

Eliminating Wire Crossings for Molecular Quantum-dot Cellular Automata Implementation

Amitabh Chaudhary, Danny Z. Chen,
Xiaobo Sharon Hu, & Kevin Whitton
Dept. of Comp. Sci. & Eng.
University of Notre Dame
Notre Dame, IN 46545, USA

Email: {achaudha, chen, shu, kwhitton}@cse.nd.edu

Michael Niemier
& Ramprasad Ravichandran
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA

Email: {mniemier, raam}@cc.gatech.edu

Abstract—When exploring computing elements made from technologies other than CMOS, it is imperative to investigate the effects of physical implementation constraints. This paper focuses on molecular Quantum-dot Cellular Automata circuits. For these circuits, it is very difficult for chemists to fabricate wire crossings (at least in the near future). A novel technique is introduced to remove wire crossings in a given circuit to facilitate the self assembly of real circuits – thus providing meaningful and functional design targets for both physical and computer scientists. The technique eliminates all wire crossings with minimal logic gate/node duplications. Experimental results based on existing QCA circuits and other benchmarks are quite encouraging, and suggest that further investigation is needed.

I. INTRODUCTION

While it is important to study the theoretical properties of an emergent device, the feasibility of employing such a device to *build* interesting circuits must also be investigated simultaneously. The latter effort cannot only pave the way for actually implementing killer applications with new devices, but also provides valuable *guidance* to physical scientists as to where to focus their research efforts in order to gain performance wins of any kind. This is particularly critical for devices that are significantly different than CMOS transistors.

We consider the nano-scale, quantum-dot cellular automata (QCA), and investigate the impact of physical implementation constraints on realizing reasonably complex circuits. QCA, first proposed in the early 1990s [34], operates on binary data at the nanometer-scale. Logical operations and data movement are accomplished via Coulombic interaction rather than electric current flow. QCA circuits could be clocked at extremely high frequencies (1-10 THz), potentially leading to circuits with densities that are one or two orders of magnitude beyond what end-of-the-curve CMOS can provide [27], and dissipate very little power [33]. As circuits made from QCA devices could provide various “wins” over end-of-the-curve CMOS, in the last several years there has been an influx of QCA-related research [27], [35], [32].

Simple logic gates and a wire have been *implemented* using metal “dots” [30]. Unfortunately, metal QCA has two significant shortcomings: it must operate at a very low temperature, and mass fabrication faces the same challenges as CMOS. A promising implementation alternative is to use chemical molecules as QCA cells (mQCA) that would self assemble

to form larger circuits, and operate at room temperature [21].

At present, there is limited work examining physical design problems associated with QCA circuits [1], [15]. This work not only falls into this category, but also has been done in close coupling with physical scientists to explicitly consider buildability issues. Our work targets a molecular QCA implementation, yet also introduces a significant design constraint: with mQCA, wire crossings will be difficult to build. Thus, to help facilitate the *fabrication* of computationally interesting circuit designs with mQCA, we develop and present an approach that will eliminate all wire crossings for a given circuit via logic gate/node duplication. The goal of our algorithms is to minimize the number of gates that must be duplicated. We will provide an initial comparison of QCA designs with no crossings to CMOS designs, as well as to QCA designs that assume fabricatable wire crossings. This work should help to facilitate the implementation of small to medium sized mQCA circuits, and also illustrates the criticality of interactions between computer and physical scientists.

Eliminating wire crossings also finds applications in the physical synthesis of Binary Decision Diagram (BDD) based regular circuit structures. Non-Crossing Ordered BDD (NCOBDD) was first introduced by Cao and Koh [3] as a viable alternative to overcome the timing closure problem and offset the process variation in aggressively scaled silicon technology [25]. By eliminating wire crossings in a reduced ordered BDD (ROBDD), a more compact BDD structure can be obtained, which can be directly mapped to a regular circuit structure such as an FPGA or pass transistor logic (PTL) circuit. Such designs allow precise determination of delays and reduce cross-talk effects [3].

II. BACKGROUND AND RELATED WORK

QCA represents information by using binary numbers, but replaces a current switch with a cell having a bi-stable charge configuration. A device can consist of exactly 2 or 4 quantum dots and exactly 1 or 2 excess electrons respectively. One configuration of charge represents a binary ‘1’, the other a binary ‘0’ (Fig. 1(a)), but no current flows into or out of a cell [34], [18]. The charge configuration of one device alters the charge configuration of the next device. This device-device

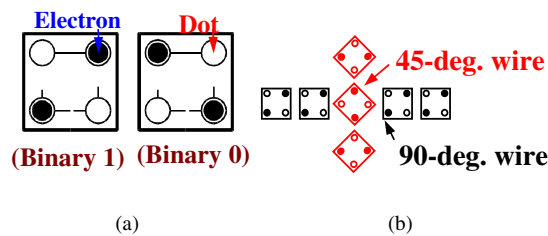


Fig. 1. (a) QCA representation of binary 1 and 0. (b) Wire crossing in QCA.

interaction allows for the computation of any Boolean function ([34], [27]), and also forms interconnects. An electric field (QCA's "clock") is also required which essentially turns QCA cells "on" (and allows them to represent a '1' or '0') or "off" (and places them into a null state) – see [12].

