Fault Tolerant Quantum Cellular Array (QCA) Design using Triple Modular Redundancy with Shifted Operands

Tongquan Wei
ECE Department
Polytechnic University at
Brooklyn NY USA 11201
tongquan@photon.poly.edu

Kaijie Wu
ECE Department
University of Illinois at
Chicago IL USA 60607
kaijie@ece.uic.edu

Ramesh Karri
ECE Department
Polytechnic University at
Brooklyn NY USA 11201
ramesh@india.poly.edu

Alex Orailoglu
CSE Department
University of California at
San Diego CA USA 92093
alex@cs.ucsd.edu

Abstract: Due to their extremely small feature sizes and ultra low power consumption, Quantum-dot Cellular Automata (QCA) technology is projected to be a promising nanotechnology. However, in nanotechnologies, manufacture time defect levels and operational time fault rates are expected to be quite high. Straightforward Triple Modular Redundancy (TMR) based fault tolerance is inappropriate for QCA nanotechnology since wire delays dominate the logic delays and faults in wires dominate the faults in a QCA based design. Furthermore, long wires are necessary in TMR based designs. In this paper we show that fault-tolerance can be obtained by using TMR with Shifted Operands (TMRSO). TMRSO uses shorter wires of QCA cells and exploits the self-latching property of clocked QCA arrays to provide the same level of fault tolerance capability as straightforward TMR while being significantly faster and smaller. This technique can be applied to a variety of operations; we have validated TMRSO on adders. Implementation results obtained using QCADesigner [6] show that an 8-bit adder using TMRSO has more than 50% area reduction and more than 100% throughput improvement when compared to a TMR implementation.

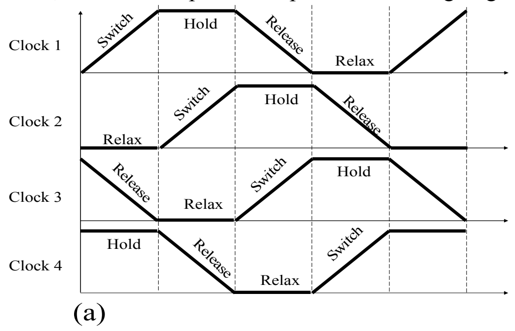
I. Introduction

Scaling of CMOS devices is being aggressively pursued by shrinking transistor dimensions, reducing power supply voltages and increasing operating frequencies. Such aggressive scaling adversely results in non-ideal behaviors such as high leakage current and high power density levels. These issues will eventually become road blocks and slow down the scaling trend that has been operative for years. Quantum-dot Cellular Automata (QCA) proposed in the 1990s [1] are attracting a lot of attention due to their extremely small feature sizes and ultra low power consumption. However, at nanometer scales, it is extremely hard to achieve the required manufacturing tolerances. Hence, fault-tolerance assumes the role of an enabling design technology for QCA and other nanotechnologies. Straightforward application of Triple Modular Redundancy (TMR) based fault tolerance is inappropriate for QCA nanotechnology because, unlike CMOS technology, both the wires and the logic gates are built from quantum cells and wire delays end up dominating the logic delays in QCA. Furthermore, a TMR based design will result in long wires of QCA cells and more associated faults.

This paper presents a novel QCA fault tolerance technique for arithmetic circuits. This method, Triple Modular Redundancy with Shifted Operands (TMRSO), illustrated on an adder, is compared with a TMR adder with respect to fault tolerance capability, throughput and complexity. In contrast to a TMR adder, a TMRSO adder is shown to be able to provide at least the same level of fault tolerance capability while being much faster and smaller. The rest of this paper is organized as follows. In Section II, the clocking scheme of QCA and defects in QCA elements are presented. In Section III, the TMRSO technique is investigated. In Section IV, the TMRSO technique is compared to the TMR technique using QCADesigner [6], a state-of-the-art QCA design tool. Finally, Section V concludes the paper.

