neighbor ECA evolution for Rule #240, and (d) *reversible* overlapping neighbor ECA evolution for Rule #{170,240,204} where x means a cell with contradictory values, i.e., a cell with more than one value at the same spatial location.

Utilizing Figure 4b as an example, one can note that by incorporating the property of reversibility in the ECA, one obtains after a single forward pass, using reversible maps, a

new relatively complex discrete dynamics that can be obtained with *irreversible* ECA, for the same initial conditions, only with much higher complexity in terms of the need to perform many forward iterations (i.e., several forward passes). For example, this can be seen in Figure 4, in order to achieve the evolved ECA in Figure 4b from Figure 4a, that one needs to evolve the ECA in Figure 4a using the individual evolution Rules {#170, #240, #204} at a time in each level and thus one needs a total of $3\frac{iterations}{level}\cdot 4levels = 12$ forward passes. This fact can reflect itself in the circuit design as an optimization tradeoff between spatial complexity (memory used) and time (i.e., temporal) complexity of number of iterations, and thus depending on the cost one could select the type of optimization, i.e., *spatial optimization in irreversible maps versus temporal optimization in reversible maps*.

Given: (1a) 1-D array length l, (1b) time steps n, (2) initial condition (distribution), and (3) reversible maps $\{f_1, f_2, f_3\}$, the *analysis* of CRECA refers to the finding of the CRECA evolution at step $(t+1)$ from step t. While analysis is useful to explore the whole space of all potential reversible system dynamics, the opposite problem of *synthesis* is the more interesting problem. The synthesis problem can be stated as follows: Given (1a) 1-D array length l, (1b) discrete time steps n, (2) a priori *known* or *assigned* initial condition, and (3) the result of a *total* spatial evolution over time, produce the reversible maps $\{f_1, f_2, f_3\}$. It turns out that, while CRECA analysis is relatively an easy problem, CRECA synthesis is a more difficult one.

Different types of reversible ECA have been studied [23,31,33,34,35, 36,47,54,55]. Yet, none of these approaches considered the important modeling and processing case which uses Swap-based operations (primitives) to represent reversible ECA even in the presence of noise. As mentioned previously, modeling and processing using Swap-based circuits is important since many of the two-valued and many-valued quantum circuit implementations use two-valued and multiple-valued quantum *Swap*-based and *Not*-based gates. This can be important, since the Swap and Not gates are basic primitives in quantum computing, from which many other m-valued fundamental gates are built, such as [3,63]: (1) m-valued Not gate, (2) m-valued Controlled-Not gate (m-valued C-Not gate or m-valued Feynman gate), (3) m-valued Controlled-Controlled-Not gate (m-valued C^2-Not gate or m-valued Toffoli gate), (4) m-valued Swap gate, and (5) m-valued Controlled-Swap gate (m-valued C-Swap gate or m-valued Fredkin gate). The following is an algorithm to generate one possible Swap-based CRECA (SCRECA).

The SCRECA Algorithm can be used for the representation (i.e., modeling) and the operation (i.e., processing) reversibly on the final resulting lattice of the ECA evolution, whether that evolution was conducted using a non-overlapping neighbor ECA evolution or an overlapping neighbor ECA evolution.

Since the elementary cellular automaton in its fundamental form is a 3-cell block, then the SCRECA algorithm is based on mapping a partition of the spatial state of the automata in blocks of three cells and then finding a suitable permutation matrix reflecting the behavior of the conservative reversible system. This simplicity of the Swap-based SCRECA algorithm is also the reason for its ability to model complex evolutions.

As SCRECA is (1) reversible and (2) conservative, the output of each automaton $\vec{a}_{k,t+q}^{T}$ can be always obtained as a *permutation* of input $\vec{a}_{k,t+p}^{T}$. Therefore, the matrix $\left[\left[M_{(q-1)q}\right]\ldots\left[M_{(p+1)(p+2)}\right]\left[M_{p(p+1)}\right]\right]^{T}$ (i.e., the total matrix that results from multiplying the

Algorithm SCRECA

1. For a BECA (Boolean ECA or binary-input binary-output ECA) map specified as follows: (1) spatial length (i.e., array length) is l, and (2) temporal length (i.e., number of steps) is n.
2. **If** the length l is not a multiplication of 3, **Then** add minimum number of columns with zero values to make the length (l/3) is an integer, **Else** goto 3.
3. For temporal (row) index K = 1,2,3,…,n, spatial automaton a_k with index k = 0,1,2,…, (l/3)-1, and evolution matrix from an arbitrary time (z-1) to time z: [$M_{(z-1)z}$], find transformation matrix from time (row) p (i.e., $t + p$) to time (row) q (i.e., $t + q$) as follows:

$$\vec{a}^{T}_{k,t+q} = \vec{a}^{T}_{k,t+p}\left[\left[M_{(q-1)q}\right]\ldots\left[M_{(p+1)(p+2)}\right]\left[M_{p(p+1)}\right]\right]^{T} \text{ or}$$

$$\vec{a}_{k,t+q} = \left[\left[M_{(q-1)q}\right]\ldots\left[M_{(p+1)(p+2)}\right]\left[M_{p(p+1)}\right]\right]\vec{a}_{k,t+p}$$

   where t, p, and q are positive integers.
4. For top-to-down and left-to-right evolution, **Goto** 5.
5. Since the CRECA is *conservative*, then by using the Swap-based reversible primitives, synthesize a reversible circuit for the CRECA map as follows:

5a. **For** (m = 0; m < n; m++)

For (k = 0; $k \le \frac{l}{3} - 1$; k++)

5b. Perform one-to-one spatial mapping of permuted automaton cell a_k between levels m and (m+1) into (3, 3) Swap-based reversible primitive (cf. Figure 5) between stages m and (m+1) in the corresponding reversible circuit.

6. **End**

matrices $\left\{\left[M_{(q-1)q}\right],\ldots,\left[M_{(p+1)(p+2)}\right],\left[M_{p(p+1)}\right]\right\}$) is always a *permutation matrix*. For example, for a 3-input 3-output permutation, Figure 5 shows all possible reversible (3, 3) Swap-based permutations using the Wire (Buffer) and Swap reversible logic primitives. The matrix representation in Figure 5 is obtained by solving for the output spatial state (vector) permutation from the input state (vector). This is shown for Figure 5d as an example as follows: $\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}\begin{pmatrix}u\\v\\w\end{pmatrix}=\begin{pmatrix}u\\w\\v\end{pmatrix}$, where $\begin{pmatrix}u\\v\\w\end{pmatrix}$ is the input vector and $\begin{pmatrix}u\\w\\v\end{pmatrix}$ is the output vector in Figure 5d, respectively.

One notes that the above algorithm SCRECA can be used *spatially* for any of the following three cases: (a1) evolving a single automaton, (b1) evolving any set of automata at the same time, and (c1) evolving all of automata at the same time. Also, one notes that the above algorithm SCRECA can be used *temporally* for any of the following three cases: (a2) step-by-step automata evolution between two consecutive times (i.e., steps) (k-1) and k, (b2) automata evolution between any two times (steps) p and q, and (c2) total automata evolution for all of steps n. While temporal complexity in methods (a2) through (c2) result in step-by-step evolution matrix transformations (i.e., matrix multiplications), methods (a1) through

(c1) result in spatial complexity that determines the number of input-output permutation primitive, e.g., (3, 3), (6, 6), (9, 9), etc. The determining factor for using methods (a1) - (c2) can be, for example, the complexity of possible circuit implementation of the resulting reversible circuit models.

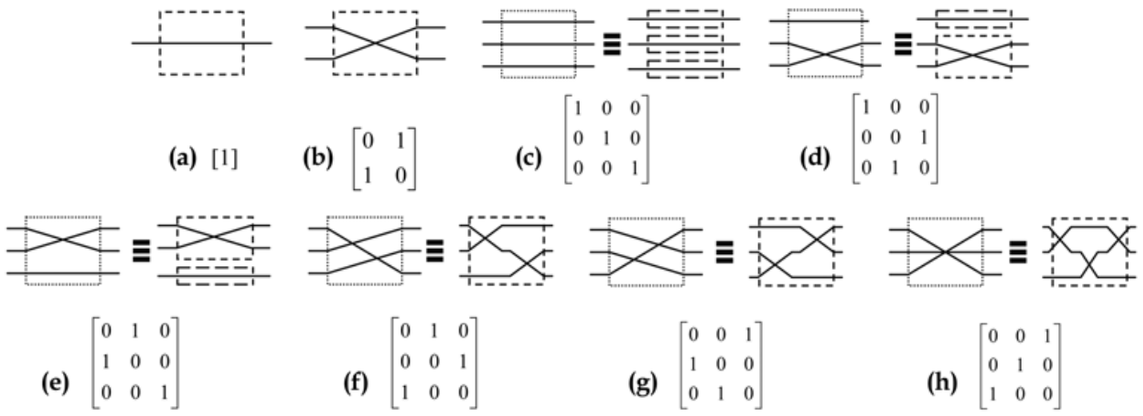


Fig. 5. Two-valued (binary) reversible and conservative permutation-based circuits: (a) (1, 1) Wire (i.e., Buffer), (b) (2, 2) Swap, and (c) - (h) all possible reversible (3, 3) Swap-based primitives.

Example 7. For a discrete system defined as follows: (1) Object: set of two objects $\{I_1, I_2\} \rightarrow$ automata; (2) Characteristic Features: location $\rightarrow$ automaton; (3) Data: spatial grid locations $\rightarrow$ cells, given an initial condition as {01010}, Figure 6a shows a possible top-to-down and left-to-right discrete dynamics of the two objects over 3-step period of time, where the tuple (t, s) indicate the cell indices in ECA. A possible conservative reversible discrete dynamics for Figure 6a can be obtained using Table 3 as shown in Figure 6b by adding an auxiliary cell C.

