Evaluating the Impact of Interconnections in Quantum-Dot Cellular Automata

Frank Sill Torres[1,2] Robert Wille[1,3] Marcel Walter[2] Philipp Niemann[1,2] Daniel Große[1,2] Rolf Drechsler[1,2]
[1]Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
[2]Group of Computer Architecture, University of Bremen, Germany
[3]Johannes Kepler University Linz, Austria
{frasillt, m_walter, pniemann, grosse, drechsler}@uni-bremen.de, robert.wille@jku.at

***Abstract*—Quantum-Dot Cellular Automata (QCA) are an emerging nanotechnology with remarkable performance and energy efficiency. Computation and information transfer in QCA is based on field forces rather than electric currents. As a consequence, new strategies are required for design automation approaches in order to cope with the arising challenges. One of these challenges rises from the fact that QCA is a planar technology. That means, logic gates as well as interconnection elements are mostly located in the same layer. Hence, it is expected that interconnections have higher influence on the final design costs than in conventional integrated technologies. For the first time, this paper presents an extensive study on the quantification of this impact. Therefore, we consider the entire design flow for QCA circuits from the initial synthesis (using different synthesis approaches) to the corresponding placement on a QCA grid. Then, we characterize the respectively obtained QCA circuits in terms of area, delay and energy costs. The obtained results indicate that the impact of interconnections in QCA is indeed substantial. Design costs including or not including interconnections differ by several orders of magnitudes, which motivates to completely re-think how logic synthesis for QCA circuits shall be conducted in the future.**

***Keywords*—Quantum-dot Cellular Automata, Interconnections, Layout design, Field-Coupled Nanocomputing**

I. Introduction

Quantum-dot Cellular Automata (QCA) [1], [2] are an emerging technology in which computations are conducted in a fundamentally different way compared to conventional systems relying e. g. on CMOS. Here, information is stored in terms of the polarity of small cells and can be propagated to adjacent cells using electrostatic force (Coulomb interaction). This results in a *Field-Coupled Nanotechnology* (FCN) that allows to represent and process binary information [3]. Moreover, this way of representing and processing information is doable with highest processing performance and remarkably low energy dissipation—as confirmed by several theoretical and experimental studies (see e. g. [4], [5]). Overall, this makes QCA a promising alternative to conventional integrated circuit technologies. As a consequence, numerous contributions on their physical realization have been made in the past, e. g. based on molecules [6], nano-magnets [7], or silicon atoms [8], [9].

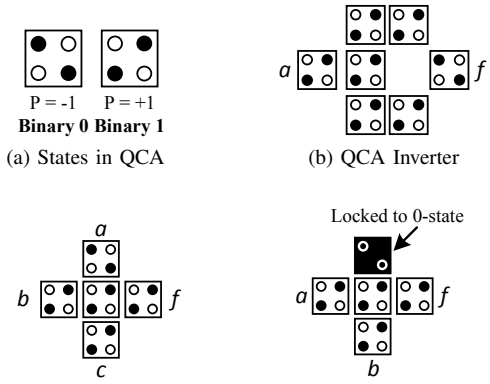
In parallel, there has been some research on the design of QCA circuits. In the past, the majority of QCA circuits have been derived manually—including e. g. realizations of arithmetic circuits [10], processors [11], or FPGAs [12]. Besides that, there are also recent developments towards logic synthesis of QCA circuits (see e. g. [13]–[19]). These contributions are essential since, as for conventional circuitry, complex systems can eventually only be realized with the help of efficient *automatic* design methods.

Most of the currently available methods follow a two-stage design flow in which the desired function is synthesized first in terms of a conventional circuit (disregarding any technology constraints). Afterwards, the resulting (conventional) circuit is mapped into a proper QCA circuit using corresponding building blocks (without any further modifications of the initial netlist). Consequently, the resulting circuits are mainly optimized with respect to conventional cost metrics thus far and, hence, are likely non-optimal with respect to QCA-specific costs in terms of area, delay, and energy dissipation.

In order to address this problem and to "lift" these physical objectives to a higher level of abstraction (namely the abstraction level in which the actual logic synthesis is conducted), a corresponding cost model for logic synthesis has recently been introduced in [20]. Here, proper and technology-specific cost functions for area, delay, and energy dissipation for a gate library including frequently used elementary building blocks is provided, which, in principle, can easily be incorporated into existing logic synthesis methods—thereby allowing for a QCA-specific synthesis. However, the considerations in [20] also unveiled that a pure focus on the costs of single gates (as common in conventional logic synthesis) is not sufficient in order to properly guide the synthesis process. In fact, also *interconnections* such as wires, fan-outs and crossovers (whose costs are usually considered negligible in conventional logic design) have a significant impact on area, delay, and energy dissipation.

However, thus far, it remains unknown whether this impact of interconnections in QCA circuits indeed has a substantial effect and, hence, should explicitly be considered in the future. In this work, we are addressing this issue by conducting several empirical evaluations. To this end, we utilize the model proposed in [20] as well as existing methods for logic synthesis (developed for conventional circuits) and QCA placement. The obtained results show that the impact is indeed substantial. Considering interconnections vs. not considering interconnections actually yields to differences amounting to several orders of magnitudes. This clearly motivates to completely re-think how synthesis for QCA circuits shall be conducted in the future.

The remainder of this work is organized as follows. First, the basics on QCA are reviewed in Section II. Afterwards, Section III introduces the model for logic synthesis of QCA designs in order to keep this work self-explanatory. The following Section IV discusses the realization and function of interconnections in QCA designs. Section V presents the applied environment for analysis of the impact of interconnections in QCA designs, while Section VI discusses the simulation results. Finally, the work is concluded in Section VII.



