Swarm and Evolutionary Computation 10 (2013) 25–30

Contents lists available at SciVerse ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular Paper

Multi-output majority gate-based design optimization by using evolutionary algorithm

Mohammad A. Tehrani [a], Keivan Navi [a,b,*], Ali Kia-kojoori [a]

[a] Nanotechnology and Quantum Computing Laboratory, G.C., Shahid Beheshti University, Tehran, Iran
[b] Electrical and Computer Engineering, University of California at Irvine, Irvine, CA 92714, USA

ARTICLE INFO

Article history:
Received 29 February 2012
Received in revised form
8 November 2012
Accepted 12 December 2012
Available online 20 December 2012

Keywords:
Quantum-dot cellular automata
Majority gate
Logic optimization
Evolutionary algorithm

ABSTRACT

In this paper, a novel efficient method for optimizing multi-output majority gate based designs is proposed. Majority gate is a fundamental Boolean operator in some nano-scale technologies such as quantum-dot cellular automata (QCA). As a result, the design optimization must be directly implemented on majority gates instead of optimizing the design for AND–OR gates. In some other nanotechnologies, a fundamental element is Minority gate which could be simply converted to majority gate by the De Morgan's theorem. Here, the proposed optimization method works on the basis of evolutionary computation and can reduce both the number of majority gates and the worst-case delay of the circuit. The method is compared to some other optimization algorithms and its efficiency is verified.

1. Introduction

Complementary metal-oxide semiconductor transistor (CMOS) technology has faced many serious challenges in recent decades such that a proper substitute technology seems quite necessary. Many efforts have been made on several areas of nanoelectronics because they have been considered as superior candidates for replacing the CMOS technology [1]. Quantum-dot cellular automata (QCA) is a promising nanotechnology whose power consumption is considerably lower than CMOS and could work significantly faster [2,3].

The basic device element of QCA is a cell consisting of four quantum-dots at the corners. There are two surplus electrons trapped in a cell which can tunnel freely within the cell between quantum-dots. Because of the high barriers at the border of each cell, electron tunneling outside the cell is blockaded. Electrons can take several positions in the cell. However, they are stable enough for being assumed as the steady state, only when they are on the antipodal sites. As shown in Fig. 1, one state denotes logic 0 and the other logic 1 [3].

In this technology, the basic logic elements are majority and inverter gates. Fig. 2 depicted a typical majority gate and several implementations of inverter [3–5].

In QCA there are four cyclic clock phases which divides the circuit into several segments. Inputs usually enter the circuit in the first phase and propagate step by step through the rest of the circuit in each clock phase. Eventually, fourth phase of each clock delivers the signal to the first phase of the next clock cycle. Fig. 3 illustrates the signal propagation from input to the output in four phases (one clock cycle) [6]. Accordingly, it is necessary to use several clock phases between inputs and outputs. Otherwise, input signals are unable to propagate through the whole circuit and reach the outputs without amplification [7].

In a majority gate, when three amplified signals reach to a voter cell, their amplitude should be normalized to have a fair voting. Hence, the number of the cells between the voter and where the clock phases have changed must be equal. To simplify the design process, this distance usually assumed as one cell. It means that the clock phase should change one cell before every voter cell (Fig. 3) [4].

Majority gate based design had been introduced before the invention of BJT and MOSFET devices [8]. However, it has not become popular due to its hardware inefficiency in these technologies. Using BJT and CMOS widely for several decades, almost all optimization methods have been developed for "AND–OR" based designs [4,9]. As in the QCA technology, implementing the majority function is far easier it is worthwhile to propose effective optimization methods for majority gate-based design.

Like many other optimization problems, several parameters should be considered and some criteria must be met. So a heuristic method seems necessary to search the immense space of candidate solutions. Evolutionary algorithm (EA) is a heuristic method which imitates the behavior of natural evolution. One of the most prominent attributes of EA is the capability of performing "local

* Corresponding author at: Electrical and Computer Engineering, University of California at Irvine, Irvine, CA 92714, USA. Tel.: +1 847 444 9798.
E-mail addresses: m_tehrani@sbu.ac.ir (M.A. Tehrani), knavi@uci.edu, navi@sbu.ac.ir (K. Navi), a.kia@mail.sbu.ac.ir (A. Kia-kojoori).

2210-6502/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.swevo.2012.12.002

search" which leads to the answers closer to the global optimum [10,11].

The main concern of this article is to propose a novel method for reducing the number of majority gates in logical circuits based on an evolutionary algorithm which could be applied on both single-output and multi-output designs.

The remainder of the paper is organized as follows. In Section 2, the previous related works are studied. The proposed method is presented in Section 3. Section 4 includes the simulation results and comparison and finally Section 5 concludes the paper.

