FAULT MODELS AND YIELD ANALYSIS FOR QCA-BASED PLAS[1]

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

Various implementations of the Quantum-dot Cellular Automata (QCA) device architecture may help many performance scaling trends continue as we approach the nano-scale. Experimental success has led to the evolution of a research track that looks at QCA-based design. The work presented in this paper follows that track and looks at implementation friendly, programmable QCA circuits. Specifically, we analyze a novel, QCA-based, Programmable Logic Array (PLA) structure, develop an implementation independent fault model, discuss how expected defects and faults might affect yield, and look at the design in the context of a magnetic implementation of QCA.

1. INTRODUCTION

In this paper, we consider the effects of manufacturing defects on the yield of the PLA design proposed in [1] for the Quantum-dot Cellular Automata (QCA) device architecture. QCA accomplishes logical operations and moves data via nearest-neighbor interactions rather than with electric current flow. It has the potential to achieve fast logic evaluation with low power at the nano-scale. Four different implementations are being researched. Initial experiments were conducted on a *metal-dot* implementation of a QCA device [2]. Candidate QCA *molecules* have been shown to switch [3]. All components required for a functionally complete logic set have been experimentally demonstrated at room temperature for a *magnetic* implementation of QCA. If 10^{10} nanomagnets are adiabatically switched 10^8 times each second, they should only dissipate approximately 0.1W of power [4]. Nanomagnets with feature sizes of about 20 nm are feasible [5]. *Semiconductor-based* QCA devices have also recently been realized [6].

It is well recognized that any realized circuit or architecture at the nano-scale will have a higher percentage of defective devices than CMOS based circuits. Largely for this reason, reconfigurable logic has been studied by a number of research groups for different nano-scale technologies [7, 8, 9]. These studies considered not only the logic structure, but also what redundancy would ultimately be required in order to achieve a desired yield. This research component – missing from [1] – is presented here for the QCA-based PLA design.

[1]The research was supported in part by the US National Science Foundation under Grants CCF-0541324, CCF-0621990, and CCR-02-10153.

The QCA PLA is a promising, regular, and reprogrammable structure that can utilize redundancy to make usable nano-circuits. The design is compact and easily extensible. We present a fault model that considers the unique effects of QCA defects for many implementations. We analyze these faults and the PLA yield to reveal trends that a QCA PLA might exhibit. Finally, we use a case study to illustrate the validity of some observations for a magnetic QCA implementation.

We believe that this work should be relevant in the near-term as application spaces exist for reconfigurable logic at the nano-scale, and ideally for reconfigurable logic made from various implementations of QCA. For example, circuits made from nano-scale magnets should be non-volatile, dense, low power, radiation hard, and could have a natural interface to MRAM. These properties suggest that this technology would be ideal for processing in military and/or space applications [10].

2. BACKGROUND

We begin by introducing the QCA device architecture, reviewing the PLA design of [1] that will form the basis for our fault model, and reviewing other related work.

2.1. 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' (Fig. 1a) [11]. Logical operations and data movement are accomplished via Coulomb (or nearest-neighbor) interactions. QCA devices 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.

Figs. 1b-e illustrate the building blocks that would be used to construct QCA circuits [12]. A QCA wire (Fig. 1b) is just a line of QCA devices. The wire is driven at the input cell by a cell with a fixed/held polarization. The majority gate (Fig. 1c) implements the logic function $AB + BC + AC$. The output cell assumes the polarization of the majority of the 3 input devices [11]. By setting one input of a majority gate to a logic '0' or '1', the gate will

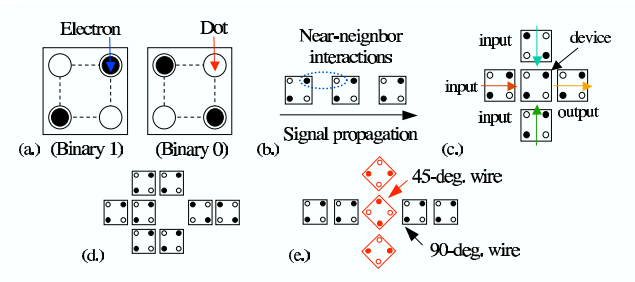

Fig. 1. Schematics of the fundamental structures needed to build QCA circuits: (a) basic device, (b) wire, (c) logic gate, (d) inverter, and (e) crossover.

execute an AND or OR function respectively. An inverter can be easily built with QCA devices (Fig. 1d). QCA wires with different orientations (Fig. 1e) can theoretically cross in the plane without destroying the binary value on either wire.

