Quantum-Dot Cellular Automata (QCA) Circuit Partitioning: Problem Modeling and Solutions

Dominic A. Antonelli † Danny Z. Chen † Timothy J. Dysart † Xiaobo S. Hu †
Andrew B. Kahng ⋆ Peter M. Kogge † Richard C. Murphy † Michael T. Niemier †

†CSE Department
University of Notre Dame
384 Fitzpatrick Hall
Notre Dame, IN 46556
USA

{dantonel,dchen,tdysart,
shu,kogge,rcm,mniemier}@cse.nd.edu

⋆CSE and ECE Departments
University of California, San Diego
Mailcode 0114
La Jolla, CA 92093-0114
USA

abk@ucsd.edu

ABSTRACT

This paper presents the Quantum-Dot Cellular Automata (QCA) physical design problem, in the context of the VLSI physical design problem. The problem is divided into three subproblems: partitioning, placement, and routing of QCA circuits. This paper presents an ILP formulation and heuristic solution to the partitioning problem, and compares the two sets of results. Additionally, we compare a human-generated circuit to the ILP and Heuristic solutions. The results demonstrate that the heuristic is a practical method of reducing partitioning run time while providing a result that is close to the optimal for a given circuit.

Categories and Subject Descriptors: B.6.3 Design Aids: Optimization

General Terms: Design, Algorithms

Keywords: Circuit Partitioning, Computer Aided Design, Quantum-Dot Cellular Automata (QCA)

1. INTRODUCTION AND MOTIVATION

Moore's law dictates that the number of devices integrated on a single die doubles every 18 months. Since 1965 this scaling has governed the manufacturing of integrated circuits. Studies indicate that as early as 2012, the CMOS transistor may hit physical scaling limits that inhibit this aggressive packing of devices. Although recent advances have given significantly more life to this device model, and some in the field contend that molecular transistors may be feasible in the near future, other devices may be better suited to the computing environment of the future. Understanding the design automation and computing models associated with a given set of devices is critical to evaluating their fitness.

One such device type, Quantum-Dot Cellular Automata (QCA) [12, 16, 18], first proposed in the early 1990s [12], transfers information by the propagation of polarized charge, rather than the flow of current. This device has the potential to greatly simplify the construction of circuits because every component of the circuit is represented by a cell and only one type of gate (the "majority gate") is needed (inversion can be performed in QCA "wires"). Furthermore, because the cells operate using Coulombic interaction (e.g., like charges repel), no current flows between the cells and no power (or information) is dissipated by the internals of the cell. Conservative estimates indicate that room temperature devices could be clocked in the 1-10 terahertz range and be 100 times more dense than a CMOS device at the end of the CMOS curve (e.g., the smallest non-molecular theoretically operable CMOS device), and dissipate very little power.

QCA has been realized using metal-dot cells [3, 19, 1, 20, 8] and there is tremendous opportunity in molecular implementations [14, 11] that allow for room temperature circuit operations. Current lines of investigation include choosing candidate molecules [13, 7, 21], clocking on the molecular scale [10], and circuit self-assembly [4]. Molecular scale circuit implementations are a near term goal, and one of the primary foci of this work is to provide a methodology for producing valid, correctly clocked circuits that can be used by those performing fabrication work.

Thus far, virtually all intellectual inquiry about emerging nano-scale systems has focused on the device physics, with some limited effort given to circuit design. There is significant emerging work in the area of constructing traditional computer architectures with QCA, and understanding which computational models best fit the device physics [18, 15, 16, 17]. However, given that nano-scale devices pose new and major challenges to circuit designers, particularly in terms of managing the transfer of state information between very fine-grain modules of computation (potentially consisting of a single gate), significant design automation is required to correctly construct QCA circuits (both in terms of the properties required for the devices making up the circuit to function physically, and the timing necessary for them to correctly implement the desired logic function).

