Evolved Reversible Cascades Realized on the CAM-Brain Machine (CBM)

Andrzej Buller
ATR Human Information Science Labs
2-2-2 Hikaridai, Keihanna Science City
Kyoto 619-0288 Japan
tel. +81 774 95 1009, buller@atr.co.jp
http://www.atr.co.jp/his/~buller

Marek Perkowski
Department of Electrical Engineering and Computer Science, KAIST, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Korea
mperkows@ee.kaist.ac.kr

Abstract

This paper presents a method of automated synthesis of reversible cascades in a special cellular automaton called CAM-Brain Machine (CBM). Reversible circuits are investigated because they are expected to dissipate much less energy than their irreversible counterparts. It is believed that one day they will be implemented as nano-scale 3-dimensional chips. A circuit is reversible if the number of its inputs equals the number of its outputs and there is a one-to-one mapping between spaces of input vectors and output vectors. This paper provides (1) a brief introduction to reversible logic concentrating on definitions and properties of the Feynman, Toffoli, Fredkin and other gates, (2) an introduction to logic synthesis for the CBM and (3) a collection of reversible structures evolved using a dedicated GA and located in the CBM using the NeuroMaze 3.0 Pro, a software tool for computer-aided design of CBM-style structures.

1. Introduction

This paper presents a method of automated synthesis of reversible cascades in a cellular automaton. A layout of a desired cascade is generated by a GA developed in the Portland Quantum Logic Group (Lukac & Perkowski 2002; Lukas et al. 2002) and then converted to certain states of the cells constituting the cellular automaton (Buller 2003b). The ATR CAM-Brain Machine (Korkin et al. 2000) was used as the research platform. This research is being conducted as a part of the Quantrix Project, launched as one of four themes explored in the framework of the Artificial Brain Project conducted at the ATR Human Information Science Laboratories, Kyoto (Buller & Shimohara 2003). It contains first results obtained in search for scientific grounds for a new evolvable hardware for on-board brains of intelligent robots. Reversible circuits appear to be a promising solution because they are expected to dissipate much less energy than their irreversible counterparts exploited nowadays (cf. Bennett 1973). It is believed that one day they will be implemented as nano-scale 3-dimensional chips that nowadays are impossible because of the still unsolved problem of heat produced in traditional logic gates.

A circuit is reversible if the number of its inputs equals the number of its outputs and there is a one-to-one mapping between spaces of input vectors and output vectors (Perkowski et al. 2001). The qualification 'reversible' comes from the fact that every reversible gate or circuit provides unique deduction of the input vector based on the given gate definition and the output vector. During the development of Reversible Logic various basic sets of reversible gates were introduced. One of the sets, called the CNTS Library (Shende et al. 2002), includes the NOT gate that for 0 returns 1, while for 1 it returns 0, as well as the Feynman, Toffoli and SWAP gates that will be introduced in Section 3 together with more complex Fredkin gate. In the present mainstream of the Reversible Logic-related research the circuits are being synthesized as cascades that can be drawn as arrays of separate horizontal lines representing wires with vertical symbols representing basic gates influencing locally the signals passing along the wires. Hence, fan-outs are not allowed. This restriction is followed in this paper.

Cellular Automaton (CA) is defined as a computing device based on three elements: a set of connected sites (cells), a set of states that are allowed on the sites (cells), and a set of rules for how the states are updated (cf. Gershenfeld 1999: 102). For implementation of the reversible cascades we used

CA adjusted based of the following assumptions (Buller 2003b): (1) the state of every cell is defined using one binary variable called the pulsing state variable, as well as six binary variables called the frozen state variables, and (2) there is only one cell transition rule, that is the Boolean function S_1 that returns 1 when exactly one of its inputs is equal to one and returns zero otherwise. This research is not related to reversible CA (see Morita & Harao 1989; Morita & Ueno 1992) because the employed CA itself is not reversible. This approach is being developed not to directly obtain reversible devices, but to obtain a platform for modeling large-scale reversible structures.

The ATR's CAM-Brain Machine (CBM) is a dedicated FPGA-based hardware for experiments with 3-D CA designed to evolve and emulate large-scale para-neural networks (Korkin et al. 2000). Since the genetic algorithm located in the CBM proved to be too weak to be used for synthesis of useful structures, the ATR's Brain Building Group developed the NeuroMaze 3.0 Pro™, software for computer aided designing and testing of neural modules to be run on the CBM (Liu 2002). This software appeared a convenient tool for rapid modeling and testing of reversible cascades. Hence, a number of reversible structures evolved using a dedicated Genetic Algorithm developed in the Portland Quantum Logic Group (Lukac & Perkowski 2002; Lukas et al. 2002) have been successfully converted into cellular-automatic structures and run on the CBM.

