Memory in Motion: A Study of Storage Structures in QCA

Sarah Elizabeth Frost
University of Notre Dame
Dept. of Comp. Sci. and Eng.
Notre Dame, IN 46556, USA
sfrost@nd.edu

Arun Francis Rodrigues
University of Notre Dame
Dept. of Comp. Sci. and Eng.
Notre Dame, IN 46556, USA
arodrig6@nd.edu

Andrew Walter Janiszewski
University of Notre Dame
Dept. of Comp. Sci. and Eng.
Notre Dame, IN 46556, USA
ajanisze@nd.edu

Randal Thomas Rausch
University of Notre Dame
Dept. of Comp. Sci. and Eng.
Notre Dame, IN 46556, USA
rrausch@nd.edu

Peter M. Kogge
University of Notre Dame
Dept. of Comp. Sci. and Eng.
Notre Dame, IN 46556, USA
kogge@wizard.cse.nd.edu

Abstract

Quantum Cellular Automata (QCA) is a new technology that replaces current flow as an information carrier by coulombic interactions of electrons within confined configurations. Prior work has investigated its potential as very dense logic. This paper uses a defining characteristic of such devices, namely pipelining that occurs even at the wire level, as the mechanism for memory structures that are formed recursively, rather than as conventional CMOS arrays, and hold the potential for extremely dense storage with embedded processing capabilities.

1. Introduction

As computing components continue to shrink according to Moore's law, serious barriers to transistor based computing loom ahead [6]. Already dissipating heat from high performance high density CMOS components is a significant challenge, and one which will only become more difficult. The exponential increases in fabrication plant costs also endanger CMOS advancement. More threatening are the inherent physical constraints which will be encountered as feature size shrinks. Quantum mechanical effects, interconnect limitations, and lithographic difficulties will halt the advancement of transistor based fabrication, perhaps as soon as 2010.

Quantum Cellular Automata (QCA) offers a novel alternative to the transistor paradigm [9] [1]. A QCA cell (Figure 1(a)) is comprised of four sites, two of which can be inhabited by electrons. Coulombic repulsion between the electrons will drive them to opposite corners of the cell. The state of a cell is represented by the configuration of the electrons within. Similarly, Coulombic interaction between neighboring cells can be used to propagate configurations to neighboring cells (Figure 1(b)) by causing neighboring cells to tunnel from one site to another. In this manner state can be propagated along a line of adjacent QCA cells, forming a wire. Other structures can be built to perform logical operations [14]. Coulombic interaction causes electrons to move within a cell, not between cells, so there is no current flow. This avoids many of the heat dissipation and power consumption problems of transistor computing. Unlike CMOS, in QCA the transmission media and logical elements are both comprised of the same basic block - the cell. As such, QCA has been called "processing-in-wire."

To avoid the loss of coherence in QCA circuits, a four phase clocking system has been developed [5]. In this system, an electric field is applied to the cells. This field raises or lowers the tunneling barriers between electron sites within a QCA cell. This has the effect of preventing or allowing electrons from changing positions or influencing neighboring cells (see figure 2). Cells can be grouped into zones so that the field influencing all the cells in a zone will be the same. A zone cycles through 4 phases. In the **Switch** phase, the tunneling barriers in a zone are raised. While this occurs, the electrons within the cell can be influenced by the Coulombic charges of neighboring zones. Zones in the **Hold** phase have a high tunneling barrier and will not change state, but can influence other adjacent zones. Lastly, the **Release** and **Relax** decrease the tunneling barrier so that the zone will not influence other zones. These zones can be of irregular shape, but their size must be within certain limits imposed by fabrication and dissipation concerns. Proper placement of these zones is critical to design efficiency.

Already QCA cells have been implemented [12] using metal island dots under low temperatures, but an interesting possibility is the implementation of QCA cells on the molecular level [8]. By using single molecules as the cells and

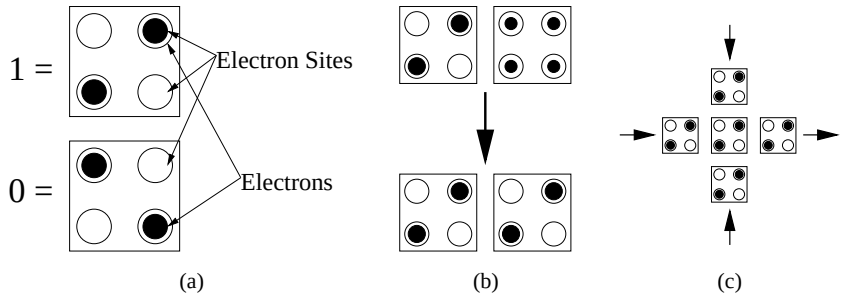

Figure 1. (a) QCA cells represent state by electron configuration. (b) Coulombic Interaction between neighboring cells propagates state. (c) A QCA majority gate

