Implementing the Toffoli gate in Quantum-dot Cellular Automata

I. seminar project

Biserka Cvetkovska,
Ivana Kostadinovska,
and Jirka Daněk

November 2013

This report presents results of our seminar work in course *Unconventional information processing methods and platforms* taught by prof. M. Mraz at University of Ljubljana in the winter semester of the academic year 2013/2014.

1 Introduction

The idea of this seminar project was to build the reversible Toffoli logic gate in Quantum-dot Cellular Automata (QCA). For that manner, we created a practical implementation of the gate in the QCADesigner and gave some examples that demonstrate how the gate can be used as a part of more complicated reversible circuits.

1.1 Quantum-dot Cellular Automata (QCA)

A quantum-dot cellular automata is a finite state machine consisting of a finite or infinite grid of quantum-dot cells. A quantum-dot cell is a set of four quantum dots located at the corners of the cell and an electron pair. By providing tunneling junctions with potential barriers, which are raised to prevent electron movement and lowered to permit

(a) Quantum-dot cell. (b) The two possible states of a QCA cell.

Figure 1: QCA cell.

electron movement, three states can occur. When barriers are low, the electrons can localize on any dot and the Null state occurs, but when the barrier is raised, the cell is polarized and the other two states can occur. These two states represent the logic "1" and "0". Because of Coulombic interactions, cells which are located near each other are forced into matching polarizations. The propagation of polarization provides information transfer.

1.1.1 Basic building structures

Basic structure in QCA technology is the wire and the majority gate or majority voter (MV). Other structures like AND and OR can be implemented using the MV by setting one of its inputs to a constant value.

1.1.2 Clock

The propagation of signals in all QCA circuits is driven by a codlock signal. Each QCA cell belongs into one of four clock zones. The clock signal periodically goes through four phases. There are two transitory phases that separate the two main phases – the hold phase when the cell holds onto its value and the release phase when the cell assumes a new value. The clock signals in the four clock zones are shifted from each other by a quarter of a period. The difference from clock in a CMOS circuits is that in QCA the clock is an external electric field that drives each individual cell, while in CMOS it is a signal inside the circuit that is used to synchronize larger structures.

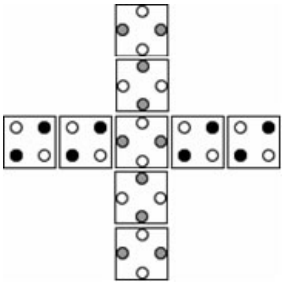
1.1.3 Wire

The simplest practical cell arrangement is given by placing quantum-dot cells in series, to the side of each other. For example, Figure 2b (p. 3) shows a 90 degrees wire. When the dots in the wire are rotated by 45 degrees this arrangement is called the 45-degree wire.

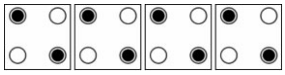
Wire crossing on the other hand can be done using two quantum-dot wires (one 90 degrees wire and one 45 degrees wire). The wire composed of one type passes perpendicularly "through" a wire of the other type, Figure 2a (p. 3). The first type of wire always propagates the same polarization, but the second type changes the polarization from one cell to the next. But, at crossing point there is no polarization change in either wire, therefore both wires preserve their own information.

1.1.4 Majority gate

As discussed earlier the most important logic gate in QCA is the majority gate. In this structure, the electrical field effect of each input on the output is identical and additive, with the result that whichever input state ("binary 0" or "binary 1") is in the majority becomes the state of the output cell — hence the gate's name. Majority gate can be used to implement logic conjunction (AND) and disjunction (OR), which together with negation (NOT) gives us a complete logic system.

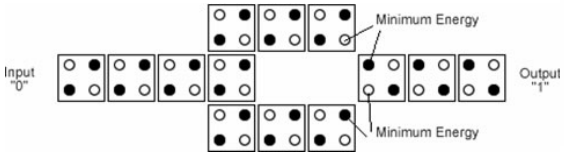


(a) Wire crossing.

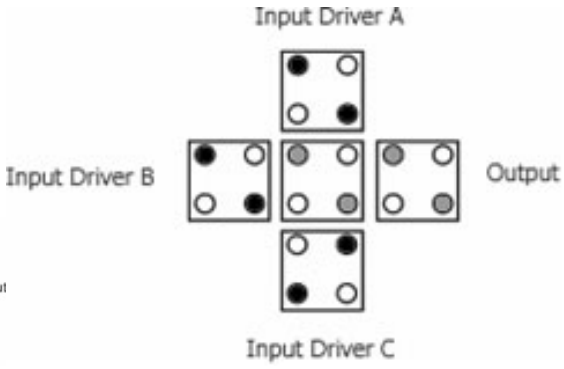


(b) 90-degree wire

Figure 2: Wires



(a) Negation.



