Programmable Comparators Based Array for Regular QCA Implementation

Vladimir Ostrovsky,
Tel-Aviv University,
vladio@post.tau.ac.il

Osnat Keren,
Bar-Ilan University,
kereno@eng.biu.ac.il

Ilya Levin,
Tel-Aviv University,
ilia1@post.tau.ac.il

Abstract

The paper presents a novel universal Quantum Cellular Automata (QCA) gate called boundary comparator. This gate implements a Boolean function in its boundary form, which is a superposition of elementary boundary functions i.e. a threshold function having weights equal to integer powers of 2. The boundary comparators are arranged in a form of array forming homogeneous programmable structure. The paper proposes a method of synthesis of Boolean functions on the base of boundary functions. The method uses autocorrelation values of the initial function for minimization of a number of bounds. The structures of the boundary comparator as well as the structure of the comparator-based array are presented. Benchmark results allow evaluating efficiency of the proposed structure in comparison with known QCA solution.

1 Introduction

Quantum Cellular Automata (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. 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 [9] a number of heuristic approaches were developed [1, 3].

In all of the above-mentioned works, quality of the solutions was evaluated by the number of majority elements in the schemes to be synthesized. This criterion approximately corresponds to the area overhead. It should be noted that that reduction of the area overhead is not the only criterion and in many cases is not even the main criterion imposed to the scheme. There are other, not less important criteria, for example: a) reparability i.e., a possibility to reconfigure the scheme in order to replace faulty elements, b) simplicity of verification, testing and on-line checking, and c) "understandability" of the scheme and simplicity of designing it. The latter criterion supposes that there is a correspondence between functional description of the device and structural elements of the scheme. The importance of these criteria grows with increase of the density of the chip and the complexity of the schemes implemented by each specific chip. When evaluating quantum-computing devices, these criteria play an important role. Programmable logical arrays (PLAs) satisfy these criteria almost completely. Solutions [2, 10] describe quantum PLA-like homogeneous structures comprising QCAs. However, while quantum PLAs satisfy the criteria of reparability and simplicity of designing, they are rather difficult for verification and checking. In order to satisfy the above criteria, the present paper proposes a novel approach to the QCA design. According to the proposed approach, the scheme is presented as a superposition of threshold functions having weights that are integer powers of 2. It is known that any Boolean function can be represented as a superposition of boundary functions. Any superposition of boundary functions is non-ambiguously implementable by a majority scheme having minimal number of inverters. Such a scheme is homogeneous, it can be easily checked and it is reparable.

Material in the paper is presented in the following order. Fundamental of the proposed approach are given in Section 2. Section 3 describes comparator based array architecture. Evaluation of the implementation cost of the proposed solution and its optimization are presented in Section 4. Experimental results are discussed in Section 5. Conclusions are given in Section 6.

2 Preliminaries

Boundary functions form a class of threshold functions for which the weight v_i associated with the input variable x_i, equals to 2^i. Denote by $\Gamma(x_{n-1}, \ldots, x_0; a)$ a boundary function with bound (threshold) value a, that is,

$$\Gamma(x_{n-1}, \ldots, x_0; a) = \begin{cases} 1 & \sum_{i=0}^{n-1} x_i 2^i \geq a \\ 0 & \sum_{i=0}^{n-1} x_i 2^i < a \end{cases} .$$

978-0-7695-3379-7/08 $25.00 © 2008 IEEE
DOI 10.1109/NDCS.2008.14

IEEE computer society

If the arguments and their corresponding weights are known from the context, then the boundary function can be denoted as $\Gamma(a)$.

Let $M(a,b,c)$ denote a majority operator $M(a,b,c) = ab \vee ac \vee bc$. A boundary function can be expressed in terms by majority operators as follows:

Theorem 1 *Let $a = (\alpha_{n-1}, \ldots, \alpha_0)$ be the binary vector representing the value of a in base 2. Then,*

$$\Gamma(a) = M(x_{n-1}, \bar{\alpha}_{n-1}, M(x_{n-2}, \bar{\alpha}_{n-2}, M(\cdots M(x_0, \bar{\alpha}_0, 1))). \quad (1)$$

Example 1 *The boundary function $\Gamma(x_4, x_3, x_2, x_1, x_0, a)$ where $a = (01101) = 13$, can be represented as*

$$\begin{aligned} y &= M(x_4, 1, M(x_3, 0, M(x_2, 0, M(x_1, 1, M(x_0, 0, 1))))) \\ &= x_4 + (x_3(x_2(x_1 + x_0))) = x_4 + x_3x_2(x_1 + x_0). \end{aligned}$$

Note that the inner majority operation in Eq. 1 is $M(x_0, \bar{\alpha}_0, 1)$. If α_0 equals zero, then $\Gamma(a)$ does not depend on the variable x_0. In general,

Lemma 1 *Let $z(a)$ be the largest integer for which $a = (\alpha_{n-1}, \ldots \alpha_0)$ satisfies*

$$\alpha_i = \begin{cases} 0 & i < z(a) \\ 1 & i = z(a) \end{cases} .$$

The number of majority blocks requited for the implementation of $\Gamma(a)$ is $n - z(a) - 1$.

