Designing of QCA Schemes by Boundary Functions

Vladimir Ostrovsky
Tel Aviv University,
Tel Aviv 69978, Israel
e-mail:
vladio@post.tau.ac.il

Ilya Levin
Tel Aviv University,
Tel Aviv 69978, Israel
e-mail:
i.levin@computer.org

Osnat Keren
Bar Ilan University,
Ramat Gan, Ramat Gan
52900, Israel,
e-mail:
kereno@eng.biu.ac.il

Abstract

A novel universal QCA gate is introduced and called a boundary comparator. This gate implements a Boolean function in its boundary form, as a superposition of elementary boundary functions or as threshold functions having weights equal to integer powers of 2. An array of the boundary comparators is called Comparator based Programmable Array (CPA) and forms a homogeneous regular structure programmable for implementing any logic function. The paper describes a) presenting a Boolean function in its boundary form; b) structure of the boundary comparator, and c) structure of the CPA. The proposed CPA is testable, reparable and debuggable.

1. Introduction

Progress in the manufacture of digital computational and control devices, their microminiaturization, and the growing number of new applications, change the basic requirements to such devices. An increase in complexity of a scheme implemented by a chip results in a rise of the chip's cost, mainly due to the fact that a wider chip area decreases the yield. In view of that, available redundancy and simplicity of replacing any defective elements with the redundant elements have important parts in the play. These same properties characterize the chip reparability i.e., its lifetime.

Another important parameter of the chip is its testability i.e., the suitability for revealing faults. Testability characterizes suitability of the chip both for verification, and for concurrent checking or off-line testing. Microminiaturization complicates the task of checking due to the following two reasons. Firstly, internal poles of the chip are hardly accessible thus decreasing the amount of information, which is required for making conclusions about its workability, presence or absence of faults, and others. Secondly, it becomes more and more difficult to allocate structural elements in the more complex chip. In turn, it hampers the testing, debugging and repairing of the chip. As the result, the traditional optimization criterion steps aside and is presently replaced with criteria of structural simplicity and testability of a scheme, which are more important for modern VLSI schemes.

The above considerations are fully applicable to quantum cellular automata (QCA). Moreover, designing QCA schemes satisfying the above requirements will be hampered by the fact that the basic element of the QCA is a three-input majority element. Most of the methods and software tools, which are presently used for designing digital devices, deal with traditional additive forms of Boolean functions: SOP and ESOP. Therefore, the methods of designing QCA, proposed in some papers [1- 4], are based on transformation of such forms into QCA oriented forms. The aim of the transformations is presenting a traditional scheme as a composition of predetermined logic blocks followed by their replacement with specific scheme comprising majority and inventor elements. The structure of a scheme synthesized in this manner is hardly connected with a functional description of that scheme, which hampers its verification and testing. Such a scheme is problematic for modifying it when debugging, it is also hard for reconfiguration and repairing.

In order to overcome the above-mentioned drawbacks, and to take into account specific properties of the hardware at the early stage of functional description of the device to be designed, we propose a

principally novel approach in the present paper. We develop here an approach that is based on representation of the scheme to be synthesized in a form of a superposition of specific threshold functions. The specific of such a threshold function is in that weights of its inputs are different while all being an integer power of 2. In order to distinguish such threshold functions we call them *boundary functions*. Obviously, any Boolean function can be presented as a superposition of boundary functions. Any superposition of boundary functions has a canonical implementation as a majority scheme with a minimal number of invertors. Such an implementation possesses homogeneity, can be easily tested and allows quite simple replacement of a defective element with a redundant one.

The material is presented in the following order. Section 2 comprises a description of basic quantum elements and an overview of known methods for designing QCA. Section 3 describes boundary functions and some of their properties. Section 4 relates to synthesis of quantum comparators. Implementation of QCA in the form of a Comparator based Programmable Arrays (CPA) is presented in Section 5. In the Conclusion, we discuss advantages and drawbacks of the proposed method of design, as well as ways for optimizing the schemes to be synthesized.

