

Fabricatable Interconnect and Molecular QCA Circuits

Amitabh Chaudhary, Danny Ziyi Chen, *Senior Member, IEEE*, Xiaobo Sharon Hu, *Senior Member, IEEE*, Michael T. Niemier, *Member, IEEE*, Ramprasad Ravichandran, and Kevin Whitton

Abstract—When exploring computing elements made from technologies other than complementary metal–oxide–semiconductor, it is imperative to investigate circuits and systems assuming realistic physical implementation constraints. This paper looks at molecular quantum-dot cellular automata (QCA) devices within this context. With molecular QCA, physical coplanar wire crossings may be very difficult to fabricate in the near to midterm. Here, we consider how this will affect interconnect. We introduce a novel technique to remove wire crossings in a given design in order to facilitate the self-assembly of real circuits—thus, providing meaningful and functional design targets for both physical and computer scientists. The proposed methodology eliminates all wire crossings with minimal logic gate/node duplications. Simulation results based on existing QCA circuits and other benchmarks are presented, and suggest that further investigation is needed.

Index Terms—Bipartite graphs, crossing elimination, molecular electronics, quantum-dot cellular automata (QCA), shortest path.

I. INTRODUCTION

THE 2005 edition of the International Technology Roadmap for Semiconductors stresses that in order to achieve long-term MOSFET scaling (2014–2020, $L_g = 15$ nm), experimental advances relating to physical devices are clearly needed [1]. For silicon-based systems, much of this experimental work is being done with the belief that technology requirements for which there are currently “no known solutions” will ultimately be achieved. However, another research track is currently working to address a second possibility—one where red areas of the roadmap will *never* be achieved, and any workaround will indeed be irrelevant. Much of this work has focused on computational models and emerging technologies that are not based on CMOS MOSFETS, but can still offer their own unique “wins.”

Manuscript received July 13, 2006; revised December 10, 2006. The work of D. Z. Chen was supported in part by the U.S. National Science Foundation under Grant CCF-0515203. The work of X. S. Hu was supported in part by the U.S. National Science Foundation under Grant CNS-0410771. The work of M. T. Niemier was supported in part by the U.S. National Science Foundation under Grant CPA-0541324. This paper was recommended by Associate Editor S. Vrudhula.

A. Chaudhary, D. Z. Chen, X. S. Hu, and M. T. Niemier are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: mniemier@nd.edu).

R. Ravichandran is with Carnegie Mellon University, Pittsburgh, PA 15213 USA.

K. Whitton is with Motorola, Inc., Schaumburg, IL 60196 USA.

Digital Object Identifier 10.1109/TCAD.2007.906467

Regardless of technology, a successful electronic device must address the following.

- 1) *Interconnect*: A natural way to connect each stage of processing with subsequent stages is needed. For FETs, this becomes particularly challenging at the molecular scale [2].
- 2) *Power gain*: Digital computing requires true power gain. Unavoidable energy losses in each stage must be made up or a signal cannot continue to drive subsequent stages. Power gain is one feature that has made FETs so resilient and attractive.
- 3) *Low power dissipation*: As was also seen in [2], ultralow power dissipation is necessary at molecular densities to avoid catastrophic overheating.
- 4) *Defect tolerance*: Some randomness in the positioning and orientation of individual molecules is thermodynamically unavoidable. Defect tolerance at any level will be required.
- 5) *Appropriate nanosystem architectures*: We need to learn how to exploit new molecular-scale devices. Failure to match architecture to device characteristics could squander potential gains.

This paper addresses *interconnect* for circuits and systems of molecular quantum-dot cellular automata (QCA) cells. QCA accomplishes logical operations and moves data via nearest-neighbor interactions rather than with electric current flow. Four different implementations are possible. Both initial and current experiments have focused on a metal-dot implementation of a QCA device [3]–[5]. Semiconductor-based QCA devices have also been realized and should operate at significantly higher temperatures than metal-dot devices [6]–[9]. *Experiments* have shown that a molecule is capable of switching between configurations that could represent binary 0 and 1 states. A theoretical analysis of a molecular implementation has shown that QCA-based circuits: 1) could potentially be clocked at an extremely high frequency (adiabatically at 1 THz [10]–[12]); 2) could potentially lead to circuits with densities that are three to four orders of magnitude beyond what end-of-the-curve CMOS can provide [13], [14]; and 3) should dissipate very little power [10]. In a magnetic implementation, device switching times are slower. However, nanoscale magnets offer the advantages of large energy differences between states, tolerance to the effects of radiation, and the demonstrated room temperature operation of a functionally complete logic set [15]–[17]. Applications should require no power other than power required to switch their state [18].

Because nearest neighbor interactions are strongly distance dependent, the smaller that QCA devices can be made, the better they work (hence, the potential promise of molecular QCA devices). Additionally, information transmission and processing is carried out by the same entities—QCA cells—rather than by separate devices and wires. In CMOS circuitry, the preponderance of real estate on the chip is taken up by *wires*, so shrinking CMOS devices does not necessarily produce corresponding gains in *device* density. When smaller QCA cells are made, QCA interconnect shrinks too.

That said, CMOS does have many features that will cause both industry and academia to pursue scaling for the foreseeable future. The robustness of individual devices, relatively high fabrication yields, and existing infrastructure are just some of the characteristics that any new technology will need to eventually mimic. Additionally, with regard to interconnect, although wire delays [19] and pitches may eventually limit scaling, the multiple layers of interconnect associated with CMOS have allowed designers to cross signals and wire up different parts of a circuit with relative ease. Although providing similar functionality has not been the most pressing focus for emerging nanoscale devices, many should still be able to promise at least one additional circuit “layer” for crossings (as efficient methods are essential to continue scaling at the system level). For example, systems of nanowires and nanotubes are orthogonal to each other and can leverage CMOS-like vias to cross signals. Some implementations of QCA may also have a natural crossover. For semiconductor-based QCA devices, crossovers could be achieved with simple metal lines. By capacitively coupling one end of the line to one QCA cell, and the other end to another QCA cell, the value of one cell could be transferred to another [20]. With nanomagnets, perpendicular, out-of-plane magnetization using Co–Pt multilayer structures could facilitate crossings [18].

When considering molecular QCA, certain circuit substrates may make wire crossings difficult to fabricate (particularly in the near to midterm) and, hence, undesirable (see Sections II and III for more details). Thus, to help facilitate the *fabrication* of computationally interesting circuit designs, 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 also provide an initial comparison of near-to-midterm QCA designs to CMOS designs, as well as to QCA designs that assume fabricatable wire crossings.

We do not consider duplication to be the “be all, end all” solution to this problem. In Section VII, we will discuss how duplication can be used in *conjunction* with other methods for handling the crossing problem and can result in very implementable circuit structures that could simultaneously provide significant performance wins over end-of-the-roadmap CMOS.

In this paper, we will define the node-duplication-based crossing elimination (NDCE) problem. We will formulate the efficient elimination of wire crossings as an optimization problem and present a heuristic for solving it. Our heuristic iterates over the “layers” of the input circuit, solving in each iteration one of two subproblems. These subproblems are essentially the versions of the crossing elimination problem restricted to two-

layered circuits. We then present an integer linear programming (ILP) formulation for the first subproblem, and a linear-time algorithm for optimally solving the second subproblem, along with a time analysis and a detailed correctness proof of the algorithm.

In addition to easing implementable interconnect in all QCA systems,¹ eliminating wire crossings also finds applications in the physical synthesis of binary-decision-diagram (BDD)-based regular circuit structures. Non-crossing ordered BDD (NCOBDD) were first introduced by Cao and Koh [21] as a viable alternative to overcome the timing closure problem and offset the process variation in aggressively scaled silicon technology [22]. 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 a field-programmable gate array (FPGA) or pass transistor logic circuit. Such designs allow precise determination of delays and reduce crosstalk effects [21].

The work presented here is based on our previous work in [13]. However, in addition to a more detailed discussion of experiments and potential physical implementations, this paper contains the complete proofs and details for efficient implementation of the algorithm for the layer propagation problem. Specifically, it proves the relationship between sharings and duplications (Lemma 1), the correctness of the algorithm (Theorem 1), and it lays out the linear time implementation (Section V-C).

The rest of this paper is organized as follows. In Section II, we first review the basics of QCA circuits and a plausible molecular implementation of QCA. This will explain why our work in computer-aided design is important, and how it is directly tied to an implementation target. To the best of our knowledge, no existing work in QCA design has such a close mapping. In Section III, we will address ways to avoid crossings in a circuit (Section III-A), introduce the focus of this paper (duplication) and discuss related work (Section III-B), and define our algorithmic objectives for the NDCE problem (Section III-C). In Section IV, we discuss an ILP formulation of the two-layer NDCE problem (to address the first subproblem), and in Section V, we present a linear time algorithm to solve the second subproblem. Section VI considers a time analysis of our algorithm, as well as how duplication will affect circuit area. In Section VII, we illustrate by example how the methods developed here can be used in conjunction with other methods to make circuit designs both more efficient and easier to implement. Section VIII provides some concluding remarks.

