SANDIA REPORT

SAND2006-5990
Unlimited Release
Printed

On the Design of Reversible QDCA Systems

S.E. Frost-Murphy, M. Ottavi, M.P. Frank, E.P. DeBenedictis

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

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-AC04-94-AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2006-5990
Unlimited Release
Printed

On the Design of Reversible QDCA Systems

S.E. Frost-Murphy
University of Notre Dame
Notre Dame, IN 46556, and
Multiscale Computational Materials Methods Department

M. Ottavi
Scalable Computing Systems Department

M.P. Frank
Florida State University
Tallahassee, FL 32310

E.P. DeBenedictis
Scalable Computing Systems Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185

Abstract

This work is the first to describe how to go about designing a reversible QDCA system. The design space is substantial, and there are many questions that a designer needs to answer before beginning to design. This document begins to explicate the tradeoffs and assumptions that need to be made and offers a range of approaches as starting points and examples. This design guide is an effective tool for aiding designers in creating the best quality QDCA implementation for a system.

Contents

List of Figures

List of Tables

Preface

The purpose of this document is as follows: Assume you are tasked to implement a system, but can choose any design style. You decide to try QDCA to see if it offers advantages over other options. This design guide then becomes effective for creating the best quality QDCA implementation. To figure out if a QDCA implementation is actually better, trial designs of QDCA and alternatives would need to be compared.

There are two independent decisions to be made at the top level: implementation technology and a logic schema that includes reversibility. A third consideration on fault tolerance follows.

Implementation Technology

A handful of QDCA technologies have been proposed, and there are variants within each. For example, there are molecular implementations, "quantum fortress", electrostatically gated quantum dots, metal island, etc. Each of these differ in cell size, operating temperature, manufacturability, and fault tolerance.

Within each of the technology options above, there will be sub-options. For example:

1. How many physical clocks are allowed; clock drivers integrated on chip
2. Clock zones limited to columnar regions, arbitrary patterning on top of chip, or clock patterning top and bottom
3. High or low ratio of dissipation versus bit erasure energy
4. Physical support for crossovers yes/no.

Logic (Reversibility) Schema

Generally, it will be a good idea to have reversibility at the lower levels. Most logic occurs at the lower levels (the "inner loop") so making the lower levels reversible translates most directly to power savings. However, the effort of making logic reversible cascades to higher levels where the benefit is less (i. e. the "outer loop"). A generally reasonable strategy is to draw a line at some logic level, with reversibility employed below the line and not above it.

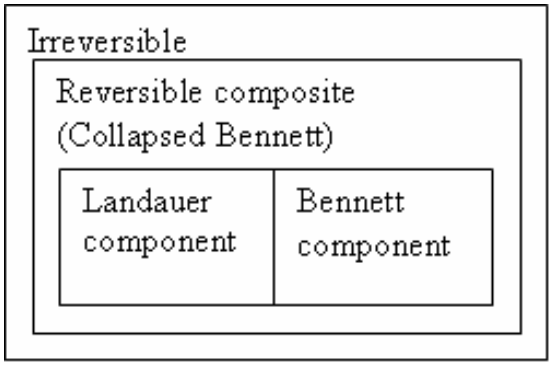


Figure 1. Exemplary Hierarchical Composition of Logic Schema

The reversible logic part of the design will need to comply with some sort of reversibility schema. The Bennett and Landauer clocking schema were proposed in the early papers, but over time it seems that these are just two instances of schema out of a larger space (see figure). QDCA cells obey a fairly simple set of rules about when they will work reversibly, work irreversibly, and don't work at all. Given the technology and sub-options as discussed above, it is feasible to define a low-level clocking structure (of which Landauer and Bennett are instances). It is then possible to hierarchically combine low-level modules into higher-level ones more reminiscent of a classical microprocessor. "Collapsed Bennett" design is an example of a higher-level schema.

Fault Tolerance

It is expected that manufacturing of QDCA systems will need to cope with the occurrence of a high level of defects at manufacturing.

With molecular implementations, QDCA cells each made of two dipoles or dots will be deposited on a patterned substrate. At this level however, new types of defect are likely to occur. Missing or additional cells are inevitable for molecular implementation, because the process of cell deposition is very sensitive a small variation in process parameters may result in a defect.

In order to provide a model for test, the functional effect of these defects will need to be characterized, moreover to increase the production yield some techniques of fault tolerance will need to be provided to the system. Some of the defects that will be enountered are discussed in terms of design rules in chapter two.

Although a detailed analysis of the possible fault tolxxerant techniques is not in the

scope of this report, the the adoption of these techniques will introduce a consequent redundancy in terms of space or time that will need to be considered in the design.

Chapter 1

Introduction to QDCA

The Problem

The goal of computer designers and manufacturers is to produce smaller, faster computers. In 1965, Gordon Moore described the success of the industry in this matter noting that between 1959 and 1965, the number of components on a die grew exponentially [26]. This trend has continued with the number of transistors on a die nearly doubling every 18-24 months. This success has been achieved primarily by shrinking the size of the transistor, aided by the increasing size of the die. For instance, Intel's 4004 released in 1971 was made of 2300, 10 micron transistors on a 12 mm^2 die [41]. In contrast, today's chips contain tens or hundreds of millions of transistors near 0.07 microns on dies on the order of several hundred square millimeters.

However, the current strategy of shrinking the transistors and maintaining the same design paradigm will soon be insufficient to meet physical, economic, and architectural barriers. The smallest transistors in production today operate despite quantum effects. In the near future, the operation of transistors will be dominated by the quantum world. The current device, the CMOS transistor, will need to be replaced by one that embraces these quantum effects and takes advantage of the physics that governs at the nano-scale. Fabrication costs, short lifetime of chip generations, rising capital costs, and demand for computing power from consumers all create economic challenges for the semiconductor industry [39]. Finally, as the gap between processor and memory speeds continues to grow, the von Neumann bottleneck will create a greater and greater architectural barrier to continued performance increases.

These barriers point to the need for a new kind of fundamental device and architecture, such as quantum-dot cellular automata (QDCA). The device characteristics of QDCA, which will be introduced below, are quite different from CMOS characteristics. This changes the cost landscape which in turn changes the look of efficient designs. The design frameworks presented in this document take advantage of the characteristics of QDCA, in particular the natural marriage of QDCA and reversible computing.

Table 1.1. Truth Table of AND Operation

Irreversible AND			Reversible AND				
A	B	A and B	A	B	A	B	A and B
0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	0
1	0	0	1	0	1	0	0
1	1	1	1	1	1	1	1

Brief Introduction to Reversible Computing

Reversible computing builds on a well-established thermodynamics history starting with Maxwell's Demon, Maxwell's 1867 thought experiment that showed that destroying, or erasing, information results in heat dissipation, specifically at least $kt * ln(2)$ where k is Boltzman's constant and T is the temperature of the system.

In traditional computing based on CMOS technology, the energy dissipated by the device and clock independent of the function being performed dominated any energy dissipation due to irreversibility. However, non-traditional technologies such as QDCA offer a new opportunity to experimentally verify the connection between physical devices and information.

The key insight of reversible computing is that information does not need to be destroyed during computation. There is a fundamental connection between logical reversibility and physical reversibility, and if a logically reversible system is implemented by physically reversible devices, there need not be any power dissipation due to information erasure.

To be reversible, a function needs to be one-to-one. Any function can be made to be one-to-one by saving the inputs. For instance, it is clear from examining the truth table of the AND operation (table 1) that AND is naturally irreversible since there are three zeros in the output making it impossible to determine what the inputs were from just the output. However, by copying the inputs to the output the function becomes one-to-one. In this way, any irreversible function can be made to be reversible at the expense of carrying additional information, or garbage data, forward through the computation.

To take full advantage of reversible computing, the physical implementation of the logic must be physically reversible. Traditional CMOS is not physically reversible since V_{dd} is constantly being dumped to ground. In contrast, QDCA has the potential for very low power operation to the point that energy dissipated due to information destruction will be a significant if not dominant factor of the overall heat dissipation of the system. For perhaps the first time, QDCA systems may allow the connection between information destruction and heat generation to be seen and used in a real

system.

A Computer Architect's Introduction to Quantum-dot Cellular Automata

QDCA is a novel alternative to the transistors, silicon, and CMOS paradigm. Rather than using charge movement, current, to propagate signals and perform operations, QDCA uses devices as charge holders, using Coulombic repulsion of electrons as the primary computing force. A QDCA cell consists of four quantum dots arranged in a square with two excess electrons that can occupy the dots. Because the electrons are repelled by each other, they naturally reside in opposite corners. As a result, the cell has two stable states. The first is an electron in the bottom left corner and the top right corner. A cell with this configuration has a polarization of +1 and represents logical "1". The second stable state is an electron in the top left corner and the bottom right corner, a polarization of -1 representing a logical 0 (figure 1.1). The electrons can tunnel between the quantum dots allowing them to change configurations.

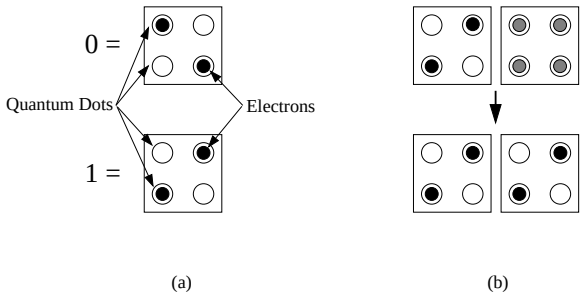


Figure 1.1. QDCA Cell (a) Polarization and corresponding logic values, (b) Signal propagation in QDCA. The cell on the left is polarized, the cell on the right is unpolarized. The cell on the right transitions to assume the polarization of the driving left cell.

Traditionally in QDCA, computation is performed by controlling the tunneling with a four phase "clock" signal (figure 1.3). There are other clocking strategies that will be discussed later in this work in chapter three. Unlike CMOS circuits, the QDCA clock is a fundamentally different phenomenon than the data. The clocking wires generate an electric field that controls the tunnelling of electrons between dots on the QDCA layer (figure 1.2). The clocking field will be generated by fine wires near the QDCA layer. These wires will need to be connected by thicker wires to the signal generators.

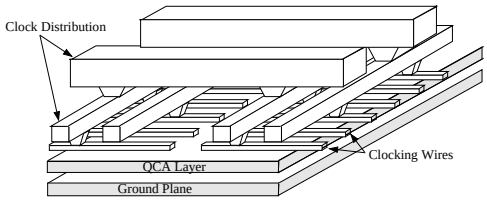


Figure 1.2. Fine wires near the QDCA layer create the clocking fields. Thicker wires distribute the clocking signals to the fine wires.

The clock raises and lowers the barriers between the dots, alternately prohibiting and allowing the electrons to tunnel between dots. The raising and lowering behavior of the clock signal is described by four phases called switch, hold, release, and relax. In the switch phase, the barriers begin low, allowing tunneling, and are raised to prohibit tunneling. In this phase, the cell transitions from having no value to having a definite value. The hold phase follows switch in which the barriers are maintained high, preserving the value assumed during switch. In the release phase, the barriers are falling, allowing the cell to go from a well-defined state to an undefined state in which the cell has no natural polarization. Finally, the relax phase maintains low barriers and no polarization.

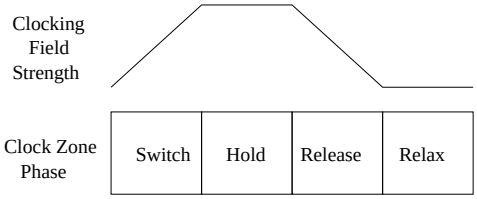


Figure 1.3. Propagation of clock signal in a single cell through time.

If QDCA cells are lined up side by side and clocked appropriately, they act as a wire, propagating a signal down its length (figure 1.4a). Cells laid out in this side by side manner are called 90 degree cells. The alternative is 45 degree cells which are laid out corner to corner (figure 1.4b). In a 45 degree wire, the signal is inverted at each cell. If the first cell holds a "1", the second cell will hold a "0", followed by a "1" in the third cell, and so on.

QDCA cells exist on a single plane. Theoretically, the two types of wires are able to crossover each other in this single plane without effecting the values being transmitted (figure 1.5). This makes complex circuits possible. This strategy will require very precise fabrication techniques. In addition to this physical crossover strategy,

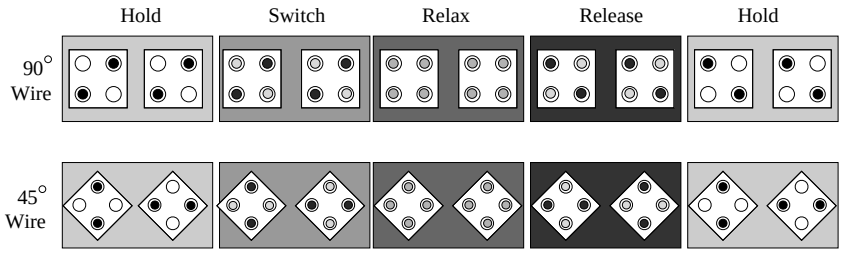


Figure 1.4. Shaded boxes indicate clocking zones. a) 90 degree cells forming a "wire". b) 45 degree cells forming a wire.

temporal and logical crossover strategies have also been proposed.

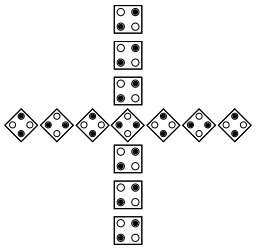


Figure 1.5. Wire crossover.