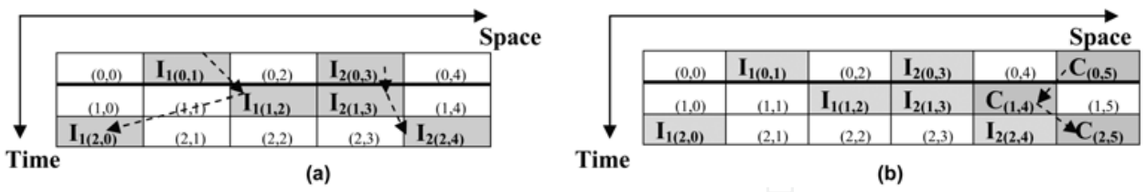
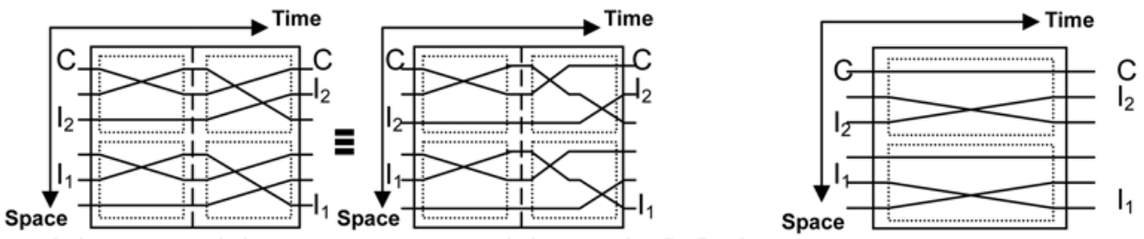


Fig. 6. An example of an evolution dynamics: (a) evolution of two objects, and (b) *reversible* discrete dynamics for Figure 6a that is obtained using Table 3.

The reversible circuit model for the evolution in Figure 6b can be obtained by interconnecting reversible *Swap*-based primitives from the permutation circuits that are shown in Figure 5. The two-stage *step-by-step* evolution of the conservative reversible discrete dynamics in Figure 6b can be modeled using the *Swap*-based reversible circuit in Figure 7a, and the two-stage *total* conservative reversible evolution can be modeled using the *Swap*-based reversible circuit in Figure 7b.



(a) (b)

Fig. 7. Conservative and reversible Swap-based circuit model for the reversible dynamics in Figure 6b: (a) (6,6) conservative reversible circuit for the two-stage *step-by-step* evolution of the discrete dynamics in Figure 6b, and (b) (6,6) circuit for the two-stage *total* evolution in Figure 6b.

One notes that the two objects I_1 and I_2 and the additional object C are of the same type, i.e., logically can be represented as value "1". The two-stage *step-by-step* evolution of the conservative reversible discrete dynamics in Figure 7a is mathematically represented by the following transformation matrices (cf. Figures 5d and 5e), where 0_{3x3} is a zero-value matrix of dimension 3x3:

$$\text{Stage 1: } \begin{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0_{3x3} \\ 0_{3x3} & \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}, \text{ Stage 2: } \begin{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} & 0_{3x3} \\ 0_{3x3} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

and the two-stage *total* evolution of the reversible discrete dynamics in Figure 7b is represented by the following transformation:

$$\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} & 0_{3x3} \\ 0_{3x3} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} & 0_{3x3} \\ 0_{3x3} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0_{3x3} \\ 0_{3x3} & \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}$$

Suppose now, due to an external unknown stimulus in Figure 6b (which is modeled using Table 3) that it is acceptable for the output vector $\{s_{t+1}(i-1)=0, s_{t+1}(i)=0, s_{t+1}(i+1)=1\}$ in the conservative reversible map in Table 3 to be $\{s_{t+1}(i-1)=0, s_{t+1}(i)=1, s_{t+1}(i+1)=1\}$, i.e., one does not care if $s_{t+1}(i)$ has value "0" or value "1", (which happens a lot in circuit design [82] for instance), and thus one would like to re-use the same map in Table 3. A sub-map of such (1) reversible and (2) conservative map would be as shown in Table 4a. One way to make Table 4a conservative and reversible is by using the method suggested in Section 2, i.e., by incorporating redundancy in inputs (K_t) and outputs (K_{t+1}).

Each row in the new map is a (4, 4) automaton and thus the new map would have 16 possible automata (i.e., each Rule is made of 16 "0/1" entries).

K_t	$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i-1)$	$s_{t+1}(i)$	$s_{t+1}(i+1)$	K_{t+1}
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	**0**	**0→1**	**1**	**0**
0	0	1	1	1	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	0
0	1	1	0	**0**	**1**	**1**	**1**
0	1	1	1	1	1	1	0

(a)

K_t	$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i-1)$	$s_{t+1}(i)$	$s_{t+1}(i+1)$	K_{t+1}
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	**0→1**	1	0
0	0	1	1	1	0	**1→0**	1
0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	0
0	1	1	0	0	1	1	1
0	1	1	1	1	1	**1→0**	1

(b)

Table 4. Reversible and conservative maps in the existence of noise: (a) reversible conservative noisy map model for Table 3 with single erroneous cell, and (b) reversible conservative noisy map model for Table 3 with multiple-errors.

Note that Table 4a will be reversible for both cases if the value of the output noisy cell $s_{t+1}(i)$ is "0" or "1". The CRECA and its conservative reversible circuit model can be obtained for Table 4 by using the SCRECA algorithm utilizing (4, 4) Swap-based reversible primitives that can be obtained using all possible input-output permutations similar to the (3, 3) Swap-based primitives in Figure 5. *Table* 4 *shows that one way to incorporate noise while maintaining reversibility is to add more spatial complexity in terms of adding extra hardware (sub-circuits) that correspond to the redundant input/output variables*. The problem of obtaining a reversible map from an irreversible map is important because, as will be seen in the next section, quantum circuits are inherently reversible and thus does not consume power, while irreversible circuits and systems, due to an irreversible mapping, do consume power [9,40,63].

5. Quantum computing for the conservative reversible ECA

Quantum computing (QC) is a method of computation that uses a dynamic process governed by the Schrödinger Equation (SE) [3,63]. The one-dimensional time-dependent SE (TDSE) takes the following general form:

$$-\frac{(h/2\pi)^2}{2m}\frac{\partial^2|\psi\rangle}{\partial x^2}+V|\psi\rangle=i(h/2\pi)\frac{\partial|\psi\rangle}{\partial t} \tag{5}$$

$$or \quad H|\psi\rangle=i(h/2\pi)\frac{\partial|\psi\rangle}{\partial t} \tag{6}$$

where h is Planck constant ($6.626\cdot10^{-34}$ J·s), $V(x, t)$ is the potential, m is particle mass, $i=\sqrt{-1}$, $|\psi(x, t)\rangle$ is the quantum state, H is the Hamiltonian operator ($H = -[(h/2\pi)^2/2m]\nabla^2 + V$), and ∇^2 is the Laplacian operator.

Although more complex forms of the Schrödinger Equation were proposed such as (1) the *relativistic* TDSE, and (2) the *master* TDSE, Equation (5) is still a good and acceptable useful form to model systems' dynamics utilizing quantum mechanical principles.

While the above holds for all physical systems, in the quantum computing (QC) context, the time-independent SE (TISE) is normally used [3,63]:

$$\nabla^2\psi = \frac{2m}{(h/2\pi)^2}(V-E)\psi \tag{7}$$

where the solution $|\psi\rangle$ is an expansion over orthogonal basis states $|\phi_i\rangle$ defined in a complex linear vector space called Hilbert space **H** as follows:

$$|\psi\rangle = \sum_i c_i |\phi_i\rangle \tag{8}$$

where the coefficients c_i are called *probability amplitudes*, and $|c_i|^2$ is the *probability* that the quantum state $|\psi\rangle$ will collapse into the (eigen) state $|\phi_i\rangle$. The probability is equal to the inner product $|\langle\phi_i|\psi\rangle|^2$, with the unitary condition $\sum|c_i|^2 = 1$. In QC, a linear and unitary operator $\Im$ is used to transform an input vector of quantum binary digits (qubits) into an output vector of qubits [3,63]. In two-valued QC, a qubit is a vector of bits defined as follows:

$$\text{qubit}_0 \equiv |0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \text{qubit}_1 \equiv |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{9}$$

A two-valued quantum state $|\psi\rangle$ is a superposition of quantum basis states $|\phi_i\rangle$, such as those defined in Equation (9). Thus, for the orthonormal computational basis states $\{|0\rangle, |1\rangle\}$, one has the following quantum state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{10}$$

where $\alpha\alpha^* = |\alpha|^2 = p_0 \equiv$ the probability of having state $|\psi\rangle$ in state $|0\rangle$, $\beta\beta^* = |\beta|^2 = p_1 \equiv$ the probability of having state $|\psi\rangle$ in state $|1\rangle$, and $|\alpha|^2 + |\beta|^2 = 1$. The calculation in QC for multiple systems (e.g., the equivalent of a register) follows the tensor (Kronecker) product ($\otimes$) [3]. For example, given two states $|\psi_1\rangle$ and $|\psi_2\rangle$ one has the following QC:

$$\begin{aligned}|\psi_1\psi_2\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle \\ &= \left(\alpha_1|0\rangle + \beta_1|1\rangle\right) \otimes \left(\alpha_2|0\rangle + \beta_2|1\rangle\right) \\ &= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle\end{aligned} \tag{11}$$

A physical system, describable as follows:

$$|\psi\rangle = c_1|\text{Spinup}\rangle + c_2|\text{Spindown}\rangle \tag{12}$$

(such as the Hydrogen atom), can be used to physically implement a two-valued QC. Another common alternative form of Equation (12) is:

$$|\psi\rangle = c_1\left|+\frac{1}{2}\right\rangle + c_2\left|-\frac{1}{2}\right\rangle = c_1|0\rangle + c_2|1\rangle \tag{13}$$

A Quantum circuit, that implements specific mapping, can be modeled as interconnects of certain quantum primitives [3]. As quantum circuits must be reversible, the conservative

reversible primitives in Figures 5a and 5b are used for quantum circuit synthesis [3,63], and therefore the reversible circuits in Figure 5 can be used for QC. Yet, since inputs in the quantum domain are vectors of bits rather than scalar bits as in the classical domain, the quantum evolution of inputs to outputs in Figure 7 (as an example) is modeled in QC differently. In QC, a (1, 1) Wire and a (2, 2) Swap quantum primitives are modeled as in Figure 8.

$$\text{(1, 1) Wire: } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{(2, 2) Swap: } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) (b)

Fig. 8. Quantum modeling of: (a) Wire (Buffer) gate and (b) Swap gate.

