

Defect Tolerance in QCA-Based PLAs

Michael Crocker, X. Sharon Hu, and Michael Niemier
 Department of Computer Science and Engineering
 University of Notre Dame
 Notre Dame, IN 46556, USA
 Email: {mcrocker,shu,mniemier}@nd.edu

Abstract—Defect tolerance will be critical in any system with nano-scale feature sizes. This paper examines some fundamental aspects of defect tolerance for a reconfigurable system based on Magnetic Quantum-dot Cellular Automata (MQCA). MQCA performs logical operations and moves data by manipulating the polarizations of nano-scale magnets, has been experimentally demonstrated, and operates at room temperature. We consider how specific defects will impact device functionality. Within this context, we introduce techniques for mapping Boolean logic functions to a defective system architecture (a reconfigurable programmable logic array design for MQCA). Simulation results show that our new mapping techniques can achieve much higher yields than existing techniques for nanowire crossbar PLAs.

I. INTRODUCTION

At present, there are a number of research efforts that have focused on different devices that might either replace or augment CMOS technology such that the performance scaling trends that we have seen for the last 30 years – and expect to see for the next 10-15 years – might continue beyond the year 2020. The work presented here looks at the Quantum-dot Cellular Automata (QCA) device architecture – and more specifically a reconfigurable, systems-level architecture realized with a magnetic implementation of QCA devices. This work is based on the work in [12] and [5], but represents two important steps toward a realistic, computationally interesting system. First, we show via physical-level simulation what logical faults we can expect in a physically-realized QCA-based circuit. We consider how these faults will affect the functionality of a reconfigurable PLA and propose mapping techniques to improve overall yield.

QCA accomplishes logical operations and moves data via nearest-neighbor interactions rather than with electric current flow. A given implementation could potentially lead to fast and/or low power circuits with nanometer feature sizes. Different implementations are being researched and include devices based on a metal-dot structure [1], individual molecules [22], and semiconductor technology [17]. A magnetic implementation of QCA is also possible. For nanomagnet-based QCA (MQCA), wires, gates, and inverters, operating at room temperature, have all been experimentally realized and verified. It has been estimated that if 10^{10} nanomagnets are adiabatically switched 10^8 times each second, they should only dissipate approximately 0.1W of power [14]. MQCA will be considered in more detail as part of a case study to be presented later.

Of course, it is also well recognized that any circuit with nanometer feature sizes will almost certainly be more defective than what we have come to expect from previous generations

of CMOS-based circuits. Largely for this reason, many research groups studying emerging technologies have proposed specific reconfigurable logic structures and investigated means to map Boolean functions to redundant, reconfigurable architectures [6], [26], [24]. A programmable logic array (PLA) structure for QCA was introduced in [12] while QCA-specific defect and fault modeling was studied by the authors of [5].

The work presented in this paper forms a bridge between the efforts of [12] and [5] by providing important observations and necessary tools that are useful for more detailed and realistic estimations regarding whether or not QCA will ultimately offer performance wins at the systems-level. Specifically, we examine different types of faults that we would expect given a realized MQCA circuit. It will be shown that certain type of functional faults in QCA PLAs have not been encountered before and proper treatments of such faults can significantly impact PLA yields. Using these specific faults as context, we introduce two approaches to mapping Boolean logic functions to a faulty QCA-based PLA with redundant rows and columns. The approaches allow a tradeoff between mapping algorithm efficiency and resulting PLA yield.

We begin in Sec. II by discussing the experimental state of the art and the QCA-based PLA design. We review mapping techniques for nanowire crossbars in Sec. III as we leverage this work in our proposed mapping methodology. In Sec. IV, we illustrate what faults we might expect for MQCA using physical-level simulation. In Sec. V we present two different methodologies for mapping logic functions to the QCA-based PLA design. We present our results in Sec. VI and discuss future work in Sec. VII.

II. BACKGROUND

A. QCA Basics

The initial description of a QCA device called for encoding binary numbers into cells that have a bi-stable charge configuration. A QCA cell would consist of 2 or 4 “charge containers” (i.e. quantum dots) and 1 or 2 excess charges respectively. One configuration of charge represents a binary ‘1’ and the other a binary ‘0’ [16]. 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. In a magnetic implementation of QCA, charge configurations are replaced with magnetic polarizations of single domain magnets. We note that the rest of this section will be devoted to MQCA-based circuit building blocks in order to lay the foundation

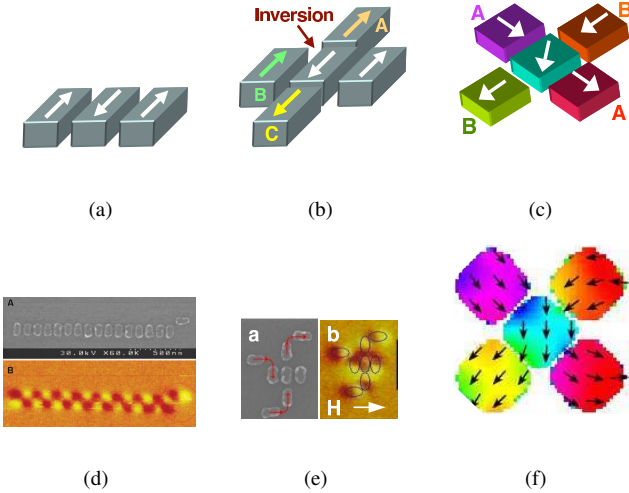


Fig. 1. Cartoon representations of a wire segment (a), a majority gate (b), and a crossover (c). Wire segments have been experimentally demonstrated (d) as have majority gates (e). Crossovers function correctly in simulation (f).

for the case study to be presented in Sec. IV. For more detail about electrostatic implementations, we refer the reader to [1], [22], [17], and [11].