(b) Majority gate.

Figure 3: Negation and majority gate.

1.1.5 Negation

The NOT gate is not constructed using the MV gate. It has a single input and output and it simply returns the opposite of the input. A standard implementation of the NOT gate is given in Figure 3a (p. 3).

Negation can be also implemented by having two cells (both either 90-degree or 45-degree) touch at the corner. Another possibility for negation is when a 45-degree wire connects to a 90-degree wire.

1.2 Reversibility

A reversible logic gate is an n-input n-output logic device with one-to-one mapping. This helps to determine the outputs from the inputs and also the inputs can be uniquely recovered from the outputs. The simplest example of a reversible logic function is negation. An example of a nonreversible logic function is logical conjunction or disjunction, since there are several input configurations that produce the same output.

$$(x_1, x_2, x_3) = \text{Toffoli}(\text{Toffoli}(\text{x}_1, \text{x}_2, \text{x}_3))$$

As with the NAND function which is universal there has been a lot of research trying to devise interesting universal reversible gates. Using only one type of a gate in a circuit simplifies manufacturing process.

1.3 The Toffoli gate

The Toffoli gate is an universal reversible logic gate proposed in by Tommaso Toffoli and can be used in the construction of any reversible circuit. Its structure is as follows. The gate has three inputs that can be labeled x_1, x_2, x_3 and three outputs y_1, y_2, y_3 described by equations in Figure 4b (p. 5).

The working of the Toffoli gate can be also described by equations in Figure 4c (p. 5). The Toffoli gate negates its first input if both the second and third input are equal to logic "1". The second and third input are passed through unchanged.

The input values that are passed through unchanged (only to provide reversibility) are sometimes called garbage outputs [1].

1.4 QCADesigner

Before we proceed to the implementation of the Toffoli Gate, we would like to add a few introductory words on the tool we used for drawing the QCA structures. The QCADesigner [6] is a design and simulation computer program for QCA circuits developed by the Walus Group at the University of British Columbia. It allows to design and simulate QCA circuits. It is written using the GTK2 graphic library and it is published under the General Public Licence. The official pages provide source code and prebuild binaries for Windows and deb and rpm packages for GNU/Linux.

Input			Output		
x_1	x_2	x_3	y_1	y_2	y_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

(a)

$$
\begin{aligned}
y_1 &= x_1 \oplus (x_2 \wedge x_3) \\
&= (x_1 \wedge \overline{x_2}) \vee (x1 \wedge \overline{x_3}) \vee (\overline{x_1} \wedge x_2 \wedge x_3) \\
y_2 &= x_2 \\
y_3 &= x_3
\end{aligned}
$$

(b)

$$
\text{Toffoli}(\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3) = \begin{cases} (\neg x_1, x_2, x_3) & \text{if } x_2 = x_3 = 1 \\ (x_1, x_2, x_3) & \text{otherwise.} \end{cases}
$$

(c)

Figure 4: Toffoli gate.

2 Implementing the Toffoli Gate

We implemented the Toffoli gate in QCADesigner in three stages. First we implemented the equation describing the first output, then we added the other two outputs and finally we modified the design to make it more compact while keeping the functionality unaffected.

We found a circuit implementing the XOR function in paper by Shah et al. [5] that was provided to us by prof. Mraz. Together with the basic QCA blocks described earlier we now have all the components that are needed to implement the Toffoli gate.

2.1 Previous work

Before implementing the gate ourselves, we searched the literature for existing implementations.

We found figures depicting the Toffoli gate in a paper by Chandra and Netam [2, fig. 7 on p. 73] and also in another paper by Mohammadi et al. [3, fig. 2b on p. 55]. Designs in those two papers are identical and do not actually implement the Toffoli gate even though both papers claim to be so. Still, the illustrations in the papers proved useful to demonstrate some of the design techniques used together in a larger example.

Next we studied the diploma thesis by Rolih [4] which is primarily concerned with three state logic but two state logic is also discussed. In this paper a working design of the Toffoli gate is depicted in Figure 4.5 (p. 29). Compared to this design, the one we come up with uses more cells (101 compared to 44) and the computation takes longer ($1\frac{1}{4}$ clock cycle compared to 1). On the other hand, since inputs and outputs in our design are easily accessible, it can be more easily integrated as a component into a larger circuit.