In Figure 8, the matrix representation is equivalent to the input-output (I/O) mapping representation of quantum gates, and we consider the example of the Swap gate to illustrate this point as follows. The Swap gate performs the following I/O mapping:

00	00
01	10
10	01
11	11

If one considers each *row* in the input side of the I/O map as an input vector represented by the natural binary code of 2^{index} with row index starting from "0", and similarly for the output row of the I/O map, then the matrix transforms the input vector to the corresponding output vector by transforming the code for the input to the code for the output. For example, the following matrix equation is the I/O mapping that uses the Swap matrix from Figure 8b:

$$\text{[Swap matrix]} \cdot \text{[input code]} = \text{[output code]}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

One notes from this example, that the Swap gate, and similarly the Buffer (Wire) quantum gate in Figure 8a, is merely a permuter, i.e., it produces output vectors which are permutations of the input vectors. Another method for obtaining the matrix representations in Figure 8 is as follows [3]: Utilizing the algebraic addition and multiplication operations over the 2nd-radix (two-valued) Galois field [3,67], one obtains the following quantum transformations of the binary input qubits into the output qubits using the two-valued Swap quantum gate:

$$|00\rangle \to |00\rangle, |01\rangle \to |10\rangle, |10\rangle \to |01\rangle, |11\rangle \to |11\rangle$$

and by solving for the set of linearly independent equations over the two-valued Galois field, one obtains the Swap evolution matrix in Figure 8b. Therefore, the evolution of input to output for QC in Figure 7a is performed as follows, where ⊗ is the tensor (i.e., Kronecker) product, | is the quantum parallel interconnection, and — is the quantum serial interconnection [3]:

$$\text{Input}_1 = |C0I_2\rangle = |101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = |0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0\rangle^T$$

$$\text{Input}_2 = |0I_10\rangle = |010\rangle = |0\rangle \otimes |1\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = |0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0\rangle^T$$

Sub-System$_1$ = Sub-System$_2$ = [(Swap | Wire) — ((Swap | Wire) — (Wire | Swap))]

$$\to (\text{Swap} \mid \text{Wire}) = \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} \otimes \begin{bmatrix}1&0\\0&1\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix}\end{bmatrix}.$$

$$\to (\text{Wire} \mid \text{Swap}) = \begin{bmatrix}1&0\\0&1\end{bmatrix} \otimes \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} & \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\\ \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix} & \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\end{bmatrix}.$$

$$\Rightarrow [(\text{Swap} \mid \text{Wire}) — ((\text{Swap} \mid \text{Wire}) — (\text{Wire} \mid \text{Swap}))] =$$

$$\begin{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix}\end{bmatrix} \cdot \begin{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix}\end{bmatrix} \cdot \begin{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} & \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\\ \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix} & \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\end{bmatrix} \cdot \begin{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} & \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\\ \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix} & \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1&0&0&0&0&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&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\end{bmatrix}.$$

$$\Rightarrow \text{Output}_1: \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |1\rangle \otimes |1\rangle \otimes |0\rangle = |110\rangle = |CI_2 0\rangle,$$

$$\text{Output}_2: \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |0\rangle \otimes |0\rangle \otimes |1\rangle = |001\rangle = |00I_1\rangle.$$

One can note that the classical method using Figure 5 and the quantum method using Figure 8 produce formally the same output in two distinct ways: *the first uses linear transformation upon classical bits while the second uses (equivalently) distinct linear transformation upon qubits*.

While the previous example considered that the probability of the quantum system is 100% in one state and 0% in the rest (i.e., $|\psi_i\rangle = 1.0|0\rangle + 0.0|1\rangle$ or $|\psi_i\rangle = 0.0|0\rangle + 1.0|1\rangle$), the following example illustrates the evolution of a general superimposed quantum state.

Example 8. Feynman gate is the quantum XOR and plays a fundamental role in quantum computing [63]. The fundamental Swap gate (cf. Figure 8b) can be decomposed into serially-interconnected Feynman-based primitives. This example illustrates the evolution of the input superimposed quantum states into output quantum states using the quantum circuit in Figure 9d which is related to the fundamental Swap gate via using the quantum circuit in Figure 9c. The quantum matrix representations for the basic gates in Figures 9a and 9b is shown within the two Figures respectively, and is obtained using any of the two methods that were shown previously for obtaining the quantum matrix representation for the Swap gate. The quantum matrix representation for the circuit in Figure 9d is obtained using the regular matrix multiplication of the matrix representations of the serially interconnected C-Not and flipped C-Not gates.

For the two quantum input states: $|\psi_1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$, $|\psi_2\rangle = a|0\rangle + b|1\rangle$, the evolution of the superimposed input quantum state:

$$|\psi\rangle = |\psi_1\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \left(a|0\rangle + b|1\rangle\right) = \frac{1}{\sqrt{2}}\left(a|00\rangle + b|01\rangle + a|10\rangle + b|11\rangle\right)$$

using the circuit in Figure 9d is obtained as follows:

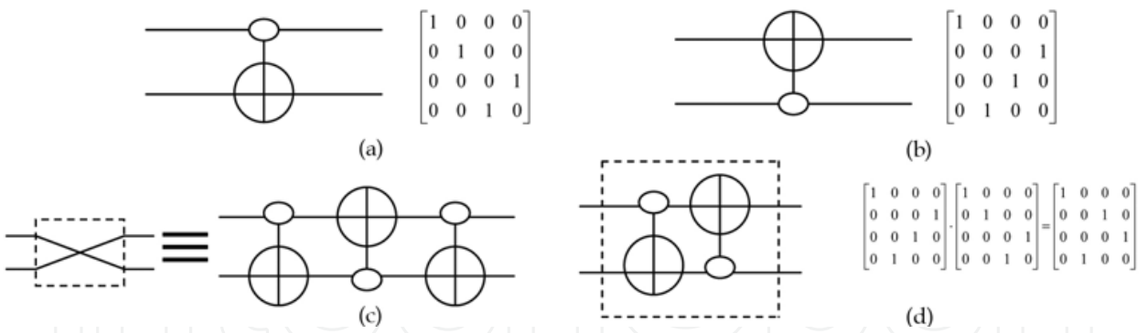


Fig. 9. Quantum gates and circuits and their quantum matrix representations: (a) 2-valued Controlled-Not gate (2-valued C-Not gate or 2-valued Feynman gate), (b) flipped 2-valued Controlled-Not gate (flipped 2-valued C-Not gate or flipped 2-valued Feynman gate), (c) Swap gate as an equivalent to a serial cascading of CNot-FlippedCNot-CNot gates, and (d) a sub-circuit of the quantum circuit in Figure 9c which is composed of a serial interconnection between a C-Not gate and a flipped C-Not gate.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a/\sqrt{2} \\ b/\sqrt{2} \\ a/\sqrt{2} \\ b/\sqrt{2} \end{bmatrix} = \begin{bmatrix} a/\sqrt{2} \\ a/\sqrt{2} \\ b/\sqrt{2} \\ b/\sqrt{2} \end{bmatrix}$$

Thus, the superimposed output quantum state is:

$$|\psi'\rangle = \frac{a}{\sqrt{2}}|00\rangle + \frac{a}{\sqrt{2}}|01\rangle + \frac{b}{\sqrt{2}}|10\rangle + \frac{b}{\sqrt{2}}|11\rangle = \left(a|0\rangle + b|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) = |\psi_2\rangle \otimes |\psi_1\rangle = |\psi_2\psi_1\rangle$$

One can note from Example 8 that the quantum matrix holds the orthonormal axes in the corresponding quantum space that define that particular quantum space, the quantum state is a vector of probabilities in that quantum space, and the quantum evolution is a transformation of the vector of probabilities in the corresponding quantum space. Equivalently, one may interpret the quantum evolution as the application of the transformed quantum space (that is the transformed orthonormal axes) upon the input vector of probabilities. This can be directly observed from the following Equation:

$$|\psi'\rangle = \begin{bmatrix} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} a/\sqrt{2} \\ b/\sqrt{2} \\ a/\sqrt{2} \\ b/\sqrt{2} \end{bmatrix} = \begin{bmatrix} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a/\sqrt{2} \\ b/\sqrt{2} \\ a/\sqrt{2} \\ b/\sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{bmatrix} \begin{bmatrix} a/\sqrt{2} \\ a/\sqrt{2} \\ b/\sqrt{2} \\ b/\sqrt{2} \end{bmatrix} = \begin{bmatrix} |00\rangle & |11\rangle & |01\rangle & |10\rangle \end{bmatrix} \begin{bmatrix} a/\sqrt{2} \\ b/\sqrt{2} \\ a/\sqrt{2} \\ b/\sqrt{2} \end{bmatrix} = \frac{1}{\sqrt{2}}\left(a|00\rangle + b|11\rangle + a|01\rangle + b|10\rangle\right)$$

Although Example 8 demonstrates the method for evolving the input superimposed quantum states into output quantum states for two-valued quantum circuit, the same

method is straightforward extended to the many-valued case that will be introduced in the following section.

For the superposition of quantum states in the corresponding quantum ECA (QECA), one can spatially obtain the total superimposed quantum state from the individual quantum cells (quantum states), by superimposing *recursively* two quantum cells (states) at a time, due to the fact that the tensor (Kronecker) product possesses the associative property, as follows:

$$|\psi\rangle = |\psi_1\psi_2...\psi_n\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle = (...(((|\psi_1\rangle \otimes |\psi_2\rangle) \otimes |\psi_3\rangle) \otimes |\psi_4\rangle) \otimes ... \otimes |\psi_n\rangle) \quad (14)$$

The decomposition in Equation (14) can be directly utilized in using a binary tree for the representation of each pair of the quantum states' tensor product, and the resulting quantum decision tree (QDT) [3] can be used as a data structure for simulating the evolution dynamics in the corresponding QECA. This is shown in Figure 10 for the two quantum states of the Wire (Buffer) in Equations (15) and (16), respectively.

$$|\psi_A\rangle = \begin{bmatrix} |0\rangle & |1\rangle \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \quad (15)$$

$$|\psi_B\rangle = \begin{bmatrix} |0\rangle & |1\rangle \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix} \quad (16)$$

Where $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ is the Buffer quantum operator [3].

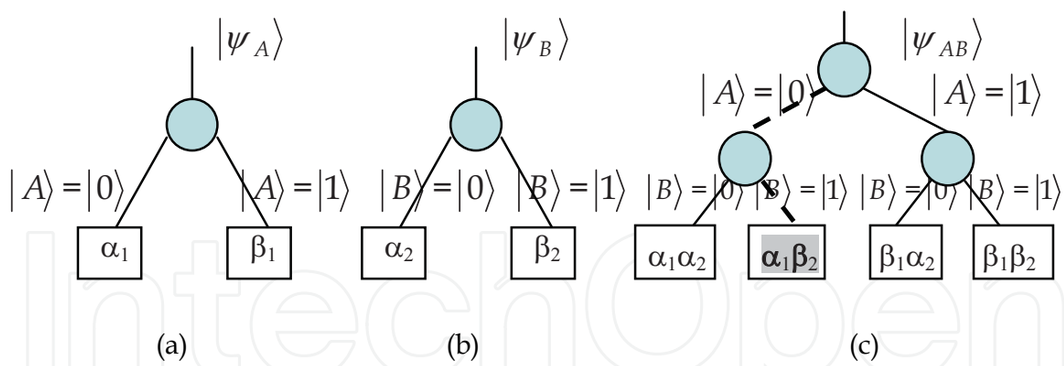


Fig. 10. Two-valued computational basis states QDT representations for: (a) Equation (15), (b) Equation (16), and (c) Equation (11).