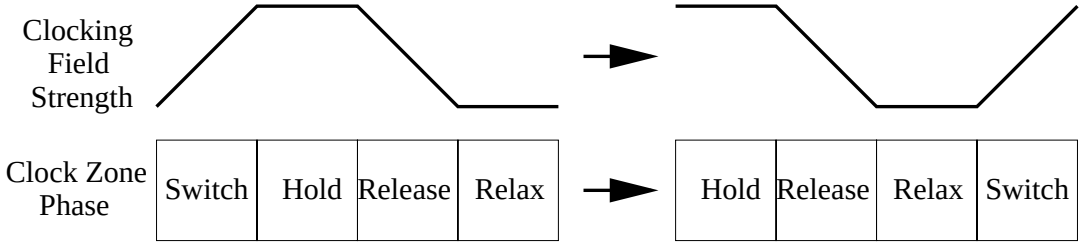

Figure 2. Data moves from a zone which is in Hold to a zone in the Switch phase

regions within the molecule as sites for electrons, molecular QCA holds the promise of densities upwards of 10^{13} devices per cm^2. Clock speeds for these cells could be in the 1 to 10 Terahertz range at room temperature. Power consumption should be far less than end-of-line high performance transistors.

Design and demonstration of a variety of basic logic structures in QCA [14], combined with the lure of higher densities and speeds, has prompted research into larger scale QCA architectures [10] [4]. Research has shown that logic elements can be built from QCA. However for a full computer, we must also store data. Previous QCA memory designs have focused on functionality, not optimization, and have resulted in bulky and slow designs.

QCA is so fundamentally different than CMOS that simply scaling and translating CMOS designs for memory and logic has proven inefficient. To achieve high densities and performance, QCA systems must adopt new design paradigms which embrace and take advantage of the "processing-in-wire" [11] nature of this technology. In section 3.1, we explore several of these issues and how they apply to the design of efficient memory structures. The exploration of these issues lead to the construction of the H-memory structure. A comparison of this structure (in section 4) shows that it surpasses other QCA memories and promises significant improvements over end-of-the-line transistor memory technology. Additionally, the nature of the H-memory design takes full advantage of the processing-in-wire aspect of QCA, allowing exciting possibilities for optimization, use as a cache, and integration with logic elements. We explore these possibilities in section 5.

2. Present Memory Structures

In current CMOS memory, one bit memory cells are arranged in a two dimensional grid structure. Incoming addresses are decoded to generate a two dimensional select signal, one dimension of which selects a row of the memory grid and one which selects a column or set of columns. The memory cells which are at the intersection of these two select signals is activated. In the case of a read, the cell will send the value of its state to the memory output; in the case of a write, the cell will change its stored value to that determined by a input to the memory. The internal structure of a memory cell is either two interlocking invertors or a capacitor. Several memory grids can be combined to to create words of arbitrary length.

Several problems present themselves if we try to "translate" this structure into QCA. The first problem is that of density. The single bit cell inefficiently consumes 4 clocking zones. Additionally, control becomes an issue. A CMOS memory relies upon setting the the row and column select signals in a negligible amount of time. In QCA, generating and propagating these signals is complex and in the multiple clock cycle range.

Clearly, QCA memory designs must circumvent the limitations of a grid design and single bit cell. The density of QCA memory can be increased by storing multiple bits within each memory cell. A spiral pattern allows multiple clock zones to be "shared" by QCA cells. A pure spiral is appropriate for small words, larger words could utilize different configuratoins, such as a series of smaller spirals. Control signals generated at the "edge" of a memory structure are not able to propagate to the memory cell in time. To take advan-

tage of QCA, control must be embedded within the memory structure, adjacent to the memory cells themselves.

It was with this intent that alternative memory storage schema were explored. This search resulted in the design of a new, uniform access memory structure. In the following section we develop the rationale for such a design, the major macros, and the overall charecteristics.

3. The H-Memory Structure

The basic structure of the memory is a recursive H structure (see Figure 3), similar to those used to achieve zero-skew clock routing in systolic arrays [13].

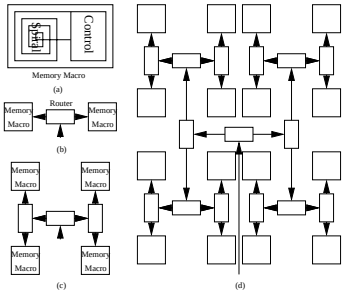

Figure 3. (a) A memory macro contains a single word. (b) Router macros connect memory macros. (c,d) A recursive H-structure is built with routers forming interior nodes and memory macros forming leaves.

