HDLQ: A HDL Environment for QCA Design

MARCO OTTAVI, LUCA SCHIANO, and FABRIZIO LOMBARDI
Northeastern University
and
DOUGLAS TOUGAW
Valparaiso University

Emerging technologies have attracted a substantial interest in overcoming the physical limitations of CMOS as projected at the end of the Technology Roadmap; among these technologies, quantum-dot cellular automata (QCA) relies on different and novel paradigms to implement dense, low power circuits and systems for high-performance computing. As applicable to existing technologies, a hierarchical process can be utilized to facilitate the design of QCA circuits. Tools and methodologies both at system and physical levels are required to support all design phases. This article presents an HDL model to describe QCA "devices" (also referred elsewhere in the technical literature as building blocks, i.e., majority voter, inverter, wire, crossover) and facilitate the evaluation of their design. This tool, referred to as HDLQ, allows a designer to verify the logic characteristics of a QCA system, while supporting within a design environment different operational mechanisms (such as fault injection) and the unique features of QCA (such as bidirectionality and timing/clocking partitioning). The applicability of this design environment to various memory circuits for logic and timing verification is presented in detail. Various defective conditions for kinks due to thermodynamic effects and permanent faults due to manufacturing defects are considered for injection.

Categories and Subject Descriptors: J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design (CAD)*; B.6.3 [**Logic Design**]: Design Aids—*Verification*

General Terms: Design, Varification

Additional Key Words and Phrases: QCA, HDL, fault injection, CAD

1. INTRODUCTION

As CMOS technology approaches its fundamental physical limits, recent years have seen extensive research in nanotechnology for manufacturing the next generation of integrated circuits (ICs) [Compano et al. 1999]. New paradigms have been proposed to overcome computational limitations by exploiting

Author's addresses: M. Ottavi (contact author), L. Schiano, F. Lombardi, Department of Electrical and Computer Engineering, Northeastern University, 360 Huntington Ave., Boston MA 02115; email: mottavi@ece.neu.edu; D. Tougaw, Department of Electrical and Computer Engineering, Valparaiso University, 206 Gellersen Hall, Valparaiso, IN 46383.

extremely small feature size and high density. Among emerging technologies, *quantum-dot cellular automata* (QCA) [Lent et al. 1994; Smith 1999] utilizes both Coulombic interactions and quantum mechanic tunneling with innovative techniques that radically depart from the CMOS-based processing model of VLSI. Moreover, QCA not only gives a solution at nano scale, but also offers a new method of computation and information transfer [Orlov et al. 1997] with low-power operation [Timler and Lent 2003]. Recent developments in cell manufacturing (involving the deposition of molecules on a substrate surface) have substantially changed the nature of QCA fabrication [Qi et al. 2003; Jiao et al. 2003]; in terms of feature size, a QCA cell of a few nanometers in size has been fabricated through a molecular implementation by a self-assembly process [Qi et al. 2003]. This QCA fabrication process has received considerable attention, resulting in a very promising progress for patterning molecular QCA cells on functional layouts [Hang et al. 2002, 2003].

In addition to advances in cell manufacturing and fabrication, research in circuit and system-level analysis of QCA has been pursued; different high-level architectures have been proposed, such as memories [Frost et al. 2002; Vankamamidi et al. 2005; Ottavi et al. 2005a, 2005b] and microprocessors [Niemier and Kogge 1999].

High-level modeling and simulation of a relatively small set of cells can be accomplished using tools to provide a design-flow process, as currently available for CMOS. For example, Henderson et al. [2004] compares the QCA design flow to the steps involved in a typical CMOS design process. Similarly to a CMOS top-down design methodology, CAD and simulation tools must be formulated for QCA to address high-level requirements such as verification, validation, and physical-level requirements, for example, the structural model of a cell layout. A few tools based on the solution of quantum equations for cell interactions in QCA (MAQUINAS, QCADesigner [Walus et al. 2003, 2004; Walus 2006], QBART [Niemier et al. 2000; Niemier and Kogge 2001]) have been developed to analyze the layout of small circuits. However, these tools are not suitable when, for example, a circuit-level simulation requires new timing mechanisms, or the number of QCA cells is increased. Therefore, new design environments suitable for CAD implementation must be devised for QCA. These environments should be flexible to include additional and novel features; for example, defect-prone manufacturing at a nanoscale (especially for molecular-based devices) and small feature size should be included in this process. Moreover, it has been shown that for some circuits, timing zones are not necessarily placed in a cascade (one-dimensional) arrangement in the layout, as commonly utilized for the four-phase adiabatic switching process. New arrangements for information flow in QCA require trapezoid rules for generating feedback paths in sequential designs and back-and-forth data movement for storage in an memory line.

The goal of this article is to introduce an HDL-based design tool and associated environment for QCA circuits. This tool allows us to overcome the limitations of current simulators with respect to circuit-level evaluation. The proposed environment, is referred to as HDLQ, is flexible to integrate new features for designing QCA circuits. In particular, the proposed model allows the evaluation of QCA circuits and systems through a structural logic and timing

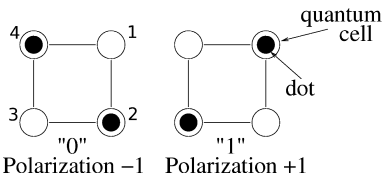

Fig. 1. QCA cell and polarization states.

simulation; specific attributes such as bidirectionality of QCA devices and nontraditional quasiadiabatic switching techniques for timing (as applicable to QCA memory systems) are introduced in the HDL model.

The article is organized as follows: Section 2 provides an overview of QCA; in Section 3, circuit design and the proposed HDL model are introduced in detail. Section 4 presents fault injection with two different types of fault/defect in QCA circuits. Section 5 introduces the application of the proposed tool to timing; HDLQ is used to evaluate different memory cell architectures requiring nonconventional switching for quasiadiabatic operation. Conclusions are provided in the last section.

2. REVIEW OF QCA

A QCA cell can be viewed as a set of four charge containers (or "dots") positioned at the corners of a square (Figure1). The cell contains two extra mobile electrons that can quantum mechanically tunnel between dots, but not cells. The electrons are forced to the antipodal corner positions by Coulomb repulsion. If the two extra electrons are completely localized on dots 1 and 3, the polarization is + 1 (binary 1); if they are localized on dots 2 and 4, the polarization is −1 (binary 0).

Unlike conventional logic circuits, in which information is transferred by electrical current, QCA operates by the Coulombic interaction that connects the state of one cell to the state of its neighbors. The configuration of the polarization of a set of cells reflects the lowest energy state (ground state). This results in a technology in which information transfer (interconnection) is the same as information transformation (logic manipulation) with low-power dissipation [Tougaw and Lent 1994].