Summarizing, in order to have a desired reversible circuit run on a dedicated FPGA-based hardware (CBM), we (1) evolve the circuit using a special GA, (2) simplify the circuit using peephole optimizing transforms based on rules of EXOR algebra and tree search, (3) convert a code produced by the GA into cellular-automatic structure, (4) execute the structure in the CBM. Note that in our approach the evolution is only on the level of logic synthesis of complex logic gates and not on the low level cellular cells which approach would make the GA responsible for logic, timing, placement and routing. The approach we propose here combines evolutionary algorithm (EA) software, standard Computer Aided Design (CAD) and some human intervention, which seems to be a more realistic way to create complex circuits in the CBM than the entirely evolutionary approaches proposed earlier (eg. de Garis et al. 1999). Reversible circuits containing up to 20 gates has already been evolved (Lukac et al. 2003; Perkowski et al. 2003).

2. Nomenclature and lemmas

x' - Boolean negation ($0' = 1$, $1' = 0$)
xy – Boolean product (00 = 0, 01=0, 10 = 0, 11 = 1)
$x + y$ – Boolean sum (0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 1)
$x \oplus y$ - Exor (exclusive OR) ($0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$, $x \oplus 1 = x'$, $x \oplus 0 = x$, $x \oplus x' = 1$)

Exor Logic lemmas (Sasao 1999: 44)

$(x \oplus y) \oplus z = x \oplus (y \oplus z)$, $x(y \oplus z) = xy \oplus xz$, $x \oplus y = y \oplus x$, $x \oplus x = 0$

3. Reversible Logic

This section introduces the basic reversible gates and provides examples of using them for synthesis of reversible cascades.

3.1. CNOT (Feynman Gate)

The CNOT (Controlled NOT) gate, called also Feynman gate, is represented using a compound of three symbols: ⊕, •, and | that represent an inverter, a control and a connection, respectively (Figure 3.1a). It concerns two and only two wires. The logical value in the wire to which the control • is attached is the same both immediately before and immediately after the control. As for the wire on which the inverter ⊕ is attached, the CNOT's behavior depends on the value detected by the control. When the value detected by the control is 1, the gate affects the wire the same way as NOT. When the

value detected by the control is 0, the logical value in the wire to which the inverter is attached is the same both immediately before and immediately after the inverter (Figure 3.1b). Since $\mathbf{0} \oplus y = y$ for any Boolean value of y, for $x = 0$ the Feynman gate behaves as a fan-out element (Figure 3.1c).

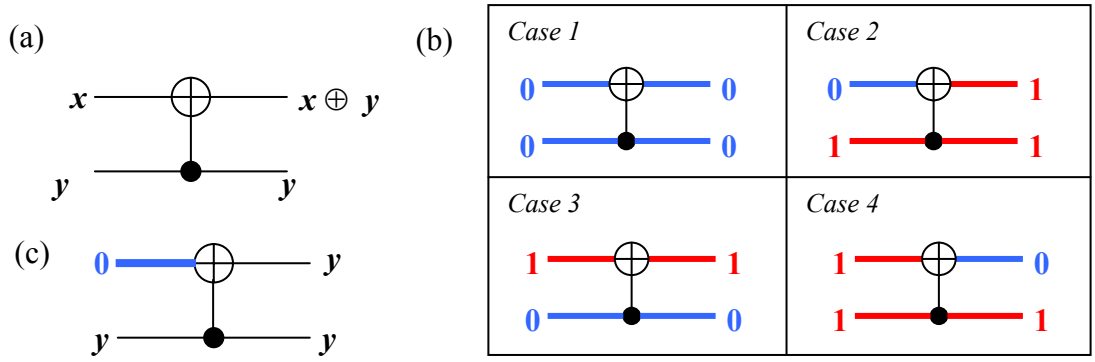


Figure 3.1. CNOT (Feynman) gate (a) symbol, (b) illustration of behavior, (c) applied as a fan-out element.

3.2. Toffoli Gate

The Toffoli gate is represented as a compound of one inverter ⊕, two controls, and the vertical connection | (Figure 3.2a). It concerns three and only three wires. The logical values in the wires to which the controls • are attached are the same both immediately before and immediately after a given control. As for the wire on which the inverter ⊕ is attached, the Toffoli's performance depends on the values detected by the controls. When the product of the values detected by the controls is 1, the gate affects the wire the same way as NOT. When the product of values detected by the controls is 0, the logical value in the wire to which the inverter is attached is the same both immediately before and immediately after the inverter. (Figure 3.2b).