2. Logic Synthesis by QCA

QCA cell contains four quantum dots and two mobile electrons. Due to Coulombic interactions, the electron pair assumes one of the two configurations. These configurations are considered as digital states. A majority gate is a primitive gate in QCA that implements the function $maj(a,b,c) = ab \vee ac \vee bc$. The fundamental QCA logic primitives also include a QCA wire and QCA inverter. This fact initiated a number of studies aimed to find an effective method for synthesis of QCA based logic structures. Since the problem of the optimal majority based syntheses in NP-complete [5] a number of heuristic approaches were developed [1- 4].

The majority gate is a particular case of threshold gates. A methodology for synthesis QCA schemes by threshold logic functions was studied in [6]. The Authors proposed a comprehensive threshold network synthesis methodology.

Another important direction in designing the QCA circuits is using regular homogeneous structures. A number of studies were done in using PLA-like structures as a basis for the QCA design [7, 8]. Owing to its highly regular structure, such a PLA structure can be easily fabricated using a bottom-up self-assembly process. The implementation of the nano PLA can be supported by multiple nanoelectronic devices under a crossbar based architecture.

In the present paper, we combine together a threshold function based synthesis and the homogeneous array structure. We restrict the basic set of threshold functions by a specific class having weights equal to integer powers of 2 and construct a universal gate comprising such functions, called a boundary comparator. This comparator being arranged into array structure and called Comparator based Programmable Array (CPA) forms a homogeneous regular structure that can be programmed for implementation of any desired logic function.

3. Boundary Functions

Definition. Let a ***boundary function*** be such a threshold function, which has weights v_i of its arguments equal to an integer power of 2:

$$v_i = 2^i, i = 0, \ldots, n-1,$$

where n - the number of arguments of the function.

Let us use a symbol Γ to indicate a boundary function:

$$y = \Gamma\left(x_{n-1}, \ldots, x_0; a\right), \tag{1}$$

where $x_{n-1}, \ldots, x_0$ are the arguments of the function, and a is the ***bound***.

Let us assume that the order of arguments in the equation (1) defines their weight, that is:

$$y = \begin{cases} 1 & if \quad \sum_{i=0}^{n-1} x_i 2^i \geq a \\ 0 & if \quad \sum_{i=0}^{n-1} x_i 2^i < a \end{cases}.$$

If the arguments and their weights are known from a context, the list of arguments in the equation (1) can be dropped and the function can be written down in its shortened form: $y = \Gamma(a)$.

By using the weights, each input vector is associated with a number. Clearly, different input vectors have different numbers. Thus, the n-dimensional cube is projected onto the closed interval $\left[0, 2^n - 1\right]$.

On this interval, the bound a separates the set of input vectors where the function equals 1 from the set of vectors where the function equals 0, as shown on Fig. 1.

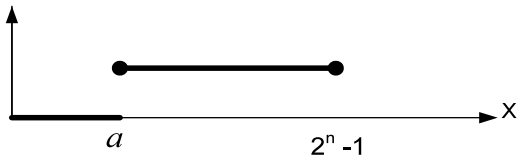


Fig 1. Graphical representation of the boundary function

Any logic function may be defined by a set $A = \{a_1, \ldots, a_k\}$, of bounds or points on the X axis, where, $0 \leq a_1 < a_2 < a_k \leq 2^n - 1$. For example, let $y_1(x_3, x_2, x_1, x_0) = 1$ for the input vectors associated with the integer values {3,4,5,6,10,11,12,13}. It is clear that the value of the function is changed in the following three points: $a_1 = 3, a_2 = 7, a_3 = 10, a_4 = 14$.

Let a function defined by a set of bounds $A = \{a_1, \ldots, a_k\}$ be denoted as $y = F\left(x_{n-1}, \ldots, x_0; \{a_1, \ldots, a_k\}\right)$ or $y = F\left(x_{n-1}, \ldots, x_0; A\right)$, or if the weights of variables are known: $y = F(A)$. For example, $y_1 = F\left(\{3, 7, 10, 14\}\right)$.