II. OVERVIEW OF MOLECULAR QCA

A. QCA Fundamentals

QCA represents information by encoding binary numbers into cells that have a bistable charge configuration. A QCA cell can consist of two or four “charge containers” (i.e., quantum

¹Despite the aforementioned methods for crossing signals in nonmolecular implementations of QCA, to date, none have been experimentally realized. Thus, this work should impact these implementations too—both in terms of allowing for faster circuit prototyping and in determining how important this structure actually is to performance (i.e., are indirect methods good enough?).

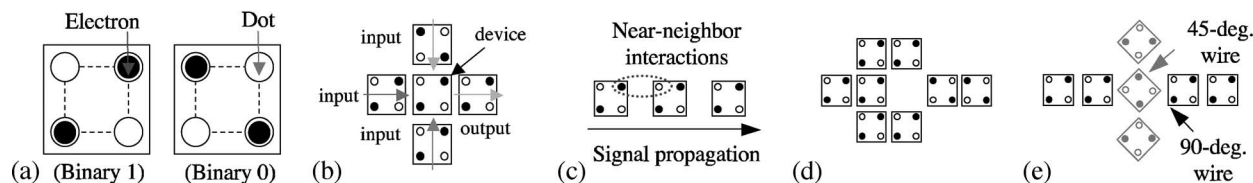


Fig. 1. Representations of (a) four-dot QCA cell, (b) majority gate, (c) wire, (d) inverter, and (e) physical wire crossing.

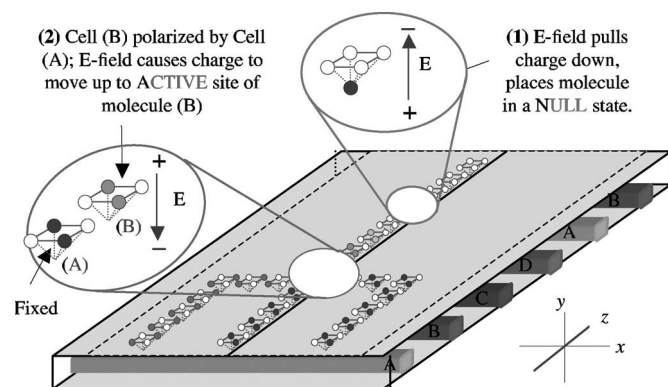


Fig. 2. Three-dimensional view of the proposed clock structure for charge-based QCA devices.

dots) and one or two excess charges, respectively. One configuration of charge represents a binary “1” and the other a binary “0” [Fig. 1(a)] [23]. Logical operations and data movement are accomplished via Coulomb (or nearest-neighbor) interactions. QCA cells interact because the charge configuration of one cell alters the charge configuration of the next cell.

A QCA wire [Fig. 1(c)] is just a line of QCA cells. The wire is driven at the input cell by a cell with a fixed/held polarization. The cells do not need to be exactly spaced the same distance apart. The majority gate [Fig. 1(b)] implements the logic function $AB + BC + AC$ (where A , B , and C are arbitrary inputs). The output cell assumes the polarization of the majority of the three input cells [23]. By setting one input of a majority gate to a logic “0” or “1,” the gate will execute an AND or OR function, respectively. An inverter can also be easily built with QCA devices [Fig. 1(d)]. QCA wires with different orientations [Fig. 1(e)] can theoretically cross in the plane without destroying the binary value on either wire.

For charge-based QCA, a physical manifestation of a clock will most likely take the form of lithographically defined conducting metal wires [24] (see Fig. 2). QCA devices will sit above the wires, and a ground plane would sit above the devices. The wires will produce an electric field that will help to induce data flow in the QCA logic and interconnect (see [24] and [25] for more detail). When a strong positive electric field is applied to a device, electrons will be drawn out of the active region of the cell as negatively charged electrons are attracted to a region of positive charge [see Fig. 2 inset (1)]. This puts the molecule in a NULL state where it will not participate in a computation or hold information. When the field becomes strongly negative, the electrons will be forced up into a *locked* state.

Which of the two upper “dots” are occupied is determined by a driving cell that splits the degeneracy and determines whether a QCA device is a 1 or 0 [see Fig. 2(a), inset (2)]. By applying time-varying voltages on the metal wires, we can “clock” the

cells’ activity to produce a flow of information. The required voltages are periodic, and adjacent wires have a $\pi/2$ phase shift between them. Every fourth wire will have the same applied signal. The four-phase signal on the buried wires should induce sinusoidal regions of activity (i.e., computation).

B. Molecular QCA

1) *Devices*: QCA devices (and circuits) not only have actually been made from metal [3]–[5], [26], [27], various semiconductor implementations [6]–[9], and nanomagnets [15], [16] but can also be made from a single chemical molecule. Molecular QCA devices can be made from mixed-valence compounds, which contain multiple redox centers in different oxidation states. Each “quantum dot” would be a single redox center, and the redox centers that make up a two- or four-dot QCA cell would be rigidly held together by covalent bonds [28]. A recent experiment demonstrates that applying reasonable electric fields can move a charge between two redox sites of a molecule engineered to function as a two-dot QCA cell [29]. This room-temperature experiment shows a self-assembled monolayer of molecules switching between configurations that could be used to chemically represent a binary 0 or 1. Four-dot QCA molecules have also been made [28], and promising input–output methodologies exist.

2) *Making Circuits*: The requirements for positioning molecular QCA cells are stringent. More than in any other implementation, individual molecules must be placed in specific locations with nanometer precision. Here, we will begin to explain exactly what precision is necessary to make QCA circuits and why the crossover construct will be physically hard to build.

Recent results in DNA tiling suggest that DNA could be used as a “circuit board” to position QCA cells. One viable target assumes DNA tiles fabricated by Winfree *et al.* [30] as the scaffolding. QCA molecules would covalently attach to modified DNA nucleotides. Assuming the tiles retain the B-DNA duplex, sites in the major groove could serve as attachment points with a periodicity of 3.6 Å along the helix axis, and 2 or 4 Å perpendicular to the helix axis (creating the potential for densities of $10^{13}/\text{cm}^2$). This paper has significant critical mass, as many research groups are currently looking at creating and attaching nanoparticles to DNA-based scaffoldings (i.e., [31]–[33]).

3) *Guided Self-Assembly for Systems*: DNA rafts with molecules can be “guided” to certain places to form more complex systems. Above, we discussed using chemical synthesis and bottom-up self-assembly to form molecular circuit boards. This section looks at how those circuit boards will be used to form systems. “Directed assembly” envisions using 10–100-nm top-down lithography to guide the attachment of circuit boards

to CMOS substrates. Interactions between the circuit boards will allow them to dock with specific partners, while maintaining subnanometer registry from one circuit board to the next.

However, [34] illustrates that silicon is a viable substrate for DNA tile binding if it is coated with a cationic adhesion layer. (DNA has a negative charge associated with it. A cation is a group of atoms carrying a positive charge. Hence, there will be a mutual attraction and “stickiness.”) Results show that the small molecule 3'-aminopropyltrimethoxysilane (APTES) can be patterned by a process called *molecular liftoff* to make a sticky surface that will anchor DNA rafts. Liftoff is a process where silicon coated with polymethyl methacrylate (PMMA) is etched with electron beam lithography. APTES is deposited in the etched areas, and the PMMA is removed. DNA rafts are then deposited and stick to the APTES. They cannot be washed off. Experimental results show that rafts stick on the APTES, but not on the silicon [34]–[37].

These experiments were done to develop a method for binding DNA parts to a silicon substrate. This is particularly important as it provides not only a substrate to attach our circuit components to but also that substrate can simultaneously be used to implement the clock structure discussed earlier—i.e., only metal CMOS wires are required.

Thus, DNA has the *potential* to: 1) act as a circuit board; 2) be made large enough to hold computationally interesting circuit components; and 3) be built with reasonable yields. It should also be possible to enforce extremely rigid cell orientation on the DNA tile—which is good for a four-dot molecular QCA cell. There are good precedents for rigid placements of metal complexes on duplex DNA [38] (meaning that it should be possible to position QCA devices so that they all have the same orientation).

III. PROBLEM STATEMENTS

Although it appears to be possible to enforce cells with one specific rotation, creating a circuit part that has cells with two different rotations, on the same DNA tile or raft, is a much more difficult problem to experimentally solve. The theoretical layouts for wire crossings require subangstrom control over the position and orientation of the QCA cells—and we would need to selectively rotate nanometer-sized devices with 45° precision to build them. Although not impossible, our first implementation target will probably be a system that has cells with just one orientation. In addition, although we might eventually leverage other methods that can position molecular QCA devices with some degree of determinism (References [39]–[41] are also promising), we will realistically encounter the same problems. This has immediate consequences in that the coplanar wire crossing shown in Fig. 1(e) would not be available for any design. This assumption is the major motivation for the work presented here.