(a) States in QCA (b) QCA Inverter

(c) QCA Majority (d) QCA AND

Fig. 1: QCA realizations of basic operations

II. QCA Circuits and Their Design

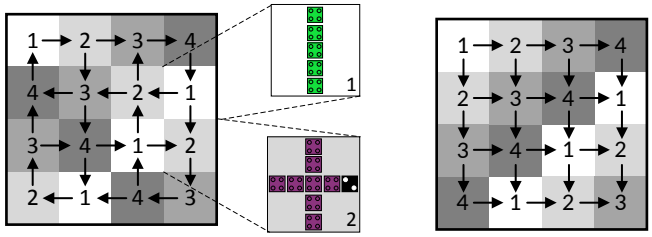
The basic element of *Quantum-Dot Cellular Automata* (QCA) [1], [2] are *cells* that interact via local fields and, thus, allow for the realization of logic functions. A QCA *cell* has a square shape and contains in each corner a *quantum dot* which is a structure able to confine an electric charge [2], [21]. Further, each QCA cell possesses two free and mobile electrons that are able to tunnel between adjacent dots, while a potential barrier prevents tunneling to the outside of the cell. Due to Coulomb interaction, the two electrons tend to locate themselves at opposite corners of the cell—eventually leading to two possible *cell polarizations* (namely $P = -1$ and $P = +1$ which can be defined as binary 0 and binary 1, respectively).

Example 1. *Both stable states are depicted using the QCA cells in Fig. 1a, where circles denote quantum dots (○) and black bullets illustrate electrons (●). Usually, the state shown in the left-hand side of Fig. 1a is defined as binary 0, while the state shown in the right-hand side of Fig. 1a is defined as binary 1.*

QCA cells placed next to each other interact via Coulomb forces such that the polarization of one cell influences the polarization of the other. Consequently, basic Boolean operations such as NOT, AND, OR, Majority, etc. can be realized.

Example 2. *Figures 1b to 1d exemplarily depict some basic logic functions implemented by QCA cells. More precisely, Figure 1b shows the realization of the NOT function, where e. g. a binary 1 is copied to two paths, which are then combined diagonally, such that the binary 1 is inverted to a binary 0 (from left to right). The structure depicted in Fig. 1c implements the Majority function, where e. g. a binary 0 from input* a *competes with two binary 1s coming from inputs* b *and* c*. The output follows the majority of the input values, which in this case is a binary 1. Figure 1d shows the realization of the AND function which is similar to the Majority gate. However, the top QCA* cell *is locked to the 0-state such that both inputs* a *and* b *compete with each other and the binary 0. The structure can be turned into an OR gate by changing the top* cell *to the 1-state.*

In order to avoid metastability, QCA cells have to enter a neutral state before assuming a new polarization [22]. Further, it has to be ensured that data is only passed between QCA structures if the source structure remains in a stable state, while the receiving structure is able to change its polarization. To this end, the state of the QCA cells can be controlled by an external electromagnetic field, in the following named *clock*, that regulates the interdot barriers within a QCA cell controlling whether the cell can be polarized or not [23]. As an external clock consists of four phases, usually four external clocks numbered form 1 to 4 and phase-shifted by one phase are employed.



(a) Fixed arrangement of clock zones based on USE clocking scheme proposed in [24]

(b) Diagonal Clocking where information flows from top left to bottom right

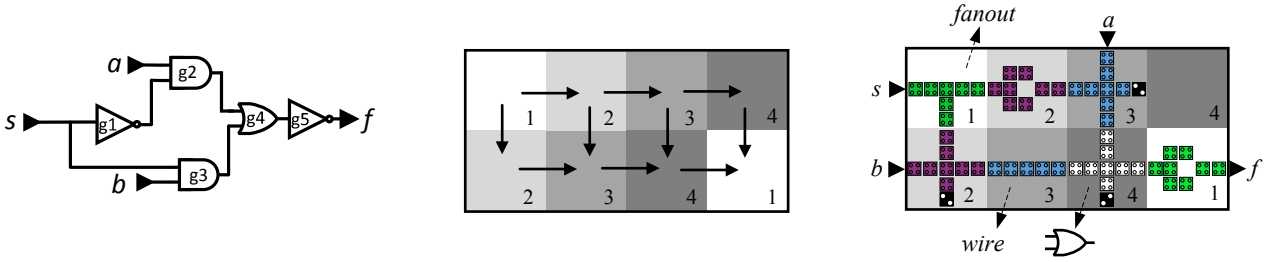
Fig. 2: Exemplary grids of clock zones

Now, in order to provide a proper data transfer between QCA gates, it has to be ensured that the output value of a gate is applied to the input of a following gate at exactly that time when the following gate is able to accept a new value, i. e. to change its polarization. Therefore, all inputs of a QCA gate must be fed by structures that are driven by a preceding clock signal, e. g. a gate of clock 2 receives its data from gates of clock 1.

In order to comply with fabrication constraints, cells are commonly grouped in square or rectangular shaped *clock zones*, or *tiles*, such that all cells within a clock zone are controlled by the same external clock [24], [25]. These clock zones are organized as a grid based on fixed or free arrangement styles. Further, each clock zone may contain structures that form a gate or interconnection elements, such as wires or fan-outs.

Example 3. *Consider the two grids of clock zones depicted in Fig. 2, where each square shape represents a clock zone in which QCA cells realizing a gate or interconnection may be placed. Each clock zone is related to one of the four external clocks (indicated by the numbers and a corresponding coloring), which control all QCA cells within the respective clock zone. Following the proposal from [24], [26], [27], each clock zone of this example has a size of* 5×5 *QCA cells. Further, the grids are organized such that each clock zone has at least one neighboring zone that can provide data and one that can receive data. The possible data flows between adjacent clock zones are indicated by arrows.*

When placing QCA structures (gates and interconnections) on the grid, the data flow constraints resulting from the particular arrangement of clock zones need to be satisfied. This can become rather complex, as all inputs of a gate have to arrive simultaneously, although they might originate from paths with different lengths. Then, additional *wires* have to be added so that the respective signals are delayed and all arrive at the same time. In order to cope with this complexity, a two-stage design flow for QCA circuits got established in which, first, the desired function is synthesized (using conventional methods such as [28], [29]) and, afterwards, the resulting conventional circuit is explicitly placed on a QCA grid satisfying all constraints (using methods such as [19], [30]). This way, the tedious tasks of synthesis and determining a proper placement are separated.