The following equality holds,

$$y = F\left(\{a_1, \ldots, a_k\}\right) = \Gamma(a_1)\overline{\Gamma(a_2) \ldots \Gamma(a_k)} = \Gamma(a_1) \oplus \cdots \oplus \Gamma(a_k) = \Gamma(a_1)\overline{\Gamma(a_2)} + \cdots + \Gamma(a_{k-1})\overline{\Gamma(a_k)}. \quad (2)$$

This equality allows expressing the value of an arbitrary function that is defined by its set of bounds. For example:

$$y_1 = \Gamma(3)\overline{\Gamma(7)}\,\overline{\Gamma(10)}\,\overline{\Gamma(14)} = \Gamma(3) \oplus \Gamma(7) \oplus \Gamma(10) \oplus \Gamma(14) = \Gamma(3)\overline{\Gamma(7)} + \Gamma(10)\overline{\Gamma(14)}. \quad (3)$$

Both the logical product and the logical sum functions can be expressed as a single boundary function each, as follows:

$$x_{n-1} \,\&\, \cdots \,\&\, x_0 = \Gamma\left(2^n - 1\right), \; x_{n-1} + \cdots + x_0 = \Gamma(1). \quad (4)$$

Note, the SOP and the POS canonical representations of Boolean functions can be considered as particular case of the boundary based representations.

Taking into account (4), formula (2) can be transformed to:

$$F\left(\left\{a_1,\ldots,a_k\right\}\right)=\Gamma\left(\Gamma\left(\Gamma\left(a_1\right),\overline{\Gamma\left(a_2\right)};3\right),\ldots,\Gamma\left(\Gamma\left(a_{k-1}\right),\overline{\Gamma\left(a_k\right)};3\right);1\right). \quad (5)$$

In Section 5, we show that if each of arguments of the function is represented in dual-rail form (both direct and negated values are available) the boundary functions form the universal or functionally complete basis in the Boolean algebra.

4. Quantum comparator

Let us represent a boundary function $y=\Gamma\left(X;a\right)$, where $X=\left\{x_{n-1},\ldots,x_0\right\}$ in a form of superposition of three-input majority functions. In this section we address the bound a by using its binary representation, $a=(\alpha_{n-1}\ldots\alpha_0),\alpha_i\in\left\{0,1\right\}$. We use a simple disjoint decomposition [9] for this purpose. Notice, that any boundary function has the simple disjoint decomposition.

Let us present a bound a as a concatenation of two strings: $a=(u.v)$. This presentation of a divides the ordered set X of arguments into the two disjoint subsets X_u and X_v, $X_u\cup X_v=X, X_u\cap X_v=\varnothing$. For example, let $X=\left\{x_4,x_3,x_2,x_1,x_0\right\}$ and $a=01101$. Define a as a concatenation of the numbers $u=01$ and $v=101$: $a=01.101$. Consequently: $X_u=\left\{x_4,x_3\right\}$ and $X_v=\left\{x_2,x_1,x_0\right\}$.

Theorem 1. Any boundary function has a simple disjoint decomposition:

$$\Gamma\left(X_u,X_v;u.v\right)=\Gamma\left(X_u,\Gamma\left(X_v;v\right);u.1\right) \quad (6)$$

(Due to the limitations on the paper the size all theorems are presented without proofs.)

Example: $\Gamma\left(\left\{x_4,x_3\right\},\left\{x_2,x_1,x_0\right\};01.101\right)=\Gamma\left(x_4,x_3,\Gamma\left(x_2,x_1,x_0;101\right);011\right)$.

Appling (6) for the particular example, when $X_u=\left\{x_{n-1}\right\}, X_v=\left\{x_{n-2},\ldots,x_0\right\}$ and $u=\alpha_{n-1}, v=\alpha_{n-2}\ldots\alpha_0$ we have:

Consequence 1.1.

$$\Gamma\left(\left\{x_{n-1}\right\},\left\{x_{n-2},\ldots,x_0\right\};\alpha_{n-1}.\alpha_{n-2}\ldots\alpha_0\right)= maj\left(\Gamma\left(x_{n-2},\ldots,x_0;\alpha_{n-2}\ldots\alpha_0\right),x_{n-1},\bar{\alpha}_{n-1}\right) \quad (7)$$

Example: $\Gamma\left(0.1101\right)=maj\left(\Gamma\left(1101\right),x_4,1\right)=x_4+\Gamma\left(1101\right)$

Applying the formula (7) recursively we have:

Consequence 1.2.

