A Serial Memory by Quantum-Dot Cellular Automata (QCA)

Vamsi Vankamamidi, Marco Ottavi, *Member*, *IEEE*, and Fabrizio Lombardi, *Senior Member*, *IEEE*

Abstract—Quantum-dot Cellular Automata (QCA) has been widely advocated as a new device architecture for nanotechnology. QCA systems require extremely low power, together with the potential for high density and regularity. These features make QCA an attractive technology for manufacturing memories in which the paradigm of memory-in-motion can be fully exploited. This paper proposes a novel serial memory architecture for QCA implementation. This architecture is based on utilizing new building blocks (referred to as tiles) in the storage and input/output circuitry of the memory. The QCA paradigm of memory-in-motion is accomplished using a novel arrangement in the storage loop and timing/clocking; a three-zone memory tile is proposed by which information is moved across a concatenation of tiles by utilizing a two-level clocking mechanism. Clocking zones are shared between memory cells and the length of the QCA line of a clocking zone is independent of the word size. QCA circuits for address decoding and input/output for simplification of the Read/Write operations are discussed in detail. An extensive comparison of the proposed architecture and previous QCA serial memories is pursued in terms of latency, timing, clocking requirements, and hardware complexity.

Index Terms—QCA, memory architecture, emerging technologies.

1 Introduction

In the past few decades, the exponential scaling in feature size and increase in processing power have been successfully achieved by CMOS as the most popular technology for VLSI; however, there is substantial evidence [13] that emerging technologies (mostly based at nanoscale ranges) will be required to supersede the fundamental physical limits of CMOS devices. Among these new technologies, *Quantum-dot Cellular Automata* (QCA) gives a solution at nanoscale and also offers a new method of computation and information transformation using new paradigms. For example, interconnections for signal transfer are used for logic computation and manipulation by which the so-called processing-in-wire paradigm is accomplished.

Orlov et al. [12] reported an experimental demonstration of a metallic QCA cell; such a device is composed of four metal dots, connected with tunnel junctions and capacitors. Microsized QCA devices have been fabricated with metal cells that operate at 50 mK [12] (that is, cryogenic). Experiments have confirmed that the switching of a single electron in a double-dot cell can control the position of a single electron in another double-dot cell. The basic logic behavior with these cells has been demonstrated in [10], using its basic block as a majority voter (MV). It has been reported [7] that room temperature operation requires QCA cells to be fabricated in the range of 1-5 nm in size. Even though the manufacturing of nanometer-scale QCA cells is still being investigated, Lieberman et al. [7] have proposed some possible realizations of molecular QCA; it describes the progress toward making QCA molecules and establishing the attachment chemistry for a substrate compatible with QCA. There is strong evidence that QCA readily adapts to assembly and molecular growth for self-assembly at an extremely small cell size.

Many works have been reported on the system-level features of QCA. Different devices and circuits have been proposed for QCA implementation. These include a carry look-ahead adder, a barrel shifter, microprocessors, and FPGAs [6], [14], [15], [11]. A fundamental issue that must be addressed when designing QCA circuits is timing and clocking. Signal propagation in QCA systems can be accomplished along serial timing zones using the one-dimensional technique in [1]. This one-dimensional arrangement results from the four phases required for correctly operating the QCA cells by pipelining. Long vertical lines consisting of many QCA cells are commonly required to route signals, thus imposing stringent timing constraints on the pipelining process. Moreover, correct switching among cells (that is, kink-free operation) in a timing zone is affected by thermal fluctuations. A trapezoid clocking scheme has been proposed in [2] to provide feedback paths while generating additional processing-in-wire capabilities in QCA designs.

QCA have many desirable features for processing [1]; for example, clocking and timing can be adjusted as a function of the cells in a Cartesian layout. Low power consumption (power gain has been demonstrated by the clocking of the cells), high density, and regularity are readily applicable to QCA; therefore, memory is well suited for implementation using this technology. However, large memory designs in QCA present unique characteristics due to their architectural structure (such as the tournament bracket in cell

- *V. Vankamamidi and F. Lombardi are with the Electrical and Computer Engineering Department, Northeastern University, 360 Huntington Avenue, Boston, MA 02115.
E-mail: {vvankama, lombardi}@ece.neu.edu.*
- *M. Ottavi is with Advanced Micro Devices Inc., Boston Design Center, 90 Central Street, Boxborough, MA 01719. E-mail: mottavi@ece.neu.edu.*

*Manuscript received 7 July 2005; revised 15 Jan. 2007; accepted 15 June 2007; published online 26 Sept. 2007.
Recommended for acceptance by T. Brun.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-7-25-2005.
Digital Object Identifier no. 10.1109/TC.2007.70831.*

placement). Moreover, sequential circuits and memory elements cannot be directly mapped into QCA using the same criteria as conventional CMOS technology. For storage, QCA utilizes the so-called paradigm of memory-in-motion, that is, the state of a memory must be kept in movement in the QCA cells.

The objective of this paper is to present a new serial memory architecture for QCA implementation. The design of this memory is based on the utilization of basic building blocks referred to as tiles. Tiles are used in the memory cell to construct a loop for moving the memory state in different QCA circuits (memory-in-motion), as well as input/output capabilities for the Read/Write operations. The combination of tile-based design and memory-in-motion by state looping results in a novel timing/clocking arrangement by which semi-adiabatic switching can be implemented using two additional signals within a two-stage operational cycle. The serial memory proposed in this paper uses different tiles to allow bidirectional signal propagation. The closed QCA loop, which is required to store data, is formed by using a pair of parallel wires connected together at both ends. The resulting rectangular-shaped loop is partitioned into multiple columns of tiles. Each tile alternates between one of the two stages in the operational cycle, Hold and Switch; adjacent tiles are always in different stages, so, at any given time, half of the tiles are in the Hold stage and the other half are in the Switch stage. When a tile is in the Hold stage, it holds 2 bits of data, one for each horizontal wire, and, when it is in the Switch stage, it holds no data.

This paper is organized as follows: Section 2 presents a brief review of QCA. In Section 3, the different QCA memory designs [5], [3], [4] that have appeared in the technical literature are discussed. In Section 4, the basic principles of tiling for the proposed memory architecture are outlined. Section 5 describes the clocking mechanism and timing requirements for the proposed serial memory. Section 6 presents in detail the operations of the different tiles. Section 7 presents the evaluation of the proposed memory cell by simulation using QCADesigner. A comparison between the proposed memory architecture and other serial designs found in the technical literature is pursued in Section 8. Sections 9 and 10 analyze memory latency and address decoding complexity as further figures of merit. Memory density is addressed in Section 11.

2 Review of QCA

QCA is a new device architecture that is amenable to nanometer scale [1]. QCA stores logic states not as voltage levels but, rather, based on the position of individual electrons [14]. A quantum cell can be viewed as a set of four charge *containers or dots*, positioned at the corners of a square cell. Computation is realized by the Coulombic interaction of extra electrons in quantum dots. Each quantum dot is a nanometer-scaled square with wells at each corner of the cell. The two extra electrons that are present in each cell can quantum mechanically tunnel between wells, but they cannot tunnel out of the cell. *Electron repulsion* causes the extra electrons to occupy diagonally opposite wells. These two electron configurations are used to *encode* binary information in the cells. Fig. 1 shows a QCA cell and the Boolean nature of the *polarization* for its two electron configurations.

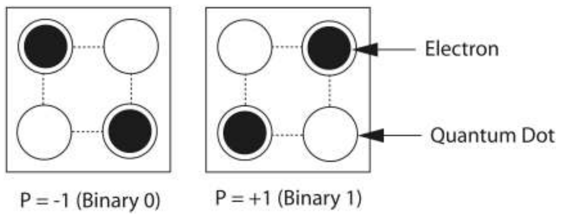


Fig. 1. Binary behavior of a QCA cell.

The unique feature of QCA-based designs is that logic states are not stored in voltage levels, as in conventional electronics, but they are represented by the *position* of individual electrons. Unlike conventional logic, in which information is transferred by electron phenomena [14], QCA operates by the Coulomb interaction that relates the state of one cell to the state of its neighbors. This results in a technology in which *information transfer* (interconnection) is the same as *information transformation* (logic manipulation).

