The key elements of logic design in ternary quantum-dot cellular automata

Primoz Pecar and Iztok Lebar Bajec

University of Ljubljana, Faculty of Computer and Information Science,
Trzaska 25, 1000 Ljubljana, Slovenia,
primoz.pecar@fri.uni-lj.si

Abstract. The ternary Quantum-dot Cellular Automata (tQCA) were demonstrated to be a possible candidate for the implementation of a future multi-valued processing platform. Recent papers show that the application of the adiabatic pipelining can be used to solve the issues of the tQCA logic primitives. The architectures of the resulting tQCAs become similar to their binary counterparts and the physical design rules remain similar to those for the binary QCA domain. The design of complex processing structures is, however, usually based on logic design. The foundation of logic design is functionally complete set of elementary logic primitives (functions). The currently available tQCA logic primitives, i.e. tQCA majority gate and tQCA inverter gate, do not constitute a functionally complete set. We here present a tQCA implementation of the ternary characteristic functions, which together with the tQCA majority gate and the ternary constants constitute a functionally complete set according to multi-valued Post logic.

Keywords: ternary quantum-dot cellular automaton, multi-valued Post logic, ternary characteristic functions, ternary functionally complete set

1 Introduction

Quantum-dot cellular automata (QCA) were demonstrated to be a promising processing platform that could bridge the technological limitations of current CMOS technology. The foundations of binary QCA (bQCA) processing were set in the early 1990s with the introduction of the bQCA cell followed by its demonstration in a laboratory environment soon after [11]. Later research led to the development of the first binary processor in QCA [1].

Exploiting binary logic, as the basis of elementary computer structures, is a legacy of the technological limitations that computer designers had to overcome in the past. The desired simplicity of data representation was achievable only with the binary system and its realization with a simple two-state switch. The most effective representative of such a two-state switch is the transistor or CMOS circuit. The pioneers of computer design were well aware of the advantages of multi-valued logic. Both, ternary logic and ternary based processing have been extensively researched over the past five decades [2–4, 6, 16, 18]. The actual

working platform designs are, however, unable to keep up with the theoretical advancement. The main obstacle is the shortage of building blocks that offer native ternary support. Currently known solutions are built mostly on CMOS technology, whose binary nature prevents effective and economically justifiable ternary computer design.

The earlier mentioned QCAs were demonstrated as a possible solution to the problem. The first advancement of QCA to native ternary processing was made with the redesign of the bQCA cell. The cell's geometry was altered to allow the representation of three logic values and hence named as the ternary QCA (tQCA) cell [7–9]. Adiabatic pipelining was later introduced to solve the issues of the elementary tQCA logic gates [14, 15]. As a plus the architectures of the proposed ternary QCAs equal those employed for the implementation of the corresponding binary logic functions, which opens up the possibility to use physical design rules similar to those developed for the binary QCA domain.

This is all encouraging but the design of complex processing elements is still at its first steps. In order to promote efficient composition of complex ternary processing elements one should follow a systematic logic design. The later is based on a set of ternary logic functions, that constitute a ternary functionally complete set. Using it one can implement any arbitrary logic function. It is therefore imperative to identify such a set and implement it in the tQCA platform.

The principal functionally complete set of the binary logic system comprises binary conjunction, disjunction and negation and the corresponding binary QCAs are available. Ternary logic, the simplest multi-valued logic, represents a generalization of binary logic so one cannot simply use the binary functionally complete set [2]. Similarly the designs proposed for the ternary CMOS platform cannot be relied upon. These typically employ implementations, like the TXOR gate, that exploit the platform's physical properties.

Here we present a tQCA implementation of the ternary functionally complete set according to chain-based Post logic [2]. The set comprises the ternary majority gate and ternary characteristic functions. While the majority gate was implemented using proven approaches from bQCA design [15], this was not the case for the characteristic functions. They were developed by observing the behavior of simple tQCA segments and their subsequent composition according to physical design rules, thus illustrating the bottom-up approach [5, 9].