As an example, Figure 10c shows the quantum decision path $|AB\rangle = |01\rangle$ in a dashed dark line that leads to the *highest* probability $\alpha_1\beta_2$ into which the QECA spatially superimposed state will collapse.

The QDTs in Figure 10 use the quantum computational basis states to model the ECA dynamics. Other quantum basis states such as the 1-qubit quantum systems' orthonormal composite basis states $\left\{\frac{|0\rangle + |1\rangle}{\sqrt{2}}, \frac{|0\rangle - |1\rangle}{\sqrt{2}}\right\}$ and the 2-qubit quantum systems' Einstein-

Podolsky-Rosen (EPR) basis states $\left\{\frac{|00\rangle+|11\rangle}{\sqrt{2}},\frac{|00\rangle-|11\rangle}{\sqrt{2}},\frac{|01\rangle+|10\rangle}{\sqrt{2}},\frac{|01\rangle-|10\rangle}{\sqrt{2}}\right\}$ can be used [3,63] for the quantum representation of ECA, where various tree representations will lead to different computational optimizations in terms of (1) number of internal nodes used (i.e., memory used or spatial complexity) and (2) the speed of implementation operations using such representation (i.e., temporal complexity). For instance, by using the quantum Walsh-Hadamard operator $\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$ [3,63], Equations (17) and (18) represent the equivalence of Equations (15) and (16) in terms of the orthonormal composite basis states $\left\{\frac{|0\rangle+|1\rangle}{\sqrt{2}},\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right\}$ as follows [3]:

$$\begin{aligned}|\Psi_A\rangle=\alpha_1|0\rangle+\beta_1|1\rangle&=\alpha_1\frac{|+\rangle+|-\rangle}{\sqrt{2}}+\beta_1\frac{|+\rangle-|-\rangle}{\sqrt{2}},\\&=\frac{\alpha_1+\beta_1}{\sqrt{2}}|+\rangle+\frac{\alpha_1-\beta_1}{\sqrt{2}}|-\rangle=\lambda_1|+\rangle+\mu_1|-\rangle.\end{aligned}\tag{17}$$

$$\begin{aligned}|\Psi_B\rangle=\alpha_2|0\rangle+\beta_2|1\rangle&=\alpha_2\frac{|+\rangle+|-\rangle}{\sqrt{2}}+\beta_2\frac{|+\rangle-|-\rangle}{\sqrt{2}},\\&=\frac{\alpha_2+\beta_2}{\sqrt{2}}|+\rangle+\frac{\alpha_2-\beta_2}{\sqrt{2}}|-\rangle=\lambda_2|+\rangle+\mu_2|-\rangle.\end{aligned}\tag{18}$$

where $\left\{|+\rangle=\frac{|0\rangle+|1\rangle}{\sqrt{2}},|-\rangle=\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right\}$ and $\left\{|0\rangle=\frac{|+\rangle+|-\rangle}{\sqrt{2}},|1\rangle=\frac{|+\rangle-|-\rangle}{\sqrt{2}}\right\}$.

Consequently, measuring $|\Psi_A\rangle$ with respect to the new basis $\{|+\rangle,|-\rangle\}$ will result in the state (basis) $|+\rangle$ with probability $\frac{|\alpha_1+\beta_1|^2}{2}$ and the state (basis) $|-\rangle$ with probability $\frac{|\alpha_1-\beta_1|^2}{2}$. Similarly, measuring $|\Psi_B\rangle$ with respect to the new basis $\{|+\rangle,|-\rangle\}$ will result in the state (basis) $|+\rangle$ with probability $\frac{|\alpha_2+\beta_2|^2}{2}$ and the state (basis) $|-\rangle$ with a probability of $\frac{|\alpha_2-\beta_2|^2}{2}$. Figure 11 shows the corresponding QDTs using Equations (17) and (18), respectively [3].

As an example, Figure 11c shows the quantum decision path $|AB\rangle=|+-\rangle$ in a dashed dark line that leads to the *highest* probability $\lambda_1\mu_2$ into which the QECA spatially superimposed state will collapse.

The representation in Equation (14) is the quantum superposition over the spatial axis of QECA and leads to the tensor (Kronecker) matrix multiplication for all of the 3-block cells, and the quantum evolution (dynamics) is performed over the time (temporal) axis of QECA

that transforms the input qubits into the corresponding output qubits using the spatially-obtained evolution matrix.

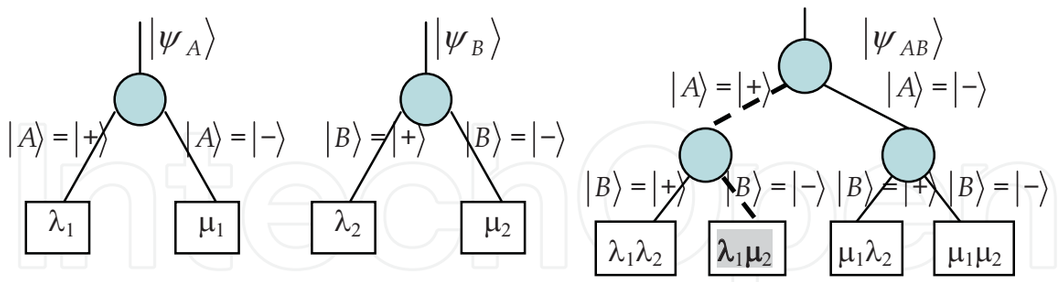


Fig. 11. Two-valued orthonormal composite basis states QDT representation for:
(a) Equation (17), (b) Equation (18), and (c) Equation (11).

6. Extensions to the many-valued conservative reversible ECA

Binary ECA that was introduced previously (cf. Figures 1 and 2) can be extended to the general case of many-valued ECA [113], where each ECA cell can take any value of "many" different values (cf. Figure 3). The next state of any cell in the many-valued ECA depends upon its present "neighborhood," which includes (a) the state of the cell itself and (b) those of its immediate neighbors to the left and right which also can take one of "many" values.
Many-valued QC can also be accomplished for the case of many-valued ECA. For the three-valued QC, the qubit becomes a 3-dimensional vector, called quantum discrete digit (qudit), and in general, for many-valued QC (MVQC) the qudit is of dimension "many". For example, one has for 3-state QC (in Hilbert space *H*) the following qudits:

$$\text{qudit}_0 \equiv |0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \text{qudit}_1 \equiv |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \text{qudit}_2 \equiv |2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{19}$$

A three-valued quantum state is a superposition of three quantum orthonormal basis states (vectors). Thus, for the orthonormal computational basis states $\{|0\rangle, |1\rangle, |2\rangle\}$, one has the following quantum state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$, where $\alpha\alpha^* = |\alpha|^2 = p_0 \equiv$ probability of having state $|\psi\rangle$ in state $|0\rangle$, $\beta\beta^* = |\beta|^2 = p_1 \equiv$ the probability of having state $|\psi\rangle$ in state $|1\rangle$, $\gamma\gamma^* = |\gamma|^2 = p_2 \equiv$ the probability of having state $|\psi\rangle$ in state $|2\rangle$, and $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$.
The calculation in QC for many-valued multiple systems follow the tensor product in a manner similar to the one demonstrated for the higher-dimensional qubit in two-valued QC [3]. It has been shown that a physical system comprising trapped ions under multiple-laser excitations can be used to reliably implement MVQC [3]. A physical system in which an atom (particle) is exposed to a specific potential field (function) can also be used to implement MVQC (two-valued being a special case). In such an implementation, the resulting distinct energy states are used as the orthonormal basis states.

In ternary QC, as a special case of the general many-valued (*m*-valued or *m*-ary) QC, a (1, 1) Wire and a (2, 2) Swap quantum primitives are modeled as in Figure 12 [3] (as compared to the binary case in Figure 8).

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) (b)

Fig. 12. The ternary unitary matrices as mathematical representations of: (a) Wire (i.e., Buffer) (**W**) quantum gate and (b) Swap (**S**) quantum gate.

In the many-valued ECA context, the general m-valued QC will be performed as one-to-one mapping permutation-based automaton cell between levels *m* and (*m*+1) into (3, 3) Swap-based reversible primitive (cf. Figure 13) between stages *m* and (*m*+1) in the corresponding reversible circuit. The idea of the previously introduced Algorithms CRBF and SCRECA (in Sections 2 and 4, respectively) can be generalized in a straightforward manner from the case of binary to the case of many-valued. The only difference would be that all Swap-based gates must be of "many" valued in their (a) representations and (b) operations.

The many-valued quantum circuits obtained through the implementation of the new quantum logic synthesis method in this Section *evolve* (i.e., *transform*) the input qudits into output qudits using the underlying mathematical transformation that is demonstrated in the following example.

Example 9. The following circuit in Figure 14 represents a parallel quantum interconnect between ternary Wire (**W**) (i.e., Buffer) and ternary Swap (**S**) quantum primitives from Figures 12 and 13.

Let us evolve the ternary input qubit $|120\rangle = |1\rangle \otimes |2\rangle \otimes |0\rangle$ using the ternary quantum circuit in Figure 14 (which is also the special permutation circuit in Figure 13d). This is done using the following general two quantum synthesis rules [3]: (1) the total multiple-valued quantum evolution transformation $[M]$ of the total serially-interconnected quantum circuit is equal to the matrix multiplication of the individual evolution matrices $[N_q]$ that correspond to the individual quantum primitives, i.e., $[M]_{serial} = \prod_q [N_q]$, and (2) the total evolution transformation $[M]$ of the total parallel-interconnected quantum circuit is equal to the tensor (i.e., Kronecker) product of the individual evolution matrices $[N_q]$ that correspond to the individual quantum primitives, i.e., $[M]_{parallel} = \otimes [N]_q$.