(a) Netlist represenatation of logic function

(b) Grid of clock zones with arrows indicating allowed data flows

(c) QCA circuit implementing function $f(a,b,s) = \overline{a \cdot \bar{s} + b \cdot s}$

Fig. 3: QCA circuit design

Example 4. *Consider the logic function with three variables* $f(a,b,s) = \overline{a \cdot \bar{s} + b \cdot s}$ *which shall be realized as a QCA circuit. To this end,* f *is decomposed into gates for which QCA realizations are available (see Fig. 3a). Next, all gates have to be located on a grid of clock zones (see Fig. 3b) such that all inputs of a gate come from structures located in a preceding clock zone. Figure 3c shows a possible solution, in which this is ensured for all gates.*

III. Cost Model for Logic Synthesis

The design flow reviewed above allows for an efficient and scalable realization of QCA circuits but suffers from the fact that the first step (the actual synthesis) still relies on conventional methods employing conventional cost metrics. In order to address that, and to explicitly consider QCA-specific cost metrics such as area, delay and energy dissipation, a dedicated cost model for logic synthesis of QCA is required. A corresponding model has recently been proposed in [20] and, for sake of completeness, is reviewed in the following.

A. Area, Delay and Energy Dissipation

In QCA, the *area* can be defined via the number of required QCA cells, the number of occupied clock zones, or via the area of the grid created by the circuit. For example, the circuit depicted in Fig. 3c consists of 55 QCA cells, occupies 7 clock zones and creates a grid of 8 clock zones, with each clock zone having a size of 5×5 QCA cells.

The *delay* of a QCA circuit is defined by the maximum amount of time a signal requires to pass from an input to an output. It should be noted that in QCA the delay is independent of the input signals, because there is no difference in the timing behavior for polarizations $P = -1$ and $P = +1$ [1]. Further, input slopes and output load, as considered in CMOS technologies, can be ignored for QCA. This follows from the fact that the polarizations of all cells in a clock zone are stabilized during a clock phase [1]. Thus, the delay follows from the frequency of the external clock and the number of traversed clock zones between the input and the output. For example, the delay between input *a* and output *f* in the QCA structure shown in Fig. 3c is 3 clock zones, while the delay between input *s* and output *f* is 5 clock zones, which leads to an overall delay of 5 for the whole circuit.

Finally, the *energy* dissipation of a circuit is computed by summation of the energy dissipation of all cells. For an individual QCA cell, this energy transfer between the cell and the environment (E_{env}) can be determined by

$$\begin{aligned} E_{env} &= \frac{\hbar}{2} \int \left(\frac{d}{dt} \vec{\lambda} \cdot \vec{\Gamma} \right) dt' \\ &= -\frac{\hbar}{2\tau} \int \left[\left(\vec{\Gamma} \cdot \vec{\lambda} + |\vec{\Gamma}| \tanh \eta_{th} \right) \right] dt', \end{aligned} \tag{1}$$

where $\hbar$ is the reduced Planck constant, $\vec{\lambda}$ denotes the so-called *coherence vector* and represents the current state of the cell, τ is a technology-dependent relaxation time parameter, and $\eta_{th} = \hbar |\vec{\Gamma}| \cdot (2 k_B T)^{-1}$ refers to the thermal ratio, with k_B being the Boltzmann constant and T denoting the temperature. Further, $\vec{\Gamma}$ means the energy vector that is related to the cell's *steady-state*, a virtual state that characterizes the future behavior of the cell and depends on the current tunneling behavior γ as well as the Coulombic force induced by neighboring cells [4], [20], [31] and follows from

$$\vec{\Gamma} = \frac{1}{\hbar} \left[-2\gamma, 0, \Phi \right], \tag{2}$$

where $\Phi = \sum_{j \in N(i)} E_{kink}^{i,j} P_j$ models the Coulombic interactions with cells from the neighborhood $N(i)$ of the cell i with P_j being the other cells' polarization and $E_{kink}^{i,j}$ denoting the so-called *kink energy* between two cells i and j which quantifies the energy cost of both cells having opposite polarizations [4], [20], [31].

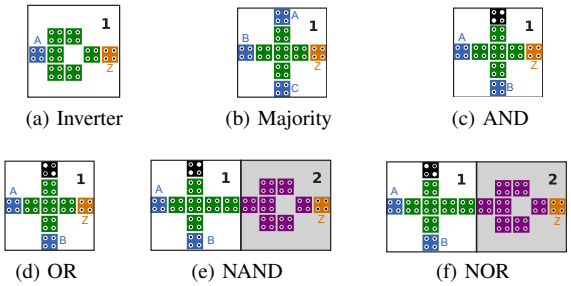
Using a simulation program like *QCADesigner-E* [20], [32] [1], which implements the quantum state QCA model, one can determine the cells' coherence vectors and steady states and, by this, the energy dissipation of all structures of a QCA circuit. For example, for the input case $a = b = s = 0$ the energy dissipation of the QCA circuit depicted in Fig. 3c results to 1.56 meV for a clock frequency of 25 GHz and the standard parameters of *QCADesigner-E*.

B. Resulting Cost Model

Based on the consideration from above, a cost model for area, delay and energy dissipation of QCA circuits which can be used during the first step of the QCA design flow (i. e. the actual synthesis) has been introduced in [20]. This model assumes the following established design paradigms:

1) The elementary logic building blocks considered for synthesis are given by the standard gates shown in Fig. 4.

[1] The tool has been made publicly available as open-source at https://github.com/FSillT/QCADesigner-E.



(a) Inverter (b) Majority (c) AND

(d) OR (e) NAND (f) NOR

Fig. 4: Layout of standard gates (adapted from [33])

2) Interconnection elements as wires, fan-outs, and crossovers are modeled in the same way as the logic blocks.
3) A tile-based clocking scheme [24]–[27] is employed.
4) Varying the frequency of the external clock can enable faster and slower designs which differ in their energy dissipation.

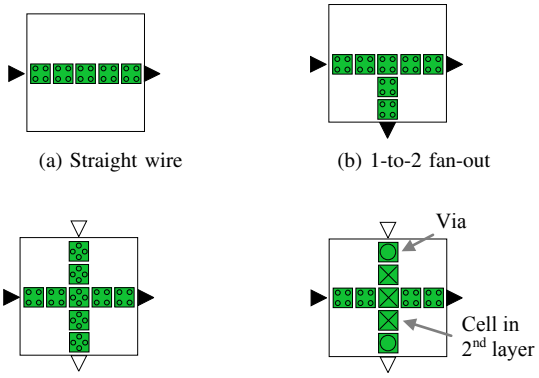
The resulting model is summarized in Table I. The model distinguishes between interconnection elements and logic gates. Further, the column *Area* indicates the area in μm^2 occupied by each element, while the column *Delay* refers to the path length between each input and output in terms of clock zones. The following columns *Energy Dissipation* list the energy dissipation of each QCA element for all possible input combinations, depending on the number of inputs. Here, energy dissipation has been evaluated for a *Regular mode* with a main clock frequency of $f_{clk} = 25\ GHz$ and a *Fast mode* with $f_{clk} = 100\ GHz$. The time value of the delay follows directly from the product of the path length and the clock period of the chosen operation mode.

