# General Floorplan for Reversible Quantum-dot Cellular Automata

Sarah E. Frost-Murphy
University of Notre Dame and
Sandia National Laboratories*
Dept. of Computer Science
and Engineering 384
Fitzpatrick Hall
Notre Dame, IN 46556
smurphy@cse.nd.edu

Erik P. DeBenedictis
Sandia National Laboratories
Scalable Computing Systems
PO BOX 5800, MS-1322
Albuquerque, NM 87185-1322
epdeben@sandia.gov

Peter M. Kogge
University of Notre Dame
Dept. of Computer Science
and Engineering
384 Fitzpatrick Hall
Notre Dame, IN 46556
kogge@cse.nd.edu

## ABSTRACT

This paper presents the Collapsed Bennett Layout, a general purpose floorplan for reversible quantum-dot cellular automata (QCA) circuits. In order to exploit the full density and speed potential of emerging nanodevices, the principles of reversible computing need to be incorporated into the design of nanoscale circuits and systems. The Collapsed Bennett Layout implements Bennett's algorithm in hardware, allowing any arbitrary logic function to be implemented reversibly in QCA.

## Categories and Subject Descriptors

C.1 [**Processor Architectures**]: Other Architecture Styles

## General Terms

Design, Theory

## Keywords

quantum-dot cellular automata, reversible computing

## 1. INTRODUCTION

Without incorporating the principles of reversible computing into nanoscale designs, the density and speed gains promised by emerging nanotechnologies may remain beyond reach because of the dissipation associated with the erasure of a bit. This work introduces a general purpose floorplan for reversible quantum-dot cellular automata circuits, the Collapsed Bennett Layout. Reminscient of Bennett's alogorithm, this layout allows any irreversible QCA circuit to be executed reversibly. This work begins by briefly introducing the quantum-dot cellular automata (QCA) paradigm, pointing out the magnitude of the power problem for nanoscale systems, and discussing reversible computing as a way to address the problem. Bennett's algorithm is then briefly discussed followed by the presentation of the collapsed Bennett layout.

*Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.


*CF'07,* May 7–9, 2007, Ischia, Italy.


## 2. QUANTUM-DOT CELLULAR AUTOMATA

The quantum-dot cellular automata (QCA) paradigm has been widely described. In brief, charge configuration is used to represent information rather than current. Each QCA cell consists of four quantum-dots arranged in a square and two excess electrons that repel each other and can tunnel quantum mechanically between the dots of the cell. Since the electrons repel eachother, there are two stable configurations that correspond to a binary zero and binary one (figure 1). If the QCA cells are arranged in a line, they form a wire. Directionality can be imposed by a localized electric field that controls the tunneling of the electrons. When the field is high, the configuration of the QCA cells are locked. When the field is low, the configuration relaxes and the cell has no value. The basic gates in the QCA paradigm are the majority gate and inverter which form a functionally complete set. The majority gate functions as an AND gate if one of its inputs is a fixed logical zero input and as an OR gate with a fixed one input (figure 2).

Logically irreversible circuits and architectures have been studied including a simple processor [7][8] and memory[4][3]. These works cite many additional papers describing different aspects of the QCA paradigm including physical realizations of the devices. Logically reversible circuits and design techniques have also begun to be explored [5].

## 3. THE POWER PROBLEM FOR NANODEVICES

Landauer's principle, the justification of reversible computing, says that the erasure of a bit leads to at least $kTln(2)$ Joules of heat dissipation, where $k$ is Boltzmann's constant and $T$ is the temperature of the system (300K is room temperature)[6]. If it is true, a processor design that takes advantage of the density and speed potential of nanoscale devices will need to consider reversibility so the heating due to the erasure of bits does not overwhelm the system. For instance, consider the following scenario. Suppose there is a device (e.g. a transistor, a carbon nanotube switch, xa QCA

