# On the Impact of the Synchronization Constraint and Interconnections in Quantum-dot Cellular Automata

Frank Sill Torres[1] Pedro A. Silva[2] Geraldo Fontes[2] Marcel Walter[3] José Augusto M. Nacif[2]
Ricardo Santos Ferreira[2] Omar Paranaiba Vilela Neto[4] Jeferson F. Chaves[4,5] Robert Wille[6]
Philipp Niemann[3,7] Daniel Große[3,7] Rolf Drechsler [3,7]

[1]Department for Resilience of Maritime Systems, German Aerospace Center, Bremerhaven, Germany
[2]Universidade Federal de Viçosa, Brazil
[3]Group of Computer Architecture, University of Bremen, Germany
[4]Universidade Federal de Minas Gerais, Brazil
[5]Centro Federal de Educação Tecnológica de Minas Gerais, Brazil
[6]Johannes Kepler University Linz, Austria
[7]Cyber-Physical Systems, DFKI GmbH, Bremen, Germany

***Abstract*—Quantum-dot Cellular Automata (QCA) is an emerging nanotechnology with remarkable performance and energy efficiency. Computation and information transfer in QCA are 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 is the transport of information, which is affected by two particularities of the QCA technology. First, information flow in QCA is controlled by external clocks, and second, QCA is a planar technology in which gates, as well as interconnections, are mostly located in the same layer. The former demands proper synchronization already during the circuit design phase, while the latter results in high area costs for interconnections. This work focuses on both constraints and discusses its impact on the implementation of QCA circuits. Further, the concept of local and global synchronicity in QCA circuits is explored. The obtained results indicate that relaxing the global synchronicity constraint can reduce design size by about 70% while the throughput performance declines by similar values. Additionally, it can be shown that the impact of interconnections in QCA, like wires, fan-outs, and crossovers, is indeed substantial. That means, up to 75% of the total area is occupied by interconnections.**

## I. Introduction

The Quantum-dot Cellular Automata (QCA) nanotechnology offers a promising alternative to conventional circuit technologies. In QCA, computations, and data transfer are carried out via local fields between nanoscale devices, the so-called QCA cells, that are arranged in patterned arrays [1]. Further, information is represented in terms of the polarization of the cells. Theoretical and experimental results indicate that QCA-based approaches have the potential to allow for systems with highest processing performance and remarkably low energy dissipation [2], [3]. Consequently, numerous contributions on their physical realization have been made in the past, e. g., molecular Quantum Cellular Automata [4], atomic Quantum Cellular Automata [5] or nanomagnetic logic [6]. Many contributions to the progress of these technologies have been made relatively recently, e. g., [7], [8].

QCA applies external clocks in order to prevent metastability and to control the data flow among logic elements [9]. These clocks modify the state of QCA cells such that a cell is in reset or can change or not its polarization, and thus, its logic value. Commonly, four clocks, numbered from 1 to 4 and phase-shifted by 90 degrees, are applied. For fabrication purposes, cells are usually grouped in a grid of square-shaped clock zones such that all cells within a clock zone are controlled by the same external clock [10], [11]. It is important to note that correct data flow is only possible between cells controlled by consecutively numbered clocks. That means, cells controlled by clock 1 can solely pass their data to cells controlled by clock 2, etc. and, finally, from clock 4 to clock 1.

Consequently, data are passed between cells in a pipeline-like fashion controlled by the external clocks (more details will be given in Section II). This behavior led to the common assumption that QCA circuits must not employ only a local but also global pipeline-like behavior, e. g., in [12], [13]. That means, it is assumed that all signal paths arriving at the same logic gates must have equal length and that all signals must always arrive at the respective logic gates in a synchronized manner. This requirement puts some limitations on the design automation of QCA circuits and demands some design overhead, as will be discussed in Section III.

This design overhead also includes a more extensive use of *interconnections* such as wires, fan-outs and crossovers. In contrast to conventional logic design, these elements should not be considered negligible in QCA, which is also discussed in Section II-D. In fact, there is still less research on the actual impact of interconnections in QCA circuits and whether it should explicitly be considered in the future [3].

The intention of this work is to highlight the possibility to design QCA circuits that do not possess a global pipeline-like behavior and to explore the consequent impact on the interconnections.[1] Therefore, we introduce the basics of Quantum-dot Cellular Automata (Sections II) before we discuss the aspect

[1]Preliminary versions of this work have been published before in [14], [15].

of synchronicity in QCA designs (Section III). Based on the conclusions of this discussion we propose modifications to existing QCA placement and routing algorithms (Section IV), which is followed by a simulative comparison of QCA designs that are fully synchronized or not and an exploration of the actual impact of the interconnections (Section V). Finally, we draw some conclusions (Section VI).

## II. Quantum-dot Cellular Automata

This section introduces the nanotechnology Quantum-dot Cellular Automata (QCA) and discusses basic aspects of QCA circuit design.

### A. QCA states and logic gates

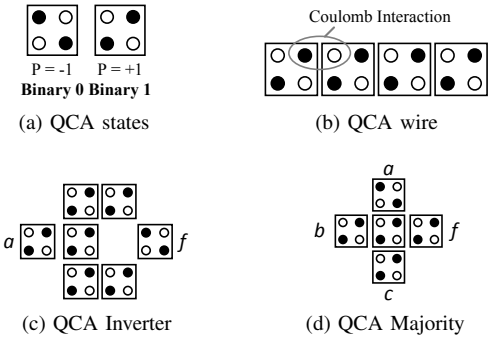
*Quantum-dot Cellular Automata* (QCA) are a field-coupled nanotechnology that executes computations fundamentally different from current technologies. In QCA, information is stored in terms of the polarization of nanosized cells and can be propagated to adjacent cells using Coulomb forces.

The basic element of QCA is a *cell* that is usually composed of four quantum dots which are able to confine an electric charge [16], [17]. These quantum dots are arranged at the corners of a square, such as depicted in Fig. 1a. Further, each cell contains two free and mobile electrons, which are able to tunnel between adjacent dots, while tunneling to the outside of the cell is prevented by a potential barrier. The electrons within a cell experience mutual repulsion due to Coulomb interaction and, thus, tend to locate at opposite corners of the square. Consequently, an isolated cell might assume one of the two *cell polarizations* $P = -1$ and $P = +1$ as depicted in Fig. 1a. This allows for an encoding of binary information by identifying $P = -1$ with a binary 0 and $P = +1$ with a binary 1.