At present, four major "building blocks" are being considered for implementable systems of mQCA devices: (1) molecular QCA cells where one cell is approximately 1.2 nm by 1.2 nm with 3 nm center-to-center cell spacing, (2) DNA-based substrates for the molecules to attach to, which consist of tiles capable of holding 8 QCA cells each (similar to a 2 x 4 LEGO(tm) block); these would join together to form DNA rafts that hold QCA logic, (3) a silicon-based, electric field generating, clock structure, and (4) a means for integrating the QCA logic and the clock structure (a process called "liftoff" where DNA rafts are attached to the silicon clock structure in electron beam lithography etched tracks) [24]. The chemical process of assembling the DNA rafts, i.e., self-assembly, is the key to forming a circuit or system. It is desirable to have fewer distinct types of DNA rafts.

These building blocks unfortunately make it difficult to fabricate certain constructs of interest to the circuit designer – namely the co-planar wire crossing illustrated in Fig. 1(b). mQCA cells with different orientations can theoretically cross in the plane without the destruction of either value on either wire [34]. However, chemically, the physical layouts for wire crossings require sub-Angstrom control over the position and orientation of QCA cells and are difficult to fabricate. Thus, it is desirable to eliminate the need for the construct shown in Fig. 1(b) from initial design targets. This could be especially problematic as we are already considering just two dimensions of circuit complexity, and there is currently no analog to "multiple layers of metal".

One way to eliminate crossings in a given circuit is to duplicate logic gates such that the crossed connections can be removed. This is equivalent to planarizing a circuit. From a physical implementation standpoint, gate duplication will unavoidably result in more QCA logic gates. Furthermore, removing wire crossings with gate duplication essentially "pre-pones" all wire crossings by pushing them to the inputs of the circuit, which will result in more inputs. Both of the above consequences will most likely increase the overall circuit area. For small to medium sized circuits, the increase in number of QCA cells is not a significant problem for the following two reasons: (i) As the dimension of a molecular QCA cell is 1.2 nm by 1.2 nm, the resulting circuit should still have an area advantage over the end-of-the-curve CMOS implementation

even with many more QCA cells, and (ii) More importantly, the area increase caused by gate duplication does not introduce new gate types. This is much desired for the self-assembly process as a chemist would only have to map logic to DNA rafts one time, and parts (i.e. logic nodes) could be re-used. Mapping QCA cells to DNA rafts to deterministically interconnect logic nodes in varying circuits would be a **much** more difficult – if not impossible – task. Thus, another positive "side effect" of eliminating wire crossings via gate duplication is that all interconnects should become local, and essentially consist of simple, short, uniform, and reusable QCA wire segments. However, for large circuits, architectural innovations are needed.

Generally speaking, gate-duplication based wire-crossing elimination is to simply remove edge crossings in a directed graph through node duplication. We refer to this problem as the node-duplication based crossing-elimination (NDCE) problem. At first sight, the NDCE problem appears to be closely related to the crossing minimization problem (CM) on directed layered graphs that has been well studied in the graph drawing area [6], [7], [8] and arises in other applications such as visualization [6] and DNA mapping [36]. Liebers has presented a survey on planarizing graphs which includes an annotated bibliography [20]. CM has also been widely used in VLSI layout, e.g., [17], [5], [23], [31] to name a few.

While the NDCE problem seeks to avoid *all* edge crossings by inserting new nodes (gates), the CM problem aims to minimize edge crossings (edge crossings may still exist after the minimization). Though computationally the general versions of both problems are NP-hard, the combinatorial structures of the two problems are somewhat different. For example, for the key basic case which is defined on a two-layer graph with the order of the output nodes already fixed, the CM version is still NP-hard [8] but the NDCE version is linear-time solvable (as we will show). Due to the different structures of the two problems, our algorithm for the NDCE problem is quite different from those for the CM problem. Since the goal of NDCE is different from that of CM, comparing results of solving these two problems can be misleading and will not be attempted in our work.

To achieve planarity of logic circuits designed with Binary Decision Diagrams (BDDs) and Multi-values decision diagrams (MDDs), Sasao and Butler examined the necessary properties for BDDs and MDDs to be planar [29]. Specifically, they showed that threshold functions, symmetric functions and monotone increasing functions lead to planar diagrams. These functions can be useful in deriving certain logic circuits for QCA implementation. However, these functions alone cannot adequately deal with the general wire-crossing elimination problem.

Node duplication has been exploited by various performance and area optimization approaches in the design automation field. For example, cell duplication is used in circuit partitioning [22], [37], [9], in placement enhancement [26], [13], and in FPGA timing optimization [2], [10]. Yet, none of these takes wire-crossing elimination into consideration. To the best of

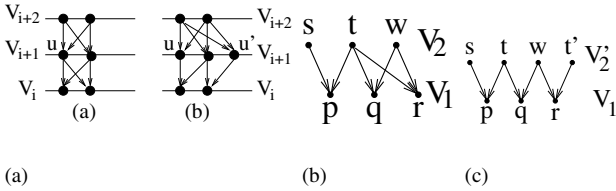


Fig. 2. (a) A node-duplication in V_{i+1} to avoid an edge crossing between V_i and V_{i+1} . (b) A 2-layer input graph G . (c) Output graph G' of G .

our knowledge, the only work that examined problems similar to NDCE is the NCOBDD synthesis by Cao and Koh [3], [4]. In [3], an interesting, albeit exponential time, back-tracking mechanism is proposed for optimal node duplication in a given ROBDD. An elegant decomposition technique is discussed in [4] to improve the efficiency of constructing NCOBDD. However, the work is limited to graphs in which each node has no more than two out-going edges.