Fig. 1 illustrates three important building blocks for MQCA circuits. A **wire** (Fig. 1a) is just a line of magnets that are antiferromagnetically coupled with each other. This structure has been experimentally demonstrated (Fig. 1d) and operates at room temperature [2], [4]. The basic logic **gate** in MQCA is based on the majority voting function – where the output is the logical value associated with the majority of the 3 input devices. By setting one input of a majority gate to a logic ‘0’ or ‘1’, the gate will execute an AND or OR function, respectively. Note that in MQCA, the gate is actually an *inverting* majority gate (Fig. 1b). This structure has also been experimentally demonstrated and operates at room temperature (see Fig. 1e, [14]). Structures have also been devised to cross two signals in the plane by leveraging magnets shaped so that they can represent two bits of information simultaneously (Fig. 1c) [19] – as the diamond-shaped magnets have no preferred magnetization and are readily influenced by the logical state of its neighbors. This structure has not yet been experimentally demonstrated, but no exotic shapes are required and the design has been verified via micromagnetic simulation [19].

B. PLA Design

By using the device architecture just discussed, a reconfigurable PLA architecture was introduced in [12]. We briefly review the core of this design here. Traditional MOSFET PLAs have been made from NAND or NOR logic. Due to the unique operation of QCA wires and gates, the QCA PLA discussed in [12] uses AND and OR logic. A schematic of one PLA cell for the AND plane is shown in Fig. 2a.

A PLA cell is equivalent to a crosspoint in a traditional PLA. The PLA cell structure contains a re-programmable *select bit* (denoted by “Select” or “S”) and two majority gates – one configured to act as an AND gate and the other configured to function as an OR gate. Referring to the layout in Fig. 2b:

$$\text{if } S=0, (\text{Implicant Out}) = (\text{Literal In}) \bullet (\text{Implicant In})$$

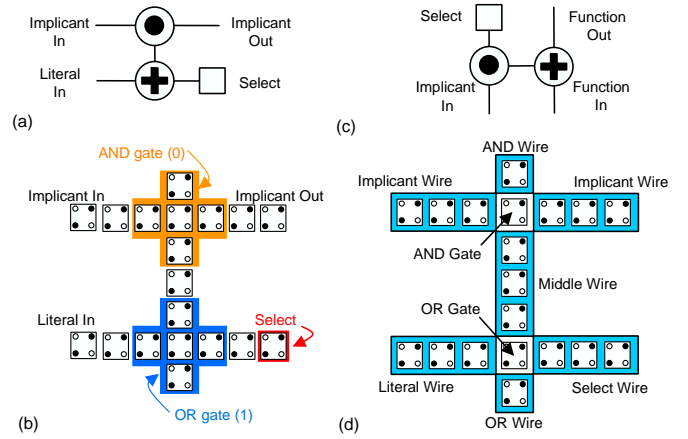


Fig. 2. (a) QCA AND Plane Cell Schematic, (b) AND Plane Cell Layout, (c) QCA OR Plane Cell Schematic, (d) AND Plane Cell Regions.

if $S=1$, $(\text{Implicant Out}) = (\text{Implicant In})$

Thus, if $S=0$, the PLA cell acts as an AND gate (*logic mode*), and if $S=1$, the PLA cell will act as a wire (*wire mode*). The ability to conditionally set each select bit makes the PLA re-programmable. In the OR plane, the position of the AND and OR gates in one “cell” is reversed (Fig. 2c). The select bit should be set to 1 for *logic mode* and 0 for *wire mode*:

if $S=1$, $(\text{Function Out}) = (\text{Implicant In}) + (\text{Function In})$

if $S=0$, $(\text{Function Out}) = (\text{Function In})$

By leveraging the structures just discussed, it is relatively easy to construct the logic required for a PLA of arbitrary size. We refer the reader to [5] for more detailed examples.

C. Faults and the QCA PLA Design

Employing the well known stuck-at fault model, the authors in [5] studied the impacts of such faults on system-level functionality. They showed how stuck-at faults would logically affect different parts of the PLA. Besides the faults similar to conventional PLA structures such as broken literal wires, QCA PLAs may have some unique fault types. For example, if a stuck-at-1 fault occurs in the Select Wire region of a PLA cell (see Fig. 2a,d) that needs to be programmed into logic mode, the fault does not adversely affect the behavior, since the select bit still outputs a “1” as required. Also, a PLA cell operating incorrectly does not necessarily ruin the operation of an entire row. The logic implemented in the PLA can take on many patterns, allowing for the use of faulty PLA cells. Similarly, if a stuck-at-0 fault occurs on the Literal Wire, the PLA cell can be programmed to wire mode, and it would still be useful for generating an implicant.

III. RELATED WORK

In this section, we first briefly review existing PLA fault tolerance research. We then discuss one specific PLA mapping approach in more detail as it forms the basis of our mapping work for QCA PLAs.

For MOSFET PLAs, the crosspoint fault model was developed as a more accurate model for PLA faults [25]. The crosspoint faults better encompass the fault behaviors specific

to PLAs. However, the crosspoint model is not sufficient for many nano-scale PLAs, as there are nano-scale defects and faults not present in MOSFET PLAs.

In terms of redundancy, there was a great deal of research effort put into MOSFET PLA *repair* in the 1980s and early 1990s. The idea of *mapping* to PLAs was not as developed. For the repair process, redundant rows were introduced to replace programmed rows that were defective. There were a few research efforts that did look at fault detection and made strides towards full mapping. In [7], a field-programmable PLA (FPLA) was designed with fixed inputs and outputs, but the implicant terms had full arrangement flexibility. Using a bipartite graph representation of the FPLA, a matching algorithm could be used to determine successful mappings of the desired Boolean functions. Beyond that, most of the mapping and resource allocation work moved on to FPGAs. However, the popularity of re-programmable PLAs is increasing again in the area of emerging nanotechnologies.

We are aware of one fault study specifically for QCA PLAs [5]. This work provides a fault model, and gives a brief yield study. However, the yield analysis does not consider the mapping of benchmarks. Instead, a simplification is made in order to get first-order yield numbers for PLAs of a general size. A more complete nano-scale PLA fault model has been adopted in the context of nanowire crossbars [6]. While not specific to QCA, we can leverage PLA-based mapping techniques developed for other emerging technologies.