Regardless of implementation, a circuit or system made from QCA devices will require a clock structure that will take the form of lithographically defined, conducting metal wires [13]. The clock is required to maintain state for electrostatic QCA, and to remove state for magnetic QCA. For detailed descriptions on the clocking structure, readers can refer to [14].

2.2. Reprogrammable PLA Cells and Array Structure

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 [1] uses AND and OR logic. A schematic of one PLA cell for the AND plane is shown in Fig. 2a. For this paper we will refer to the structures at the crosspoints of the PLA as "PLA cells" and the QCA building blocks as "QCA devices." This structure contains a reprogrammable *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:

if S=0, *(Implicant Out) = (Literal In)* • *(Implicant In)*

if S=1, *(Implicant Out) = (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 reprogrammable. 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, *(Function Out) = (Implicant In) + (Function In)*

if S=0, *(Function Out) = (Function In)*

By leveraging the structures just discussed, it is relatively easy to construct the logic required for a PLA of arbitrary size. Fig. 3 shows how the "cell" construct can be used to make entire AND and OR planes. Select bits are set to perform a majority voting function (their binary values are in the inset triangles).

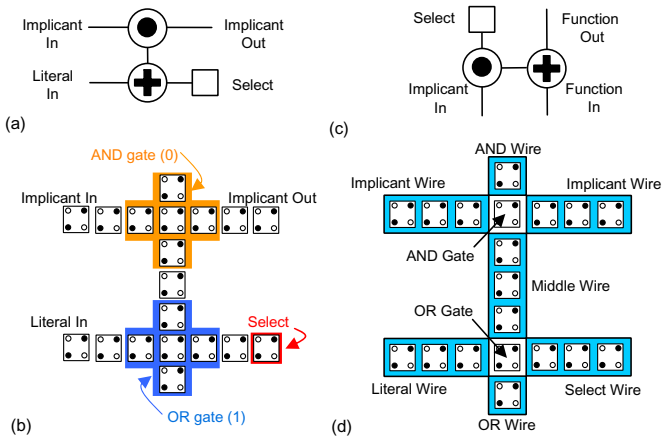

Fig. 2. (a) QCA AND Plane Cell Schematic, (b) AND Plane Cell Layout, (c) QCA OR Plane Cell Schematic, (d) AND Plane Cell Regions.

2.3. Related Work

We are not aware of any defect and yield studies specifically for QCA PLAs. A PLA fault model has been adopted in the context of nanowire crossbars [7]. For MOSFET PLAs, the crosspoint fault model was developed as a more accurate model for PLA faults [15]. The crosspoint model considers how defective or misplaced transistors cause the Boolean logic implemented by a PLA to change. There are four crosspoint faults: growth, shrinkage, appearance, and disappearance. The crosspoint model is not sufficient for a QCA PLA, as there are faults that are not present in a MOSFET PLA.

3. FAULT MODELING

In this section, we discuss defects and the consequent faults. Here, a "defect" is defined as a deviation of a QCA device from the ideal device in terms of shape, orientation, location, etc. A defective QCA device does not necessarily exhibit the wrong logical behavior. Defect tolerance is the ability for a device to operate properly even though it is not perfect. A "fault" occurs when a defect is severe enough to cause incorrect logical behavior in a QCA device. We first discuss possible defects and faults for different QCA implementations, then examine the effects that those defects and faults can have on the QCA PLA structure, and finally quantify the fault probabilities based on defect rates.

3.1. QCA Defects and Faults

We first introduce the unique types of faults in QCA devices and then discuss what kinds of defects cause such faults. Due to its different operating principles, QCA devices experience more types of faults than traditional MOSFET devices. Besides the stuck-at fault (which is equivalent to the MOSFET one), QCA devices have

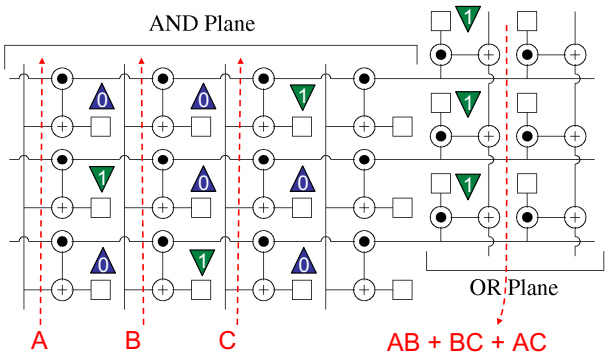

Fig. 3. A PLA array that implements the majority voting function

two other types of unique faults. The first is the non-deterministic fault (ND) where the output of a circuit may vary during run time for the same inputs. The second is the inversion fault where the output of a circuit becomes the inverted value of the desired output.

