QCA-LG: A tool for the automatic layout generation of QCA combinational circuits

Tiago Teodósio and Leonel Sousa
INESC-ID/IST, TU Lisbon
R. Alves Redol 9, 1000-029, Lisboa, PORTUGAL
Email: {ttte,las}@sips.inesc-id.pt

***Abstract*—Quantum-dot Cellular Automata (QCA) is a promising successor for CMOS transistor technology, while allowing the implementation of logic circuits using quantum devices, such as quantum dots or single domain nano magnets, a new set of tools must be developed to assist the design and implementation process. Examples of such tools are the QCADesigner for handmade layout and physical simulation, and also tools for majority logic optimization. Since no tool is available for assisting the QCA layout generation, we propose tool to automatically generate the layout of QCA circuits. This tool, designated by QCA-Layout Generator (QCA-LG), was integrated in a general QCA technology design flow, accepting the most used formats of the synthesis tools and producing the layout output according to the QCADesigner tool. Therefore, the layout of a logical circuit described in VHDL is automatically generated, and can be further optimized by hand and simulated using the QCADesigner. Examples of layouts automatic generated by the QCA-LG are presented for simple logical circuits, and are also compared with optimal layouts designed by hand.**

I. Introduction

In the medium term scenario, the present CMOS technology will fail to support the growth rates needed by the industry. The QCA is a computation paradigm that can be implemented in different physical systems, and at least the following were proposed: *i*) metal-island quantum-dots and tunneling effect junctions [1]; *ii*) nano magnetic particles with single magnetic moment domain behaviour [2]; *iii*) molecular [3]. The quantum-dot is one of the most promising technologies, achieving a good efficiency in terms of operation speed and density of integration. As in QCADesigner [4], in this project is assumed the quantum-dot implementation of QCA. In the last few years, software tools have been developed to assist the design and of QCA circuits and layouts. A design flow for QCA technology has been proposed in [5]. In this design flow, only the last three phases, logic mapping, technology mapping, and simulation, are different from the design flow for the CMOS technology. Logic mapping can be optimized by favouring the usage of majority function. Although this logic mapping can be done with available tools, such as the ones from Synopsys[6] and MVSIS[7], results are not optimal. Therefore, two methods [8], [9] have been proposed to automatically map the logic functions to majority gates.

QCADesigner [4] is a well known simulation tool and layout editor. QCADesigner is used to manually draw the layout and perform physical simulation. This tool, that has been widely used to develop and simulate QCA circuits, has the possibility of importing and exporting already generated layouts into files, according to a predefined format.

The main goal of this paper is to propose and to present the QCA-LG tool that was specifically designed and programmed to automatically generate the layout of QCA combinational circuits. This technology mapping tool maps a logic combinational circuit into QCA gates and produces the corresponding layout. The QCA-LG supports some well-known netlist formats adopted by commercial and academic synthesis tools; it can store the generated layouts in the format defined by the QCADesigner.

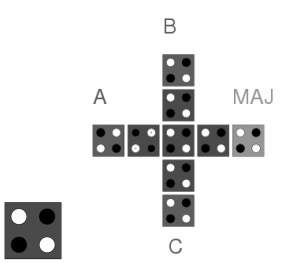
Therefore, it is achieved for the first time a set of tools that can map the combinational functions in QCA layouts that can be visualized and physically simulated.

This paper is organized as follows. In section II a brief introduction to QCA technology is presented. The QCA-LG tool is presented in section III, and layouts generated by the QCA-LG for simple combinational circuits are presented in section IV. Section V concludes the paper.

II. QCA circuits and technology

QCA is a new computing paradigm based on quantum dots [10], [11]. QCA computing is based on the electrostatic interaction between QCA cells. The basic four-dot QCA cell in Figure 1(a) can be viewed as a square charge container with four quantum dot locations at the corners. Each cell has two electrons in excess, localized on antipodal quantum dots due to the Coulomb repulsion. In Figure 1, white circles represent unoccupied dots while a black circle denotes that the dot is occupied by an electron. These electrons are only allowed to tunnel between quantum dots inside a cell, and therefore, only two different charge distributions are possible in a cell. Although these two states of an isolated cell are equal in energy, they are observable and can be used to encode the logical levels '0' and '1' of a bit. Moreover, when two or more cells are put close to each other, the two states are not energetically equal, due to the electrostatic interaction between those cells, and thus, the lowest energy state of a QCA system is achieved. Basic logic gates, such as AND, NOT and OR can be implemented by placing together QCA cells according to different configurations. Among these basic gates the basic structure of QCA logic is the three input majority gate M depicted in Figure 1(b), whose output is defined by the majority vote of the three inputs: $M(A, B, C) = A.B + B.C + A.C$;

thus, the logic product can be performed as $M(A, B,' 0')$ and the logic sum as $M(A, B,' 1')$.