3.1. Design Rules

In QCA, conventional design wisdom must be rethought. Whereas transistor and current based designs have transistors, wires, and capacitors to construct memory out of, QCA memories must be constructed only of QCA cells and clocking zones. The ramifications of this directed the working design rules for our design.

Unlike transistor and current based designs, signals do not propagate down a wire in negligible time. Instead, a signal moves down a string of QCA cells, traveling one clock zone every phase of the four phase clock (see figure 2). This lead to the driving design rule, that first and foremost, **latency is directly related to design size.**

Additionally, unlike current based systems, QCA has no direct equivalent of a capacitor to hold state in. To store state it must be kept in a ring of QCA cells arrayed through a 'loop' of clocking zones. **Memory must be kept in motion.**

QCA technology will be comprised of the QCA cells and the clocking zones which are generated by traditional CMOS circuitry underneath the QCA molecules. Because the clocking zones will be generated by a different technology from the molecular cells, it is preferred to have regular clock zones of a minimum size. An additional concern is raised by dissipation effects. The longer a QCA "wire" is, the greater the chance that a signal will become incoherent. The exact maximum size will depend on the molecule used. **Clock zones should be regular in shape and bounded in size.**

These rules lead to the employment of serial logic rather than parallel. Because latency is directly related to wire length, parallel transmission runs into the problem that "turning corners" becomes difficult. (The "inside" bit wire is significantly shorter than the "outside".) A price is paid for streaming serial data, but this is quickly compensated for by simpler and smaller control logic.

3.2. Early Designs

Early designs demonstrated the hazards of non-uniform access times. As layout is the direct source of timing issues, it became evident that a uniform path length was preferable if possible, if only as a worthwhile first effort in designing a complete memory substructure. It has since been seen that this is a very efficient approach to QCA memory design.

3.3. General Structures and Protocol

The H-Memory presented here is a QCA realization of a complete binary tree, where each leaf contains a memory cell and each node houses a routing circuit. It recursively defines a highly compact and scalable memory system with the desired uniformity in access times.

To access a memory cell, a data packet is constructed that will be serially sent down the address/data line. At the head of the packet is the address of the memory cell, most significant bit first. Following the address is a one bit operation code signaling a read or a write operation. If the operation is a write, the data packet ends with the word to be written. For instance, in a 256 word H-Memory with 12 bit words, the data packet for a write operation would consist of 21 bits: an 8 bit address, a 1 bit op code, and a 12 bit word to be written. These numbers were chosen to match a particular trial CPU design we were investigating. For the read operation in this memory, the data packet would consist of just 9 meaningful bits.

In parallel with the data packet is the select packet. The select packet is a serial one that will signal the appropriate memory cell that the operation described by the data packet is to be done at the particular cell (see figure 4. This select packet serves a similar purpose to that of row and column select signals in traditional CMOS memories but is "logarithmic." The size of the select packet is not dependent on the operation to be performed. This is because a read operation needs the same amount of time as the write operation. Instead of waiting for the bits of data to enter the memory, the wait is for the bits to cycle out of memory. The details of these operations are discussed in section 3.4.

The full data packet enters the memory at the central node and is sent to every leaf in the memory. The select packet arrives only at the memory cell to be operated upon. Each bit of the address is responsible for making one of the decisions

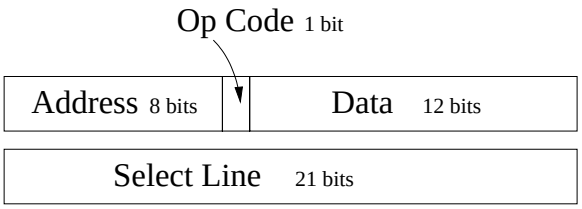

Figure 4. Data packet components and companion select packet.

to turn to the right or left child router. At each turn, a bit at the head of the select packet is consumed. Address bits which have been used arrive before the shortened select stream and thus are ignored.

The data out line of each memory cell is again serial, and is combined back along the H-structure. If a memory cell is not being read, its data out line sends out zeros. At each node, the left and right data-out lines are combined by an OR gate. This guarantees a uniform length and time for data to return to the center of the H-structure and exit the memory.

3.4. Memory Macro

At each leaf of the H-structure is a memory macro (see figure 5) which can read, write, and maintain the currently stored data. The memory macro can be divided into two logical parts, the memory macro control and the memory loop. The memory macro control determines the action of the memory macro based on the signals it receives. The memory loop does the actual storage of data and is simple a tightly wrapped spiral.

Each memory macro has two input lines – a select line and a data line. If a memory macro is to be read from or written to, the select line delivers a string of ones that arrive concurrently with the control signals and data, otherwise the select line will deliver a string of zeros.