A. Ways to Avoid Crossings in a Design

One way to eliminate crossings in a given circuit is to duplicate logic gates such that 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 “prepones” all wire crossings by pushing them to the inputs of the circuit, which will (of course) result in more inputs. Both of the above consequences will most likely increase the overall circuit area.²

For small- to medium-sized circuits, we do not believe that any perceived increase in the number of QCA cells will have a significant effect on the circuit area. (This will be further discussed in Section VI.) More importantly, although a circuit may be larger and more complex, this is not because new circuit components have been added. Rather, copies of the same logic blocks are reused—and a chemist would only have to design each logic block once. For example, designing DNA rafts for deterministic interconnect in varying circuits would be a much more difficult task—important when the fabrication process is still maturing. That said, not surprisingly, as circuit size and complexity increases, duplication alone will not generate efficient and scalable circuits. However, when used in conjunction with other methods (i.e., logic and time), “no physical wire crossings” may not be a restriction. In fact, performance gains with regard to area and speed are possible for a wide variety of circuits and systems that have only one cell type—even when providing added defect tolerance [42].

B. Duplication

Here, we provide a method to address duplication—and will consider it in the context of a “hybrid” solution in Section VII. Generally speaking, gate-duplication-based wire-crossing elimination simply removes edge crossings in a directed graph through node duplication. We refer to this problem as NDCE problem. At first sight, the NDCE problem appears to be closely related to the crossing minimization (CM) problem on directed layered graphs that has been well studied in the graph drawing area [43]–[45] and arises in other applications such as visualization [43] and DNA mapping [46]. Liebers has presented a survey on planarizing graphs, which includes an annotated bibliography [47]. CM has also been widely used in very-large-scale-integration layout, e.g., [48]–[51] to name a few.

Although 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). Although, 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 fixed, the CM version is still NP-hard [45], 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 this paper.

²We can interface between the microscale and nanoscale by using a T-junction. The T-junction connects a path of QCA cells that leads into QCA logic circuitry to a device that permanently represents a “1” or “0.” Whether the “1” or the “0” serves as the input will be determined by the CMOS clock.

To achieve planarity of logic circuits designed with BDDs and multivalued decision diagrams (MDDs), Sasao and Butler have examined the necessary properties for BDDs and MDDs to be planar [52]. Specifically, they have shown that threshold functions, symmetric functions, and monotone increasing functions lead to planar diagrams. These functions can be useful in deriving certain logic circuits for a molecular 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 [53]–[55], in placement enhancement [56], [57], and in FPGA timing optimization [58], [59]. Yet, none of these take wire-crossing elimination into consideration.

To the best of our knowledge, the only work that examined problems similar to NDCE is the NCOBDD synthesis by Cao and Koh [21], [60]. In [21], an interesting, albeit exponential time, backtracking mechanism is proposed for optimal node duplication in a given ROBDD. An elegant decomposition technique is discussed in [60] to improve the efficiency of constructing NCOBDD. However, the work is limited to graphs in which each node has no more than two outgoing edges.

C. Algorithmically Addressing Duplication

Here, 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 II-B, QCA circuits function under the influence of a clock field; and as a circuit should more reliably function if input signals arrive at logic gates at about the same time, this naturally gives rise to QCA circuits being “layered.” If a circuit design has wires across nonadjacent clock zones, buffers must be inserted in order to guarantee correct functionality, similar to synchronous circuits [61], [62]. 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. 3(a)]. For modeling QCA or other circuit design applications, we assume that only V_k contains zero in-degree nodes. 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. 3 for an example. The objective is to minimize the number of inserted nodes. The NDCE problem is NP-hard.

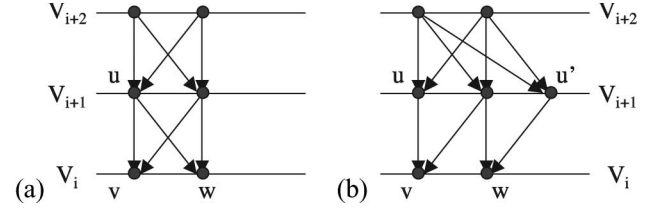


Fig. 3. Node duplication in V_{i+1} to avoid an edge crossing between V_i and V_{i+1} .

This way 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. 3(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.

Note 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 naturally propagate 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 *layer propagation*).

Note that in the above iterative algorithm, the edge crossing propagation goes “upward” (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: 1) determine an order of the nodes in V_1 and 2) determine a node order and possible node insertions for V_2 . As shown in Section IV, the two-layer case is NP-hard. In Section IV, we give an ILP formulation for this case. Since solving the ILP formulation is time consuming on large-size inputs, we also use a heuristic method to solve the two-layer case in polynomial time as follows.

- 1) We first choose a node order for V_1 (e.g., 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 .
- 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 shown in Section V, this problem can be optimally solved by a nontrivial linear time algorithm.

IV. ILP FORMULATION FOR THE TWO-LAYER NDCE PROBLEM

Given a directed two-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 two-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 [63] to the two-layer NDCE problem [64].

Based on the NP-hard conclusion, it is unlikely that polynomial time algorithms exist for exactly solving the two-layer NDCE problem. Here, we present an ILP formulation for the two-layer NDCE problem such that various ILP solvers can be exploited to solve the problem when the size of a two-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 noncrossing 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 $1 \leq i \leq |V_2|, 1 \leq j \leq |E|$. A straightforward way to model the edges needs $|V_1|^2 \cdot |V_2| \cdot |E|$ variables to represent 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 nonzero x_{ij} indicates the presence of a node in layer 2. If the sum of x_{ij} is greater than $|V_2|$, then 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:

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

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

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

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

Equations (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, whereas (7) specifies that each location j in layer 2 can have at most one node.

Additional constraints are needed concerning the edges. The first set of these are used to preserve the total number of edges connected to a node in layer 1 and layer 2, respectively, defined as

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

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

To ensure that edges in the resulting graph are always connected with the correct nodes, we introduce two additional constraints as

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

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

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 crossings occur in the resulting graph as shown in the following:

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

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. 4(a). 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

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

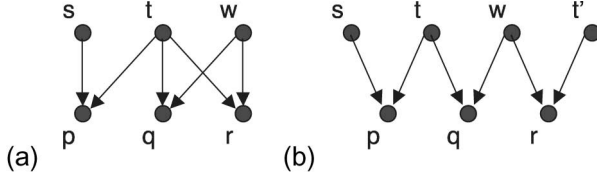


Fig. 4. (a) Two-layer input graph G . (b) Output graph G' of G .

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, e.g., location 1, we obtain the following constraint based on (9):

$$\sum_{j=1}^6 z_{sj1} + \sum_{j=1}^6 z_{tj1} + \sum_{j=1}^6 z_{wj1} = 2y_{p1} + 2y_{q1} + 2y_{r1}. \quad (15)$$

For a node in V_2 , e.g., 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 accordingly derived. 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

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

No-crossing constraints for other connections can be similarly constructed. Solving the ILP, we obtain the resulting graph as given in Fig. 4(b).

V. LAYER PROPAGATION ALGORITHM

In this section, we present an algorithm for the layer propagation problem, which was introduced in Section III-C. 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 the shortest path between two nodes in a graph of polynomial size in the size of the input and 2) we show how the 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 two-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 [i.e., as in Fig. 4(a) and (b)].

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 $f(u_i)$.

A. Partial Order on Edges and Sharings

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' . In addition, 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, the edges in $E(G)$ follow a partial order: given two edges (u_x, v_a) and (u_y, v_b) in $E(u')$, if $(a < b)$, then it must be that $(x \leq y)$.

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 . If in a solution σ_2 , v_i and v_{i+1} do have a common neighbor, we call it a *sharing*. The number of sharings in a solution σ_2 is the number of pairs of consecutive nodes in V_1 that share a common neighbor in V'_2 . For example, in Fig. 4(a) and (b), there are two sharings: 1) pair (p, q) share t and 2) pair (q, r) share w . We should clarify the notion of a sharing when there are nodes in V_1 of degree one. For example, suppose there are three consecutive nodes in V_1 , e.g., v_i , v_{i+1} , and v_{i+2} that are all adjacent to u in σ_2 . This implies that v_{i+1} has degree one. These nodes contribute two sharings: pair (v_i, v_{i+1}) and pair (v_{i+1}, v_{i+2}) , both pairs sharing u . We state and prove the following simple lemma that establishes the utility of the number of sharings.

Lemma 1: The minimum number of duplications in a solution to the layer propagation problem is $m - n_2 - s^*$, where s^* is the maximum number of sharings in a solution to the layer propagation problem.