Further, the polarizations of neighboring cells influence each other—again by Coulomb interaction. This allows for the design of wires as well as logic gates. For example, Fig. 1b shows a QCA wire where a signal is propagated through several cells from left to right by Coulomb interaction. Further, Fig. 1c depicts an Inverter gate, where, again from left to right, an input signal is copied to two paths, which are then combined diagonally, such that the input value is inverted. Finally, Fig. 1d shows a *majority gate*, where the output is identically to the majority of the input signals. Further classical logic operations such as AND and OR gates can be easily derived from the majority gate by locking one of its inputs to a binary 0 (leading to an AND) or binary 1 (leading to an OR).

### B. QCA clocking

In order to execute these and more complex logic operations, a dedicated *clocking* is required which, starting with the initialization of the QCA cells, properly propagates information among the cells and avoids metastable states [18]. To this end, external clocks are employed which regulate the intercellular tunneling barriers within a QCA cell such that the cell can be polarized (i. e., tunneling is prevented) or not (i. e., electrons may tunnel between adjacent quantum dots within the cell). Typically, a clock consists of four phases:



(a) QCA states (b) QCA wire (c) QCA Inverter (d) QCA Majority

Fig. 1: QCA states and basic cells

- In the so-called *relax* phase, the cell is depolarized and does not contain any information.
- During the following *switch* phase, the interdot barriers are raised which forces the cell to polarize into one of the two antipodal states (according to the polarization of surrounding cells).
- In the following *hold* phase, the cell keeps its polarization and may act as input for adjacent cells.
- During the final *release* phase, the interdot barriers are lowered again thereby removing the previous polarization of the cell.

Normally, four clocks shifted by 90 degrees are provided in order to enable the propagation of information among cells [9]. Using these clocks, the data flow can be controlled by applying appropriately shifted clock signals such that the cells which shall pass their data are in the *hold* phase at the same time when the cells that shall receive the data are in the *switch* phase. For fabrication purposes, cells are usually grouped in a grid of square-shaped clock zones such that all cells within a clock zone are controlled by the same external clock [10].

Fig. 2 depicts an exemplary QCA wire that has the extension of four clock zones. Moreover, possible locations for further cells are indicated in gray in the most left clock zone. All QCA cells within the same clock zone are controlled by the same clock signal. Note that consecutive clock signals are shifted by one phase. That means, if clock zone 1 is in the *hold* phase then clock zone 2 will be in the *switch* phase, clock zone 3 will be in the *relax* phase and clock zone 4 will be the *release* phase. In this state, cells in clock zone 2 polarize according to the polarization of the adjacent cells in clock zone 1 while cells in clock zone 3 and 4 are without polarization. During the next clock phase, clock zone 2 changes to *hold*, while clock zone 3 is in the *switch* phase. Consequently, data is passed from zone 2 to 3 (and so on).

### C. QCA Circuit Design

In order to design a QCA circuit, traditional design solutions for logic synthesis of conventional circuits can be employed for generation of initial netlists. Therefore, already available realizations of typical gates such as Inverter, OR, AND, XOR, etc. can be applied [19]. During the following placement and routing (P&R), these gates must be arranged such that the

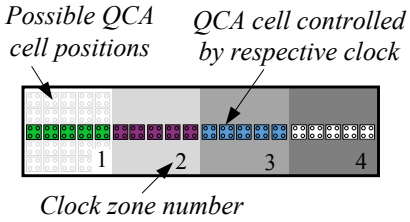
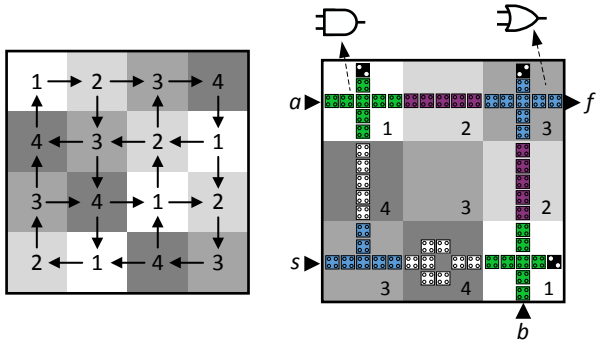



Fig. 2: QCA wire with cells in four clock zones




(a) USE grid with clocking zones (b) Multiplexer mapped to USE grid

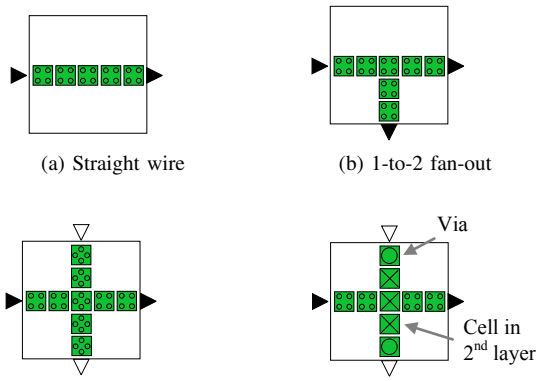
Fig. 3: USE clocking scheme

corresponding clocking is respected, i. e., data is properly passed from one gate to another. To this end, usually a fix arrangement of clock zones is imposed on a QCA layout [11], [20].

There have been several proposals for clocking schemes, of which 2DDWave [20] and USE (*Universal, Scalable and Efficient*) [11] are just two representatives. Without loss of generality, we apply the latter in this work, which is characterized by a highly regular architecture and the ability to utilize feedback paths, which renders USE suitable for design automation of QCA circuits for a variety of netlists.

USE defines a grid of clock zones, which are arranged such that all inner clock zones have two adjacent neighboring clock zones that can provide data and two neighbors that can receive data. The clock zones are numbered from 1 to 4, whereby consecutive numbered zones have clock signals shifted by 90 degree. Fig. 3a depicts the concept of USE (each square is a clock zone that contains $5 \times 5$ QCA cells, following the proposal from [11]). Further, the arrows indicate the possible data flow between adjacent clock zones.