The basic logic gate in QDCA is the three input majority gate (figure 1.6a). Three input cells are arranged on the edges of a center "device cell." The output of the gate is on the fourth edge of the device cell. The input cells and the device cell share the same clock zone. Because of this and simple coulombic repulsion, the device cell assumes the value of the majority of the inputs. When this device cell is frozen in the hold phase, it drives the output cell which then proceeds as a normal QDCA wire. It is notable that the majority gate is a natural, native device in QDCA. It requires nothing more than the QDCA cells and clocking already introduced. This majority gate can be converted to either an AND gate or an OR gate by fixing one of the inputs to be permanently "0" (figure 1.6c) or "1" (figure 1.6d) respectively.

An inverter is needed for logical completeness, and is formed by taking advantage of the 45 degree interaction (figure 1.6b).

Notice that the majority gate is not natively reversible. However, as discussed above, it can be made to be reversible by saving its inputs. This can be done either by the QDCA circuit or by the clocking strategy. This choice will be further discussed in chapter four.

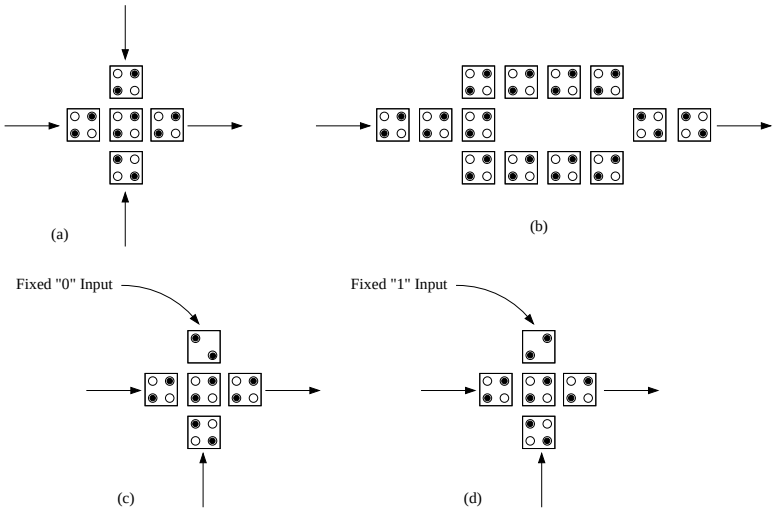


Figure 1.6. a) Three-input Majority Gate, b) Inverter, c) AND gate d) OR gate

In addition to these basic gates, the design landscape also includes three other important features. The first is the inherent latching in wires. In essence, the wires are shift registers. This adds a new dimension to designing QDCA circuits rather than CMOS circuits, allowing a designer to pipeline at a very fine level. Connected to this inherent latching and pipelining, the second feature of QDCA is the close connection between layout and timing [35]. There is an upper and lower bound on the size of clocking zones. Distances and time, then, are very tightly coupled. Finally, bits in QDCA designs are always in motion. The clock and the cells are made of different technologies. Perhaps in the future it may be feasible to have the circuit influence the operation of the clock, but for the design strategies presented in this work, it is assumed that once the clock starts running, it continues to operate independently of the information in the circuit. This, too, changes the design decisions made. These aspects of design will also be discussed somewhat further in chapter five

Prior QDCA Architecture Work

Computer engineering QDCA research first focused on device basic logical devices and an adder as an example of a QDCA circuit [44]. Niemier's work was the first look at the effect QDCA has on architecture and system design. His initial work focused on the hand designing of a simple but complete processor in QDCA much as the first Intel 8086 processor was designed [28][17]. In the course of this work, Niemier identified several key elements of circuit design in QDCA including the connection between layout and timing [35], the potential of processing-in-wire and fine-grained

pipelining [32] [31] [34] [30], and initial floorplanning for logic [28]. In addition, since the first molecular QDCA circuits that will be fabricated will need to be regular structures, the design of implementable FPGAs was explored [36] [33]. Another key work explored the layout parameters and layout rules that will govern the layout of QDCA circuits [29].

Memory systems have also been explored. Frost designed a very dense, finely pipelined memory to work in conjunction with Neimier's processor [13][11]. A novel execution model, the bouncing threads execution model, was explored in conjunction with the H-memory model [14]. Memory cells have been explored by Ottavi [38], Walus [45], and earlier by Fountain and Berzon [4].

Research is also being pursued to build fault models for QDCA circuits in order to build fault tolerant circuits and to build CAD tools to facilitate testing and design of circuits [10]. In addition, the first algorithm that addresses the circuit partitioning problem in QDCA has been developed [1].

The Real Device

QDCA is very real. QDCA cells have been fabricated and their operation experimentally verified [3] [37]. These QDCA cells were constructed with metal dots on a micron scale and operate at 70 mK. As the size of the cell grows smaller, the operational temperature will rise [23]. A molecular implementation, then, would allow room temperature operation as well as offering significant potential density gains in circuits. Lieberman, *et al* have investigated several two dot-molecules such as the Creutz-Taube ion and mixed-valence ruthenium dimers. In addition, they have explored options for attaching these molecules to etched self-assembled monolayers [25]. Other groups at Notre Dame are investigating four-dot molecules[24] and alternate fabrication strategies such as DNA tiling.

In addition to the QDCA cells, a functioning QDCA circuit requires a clock signal and input/output capabilities. Lent, *et al* have designed an implementable clocking scheme in which buried metal wires are used to create the clocking field [15]. Bernstein, *et al* are investigating mechanisms for detecting the output of QDCA circuits. The output of the metal-dot systems were detected using single electron transistor electrometers [3] [25].

Current estimates place fabrication of simple molecular circuits being possible within three to five years. More complex circuits and large scale fabrication will require more time, but are expected to be possible before the end of the roadmap is reached and nanoscale devices are required to meet density, speed, power and performance demands.

Original Contributions

This work is the first attempt to describe how to go about designing a reversible QDCA system. The design space is substantial, and there are many questions that a designer needs to answer before beginning to design. This document begins to make the tradeoffs and assumptions that need to be made explicit and offer a range of approaches as starting points and examples.

Organization

This design guide is organized in a roughly bottom up fashion. Chapter 2 discusses some physical properties of the device that designers should be aware of. Chapter 3 discusses ways in which the clocking field can be organized. CHpater 4 discusses the circuit implementation of the clocking signal. Chapter 5 explores circuit design QDCA, and Chapter 6 discusses some architectural approaches to designing QDCA systems.

Chapter 2

Physical Properties

There is a subtantial body of literature describing the QDCA device. There are a few low-level properties in particular that designers should be cognizant of. These properties include kink energy, gain, and what parameters should be considered for Conway-Mead type design rules for QDCA. In addition, in considering the design of reversible systems, it may be worthwhile to examine adiabaticity in the context of the clocking wires.

Kink Energy

The kink energy of a system composed of two QDCA cells is defined as by the Coulombic energy of the two cells in opposite polarization (as shown in Figure2.1) with respect to the energy of the cells being in the same polarization status.

The Coulombic energy between two quantum dots is:

$$E_{i,j} = \frac{1}{4\pi\epsilon_0\epsilon_r}\frac{q_i q_j}{d_{ij}}$$

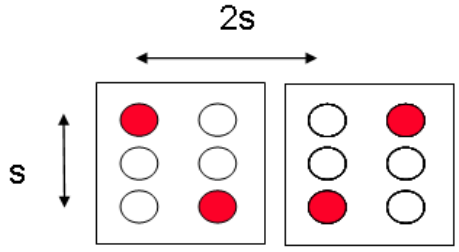


Figure 2.1. Considered sizes of Cell

Figure 2.2. Kink Energy: (a) Ground State (b) Excited state

The energy between two neighboring cells indexed (1) and (2) is therefore:

$$E_{1,2} = \sum_{i=1}^{4}\sum_{j=1}^{4} \frac{1}{4\pi\epsilon_0\epsilon_r}\frac{q_{1i}q_{2j}}{d_{ij}}$$

Where the contributions internal to each single cells have been neglected as they amount to the same value in any cell.

Therefore the Kink energy among the two cells is the difference between the energies of the configurations (a) and (b) shown in figure 2.2.

The energy of configuration (a) is the ground state energy and can be written as:

$$E_a = \frac{1}{4\pi\epsilon_0\epsilon_r}q^2\left(\frac{1}{2s}+\frac{1}{2s}+\frac{1}{\sqrt{(3s)^2+s^2}}+\frac{1}{\sqrt{s^2+s^2}}\right)$$

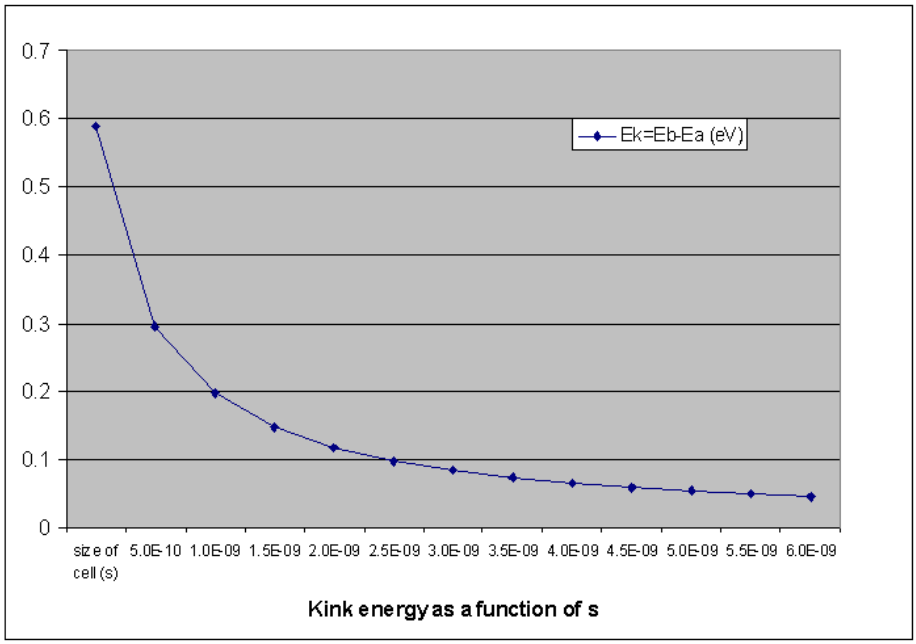
The energy of configuration (b) is the energy of the first excited state and can be written as:

$$E_b = \frac{1}{4\pi\epsilon_0\epsilon_r}q^2\left(\frac{1}{3s}+\frac{1}{s}+\frac{2}{\sqrt{(2s)^2+s^2}}\right)$$

The Kink energy is $E_k = E_b - E_a$ and it is therefore a function of the dielectric constant between the dots, the size of the cell and the charge on each dot $E_k = f(\epsilon_r, s, q)$. Assuming $q = 1$ and $\epsilon_r = 1$ (vacuum) the E_k is plotted in Figure 2 as a function of s

Gain

Gain is critical for any electronic device to guarantee signal restoration. The theory of ower gain in clocked QDCA was explored by Timler and Lent [43], and gain was



experimentally demonstrated by Kummamuru, *et.al.* [19]. The details of the calculations and experiment will be left for the reader to explore. It is very significant that QDCA demonstrates gain, though, because many emerging devices lack gain and therefore have a very limited range of circuit design techniques and architectures. QDCA, though, does demonstrate gain and has a wide variety of circuit and architectural opportunities.

Parameters for Conway-Mead Type Design Rules

One of the challenges when working with any emerging device is that there is not a set of well ununderstood abstractions, or interfaces between device physicists, circuit designers, and computer architects. For instance, Conway and Mead's work in defining design rules for MOS in terms of a fundamental unit λ truly revolutionized the design of circuits and computers.

For emerging devices, it is an ongoing challenge to try to define design rules that capture everything the circuit designer needs to know about the underlying structure to guarantee a circuit that works as expected without muddying the waters with too much information about the details of the device physics. Niemier took the first steps toward design rules for molecular QDCA by identifying possible sources of error in QDCA circuit fabrication and proposing the beginnings of design rules for QDCA [29]. The discussion that follows is a summary of his work.

For Mead and Conway, the sources of error avoided by the design rules are related to the resolution of the process fabricating the circuit. The Mead-Conway rules use

the unit length λ to minimize the impact of fabrication errors such as over etching, misalignment of mask levels, distortion of the silican wafer due to high temperature processing, and over or under exposure of resist. The Mead-Conway rules abstract away from these sources of error to communicate to the circuit designer the information needed to design circuits that will most likely function as expected. The abstraction covers information such as the minimum width of a wire to guarantee current flow, the minimum wire spacing to separate information, and the overlap between layers to assure good contact.

Similarly, the possible errors result from improper electrostatic interactions that effect of which scale with differently with distance d. These include charge-charge interactions ($1/d$), charge-dipole interactions ($1/d^2$), dipole-dipole interactions ($1/d^3$), charge induced dipole interactions ($1/d^4$), dipole induced dipole interactions ($1/d^5$), dispersion effects ($1/d^6$), and van der Waals effects ($1/d^{12}$). These errors occur due to cells not attaching to the substrate, improper distance between cells, misalignment of cells, improper cell rotation, and cells of differing heights. Analogous to the Mead-Conway rules, the QDCA rules abstract these errors to five broad categories. The rules seek to define spacing to guarantee information transfer, spacing to guarantee information isolation, spacing and rotation rules to guarantee proper gate operation, spacing and rotation rules for physical crossovers, and a set of rules to guarantee the clocking field behavior.