Proof: We first prove that in every solution to the layer propagation problem, the number of duplications is $m - n_2 - s$, where s is the number of sharings in the solution; the theorem then follows. The solution to the layer propagation problem that has the maximum possible number of duplications has a distinct node in V'_2 corresponding to each edge in $E(G)$. The number of duplications in it is $m - n_2$. To reduce the number of duplications, there has to be a node in V'_2 that is the common neighbor of two nodes in V_1 , which, because of the partial order respected by the edges, have to be consecutive. (Remember, we assume there are no zero in-degree nodes in V_1 .) In general, for any solution that has d duplications where $d < m - n_2$, there have to be $m - n_2 - d$ instances of a node

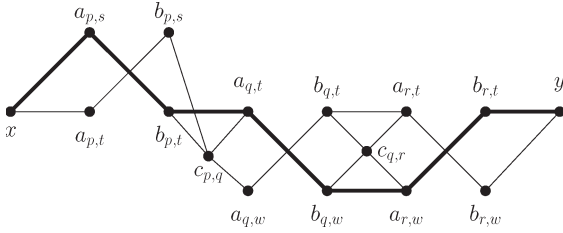


Fig. 5. Graph H for the problem in Fig. 4(a). The shortest path is in bold.

in V'_2 being a common neighbor of two consecutive nodes in V_1 , i.e., there are $m - n_2 - d$ sharings. ■

As an example, in Fig. 4(a) and (b), $m = 6$, $n_2 = 3$, $d = 1$, and $s = 2$, which is consistent with the theorem. The theorem implies that to minimize the number of duplications, we need to maximize the number of sharings.

B. Shortest Paths

We now define a graph H such that computing the maximum number of sharings s^* in G' corresponds to computing the 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 example, for the graph in Fig. 4(a), 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. 5. 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. 5]. 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. 5).

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 , e.g., (u, v_i) , then there is an edge between $a_{v_i,u}$ and $b_{v_i,u}$. If, however, v_i has two 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 exactly solve the layer propagation problem as stated in the following theorem.

Theorem 1: The optimal solution for the layer propagation problem on G can be determined from the shortest x – y path in H . The minimum number of duplications in a solution to the layer propagation problem on G is $l^* + m - 3n_1 - n_2$, where l^* is the shortest x – y path in H .

Proof: We begin by proving the following two statements.

- 1) For every solution σ_2 with s sharings for the layer propagation problem on G , there is a corresponding x – y path in H of length $3n_1 - s$.
- 2) For every x – y path in H of length l , there is a solution σ_2 with $3n_1 - l$ sharings for the layer propagation problem on G .

To prove Statement 1, let σ_2 be $u_1, \dots, u_{m'}$, where $u_i \in V'_2$. The partial order on the edges implies that we can construct a sequence of subsequences from σ_2 of the form

$$(u_{j_{1,1}}, \dots, u_{j_{1,k_1}}), (u_{j_{2,1}}, \dots, u_{j_{2,k_2}}), \dots, (u_{j_{n_1,1}}, \dots, u_{j_{n_1,k_{n_1}}})$$

where $u_{j_i,h}$ is the h th neighbor of i th node v_i in σ_1 . We use k_i to denote the degree of v_i . Note that all the $u_{j_i,h}$ are different for different values of h , but u_{j_i,k_i} may or may not be distinct from $u_{j_{i+1},1}$. Begin constructing the corresponding x – y path in H by adding the edge $(x, a_{v_1,u_{j_{1,1}}})$. For the vertex v_1 , first add the edge $(a_{v_1,u_{j_{1,1}}}, b_{v_1,u_{j_{1,k_1}}})$. Now, if v_1 and v_2 share the node $u_{j_{1,k_1}} = u_{j_{2,1}}$, add the edge $(b_{v_1,u_{j_{1,k_1}}}, a_{v_2,u_{j_{2,1}}})$, or else add the two edges $(b_{v_1,u_{j_{1,k_1}}}, c_{v_1,v_2})$ and $(c_{v_1,v_2}, a_{v_2,u_{j_{2,1}}})$, and so on for each v_i . (If at any point v_i has degree one, we add the corresponding edges, as in the regular case, except here $u_{j_{i,1}} = u_{j_i,k_i}$.) Finally, add the edge $(b_{v_{n_1},u_{j_{n_1,k_{n_1}}}}, y)$. The path has two edges adjacent to x and y , and one edge to connect a node in A_{v_i} to a node in B_{v_i} for every v_i in σ_1 . In addition, there is an edge between a node in B_{v_i} and $A_{v_{i+1}}$ if they have a sharing, and two edges otherwise. Thus, the length of the path is $2 + n_1 + s + 2(n_1 - 1 - s) = 3n_1 - s$, as required.

To prove Statement 2, let the x – y path be $x, a_{v_1,u_{j_{1,1}}}, b_{v_1,u_{j_{1,k_1}}}, a_{v_2,u_{j_{2,1}}}, \dots, b_{v_{n_1},u_{j_{n_1,k_{n_1}}}}, y$. We use this to construct the corresponding sequence of subsequences. For the first subsequence (for node v_1), we know the first and last nodes: $u_{j_{1,1}}$ and $u_{j_{1,k_1}}$. We add all the remaining neighbors of v_1 within the subsequence in any order. Similarly, we can construct the entire sequence of subsequences: $(u_{j_{1,1}}, \dots, u_{j_{1,k_1}}), \dots, (u_{j_{n_1,1}}, \dots, u_{j_{n_1,k_{n_1}}})$. To construct σ_2 from this, we use the same order of vertices, except that whenever there is a sharing, i.e., $u_{j_i,k_i} = u_{j_{i+1},1}$, we just list the node once. Using a similar argument as for Statement 1, we can show that the number of sharings is $3n_1 - l$.

From the above two statements, it follows that the length l^* of the shortest x – y path in H is $3n_1 - s^*$, where s^* is the maximum number of sharings in a solution to the layer propagation problem on G . From Lemma 1, it follows that the minimum number of duplications in a solution to the layer propagation problem is $l^* + m - 3n_1 - n_2$. From the proof for Statement 2, we can use the shortest x – y path in H to get a solution σ_2 for the layer propagation problem in G with the minimum number of duplications. ■

C. Linear Time Implementation

We now discuss the techniques that allow us to compute the shortest x – y path in H in time that is linear in the size of G . We shall compute the shortest x – y path without

explicitly creating and visiting all the edges of H . We do this by computing the distance-from- x value $d(z)$ of each node $z \in V(H)$ in steps based on the sequence σ_1 (that is, we make H a directed acyclic graph based on σ_1). For each $v_i \in \sigma_1$, we first compute the distances for nodes in A_{v_i} , followed by distances for nodes in B_{v_i} , and then move to nodes corresponding to v_{i+1} .

Assume that we have computed the distances for nodes corresponding to v_{i-1} and earlier. Now $d(a_{v_i,u})$ is the minimum of $d(b_{v_{i-1},u}) + 1$ (if $b_{v_{i-1},u}$ exists) and $\min_w d(b_{v_{i-1},w}) + 2$. The value $\min_w d(b_{v_{i-1},w})$ can be easily computed by scanning through the values for $B_{v_{i-1}}$. To determine whether $b_{v_{i-1},u}$ exists, we use links that are created in the following preprocessing step. For every $u \in V_2$, sort all edges in G such that $(u, v_i) < (u, v_j)$ if $i < j$. This is done by the radix sort, in linear time. Next, we scan through this sorted list then, for every consecutive pair of the form $(u, v_{i-1}), (u, v_i)$, add an edge between $b_{v_{i-1},u}$ and $a_{v_i,u}$. This preprocessing takes time linear in m .

Then, we compute the distances for nodes in B_{v_i} . Let k denote $|A_{v_i}| > 1$ (the case of $|A_{v_i}| = 1$ is trivial), and $U \subseteq V_2$ be the neighbors of v_i in G . $d(b_{v_i,u})$ is $\min_w d(a_{v_i,w}) + 1$, where $w \in U \setminus \{u\}$. We order the nodes in A_{v_i} in an arbitrary order in which $a_{v_i,u}$ has a rank, e.g., l . Let p (respectively, q) be the minimum distance value among all nodes in A_{v_i} with ranks less (respectively, greater) than l . Then, $d(b_{v_i,u})$ is $\min\{p, q\} + 1$. The “ p ” and “ q ” values for all nodes in A_{v_i} can be computed by two scans of A_{v_i} , i.e., one in increasing order and the other in decreasing order of A_{v_i} . Thus, this computation is also linear in m .

VI. SIMULATION RESULTS

In evaluating the proposed approach, we focus on two important aspects: 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.