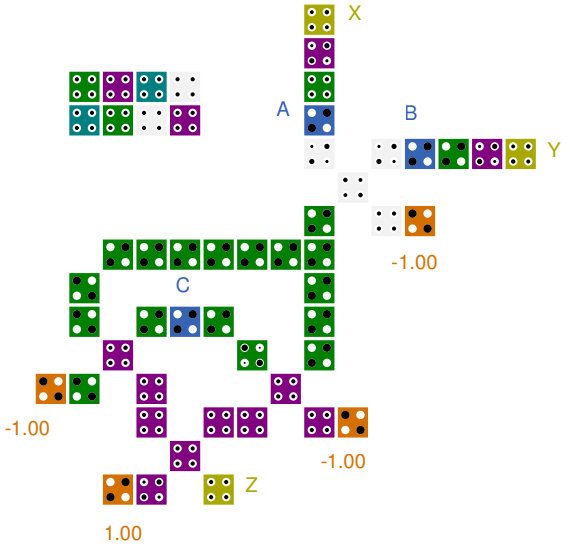


Figure 5: Implementation of Toffoli gate by Rolih [4]. The two rows of cell show color correspondence between this and the original figures.

2.2 Designing the first output

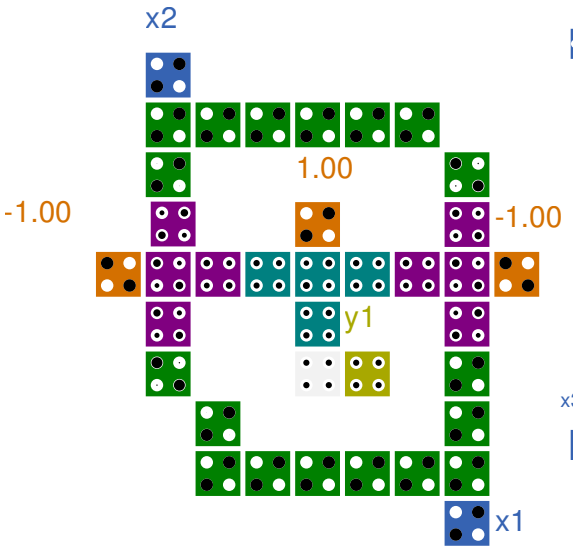
The function for the first output was obtained by connecting inputs x_2 and x_3 to an AND gate, output of which was connected to a XOR gate, together with input x_1.

2.3 Adding the other two outputs

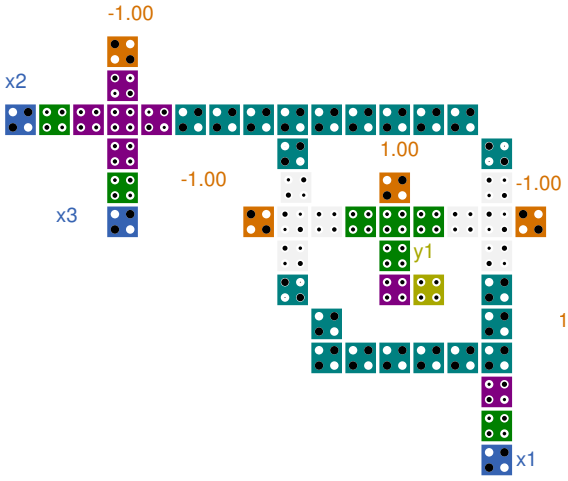
To make the design look intuitive and to simplify connecting gates we decided to have all inputs on the left side of our circuit and outputs on the right side. We used wire crossing to deliver one of the inputs to the place inside the structure where it was needed and also to take the output signal y_1 from inside out.

2.4 Modifying the design

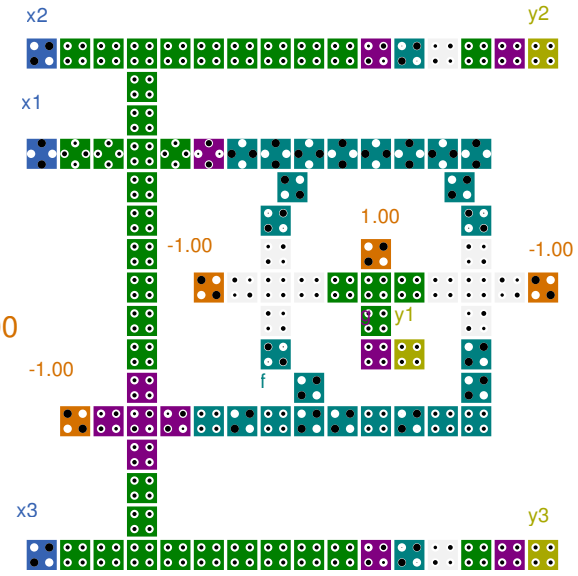
To improve the structure aesthetically and to conserve some space, we decided to modify the structure while keeping the function essentially intact. The modification in computation of output $y1$ consist of first negating inputs into the AND gate and then changing the AND gate into an OR gate. This allowed to shift some cells a little bit to the right. The first two changes together resulted in negating the output. The output y_1 is a 45 wire, which means that applying another negation to it can be done easily.