QCA cells can be designed to realize a binary wire, an inverter, and an MV. The basic logic gate in QCA is the MV. The MV with logic function $MV(A,B,C) = AB + AC + BC$ can be realized by only five QCA cells (compared to a CMOS implementation, which requires 16 transistors). The MV is said to have three legs for the input signals. Logic AND and OR functions can be implemented from an MV by setting one input leg (the programming input) permanently to 0 and 1, respectively. Let a control cell be that cell controlling the behavior of an MV as an AND or OR logic gate. Cells that are positioned adjacent to each other tend to align and produce a binary wire, whereas cells positioned diagonally from each other align in an opposite fashion and produce logic complementation (that is, an inverter).

In VLSI systems, timing is controlled through a reference signal (that is, the clock); however, timing in QCA is accomplished by clocking in four distinct and periodic phases [1]. A QCA circuit is partitioned into *serial* (one-dimensional) zones and each zone is maintained in a phase.

The use of a quasi-adiabatic switching technique for QCA circuits requires a four-phased clocking signal, which is commonly supplied by conducting wires buried under the QCA circuitry for modulating the electric field [17]. The four phases are Relax, Switch, Hold, and Release. Fig. 2 depicts a cell in its *four clock phases*. During the Relax phase, there is no interdot barrier and a cell remains unpolarized. During the Switch phase, the interdot 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.

The timing zones of a QCA circuit or system are arranged by following the periodic concatenation of these four clock phases. Zones in the Hold phase are followed by zones in the Switch, Release, and Relax phases. This clocking mechanism provides inherent pipelining and allows multibit information transfer for QCA. As a zone in the Hold phase is followed by a zone in the Switch phase (and preceded by a zone in the Release phase), the

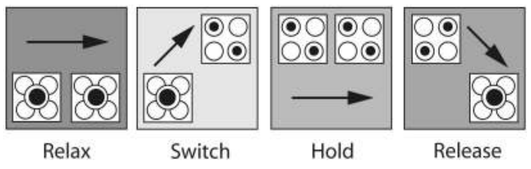


Fig. 2. QCA clock phases.

computation in QCA is strictly one-dimensional (that is, unidirectional and consistent with signal propagation).

Currently, QCA circuits and systems follow the clocking zone partition scheme in [1]. Designs are partitioned into multiple clocking zones only along one dimension (say, the x-axis), thus effectively creating *columns* (as *zones*). Clocking and pipelining require designs to maintain sets of four adjacent zones at any time (as according to the four phases, that is, Switch, Hold, Release, and Relax). For the four phases, clocking to a zone (and the design as a whole) is applied through an underlying clocking circuitry by a signal, as shown in Fig. 3 [9]. Such a signal generates the required electric field to modulate the tunneling barrier of all cells in the zone (adiabatic switching). To maintain zones in sets of four phases, four conducting wires (carrying the signal in Fig. 3) are required. Each clock has a phase that is shifted by $\frac{\pi}{2}$.

3 QCA and Memories

As a nanotechnology, QCA has been advocated for high-density memories. QCA offers many features for designing memories, such as regularity, fast switching, and low power. A straightforward approach to implement a memory by QCA is to maintain a cell (or a clocking zone) in the Hold phase as long as its value must be retained for storage. The main problem with this approach is the requirement of an explicit control of the clock signal from the memory decoder (which is implemented in QCA). This is not viable because it causes problems in timing and a significant increase in the complexity of the underlying circuitry (as interface to CMOS). For a truly QCA-based implementation, it is well known that memory must be kept in motion, that is, the memory state has to be continuously moved through a set of QCA cells (such as those connected in a loop).

Traditionally, it is possible to distinguish two types of memory architectures: parallel and serial architectures. Table 1 summarizes the comparison of parallel and serial QCA memory architectures. For a parallel architecture, the one-bit-per-memory cell feature reduces latency, but the duplication of the Read/Write circuitry for each memory bit, the higher counts (QCA cell, Control cell, and Clocking zone), and the more complex clocking circuitry result in a faster operation at a reduced density.

TABLE 1
Comparison of Parallel and Serial QCA Memory Architectures

Feature	Parallel Architecture	Serial Architecture
Latency	Low	High
Read/Write Circuitry	Duplicated for every bit	Shared between multiple bits
QCA-cell and Control-cell Count	High	Low
Zone Count and Clocking Circuitry	Complex	Simple
Memory Density	Low	High

The most obvious advantage of a serial over a parallel architecture is with respect to hardware; in a parallel architecture, each bit stored in the memory requires a separate Read/Write circuitry. In QCA, the Read/Write circuitry is more complex (requiring multiple logic gates) than a QCA wire loop (for actually storing the data bit). Therefore, compared to a serial design in which a single Read/Write circuit is shared between multiple bits of data (as stored in each QCA loop), a parallel design requires a higher number of QCA and control cells. Moreover, the Read/Write circuitry must be partitioned into multiple clocking zones, which causes the number of clocking zones and the complexity of the underlying clocking circuitry to increase. Parallel designs also encounter problems with synchronization and coordination of the clock phases of different Read/Write circuits with the Control/Select signals that are transferred to a common QCA line.

Many QCA memory designs have been proposed in the technical literature. Researchers at Notre Dame University have introduced the H-Memory architecture [4], whose main objectives are high density and uniform access time. The H-Memory is a complete binary tree structure with control circuitry at each node. As the memory spirals at the leaf nodes, an integration of logic and memory is accomplished in the layout, even though, logically (as encountered in conventional designs), control circuitry and storage are separate. Fig. 4a shows this spiral architecture. However, unlike conventional designs, the control and data bits are serialized. The bitstream enters the memory structure at the root node and traverses down the tree by utilizing one control bit for routing at every node in the path. The architectural choice of dealing with serial bitstreams also results in a complex control logic. The router at each internal node has 10 gates and six feedback loops; each loop requires four clocking zones for its implementation. The circuitry at the leaf nodes (that is, the memory cells) requires 11 gates per node. Also, the memory cell at each leaf node is a spiral, allowing the storage of several bits

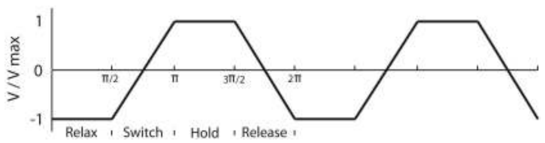
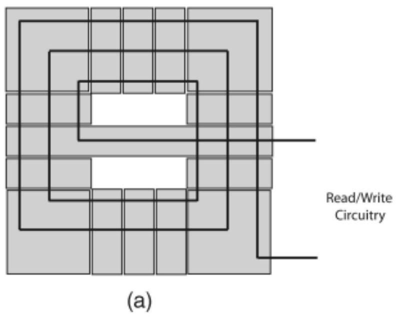


Fig. 3. Four-phased signal for clocking zones in QCA, adiabatic switching.



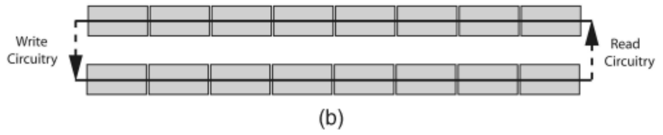


Fig. 4. Serial memory cell architectures. (a) Memory spiral architecture presented by Frost et al. (b) Memory loop architecture used by Berzon and Fountain.

while sharing clocking zones between multiple concentric loops. In this design, the memory size and the cell count at each spiral do not have a linear relationship; each outer loop has an increasing diameter, thus requiring more cells to implement (although its storage capacity remains constant).

Berzon and Fountain [3] made an early attempt to design a QCA memory using the so-called SQUARES formalism. The basic principle of this technique is to define a set of equally sized blocks, each performing a basic function in QCA (such as logic or interconnect). The basic blocks can then be tiled together to design more complex QCA circuits. The obvious advantage of this technique is the ease in the geometric layout; also, such formalism allows a design to be modular such that the tiles (blocks) are effectively considered as black boxes. However, as all blocks are of the same size (in SQUARES, a 5×5 grid), an unutilized area appears in each block, thus causing spatial redundancy and lower density in the overall design. This results in a serial memory architecture (shown in Fig. 4b) in which each memory cell consists of a QCA wire connected as a loop. The wire is divided into clocking zones whose number is four times the number of bits stored in the loop. Even for a modest memory size, this results in a large number of zones, thus requiring a considerable amount of clocking circuitry for the signals. Finally, additional control circuitry (such as comparators) must be utilized to make the memory bit addressable. This results in a quite high hardware overhead per memory cell.