Fig. 3b depicts the *2:1 MUX* function $f = a\bar{s}+bs$ placed on a USE grid. Using conventional synthesis tools, the gate netlist has to be mapped onto the USE grid such that the output of one QCA structure, i. e., gate or wire, is always propagated to the input of a QCA structure containing the next gate or wire. In total, the resulting QCA circuit possesses design costs of $3 \times 3$ clock zones which is equal to the dimension of $15 \times 15$ QCA cells and a critical path of 5 clock zones ($s \rightarrow f$).




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

(c) Planar crossover (d) Multilayer crossover

Fig. 4: QCA Interconnection elements. Same colored arrows indicate related input and output signals.

### D. Interconnections in QCA Circuits

Interconnections establish the connections amongst the logic elements. Its placement is not a trivial task 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 1.** *Fig. 4a 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. 4b. Fig. 4c 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. Fig. 4d 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).*

Hence, interconnections are implemented 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. 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).

## III. Synchronicity of QCA Circuits

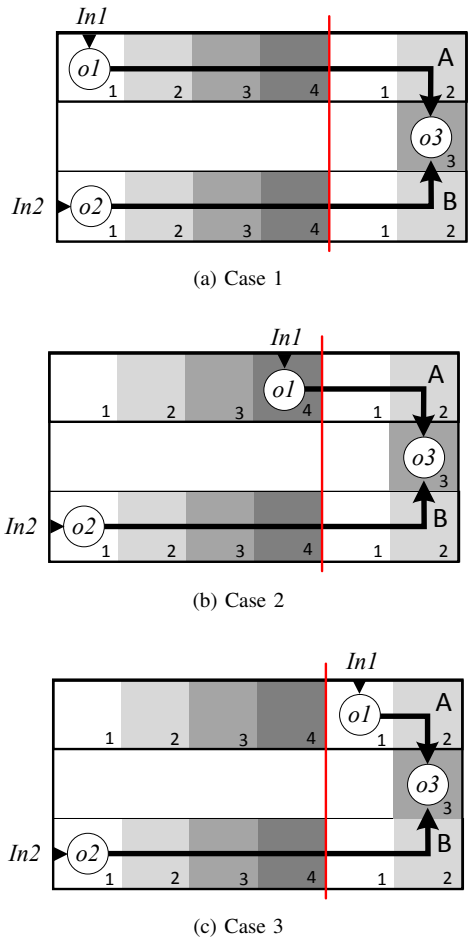
In this section, we discuss the difference between global and local synchronicity in QCA designs. Further, we show that—contrary to the state of the art—global synchronicity is not a mandatory constraint in QCA designs.

### A. Global and Local Synchronicity

When considering synchronicity in QCA circuits, one has to distinguish between local and global synchronicity. The former means the data flow constraint, discussed in Section II, which requires that data can only be transferred between cells located in consecutive numbered clock zones. On the other hand, global synchronicity refers to the global pipeline-like behavior of QCA circuits.

The example depicted in Fig. 5 and Fig. 6 shall highlight the differences. Note that, for the sake of simplicity but without loss of generality, this example does not apply the USE clocking. The circuit shown in Fig. 5 has two primary inputs *In1* and *In2* and three arbitrary operations *o1*, *o2* and *o3*, with the first two having one input while the latter operation possesses two inputs. Each of the three depicted cases differs in the position of input *In1*. The curves in Fig. 6 relate to the clock signals of all four zones, the input signals, that change when clock 1 enters in switch phase (falling clock slope), and the data at points A and B, which both contain inputs of operation *o3*.

In all three cases, local synchronicity is guaranteed. That means all data flow is only between cells in consecutive numbered clock zones. Further, in case 1 global synchronicity is given, i. e., operation *o3* receives related input signals. This is also true for case 2, even though both *In1* are connected with different clock zones. However, the distance between both is less than 4 clock zones (see also the indicated red line). Consequently, in the following clock zone 1 all data from *In1* and *In2* are synchronized again, because all clock zones with number 1 change into *switch* phase at the same time. In contrast, case 3 misses the global synchronicity, because data of input *In1* arrive one clock cycle before the related data of input *In2*. That means, in this case, operation *o3* processes input data that do not belong together.




(a) Case 1

(b) Case 2

(c) Case 3

Fig. 5: QCA circuit possessing and missing the global synchronicity. The red line indicates the limit until where *In1* could be placed such that paths *In1*→*o3* and *In2*→*o3* are synchronous. For the sake of simplicity, this example is not using USE.

### B. Unsynchronized QCA Circuits

A fundamental characteristic of globally synchronized designs is that new data can be applied to the primary inputs of the circuit in each clock cycle. After the first input data passed the circuit, correspondingly new results arrive at the circuit's primary outputs in each clock cycle—resulting in a circuit throughput of 1. Furthermore, a globally synchronized circuit does not require synchronization elements like latches, as, by definition, all related data are always synchronized.

However, in contrast to many related statements in the literature, e. g., in [12], [13], global synchronicity (GS) is not a mandatory constraint in QCA circuits [21]. For example, the circuit depicted in Fig. 5c misses GS, because data from both inputs are not arriving at the same time at operation *o3*. A common solution to this problem would be the relocation of *In1* or *In2* such that paths have equal lengths, as, e. g. in Fig. 5a. However, this usually comes at high costs in terms of area [3]. Instead, we propose to reduce the frequency with which new input data are applied. That means for the given example, data connected at *In1* and *In2* must be kept stable for two clock cycles[2]—leading to a reduced throughput of $1/2$. On the other hand, this approach allows for the reduction of area costs, latency, and design complexity. Consequently, there is a trade-off between performance and area costs.