Specifically, Niemier's design rules are:

- Cell Spacing

  1A: Maximum allowed spacing between cells that will transmit data. Cells that are too far apart will not have the required coulombic interaction to reliably assume the same configuration.

  1B: Minimum distance cells can be apart and still transmit data. Depending on the particular implementation, cells may need to be a minimum distance apart to transmit data. For instance, if cells are too close, electrons may tunnel between cells rather than within a cell.

- Wire

  2A: Wire lengths with no disorder. Even in a perfectly fabricated wire, there will be a limit on the length of a wire in a zone controlled by a single clocking signal. This will depend on the kink energy which will depend on the particular QDCA implementation.

  2B: Wire lengths and disordered wires. The kink energy decreases with disorder. Different types of disorder with be possible with different implementations and substrates.

  2C: Distance between two parallel wires. To avoid the analogous error to crosstalk, wires will need to be a minimum distance apart.

2D: Incomplete wires. This can be thought of as a special case of rule 1A, the maximum spacing between cells.

- Crossovers

  3: Crossover (governed by 1A and 2B). The physical crossing of 45 degree wires and 90 degree wires will be goverened partly by rules 1A and 2B, but the set of rotations allowed and required will be more specific than the general 1A and 2B cases.

- Majority Gate

  4: Majority gate (governed by 1A and 2B). Similar to the crossover case above, the majority gate can be thought of as a special case of rules 1A and 2B. However, there is the added complication of the interaction of the device cell with the inputs.

- Rippers

  5: Ripper (governed by 1A, 1B, and 2B). A value can be "ripped" off a 45 degree wire bt a 90 degree cell (the opposite interaction of the crossover). This, too, is a special case of rules 1A, 1B, and 2B.

- Clock Related

  6A: Move charge in clocking wires adiabatically. In addition to standard VLSI rules, to maintain low power operation, the clocking signals should be generated adiabatically. This will influence the types of circuits that generate and distribute the clocking signals.

  6B: Voltage needed across QDCA layer to properly clock it. The QDCA implementation will define the voltage that needs to be felt across the QDCA layer. The thickness and makeup of the substrate will then dictate what sort of voltage the clocking wires need to generate.

  6C: Clocking wire placement error. The clocking wires need to be fabricated in a normal process that will have fabrication errors as discussed by Mead and Conway. These need to be taken into account when designing the entire QDCA system.

  6D*: Clock signal phase error. This rule was identified by Ottavi. There may be error in the generation of the clocking signal so the relative phases of the clocking signals would be off. This effect needs to be accounted for as well.

- Misc. Other Rules

  M1: Switching and discharge times of the clock. The clock may have very different switching times than the QDCA cells. This may result in the degradation or disruption of data on the QDCA layer.

  M2: The y-component of a generated electric field. This is somewhat related to rule 6B, but it also encompasses the shape of the electric field being generated and the appropriate distance of the wire from the QDCA layer.

Table 2.1. Niemier's Molecular QDCA Design Rules

Label	Rule
1A	maximum allowed spacing between cells to transmit data
1B	minimum allowed spacing between cells to transmit data
2A	wire lengths with no disorder
2B	wire lengths with disordered wires
2C	distance between two parallel wires
2D	incomplete wires
3	crossover (governed by 1A and 2B)
4	majority gate (governed by 1A and 2B)
5	rippers (governed by 1A, 1B, and 2B)
6A	move charge in clocking wires adiabatically
6B	voltage needed across QDCA layer to properly clock it
6C	clocking wire placement error
M1	switching and discharge times of the clock
M2	y-component of a generated electric field
M3	power dissipation
M4	multiple clock systems

M3: Power dissipation. It is unclear what design rules would be required or helpful to handle power dissipation issues. However, this will be an important consideration for any QDCA system.

M4: Multiple clocks in a system. Different regions of the QDCA circuit may be clocked by completely different clocking circuits. These regions will still need to be able to communicate with eachother.

Niemier's work was the first attempt to define design rules for any emerging nanotechnology. His work began to define what kinds of rules are needed for QDCA. There is still more work to be done to define the rules and develop some sort of QDCA equivalent to λ. This discussion should communicate a sense for what sorts of challenges are important at the fabrication level and how they will effect the circuit and system level.

Adiabaticity: A Case Study

The clocking system will be explored in depth in chapters three and four. However, it may be useful to consider the details of adiabaticity using the clocking system as the example.

Consider an ideal sinusoidal voltage source

$$v_{\mathrm{s}} = V \sin(\omega t), \tag{2.1}$$

where V is the voltage amplitude and $\omega = 2\pi f$ is the angular frequency.

Let us now consider, in a simple lumped-element model, the effect when this source is connected to a load with a capacitance of C through a path with series resistance of R. Let i, v, q be the instantaneous current towards the load through the resistor, the instantaneous voltage across the capacitor, and the instantaneous charge stored on the capacitor, respectively. These are all functions of t. We'll assume that $v(0) = 0$ (the capacitor is initially discharged).

Now, let's build up our model of the circuit dynamics. From the definition of capacitance, we have

$$C = \mathrm{d}q/\mathrm{d}v, \tag{2.2}$$

while from the definition of current and the fact that charge builds up on a capacitor, we have that the instantaneous current is

$$i = \mathrm{d}q/\mathrm{d}t. \tag{2.3}$$

Meanwhile, Ohm's Law gives us that the instantaneous current is also

$$i = (v_{\mathrm{s}} - v)/R. \tag{2.4}$$

Combining eqs. 2.2-2.4 and solving for $\mathrm{d}v/\mathrm{d}t$, we obtain the differential equation:

$$\frac{\mathrm{d}v}{\mathrm{d}t} = (v_{\mathrm{s}} - v)/RC, \tag{2.5}$$

or, writing out v_{s} explicitly,

$$\frac{\mathrm{d}v}{\mathrm{d}t} = (V \sin \omega t - v)/RC. \tag{2.6}$$

The solution to eq. 2.6 (derived in the appendix) is

$$v(t) = \frac{V}{\sqrt{(RC\omega)^2 + 1}} \sin[\omega t - \tan^{-1}(RC\omega)], \tag{2.7}$$

where notice that the signal at the load has been taken down in amplitude by the damping factor $d = \sqrt{(RC\omega)^2 + 1} > 1$ and lags in phase by $\theta = \tan^{-1}(RC\omega)$. Both terms depend on the critical dimensionless parameter $\alpha = RC\omega = t_c/t_r$ where $t_c = RC$ is the time constant (the e-folding time for the exponential decay) for charging the load C through resistance R, while $t_r = t_{cyc}/2\pi$ is the time for the source signal to rotate 1 radian, where $t_{cyc} = 1/f = 2\pi/\omega$ is the clock cycle period. We might call α the "quickness" of the clock oscillation, judged relative to the circuit's natural transition time of t_c.

Now, plugging (2.7) back into (2.5), the voltage drop across the resistor is

$$v_s - v = \frac{V\alpha}{\sqrt{\alpha^2 + 1}} \cos[\omega t - \tan^{-1}\alpha], \tag{2.8}$$

so by (2.4) the current is

$$i = \frac{CV\omega}{\sqrt{\alpha^2 + 1}} \cos[\omega t - \tan^{-1}\alpha]. \tag{2.9}$$

Using $p = iv$, the instantaneous power dissipated in the resistor is then

$$p = \frac{CV^2 RC\omega^2}{\alpha^2 + 1} \cos^2[\omega t - \tan^{-1}\alpha]. \tag{2.10}$$

Over one complete cycle of length $t_{cyc} = 2\pi/\omega$, the energy dissipated is thus

$$\begin{aligned}
E_{cyc} &= \int_{t=0}^{2\pi/\omega} p \, dt, && (2.11)\\
&= \frac{CV^2 RC\omega^2}{\alpha^2 + 1} \int_{t=0}^{2\pi/\omega} \cos^2[\omega t - \tan^{-1}\alpha] \, dt && (2.12)\\
&= \frac{CV^2 RC\omega^2}{\alpha^2 + 1} \int_{\theta=0}^{2\pi} \cos^2\theta \, \frac{d\theta}{\omega} && (2.13)\\
&= CV^2 \frac{\pi\alpha}{\alpha^2 + 1} = \frac{\pi}{\alpha + \alpha^{-1}} CV^2 && (2.14)
\end{aligned}$$

where in (2.13) we have temporarily substituted $\theta = \omega t$ and removed the phase lag, which is irrelevant to the full-cycle integration.

The most important thing to note about eq. 2.14 is its behavior for small quickness $\alpha \to 0$, that is for slow charging, when the radial time $t_r = t_{cyc}/2\pi \gg RC$. Just as with the classical case of adiabatic charging with a linear ramp, note that here too, as the signal rise time increases and the clock frequency decreases, the energy dissipated per cycle decreases roughly proportionately, since $\alpha/(\alpha^2+1) \to \alpha$ as $\alpha \to 0$.

Note also that for very large quickness $\alpha \to \infty$, it is also the case that $E_{cyc} \to 0$, since $\alpha/(\alpha^2+1) \to 1/\alpha$ as $\alpha \to \infty$. However, in this case, the low dissipation can be attributed to the fact that the load voltage does not have time to change very much in a cycle, due to the substantial size of the damping factor $d = \sqrt{\alpha^2+1}$, which approaches α as $\alpha \to \infty$.

The maximum dissipation per cycle is $E_{cyc} = \frac{\pi}{2}CV^2$ which occurs when $\alpha = 1$, that is when $\omega = 1/RC$. This is the case of "least adiabatic" charging, but in fact dissipates only $\pi/8 \approx 40\%$ as much energy as the $4CV^2$ that would be dissipated by a square wave taking the load through the identical range of voltages $[-V, +V]$.

In general, we can characterize the *degree of adiabaticity* of a given process as the ratio between the energy transferred E_{tr} and the energy dissipated E_{diss}. For the charging and discharging of a load between $-V$ and $+V$, the total amount of electrostatic energy moved onto and off of the load is $E_{tr} = \frac{1}{2}C(2V)^2 = 2CV^2$, whereas with a sinusoidal driver we saw that the actual dissipation in a cycle was only $E_{diss} = CV^2\pi\alpha/(\alpha^2+1)$. Therefore, the degree of adiabaticity A of the complete sinusoidal charge/discharge process is

$$A = \frac{2(\alpha^2+1)}{\pi\alpha} = \frac{2}{\pi}(\alpha + \alpha^{-1}), \tag{2.15}$$

which has a minimum of $4/\pi = 1.27$ when $\alpha = 1$. Or, putting things another way, we can define the *energy efficiency* $\eta = 1 - 1/A = (E_{tr} - E_{diss})/E_{tr} = E_{rec}$ which is the ratio between the amount of energy recovered $E_{rec} = E_{tr} - E_{diss}$ and the amount of energy transferred. Phrased this way, the efficiency of the sinusoidal charge/discharge cycle is

$$\eta = 1 - \frac{\pi}{2(\alpha + \alpha^{-1})} \tag{2.16}$$

whose minimum is

$$\eta = 1 - \pi/4 \approx 21.46\% \tag{2.17}$$

when $\alpha = 1$, whereas the efficiency approaches 100% as $\alpha \to 0$, with the distance from 100% in that limit being proportional to α since the expression (2.16) for η approaches $1 - \pi\alpha/2$. Note, in contrast, that the energy efficiency of a standard

abrupt (square wave) charge/discharge process always approaches 0% whenever the load voltage range approaches full-swing, since the energy delivered from the constant-voltage source after the rising edge is CV^2, and exactly this much energy is dissipated upon charging and then discharging the load (half of it or $\frac{1}{2}CV^2$ after each clock edge). Whereas for the adiabatic driver, the case $\alpha \to 0, d \to 1$ where the load voltage range approaches full swing (and also with phase lag θ approaching zero) is also the same limit in which the energy efficiency of the charge transfer approaches 100%.

We can thus see that in all cases, sinusoidal charging dissipates less energy per complete charge-discharge cycle than sharp-edged square-wave charging, dissipating at most about 21% of the energy transferred, and at best nearly 0% when the clock period t_{cyc} is large compared to $2\pi RC$, since in this limit, as $\alpha = RC/t_{\text{cyc}} \to 0$, the fraction of the capacitor charging energy that is actually dissipated on each cycle approaches $\pi\alpha/2$, that is, it goes down in proportion to the quickness of the clock transitions, as would be expected for an asymptotically adiabatic process.

Solution of Differential Equation

This section shows how to solve the differential equation (2.6) from first principles, without delving into formulations in terms of complex impedances which may be non-intuitive for some readers.

We know from our general background knowledge that a circuit composed of linear elements (resistors, capacitors, and inductors) and driven by constant-frequency sinusoidal sources will always attain what is known as an AC steady state. Thus, the solution to (2.6) must also be a sinusoid of constant amplitude, frequency, and phase shift; furthermore, it must have the same average (DC) level as the source, since the resistor cannot support a constant DC voltage drop.

Thus, we know that the solution to (2.6) must be of the form

$$v(t) = V_{\text{L}} \sin(\omega_{\text{L}} + \theta_{\text{L}}) \tag{2.18}$$

where V_{L} is the amplitude of the voltage swing (from $-V_{\text{L}}$ to $+V_{\text{L}}$) of the signal on the load node, ω_{L} is the angular frequency of this signal which we will see must be the same as the driving frequency ω, and θ_{L} is the relative phase of the load.