QCA can also be characterized at logic level. One of the basic logic gates in QCA is the so-called majority voter (MV) with logic function Maj $(A,B,C) = AB + AC + BC$. MV can be realized by five QCA cells, as shown in Figure 2(1b). Logic AND and OR functions can be implemented from the MV by setting an input (the so-called programming or control input) permanently to a "0" or "1" value. The inverter (INV) is the other basic gate in QCA and is shown in Figure 2(1a). The binary wire and inverter chain (as interconnect fabric) are shown in Figures 2(1c) and (1d), respectively. In VLSI systems, timing is controlled through a reference signal (i.e., the clock), however, timing in QCA is accomplished by clocking in four distinct and periodic phases [Hennessy and Lent 2001] (as shown in Figure 2 (2)). In particular, a QCA circuit is partitioned into *serial* zones, and each zone is maintained in a phase. The use of a quasiadiabatic switching technique for QCA circuits requires a

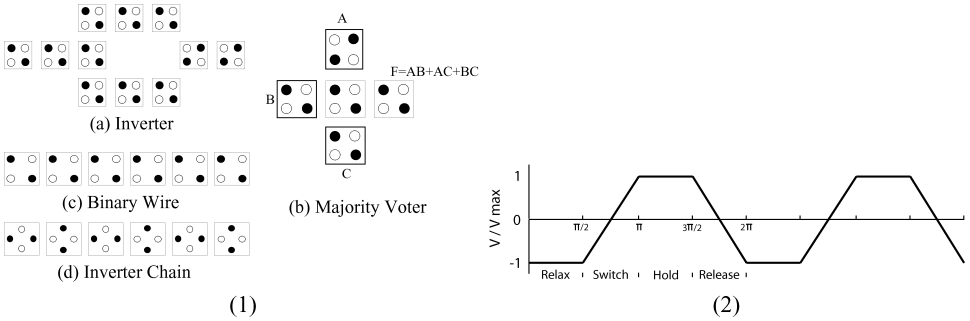

Fig. 2. (1) Basic QCA devices; (2) Cmos clock signal.

four-phased clocking signal, commonly supplied by CMOS wires buried under the QCA circuitry, for modulating the inter-dot tunneling barrier. For quasiadiabatic switching, the four timing phases are Relax, Switch, Hold, and Release. During the Relax phase, there is no inter-dot barrier and a cell remains unpolarized. During the Switch phase, the inter-dot barrier is slowly raised and a cell attains a definitive polarity under the influence of its neighbors. In the Hold phase, barriers are high and a cell retains its polarity. Finally, in the Release phase, barriers are lowered and a cell loses its polarity. A signal is effectively "latched" when one clocking zone goes into the Hold phase and acts as input to the subsequent zone.

3. CIRCUIT DESIGN AND HDL MODEL

A hierarchical process for QCA circuit design has been initially formulated in Henderson et al. [2004]. With an increase in the size and complexity of QCA circuits, a brute-force approach based on directly generating the QCA layout and then simulating its quasiadiabatic switching is not very efficient; Henderson et al. [2004] advocates the use of structured methodologies based on a CMOS-like design process such as the immediate embedding of logic functions. It has been widely reported that a design which directly translates CMOS circuits into QCA is highly interconnection-limited due to the large overhead in area and clocking.

As an environment, a top-down CMOS process usually starts from the conceptual design until a behavioral model is finally utilized for its evaluation. A structural logic model is then generated, followed by a transistor (switch-level) model and layout design. Each step of this process is supported by proper tools both for design and simulation (e.g., HDL tools for the behavioral and structural logic levels, CAD and Spice for transistor-level analysis). Similarly, in Henderson et al. [2004], a multistep process has been proposed for QCA circuit design. This article focuses on the behavioral and structural logic-level steps of the QCA design process, and proposes a novel HDL-based model.

With an increasing interest in QCA devices, a few *technology-dependent* design and simulation tools are currently available, such as MAQUINAS [Tougaw and Len 1996], QCADesigner [Walus et al. 2003; Walus 2006], and QBART [Niemier et al. 2000]. MAQUINAS and QCADesigner simulate QCA

circuits at the cell level with different degrees of complexity in evaluation. Both rely on a simulation engine that solves the Schroedinger equation for modeling physical interactions within the considered set of cells in the circuit; AQUINAS focuses on the simulation of molecular-based QCA, whereas QCADesigner simulates metal-based QCA. A shortcoming of simulation with an engine based on the solution of quantum mechanical equations is its computational complexity; this heavily impacts the simulation time of tools like MAQUINAS and QCADesigner, even when only simple circuits are considered. Other limitations of these QCA simulators are related to their recent development, and the restricted set of optional features that are available to a designer in their current release version. Still, there is a lot of work in progress on these tools, and their fast development and widespread use have been made possible by the open-source nature of these software programs [Walus 2006]. Different from these simulation tools, QBART introduces a level of abstraction that can be utilized to reduce the computational complexity of QCA analysis. QBART defines a grid-like environment in which QCA devices are instantiated by introducing mostly *ad hoc* simulation rules. However, it is limited in execution due to the inflexible nature of the circuit model and its difficulty in changing features such as timing and signal bidirectionality; QBART differs from the generally accepted approach that the adoption of a high-level description (HDL) makes possible the generation of simulation-based models for behavioral and structural logic-level analysis. However, at the circuit level, many hurdles are encountered for QCA design, hence a behavioral model is almost a necessity. In particular, there are three main motivations to justify the development of an HDL-based behavioral model for QCA:

—It reduces the complexity in terms of both design and simulation times of the different processes, inclusive of verification, as involved in QCA design.
—It allows flexibility in timing/clocking analysis; for example, it can be used to analyze clocking schemes that do not follow the strict sequentiality of four-phase clocking for quasiadiabatic switching. Examples of these schemes are encountered for the line-based memory schemes introduced in Vankamamidi et al. [2005] and Ottavi et al. [2005a].
—It allows us to model the testing and validation of a QCA system [Tahoori et al. 2004].
—It permits the introduction of new capabilities in the design process, such as fault injection.

Figure 3 shows the flowchart of the proposed behavioral and logic design environment of HDLQ. A detailed database, of QCA devices is required to provide the physical information. From the database, for example, manufacturing defect models and a library of QCA devices can be extracted; a complete QCA library inclusive of fault models can be established through a low-level simulation performed using available tools, such as QCADesigner and MAQUINAS. Previous works have reported novel QCA devices and defects that can be included in this library [Huang et al. 2005]. An extended set has been considered in HDLQ to provide the designer with a

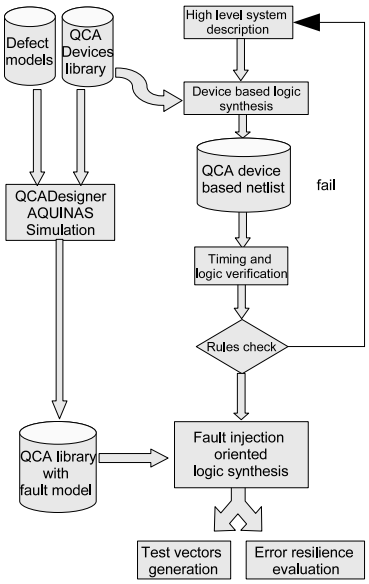

Fig. 3. QCA design flowchart.

wide range of options for selecting QCA devices at design time. In particular, the set considered in the HDLQ library is derived from that generated in Momenzadeh et al. [2005] and departs from similar sets considered in previous works [Berzon and Fountain 1999; Henderson et al. 2004]. As for the defect models, the reference model is that reported in Momenzadeh et al. [2005] and established through QCADesigner-based simulations.

By utilizing an HDL simulation approach, the design flow can then proceed with timing and logic analysis inclusive of verification and by utilizing the device library (Figure 3). The basic operation of HDLQ is as follows: After partitioning the design into QCA devices, the Verilog code is built with primitive block models. Input and clock waveforms are then handled in the test-bench file. Following this step, HDLQ allows the designer to proceed with a fault injection-oriented logic synthesis. The HDL QCA library inclusive of fault models provides the designer with the capability of verifying the QCA design and assessing relevant features, such as obtaining a test set, validating its defect resilience, and checking its timing/clocking mechanism, as required, for instance, for QCA memories [Ottavi et al. 2006].