In [24], a resource allocation method is proposed for nanowire crossbar PLAs. This nanowire crossbar architecture is similar to the one discussed in [10]. At a high level, this approach consists of two major steps: (i) model the given Boolean function and the PLA structure as two graphs, and (ii) determine a graph monomorphism matching between the two graphs. **This approach allows for rearrangement flexibility of inputs and outputs as well as implicant terms, resulting in more possible mappings and higher yields as compared to the mapping algorithm given in [7]. However, more rearrangement flexibility leads to a more computationally intensive mapping algorithm. While other research projects attempt to reduce computation time through different heuristics in the mapping algorithm [23], our goal is to improve the algorithm to identify more possible mappings.** We review the work from [24] in more detail below since it is closely related to our work.

The graphs used to model the sum-of-product Boolean functions and the PLA structure both take the form of back-to-back bipartite graphs. More specifically, there are three layers of vertices S_1 , S_2 , and S_3 . Edges are directed and only allowed from vertices in S_1 to S_2 and from S_2 to S_3 .

For the graph representation of Boolean functions, which will be referred to as the **Logic Graph (LG)**, vertices in S_1 correspond to input literals, vertices in S_2 correspond to implicants, and vertices in S_3 will correspond to output functions. Edges between the vertices in S_1 and S_2 represent the combination of literals into implicants using the AND Boolean operation. Edges between vertices in S_2 and S_3 represent the combination of implicants into functions using the OR Boolean operation. As an example LG, consider the

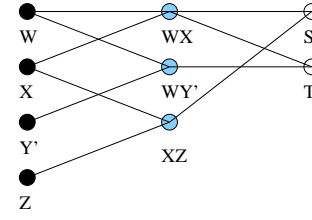


Fig. 3. Logic Graph Representation. The logical functions $S = WX + XZ$ and $T = WX + WY'$ are represented in graph form.

two following sum-of-product functions: $S = WX + XZ$ and $T = WX + WY'$. The graph representation of these two logic functions is shown in Fig. 3. For the representation of Boolean functions, there are no extra edges or vertices beyond the minimum needed to represent the logic.

The graph for the nanowire crossbar-based PLA structure, referred to as the **Crossbar Structure Graph (CSG)**, captures both redundant nanowires and defects in the model. **To create a CSG from a fabricated PLA structure, information must first be obtained from a fault-detection procedure. For the purposes of this paper, many example faulty PLA structures are created digitally, and the fault-detection information is considered accurate and complete (see Sec. VI).** If there are no defects in the nanowire crossbar structure, the representative back-to-back bipartite graph would have vertices in S_1 equal to the inputs, vertices in S_2 equal to implicant rows, and vertices in S_3 equal to outputs in the PLA. In addition, the graph would be a complete graph.

However, large collections of nanowires are unlikely to have zero defects. There are three main defects that affect the nanowire crossbar: broken nanowires, stuck-open crosspoints, and stuck-closed crosspoints. Since each defect causes certain resources to be faulty, they must be reflected in the CSG. As an example, Fig. 4a illustrates a defective PLA with 5 inputs, 5 implicant rows, and 3 outputs. There are two broken nanowires shown as broken lines. There are two stuck-open crosspoints indicated by an 'X' for each. Finally, there is one stuck-closed crosspoint shown as a dark square.

To create a CSG that represents a defective PLA structure, a complete bipartite is initially used but certain rules are applied to remove edges and vertices based on the defect pattern. Vertices in the graph correspond to nanowires in the crossbar, while edges between vertices correspond to crosspoints in the crossbar. For broken nanowires, the corresponding vertex must be removed (and thus all adjacent edges). For nanowire crosspoints that have a stuck-open defect, the corresponding edge must be removed from the graph. While there is no discussion of stuck-closed defects in [24], there is mention of these defects in [10] along with a scheme that can be used to modify the structure graph. All nanowires that are connected through a stuck-closed crosspoint are removed from the graph representation and are instead used to route a single signal. For the sake of the mapping, both vertices and all adjacent edges are removed from the structure graph. Fig. 4b shows the CSG for the PLA structure in Fig. 4a.

Once the graph for the defective nanowire crossbar structure has been generated, a mapping can be achieved by finding a monomorphism between the graphs [24]. We refer to this

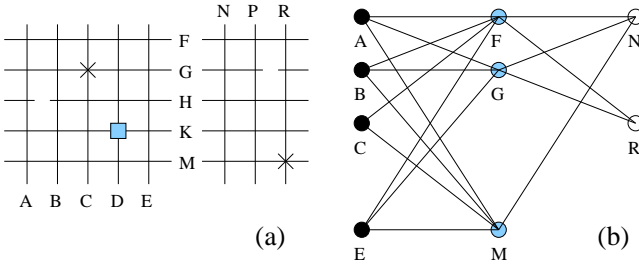


Fig. 4. (a) A defective and redundant nanowire crossbar structure. The AND plane is 5x5 cells, and the OR plane is 3x5 cells. There are 5 faults. (b) Crossbar Structure Graph Representation. Vertices and edges have been removed based on the location and types of PLA cell faults.

mapping method as the **Crossbar Based PLA Mapping Method (CBPM method)**. This method includes both the generation of the structure graph from the known defects and the use of monomorphism to find a matching. According to [24], the authors used an implementation of monomorphism matching discussed in [3].

IV. MQCA CASE STUDY

Regardless of implementation, almost every QCA device will be defective in that it deviates from its ideal shape, location, or orientation. For example, upon examining Figs. 1d and 1e, it is obvious that none of the fabricated magnets have the ideal “rounded rectangle” shape espoused in [15]. That said, successful experiments and simulations clearly indicate that defects must reach a certain severity before a fault occurs.

Because fabrication variations do not always lead to faulty QCA devices, the authors of [5] introduce the term “Effective Defect Rate” (EDR), to quantitatively capture the impact of QCA defects on device functionality. In other words, the EDR of defect type x is the probability that the occurrence of defect x causes a QCA device to malfunction. Like the work presented here, the authors of [5] also performed an MQCA-based case study and reported that nanomagnets could be misshapen and still produce the correct logical polarization. Thus, EDRs should be significantly lower than the probability of simply seeing fabrication variation from device to device.