An important parameter of globally unsynchronized QCA circuits is the frequency with which new input data can be sent to the circuit. This frequency depends on the maximum difference between the arrival times of all input signals of any gate of the QCA circuit. Again, since signal paths starting at different primary inputs but leading to the same gate, can be of different length in the circuit (as demonstrated above), information applied to the primary inputs in the same time step, do not necessarily arrive at that very gate at the same time step. To preserve functionality nonetheless, i.e. to assure

[2]This feature requires additional clocks with clock periods that are multiples of the four basic clocks. It should be noted that adding these clocks can be done with only a low increase in design costs. First, the additional clocks can be generated by the same circuits applied for the generation of the four basic clocks [9], [22]. Second, the distribution of the clocks signals can be done by existing wire routing technologies, as shown in and [20], [23]

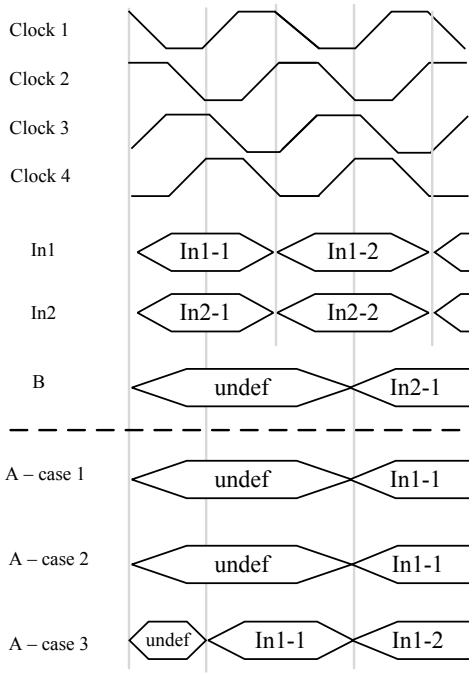
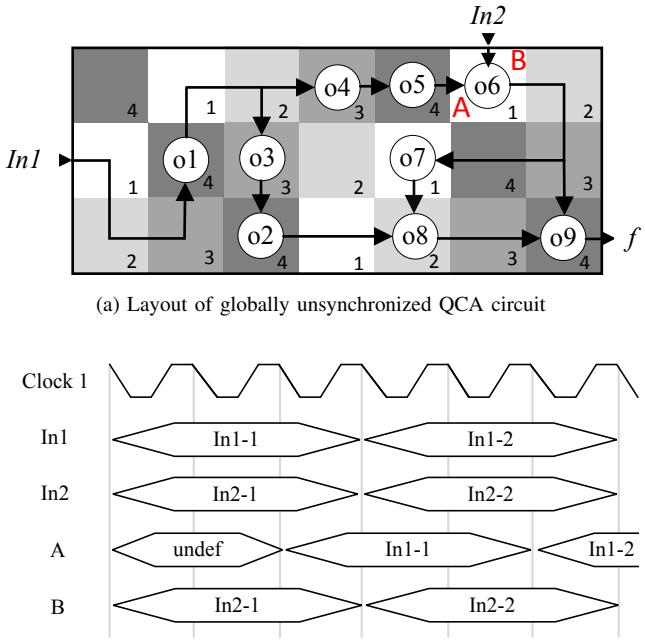



Fig. 6: Curves of clock zone signals, inputs and data at points A and B for all three cases shown in Fig. 5




(a) Layout of globally unsynchronized QCA circuit

(b) Curves of clock 1, inputs and signals at points A and B

Fig. 7: Unsynchronized QCA circuit

that designs are not asynchronous, it must be assured for all gates that their inputs are synchronous for at least one clock cycle before new inputs arrive. As mentioned above, this is done by using additional clocks with clock periods that are multiples of the periods of the four basic clocks.

The following example shall detail the related analysis. Fig. 7a depicts an exemplary QCA circuit that does not possess global synchronicity. Several of the operations *oX* have two inputs that have diverging arrival times. In detail, the inputs of *o6* arrive after 1 and 9 clock phases, the inputs of *o8* after 10 and 14 clock phases, and the inputs of *o9* after 12 and 16 clock phases. As each clock cycle lasts for 4 clock phases, the maximum difference in terms of clock cycles results from the ceiling division by 4. This in case of, e. g., *o6* means $\lceil 9/4 \rceil - \lceil 1/4 \rceil = 2$ is the maximum difference in terms of clock cycles. Analogously, for, e. g., *o8* and *o9* it is 1. Hence, both inputs *In1* and *In2* must not change for two additional clock cycles in order to assure correct operation. Fig. 7b depicts the curves of the clock in clock zones 1, the inputs and the signals at points A and B, both highlighted in Fig. 7a. One can note that only at the third clock cycle, operation *o6* has synchronized inputs, i. e., its both input signals (In1-1 and In2-1) been sent at the same time.

The presented circuit has a latency of 16 clock phases, i. e., 4 clock cycles. If the input frequency is reduced to $1/3$ of the clock frequency then the first correct results will arrive after 6 clock cycles. Next, every three clock cycle new correct outputs will be available.

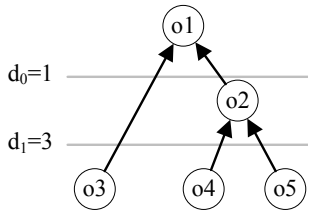
## IV. Modified Place and Route Algorithms

In Electronic Design Automation, the placement and routing (P&R) generates a final layout starting from a gate-level netlist. Here, placement means the location of gates on the grid, while routing refers to the connection of these gates via wires. In this section, we present a heuristic and an exact QCA P&R algorithm, that has been modified such that it is possible to relax the GS constraint.

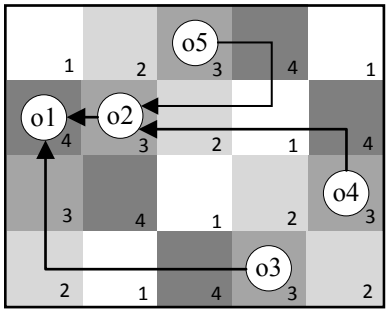
### A. Heuristic Algorithm

In QCA, the P&R is NP-complete leading to high computation costs even for small circuits [24]–[27]. Hence, we propose in [28] a P&R algorithm based on a divide-and-conquer strategy that notably reduces the complexity. The presented P&R algorithm applies the USE clocking scheme but can be easily adapted for other ones too. The approach starts with a decomposition of the gate-level graph that is guided by reconvergent paths. Next, for each partition, the corresponding QCA layout is generated. In the final step, the entire circuit is rebuilt by aligning the nodes that overlap partitions, followed by a routing of all inter-subgraphs wires. In the case of the latter, the algorithm assures that all interconnections between two graph depth levels are locally and globally synchronized.