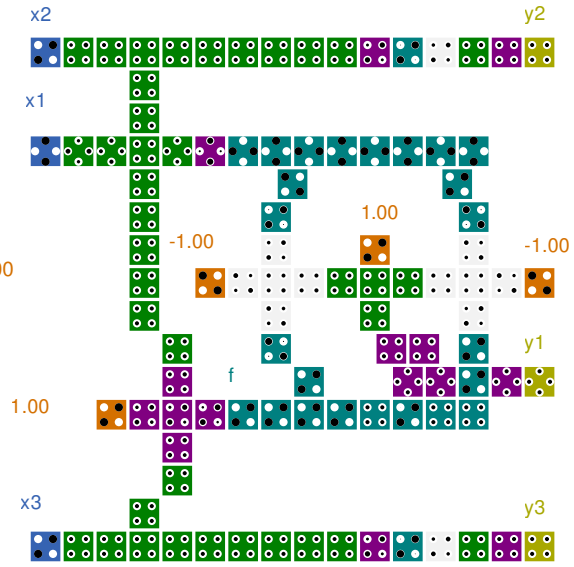
(a) The beginning.



(b) First stage.



(c) Second stage.



(d) Final stage.

Figure 6: The progression of our implementation of Toffoli gate.

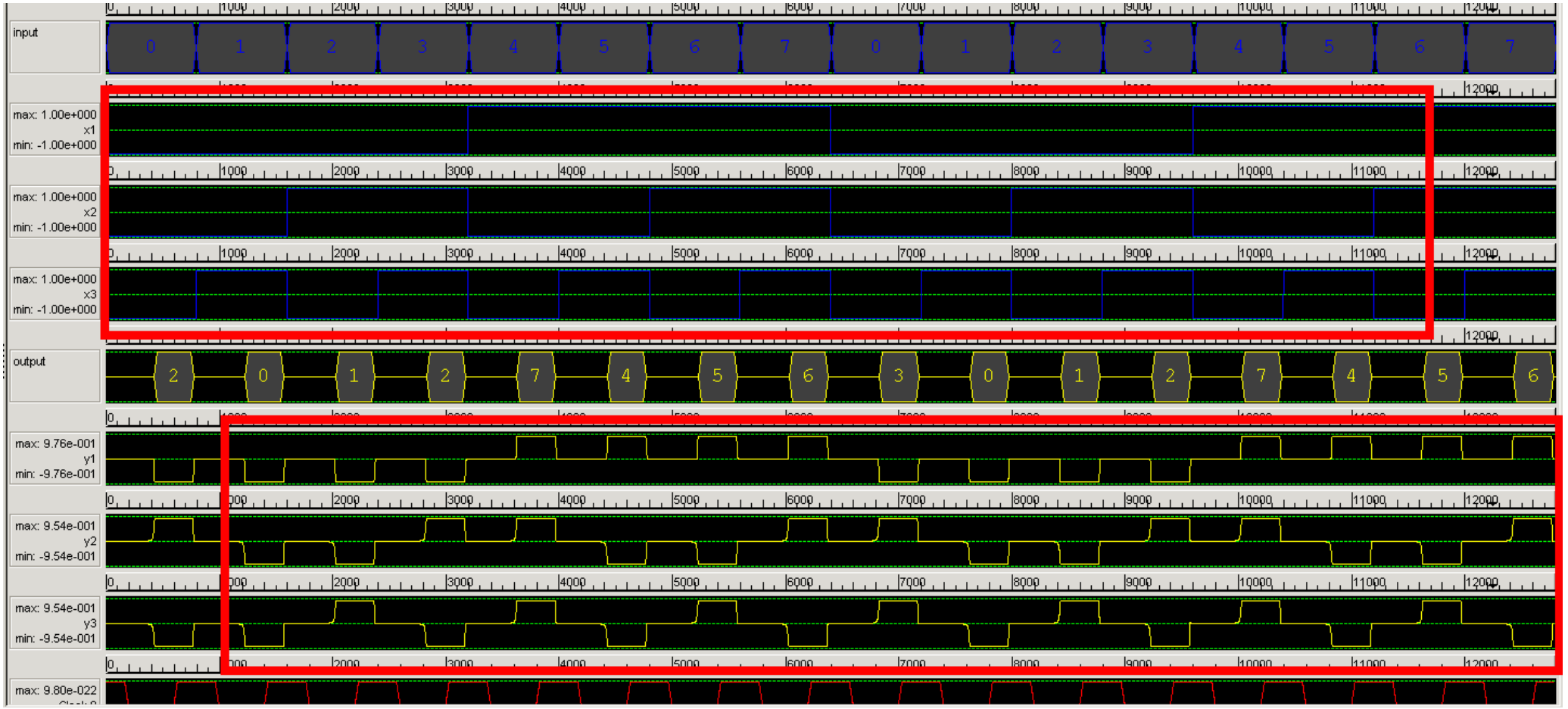


Figure 7: The working of our circuit.

2.5 Problems

We have not succeeded in running QCADesigner in Linux. The binary packages used out of date shared libraries and the source distribution did not compile successfully. Even though we eventually managed to get it compile, when executed it crashed immediately after the splash screen. We decided to use it on Windows XP and Windows 7 machines.

While designing the circuit, we did not have to debug it per se. Rather we sometimes needed to correct our misconceptions about how the QCA works. We describe one of them in the following subsection.

2.5.1 Outputs in QCA are influenced by clocking

Even though the QCADesigner does not visually differentiate it by color, the output elements belong to and are affected by clock zones. One of our problems we had was caused by assigning one of the inputs into a wrong clock zone. Because it was one clock zone after the right one, the signal looked distorted. Changing the clock zone fixed the problem.

3 Use of Toffoli gate

Toffoli gate is universal logic gate, meaning any logic function can be constructed using only Toffoli gates. This is easily proven by constructing a NAND gate using only a Toffoli gate or alternatively by constructing AND, OR and NOT gates, another complete logic system. To simplify the equations, we define a hypothetical Toffoli' gate which is a Toffoli gate without the two garbage outputs.

$$\text{Toffoli}'(\text{a},\text{b},\text{c}) = \begin{cases} \neg a & \text{if } a = b = 1 \\ a & \text{otherwise.} \end{cases}$$

With our new Toffoli' gate, $\text{NAND}(a, b)$ can be constructed as

$$\text{NAND}(a, b) = \text{Toffoli}'(1, \text{a}, \text{b}),$$

$\text{NOT}(x)$ becomes