3.4.1 Memory Loop

The data loop is a spiral that feeds back into the loop control MUX. The memory macro in figure 5 stores one twelve bit word to allow an easy interface with other QCA based processors being designed [10]. Since the data in the memory loop is always moving, a bit is being written to the data loop every clock cycle. If the memory macro is not selected or if the memory macro is doing a read operation, the data currently in the data loop is recycled by the loop control MUX and is rewritten to the data loop. For a write operation, each bit is replaced by a new bit of data.

In a write, when the write enable bit and the word to be written arrive at the memory macro the operation code bit is consumed, activating the write path. This first bit of the new data and the stored data reach the MUX inouts at the same time. Since the write path is activated, the first bit written into the data loop is the first bit of the new data.

This design exploits the features unique to QCAs. The design of the memory macro control takes good advantage of the majority gate. Keeping the memory in motion makes reading and writing simple while countering dissipation and keeping the memory coherent.

3.5. Router Macro

Each router (see figure 6) has been made into a deterministic finite automaton which sends the select bit stream in only one direction based on one bit from the address stream. See figure 6. By embedding routers along the memory path, control logic (and therefore size and latency) for individual memory cells diminished greatly.

The critical bit in the address stream's arrival is timed to coincide with the first bit of the select stream. However, the machine must be able to access its previous select input and output. Luckily, QCA is capable of latching previous values by simply employing a four clock zone loop which feeds back into the input state.

This deceptively simply feature is quite useful in many designs. By routing a path through extra clocking zones, the values being transmitted are necessarily delayed, allowing feedback. Turning that path back into a majority gate provides a delayed signal. As will be described, this process was instrumental to router design.

There are three majority gates for each memory access path per router. The primary gate has as its output the new select stream, and takes as input the select stream and two other gate outputs. These gates are slightly more complicated. The lower is a majority gate where one input is frozen to binary '0' creating an AND gate. Its other inputs are the select stream and the previous output of the main select gate; the purpose behind this gate is to continue to transmit the stream once it has started (if select out is high and select in is high, return high) and stop when the select stream input goes low.

The top gate controls the transmission of the first bit (as the lower gate can never return a binary '1' until one has already been returned by the machine). It does this by simply checking if the previous value of select entered into the machine was low, and the current bit in the address stream is correct for the direction to be taken in the tree. The output will actually be high constantly in all left branches while nothing is being transmitted; this is not a problem, as the other two inputs to the main majority gate must necessarily be low in all such cases.

3.6. Density and Latency

Another benefit of the recursive H structure is its scalability. A general H-tree layout is highly packed and consumes area $O(n)$, where n is the number of words stored and the maximum wire length is bounded by $O(\sqrt{n})$[15]. We have completed and simulated a detailed H-memory comprised of 256 12 bit words. This design exhibits a density of 218 $cell/bit$. With a 12-bit word, only 8.5% of the total memory area is comprised of memory loop area. The bulk