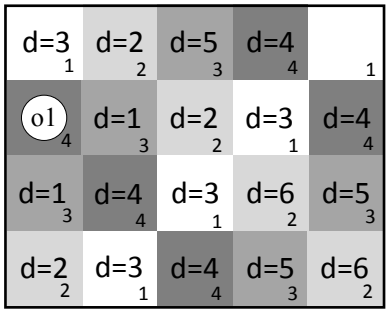
The example in Fig. 8 demonstrates the principal steps of the P&R algorithm presented in [28]. First, the circuit is represented as a graph where each node is a gate (see Fig. 8a). Next, a distance is defined between each level. This is followed by the placement of the nodes; level by level starting from the primary outputs (see Fig. 8b). In the depicted example, the inter-level distance $d_0$ between node $o1$, which connects to the output, and $o2$ is 1. Fig. 8c illustrates the corresponding




(a) Gate-level sub-graph with synchronized inter-level distances

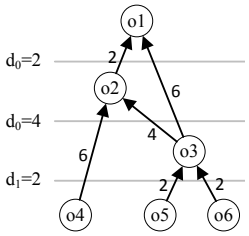



(b) Corresponding layout

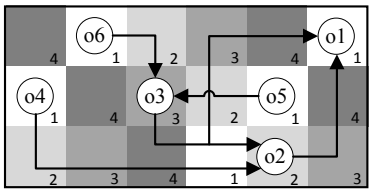



(c) Distances to node $o1$

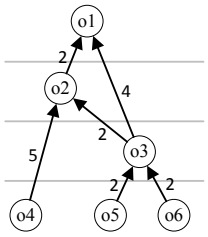
Fig. 8: Example for unmodified P&R algorithm assuring global synchronicity



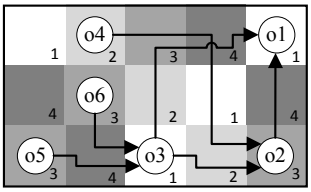

(a) Graph for unmodified P&R algorithm (with GS)




(b) Layout of unmodified P&R algorithm (with GS constraint)




(c) Graph for modified P&R algorithm (no GS)




(d) Layout of modified P&R algorithm (no GS)

Fig. 9: Example comparing of graphs and resulting layouts for P&R assuring and ignored global synchronicity (GS) constraint. The numbers on the edges indicate the distances between nodes.

differences of each clock zone to the clock zone containing node $o1$. As one can see, there are only two possible positions for node $o2$ if the distance to $o1$ shall be 1. In this example, the algorithm chooses the right neighboring clock zone of $o1$. Further, the distance $d_1$ between the nodes in level 1 and level 2 has been defined with 3. Note that for the distance between nodes $o1$ and $o3$ this sums up to 4. Hence, the algorithm tries to place the nodes $o4$ and $o5$ such that both have a distance of 3 to node $o2$, while $o3$ is placed such that it has a distance of 4 to node $o1$. In order to improve the results, the proposed P&R algorithm varies the inter-level distances $d_0, d_1, \ldots, d_{n-1}$ with $1 \leq d_i < max(4, n)$, where $n$ means the graph depth. However, it is assured that the distance between nodes in same level to nodes in higher level is always the same, i. e., global synchronicity is given for all tries.

In order to relax the GS constraint, we modified the algorithm such that each edge of the graph possesses its own distance. This enables that nodes in the same level can have a different distance to nodes in a higher level. Nevertheless, distances must be chosen such that local synchronicity is assured.

The example depicted in Fig. 9 shall highlight these modifications. Fig. 9a and Fig. 9b show the graph and the respective layout if global synchronicity (GS) is assured. The numbers at the edges in Fig. 9a indicate the distance between the nodes. As one can see, these numbers follow the definitions of the inter-level distances, also shown in the figure. Consequently, the distances between nodes $o4$, $o5$ and $o6$ to node $o1$ is always 8. In contrast, Fig. 9c illustrates the same graph, but with new distances. As one can see, the distance between nodes in same level to a node in a different level is not always the same. For example, the distance between $o4$ and $o1$ reduces to 7, and it changes to 6 between $o5$, $o6$ and $o1$. Consequently, the maximum delay, i. e., its latency, of the circuit has been reduced from 8 to 7. Furthermore, the layout can be more compact, as Fig. 9d indicates. Here, the grid layout could be reduced from 18 clock zones to 15, while the number of occupied clock zones reduced from 16 to 13.

### B. Exact Algorithm

In [25], the authors presented an automatic and exact design method for QCA circuits. In contrast to heuristic or manual approaches, this method guarantees the satisfaction of desired design objectives and physical constraints. For this purpose, a reasoning engine (e. g., an SMT solver [29]) is used.

Reasoning engines can be applied to find solutions to decision problems. They use effective heuristics to traverse search spaces in smart ways. Even though they cannot break complexity bounds, in practice, they are often able to come up with solutions to known hard problems. From an application view, a reasoning engine can be seen as a black box that receives problem formulations and then generates valid solutions to these problems or proves that no such solution exists.

Thus, the main challenges are the definition of an adequate formal description of the design constraints and the translation of the obtained assignments to a solution of the original problem instance. In this section, we give an intuitive idea about how we formulated the QCA P&R problem. A detailed description of the quantifier-free first-order logic representation can be found in [25].

When working with decision engines, one first defines the search space which is the set of all *possible solutions* to a problem. The search space is then restricted by constraints until only *valid solutions* remain. As the input to the P&R problem is a netlist and a USE grid, the search space is equal to all mappings from the netlist elements to the grid tiles. Intuitively formulated, every gate and every wire can be placed on any tile. This formulation is then extended by constraints which contain but are not limited to

1) every gate has to be placed exactly once,
2) wires must only be placed between the gates they connect,
3) tiles can be left empty but there can only be one gate or alternatively up to two wires per tile,
4) consecutive gates/wires have to be placed in consecutively numbered clock zones,
5) all elements have to be placed so that the data flow on the grid is equivalent to the input netlist.

In order to satisfy the GS constraint, it must be additionally enforced that all paths leading to any multi-input gate must have the same length on the grid. If on the other hand, a designer is fine with a globally unsynchronized circuit, i. e., with a relaxed GS requirement which leads to lower throughput but presumably smaller circuit area, no additional formal constraint needs to be applied.