II. Clocked QCA system and defects in QCA elements

The clock scheme of QCA makes a QCA system inherently suitable for pipeline computations. Figure 1 (a) shows a popular clock scheme. Each clock has a 90 degree phase shift from its previous clock. Figure 1 (b) shows an array of QCA cells that uses such a clock scheme. The input cell and cell 1 are controlled by **clock 1,** cell 2 is controlled by **clock 2**, cell 3 is controlled by **clock 3**, and output cell is controlled by **clock 4**. Each row in Figure 1 (b) denotes one phase of a clock. At the beginning, all cells are in the relaxed phase, that is, in an unpolarized neutral state. When **clock 1** is in the switch phase, cell 1 is polarized by the input cell. When **clock 1** is in the hold phase and **clock 2** is in the switch phase, cell 1 serves as an input to cell 2. When **clock 2** is in the hold phase and **clock 1** is in the release phase, cell 2 transmits its value to cell 3 and **clock 3** is in the switch phase. When **clock 3**, **clock 2** and **clock 1** are in the hold phase, release phase, and relax phase, respectively, cell 3 passes its value to output cell and **clock 4** is in the switch phase. Finally **clock 4** switches to the hold phase and the cycle repeats itself as a new input is clocked into the cell 1 in the switch phase of **clock 1**. The whole array shown in Figure 1 (b) works like a D-latch, where the input is sampled at the rising edge of **clock 1** and



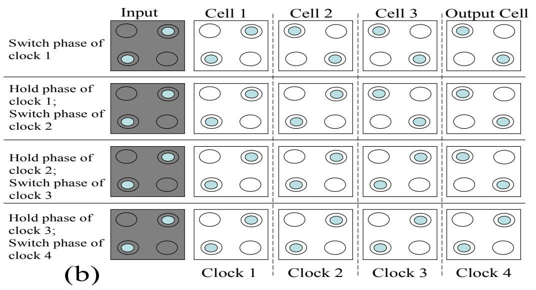

Input
Cell 1
Cell 2
Cell 3
Output Cell
Switch phase of clock 1
Hold phase of clock 1; Switch phase of clock 2
Hold phase of clock 2; Switch phase of clock 3
Hold phase of clock 3; Switch phase of clock 4
(b)
Clock 1
Clock 2
Clock 3
Clock 4

Figure 1 (a) 4 clocks with each one lagging by 90 degrees the previous (b) A QCA array using 4 clocks works like a D-latch

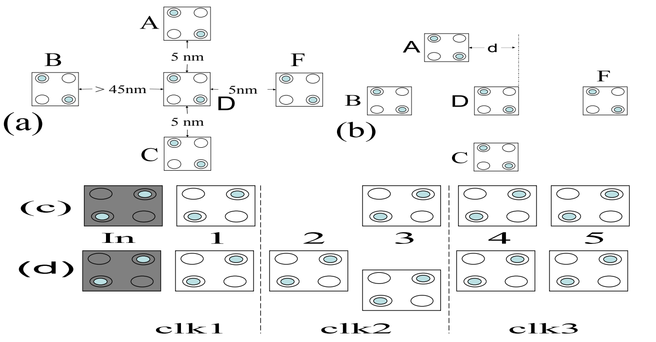


Figure 2 (a) One cell has a displacement from the desired location (b) One cell is misaligned (c) QCA array with missing cells (d) QCA array with misaligned cells

appears at the output at the rising edge of **clock 4**. In fact, any array of cells with a clock sequence(**clock 1→clock 2→clock 3→clock 4**) functions as a D-latch since the cells in one clock get latched and stay latched until the cells of the next clock switch to the hold phase. This is referred to as the *self latching* and *adiabatic pipelining* property of QCA devices [12].

Recently clocked designs using QCA nanotechnology have been proposed. Tougaw and Lent designed a QCA-based one-bit full adder [2]. W. Wang et al. presented a QCA full adder with fewer cells in [4]. A bit-serial adder proposed in [8] modifies the full adder implementation of [4] to include a feedback connection between carry-out and carry-in. A QCA-based carry-look-ahead adder is obtained by connecting the carry out of a full adder to the carry in of the next full adder [3].

All of these logic implementations focus on designing area efficient operators by assuming the proper operation of QCA cells and arrays. However, the proper operation of cells and arrays depends on precise manufacturing. Current semiconductor technologies that are being considered for the QCA implementation would operate correctly only at cryogenic temperatures. To be able to operate correctly at room temperature, the diameter of quantum dots in QCA cells has to be reduced to approximately 2 nm. At these dimensions, it is hard to maintain acceptable process variations. QCA cells could be misplaced or misaligned with respect to their neighbors [5]. Figure 2 shows some of these defects where the cell size is 20nm × 20nm and the distance between adjacent QCA cells is 5nm. In Figure 2 (a) output F will not be polarized when the distance between cell B and cell D is larger than 45 nm. In Figure 2 (b) cell A is misaligned with cell D. The value of output cell F solely depends on input cell B when the displacement between cell A and cell D is larger than 10 nm. Similar defects could exist in QCA interconnect arrays as is shown in Figure 2 (c) and Figure 2 (d). In Figure 2 (c) cell 2 is missing while in Figure 2 (d) cell 3 is misaligned with its neighbors.