$$\text{NOT}(x) = \text{Toffoli}'(\text{x}, 1, 1),$$

finally the $\text{AND}(a, b)$ gate is

$$\text{AND}(a, b) = \text{Toffoli}'(0, \text{a}, \text{b}).$$

Having constructed these gates, the $\text{OR}(a, b)$ gate can be then constructed from the previous gates using one of the De Morgan rules

$$\begin{aligned} \text{OR}(a, b) &= \neg(\neg a \wedge \neg b) \\ &= \text{Toffoli}'(\text{Toffoli}'(\text{Toffoli}'(\text{a}, 1, 1), \text{Toffoli}'(\text{b}, 1, 1), 0), 1, 1). \end{aligned}$$

4 Conclusion

The article by Chandra and Netam [2] suggests that while the Toffoli gate is theoretically interesting, its practical applications in QCA reversible circuits are limited. When reversible circuits are designed, authors usually use the standard gates only as a first step and develop modified reversible gates that would allow them to minimize the amount of garbage outputs in the specific problem they are solving and use less cells.

References

[1] Md Selim Al Mamun, Indrani Manda, and Md Hasanuzzaman. Design of universal shift register using reversible logic. 2012.

[2] Saroj Kumar Chandra and Deepak Kant Netam. Exploring quantum dot cellular automata based reversible circuit. *interaction*, 2(1March), 2012.

[3] Zahra Mohammadi, Majid Mohammadi, and Mahdi Hasani. Designing of testable reversible qca circuits using a new reversible mux 2×.

[4] Mark Rolih. *Analiza možnosti realizacije logičnih reverzibilnih vrat v trostanjskem kvantnem celičnem avtomatu*. PhD thesis, Univerza v Ljubljani, 2013.

[5] NA Shah, FA Khanday, and J Iqbal. Quantum-dot cellular automata (qca) design of multi-function reversible logic gate. *Communications in Information Science and Management Engineering*.

[6] Konrad Walus, Timothy J Dysart, Graham A Jullien, and R Arief Budiman. Qcadesigner: A rapid design and simulation tool for quantum-dot cellular automata. *Nanotechnology, IEEE Transactions on*, 3(1):26–31, 2004.