That said, some fabrication variation will result in logical faults as discussed in Sec. II-C. While [5] enumerated what faults might be associated with the various implementations of the QCA device architecture, and suggested what faults might be associated with specific implementations, it did not present any specific experimental or simulation-based evidence that would confirm these projections. We do this here for MQCA and will leverage this information in Sec. V to consider different methodologies for mapping logic to a PLA in the presence of faults.

A. Simulation Methodology

For our simulations, we consider how localized fabrication variation might affect dataflow. We chose to take this approach as opposed to the digitization approach discussed in [5] after contacting the authors of [14] to better understand what their scanning electron microscope (SEM) images actually illustrated. Essentially, the images are a top-down view. However, they do not provide accurate information about nanomagnet

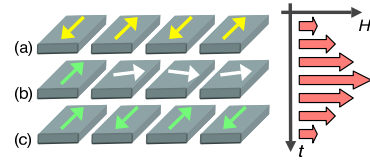


Fig. 5. Operating scheme of a wire: (a) initial configuration, (b) high-field (“null”) state, (c) after the application of the input, and the final ordered state.

shape, since variation in the third dimension is not clear. Based on our discussions, we determined that a better course of action was to use the two dimensional SEM images to provide more general defect information. For example, notice that some of the magnets from the wire in Fig. 1d clearly have a “slanted” edge. We took this information, and created a defective nanomagnet shape that is representative of this misshapeness (Fig. 6a). Depending on the severity of this slant, we have discovered that a magnet may or may not transmit data correctly (see Sec. IV-B). Empirical evidence based on experimental data presented in [15] corroborates this observation.

We simulated MQCA wires (Fig. 1d) and gates (Fig. 1e) using the NIST-developed micromagnetic simulator called OOMMF [8]. The tests included a clock that took the form of a periodically oscillating external magnetic field that drove a system to an initial state, and then controlled the relaxation of the system to a ground state. Conceptually, a magnetic field is applied along the hard (or shorter) axes of a group of nanomagnets. For example, Fig. 5a illustrates a wire of nanomagnets that has relaxed to a logically correct, antiferromagnetically coupled ground state. In Fig. 5b, the external field turns the magnetic moments of all magnets horizontally into a neutral logic state against the preferred magnetic anisotropy (i.e. along the hard axes of the magnets). This is an unstable state of the system, and as the field is removed, the nano-magnets relax into a new antiferromagnetically ordered ground state in accordance with the new input (Fig. 5c).

As the clock will be required for functional circuits, we have also carefully studied the clocking circuitry proposed in [19] to model the time evolution of a given circuit as realistically as possible. When we attempted to mimic wire switching experiments similar to those discussed in [19] and [5], we found that our wires would not switch correctly and retained some of the remnant magnetization associated with the previous computation (see Fig. 6b). Upon closer examination, we determined that in the previous work, the authors terminated their wire segments with a block of magnetic material to mimic an infinitely long wire and assist with the removal of remnant magnetizations. Ultimately we must ensure that remnance can be removed by having the last magnet of one wire group couple to just the first magnet of the next. To achieve this goal, we have developed a modified clocking scheme that still leverages the structures proposed in [19], but also successfully removes remnance. A detailed discussion is beyond the scope of this paper, but we note that two adjacent wires will be excited simultaneously (see Fig. 6c). By exciting the second wire, the last magnet in Group A can couple to the first (nulled) magnet in Group B. This process is repeated for Groups B and

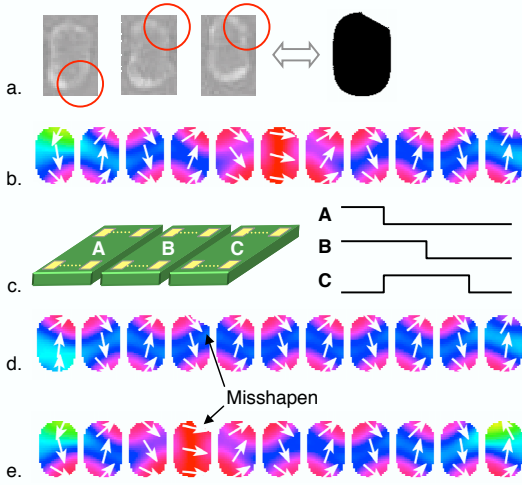


Fig. 6. (a) Experimentally realized magnets have slanted edges, (b) The 6th magnet in this chain does not have the correct polarization. The rest of the magnets in the wire (7-11) are still in their initial ground state. (c) 3 clocked groups of magnets (A-C); 2 groups are excited simultaneously. (d) Wire with misshapen magnet in a logically correct ground state. (e) When input is changed, misshapen magnet prevents data propagation and induces a stuck-at fault.

C, etc. We use this more realistic methodology to study the effects of defects on successful dataflow.

B. Simulation Results

We now consider how a magnet with a misshapen edge might affect signal propagation. For this discussion we consider an 11 cell wire initialized to a logically correct ground state (see Fig. 6d) and assume that the first magnet in this image is the last magnet in Group A. Group B (magnets 2-6 of Fig. 6d) and Group C (magnets 7-11 of Fig. 6d) would be clocked as illustrated in Fig. 6c. Thus, we simultaneously excite Groups A and B and flip the magnet in Group A. We then remove the clock field associated with Group A and turn on the field associated with Group C. (Thus, Groups B and C are excited simultaneously.) The desired result of these simulations is to see the magnets in Group B switch in accordance with the polarization of the magnet in Group A. However, as seen in Fig. 6e, while the first magnets in Group B do switch correctly, when the signal arrives at the misshapen magnet, the signal does not continue to propagate through the wire. This is because the nulling field does not sufficiently remove the remnant magnetization associated with the initial ground state. The net result is a “stuck at” fault. Depending on the initial state of the wire and where the defect is, stuck at zero and stuck at one faults are obviously possible.