IV. Interconnections in QCA Circuits

Interconnections play a vital role in QCA circuits. On the one hand, they have to establish the connections amongst the logic elements, while they also must assure the correct timing of data flows. These tasks are more difficult due to the fact that QCA is mainly a planar technology, i. e. most of the interconnection structures are fabricated in the same layer as the actual logic. In general, one can distinguish three principle structures, namely

- *Wires*, i. e. straight-forward or bent connections between two QCA cells,
- *1-to-n Fan-outs*, i. e. structures that copy one input to *n* outputs, and
- *Crossovers*, i. e. crossings of two independent wires (which can be planar or within a multi-layer structure).

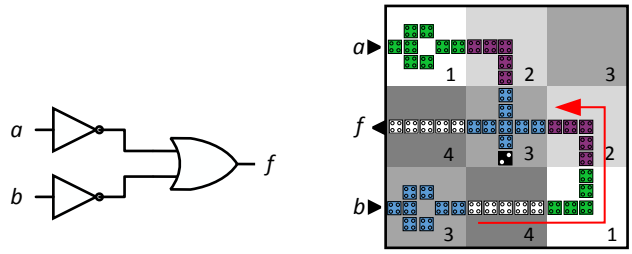
Example 5. *Figure 5a shows an example for a straight wire. An exemplary 1-to-2 fan-out, i. e. a fan-out with one input and two outputs, is shown in Fig. 5b. Figure 5c shows a planar crossover where one signal is transported in the direction of the rotated cells, while the second signal is routed in direction of the non-rotated cells. Figure 5d depicts a multi-layer crossover, where* via *cells copy the signal to a second layer located above the main layer. Note that cells in both layers are controlled by the same clock and that in the very center there is also a regular cell in the main layer (hidden by the via cell on top of it).*



(a) Straight wire (b) 1-to-2 fan-out

(c) Planar crossover (d) Multilayer crossover

Fig. 5: QCA Interconnection elements. Same colored arrows indicate related input and output signals.



(a) Simple circuit to be placed (b) Final implementation with red arrow indicating additional interconnections in order to comply with data flow requirement.

Fig. 6: Synchronization via additional wires

Besides connecting logic elements, interconnections are also employed to satisfy the data flow constraints discussed in Section II.

Example 6. *Consider the exemplary circuit in Fig. 6a which shall be placed on a grid of QCA clock zones. Following the data flow constraints, the OR gate in the center of Fig. 6b (clock zone 3) can only receive its inputs from QCA structures in both neighboring clock zones 2. Consequently, additional wires are required, indicated by the red arrow, in order to pass the output signal of the lower inverter to the input of the OR gate.*

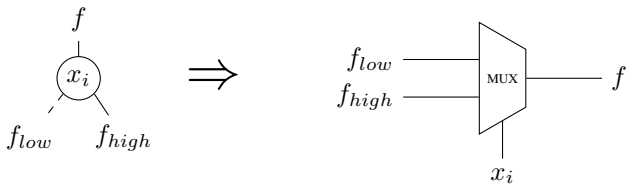
Hence, interconnection elements are realized by the very same basic QCA cells as the elementary logic gates. Accordingly, they have a significant impact on area and delay costs of at least one clock zone. Additionally, their energy costs are comparable to an inverter gate (see also Table I). This is in strong contrast to conventional logic synthesis (where the effects e. g. of wires, fan-outs, etc. with respect to area, depth, and energy dissipation are usually neglected). However, the significance of this impact has not been thoroughly investigated yet. Because of that, it remains unknown how substantial the impact is and whether this indeed requires a dedicated consideration of interconnections already in the logic synthesis phase.

In this work, we conducted such a detailed evaluation. In the following, we review the corresponding environment which has been used for those evaluations. Afterwards, the obtained results are summarized and discussed.

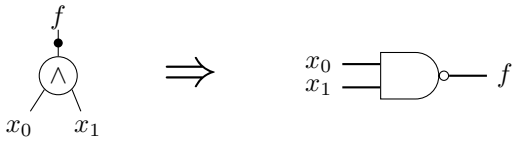
TABLE I: Model for the logic synthesis of Quantum-Dot Cellular Automata[a] (proposed in [20])

		Area [μm^2][c]	Delay[d] [clk zones]			Energy Dissipation [meV]													
						Regular mode (f_{clk} = 25 GHz) with respect to the following input assignments								Fast mode (f_{clk} = 100 GHz) with respect to the following input assignments							
			$A \to Z$	$B \to Z$	$C \to Z$	000	001	010	011	100	101	110	111	000	001	010	011	100	101	110	111
Interconn. Element	Wire	0.01	1			0.09	0.09							0.82	0.82						
	Wire pair	0.01	1	1		0.17	0.17	0.17	0. 17					1.60	1.61	1.61	1.60				
	Fan-out	0.01	1			0.12	0.12							1.15	1.15						
	Crossover	0.01	1	1		0.28	0.28	0.28	0.28					2.57	2.58	2.58	2.57				
Logic Gate[b]	Inverter	0.01	1			0.13	0.13							1.19	1.19						
	Majority	0.01	1	1	1	0.15	0.82	0.82	0.82	0.82	0.82	0.82	0.15	1.41	1.77	1.78	1.77	1.77	1.78	1.77	1.41
	OR	0.01	1	1		0.18	0.79	0.79	0.12					1.30	1.52	1.54	1.19				
	AND	0.01	1	1		0.12	0.79	0.79	0.18					1.19	1.54	1.53	1.30				
	NOR	0.02	2	2		0.31	0.92	0.92	0.25					2.49	2.72	2.73	2.38				
	NAND	0.02	2	2		0.25	0.92	0.92	0.31					2.38	2.73	2.72	2.49				

[a] Technology parameters taken from [20], simulation parameters are: γ_{shape} = GAUSSIAN, T_{step} = 1E-17 s, γ_{slope} = 1E-12 s for f_{clk} = 25 GHz, and γ_{slope} = 1E-13 s for f_{clk} = 100 GHz.
[b] Related layouts are depicted in Fig. 4.
[c] Each tile contains 5x5 QCA cells, with each cell having the size of 400 nm^2 see also [24], [25].
[d] Delay is measured in numbers of clock zones a signal must pass from input to output cell.