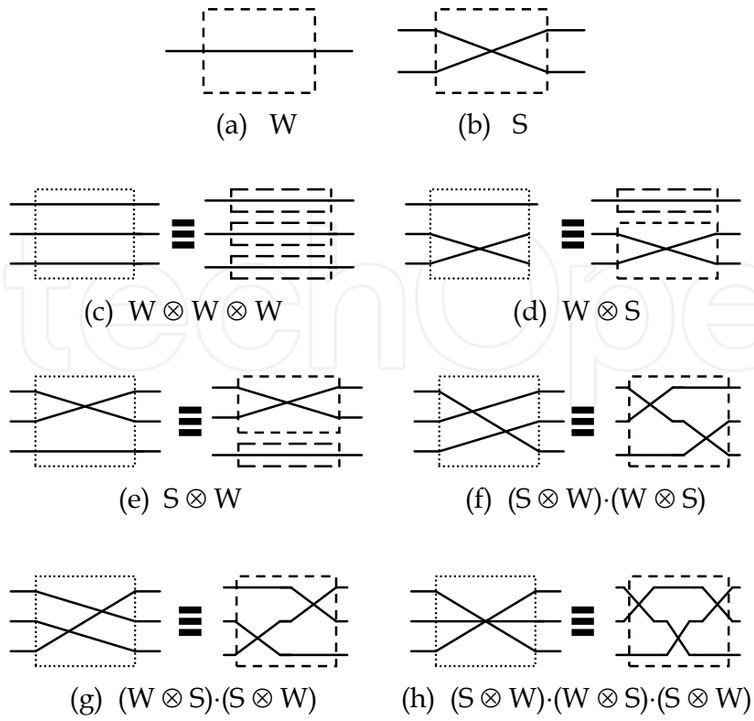


Fig. 13. Many-valued (*m*-ary) reversible and conservative permutation-based circuits: (a) *m*-ary (1, 1) Wire (W), (b) *m*-ary (2, 2) Swap (S), and (c) - (h) all possible *m*-ary reversible and quantum (3, 3) Swap-based primitives. The symbol ⊗ is the Kronecker (i.e., tensor) product.

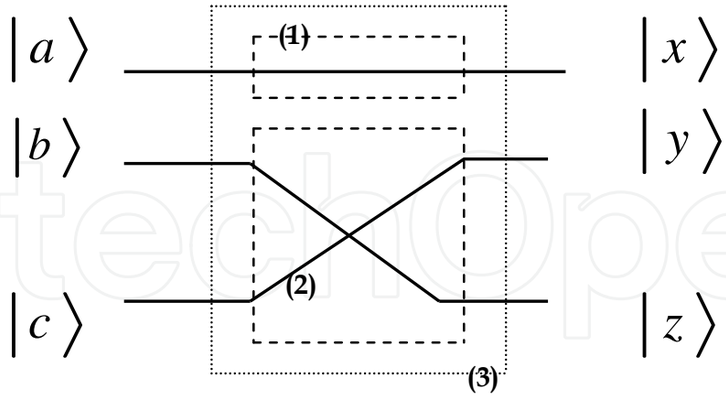


Fig. 14. A ternary quantum circuit composed of a parallel interconnect of a ternary Wire (**W**) and a ternary Swap (**S**) quantum gates.

The evolution matrices (transformations) of the parallel-interconnected dashed boxes in (1) and (2) in Figure 14 are as follows, where the symbol | means a parallel interconnection, and the utilized *unitary* matrices are the mathematical representations of the quantum gates in Figures 12 and 13 in the quantum domain [3,63].

$$\Rightarrow (3) = (1) \mid (2) = \text{Wire} \otimes \text{Swap} \Rightarrow \mathbf{M} = \mathbf{W} \otimes \mathbf{S} =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\therefore \mathbf{M} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & & \\ & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & \\ & & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{bmatrix}.$$

For the input qudit

$$\vec{I} = |120\rangle = |1\rangle \otimes |2\rangle \otimes |0\rangle = \left(0_{3x1} \quad 0_{3x1} \quad 0_{3x1} \quad 0_{3x1} \quad 0_{3x1} \quad 1 \quad 0 \quad 0 \quad 0_{3x1} \quad 0_{3x1} \quad 0_{3x1}\right)^T$$

$\therefore$ The output qudit

$$\vec{O} = [M]\vec{I} = \left(0_{3x1} \quad 0_{3x1} \quad 0_{3x1} \quad 0 \quad 0 \quad 1 \quad 0_{3x1} \quad 0_{3x1} \quad 0_{3x1} \quad 0_{3x1} \quad 0_{3x1}\right)^T$$

$$= |1\rangle \otimes |0\rangle \otimes |2\rangle = |102\rangle .$$

By observing the transformed output qudit from Example 9, one can note that the output qudit is a Swap-based transformation of the input qudit, i.e., the element values of the input qudit has been permuted within the output qudit. Similarly, all the fundamental *m*-valued (*m*-ary) Swap-based circuits in Figure 13 do this basic quantum operation of permutations. If a circuit is composed of interconnecting *m*-valued reversible elements (primitives or gates) then the overall *m*-valued circuit is also reversible. Thus, analogously to the binary case, and due to the permutation operations, the many-valued ECA circuits that are constructed from such basic elements as in Figure 13 are both conservative and reversible.

For the superposition of quantum states in the corresponding many-valued quantum ECA (*m*-valued QECA), then one can obtain the total superimposed quantum state from the individual quantum cells (quantum states), by superimposing *recursively* two quantum cells (states) at a time according to Equation (14). The decomposition in Equation (14) can be directly utilized in using an *m*-ary tree for the representation of each pair of the quantum states' tensor product, and the resulting many-valued quantum decision tree (MVQDT) [3] can be used as a data structure for simulating the evolution dynamics in the corresponding QECA. As an example, for the ternary case, the ternary Wire (Buffer) QDT is shown in Figure 15 for the two quantum states in Equations (20) and (21), respectively.

$$|\psi_A\rangle = \begin{bmatrix} |0\rangle & |1\rangle & |2\rangle \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \end{bmatrix} \tag{20}$$

$$|\psi_B\rangle = \begin{bmatrix} |0\rangle & |1\rangle & |2\rangle \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_2 \\ \beta_2 \\ \gamma_2 \end{bmatrix} \tag{21}$$

Analogously to the two-valued case, the QECA superimposed state $|\psi_{AB}\rangle$ in Figure 15 will collapse into the quantum state with the highest probability.

The ternary QDTs in Figure 15 use the quantum computational basis states to model QECA dynamics. For instance, by using the quantum Chrestenson gate $C^{(3)}_{(1)normalized} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & d_1 & d_2 \\ 1 & d_2 & d_1 \end{bmatrix}$ [3], Equation (24) presents the equivalence of Equations (20) and (21) in terms of the ternary orthonormal composite basis states as follows [3]: by using the ternary quantum signal $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$ as an input to the ternary normalized quantum Chrestenson gate, one obtains the following quantum signal at the output of the gate [3]:

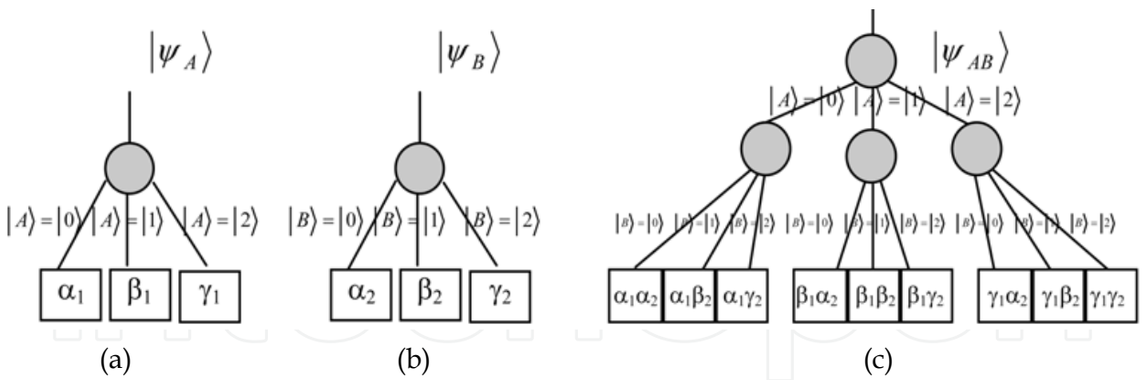


Fig. 15. Ternary computational basis states QDT representations for: (a) Equation (20), (b) Equation (21), and (c) the superposition of Equations (20) and (21): $|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$.

$$|\Psi\rangle' = \begin{bmatrix} |0\rangle & |1\rangle & |2\rangle \end{bmatrix} \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & d_1 & d_2 \\ 1 & d_2 & d_1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} |0\rangle & |1\rangle & |2\rangle \end{bmatrix} \begin{bmatrix} \frac{\alpha+\beta+\gamma}{\sqrt{3}} \\ \frac{\alpha+d_1\beta+d_2\gamma}{\sqrt{3}} \\ \frac{\alpha+d_2\beta+d_1\gamma}{\sqrt{3}} \end{bmatrix},$$

$$= \frac{\alpha+\beta+\gamma}{\sqrt{3}}|0\rangle + \frac{\alpha+d_1\beta+d_2\gamma}{\sqrt{3}}|1\rangle + \frac{\alpha+d_2\beta+d_1\gamma}{\sqrt{3}}|2\rangle. \tag{22}$$

$$|\Psi\rangle' = \begin{bmatrix} |0\rangle & |1\rangle & |2\rangle \end{bmatrix} \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & d_1 & d_2 \\ 1 & d_2 & d_1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix},$$

$$= \begin{bmatrix} \frac{|0\rangle+|1\rangle+|2\rangle}{\sqrt{3}} & \frac{|0\rangle+d_1|1\rangle+d_2|2\rangle}{\sqrt{3}} & \frac{|0\rangle+d_2|1\rangle+d_1|2\rangle}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}, \tag{23}$$

$$= \begin{bmatrix} |\mathrm{I}\rangle & |+\rangle & |-\rangle \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \alpha|\mathrm{I}\rangle + \beta|+\rangle + \gamma|-\rangle.$$

Where:

$$\{\ |+\rangle = \frac{|0\rangle+d_1|1\rangle+d_2|2\rangle}{\sqrt{3}}, |\mathrm{I}\rangle = \frac{|0\rangle+|1\rangle+|2\rangle}{\sqrt{3}}, |-\rangle = \frac{|0\rangle+d_2|1\rangle+d_1|2\rangle}{\sqrt{3}}\ \} \text{ and}$$

$$\{|0\rangle = \frac{|+\rangle+|\mathrm{I}\rangle+|-\rangle}{\sqrt{3}}, |1\rangle = \frac{d_2|+\rangle+|\mathrm{I}\rangle+d_1|-\rangle}{\sqrt{3}}, |2\rangle = \frac{d_1|+\rangle+|\mathrm{I}\rangle+d_2|-\rangle}{\sqrt{3}}\}.$$