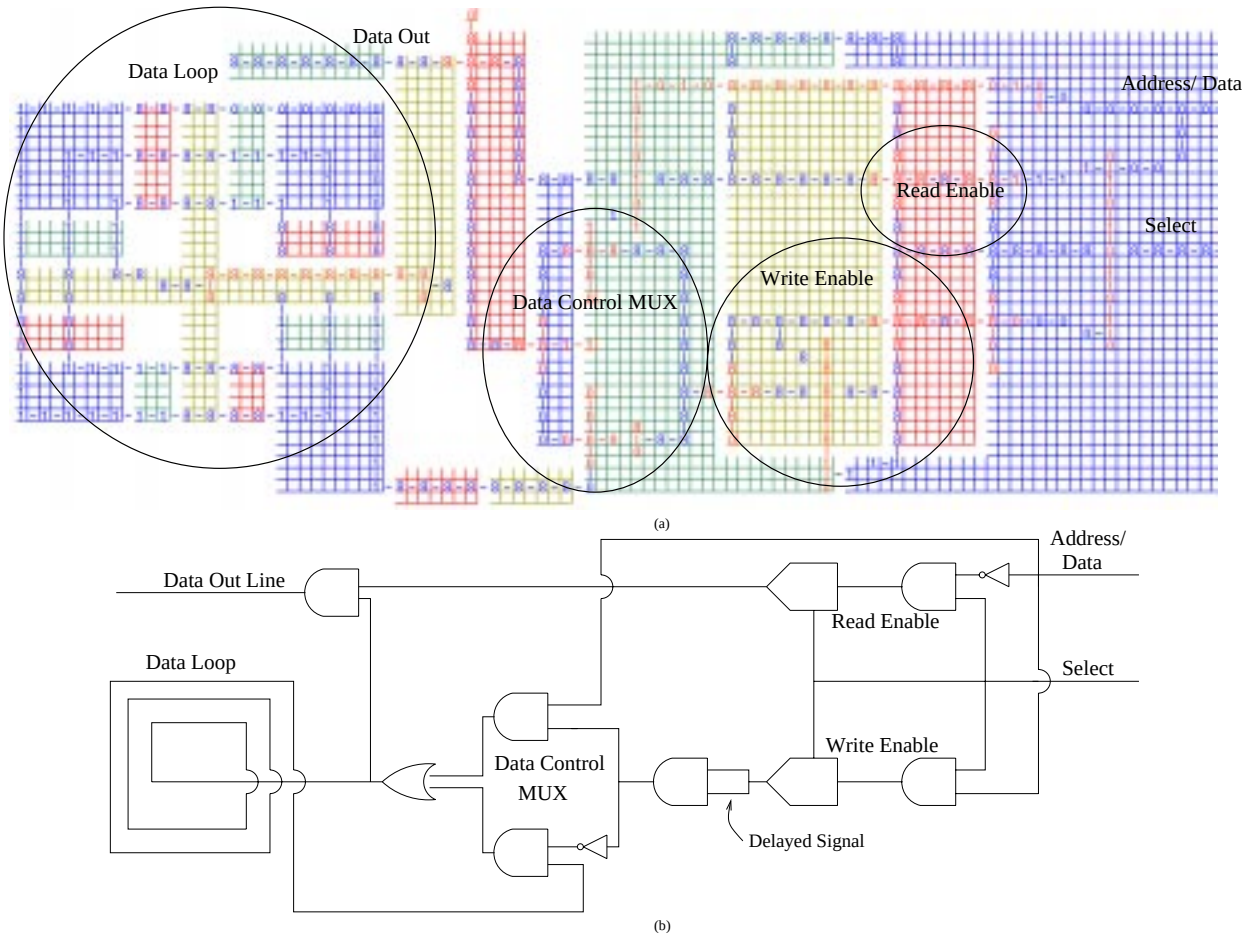

Figure 5. (a) The H memory macro in QCA. (b) The logic level equivalent.

of the design is unused space or devoted to the routing and control mechanisms. As larger memory words are used, the area required per bit drops quickly (figure 7).

Because latency is directly related to design size, the number of clock cycles required for a word to be accessed is a function of the size of the word and the number of words stored in the memory. The rate of increase in access time can be seen in figure 8.

4. Comparisons

Previous work in designing QCA memory structures has focused on translating a conventional silicon SRAM architecture directly into QCA.

The memory designed by the SQUARES architecture [2] was one such attempt to mimic a CMOS grid. The component based architecture made inefficient use of space (consuming at least 2200 $cells/bit$) with resulting high latency.

Because the SQUARES cell did not focus on optimization, it made a poor candidate for comparison against the H-memory. Hence it was decided to design a new memory architecture, the QCA-GRID, which emphasized high density and low latency but still followed a conventional SRAM grid architecture. To achieve these goals several optimizations were used. The QCA-GRID was not a direct translation of the SRAM logic, but retained the conventional grid layout, including row and column selects. Proper layout of clockzones allowed memory loops to "share" zones, increasing density and provided channels for select and data signals. These optimizations allowed densities to approach 324 $cells/bit$.

4.1. Density Comparisons

It should be possible to space molecular QCA cells 4.2 to 10 nm apart, allowing incredible densities compared to end-of-line CMOS technology. Assuming this cell spacing, the present 12-bit designs would exhibit densities of 218 $cell/bit$ and 324 $cell/bit$, or between 24.12 and 4.26 $GBit/cm^2$, a density which should be equaled by CMOS sometime in the next decade [6]. However, if one samples the density of just the memory loop of the H-memory, the spiral architecture allows a density of only 18.75 $cell/bit$ or between 281.58 and 49.67 $GBit/cm^2$, placing potential QCA densities well above those of CMOS (see figure 9).

The performance of of the 12-bit H-memory structure is due in large part to the relatively small word size. By placing