(a) BDD-based synthesis



(b) AIG-based synthesis

Fig. 7: Illustration of synthesis approaches

V. Evaluation Environment

In this section, we briefly summarize the synthesis and placement methods that have been considered for the evaluations. Further, we briefly review the chosen benchmarks.

A. Synthesis Methods

The typical input for synthesis approaches is a Boolean function representation such as two-level representations (i. e. *Sum of Products* (SoPs) and *Exclusive Sum of Products* (ESoPs)) or graphical representations like *Binary Decision Diagrams* (BDDs) [28], *AND-Inverter Graphs* (AIGs) [29], etc. The corresponding function representation is then mapped to a netlist of gates using the gate library available in the considered technology. To this end, a one-to-one relation between the function representation and the considered gate library is desired, as it implies that a simplified/reduced representation will allow for a cheaper circuit realization. For example, a node of a BDD corresponds to a MUX gate (as illustrated in Fig. 7a), while a node of an AIG directly corresponds to a (N)AND gate (as illustrated in Fig. 7b). These relations give rise to corresponding synthesis approaches (termed BDD-based and AIG-based synthesis) that will be considered in our evaluations.

To this end, we utilized the well-established academic synthesis tool ABC [34]. The tool is able (a) to read in a Boolean function specification from many different formats (e. g. BLIF, PLA, Verilog), (b) to convert between different internal representations (e. g. AIG to BDD and vice versa), and (c) to perform optimizations (e. g. variable reordering for BDDs and rewriting techniques for AIGs).

Further, we applied a commercially available synthesis software.[2] As input, we used verilog netlists. The target library is based on the model for logic synthesis listed in Table I.

B. Placement Method

Area-efficient placement of conventional circuits to QCA structures turned out to be a complex task. So far, to the best of our knowledge, no fully automated solution exists that can handle functions of relevant size and produces QCA circuits with satisfying area costs. This discrepancy between CMOS and QCA design is due to different physical and logical constraints for which one must come up with new solutions. Classical CMOS approaches simply are not applicable in the QCA domain.

More precisely, clock zone constraints as well as data flow constraint (as discussed in Examples 3 and 4) need to be taken into account. To satisfy these, we developed a heuristic placement algorithm that assumes the fixed clocking scheme shown in Fig. 2b and diagonally places a given netlist starting with the inputs in the upper left and finishing with the outputs in the lower right. All gates are placed in topological order and wires are Manhattan-routed [35]. Even though the resulting area needs are not optimal, this algorithm can handle large netlists due to its linear runtime complexity.

C. Benchmark Circuits

As benchmarks, we applied functions provided by the *EPFL Combinational Benchmark Suite* [36]. This is a recently developed set of natively combinational circuits designed for aiding the comparison of modern logic optimization approaches. Table II lists the related circuits that have been selected for the purpose of this evaluation together with their respective number of primary inputs and primary outputs as well as the number of AND nodes in the provided AIG representation.

[2] In alignment with the related NDA, we may not provide the producer's name here.

TABLE II: Applied EPFL Arithmetic and Random/Control Benchmarks (taken from [36])

Benchmark name	Inputs	Outputs	AND nodes
Adder (adder)	256	129	1 020
Barrel shifter (bar)	135	128	3 336
Max (max)	512	130	2 865
Sine (sin)	24	25	5 416
Alu control unit (ctrl)	7	26	174
Coding-cavlc (cavlc)	10	11	693
Decoder (dec)	8	256	304
i2c controller (i2c)	147	142	1 342
Int to float converter (int2float)	11	7	260
Priority encoder (priority)	128	8	978
Lookahead XY router (router)	60	30	257

VI. Results and Discussions

This section presents and discusses the obtained results. All benchmark circuits have been synthesized using the three synthesis approaches discussed in Section V-A, i. e. BDD-based synthesis, AIG-based synthesis, as well as the commercial synthesis tool (denoted as BDD, AIG, and Comm, respectively). From the obtained netlists, a layout has been generated using the placement approach discussed in Section V-B.

Table III lists all obtained results for both the intermediate netlists (post-synthesis), which has been generated by the synthesis approaches, as well as the final layout (post-layout). The columns *#Gates* and *#Elements* list the number of logic gates after synthesis and the number of all QCA elements, i. e. logic gates *and* interconnection elements, after layout generation. The remaining columns quantify the physical cost of the corresponding designs based on the cost model listed in Table I. More precisely,

- The columns *Gate area* and *Design area* provide a direct translation of the number of gates and elements into area using the model presented in Table I.
- The *Delay* of the designs, i. e. the longest path within the netlist after synthesis and the maximum delay between inputs and outputs after layout generation, respectively, is provided considering a fast operation mode, i. e. $f_{clk} = 100\ GHz$ [20].
- The columns *Energy (regular)* and *Energy (fast)* list the energy dissipation in meV for regular and fast operation mode, i. e. $f_{clk} = 25\ GHz$ and $f_{clk} = 100\ GHz$, respectively.

The numbers clearly indicate that: *Interconnections have a very high impact on the area, delay, and energy costs of QCA circuits and must not be ignored.* This can easily be explained by the fact that, once interconnections are considered, they have to be realized as explicit elements. This drastically increases the number of elements compared to the number of gates which are the only elements to be considered when interconnections are ignored (cf. columns *#Gates* vs. *#Elements*).

Figure 8 exemplarily visualizes this finding on a precise example. Here, the layout of a 2-bit Ripple-Carry-Adder is shown which is composed of 19 logic gates (highlighted green) and plenty of interconnections (highlighted purple).[3] As can be clearly seen the interconnections dominate the overall design. After all, this dominance explains the huge differences in both columns *Gate Area* and *Design Area* as well as in the energy costs reported in Table III. When interconnections are not considered, the huge purple parts of the circuits are counted to have zero costs (as common in conventional circuits). But

[3]For visualization purposes, we removed all clock zone information.