V. MAPPING TO FAULTY QCA PLAS

In this section, we present two approaches to mapping a given logic function to a PLA with the stuck-at faults discussed above. Such a PLA may have redundant rows and columns to improve yield. The first method is a natural extension to the graph matching approach proposed for nanowire crossbar PLAs [24]. Though the extension is simple to implement and quite effective, it is still rather pessimistic in terms of PLA

yields. That is, it may deem a PLA cannot implement the given functions even though an implementation does in fact exist. The second approach exploits the unique features of the faults in QCA PLAs to construct new graph models for the mapping problem, and hence allows for a more versatile usage of faulty crosspoints. Much higher yields can be achieved by the second approach as will be shown in Sec. VI.

A. Extending Crossbar-Based PLA Mapping to QCA PLAs

The CBPM method discussed in Sec. III can be extended to map Boolean functions to QCA-Based PLAs. Recall that the CBPM method basically consists of two major steps: (i) constructing the crossbar structure graph (CSG) for a given PLA array and the logic graph (LG) for given Boolean logic functions, and (ii) applying a monomorphism algorithm to match the LG to the CSG. In extending this method to QCA PLAs, the key lies in how to correctly capture the different types of faults present in a QCA PLA in the structure graph. We give the detailed discussion below.

At the level of QCA PLA cells, there are three major categories of faults. That is, each faulty PLA cell is either (a) *stuck in wire mode*, (b) *stuck in logic mode*, or (c) *completely unusable*. The crossbar defects are very similar. Crosspoints can be (a) stuck-open, (b) stuck-closed, or (c) a nanowire can be broken. Because of this similarity, only a small change needs to be made to the CBPM method

To handle the faults in QCA PLAs, we propose to extend the graph model used for crossbar PLAs (see Sec. III) such that the same monomorphism-based graph matching can still be applied. The CSG must be modified, while the same LG can be used. We refer to the new graph as **Enhanced Crossbar Structure Graph (ECSG)**. For a given faulty PLA, we construct the ECSG as follows: we start with a complete back-to-back bipartite graph where vertices represent the inputs, rows, and outputs of the PLA, and edges represent PLA cells. We then examine each fault in the PLA to decide which vertices and edges must be removed to reflect the faults in the PLA structure.

If a QCA PLA cell is stuck in wire mode, the corresponding edge in the ECSG is removed. This is identical to the CSG for stuck-open defects. If a QCA PLA cell is completely faulty, again, the corresponding edge is removed. However, there is an additional consequence. Because the PLA cell is completely faulty, anything that depends on the output of that PLA cell will also be faulty. This means that no signal can pass through the row or column that the cell’s output is connected to. For the ECSG, this means that the vertex pointed to by the corresponding edge must be removed. Any edges adjacent to the vertex must also be removed. This is identical to the CSG for a broken nanowire defect.

The third defect is the stuck-closed crosspoint. While a PLA cell stuck in logic mode is similar, the consequences for the ECSG are very different. In the case of a nanowire crossbar, a stuck-closed defect means that two nanowires are connected across a low-resistance crosspoint. It is necessary to remove both wires from the mapping representation and map a single signal instead. The same is not true for a PLA cell stuck in

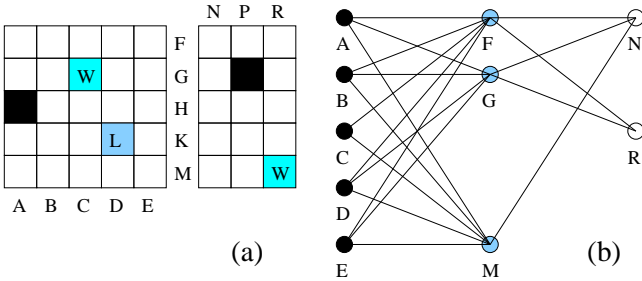


Fig. 7. (a) A defective and redundant QCA PLA structure. The AND plane is 5x5 cells, and the OR plane is 3x5 cells. There are 5 faulty cells. (b) The ECSG for this defective PLA structure. Vertices and edges have been removed based on the location and types of PLA cell faults.

wire mode. This fault only means that the cell will perform a logical operation on the two crossing signals. There is no reason to remove both the row and column wires. Instead, a QCA PLA can treat this in the same way it treats a fully faulty PLA cell. Only the wire that is connected to the output of the faulty PLA cell will be faulty.

We use a simple example below to illustrate the construction of an ECSG. A QCA PLA structure is given in Fig. 7a with faults that correspond to the defects given in the example shown in Fig. 4a. Like the example nanowire structure, the QCA PLA has 5 inputs, 5 implicant rows, and 3 outputs. There are a total of 5 faulty PLA cells. Two are fully faulty (the dark squares), two are stuck in wire mode (the squares labeled with W), and one is stuck in logic mode (the square labeled with L). The resulting graph is shown in Fig. 7b. The two fully faulty cells lead to the removal of vertex H and P and their adjacent edges. The stuck at logic fault caused the removal of vertex K and its adjacent edges. The two stuck at wire PLA cells caused the removal of edges (C, G) and (M, R) .

In summary, the graph representation of a defective QCA PLA structure can be generated in a similar way as that for a defective nanowire crossbar. Fully faulty PLA cells correspond to broken nanowires. PLA cells stuck in wire mode correspond to stuck-open crosspoints.

Finally, PLA cells stuck in logic mode are treated in a somewhat different manner. Because the ECSG that represents the faulty QCA PLA is a back-to-back bipartite graph, mapping logic to a QCA PLA structure can be done using the same matching algorithm [3] as for nanowire crosspoints. We call this whole procedure, from the graph generation to the matching of the graphs, the **Enhanced Crossbar Based PLA Mapping Method (ECBPM method)**. We will discuss the effectiveness and efficiency of the ECBPM method in Sec. VI.

B. Yield-Increasing Mapping for QCA PLAs

Though the ECBPM method discussed in Sec. V-A is easy to implement and quite efficient in terms of finding mappings (to be shown later), it can be rather pessimistic in terms of yield. This is due to the fact that QCA PLA cells are just as likely to become stuck in logic mode as in wire mode – but a much larger number of edges would be removed from the ECSG for a stuck in logic fault than for a stuck-in-wire fault. Such removals could significantly hurt defect tolerance and yield compared to what could be achieved by a mapping that treats stuck-in-logic faults more intelligently. In this section,

we introduce a new method of dealing with stuck in logic faults, which leads to PLA mappings with higher yield.