(a) QCA cell **(b)** QCA layout of a majority gate

Fig. 1. QCA cell and gates.

Traditional implementations have been considered in this work for all the basic logic gates, except the QCA inverters. A binary wire is formed by juxtaposing QCA cells in a linear array, that can be also seen as a set of half cells [12]. Since it is known that a wire of half cells works as an inverter chain [13], a QCA inverter can be attained by simply introducing or removing a half cell to a given wire. We adopt this approach in this work because is more efficient: it was chosen to remove a half cell and to distribute the spare space along the inter cell spacing in a wire, in a way that start and end points of the original wire composed by complete cells is preserved (see Figure 2). In order to safely remove a half cell from a given wire, the wire must have a minimum number of cells, otherwise it is not possible to distribute the half cell distance in a way that they properly interact with each others.

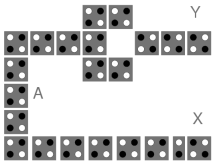


Fig. 2. Traditional and the proposed structure for signal inversion: the input cell is labeled as A, the usual inverter's output is Y, and the output of the in wire inverter is X.

The QCA binary wires are made of QCA cells and two different approaches have been proposed to solve crossing conflicts: in plane crossing and multi layer crossing [14]. Multi layer crossover was adopted by the QCA-LG tool. However, with small changes, the QCA-LG can also support in plane crossover. Although multi layer crossover seems safer and gives better simulation results, it is expected to be harder to fabricate than in plane crossovers. It should be noticed that, in multi layer crossover a connection through an even number of separation layers acts as a QCA inverter.

For the clocking mechanism in QCA, four clock signal of equal frequencies can be considered: one of these clocks is considered the reference ($phase = 0$) and the others are delayed one, two, and three quarters of a period, respectively ($phase = \pi/2, \pi, 3\pi/2, $). Each of these clock signals imposes its pace to a given set of layout regions[15]. Signals are pushed from one clock region to another as the phase of the clock of these regions increases. A given clock zone receives the signal from adjacent clock zones, having a clock in advance by one quarter period.

III. QCA-LG: A TOOL FOR GENERATING QCA LAYOUTS

The main goal of this work is to automatically generate a quantum dot layout for a given combinational circuit, in a format compatible with the QCADesigner tool, and not to develop a professional "Place and Route" tool. The flow of the program actions is shown in Figure 3. The input interface

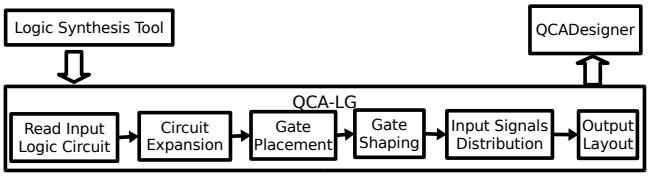


Fig. 3. Block diagram of the operations carried out by the tool.

allows data be read from files representing logic circuits, according to two different formats: *i*) *LSI*, which is a format supported by the Synopys symthesis tools; *ii*) *Gate*, the logic netlist format used by the original MVSIS and the SIS[16] successor tools.

A Lex&Yacc parser was specified for each of the two different formats. The use of Lex&Yacc facilitates support of new formats.

Libraries with only the supported components have to be specified in order to, given a design, generate a netlist bases only on those components. Libraries with only majority gates, NOT gates, and 2-inputs AND, OR, NAND and NOR gates were specified, both for the *LSI* and the *Gate* tools. The circuits are intrenally represented by the tools as directed graphs and stored in a hash, where each object represents a gate of the circuit or a primary input.

A. Circuit Expansion

Expansion is the first operation performed over the circuit representation; every shared node is duplicated. At the end of this step the fanout of every gate is only one. This operation makes the place and route task easier, avoiding wire crossing at the expense of an increase in the circuit area.