Walus et al. [5] proposed a conventional parallel memory architecture (such as that encountered in CMOS-based RAM design) for QCA, that is, by storing 1 bit at each memory cell. The single-bit memory cells allow the design of a simple Read/Write circuitry; each memory cell is implemented using 158 QCA cells and the Select signals are separately generated using decoders. This approach has the same disadvantage as in [3]; as data in each memory cell is stored using a closed QCA wire loop partitioned into four clocking zones, it needs a large number of clocking zones, thus complicating the underlying circuitry for clocking (the dimensions of the clocking zones for the memory loop are small, that is, a QCA wire with fewer than 10 cells).

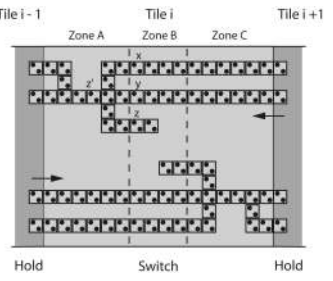


Fig. 5. QCA implementation of the Internal memory tile.

However, the dimensions of the conducting wire distributing the clock is at least an order of magnitude higher, thus making the clocking of such small zones very difficult if not infeasible. Also, this memory architecture does not exploit full parallelism in its operation: Unlike CMOS designs, the Select signal in QCA takes multiple clock cycles to propagate and does not simultaneously reach all cells in a row, that is, bits in the same row are read at different clock cycles.

In the single-bit memory design introduced in [18], each QCA line for a memory cell spans three clocking zones, as required to retain its memory value. By increasing the length of the QCA wire over a number of zones (in a number given by a multiple of three), more than 1 bit of data can be stored in the wire. The principle of memory-in-motion is kept within a single-bit design by transferring the stored value back and forth within the clocking zones. During the Read cycle, multiple bits in a line must be transferred to one end of the wire for output; however, there is no feedback loop to transfer the bits back to the beginning of the wire; consequently, they can be lost. Therefore, a serial architecture that stores multiple bits at each memory cell must have a feedback path (such as a continuous loop).

4 Memory Design by Tiling

In this section, the basic principles of a novel architecture for a serial QCA memory are presented. The proposed architecture still utilizes the concept of memory-in-motion within a QCA loop. Some of the advantages of the proposed serial architecture are the novel QCA design for storing the memory bits and the associated Read/Write control circuitry. The proposed design is independent of the address decoding logic and can be used with the decoding circuits proposed for other QCA memory architectures [3], [4], [5]. QCA cells are arranged into simple basic QCA blocks referred to as *tiles*. Three types of tiles are utilized: 1) Internal memory tile (shown in Fig. 5), 2) Output tile (shown in Fig. 6), and 3) Input tile (shown in Fig. 7).

Tiles are connected in a loop using two horizontal wires (referred to as the upper and lower wires) (Fig. 8). The memory cell in the proposed serial architecture consists of two long horizontal QCA wires connected at both of their ends by two short vertical wires, which create a loop for the memory-in-motion implementation. The Input and Output tiles and related circuits (for the Read and Write operations) are located at opposite sides of the horizontal wires. The Internal memory tiles are located between the Input and Output tiles. In this architecture, the loops are stacked, thus

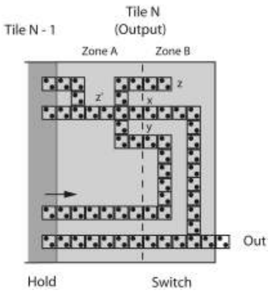


Fig. 6. QCA implementation of the Output tile.

resulting in a highly compact memory layout. Together with a novel clocking strategy, data is allowed to move in each horizontal wire along two different directions while still being connected into a continuous loop. Fig. 8 shows the architecture for one memory loop using tiles; note that the size of the register is equal to the number of tiles and clocking zones are shared between all registers in the memory.

A memory loop partitioned into n tiles can store n bits of data, that is, the number of tiles required to implement a serial memory cell is equal to its word size. In the proposed architecture, all memory cells are arranged into a column. Tiles partition the loop of a particular memory cell and the memory loops of all other memory cells in that column. The exception is the Input tile, which is used to multiplex new input values into the memory loop (and they cannot be shared with other memory cells). Therefore, the number of tiles that are required to implement a memory of m words (for a word that is $n\text{-bits}$ wide) is given by

$$T_{m\times n} = m + (n-1), \tag{1}$$

where m corresponds to the number of Input tiles required for each of the m words and $n-1$ corresponds to the tiles that are shared between all memory cells for implementing the remaining $n-1$ bits.

For establishing the number of required QCA cells, the number of bits stored in a memory loop must be equal to the number of tiles. Therefore, the number of QCA cells required per bit is equal to the number of QCA cells of the memory loop in any tile. A total of 74 cells are required for the Internal memory tile (Fig. 5), whereas 24 and 54 cells are required for the Input and Output tiles (Figs. 6 and 7), respectively. A total of 74 cells are required to store 1 bit of data, that is, the QCA cell count is linearly related to the memory size.

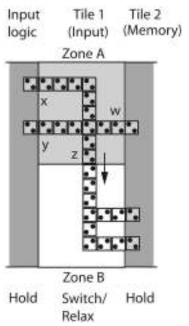


Fig. 7. QCA implementation of the Input tile.

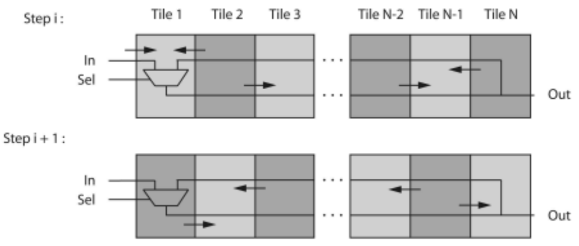


Fig. 8. Proposed $N\text{-bit}$ wide memory.

In the proposed serial architecture, timing and clocking are implemented using a two-level arrangement; the first level is tile based. Each tile is divided into zones which are utilized for timing purposes for the different QCA phases. The Internal memory tile has three zones, the Output tile has two zones, and the Input tile has two zones. The operational cycle consists of two stages (made of multiple steps) that are tile dependent, affecting the different zones. The second level is loop based; each loop is partitioned into multiple columns of clocking zones; each column of clocking zones spans the same section of all loops arranged into a stack. The number of clocking zones into which the horizontal wires of the loop are partitioned determines the word size of each memory cell; the number of loops that are stacked determines the memory size.

To store data in the loops, bits move in opposite directions along the two horizontal wires. However, in the proposed architecture, similar sections of the horizontal wires are in the same clocking zone. Using a conventional four-phased clocking mechanism, data always moves in the same direction; to resolve this issue and retain the advantages of a serial architecture, tiles with different operational features must be utilized. The proposed memory is depicted in block diagram form in Fig. 8; each tile is alternatively in two different stages (referred to as the Hold and Switch stages) of the operational cycle, that is, adjacent tiles are always in different stages. When a tile is in the Hold stage, it retains the bit values that are stored in the two horizontal wires of the loop and holds them as input for the next tile. When a tile is in the Switch stage, it switches to the new input bit value, thus moving data among adjacent tiles. The QCA cells in the tiles of a wire and the associated clocking strategy allow bits to move only in one direction at one time, that is, counterclockwise (right to left for the upper wire) for the purpose of this paper. The Hold and Switch stages involve different clocking zones for the tiles and phases of the four-phased clock signal (which includes Release and Relax).

Input and Output tiles require two clocking zones per tile; all Internal memory tiles require three clocking zones per tile. Therefore, the total number of clocking zones for implementing a memory of size $m \times n$ using the proposed architecture is given by

$$Z_{m\times n} = (2 \times m) + (3 \times (n-2)) + 2, \tag{2}$$

where $n-2$ is the number of intermediate memory tiles. Therefore, the number of clocking zones required to implement the proposed architecture is efficient. In the proposed design, the maximum line length does not increase with word or memory size. Moreover, the

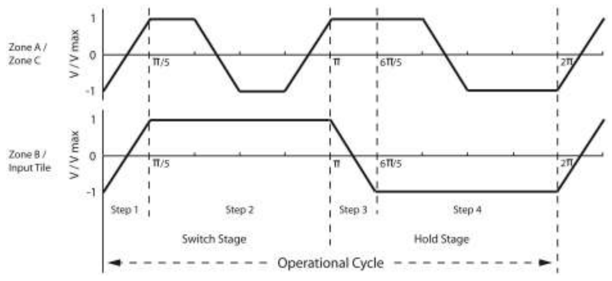


Fig. 9. Clock signals for the required switching mechanism.

proposed architecture retains the advantages of single-bit memory design, whereas clocking zones between all memory cells are shared, thus reducing the complexity of the control circuitry. A serial design has a single Read/ Write logic for multiple bits in each memory cell, so, when the number of bits per cell increases, the hardware overhead per bit is also reduced.