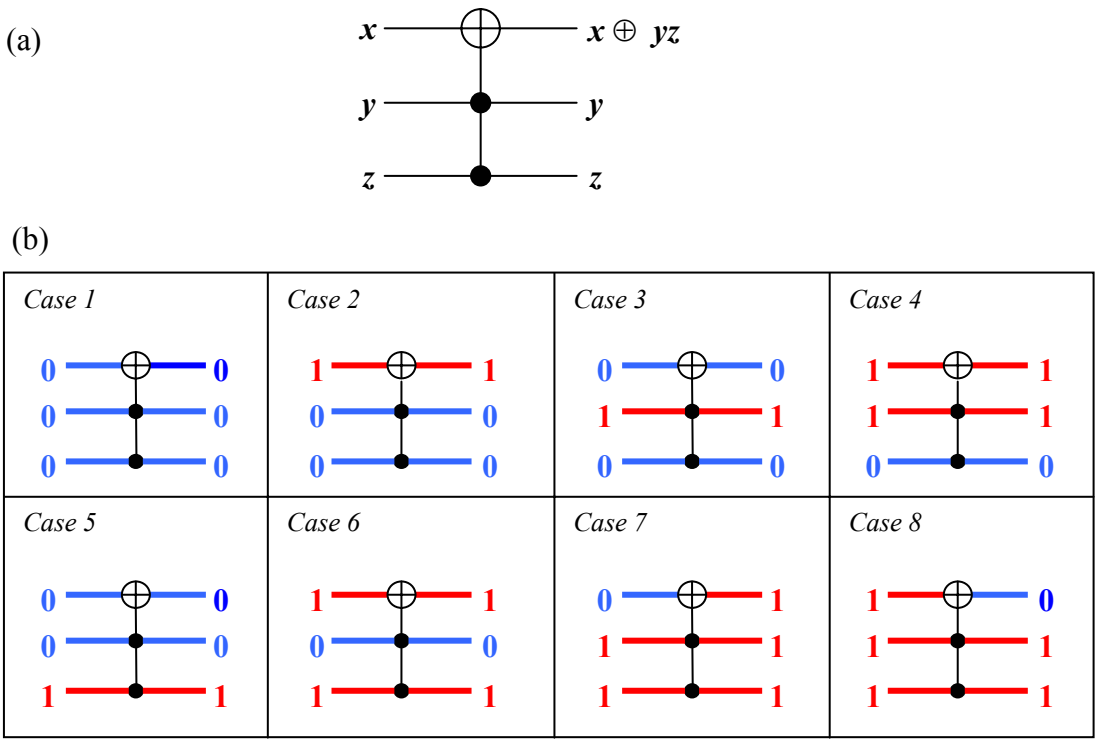


Figure 3.2. Toffoli gate (a) symbol, (b) illustration of behavior

3.3. SWAP Gate

The SWAP gate is represented as a compound of two copies of the symbol × and the vertical connection | (Figure 3.3). It concerns two and only two wires and works in such a way that for both of the wires the logical value in one wire immediately after × is equal to the logical value in the other wire immediately before the other ×.

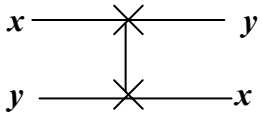


Figure 3.3. SWAP gate

3.4. CNTS Library and Reversible Cascades

The CNTS Library proposed by Shende et al. (2002) takes its name from first letters of the names of the gates: CNOT, NOT, Toffoli and SWAP. It is an intellectual challenge to synthesize a desired Boolean function using exclusively the gates taken from the Library. A lot of effort in the field is devoted to search of efficient automation of Reversible Logic Synthesis. Genetic Algorithms and other heuristics are investigated and some promising results are being reported (Lukas et al. 2002; Lukac & Perkowski 2002, Lukac et al 2003, Perkowski et al 2003).

The recently dominating trend in Reversible Logic Synthesis is to arrange gates into cascades, i.e. arrays of horizontal wires interconnected using consecutive gates. The simplest possible cascade consists of two NOT-gates and returns the same logical value as this provided to its input.

3.5. Constant inputs and garbage outputs

In the realm of Reversible Logic it is seldom possible to use as many inputs and outputs as in classic logic synthesis. There are tree reasons. First, we may want to synthesize a function that by definition has a different numbers of inputs and outputs, usually real life functions have more inputs than outputs. While the basic requirement for reversible circuit is that a number of inputs is equal to the number of outputs. Second, even if the desired function has itself as many inputs as outputs, it may be not a reversible function and has thus to be converted to a reversible functions by adding input signals (set to constant values) and output signals (not used), as shown in (Perkowski et al 2001). The basic reversible gates used in such a new reversible circuit may produce some useful and some useless values. These useless values are called garbage. It is one of the goals of reversible logic synthesis technique to create systematic algorithms with as small garbage as possible. Sometimes the garbage of the entire circuit can be reduced via creating the so-called mirror circuit (Perkowski et al. 2001) but at the price of adding more intermediate variables. Nevertheless, increasing the width of the circuit is sometimes undesirable, for example when the reversible logic is to be implemented as a quantum computing device.

Thus, a smart design is when the designer manages to make use of all outputs produced by the components of his circuit, thus introducing no input signals. The smaller the number of employed wires the better the design of a defined initial function in a reversible cascade. This task is quite difficult and different from standard logic synthesis. So far no good methods exist for reversible synthesis of functions of many variables and high quality algorithms have been created only for few variables. Evolutionary algorithms are some of the most successful methods for reversible design so far, which is in contrast to the classical logic design, where evolutionary methods are not yet competitive to general purpose two- and many-level design tools that are capable of producing better-than-human designs for functions with hundreds of inputs and outputs and where they totally eliminate human logic minimization from modern industrial design processes.