The logic replication is not supposed to increase the circuit delay, but it can happen, since the extra area implies extra length in some wires, and sometimes this forces wires to be split up over some more clock zones.

Circuit expansion is performed by visiting the nodes of the graph (gates) using a breadth-first approach exploration of the circuit. Starting from each primary output towards the inputs, every node is marked as visited if it was not visited yet; on the other hand, every time a node is revisited it is duplicated, not only the node itself but also, all the nodes included in the

sub tree rooted by it. The duplication is a recursive process that may "explode" if there are loops in the circuit. However, in combinational logic circuits no computational loops should occur.

This operation splits the circuit in independent sub circuits; each one of these sub circuit is rooted at a primary output of the circuit and is separately processed in the *gate placement* step.

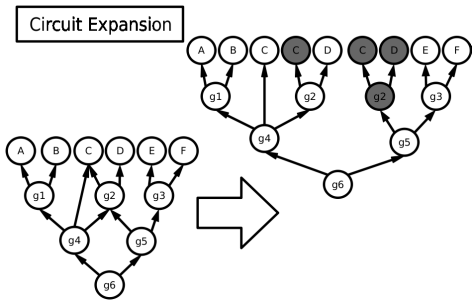


Fig. 4. Circuit Expansion operation; grey gates are copies.

B. Gate Placement

From the gate level to the physical level, three different referentials of 2D Cartesian coordinates are used:

- *gate level coordinates* - used to define the relative positions of the gates; each unit in the vertical direction represents one gate level, the vertical distance between the inputs and the corresponding output; a unit in the horizontal direction represents the minimum width of one wire, in practice three cells, so that the separation of two adjacent wires corresponds to the width of two QCA cells.
- *cell level coordinates* - the unit in this referential corresponds to the size of one cell in each direction.
- *physical coordinates* - which are obtained by translating the cell coordinates into the physical coordinate system, namely the one adopted by the QCADesigner; the unit is the nanometer, and each QCA cell has 18.0 units of width and length, and the spacing between each cell is 2.0 units in booth directions.

The coordinates of a gate in a sub-circuit are computed in three phases, all manipulating coordinates in the gate level referential.

The first phase determines the level of each gate in the graph, by applying breadth-first search starting from the primary outputs. The graph level is the *y axis value* in gate level coordinates and to the inputs of a gate it is assigned the gate level plus one.

In the second step the gates within each level are numbered; these numbers range from 0 to $(3{*}y_{max})-1$. Three subsequent numbers are assigned to each of the inputs of the n_{gate}, starting from the left most input $n = 3 * n_{gate}$ and ending in the right most input $n = 3 * n_{gate} + 2$. Note that only for Majority Gates there is a central $n = 3 * n_{gate} + 1$.

In the third phase, the x gate level coordinate is calculated according to the following expression:

$$x = (1 + 2n)\frac{(3^{y_{max}-y} + 1)}{2} - n, \qquad (1)$$

corresponding to the coordinates of the nodes in a complete ternary tree.

After these three steps, the gates in the present sub circuit are arranged as an incomplete ternary tree (see Figure 5), in which some nodes lack the middle child. This is the case of all non majority gates, which leads to an undesired waste of area. Therefore, it can be concluded that the proposed method is optimal for circuits exclusively based on majority gates, which are the most efficient in the QCA computation paradigm. It is important to notice that inverters are ignored in the this step, because they will be treated separately.

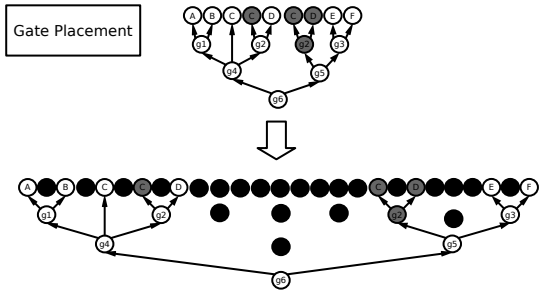


Fig. 5. Gate placement operation; black circles show wasted area.

C. Gate Shaping

After defining the place in the gate level coordinates, the place and size of each gate are computed.