VHDL and Verilog are the most widely used HDL standards. Verilog HDL has been utilized in this work to build a QCA device library for assembling different circuits with the objective of using it for fast assembly of larger designs. In Henderson et al. [2004], a QCA design environment was introduced by using a VHDL-based simulation tool in which QCA blocks can be assembled and simulated in their own clocking zones. However, in Henderson et al. [2004], the VHDL model is limited mostly to fixed timing features for simulating QCA circuits; moreover, its structure does not include additional features such as fault injection. This work expands on Henderson et al. [2004] by

generating a comprehensive HDL engine to allow for simulation of new timing schemes [Vankamamidi et al. 2005; Ottavi et al. 2005a], and for runtime injection of defects, permanent faults [Momenzadeh et al. 2005], and transient faults caused by thermodynamic effects, which result in kinks on long QCA wires. Furthermore, the proposed environment allows flexibility in QCA modeling to include bidirectionality of devices; this feature was not addressed in Henderson et al. [2004]. These features of HDLQ are described in detail next.

3.1 Bidirectional Flow and Timing Simulation

One of the most challenging aspects in describing QCA devices using HDL is information flow with respect to the definition of the timing/clocking mechanism. Different from CMOS, QCA devices are bidirectional, thus requiring careful handling when used with the fixed input/output structure of HDL modules. This characteristic is important when novel clocking schemes are applied to QCA, mostly at the circuit level [Vankamamidi et al. 2005; Ottavi et al. 2005a]. Therefore, different from Henderson et al. [2004], the library in HDLQ includes both unidirectional and bidirectional implementations, as well as the relevant logic and physical features of each QCA device. The bidirectional modules rely on the extensive definition of "inout" ports; they are organized such that simultaneous occurrences (i.e., concurrence) of opposite signals on the same net never happen, irrespective of the adopted design structures. This conforms to the actual behavior of QCA circuits because QCA devices are set to an undetermined value prior to the switch phase.

As for timing, a continuous-time simulation is not easy to implement in Verilog. However, a model of the signal propagation mechanism through different timing zones has been proposed and implemented. In the proposed environment, every QCA device from the device set introduced in Momenzadeh et al. [2005] is described by a Verilog module that is defined in its clock state by an input (four-valued clock) variable. Under a quasiadiabatic switching mechanism, this is given as follows.

—When a device is in the Release or Relax phase, its outputs are in a high-impedance state (denoted by Z).
—When the Switch phase is entered, computation is performed, and the logic outputs reflect the operation on a time-dependent basis for the inputs of the device.
—When the device is in Hold phase, the last attained logic value is locked at the output for the duration of the whole phase.

A detailed description of the code implementing a majority voter is provided in Ottavi et al. [2006] for the interested reader.

3.2 Fault Injection

As stated in Clark and Pradhan [1995], "[F]ault-injection involves the deliberate insertion of faults or errors into a computer system in order to determine its response. It has proven to be an effective method for measuring the parameters of analytical dependability models, validating existing fault-tolerant systems,

synthesizing new fault-tolerant designs, and observing how systems behave in the presence of faults." This statement provides the motivation for fault injection to nanotechnology-based systems; it is widely acknowledged that systems at the nano scale will have a high probability of failure due to manufacturing defects or thermal runtime faults. A capability for fault injection has been introduced in the proposed simulation engine and design environment to properly simulate the effects of manufacturing defects [Momenzadeh et al. 2005]. This has also been used to perform a time-dependent fault injection of a transient fault that is modeled by the so-called kink effect (i.e., the inversion of a logic value on a line).

Fault injection through the HDL model is accomplished at the logic level. Starting from the fault set [Momenzadeh et al. 2005], the Verilog model of each QCA device is characterized by a number of auxiliary inputs; this is required for unambiguously identifying the effects of faults on a device that is not fault-free. As an example, consider straight or L-shaped wires in QCA. As shown in Momenzadeh et al. [2005], these wires can only be affected by one type of logic fault, namely, the inversion of the output. Therefore, only one more input signal is added to the block for discriminating between fault-free and faulty conditions. The majority voter can be affected by two types of logic fault [Momenzadeh et al. 2005]. Therefore, two inputs are necessary to distinguish the fault-free condition from the two faulty behaviors, namely, $fault0$ and $fault1$. So, ($fault0 = 0$, $fault1 = 0$) is the fault-free configuration; ($fault0 = 0$, $fault1 = 1$) is the stuck-at-B fault, modeling a missing cell deposition defect on one of the other input cells; ($fault0 = 1$, $fault1 = 0$) is the fault changing the MV function to $F = MV(A', B, C') = A'C' + A'B + C'B$ for modeling a missing cell deposition defect in the central cell of the MV.

4. FAULT INJECTION

Hereafter, two examples of fault injection are reported; sample layouts are used to identify the capabilities of HDLQ for fault injection. The first example is the simulation of a time-dependent transient fault on a long wire due to a kink induced by thermal noise. The second example reports the simulation of a layout of an XOR circuit affected by a permanent defect, such as an additional/missing cell placement at manufacturing. Fault injection allows us to define the impact of faults of different nature on a QCA design, as well as to determine the test vectors for detection (as shown in Figure 3).

4.1 Kink On a QCA Wire

To show the effects of a fault of transient nature, a QCA circuit with three long wires as inputs to an MV is considered; the fault results from the occurrence of a kink due to thermal effects. It is assumed that the wires cross multiple clocking zones.

The Verilog model allows injecting the transient fault on a long line by setting an input variable "fault" on the desired wire-type module. Due to its transient nature, and to correctly model a kink, the fault signal should be set to a bit value of 1 in the test-bench file for the duration of a Switch phase. In Figure 4,

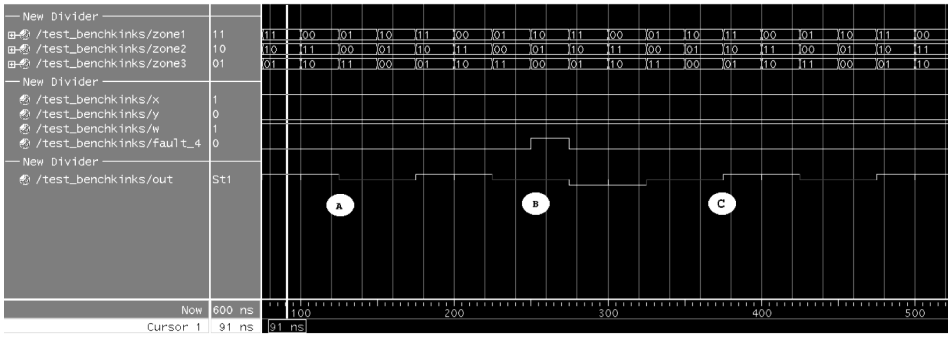

Fig. 4. Waveforms showing the effect of a kink in long wires.