3.7. Fredkin Gate

The Fredkin gate is a 3-wire reversible device that can return various functions of selected input variables, including AND, OR, controlled SWAP and implication. Formally, it converts ***x***, ***y*** and ***z*** into

$\mathbf{x}$, $xz \oplus x'y$, $xy \oplus x'z$, respectively (Figure 3.4). Nevertheless, in order to note its useful properties, let us consider three cases when a given input is constant. The cases are shown in Figures 3.5 and 3.6.

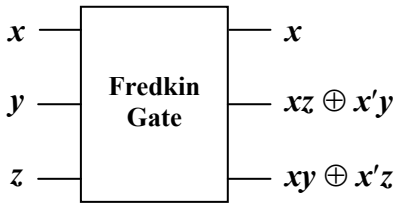


Figure 3.4. Fredkin Gate

Let x be set us constant value. For x= **0**, the Fredkin gate will return **0,** $0z \oplus 1y$ and $0y \oplus 1z$, that equal to **0,** y and z, respectively. The calculations leading to this result are in Figure 3.5. For x= **1**, the Fredkin gate will return **1,** $1z \oplus 0y$ and $1y \oplus 0z$, respectively that equal to **1,** z and y, respectively. This way the Fredkin gate operates as a controlled SWAP gate (Figure 3.5).

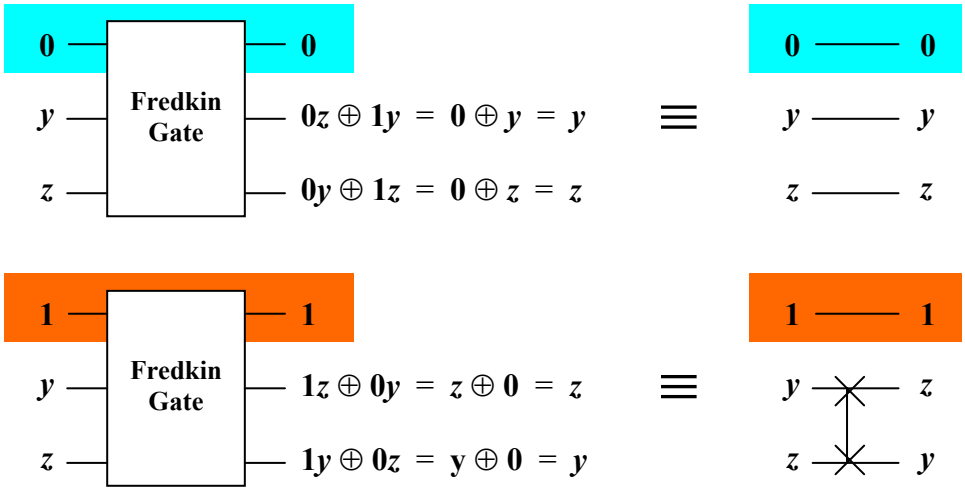


Figure 3.5. Fredkin Gate as a controlled SWAP

Now let y has constant value. For y= **0**, the Fredkin gate will return $\mathbf{x}$, $xz \oplus \mathbf{x}'\mathbf{0}$ and $x\mathbf{0} \oplus \mathbf{x}'z$, respectively that equal to $\mathbf{x}$, xz and $x'z$, respectively. The calculations leading to this result are in Figure 3.6. For y= **1**, the Fredkin gate will return $\mathbf{x}$, $xz \oplus \mathbf{x}'\mathbf{1}$ and $x\mathbf{1} \oplus \mathbf{x}'z$, respectively. Observe that:

$$xz \oplus x'\mathbf{1} = xz \oplus x' = xz \oplus (x \oplus \mathbf{1}) = (xz \oplus x) \oplus \mathbf{1} = (xz \oplus x)' =$$
$$= (xz \oplus x\mathbf{1})' = (x(z \oplus \mathbf{1}))' = (xz')' = x' + z = x \Rightarrow z$$
$$x\mathbf{1} \oplus \mathbf{x}'z = x \oplus \mathbf{x}'z = (x' \oplus 1) \oplus \mathbf{x}'z = (x'\mathbf{1} \oplus \mathbf{1}) \oplus \mathbf{x}'z =$$
$$= (x'\mathbf{1} \oplus \mathbf{x}'z) \oplus \mathbf{1} = (x'\mathbf{1} \oplus \mathbf{x}'z)' = ((x'(1 \oplus z))' = (x'z')' = x + z$$

This way the Fredkin gate appears to be a device that can return Boolean product, Boolean sum, as well as implication (Figure 3.6).

Although the Fredkin gate was invented as a primitive to be used in quantum computing, it can be built of primitives taken from the CNTS Library. Figure 3.7 shows one of solutions.