Before introducing our new method, let us first review the implications of the edges in the LG and ECSG. An edge in an LG represents the existence of a logic function (AND or OR). An edge in the ECSG represents the capability of a programmable PLA cell being programmed into a logic gate. If this edge is matched to an edge in the LG, the respective PLA cell is programmed into a logic gate. Otherwise, the PLA cell is programmed into a simple wire. If a PLA cell is stuck at logic, it could still be used properly if this cell “happens” to be selected to function as a gate. Therefore, if we could distinguish the edges in a PLA structure graph between wire mode only, logic mode only, or both modes, it would be possible to match these edges in a more intelligent manner.

We now introduce our new graph model. We use a **QCA-based PLA Structure Graph (QSG)** to represent a QCA PLA. The vertices in a QSG are derived the same way as those in a CSG. That is, literal wires correspond to the first layer of vertices, implicant wires correspond to the second layer of vertices, and output wires correspond to the third layer of vertices. Between any two vertices in the adjacent layers, there may exist an edge of one of the following three types:

- Type 1: The PLA cell is fault free
- Type 2: The PLA cell is stuck in logic mode
- Type 3: The PLA cell is stuck in wire mode

Fig. 8a is the QSG for the PLA structure given in Fig. 7a. Due to the two completely unusable PLA cells, vertices H and P and their adjacent edges are missing from the QSG. The two stuck at wire PLA cells give the two dotted edges while the one stuck at logic PLA cell results in the one dashed edge. Compared with the ECSG in Fig. 7b, the QSG has more edges and vertices.

To leverage the graph monomorphism algorithm, we also modify the way that a Boolean logic function is modeled. Instead of modeling only the logic functions as edges in an LG, we model both the logic functions and the signal passages. Specifically, we introduce a new graph called the **Complete Logic Graph (CLG)**. Similar to the LG, the CLG is a back-to-back bipartite graph where the first layer vertices represent the literals in the given function, the second layer vertices represent the implicants, and the third layer vertices represent the functions. Different from an LG, the CLG is a *complete* back-to-back bipartite graph. A subset of the edges correspond exactly to the edges in the LG and are called Type 2 edges, while the rest of the edges are called Type 3 edges. The CLG corresponding to the example LG in Fig. 3 is shown in Fig. 8b. Type 2 edges are shown as dashed edges while Type 3 edges are shown as dotted edges.

Mapping a given set of Boolean logic functions to a QCA PLA structure can now be solved by employing a monomorphism matching algorithm for attributed relational graphs (ARG). In particular, the edges types are treated as attributes such that Type 2 edges can be matched to either Type 1 or Type 2 edges while Type 3 edges can be matched to either Type 1 or Type 3 edges. By incorporating these attributes, we do not prematurely eliminate certain edges and hence increase the probability of finding a matching between

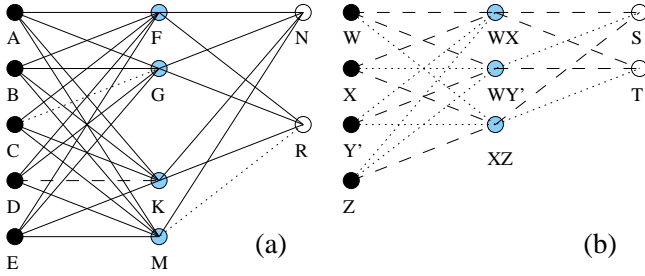


Fig. 8. (a) The QSG corresponding to the QCA PLA structure in Fig. 7a. The solid edges represent PLA cells that can be programmed into either logic or wire mode (Type 1), while the dashed edges represent the cells that are stuck in logic mode (Type 2), and dotted edges represent cells that are stuck in wire mode (Type 3). (b) The CLG corresponding to the LG in Fig. 3. The dashed edges show the logic to be mapped (Type 2), while the gray edges show which connections need to pass signals (Type 3).

TABLE I
PLA YIELD FOR MAPPING TWO BOOLEAN FUNCTIONS TO AN (8,6,4) PLA USING ALL THREE METHODS

Fault Rate	1%	5%	10%
CBPM	100%	92.3%	63.0%
ECBPM	100%	94.4%	71.5%
QCAPM	100%	99.4%	94.5%

the CLG and QSG. There exist both exact and approximate algorithms for ARG monomorphism matching (e.g. [3], [9]) which can be readily applied to solve our PLA mapping problem. Elaborating on the inner working of such algorithms is beyond the scope of this paper.

We would like to point out that our new QSG/CLG based PLA mapping technique (referred to as **QCA-based PLA Mapping Method (QCAPM method)**) does lead to more complex graphs having to be matched. This could significantly increase the computation time for solving the ARG monomorphism problem for large problem instances. For such cases, approximate algorithms should be used instead of exact ones. We will use experimental data to illustrate the effectiveness and efficiency of our new QCAPM approach.

VI. EXPERIMENTAL RESULTS

In this section, we use a set of experiments to evaluate the different PLA mapping methods discussed in this paper, i.e., the original CBPM method, the improved ECBPM method, and the new QCAPM method. We focus on two important aspects in comparing these methods: potential yield for QCA-based PLAs and runtime efficiency.

The first sets of experiments that we ran were aimed at comparing the yields possible for the three mapping methods. For the purpose of complete analysis, we started with a very simple set of Boolean functions. The first test utilized the Boolean functions in Fig. 3. To implement these two Boolean functions, four inputs and three implicants are required. Therefore, we decided to map the functions to a defective PLA of size (8,6,4). We created 1,000 PLA structures with random faults, and attempted to map the functions to the test structures. Fault rates of 1%, 5%, and 10% were used during the generation of the faulty PLAs. Results are presented in Table I.