A. Efficiency and Execution Time

Our algorithms are effective and efficient in solving the NDCE problem. 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 [65], referred to as S12) as well as a subset of circuits in the IEEE International Symposium on Circuits and Systems (ISCAS) benchmark suites (digital combinational logic, as discussed in [61], [66]). The results of this paper are summarized in Table I. 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 resulting no-crossing circuits, whereas

TABLE I
RESULTS OF APPLYING THE NDCE ALGORITHMS
(RUN ON A SUN ULTRA 20 WORKSTATION)

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	2,282	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

the last two columns summarize the running times for both the ILP-based and random-order-based NDCE algorithms. For some ILP instances, the problem sizes are too large to generate results within a reasonable time and “XX” is used to indicate this. For each random-order-based result, it is an average of ten runs, and the difference between the minimum and the maximum of each ten runs ranges between 3% and 20%.

From Table I, it can be concluded that our algorithms are effective in solving the NDCE problem, particularly 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 the best global solution. This scenario is evident in the C27 example, where the order randomly selected for the output layer turned out to be a better choice for the upper layers than the order determined by the two-layer ILP approach. Theoretically, it is possible to formulate the multilayer problem as an ILP as well. However, we feel that the resulting ILP instance would be impractically large and the formulation itself does not intellectually contribute much. Therefore, we decided not to pursue this direction.

The data in Table I also reveal that the running times for ILP solutions can be quite long. This is due to the cubic growth rate of the number of variables and constraints in the ILP formulation as the numbers of gates and wires in the first two layers increase. 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. Other methods such as a genetic algorithm for selecting the node order of the output layer deserve further study.

We have also compared our technique with the one in [21] for BDDs (see the last five circuits). Since the dynamic ordering used in generating the ROBDDs is not given in [21], we simply picked an arbitrary order to create the BDDs for the circuits in [21]. For most circuits, both methods give the same results. For some, our method is either slightly worse or better than that in [21]. (Note that the technique in [21] is not applicable to general circuits.)

B. Bounding Box Area and Logic Density

From a layout perspective, pure duplication produces both expected and surprising results. Not surprisingly, the number of

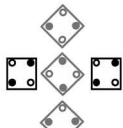
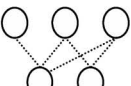
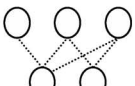
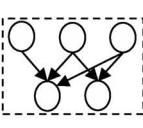
	 # of these constructs	 i.e. # of QCAs in the dotted lines	 i.e. # of QCAs in the "o"s (logic nodes)	Total number of QCA cells in a circuit: (Data generated by adding results of previous 2 columns)	 area of dashed line box (μm^2)
Simple 12 Control Logic					
Before	129	7685	1949	9634	1.62
After	0	4325	4508	8833	4.78
After/Before	1	0.563	2.31	0.917	2.95
ISCAS Circuit #1 (c17) (simple combinational logic)					
Before	1	159	195	354	0.019
After	0	217	228	445	0.029
After/Before	1	1.36	1.17	1.26	1.52
ISCAS Circuit #2 (c432) (control logic for a input controller)					
Before	1267	80141	6541	86655	8.82
After	0	18886	16942	35828	3.45
After/Before	1	0.236	2.60	0.413	0.391
ISCAS Circuit #3 (c880) (8 bit ALU)					
Before	3079	327237	27191	354428	62.6
After	0	980292	816039	1796331	1658.9
After/Before	1	3.00	30.0	5.07	26.5

Fig. 6. Impact of duplication on layouts—assuming the fabrication process discussed earlier.

nodes duplicated may rapidly grow as circuits become bigger. To examine the impact of such an increase, we discuss QCA layouts for several representative circuits.

As the focus of this paper is to evaluate how the nonuse of physical wire crossings will affect overall area and latency, we are less concerned with simulating our entire layouts at the logic and physical levels, and more concerned with overall area, the length of the critical path through the circuit, etc. That said, we do note that some subcircuits were simulated at the physical level with M-AQUINAS [42] and a statistical mechanical simulator [25], [67]. (M-AQUINAS assumes a wave-based clocking scheme suitable for a molecular implementation. These simulation methods are good in that they incorporate the physics of a clock that adds a helpful time component to the simulation. That said, M-AQUINAS only allows a QCA device to be in one of two states (a binary 1 or 0). With a molecular implementation, any combination of the four redox sites may be reduced or oxidized—resulting in six possible active states. The simulation methods in [25] account for these.)

A QCA place and route (PR) tool was also developed and used to convert the graph representations to QCA layouts. This includes replacing nodes with relevant logic where appropriate. The PR algorithm aims to obtain a dense QCA layout. We assume 3-nm center-to-center spacings and a three-cell “pitch” between QCA wires. (Although it may be possible to space wires closer, both M-AQUINAS and statistical mechanics show that there will be no crosstalk between parallel wires with a three-cell pitch.)

To estimate the area of the circuits with wire crossings, we assume that the wire-crossing implementation shown in Fig. 1(e) was used. As the clock wire pitch is a function of a given lithographic process, it is reasonable to say that the

best way to compare the circuits before and after duplication is to consider just the ratio of the number of cells in a circuit and the ratio of the area of a bounding box. Therefore, we can provide an initial comparison to designs that use all of the possible theoretical constructs, i.e., wire crossings, to a design that assumes the realistic timeline of a specific and viable fabrication process.

We summarize our results in Fig. 6. For each circuit, data before and after duplication as well as the ratio between the two are given in adjacent rows. Column 2 reports the number of wire crossings from a QCA layout via our PR tool. Column 3 refers to the number of QCA cells in the interconnect (anything not in a logic node). Column 4 lists the number of QCA cells in the logic nodes. Column 5 lists the total number of QCA cells in the design. Column 6 reports the bounding box area in squared micrometers (assuming 3-nm center-to-center cell spacing).

These data show that for relatively small circuits, node duplication can be a viable approach in eliminating wire crossings. For the Simple12 control path, duplication increases the area by a factor of 3. However, the number of QCA cells required to construct this circuit has gone down (good when one considers fabrication and possible defects). Most importantly, the number of cells required to connect logic nodes has almost been cut in half. Long and convoluted wires that existed preduplication have been replaced by shorter local wires in the interconnect post duplication (good when one considers mapping cells/parts to DNA). This explains the drop in the number of QCA cells in the interconnect.

Fig. 7 illustrates some of these trends. The block diagram in Fig. 7(a) represents our Simple12 control logic before our heuristics were applied. Boxes represent groups of AND, OR, or majority gates, whereas lines represent interconnect. As one

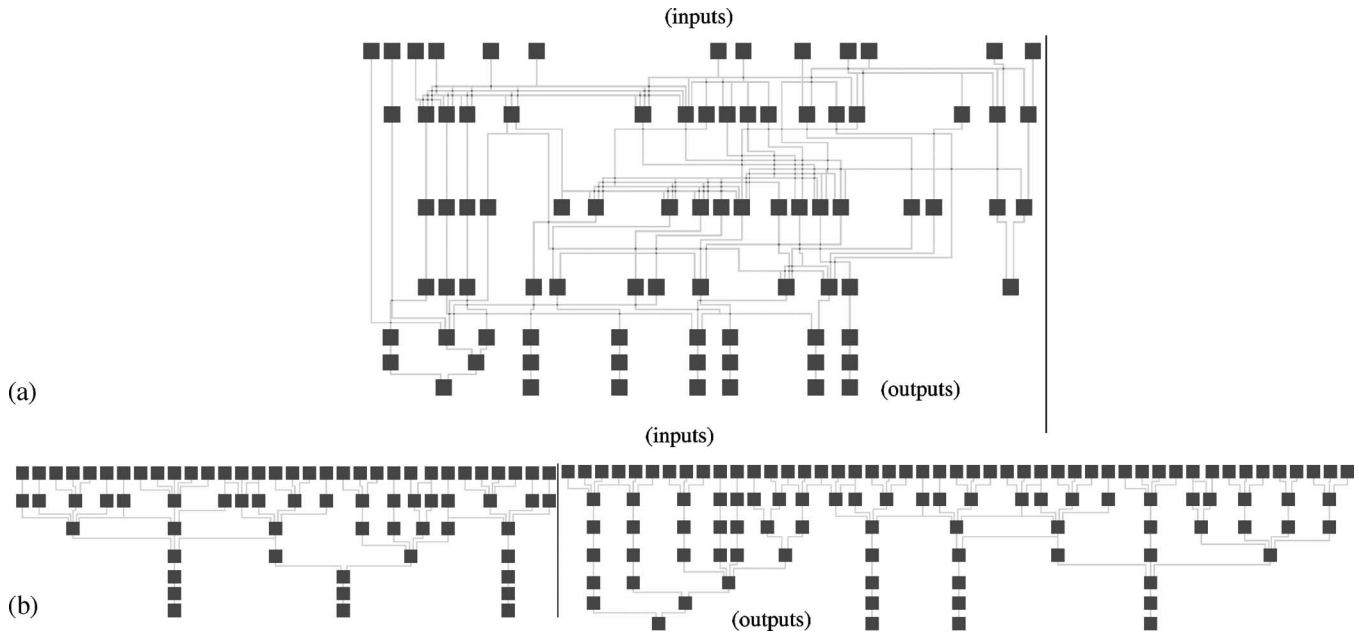


Fig. 7. S12 benchmark (a) before duplication and (b) after duplication (internal signal—with open path to output—is also a logical output and not shown).

can see, there are many long and convoluted wires in this circuit that would require a highly deterministic fabrication process to physically make. The block diagram in Fig. 7(b) represents the same circuit after our heuristics were applied. Highly deterministic interconnect has been all but eliminated—although overall area has grown. One of the reasons that circuit area quickly grows is that with duplication alone, to eliminate the last remaining crossings, we must duplicate larger parts of the current circuit. We will discuss ways to avoid this problem in Section VII.