Starting from (2.18), we can take its derivative

$$\frac{\mathrm{d}v}{\mathrm{d}t} = V_{\text{L}}\omega_{\text{L}} \cos(\omega_{\text{L}}t + \theta_{\text{L}}) \tag{2.19}$$

which we can then plug into the left side of (2.6), along with (2.18) itself in place of

v on the right, to get:

$$V_{\mathrm{L}}\omega_{\mathrm{L}}\cos(\omega_{\mathrm{L}}t+\theta_{\mathrm{L}}) = \frac{V\sin\omega t - V_{\mathrm{L}}\sin(\omega_{\mathrm{L}}+\theta_{\mathrm{L}})}{RC}. \tag{2.20}$$

We now have an ordinary (no longer differential) equation which we need merely solve in order to find the unknown parameters V_{L}, ω_{L}, and θ_{L} as functions of the known parameters V, ω, and RC. Since the equation must hold true for all values of $t \in (-\infty, +\infty)$, there is hope to determine all three unknown parameters using just this single equation.

In what follows, we often substitute $t_{\mathrm{c}} = RC$ for conciseness. Multiplying both sides of (2.20) by t_{c} and gathering all of the terms containing unknowns on the left side of the equation, we get

$$t_{\mathrm{c}}V_{\mathrm{L}}\omega_{\mathrm{L}}\cos(\omega_{\mathrm{L}}t+\theta_{\mathrm{L}}) + V_{\mathrm{L}}\sin(\omega_{\mathrm{L}}t+\theta_{\mathrm{L}}) = V\sin\omega t. \tag{2.21}$$

Factoring out V_{L} from the left side,

$$V_{\mathrm{L}}\left[t_{\mathrm{c}}\omega_{\mathrm{L}}\cos(\omega_{\mathrm{L}}t+\theta_{\mathrm{L}}) + \sin(\omega_{\mathrm{L}}t+\theta_{\mathrm{L}})\right] = V\sin\omega t. \tag{2.22}$$

To match this up with the right-hand side, we would prefer if the left-hand side was expressed as a single sinusoidal function of t. Fortunately, the term in brackets is of the form $a\cos x + \sin x$ which reduces to a single sinusoidal function. In other words, we can always write

$$a\cos x + \sin x = b\sin(x+\phi) \tag{2.23}$$

where b and ϕ are both closed-form functions of a. To see this, note that equation (2.23) is merely the real part of

$$a\mathrm{e}^{\mathrm{i}(x+\pi/2)} + \mathrm{e}^{\mathrm{i}x} = b\mathrm{e}^{\mathrm{i}(x+\phi)}. \tag{2.24}$$

Factoring the exponentials,

$$a\mathrm{e}^{\mathrm{i}x}\mathrm{e}^{\mathrm{i}\pi/2} + \mathrm{e}^{\mathrm{i}x} = b\mathrm{e}^{\mathrm{i}x}\mathrm{e}^{\mathrm{i}\phi}, \tag{2.25}$$

and we can divide out $\mathrm{e}^{\mathrm{i}x}$, leaving

$$a\mathrm{i} + 1 = b\mathrm{e}^{\mathrm{i}\phi}. \tag{2.26}$$

This equation makes it obvious that

$$\phi = \tan^{-1} a, \tag{2.27}$$
$$b = \sqrt{a^2+1}, \tag{2.28}$$

so the desired identity is

$$a\cos x + \sin x = \sqrt{a^2+1}\sin(x + \tan^{-1} a). \tag{2.29}$$

This is exactly the sort of thing we need to simplify the left-hand side of equation (2.22). The square root and arctangent functions do not present a problem since those expressions are just constants (not functions of x, or in our case t). Applying (2.29) to (2.22), we get

$$V_{\mathrm{L}}\sqrt{(t_{\mathrm{c}}\omega_{\mathrm{L}})^2+1}\sin\left[\omega_{\mathrm{L}}t + \theta_{\mathrm{L}} + \tan^{-1}(t_{\mathrm{c}}\omega_{\mathrm{L}})\right] = V\sin\omega t. \tag{2.30}$$

Since $t_{\mathrm{c}}\omega_{\mathrm{L}}$ appears twice, we begin using α in place of it:

$$V_{\mathrm{L}}\sqrt{\alpha^2+1}\sin(\omega_{\mathrm{L}}t + \theta_{\mathrm{L}} + \tan^{-1}\alpha) = V\sin\omega t. \tag{2.31}$$

Now, an equation between two sinusoidal functions of t can only hold true for all values of t if the frequencies, amplitudes, and phases of these two functions are all identical. Thus from (2.31) we obtain the three equations:

$$\omega_{\mathrm{L}} = \omega \tag{2.32}$$
$$V_{\mathrm{L}}\sqrt{\alpha^2+1} = V, \tag{2.33}$$
$$\theta_{\mathrm{L}} + \tan^{-1}\alpha = 0. \tag{2.34}$$

Solving for V_{L} and θ_{L}, we have

$$V_{\mathrm{L}} = \frac{V}{\sqrt{\alpha^2+1}} \tag{2.35}$$
$$\theta_{\mathrm{L}} = -\tan^{-1}\alpha. \tag{2.36}$$

At this point, we observe that the voltage swing on the load is taken down by the damping divisor $d = \sqrt{\alpha^2+1}$, and that the load incurs a phase lag of $\theta = -\theta_{\mathrm{L}} = \tan^{-1}\alpha$. Plugging the equations for V_{L} and θ_{L} back into (2.18), we finally have that the unique real solution to the differential equation (2.6) is

$$v(t) = \frac{V}{\sqrt{\alpha^2+1}}\sin(\omega - \tan^{-1}\alpha). \tag{2.37}$$

Chapter 3

Clock Schemes

Clocking Details

There are several clocking details that are determined by the clocking implementation chosen. These include the number of clocking signals available, the number of QDCA cells that can be controlled by a single wire, the size of the area clocked by a single wire, the number of layers of clock wires, etc. For a discussion of these issues, see chapter 4.

Before Design

Before discussing the particular clocking strategies, it may be useful to discuss the types of clocking approaches. There are two broad issues to be briefly discussed. The first is in regards to the broadest notion of floorplanning. The second is in regards to how the clocking wires are used.

Floorplanning Approaches

Historically, clocking floorplans for QDCA have used one of two general approaches: zone regions and columnar regions. The zone clocking regions approach assumes that precise regions can be defined in both the x and y directions (figure 3.1). In other words, small square or rectangular shapes can be used in creating clocking floorplans. The columnar approach, on the other hand, assumes precise definition in only one direction, leading to floorplanning with long (compared to the QDCA cell) columnar shaped regions (figure 3.3). The zone floorplanning approach leads to greater circuit densities and shorter feedback loops. The columnar approach is assumed to be more realistic (i.e. easier to implement) in the short-term.

Clock Signal Approaches

The second point concerns how the clocking wires are used. There are two approaches here as well. The first has also traditionally been called "zone clocking" since it was usually associated with the zone floorplan methodology discussed above. This approach assumes that entire regions of the QDCA circuit are controlled at the same time. The clocking field controlling the entire region rises and falls uniformly across the controlled zone (figure 3.2). The second approach is wave type clocking where the clocking field across a region is non-uniform. The clocking wires are used to create a traveling wave, or computational wave, where computation occurs on the leading edge of the wave. This leading edge, or gradient, is pushed forward through and across the QDCA circuit independent of any "zones". It is important to note that the same set of clocking wires can be used to generate either type of clocking signal. It is a question of what waveforms the wires produce rather than how they are placed.

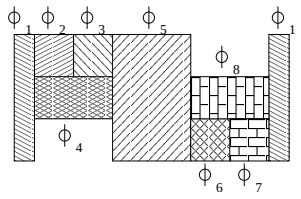


Figure 3.1. Example of a zone floorplanned circuit. This clocking zone layout is from a reversible crossover circuit. Notice that small zones abut each other on both the x and y dimensions.

Clocking Strategies

Several clocking strategies have been proposed. This is not an exhaustive list of precise clocking implementations, but rather a set of strategies that can be employed in combination in different sections of a system. The four basic strategies discussed here are Landauer clocking, Bennett clocking, bi-directional pulse clocking, and uni-directional Bennett clocking.

Landauer

Landauer clocking is a strategy in which the clocking wave moves in only one direction. It can be thought of as a traveling wave in which data is latched at the peaks,

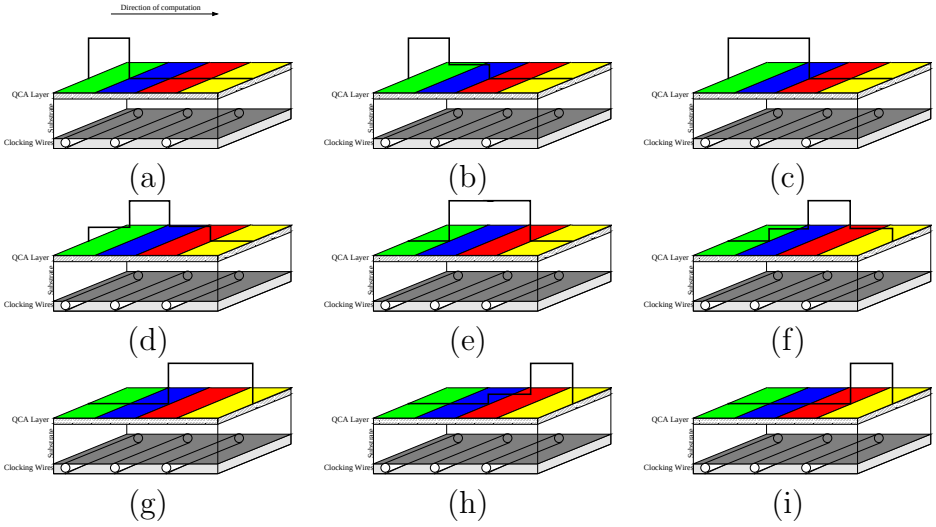


Figure 3.2. Zone Clocking: Entire regions rise and fall together. Here, a simple set of clocking zones computes from left to right.

computation is occurring on the leading edge, and cells are relaxing on the falling edge (figure 3.6).

The advantages of Landauer clocking include natural support for very fine-grained pipelining and high throughput on the architectural side, and it requires only simple clock forms on the lower-level design side of things. Disadvantages include that Landauer clocking requires logic gates with very fine-grained reversibility.

Retractile Cascade

In the retractile cascade, or Bennett, clocking scheme, computation proceeds in one direction but the clock is used to decompute as well as compute. The clocking wave sweeps across the circuit, remaining high, computing at the leading edge of the wave (figures 3.7(a-f)). When the wave reaches the edge of the circuit (figure 3.7(g)), the output remains latched in an area of high clock field and the decomputation begins as the clock is released from the right to the left (figures 3.7(h-j)). At the end of the complete clocking cycle, the original input and the final output remain latched (figure 3.7(k)).

The main advantage of Bennett clocking is that any irreversible circuit inside the Bennett clocked region will be executed reversibly. The catch, though, is that at the end of the clocking cycle both the original input and the output are latched. This can introduce challenges for high level reversible pipelining. In addition, Bennett clocking

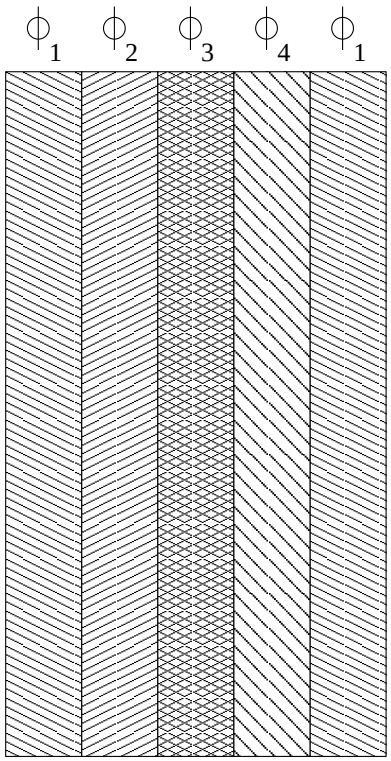


Figure 3.3. Example of a zone clocked circuit. This clocking zone layout is from a reversible crossover circuit. Notice that small zones abut each other on both the x and y dimensions.

requires non-uniform clocking signals (figure 3.8). Unlike Landauer clocking where one waveform but multiple phases are required, in Bennett clocking each clocking wire has a different wave to generate the Bennett clocking behavior.

Bi-directional Shift (aka Pulse)

The bidirectional shift, or pulse, clocking strategy allows a single region of high clock field to travel either left or right, controlled by the same set of physical wires (figures 3.9, 3.10).

It is reminiscent of the Landauer scheme in that computation occurs on the leading edge of the wave and the circuit is released on the trailing edge. There are a few important differences, though. First, rather than a periodic wave, the shift involves only a single pulse. Second, the pulse travels both left to right and right to left on the same set of clocking wires. These mean that the wave forms generated in this

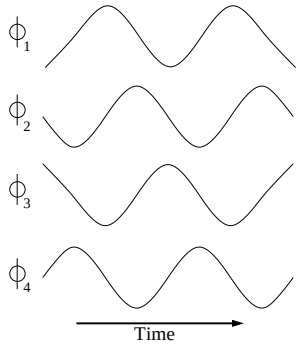


Figure 3.4. Example of wave forms generated by each clocking wire for a four phase wave-style clock signal. This corresponds to a Landauer type clocking signal as in figure 3.6.

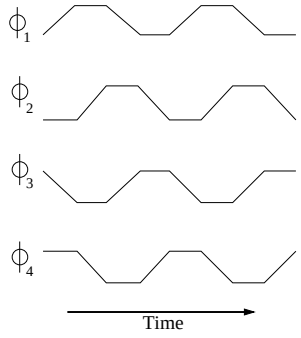


Figure 3.5. Example of wave forms generated by each clocking wire for a zone-style clock signal. This corresponds to a clocking signal as in figure 3.2.

clocking scheme are more complicated than the simple periodic forms generated for the Landauer clocking.