2. Previous works

Many efforts have been made on the circuit optimization, using evolutionary computing methods. However, most of them do not utilize majority gate as their basic element [12–14]. Some synthesis methods have also been introduced for optimizing the majority gate-based designs without using evolutionary algorithms [9,15,16]. The first attempt for optimizing the majority gate-based designs with the genetic algorithms was made in 2007 by Bonyadi et al. [17] on a single output circuit (SO_GA1). In this method, a tree structure is used whose internal nodes consist of majority functions and inverters, and external nodes (leaves) are logical "1" or input variables. A Depth-First Search (DFS) traverse of the tree constructs the chromosome vector. As shown in Fig. 4, an individual chromosome for each output must be created. The performance of this method has been enhanced in [18] by using Quine–McCluskey to initialize the population of GA. It also utilized a new approach based on hamming distance to find the constituent function set for the given logical function. Although the latter method works faster, the results remain as that of [17].

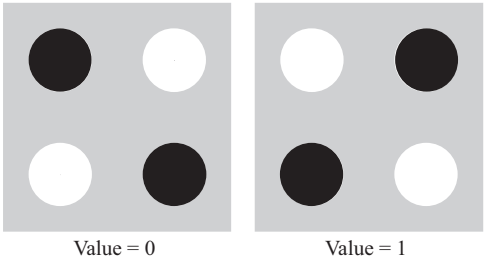

Fig. 1. Logical value representation with QCA cell.

Another design has also been introduced on the basis of SO_GA1 for multi-output circuits (MO_GA) [19]. It was first implemented for only two outputs in [19] and then extends for the arbitrary number of outputs in [20]. In this method, a function finds the common terms in two chromosomes and omits the repeated parts as shown in Fig. 5. First the algorithm iterates for each output to find the best fit chromosomes. Then, the chromosomes are combined together to find the best result.

Despite the fact that inverter gate could be implement with only two cells, both SO_GA1 and MO_GA assume inverter overly important and tried to reduce its number as well as the number of majority gates. On the other hand, the most important parameter that should be minimized in QCA circuits is the number of clock cycles between input and output through the worst-case path (on which the speed of process mainly depends). The number of inverters does not affect the speed of circuit because it is necessary to change the clock phase before every majority gate and the clock phase should not be changed for an inverter.

The next optimization method for majority gate-based designs based on SO_GA1 introduced in [21] (SO_GA2). This model provides different weights for majority gate and inverter in its fitness functions. Besides, it not only minimizes the number of cells but also tries to reduce the length of worst-case path by reducing the level of chromosome's tree.

3. The proposed method

All the mentioned methods optimize QCA circuits from their own approach, all of which are correct but do not cover all aspects of optimization. It is worthwhile to have a novel method to minimize the number of clock phase in multi-output circuit as well as the number of cells.

Moreover, as is shown in Fig. 2, there are some implementations of QCA inverter, whose cell number does not differ from a corresponding wire and a signal could be inverted through the interconnecting wires. Hence, the number of the inverters does not affect cell complexity and could be neglected.

Assume a tree with L levels where the root is first level and each node represents a majority gate. The tree has $3^{(L-1)}$ leaves to

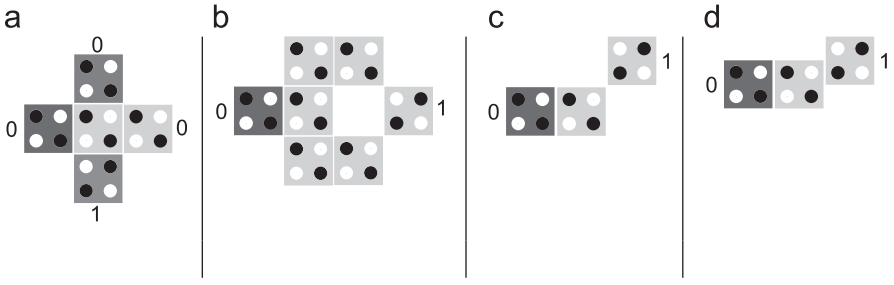

Fig. 2. (a) QCA majority voting gate. (b–d) Several implementations of inverter in QCA.

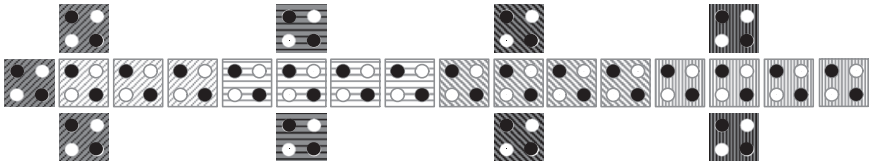

Fig. 3. Four cascaded majority gates. The clock phase must be changed on inputs of each majority gate. Several pattern represents the clock phases, and bold cells show the inputs.