If $\alpha_0=1$, then:

$$\Gamma\left(X;a\right)=maj\left(\bar{\alpha}_{n-1},x_{n-1},maj\left(\bar{\alpha}_{n-2},x_{n-2},maj\left(\ldots maj\left(\bar{\alpha}_1,x_1,x_0\right)\right)\right)\right). \quad (8)$$

Otherwise, the boundary function does not depend on x_0.

Example:

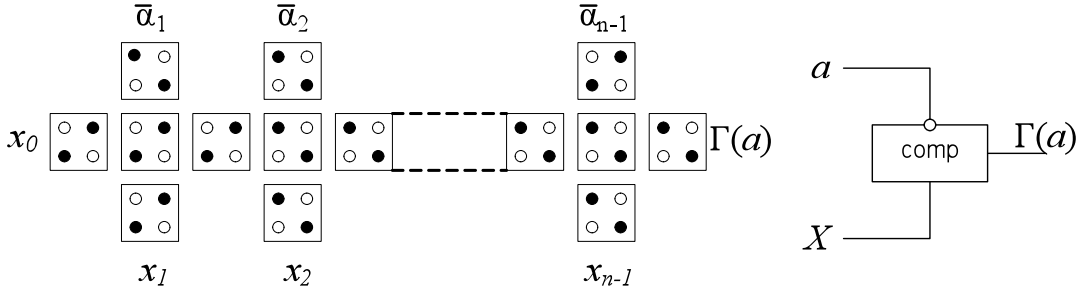
$$\Gamma\left(01101\right)=maj\left(1,x_4,maj\left(0,x_3,maj\left(0,x_2,maj\left(1,x_1,x_0\right)\right)\right)\right)=x_4+x_3x_2\left(x_1+x_0\right)$$

The scheme of the comparator is built according to (8), and shown in Fig 2a. A similar solution was proposed in [10]. However, in the present paper the solution is obtained analytically on the base of properties of the boundary functions.

Our scheme of the comparator can be considered as a cascade of universal programmable blocks, known as Maitra cascade [11]. According to that, one of inputs of the majority element of our scheme (α-input) has to be considered a control input, while two other inputs are information inputs. The

universal programmable block is programmed by its control inputs for implementation of one of two possible functions: the AND $(\alpha = 0)$and the OR $(\alpha = 1)$. The Maitra cascades are well studied. At the same time, in most of the studies the universal programmable block of the cascade has at least two control inputs. Furthermore in such studies, the universal programmable block can be programmed for implementation of non-monotonic functions, which are inappropriate for QCA-based logic design.

We will use the presented comparator as a basic element in QCA schemes. The graphical notation of the element is presented in Fig. 2b.