The purpose of this work is to identify some important issues in QCA physical design automation, provide solutions to the subproblems of QCA layout partitioning and scheduling, and suggest a general methodology for other subproblems found in QCA physical design automation. These problems are uniquely QCA-driven since QCA circuits are inherently pipelined at the gate level, and the timing of signal delivery requires that all the signals for a given gate arrive simultaneously. In particular, each QCA wire inherently holds state information for a clock cycle because the wire is constructed from QCA devices, and the state is held in the position of the charge on two devices which are physically interacting. Due to this and other QCA specific features, QCA physical design introduces a number of interesting problems that are different from traditional VLSI physical design [2].

This paper examines the problem of partitioning a QCA circuit into *clocking zones*, which are required for the functioning of a QCA cell. As the circuit is partitioned, a schedule is created that strictly enforces the timing requirements of the circuit – specifically, that each of the signals arrives at its destination simultaneously. Furthermore, it provides for the tradeoff between the objective functions of constructing the minimum latency circuit, while simultaneously placing restrictions on the amount of wasted area in that circuit.

We propose an ILP formulation and heuristic algorithms for the QCA circuit partitioning problem. We also compare the output of the ILP and heuristic solutions using both actual and randomly generated circuits. The real example, chosen from the ALU of a complete processor laid out in QCA [15], gives the additional advantage of allowing comparisons to a full-custom layout. Our heuristic algorithms are based on a network flow model of the problem.

To the best of our knowledge, this is the first attempt to model and solve the QCA circuit partitioning problem. For traditional VLSI circuit partition work, see [2, 22]. For QCA circuit physics and design work, see [12, 16, 18, 17, 15].

Section 2 provides a brief introduction to QCA circuits. Section 3 describes the formulation of the overall problem and the impact of QCA device physics upon it. Section 4 examines the specific Partitioning and Scheduling problem, and presents the ILP and heuristic formulations. The experimental results are presented in Section 5, and conclusions in Section 6.

2. QCA BASICS

Before addressing layout, it is necessary to review some of the basic properties of QCA circuits. This discussion begins with a brief overview of the device physics, then considers basic logic, and finally introduces clocking to tie everything together. The layout rules for QCA are relatively simple, and fortunately, the layout process is flexible.

2.0.1 Basic Quantum Dot Construction

A single device is used for the construction of all components of an entire circuit (computational elements and wires). This QCA device consists of a cell with four quantum dots located in the corners and two mobile electrons. By providing tunneling junctions with potential barriers controlled by local electric fields that are raised to prohibit electron movement and lowered to allow electron movement, an isolated cell can have one of three states. A null state occurs when the barriers are lowered and the mobile electrons are free to localize on any dot. The other two states are polarizations that occur when the barrier is raised, and serve to minimize the energy states of the cell. These two states are denoted as $P = +1$ (binary 1) and $P = -1$ (binary 0). Figure 1 shows basic QCA cells and their two possible orientations. Cells placed near each other are forced into matching polarizations due to the Coulombic interactions between the cells. More details regarding the QCA device physics can be found in [12].

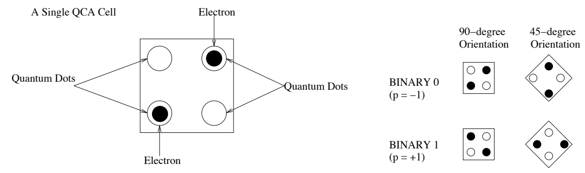

Figure 1: QCA Cells and Their Orientation

2.0.2 QCA-based Logic

The first consideration for logic is how to move data down a wire. This can be done by placing cells next to each other in a line and allowing them to interact. From the Coulombic interactions, if one cell at the end has a fixed polarization, then each cell in the line (wire) will take on the same value. However, since these devices are all in a two-dimensional plane, there is no analogy to "layers of metal" as there are in VLSI. As such, wire crossings must be constructed. Theoretically, cells can be oriented at either 90 degrees or 45 degrees (as depicted in Figure 1) in a specific layout to implement a wire crossing. Since manufacturing nano-scale cells with two different orientations is likely to be challenging, we seek to minimize the number of wire-crossings.