III. OVERVIEW

In this section, we define the NDCE problem, give an overview of our heuristic NDCE algorithm, and highlight the key subproblems of our solution (the subsequent sections will be devoted to the solutions for these subproblems).

As discussed in Section 2, QCA circuits function under the influence of a clock field. To ensure proper evaluation and propagation of signals, a driving gate and its driven gates must be in two separate, yet consecutive clock zones [18]. This naturally gives rise to QCA circuits being “layered”. If a circuit design has wires across non-adjacent clock zones, buffers must be inserted in order to guarantee correct functionality, similar to synchronous circuits [1], [19]. Based on this fact, we study the NDCE problem based on the layered graph model.

Given an integer $k \geq 2$, let $G = (V, E)$ be a directed k -layered graph such that $V = \cup_{i=1}^k V_i$ and for each directed edge $(u, v) \in E$ from u to v , $u \in V_{i+1}$ and $v \in V_i$ for some $i \in \{1, 2, \dots, k-1\}$ (e.g., see Fig. 2(a)(a)). For modeling QCA or other circuit design applications, we assume that only V_k contains zero in-degree nodes; further, every non-zero in-degree node has at least two incoming edges. The set V_i is called the i -th layer of G . The NDCE problem seeks a planar layout of G , by possibly inserting new nodes (i.e., duplicating certain gates) into the layers V_2, V_3, \dots, V_k to avoid edge crossings in the layout. (Layer V_1 contains only the output nodes and does not need node-duplication.) A node-duplication (if needed) is done as follows: For two edges (u, v) and (u, w) such that $u \in V_{i+1}$ and $v, w \in V_i$, we duplicate node u by adding a new node u' to V_{i+1} to replace the edge (u, w) by a new edge (u', w) ; the new node u' has incoming edges from the same subset of nodes in V_{i+2} as u (if any). See Fig. 2(a) for an example. The objective is to minimize the number of inserted nodes. We can show that the NDCE problem is NP-hard.

This fashion of eliminating edge crossings via node duplication models the wire-crossing elimination problem for molecular QCA circuit implementation, where QCA gates are duplicated to avoid wire crossings.

Note that, to avoid edge crossings in the layout of G , we can (1) choose a “good” order of the nodes in each V_i , and

(2) possibly insert new nodes into the ordered node sequence of V_i ($i > 1$). However, as shown in Fig. 2(a)(b), inserting new nodes to V_{i+1} to avoid edge crossings between V_i and V_{i+1} may create more crossings between V_{i+1} and V_{i+2} , i.e., node-insertions may “propagate” edge crossings to adjacent layers.

Observe that only the node order needs to be considered for layer V_1 (the output-node layer) since node duplications in this layer would not help in reducing edge crossings. Furthermore, edge crossings propagate naturally to higher layers due to node-duplication. Hence, our heuristic algorithm for the NDCE problem uses the following iterative process: (1) eliminate edge crossings between V_1 and V_2 (called the *two-layer case*), and (2) for $i = 2, 3, \dots, k-1$, based on the (already fixed) node ordering of V_i (V_i may contain inserted nodes), determine the node ordering and node-insertion on V_{i+1} to avoid edge crossings between V_i and V_{i+1} (called the *layer propagation*).

Note that in the above iterative framework, the edge crossing propagation goes “upwards” (i.e., to higher layers). For this algorithm to work well, we need to solve two key subproblems: the two-layer case on V_1 and V_2 , and the layer propagation from V_i to V_{i+1} ($i \geq 2$).

To solve the two-layer case on V_1 and V_2 , we need to (i) determine an order of the nodes in V_1 , and (ii) determine a node order and possible node-insertions for V_2 . As to be shown in Section IV, the two-layer case is NP-hard. In Section IV, we give an integer linear programming (ILP) formulation for this case. Since solving the ILP formulation is time-consuming on large-size input, we also use a heuristic method to solve the two-layer case in polynomial time, as follows: We first choose a node order for V_1 (say, from a set of random node orders for V_1), and then apply our algorithm for the layer propagation problem to determine the order and node-insertions for V_2 ; the best output over the set of random node orders for V_1 is used as our output for the two-layer case.

The layer propagation problem from V_i to V_{i+1} assumes that the node order (possibly with inserted nodes) for V_i is already given, and we need to determine the order and node-insertions for V_{i+1} to avoid all edge crossings between V_i and V_{i+1} . As we show in Section V, this problem can be solved optimally by a nontrivial linear time algorithm.

IV. 2-LAYER NDCE PROBLEM

Given a directed 2-layer graph $G = (V_1, V_2, E)$, where V_1 and V_2 are the sets of nodes in layers 1 and 2, respectively, the 2-layer NDCE problem is to determine an order for the nodes in V_1 , and determine a node order and possible node duplications in V_2 . This problem is, unfortunately, NP-hard. This can be shown by a polynomial time reduction of the Hamiltonian path problem [11] to the 2-layer NDCE problem. We briefly discuss some key observations below but omit the detailed proof due to the page limit.

Let the optimal solution to the 2-layer problem assign the order σ_1 to the nodes in V_1 , and the order σ_2 to V_2' , which is the set possibly with duplications added to V_2 . As is explained

in detail later in Section V, the optimal solution is such that σ_1 has the largest possible number of pairs of consecutive nodes that share a common node in σ_2 (we call such pairs *sharings*). In fact, the number of duplications is $m - s^* - |V_2|$, where m is the number of edges of G and s^* is the largest possible number of sharings. (Note that a graph cannot have more than $|V_1| - 1$ sharings.)