The ND fault occurs primarily in electrostatic implementations. Looking at molecular QCA, the output of a circuit has a probability of being "1" or "0", which is based on the Boltzmann distribution of electrons in the QCA devices [14]. Circuits that operate properly have a high probability to output one value and a low probability to output the other. However, if a circuit is unstable, it will have probabilities approaching 50% for outputs of both "1" and "0". For these types of circuits, small fluctuations in the environment surrounding the circuit (such as a change in temperature) can cause it to behave differently each time the circuit is used. Simulations show that a defect-free wire circuit that operates at a high probability can become unstable with a single missing QCA device. The ND fault captures this unpredictable behavior of a QCA circuit.

The inversion fault can easily occur in both electrostatic and magnetic implementations. It is often instigated by device misalignment. The misalignment can cause a QCA device to take on the opposite polarization of its neighbor. This phenomenon is exploited to our advantage in the inverter design (Fig. 1d), where neighboring QCA devices are placed next to each other in a diagonal direction to achieve inversion. However, an undesirable misalignment during fabrication can cause an unwanted inversion on a wire. This misbehavior is categorized as an inversion fault.

Defects in each QCA implementation may cause any one of the three faults (stuck-at, ND, and inversion) discussed above. Proper association of defects to faults helps in studying the fault probabilities and yield. By examining experimental data and performing simulations at the physical level [1, 14, 16], we have matched defects to faults for two QCA implementations. In Table 1, we summarize the faults that are induced by five typical defect types, i.e., shifts, rotations, missing, stray charges and misshapenness. While variations can occur, Table 1 represents the "common case."

There are five primary defects that we consider. For all implementations of electrostatic QCA, stray charges (α) can affect whether or not an individual QCA device retains its correct value (see Fig. 4a). (There are no magnetic monopoles.) Device shifts (β) and rotations (γ) occur when the position or the orientation of the QCA device deviates from the ideal (Fig. 4b and Fig. 4c respectively). Each of these defects can cause QCA devices to interact with one another in ways they normally would not. For example, a shifted nanomagnet (i.e. due to lithographic variations) might cause part of a wire to switch before it normally should (this will be illustrated later in Sec. 4.2). Similarly, a shifted QCA molecule may cause a signal to be inverted as discussed above.

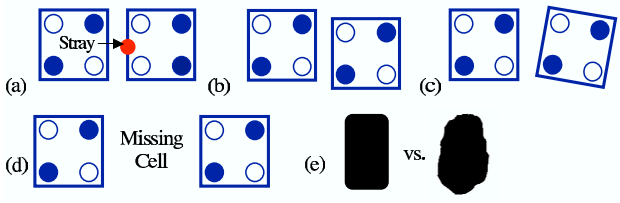

Fig. 4. Possible defects: (a) stray charge, (b) shifted device, (c) rotated device, (d) missing device, (e) misshapen device.

Table 1. Connecting Defects to Faults

Molecular		Magnetic	
Defect	Fault	Defect	Fault
shifts	inversion	misshape	stuck at / inversion
rotations	inversion	shifts	inversion / stuck at
missing	ND		
stray charge	stuck at		

Missing devices (δ), shown in Fig. 4d, cause nearest neighbor interactions to be weaker, as interactions are distance dependent. Such defects could be more frequent in the molecular implementation because of self-assembly. Finally, if a device deviates from its ideal shape (ϵ), as shown in Fig. 4e, it may interact with its neighbors incorrectly. Devices realized by lithography (e.g., nanomagnets) are especially susceptible to this type of defect (see Sec. 4.2).

We now look at how the defects and faults discussed above affect the logical behavior of the PLA design in [1]. The probability of a PLA cell becoming faulty depends on the number of QCA devices in it that could cause the PLA cell to fail, for a given type of fault model. We use variable C_x to represent the number of failure points for each defect type x. In other words, C_x represents the total number of sources susceptible to fault-causing defects in a PLA cell. As seen in Fig. 2b, there are 19 QCA devices in each PLA cell, and each is a potential point of failure. In the worst case, $C_x = 19$ for all defect types, and any fault that occurs in a PLA cell causes the entire row or column to fail. However, such a worst case is too pessimistic to be realistic.