Thus, one obtains at the input of the quantum Chrestenson gate the following quantum state:

$$\begin{aligned}
&\therefore |\Psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle, \\
&\quad = \alpha\frac{|+\rangle + ||\rangle + |-\rangle}{\sqrt{3}} + \beta\frac{d_2|+\rangle + ||\rangle + d_1|-\rangle}{\sqrt{3}} + \gamma\frac{d_1|+\rangle + ||\rangle + d_2|-\rangle}{\sqrt{3}}, \\
&\quad = \frac{\alpha + d_2\beta + d_1\gamma}{\sqrt{3}}|+\rangle + \frac{\alpha + \beta + \gamma}{\sqrt{3}}||\rangle + \frac{\alpha + d_1\beta + d_2\gamma}{\sqrt{3}}|-\rangle = \lambda|+\rangle + \mu||\rangle + \eta|-\rangle \\
&\therefore \left\{ \sqrt{p_{||\rangle}} = \frac{\alpha + \beta + \gamma}{\sqrt{3}}, \sqrt{p_{|-\rangle}} = \frac{\alpha + d_1\beta + d_2\gamma}{\sqrt{3}}, \sqrt{p_{|+\rangle}} = \frac{\alpha + d_2\beta + d_1\gamma}{\sqrt{3}} \right\}.
\end{aligned} \tag{24}$$

Consequently, measuring $|\Psi\rangle$ with respect to the new basis $\{|\ |>, |+>, |->\}$ will result in the state (basis) $\{||\rangle\}$ with probability equals to $\frac{|\alpha+\beta+\gamma|^2}{3}$, will result in the state (basis) $\{|-\rangle\}$ with probability equals to $\frac{|\alpha+d_1\beta+d_2\gamma|^2}{3}$, and will result in the state (basis) $\{|+\rangle\}$ with probability equals to $\frac{|\alpha+d_2\beta+d_1\gamma|^2}{3}$, where:

$$d_2 = -(1+d_1) = -\frac{1}{2}(1+\sqrt{3}i) = e^{\frac{4\pi i}{3}}, \quad d_1 = -(1+d_2) = -\frac{1}{2}(1-\sqrt{3}i) = e^{\frac{2\pi i}{3}}.$$

Similar to the composite-based QDT in Figure 11, one can use the ternary composite basis states {| | >, | + >, | - >} to construct the ternary orthonormal composite basis states QDT as shown in Figure 16. Analogously to the two-valued case, the QECA superimposed state $|\psi_{AB}\rangle$ in Figure 16 will collapse into the quantum state with the highest probability.

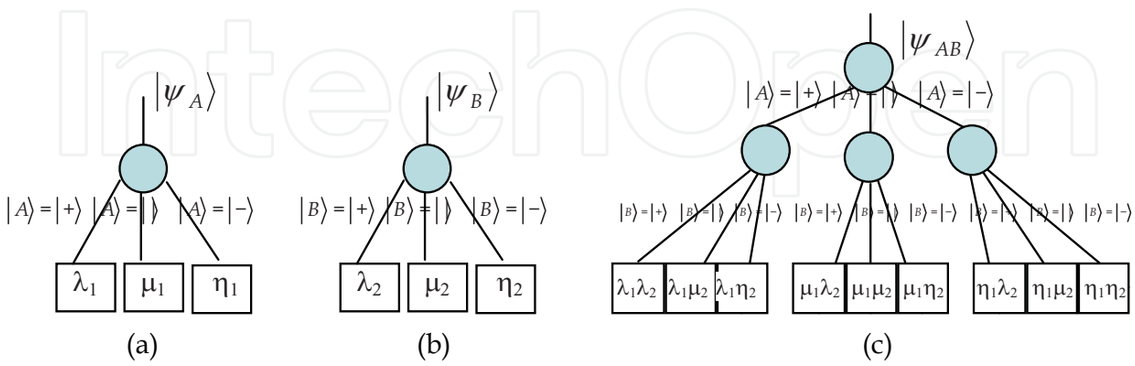


Fig. 16. Ternary computational basis states QDT representations for: (a) Equation (24) for $|\psi_A\rangle$, (b) Equation (24) for $|\psi_B\rangle$, and (c) the superposition of: $|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$.

In general, a data structure called multiple-valued quantum decision diagrams (MvQDDs) can be constructed for the corresponding multiple-valued quantum decision trees [3]. The rules for such quantum decision diagrams are the same as in classical decision diagrams [3]: (1) *join* isomorphic nodes, and (2) *remove* redundant nodes. MvQDDs can result in a more efficient representation than the corresponding MvQDTs in terms of memory space (more compact) and faster processing speed [3].

For any quantum evolution matrix (i.e., the quantum superposition over the spatial axis of QECA that leads to the tensor (Kronecker) matrix multiplication for all of the 3-block cells) the evolution (as was shown in Example 8) leads to: (a1) the permutation of the probability amplitudes or (b1) the permutation of the orthonormal bases. Consequently, in terms of the QDT representation, either: (a2) the evolution matrix leads to the permutation of the QDT leaves (probability amplitudes) while retaining the order of the QDT paths or (b2) the paths in the corresponding QDT will be permuted while retaining the order of the QDT leaves.

The state in the general case of *m*-valued (*m*-ary) QECA (which is the spatial superposition of the individual states) can be either: (1) *decomposable* as shown in Equation (25), or (2) *non-decomposable* (i.e., *entangled*) as shown in Equation (26).

$$\left|\psi_{12\ldots n}\right\rangle = \prod_{p=1}^{n}\left(\sum_{k=1}^{d}\alpha_k\left|D_k\right\rangle\right)_p \tag{25}$$

$$\left|\psi_{12\ldots n}\right\rangle \neq \prod_{p=1}^{n}\left(\sum_{k=1}^{d}\alpha_k\left|D_k\right\rangle\right)_p \tag{26}$$

where α_k is the probability amplitudes and $\left|D_k\right\rangle$ is the elementary (fundamental) basis states.

As an example of the entanglement in the QECA context, for the two-valued case, if the state vectors of quantum systems A and B were entangled with each other, then if one changes the state vector of one system A, then the corresponding state vector of the other system B is also changed, instantaneously, and independently of the medium through which some communicating signal must travel. By measuring one of the state vectors of a quantum system A, the state vector collapses into a knowable state. Instantaneously and automatically, the state vector of the other quantum system B will collapse into the other knowable state.

7. Conclusions and future work

In this research, an algorithm to model noisy discrete systems utilizing conservative reversible elementary cellular automata (CRECA) is introduced and the corresponding *m*-ary quantum computing is presented. The ECA representations in the quantum domain of (a) *m*-ary orthonormal computational basis states quantum decision trees (QDTs) and (b) *m*-ary orthonormal composite basis states QDTs are also introduced as quantum representations for the modeling and manipulation of the quantum ECA (QECA) dynamics. The new CRECA circuits and systems can play an important role in the synthesis of future

reversible circuits and systems that consume minimal power such as in the quantum technology.

As was introduced in this research, the new method of adding auxiliary variables, to incorporate the effect of noise while retaining conservativeness and reversibility, is costly in terms of the resulting structural complexity. Therefore, future work will include the creation of other less complex and more efficient CRECA algorithms to incorporate the effect of noise.

Future work will also include items such as: (1) The investigation of implementing the proposed new synthesis algorithms using Quantum Dot Cellular Automata; (2) The investigation of using the new reversible and quantum ECA algorithms that were introduced in this research for testing logical reversible and quantum circuits to (a) test, (b) localize and (c) correct circuit errors; (3) The investigation of chaos in reversible ECA and quantum ECA; (4) The investigation of the previously introduced methods for the case of rule-varying ECA; (5) The incorporation of the reversibility property in error correcting codes to correct for noisy cells within the context of ECA; (6) Some CAs are considered to conserve a kind of energy measure, a Hmiltonian, and these are reversible, thus the investigation of defining a Hamiltonian that is conserved within the context of reversible and quantum ECA that has been presented in this research will be conducted; (7) The generalization of the results introduced in this research to the general case of reversible *m*-valued *k*-dimensional CA; (8) Exploration of using genetic algorithm (GA) and genetic programming (GP) to evolve reversible CA rules to perform particular computational tasks; (9) The investigation of configuring the local settings (rules and initial conditions) of a CRECA from a given prescribed global situation (behavior) which is generally called the *inverse problem* will also be performed; (10) The development of a 3-block reversible overlapping-based neighborhood ECA evolution algorithm that doesn't result in conflict values inside map cells (which is naturally forbidden since the value and its "opposite" cannot possess the same spatial location (address) at the same time); (11) Since CA are increasingly being studied as a class of efficient parallel computers, and as the main bottleneck in applying CA more widely to parallel computing is programming, future work will involve the investigation of the automatic programming of reversible CA using GA and GP; and (12) The implementation of Soft Computing (i.e., Computational Intelligence) techniques for the newly introduced types of Reversible Cellular Automata (RCA) such as: (a) Fuzzy Reversible Cellular Automata (FRCA), (b) Fuzzy Evolutionary Reversible Cellular Automata (FERCA), and (c) Neuro-Fuzzy Reversible Cellular Automata (NFRCA).

8. References

[1] A. Adamatzky (Ed.), *Collision-Based Computing*, Springer-Verlag, 2002.

[2] A. Albicki and M.Khare, "Cellular Automata used for Test Pattern Generation," *Proc. ICCD*, pp. 56-59, 1987.

[3] A. N. Al-Rabadi, *Reversible Logic Synthesis: From Fundamentals to Quantum Computing*, Springer-Verlag, N.Y., 2004.

[4] A. N. Al-Rabadi and W. Feyerherm, "Reversible Conservative Noisy Elementary Cellular Automata (ECA) Circuits and their Quantum Computation", *Proc. of the IEEE/ACM*

International Workshop on Logic and Synthesis (IWLS), pp. 273-279, Temecula, California, June 2-4, 2004.

[5] M. Arabib, "Simple Self-Reproducing Universal Automata," *Information and Control*, 9:177-189, 1966.

[6] H. Aso and N. Honda, "Dynamical Characteristics of Linear Cellular Automata," *J. Comput. Syst. Science*, 30:291-317, 1958.

[7] J. Austin, "The Cellular Neural Network Associative Processor, C-NNAP," *Proc. 5th Intl. Conf. on Image Processing and its Application*, pp. 622-626, July 1995.

[8] F. Bagnoli, F. Franci, and R. Rechtman, "Opinion Formation and Phase Transitions in a Probabilistic Cellular Automaton with Two Absorbing States," *Proc. of the 5th International Conference on Cellular Automata for Research and Industry (ACRI)*, Switzerland, pp. 249-258, October 2002.

[9] C. Bennett, "Logical Reversibility of Computation," *IBM J. of Research and Development,* 17, pp. 525-532, 1973.