## V. Experimental Results

This section compares the design costs for selected benchmark circuits [30]–[32] and circuits generated by the ABC synthesis tool [33] if placed and routed with and without global synchronicity.

The obtained results for the heuristic algorithm presented in Section IV-A are listed in Tab. I, while Tab. II shows the results for the exact algorithm discussed in Section IV-B. Please note that due to time-out constraints only a limited number of benchmarks could be run with the exact algorithm. In detail, the run-times of the heuristic approach were in the range between several seconds to minutes, while the exact approach required for its execution from several minutes to hours. Furthermore, results of both algorithms are not directly comparable, because the heuristic algorithm applies parallel wires within the same clock zone, while the exact one supports solely single wires. This is an intended difference, emphasizing the main motivation to study the impact of synchronicity for different kinds of algorithms.

In both tables, the columns *Gates* refer to the number of elements to place which are not wires. Since, the input to both algorithms is an AIG, the number of gates is equal to the number of AIG nodes plus the number of complemented edges plus the number of fan-outs.[3] *Grid area* indicate the complete area of the generated QCA grid, including empty clock zones, while the columns *Occupied clk-zones* refer solely to the clock zones that contain QCA cells. The columns *Latency* report the length of the longest path in terms of clock zones and the column *Throughput* lists the throughput of the designs without global synchronicity in comparison to its fully synchronized counterpart. Finally, the column *Int.* lists the number of clock zones that contain interconnections, like wires, 1-to-n fan-outs and crossovers.

One can note that disregarding global synchronicity can lead to reduction of occupied clock zones and latency, but not in all cases (e. g., *FA-MAJ* and *B1_r2* for the heuristic algorithm). This reduction, though, comes at the costs of a declining throughput. Further, in most cases the reduction of occupied clock zones and area is comparable, with exception of the benchmark *t* using the heuristic algorithm. Here, the grid area increases while the number of occupied clock zones is reduced. Thus, the qualification of this result depends on the possibility to use the unoccupied area for further circuits, which is, e. g., the case of both presented P&R algorithms.

### A. Evaluation of impact of synchronization constraint

Figs. 10 and 11 compare the reduction of occupied clock zones, latency and throughput if global synchronicity is ignored for the heuristic as well as the exact algorithm. Results indicate that disregarding global synchronicity can reduce the occupied area by up to 67% (*clpl* for the heuristic algorithm) and in average by 33% for the heuristic algorithm and 19% for the exact one. An exception is circuit *b1_r2*, for which on increase of the occupied area can be observed when using the heuristic algorithm and ignoring global synchronicity.

In case of the latency, reductions of up to 25% (*clpl* and *t* for the heuristic algorithm) and in average by 13% for both algorithms can be reported. However, the heuristic algorithm could improve the latency only in about 50% of all benchmarks, while the exact one could enhance the latency in 3 out of 4 cases.

[3]Even though QCA is a Majority-based technology, it is known that MIGs are less suitable for placement and routing than AIGs. This is mostly due to the fact that a Majority gate in QCA needs to route 4 wires from 4 different sides while an AND/OR only needs to route 3 wires. Therefore, a Majority gate blocks a lot of circuit area with just wiring. Furthermore, floor plans get less area efficient as well.

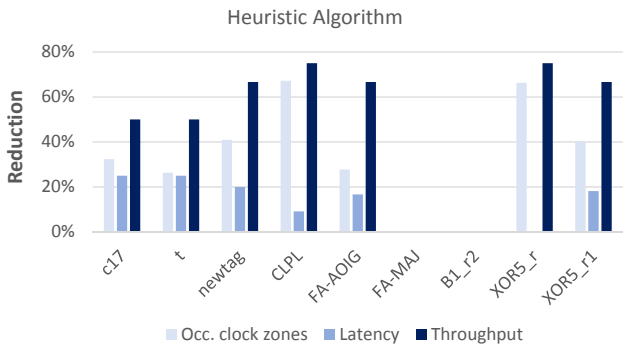



Fig. 10: Reduction of occupied clock zones, latency and throughput if global synchronicity is ignored using the heuristic algorithm

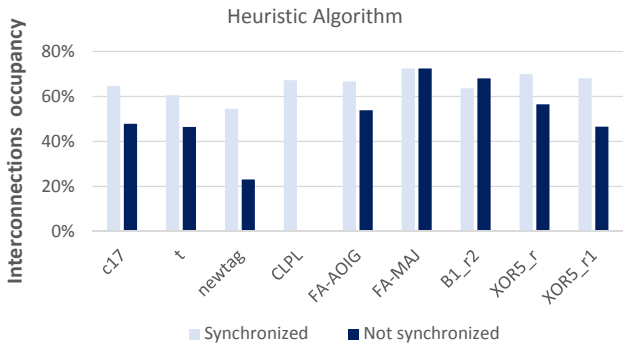



Fig. 12: Relation between total number of occupied clock zones and clock zones containing interconnections using the heuristic algorithm

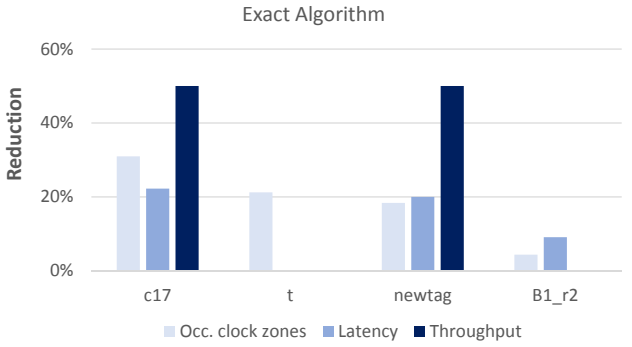



Fig. 11: Reduction of occupied clock zones, latency and throughput if global synchronicity is ignored using the exact algorithm

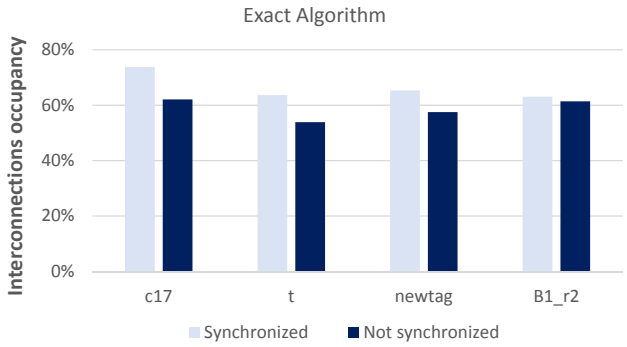



Fig. 13: Relation between total number of occupied clock zones and clock zones containing interconnections using the exact algorithm

In contrast, the throughput declined by up to 75% (*clpl* for the heuristic algorithm), i. e., factor 4, and in average by 50% for the heuristic algorithm and 25% for the exact one. In most cases, the value for throughput reduction is comparable to area reduction. However, in three cases, namely *t* and *FA-AOIG* for the heuristic algorithm and *newtag* for the exact algorithm, there is a strong discrepancy between area and throughput reduction. Hence, it is up to the designer to decide which of the parameters he wants to prioritize.