Any logic function $y = f(X), X = (x_{n-1}, \ldots, x_0)$, can be defined by an ordered set $A = \{a_1, \ldots, a_k\}$ of bound-points, where, $0 \le a_1 < \cdots < a_k \le 2^n - 1$. That is,

$$y = F(A) = \Gamma(a_1) \oplus \Gamma(a_2) \oplus \cdots \oplus \Gamma(a_k)$$

If k is even, then the function can be represented as

$$y = F(A) = \Gamma(a_1)\bar{\Gamma}(a_2) \vee \cdots \vee \Gamma(a_{k-1})\bar{\Gamma}(a_k), \quad (2)$$

where $\bar{\Gamma}(a)$ is the complement of $\Gamma(a)$. If k is odd, then,

$$y = F(A) = \Gamma(a_1)\bar{\Gamma}(a_2) \vee \cdots \vee \Gamma(a_{k-2})\bar{\Gamma}(a_{k-1}) \vee \Gamma(a_k).$$

Example 2 *Consider a four input function $f(x_3, x_2, x_1, x_0)$ that is specified by the truth vector*

$$f = (f(0), f(1), \ldots, f(15)) = (0101011110011100).$$

The set A is $\{1, 2, 3, 4, 5, 9, 11, 14\}$, thus,

$$Y = F(A) = \Gamma(1)\bar{\Gamma}(2) \vee \Gamma(3)\bar{\Gamma}(4) \vee \Gamma(5)\bar{\Gamma}(9) \vee \Gamma(11)\bar{\Gamma}(14).$$

3 Comparator based Programmable Array

In this section, we construct a universal gate comprising such functions, called a boundary comparator. We arrange comparators in an array structure, called Comparator based Programmable Array (CPA), which forms a homogeneous regular structure that can be programmed for implementation of any desired logic function.

The scheme of the comparator built according to Theorem 1, is shown in Fig. 1-top. A similar solution was proposed in [7]. We will use the presented comparator as a basic element in QCA schemes. The graphical notation of the element is presented in Fig. 1-bottom. If the inputs enter the system in a dual rail manner, that is, the inputs and their negated values are available, then the comparator is a universal gate; Any product (or a sum) of w literals can be realized by a comparator of size $w - 1$ with bound value $a = 2^w - 1$ (or $a = 0$).

We propose to arrange the set of comparator blocks into a homogeneous structure as shown in Fig. 2. The Comparator based Programmable Array (CPA) has a comparator array followed by an AND-OR strip. 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 set of bounds A. For this aim, inverse values of binary codes of bounds-points are stored on the comparator's inputs. The scheme is constructed according to Eq. 2, which refers to even number of bound-points (k is even). The set of elements and the type of their connection is universal and independent of the function to be implemented. In case k is odd, the bottom majority element in the AND part of the AND-OR strip is fed with "1" and "0", and thus, it functions as a wire.

The functional description of the scheme is directly connected with its structure. Therefore, the scheme has a universal set of $2k$ test vectors. The scheme can be efficiently checked concurrently since all boundary functions are monotonic, and their values satisfy $\Gamma(a_i) > \Gamma(a_{i+1})$. 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 than the QCA scheme can be inverter free.

4 Implementation cost and Optimization

The implementation cost of the suggested scheme depends on the size k of the set A. The AND-OR strip requires $(\lfloor k/2 \rfloor + \lceil k/2 \rceil - 1) = k - 1$ majority elements. The implementation cost of a function by a CPA is $kn + (k-1) = k(n+1) - 1$ majority elements. The implementation cost

of a function as a combinatorial circuit is

$$N = (k(n-1) - z(A)) + (k-1) = kn - z(A) - 1. \quad (3)$$

where $z(A) = \sum_{i=1}^{k} z(a_k)$.

Notice that the number of bound-points (and thus the implementation cost) is sensitive to the ordering of the input variables. For example,

Example 3 *Consider the logic function of Ex. 2. The number of bound-points for the natural ordering* (x_3, x_2, x_1, x_0) *is* $k = 8$ *and* $z(A) = 4$. *Thus, the implementation cost for the natural ordering is* $N = 4 * 8 - 4 - 1 = 26$ *majority blocks. However, the truth vector defined by the ordering* $\pi = (x_3 x_0 x_1 x_2)$, *is,* $f_\pi = (0001111111000110)$. *The reordered function* f_π *has only four bound-points,* $A_\pi = \{3, 10, 13, 15\}$ *and* $z(A_\pi) = 1$. *Therefore, it has* $N_\pi = 4 * 4 - 1 - 1 = 14$.