Now that signals can be passed along wires, gates need to be constructed to compute various logic functions. The basic logic gate in QCA is the majority gate, as can be seen in Fig. 2. The majority gate implements the logic function $F = AB + BC + AC$, where A, B, and C are inputs and F is the single output. By fixing a single input to 0, an AND gate can be implemented whereas fixing an input to 1 creates an OR gate. Fortunately, creating a fixed cell can be done within the manufacturing process and constant signals do not need to be routed within the circuit. Inversion can be done within the wire by slightly off-centering the wire, and can be considered "free." These gates provide a universal set of logic gates which allows for the continuation of the binary computing paradigm.

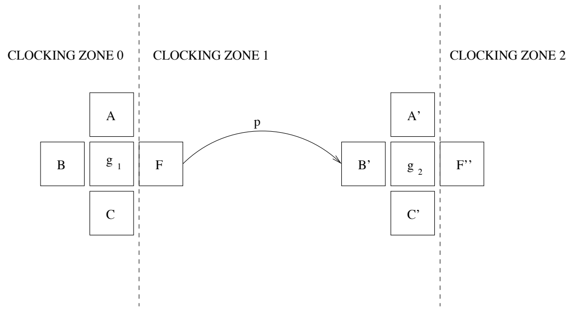

Figure 2: Majority Gates

2.0.3 QCA Clocking

Clocking plays a key role in controlling the QCA logic functionality. In order to have active computation, signals pass through *clocking zones*, which represent areas where this computation is occurring [17]. These clocking zones are a direct consequence of the QCA device physics. The clocking zones create the electric field which lowers and raises the potential barriers that allow the free electrons to tunnel or not. The clocking zones are physically adjacent, so the computation must proceed from one to the next in sequential order. As a particular clocking zone is performing a computation, the clocking zone before it must hold its outputs steady, and the clocking zone after it must perform no computation. This is a somewhat more simplified model than the 4-phase clock discussed in [15], but is more faithful to modern QCA clocking notions [5].

It is required that a signal only pass through active clocking zones and that all signals arrive at their required area of computation simultaneously. One direct method for generating such clocking zones is to assume a grounded plate above and parallel to the plane of the circuit. Below the plane of the circuit is a set of parallel wires, again in a parallel plane, but with the wires perpendicular to the signal flow. All wires carry the same clock signal but with neighboring wires 90-degrees out of phase[5]. (The same phase thus repeats every 4 wires.)

The timing rules of QCA circuits are strict and must be obeyed if the circuit is to function properly. For majority gates, its output cell must be in a clocking zone separate from that of the other cells in the gate. For example, in Fig. 2, the output cell of the left majority gate (cell F), is in clocking zone 1, while its inputs are in clocking zone 0. Since majority gates line up on the edges of each clocking zone while routes between the outputs of such gates and the input that they drive are constructed in the middle, gate-level pipelining is inherent within a QCA system. Figure 2 shows this clearly. The output from the gate on the left, which is in clocking zone 0, appears in cell F, and is held in the wire p. This value is then seen on input B' of the right majority gate, and is consumed by gate g_2. F, p, B', and C' are all in clocking zone 1. The result of the computation at gate g_2 is then seen in clocking zone 2. To have a minimum number of clocking zones, which is the minimal latency of the circuit, p must be contained in clocking zone 1 only. If p was in clocking zone 2 as well, this small system would then require 3 clocking zones, which would not be minimum.

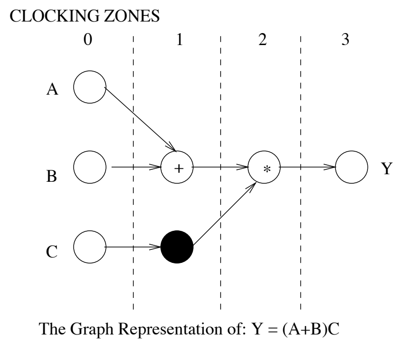

Figure 3: Example Circuit with Clocking Zones

Figure 3 depicts the circuit $Y = (A + B)C$. The inputs (represented by three separate nodes for clarity) appear in clocking zone 0. The OR must be computed in clocking zone 1 or later, and, as both the result of the OR and the primary input C are required before the AND can be computed, a buffer (indicated by the black dot) must be inserted into clocking zone 1 for signal C. Finally, the result of clocking zone 2's computation is available in clocking zone 3 for the output signal Y. Although this example is trivial, it depicts the fundamental timing problem for QCA circuits.