[10] S. A. Billings and Y. Yang, "Identification of Probabilistic Cellular Automata," *IEEE Trans. on System, Man, and Cybernetics*, Part B, 33(2):1-12, 2002.

[11] C. Burks and D. Farmer, "Towards Modeling DNA Sequences as Automata," *Physica D*, 10:157-167, 1984.

[12] K. Cattel and J.C. Muzio, "Synthesis of One-dimensional Linear Hybrid Cellular Automata," *IEEE Trans. on CAD*, 15:325-335, 1996.

[13] S. Chakraborty, D. Roy Chowdhury, and P. Pal Chaudhuri, "Theory and Application of Non-Group Cellular Automata for Synthesis of Easily Testable Finite State Machines," *IEEE Trans. on Computers*, 45(7): 769-781, July 1996.

[14] S. Chakraborty, *Some Studies on Theory and Applications of Additive Cellular Automata*, Ph.D. Thesis, I.I.T., Kharagpur, India, 1996.

[15] S. Chattopadhyay, S. Adhikari, S. Sengupta, and M. Pal, "Highly Regular, Modular, and Cascadable Design of Cellular Automata-based Pattern Classifier," *IEEE Trans. on VLSI Systems*, 8(6):724-735, 2000.

[16] D. R. Chowdhury, S. Basu, I. S. Gupta, and P. Pal Chaudhuri, "Design of CAECC-Cellular Automata based Error Correcting Code," *IEEE Trans. on Computers*, 43(6):759-764, June 1994.

[17] D. R. Chowdhury, S. Chakraborty, B. Vamsi, and Pal Chaudhuri, "Cellular Automata based Synthesis of Easily and Fully Testable FSMs," *Proc. ICCAD*, pp. 650-653, Nov. 1993.

[18] L. O. Chua and L. Yang, "Cellular Nellular Networks: Application," *IEEE Trans. on Circuits and Systems*, 35(10):1273-1290, 1988.

[19] F. Corno, M. S. Reorda, and G. Squillero, "Evolving Effective CA/CSTP: BIST Architectures for Sequential Circuits," *Proc. ACM Symposium on Applied Computing*, pp. 345-350, ACM Press, 2001.

[20] A. K. Das, *Additive Cellular Automata: Theory and Application as Built-in Self-test Structure*, Ph.D. Thesis, I.I.T., Kharagpur, India, 1990.

[21] A. K. Das, A. Sanyal, and P. Pal Chaudhuri, "On Characterization of Cellular Automata with Matrix Algebra," *Information Sciences*, 61(3): 251-277, 1992.

[22] S. Dormann, A. Deutsch, and A. T. Lawniczak, "Fourier Analysis of Turing-like Pattern Formation in Cellular Automaton Models," *Future Generation Computer Science*, 17:901-909, 2001.

[23] J. Durand-Lose, "Representing Reversible Cellular Automata with Reversible Block Cellular Automata," *Discrete Mathematics and Theoretical Computer Science Proceedings*, AA (DM-CCG), pp. 145-154, 2001.

[24] P. Flocchini, F. Geurts, A. Mingarelli, and N. Santoro, "Convergence and Aperiodicity in Fuzzy Cellular Automata: Revisiting Rule 90," *Physica D: Nonlinear Phenomena*, 142(1-2):20-28, August 2000.

[25] U. Frisch, B. Hasslacher, and Y. Pomeau, "Lattice Gas Automata for the Navier-Stokes Equation," *Phys. Rev. Lett.*, 56(14): 1505-1508, 1986.

[26] N. Ganguly, P. Maji, S. Dhar, B. K. Sikdar, and P. Pal Chaudhuri, "Evolving Cellular Automata as Pattern Classifier," *Proc. 5th International Conference on Cellular Automata for Research and Industry (ACRI)*, Switzerland, pp. 56-68, October 2002.

[27] G. Grinstein, C. Jayaprakash, and Y. He, "Statistical Mechanics of Probabilistic Cellular Automata," *Phys. Rev. Lett.*, 55: 2527-2530, 1985.

[28] H. Gutowitz, "A Hierarchical Classification of CA," *Physica D*, 45:136-156, 1990.

[29] P. D. Hortensius, R. D. McLeod, W. Pries, D. M. Miller, and H. C. Card, "Cellular Automata based Pseudo-Random Number Generators for Built-in Self-Test," *IEEE Tans. on CAD*, 8(8): 842-859, August 1989.

[30] P. D. Hortensius, R. D. McLeod, and H. C. Card, "Cellular Automata Based Signature Analysis for Built-in Self-Test," *IEEE Trans. on Computers*, C-39(10): 1273-1283, October 1990.

[31] K. Imai, "Reversible Cellular Automata," *Information Epistemology and Computation (IEC) Laboratory*, Graduate School of Engineering, Hiroshima University, Japan.

[32] D. Kagaris and S. Tragoudas, "Von Neumann Hybrid Cellular Automata for Generating Deterministic Test Sequences," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, 6(3): 308-321, 2001.

[33] J. Kari, "Reversibility of 2D Cellular Automata is Undecidable," *Physica D*, 45:379-385, 1990.

[34] J. Kari, "Reversibility and Surjectivity Problems of Cellular Automata," *J. of Comp. and Sys. Science*, 48(1): 149-182, 1994.

[35] J. Kari, "Representation of Reversible Cellular Automata with Block Permutations," *Math. Sys. Theory*, 29: 47-61, 1996.

[36] J. Kari, "On the Structural Depth of Reversible Cellular Automata," *Fundamenta Informaticae,* 38(1-2): 93-107, 1999.

[37] O. Kirchkamp, *Spatial Evolution of Automata in the Prisoners' Dilemma*, Manuscript, Bonn University, 1994.

[38] P. Kurka, "Languages, Equicontinuity and Attractors in Cellular Automata," *Ergodic Theor., Dynamic System*, 17: 229-254,1997.

[39] O. Lafe, "Data Compression and Encryption using Cellular Automata Transforms," *IEEE International Joint Symposia on Intelligence and Systems (IJSIS)*, pp. 234-241, 1996.

[40] R. Landauer, "Irreversibility and Heat Generation in the Computational Process," *IBM J. of Research and Development,* 5, pp. 183-191, 1961.
[41] C. Langton, "Self-Reproduction in Cellular Automata," *Physica D*, 10:134-144, 1984.
[42] C. Langton, "Computation at the Edge of Chaos," *Physica D*, 42:12-37, 1990.
[43] W. Li, N. H. Packard, and C. G. Langton, "Transition Phenomena in Cellular Automata Rule Space," *Physica D*, 45: 77-94, 1990.
[44] M. Mahajan, *Studies in Language Classes Defined by Different Time-Varying Cellular Automata*, Ph.D. Thesis, I.I.T., Madras, 1992.
[45] O. Martin, A. M. Odlyzko, and S. Wolfram, "Algebraic Properties of Cellular Automata," *Comm. Math. Phys.*, 93: 219-258, 1984.
[46] M. Matsumoto, "Simple Cellular Automata as Pseudorandom *m*-Sequence Generators for Built-in Self-Test," *ACM Trans. on Modeling and Computer Simulation (TOMACS)*, 8(1): 31-42, 1998.
[47] H. V. McIntosh, "Reversible Cellular Automata," *Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias,* Universidad Autónoma de Puebla, México, 1991.
[48] S. Misra, *Theory and Application of Additive Cellular Automata for Easily Testable VLSI Circuit Design*,Ph.D. Thesis, I.I.T., Kharagpur, India, 1992.
[49] M. Mitchell, P. T. Hraber, and J. P. Crutchfield, "Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations," *Complex Systems*, 7: 89-130, 1993.
[50] E. Moore (editor), *Sequential Machine: Selected Papers*, Addison-Wesley Publishing Company Inc.,Redwood City, CA, 1964.
[51] J. H. Moore and L. W. Hahn, "A Cellular Automata-based Pattern Recognition Approach for Identifying Gene-Gene and Gene-Environment Interactions," *American Journal of Human Genetics*, 67(52), 2000.
[52] F. J. Morales, J.P. Crutchfield, and M. Mitchell, "Evolving Two-dimensional Cellular Automata to Perform Density Classification: A Report on Work in Progress," *Parallel Computing*, 27:571-585, 2001.
[53] J. Moreira and A. Deutsch, "Cellular Automaton Models of Tumor Development: A Critical Review," *Advances in Complex Systems*, 5(2&3): 247-269, 2002.
[54] K. Morita and S. Ueno, "Parallel Generation and Parsing of Array Languages using Reversible Cellular Automata," *International Journal of Pattern Recognition and Artificial Intelligence*, 8: 543-561, 1994.
[55] K. Morita, "Reversible Simulation of One-Dimensional Irreversible Cellular Automata," *Theoretical Computer Science*, 148: 157-163, 1995.
[56] G. Mrugalski, J. Rajski, and J. Tyszer, "Cellular Automata-based Test Pattern Generators with Phase Shifter," *IEEE Trans. on CAD*, 19(8): 878-893, August 2000.
[57] M. Mukherjee, N. Ganguly, and P. Pal Chaukhuri, "Cellular Automata Based Authentication," *Proc. 5th International Conference on Cellular Automata for Research and Industry (ACRI)*, Switzerland, pp. 259-269, October 2002.
[58] J. C. Muzio *et. al.*, "Analysis of One-dimensional Linear Hybrid Cellular Automata over GF(q)," *IEEE Trans. on Computers*, 45(7): 782-792, July 1996.
[59] S. Nandi, *Additive Cellular Automata: Theory and Application for Testable Circuit Design and Data Encryption*, Ph.D. Thesis, I.I.T., Kharagpur, India, 1994.

[60] S. Nandi *et. al.*, "Analysis of Periodic and Intermediate Boundary 90/150 Cellular Automata," *IEEE Trans. on Computers*, 45(1): 1-12, January 1996.

[61] S. Nandi, B. K. Kar, and P. Pal Chaudhuri, "Theory and Application of Cellular Automata in Cryptography," *IEEE Trans. on Computers*, 43(12): 1346-1357, December 1994.

[62] J. V. Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks (ed.), Univ. of Illinois Press, Urbana and London, 1966.

[63] M. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information,* Cambridge University Press, 2000.

[64] H. Nishio, "Real Time Sorting of Binary Numbers by One-Dimensional Cellular Automata," *Technical Report*, Kyoto University, 1981.

[65] H. Nishio and Y. Kobuchi, "Fault Tolerant Cellular Space," *J. Comp. Sys. Science*, 11:150-170, 1975.

[66] S. Omohundro, "Modeling Cellular Automata with Partial Differential Equations," *Physica D*, 10: 128-134, 1984.