Based on the NP-hard conclusion, it is unlikely that polynomial time algorithms exist for solving the 2-layer NDCE problem. We present an ILP formulation for the problem such that various ILP solvers can be exploited to solve the problem when the size of a 2-layer graph is not very large. In the formulation, we strive to introduce the minimal numbers of variables and constraints. To model the node order in Layer-1 we introduce variables y_{kl} , where

$$y_{kl} = \begin{cases} 1 & \text{if } v_k \in V_1 \text{ is at location } l \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

and $1 \leq k, l \leq |V_1|$ ($|V_1|$ is the number of nodes in Layer-1). For Layer-2, both the node order and node duplications need to be determined. That is, a node in Layer-2 can be duplicated (i.e., appear more than once). The maximum number of nodes in the resulting Layer-2 is bounded by the total number of edges, $|E|$, since in the worst case each node in Layer-2 only connects with one edge for which a non-crossing order always exists. We use variables x_{ij} to model the node order and duplication for Layer-2, where

$$x_{ij} = \begin{cases} 1 & \text{if } v_i \in V_2 \text{ is at location } j \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

and $1 \leq i \leq |V_2|$, $1 \leq j \leq |E|$. To model the edges, a straightforward way needs $|V_1|^2 \cdot |V_2| \cdot |E|$ variables to representation all possible connections between x_{ij} and y_{kl} . However, by careful formulation of the constraints, we only need $|V_1| \cdot |V_2| \cdot |E|$ variables defined as

$$z_{ijl} = \begin{cases} 1 & \text{if an edge exists between locations } j \text{ and } l \\ & \text{and the edge connects to node } v_i \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $1 \leq i \leq |V_2|$, $1 \leq j \leq |E|$, and $1 \leq l \leq |V_1|$.

Observe that each non-zero x_{ij} indicates the presence of a node in Layer-2. If the sum of x_{ij} is greater than $|V_2|$, duplicate nodes have been introduced. Therefore, the objective of the ILP formulation is to minimize the sum of x_{ij} , i.e.,

$$\text{Minimize: } \sum_i \sum_j x_{ij} \quad (4)$$

To enforce a valid assignment of nodes to locations, we introduce the following four constraints.

$$\sum_k y_{kl} = 1 \quad \forall 1 \leq l \leq |V_1| \quad (5)$$

$$\sum_l y_{kl} = 1 \quad \forall 1 \leq k \leq |V_1| \quad (6)$$

$$\sum_i x_{ij} \leq 1 \quad \forall 1 \leq j \leq |E| \quad (7)$$

$$\sum_j x_{ij} \geq 1 \quad \forall 1 \leq i \leq |V_2| \quad (8)$$

(5) and (6) require that each node in V_1 be assigned to one and only one location as no duplication is allowed in Layer-1. Since node duplication may be required in Layer-2 in order to remove crossings, (8) forces that each node v_i in V_2 can appear at one or more locations while (7) specifies that each location j in Layer-2 can have at most 1 node.

Additional constraints are needed concerning the edges. The first set of these, (9) and (10), are used to preserve the total number of edges connected to a node in Layer-1 and Layer-2, respectively.

$$\sum_i \sum_j z_{ijl} = \sum_k y_{kl} |E_k| \quad \forall 1 \leq l \leq |V_1| \quad (9)$$

$$\sum_j \sum_l z_{ijl} = |E_i| \quad \forall 1 \leq i \leq |V_2| \quad (10)$$

To ensure that edges in the resulting graph are always connected with the correct nodes, we introduce two additional constraints, (11) and (12).

$$z_{ijl} \leq x_{ij} \quad \forall 1 \leq i \leq |V_2|, 1 \leq j \leq |E|, 1 \leq l \leq |V_1| \quad (11)$$

$$z_{ijl} \leq \sum_{k|(i,k) \in E} y_{kl} \quad i, j, l \text{ are the same as above} \quad (12)$$

Inequality (11) specifies that an edge between node v_i at location j of Layer-2 and location l of Layer-1 can exist (i.e., $z_{ijl} = 1$) only if node v_i is assigned to location j (i.e., $x_{ij} = 1$). Similarly, (12) indicates that an edge between node v_i at location j of Layer-2 and location l of Layer-1 can only exist if at least one of the Layer-1 nodes connected to node v_i of Layer-2 is assigned to location l . In (12), we use a summation over k to capture the edges connected to node v_i , and thus avoid the introduction of more edge related variables. This is only possible for Layer-1 since each node in Layer-1 can appear at only one location. The last constraint enforces that no crossing occurs in the resulting graph as given below:

$$\sum_i z_{ijl} + \sum_i z_{ij'l'} \leq 1 \quad \forall 1 \leq j < j' \leq |E|, 1 \leq l' < l \leq |V_1| \quad (13)$$

Note that a crossing occurs if two locations j and j' in Layer-2 are connected to two locations l and l' in Layer-1, respectively, while $j < j'$ and $l > l'$.