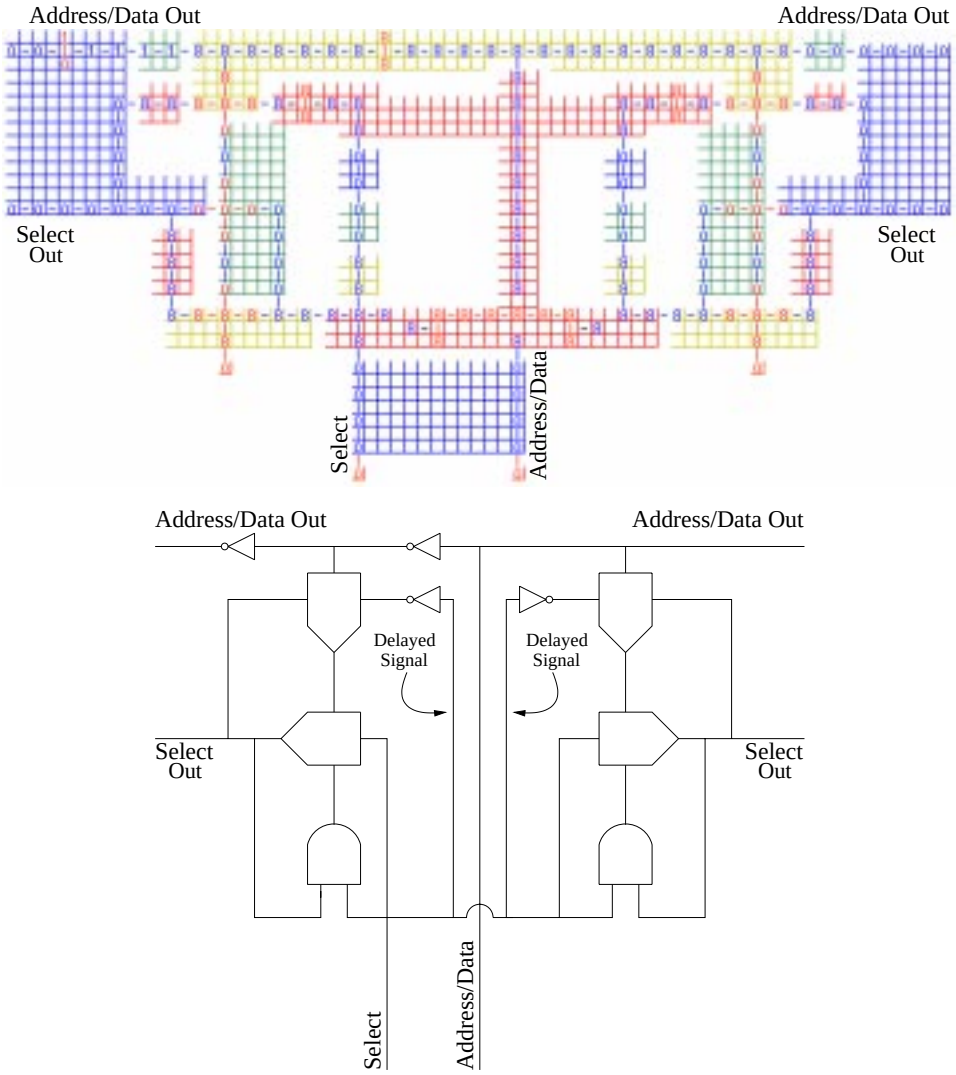

Figure 6. (a) The H memory router in QCA. (b) The logic level equivalent.

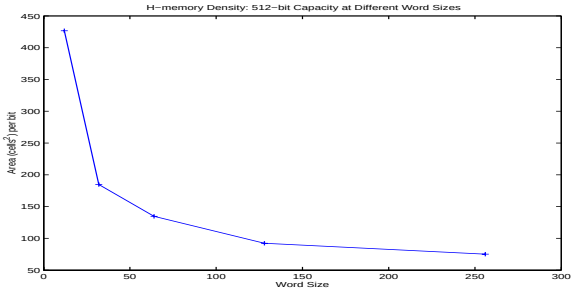

Figure 7. The area required per bit falls significantly as the word size increases.

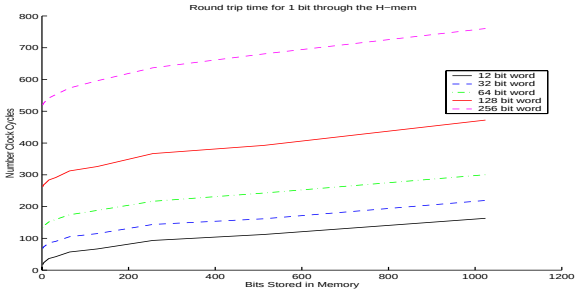

Figure 8. Access time several word sizes versus the number of words in the memory.

multiple words at each leaf and optimizing the basic structure (see section 5.1), H-memory density will grow towards the maximum QCA density.

4.2. Latency Comparisons

A comparison against CMOS is difficult at this point because so much will depend upon fabrication mechanisms. However, an analysis of the optimized QCA memories, the H-memory and the QCA-GRID, is possible.