The bi-directional shift has a substantial advantage in that it allows data to be shifted in two directions using only one set of hardware. However, like Bennett clocking, it requires each clocking wire to generate a unique signal (figure 3.11).

Uni-directional Bennett

In the uni-directional Bennett clocking strategy, the clocking wave sweeps across the circuit, remaining high, computing at the leading edge of the wave (figure 3.12(a-i)).

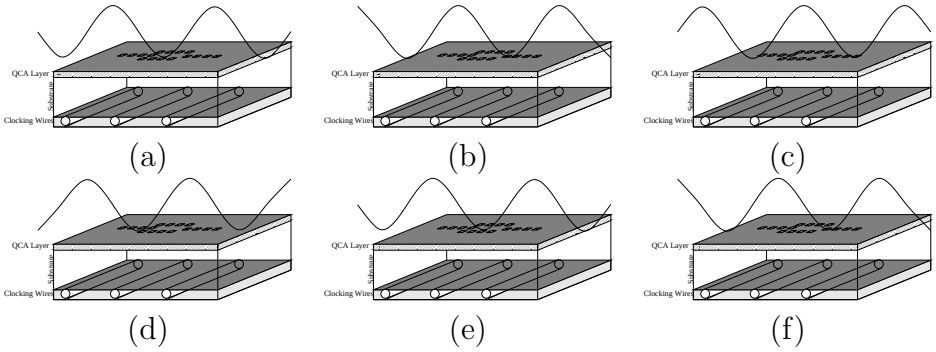


Figure 3.6. Landauer Clocking: Computation proceeds from left to right. Data is latched at each peak. Computation occurs on the rising edge (to the right of each peak) as the wave moves to the right.

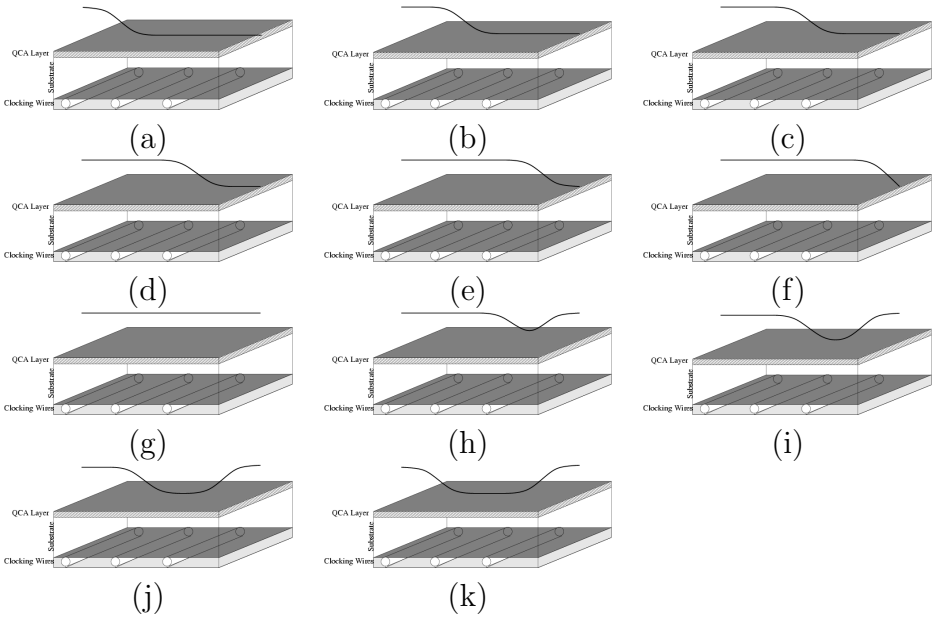


Figure 3.7. Retractile Cascade Clocking: Computation proceeds from left to right. Data is latched and remains latched until the result has been latched at the right. The clock is then retracted until only the input and output are latched.

When the entire circuit is held high, the circuit is released from the original input side (figure 3.12(j-p)). At the end of the clock cycle, the only data latched is the output (figure 3.12(q)).

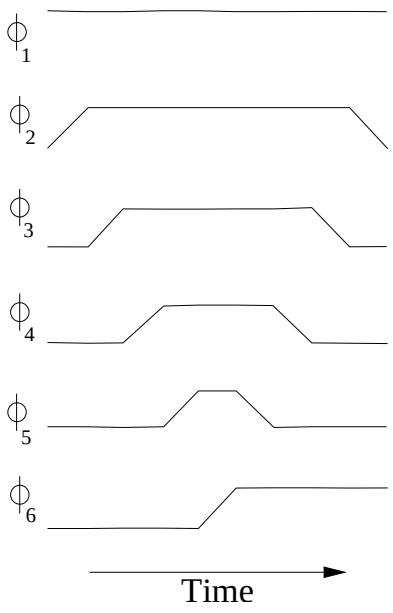


Figure 3.8. Example of wave forms needed to generate retractile cascade, or Bennett, clocking.

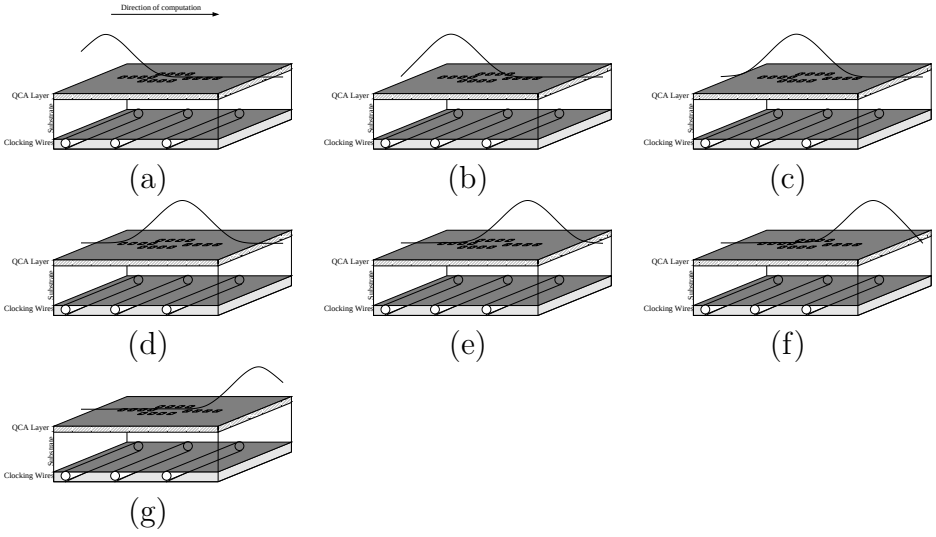


Figure 3.9. Bi-directional Shift: Shift Right

The uni-directional Bennett clocking scheme can be thought of as a combination of the Bennett and Landauer schemes. There are logic gates that are reversible if the entire gate is charged before being released from one side or the other. The needs for the clock in this case are similar to the Landauer scheme in that the clock wave moves

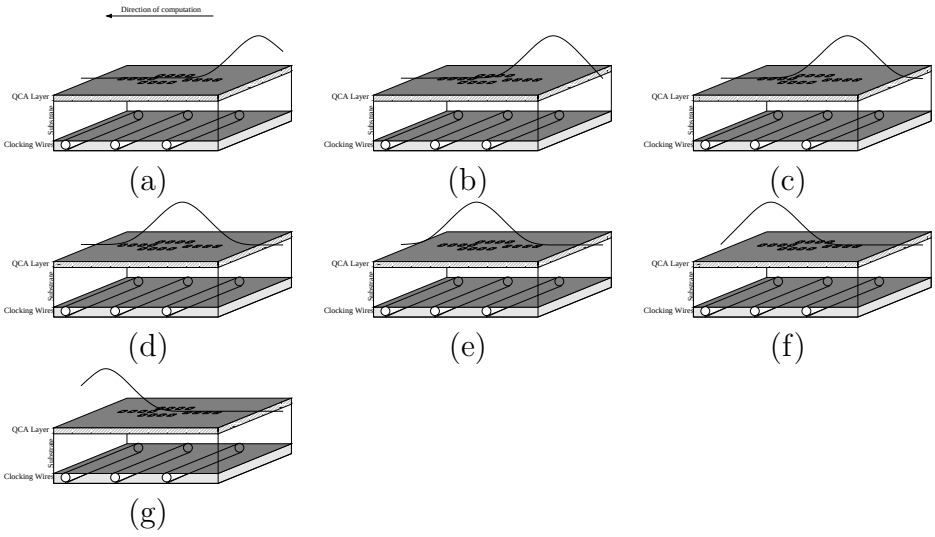


Figure 3.10. Bi-directional Shift: Shift Left

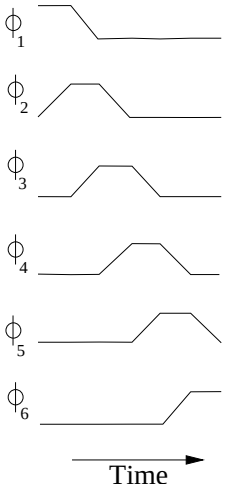


Figure 3.11. Example of wave forms needed to generate the bi-directional shift to the right.

in just one direction, but as in the Bennett scheme, the entire circuit is charged prior to being released. The key difference coming in the direction of the release.

The uni-directional Bennett scheme combines the best of Bennett clocking with the advantages and disadvantages of Landauer clocking. Uni-directional Bennett has the distinct advantage over Bennett clocking of having only the output latched at the end of the clock cycle. This implies higher throughput than Bennett clocking although

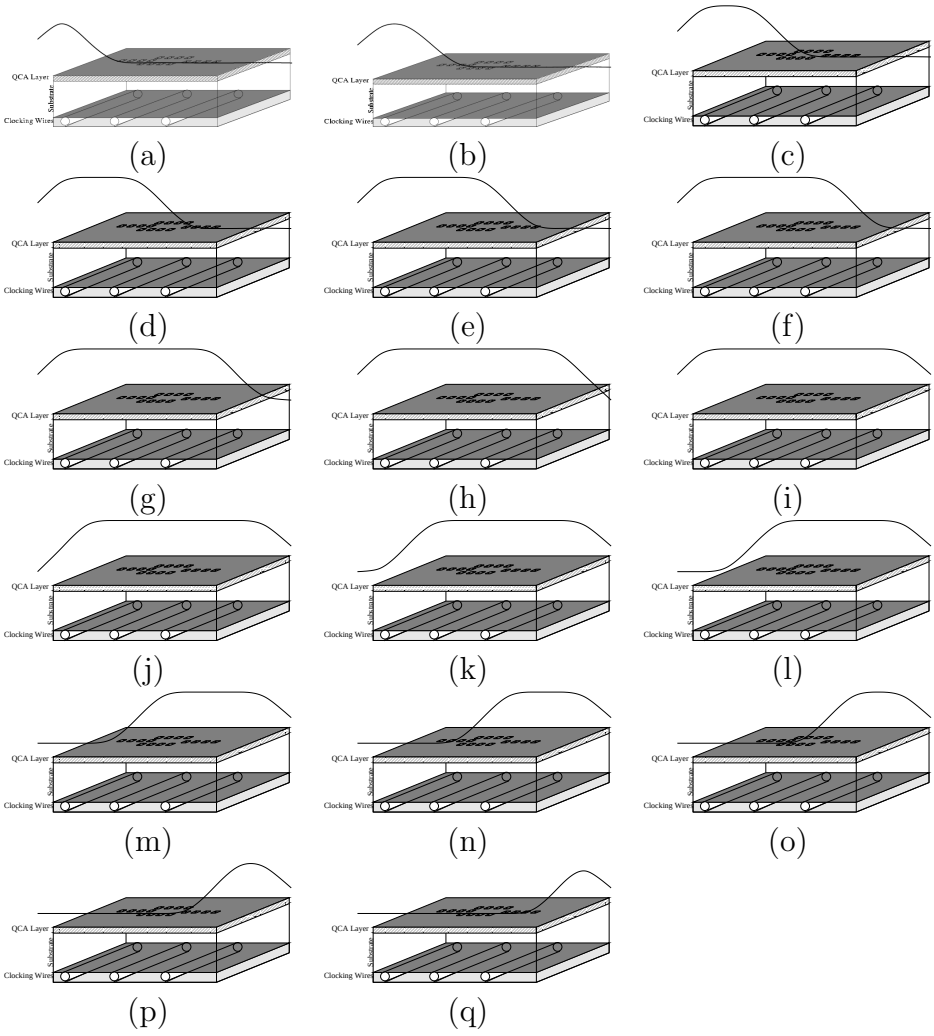


Figure 3.12. Uni-directional Bennett

lower than strict Landauer clocking throughput. It also means that pipelining is more natural in uni-directional Bennett than in Bennett clocking since at the end of the cycle only the output is still latched. Also, while uni-directional Bennett, like Landauer clocking, requires logic gates to be reversible if the system is to be reversible, there is greater flexibility in designing the reversible gates since only the entire gate needs to be reversible as opposed to each segment of the gate.