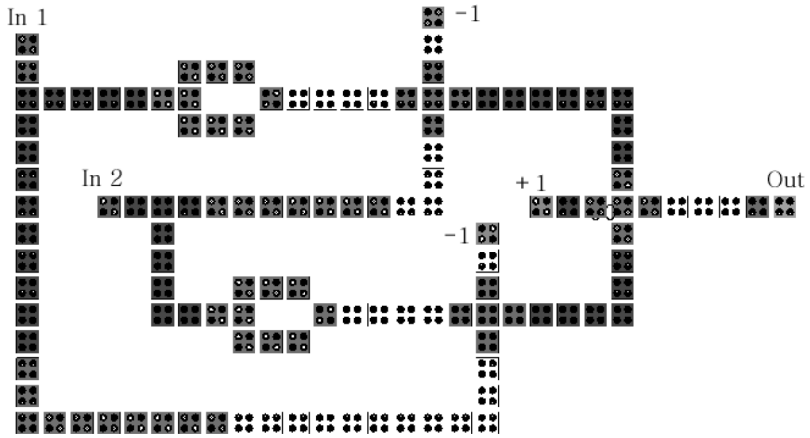

Fig. 5. XOR circuit in QCA.

the fault-free condition is marked by "A": The inputs x and w are at bit value of 1, while the input y is at bit value of 0; thus the majority output is at bit value of 1. When a kink ("B" in Figure 4) is issued through a signal *fault*_4, this causes the occurrence of a transient fault on a QCA wire (and therefore, its value is complemented prior to reaching the connecting device); in this case, the output temporary switches to a bit value 0 ("B" in Figure 4), before taking again a bit value of 1 when the kink effect is over ("C" in Figure 4). This shows that HDLQ permits us to model the effects of faults of a transient nature, while capturing their logic behavior.

4.2 Permanent Defect in a QCA Circuit

As an additional example, the XOR circuit in QCA is shown in Figure 5 [Momenzadeh et al. 2005]. Its logic block diagram is given in Figure 6; this diagram also shows the timing zones and all QCA devices. A complete fault analysis of this circuit has been presented in Momenzadeh et al. [2005]. Consider the set of possible faults for the majority voter MV3. In Momenzadeh et al. [2005], it has been shown that an additional cell deposition defect does not cause a

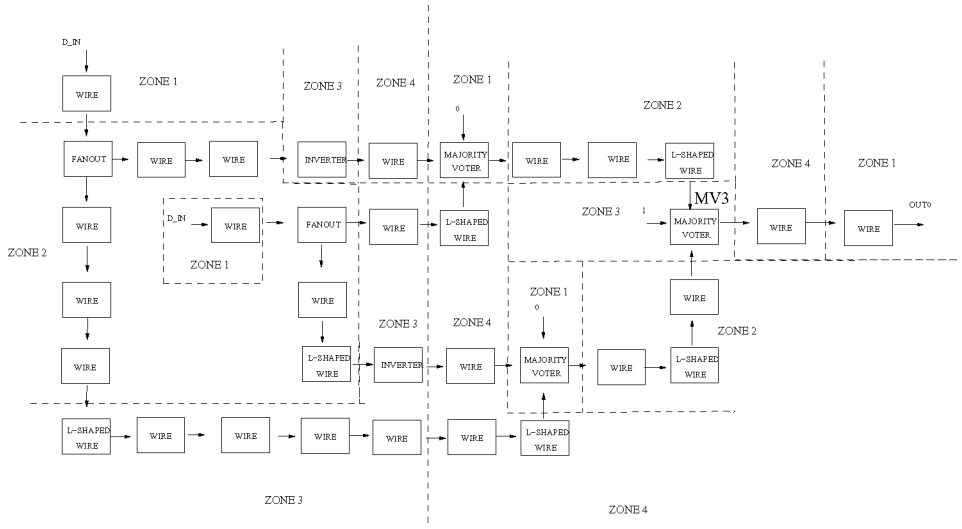

Fig. 6. Block diagram of XOR circuit in QCA.

faulty behavior in the MV. However, a missing cell deposition defect can affect functionality of the MV ($F(MV)$) in two different ways:

(1) $F(MV) = B$, where B is the polarization of the QCA input cell located in a symmetric position with respect to the output cell of the MV.
(2) $F(MV) = (A', B, C')$, instead of the fault-free $F(MV) = (A, B, C) = AB + CB + AC$.

Both of these logic faults propagate to the XOR primary output and can be tested by using the vector set $D\_in1D\_in2 = (00, 11)$.

Figure 7 shows the effect of a permanent fault that changes the functionality of MV3 in Figure 6 into a stuck-at-B, where $B = 1$; the bottom part of Figure 7 refers to the fault-free case. When the input vector 00 is sampled (this event is denoted as "A" in Figure 7), after the required latency, the output is given by signal $m34$ in Figure 7 and takes a value of 0. The behavior of the faulty circuit is shown in the top part of Figure7; the output remains at the erroneous value of 1.

Similarly, the behavior of the circuit can be simulated by considering defects in other QCA devices.

5. TIMING SIMULATION

Examples of the application of the proposed HDL methodology to timing simulation of memory elements are reported in this section so as to outline in detail the applicability of the proposed HDL environment. As representative of different timing features, memory cells for three QCA-based memory designs have been considered.

—A loop scheme for serial write/serial read memory architectures similar to Berzon and Fountain [1999] and Walus et al. [2003].

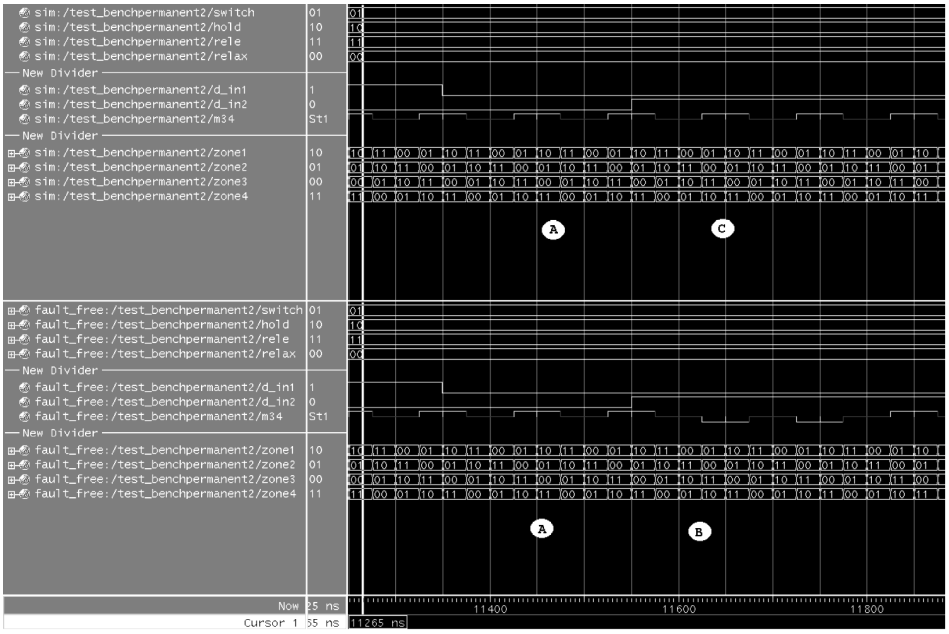

Fig. 7. Waveforms obtained by injecting a permanent fault on the output MV of XOR.

—A hybrid memory architecture for serial write/parallel read operations [Ottavi et al. 2005a].

—A line-based memory [Vankamamidi et al. 2005; Ottavi et al. 2005a].

For each of these architectures, the following features are presented:

—A layout of the basic cell.

—A block diagram showing the organization of the module in a Verilog-based simulation with the HDLQ library introduced previously in this article.

—The waveforms obtained by the Verilog simulation in ModelSim.

5.1 Loop-Based Memory Cell

Figure 8 shows the QCA implementation of a memory cell based on the classic "memory-in-motion loop" structure. QCA-based realizations of similar architectures have been presented in Berzon and Fountain [1999] and Walus et al. [2003] for serial write/parallel read memories. Only the core of the memory cell is considered, that is, the addressing circuitry is not considered.