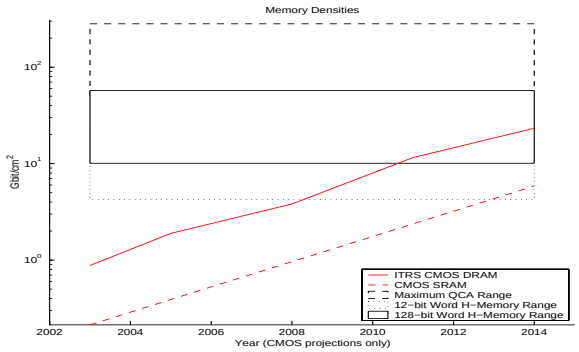

Figure 9. Projected QCA and CMOS memory density.

Because the QCA-GRID memory utilizes a grid-like storage memory access incurs overhead to generate control signals and reassemble data from the grid. In contrast, the H-memory structure embeds the router macros (see section 3.5) on route to the memory elements, taking advantage of the "processor-in-wire" paradigm. As a result, the H-memory structure is able to provide roughly 50% lower latencies than the optimized QCA-GRID.

5. Future Work and Conclusions

The H-memory structure presented above presents many benefits. In addition to significant improvements over CMOS, it also compares favorably to other QCA memory designs. As well as the benifits of latency, power dissipation, and density, the H memory offers significant architectural advantages. The "Zero Skew" properties of the recursive H pattern have already been utilized for clocking signals, we utilize them to provide uniform memory access. This uniformity, combined with the fact that multiple memory accesses can be pipelined through the structure, could allow for easy multithreading.

The flexibility of the H-structure opens the possibilities for a number of optimizations.

5.1. Optimizations

The schematics presented thus far are the first cut designs for the H-memory. There are several ways the design can be optimized. The most basic of these areas are reducing empty space and maximizing the sharing of clocking zones. In the memory cell shown in figure 5, there are six regions of empty space which together occupy approximately 28% of the area of the memory macro. In addition, there is empty space in the tree. For each subtree of four memory macros, approximately a quarter of the space occupied by the subtree is left empty. The wasted space can be minimized by moving from a long rectangular memory macro shape towards a more square shape. More involved optimizations include enabling multiple read ports, storing multiple words at each leaf, allowing non-uniform access times, supporting elementary processing in memory, and providing support for semaphores.

The density of the design can be substantially increased. The space between the read and write enables and the loop control MUX can be used. The center of the splitter macro design also provides an opportunity for improving density. The density can also be improved by combining multiple wires into adjacent clocking zones, decreasing wasted space and allowing more regular clock zones. Sharing clocking zones will increase the density of the overall structure. Improving the space efficiency will also improve the time efficiency because of the close connection between layout and timing in designing with QCAs.

5.2. Cache Functionality

Using H-Memory as a structural basis for a cache hierarchy shows excellent potential. Only a small number of modifications need to be made to the cell and the router devices. While memory access times will continue to be a challenge for future QCA designs, speculative size estimates show significant promise. Further, due to the possibility for much higher levels of associativity in every level of cache, hit rates should be very impressive by comparison.

A fundamental paradigm shift in QCA will be required by the disappearance of single cycle cache hits. These are an impossibility in so finely pipelined an architecture. However, this very pipelining could be exploited to tolerate the increased cache latency, by the use of threads (see section 5.3).

Cache cells would necessarily contain two loops rather than just one for raw memory to accommodate some form of tagging. Some control circuitry could potentially be shared, so this could have a minimal impact on the memory cell size. It would add a constant time constraint (on the order of the size of the tag) for hit verification. An equivalence unit for tag comparison can be constructed very easily in QCA for serial data. The most striking change in this new medium, however, is the ability to realize far greater levels of associativity than have been previously seen.

Router macros can be modified to pass select streams to more than one cell. A negligible amount of logic is required to implement a psuedo-least recently used algorithm. The relative ease of implementing caches will help tolerate the latencies of QCA memory.

5.3. Integrated Logic

In the traditional silicon world, the observation that the memory access latency is a significant bottleneck on computation has lead to a number of attempts to merge processing logic with memory storage [7] [3]. In QCA, distance is tied even more closely to latency, so integrating logic into QCA memory structure seems natural.

The H-memory structure is uniquely suited to this application. Because it is based upon a binary tree, there is space to add logic at interior nodes. One mechanism would be to