3. PROBLEM FORMULATION

Given a logic network based on QCA gates and represented as a directed acyclic graph (DAG), we wish to automatically generate a QCA physical layout that realizes the circuit using a minimum number of clocking zones, while simultaneously minimizing wasted area of the overall circuit. The wasted area is determined by the sum of the height differentials between the tallest clocking zone and all other clocking zones. Therefore, a uniform clocking zone height would have no wasted area and is desired. These two criteria form the core requirement in solving the QCA physical design problem.

Using VLSI design as an analogy, we can envision the QCA physical design process as consisting of partitioning, placement, and routing. However, some constraints and/or requirements in these subproblems are unique for QCA circuits.

1. **Partitioning**: This stage divides the DAG into partitions (or clocking zones) which fulfill the scheduling constraint, namely, all signals arrive at their destination simultaneously. When imagining the circuit laid out in two dimensions, this can be viewed as choosing the horizontal position for each gate. The number of cells within a clocking zone may impact the overall height and thus the area of the circuit. More importantly, each clocking zone must be of a similar height so that the clock (which is an electric field created under the clocking zone that controls the dot's state) can be easily and uniformly distributed.
2. **Placement**: This stage places the physical devices within their assigned clocking zones such that the number of wire crossings is minimized. In a QCA environment, wire crossings are expensive because they require either a large planar circuit to exchange the position of two signals, or a change in the orientation of individual quantum dots (offset by 45-degrees, see Section 2). These are expensive manufacturing requirement.
3. **Routing**: This stage constructs the optimal "wire" routing to implement the given circuit with the minimum number of wire crossings.

Just as in VLSI physical design, the three subproblems are closely related. Each of the tradeoffs (for wire crossing and routing) must affect and feed-back into the partitioning to preserve timing constraints. As in solving the VLSI physical layout problem, this can be dealt with by an iterative strategy.

In this paper, we focus on the partitioning problem. Our aim is to provide the best partition of gates into clocking zones in terms of the overall circuit area. Furthermore, we strive to develop general methodologies that can be reused in solving the other two subproblems. In the following, we introduce the necessary notation that formally defines the partitioning problem.

Let $G = (V, E)$ be a Directed Acyclic Graph describing a QCA circuit (e.g., Figure 4). G has a source and terminal vertex. All primary inputs are from the source, and primary outputs converge at the terminal. The remainder of the graph consists of vertices, which represent *majority gates*, and the edges in E represent data dependencies. If a vertex v_i represents an input, its indegree, $d_{in}(v_i)$, is 0 and its outdegree, $d_{out}(v_i)$, is ≥ 1. If v_i represents an output, its indegree is ≥ 1 and its outdegree is 0. If a vertex v_i represents a majority gate, it has the following properties (assuming a maximum fan out parameter F_{max}):

1. $d_{in}(v_i) \leq 3$
2. $1 \leq d_{out}(v_i) \leq F_{max}$

In summary, the requirements of the partitioning problem are to divide the circuit into legal clocking zones such that:

1. Ensure that all input signals to a gate arrive simultaneously;

2. Minimize the number of clocking zones, and thus, the total latency of the circuit; and
3. Create a uniform clocking zone height (the height of the clocking zone is determined by the number of gates and wire routing channels passing through that zone).

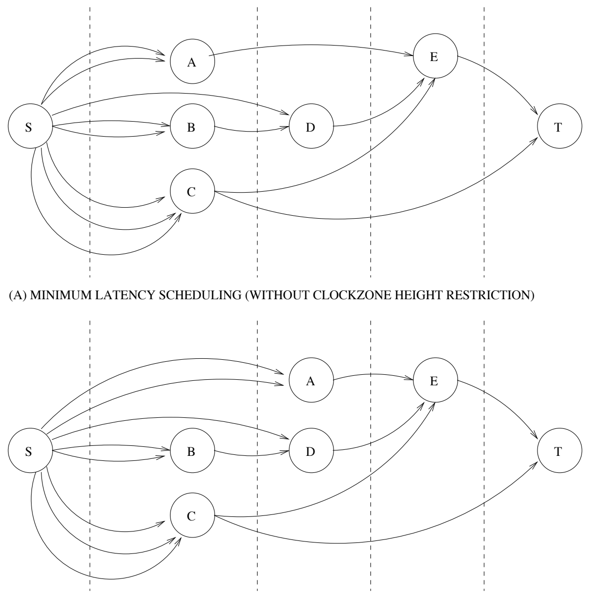
S
A
B
C
D
E
T
(A) MINIMUM LATENCY SCHEDULING (WITHOUT CLOCKZONE HEIGHT RESTRICTION)
(B) MINIMUM LATENCY SCHEDULING (WITH CLOCKZONE HEIGHT RESTRICTION)