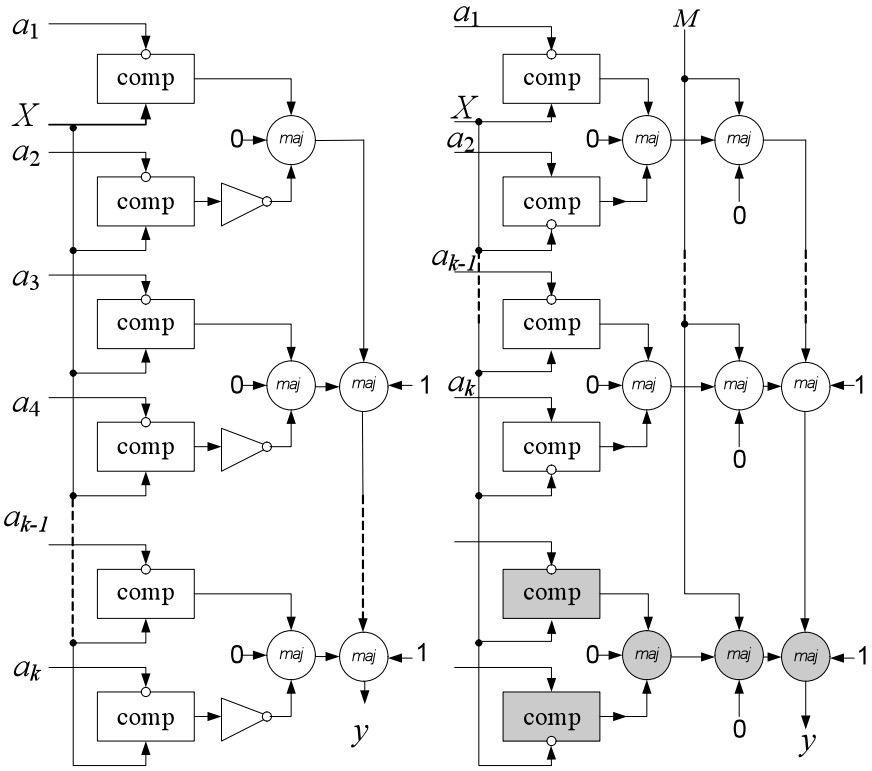


a) *b*)

Fig. 2. QCA comparator; (*a*) scheme; (*b*) graphical notation

5. Implementation of QCA by Comparator based Programmable Arrays

We propose to arrange the set of our comparator blocks into a homogeneous structure – Comparator based Programmable Array (CPA) shown in Fig. 3a.



a) *b*)

Fig. 3. Comparator based QCA array: *a*) one-rail input variables; *b*) dual-rail input variables

We use the CPA as the base of the QCA synthesis. The CPA can be programmed for implementation of a certain function defined by its bounds $a_1, \ldots, a_k$. For this aim, inverse values of binary codes of bounds are stored on the comparator's inputs. The scheme is constructed according to (5). The set of elements and type of their connection is universal and independent of the function to be implemented.

Moreover, the functional description of the scheme is directly connected with its structure. Notice, the scheme has the universal set of $2k$ test input vectors. The scheme can be efficiently checked concurrently since all boundary functions are monotonic; for their values always correct: $\Gamma(a_i) > \Gamma(a_{i+1})$. The scheme also allows a simple repair procedure. Some of faulty comparators can be replaced with fault-free reserve comparators. Finally, the scheme does not contain negations of input variables. If this limitation is relaxed, i. e., it is allowed to use both direct and inverse values of input variables then the QCA scheme can be inverter free according to the following theorem.

Theorem 2.

$$\Gamma(x_{n-1}, \ldots, x_0; a) = \overline{\Gamma(\bar{x}_{n-1}, \ldots, \bar{x}_0; \hat{a})}, \text{ where } \hat{a} = 2^n - a. \quad (9)$$

Since the least significant bit of a equals 1, then, to obtain $\hat{a}$ all other bits have to be negated. Thus, it is possible to implement $\overline{\Gamma(x_{n-1}, \ldots, x_0; a)}$ by using the negated values of input variables and the bounds $\hat{a}$. The corresponding circuit is presented in Fig. 3.b. This scheme contains circuitry enabling replacing faulty couples of boundary elements with a reserve couple. The reserve couple is underlined by grey color. The replacement can be performed by using a mask M: each pair of boundary elements corresponds a bit of the mask. If this bit is equal to 1, then a pair of boundary elements is connected to the OR element forming the output vector; if this bit is equal to 0, then the pair is disconnected.

Notice, that the number of comparators and, consequently, the QCA complexity depends of weights of functions' arguments or (which is the same) it depends on the order of arguments. For example, if we change the order of arguments to x_2, x_1, x_0, x_3 in above discussed function $y_1(x_3, x_2, x_1, x_0)$ only two (instead of four) boundary points are required to specify the function, that is, $y_1(x_2, x_1, x_0, x_3) = \Gamma(5)\overline{\Gamma(13)}$. The optimal ordering of variables can be achieved by using correlation analysis [12] or other optimization methods.