Table 3.1. Comparison of Clocking Strategies

Clocking Strategy	Pros	Cons
Landauer	• Fine-grained pipelining • High throughput	• Reversibility requires reversible logic family
Bennett	• Imposes low-level reversibility on any logic family	• Lowers throughput • Requires non-uniform clock signals • Input and output both latched at end of cycle
Bi-directional Shift	• Transmits data in two directions	• Requires non-uniform clocks
Uni-directional Bennett	• Only the output is latched at end of cycle (as opposed to Bennett) • Allows more gate families than strict Landauer clocking • More pipeline friendly than Bennett	• Lower throughput than strict Landauer • Reversibility requires reversible logic family

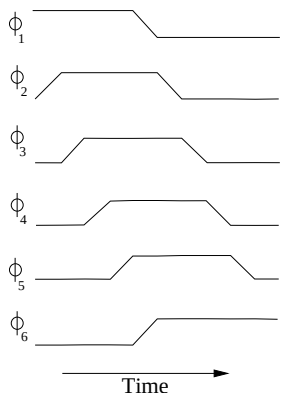


Figure 3.13. Example of wave forms needed to generate uni-directional Bennett clocking.

No Clocking

It is worth mentioning briefly that there is another clocking philosophy that has not been extensively explored. That is choosing not to clock the QDCA circuit. It is impractical for an entire circuit to be unclocked due to problems with kink energy and the potential for errors to occur due to thermal properties when large numbers of QDCA cells interact without the restorative properties of the clock. However, there is some indication that unclocked regions within a clocked circuit can be exploited. For instance, a signal could be latched by a clock at the edge of an unclocked region and then launched through the unclocked region to be "caught" by the clock at the other end of the region [1]. Another powerful paradigm that could be exploited in these regions is that of cellular automata. The cellular automata side of QDCA has not been exploited in previous architecture work, but the undeveloped potential is there and could be used as a component in a larger, clocked system.

Summary

Several topics related to clocking strategies have been discussed here. They include:

- Zone floorplanning vs Columnar floorplanning
- Zone clocking vs. Wave clocking
- Clocking Strategies: Landauer, Bennett, Bi-directional Shift, Uni-directional Bennett, (and no clocking).

[1] Named the "Hail Mary" architecture by Mike Frank

The clocking strategies each have advantages and disadvantages. In short:

- Landauer Clocking: Landauer clocking signals are regular, periodic signals that are offset in phase but identical otherwise. This clock strategy lends itself to circuits with fine-grained pipelining and high throughput. It also requires the circuit to be reversible at very fine levels.
- Bennett Clocking: Bennett clocking signals are periodic but more complicated in shape than Landauer clocking signals. Bennett clocking imposes reversibility on any irreversible circuit at the cost of time. In addition, since Bennett clocking leaves both the input and output latched at the end of a cycle, Bennett clocking introduces challenges when it is part of a pipelined system.
- Bi-directional Shift (Pulse) Clocking: This is the only clock that allows information to flow in two directions on the same circuit. It requires a circuit with very fine grained reversibility (i.e. a wire) to be reversible. Like Bennett clocking, it requires more complicated clocking signals than Landauer clocking.
- Uni-directional Bennett Clocking: Combines the throughput and pipelinability of Landauer clocking with the reversibility of Bennett clocking while minimizing the costs of each. Uni-directional Bennett clocking is slower than Landauer clocking and requires complicated clocking signals. However, it is more pipeline friendly than standard Bennett clocking, and it allows more freedom in designing reversible circuits than Landauer clocking. Uni-directional Bennett clocking incurs a higher space overhead than Bennett clocking, but significantly lessens the time overhead. Similarly, the time overhead is greater than that of Landauer clocking, and the space overhead is significantly lessened.
- No Clocking: Areas with no clock seem to have a great deal of potential as components within a clocked system that should be explored.

Chapter 4

Clock Circuit Issues

This chapter introduces and characterizes a novel circuit design for low power clock distribution for QDCA. The characterization of the clocking circuitry includes the evaluation of the coupling capacitances among the wires and with the ground plane; moreover also the dissipative effects of the wires and the substrate are taken into consideration. The phase shift between neighboring wires is included in the evaluation as well as the number of wires driven by the same clock line, consequently the clocking circuitry electrical characteristics are a function of the chosen design parameters. The model of the clock tree is initially modeled as a RC circuit, then a resonant RLC circuit is proposed and its power dissipation performance is compared to an RC circuit as a function of the quality factor Q of the resonating circuit: it is shown that this approach can greatly reduce the power dissipated on the clocking layer of a QDCA circuit.

Overview on clock distribution circuit for QDCA

In the early proof of concept [18] work, the clock was explicitly delivered to every cell through metal wires. This was sufficient for the goals of the experiments, but it has some obvious shortcomings and prohibits large scale integration. In order to overcome these and to facilitate a shift toward molecular QDCA, a clocking scheme was envisioned that would use a sequence of metal wires buried beneath the QDCA layer that would generate an E-field that would control the tunneling within the QDCA layer [15]. By variably controlling the strength of the field at different points, directionality can be imposed on the QDCA circuit. In the typical scheme the wires are divided into four groups, and each wire is assigned a phased sinusoidal voltage source $V(t) = Vsin(2\pi f_0 t + \phi_i)$. The phases of the wires are $\phi_i = (i \cdot \pi/2, i = 0, 1, 2, 3)$. Note that at least three phases are needed to provide directionality to the flow of information on the QDCA layer.

Figure 4.1 shows a cartoon view of a possible implementation of the four phased clock distribution for QDCA. The wires are actually on the top of the QDCA layer to take advantage of the typical planar process for the metalization. A ground plane is found on the other side of the QDCA layer from the clocking wires in order to terminate

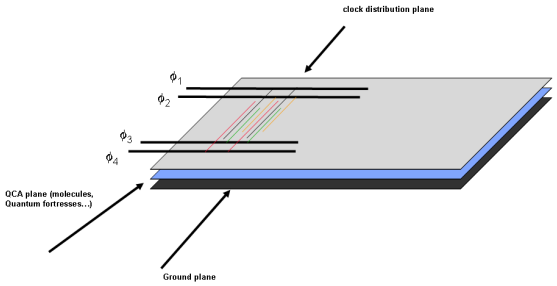


Figure 4.1. Possible clock distribution circuitry for QDCA

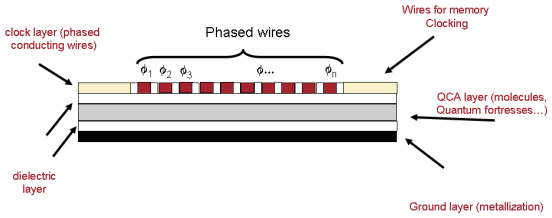


Figure 4.2. Cross Section of a generic QDCA implementation

the E-field lines. Four wires $(\Phi_1, \Phi_2, \Phi_3, \Phi_4)$ carry the four phase shifted signals. The actual distribution on the QDCA layer is obtained through smaller wires branched out from the main carriers.

Capacitive coupling

The overall capacitive effects on the clocking wires can be considered as the sum of two main contributions:

$$C_{tot} = C_W + C_L$$

where C_W represents the coupling with neighboring wires and C_L represents the coupling with the ground plane. In the following subsections we analyze in detail these two contributions.

Capacitive coupling with the neighboring wires

The clock wires experience a capacitive coupling with their neighbors. For a generic wire, the strongest coupling occurs with the two closest neighbors. The capacitance model between wires (that is usually called C_m) is typically obtained in VLSI by

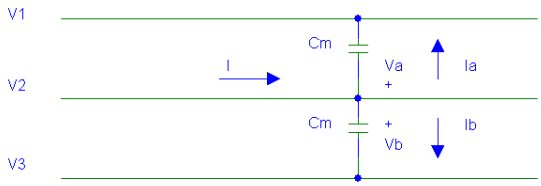


Figure 4.3. Parasitic coupling with neighbor wires

solving Green's function with a multipole expansion (as in the FastCap software [27]), by using a finite difference method to solve the Poisson equation (1Poisson [40]), or by using finite elements (FIERCE [7]). Since this work is the first one addressing this problem, we make the simplifying assumption that the value of the capacitance is obtained from the simple plane capacitor formula

$$C_m = \epsilon_0 \epsilon_r^w \cdot \frac{A_w}{d_w}$$

where d_w is the distance between two neighbor wires, A_w is the area of the wire exposed to the neighbors and ϵ_r^W is relative permittivity of the dielectric between the wires. Future work can address the refinement of the model.

In addition, there is an effect similar to what happens in VLSI when the wires of a bus are affected by crosstalk [16], the time varying signal on the two neighboring wires affects the total charge Q that needs to be provided to the target wire to obtain a certain value of V on it. This effect can be modeled with a k multiplicative factor applied to C_m.

Consider a wire coupled with its two neighbors through the capacitance C_m as shown in Figure 4.3. The equivalent capacitance seen on the middle wire can be calculated as follows:

$$V_1(t) = sin(\omega t - \phi)$$
$$V_2(t) = sin(\omega t)$$
$$V_3(t) = sin(\omega t + \phi)$$

where ϕ represents the phase shift between neighboring wires.

Further,

$$V_a(t) = V_2(t) - V_3(t) = -2sin(\frac{\phi}{2}) \cdot cos(\omega t + \frac{\phi}{2})$$

$$V_b(t) = V_2(t) - V_1(t) = 2sin(\frac{\phi}{2}) \cdot cos(\omega t - \frac{\phi}{2})$$

from the definition of capacitance:

$$I_b(t) = C_m \frac{dV_b(t)}{dt}$$
$$I_a(t) = C_m \frac{dV_a(t)}{dt}$$

and from the Kirkhoff's Law

$$I(t) = I_a(t) + I_b(t) = C_m(\frac{dV_a}{dt} + \frac{dV_b}{dt})$$

being:

$$\frac{dV_a}{dt} = 2\omega sin(\frac{\phi}{2}) sin(\omega t + \frac{\phi}{2})$$
$$\frac{dV_b}{dt} = -2\omega sin(\frac{\phi}{2}) sin(\omega t - \frac{\phi}{2})$$

Therefore the current is:

$$\begin{aligned} I(t) &= 2C_m\omega sin(\frac{\phi}{2}) \left[sin(\omega t + \frac{\phi}{2}) - sin(\omega t - \frac{\phi}{2}) \right] \\ &= 4C_m\omega sin(\frac{\phi}{2}) \left[cos(\omega t) sin(\frac{\phi}{2}) \right] \end{aligned}$$

The capacitance seen on the middle wire is therefore (from the definition of capacitance)

$$C_W = \frac{I(t)}{\frac{dV_2(t)}{dt}}$$

where

$$\frac{dV_2(t)}{dt} = \omega cos(\omega t)$$

and therefore finally:

$$C_W = k \cdot C_m = 4sin^2(\frac{\phi}{2}) \cdot C_m$$

Capacitive coupling with the ground plane and dissipative effects

As shown in Figure 4.2, the QDCA layer is sandwiched between the clocking wires and the ground plane. The capacitance through the QDCA layer depends on the relative permittivity of the chosen material to implement the QDCA circuits and on its vertical size. Similar to what was seen for C_W, a simplifying assumption is made to model C_L as:

$$C_L = \epsilon_0 \epsilon_r^q \cdot \frac{A_q}{d_q}$$

where d_q is the distance between the wire and the ground plane, A_q is the area of the wire facing the QDCA layer and ϵ_r^q is relative permittivity of the QDCA layer.

Resonant RLC circuit for low power clock distribution

Consider a simple RC circuit (see fig 4.4) as the model of the clock distribution where R_W represents the overall resistance of the clocking wire ($R_W = R_{W1} + R_{W2}$, where R_{W1} is the resistance of the distribution wire with larger cross-section, R_{W2} is the resistance of the clocking wire with smaller cross-section), C_W represents its capacitance with its two neighboring wires, and C_L and R_L represent the capacitance and the dissipative effect with the ground plane through the dielectric composing the QDCA layer. The total capacitance is $C_{tot} = C_W + C_L$, and the power dissipated per clock period can be calculated as follows.

From the definition of voltage as the energy per unit charge, the energy stored on the ideal capacitor should be $QV = C_{tot}V^2$ since all the work done on the charge in moving it from one plate to the other would appear as energy stored. However, since the work done to put a dq charge at a potential V is $dU = Vdq$, the total energy to put Q charge on the capacitor is

$$U = \int_0^Q V dq = \int_0^Q \frac{q}{C_{tot}} dq = \frac{Q^2}{2C_{tot}} = \frac{C_{tot}V^2}{2}$$

This expression shows that just half of the $QV = C_{tot}V^2$ work appears as energy stored in the capacitor. For a finite value of R_W and assuming that R_L is negligible compared to C_L, half of the energy supplied by the power supply for charging the capacitor is dissipated as heat in the resistor, regardless of the size of the resistor. Considering a sinusoidally changing voltage supply from 0 to V, then, the energy dissipated on R_W in the charge and discharge of the capacitor C_{tot} occurring in a period is $C_{tot}V^2$. Therefore, for a frequency of operation f_0, the power dissipated on

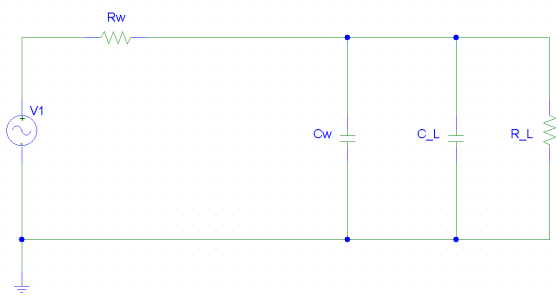


Figure 4.4. Model of Clocking with an RC circuit

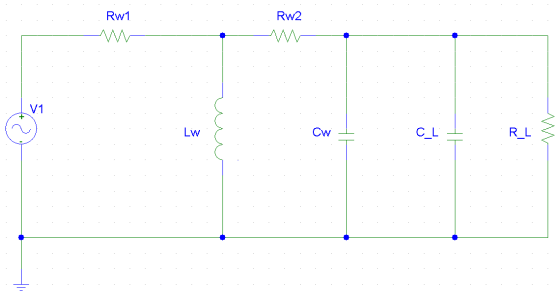


Figure 4.5. Model of Clocking with an RLC resonating circuit

the clocking distribution circuit is:

$$P_{RC} = C_{tot} V^2 f_0 \tag{4.1}$$

The RC circuit in figure 4.4 is a low band pass filter characterized by its cutoff frequency, $f_0 = \frac{1}{R_{tot} C_{tot}}$ This is the frequency at which the amplitude of the voltage on the capacitor is halved, meaning that the operating frequency is limited by the RC constant.