In general, the number of bound-points in f and in $\bar{f}$ differs by one. If $f(0) = 0$, then $\bar{f}$ has $k+1$ bound-points, and, if $f(0) = 1$, then $\bar{f}$ has $k-1$ bound-points. To simplify the discussion, we denote as $\hat{k}$ the minimum between the number of bound-points of f and of $\bar{f}$.

Denote by $\hat{k}_0$ and $\hat{k}_1$ the number of bound-points in the functions $f(0, x_{n-2}, \ldots x_0)$ and $f(1, x_{n-2}, \ldots x_0)$, respectively. Then, $\hat{k} = \hat{k}_0 + \hat{k}_1 + \delta_{0,1}$, where $\delta_{0,1} = 1$ if $f(0,1,1,\ldots 1) \neq f(1,0,0,\ldots 0)$, and it equals zero otherwise.

Let the variables X_{n-i} stand for the $n-i$ most significant variables, $X_{n-i} = (x_{n-1}, \ldots, x_i)$, and denote by $U = (u_{n-i-1}, \ldots u_0) \in \{0,1\}^{n-i}$ an assignment of X_{n-i}. The function $g_u(x_{i-1}, \ldots x_0) = f(U, x_{i-1}, \ldots x_0)$ is a logic function of i variables. Define,

$$f_i(X_{n-i}) = \begin{cases} g_u(0) & X_{n-i} = U \text{ and } u_0 = 1 \\ g_u(2^i - 1) & X_{n-i} = U \text{ and } u_0 = 0 \end{cases}.$$

Then,

Theorem 2 *The number of bound-points* $\hat{k}$ *for the function* $f(x_{n-1}, \ldots, x_0)$ *with the natural ordering equals* $\hat{k} = \sum_i R_i(0) - R_i(1)$, *where* $R_i(\tau)$ *is the autocorrelation function of* $f_i(Z)$, *and,*

$$R_i(0) = \sum_Z f_i(Z),\ R_i(1) = \sum_Z f_i(Z) f_i(Z \oplus (0 \cdots 01)). \quad (4)$$

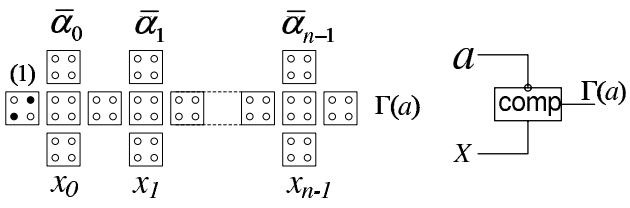

Figure 1. Comparator: representation in cells (top) and logic block (bottom).

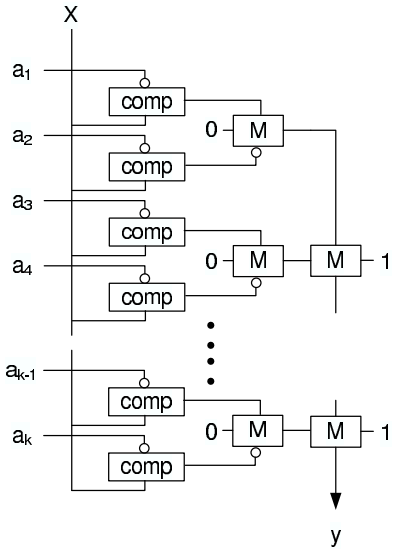

Figure 2. Comparator array structure

A procedure, similar to the one presented in [5], may be used to optimize $\hat{k}$. Note, that Eq. 4 follows the formal definition of the autocorrelation function [4]. The computation of these values following their definition is of complexity of order 2^n. However, there exist efficient methods that calculate these values with relatively small complexity, refer to [4, 6]. In particular, it is possible to compute R for a function represented as a Binary Decision Diagram (BDD) with complexity that is proportional to the BDD size [8].

5 Experimental results and discussion

In this section, we discuss the implementation cost of the suggested solution in two ways; First we examine the efficiency of the suggested CPA as a regular array of $\hat{k}$ comparators, and compare it to the cost of a quantum PLA [2]. Second, we refer to the suggested solution as a combinatorial circuit and compare it to the SOP and ESOP implementations presented in [1]; the comparison is done in terms of the number of majority elements.