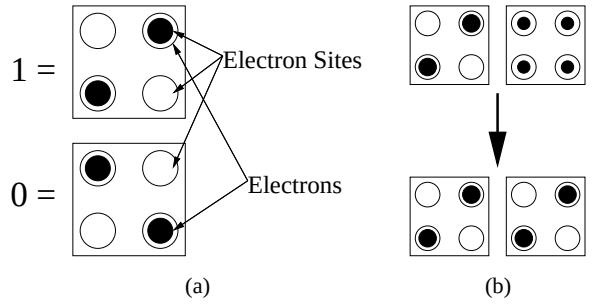


**Figure 1: (a) QCA cells consist of four quantum-dots arranged in a square and two excess electrons that repel eachother forming two stable states. (b) The value of a QCA cell is determined by its neighbor. The cell on the right begins with its electrons delocalized and assumes the configuration of its neighboring cell on the left.**

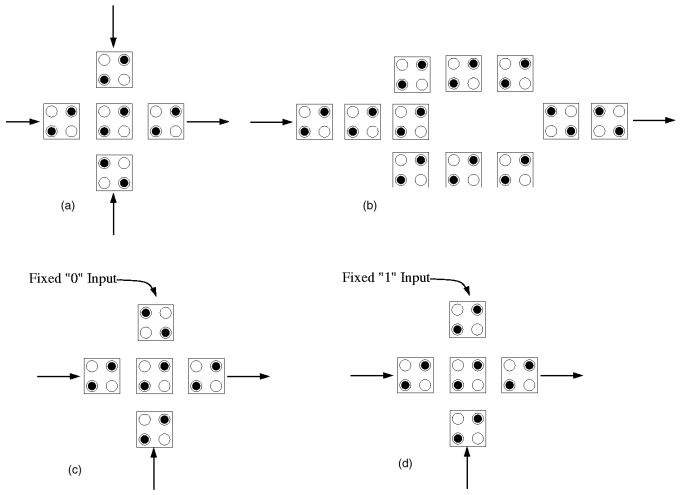


**Figure 2: (a) The fundamental QCA gate is the majority gate with three inputs and one output. The output value is the majority of the inputs, or AB+AC+BC. (b) A QCA inverter. (c) An AND gate is formed if one of the inputs is a fixed logical zero. (D) An OR gate is formed with one input fixed at logical one.**

cell, etc.) that covers an area of 1 $nm^2$ and can be clocked at 1THz. A circuit (e.g. an ALU or FPU) is built out of the device in which just one bit for every 10 devices each cycle is erased (e.g. only one device out of ten is switching each clock cycle), and assume it operates at room temperature. The dissipation due to bit erasures alone would be:

$$1\frac{device}{nm^2} * 10^{14}\frac{nm^2}{cm^2} * \frac{1}{10}\frac{bit}{devices} * \frac{kTln(2)}{bit} * 1THz = 28,696\frac{W}{cm^2} \quad (1)$$

This is a particularly alarming number considering first that this applies for *any* such device and second that the rule of thumb limit for air cooling with heat sinks is 100 $W/cm^2$. Even backing away from this dissipation result by a few orders of magnitude for density inefficiency, slower switching speed, etc., one is still left with a staggering amount of dissipation due to bit erasures alone. This indicates how critical avoiding these bit erasures will be for nanoscale technologies.

## 4. REVERSIBLE COMPUTING

The reversible computing paradigm avoids the erasure of bits and so avoids the dissipation associated with erasure. Landauer proposed the idea of reversible computing in 1961, but believed at the time that reversibility would be unworkable for arbitrary computation [6]. Bennett showed first that arbitrary reversible computation was possible [1] and later that the space overhead of imposing reversibility on an irreversible circuit or algorithm was on the order of the log of the space required for the irreversible version [2].

The key insight of reversible computing is that information does not need to be destroyed during computation. Rather than erasing a value, it can be "decomputed" to return the location storing the information to its previous value. Decomputation is made possible either by saving all the inputs of an irreversible function (e.g. an AND gate) or by using inherently reversible functions (e.g. a Toffoli gate). A reversible function is one which is one-to-one and onto meaning that the function and its inverse both have a unique inverse. In other words, no matter which direction a function is being applied, the previous state can be uniquely identified. A computation involving only inherently reversible operations, then, is really a unique transformation of the input. Further, it has been shown that there are several classes of reversible gates that are universal [9][10].