The distribution of the clock could be done through a resonant parallel RLC circuit. A resonating circuit has a very limited current drain from the source at its resonating frequency. This has been studied for low power clock distribution in conventional VLSI [9] [6]. It is useful to compare the power dissipation of the RC and RLC circuits. We will show that if the resonant circuit has a good quality (measured by the Q factor) the introduction of the resonant tank improves dramatically the power performances.

Consider the RLC circuit in figure 4. L_W represents the inductor introduced to generate the resonant circuit. As before, the total capacitance is $C_{tot} = C_W + C_L$.

The relation between L_w and C_{tot} to obtain resonance is $2\pi f_0 L_w = 1/(2\pi f_0 C_{tot})$. Therefore, the resonant frequency can be obtained by:

$$f_0 = \frac{1}{2\pi\sqrt{L_W C_{tot}}} \tag{4.2}$$

The value of L_W can be tuned for the value of the parasitic capacitances C_{tot} to keep the product of the two constant

Since at f_0 the reactive loads cancel each other, the current absorbed from the voltage supply flows only through the resistive load. Therefore the power dissipated at resonance is

$$P_{RLC} = \frac{V^2}{2R_{tot}} \tag{4.3}$$

where $R_{tot} = R_{W1} + R_{W2} + R_L$. To compare this result with the previous for RC in eq. 4.1 (similar to [9]) we introduce the definition of quality factor Q as:

$$Q = 2\pi f_0 \cdot \frac{Maximum\ Energy\ Stored}{Average\ Power\ Dissipation} \tag{4.4}$$

The maximum energy stored in the circuit is the amount of energy resonating between C_{tot} and L_W which can be expressed as the maximum amount of energy on the capacitance: $\frac{C_{tot}V^2}{2}$. As shown above average power at resonance is $P_{RLC} = \frac{V^2}{2R_{tot}}$, therefore

$$\begin{aligned} Q &= 2\pi f_0 \cdot \frac{\frac{C_{tot}V^2}{2}}{\frac{V^2}{2R_{tot}}} \\ &= 2\pi f_0 \cdot R_{tot} C_{tot} \end{aligned} \tag{4.5}$$

From 4.6 and 4.3

$$P_{RLC} = \frac{\pi C_{tot} f_0 V^2}{Q}$$

The ratio between P_{RLC} and P_{RC} is therefore:

$$\frac{P_{RLC}}{P_{RC}} = \frac{\pi C_{tot} V^2 f_0}{Q} \cdot \frac{1}{C_{tot} V^2 f_0} = \frac{\pi}{Q} \tag{4.6}$$

Therefore, if $Q \geq \pi$ the RLC resonating circuit is dissipating less power than the RC.

Parallel load effect

Previously we considered that the voltage supply is connected to a single clocking wire, however, for real clocking of QDCA circuits, the same voltage supply will most likely be connected to several (n) parallel clocking wires as shown in Figure 4.1. In this section we will introduce n as a parameter for the evaluation of the quality of the resonator. The introduction of n parallel loads will obviously affect both R_{tot} and C_{tot} yielding new values of $R'_{tot} = R_{W1} + \frac{R_{W2}}{n} + \frac{R_L}{n}$ and $C'_{tot} = n \cdot C_L$.

The parallel effect of the load equally effects C_L and R_L. The partition of current on them, then, remains the same. Therefore, the equation 4.1 for the RC circuit with n parallel wires is only slightly modified to:

$$P'_{RC} \geq n \cdot C_{tot} V^2 f_0$$

whereas for the resonant RLC circuit we have:

$$P'_{RLC} = \frac{V^2}{2R'_{tot}} = n \cdot \frac{\pi C_{tot} f_0 V^2}{Q} \tag{4.7}$$

where $R'_{tot} = R_{W1} + R_{W2} + R_L$. The increase of n scales the power stored in the clocking circuit by the same factor. Therefore the ratio of P_{RLC} and P_{RC} assumes the same expression of equation 4.6 and using n parallel wires does not affect the condition $Q \geq \pi$ for the quality factor of the resonating circuit.

Finally, to maintain the target resonating frequency f_0 the scaling of C_{tot} by n must be balanced by a symmetric scaling of L_W, therefore $L'_W = \frac{L_W}{n}$.

Evaluations

In this section we evaluate the power dissipation per unit area (P_d measured in W/cm^2) of a possible layout for clock distribution. The power dissipation per square

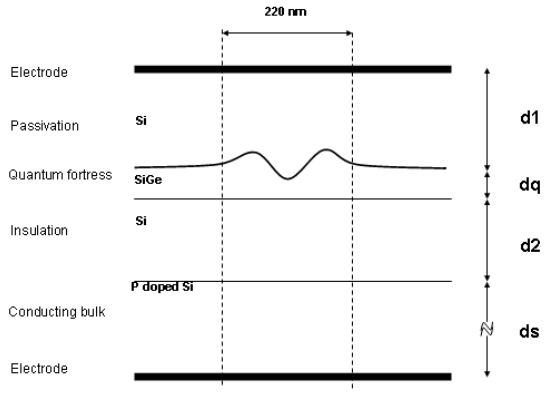


Figure 4.6. Crossectional diagram of a quantum fortress type QDCA chip

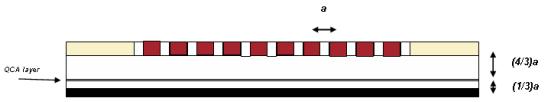


Figure 4.7. Geometric constants according to [15]

centimeter is compared to the typical limit of $100W/cm^2$ that represents a critical limit for the capability of heat dissipation in VLSI. The considered QDCA implementation is based on quantum fortresses and the dimensions involved in the computation of the parameters are summarized in figure 4.6. Moreover, parameters reported in table 4 are chosen according to the geometric rules defined in [15] and are shown in figure 4.7: the geometric constant $a = 220nm$ is such that each QDCA cell is clocked by a single wire.

First, consider the effect of the Q of the RLC resonator has on the power dissipation of the circuit for a range of frequencies (figure 4). Notice that for Q=1, there is effectively no resonance, and $P_{RLC} = P_{RC}$. As Q increases, there is a substantial decrease in power consumption. Notice, too, that the intersection with the 100 W/cm^2 line occurs at higher frequencies as the Q increases. Note that the plots assume that for each frequency the tuning L_W is set to allow resonance at that specific frequency.

The plot also shows also a Q* value of about 300 that is obtained by the given geometry and parameters reported in Table 4 and therefore represents the actual Q for the chosen geometries.

It is also important to look at the power dissipation for a given frequency (e.g. 1 GHz) and varying supply voltages. As can be seen in figure 4, for a resonating RLC circuit with a quality factor of Q* the circuit can be driven with a voltage up to 20

Table 4.1. Physical parameters

Parameter	Value	Unit	Description
a	2.20E-07	m	geometric constant
L	1.00E-02	m	length of a wire
h	3.00E-08	m	height of a wire
w	3.00E-08	m	width of a wire
Δ	2.20E-07	m	pitch between wires
d	2.05E-07	m	distance between wires (Delta-w/2)
A_q	3.00E-10	m^2	Area facing the QDCA layer
A_w	3.00E-10	m^2	Area facing the other wires
S	9.00E-16	m^2	Area of the section
ρ**(Cu)**	1.7E-08	$ohm*m$	resistivity of the wire
ρ **(Si)**	6.40E+02	$ohm*m$	resistivity of intrinsic Silicon
ρ **(P doped Si)**	1.00E-05	$ohm*m$	resistivity of P doped Silicon
ϵ_r**(SiGe)**	1.41E+01		relative permittivity of Silicon Germanium alloy
ϵ_r**(Si)**	11.68		relative permittivity of Silicon (also doped)
ϵ_0	8.85E-12	$m^{-3}kg^{-1}s^4A^2$	permittivity of free space
V_1	13	V	Source voltage amplitude
ϕ	1.570796	$radiant$	phase shift between adjacent wires
$d1$	2.93E-07	m	thickness layer 1
dq	2.44E-08	m	thickness QDCA layer
$d2$	2.44E-08	m	thickness layer
ds	2.44E-08	m	thickness of substrate

Table 4.2. Circuit parameters

Circuit Parameter	Value	Unit	Description
C0	8.55932E-14	F	Capacitance with the substrate
k(phi)	2		multiplicative factor
Cm	1.30E-14	F	Capacitance with adjacent wire
Ctot	1.12E-13	F	Total Capacitance
Rw	1.89E+05	Ohm	Resistance of a wire
Rl	7.30E+05	Ohm	Parasitic Resistance of the QDCA layers
Rtot	9.19E+05	Ohm	Total resistance

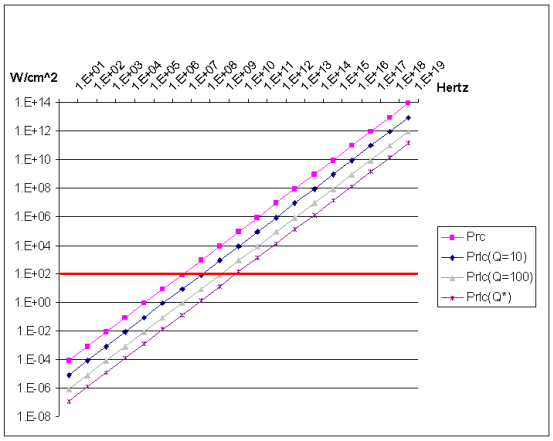


Figure 4.8. P_d as a function of frequency for different values of Q. In red: $100W/cm^2$ limit

V without hitting the limit of power dissipation 100 W/cm^2. This result provides a valuable degree of flexibility since at the moment it is unknown what driving voltage will be required to obtain the switching of the quantum fortress based QDCA cells from the locked to the relaxed state.

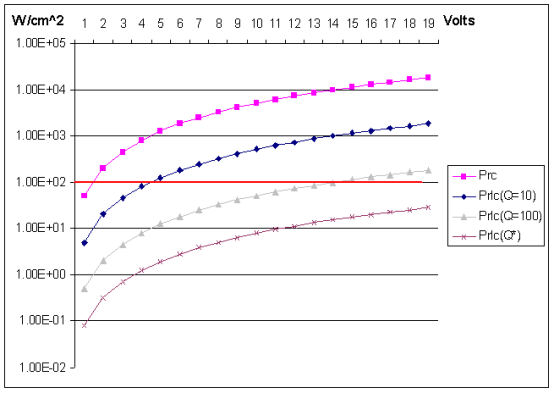


Figure 4.9. P_d as a function of supply voltage for different values of Q at f=1GHz. In red: $100W/cm^2$ limit

Finally we analyze the impact of the phase shift between neighboring wires on the power consumption. Figure 4 shows the impact of a reduced phase shift between neighboring wires given a frequency of 1 GHz and a supply voltage of 1 V.

From 4 it can be seen that for $V_1 = 1V$ the RC circuit dissipates less than 100 W/cm^2.

Finally in figure 4 we analyze the effect of a reduced phase shift between neighboring

wires on the power consumption for $f = 1GHz$ and $V_1 = 1V$ It can be seen with a $\phi < \pi/2$ the power dissipation can be reduced under the limit of 100 W/cm^2 without a resonating circuit. This effect is due to the reduced capacitive load seen throughout the clocking circuitry and provides an extra parameter to reduce the power dissipation in a QDCA clocking circuit when an RLC circuit is not used.

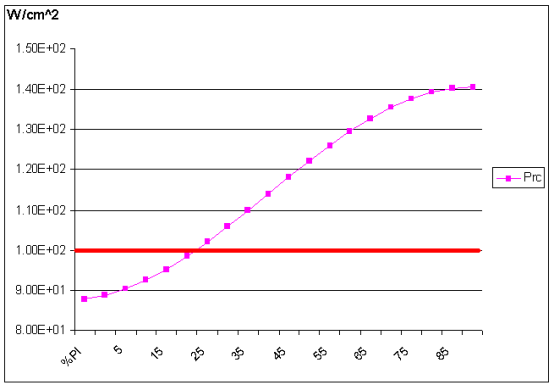


Figure 4.10. P_d as a function of ϕ at f=1 GHz and V_1=1 V. In red: $100W/cm^2$ limit

Conclusion

This chapter has addressed the characterization of the clock distribution circuits for QDCA. While there is a substantial body of literature on QDCA circuit design, little has been said about the clock distribution circuits required to make the QDCA circuits operable. This chapter has provided an electrical characterization of the parameters involved in the clocking circuitry and compared two approaches to implementing the clocking circuitry: a simple RC circuit and an improved resonating RLC circuit.

The analysis of the RLC circuit shows that it reduces power consumption below that possible with the RC allowing the clocking circuit to operate at higher computational frequencies while dissipating less than 100 W/cm^2.