The solution from Figure 3.7, as well as many other useful reversible gates and circuits have been obtained using evolutionary programming system developed in the Portland Quantum Logic Group (Lukac et al., 2003).

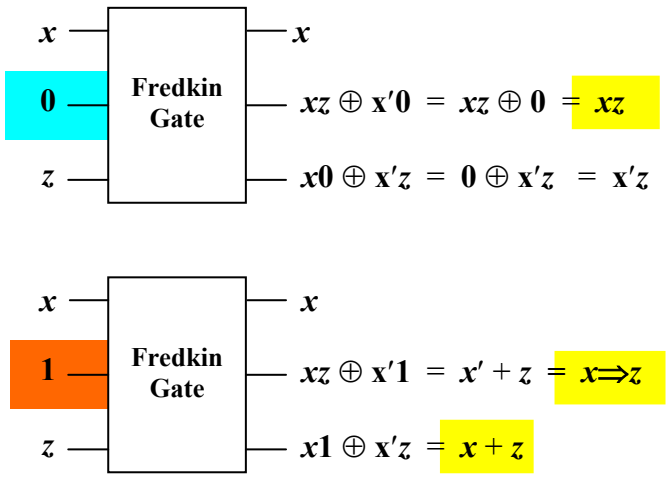


Figure 3.6. Fredkin Gate as AND/OR/implication gate. Similar effect can be obtained for constant z. Indeed, for $z = 0$, the Fredkin gate returns x, $x'z$ and xy, respectively, while for $z = 1$, it returns x, $x + z$ and $x \Rightarrow z$, respectively.

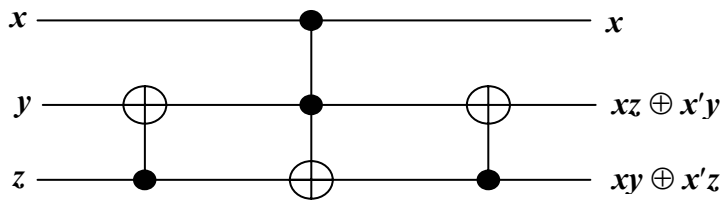


Figure 3.7. Fredkin Gate as a cascade of two Feynmans and one Toffoli.

4. Cellular Automaton (CA) for Reversible Modeling

As a medium for the modeled reversible computing Buller (2003b) employed a cellular automaton (CA) run on ATR's CAM-Brain Machine (CBM). The cellular automaton is 3-dimensional and works according to one simple rule. It can be imagined as a set of cubic cells arranged in such a way that each of the cubes has up to six neighbors. Every cell has a *door* in each of its 6 walls. The set of open doors and the set of closed doors must be determined in the framework of the automaton's initial state and kept unchanged for entire calculation process. Hence, the doors are called *frozen state variables*.

Every cell can be either activated or not activated. Hence, the variable representing activation of a given cell is called *activation* or *pulsing state variable*. A given cell gets activated in time t, if and only if the number of its doors opened toward neighbors activated in time t-1 is equal to just 1.

In order to describe the idea more precisely, let us employ the elementary symmetric function S_1 that returns 1 if and only if one of its six inputs is equal to one (Sasao 1999: 99). Let us assume that binary a_1, a_2, …, a_6 represent activations of six neighbors of a given cell, while binary variables d_1, d_2, …, d_6 are doors toward the neighbors. Let a_0 be activation of the cell itself. The cellular automaton has been adjusted to work in such a way that for every cell $a_{0,t+1} = S_1(d_1a_{1,t}, d_2a_{2,t}, \ldots, d_6a_{6,t})$.

4.1. Graphic representation of cell's state

A convenient way to show a state of a given cell using a graphic planar representation. We propose the "arrow metaphor" where ▷, ◁, ▽, △, × and ○ represent open doors to Western, Eastern, Northern, Southern, Upper and Lower neighbor, respectively, all located in a square representing cell activation (pulsing state variable) (Figure 4.1). Let color of the square be white or red for activation 0 or 1, respectively.

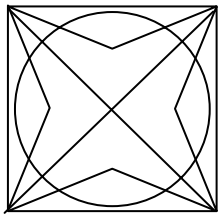


Figure 4.1. Graphic planar representation of such state of CA that all doors are open, while pulsing state variable (activation) is 0. ▷, ◁, ▽, △, × and ○ represent frozen state variables equal to 1, which can be interpreted as open doors to Western, Eastern, Northern, Southern, Upper and Lower neighbor, respectively. This figure was proposed by Michal Joachimczak as a notation in computer aided design of cellular-automatic structures. Color of the square represents pulsing state variable (white for activation 0, red for activation 1).

4.2. Channel and Exor

Channel is elementary structure of the discussed CA. It employs only cells that have only one gate open. Figure 4.2 and 4.3 show two samples of propagation of activation in the channels. If a cell has two and only two gates opened, it can serve as Exor gate (Fig. 4.4).