5 Clocking and Timing

The proposed serial memory requires clocking signals that are different from the ones used in previous QCA memory designs. Signals for this architecture also utilize the same four phases for semi-adiabatic switching; however, for proper clocking, the times of the signals for the Relax and Hold phases must be substantially different. All three zones in each Internal memory tile (that is, A, B, and C in Fig. 5), as well as the other tiles, do not switch in the same fashion. Therefore, multiple signals are required to clock the memory.

Consider initially the Internal memory tile. Zone 1 (A) and Zone 3 (C) of each memory tile switch identically and are always in the same phase. Therefore, a single signal can be used to clock both of them. However, Zone 2 (B) switches differently, thus requiring a second clock signal. Although the Output tile has only two zones (Fig. 6), its switching mechanism is similar to the Internal memory tiles. Therefore, its Zone 1 and Zone 2 can be clocked with the same signals used for the memory tiles. The Input tile (Fig. 7) is partitioned horizontally and is switched differently from the memory tiles; the Input tile must multiplex the new memory state (as input) depending on the Select signal. Therefore, the clock signal that is required to achieve this switching strategy is the same as for Zone 2 (B) of the memory tile. The Input tile requires no separate clock signal (for an Input tile, the clock signal for its Zone 2 is just a phase-shifted version of the signal for its Zone 1). Thus, two additional signals that are different from the conventional clocking arrangement of QCA are required to clock the proposed serial memory. The three signals that are required for clocking the proposed memory architecture are periodic in nature. The first half of the clock cycle corresponds to the *Switch stage*, whereas the second half corresponds to the *Hold stage*. Fig. 9 shows the waveforms of the two required clock signals over one clock period (*operational cycle*).

All tiles in the memory architecture (including the Input and Output tiles) are in one of the two stages of the operational cycle, that is, Switch and Hold. When a tile is in the Switch stage, its adjacent tiles are in the Hold stage. As all tiles alternate in stages, during the $k\text{th}$ $(k+1)$ operational cycle all Internal memory tiles with an even index are in the Switch (Hold) stage, whereas Internal memory tiles with an odd index are in the Hold (Switch) stage.

The operational cycle consists of two so-called stages. For an Internal memory tile, the two stages consists of four steps (two per stage) as follows:

- *Switch stage.* In the Switch stage (Step 1, 0 to $\frac{\pi}{5}$), all zones of the tile are in the Switch phase. At the same time, the neighboring zones are in the Hold stage (π to $\frac{6\pi}{5}$) and act as inputs. Hence, the input values are multiplexed and the output is moved to Zone 2 of the Switch tile. During Step 2 ($\frac{\pi}{5}$ to π), Zone 2 is retained in the Hold phase and Zone 1 and Zone 3 are cycled through the remaining phases and returned to the Switch phase. At the same time, the neighboring zones that are in Step 4 of the Hold stage ($\frac{6\pi}{5}$ to 2π) are released and retained in the Relax phase. Therefore, the new multiplexed values of Zone 2 are propagated to one of the desired zones (either Zone 1 or 3).
- *Hold Stage.* During Step 3, Zone 1 and Zone 3 are retained in the Hold phase such that they act as the input for adjacent tiles (which are now in the Switch stage). The previous memory values (corresponding to the Switch stage in Zone 2) are released. During Step 4, all zones are returned to the Relax phase such that they can switch together again at the beginning of the next operational cycle.

For an Input tile, its stages (made up of four steps) are as follows: During Step 1 in the Switch stage, Zone 1 is in the Switch phase (the neighboring zone of the memory tile on one side and the new input signals on the other side are in the Hold phase). Since the clock signal for Zone 2 is $\frac{3\pi}{5}$ phase delayed of Zone 1, Zone 1 is in the Relax phase; the MV in Zone 1 multiplexes the input value and the old memory state, thus resulting in the new memory state. During Step 2, Zone 1 is retained in the Hold phase until Zone 2 reaches the Switch phase, so the new memory state is propagated from Zone 1 to Zone 2. At this time, the neighboring zone of the memory tile is in the Relax phase. During Step 3 (corresponding to the Hold stage), Zone 2 is in the Hold phase and the new memory state is propagated to the adjacent memory tile (which is in the Switch stage). In Step 4, both zones in the Input tile are returned to the Relax phase.

The Output tile has the same operational cycle as an Internal memory tile, so the description of its operational cycle is omitted.

6 QCA Tiles

Consider initially the memory tile of the proposed QCA architecture. Each memory tile consists of a column of three clocking zones spanning the internal section of the two horizontal wires; this is applicable to all memory cells that have been arranged into a stack. When a tile is in the Switch stage, its two adjacent tiles are in the Hold stage; therefore, each horizontal wire in the Switch stage tile has two inputs that are in the Hold phase at each of its two ends. To have a counterclockwise memory motion, the upper horizontal wire of each memory loop must be multiplexed with the two inputs, so it transfers the input from the previous tile (on the right) to the next tile (on the left). As the lower horizontal wire is multiplexed, it transfers the input from the tile on the left to the tile on the right. This switching mechanism can be achieved through an MV that functionally acts as a diode (that is, blocking the movement of data) using the clocking zones available in each tile. This MV is placed in the clocking zone near the input whose value must be dominated (masked) to obtain the unidirectional memory motion. The input whose value must be transferred is duplicated and connected to two inputs of the MV, whereas the input whose value must be blocked is connected to the third input only. Therefore, the two input values are multiplexed by the MV and the output is the desired value. Using the MV, this new output is forced back to the wire whose input value was blocked. At the beginning of the Switch stage, the horizontal wires of a memory loop have the two inputs at both ends; however, at the end of the Switch stage, the wire has only the value of the required input.

Fig. 5 shows the QCA circuitry in a memory tile (referred to as the Internal tile or tile i) for the two horizontal wires of a memory loop. This tile operates over an operational cycle made up of two steps:

- *Step 1.* All three clocking zones (A, B, and C) of memory tile i in the Switch stage are in the Switch phase, whereas clocking zones of adjacent tiles ($i-1$ and $i+1$) are in the Hold phase. Therefore, the desired input that is duplicated and connected to two inputs of the MV (that is, x and y) is moved to the output (z').
- *Step 2.* The middle clocking zone of tile i (that is, zone B) of the Switch stage is kept in the Hold phase, whereas the other two zones (A and C) are cycled through their phases and returned to the Switch phase. At the same time, the adjacent two tiles ($i-1$ and $i+1$) are relaxed from their Hold phase. Therefore, the direction of signal propagation changes; a wire that was previously an output for the MV now becomes an input and vice versa. However, two wires still remain as inputs to the MV; as the three inputs of the MV have the same signal, the output follows this value. The output is fan out, that is, this signal is duplicated and propagated to the next tile during the subsequent operational cycle.

By the end of Step 2, all cells of the upper horizontal wire align to the input from the tile on the right while blocking the input of the left tile; the lower horizontal wire aligns to the input from the left tile while blocking the input from the right tile. Thus, in one operational cycle, data in the memory loop moves by one tile in a counterclockwise direction (right to left). In the next operational cycle, the tile that was in the Switch stage with new data on the horizontal wires is changed to the Hold stage and the two adjacent tiles (which were in the Hold stage) are changed to the Switch stage, thus enabling further motion of data.

Two additional tiles are required for the input/output of the memory loop, as well as connecting the upper and lower horizontal wires.

- *The Output tile.* The Output tile propagates data in the lower horizontal wire to the output read logic. It also has a vertical QCA wire to transfer data to the upper horizontal wire such that the loop is established. However, when the Output tile switches to accept a signal from the lower horizontal wire (which is in the Hold stage), the upper horizontal wire is also in the Hold stage. Therefore, duplication of the lower horizontal wire (to dominate the upper wire) and an appropriate switching strategy for the memory tiles are required. As the Output tile performs only one transfer (that is, from the lower to the upper horizontal wires), it only requires two clocking zones and one MV.