We consider a more realistic analysis of fault probabilities based on the observation that certain regions of the PLA cell can withstand a fault and still operate properly. We use the PLA AND plane as an example (as the OR plane can be treated in much the same way). To facilitate this analysis, we divide the AND plane schematic into eight regions according to the basic functionalities of the QCA devices (see Fig. 2d).

By examining how each of the three possible faults would logically affect these regions, we were able to find cases in which a fault does not make the PLA cell inoperable. For example, if a stuck-at-1 fault occurs in the Select Wire region of a PLA cell that needs to be programmed into the logic mode, the fault does not adversely affect the behavior, since the select bit still outputs a "1" as

Table 2. Number of QCA devices (C_x) susceptible to faults

QCA Device Location	Device Count	AND Plane Row Failures				
		ND	invert	SA1	SA0	SA
AND wire	1	1				
OR wire	1	1				
literal wire	3		3			
implicant wire	6	6	6	6	6	6
select wire	3	3				
middle wire	3	3	3		3	3
AND gate	1	1		1	1	1
OR gate	1	1			1	1
total devices (C_x)	19	16	12			11

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. For example, 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.

Therefore, in a more realistic case, some of the susceptible sources can be discounted, as they do not lead to incorrect operations. We have determined which of the PLA cell regions are susceptible to each fault and which are not. This leads to different values of C_x for each fault type. The reduced values of C_x for the AND plane, based on the more realistic assumptions, can be found in Table 2. The 8 regions are listed in the first column. The number of QCA devices in each region is given in the second column. The 5 fault types and possible numbers of devices causing the faults are given in columns 3 to 7. Each region susceptible to the fault type is marked by putting the number of QCA devices into the table.

As an example, consider the ND fault (third column). Every region of the PLA cell except the Literal Wire region is susceptible to this fault because the PLA cell can be programmed to *wire* mode such that the fault in the Literal Wire does not affect the Implicant Wire. Since the PLA cell can withstand a fault in the Literal Wire, the entry in the table for that region is empty. For each fault, the numbers of cells are added to obtain the final C_x value.

While the more realistic case above reduces the value of C_x for each fault type, the considerations are still somewhat pessimistic. There are some techniques that we have not considered in our assumptions. For example, device-level redundancy [14] can be used to make the PLA cell more defect resistant in electrostatic QCA, possibly preventing defects from causing faults. Also, we only considered one way to program around faults, by setting the affected PLA cell to *wire* mode. It is possible that some inversion faults could be fixed by reordering the input literals or re-writing the logic functions. Such techniques could significantly reduce fault probabilities and improve yield. Due to page limits, we omit further discussion on this and leave it for future work.

3.2. Quantifying Defects and Faults

It is well recognized that defect rates will increase when dealing with devices with nanometer feature sizes. For nano-scale QCA implementations, almost every QCA device is *defective* in that it deviates from its ideal shape, location or orientation. For example, in a lithography based implementation of magnetic QCA, none of the nanomagnets have exactly the ideal rounded rectangle shape (see Fig. 8a). For molecular QCA, all devices could be slightly misaligned. Fortunately, both experiments and simulations indicate that defects must reach a certain severity before a fault occurs.

Since the occurrence of defects does not always lead to faulty QCA devices, we introduce the term **Effective Defect Rate (EDR)**, to quantitatively capture the impact of QCA defects on device functionality. The EDR of defect type x, denoted as r_x, is the probability that the occurrence of defect x causes a QCA device to malfunction. For the defect types discussed in Sec. 3.1, the EDRs are defined as:

$$r_\alpha = P(\text{a stray charge causes a fault/}nm^2)$$
$$r_\beta = P(\text{a shifted QCA device causes a fault})$$
$$r_\gamma = P(\text{a missing QCA device causes a fault})$$
$$r_\delta = P(\text{a rotated QCA device causes a fault})$$
$$r_\epsilon = P(\text{a misshaped QCA device causes a fault})$$

where P denotes the probability. For the stray charge defect, we assume that the distribution of the stray charges is uniform.