However, while it has been demonstrated that there are universal reversible gates and functions, there is a long history of irreversible design – both in algorithms and in circuits. Bennett's work shows how to execute any arbitrary algorithm reversibly. This can be naturally extended to circuit design.

## 5. BENNETT'S ALGORITHM

At a high level of abstraction, reversibility can be forced onto any algorithm or circuit by saving the inputs and all intermediate results. However, depending on the function, this can lead to an exponential explosion in the amount of data that needs to be stored. Bennett proposed an algorithm that minimizes the amount of data that needs to be stored at the cost of execution time [2]. Using Bennett's algorithm, any irreversible algorithm can be divided into segments that will be calculated, have the intermediate result latched so it can be used by the next segment, and then when it is no longer needed, decompute the intermediate result so it does not have to be stored. Bennett's algorithm is an optimal ordering for computing and decomputing the segments. While Bennett's original work envisioned these segments as segments of an algorithm, they can also be thought of as segments of logic that implement the algorithm. Figure 3 shows an example of the order of computations and decomputations for an algorithm broken into eight segments.

## 6. THE COLLAPSED BENNETT LAYOUT

DeBenedictis introduced the beginnings of a potentially implementable layout that collapses Bennett's reversible al-

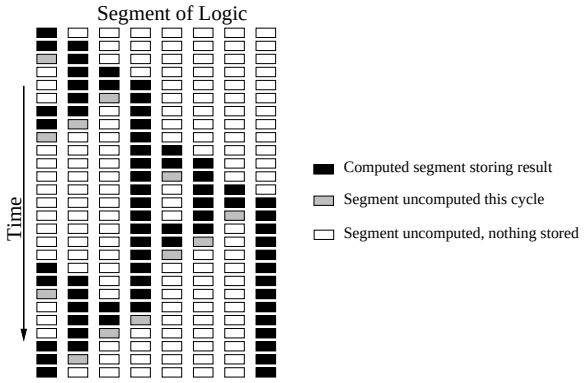


**Figure 3: Bennett's algorithm divides an algorithm or circuit into stages (8 stages in this example) and selectively computes and decomputes them to store the least amount of data necessary to maintain the reversibility of an irreversible algorithm.**

gorithm tree into a single level of re-usable logic with a stack at either end of the combinational logic and a shifting mechanism to pop the top of one stack and push it onto the other stack (i.e. shift left or shift right). This model was further developed and a simple ripple carry adder was designed for it to demonstrate the proposed operation.[5] Rather than using unique circuits to implement each segment of the algorithm or function, this layout assumes that a single segment of logic can be re-used by each segment of the computation. For instance, the logic could be as general as a complete CPU or specialized for a specific function such as a part of an ALU.

A schematic of the components can be seen in figure 4. It consists of a left stack, an area of combinational logic, a shifter unit, a shift-disable area, and a right stack. In addition, the logic unit as a whole can be disabled by adjusting the voltage bias across the area, and separately the shifter area can be similarly disabled.

The final piece of the set-up description is the clocking signal generation. The required signals can be seen in figure 7. One of the modifications to traditional QCA design is that more than the traditional four clocking signals (switch, hold, release, and relax) may be useful. This floorplan requires $6+n$ clocking signals where $n$ is the number of stages in the logic portion of the layout. The optimal number of stages in the logic portion is dependent on the complexity of the function to be calculated and the cost of additional clocking signals for a particular device implementation.

The collapsed Bennett model operates by combining four "functions": compute, decompute, shift left, and shift right. In compute, the input at the top of the left stack is used to compute a value that is pushed onto the right stack. In decompute, a value is popped from the right stack and decomputed. In shift left, the value is popped from the right stack and pushed onto the left stack. Similarly in shift right, the value is popped from the left stack and pushed onto the right stack. Using this combination of functions, Bennett's algorithm can be implemented using a single copy of the logic portion.