Figure 4: Example Schedules

In considering the height of a clocking zone, the cells occupied by wiring channels cannot be ignored. In the current QCA technology, a wire channel can take up to 3 cells in height while a majority gate takes 5. Furthermore, whenever a wire crosses a clocking zone (regardless of its length), it introduces a unit of delay to the signal it carries. We use a simple example to show how the above constraints impact the partitioning process.

Figure 4 shows an example of two partitions. Figure 4(A) minimizes the latency of the circuit while the partition in Fig. 4(B) moves majority gate A one clocking zone to the right. As a result, we have traded two routing channels for a majority gate without impacting the overall latency of the schedule. In effect, the clocking zone containing B and C is now "shorter". Clearly, the QCA features make the partitioning problems somewhat different from the partitioning and scheduling problems studied in electronic design automation.

4. ALGORITHMS FOR PARTITIONING

A common mechanism for constructing a valid schedule given the timing requirements is to insert delay buffers through retiming [9]. The basic problem of minimizing the number of delay buffers inserted into the circuit while ensuring that all signals arrive simultaneously has been solved in polynomial time using a network flow model [6]. If we think of a wire in a QCA circuit as carrying the same number of delay buffers as the number of clocking zones it crosses, the partitioning problem is similar to the problem studied in [6]. However, the additional requirement that the difference in clocking zone heights be minimized while keeping the number of buffers to a minimum increases the complexity significantly. We will first present an ILP formulation of the problem, and then present a heuristic algorithm to solve the problem.

4.1 ILP Formulation

We use the variable X_{il} to represent the assignment of vertex V_i in $G(V,E)$ to clocking zone l. Specifically,

$$X_{il} = \{ {}^{1 \text{ if } v_i \text{ is assigned to clockingzone } l}_{0 \text{ otherwise}}$$

Note that any gate in a QCA circuit has only one output. However, the output can fan out to many other gates. To reduce each clocking zone height, we would like to maximize the sharing among the fanouts of a gate output. This is equivalent to delaying forking the wire of the output to as late as possible. If one models delay buffers based on edges, the fanout sharing can become complicated and lead to many more variables than necessary. We observe that the best sharing among fanouts will introduce the same number of buffers as the number of clocking zones between the gate output and the latest input driven by the fanout. Therefore we only associate buffers with gates, rather than edges. Specifically, buffers are represented by the variables P_{il}, which are defined as follows:

$$P_{il} = \{ {}^{1 \text{ if a buffer is inserted on the output node } i \text{ at clocking zone } l}_{0 \text{ otherwise}}$$

L represents the maximum allowed latency (which must be greater than or equal to the length of the critical path in G), and C is the height of the tallest clocking zone , e.g., $C = max(M_l C^M + B_l C^P), \forall l \leq L$, where C_l is the number of majority gate cells in clocking zone l, and B_l is the number of buffers in clocking zone l. C^M represents the height of a majority gate, and C^P represents the height of a buffer.