### B. Evaluation of impact of interconnections

Figs. 12 and 13 compare the relation between the total number of occupied clock zones and clock zones containing interconnections for both algorithms and for synchronized and unsynchronized designs. Results indicate that if using the synchronization constrain interconnections occupy up to 74% of all used clock zones (*c17* for the exact algorithm), while the average occupancy of the interconnections is about 65% for both algorithms. If the synchronization constraint is ignored the maximum number remains similar, i. e., 72% (*FA-MAJ* for the heuristic algorithm), while the average values improve to 46% for the heuristic algorithm and 59% for the exact one.

A special case is the circuit *clpl*, which contains no wires if global synchronicity is ignored. This results from the specific architecture of the circuit, which consists solely of a chain of two-input gates that have a primary input each. Consequently, the gates could be placed side-by-side as a long line.

In summary, these results clearly indicate that interconnections usually have a very high impact on the area costs of QCA designs and must not be ignored.

## VI. Conclusion

Quantum-dot Cellular Automata (QCA) is a promising nanotechnology with remarkable characteristics in terms of performance and energy consumption. QCA apply external clocks for control of information transfer such that circuits can have a pipeline-like behavior. We revealed in this work that, in contrast to what is common believe, this behavior is not a mandatory constraint for QCA circuits. Therefore, we discussed the differences between local and global synchronicity (GS) in QCA circuits. Further, we showed how placement and routing algorithm can be modified in order to allow to consider or not the GS constraint. Simulation results for selected benchmarks indicate that relaxing the GS constraint can lead to area reductions of about 70%, while the throughput reduces in similar range. Further, the latency could be improved by up to 25%. Hence, designers have a further degree of freedom in order to explore the full potential

TABLE I: Simulation Results for heuristic Algorithm (presented in Section IV-A)

| | | | | Synchronized | | | | Unsynchronized | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Benchmark** | **Gates** | **Inputs** | **Outputs** | **Grid area** | **Occupied clk-zones** | **Latency** | **Int.**[a] | **Grid area** | **Occupied clk-zones** | **Latency** | **Throughput** | **Int.**[a] |
| c17 | 12 | 5 | 2 | 56 | 34 | 8 | 22 | 35 | 23 | 6 | 0.50 | 11 |
| t | 15 | 5 | 2 | 49 | 38 | 8 | 23 | 56 | 28 | 6 | 0.50 | 13 |
| newtag | 20 | 8 | 1 | 80 | 44 | 10 | 24 | 48 | 26 | 8 | 0.33 | 6 |
| CLPL | 21 | 11 | 5 | 132 | 64 | 11 | 43 | 33 | 21 | 10 | 0.25 | 0 |
| FA-AOIG | 12 | 3 | 2 | 56 | 36 | 12 | 24 | 36 | 26 | 10 | 0.33 | 14 |
| FA-MAJ | 8 | 3 | 1 | 35 | 29 | 10 | 21 | 35 | 29 | 10 | 1.00 | 21 |
| B1_r2 | 16 | 3 | 4 | 60 | 44 | 11 | 28 | 96 | 50 | 11 | 1.00 | 34 |
| XOR5_r | 37 | 5 | 1 | 252 | 252 | 24 | 86 | 140 | 85 | 24 | 0.25 | 48 |
| XOR5_r1 | 31 | 5 | 1 | 195 | 97 | 22 | 66 | 160 | 58 | 18 | 0.33 | 27 |

[a] Interconnections (wires, fan-outs and crossovers)

TABLE II: Simulation Results for exact Algorithm (presented in Section IV-B)

| | | | | Synchronized | | | | Unsynchronized | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Benchmark** | **Gates** | **Inputs** | **Outputs** | **Grid area** | **Occupied clk-zones** | **Latency** | **Int.**[a] | **Grid area** | **Occupied clk-zones** | **Latency** | **Throughput** | **Int.**[a] |
| c17 | 12 | 5 | 2 | 48 | 42 | 18 | 31 | 35 | 29 | 14 | 0.50 | 18 |
| t | 15 | 5 | 2 | 40 | 33 | 11 | 21 | 30 | 26 | 11 | 1.00 | 14 |
| newtag | 20 | 8 | 1 | 60 | 49 | 20 | 32 | 49 | 40 | 16 | 0.50 | 23 |
| B1_r2 | 16 | 3 | 4 | 54 | 46 | 22 | 29 | 50 | 44 | 20 | 1.00 | 27 |

[a] Interconnections (wires, fan-outs and crossovers)

of the QCA technology. Additionally, we could show that interconnection heavily affect the area costs. In case of the analyzed benchmarks, in average nearly 60% of the total design area is occupied by interconnection like wires, fan-outs and crossovers. That means, future design methods for QCA circuits should focus on this factor, requiring a rethinking of how interconnections are considered on all design layers.

## 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, Apr 1997.

[2] J. Timler and C. Lent, "Power Gain and Dissipation in Quantum-dot Cellular Automata," *Journal of Applied Physics*, vol. 91, no. 2, pp. 823–831, 2002.

[3] F. Sill Torres, R. Wille, P. Niemann, and R. Drechsler, "An Energy-Aware Model for the Logic Synthesis of Quantum-Dot Cellular Automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3031–3041, Dec 2018.