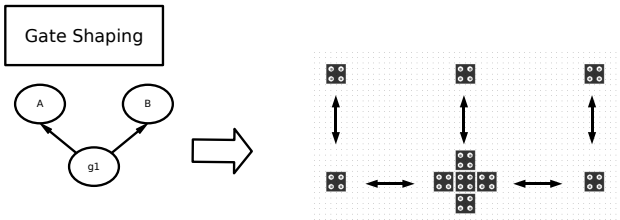


Fig. 6. Gate shaping operation.

At this point, takes place the mapping from the gate level referential to the cell level referential. This mapping corresponds to simple linear equations in each dimension, of the type $x_{cell} = (x_{gate} + x_{gate}offset)x_{factor}$ and $y_{cell} = (y_{gate})y_{factor} + y_{cell}offset$ where $x_{gate}offset$ is used to separate the various sub circuits along the layout, and $y_{cell}offset$ is used to adjust the gate position, namely when the input wires have to be split into more than one clock zone, as it will be explained later.

QCA Inverters are built directly in the wires. NAND and NOR gates are processed as AND and OR gates, respectively, with an additional Inverter at the gates output.

1) Integrated routing: The input wires of each gate are directly connected to the corresponding output cell. The input wire length is calculated from the coordinates of each gate and the coordinates of its inputs. If the wire length exceeds the maximum clock zone length [17], it will be split into more than one clock zone.

For gates with two or three inputs, if the longest wire has to be split into more than one clock zone, then the shortest wire also has to be split into the same number of clock zones to maintain the input arrival timing. This usually means the shortest wires have to become longer, in order to respect the minimum clock zone length. Since the horizontal alignment of the first cell of all input wires must be ensured, when one wire gets longer the others must also get longer in the vertical direction. When an increase in wire length occurs, it is necessary to guaranty that the number of cells per clock zone does no exceed the maximum clock zone length, repeating the last steps if needed.

A half QCA cell replaces a full QCA cell when a given input wire is connected to a NOT, NAND or NOR gate. This replacement implies some extra space between cells. This feature can not be easily achieved when drawing the layout by hand using QCADesigner; even if "snap to grid" option is turned off, the half QCA cell is not available as an individual design object. However, QCADesigner is able to import and simulate any rectangular shaped object with an arbitrary number of quantum dots.

2) Circuit synchronization: In order to ensure the proper operation of the circuit, it is necessary to set the clock zone of each cell output to the preceding clock zone of the input cell of the next gate towards the primary outputs. To achieve this goal, the clock zone of the majority gate is set arbitrarily and then the clock zones of the input wires are set accordingly given that signals flow from the inputs, in an earliest clock zone, along the input wire (ascending clock zones) to the output, which corresponds to the latest clock zone. The clock zone in which the input wires start is then saved in the gate's record for future use.

As the circuit is drawn from the outputs towards the inputs, except for the primary outputs, every gate is drawn after the gate they feed. Thus the clock zone of the output cell of each gate is known as soon as its directly dependant gate is drawn.

D. Input Signals Distribution

As one given primary input signal can be used as input for more than one logic gate, that signal must be distributed from one unique source to all the places where it is needed. The expansion of the circuit generates copies of some logic gates, which means that some input signals must be delivered to some extra points.

An input signal also must arrive at its destination(s) synchronous with the other signals. To assure this temporal coherence, signals' destinations are sorted by (descending) priority, considering that the primary input of the circuit is place above the leftmost destination and as low as possible. A main distribution horizontal wire will start above the left most input copy, and will be extended to the right until it reaches the *x* position of the right most input. All the distribution wires are placed in a different layer than the logic gates. This operation is performed using the cell level coordinates; the expression used to evaluate the "urgency" to arrive at a given destination (x,y) at a given moment (clk) is:

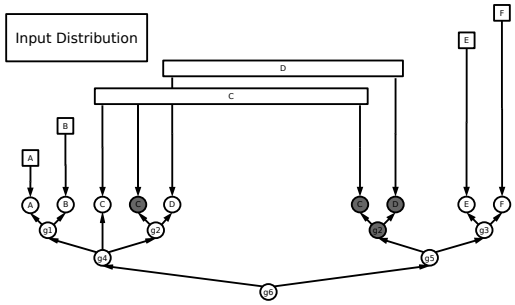


Fig. 7. Representation of the Input Signals Distribution operation.

$$pri(x, y, clk) = (x - x_{min}) + (y - y_{max}) - (clk - clk_{min}) * Zone, \quad (2)$$

where x_{min} is the minimum horizontal coordinate value found among all destinations of the signal (left most), y_{max} is the maximum vertical coordinate (lower, as y values grow downwards) and clk_{min} is the minimum earliest zone. Zone is the maximum length allowed for a clock zone. The destination is considered to have higher priority as the distance to the primary input grows and the number of clock zones to transverse gets reduced.