To ensure a valid schedule, the following inequalities representing the precedence constraints must be satisfied:

$$\sum_l X_{jl} l - \sum_l X_{il} l \geq 1, \forall (i,j) \in E \tag{1}$$

$$P_{il} \geq \sum_{h=1}^{l-1} (X_{ij} - X_{jh}) - X_{jl}, \forall (i,j) \in E, \forall l \tag{2}$$

To ensure a valid solution, X_l must satisfy the following:

$$\sum_l X_{il} = 1 \tag{3}$$

Our goal of minimizing the difference in clocking zone heights can be thought of as minimizing the maximum clocking zone height, which can be captured by the following constraint:

$$C \geq C^M \sum_i X_{il} + C^P \sum_i P_{il}(i), \forall l \tag{4}$$

Equation (1) ensures that data dependencies are considered and separated by at least one clocking zone (recall that QCA is inherently pipelined at the gate level). The second constraint, Equation (2), states that a buffer is needed at clocking zone l for gate v_i if v_i is scheduled before l and v_j is scheduled at or after l. The right hand side of Equation (4) is the sum of the number of majority gates and buffers in each clocking zone. Minimizing the value of C is the objective function for this ILP formulation.

4.2 Heuristic Solutions

The above ILP formulation is guaranteed to produce the optimal solution. However, solving the ILP can be quite time consuming for a large QCA circuit. In this section we describe a simple heuristic approach to solving the partition problem. The beauty of our heuristic approach is that it is a rather general methodology which can be employed in solving the other subproblems in QCA physical design automation.

Our heuristic makes use of the polynomial-time retiming algorithm presented in [6]. This algorithm solves the minimum buffer insertion problem for a pipelined circuit by relying on a network flow formulation. Our main idea is to apply the algorithm in [6] to a DAG that represents a QCA circuit, "perturb" the resulting solution by introducing extra constraints which confine some specific vertices to certain clocking zones, and then apply the algorithm in [6] again. The above can be repeated a number of times. The key to this heuristic is in the perturbation. We describe our approach in more detail below.

In solving the partitioning problem, we are willing to trade off latency to reduce clocking zone height and make the circuit more uniform. Our heuristic uses the network flow model to construct the minimum buffer insertion, and then examines the resulting graph to find clocking zones which are *impacted*. An *impacted* clocking zone is one which is "too tall", the difference between the maximum height and the average height is larger than a predefined threshold. The heuristic then adds a constraint to a node v in that clocking zone, forcing v to be scheduled later or earlier, and re-runs the network flow based algorithm on the modified graph. Adding these constraints must be done carefully to keep the problem polynomial time solvable (the proof is omitted due to page limit constraints). Given a set of clocking zone and design constraints (H_{max}, the maximum permissible clocking zone height, C), the general heuristic works as follows:

1. Run the network flow minimum latency scheduler to determine if an acceptable solution is found on G. If so, complete.

2. If the solution is unacceptable (because of too much variation in clocking zone heights), then modify G. An example of modifying G is to choose a vertex or subset of vertices not in the critical path, but in an impacted clocking zone, and create a precedence constraint requiring it to be *before* or *after* the node in the critical path that shares the same clocking zone in which it is currently scheduled. G could also be modified in several other ways such as by moving an impacted node to a later clocking zone (if legal),

3. Go to step 1.

By employing several different methods of modifying G, it is possible to develop a system with a more uniform clocking zone height. Additionally, by having these different methods, the heuristic may select a larger set of vertices which it could use to modify G. A major benefit of this heuristic solution is that it could be used to solve the Placement and Minimum Wire Crossing and Routing subproblems introduced in Section 3.

5. EXPERIMENTAL RESULTS

To test the ILP formulation and the heuristic, we first used a segment of a simple ALU, designed to perform addition, subtraction, AND, and OR operations. This ALU has the advantage of having been laid out by hand [15] and a single bit-slice is depicted in Figure 5. Circuits with one, two, and four bit-slices were tested with the graphs containing 24, 47, and 93 nodes respectively. Table 1 shows the critical path length (maximum latency) and the maximum clocking zone heights for both the ILP and heuristic methods. These solutions are either the same or within one cell in height for each of the three ALU segments. When comparing these to the maximum height of the full custom layout, which is 33, we see that the ILP and heuristic solutions are a slight improvement over the full custom layout, which is good.