Clocking zones are derived from the same clock (shown in Figure 2(2)), with a phase displacement of $\frac{\pi}{2}$. Therefore, when Zone 1 is in Hold, Zone 2 is in Switch, Zone 3 is in Release, and Zone 4 is in Relax. The loop is realized such that once the information bit enters the loop, it is forced to traverse the QCA devices in all four phases, as indicated in Figure 9. Figure 10 shows the waveforms obtained by simulating the Verilog model of the serial memory loop with ModelSim [Graphics 2006]. After having initialized to zero the bit stored in the loop, the following repetitive sequence of operations takes place to verify the

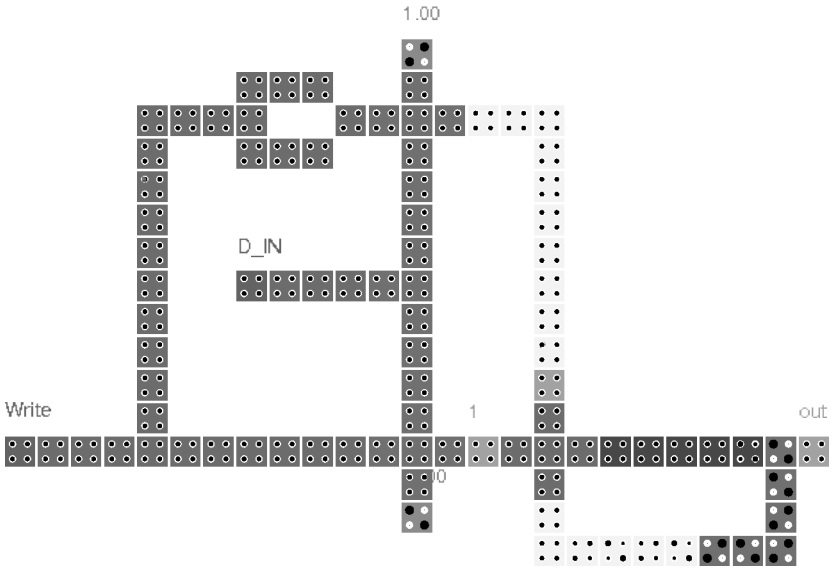

Fig. 8. QCA loop-based memory cell layout. Each shade corresponds to a different clocking zone.

correctness of the model:

—Write a 1 (this event is denoted as "A" in Figure 10);
—store the previous value, allowing a possible read operation being performed in the same clocking cycles ("B" in Figure 10);
—write a 0 (denoted as "D" in Figure 10); and
—store the previous value, thus allowing, (again) a possible read operation.

As shown in Figure 9, the information bit (d_in) is latched when Zone 1 is in the Switch phase, and it enters the loop after the MV operation. This last operation occurs when Zone 4 is in the Switch phase (four clock cycles after the clock's command) and the loop's output (out) has been arbitrarily assigned to a QCA device in the second clocking zone. Two additional clock cycles must be added to the write operation for latency. Therefore, the bit value 1 is written to out six clock cycles after the "write a 1" instruction is given to the serial memory ("C" in Figure 10). Similarly, bit value 0 is written to out six clock cycles after the "write a 0" instruction is given to the memory circuit; these events are denoted in Figure 10 as "E" and "D," respectively.

5.2 Hybrid Memory Cell

Figure 11 shows the QCA realization of a hybrid memory (serial write/parallel read) [Ottavi et al. 2005a]. Similar to the loop structure, data is serially written and stored through the multizone memory loop. In the hybrid memory, $N = 2^n$ words of m bits each (are) organized in m parallel loops, hence capable of storing N bits each, that is, N arrays of four QCA devices, each assigned to a different clocking zone. To perform either the serial write or the parallel read operation, an offset must be added to the input address to select the correct bit (word). A

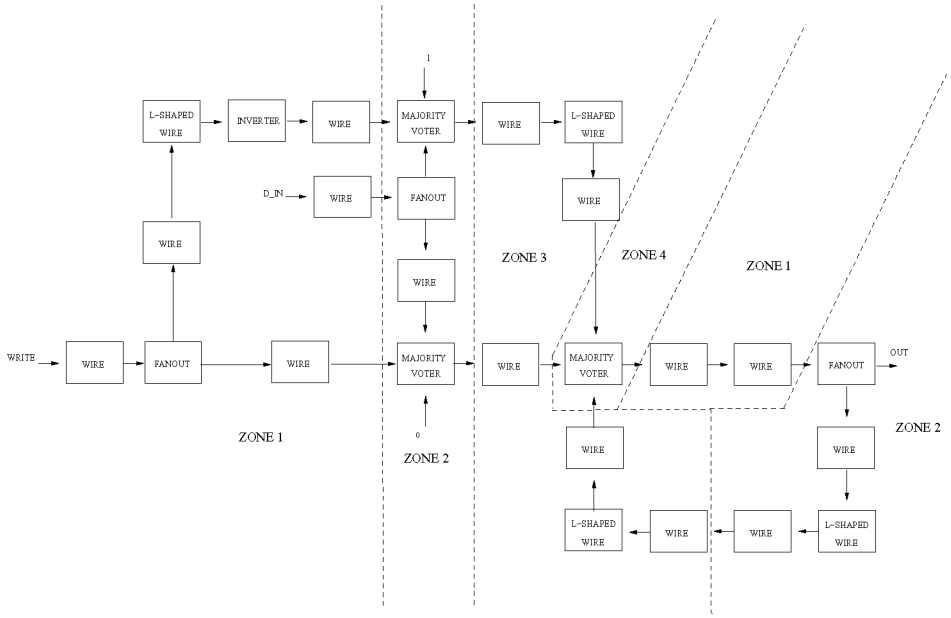

Fig. 9. Block diagram of the loop-based memory model.

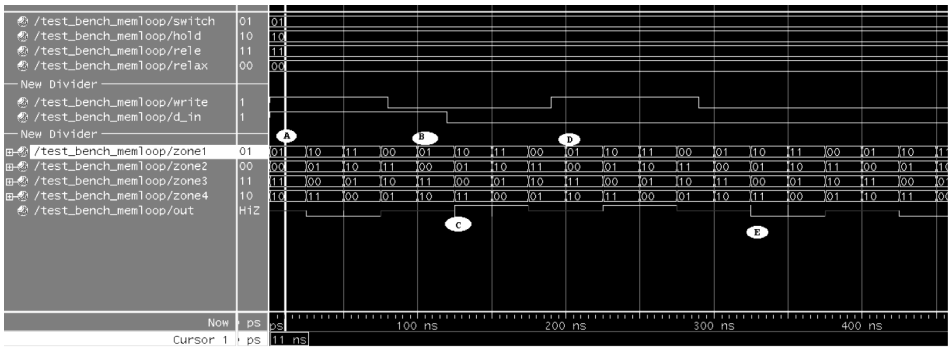

Fig. 10. Waveforms for the loop-based memory model.

2^n-bit counter is therefore used: For a write operation, the desired value in the correct location inside a word can take up to $2^n - 1$ clock cycles and the counter is used to enable the write operation only at the desired cycle; for a parallel read operation of the m-bit word, the counter is used to select desired outputs in the 2^n-bit loop structures.

Figure 12 shows the schematic implementation of a hybrid-memory cell with Verilog QCA library modules and its organization in terms of clocking zones. As an example, four words ($N = 4$) are considered in the memory; therefore, four bits must be stored in the loop. These bits are made accessible during a read operation. As for the QCA devices used in the loop, one MV connects the loop to the peripheral circuitry, while all other devices are either straight or L-shaped wires. Fanouts are used to generate the outputs ($out[0 - 3]$).