[4] V. Arima, M. Iurlo, L. Zoli, S. Kumar, M. Piacenza, F. Della Sala, F. Matino, G. Maruccio, R. Rinaldi, F. Paolucci, M. Marcaccio, P. G. Cozzi, and A. P. Bramanti, "Toward quantum-dot cellular automata units: thiolated-carbazole linked bisferrocenes," *Nanoscale*, vol. 4, pp. 813–823, 2012.

[5] T. R. Huff, H. Labidi, M. Rashidi, M. Koleini, R. Achal, M. H. Salomons, and R. A. Wolkow, "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.

[6] X. K. Hu, H. Dey, N. Liebing, G. Csaba, A. Orlov, G. H. Bernstein, W. Porod, P. Krzysteczko, S. Sievers, and H. W. Schumacher, "Edge-Mode Resonance-Assisted Switching of Nanomagnet Logic Elements," vol. 51, no. 11, 2015.

[7] S. Bohloul, Q. Shi, R. A. Wolkow, and H. Guo, "Quantum Transport in Gated Dangling-Bond Atomic Wires," *Nano Letters*, vol. 17, no. 1, pp. 322–327, 2017.

[8] C. S. Lent *et al.*, "Molecular Cellular Networks: A non von Neumann Architecture for Molecular Electronics," in *IEEE International Conference on Rebooting Computing (ICRC)*, 2016.

[9] K. Hennessy and C. Lent, "Clocking of molecular quantum-dot cellular automata," *Journal of Vacuum Science & Technology B*, vol. 19, no. 5, pp. 1752–1755, 2001.

[10] J. Huang, M. Momenzadeh, L. Schiano, M. Ottavi, and F. Lombardi, "Tile-based QCA Design Using Majority-like Logic Primitives," *J. Emerg. Technol. Comput. Syst.*, vol. 1, no. 3, pp. 163–185, Oct. 2005.

[11] C. Campos, A. Marciano, O. Neto, and F. Sill Torres, "USE: A universal, scalable, and efficient clocking scheme for QCA," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 3, pp. 513–517, 2016.

[12] J. Huang, M. Momenzadeh, and F. Lombardi, "Design of Sequential Circuits by Quantum-dot Cellular Automata," *Microelectronics Journal*, vol. 38, no. 4, pp. 525 – 537, 2007.

[13] L. A. Lim, A. Ghazali, S. C. T. Yan, and C. C. Fat, "Sequential circuit design using Quantum-dot Cellular Automata (QCA)," in *2012 IEEE International Conference on Circuits and Systems (ICCAS)*, Oct 2012, pp. 162–167.

[14] F. Sill Torres, R. Wille, M. Walter, P. Niemann, D. Große, and R. Drechsler, "Evaluating the Impact of Interconnections in Quantum-dot Cellular Automata," in *Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 649–656.

[15] F. Sill Torres, P. A. Silva, G. Fontes, J. A. Nacif, R. S. Ferreira, O. P. V. Neto, J. Chaves, and R. Drechsler, "Exploration of the Synchronization Constraint in Quantum-dot Cellular Automata," in *Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 642–648.

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

[17] C. Lent, P. Tougaw, W. Porod, and G. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, no. 1, p. 49, 1993.

[18] E. Blair and C. Lent, "An architecture for molecular computing using quantum-dot cellular automata," in *IEEE-NANO 2003*, vol. 1, 2003, pp. 402–405.

[19] D. A. Reis, C. A. T. Campos, T. R. B. S. Soares, O. P. V. Neto, and F. Sill Torres, "A Methodology for Standard Cell Design for QCA," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 2114–2117.

[20] V. Vankamamidi, M. Ottavi, and F. Lombardi, "Two-dimensional schemes for clocking/timing of QCA circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 34–44, Jan 2008.

[21] F. Sill Torres, M. Walter, R. Wille, D. Große, and R. Drechsler, "Synchronization of Clocked Field-coupled Circuits," in *18th IEEE International Conference on Nanotechnology (IEEE-NANO)*, July 2018.

[22] A. Tang, Y. Kim, T. J. Reck, G. Chattopadhyay, I. Mehdi, B. J. Drouin, K. B. Cooper, N. J. Livesey, and M. F. Chang, "Ddfs and sd approaches for fractional frequency synthesis in terahertz instruments," *IEEE Transactions on Terahertz Science and Technology*, vol. 8, no. 4, pp. 410–417, July 2018.

[23] R. Wille, M. Walter, F. Sill Torres, D. Große, and R. Drechsler, "Ignore Clocking Constraints: An Alternative Physical Design Methodology

for Field-Coupled Nanotechnologies," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 651–656.

[24] M. Walter, R. Wille, D. Große, F. Sill Torres, and R. Drechsler, "Placement and Routing for Tile-based Field-coupled Nanocomputing Circuits is NP-complete (Research Note)," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 3, 2019.

[25] M. Walter, R. Wille, D. Große, F. Sill Torres, and R. Drechsler, "An Exact Method for Design Exploration of Quantum-dot Cellular Automata," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 503–508.

[26] T. Teodósio and L. Sousa, "QCA-LG: A Tool for the Automatic Layout Generation of QCA Combinational Circuits," in *Norchip 2007*, 2007.

[27] A. Trindade, R. Ferreira, J. A. M. Nacif, D. Sales, and O. P. V. Neto, "A Placement and Routing Algorithm for Quantum-dot Cellular Automata," in *Proceedings of the 29th Symposium on Integrated Circuits and Systems Design: Chip on the Mountains*, ser. SBCCI '16. Piscataway, NJ, USA: IEEE Press, 2017, pp. 12:1–12:6.

[28] G. Fontes, P. A. R. L. Silva, J. A. M. Nacif, O. P. V. Neto, and R. Ferreira, "Placement and Routing by Overlapping and Merging QCA Gates," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018.

[29] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *TACAS/ETAPS*, Berlin, Heidelberg, 2008.

[30] S. Yang, "Logic Synthesis and Optimization Benchmarks," Tech. Rep., Dec. 1989.

[31] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in *Circuits and Systems, 1989., IEEE International Symposium on*, May 1989, pp. 1929–1934 vol.3.

[32] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proceedings of IEEE Int'l Symposium Circuits and Systems (ISCAS 85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[33] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *CAV*, 2010, pp. 24–40.