In section 2 we present a brief overview of the tQCA, the adiabatic pipelining concept and the elementary tQCAs. In section 3 we continue with a brief overview of the multi-valued Post logic and the corresponding ternary functionally complete set. In section 4 we describe the tQCA implementation of the ternary characteristic functions.

2 tQCA overview

In general, a QCA is a planar array of quantum-dot (QCA) cells [11]. The fundamental unit of a ternary QCA is a tQCA cell [7]. It comprises eight quantum dots

arranged in a circular pattern and two mobile electrons. The electrons can only reside at quantum dots or tunnel between adjacent quantum dots, but cannot tunnel outside the cell. The Coulomb interaction between the electrons causes them to localize in quantum dots that ensure their maximal separation (energetic minimal state). The four arrangements, which correspond to energetic minimal states (ground states), are marked as A, B, C and D (see Fig. 1). Relying on the

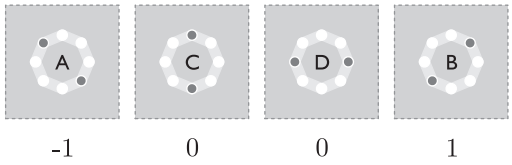


Fig. 1. The four possible arrangements that ensure maximal separation of electrons are mapped to balanced ternary values -1, 0 and 1.

principle of ground state computing, the four states can be interpreted as logic values. We here employ the balanced ternary logic, so A is interpreted as logic value -1, B as logic value 1 and C and D as 0. The arrangement D is typically not allowed (desired) for input or output cells [8, 9, 14]. Placing one or more cells in the observed cell's neighborhood, usually causes one of the arrangements to become the favored ground state. The cell to cell interaction is strictly Coulombic and involves only rearrangements of electrons within individual cells, thus it enables computation. With specific planar arrangements of cells it is possible to mimic the behavior of interconnecting wires as well as logic gates [19]. By interconnecting such building blocks more complex devices capable of processing can be constructed.

The reliability of the behavior of a QCA device depends foremost on the reliability of the switching process, i.e. the transition of a cell's state that corresponds to one logic value to a state that corresponds to another and vice versa. It is achieved by means of the adiabatic switching concept, where a cyclic signal, namely adiabatic clock, is used to control the cells' switching dynamic [20, 14]. The signal comprises four phases. The switch phase serves the cells' gradual update of the state with respect to their neighbors. The hold phase is intended for the stabilization of the cells' states when they are to be passed on to the neighbors that are in the switch phase. The release phase and the relax phase support the cells' gradual preparation for a new switch.

Recent research [15] showed that the correct behavior of tQCA logic gates requires a synchronized data transfer, achievable through a pipelined architecture based on the adiabatic clock. The four phased nature of the clock signal allows any tQCA to be decomposed to smaller stages, or subsystems, controlled by phase shifted signals, each defining its own clocking zone. Let 0 denote the clocking zone controlled by the base signal (usually the clocking zone of the input cells) and $i = \{0, 1, 2, 3\}$ the clocking zone controlled by the base signal

phase shifted by i phases. Subsystems that are in the hold phase act as inputs for subsystems that are in the switch phase. A subsystem, after performing its computation locks its state and acts as the input for the following subsystem. As the transaction and processing in the second subsystem is finished it can lock its state while the first prepares for accepting new inputs. With the correct assignment of cells to clocking zones, the direction of data flow can be controlled. Large regions of nearby cells are usually assigned to the same clocking zone in order to eliminate the challenges that would be caused by attempting to deliver a separate clock signal to every cell.

The latency of a QCA circuit is determined by the number of clocking zones along its critical path. A sequence of four clocking zones causes the delay of one clock cycle. Consequently minimizing the number of clocking zones leads to better designs [13].

The tQCA logic primitive that enables propagation of data from the input cell to the output cell (see Fig. 2) is called tQCA wire. When the input cell's

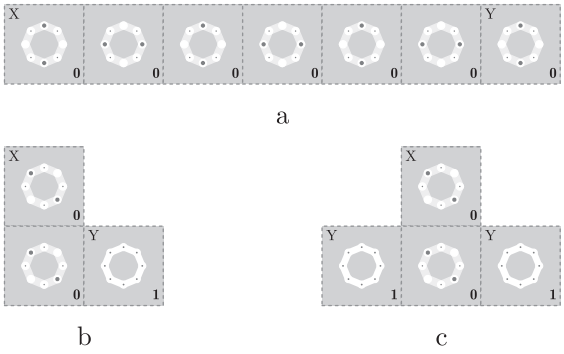


Fig. 2. Using clocking zones for a robust tQCA wire: straight wire (a), corner wire (b) and fan-out (c).

state is A (logic value -1) or B (logic value 1) all cells propagate the same state. However, when the input cell's state is C (logic value 0) the cells propagate the state in an alternating fashion. This effectively means that wires have to be of odd lengths [9]. Having that in mind the tQCA wire can be described as a processing element performing the logic function:

$$y = \mathrm{w}(x) = x, \tag{1}$$

where $x \in \{-1, 0, 1\}$ corresponds to the state of cell X and $y \in \{-1, 0, 1\}$ corresponds to the state of cell Y. The correct behavior of the corner wire and fan-out is ensured by means of a pipeline of two stages, as presented on Figs. 2b and 2c. The first stage ensures the propagation of the input value to the corner, and the second stage ensures its propagation towards the output cell.

3 The functionally complete set

In the 1920s Emil Post introduced a logic system intended for manipulation of multi-valued logic functions, known as chain-based Post logic [17]. A multi-valued logic function is a discrete function whose input and output variables take two or more truth values. Formally, an n variable multi-valued (m-valued) function $f(x_1, \ldots, x_n)$ is a mapping $f : M^n \mapsto M$, with the variables x_i taking truth values from totally ordered set of m elements, M [2]. In case of $m = 2$ Post logic reduces to Boolean logic, which is currently employed in computer design, however this article focuses on the ternary logic system with $m = 3$. Due to the fact that it does not matter how the elements are denoted [18], we here use the balanced set of elements, $M = \{-1, 0, 1\}$.

An arbitrary n variable multi-valued function can be realized as a single primitive or as a composition of primitives. In logic design a primitive is often called a logic gate. The set of all n variable m-valued functions denoted ${O_m}^{(n)}$ consists of $m^{(m^n)}$ functions. In order to make logical design practical is it essential to identify a subset F of functions, i.e. logic gates, whose composition can be used to realize any function from ${O_m}^{(n)}$. Such a set F is denoted a functionally complete set and its elements are denoted elementary logical primitives.

Well known functionally complete sets in Boolean logic are $\{\mathrm{OR}, \mathrm{NOT}\}$ and $\{\mathrm{AND}, \mathrm{NOT}\}$, where OR denotes binary disjunction, AND binary conjunction and NOT binary negation. Post generalized binary disjunction to multi-valued logic. The multi-valued disjunction, denoted $\vee$, is introduced as the multi-variable max operator that returns the highest truth value of all n input variables:

$$y = \bigvee_i x_i \equiv \max(x_i), \quad i = 1, 2, ..., n. \tag{2}$$

The symbol $x_i \in M$ denotes the i-th multi-valued input variable. Similarly the generalization of binary conjunction is multi-valued conjunction, denoted $\wedge$, introduced as multi-variable min operator returning the lowest truth value of n input variables:

$$y = \bigwedge_i x_i \equiv \min(x_i), \quad i = 1, 2, ..., n. \tag{3}$$

With the generalization of the binary negation achieved as

$$y = \neg x \equiv -x, \tag{4}$$

where $x \in M$ we can obtain sets $\{\vee, \neg\}$, $\{\wedge, \neg\}$ or $\{\vee, \wedge, \neg\}$, but in ternary logic these are not functionally complete. Post solved the problem with the function called cyclic negation, denoted $\underline{x}$. It is given as expression

$$y = \underline{x} \equiv ((x + 2) \bmod m) - 1, \tag{5}$$

where $x \in M$. Post proved that the two-variable $\wedge$ and $\underline{x}$ constitute a functionally complete set. After one complete set is identified, others can be found by constructing logic gates by means of which one can implement all the elementary

primitives belonging to the original set. Every set of functions created in this way is also functionally complete.

The selection of the functionally complete set depends on the properties of the chosen implementation platform. Obviously the elementary logic primitives should be as compact as possible so as to minimize the space they occupy and subsequently their cost and they should exploit the platform's advantages.

In order to find optimal solutions logic design of complex processing structures relies upon various minimization techniques. Well known approaches are the Karnaugh map and the Quine-McCluskey algorithm, which both operate on functions expressed in the normal form [2]. The functionally complete set $\{\wedge, \underline{x}\}$ is not suitable for representing functions in a normal form. For this purpose we would need the functionally complete set consisting of $\wedge$, $\vee$ and characteristic functions. The characteristic function termed also the unary literal operator of the multi-valued variable is defined as

$$y = f^z(x) = \begin{cases} 1, \text{ if } x = z \\ -1, \text{ otherwise,} \end{cases} \tag{6}$$

where $x, z \in M$. The disjunctive normal form of the ternary cyclic negation can be given as

$$\underline{x} = (f^{-1}(x) \vee 0) \wedge (f^0(x) \vee 1) \wedge (f^1(x) \vee -1) \tag{7}$$

which shows that two-variable $\wedge$, two-variable $\vee$ together with literals f^{-1}, f^0, f^1 also constitute a functionally complete set [5].

The tQCA implementation of the ternary $\wedge$ and $\vee$ function is based on tQCA majority voting gate. Due to the lack of implementations of other multi-input ternary logic functions, it is currently the fundamental building block in tQCA design. It is constructed as a crossing of three ternary wires and can be implemented in two possible ways [15] presented in Fig. 3. The structure has

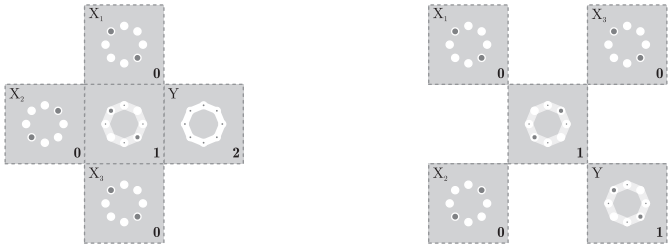


Fig. 3. Two possible pipeline implementations of the ternary majority voting gate.

three input cells denoted X_1, X_2 and X_3, a device cell in the center and an output cell Y. It acts as majority voting logic; the output reflects either the logic value that has been present at the majority of the inputs or logic value 0 if the majority cannot be determined (e.g. in the case of the input combination $x_1 = -1$, $x_2 = 1$, $x_3 = 0$). The described behavior can only be achieved through an appropriate

assignment of clocking zones. The gate's behavior can be described with the logic function:

$$y = \mathrm{m}(x_1, x_2, x_3) = x_1 x_2 \vee x_2 x_3 \vee x_1 x_3, \tag{8}$$

where $x_1, x_2, x_3 \in M$ correspond to the states of input cells X_1, X_2, X_3 and $y \in \mathrm{M}$ corresponds to the state of the output cell Y. The ternary two variable $\vee$ and $\wedge$ functions can be expressed as

$$\begin{aligned} &y = x_1 \vee x_2 = \mathrm{m}(1, x_1, x_2) \\ &\text{and} \\ &y = x_1 \wedge x_2 = \mathrm{m}(-1, x_1, x_2), \end{aligned} \tag{9}$$

where $x_1, x_2, y \in M$. That is, the ternary disjunction can be implemented by fixing one input logic value of the ternary majority voting gate to 1, and the ternary conjunction can be implemented by fixing one input logic value to -1.

4 Implementation of the tQCA characteristic functions

The implementation of ternary characteristic functions is based on the bottom-up approach, i.e. the concept of combining compact and simple structures. Searching over all possible solutions turns out to be computationally complex [12]. Therefore the search was focused on symmetrical structures with odd number of inputs. Various tQCA structures composed of a small number of tQCA cells grouped in to one clocking zone were constructed. Their behavior was analyzed using the ICHA simulation approach [10, 15]. It was based on the following parameters: quantum dots had a diameter of 10 nm, the distance between adjacent quantum dots was 20 nm, cell centers were placed on a 110 nm grid. All other relevant parameters were evaluated for a GaAs/AlGaAs material system.

The truth tables for the most promising tQCA structures were computed by mapping the tQCA cell's states to appropriate ternary logic values, as was presented in chapter 2. The obtained tables represented the search space for iterative deepening based design method [5].

Following the described approach three structures were used. The simplest one is relies on the fact that two cells arranged diagonally, assigned to the same

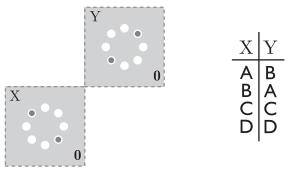
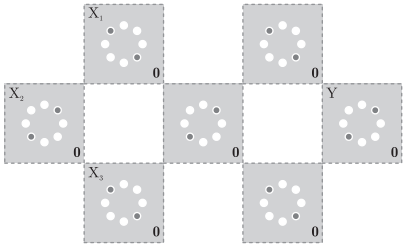


Fig. 4. The ternary inverter (a) and its behavior (b).

one clocking zone, assume alternate states when one is in state A or B and the

same state when one is in state C or D Fig. 4a. The structure performs as a unary function $\mathrm{I} : \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\} \mapsto \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$. Comparing the behavior in Fig. 4b and equation (4) reveals that the given structure evaluates ternary negation where $x \in \mathrm{M}$ corresponds to the state of cell X and $y \in \mathrm{M}$ corresponds to the state of cell Y.

The most useful structure, which made the implementation of ternary characteristic structures feasible is presented in Fig. 5. The structure has three input



a

X_1	X_2	X_3	Y	X_1	X_2	X_3	Y	X_1	X_2	X_3	Y	X_1	X_2	X_3	Y
A	A	A	B	B	A	A	A	C	A	A	B	D	A	A	B
A	A	B	A	B	A	B	A	C	A	B	A	D	A	B	A
A	A	C	B	B	A	C	A	C	A	C	C	D	A	C	B
A	A	D	B	B	A	D	A	C	A	D	A	D	A	D	D
A	B	A	B	B	B	A	B	C	B	A	B	D	B	A	B
A	B	B	B	B	B	B	A	C	B	B	A	D	B	B	A
A	B	C	B	B	B	C	A	C	B	C	C	D	B	C	B
A	B	D	B	B	B	D	A	C	B	D	A	D	B	D	D
A	C	A	B	B	C	A	D	C	C	A	B	D	C	A	B
A	C	B	D	B	C	B	A	C	C	B	A	D	C	B	A
A	C	C	B	B	C	C	A	C	C	C	C	D	C	C	B
A	C	D	B	B	C	D	A	C	C	D	A	D	C	D	D
A	D	A	B	B	D	A	C	C	D	A	B	D	D	A	B
A	D	B	C	B	D	B	A	C	D	B	A	D	D	B	A
A	D	C	B	B	D	C	A	C	D	C	C	D	D	C	B
A	D	D	B	B	D	D	A	C	D	D	A	D	D	D	D

b

Fig. 5. The structure S_1 used for implementation of characteristic functions (a) and its behavior (b). The combinations marked with black rectangle are basis of f^{-1} and f^1 implementation and the combinations marked with shaded rectangle are basis of f^0 implementation.

cells denoted X_1, X_2 and X_3, three device cells and an output cell Y. The structure performs as a three variable function $\mathrm{S}_1 : \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}^3 \mapsto \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$.

Observing its behavior where cells X_1 and X_2 are fixed to state B and cell X_3 acts as input cell (see black rectangle in Fig. 5b) and comparing with equation (6) reveals that the structure can perform as f^{-1} or f^1 literal expressed as

$$\begin{aligned} &y = f^{-1}(x) \equiv \mathrm{S}_1(\mathsf{B}, \mathsf{B}, X) \\ &\text{and} \\ &y = f^1(x) \equiv \mathrm{S}_1(\mathsf{B}, \mathsf{B}, \mathrm{I}(X)), \end{aligned} \tag{10}$$

where $X \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ corresponds to the logic variable $x \in M$ and $y \in M$ corresponds to the state of cell Y (see Fig. 6).

The construction of the structure implementing f^0 literal proved to be more difficult. The basis represents the behavior marked with the shaded rectangles in Fig. 5b. If cell X_2 is fixed to state A and cell X_3 is declared as an inverted input, the state of cell X_1 has to change according to the input state, i.e. in the case of input states A or C the required state is D, otherwise the required state is B. This was achieved with the structure presented in Fig. 7a. The structure has three input cells denoted X_1, X_2 and X_3 and an output cell Y. Observing its

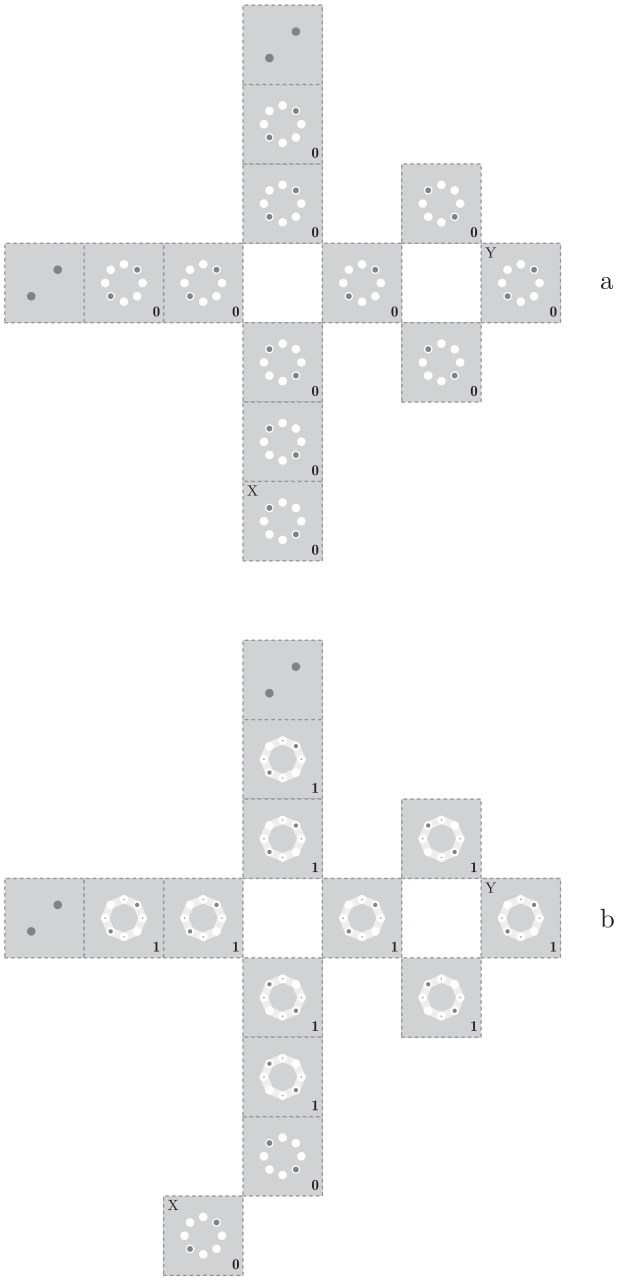
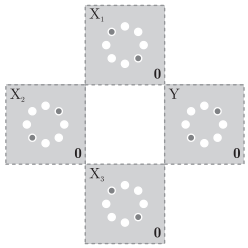


Fig. 6. The structures implementing f^{-1} literal (a) and f^1 literal (b).

behavior (see Fig. 7b) one can notice that the state of output cell Y is not well defined (marked as N) for all possible input combinations. However, for input



X_1	X_2	X_3	Y	X_1	X_2	X_3	Y	X_1	X_2	X_3	Y	X_1	X_2	X_3	Y
A	A	A	B	B	A	A	A	C	A	A	C	D	A	A	B
A	A	B	D	B	A	B	A	C	A	B	A	D	A	B	A
A	A	C	B	B	A	C	A	C	A	C	C	D	A	C	N
A	A	D	D	B	A	D	A	C	A	D	A	D	A	D	D
A	B	A	B	B	B	A	B	C	B	A	B	D	B	A	B
A	B	B	D	B	B	B	A	C	B	B	A	D	B	B	D
A	B	C	B	B	B	C	C	C	B	C	C	D	B	C	B
A	B	D	B	B	B	D	A	C	B	D	N	D	B	D	D
A	C	A	B	B	C	A	N	C	C	A	B	D	C	A	B
A	C	B	D	B	C	B	A	C	C	B	A	D	C	B	D
A	C	C	B	B	C	C	A	C	C	C	C	D	C	C	N
A	C	D	D	B	C	D	A	C	C	D	N	D	C	D	D
A	D	A	B	B	D	A	N	C	D	A	C	D	D	A	B
A	D	B	N	B	D	B	A	C	D	B	A	D	D	B	A
A	D	C	B	B	D	C	C	C	D	C	B	D	D	C	N
A	D	D	B	B	D	D	A	C	D	D	N	D	D	D	D

a b

Fig. 7. The structure S_2 (a) and its behavior (b).

combinations where cell X_1 is fixed to state A and cell X_3 is fixed to state D (see shaded rectangle in Fig. 7b) the structure performs as a unary function $S_2 : \{A, B, C, D\} \mapsto \{B, D\}$. Therefore, the f^0 literal is expressed as

$$y = f^0(x) \equiv S_1(S_2(X), A, I(X)), \tag{11}$$

where $X \in \{A, B, C\}$ corresponds to the logic variable $x \in M$ and $y \in M$ corresponds to the state of cell Y (see Fig. 8).

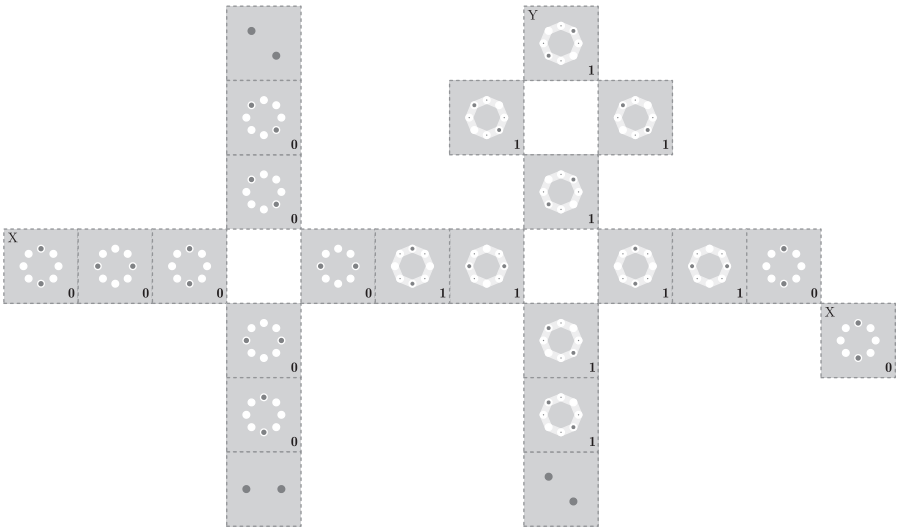


Fig. 8. The structure implementing f^0 literal.

5 Conclusion

A functionally complete set represents the foundation of logical design. Here we present the most general functionally complete set comprised of the ternary characteristic functions, the ternary disjunction and the ternary conjunction. The tQCA that implements the ternary disjunction or conjunction is actually the ternary majority gate, but the tQCAs that implement the ternary characteristic functions have been developed using the bottom-up concept. The presented architectures are not optimal, in the sense of space requirement and the number of required adiabatic phases, but it is a first step. In the next iterations we will search for more optimized solutions, however as the foundations are set we can also focus on the design of the ternary processor building blocks, as well.

Acknowledgment

The work presented in this paper was done at the Computer Structures and Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Slovenia and is part of a PhD thesis that is being prepared by P. Pecar. It was funded in part by the Slovenian Research Agency (ARRS) through the Pervasive Computing research programme (P2-0395).

References

1. Bernstein, G., Bazan, G., Chen, M., Lent, C., Merz, J., Orlov, A., Porod, W., Snider, G., Tougaw, P.: Practical issues in the realization of quantum-dot cellular automata. Superlattices and Microstructures 20, 447–559 (1996)
2. Dubrova, E., Jamal, Y., Mathew, J.: Non-silicon non-binary computing: Why not? In: 1st Workshop on Non-Silicon Computation. pp. 23–29. Boston, Massachusetts (2002)
3. Fitting, M., Orlowska, E. (eds.): Beyond two: Theory and applications of multiple-valued logic. Physica-Verlag, Heidelberg (2003)
4. Frieder, G., Luk, C.: Ternary computers: Part 1: Motivation for ternary computers. In: 5th annual workshop on Microprogramming. pp. 83–86. Urbana, Illinois (september 1972)
5. Janez, M., Lebar Bajec, I., Pecar, P., Jazbec, A., Zimic, N., Mraz, M.: Automatic design of optimal logic circuits based on ternary quantum-dot cellular automata. WSEAS Trans. Cir. and Sys. 7, 919–928 (2008)
6. Knuth, D.E.: The Art of Computer Programming, vol. 2. Addison-Wesley, Reading, 2 edn. (1981)
7. Lebar Bajec, I., Mraz, M.: Towards multi-state based computing using quantum-dot cellular automata. In: Teucher, C., Adamatzky, A. (eds.) Unconventional Computing 2005: From Cellular Automata to Wetware. pp. 105–116. Luniver Press, Beckington (2005)
8. Lebar Bajec, I., Zimic, N., Mraz, M.: The ternary quantum-dot cell and ternary logic. Nanotechnology 17(8), 1937–1942 (2006)

9. Lebar Bajec, I., Zimic, N., Mraz, M.: Towards the bottom-up concept: extended quantum-dot cellular automata. Microelectronic Engineering 83(4-9), 1826–1829 (2006)
10. Lent, C., Tougaw, P.: Lines of interacting quantum-dot cells: A binary wire. Journal of Applied Physics 74(10), 6227–6233 (1993)
11. Lent, C., Tougaw, P., Porod, W., Bernstein, G.: Quantum cellular automata. Nanotechnology 4, 49–57 (1993)
12. Lusth, J.C., Dixon, B.: A characterization of important algorithms for quantum-dot cellular automata. Inf. Sci. 113, 193–204 (1999)
13. Niemier, M.T., Kogge, P.M.: Problems in designing with QCAs: Layout = timing. International Journal of Circuit Theory and Applications 29, 49–62 (2001)
14. Pecar, P., Mraz, M., Zimic, N., Janez, M., Bajec, I.L.: Solving the ternary QCA logic gate problem by means of adiabatic switching. Japanese Journal of Applied Physics 47(6), 5000–5006 (2008)
15. Pecar, P., Ramsak, A., Zimic, N., Mraz, M., Lebar Bajec, I.: Adiabatic pipelining: A key to ternary computing with quantum dots. Nanotechnology 19(49), 495401 (2008)
16. Porat, D.I.: Three-valued digital systems. Proceedings of IEE 116(6), 947 – 954 (1969)
17. Post, E.L.: Introduction to a general theory of elementary propositions. American Journal of Mathematics 43(3), 163–185 (1921)
18. Rine, D.C. (ed.): Computer science and multiple-valued logic: Theory and applications. North-Holland, Amsterdam, second edn. (1984)
19. Tougaw, P.D., Lent, C.S.: Logical devices implemented using quantum cellular automata. Journal of Applied Physics 75(3), 1818–1825 (1994)
20. Tougaw, P., Lent, C.: Dynamic behaviour of quantum cellular automata. Journal of Applied Physics 80(8), 4722–4736 (1996)