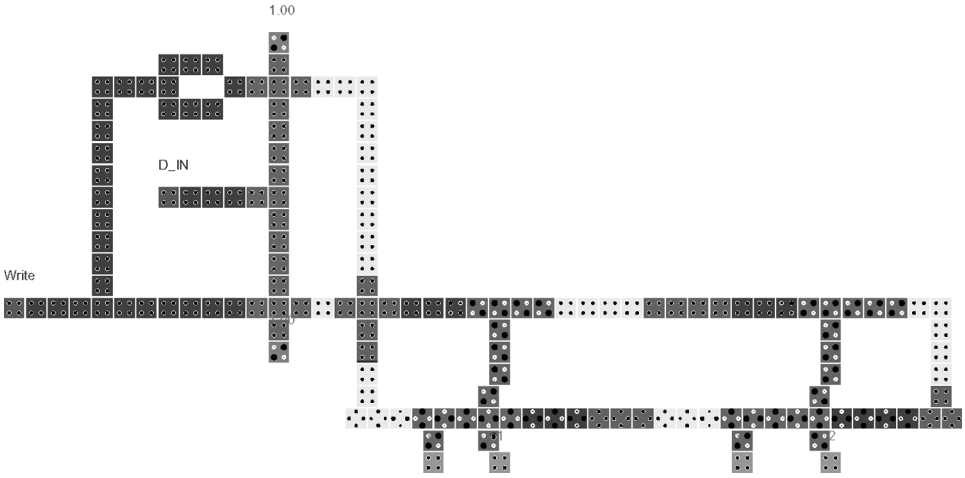

Fig. 11. QCA hybrid-memory cell layout. Each shade corresponds to a different clocking zone.

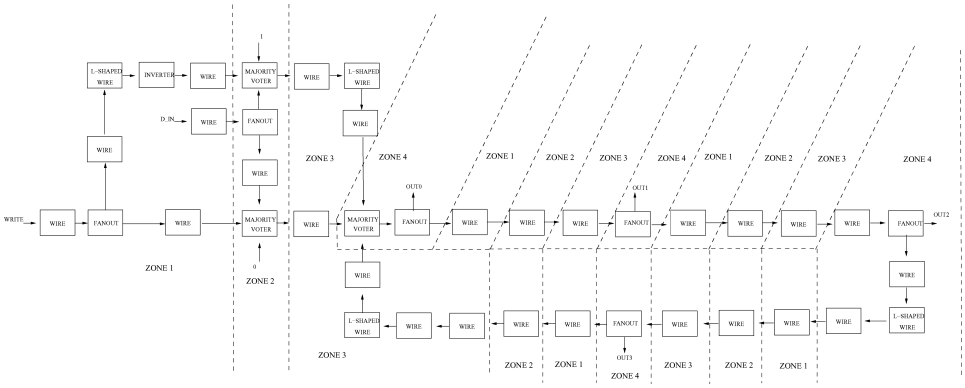

Fig. 12. Block diagram of the hybrid-memory model.

Figure 13 shows the waveforms obtained by ModelSim [Graphics 2006] for the Verilog model of the hybrid memory cell. The four bits stored in the loop are initialized to 0 and then serially written to a bit value of 1. The signal $bits\_stored$ consists of the concatenation of bits $out[0-3]$; the four bits belong to four different words and in this case, for ease of presentation, they are read together. As shown in the waveforms, there is a latency prior to writing a bit in the memory after the write signal is issued. In this simplified model, the addressing circuitry is not fully described [Ottavi et al. 2005a]; the input data (d_in) requires three full clock cycles for propagation to the first memory cell bit corresponding to $out0$ in Figure 12, that is, this is equal to the distance in cycles between a Zone 1 in a Switch phase and the following Zone 4 in a Switch phase. To write the nth bit of the memory cell, $4(n-1)$ clock cycles must be added to this initial offset. As an example, in Figure 13, $3 + 3 \times 4$=15 clock cycles are required to write the fourth bit of the memory when $bits\_stored$ reaches the value "1111".

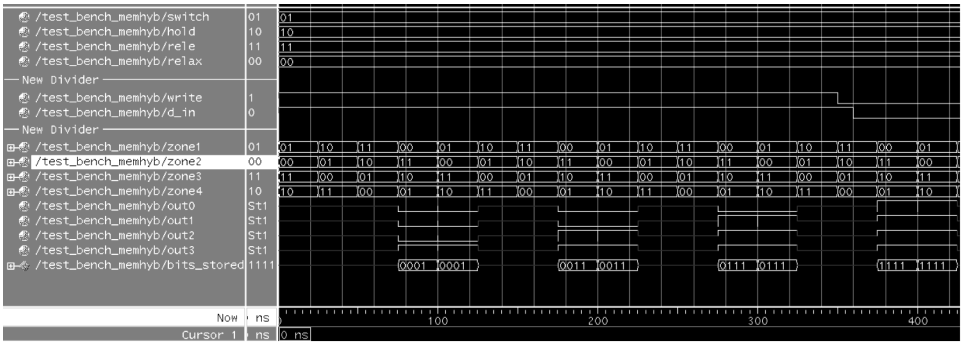

Fig. 13. Waveforms for the hybrid-memory model.

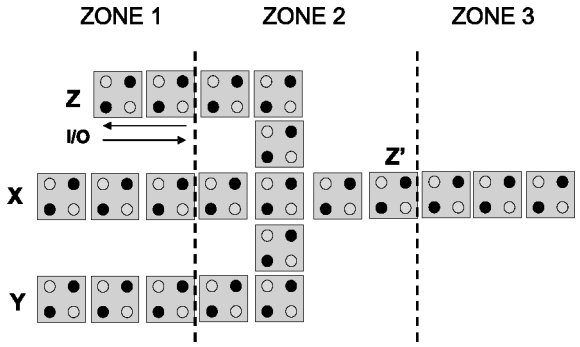

Fig. 14. QCA line-based memory cell layout. Each shade corresponds to a different clocking zone.

5.3 Line-Based Memory Cell

Figure 14 shows the QCA circuit for realizing the storage element in a line-based memory. The clock signals for the required switching mechanism are also provided. This design relies on the bidirectional nature of QCA devices. The MV ports are connected to Z and Z' and are used either as input or output, according a to the phase of the timing zone and the neighbors of the MV. The behavior of this cell can be described as follows for the four steps of a memory operation [Vankamamidi et al. 2005].

—*Step* 1: Zone 1 and Zone 2 are in the Switch phase, while Zone 3 is in the Hold phase. The stable inputs X and Y come from an adjacent clocking zone in the Hold phase and propagate to the MV. At the same time, Z' behaves as an input to the MV; in this case, Z assumes the value given by by the triplet (X, Y, Z'). The value of Z' represents the value stored in the memory cell after a previous write operation, that is, in the Hold phase for Z (if X and Y have different values), or overwritten to a new value (for both X and Y of the same value).