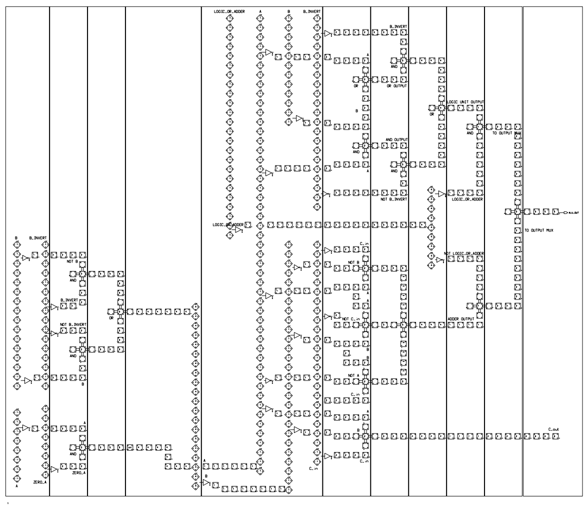

Figure 5: An ALU Designed by Hand

Table 1: ALU Bit-slice Results.

Num. Bit-slices	Crit. Path Len.	Optimal	Heuristic
1	10	31	31
2	10	51	52
4	12		78

A second group of tests was created to directly compare the ILP and heuristic solutions. The circuits for these tests were randomly generated and contain a specific number of gates. For this group of tests, 20 graphs were generated and tested for 10 gate, 20 gate, and 50 gate circuits. Table 2 shows the average, minimum, and maximum values for the critical path length, ILP maximum height, and heuristic maximum height. For reference, the average running time of the optimal ILP solution (for 50 nodes only) is 385 seconds, with a range of 10 seconds to 45 minutes, and a median of 45 seconds. The large average running time is due to the three tests that took longer than 5 minutes to complete. All other tests completed in the order of seconds. We see here that for these smaller circuits, the average height is within 10% - 15% of the optimal, while the minimum and maximum heights are the same. The heuristic algorithm required less than one second for small (less than 100 node) graphs, and up to 20 seconds for large (2000 node graphs) per iteration. The solution was found in one to 100 iterations (depending on the circuit).

Table 2: Small Graph Results.

Num. Nodes	Crit. Path Len. Avg,Min,Max	Optimal Avg,Min,Max	Heuristic Avg,Min,Max
10	6.3, 4, 8	18.35, 14, 23	18.6, 14, 23
20	8.3, 6, 12	29.7, 25, 38	32.25, 25, 38
50	10.75, 8, 13	60.83, 51, 69	70.28, 59, 87

Lastly, we generated groups of circuits with 100, 1000, and 2000 nodes through the heuristic solver. For graphs with 100 and 1000 nodes, five tests were conducted, and one was conducted for a graph with 2000 nodes. The results are summarized in Table 3 and contain the same information as the previous two tables. The ILP solutions are not found as they cannot be computed in a reasonable amount of time.

Table 3: Large Graph Results.

Num. Nodes	Crit. Path Length Avg., Min., Max	Heuristic Avg., Min., Max
100	15.6, 12, 16	129.8, 118, 143
1000	24.2, 24, 26	1426, 1402, 1463
2000	29	3061

From these results, we see that the optimal and heuristic solvers are nearly the same for small graphs and that they are comparable to a full-custom layout. For larger graphs, we have also shown that the heuristic solver can find solutions in a reasonable time frame. The results generated here show that our ILP formulation returns a desired answer and that the heuristic formulation enables us to find a solution that is close to the optimal, but is computed considerably faster, which enables us to partition large circuit designs.

6. CONCLUSIONS

In this paper, we have presented the QCA physical design problem, and related it to problems seen in VLSI physical design. To solve this problem, three subproblems were formulated to handle partitioning, placement, and routing of a QCA circuit. We then developed an ILP formulation and heuristic solutions to solve the partitioning problem. Comparing the results of the ILP and heuristic solutions shows that the heuristic is a practical method to reduce design times while providing a result that is close to optimal for a given circuit. Additionally, we compared these solutions to an ALU bit-slice that was generated by hand and found them to be in agreement.

Acknowledgments

This work was supported in part by the University of Notre Dame, in part by the National Science Foundation under grant CCR-0210153, and in part by SRC under grant id 2000HJ836. D.Z. Chen's research is supported in part by NSF Grant CCR-9988468. X.S. Hu's research is supported in part by NSF under grant number CCR02-08992. A.B. Kahng's research is supported in part by the MARCO Gigascale Systems Research Center.