In order to study the fault tolerance capability of QCA-based PLAs, we use f_x to represent the probability of a fault occurring in a PLA cell due to defect type x. If the EDR of x is known, f_x can be readily computed. Consider the "device shift" (β) defect. Given the number of QCA devices susceptible to fault-causing shifts in a PLA cell (C_β), we have

$$f_\beta = 1 - (1 - r_\beta)^{C_\beta} \quad (1)$$

The computation of f_γ, f_δ and f_ϵ can be done in the same way. For stray charge defects, we compute the fault probability as follows:

$$f_\alpha = 1 - (1 - A * r_\alpha)^{C_\alpha} \quad (2)$$

where A is the *effective* area covered by a PLA cell. It is not sufficient to simply use the area of the QCA devices in a PLA cell, since a stray charge in the vicinity of the cell could still cause the PLA to malfunction. Furthermore, as there may be different sources of stray charges, the effective area and EDR for each kind of charge may need to be evaluated separately, leading to a different fault probability. The overall PLA cell fault probability can then be evaluated as

$$F_{cell} = 1 - \Pi_{x \in \text{all fault types}}(1 - f_x) \quad (3)$$

Computing the fault probability of each row and column in a PLA is straightforward once the fault probability for each cell is known, and we omit the details. From the fault probability computation, it is clear that the EDR is critical for estimating the fault tolerance of a PLA, and the value of C_x can significantly impact the yield.

4. YIELD ANALYSIS

In this section, we discuss the behavior of the PLA yield for different EDRs and PLA sizes. We also examine potential EDRs for a magnetic implementation of QCA. As in other PLA yield analysis work [7], we assume that redundant rows and columns may be

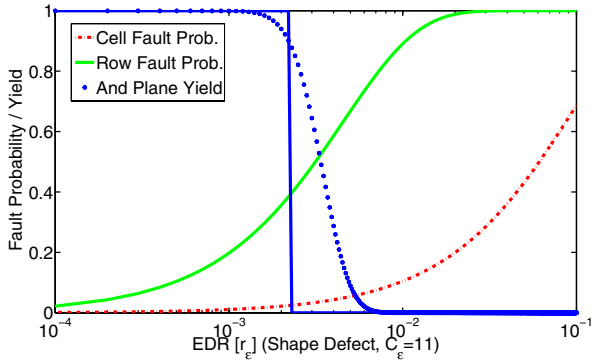

Fig. 5. Example fault/yield vs. EDR curves with the cutoff EDR shown as a vertical line. The PLA AND plane has 20 columns and 20 rows.

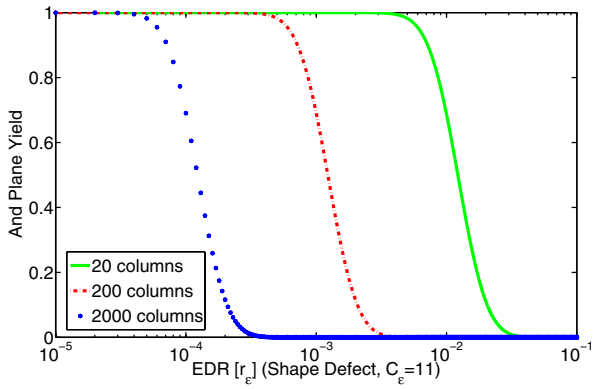

Fig. 6. This plot shows how increasing the input size in the AND plane reduces the tolerated EDRs for the same yield. The AND plane has 10 rows.

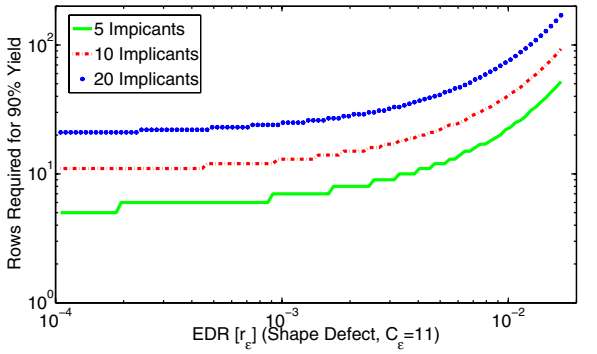

Fig. 7. This graph shows the number of actual PLA rows required to achieve certain number of functional implicants as a function of the EDR. The AND plane has 20 columns.

used. Calculating the QCA PLA yield is straightforward using the fault probabilities from Sec. 3.2. The yield of each plane in the PLA can be determined using an M-choose-N calculation based on the row and column fault probabilities. The yield of the whole PLA is the product of the yields of both the AND and OR planes.