For c432 (control logic for an input controller), again, the number of cells required has dropped by a factor of 4, and by a factor of 2.6 overall. Additionally, the total area required for this circuit has actually decreased by a factor of almost 2. We believe that this occurs because wire crossings have been eliminated. To implement a wire crossing, wires with two different orientations are required. Thus, if the two signals that were initially on 90° wires needed to cross, one value would have to be ripped onto another wire—and any new 45° wires must incur successive three-cell pitch separations—causing area to grow.³

To get a feel for how molecular QCA circuits compare with future CMOS circuits, we estimated the area of the CMOS implementation of c432 based on the standard-cell design given in [68]. We used $\lambda = 25$ nm for a hypothetical 45-nm technology. The CMOS design would take $7812 \mu\text{m}^2$, which is more than 2000 times larger than the no-crossing QCA design. (We readily

admit that this is only the first comparison, and further study is needed. However, the initial results are encouraging.)

Unfortunately, for more complex circuits, the above trends do not hold. For the c880 example (an 8-bit arithmetic logic unit), 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, whereas 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 again, we must duplicate larger and larger parts of the circuit. This is somewhat expected since, theoretically, removing wire crossings via node duplication can cause the number of duplications to exponentially grow.

VII. DISCUSSION AND CASE STUDY

This paper will form the foundation for a more manageable and efficient method for avoiding physical wire crossings—namely, *selective* duplication combined with logical crossings. To provide a better feel for how each method (duplication and logical crossings) will impact circuit layout, we consider a 1-bit full adder. A schematic of a majority gate-based adder appears in Fig. 8. As one can see, there are three crossovers that would precede each bit slice. We will consider how we can remove these physical crossovers using only duplication, only logical crossings, and a combination of duplication and logical crossings. We will comment on area and latency.

Fig. 9(a) depicts how we can eliminate crossings using only duplication. However, as seen in Section VI, this can easily become unwieldy—more and more of the circuit will have to be duplicated as successive bit slices are added. However, a combination of three XOR gates [see Fig. 10(a)] can also

³We do not specifically examine recurrent connections in this paper. However, these could be handled by solving the duplication problem in a hierarchical manner—i.e., finding solutions for the lower level modules, and then replacing these modules by simple nodes. There may be some complications due to the ordering of the input and output of different instances of the same module. We will examine this in our future work. As will be seen, using logical crossings can also help.

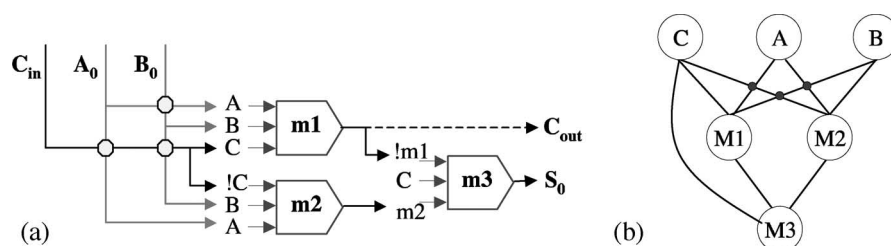


Fig. 8. (a) Adder based on majority gates and (b) its graph representation.

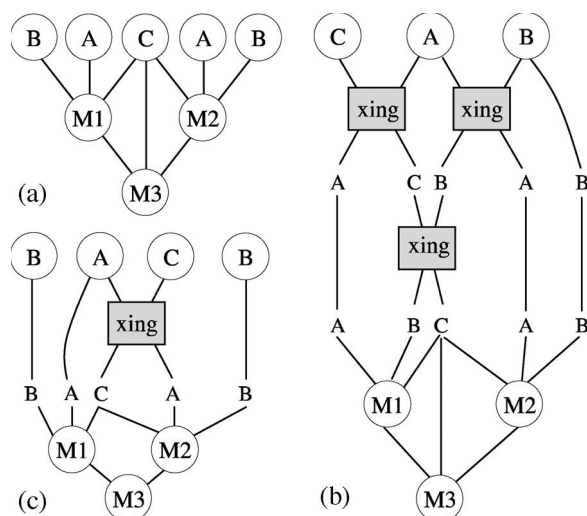


Fig. 9. (a) Graph representation of using duplication to eliminate the crossings in Fig. 8(b). (b) Logical crossings are used to eliminate physical crossings. (c) Logical crossings and duplication are used to eliminate physical crossings.

logically swap two bits. If each XOR gate is implemented with a combination of NAND, AND, and OR gates [see Fig. 10(b)], the resulting circuit will be completely planar and require just one device orientation.⁴

Fig. 9(b) shows how logical crossings might be used to eliminate the crossings in the graph representation of Fig. 8(b). That said, each logical crossing requires 12 majority gates and has 8 majority gates on the critical path through it. Thus, this design adds 16 gate delays to each bit slice of our adder. Additionally, these gates do not accomplish any processing, they only reposition data signals.

This delay could be cut in half by combining duplication and logical crossings. Instead of using only logical crossings, we could duplicate the B input and use one logical crossing, as seen in Fig. 9(c). This removes one logical crossing and eight gate delays from the critical path of the circuit. Although this method would add area due to wire routing, additional logical crossings [as in Fig. 9(b)] also add area and increase latency. Evaluating combinations of these methods is the subject of future work.

We note that here we discuss latency more in the context of gate delay rather than execution time. Device switching times lie in electron transfer rates in perturbed mixed valence

compounds (the class of molecules from which QCA cells have been derived). A clocked QCA device is essentially a perturbed system—the electric field induces switching. In a perturbed system, switching times between 10^{-13} and 10^{-15} s are reported [11], [12]. Additionally, according to [10], clock rates of greater than 1 THz are theoretically possible.⁵ That said, none of these structures have been fabricated and switching times and/or clock rates may be slower. Thus, we do not want to exaggerate the potential performance of a specific circuit. On the other hand, fewer devices on the critical path will most certainly translate to lower latency.

In general, removing crossings with node duplication could increase the lengths of certain interconnects and, thus, impact latency. However, because QCA circuits (both computation nodes and interconnects) are inherently pipelined due to the use of clock structure [65], increasing interconnect lengths do not need to increase the throughput rate of a QCA circuit. Hence, the method introduced here is applicable to larger circuits.

VIII. CONCLUDING REMARK

In this paper, we have discussed in detail one possible way to physically implement circuits made from QCA cells. Although it is not the only implementation mechanism possible, it is quite possible that it will be one of the firsts. We also highlighted that performance wins were possible with regard to area, speed, and power for the technology. Clearly, even with a “restrictive” fabrication process, systems made from molecular QCA devices are worth pursuing.

Nevertheless, if we are to move toward useful circuits and systems, we must overcome the potential wire-crossing constraint. In this paper, we have presented one approach for removing all wire crossings from a molecular QCA circuit or system schematic. We reiterate here that we never intended this method to, alone, be a “be all, end all” solution to this physical constraint. As we expected, pure duplication can cause an exponential increase in the number of gates required for certain types of circuits. However, it also produces several positive side effects as well.

- 1) Interconnect is greatly simplified and, deterministic interconnect paths are all but eliminated.
- 2) Fewer “parts” will have to be mapped to DNA. Those that are can be frequently reused.

⁴Components needed for molecular implementations of this construct have been verified with M-AQUINAS and the statistical mechanical simulator and function as intended. Additionally, because molecules are approximately 1×1 nm, the area of a logical crossing should be less than that of a metal1–metal2 physical crossover in 22-nm CMOS.

⁵Does not adiabatic mean slow? For molecular QCA, a 1-THz clock is slow compared to the time required for one electron to switch from one side of a molecule to another; 10^{-12} is three orders of magnitude less than 10^{-15} for instance. Of course, the possible switching times will ultimately depend on the capabilities of the CMOS circuitry.