An iterative method is used to determine where the main distribution wire has to be placed, and where the signal has to be derive to secondary distribution wires, to achieve logic gate where it is needed. On each iteration, the borders of the clock zones in the main distribution wire are updated according to the most restrictive destination. The sorted set of destinations is scanned and, given the present position of the main distribution wire, it is checked if it is possible to respect the maximum and minimum length allowed for a clock zone. If the signal can travel correctly to every destination, the main distribution wire achieves its final position, otherwise the main distribution wire is pushed up; this operation is carefully performed in order to avoid collisions with main distribution wires of other primary inputs. These actions are taken in each iteration. The initial conditions are critical to ensure this iterative method stops, they depend heavily on the most restrictive destination. Consequently the sorting step is fundamental to assure the success of the method.

The output layout format aims to be compatible with QCADesigner (version 2.0.3) [4]. The produced layout is stored as a QCA layout block, and can be imported into any QCADesigner layout.

IV. Generated QCA layouts

Figure 8 presents an handmade designed layout of a binary 2 to 1 multiplexer. Simulation results for this layout were

obtained with QCADesigner, where it was observed the existence of one clock cycles delay between inputs and the correct output, as expected.

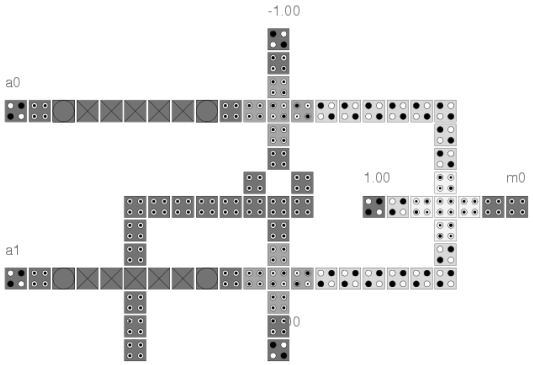


Fig. 8. Handmade layout of a 2 to 1 multiplexer.

The automatically generated layout for the binary 2 to 1 multiplexer is presented in Figure 9. This layout was also simulated and the delay between the inputs and the correct output was of two clock cycles.

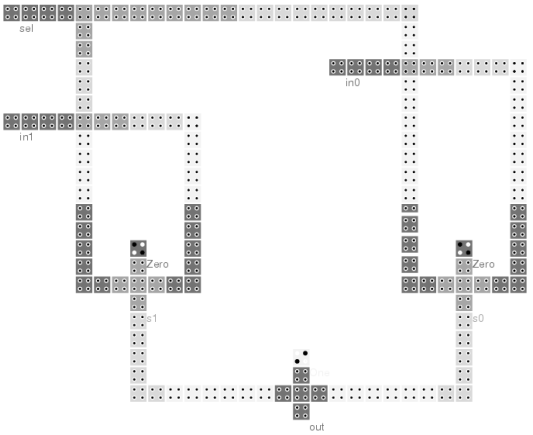


Fig. 9. Layout of a 2 to 1 multiplexer generated by the tool.

The QCA-LG tool was also used to produce the QCA layout of several other combinational circuits, including decoders, half and full adder, which can be found at the website dedicated to the project (*http://sips.inesc-id.pt/las/QCA-LG/*).

V. Conclusions

The proposed QCA-LG tool can be used to produce suitable QCA layouts for combinational circuits. The QCA-LG accepts standard netlist formats, such as *LSI* and *Gate*, and generates QCA layouts that can be physically simulated by the well known QCADesigner tool. The presented results show that the developed QCA-LG tools are able to automatically generate layouts for small sized circuits. However, for medium and large sized circuits the wasted area can be significant, namely for circuits not exclusively composed by majority gates. More research has to be performed in order to optimize the gates placement, which is statically performed and can be one of the main sources of inefficiency.