There is little work on testing and detecting defects in QCA designs. Baradaran et al. proposed a testing scheme for QCA based designs [9]. Fijany and Toomarian proposed a fault tolerant implementation of a majority gate called *block majority gates* [10]. A block majority gate shows fault tolerance to some of the defects but it does not address the faults in interconnections. Furthermore, its large size may eliminate the advantage of small feature sizes of QCA logic.

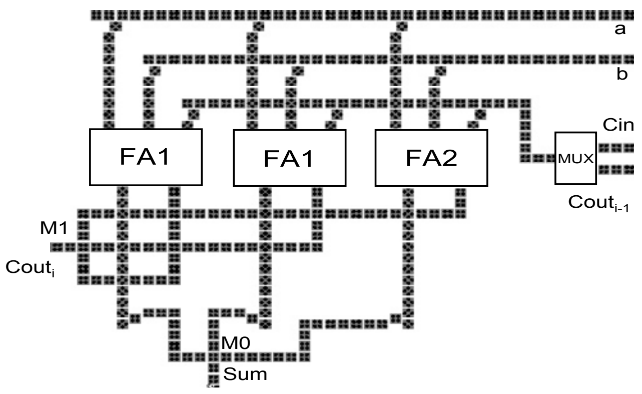


Figure 3 A one-bit TMR adder

III. Fault tolerant QCA

1. *Fault tolerance using Triple Module Redundancy*

In this paper we investigate fault tolerant QCA building block design using a fault tolerant adder as an example. Triple Modular Redundancy (TMR) [11] is a straightforward way to provide fault tolerance capability. A one-bit TMR adder designed using QCADesigner [6] is shown in Figure 3. In a one-bit TMR adder, there are three adders operating in parallel and a two-out-of three majority voting ensures that no single fault in a one-bit adder will yield an incorrect result. The one-bit TMR adder displays a minor degradation in performance due to the additional majority voter.

However, TMR is inefficient for multiple-bit addition in QCA. Consider an 8-bit carry-look-ahead QCA adder proposed in [3]. This adder uses eight full adders in parallel, with the carry out of the preceding full adder supplying the carry in of the successive full adder, forming a long carry propagation chain. In a CMOS implementation only the logic delays are considered and hence the carry chain delay is proportional to the sum of the delays through the carry generation logic. In contrast, in QCA the wire delay dominates the logic delay and hence determines the critical path and the clock duration of the QCA adder with TMR. This is because in QCA nanotechnology, both wires and gates are constructed from QCA cells. While a majority gate uses only 5 QCA cells, the carry chain used in such TMR adders can have more than 100 cells. Overall, the length of the carry chain is proportional to the number of full adders it spans.

2. *Triple Modular Redundancy using shifted operands (TMRSO)*

TMR is not a good choice for designing fault tolerant QCA designs since wires, faults in wires, and wire delays dominate in this nanotechnology. We propose TMR using Shifted Operands (TMRSO) as a new approach to designing fault tolerant QCA designs with lower area overhead and better performance than straightforward TMR. This new method exploits the self-latching and adiabatic pipelining properties of QCA devices to maximize throughput of a system since more than one calculation can be in the pipeline at a given time [12]. TMRSO technique can be applied to data path operations including additions, subtractions, multiplications etc. Since additions are the building blocks of most of the data path operations, we will illustrate the TMRSO concept using a 2-bit adder.