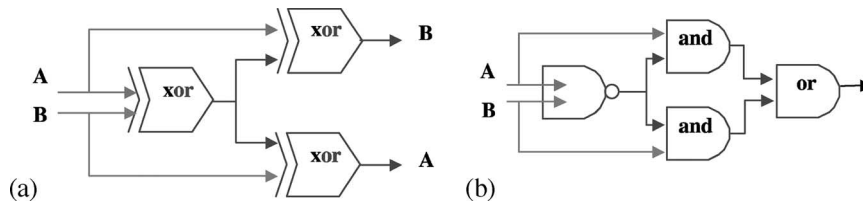


Fig. 10. (a) Crossing signals with three XOR gates. (b) Planar XOR gate.

- 3) For small- to medium-sized circuits, circuit area as well as the number of molecular QCA devices required to implement the circuit actually *decreases*.

Additionally, combining duplication with other methods that can also swap signals with just one device type appears promising. This will be the focus of our future work. We believe that the net effect of these studies will provide significant insight as to how important the wire crossing construct is to overall performance. Such insight will directly affect the experimental path for physical scientists—either entirely removing the wire crossing construct from it (and hence, devices with different rotations) or showing what is essential for system-level wins with this technology.

REFERENCES

- [1] *International Technology Roadmap for Semiconductors*, Dec. 2005. I. T. R. for Semiconductors, 2005 ITRS Update. [Online]. Available: <http://www.itrs.net/Common/2004Update/2004Update.htm>
- [2] C. S. Lent, M. Liu, and Y. Lu, "Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling," *Nanotechnology*, vol. 17, no. 16, pp. 4240–4251, Aug. 2006.
- [3] I. Amlani, A. Orlov, G. Snider, and C. Lent, "Demonstration of a functional quantum-dot cellular automata cell," *J. Vac. Sci. Technol. B, Microelectron. Process. Phenom.*, vol. 16, no. 6, pp. 3795–3799, Nov. 1998.
- [4] C. Lent, G. Snider, G. Bernstein, W. Porod, A. Orlov, M. Lieberman, T. Fehlner, M. Niemier, and P. Kogge, "Quantum-dot cellular automata," in *Electron Transport in Quantum Dots*, J. P. Bird, Ed. New York: Kluwer, 2003, pp. 397–433.
- [5] G. Snider, A. Orlov, I. Amlani, G. Bernstein, C. Lent, J. Merz, and W. Porod, "Quantum-dot cellular automata: Line and majority logic gate," *Jpn. J. Appl. Phys.*, vol. 38, no. 12B, pp. 7227–7229, Dec. 1999.
- [6] T. Vandervelde, P. Kumar, T. Kobayashi, J. Gray, T. Pernell, J. Floro, R. Hull, and J. Bean, "Growth of quantum fortress structures in $\text{Si}_{1-x}\text{Ge}_x/\text{Si}$ via combinatorial deposition," *Appl. Phys. Lett.*, vol. 83, no. 25, pp. 5205–5207, Dec. 2003.
- [7] M. Mitic, M. Cassidy, K. Petersson, R. Starrett, E. Gauja, R. Brenner, R. Clark, A. Dzura, C. Yang, and D. Jamieson, "Demonstration of a silicon-based quantum cellular automata cell," *Appl. Phys. Lett.*, vol. 89, no. 1, pp. 013503.1–013503.3, Jul. 5, 2006.
- [8] C. Smith, S. Gardelis, A. Rushforth, R. Crook, J. Cooper, D. Ritchie, E. Linfield, Y. Jin, and M. Pepper, "Realization of quantum-dot cellular automata using semiconductor quantum dots," *Superlattices Microstruct.*, vol. 34, no. 3–6, pp. 195–203, Sep.–Dec. 2003.
- [9] A. Fujiwara, S. Horiguchi, M. Nagase, and Y. Takahashi, "Threshold voltage of Si single-electron transistor," *Jpn. J. Appl. Phys. 1, Regul. Pap. Short Notes Rev. Pap.*, vol. 42, no. 4B, pp. 2429–2433, Apr. 2003.
- [10] J. Timler and C. Lent, "Power gain and dissipation in quantum-dot cellular automata," *J. Appl. Phys.*, vol. 91, no. 2, pp. 823–831, Jan. 2002.
- [11] P. Barbara, T. Meyer, and M. Ratner, "Contemporary issues in electron transfer research," *J. Phys. Chem.*, vol. 100, no. 31, pp. 13 148–13 168, Aug. 1996.
- [12] P. Perez-Tejeda, F. Sanchez-Burgos, and M. Galan, "Calculation of rate constants from UV–vis spectroscopic data: An application of the Marcus–Hush model," *J. Mol. Struct., Theochem*, vol. 371, pp. 153–160, 1996.
- [13] A. Chaudhary, D. Chen, X. Hu, M. Niemier, R. Ravichandran, and K. Whitton, "Eliminating wire crossings for molecular quantum-dot cellular automata implementation," in *Proc. ICCAD*, 2005, pp. 565–571.
- [14] M. Niemier, "The effects of a new technology on the design, organization, and architectures of computing systems," Ph.D. dissertation, Univ. Notre Dame, Notre Dame, IN, Jan. 2004.
- [15] R. Cowburn and M. Welland, "Room temperature magnetic quantum cellular automata," *Science*, vol. 287, no. 5457, pp. 1466–1468, Feb. 2000.
- [16] G. Bernstein, A. Imre, V. Metlushko, A. Orlov, L. Zhou, L. Ji, G. Csaba, and W. Porod, "Magnetic QCA systems," *Microelectron. J.*, vol. 36, no. 7, pp. 619–624, Jul. 2005.
- [17] A. Imre, G. Csaba, L. Ji, A. Orlov, G. Bernstein, and W. Porod, "Majority logic gate for magnetic quantum-dot cellular automata," *Science*, vol. 311, no. 5758, pp. 205–208, Jan. 13, 2006.
- [18] A. Imre, "Experimental study of nanomagnets for magnetic quantum-dot cellular automata (MQCA) logic applications," Ph.D. dissertation, Univ. Notre Dame, Notre Dame, IN, Apr. 2005.
- [19] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.
- [20] G. Snider, private communication, 2006.
- [21] A. Cao and C.-K. Koh, "Non-crossing OBDDs for mapping to regular circuit structures," in *Proc. Int. Conf. Comput. Des.*, 2003, pp. 338–343.
- [22] B. Mochocki, X. Hu, and G. Quan, "A realistic variable voltage scheduling model for real-time applications," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2002, pp. 726–731.
- [23] C. Lent and P. Tougaw, "A device architecture for computing with quantum dots," *Proc. IEEE*, vol. 85, no. 4, pp. 541–557, Apr. 1997.
- [24] K. Hennessy and C. Lent, "Clocking of molecular quantum-dot cellular automata," *J. Vac. Sci. Technol. B, Microelectron. Process. Phenom.*, vol. 19, no. 5, pp. 1752–1755, Sep.–Oct. 2001.
- [25] M. Niemier, M. Crocker, X. S. Hu, and M. Lieberman, "Using CAD to shape experiments in molecular QCA," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 8, 2006, pp. 907–914.
- [26] I. Amlani, A. Orlov, G. Toth, G. Bernstein, C. Lent, and G. Snider, "Digital logic gate using quantum-dot cellular automata," *Science*, vol. 284, no. 5412, pp. 289–291, Apr. 1999.
- [27] R. Kumamuru, J. Timler, G. Toth, C. Lent, R. Ramasubramaniam, A. Orlov, and G. Bernstein, "Power gain in a quantum-dot cellular automata latch," *Appl. Phys. Lett.*, vol. 81, no. 7, pp. 1332–1334, Aug. 2002.
- [28] J. Jiao, G. Long, F. Grandjean, A. Beatty, and T. 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. Amer. Chem. Soc.*, vol. 125, no. 25, pp. 1522–1523, Jun. 2003.
- [29] H. Qi, S. Sharma, Z. Li, G. Snider, A. Orlov, C. Lent, and T. Fehlner, "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. Amer. Chem. Soc.*, vol. 125, no. 49, pp. 15 250–15 259, Dec. 2003.
- [30] E. Winfree, F. Liu, L. Wenzler, and N. Seeman, "Design and self-assembly of two-dimensional DNA crystals," *Nature*, vol. 394, no. 6693, pp. 539–544, Aug. 1998.
- [31] T. Labeau, S. Park, S. Ahn, and J. Reif, "Stepwise DNA self-assembly of fixed-size nanostructures," in *Proc. Found. Nanoscience, Self-Assembled Archit., Devices*, 2005, pp. 179–181.
- [32] P. W. K. Rothmund, N. Papadakis, and E. Winfree, "Algorithmic self-assembly of DNA sierpinski triangles," *PLOS Biol.*, vol. 2, no. 12, pp. 2041–2053, Dec. 2004.
- [33] P. Rothmund, "Design of DNA origami," in *Proc. Int. Conf. Comput.-Aided Des.*, 2005, pp. 471–478.
- [34] W. Hu, G. Bernstein, K. Sarveswaran, and M. Lieberman, "Low temperature development of PMMA for sub-10-nm electron beam lithography," in *Proc. 3rd IEEE Conf. Nanotechnol.*, 2003, vol. 2, pp. 602–605.
- [35] Q. Hang, Y. Wang, M. Lieberman, and G. Bernstein, "Molecular patterning through high-resolution polymethylmethacrylate masks," *Appl. Phys. Lett.*, vol. 80, no. 22, pp. 4220–4222, Jun. 2002.