We will illustrate the ILP formulation for the example graph in Fig. 2(b). For this example, $V_1 = \{p, q, r\}$ and $V_2 = \{s, t, w\}$. We name the locations for V_1 as $\{1, 2, 3\}$ from left to right and those for V_2 as $\{1, 2, 3, 4, 5, 6\}$. The objective function is then

$$\text{Minimize: } \sum_{j=1}^6 (x_{sj} + x_{tj} + x_{wj}) \quad (14)$$

We omit the constraints corresponding to (5)-(8) as they are straightforward. To ensure that the correct number of edges are connected to a location in Layer-1, say location 1, we obtain the following constraint based on (9):

$$\sum_{j=1}^6 z_{sj1} + \sum_{j=1}^6 z_{tj1} + \sum_{j=1}^6 z_{wj1} = 2y_{s1} + 2y_{t1} + 2y_{w1} \quad (15)$$

For a node in V_2 , say t , we have

$$\sum_{j=1}^6 z_{tj1} + \sum_{j=1}^6 z_{tj2} + \sum_{j=1}^6 z_{tj3} = |E_t| = 3 \quad (16)$$

Constraints for other locations and nodes can be derived accordingly. To guarantee the correct connections between nodes of the resulting graph, we need to constrain the edge variables based on (11) and (12). Consider the edge variable corresponding to a connection between location 1 in Layer-1 and location 4 in Layer-2, we have

$$z_{s41} < x_{s4}, \quad z_{s41} < y_{p1} \quad (17)$$

$$z_{t41} < x_{t4}, \quad z_{t41} < y_{p1} + y_{q1} + y_{r1} \quad (18)$$

$$z_{w41} < x_{w4}, \quad z_{w41} < y_{q1} + y_{r1} \quad (19)$$

Similarly, constraints for other edge variables can be built. To enforce that the connection between location 1 in Layer-1 and location 4 in Layer-2 does not cross that between location 3 in Layer-1 and location 2 in Layer-2, we need

$$z_{s41} + z_{t41} + z_{w41} + z_{s23} + z_{t23} + z_{w23} \leq 1 \quad (20)$$

(See (13).) No-crossing constraints for other connections can be constructed similarly. Solving the ILP, we obtain the resulting graph as given in Fig. 2(c).

V. LAYER PROPAGATION ALGORITHM

In this section we present an algorithm for the layer propagation problem, which was introduced in Section III. Given a fixed order of the first layer, the algorithm determines the node order and duplications for the second layer with the *minimum* number of node duplications, and it has a time complexity that is linear in the input size. We present the algorithm in two stages: (1) we transform the layer propagation problem to that of finding a shortest path between two nodes in a graph of size polynomial in the size of the input, and (2) we show how a shortest path in (1) can be computed in time that is linear in the size of the input.

To recall, in the layer propagation problem, we are given a directed 2-layer graph $G = (V_1, V_2, E)$.

All directed edges are of the form (u, v) where $u \in V_2$ and $v \in V_1$. The order of vertices in V_1 is fixed. The problem is to determine the insertions of nodes (duplications) in V_2 as well as the order of nodes in V_2 (including the new duplicate nodes) such that all edge crossings are eliminated and the number of duplications is minimized. See Fig. 2(b) and 2(c).

Let n_1 and n_2 denote $|V_1|$ and $|V_2|$ respectively, and let m denote $|E(G)|$. Denote the given order of nodes in V_1 by σ_1 and without loss of generality let it be v_1, \dots, v_{n_1} . Let the solution be graph G' such that $V(G') = V_1 \cup V_2'$, where V_2' is the second layer with duplications added. Let the order of nodes in V_2' in the solution be denoted by $\sigma_2 = u_1, \dots, u_{m'}$, where $u_i \in V_2'$ and $m' = |V_2'| \leq m$. Note that each $u_i \in V_2'$ corresponds to a node in V_2 ; denote it by $f(u_i)$.

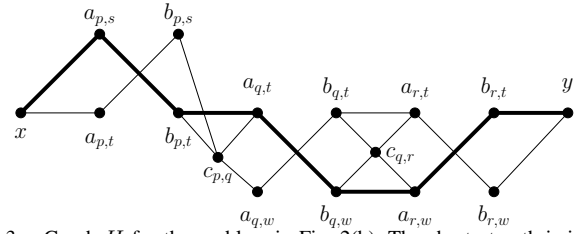


Fig. 3. Graph H for the problem in Fig. 2(b). The shortest path is in bold.

A. Partial Order on Edges

Consider the edges in the optimal output graph G' . There is a one-to-one correspondence between edges $(u_x, v_a) \in E(G')$ and $(f(u_x), v_a) \in E(G)$. Let (u_x, v_a) and (u_x, v_b) be two edges in G' . Also, let $a < b$. Since there are no edge crossings in G' , there can be no edge (u_y, v_a) in G' with $y > x$; otherwise, this edge will cross (u_x, v_b) . Similarly, there can be no edge (u_w, v_b) in G' with $w < x$. Thus there is a partial edge order, based on the nodes in V_1 that they are adjacent to, which the edges in G' must respect.

Let v_i and v_{i+1} be a pair of consecutive nodes in σ_1 . The partial order on edges implies that these two nodes can have at most one common neighbor in V_2' . A trivial non-optimal solution to the layer propagation problem is to have a distinct node in V_2' corresponding to each edge in $E(G)$. This would imply that the number of duplications is $m - n_2$. In this solution, no pair of nodes in V_1 shares a common neighbor. Instead, in our solution, suppose that there are s pairs of consecutive nodes in V_1 that share a common neighbor in V_2' , that is, there are s sharings. The number of duplications then is $m - s - n_2$. Thus, to minimize the number of duplications, we need to maximize the sharings.