  Fig. 6 shows the QCA implementation of the Output tile; the operational cycle of the Output tile is the same as that of the internal memory tile. In Step 1, the MV is switched and, due to the duplication, the signal of the lower horizontal line dominates and is transferred to the output. In Step 2, the previous output line acts as an input and transfers the signal to the upper horizontal wire. Having aligned the signal on the lower horizontal wire, in the next operational cycle, the Output tile alternates to the Hold stage and loops the signal to the upper horizontal wire.
- *The Input tile.* The Input tile has a vertical wire to connect the two horizontal wires; this transfers the signal between them such that the memory loop is constructed at the other end too. However, it is different from the Output tile because, prior to transferring data, the old memory state has to be multiplexed with the input data based on the Write control signal for acquiring the new memory state. Multiplexing is achieved through an MV, as shown previously. However, the Input tile can be affected if no horizontal partitioning is implemented for timing. The implementation of the Input tile is shown in Fig. 7. Since the tile is horizontally partitioned into two clocking zones (upper and lower), they cannot be shared with other memory loops. The functionality of the Input tile over one operational cycle is also given by a two-step process:

  1. *Step 1.* The lower clocking zone of the Input tile is in the Relax phase; the upper clocking zone is in the Switch phase. Its two inputs (from the memory tile on one side and the Write circuitry on the other side) are in the Hold phase. Hence, the output of the MV in the top zone switches to the new memory state as input value.
  2. *Step 2.* The upper clocking zone is kept in the Hold phase and the lower zone is in the Switch phase (while keeping the adjacent memory tile in the Relax phase). Therefore, the new memory value is propagated to the QCA wire connected to the lower horizontal wire. In the next operational cycle, the Input tile is in the Hold stage and the new memory value is moved back to the memory tile, which is changed to the Switch stage.

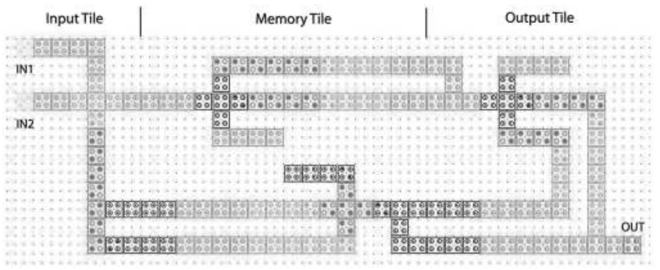


Fig. 10. Design of the proposed memory cell for a simulation by QCADesigner (logic verification).

As an Input tile must implement the logic to multiplex between the old memory value and the new input value, it requires two separate clocking zones.

7 Simulation

QCADesigner [16] provides a design and simulation environment for QCA circuits; it has multiple simulation engines and CAD capabilities. This tool has been used to verify the design of the proposed QCA memory cell. A QCA memory loop that consists of an Input tile, one Internal memory tile (1 bit), and an Output tile (Fig. 10) was assembled. As the Input and Output tiles store 1 bit each and the Internal memory tile stores 2 bits, the size of the memory cell in Fig. 10 is 4 bits. Simulation has been performed using the bistable engine of QCADesigner with a cell dimension of 18 nm and a dot size of 5 nm. However, QCADesigner does not support the clocking scheme and clock zone partitioning of the proposed memory cell; therefore, minor adjustments (mostly of a functional nature) were implemented to establish compatibility with this CAD tool. The proposed QCA memory cell has been evaluated and simulated for logic and clocking (timing) verification. In both cases, minor modifications were required; these modifications are introduced only for compatibility with QCADesigner, that is, the modified memory circuit/clocking is isomorphic to the proposed circuit/clocking scheme, as presented in the previous sections.

For *logic verification*, Fig. 11 shows the simulation results for the memory cell; the phase of the output waveform is shifted by two clock cycles with wraparound (that is, the output waveform for clock cycles 7 and 8 is shown in clock cycles 1 and 2 too). During the first four clock cycles, the output is determined only by the two inputs that are connected to the two legs of the MV in the Input tile; the third leg of this MV is not connected because it takes four cycles for the first bit to loop through the memory cell. Therefore, during this period the MV behaves as an AND gate. For the next four clock cycles (labeled five through eight in the waveform diagrams), all three legs of the MV are connected (active) and the MV inputs are 000, 010, 100, and 111. Hence, this will result in a majority function with an output of the same value as during the first four clock cycles. As observed in the simulation results, the QCA circuit behaves as expected, that is, data is looped in a correct manner through the tiles.

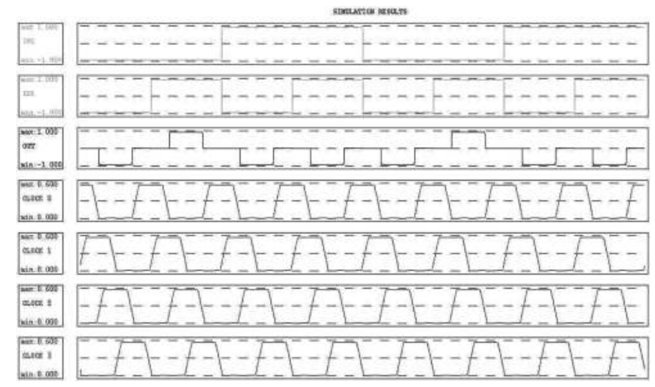


Fig. 11. Simulation waveforms for the proposed memory cell (logic and timing verification).

For the *timing verification* of the QCA memory cell, a slight modification to the clocking strategy must be performed because QCADesigner only provides the capability of clocking circuits using a conventional four-phased clock signal, as shown previously in Fig. 3. This limiting feature is also found in other CAD tools for QCA, such as AQUINAS [17]. In the proposed clocking strategy, the clocking zones next to a zone in the Switch phase must be in the Hold phase so that all three legs of the MV can be driven. To simulate this feature, the arrangement shown in Fig. 12 was utilized. The third leg of each MV in the Internal memory and Output tiles is permanently set to a value and placed in the same clock phase as the other two legs of the MV. The value of this third leg is dominated by the duplicated value on the other two legs and the resulting output of the MV is propagated through the memory loop. Thus, the counterclockwise motion of data as required by the proposed clocking strategy is still achieved within the memory loop. This modified memory cell has been simulated using QCADesigner with the above-mentioned configuration; the resulting waveforms are the same as the memory cell design (shown previously in Figs. 10 and 11).

8 Comparison

In this section, an analysis is pursued to compare the proposed serial memory with other serial memories found in the technical literature [3], [4]. The serial QCA memory architecture in [4] uses a spiral (squared shaped) that loops back to itself for storing data. The main advantage of a spiral over a loop is that the sections of each layer of the spiral can be in the same clocking zone. Even though the word size of each memory cell is increased by adding extra layers, the number of clocking zones is not increased. As

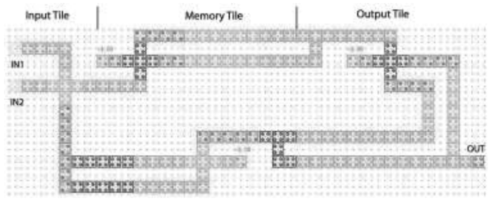


Fig. 12. Design of the proposed memory cell for simulation by QCADesigner (timing verification).

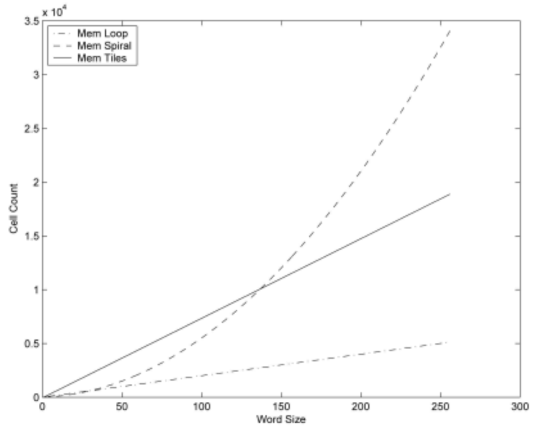


Fig. 13. QCA cell count versus word size.

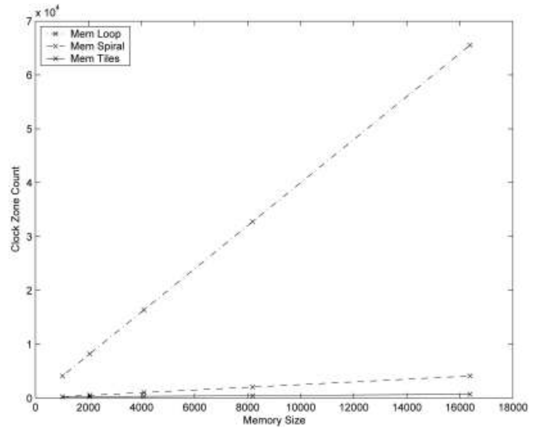


Fig. 14. Clocking zone count versus memory size.

clocking zones span multiple layers, their dimensions are sufficiently large to be clocked by the underlying wires. However, the spiral architecture in [4] has some inherent drawbacks. As the word size at each memory cell is increased by adding layers, the number of QCA cells for implementing them increases, that is, the number of QCA cells required per data bit is not constant and depends on the word size at each memory cell.