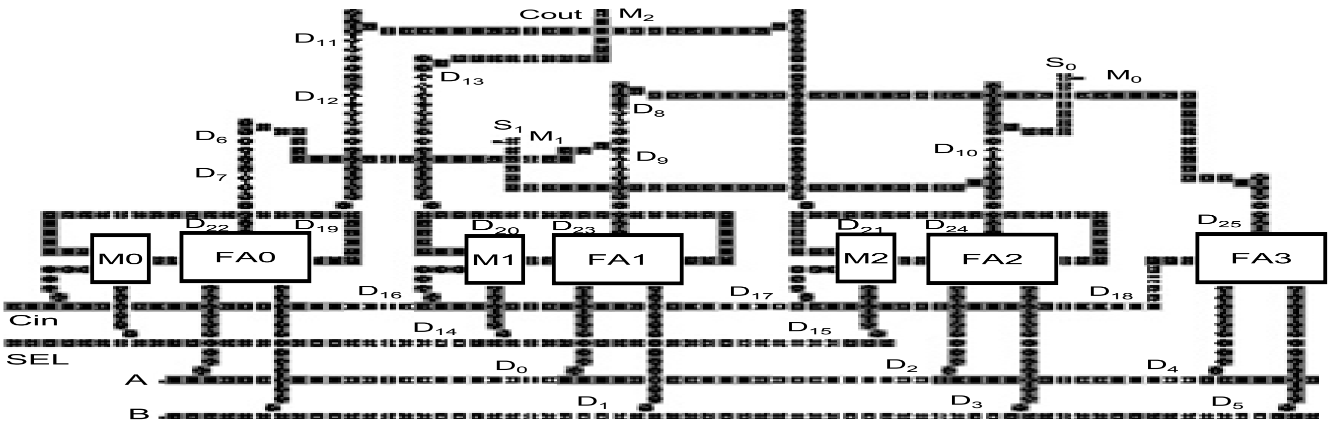


Figure 4 A 2-bit adder using TMRSO

2.1. Design of a 2-bit TMRSO Adder

A 2-bit adder that uses Triple Module Redundancy with Shifted Operands (TMRSO) designed using QCADesigner [6] is shown in Figure 4. This is an adaptation of Patel and Fung's [7] logic level time redundancy-based Concurrent Error Detection (CED) technique for permanent faults, called RE-computing with Shifted Operands (RESO). If an ALU performs a function **f**, and **x** is an input to the function, then an error in the ALU can be detected by comparing **f(x)** with the output of **right_shift (f (left_shift (x))**. For a given data input, the result of function **f** is stored in a register. This is then compared with the result obtained using shifted operands. Though this technique is a time redundancy based technique, it requires a shift-register to buffer the input and a register to buffer the output. As we have outlined in Section II, an array of QCA cells with proper clocking works as a D-latch. Similarly, a series of D-latches implements a shift-register and can be used to shift the operands and results.

2.2. Step by step operation of the adder

The TMRSO adder shown in Figure 4 operates on two inputs **A** (A_1A_0) and **B** (B_1B_0) as follows:

Clock cycle 1: Input bits $\mathbf{A_0}$, $\mathbf{B_0}$ and **Cin** are pumped down the input wire and are latched in **D0**, **D1** and **D16** respectively. **SEL** is reset to feed the carry bit **Cin** into **FA0**. At the end of this cycle $\mathbf{Cout_0}$ and $\mathbf{Sum_0}$ are latched at **D19** and **D22**, respectively.

Clock cycle 2: $\mathbf{A_0}$, $\mathbf{B_0}$ and **Cin** are shifted down to **D2**, **D3** and **D17** while $\mathbf{A_1}$ and $\mathbf{B_1}$ are fed in and latched at **D0** and **D1**. The $\mathbf{Sum_0}$ and $\mathbf{Cout_0}$ will be calculated once more on **FA1**. $\mathbf{Cout_0}$ will be latched at **D20** and $\mathbf{Sum_0}$ is ready at **FA1**'s output. Meanwhile, $\mathbf{Sum_1}$ and $\mathbf{Cout_1}$ can be calculated on **FA0** using bits $\mathbf{A_1}$, $\mathbf{B_1}$ and $\mathbf{Cout_0}$. At the end of this cycle, the first set of results ($\mathbf{Sum_1}$ $\mathbf{Sum_0}$ **Cout**) are available at **D22**, **D23** and **D19**, respectively.