The yield results were obtained using an exhaustive search of matchings between the representative structure and function

graphs. All three methods found successful mappings for the entire set of 1,000 test PLAs when the fault rate was set at 1%. The test was performed on an (8,6,4) PLA, which has 48 crosspoint cells in the AND Plane and 24 crosspoints in the OR Plane. Since there are only 2 Boolean functions, 3 implicants, and 4 literals to map onto the PLA structure, it is unlikely that having only 1% faulty cells out of 72 will lead to an unsuccessful matching. At a 5% fault rate, all methods exhibited at least a few cases out of 1,000 tests without a successful matching, and even more so at a 10% fault rate.

The results show that the QCAPM method is quite effective in providing higher yields than the other methods do. Even at 10% faults, the QCAPM method had a yield over 94% while the two crossbar-based mapping methods had yields much lower. We expected this, since our approach can utilize crosspoint that are stuck in wire and logic mode more effectively than the other approaches. The two crossbar-based methods see a drop in yield more quickly as the fault rate increases.

The most common architectural benchmarks, however, are much larger than just two Boolean functions. Since the exact graph monomorphism algorithm used in our tool suite is exponential in time complexity, as the benchmark increases in size the mapping will soon become infeasible if an exhaustive search is used. Therefore, we have implemented a runtime cutoff to stop searching for a successful matching after a certain period of time. This is a common step taken when trying to map onto reprogrammable logic. Stopping the search after a certain period of time will reduce the number of tests that return a successful matching, resulting in a reduction in the yield. However, there will generally only be a small number of tests that will incorrectly see failure because the cutoff was too small. Most failures with the cutoff will still be failures without the cutoff. Generally, if the cutoff is reached but there is a solution, then that particular test would take orders of magnitude longer to run to a successful conclusion.

To test these three methods on a larger benchmark, we used the Boolean functions for a 3-bit adder expressed in the sum-of-product form. There are 12 inputs and 31 implicant terms required to produce the 4 bits of output needed for the 3-bit adder. We attempted to map the 3-bit adder to 200 PLA structures with random faults. We used a cutoff time of 5 seconds. The yield results are illustrated in Fig. 9.

The results from mapping a larger benchmark to random faulty PLA structures indicate that the QCAPM method is similarly effective in providing higher yields as in our test on a smaller benchmark. As compared to the results given in Table I, the yield for the same fault rate is lower for all methods. However, this is more likely due to the PLA structure size. For the smaller example functions, the PLA size was twice that of the requirements for the functions. That is, there were only 4 literals, 3 implicants, and 2 functions trying to map onto a PLA with 8 inputs, 6 rows, and 4 outputs. In the case of the 3-bit adder, we tested a structure that had less redundancy. There were only 20 inputs for 12 literals, 40 rows for 31 implicants, and 10 outputs for 4 functions. By adding extra redundancy, we would see an increase in yield in all cases.

However, the trends seen for the small example are still visible for this larger example up until the fault rate reached 7%.

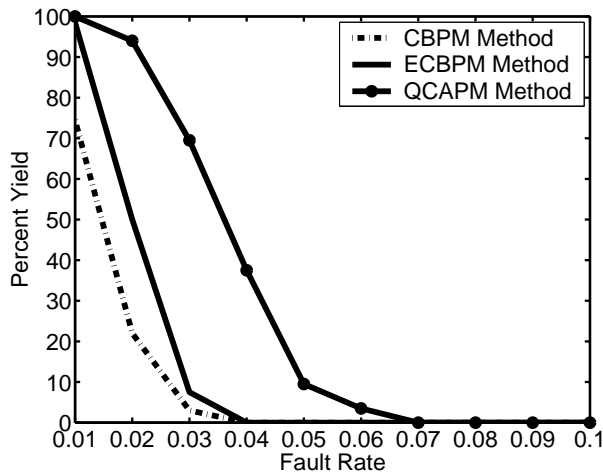


Fig. 9. This graph shows the PLA yield for the three mapping methods. Tests were performed to map the 3-bit adder functions to a (20,40,10) faulty PLA structure. As the fault rate was increased, the yield decreased for all three mapping methods.

TABLE II
RUNTIME COMPARISON OF THE THREE MAPPING METHODS FOR SUCCESSFUL MATCHINGS OF THE 3-BIT ADDER.

Fault Rate	1%	2%	3%
CBPM	94 μ sec	85 μ sec	89 μ sec
ECBPM	96 μ sec	99 μ sec	100 μ sec
QCAPM	142 μ sec	145 μ sec	146 μ sec

At that point, all methods had a 0% yield. It is interesting to note that the largest yield reduction occurred at different places and at different rates for each method. Both the CBPM and the ECBPM methods dropped off most significantly between 1% and 3% faults, while the QCAPM method dropped off most between 2% and 5%. In addition, the yield for the QCAPM method did not decrease at the same rate as the other two methods. It took a fault rate *change* (not fault rate) of 3% to reduce the QCAPM method from a yield of 95% down to around 10%. On the other hand, the CBPM method went from a 75% yield to around a 5% yield with only a 2% increase in fault rate. The ECBPM method saw the worst drop-off, losing 90% yield with an increase in fault rate of only 2%.

While the QCAPM method clearly produces better yields with the same faulty PLA structures, there is a cost for this improvement. When we ran the 3-bit adder tests used to generate Fig. 9, we also tracked the runtime of each mapping method. For tests that could not find a successful mapping, the run times were all the same, since the cutoff was the same for each. However, for tests that did find a successful mapping, there were more interesting runtime results.

The CBPM method had the shortest run times and the QCAPM method had the longest run times on successful tests. This means that if there is a relatively easy solution to the mapping problem, the CBPM method will find that solution faster. The QCAPM method will find a mapping more often. The runtime averages for successful tests are shown in Table II. On average, the CBPM method seems to be about 1.5 to 2 times faster in finding successful mappings.

VII. DISCUSSION AND FUTURE WORK

In the area of reconfigurable nanotechnology, much effort has focused on nanowire or nanotube diode logic PLAs. Defect studies, fault models, routing algorithms, and performance calculations have been obtained from the mapping of logical benchmarks. In the area of QCA research, similar studies have been done. From reconfigurable designs [12] and defect studies [18], [21], [20] up to fault models [13], [5] and this work on routing and resource allocation, QCA is proving to be an exciting emerging technology with many implementations that target many applications.