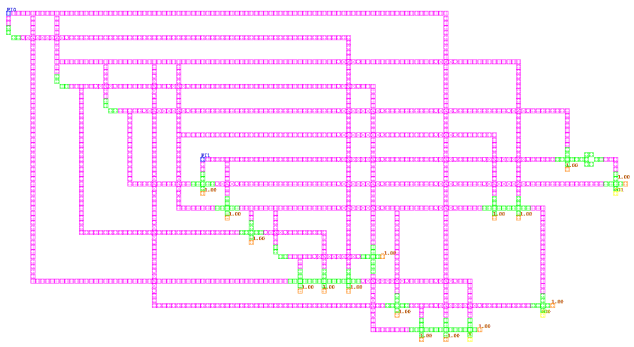


Fig. 8: QCA circuit with highlighted interconnections (purple) and logic gates (green). Blue, yellow and orange cells indicate inputs, outputs and cells with fixed polarization, respectively.

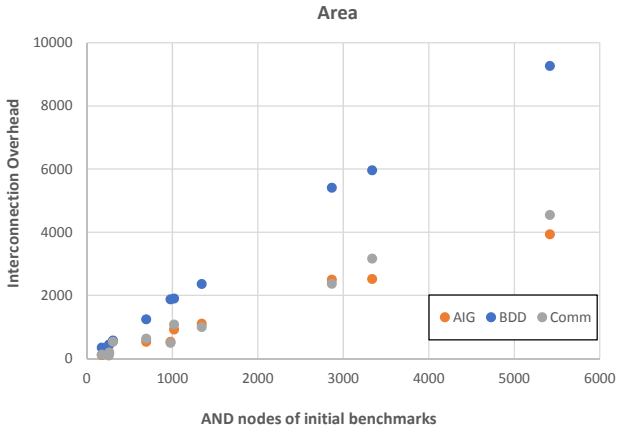


Fig. 9: Number of initial AND nodes vs. increase in area due to interconnections

since they actually are realized by explicit circuit elements (which is in contrast to conventional circuits), they actually have to be considered—suddenly leading to costs which are magnitudes larger than originally approximated. Again, it can clearly be seen that interconnections have a severe impact on QCA circuits with respect to costs.

Figure 9 relates the increase in area (due to the consideration of interconnections) to the number of AND nodes of the initial benchmarks (see also Table II) for all three algorithms. Here, *Interconnection Overhead* means the relation between the gate area, taken from column *Gate area* in Tab. III, and the area for all elements, including interconnections and gates, taken from column *Design area*. First of all, one can observe the remarkable overhead due to interconnections ranging up to a factor of 9200. Further, the results indicate a nearly linear relation between the number of initial AND nodes and the interconnection overhead. This relation results from the planar characteristic of QCA, which enables solely two dimensional routing or crossovers. Consequently, routing between logic gates follows basically the Manhattan distance, i. e. the sum of the horizontal and vertical components [35]. That means, the interconnection overhead between two logic gates located in different rows and columns of the QCA grid results from the vertical *and* the horizontal distance between both gates. Finally, one can note that the BDD-based synthesis strategy leads to the highest overhead, while the overhead of AIG-based synthesis and the commercial tool is up to ten times lower.

TABLE III: Results after synthesis and layout generation

		Post-Synthesis (note: no interconnections)					Post-Layout (note: with interconnections)				
Benchmark	**Algorithm**	**#Gates**	**Gate area [μm^2]**	**Delay [ps]**	**Energy (reg.) [meV]**	**Energy (fast) [meV]**	**#Elements**	**Design area [μm^2]**	**Delay [ps]**	**Energy (reg) [meV]**	**Energy (fast) [meV]**
adder	AIG	1 911	19	1 913	591	2 433	365 918	3 659	7 643	38 144	344 952
adder	BDD	3 949	39	3 190	1 137	4 770	1 429 871	14 299	14 973	148 835	1 351 870
adder	Comm	1 295	13	1 288	520	1 746	296 649	2 966	6 138	32 455	293 119
bar	AIG	5 326	53	1 013	1 756	6 919	2 376 342	23 763	21 480	364 581	3 322 910
bar	BDD	11 954	120	3 168	3 483	14 501	17 376 521	173 765	44 958	2 170 010	19 795 000
bar	Comm	3 742	37	380	1 756	5 198	3 076 894	30 769	19 000	519 106	4 735 280
cavlc	AIG	1 213	12	203	381	1 555	318 837	3 188	4 710	39 018	354 295
cavlc	BDD	2 667	27	630	763	3 219	1 268 135	12 681	9 883	151 900	1 382 830
cavlc	Comm	761	8	188	339	1 039	250 818	2 508	3 683	35 106	318 675
ctrl	AIG	182	2	78	61	234	10 644	106	788	1 352	12 025
ctrl	BDD	679	7	270	197	819	103 256	1 033	2 593	13 605	123 277
ctrl	Comm	132	1	80	57	178	8 536	85	678	1 166	10 316
dec	AIG	312	3	63	144	432	92 843	928	2 175	13 291	120 573
dec	BDD	636	6	80	188	795	146 123	1 461	3 015	18 297	166 158
dec	Comm	344	3	65	148	468	95 946	959	2 255	13 630	123 664
i2c	AIG	1 965	20	203	644	2 499	578 008	5 780	8 080	69 628	632 234
i2c	BDD	4 758	48	578	1 356	5 679	2 660 967	26 610	17 885	300 861	2 738 700
i2c	Comm	1 122	11	165	479	1 504	404 707	4 047	5 453	48 215	437 256
int2float	AIG	385	4	110	123	491	26 847	268	1 513	3 451	30 888
int2float	BDD	951	10	200	270	1 139	124 483	1 245	3 500	15 220	137 637
int2float	Comm	237	2	93	101	319	15 205	152	1 110	2 158	19 118
max	AIG	5 586	56	1 495	1 676	7 055	3 236 644	32 366	21 365	365 842	3 330 090
max	BDD	11 348	113	3 520	3 212	13 581	14 456 505	144 565	42 090	1 549 530	14 119 100
max	Comm	3 190	32	2 425	1 352	4 340	2 101 463	21 015	15 190	277 056	2 522 300
priority	AIG	1 308	13	1 160	384	1 635	255 094	2 551	4 885	25 547	231 078
priority	BDD	4 153	42	2 188	1 173	5 048	2 548 184	25 482	15 223	262 245	2 385 970
priority	Comm	646	6	280	259	840	80 423	804	2 930	8 238	73 612
router	AIG	301	3	113	96	374	12 452	125	1 233	1 369	11 986
router	BDD	938	9	495	262	1 135	62 289	623	3 425	6 552	58 510
router	Comm	119	1	58	54	164	3 263	33	553	393	3 262
sin	AIG	8 387	84	1 260	2 797	10 901	6 529 236	65 292	33 630	718 985	6 545 170
sin	BDD	21 176	212	2 485	5 917	25 626	32 004 305	320 043	77 218	3 444 650	31 394 600
sin	Comm	6 644	66	1 168	2 775	8 947	8 990 800	89 908	30 803	1 012 260	9 220 370