Clock cycle 3: $\mathbf{A_1}$, $\mathbf{B_1}$, $\mathbf{A_0}$, $\mathbf{B_0}$ and **Cin** are shifted down the wire and are available at the inputs of **FA1** and **FA2**. **SEL** is kept at '**1**' in this cycle so that feedback carry bits are fed to **FA1** and **FA0**. $\mathbf{A_0{+}B_0{+}Cin}$ and $\mathbf{A_1{+}B_1{+}Cout_0}$ are computed at the same time on **FA2** and **FA1**, respectively. At the end of this cycle, the second set of results ($\mathbf{Sum_1}$ $\mathbf{Sum_0}$ **Cout**) are available at **D23**, **D24**, and **D20** respectively, while the first set of results are shifted to **D7**, **D9** and **D12** accordingly.

Clock cycle 4: SEL retains its value of '**1**'. The operands $\mathbf{A_0}$, $\mathbf{B_0}$, $\mathbf{A_1}$ and $\mathbf{B_1}$ continue their propagation and flow into **FA3** and **FA2,** respectively. At the end of this clock cycle the third set of results is available at **D24**, **D25**, and **D21** respectively. Three sets of results will vote through majority gates and the final result will be sent out.

IV. TMRSO VS TMR

1. Fault tolerance capability

1.1. Defects in the input lines

The input lines of TMR and the proposed TMRSO technique are shared by all the copies. A failure in the lines may simultaneously affect two or all copies of computations and results in a faulty output. Since QCA arrays constructed by 90-degree cells show strong tolerance to missing cell and cell misalignment, we suggest that the input lines should be constructed by 90-degree cells. However, such defects introduce additional delay. If multiple defects exist in a wire where all the cells must be traversed by information flow in one clock cycle, which is the case for basic TMR, the delays caused by these defects will pile up and adversely impact performance. If multiple defects are interlaced by D-latches, which is the case for TMRSO, the delays will be isolated and distributed among copies of one-bit full adders. As a result, the effect on performance is minimized.

1.2. Defects in individual full adders

The basic idea behind the fault tolerance using TMR and TMRSO is that every bit addition will be performed on three different full adders. If one fails, the final results will still be correct due to the remaining two copies. If two or all of them fail, the final result will be incorrect. Therefore, both techniques can tolerate a single defective full adder. However, in the presence of multiple defective full adders neither of the techniques can guarantee correct results. In this case, additional redundancy is needed. While the basic N-Module Redundancy technique needs N times the area overhead and incurs an N-fold performance degradation, the proposed technique can support N-Modular Redundancy with only little increase in area overhead and minor degradation in throughput. For example, 5-Modular redundancy using shifted operands needs {n+4} full adders and {n+5} clock cycles for the computation of n bit width while the clock period remains the same.

1.3. Defects in the output majority gate

Both techniques use majority gates to vote the three results. If the majority gate fails, the result will be incorrect no matter how much redundancy is used. Therefore, both techniques require fault tolerant majority gates one of which can be found in [10].

2. *Throughput*

TMRSO successfully overcomes the shortcomings of a multiple-bit TMR adder. For an n-bit TMR adder, the maximum number of cells that the information needs to flow in one clock cycle equals 3×n×18 while for the adder using TMRSO, it is only 19 – the width of a full adder used in the design. Assuming the period of the clock is proportional to the cell delay, the clock period is 3×n×18 cell delays for TMR and 19 cell delays for TMRSO. On the other hand, an n-bit TMR adder can process an n-bit addition in one clock cycle while TMRSO needs n+3 clock cycles (n cycles for loading n-bit inputs, + 3 for computations). A detailed summary is given in Table 1. The final throughput is calculated as the number of bits divided by the product of the number of clock cycles and clock period in terms of cell delay. According to the table, the throughput of the proposed technique exceeds the throughput of TMR when the bit width of operands exceeds 2. For an 8-bit adder, the throughput improvement is more than 100%.

Table 1 Throughput comparison of TMRSO and TMR

	TMR	TMRSO
Clock period in terms of the length of the longest wire (cell delay)	3×n×18	19
Number of cycles for n-bit addition	1	n+3
Throughput (bits/cell delay)	1/54	n/(19×n+57)

3. *Area overhead*

TMRSO uses significantly less area when compared to the basic TMR. An n-bit adder using TMR has 3×n one-bit full adders while an n-bit TMRSO adder consists of (n+2) one-bit full adders with feedback and a multiplexer. Using the QCAdesigner [6], we have created a one-bit full adder and a one-bit full adder with feedback and a multiplexer. The dimensions of the one-bit full adder are 18×22 and those of the one-bit full adder with feedback and a multiplexer are 19×38 in terms of QCA cells. The full adder with feedback and a multiplexer is 1.8 times larger than a one-bit full adder. However, for an n-bit adder using TMR, each copy will have a feedback to its input, hence necessitating a multiplexer. In other words, the bounding block of the TMR adder will increase in height by the length of a multiplexer.