The problem of increasing the QCA cell count for additional layers leads to another drawback, that is, the number of clocking zones into which the memory spiral is partitioned is constant. Therefore, as the dimensions of each additional layer increases, their length in some clocking zones (corners) also increases. This could be a significant problem because the probability of kink occurrence increases with the maximum line length of a clocking zone. To avoid kinks, the switching frequency must be reduced too to ensure that the QCA cells remain in the ground state.

The first significant difference between the memory spiral [4] and the tile-based memory proposed in this paper is that the memory spiral shares clocking zones within a memory cell and, hence, it is independent of word size. In the memory presented in this paper, clocking zones are shared between different memory cells, so this scheme is independent of memory size (that is, the number of words), but it depends on the word size. As the number of memory cells is usually much larger than the number of bits in each cell, then the proposed architecture provides a better arrangement for the number of clocking zones required for timing a QCA memory. The SQUARES technique in [3] is also evaluated and compared. The modular design of this technique is different from the tiling proposed in this paper; the basic block in [3] is designed to improve different QCA functionalities. In SQUARES, the number of clocking zones is four times the number of bits stored in memory and the number of QCA cells (per bit) to implement the loop is 20. The density is low because of the complex decoding and control circuitry, as well as the low area utilization due to the SQUARES formalism. The tiles of the proposed approach are tailored to memory design and its performance.

Fig. 13 shows the cell count versus word size for the QCA memories proposed in [4] (Mem Spiral) and [3] (Mem Loop), as well as the design in this paper (Mem Tiles). The linearity of both the proposed approach and [3] is evident even though the Memory Loop [3] requires a small number of cells. A comparison is also performed with respect to clocking zone count. Fig. 14 shows the relationship between the clocking zone count and the memory size for the same three QCA architectures. In this case, [3] requires the largest number of zones, thus reducing the switching speed of the QCA memory, whereas the proposed scheme needs the least.

For QCA designs, the logic-level effects caused by the interference of the electric field from adjacent clocking wires are a significant issue because QCA cells that are located at the boundary must be in a clocking zone. This depends on the strength of the electric field in the adjacent clocking wires [9]. In [9], wave clocking was first introduced such that circuits can be designed with QCA cells at clocking zone boundaries with no modification in logic functionality; these cells must belong to either of the clocking zones to tolerate the interference in the electric fields [19].

9 Latency Considerations

By the memory-in-motion paradigm, storage is implemented in QCA by continuously moving bits in a loop. In serial architectures, multiple bits are stored in each loop. Each memory loop is associated with a single Read/Write logic circuitry. When the first bit of a memory word reaches this circuitry, the memory operation can be performed, that is, bits in the loop can be read and transferred to an output line or new input bits can be written into the loop. However, if the first bit passes the Read/Write circuitry, then a delay is incurred to account for cycling through the loop and returning it back to the Read/Write circuitry. On average, this delay (generally referred to as *memory latency*) is equal to half the time required to complete one revolution through the loop. However, the time required to pass through the loop depends on the loop size, which is a function of the number of stored bits (that is, the word size). Therefore, the word size of a memory cell must be small to reduce latency.

The serial architecture presented in this paper is only word addressable (although, with additional circuitry, the individual bits of a word in each memory loop could also be made addressable). Memory latency is incurred only for the first bit of the word, that is, all subsequent bits are accessed in successive clock cycles with no latency. However, the serial memory designs presented in [3] are bit addressable (that is, individual bits within the memory loop can be addressed). For random bit access, these designs incur a penalty for memory latency on every bit access; therefore, if bit addressing is required, parallel QCA architectures are better suited. If serial architectures are selected for bit-addressable memories, latency considerations provide an added reason to keep the word size at each memory cell small.

For a memory design with a small word size, clocking considerations (which also affect the underlying clocking circuitry) make the serial architecture presented in this paper more advantageous than the serial designs in [3] and [4] because clocking zones are shared between memory cells rather than within a memory cell, that is, a reduction in word size by one bit accomplishes a reduction of three in the number of clocking zones. Therefore, the reduction in word size at each memory cell and the increase in the total number of memory cells (to preserve a constant memory size) reduce the total number of clocking zones required for implementing the QCA memory. However, the QCA memory design in [4] uses a closed QCA spiral which shares clocking zones within but not between memory cells; its reduction in word size does not reduce the number of clocking zones for the memory cell. Moreover, the reduction in word size and the increase in memory cell count result in an increase in the total number of clocking zones for the serial memory (in [3], the clocking zone requirement is rather high because it is a function of the total memory size).

In addition to the word (loop), another characteristic that affects latency is the time for moving bits in the QCA loops. In the serial designs in [3] and [4], one bit of the memory loop passes through the Read/Write circuitry at every clock cycle. For the proposed serial architecture, one bit passes every half cycle because the Hold stage of one tile coincides with the Switch stage of the adjacent tile (Fig. 8). However, [3] and [4] use a conventional clock signal which has four phases in each cycle (the proposed architecture uses a different clock signal which has a repetition of four phases for a total number of 10 phases per clock cycle, as shown in Fig. 9). Therefore, the designs in [3] and [4] require four clock phases (one cycle) for bit movement, whereas the proposed architecture requires slightly more, that is, five clock phases (or half cycle).

A further feature that must be considered is the time period for each phase of the clock signal. This is determined by the longest QCA line of a clocking zone [1] as

$$T_s \propto C^{1.16}, \qquad (3)$$

where T_s is the switching time for the clocking zone and C is the number of QCA cells in the longest wire of the zone. This equation shows that the dependency is nearly linear (the slightly higher exponent is attributed to approximations in the calculation).

Using the number of clock phases per bit movement and the time period of each phase, the average memory latency of the proposed scheme is given by the product of these terms times half the word size (moving the first bit to the Read/Write circuitry). Fig. 15 shows the plot of memory latency with the word size (loop size) for different serial architectures. The nonlinear behavior of the spiral architecture is due

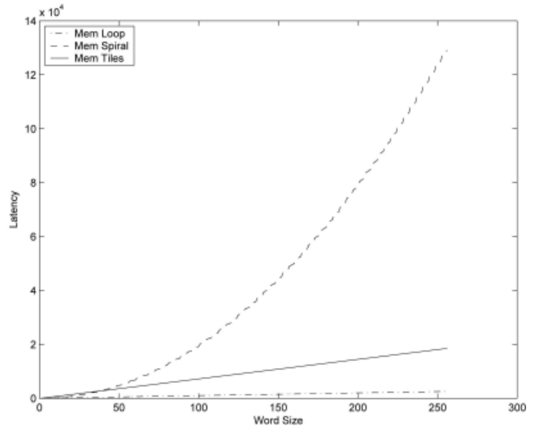


Fig. 15. Comparison of latency for bit access.

to the increase in word size with each additional layer and the length of the QCA line in a clocking zone. As the time period increases, the bit movement rate is also reduced. However, for the loop and tile architectures, the length of the QCA line of a clocking zone is independent of the word size and, therefore, the bit movement rate remains unchanged, that is, it is linear with the word size.

10 Address Decoding

The circuitry that decodes signals on the address lines and selects the corresponding memory cell is generally referred to as the *address decoder*. This is an important functional part of a memory because it ultimately affects its performance, as well as density. In the proposed architecture, the m signal lines address n memory cells, where $n = 2^m$. In traditional CMOS memories, address decoding is usually achieved through standard blocks such as m-to-n demultiplexers, lookup tables (PROM), or programmable logic devices (PAL, PLA, and FPGA). As in the early stage of research, a QCA memory will require simple circuits using combinational logic for address decoding purposes. In this section, a logic block for address decoding is presented. Issues associated with its design are discussed for improvement in reliability and performance. The characteristics and hardware requirements of the proposed architecture are then compared with other QCA decoding circuits presented in the literature.