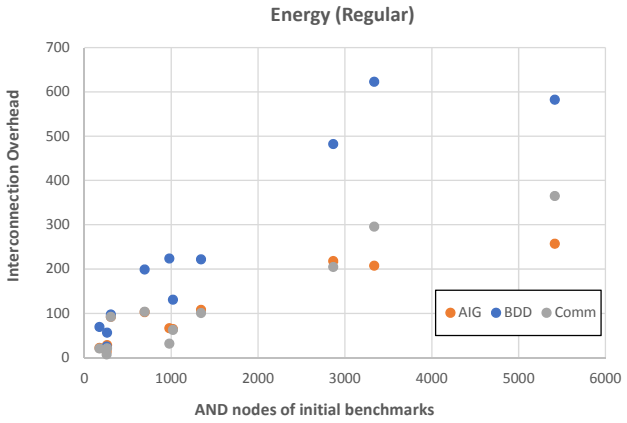


Fig. 10: Number of initial AND nodes vs. increase in energy dissipation due to interconnections (low performance mode)

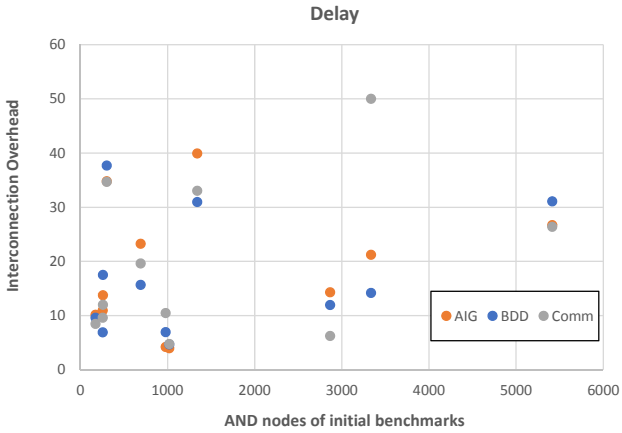


Fig. 11: Number of initial AND nodes vs. increase in delay due to interconnections

Figure 10 demonstrates the increase in energy dissipation due to the interconnection overhead, i. e. the relation between the values of both columns *Energy (reg.)* in Tab. III. We report here solely the results for the regular performance mode which are, though, very similar to the results of the high performance mode. Similar to the data related to the increase in area, the overhead factor increases with the number of initial AND nodes. However, the absolute values of the overhead factors are smaller, caused by the lower energy costs of the wires (see also the cost model in Table I). As observed before, synthesis based on AIGs and the commercial tool yield notably better results than the BDD-based synthesis.

The results for area and energy enable the conclusion that: *The impact of interconnections on the QCA design costs in terms of area and energy increases with the number of logic elements.* Further, one can say that: *The choice of the synthesis strategies can have considerable impact on the design costs.*

Figure 11 shows the relation between the delay of the synthesized netlists and the delay of the final layout. Results indicate a remarkably lower overhead compared to the increase of area and energy costs, but still at high scale. Further, in contrast to the interconnection related area overhead, the impact of the interconnections on the delay is less dependent on the number of AND nodes in the initial benchmark circuits. This results form the fact that the delay depends on a specific path which does not necessarily scale directly with the layout. The impact of all three algorithms on the increase in delay is comparable, although, BDD-based synthesis again leads to the largest overhead.

These results enable the final observation that: *The delay due to interconnections depends on the internal structure of the circuit.*

VII. Conclusion

In this work, we evaluated the impact of interconnections in QCA circuits. This is important as it heavily influences what future design methods for QCA circuits should be considered. In fact, while interconnections are usually ignored in the design for conventional circuits, we observed that they heavily affect the result with respect to area, delay, and energy dissipation. These observations motivate several objectives for future research, including the following statements:

- There is a need for a comprehensive model in order to determine interconnection costs already during synthesis.
- Having an appropriate interconnection model, new synthesis strategies for QCA designs with emphasis on reduction of interconnections must be developed.
- Logic gates and interconnections should be treated similarly. This includes the intelligent application of copying strategies of logic blocks, e. g. as proposed in [37].
- Different concepts, e. g. systolic arrays [38], must be explored.

Acknowledgments

This work was supported in part by the University of Bremen's graduate school SyDe, funded by the German Excellence Initiative.

References

[1] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proceedings of the IEEE*, vol. 85, no. 4, pp. 541–557, 1997.

[2] P. D. Tougaw and C. S. Lent, "Logical devices implemented using quantum cellular automata," *J. Appl. Phys.*, vol. 75, no. 3, pp. 1818–1825, 1994.

[3] N. G. Anderson and S. Bhanja, *Field-coupled Nanocomputing: Paradigms, Progress, and Perspectives*, 1st ed. New York: Springer, 2014.

[4] J. Timler and C. S. Lent, "Power gain and dissipation in quantum-dot cellular automata," *J. Appl. Phys.*, vol. 91, no. 2, pp. 823–831, 2002.

[5] J. Pitters, L. Livadaru, M. B. Haider, and R. A. Wolkow, "Tunnel coupled dangling bond structures on hydrogen terminated silicon surfaces," *JCP*, vol. 134, no. 6, 2011.

[6] V. Arima, M. Iurlo, *et al.*, "Toward quantum-dot cellular automata units: Thiolated-carbazole linked bisferrocenes," *Nanoscale*, vol. 4, pp. 813–823, 2012.