The relative area overhead of a TMRSO adder and a TMR adder depends on the ratio of the width of two designs. Table 2 shows a detailed comparison between TMRSO and TMR using n-bit adders. The width of an n-bit adder using TMR is 3×n×18 cells while the width of an n-bit adder using TMRSO is (n+2)×19 cells. The height of the TMRSO adder is slightly larger than the height of the TMR adder. Therefore, the overall size of the bounding box is 3618×n for the basic TMR and 1482×n + 2964 for TMRSO. As long as n >1, the area overhead of the TMRSO adder will be less than that of the TMR adder. For an 8-bit adder, the area of TMRSO is less than *half* of that of TMR.

Table 2 Area overhead comparison of TMRSO and TMR

n-bit addition	TMR	TMRSO
Full-adder in cells	18	19
Number of full adders	3×n	n+2
overall design width	3×n×18	(n+2)×19
overall design height	67	78
overall bounding box size	3618×n	1482×n + 2964

V. Conclusions

In this paper we report a novel fault tolerance technique for QCA arithmetic circuits called Triple Modular Redundancy with Shifted Operands (TMRSO). In QCA nanotechnology wire delays dominate logic delays and faults in wires dominate the faults in a design. Based on this observation, TMRSO uses shorter wires and exploits the self-latching property of QCA wires by considering delays and faults in QCA wires as well. TMRSO is applicable to arithmetic building blocks such as adders, subtractors, shifters, multipliers etc. We have validated TMRSO using an adder. The implementation results show that TMRSO exhibits superior throughput and area overhead characteristics while still maintaining the identical level of fault tolerance capability as a straightforward TMR technique.

VI. References

[1] C.S. Lent, P.D. Tougaw, W. Porod, G.H. Bernstein, "Quantum cellular automata," Nanotechnology, vol. 4, pp. 49-57, 1993.

[2] P.D. Tougaw, C.S. Lent, "Logical devices implemented using quantum cellular automata", Journal of Applied Physics, vol. 75(3), pp. 1818-1825, February 1,1994.

[3] A. Vetteth, K. Walus, V.S. Dimitrov, G.A. Jullien, "Quantum-dot cellular automata carry-look-ahead adder and barrel shifter", IEEE Emerging Telecommunications Technologies Conference, 2-I-4 (5 pages), Dallas, TX, Sept. 2002.

[4] W. Wang, K. Walus, G.A. Jullien, "Quantum-Dot Cellular Automata Adders", IEEE Nano 2003 Conference, pp. 461-464,San Francisco, CA 2003.

[5] M.B. Tahoori, M. Momenzadeh, J. Huang, F. Lombardi, "Defects and Faults in Quantum Cellular Automata at Nano Scale", VLSI Test Symposium, p.291, 2004.

[6] QCADesigner, http://www.atips.ca/projects/qcadesigner/

[7] J.H. Patel, L.Y. Fung, "Concurrent Error Detection in ALUs by Recomputing with Shifted Operands," IEEE Transactions on Computer, Vol. C.31, No.7, pp. 589 - 595, Jul. 1982.

[8] A. Fijany, N. Toomarian, K. Modarress, M. Spotnitz, "Bit-serial Adder Based on Quantum Dots", NASA technical report, NPO-20869, Jan. 2003.

[9] M.B. Tahoori, F. Lombardi, "Testing of Quantum Dot Cellular Automata Based Designs", Design Automation and Test in Europe Conference, pp.1408-1409, 2004.

[10] A. Fijany, B.N. Toomarian, "New Design for Quantum Dots Cellular Automata to Obtain Fault Tolerant Logic Gates", Journal of Nanoparticle Research, vol. 3, pp. 27-37, Feb. 2001.

[11] B.W. Johnson, **Design and Analysis of Fault-Tolerant Digital Systems**, Addison-Wesley Publishing Company, 1989.

[12] Craig S. Lent, P. Douglas Tougaw, "A Device Architecture for Computing with Quantum Dots", Proceedings of The IEEE, Vol. 85, NO.4, pp541-557, April 1997