The operation of a decoder is based on selecting one of the n Output lines by the m Select signal lines, where $n = 2^m$. Decoders usually are the preferred devices for generating mutually exclusive signals as required for addressing. In QCA, decoders can be designed using MVs that implement the AND/OR functions. Each m-to-n decoder requires a total of $(n-1)$ 2-to-1 decoders which are implemented using two MVs (as AND gates) and an inverter. Fig. 16 shows the QCA design of a 3-to-8 decoder (in this case, $Enable = 1$ and $Sel_{1,2,3} = 0$, so only $Out_o = 1$). The input A is the $Enable$ of the *memory cell* and is propagated depending on the values of the Select lines (Sel_i, $i = 1, 2, 3$), that is, at any given time, only one of the eight memory cells that are connected to the outputs (Out_j, $j = 0, \dots, 7$) is enabled. By changing the value of A, each memory cell can be enabled/disabled.

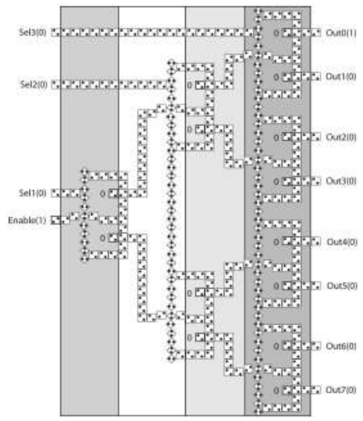


Fig. 16. The 3-to-8 QCA decoder under the one-dimensional clocking scheme.

A further issue that must be considered for address decoding is the synchronization between accessing the memory cell and the operational cycle. The memory cycle for parallel and serial designs consists of multiple conventional (four-phased, equally timed) QCA clocking cycles. Although the operational cycle of a parallel architecture consists of two QCA clocking cycles, for a serial architecture, the operational cycle is made of multiple QCA clocking cycles depending on the number of bits stored in each memory cell.

The Control signals for the cells must be asserted and valid during the first clock cycle of the operational cycle of the memory when the bit (which is stored in the memory cell) reaches the input clocking zone. For a serial architecture, the Control signals must only be asserted during the first clock cycle, when the start bit reaches the Input tile. If the signals are asserted in the middle of the memory cycle, the value on the input line could be written at an arbitrary position of the loop, thus corrupting the data in the memory cell.

Using address decoders, synchronization can be accomplished with relative ease by using a counter at its input. If the input of the decoder is enabled, then the Control signals to the addressed memory cell are effectively asserted; if the input is disabled, the Control signals to all memory cells (including the addressed cell) are not asserted. A counter (with a count equal to the number of QCA clocking cycles in the operational cycle of the memory) can be used to enable the decoder input at the correct time, that is, the signals at the memory cell are asserted only at the beginning of the memory cycle. Thus, only a single counter is required to maintain synchronization for all memory cells.

The proposed circuitry can be compared with previous works. Walus et al. [5], for example, use separate decoding logic for each row of the memory cells in a row-addressed two-dimensional architecture. A memory with N rows that is addressed by M (where $M = log_2 N$) address lines would therefore require N M-to-1 decoders. As each decoder requires $M-1$ two-input gates (for AND and OR), the total number of QCA gates (or MV) required to address the N locations using this decoding scheme is

$$G_{M-to-1} = N \times (M-1) = 2^M \times (M-1). \tag{4}$$

The use of separate decoding to address each location (row) has an advantage in terms of latency. As it involves N M-to-1 decoders (connected in parallel), the latency in address decoding is only equal to that of an M-to-1 decoder (which requires signal propagation through $log_2 M$ levels of two-input gates). So,

$$L_{M-to-1} = log_2 M. \tag{5}$$

The decoding logic presented in this paper uses a single M-to-N decoder that propagates the Select signal to one of the N locations (based on the signals in the M address lines). The hardware requirements for this architecture are considerably lower than that in [5], which uses a separate circuit for each of the N locations. The total number of two-input QCA gates (or MVs) required for the proposed decoding scheme is

$$G_{M-to-N} = 2^{M+1} - 2. \tag{6}$$

However, as a single block is used for decoding the addresses of all N memory locations, the latency is also increased. As latency is related to the number of levels of two-input QCA gates (required for signal propagation to complete the decoding process by using the M-to-N decoder), the latency is given by

$$L_{M-to-N} = M. \tag{7}$$

Therefore, the proposed decoding circuit requires significantly less hardware for implementation, thus accomplishing a higher density (albeit requiring additional clock cycles).

For comparison, consider next the memory architecture in [4]. The H-memory structure in [4] uses a different approach for QCA signal propagation because it exploits microlevel pipelining in QCA wires. In previously presented designs, each bit of the memory address space is transferred through a different QCA line, similarly to CMOS designs. Therefore, the decoding circuits of CMOS can be readily adapted to QCA. In the H-memory, the address and data bits are serialized and transferred through a single QCA wire; hence, the decoding circuitry is substantially different. The H-memory is a complete binary tree, with memory cells at the leaf nodes and decoding logic at the root and all other internal nodes. As the address and data bits enter the structure at the root node and depending on the address value, data bits are routed to a particular memory cell; therefore, one address bit is needed for making a decision at each node.

In another serial architecture, the decoding circuitry effectively implements a binary tree with simple QCA logic gates at each node, that is, one two-input gate in the case of the M-to-1 decoder [5] and two gates for the M-to-N decoder. In the H-memory, as the address is serialized, the QCA circuitry at each node is complicated; a total of six QCA gates are required with multiple feedback loops. Therefore, the number of QCA gates (that is, MVs) required for decoding the M-bit address for the N memory cell space is given by

$$G_{M-to-N} = 6 \times (2^M - 1). \tag{8}$$

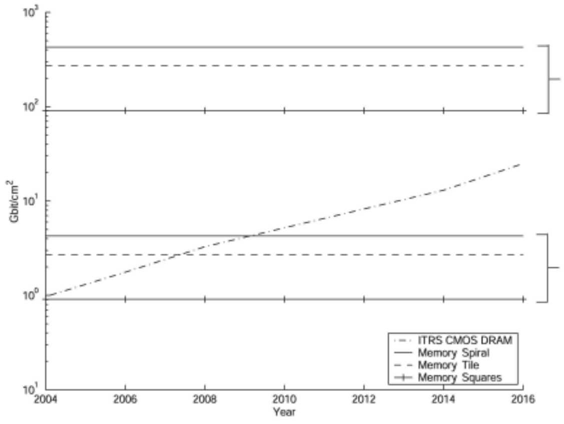


Fig. 17. Density comparisons of CMOS/QCA serial memory architectures (projected).

Latency in decoding is also increased: Although the number of levels of the nodes is the same as in previous decoding designs, the complexity and computation at each node are high (in previous designs, the signal must pass through a single QCA gate at each level). Therefore, the latency in memory address decoding is also high.

11 Memory Density

Fig. 17 shows the projected memory densities for DRAM using CMOS technology and for serial memory architectures using QCA technology. DRAM density projections are obtained from [13]. When calculating QCA memory densities, cell sizes in the range of 1 nm up to 10 nm are assumed (through either molecular or metallic implementations). The memory spiral architecture in [4] requires an area of $15d \times 15d$ QCA cells per memory cell, whereas the architecture proposed in this paper takes an area of $18.5d \times 18.5d$ QCA cells (where d is the interdot distance). The area requirements per bit are calculated for a memory of size 256 with 12-bit words (inclusive of input/output and decoding circuitry). For decoding, the spiral memory architecture uses router cells [4], whereas the proposed tile-based architecture uses the decoder presented in a previous section.

The memory architecture designed using the SQUARES formalism [5] exhibits a relatively low density. It requires an area of $32d \times 32d$ QCA cells. This low density occurs because, even though the number of QCA cells for implementing the memory loop is small, there is still a substantial amount of wasted area (the goal of SQUARES is to simplify the engineering design process using uniformly sized logic blocks; it has been shown that, in each block, the wasted area accounts for more than 50 percent); moreover, the feature of making data in each loop bit addressable results in complex control and decoding circuits.

In Fig. 17, it can be observed that the proposed serial QCA memory architecture, even with metal-dot implementations (at a cell size of 10 nm), allows memory densities that can only be matched after some years by using conventional CMOS technology. For molecular implementations (at a 1 nm range), QCA memory architectures offer incredible densities, placing them well above the range of CMOS technology.

12 Conclusions

This paper has proposed a novel serial memory architecture for QCA implementation. This architecture is based on utilizing new building blocks (referred to as tiles) in the storage and input/output circuitry of the memory. The QCA paradigm of memory-in-motion has been accomplished using a novel arrangement in the storage loop and timing/clocking; a three-zone memory tile has been proposed by which information is moved across a concatenation of tiles by utilizing a two-level clocking mechanism. In the proposed memory, clocking zones are shared between memory cells and the length of the QCA line of a clocking zone is independent of the word size. QCA circuits for address decoding and input/output for simplification of the Read/Write operations have been discussed in detail. An extensive comparison of the proposed architecture and previous QCA serial memories has been pursued in terms of latency, timing, clocking requirements, and hardware complexity. This analysis has shown that the proposed memory architecture is readily applicable to QCA implementation and provides excellent figures of merit compared with other QCA-based serial memories.