[7] I. Eichwald, A. Bartel, J. Kiermaier, S. Breitkreutz, G. Csaba, D. Schmitt-Landsiedel, and M. Becherer, "Nanomagnetic logic: Error-free, directed signal transmission by an inverter chain," *IEEE Trans. Magn.*, vol. 48, no. 11, pp. 4332–4335, 2012.

[8] R. A. Wolkow, L. Livadaru, *et al.*, *Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics*. Springer-Verlag, 2014, p. 33–58.

[9] T. R. Huff, H. Labidi, *et al.*, "Atomic white-out: Enabling atomic circuitry through mechanically induced bonding of single hydrogen atoms to a silicon surface," *ACS Nano*, vol. 11, no. 9, pp. 8636–8642, 2017.

[10] S. Perri and P. Corsonello, "New methodology for the design of efficient binary addition circuits in QCA," *TNANO*, vol. 11, no. 6, pp. 1192–1200, 2012.

[11] E. Fazzion, O. L. Fonseca, J. A. M. Nacif, O. P. V. Neto, A. O. Fernandes, and D. S. Silva, "A quantum-dot cellular automata processor design," in *SBCCI*, 2014.

[12] M. Kianpour and R. Sabbaghi-Nadooshan, "A novel quantum-dot cellular automata CLB of FPGA," *J. Comput. Electron.*, vol. 13, no. 3, pp. 709–725, 2014.

[13] M. Momenzadeh, J. Huang, M. B. Tahoori, and F. Lombardi, "Characterization, test, and logic synthesis of and-or-inverter (aoi) gate design for qca implementation," *TCAD*, vol. 24, no. 12, pp. 1881–1893, Dec 2005.

[14] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," *TNANO*, vol. 3, no. 4, pp. 443–450, Dec 2004.

[15] K. Kong, Y. Shang, and R. Lu, "An optimized majority logic synthesis methodology for quantum-dot cellular automata," *TNANO*, vol. 9, no. 2, pp. 170–183, 2010.

[16] M. G. A. Martins, V. Callegaro, F. S. Marranghello, R. P. Ribas, and A. I. Reis, "Majority-based logic synthesis for nanometric technologies," in *IEEE-NANO*, 2014, pp. 256–261.

[17] P. Wang, M. Niamat, and S. Vemuru, *Majority Logic Synthesis Based on Nauty Algorithm*. Berlin, Heidelberg: Springer, 2014, pp. 111–132.

[18] R. K. Nath, B. Sen, and B. K. Sikdar, "Optimal synthesis of qca logic circuit eliminating wire-crossings," *IET-CDS*, vol. 11, no. 3, pp. 201–208, 2017.

[19] M. Walter, R. Wille, D. Große, F. S. Torres, and R. Drechsler, "An Exact Method for Design Exploration of Quantum-dot Cellular Automata," in *DATE*, 2018, pp. 509–514.

[20] F. S. Torres, R. Wille, P. Niemann, and R. Drechsler, "An energy-aware model for the logic synthesis of quantum-dot cellular automata," *TCAD*, 2018.

[21] W. Liu, E. E. Swartzlander Jr, and M. O'Neill, *Design of semiconductor QCA systems*. Artech House, 2013.

[22] M. Taucer, F. Karim, K. Walus, and R. A. Wolkow, "Consequences of many-cell correlations in clocked quantum-dot cellular automata," *TNANO*, vol. 14, no. 4, pp. 638–647, 2015.

[23] K. Hennessy and C. S. Lent, "Clocking of molecular quantum-dot cellular automata," *J. Vac. Sci. Technol. B*, vol. 19, no. 5, pp. 1752–1755, 2001.

[24] C. A. T. Campos, A. L. P. Marciano, O. P. V. Neto, and F. S. Torres, "USE: A universal, scalable, and efficient clocking scheme for QCA," *TCAD*, vol. 35, no. 3, pp. 513–517, 2016.

[25] J. Huang, M. Momenzadeh, L. Schiano, M. Ottavi, and F. Lombardi, "Tile-based QCA design using majority-like logic primitives," *JETC*, vol. 1, no. 3, pp. 163–185, 2005.

[26] V. Vankamamidi, M. Ottavi, and F. Lombardi, "Two-dimensional schemes for clocking/timing of QCA circuits," *TCAD*, vol. 27, no. 1, pp. 34–44, 2008.

[27] M. Janez, P. Pecar, and M. Mraz, "Layout design of manufacturable quantum-dot cellular automata," *Microelectroics Journal*, vol. 43, no. 7, pp. 501–513, 2012.

[28] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *TC*, vol. C-35, no. 8, pp. 677–691, Aug 1986.

[29] A. Mishchenko, S. Chatterjee, and R. Brayton, "Dag-aware aig rewriting: a fresh look at combinational logic synthesis," in *DAC*, July 2006, pp. 532–535.

[30] A. Trindade, R. S. Ferreira, J. A. M. Nacif, D. Sales, and O. P. V. Neto, "A placement and routing algorithm for quantum-dot cellular automata," in *SBCCI*, 2016.

[31] S. Srivastava, S. Sarkar, and S. Bhanja, "Estimation of upper bound of power dissipation in qca circuits," *TNANO*, vol. 8, no. 1, pp. 116–127, Jan 2009.

[32] K. Walus and G. A. Jullien, "Design tools for an emerging SoC technology: Quantum-dot cellular automata," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1225–1244, 2006.

[33] D. A. Reis, C. A. T. Campos, T. R. Soares, O. P. V. Neto, and F. S. Torres, "A methodology for standard cell design for QCA," in *ISCAS*, 2016.

[34] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *CAV*, 2010, pp. 24–40.

[35] E. F. Krause, *Taxicab Geometry*. Dover, 1987.

[36] L. Amarù, P.-E. Gaillardon, and G. De Micheli, "The epfl combinational benchmark suite," *Int'l Workshop on Logic Synth.*, 2015.

[37] W. J. Chung, B. Smith, and S. K. Lim, "Node duplication and routing algorithms for quantum-dot cellular automata circuits," *IEEE Proceedings on Circuits, Devices and Systems*, vol. 153, no. 5, pp. 497–505, 2006.

[38] R. P. Brent and H. T. Kung, "Systolic vlsi arrays for polynomial gcd computation," *TC*, vol. C-33, no. 8, pp. 731–736, Aug 1984.