- [36] Q. Hang, Y. Wang, M. Lieberman, and G. Bernstein, "A liftoff technique for molecular nanopatterning," *J. Nanosci. Nanotechnol.*, vol. 3, no. 4, pp. 309–312, Aug. 2003.
- [37] W. Hu, K. Sarveswaran, M. Lieberman, and G. H. Bernstein, "High resolution electron beam lithography and DNA nano-patterning for molecular QCA," *IEEE Trans. Nanotechnol.*, vol. 4, no. 3, pp. 312–316, May 2005.
- [38] D. Hurley and Y. Tor, "Ru(II) and Os(II) nucleosides and oligonucleotides: Synthesis and properties," *J. Amer. Chem. Soc.*, vol. 124, no. 14, pp. 3749–3762, Apr. 2002.
- [39] M. de Wild, S. Berner, H. Suzuki, H. Yanagi, D. Schlettwein, S. Ivan, A. Baratoff, H.-J. Guentherodt, and T. Jung, "A novel route to molecular self-assembly: Self-intermixed monolayer phases," *ChemPhysChem*, vol. 3, no. 10, pp. 881–885, Oct. 2002.
- [40] M. Stoykovich, M. Muller, S. Kim, H. Solak, E. Edwards, J. J. de Pablo, and P. Nealey, "Directed assembly of block copolymer blends into non-regular device-oriented structures," *Science*, vol. 308, no. 5727, pp. 1442–1446, Jun. 3, 2005.
- [41] L. Demers, D. Ginger, S. Park *et al.*, "Direct patterning of modified oligonucleotides on metals and insulators by dip-pen nanolithography," *Science*, vol. 296, no. 5574, pp. 1836–1838, Jun. 7, 2002.
- [42] E. P. Blair, "Tools for the design and simulation of clocked molecular quantum-dot cellular automata circuit," M.S. thesis, Univ. Notre Dame, Notre Dame, IN, 2003.
- [43] G. D. Battista, P. Eades, R. Tamassia, and I. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ: Prentice-Hall, 1998.
- [44] V. Dujmovic and S. Whitesides, "An efficient fixed parameter tractable algorithm for 1-sided crossing minimization," *Algorithmica*, vol. 40, no. 1, pp. 15–31, Jun. 2004.
- [45] P. Eades and S. Whitesides, "Drawing graphs in two layers," *Theor. Comput. Sci.*, vol. 131, no. 2, pp. 361–374, Sep. 1994.
- [46] M. Waterman and J. Griggs, "Interval graphs and maps of DNA," *Bull. Math. Biol.*, vol. 48, no. 2, pp. 189–195, Mar. 1986.
- [47] A. Liebers, "Planarizing graphs—A survey and annotated bibliography," *J. Graph Algorithms Appl.*, vol. 5, no. 1, pp. 1–74, 2001.
- [48] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*. New York: Wiley, 1990.
- [49] H. Chen and D. Lee, "On crossing minimization problem," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 17, no. 5, pp. 406–418, May 1998.
- [50] M. Marek-Sadowska and M. Sarrafzadeh, "The crossing distribution problem," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 4, pp. 423–433, Apr. 1995.
- [51] X. Song and Y. Wang, "On the crossing distribution problem," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 4, no. 1, pp. 39–51, Jan. 1999.
- [52] T. Sasao and J. T. Butler, "Planar decision diagrams for multiple-valued functions," *Multi-Valued J.*, vol. 1, no. 1, pp. 39–64, 1996.
- [53] L. Liu, M. T. Kuo, C. Cheng, and T. Hu, "A replication cut for two-way partitioning," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 5, pp. 623–630, May 1995.
- [54] H. Yang and D. Wong, "New algorithms for min-cut replications in partitioned circuits," in *Proc. Int. Conf. Comput.-Aided Des.*, 1995, pp. 216–222.
- [55] M. Enos, S. Hauck, and M. Sarrafzadeh, "Replication for logic bipartitioning," in *Proc. Int. Conf. Comput.-Aided Des.*, 1997, pp. 342–349.
- [56] I. Neumann, D. Stoffel, H. Hartje, and W. Kuntz, "Cell replication and redundancy elimination during placement for cycle time optimization," in *Proc. Int. Conf. Comput.-Aided Des.*, 1999, pp. 25–30.
- [57] M. Hrkic, J. Lillis, and G. Beraudo, "An approach to placement-coupled logic replication," in *Proc. Des. Autom. Conf.*, 2004, pp. 711–716.
- [58] G. Beraudo and J. Lillis, "Timing optimization of FPGA placements by logic replication," in *Proc. Des. Autom. Conf.*, 2003, pp. 196–201.
- [59] W. Fang and A. Wu, "Performance-driven multi-FPGA partitioning using functional clustering and replication," in *Proc. Des. Autom. Conf.*, 1998, pp. 283–286.
- [60] A. Cao and C.-K. Koh, "Decomposition of BDDs with application to physical mapping of regular PTL circuits," in *Proc. Int. Workshop Logic Synthesis*, Temecula, CA, Jun. 2004, pp. 244–249.
- [61] D. Antonelli, D. Chen, T. Dysart, X. Hu, A. B. Khang, P. Kogge, R. Murphy, and M. Niemier, "Quantum-dot cellular automata (QCA) circuit partitioning: Problem modeling and solutions," in *Proc. Des. Autom. Conf.*, 2004, pp. 363–368.
- [62] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1, pp. 5–35, 1991.
- [63] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
- [64] M. Liu, "Robustness and power dissipation in quantum-dot cellular automata," Ph.D. dissertation, Univ. Notre Dame, Notre Dame, IN, 2006.
- [65] M. Niemier and P. Kogge, "Exploring and exploiting wire-level pipelining in emerging technologies," in *Proc. Int. Symp. Comput. Archit.*, 2001, pp. 166–177.
- [66] M. Hansen, H. Yalcin, and J. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test Comput.*, vol. 16, no. 3, pp. 72–80, Jul.–Sep. 1999.
- [67] Y. Wang and M. Lieberman, "Thermodynamic behavior of molecular-scale quantum-dot automata (QCA) wires and logic devices," *IEEE Trans. Nanotechnol.*, vol. 3, no. 3, pp. 368–376, Sep. 2004.
- [68] S. Khatri, A. Mehrotra, R. Brayton, A. Sangiovanni-Vincentelli, and R. Otten, "A novel VLSI layout fabric for deep sub-micron applications," in *Proc. Des. Autom. Conf.*, 1999, pp. 491–496.

Amitabh Chaudhary received the Ph.D. degree from Johns Hopkins University, Baltimore, MD, in 2002.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN. His research interests include graph theory and online algorithms.

Danny Ziyi Chen (M'93–SM'02) received the Ph.D. degree in computer science from Purdue University, West Lafayette, IN, in 1992.

Since 1992, he has been on the faculty of the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, where he is currently a Professor. He has published over 170 journal and conference papers. His main research interests include algorithm design, analysis, and implementation, computational geometry, parallel and distributed computing, computational medicine, data mining, robotics, and VLSI design.

Xiaobo Sharon Hu (S'85–M'89–SM'02) received the Ph.D. degree from Purdue University, West Lafayette, IN.

She is an Associate Professor with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN. She is currently an Associate Editor of the *ACM Transactions on Design Automation of Electronic Systems* and *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS*. Her research interests include real-time embedded systems, low-power system design, hardware-software codesign, VLSI and nanoscaling computing, and computational medicine.

Michael T. Niemier (S'00–M'03) received the Ph.D. degree from the University of Notre Dame, Notre Dame, IN, in 2004.

He is a Faculty Member with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN. His research interests include designing, facilitating, and evaluating architectures for emerging technologies with an emphasis on magnetic logic based on the QCA device architecture and self-assembling systems, as well as how technology scaling impacts computer architecture.

Ramprasad Ravichandran received the B.S. degree with high honors and the M.S. degree, with a focus in the intersection of nanotechnology and computing, from Georgia Institute of Technology, Atlanta. He is currently working toward the Ph.D. degree in the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

His current research interests include multiagent systems, machine learning, and game theory.

Kevin Whitton received the B.S. degree in computer engineering from Tri-State University, Angola, IN, in 2003 and the M.S. degree in computer engineering from the University of Notre Dame, Notre Dame, IN, in 2005.

After leaving Notre Dame in 2005, he joined Motorola, Inc., Schaumburg, IL, where he is currently a Senior Computer Engineer.