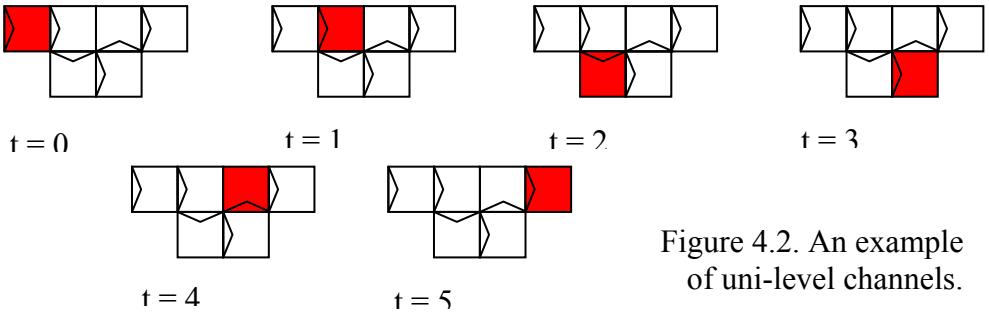


Figure 4.2. An example of uni-level channels.

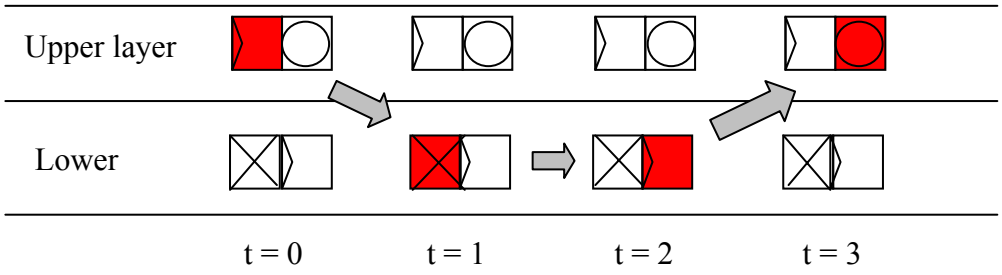


Figure 4.3. An example of multi-level channels.

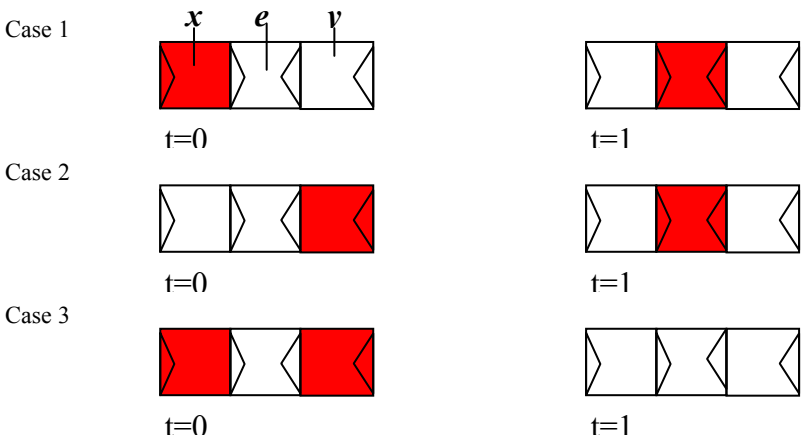


Figure 4.4. CA-based Exor. $e_{t+1} = x \oplus y$. The trivial case $x = 0$, $y = 0$ is not shown since all cells would always be blank.

4.4. Eeckhaut gate

Eeckhaut gate (Eeckhaut & Van Campenhout 2003) serves as an AND gate (Fig. 4.5 and 4.6).

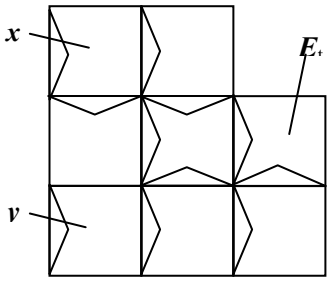


Figure 4.5. Eeckhaut gate which works as delayed AND, i.e. $\boldsymbol{E}_{t+3} = \boldsymbol{x}_t \boldsymbol{y}_t$.