B. Shortest Path

We now define a graph H such that computing the maximum number of sharings s^* in G' corresponds to computing a shortest path between two nodes in H . For each node $v_i \in V_1$, there are two sets A_{v_i}, B_{v_i} of nodes in H . Both sets contain the same number of nodes, each corresponding to an edge adjacent to v_i in G . Thus, for an edge (u, v_i) in G , there is a node $a_{v_i, u} \in A_{v_i}$ and a node $b_{v_i, u} \in B_{v_i}$. For the graph in Fig. 2(b), edges (q, w) and (r, w) introduce nodes $a_{q,w}, b_{q,w}$ and $a_{r,w}, b_{r,w}$, respectively as in Fig. 3. Intuitively, A_{v_i} is used to determine a common neighbor with $B_{v_{i-1}}$, and B_{v_i} to determine a common neighbor with $A_{v_{i+1}}$. In order to do so, there is an (undirected) edge between $a_{v_i, u}$ and $b_{v_{i-1}, u}$ for every node $u \in V_2$ that v_i and v_{i-1} have as a common neighbor (e.g. edge $(b_{q,w}, a_{r,w})$ in Fig. 3). If in the (optimal) solution u is a common neighbor that results in a sharing between v_i and v_{i-1} , the shortest path will use this edge. If, instead, v_i and v_{i-1} have no sharing, none of these edges will be in the path. For that case, we add a node c_{v_{i-1}, v_i} such that there is an edge to it from all nodes in A_{v_i} and from all nodes in $B_{v_{i-1}}$ (e.g., node $c_{q,r}$ in Fig. 3).

In addition, there are edges between nodes in A_{v_i} and B_{v_i} that depend on the number of edges adjacent to v_i in G . If v_i has just one adjacent edge in G , say (u, v_i) , then there is an edge between $a_{v_i, u}$ and $b_{v_i, u}$. If, however, v_i has 2 or more

adjacent edges in G , then the situation is quite the opposite. Let U be the set of nodes in V_2 that have an edge to v_i . For each $u \in U$, $a_{v_i,u}$ has edges with $b_{v_i,z}$, for all $z \in U \setminus \{u\}$. Doing so prevents our path from choosing the same node u (in V_2) for v_i to share with v_{i-1} as well as with v_{i+1} . We need to prevent this only when $|U| \geq 2$. We also add a node x to H with edges to all nodes in A_{v_1} and a node y with edges to all nodes in $B_{v_{n_1}}$.

Note that H has $O(m)$ nodes and $O(mn)$ edges. The construction of H allows us to solve the layer propagation problem exactly as stated in the following theorem. We omit the proof due to page limit.

Theorem 1: Every feasible solution σ_2 for the layer propagation problem on G corresponds to an x - y path in H . In addition, the length of the shortest x - y path in H is equal to $3n_1 - s^*$, where s^* is the maximum possible sharings in G' .

VI. EXPERIMENTAL RESULTS

To evaluate, we focus on the effect of node-duplication based wire-crossing elimination on circuit area, and the efficiency of the algorithms. To assess the impact of removing wire crossings on the area of a circuit, we compare not only the numbers of nodes (gates) based on the abstract graph model, but also the QCA layouts before and after applying our algorithms. We first present results based on the graph model and then discuss their impact on actual circuit layout.

We applied both the ILP-based and random-order based NDCE algorithms to a sample QCA circuit (the control path of a 12-bit accumulator-based microprocessor [27], referred to as S12) as well as a subset of circuits in the ISCAS benchmark suites. Results are summarized in Table I. The last five circuits are BDDs similar to the ones used in [3]. The second column of Table I indicates the total number of nodes (original gates and buffers for converting the connections across multiple layers to those only between two adjacent layers) in each circuit. The next two columns give the total numbers of nodes in the circuits with no crossings, and the last two columns summarize the run times for both the ILP-based and random-order based NDCE algorithms. ('XX' indicates that the problem size was too large to be solved by ILP in a reasonable time). For each random-order based result, the number reported is an average of 10 runs; the difference between the minimum and the maximum of each 10 runs is somewhere between 3% to 20%.

From Table I, we can conclude that our algorithms are effective in solving the NDCE problem, especially if random selection is used for ordering the layers. Recall that the ILP formulation gives the optimal result only for the two-layer problem. In general, an optimal two-layer (local optimal) solution does not necessarily guarantee to the best global solution. This scenario is evident in the C27 example.

As expected, the data in Table I reveals that the run times for ILP solutions can be quite long. Thus, the ILP-based NDCE algorithm is only applicable to circuits with a small number of gates and wires at the two layers closest to the output.

Circuit	# of Nodes before	# of Nodes after		Time Taken (sec)	
		ILP	Random	ILP	Random
S12	83	184	184	<1	<1
C17	16	19	19	<1	<1
C27	29	32	31	<1	<1
C432	468	1,324	1,328	2.3	<1
C499	1,403	XX	160,021	XX	58.34
C880	2,227	XX	50,832	XX	16.98
C1355	1,343	XX	156,087	XX	62.79
xor5 (BDD)	9	13	13	<1	<1
xor7 (BDD)	13	28	28	<1	<1
rd53 (BDD)	29	35	35	<1	<1
rd73 (BDD)	49	71	71	<1	<1
z4ml (BDD)	61	76	76	<1	<1

TABLE I
RESULTS OF APPLYING NDCE ALGORITHMS.

We have also compared our technique with the one in [3] for BDDs. Since the dynamic ordering used in generating the ROBDDs is not given in [3], we simply picked an arbitrary order to create the BDDs for the circuits in [3]. For most circuits, both methods give the same results. For some, our method is either slightly worse or better than that in [3]. (Due to the space limit, we omit the detailed comparison, but note that the technique in [3] is not applicable to general circuits.)