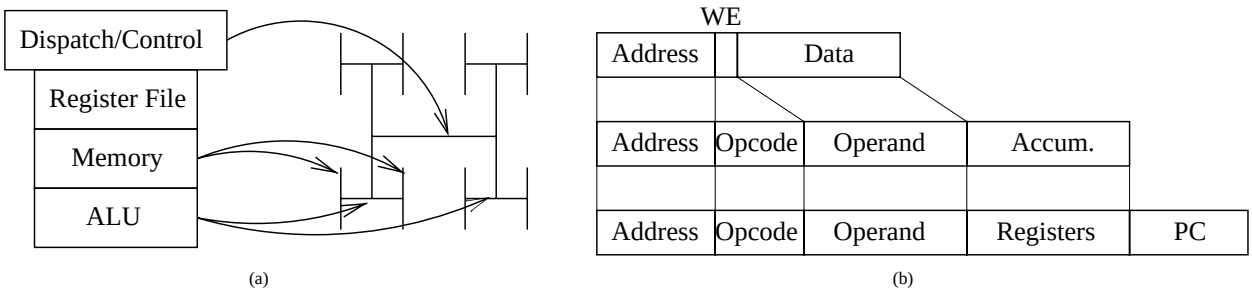

Figure 10. (a) Integrating logic into the H-memory breaks the traditional pipeline stages and distributes them through the structure. (b) The addressing format could be modified to contain an opcode and operands to complete an instruction.

add control logic at the root node, and place computational power (ALUs) at key interior nodes (see figure 10(a)). By distributing computational logic with memory, it is possible to take full advantage of the 'processing-in-wire' nature of QCA.

The basic H structure sends an access packet to a memory node with a single bit (the write-enable) telling the node to either read or write the data. We can extend this by adding more space to the write-enable, making it a full opcode, and attaching a register to the packet (see figure 10(b)). After fetching the memory from a leaf node, an ALU at an interior node can perform the arithmetic portion of the instruction on the return path. Once the now computed instruction reaches the root node, control logic there can redirect the packet to fetch its next instruction. This organization would lend itself to a simple accumulator based ISA. The packet could be further extended to contain multiple registers and more machine state, thus representing an entire thread.

5.4. Conclusions

QCA circuits are naturally pipelined at very fine levels. When used to store data, a particularly dense structure is a simple spiral wire, where data circulates in an endless loop. While arbitrarily large loops could be formed, access times become large, and dependent on how long it takes for the desired data to circulate to the readout point. As an alternative, this paper proposed a memory made of many small spirals, each containing a word, and arranged in a recursive structure. Accesses are inserted into such a structure serially, and take a constant time to reach any word, and either modify it or read out the contents. If QCA is reducible to the molecular level, densities in excess of 50 gigabits per cm^2 may be possible. This is far denser than the best of projected CMOS.

In addition to these properties, the structure of this memory allows the embedding of simple operations at each word, with little additional complexity, and thus permitting a rich array of fetch and op functions. Of even more potential novelty is the expansion of the access packet to contain the entire machine state for a simple thread. Future work will investigate a design where there is nothing but memory, and such threads bounce from word to word as dictated by the program.

References

[1] S. C. Benjamin and N. F. Johnson. A possible nanometer-scale computing device based on an adding cellular automaton. *Applied Physics Letters*, 1997.

[2] D. Berzon and T. Fountain. Computer memory structures using qca. Technical report, University College London, 1998.

[3] J. B. Brockman, P. M. Kogge, V. Freeh, and T. Sterling. Microservers: A new memory semantics for massively parallel computing. In *International Conference on Supercomputing*.

[4] T. J. Fountain. The propagated instruction processor. In *Proc. Work. on Innovative Circuits and Systems for Nanoelectronics, Delft*, 1997.

[5] K. Hennessy and C. Lent. Clocking molecular quantum-dot cellular automata. To appear in the Journal of Vacuum Science Tech.

[6] ITRS. International technology roadmap for semiconductors 2000 update. Technical report, ITRS, 2000.

[7] K. Keeton, R. Arpaci-Dusseay, and D. Patterson. Iram and smartsimm: Overcoming the i/o bus bottleneck. In *Workshop on Mixing Logic and DRAM*, 1997.

[8] C. Lent. Molecular electronics: Bypassing the transistor paradigm. *Science*, 2000.

[9] C. S. Lent and P. D. Tougaw. A device architecture for computing with quantum dots. *Proceedings of the IEEE*, 1997.

[10] M. Niemier. Designing digital systems in quantum cellular automata. Master's thesis, University of Notre Dame, 2000.

[11] M. T. Niemier and P. M. Kogge. Exploring and exploiting wire-level pipelining in emerging technologies. In *International Symposium of Computer Architecture*.

[12] A. O. Orlov, I. Amlani, G. Toth, C. S. Lent, G. H. Bernstein, and G. L. Snider. Experimental demonstration of a binary wire for quantum-dot cellular automata. *Applied Physics Letters*, 1999.

[13] N. Sherwani. *Algorithms for VLSI Physical Design Automation*, chapter 11. Kluwner Academic Publishers, 1999.

[14] P. Tougaw and C. Lent. Logical devices implemented using quantum cellular automata. *Journal of Applied Physics*, 1994.

[15] J. Ullman. *Computational Aspects of VLSI*, chapter 3. Computer Science Press, 1984.