Bennett's algorithm defines optimal time/space tradeoffs. The version of the algorithm presented here minimizes the

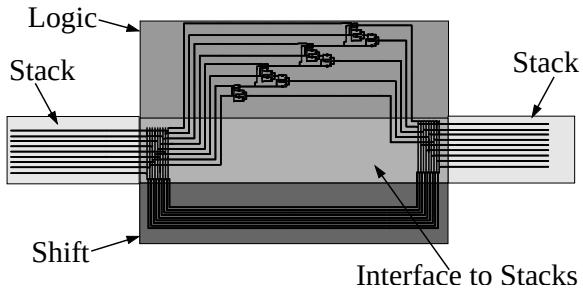


**Figure 4: The regions of the collapsed Bennett layout include two stacks, a logic or computational area, a shift area that allows data to be transferred between the stacks, and an interface between the stacks and the logic and shift regions.**

**Table 1: Execution of a Four Segment Problem by Bennett's Algorithm**

| Step | Bennett's Algorithm |
|---|---|
| 1. | Compute 1 |
| 2. | Compute 2 |
| 3. | Decompute 1 |
| 4. | Compute 3 |
| 5. | Compute 4 |
| 6. | Decompute 3 |
| 7. | Compute 1 |
| 8. | Decompute 2 |
| 9. | Decompute 1 |

space required. However, at the expense of using more space, the time component could be minimized. This tradeoff is an important one for the collapsed Bennett layout because the size of the stacks are determined by the time/space tradeoff made for the particular function being computed. An algorithm that requires $O(T)$ time and $O(S)$ space to run irreversibly can run in $O(T^{1+\epsilon})$ time and $O(SlogT)$ space.

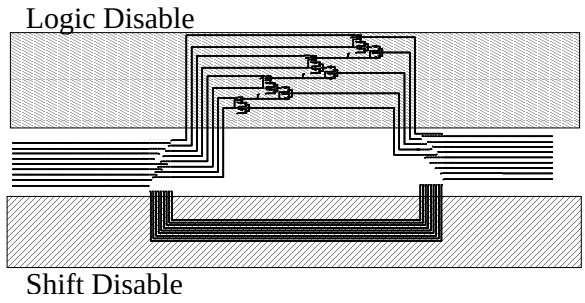


**Figure 5: There are two disable sections in this layout. The top area disables the logic, while the bottom area disables the shift. While disabled, the QDCA cells have no value and do not contribute to the computation of any nearby cells.**

A simple adder is shown in this section to illustrate the layout and the interfaces between the computation and storage. One can imagine sandwiching an entire processor be-

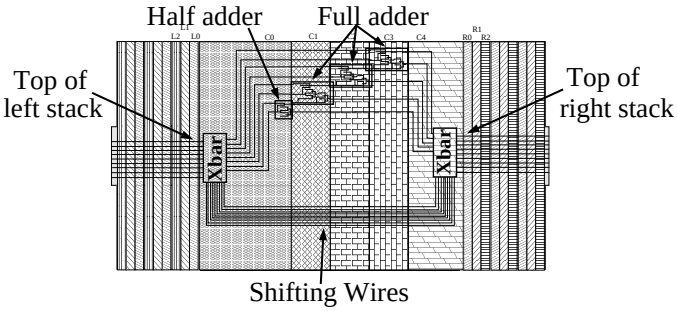


**Figure 6: Columnar clocking zones defined for this adder. The shadings correspond to the shadings in figure 7 showing the signal generation needed for the four modes of operation.**

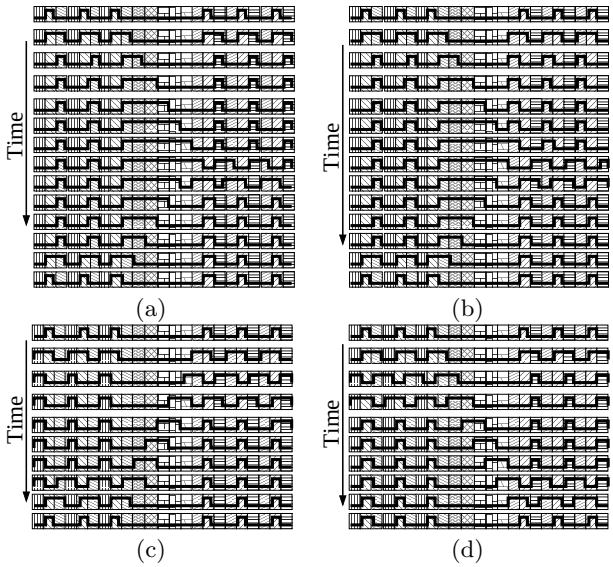


**Figure 7: Clocking signals required for four modes of operation of collapsed Bennett clocking layout: (a) Compute, (b) Decompute, (c) Shift Left, (d) Shift Right. The clocking regions correspond to the coloring in figure 6**