7. REFERENCES

[1] A. O. Orlov et al. Expiremental demonstration of clocked single electron switching in quantum-dot cellular automata. *Appl. Phys. Lett.*, 77(2):295–297, July 2000.

[2] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19:1–81, 1995.

[3] I. Amlani et al. Digital logic gate using quantum-dot cellular automata. *Science*, 284:289–291, April 1999.

[4] Q. Hang, Y. Wang, M. Lieberman, and G. H. Bernstein. Molecular patterning through high-resolution polymethylmethacrylate masks. *Appl. Phys. Lett.*, 80(22):4220–4222, June 2002.

[5] K. Hennessy and C. S. Lent. Clocking of molecular quantum-dot cellular automata. *Journal of Vacuum Science and Technology*, 19(5):1752–1755, September/October 2001.

[6] X.S. Hu, S.C. Bass, and R.G. Harber. Minimizing the number of delay buffers in the synchronization of pipelined systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(12):1441–1449, December 1994.

[7] J. Jiao, G.J. Long, F. Grandjean, A.M. Beatty, and T.P. Fehlner. Building blocks for the molecular expression of quantum cellular automata. isolation and characterization of a covalently bonded square array of two ferrocenium and two ferrocene complexes. *J. Am. Chem. Soc.*, 125:7522–7523, 2003.

[8] R. K. Kummamuru et al. Operation of a quantum-dot cellular automata (QCA) shift register and analysis of errors. *IEEE Transactions on Electron Devices*, 50(9):1906–1913, September 2003.

[9] C. Leiserson and J. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5–35, 1990.

[10] C. S. Lent and B. Isaksen. Clocked molecular quantum-dot cellular automata. *IEEE Transactions on Electron Devices*, 50(9).

[11] C. S. Lent, B. Isaksen, and M. Lieberman. Molecular quantum-dot cellular automata. *J. Am. Chem. Soc.*, 125(4):1056–1063, 2003.

[12] Craig S. Lent, P. Douglas Tougaw, and Wolfgang Porod. Quantum Cellular Automata. *Nanotechnology 4, 49*, 1993.

[13] Z. Li and T. P. Fehlner. Molecular QCA cells. 2. characterization of an unsymmetrical dinuclear mixed-valence complex bound to a au surface by an organic linker. *Inorganic Chemistry*, 42:5715–5721, 2003.

[14] M. Lieberman et al. Quantum-dot cellular automata at a molecular scale. *Ann. N.Y. Acad. Sci.*, 960:225–239, 2002.

[15] M. T. Niemier and P. M. Kogge. Logic-in-wire: Using quantum dots to implement a microprocessor. In *International Conference on Electronics, Circuits, and Systems (ICECS '99)*, Cyprus, September 1999.

[16] Michael T. Niemier. *Designing Digital Systems in Quantum Cellular Automata*. MS CSE Thesis, University of Notre Dame, April 2000.

[17] Michael T. Niemier and Peter M. Kogge. Problems in designing with QCAs: Layout = timing. *International Journal of Circuit Theory and Applications*, 29:49–62, April 2001.

[18] Michael T. Niemier and Peter M. Kogge. Exploring and Exploiting Wire-Level Pipelining in Emerging Technologies. *ISCA*, June 2001.

[19] A. O. Orlov et al. Expiremental demonstration of a binary wire for quantum-dot cellular automata. *Appl. Phys. Lett.*, 74(19):2875–2877, May 1999.

[20] A. O. Orlov et al. Clocked quantum-dot cellular automata shift register. *Surface Science*, 532-535:1193–1198, June 2003.

[21] H. Qi et al. Molecular quantum cellular automata cells. electric field driven switching of a silicon surface bound array of vertically oriented two-dot molecular quantum cellular automata. *J. Am. Chem. Soc.*, 125:15250–15259, 2003.

[22] N. A. Sherwani. *Algorihms for VLSI Design Automation*. Kluwer Academic Publishers, 1995.