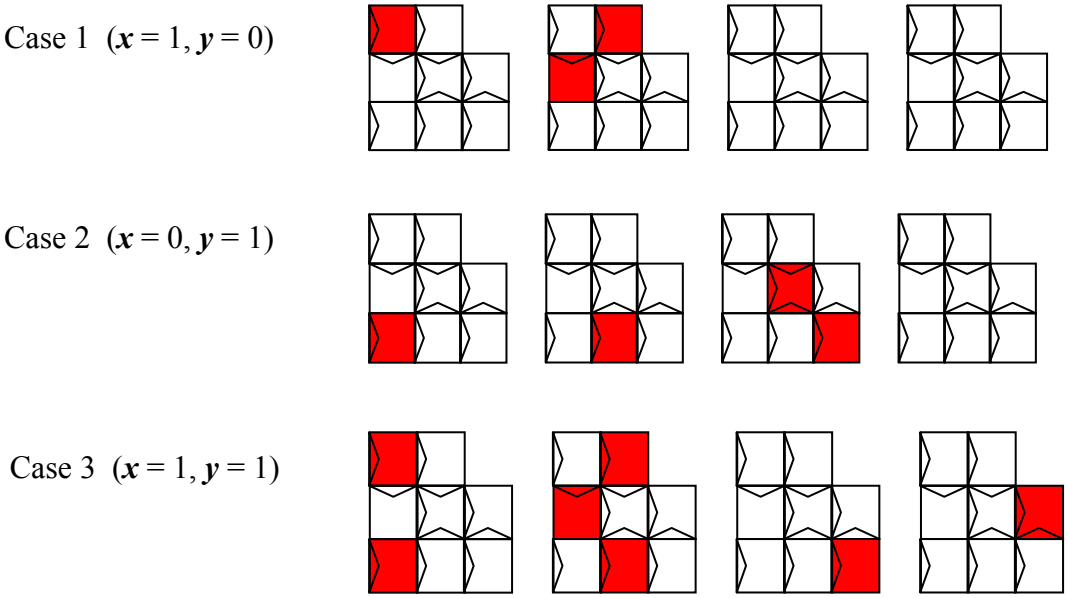
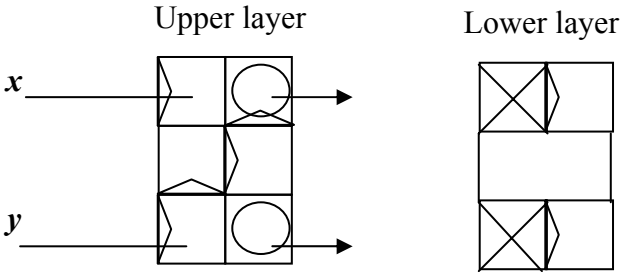


Figure 4.6. Behavior of the Eeckhaut gate

4.5. Reversible gates

Figure 4.7. shows a CA-based model of the Feynman gate (CNOT). Figure 4.8 shows a model of Tofoli gate using an Eeckhaut gate. Based on one Tofoli, two Feynmans and two 4-cell channels one can easily compose a cellular-automatic model of the Fredkin gate (Figure 4.9). The presented structures have been built under the NeuroMaze 3.0 Pro, a software tool for computer aided designing of 3-D β-PPNNs (Pulsed Para-Neural Networks) executable on the ATR's CAM-Brain Machine (CBM) (Buller 2003a). Since the employed cellular automaton does not use all functions offered by the CBM, one of successors of the CBM could be reversible-logic-specific.

Figure 4.7. Feynman gate modeled in CLM paradigm.



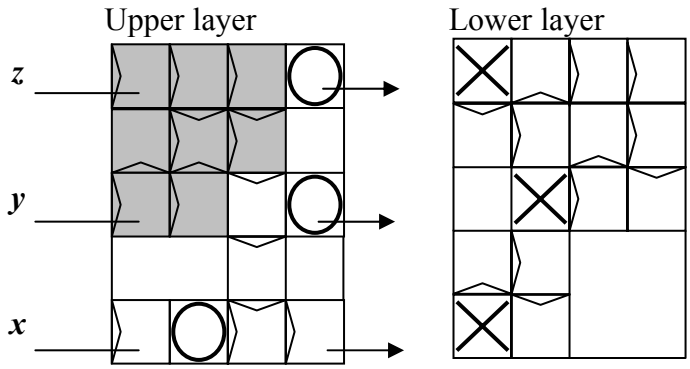


Figure 4.8. Toffoli gate modeled in CLM paradigm. One of components is Eeckhaut gate (grey color)

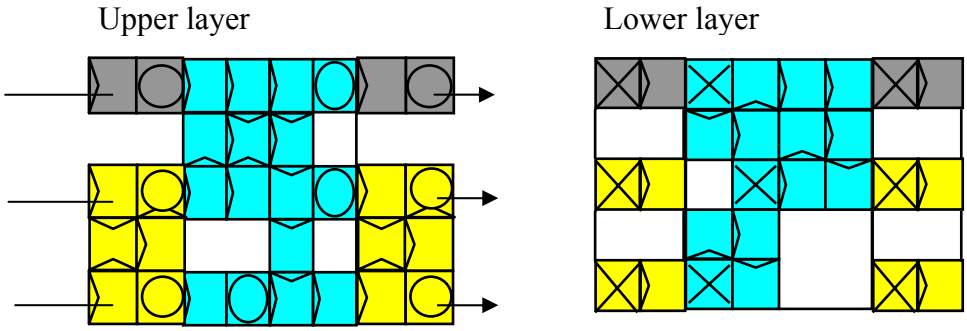


Figure 4.9. Fredkin gate composed in CLM from one Toffoli (blue), two Feynmans (yellow) and two synchronizing channels (grey).