The regular arrays (PLA and CPA) are compared in terms of the number of product terms P_{SOP} in the minimal SOP expression and the number bound-points $\hat{k}$ in the reordered function, respectively. Note that the proposed structure *inherently* has a fault detection property, while the minimal SOP representation may not have this property since two rows in the PLA's OR plane may be activated simultaneously. Thus, P_{sop} may be referred to as a lower bound on the size of a quantum PLA having fault detection property.

Tables 1 and 2 present the implementation cost of the benchmark functions $5xp1$ and $dc2$, respectively. We use the following notation:

- i is the number corresponding to a specific output
- $\hat{k}$ is the number of bound-points (BPs) for the original function using the natural ordering.
- $\hat{k}_\pi$ is the number of BPs for the ordered function.
- P_{sop} is the number of non-disjoint product terms in the minimal SOP representation [1].
- N is the number of majority elements in the implementation of the function with the natural ordering (Eq. 3).
- N_π is the number of majority elements in the implementation of the reordered function.
- N_{sop} is the number of majority elements in the implementation of the function in its minimal SOP form [1].
- N_{esop} is the number of majority elements in the implementation of the function in its minimal ESOP form [1].

It is clear from the Tables that ordering may improve the number of bound-points. It is assumed that allowing use of linear combinations of the inputs as inputs to the comparators array may further reduce the number of bound-points. For some functions, the comparator based circuits suggest a better solutions (in terms of area and number of majority elements) than quantum PLA and/or combinatorial logic based on SOP or ESOP.

6 Conclusions

We presented a novel universal quantum cellular automata gate - boundary comparator. We introduced a comparator based programmable array that is based on this boundary comparator. Any logic function can be implemented by the boundary functions and, consequently, by the boundary comparator. We provided a method for optimized implementation of the QCA schemes by the proposed array structures. The optimization method is based on calculation of the autocorrelation values of Boolean functions to be implemented. The comparator arrays provide better solution in comparison with known approaches for considerable number of benchmarks. The main advantage of the proposed solution is its regularity, which, in turn, provides the testability, a potential reconfigureability and reparability.

Table 1. Benchmark 5xp1: 7 inputs, 10 outputs

i	0	1	2	3	4	5	6	7	8	9
$\hat{k}$	32	64	95	65	123	9	11	8	15	31
$\hat{k}_\pi$	2	4	9	9	9	5	3	2	1	1
P_{sop}	12	12	19	15	10	5	8	2	1	3
N	208	416	582	382	753	30	32	31	48	151
N_π	13	26	54	46	35	14	7	3	0	6
N_{sop}	50	54	86	62	38	15	34	3	0	10
N_{esop}	37	62	57	27	17	6	7	3	0	3

Table 2. Benchmark dc2 : 8 inputs, 7 outputs

i	0	1	2	3	4	5	6
$\hat{k}$	4	6	10	20	28	50	255
$\hat{k}_\pi$	4	6	10	18	20	14	1
P_{sop}	4	7	10	12	9	8	1
N	19	35	53	113	166	304	1792
N_π	19	34	53	104	104	73	0
N_{sop}	17	40	52	70	49	44	0
N_{esop}	22	49	57	84	31	31	0

References

[1] D. Y. Feinstein and M. A. Thornton, "ESOP Transformation to Majority Gates for Quantum-dot Cellular Automata Logic Synthesis," *Proc. of the Reed-Muller Workshop (RMW)*, pp. 43-50, May 2007.

[2] Xiaobo Sharon Hu, Michael Crocker, Michael Niemier, Minjun Yan, Gary Bernstein, "PLAs in Quantum-dot Cellular Automata," *Proc. of the 2006 Emerging VLSI Technologies and Architectures* , pp. 242 -250, March 2006

[3] Zhi Huo, Qishan Zhang, Haruehanroengra S. and Wei Wang, "Logic optimization for majority gate-based nanoelectronic circuits," *Proc. of the IEEE Int. Symposium on Circuits and Systems*, pp. 1307-1310, 2006.

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

[5] O. Keren, "Reduction of average path length in binary decision diagrams by spectral methods," *IEEE Transactions on Computers*, vol. 57, no. 4, pp. 520-531 ,2008.

[6] O. Keren, I. Levin and R.S. Stankovic, "Linearization of Functions Represented as a Set of Disjoint Cubes at the Autocorrelation Domain," *Proc. of the 7th Int. Workshop on Boolean Problems*, pp. 137-144, 2006.

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

[8] J. E. Rice and J. C. Muzio, "Methods for Calculating Autocorrelation Coefficients," *Proc. of the 4th Int. Workshop on Boolean Problems* , pp. 69-76, 2000.

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

[10] Rui Zhang, P. Gupta, Lin Zhong, N.K. Jha, "Threshold network synthesis and optimization and its application to nanotechnologies," *IEEE Trans. on CAD*, vol. 24, no. 1, pp. 107-118, Jan. 2005.