4.1. Trends

We first examine the dependency of the fault probability and yield on the EDR. The fault probability for a single PLA cell grows exponentially as the EDR increases (Eq. 1 and 2). For different types of defects and also for PLA cells in the AND and OR planes, the growth rates can be different depending on the C_x values in Table 2. The fault probability for a row or column grows even faster as EDRs increase due to the multiplicative effect of multiple PLA cells. Consequently, the yield of a PLA falls off quickly as the EDR increases.

Fig. 5 illustrates the above behaviors with three different curves. Here, the misshape defects (ϵ) are considered with $C_\epsilon = 11$, the x-axis is the EDR of ϵ, while the y-axis shows both the fault probability and yield. The fast growth of the fault probabilities can be seen from the dot-dash line depicting f_ϵ for one cell and from the gray line for one row. The AND plane yield is shown as a sequence of dots. It is clear that after the EDR passes a certain threshold, the yield quickly drops off. To quantify this phenomenon, we define the "cutoff EDR" to be the highest EDR that gives a yield of $\geq 90\%$, represented by a vertical line.

EDR is not the only factor that has a significant effect on yield. The number of inputs and outputs in each plane must also be considered. Fig. 6 shows how the dependency of the AND-plane yield on the EDR changes as the number of inputs changes when the number of rows (i.e., the output of the AND-plane) is fixed. Here, the defect is a misshapen device, the x-axis is for EDR, and the y-axis is for yield. The three yield curves correspond to 2000, 200, and 20 input columns from left to right, respectively. As can readily be seen, as the number of inputs increases, the yield curve shifts to the left and the cutoff EDR decreases. In fact, the cutoff EDR is a linear function of the number of inputs.

Given the number of inputs, implicants, and outputs of the intended logic functions, the designer must select the actual size of the PLA (i.e., the number of rows and columns) in order to ensure a predetermined yield. The selection is critically dependent on the EDRs. The fault probability and yield analysis discussed above facilitates the selection process. We illustrate this with Fig. 7, which plots the number of rows needed to achieve 90% yield as a function of EDR. Again, misshape defects are considered. Three different curves are depicted corresponding to 5, 10 and 20 required implicants from bottom to top, respectively. As the EDR increases, the number of required rows also increases. The curves begin to grow rapidly as the EDR approaches the cutoff EDR.

4.2. Case Study: Magnetic QCA

This section contains an initial study of how implementation factors may impact EDRs. We choose a magnetic implementation of QCA to leverage existing experimental data, and because there are clear-cut application spaces for this implementation of the PLA design. Based on our preliminary analysis, we believe that EDRs can be sufficiently low.

Since nanomagnets are physically realized with electron-beam lithography, shifts and misshapenness caused by lithographic error are the most likely sources of device imperfection. Fig. 8a illus-

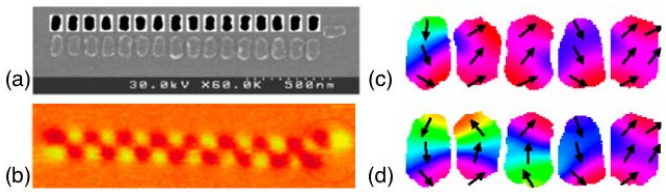

Fig. 8. (a) SEM image of nanomagnet line (from [4], with author permission), (b) MFM image showing state of line, (c)-(d) simulations based on digitized images.

trates a scanning electron microscopy (SEM) image of a line of nanomagnets driven by a magnet at the right. Fig. 8b is the corresponding magnetic force microscopy (MFM) image that provides information as to the "state" of each magnet. The shadings (which represent polarization direction) suggest that this wire is in the correct ground state "driven" by the input magnet. Close observations reveal that all the magnets deviate from the ideal rounded rectangle shape, yet the wire works. However, MFM cannot show how the state associated with this wire evolves in time – it can only provide a "before and after" picture. Understanding how the states of the magnets evolve in time is important because the wire may have reached the correct state by chance instead of by the influence of the input. That is, experimental observations alone may not lead to correct EDR estimates.

We must leverage physical-level simulations to study EDRs. We briefly discuss two wire segments that have been analyzed with the Object Oriented Micromagnetic Framework (OOMMF) [16]. To mirror actual experimental results, we digitized the individual magnets shown in the SEM image of Fig. 8a (see inset), chose a subset of those images to build a wire, and simulated that wire to determine how the state would evolve when (a) the magnets contained a remanent magnetization from a previous computation, and (b) a new input was applied.