References

[1] C.S. Lent and P.D. Tougaw, "A Device Architecture for Computing with Quantum Dots," *Proc. IEEE,* vol. 85, pp. 541-557, 1997.

[2] M.T. Niemier and P.M. Kogge, "Problems in Designing with QCAs: Layout=Timing," *Int'l J. Circuit Theory and Applications,* vol. 29, no. 1, pp. 49-62, 2001.

[3] D. Berzon and T.J. Fountain, "A Memory Design in QCAs Using the SQUARES Formalism," *Proc. Ninth Great Lakes Symp. VLSI,* pp. 168-172, 1999.

[4] S.E. Frost, A.F. Rodrigues, A.W. Janiszewski, R.T. Rausch, and P.M. Kogge, "Memory in Motion: A Study of Storage Structures in QCA," *Proc. First Workshop Non-Silicon Computation,* 2002.

[5] K. Walus, A. Vetteth, G.A. Jullien, and V.S. Dimitrov, "RAM Design Using Quantum-Dot Cellular Automata," *Technical Proc. Nanotechnology Conf. and Trade Show,* vol. 2, pp. 160-163, 2003.

[6] M.T. Niemier, A.F. Rodrigues, and P.M. Kogge, "A Potentially Implementable FPGA for Quantum Dot Cellular Automata," *Proc. First Workshop Non-Silicon Computation,* 2002.

[7] M. Lieberman, S. Chellamma, B. Varughese, Y. Wang, C.S. Lent, G.H. Bernstein, G. Snider, and F. Peiris, "Quantum-Dot Cellular Automata at a Molecular Scale," *Annals of the New York Academy of Sciences,* vol. 960, pp. 225-239, 2002.

[8] C.S. Lent, P.D. Tougaw, and W. Porod, "Quantum Cellular Automata: The Physics of Computing with Arrays of Quantum Dot Molecules," *Proc. Workshop Physics and Computation,* 1994.

[9] K. Hennessy and C.S. Lent, "Clocking of Molecular Quantum-Dot Cellular Automata," *J. Vacuum Science and Technology B,* vol. 19, no. 5, pp. 1752-1755, 2001.

[10] I. Amlani, A.O. Orlov, G. Toth, C.S. Lent, G.H. Bernstein, and G.L. Snider, "Digital Logic Gate Using Quantum-Dot Cellular Automata," *Science,* vol. 284, no. 5412, pp. 289-291.

[11] S.E. Frost, A.F. Rodrigues, A.W. Janiszewski, R.T. Rausch, and P.M. Kogge, "Memory in Motion: A Study of Storage Structures in QCA," *Proc. First Workshop Non-Silicon Computation,* 2002.

[12] A.O. Orlov, I. Amlani, G.H. Bernstein, C.S. Lent, and G.L. Snider, "Realization of a Functional Cell for Quantum-Dot Cellular Automata," *Science,* vol. 277, pp. 928-930, 1997.

[13] R. Compano, L. Molenkamp, and D.J. Paul, "Technology Roadmap for Nanoelectronics," *European Commission IST Programme, Future and Emerging Technologies,* 2000.

[14] P.D. Tougaw and C.S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata," *J. Applied Physics,* vol. 75, no. 3, pp. 1818-1825, 1994.

[15] K. Walus, A. Vetteth, G.A. Jullien, and V.S. Dimitrov, "RAM Design Using Quantum-Dot Cellular Automata," *Technical Proc. Nanotechnology Conf. and Trade Show,* vol. 2, pp. 160-163, 2003.

[16] K. Walus QCADesigner Homepage, ATIPS Laboratory, Univ. of Calgary, Canada, http://www.qcadesigner.ca/index.html, 2004.

[17] E.P. Blair and C.S. Lent, "An Architecture for Molecular Computing Using Quantum-Dot Cellular Automata," *Proc. Third IEEE Conf. Nanotechnology,* vol. 1, pp. 12-14, 2003.

[18] V. Vankamamidi, M. Ottavi, and F. Lombardi, "A Line-Based Parallel Memory for QCA Implementation," *IEEE Trans. Nanotechnology,* vol. 4, no. 6, pp. 690-698, Nov. 2005.

[19] V. Vankamamidi, M. Ottavi, and F. Lombardi, "Clocking and Cell Placement for QCA," *Proc. Sixth IEEE Conf. Nanotechnology,* vol. 1, pp. 343-346, June 2006.

Vamsi Vankamamidi received the BS degree in computer engineering from the University of Mumbai, India, in 2000 and the MS degree in electrical engineering and computer science from the University of Toledo, Ohio, in 2001. He is currently working toward the PhD degree in computer engineering at Northeastern University, Boston. As part of his dissertation, he is working on quantum-dot cellular automata (QCA), a nanoscale device architecture to supersede conventional silicon-based technology. His research interests include the design of nanoscale circuits and systems, electronic design automation, defect tolerance, and reliability.

Marco Ottavi received the Laurea degree in electronic engineering from the University of Rome "La Sapienza," Rome, in 1999 and the PhD degree in microelectronic and telecommunications engineering from the University of Rome "Tor Vergata," Rome, in 2004. In 2000, he was with the ULISSE Consortium, Rome, as a designer engineer of digital systems for space applications. In 2003, he was a visiting research assistant with the Electrical and Computer Engineering Department at Northeastern University, Boston. From 2004 to 2007, he was a postdoctoral research associate at Northeastern University and, in 2006, he was a visiting research scholar at Sandia National Laboratories, Albuquerque, New Mexico. He is currently a senior design engineer with Advanced Micro Devices, Boxborough, Massachusetts. His research interests include yield and reliability modeling, fault-tolerant architectures, and online testing and design of nanoscale circuits and systems. He is a member of the IEEE.

Fabrizio Lombardi received the BSc (Hons.) degree in electronic engineering from the University of Essex, United Kingdom, in 1977 and the master's degree in microwaves and modern optics in 1978, the diploma in microwave engineering in 1978, and the PhD degree in 1982 from University College London, University of London. He joined the Microwave Research Unit at the University College London in 1977. He is currently the holder of the International Test Conference (ITC) Endowed Chair Professorship at Northeastern University, Boston. At the same institution, during the period 1998-2004, he served as the chair of the Department of Electrical and Computer Engineering. Prior to joining Northeastern University, he was a faculty member at Texas Tech University, the University of Colorado, Boulder, and Texas A&M University. His research interests are bio-inspired nanomanufacturing/nanocomputing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in these areas and coauthored/edited seven books. Since 1 January 2007, he has been the editor-in-chief of the *IEEE Transactions on Computers*. He is also an associate editor of *IEEE Design and Test* magazine and the *ACM Journal of Emerging Technology in Computing Systems*. He has been the chair of the Committee on "Nanotechnology Devices and Systems" of the Test Technology Technical Council of the IEEE since 2003. He was an associate editor from 1996 to 2000 and an associate editor-in-chief from 2000 to 2006 of the *IEEE Transactions on Computers*. He was a guest editor of special issues in archival journals and magazines such as the *IEEE Transactions on Computers*, *IEEE Transactions on Instrumentation and Measurement*, *IEEE Transactions on VLSI*, *IEEE Micro* magazine, and *IEEE Design and Test* magazine. He has been involved in organizing many international symposia, conferences, and workshops sponsored by professional organizations. He is the founding general chair of the IEEE Symposium on Network Computing and Applications. He was a Distinguished Visitor of the IEEE Computer Society twice (1990-1993 and 2001-2004). He has received many professional awards, including the Visiting Fellowship at the British Columbia Advanced System Institute, University of Victoria, Canada (1988), two Texas Experimental Engineering Station Research Fellowships (1991-1992 and 1997-1998), the Halliburton Professorship (1995), the Outstanding Engineering Research Award at Northeastern University (2004), and an International Research Award from the Ministry of Science and Education of Japan (1993-1999). He was the recipient of the 1985/86 Research Initiation Award from the IEEE/Engineering Foundation and a Silver Quill Award from Motorola-Austin (1996). He is a senior member of the IEEE and a member of the IEEE Computer Society.