6. Conclusions

We presented a novel universal quantum cellular automata gate - boundary comparator. We introduced a Comparator based Programmable Array (CPA) that is based on this boundary comparator. The CPA is a homogeneous regular structure that can be programmed for implementation of any desired logic function.

We have developed theoretical fundamentals of the boundary functions used for representation of logic functions by the boundary comparator. The boundary functions forms a functionally complete system in the Boolean algebra. Each of logic function can be implemented by the boundary functions and, consequently, by the boundary comparator. We demonstrated implementation of the QCA schemes by the proposed CPA structures.

The main advantage of the proposed solution is its regularity, which, in turn, provides the testability, a potential reconfigureability, reparability etc. The array homogeneous structure is also desirable form manufacturing point of view.

We did not discuss methods for the optimal synthesis of CPA based QCA schemes. Obviously, this issue is an important direction of the future study. Nevertheless, we believe that even without optimization the proposed CPA structure will be useful in designing QCA logic circuits.

References

[1] David Y. Feinstein and Mitchell A. Thornton.: ESOP Transformation to Majority Gates for Quantum-dot Cellular Automata Logic Synthesis, Proceedings of the Workshop on Applications of the

Reed-Muller Expansion in Circuit Design and Representations and Methodology of Future Computing Technology (RMW), May 16, 2007, pp. 43-50.

[2] Z. Huo, Q. Zhang, S. Haruehanroengra and W. Wang.: Logic optimization for majority gate-based nanoelectronic circuits", Proceedings of the IEEE International Symposium on Circuits and Systems. 2006, pp. 1307-1310.

[3] Rumi Zhang, Konrad Walus, Wei Wang, and Graham A. Jullien: A Method of Majority Logic Reduction for Quantum Cellular Automata, IEEE Transaction on Nanotechnology, vol. 3, No. 4, December, 2004, pp. 443- 450.

[4] Mariam Momenzadeh, Jing Huang, Mehdi B. Tahoori, Fabrizio Lombardi: Characterization, Test, and Logic Synthesis of And-Or-Inverter (AOI) Gate Design for QCA Implementation, IEEE Transaction on Computer-Aided Design of Integrated Circuits and System, vol. 24, No. 12, December 2005, pp. 1881- 1893.

[5] V. Varshavsky: Logic Design and Quantum Challenge, Workshop on Physics and Computer Modeling of Devices Based on Low-dimensional Structures (PHYSICS'95), November 1995, pp. 134.

[6] Rui Zhang, P. Gupta, Zhong Lin, N. K. Jha: Threshold network synthesis and optimization and its application to nanotechnologies, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 1, Jan. 2005, pp. 107-118.

[7] Xiaobo Sharon Hu Michael Crocker Michael Niemier: Minjun Yan Gary Bernstein. PLAs in Quantum-dot Cellular Automata, Proceedings of the 2006 Emerging VLSI Technologies and Architectures (ISVLSI'06). March 2006, 6 pp.

[8] Wenjing Rao, Alex Orailoglu, Ramesh Karri: Fault Tolerant Approaches to Nanoelectronic Programmable Logic Arrays, 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007. DSN '07, pp. 216 –224.

[9] R. L. Ashenhurst. The decomposition of switching functions, Proc. of an Intl. Symposium on Theory of Switching, April 2-5, 1957, Ann. Computation Lab. Harvard University, 1959, vol.29, pp. 74-116,.

[10] Heinz-Juergen Lohmann: Comparator circuit for two N-digit binary codes, United States Patent 4012714. Mar 15, 1977.

[11] G. Fantauzzi: NORNAND Maitra Cascades, IEEE Transactions on Computers Nov. 1968 Volume: C-17, Issue: 11, 1074 – 1080.

[12] M.G. Karpovsky: Finite Orthogonal Series in the Design of Digital Devices, New York, Wiley, 1976.