However, there is still much work to be done, especially for QCA PLAs. We plan on using various tools and benchmarks to look at how QCA PLAs compare to other technologies in terms of area, delay, and power. While there has been work done to consider general power consumption for QCA [19], it was not specific to reconfigurable logic. However, the QCA PLA design as a reconfigurable architecture is an excellent vehicle for logic mapping. We intend to utilize our routing work to look at logic density and area requirements for certain logic benchmarks.

Finally, we will also explore methodologies to lower the EDR. For example, physical-level simulation has shown that we can effectively remove faults in MQCA circuit constructs by using a stronger clocking field, and we could apply this to MQCA PLAs. This suggests that we can eliminate faulty behavior at the expense of increased energy, as the applied field strength is determined by the amount of current in a clock wire [19]. We know of no analog to this behavior in other emerging technologies and it must be explored further.

REFERENCES

- [1] 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, 1999.
- [2] G. Bernstein, A. Imre, V. Metlushko, A. Orlov, L. Zhou, G. C. L. Ji, and W. Porod, "Magnetic QCA systems," *Microelectronics Journal*, vol. 36, pp. 619–624, 2005.
- [3] L. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.
- [4] R. Cowburn and M. Welland, "Room temperature Magnetic Quantum Cellular Automata," *Science*, vol. 287(5457), pp. 1466–1468, 2000.
- [5] M. Crocker, X. S. Hu, and M. Niemier, "Fault models and yield analysis for QCA-based PLAs," *Int. Sym. on FPL*, pp. 435–440, 2007.
- [6] A. DeHon and M. Wilson, "Nano-wire based sublithographic programmable logic arrays," *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, pp. 123–132, 2004.
- [7] M. Demjanenko and S. J. Upadhyaya, "Yield enhancement of field programmable logic array by inherent component redundancy," *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 8, pp. 876–884, 1990.
- [8] M. Donahue and D. Porter, "OOMMF user's guide, version 1.0, interagency report NISTIR 6367," <http://math.nist.gov/oommf>.
- [9] Y. El-Sonbaty and M. A. Ismail, "A new algorithm for subgraph optimal isomorphism," *Pattern Recognition*, vol. 31, pp. 205–218, 1998.
- [10] M. Haselmann and S. Hauck, "The future of integrated circuits: A survey of nano-electronics," *submitted to Proceedings of the IEEE*, 2008.
- [11] K. Hennessy and C. Lent, "Clocking of molecular quantum-dot cellular automata," *J. of Vac. Sci. & Tech. B*, vol. 19(5), pp. 1752–55, 2001.
- [12] X. S. Hu, M. Crocker, M. Niemier, M. Yan, and G. Bernstein, "PLAs in Quantum-dot Cellular Automata," *Int. Sym. on VLSI*, 2006.
- [13] J. Huang, M. Momenzadeh, M. Tahoori, and F. Lombardi, "Design and characterization of an and-or-inverter (AOI) gate for QCA implementation," *ACM Great Lakes Symposium on VLSI*, pp. 426–29, 2004.

- [14] 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, January 13, 2006.
- [15] A. Imre, "Experimental study of nanomagnets for Magnetic Quantum-dot Cellular Automata (MQCA) logic applications," *Dissertation, University of Notre Dame*, April 2005.
- [16] C. Lent and P. Tougaw, "A device architecture for computing with quantum dots," *Proc. of the IEEE*, vol. 85, p. 541, 1997.
- [17] M. Mitic, M. Cassidy, K. Petersson, R. Starrett, E. Gauja, R. Brenner, R. Clark, A. Dzurak, C. Yang, and D. Jamieson, "Demonstration of a silicon-based Quantum Cellular Automata cell," *Applied Physics Letters*, vol. 89, p. x, July 5, 2006.
- [18] M. Momenzadeh, H. Jing, M. Tahoori, and F. Lombardi, "On the evaluation of scaling of QCA devices in the presence of defects at manufacturing," *IEEE T. on Nano.*, vol. 4(6), pp. 740–743, Nov. 2005.
- [19] M. Niemier, M. Alam, X. S. Hu, G. Bernstein, W. Porod, M. Putney, and J. DeAngelis, "Clocking structures and power analysis for nanomagnet-based logic devices," *ISLPED '07: Proceedings of the 2007 international symposium on Low power electronics and design*, pp. 26–31, 2007.
- [20] M. Niemier, M. Crocker, X. S. Hu, and M. Lieberman, "Using CAD to shape experiments in molecular QCA," *Int. Conf. on Comp. Aided Design*, pp. 907–914, Nov 2006.
- [21] M. Ottavi, M. Momenzadeh, and F. Lombardi, "Modeling QCA defects at molecular-level in combinational circuits," *IEEE Symp. on Defect and Fault Tolerance in VLSI Sys.*, pp. 208–216, 2005.
- [22] H. Qi, S. Sharma, Z. Li, G. Snider, A. Orlov, C. Lent, and T. Fehner, "Molecular Quantum Cellular Automata cells. electric field driven switching of a silicon surface bound array of vertically oriented two-dot molecular quantum cellular automata," *J. Am. Chem. Soc.*, vol. 125, pp. 15 250–15 259, 2003.
- [23] W. Rao, A. Orailoglu, and R. Karri, "Topology aware mapping of logic functions onto nanowire-based crossbar architectures," *IEEE/ACM Design Automation Conference*, 2006.
- [24] G. Snider, P. J. Kuekes, T. Hogg, and R. S. Williams, "Nanoelectronic architectures," *Applied Physics A*, vol. 80, pp. 1183–1196, 2005.
- [25] F. Somenzi and S. Gai, "Fault detection in programmable logic arrays," *Proceedings of the IEEE*, vol. 74, no. 5, pp. 655–667, May 1986.
- [26] D. Strukov and K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," *Nanotechnology*, vol. 16, pp. 888–900, 2005.