With QCA-LG the design flow for QCA technology is almost complete. Combinational VHLD/Verilog circuits can be mapped into logic netlists with existing synthesis tools. These netlists can be transformed into QCADesigner compatible layout using QCA-LG, and validated by physical simulation.

QCA-LG is still under development and an evaluation version of the QCA-LG tools is available for download at *http://sips.inesc-id.pt/las/QCA-LG/*.

References

[1] A. O. Orlov, I. Amlani, R. K. Kummamuru, R. Ramasubramaniam, G. Toth, C. S. Lent, G. H. Bernstein, G. L. Snider, W. Porod, and J. L. Merz, "Quantum-dot cellular automata: Introduction and experimental overview," *IEEE NANO*, pp. 465–470, October 2001.

[2] A. Imre, L. Ji, G. Csaba, A. Orlov, G. H. Bernstein, and W. Porod, "Magnetic logic devices based on field-coupled nanomagnets," *IEEE Semiconductor Device Research Symposium*, pp. 25–25, December 2005.

[3] Y. Lu, M. Liu, and C. Lent, "Molecular electronicsfrom structure to circuit dynamics," *IEEE NANO*, vol. 1, pp. 62–65, June 2006.

[4] K. Walus, V. Dimitrov, G. Jullien, and W. Miller, "QCADesigner: A CAD tool for an emerging nano-technology," *Micronet Annual Workshop*, 2003, web: http://www.qcadesigner.ca.

[5] S. C. Henderson, E. W. Johnson, J. R. Janulis, and P. D. Tougaw, "Incorporating standard CMOS design process methodologies into the QCA logic design process," *IEEE Transactions on nanotechnology*, vol. 3, no. 1, pp. 2–9, March 2004.

[6] S. Inc., "Synopsys tools," May 2003, version 2003.06 for linux.

[7] D. Chai, J.-H. Jiang, Y. Jiang, Y. Li, A. Mishchenko, and R. Brayton, *MVSIS 2.0 Users Manual*, Department of Electrical Engineering and Computer Sciences.

[8] R. Zhang, P. Gupta, and N. K. Jha, "Synthesis of majority and minority networks and its applications to QCA, TPL and SET based nanotechnologies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 229–234, 2005.

[9] Z. Huo, Q. Zhang, S. Haruehanroengra, and W. Wang, "Logic optimization for majority gate-based nanoelectronic circuits," *Circuits and Systems*, p. 4 pp., May 2006.

[10] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," in *Proceedings of the IEEE*, vol. 84, no. 4, April 1997, pp. 541–557.

[11] K. Walus and G. A. Julien, "Design tools for an emerging SoC technology: Quantum-dot cellular automata," in *Proceedings of the IEEE*, vol. 96, no. 6, June 2006, pp. 1225–1244.

[12] A. O. Orlov, I. Amlani, R. K. Kummamuru, R. Ramasubramaniam, G. Toth, C. S. Lent, G. H. Bernstein, and G. L. Snider, "Experimental demonstration of clocked single-electron switching in quantum-dot cellular automata," *Applied Physics Letters*, vol. 77, no. 2, pp. 295–297, July 2000.

[13] A. O. Orlov, I. Amlani, G. Toth, C. S. Lent, G. H. Bernstein, and G. L. Snider, "Experimental demonstration of a binary wire for quantum-dot cellular automata," *Applied Physics Letters*, vol. 74, no. 19, pp. 2875–2877, May 1999.

[14] K. Walus, G. Schulhof, and G. A. Jullien, "High level exploration of quantum-dot cellular automata (QCA)," *Signals, Systems and Computers*, vol. 1, pp. 30–33, 2004.

[15] G. Toth and C. S. Lent, "Quasiadiabatic switching for metal-island quantum-dot cellular automata," *Applied Physics Letters*, vol. 85, no. 5, pp. 2977–2984, March 1999.

[16] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, *SIS: A System for Sequential Circuit Synthesis*, memorandum no. ucb/erl m92/41 ed., Department of Electrical Engineering and Computer Sciences, May 1992.

[17] K. Kim, K. Wu, and R. Karri, "Towards designing robust qca architectures in the presence of sneak noise paths," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, vol. 2. IEEE Computer Society, 2005, pp. 1214–1219.