tween two stacks in this manner for a general purpose reversible processor.

For instance, consider a very simple example of using an adder such as that in figure 4 to multiply $1*4$ by iteratively adding 1 to itself 4 times. Table 1 shows the operations associated with the execution of a four segment problem by Bennett's algorithm. Table 2 steps through the execution and shows the corresponding execution in the collapsed Bennett layout. Since one of the operands is always 1 in this example only the changing operand is shown at the top of the stack to aid in clarity. The numbering of the steps is the numbering used in table 1. Notice that each Bennett's algorithm step is associated with two collapsed Bennett operations - a shift left/right operation followed by a compute/decompute operation. At the end of execution, the original input remains at the top of the left stack, and the final output is

**Table 2: Executing a Simple Example in the Collapsed Bennett Layout**

| Step | Operation | Left Stack | Right Stack |
|---|---|---|---|
| 1. | Enter Initial Input: 0 | 0 | - |
| | Compute 1+0 | 0 | 1 |
| 2. | Shift Left | 1<br>0 | - |
| | Compute 1+1 | 1<br>0 | 2 |
| 3. | Shift Right | 0 | 1<br>2 |
| | Decompute 1+0 | 0 | 2 |
| 4. | Shift Left | 2<br>0 | - |
| | Compute 1+2 | 2<br>0 | 3 |
| 5. | Shift Left | 3<br>2<br>0 | - |
| | Compute 1+3 | 3<br>2<br>0 | 4 |
| 6. | Shift Right | 2<br>0 | 3<br>4 |
| | Decompute 1+2 | 2<br>0 | 4 |
| 7. | Shift Right | 0 | 2<br>4 |
| | Compute 1+0 | 0 | 1<br>2<br>4 |
| 8. | Shift Left | 1<br>0 | 2<br>4 |
| | Decompute 1+1 | 1<br>0 | 4 |
| 9. | Shift Right | 0 | 1<br>4 |
| | Decompute 1+0 | 0 | 4 |

located at the top of the right stack. Clearly, this is a very simple example, but it illustrates the relationship between Bennett's original algorithm and the operation of the collapsed Bennett layout.

## 7. CONCLUSION

This work reminds readers of the importance of reversible computing for nanoscale architectures that aim to exploit the speed and density potential of emerging devices. In order to fully harness this potential, reversibility must be explicitly incorporated into designs. This paper introduces a general floorplan to incorporate reversibility into any design for one such high-performance device, quantum-dot cellular automata.

## 8. REFERENCES

[1] C. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, pages 525–532, November 1973.

[2] C. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.

[3] S. Frost. Memory architecture for quantum-dot cellular automata. Master's thesis, University of Notre Dame, March 2005.

[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. In *1st Workshop on Non-Silicon Computation (NSC-1), held in conjunction with 8th Int. Symp. on High Performance Computer Architecture (HPCA-8), Boston, MS*, Feb 3 2002.

[5] S. Frost-Murphy, M. Ottavi, M. Frank, and E. DeBenedictis. On the design of reversible qdca systems. Technical Report SAND2006-5990, Sandia National Laboratories, 2006.

[6] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal*, pages 183–191, July 1961.

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

[8] M. Niemier. *The Effects of a New Technology on the Design, Organization, and Architectures on Computing Systems*. PhD thesis, University of Notre Dame, September 2003.

[9] T. Toffoli. *Cellular Automata Mechanics*. PhD thesis, University of Michigan, 1977.

[10] T. Toffoli. Reversible computing. Technical Report MIT/LCS/TM-151, MIT, 1980.