[67] C. Paar, P. Fleischmann, and P. Roelse, "Efficient Multiplier Architectures for Galois Fields GF(2^{4n})," *IEEE Trans. on Computers*, 47(2): 162-170, 1998.

[68] K. Paul and D. R. Chowdhury, "Application of GF(2^p) CA in Burst Error Correcting Codes," *Proc. VLSI*, INDIA, pp. 562-567, January 2000.

[69] K. Paul, D. R. Chowdhury, and P. Pal Chaudhuri, "Cellular Automata Based Transform Coding for Image Compression," *Proc. HiPC*, INDIA, pp. 269-273, December 1999.

[70] K. Paul, A. Roy, P. K. Nandi, B. N. Roy, M. D. Purkhayastha, S. Chattopadhyay, and P. Pal Chaudhuri, "Theory and Application of Multiple Attractor Cellular Automata for Fault Diagnosis," *Proc. Asian Test Symposium*, pp. 388-392, December 1998.

[71] Y. Pomeau, "Invariants in Cellular Automata," *J. Phys. A*, 17, 1986.

[72] K. Preston, M. J. Duff, S. Levialdi, Ph. E. Norgren, and J. I. Toriwaki, "Basics of Cellular Logic with Some Applications in Medical Image Processing," *Proc. IEEE*, 67: 826-856, 1979.

[73] R. Raghavan, "Cellular Automata in Pattern Recognition," *Information Science*, 70: 145-177, 1993.

[74] F. C. Richards, T. P. Meyer, and N. H. Packard, "Extracting Cellular Automata Rules directly from Experimental Data," *Physica D*, 45: 189-202, 1990.

[75] M. Roncken, K. Stevens, and P. Pal Chaudhuri, "CA-BIST for Asynchronous Circuits: A Case Study on RAPPID Asynchronous Instruction Length Decoder," *Proc. 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Eilat, Israel, pp. 62-72, 2000.

[76] K. Roy and S. Prasad, *Low-Power CMOS VLSI Circuit Design,* John Wiley & Sons Inc., 2000.

[77] S. Saha, P. Maji, N. Ganguly, B. K. Sikdar, and P. Pal Chaudhuri, "Evolution of Cellular Automata Based Pattern Classifier and Recognizer," *IEEE Conference on System, Man & Cybernetics*, pp. 114-119, 2002.

[78] R. M. Z. D. Santos and S. Coutinho, "Dynamics of HIV Infection: A Cellular Automata Approach," *Phys.Rev. Lett.*, 87(16), 168102, 2001.

[79] P. Sarkar, "A Brief History of Cellular Automata," *ACM Computing Systems*, 32(1):80-107, March 2000.

[80] P. Sarkar. and R. Barua, "Multi-dimensional σ - Automata, π - Polynomial and Generalized s-Matrices," *Theoretical Computer Science*, 197(1-2): 111-138, 1998.

[81] P. Sarkar. and R. Barua, "The Set of Reversible 90/150 Cellular Automata is Regular," *Discrete Applied Math.*, 84(1-3): 199-213,1998.

[82] T. Sasao, *Logic Synthesis and Optimization,* Kluwer Academic Publishers, 1993.

[83] B. Schonfisch and M. Kinder, "A Fish Migration Model," *Proc. 5th International Conference on Cellular Automata for Research and Industry (ACRI)*, Switzerland, pages 210-219, October 2002.

[84] B. Schonfisch and A. D. Ross, "Synchronous and Asynchronous Updating in Cellular Automata," *Biosystems*, 51: 123-143, 1999.

[85] S. Sen, C. Shaw, D. R. Chowdhuri, N. Ganguly, and P. Pal Chaudhuri, "Cellular Automata Based Cryptosystem," *Proc. ICICS*, Singapore, pp. 303-314, December 2002.

[86] M. Serra, T. Slater, J. C. Muzio, and D. M. Miller, "Analysis of One Dimensional Cellular Automata and their Aliasing Probabilities," *IEEE Trans. on CAD*, 9(7): 767-778, July 1990.

[87] M. Shereshevsky, "Lyapunov Exponent for One-dimensional Cellular Automata," *J. Nonlinear Science*, 2: 1-8, 1992.

[88] B. K. Sikdar, D. K. Das, V. Boppana, C. Yang, S. Mukherjee, and P. Pal Chaudhuri, "GF(2^p) Cellular Automata as a Built-In Self-Test Structure," *Proc. ASP-DAC*, Japan, pp. 319-324, 2001.

[89] B. K. Sikdar, N. Ganguly, and P. Pal Chudhuri, "Design of Hierarchical Cellular Automata for On-Chip Test Pattern Generator," *IEEE Trans. on CAD*, 21(12): 1530-1539, Dec. 2002.

[90] B. K. Sikdar, N. Ganguly, A. Karmakar, S. Chowdhury, and P. Pal Chaudhuri, "Multiple Attractor Cellular Automata for Hierarchical DIAGNOSIS of VLSI Circuits," *Proc. Asian Test Symposium*, pp. 385-390, November 2001.

[91] B. K. Sikdar, N. Ganguly, P. Majumder, and P. P. Chaudhuri, "Design of Multiple Attractor GF(2^p) Cellular Automata for Diagnosis of VLSI Circuits," *Proc. Int. Conf. on VLSI Design*, India, pp. 454-459, January 2001.

[92] M. Sipper, "Co-evolving Non-Uniform Cellular Automata to Perform Computations," *Physica D*, 92: 193- 208, 1996.

[93] A. Smith, *Introduction to and Survey of Polyautomata Theory. Automata, Languages, Development*, North Holland Publishing Co., 1976.

[94] S. Smith, R. Watt, and R. Hameroff, "Cellular Automata in Cytoskeletal Lattices," *Physica D*, 10: 168-174, 1984.

[95] A. R. Smith(III), "Real-time Language Recognition by One-Dimensional Cellular Automata," *J. Computer Sys. Science*, 6: 233-253, 1972.

[96] S. Tezuka, "A Method of Designing Cellular Automata as Pseudorandom Number Generators for Built-In Self-Test for VLSI," *Finite Fields: Theory, Applications and Algorithms*, pp. 363-367, 1994.

[97] T. Toffoli, "CAM: A High-Performance Cellular Automata Machine," *Physica D*, 10: 195-204, 1984.

[98] T. Toffoli, "Cellular Automata as an Alternative to (rather than an approximation of) Differential Equations in Modeling Physics," *Physica D*, 10: 117-127, 1984.

[99] T. Toffoli and N. Margolus, *Cellular Automata Macjomes: A New Environment for Modeling*, MIT Press, Cambridge, Mass, 1987.

[100] T. Toffoli and N. Margolus, *Invertible Cellular Automata: A New Environment for Modeling*, MIT Press, Cambridge, Mass., 1987.

[101] T. Toffoli and N. Margolus, "Irreversible Cellular Automata: A Review", *Physica D*, 45: 229-253, 1993.

[102] M. Tomassini, M. Sipper, and M. Perrenoud, "On the Generation of High-Quality Random Numbers by Two-dimensional Cellular Automata," *IEEE Trans. on Computers*, 49(10):1146-1151, 2000.

[103] Ph. Tsalides, T. A. York, and A. Thanailakis, "Pseudo-Random Number Generators for VLSI Systems Based on Linear Cellular Automata," *IEE Proc. E. Comp. Digit. Tech.*, 138(4): 241-249, 1991.

[104] P. Tzionas, P. Tsalides, and A. Thanailakis, "A New Cellular Automaton-Based Nearest Neighbor Pattern Classifier and its VLSI Implementation," *IEEE Trans. on VLSI Implementation*, 2(3): 343-353, 1994.

[105] P. Tzionas, Ph. Tsalides, and A. Thanailakis, "A Cellular Neural Network Predicting the Behavior of a Complex System Modeled as a Cellular Automaton," *Proc. 15th Annual International Symposium on Forecasting (ISF)*, Toronto, Canada, June 4-7 1995.

[106] G. Vchniac, "Simulating Physics with Cellular Automata," *Physica D*, 10: 96-115, 1984.

[107] A. Winfree, E. Winfree, and H. Seifert, "Organizing Centers in Cellular Excitable Meduim," *Physica D*, 17: 109-115, 1985.

[108] S. Wolfram, "Statistical Mechanics of Cellular Automata," *Rev. Mod. Phys.*, 55(3): 601-644, July 1983.

[109] S. Wolfram, "Universality and Complexity in Cellular Automata," *Physica D*, 10: 1-35, 1984.

[110] S. Wolfram, "Undecidability and Intractability in Theoretical Physics," *Phys. Rev. Lett.*, 54: 735-738, 1985.

[111] S. Wolfram, *Theory and Applications of Cellular Automata*, World Scientific, Singapore, 1986.

[112] S. Wolfram, *Cellular Automata and Complexity*, 2002.

[113] S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.

[114] F. Wu, "A Linguistic Cellular Automata Simulation Approach for Sustainable Land Development in a Fast Growing Region," *Computers, Environment and Urban Systems*, 20(6): 367-387, Nov. 1996.

[115] A. Wuensche and M. Lesser, *The Global Dynamics of Cellular Automata*, Volume 1, Addison-Wesley, 1992.

[116] A. Wuensche, "Classifying Cellular Automata Automatically," *Complexity*, 4(3): 47-66, 1999.

[117] M. Zwick and H. Shu, "Set-Theoretic Reconstructability of Elementary Cellular Automata," *Advances in Systems Science and Applications*, 1, pp. 31-36, 1995

Cellular Automata - Innovative Modelling for Science and Engineering

Edited by Dr. Alejandro Salcido

ISBN 978-953-307-172-5
Hard cover, 426 pages
Publisher InTech
Published online 11, April, 2011
Published in print edition April, 2011

Modelling and simulation are disciplines of major importance for science and engineering. There is no science without models, and simulation has nowadays become a very useful tool, sometimes unavoidable, for development of both science and engineering. The main attractive feature of cellular automata is that, in spite of their conceptual simplicity which allows an easiness of implementation for computer simulation, as a detailed and complete mathematical analysis in principle, they are able to exhibit a wide variety of amazingly complex behaviour. This feature of cellular automata has attracted the researchers' attention from a wide variety of divergent fields of the exact disciplines of science and engineering, but also of the social sciences, and sometimes beyond. The collective complex behaviour of numerous systems, which emerge from the interaction of a multitude of simple individuals, is being conveniently modelled and simulated with cellular automata for very different purposes. In this book, a number of innovative applications of cellular automata models in the fields of Quantum Computing, Materials Science, Cryptography and Coding, and Robotics and Image Processing are presented.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen