Computer-Aided Design of Ternary Quantum-dot Cellular Automata Circuits

MIHA JANEŽ, IZTOK LEBAR BAJEC, PRIMOŽ PEČAR, ANDREJ JAZBEC, NIKOLAJ ZIMIC
and MIHA MRAZ
University of Ljubljana
Faculty of Computer and Information Science
Tržaška cesta 25, 1000 Ljubljana
SLOVENIA

Abstract: This article builds on the ternary quantum-dot cell, which is an extension of the classic binary cell. These cells are basic building blocks of quantum-dot cellular automata. They can be be used to construct structures with input and output cells, so when using the ternary quantum-dot cells a structure is an implementation of some ternary logic function. Furthermore the structures can be interconnected and act as building blocks of the implementation of an arbitrary logic function. This paper presents a computer-aided design tool that finds an optimal implementation of a circuit. The search is based on the concept of iterative deepening. Since searching over all possible solutions would take too much time, heuristics are used to reduce the required computation time.

Key–Words: Quantum-dot cellular automata, Ternary quantum-dot cell, Computer-aided design, Ternary logic, Logic circuits, Iterative deepening, Heuristics

1 Introduction

The quantum-dot cellular automaton (QCA) is a possible future nano-scale processing platform that could perform useful computation based on a new computing paradigm [1]. QCA devices are smaller and faster than present computers and dissipate much less energy. A binary QCA is composed of binary quantum-dot (bQCA) cells, each of them having four quantum dots distributed in a square like pattern over its surface. An individual cell contains also two electrons, which can tunnel among the quantum dots. Because of the Coulomb interaction the electrons tunnel between adjacent dots. Their possible arrangements determine different states of a cell.

The use of binary logic in computers was historically necessary only due to the limited technology available at the time. This, however, does not apply for QCA. Lebar Bajec et al. extended the bQCA cell and introduced the ternary quantum-dot (tQCA) cell capable of multi-valued processing [2, 3]. The tQCA cell has eight quantum dots distributed on its surface in a circular pattern and contains two electrons. The positions of electrons determine four possible configurations, thus the cell can be in one of four different states. The tQCA cell is shown on figure 1(a) and figure 1(b) presents its four states denoted A, B, C and D respectively. The ternary representation of numbers is proposed as the most efficient [4], therefore we based

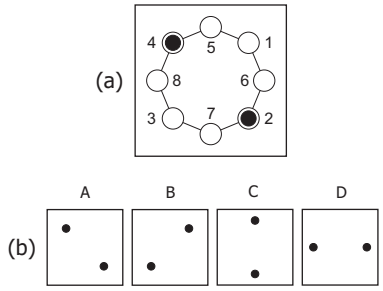


Figure 1: (a) The tQCA cell. (b) The four possible cell states.

our logic analysis on Łukasiewicz's ternary logic and set the logic values of the states as {A, B, C, D} = {0, 1, $\frac{1}{2}$, $\frac{1}{2}$} [5].

The algorithms for an exact simulation of large tQCA circuits are inefficient [6], therefore we propose a novel methodology for the design of complex tQCA circuits, based on building blocks with designated inputs and outputs. These are simple tQCA structures constructed of a small number of cells grouped in clocking zones. We developed a computer-aided design (CAD) tool that constructs a desired tQCA circuit with available blocks. The building blocks are composed of only a few cells so their behavior can be more accurately simulated fast. The implemented

tool finds an optimal solution in terms of spatial redundancy (the minimal number of tQCA cells used) and maximal speed of processing (the minimal number of clocking zones in a circuit).

2 The tQCA structures

By placing tQCA cells adjacent to each other various cellular automata with defined input and output cells can be constructed. Cells are partitioned into clocking zones to exploit the adiabatic pipelining [7, 8]. Each zone cycles through four clock phases, denoted Switch, Hold, Release and the Relax phase. The tQCA processing performed by the tunneling of electrons takes place in the Switch phase, therefore the result may be available before the clock cycles through all phases. The number of clock cycles needed for computation is determined by the number of clocking zones used in the tQCA circuit. After the setting up of the input cells' states and the relaxation of the tQCA to its ground state the result of the computation is read by determining the states of the output cells. For each of the constructed structures we computed the truth tables, i.e. for every input configuration the corresponding output state was found. Calculations were made using the tQCA simulation methods [8].

Figure 2 presents the simplest tQCA structures. On figure 2 (a) are the tQCA ternary logic inverter and its truth table. The cells of the tQCA inverter have to be divided into two clocking zones for correct operation, thus the result of the computation is available after the second clock phase. The truth table of the tQCA inverter corresponds to the truth table of negation in ternary logic. Figure 2 (b) shows the behavior of the tQCA wire at every possible input. The input cell, internal cells and the output cell are assigned to three different clocking zones, which determine the direction of data flow. Upon closer inspection it is evident that the wire must have an odd number of cells to perform correctly. A useful structure is the tQCA majority gate that can be used to implement ternary logic disjunction and conjunction, similar to the binary QCA majority gate [9, 10]. The tQCA majority gate is presented on figure 3 together with corresponding truth table.

Each tQCA structure implements some ternary logic function. Input variables are entered into the structure by setting appropriate states of input cells. After the processing, the output is obtained by reading the state of the output cell. The equations of functions, which are implemented by tQCA structures, are

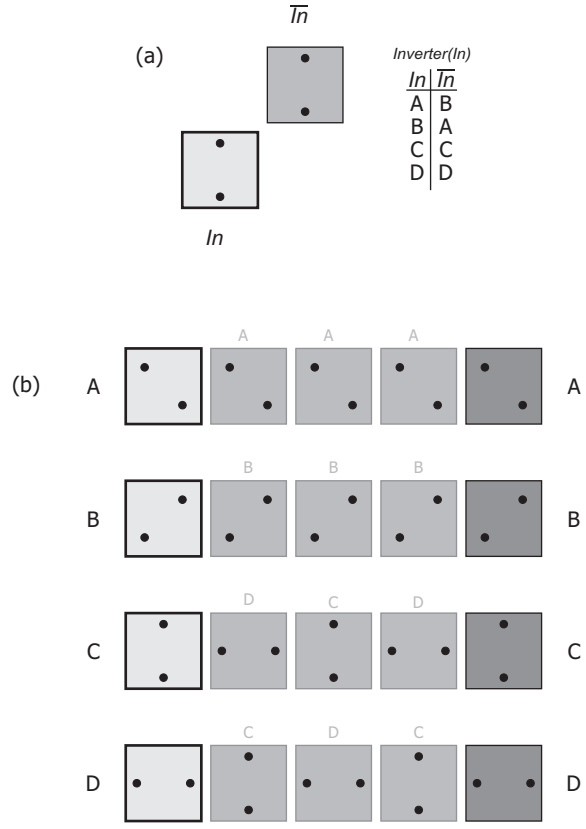
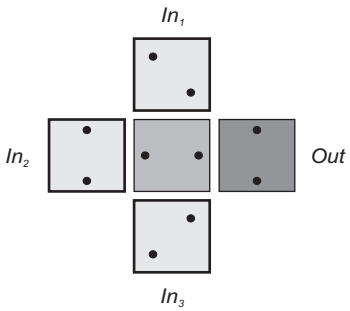


Figure 2: (a) The tQCA ternary logic inverter. The input cell is labeled In and the output cell is labeled $\overline{In}$. Different background shades of the cells denote different clocking zones. (b) Behavior of the tQCA wire at different inputs. In each case the input cell is on the left side of a wire and it is outlined with thick lines. The internal cells are to the right of the input cell and outlined with thin lines. The output cell is on the right end of a wire.

written in the form

$$Output = S_i(Input_1, Input_2, ..., Input_m), \quad i = 1, 2, ..., n \tag{1}$$

S_i denotes the function that is computed by i–th out of the n constructed tQCA structures which has m input cells. The number of input cells m varies between structures, e.g. the tQCA inverter has one and the tQCA majority gate has three input cells. $Output$ is a vector of outputs of the logic function with k input variables $Variable_1$, $Variable_2$, ..., $Variable_k$ and contains 4^k values. These are the tQCA cell states corresponding to the ternary logic values in the output column in the truth table of the function. $Input_j$, $j = 1, 2, ..., m$, can be either a constant state, a vector of $Variable$ or a vector of function outputs computed



In1	In2	In3	Out
A	A	A	A
A	A	B	A
A	A	C	A
A	A	D	A
A	B	A	A
A	B	B	B
A	B	C	C
A	B	D	D
A	C	A	A
A	C	B	C
A	C	C	C
A	C	D	A
A	D	A	A
A	D	B	D
A	D	C	A
A	D	D	D

In1	In2	In3	Out
B	A	A	A
B	A	B	B
B	A	C	C
B	A	D	D
B	B	A	B
B	B	B	B
B	B	C	B
B	B	D	B
B	C	A	C
B	C	B	B
B	C	C	C
B	C	D	B
B	D	A	D
B	D	B	B
B	D	C	B
B	D	D	D

In1	In2	In3	Out
C	A	A	A
C	A	B	C
C	A	C	C
C	A	D	A
C	B	A	C
C	B	B	B
C	B	C	C
C	B	D	B
C	C	A	C
C	C	B	C
C	C	C	C
C	C	D	C
C	D	A	A
C	D	B	B
C	D	C	C
C	D	D	D

In1	In2	In3	Out
D	A	A	A
D	A	B	D
D	A	C	A
D	A	D	D
D	B	A	D
D	B	B	B
D	B	C	B
D	B	D	D
D	C	A	A
D	C	B	B
D	C	C	C
D	C	D	D
D	D	A	D
D	D	B	D
D	D	C	D
D	D	D	D

Figure 3: The tQCA majority gate and its truth table. Input cells are labeled In_1, In_2, In_3 and the output cell is labeled Out.

by some tQCA structure. In case of the tQCA logic inverter with one input variable, vectors are described by the relations (2), (3) and (4):

$$Variable = [\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}] \tag{2}$$

$$Output = [\mathrm{B}, \mathrm{A}, \mathrm{C}, \mathrm{D}] \tag{3}$$

and

$$Output = Inverter(Variable) \tag{4}$$

The equation of ternary logic conjunction is written as

$$Output = M(V_1, V_2, \mathrm{A}) \tag{5}$$

In equation (5) input variables are labeled V_1 and V_2, whereas M denotes the tQCA majority gate. The third input cell is set to constant state A.

Each input can be an output of another tQCA structure. The tQCA logic circuits are constructed by connecting the output cell of the tQCA structure to the input cell of another structure with the tQCA wire. Thus the state of the output cell is transfered to the input of the next tQCA structure.

3 Design of minimal tQCA logic circuits

Although any ternary function can be written in the normal form as the composition of functions in a functionally complete set, the realization of this form may be constituted of many tQCA structures. Our goal was to find the smallest and the fastest tQCA circuit that implements the desired ternary logic function. The optimal logic circuit consists of the minimal number of tQCA structures and has the minimal total number of clocking zones. It is preferred that structures process in parallel, so that outputs of the earlier stage of processing are available to the next stage at the same time.

We designed the CAD tool that finds the tQCA logic circuit composed of predefined tQCA structures for an arbitrary ternary logic function with an arbitrary number of variables. The input data are vector $Output$ and the truth tables of tQCA structures that will be used to construct the circuit. The number of variables is computed from the length of vector $Output$. The result is constructed tQCA logic circuit, written in the form of a relation (1). The basic idea of the developed tool is searching the space of solutions by iterative deepening. The exhaustive search is inefficient because the complete space of solutions is very large, therefore we used heuristic methods to reduce it. The tool is implemented in the Prolog programming language because of its mechanisms for pattern matching, automatic backtracking and others that we found appropriate for the task [11, 12, 13].

The truth tables of the basic structures are given to Prolog as facts. This is very useful as additional structures can be added by simply designing a new structure, computing its truth table by simulation and adding it to the existing facts. The input data is the truth table of the logic function to be realized. Furthermore the input can be incompletely specified. Using the tool it is easy to handle the do not cares, as these are represented as anonymous variables in Prolog.

The search space is reduced by not considering configurations that always output a constant, e.g. M(A, A, $Variable$) that always outputs the vector consisting of states A, independent of the value of vector $Variable$. It is evidently more efficient to assign the constant state to the input cell instead of computing it by such function. Furthermore only one of the configurations that have identical truth tables is used, e.g. M(A, $Variable_1$, $Variable_2$) and M($Variable_1$, $Variable_2$, A) both compute the same output. The number of all possible configurations of the tQCA majority gate with three inputs, considering four constant states and three different variable vectors, is 7^3=343. Using the described technique,

In_1	In_2	$In_1 \leftrightarrow In_2$
A	A	B
A	B	A
A	C	C
B	A	A
B	B	B
B	C	C
C	A	C
C	B	C
C	C	B

→

In_1	In_2	$In_1 \leftrightarrow In_2$
A	A	X_B
A	B	X_A
A	C	X_C
B	A	X_A
B	B	X_B
B	C	X_C
C	A	X_C
C	B	X_C
C	C	X_B

Figure 4: Mapping the output states into corresponding states, represented as anonymous variables in Prolog. Figure shows the mapping in case of Łukasiewicz's equivalence function.

the number of appropriate configurations is reduced to merely 25.

The number of levels of the tQCA logic circuits is the maximal number of sequentially connected structures, e.g. the circuit described by the form (6) has two levels:

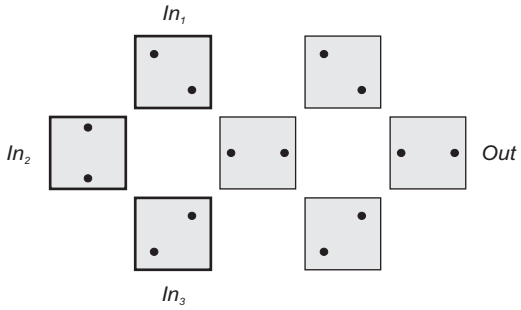
$$M(M(V_1, V_2, \mathrm{A}),\ M(V_1, V_2, \mathrm{B}),\ \mathrm{C}) \qquad (6)$$

V_1 and V_2 indicate variable vectors, whereas A, B and C are constant states. By iterative deepening the tool first finds circuits with the minimal number of levels. From obtained solutions only the circuits with minimal number of used structures and clocking zones are chosen. More than one different optimal circuits that compute the desired ternary function can be found.

Since it can be hard to find a complex circuit, the CAD tool uses heuristic methods to speed up the search. One of the heuristics used is to define as many anonymous variables as there are different output states and represent the truth table using them. In other words all of the input combinations that result in the same output state use the same anonymous variable. State D is only allowed for internal cells and does not appear in the input cells. Figure 4 shows the procedure in case of Łukasiewicz's equivalence function. State A is mapped to variable $\mathrm{X_A}$, state B to $\mathrm{X_B}$ and state C to variable $\mathrm{X_C}$. Through backtracking Prolog determines the values of variables that are mutually exclusive. The states may be temporarily mapped to different ternary logic values as initially, thus the obtained solution is not the desired function. In this case the tool maps the output states into the desired states with the fast method that finds a function of one variable, which correctly maps the intermediate output states to the final and correct output states. With this procedure the tool may not find an optimal solution, but it finds a good solution fast.

Table 1: Truth table of the ternary cyclic negation function.

X	0	$\frac{1}{2}$	1
Logic values of $\underline{X}$	$\frac{1}{2}$	1	0
Output cell states of $\underline{X}$	C	B	A



In_1	In_2	In_3	Out
A	A	A	B
A	A	B	A
A	A	C	B
A	A	D	B
A	B	A	B
A	B	B	B
A	B	C	B
A	B	D	B
A	C	A	B
A	C	B	D
A	C	C	B
A	C	D	B
A	D	A	B
A	D	B	C
A	D	C	B
A	D	D	B

In_1	In_2	In_3	Out
B	A	A	A
B	A	B	A
B	A	C	A
B	A	D	A
B	B	A	B
B	B	B	A
B	B	C	A
B	B	D	A
B	C	A	D
B	C	B	A
B	C	C	A
B	C	D	A
B	D	A	C
B	D	B	A
B	D	C	A
B	D	D	A

In_1	In_2	In_3	Out
C	A	A	B
C	A	B	A
C	A	C	C
C	A	D	A
C	B	A	B
C	B	B	A
C	B	C	C
C	B	D	A
C	C	A	B
C	C	B	A
C	C	C	C
C	C	D	A
C	D	A	B
C	D	B	A
C	D	C	C
C	D	D	A

In_1	In_2	In_3	Out
D	A	A	B
D	A	B	A
D	A	C	B
D	A	D	D
D	B	A	B
D	B	B	A
D	B	C	B
D	B	D	D
D	C	A	B
D	C	B	A
D	C	C	B
D	C	D	D
D	D	A	B
D	D	B	A
D	D	C	B
D	D	D	D

Figure 5: The tQCA structure Cf and its truth table. All cells belong to the same clocking zone.

4 Results

The disjunctive normal form of the ternary cyclic negation, defined in the table 1, is quite lengthy:

$$\begin{aligned} \underline{X} = &(f^{\mathrm{A}}(X)\ \wedge\ \mathrm{C}) \vee \\ &(f^{\mathrm{B}}(X)\ \wedge\ \mathrm{A}) \vee \\ &(f^{\mathrm{C}}(X)\ \wedge\ \mathrm{B}) \end{aligned} \qquad (7)$$

In equation (7) the symbol $\wedge$ denotes ternary logic conjunction, defined as the minimum of the input logic values. The symbol $\vee$ denotes ternary disjunction, defined as the maximum of the input logic values. Input variable is labeled X and $\underline{X}$ is the notation of the cyclic negation. The functions $f^{\mathrm{A}}(X)$, $f^{\mathrm{B}}(X)$ and $f^{\mathrm{C}}(X)$ denote the characteristic functions for states A (logic value 0), B (logic value 1) and C

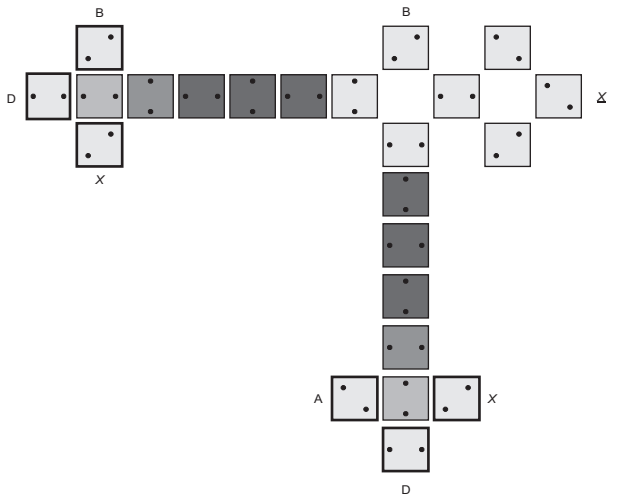


Figure 6: The optimal implementation of the ternary cyclic negation with tQCA structures. The output $\underline{X}$ is computed after 5 clock cycles.

(logic value $\frac{1}{2}$) respectively. They are defined by the relation

$$f^{Y}(X) = \begin{cases} \mathrm{B} & \text{if } X{=}Y, \\ \mathrm{A} & \text{otherwise} \end{cases} \qquad Y = \mathrm{A}, \mathrm{B}, \mathrm{C} \qquad (8)$$

Figure 5 presents the tQCA structure denoted Cf and corresponding truth table, determined by thorough simulation of structure's behavior. It enables simple implementations of tQCA circuits which compute characteristic functions for the states A and B. Circuits for $f^{\mathrm{A}}(X)$ and $f^{\mathrm{B}}(X)$ are given by equations (9) and (10) respectively.

$$f^{\mathrm{A}}(X) = Cf(\mathrm{B}, \mathrm{B}, X) \qquad (9)$$

$$f^{\mathrm{B}}(X) = Cf(\mathrm{B}, \mathrm{B}, Inverter(X)) \qquad (10)$$

The straightforward realization of the ternary cyclic negation by equation (7) is composed of many structures sequentially connected in numerous levels. However our tool finds an optimal solution, composed of only three structures connected in two levels as presented in figure 6. It can be written in equation form as

$$\underline{X} = Cf(\mathrm{B},\ M(\mathrm{B}, \mathrm{D}, X),\ M(\mathrm{A}, \mathrm{D}, X)) \qquad (11)$$

The ternary logic conjunction and disjunction on the other hand can be realized using the majority gate analogous to the realization of binary AND and OR in the binary QCA [2, 3, 9, 10]. With implemented heuristic methods the CAD tool is able to find even complex tQCA circuits for implementation of ternary logic functions that do not have a simple solution. One of these is Łukasiewicz's equivalence, implemented by the tQCA circuit described by complex form:
$X \leftrightarrow Y$ = M(B,M(B,M(A,X,Y),
M(A,$Inverter(X)$,$Inverter(Y)$)),
Cf(A,A,M(B,M(B,X,Y),Cf(X,A,Y)))).

5 Conclusion

This article describes the methodology used to design (sub)optimal tQCA logic circuit that computes an arbitrary ternary logic function. The solution is composed of predefined tQCA structures. One criterion for optimality is the number of interconnected structures, so that an optimal realization of a function occupies the smallest possible area. The second criterion is the number of levels in which the structures are connected. Indeed every level in the implementation increases the time needed to compute an output of a function. The third criterion, connected to the previous two, is the number of clocking zones in the tQCA circuit. Obviously the fastest and therefore optimal circuit computes the result after the minimal number of clock cycles. We present the CAD tool that is designed to find the implementation of the ternary logic function given as its input. The application, developed in programming language Prolog, searches for a solution based on the concept of iterative deepening. Since checking every possible configuration consumes too much computation time, we improved search techniques with heuristic methods. In this way the tool may find a suboptimal solution but the computation time is greatly reduced.

The work presented in this paper was performed at the Computer Structures and Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Slovenia and is part of a PhD thesis being prepared by Miha Janež.

References:

[1] C.S. Lent, P.D. Tougaw, W. Porod and G.H. Bernstein, Quantum cellular automata, *Nanotechnology* 4, 1993, pp. 49–57.

[2] I. Lebar Bajec, N. Zimic and M. Mraz, The ternary quantum-dot cell and ternary logic, *Nanotechnology* 17, 2006, pp. 1937–1942.

[3] I. Lebar Bajec, N. Zimic and M. Mraz, Towards the bottom-up concept: Extended quantum-dot cellular automata, *Microelectronic Engineering* 83, 2006, pp. 1826–1829.

[4] B. Hayes, Third Base, *American Scientist* 89, 2001, pp. 490–494.

[5] J. Łukasiewicz and L. Borkowski, *Selected Works*, North-Holland Publishing Company, Amsterdam, 1970.

[6] J. C. Lusth, B.Dixon, A characterization of important algorithms for quantum-dot cellular automata, *Journal of Information Sciences* 113, 1999, pp. 193–204.

[7] C. S. Lent and P. D. Tougaw, A device architecture for computing with quantum dots, *Proceedings of the IEEE* 85, 1997, pp. 541–557.

[8] P. Pečar, M. Mraz, N. Zimic, M. Janež and I. Lebar Bajec, Solving the Ternary Quantum-dot Cellular Automata Logic Gate Problem by Means of Adiabatic Switching, *Japanese Journal of Applied Physics* 47, 2008, in press.

[9] I. Amlani, A. O. Orlov, G. Toth, G. H. Bernstein, C. S. Lent and G. L. Snider, Digital Logic Gate Using Quantum-Dot Cellular Automata, *Science* 284, 1999, pp. 289–291.

[10] P. D. Tougaw and C. S. Lent, Logical devices implemented using quantum cellular automata, *Journal of Applied Physics* 75, 1994, pp. 1818–1825.

[11] I. Bratko, *Prolog programming for artificial intelligence (Third edition)*, Addison-Wesley Longman Publishing Company, Boston, 2001.

[12] P. W. Horstmann and E. P. Stabler, Computer aided design (CAD) using logic programming, *Proceedings of the 21st conference on Design automation*, 1984, pp. 144–151.

[13] J. C. Gonzalez, M. H. Williams and I. E. Aitchison, Evaluation of the Effectiveness of Prolog for a CAD Application, *IEEE Computer Graphics and Applications* 4, 1984, pp. 67–75.