Not surprisingly, the number of nodes duplicated may grow rapidly as circuits become bigger. To examine the impact of such an increase, we generated the QCA layouts for several circuits. We used a custom QCA Place and Route (PR) tool to convert graph representations to actual QCA layouts. Nodes were replaced with relevant logic where appropriate. A 3 cell pitch was assumed between parallel QCA wires – to obtain both a dense layout while simultaneously considering (and avoiding) any cross talk.

Statistics for equivalent circuits with wire crossings (construct in Fig. 1(b)) were also computed. As the clock wire pitch will be a function of a given lithographic process, we chose to compare the circuits before and after duplication by considering the ratio of the number of cells in a circuit, and the ratio of the area of a bounding box. This should provide some initial insight into how circuits with constructs believed to be implementable, compare to circuits with constructs that are theoretically possible.

Table II summarizes our results. Pre- and post-duplication circuit data, and the ratio between them, are listed in adjacent rows. Col. 2 is the number of wire crossings for a QCA layout via our PR tool, Col. 3 is the number of QCA cells in the interconnect (anything not in a logic node), Col. 4 is the number of QCA cells in the logic nodes, Col. 5 is the total number of QCA cells in the design, and Col. 6 is the bounding box area in μm^2 (using 3 nm center-to-center cell spacing).

The experimental data shows that for relatively small circuits, node duplication can be a viable approach for eliminating wire crossings. For the Simple12 control path, post-duplication increases the area by a factor of 3. However, the number of QCA cells required to construct this circuit has gone down. Most importantly, the number of cells required to connect logic nodes has almost been cut in half. Long and convoluted wires that existed pre-duplication have been

Circuit Name	Wire Cross.	QCAs in IC	QCAs in Logic Nodes	Total QCA Cells	Bounding Box Area (μm^2)
S12 Bef.	129	7685	1949	9634	1.62
S12 Aft.	0	4325	4508	8833	4.78
Aft./Bef.	1	0.563	2.31	0.917	2.95
c17 Bef.	1	159	195	354	0.019
c17 Aft.	0	217	228	445	0.029
Aft./Bef.	1	1.36	1.17	1.26	1.52
c432 Bef.	1267	80141	6541	86655	8.82
c432 Aft.	0	18886	16942	35828	3.45
Aft./Bef.	1	0.236	2.60	0.413	0.391
c880 Bef.	3079	327237	27191	354428	62.6
c880 Aft.	0	980292	816039	1796331	1658.9
Aft./Bef.	1	3.00	30.0	5.07	26.5

TABLE II
IMPACT OF DUPLICATION ON QCA LAYOUTS.

replaced by shorter, local wires in the interconnect post duplication. This explains the drop in the number of QCA cells in the interconnect. For c432, again, the number of cells required has dropped by a factor of 4, and by a factor of 2.4 overall. Additionally, the total area required for this circuit has actually decreased by a factor of 2.5!. We believe that this occurs because wire crossings have been eliminated. More specifically, if the two signals that were initially on 90-degree wires needed to cross, one value would have to be ripped onto another wire – creating additional area overhead.

We also estimated the area of the CMOS implementation of c432 based on the standard-cell design given in [14]. We used $\lambda = 25\text{nm}$ for a hypothetical 45nm technology. The CMOS design would take $7812\mu\text{m}^2$, which is more than 2000 times larger than the no-crossing QCA design. While such a comparison does not give a complete picture, it does provide motivation to further consider this research.

Unfortunately, the above trends do not hold. For the more complex c880 example, the number of QCA cells increases by a factor of 3 in the interconnect, increases by a factor of 5 overall, and the area of the circuit increases by a factor of 26.5. We believe that this problem stems from two sources. First, the number of inputs to our combinational logic has increased (i.e., c432 requires 36 individual input signals, and c880 requires 60). Thus, more signals must be routed to different layers in a graph representation of a circuit. Second, as we continue to iterate through the layers of the circuit, we must duplicate larger and larger parts of the circuit. This is somewhat expected as theoretically, removing wire crossings via node duplication causes the number of duplications to grow exponentially.

REFERENCES

[1] D.A. Antonelli, D.Z. Chen, T.J. Dysart, X.S. Hu, A.B. Khang, P.M. Kogge, R.C. Murphy, and M.T. Niemier, "Quantum-Dot Cellular Automata (QCA) Circuit Partitioning: Problem Modeling and Solutions," *Des. Auto. Conf.*, 2004, pp. 363-368.

[2] G. Beraudo and J. Lillis, "Timing optimization of FPGA placements by logic replication," *Des. Auto. Conf.*, 2003.

[3] A. Cao and C.-K. Koh, "Non-Crossing OBDDs for Mapping to Regular Circuit Structures," *Int'l Conf. on Computer Design*, 2003, pp. 338-343.

[4] A. Cao, C.K. Koh, "Decomposition of BDDs with Application to Physical Mapping of Regular PTL Circuits," *Int. Work. for Logic Syn.*, 2004.

[5] H. Chen and D. Lee, "On crossing minimization problem," *IEEE Trans. on Computer-Aided Design*, 1998.

[6] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, NJ, 1998.

[7] V. Dujmovic and S. Whitesides, "An Efficient Fixed Parameter Tractable Algorithm for 1-Sided Crossing Minimization," *Algorithmica*, 40(1) (2004), pp. 15-31.