—*Step* 2: While Zone 3 is in the Relax phase, Zone 1 goes into the Hold phase, and Zone 2 is still in the Switch phase. Therefore, the value that was written

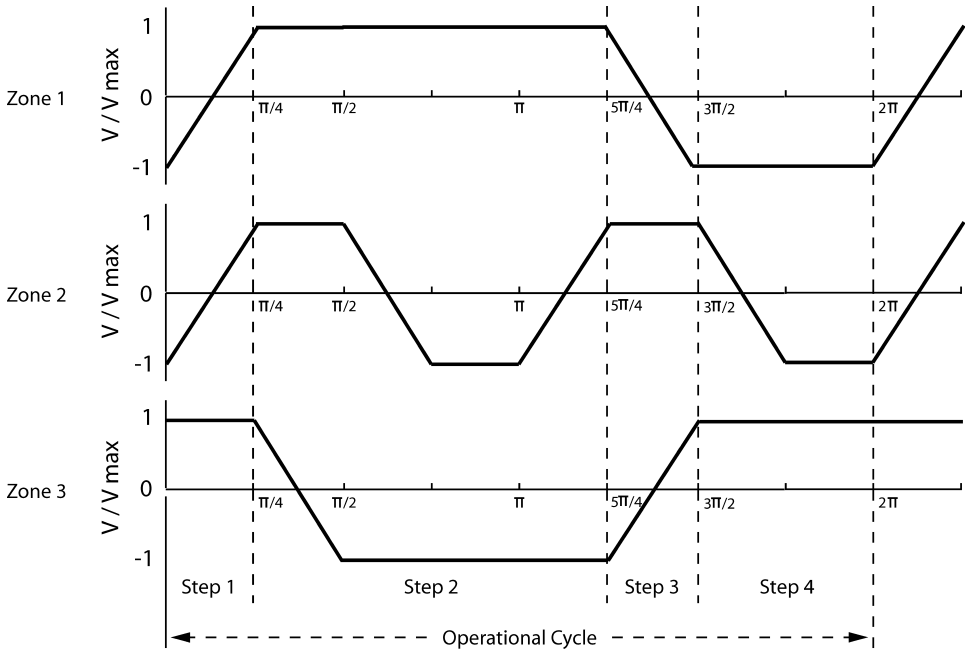

Fig. 15. Clocking scheme for QCA line-based memory cell.

in Z during the previous step is propagated to Z', that is, it is ready to be an output in the next read operation.

—*Step* 3: Zone 1 is in the Relax phase, while Zone 2 is in the Hold phase. Zone 3 is in the Switch phase, therefore Z' is propagated to the output stage of the memory cell.

—*Step* 4: In this step, Zone 3 is in the Hold phase, and the value stored in the cell is available for a possible read operation.

The clocking arrangement is shown in Figure 15 and implemented in HDL by a simple program for its simulation, while neither QCADesigner [Walus et al. 2003; Walus 2006] nor AQUINAS [Tougaw and Len 1996] can be utilized.

Figure 16 shows the partition of the QCA memory cell as introduced in a previous section. All blocks denoted by "bi" correspond to bidirectional realizations from the HDLQ device library. Bidirectional devices have been used for the MV in the memory cell and all transmission paths to nodes Z and out. This last node connects the cell to the read circuitry. After partitioning the design into devices, the Verilog code is built with primitive block models. Input and clock waveforms made of different duty cycles for the considered clocking zones are then handled in the test-bench file.

The resulting waveforms from ModelSim [Graphics 2006] are shown in Figure 17. Signals $zone[1-3]$ refer to the corresponding clocking zones, while X and Y are the inputs, Z is the node in which the bit is stored after the MV, and out is the cell output. As shown in Figure 17, the operations performed on

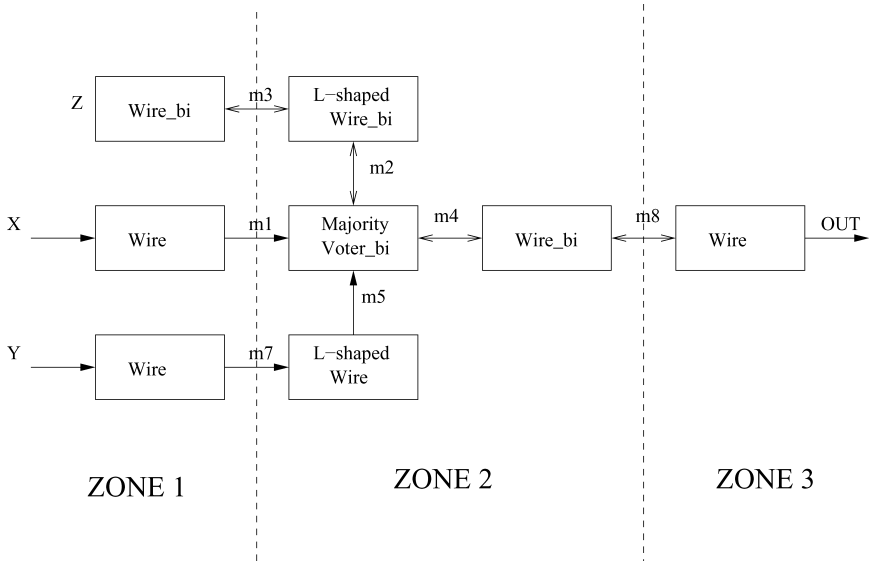

Fig. 16. Block diagram of the line-based memory model.

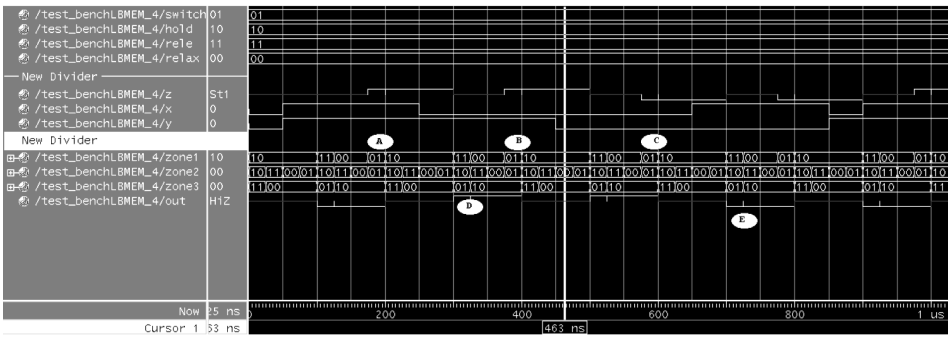

Fig. 17. Waveforms for the line-based memory model.

a memory cell are as follows:

—Write a 1 as bit value (event denoted in Figure 17 as “A”);
—retain the stored value to allow a possible read operation (“B” in Figure 17);
—write a “0” (“C” in Figure 17); and
—retain the stored value.

The output (*out*) is latched and, therefore, ready for a possible read operation when Zone 3 goes in the Switch phase. Since the clocks connected to different zones have different duty cycles, the clock signal of “Zone 2” is considered as a reference, its selection having been based on the fact that it is the only signal with a 50% duty cycle. Therefore, the output is written with a bit of value 1 (“D” in Figure 17) six clock cycles after the “write a 1” operation (“A”). Similarly, the

memory is reset to a bit of value 0 ("E" in Figure 17) after six clock cycles of latency (computed from event "C" in Figure 17).

6. CONCLUSION

Quantum-dot cellular automata is a new paradigm for nanoscale systems; QCA is radically different from CMOS VLSI technology. However, a design process flow similar to CMOS can be considered for QCA [Henderson et al. 2004]. Even though tools have been developed for designing QCA circuit layouts, such as QCADesigner, the capability of utilizing a well-known and widely accepted methodology at a higher level can be beneficial for designing complex QCA circuits.

This work has introduced an HDL-based tool to model QCA circuits at the logic level, and this tool is referred to as HDLQ. In HDLQ, a library of QCA devices has been developed in Verilog by allowing the designer to simulate large QCA circuits. By maintaining technology-related features, such as bidirectionality, and proper fault characterization, this tool guarantees fast verification at the behavioral/structural level. It also allows timing simulation based on the simultaneous use of clocking schemes with different frequency and duty cycles for the zones. This feature at present is not available in current simulators, such as QCADesigner.