Two simulation results are reported. Both are wire segments constructed using five shapes from the digitized images scaled to represent magnets that are approximately 60nm $\times$ 90nm with 10nm spacing. (Initial experiments in [4] used slightly larger magnets only because of equipment limitations.) While a detailed discussion of these results is beyond the scope of this paper, we do make the following observations: In Fig. 8c, the data does not propagate completely and correctly from left-to-right (arrows indicate polarization, and the rightmost magnet should have arrows pointing upwards). Magnet 3 does not become adequately polarized, and causes coupling problems in the rest of the wire. We note that magnet 2 was more than 10% shorter than the other four magnets and that the edge of magnet 4 is slanted – creating a greater distance between magnets 3 and 4. Both of these characteristics weaken the coupling between magnets which helps to induce this incorrect result. (Digitization may be a simulation source of error too.) Interestingly, these coupling problems can be overcome simply by shifting magnet 2 up such that it couples better with magnet 3. As seen in Fig. 8d, the wire is now driven to the correct ground state and the magnets also switched in the proper order.

The above results indicate that magnetic implementations can tolerate a significant amount of misshapenness, though the tolerance depends on other factors such as location. Therefore, estimating EDRs is not trivial, but EDRs can be significantly lower than the probability of simply having defects.

5. CONCLUSIONS AND FUTURE WORK

We have presented a fault model unique to QCA and have applied it to a redundant PLA design. Yield results discussed in Sec. 4.1 combined with the fact that nanomagnets can tolerate misshapenness suggest that magnetic QCA could be used to implement the PLA design with high yields. Ongoing work involves the fabrication of new lines of magnets which will be analyzed, digitized, and simulated to provide initial numerical data with regard to expected EDRs. We will study how the clocking field evolves in time as this can affect switching. Also, the wire segment illustrated in Fig. 8c did not switch correctly because remanent magnetization was not removed by the clocking field. We will consider the trade off of a stronger field with the expense of increased power consumption. We will also explore additional programming techniques to increase fault tolerance.

6. REFERENCES

[1] S. Hu, *et al.*, "PLAs in quantum-dot cellular automata," *Proceedings of International Symposium on VLSI*, March 2-3, 2006.

[2] I. Amlani, *et al.*, "Digital logic gate using quantum-dot cellular automata," *Science*, vol. 284 no. 5412, pp. 289–291, 1999.

[3] H. Qi, *et al.*, "Molecular quantum cellular automata cells. electric field driven switching of a silicon surface bound array of vertically oriented two-dot molecular quantum cellular automata," *J. Am. Chem. Soc.*, vol. 125, pp. 15 250–15 259, 2003.

[4] A. Imre, *et al.*, "Majority logic gate for magnetic quantum-dot cellular automata," *Science*, vol. 311 no. 5758, pp. 205–208, January 13, 2006.

[5] M. H. Kryder, "Magnetic recording beyond the superparamagnetic limit," *IEEE International Magnetics Conference*, p. 575, 2000.

[6] M. Mitic, *et al.*, "Demonstration of a silicon-based quantum cellular automata cell," *Applied Physics Letters*, vol. 89, July 5, 2006.

[7] 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.

[8] 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.

[9] G. Snider, *et al.*, "Nanoelectronic architectures," *Applied Physics A*, vol. 80, pp. 1183–1196, 2005.

[10] 2006 MAPLD International Conference, http://klabs.org/mapld06/.

[11] C. Lent and P. Tougaw, "A device architecture for computing with quantum dots," *Proc. of the IEEE*, vol. 85, p. 541, 1997.

[12] P. Tougaw and C. Lent, "Logical devices implemented using quantum cellular automata," *J. of App. Phys.*, vol. 75, p. 1818, 1994.

[13] 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.

[14] M. Niemier, *et al.*, "Using CAD to shape experiments in molecular QCA," *Int. Conf. on Comp. Aided Design*, pp. 907–914, Nov 2006.

[15] F. Somenzi and S. Gai, "Fault detection in programmable logic arrays," *Proceedings of the IEEE*, vol. 74, no. 5, pp. 655–667, May 1986.

[16] M. Donahue and D. Porter, "OOMMF user's guide, version 1.0, interagency report NISTIR 6367," http://math.nist.gov/oommf.