5. Concluding remarks

It was shown, that reversible cascades could be modeled on in a special 3-dimensional cellular automaton with cells having a pulsing state variable, as well as a set of frozen state variables. This automaton is executable on the ATR's CAM-Brain Machine (CBM). Since some successful genetic algorithms for reversible logic synthesis have been built (Lukas & Perkowski 2002; Lukas at al. 2002, Lukac et al 2003, Perkowski et al 2003), the ATR's NeuroMaze 3.0 Pro, a software for computer aided designing of pulsed neural networks could be enhanced to facilitate fully automated creation of large-scale models of reversible cascades. Indeed, owing to regular input-output layouts of the presented reversible gates, they can be attached one to another by a simple program.

Acknowledgement The research is being conducted as a part of the *Research on Human Communication* supported by the Tele-communications Advancement Organization of Japan (TAO).

References

1. Bennett C (1973) Logical reversibility of computation, *IBM Journal of Research and Development, 17*, 525-532.
2. Buller A (2003a) CAM-Brain Machine and Pulsed Para-Neural Networks (PPNN): Toward a hardware for future robotic on-board brains, *Proceedings of the Eighth International Symposium on Artificial Life and Robotics (AROB 8th '03), January 24-26, 2003, Beppu, Oita, Japan*, 490-493.

3. Buller A (2003b) Reversible Cascades and 3D Cellular Logic Machine, Technical Report TR-0012, ATR Human Information Science Laboratories, Kyoto.
4. Buller A, Shimohara K (2003) Artificial Mind. Theoretical Background and Research Directions, *Proceedings of the Eighth International Symposium on Artificial Life and Robotics (AROB 8th '03), January 24-26, 2003, Beppu, Oita, Japan*, 506-509.
5. de Garis H, Buller A, Korkin M, Gers F, Nawa NE & Hough M (1999) ATR's Artificial Brain ("CAM-Brain") Project : A Sample of What Individual "CoDi-1Bit" Model Evolved Neural Net Modules Can Do with Digital and Analog I/O, *Proceedings of The First NASA / DoD Workshop on Evolvable Hardware*, July 19-21, Pasadena, California, 102-110.
6. Eeckhaut H, Van Campenhout J (2003) Handcrafting Pulsed Neural Networks for the CAM-Brain Machine, *Proceedings of the Eighth International Symposium on Artificial Life and Robotics (AROB 8th '02), January 24-26, 2002, Beppu, Oita, Japan*, 494-501.
7. Gershenfeld N (1999) *The nature of Mathematical Modeling*, Cambridge: Cambridge Univ.Press.
8. Korkin M, Fehr G, Jeffrey G (2000) Evolving hadware on a large scale, *Proceedings, The 2nd NASA/DoD Workshop on Evolvable Hardware, July 2000, Pasadena, USA*, 173-181.
9. Liu J (2002) *NeuroMaze User's Guide, Version 3.0*, ATR HIS, Kyoto.
10. Lukac M, Perkowski M (2002) Evolving Quantum Circuits Using Genetic Algorithms, *Proceedings, The 4th NASA/DoD Workshop on Evolvable Hardware, July 2002, Washington DC, USA*, 173-181.
11. Lukac M, Pivtoraiko M, Mishchenko A, Perkowski M (2002) Automated Synthesis of Generalized Reversible Cascades using Genetic Algorithms, *Proceedings, 5th International Workshop on Boolean Problems, Freiberg, Germany, September 19-20*, 33-45.
12. Lukac M, Perkowski M, Pivtoraiko M, (2003) Evolutionary Approach to Quantum and Reversible Logic Synthesis, *Submitted to Artificial Intelligence Review Journal, Februay 2003.*
13. Perkowski M, Lukac M, Pivtoraiko M, Kerntopf P, Folgheraiter M, Lee,Kim H, Hwangbo W, Kim J-W, Choi Y-W, A Hierarchical Approach to Computer-Aided Design of Quantum Circuits, *Proceedings of 6th International Symposium on Representations and Methodology of Future Computing Technologies*, March 10 -11, Trier, Germany, 2003
14. Morita K, Harao M (1989) Computation universality of one-dimensional reversible (injective) cellular automata, *Trans. IEICE, E-72*, 758-762.
15. Morita K, Ueno S (1992) Computation-universal models of two-dimensional 16-state reversible cellular automata, *IEICE Trans. Inf. & Syste., E75-D*, 141-147.
16. Perkowski M, Al-Rabadi A, Kerntopf P, Buller A, Chrzanowska-Jeske M, Mishchenko A, Azad Khan M, Coppola A, Yanushkevich S, Shmerko V, Jozwiak L (2001) A General Decomposition for Reversible Logic, *Proceedings RM'2001, Starkville, Mississippi, USA, August 10-11, 2001*, 119-138.
17. Sasao T (1999) *Switching Theory for Logic Synthesis*, Boston: Kluwer Academic Publishers.
18. Shende VV, Prasad AK, Markov IL, Hayes JP (2002) Reversible Logic Circuit Synthesis, *Proceedings, 11th International Workshop on Logic Synthesis*, 125-130.