[8] P. Eades and S. Whitesides, "Drawing Graphs in Two Layers," *Theor. Comput. Sci.*, 131(2) (1994), pp. 361-374.

[9] M. Enos, S. Hauck and M. Sarrafzadeh, "Replication for logic bipartitioning," *Int'l Conf. on Computer-Aided Design*, 1997, pp.342-349.

[10] W. Fang and A. Wu, "Performance-driven multi-FPGA partitioning using functional clustering and replication," *Des. Auto. Conf.*, 1998.

[11] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[12] K. Hennessy, C.S. Lent, "Clocking of molecular quantum-dot cellular automata", *J. of Vac. Sci. & Tech. B*, 19,5(Sep-Oct 2001), 1752-55.

[13] M. Hrkic, J. Lillis and G. Beraudo, "An approach to placement-coupled logic replication," *Des. Auto. Conf.*, 2004, pp. 711-716.

[14] S.P. Khatri, A. Mehrotra, R.K. Brayton, A. Sangiovanni-Vincentelli and R.H. Otten, "A novel VLSI layout fabric for deep sub-micron applications," *Des. Auto. Conf.*, 1999.

[15] R. Ravichandran, N. Ladiwala, J. Nguyen, M. Niemier, S.K. Lim, "Automatic cell placement for quantum-dot cellular automata," *ACM Great Lakes Symposium on VLSI*, 2004, pp. 332-337.

[16] H.T. Kung, "Why Systolic Architectures?," *Computer*, 1982, p. 37-46.

[17] T. Lengauer, *Comb. Algs. for Int. Circuit Layout*, Wiley, New York, 1990.

[18] C.S. Lent, P.D. Tougaw, "A Device Architecture for Computing with Quantum Dots", *Proc. of the IEEE*, Vol. 85, p. 541, 1997.

[19] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, vol. 6, 1991, pp. 5-35.

[20] A. Liebers, "Planarizing graphs – a survey and annotated bibliography," *J. of Graph Algorithms and Applications*, vol. 5, no. 1, 2001, pp. 1–74.

[21] M. Lieberman, S. Chellamma, B. Varughese, Y. Wang, C. Lent, G. Bernstein, G. Snider, and F. Peiris, "Quantum-dot Cellular Automata at a Molecular Scale", *A. of New York Acad. of Sci.*, 960, p.225-39, Apr. 2002.

[22] L. Liu, T. Kuo, C.K. Cheng and T.C., Hu, "A replication cut for two-way partitioning," *IEEE Trans. on CAD*, 1995.

[23] M. Marek-Sadowska and M. Sarrafzadeh, "The Crossing Distribution Problem," *IEEE Trans. on CAD*, 14(4) (1995), pp. 423-433.

[24] Q.L. Hang, Y.L. Wang, M. Lieberman, G. Bernstein, "A liftoff technique for molecular nano patterning", *J. of Nanosci. & Nanotech.*, (3)309-12, 2003.

[25] F. Mo and R. Brayton, "Regular Fabrics in Deep Sub-Micron Integrated-Circuit Design," *Int'l Work. on Logic Synthesis*, 2002, p.7-12.

[26] I. Neumann, D. Stoffel, H. Hartje and W. Kuntz, "Cell replication and redundancy elimination during place for cycle time optimization," *Int'l Conf. on Computer-Aided Design*, 1999.

[27] M.T. Niemier, "The Effects of a New Technology on the Design, Organization, and Architectures of Computing Systems," *Ph.D. Dissertation*.

[28] R. Ravichandran, N. Ladiwala, J. Nguyen, M. Niemier and S.K. Lim, "Automatic Cell Placement for Quantum-dot Cellular Automata", *ACM Great Lakes Symposium on VLSI*, April 2004.

[29] T. Sasao and J. T. Butler, "Planar decision diagrams for multiple-valued functions," *Multiple-Valued Journal*, Vol. 1, No. 1, 1996, pp. 39-64.

[30] G.L. Snider, A.O. Orlov, I. Amlani, G.H. Bernstein, C.S. Lent, J.L. Merz and W. Porod, "Quantum-Dot Cellular Automata: Line and Majority Gate Logic", *Jpn. J. of Applied Physics*, 38, pp. 7227-7229, 1999.

[31] X. Song and Y. Wang, "On the crossing distribution problem," *ACM Trans. on Design Automation of Electronics Systems*, 1999.

[32] M.B. Tahoori, J. Huang, M. Momenzadeh and F. Lombardi, "Defects and Faults in QCA at the nano-scale", *VLSI Test Sym.*, 2004.

[33] J. Timler and C.S. Lent, "Power gain and dissipation in quantum-dot cellular automata", *J. of App. Phys.*, 91, 2002, p.823-831.

[34] P.D. Tougaw, C.S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata", *J. of App. Phys.*, Vol. 75, 1994, p. 1818.

[35] K. Walus, T. Dysart, G.A. Jullien and R.A. Budiman, "QCA Designer: A Rapid Design and Simulation Tool for Quantum-dot Cellular Automata", *IEEE Trans. on Nano.*, 3(1), 2004, pp. 26-31.

[36] M.S. Waterman and J.R. Griggs, "Interval Graphs and Maps of DNA," *Bull. Math. Biol.*, 48(2), 1986, p. 189-195.

[37] H. Yang and D.F. Wong, "New algorithms for min-cut replications in partitioned circuits," *Int'l Conf. on Comp.-Aided Des.*, 1995, pp.216-222.