Furthermore, HDLQ provides the ability to inject faults and defects, for example, by utilizing the QCA device fault set defined in Momenzadeh et al. [2005]. This feature is required for fault-tolerance design and can be used to generate test sets for a QCA circuit.

REFERENCES

BERZON, D. AND FOUNTAIN, T. J. 1999. A memory design in QCAS using the Squares formalism. In *IEEE Proceedings of the 9th Great Lakes Symposium on VLSI* (Ann Arbor, MI), 166–170.

CLARK, J. A. AND PRADHAN, D. K. 1995. Fault injection: A method for validating computer-system dependability. *Comput.*, 47–56.

COMPANO, R., MOLENKAMP, L., AND PAUL, D. J. 1999. Technology roadmap for nanoelectroincs. In *Proceeding of the European Commission IST Programme, Future and Emerging Technologies, Conference*.

FROST, S. E., RODRIGUES, A. F., JANISZEWSKI, A. W., RAUSCH, R. T., AND KOGGE, P. M. 2002. Memory in motion: A study of storage structures in QCA. *1st Workshop on Non-Silicon Computation*.

Graphics, M. 2006. Modelism. http://www.model.com.

HANG, Q. L., WANG, Y. L., LIEBERMAN, M., BERNSTEIN, G. H. 2002. Molecular patterning through high-resolution polymethylmethacrylate masks. *Appl. Phys. Lett. 80*, 4220–4222

HANG, Q. L., WANG, Y. L., LIEBERMAN, M., BERNSTEIN, G. H. 2003. A liftoff technique for molecular nanopatterning. *J. Nanosci. Nanotechnol. 3*, 309–312.

HENDERSON, S., JOHNSON, E., JANULIS, J., TOUGAW, P. D. 2004. Incorporating standard CMOS design process methodologies into the QCA logic design process. in *IEEE Trans. Nanotechnol. 3*, 2–9.

HENNESSY, K., AND LENT, C. S. 2001. Clocking of molecular quantum-dot cellular automata. *J. Vaccum Sci. Technol. 19*, 5, 1752–1755

HUANG, J., SCHIANO, L., MOMENZADEH, M., AND LOMBARDI, F. 2005. Simulation-based design of modular QCA circuits. In *5th IEEE Conference on Nanotechnology.* (Nagoya, Japan), 687–690.

JIAO, J., LONG, G. L., GRANDJEAN, F., BEATTY, A. M., AND FEHINER, T. P. 2003. Building blocking for the molecular expression of QCA, isolation and characterization of a covalently bounded square array of two ferrocenium and two ferrocene complexes. *J. Amer. Chem. Soc. (JACS Commun.) 125*, 25, 7522–7523.

LENT, C. S., TOUGAW, P. D., AND PORORD, W. 1994. Quantum cellular automata: The physics of computing with arrays of quantum dot molecules. In *IEEE Proceedings of the Workshop on Physiscs and Computing* (Los Alamitos, CA), 5–13.

MOMENZADEH, M., OTTAVI, M., AND LOMBARDI, F. 2005. Modeling QCA defects at molecular-level in combinational circuits. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*. 208–216.

NIEMIER, M. T., MICHAEL, J., KONTZ, AND PETER M. KOGGE. 2000. A design of and design tools for a novel quantum dot based microprocessor. In *Proceedings of the 27th Design Automation Conference*. 227–232.

NIEMIER, M. T. AND KOGGE, P. M. 2001. Problems in designing with QCAs: Layout=Timing. *Int. J. Circuit Theory Appl. 29*, 1, 49–62.

NIEMIER, M. T., KOGGE, P. M. 1999. Logic-In-Wire: Using quantum dots to implement a microprocessor. In *International Conference on Electronics, Circuits, and Systems (ICECS)*, 1211–1215.

ORLOV, A. O., AMLANI, I., BERNSTEIN, G. H., LENT, C. S., AND SNIDER, G. L. 1997. Realization of a functional cell for quantum-dot cellular automata. *Sci. 277*, 928–930.

OTTAVI, M., PONTARELLI, S., VANKAMAMIDI, V., AND LOMBARDI, F. 2005a. Design of a QCA memory with parallel read/serial write. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. 292–294.

OTTAVI, M., PONTARELLI, S., VANKAMAMIDI, V., AND LOMBARDI, F. 2005a. Novel approaches to QCA memory design. In *Proceedings of the 5th IEEE Conference on Nanotechnology* (Nagoya, Japan), 699–702.

OTTAVI, M., SCHIANO, L., LOMBARDI, F. 2006. Description of the majority voter building block in verilog. In Tech. Rep. available upon request. Northeastern University.

OTTAVI, M., SCHIANO, L., PONTARELLI, S., VANKAMAMIDI, V., AND LOMBARDI, F.. 2006. Timing verification of QCA memory architectures. In *6th IEEE Conference on Nanotechnology* (Cincinnati, OH) 343–346.

QI, H., SHARMA, S., LI, Z., SNIDER, G. L., ORLOV, A. O., LENT, C. S., AND FEHINER, T. P. 2003. Molecular quantum cellular automata cells: Electric field driven switching of a silicon surface bound array of vertically oriented two-dot molecular QCA. *J. Amer. Chem. Soci. 125*, 49, 15250–15259.

QI, H., AND SHARMA, S., AND LI, Z. H., AND SNIDER, G. L., AND ORLOV, A. O., AND LENT, C. S., AND FEHLNER, T. P. 2003. 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. 125*, 4, 15250–15259.

SMITH, C. G. 1999. Computation without current. *Sci.* 2, 284–274.

TAHOORI, M., HUANG, J., MOMENZADEH, M., AND LOMBARDI, F. 2004. Testing of quantum cellular automata. *IEEE Trans. Nanotechnol. 3*, 4, 432–442.

TIMLER, J. AND LENT, C. 2003. Maxwell's demon and quantum-dot cellular automata. *J. Appl. Phys. 94*, 1050–1060.

TOUGAW, P. D., AND LENT, C. S. 1994. Logical devices implemented using quantum cellular automata. *J. Appl. Phys. 75*, 3, 1818–1825.

TOUGAW, P. D., LEN, C. S. 1996. Dynamic behavior of quantum dot cellular automata. *J. Appl. Phys. 80*, 4722.

VANKAMAMIDI, V., OTTAVI, M., AND LOMBARDI, F. 2005. A line-based parallel memory for QCA implementation. *IEEE Trans. Nanotechnol. 4*, 6, 690–698.

WALUS, K., DIMITROV, V., JULLIEN, G. A., MILLER, W. C. 2003a. QCAdesigner: A CAD tool for an emerging nano-technology. In *Micronet Annual Workshop* (Toronto, ON), 292–294.

WALUS, K., DYSART, T., JULIEN, G., AND BUDIMAN, R. 2004. QCAdesigner: A rapid design and simulation tool for quantum-dot cellular automata. *IEEE Trans. Nanotechnol. 3*, 1, 26–29.

K. WALUS. 2006. QCA Designer. http:www.qcadesigner.ca.

WALUS, K., VETTETH, A., JULLIEN, G. A., AND DIMITROV, V. S. 2003b. RAM design using quantum-dot cellular automata. In *3rd IEEE Conference on NanoTechnology* (San Francisco, CA), 160–163.

Received April 2006; revised November 2006; accepted November 2006