Chapter 5

QDCA Circuit Design

As mentioned in the introduction, drawing a line in the design space and insisting on reversibility below that line while tolerating irreversibility above the line is a good approach to designing with QDCA. This chapter discusses some of the architectural tradeoffs, where they can be made, the costs and the opportunities, and some example architectures.

It is important to make a distinction here between physical reversibiltily and logical reversibility. At some point, depending on design restrictions and implentation choices, a point of diminishing returns is reached with physical reversibility where the cost of reversibility (time and space) overshadows the gains of reversibility (power). The location of this point will vary with the system and the implementations, but it is important to be aware of. Also, while physical reversibility may no longer offer substantial improvements beyond this point in terms of power savings, support for higher level logical reversibility may still be a useful design strategy. This sort of higher level consideration will not be addressed further in this document.

As with any system, before it can be designed, there is a set of decisions that need to be made. The questions that need to be answered are somewhat different in designing a QDCA system than in traditional CMOS design, though. There are two sides to the set of questions. The first is the set of questions that define what toolbox is available to pull from. The second set of questions determines how the design can be evaluated to see if it meets its time, space, and power budget goals. Table 5 lays out one way of asking these questions.

It is worth noting that any reversible part, be it clock or gate, can be abused to create a system that is dissipative. Special care should be taken in the interfaces between different clocking schemes and at points where multiple data produced by different functional units converges (i.e. feedback loops, memory requests, etc.). Common pitfalls at the interfaces between clocking regions include: data being produced and consumed at different rates (leading to bottlenecks or idling); incorrect hand-offs between regions (e.g. data produced by region one at time x but region two isn't ready to copy it until time x+Δ by which time region one has already released the data); mishandling the decomputation requirements of different regions (e.g. Bennett clocking leaves the input of a region latched while Landauer clocking does not).

Table 5.1. Questions Before Design

	Design	Analysis
QDCA Imp.	• Are physical crossovers allowed? • Are 45 degree cells allowed?	• What is the dissipation per cell switching event? • What are the physical dimensions of the cell? • What is the cell spacing? • How cells wide are the QDCA wires?
Clocking Imp.	• How many clock signals are allowed? • What signal shapes can be produced? • How many layers of clocking are available?	• What is the pitch of the clocking wires? • How big/small can the wires be? • What range of frequencies are available?
Budgets	• What space is available? • What time requirements apply? • What power budget is available?	• Given the answers to all other questions, does the design satisfy the time, space, and power requirements?

It is also worth noting that this chapter does not discuss the design of unclocked regions of QDCA cells. This is an area of design that has not been extensively explored and hopefully will be ready for inclusion in subsequent editions of this manual.

Circuit Design Strategies

Circuits can be designed with varying degrees of logical reversibility, independent of physical reversibility. The pairing of circuit design strategies with clocking strategies leads to physical reversibility or irreversibility. This section will discuss the basic options for circuit design strategy and then discuss the effect of different pairings on physical reversibility and the time/space tradeoffs of the different pairings.

There are two axes on which to classify these circuits. One is the direction of operation (uni-directional vs bi-directional), and the second is the level of reversibility (irreversible, gate reversible, sub-gate reversible). Not all six options are reasonable, though. For instance, the "bi-directional irreversible" option is nonsensical since if a gate is bi-directional, it must be reversible on some level.

Uni-directional Irreversible

The uni-directional irreversible option is the most explored circuit design strategy to date. In this strategy the circuit only produces meaningful output when the clock is run in one direction and the circuit is logically irreversible. A very basic example is a simple majority gate (figure 5.1). When the clock is reversed, the circuit no longer acts as a majority gate (specifically, it acts like a fan-out). Further, the majority gate is not a 1:1 function and information about the "losing" input is dissipated.

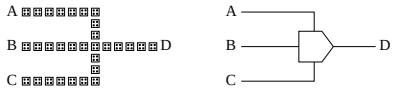


Figure 5.1. The majority gate is an example of a uni-directional, irreversible circuit.

Uni-directional Gate Level Reversible

There is a set of circuits that perform 1:1 functions but are reversible if and only if the entire gate is locked before being any of the QDCA values are released for a particular circuit implementation of the function. Further, they do not perform

the same function when the direction is reversed, making them uni-directional. Some implementations of circuits from gate classes 3, 5, 6, or 7 would fall into this category. These gate classes use a different set of functions for the inverse of the function rather than using the same gate as the inverse.

Uni-directional Sub-gate Level Reversible

There is also a set of circuits that are reversible at a much finer grain than the gate level. These tend to be much simpler circuits than the previous strategies. One example of this is a fanout circuit in which there is one input and either two or three copies of the input as outputs. If this circuit were clocked by a very fine Landauer type clock, there would be no dissipation since the level of reversibility in the circuit is comparable or superior to that in the clock. Further, this is a uni-directional circuit because if the direction of the clock were reversed, the result would be a majority gate rather than a fanout.

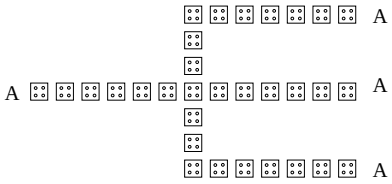


Figure 5.2. The basic fan-out circuit is an example of a uni-directional circuit that is reversible at the sub-gate level but is not bidirectional.

Bi-directional Gate Level Reversible

Bi-directional circuits perform the same function whether clocked from the left or from the right. As in the uni-directional case, a gate level reversible circuit performs a 1:1 function and does not erase any bits provided the entire gate is charged before being released. Figures 5.3 and 5.4 show the QDCA layout and a schematic of a Toffoli gate [8]. The Toffoli function negates bit C if bits A and B are both logical ones. The Toffoli function is its own inverse. This particular layout would dissipate if it was released in a Landauer type way at Area 1. Since the connetion to the copy of G1 at this point has already been released, there is no way to "copy" the value back to the stored value of G1.[1]

[1]This is a somewhat artificial example. With a simple change this circuit could be made sub-gate level reversible.

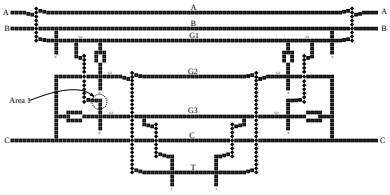


Figure 5.3. QDCA layout of an example of a bi-directional circuit that is reversible at the whole-gate level. This is one implementation of a Toffoli gate.

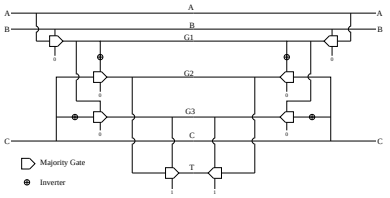


Figure 5.4. Schematic of an example of a bi-directional circuit that is reversible at the whole-gate level. This is one implementation of a Toffoli gate.

Bi-directional Sub-Gate Level Reversible

Combining two uni-directional, sub-gate level reversible components (e.g. the gate and its inverse) can create a bi-directional, sub-gate level reversible circuit as in the NAND/UnNAND circuit in figure 5.5

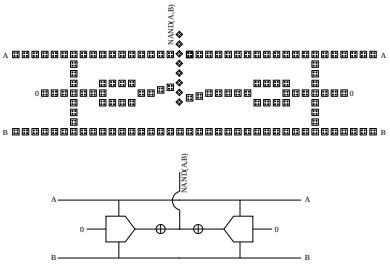


Figure 5.5. The combination of the NAND and UnNAND gates creates a bi-directional circuit that is reversible at the sub-gate level.

Matching Circuit and Clocking Strategies

Any circuit strategy can be matched with any clocking strategy, but some pairings are more advantageous than others. Table 5.2 shows which pairings are reversible and irreversible. Notice that one column in table 5.2 is completely reversible. This corresponds to the Bennett style clocking. While Bennett clocking can make any circuit style reversible, it incurs a time penalty of roughly 2x over Landauer clocking sine the clock cycle includes sweeping down and back a circuit rather than just down as in Landauer clocking.

There is also one row that is reversible regardless of the clocking scheme. Unfortunately, this is the bi-directional sub-gate level reversible circuit class with only one known member, the wire. Even though this class includes only the wire, it is important to include it in this discussion since a significant part of any design will be communication between components.

In general, Bennett and uni-directional Bennett clocking incur a time overhead. In addition, Bennett clocked circuits leave the input latched at the end of the clock cycle. To be fully reversible within a larger system using multiple clocking strategies, a Bennett clocked system must incur additional time and space penalty to account for the additional circuitry and/or time to decompute the input. The amount of additional overhead varies depending on the approach taken (i.e. the collapsed Bennett layout or the the reversible Bennett pipeline).

Gate level reversible circuits will in the general case incur a space overhead of approximately 2x. This is because any circuit can be made reversible by saving its inputs and intermediate solutions. Once the result is obtained, the intermediate results can be decomputed by means of a mirror circuit which takes as input all of the saved inputs and intermediate results and return the only the original input. The space overhead of this approach for general circuits can grow much bigger than 2x, but for small circuits on the level of a simple gate, the approximation is close.

For most of the rows in table 5.2, there are multiple entries that are reversible. There are a few basic guidelines deciding which reversible strategy to choose. In general, Bennett clocking incurs the most overhead. For each circuit strategy, if there is a reversible option using a clocking strategy other than Bennett, that other option should be used. Further, if both Landauer and uni-directional Bennett clocking are reversible options for a circuit strategy, Landauer clocking should be used since it has less overhead and is easier to implement. After applying these basic guidelines, there are six entries left to be considered (table 5.3). They are:

- Uni-directional irreversible circuit with Bennett clocking: Any irreversible circuit can be Bennett clocked, but there is a time penalty. Also, interfacing reversibly to other clocking regions introduces additional time and space penalties.

Table 5.2. Reversibility of Circuit Design Strategies and Clocking Strategies

	Landauer	Bennett	Bi-D Shift	Uni-D. Bennett
Uni-Irrev.	Irr	Rev	Irr	Irr
Uni-Gate Rev.	Irr	Rev	Irr	Irr
Uni-Sub-Gate Rev.	Rev	Rev	Irr	Rev
Bi-Gate Rev.	Irr	Rev	Irr	Rev
Bi-Sub-Gate Rev.	Rev	Rev	Rev	Rev

- Uni-directional gate level reversible circuit with Bennett clocking: this would seem to combine the cost of reversibility with the cost of imposing reversibility by clock. Except for a limited number of situations, there is little reason to design in this region. Reversible gates usually have a space overhead and combining them with Bennett clocking leads to the worst of the time/space overheads.

- Uni-directional sub-gate level reversible circuit with Landauer clocking: This is a small set of circuits, but it contains some very important ones such as fanout and a majority gate with its inputs saved and communicated to the output of the gate. Landauer clocking is very fast and has high throughput, but the amount of garbage data can grow exponentially when circuits are designed in this input-saving manner.

- Bi-directional gate level reversible circuit with Uni-directional Bennett clocking: Bi-directional gate level circuits will have some space overhead due to their reversibility, and uni-directional Bennett clocking has some time overhead when compared with Landauer clocking. However, the space overhead is likely to be less than the input-saving method, and the uni-directional Bennett strategy preserves pipelining and relatively high throughput.

- Bi-directional sub-gate level reversible circuit with Landauer or bi-directional shift clocking: The set of known bi-directional sub-gate level reversible circuits currently consists of the wire. This region of design space is important, though, because it shows that communication between different regions can be accomplished in two manners that are both reversible and efficient.

There are two basic rules-of-thumb to take away from the above discussion. First, reversible circuits incur a space penalty for being reversible. However, if the circuit is designed with reversibility in mind, that penalty can be minimized. Second, reversibility by clocking incurs a time penalty. The range of circuit design strategies and clocking strategies allows necessary flexibility for designers to optimize for their particular set of constraints.

Table 5.3. Important Circuit Design Strategies and Clocking Strategies Matches

	Landauer	Bennett	Bi-D Shift	Uni-D. Bennett
Uni-Irrev.		√		
Uni-Gate Rev.		√		
Uni-Sub-Gate Rev.	√			
Bi-Gate Rev.				√
Bi-Sub-Gate Rev.	√		√	

Special Concerns for Reversible Circuits

When considering reversible circuits, especially when Bennett clocking is involved, there is a temptation to blur what inverse is being considered and used. Below is a case study of a particular 3-input, 3-output reversible gate.

The specific invertible function we're using as an example is a particular reversible majority "gate" (i.e., operation) having only 2 garbage bits (there are other such gates), which we will abbreviate rMAJ. This function is useful in reversible computing with traditional technology. In this discussion, it can be considered as one example of a 3-input, 3-output reversible gate. The rMAJ(x,y,z) operation can be defined as follows:

$$\begin{aligned} x' &= MAJ(x, y, z) \\ y' &= x \oplus y \\ z' &= x \oplus z \end{aligned} \tag{5.1}$$

where MAJ denotes the usual (1-output, irreversible) 3-input majority operation:

$$MAJ(x, y, z) = xy + xz + yz \tag{5.2}$$

This implements a permutation consisting of a 2-cycle composed with a 3-cycle:

$$(101, 110) \circ (011, 111, 100) \tag{5.3}$$

Its truth table can be seen in table 5.4. It is a universal gate for constructing arbitrary Boolean functions embedded within reversible functions with garbage outputs. We can construct AND because when x=0, note that x' = yz. We can also construct NOT, because when x=1, y' = ¬y.

Any gate that is universal in the above sense is automatically also universal for constructing arbitrary n-bit reversible functions with no garbage. The n-bit reversible

Table 5.4. rMAJ Operation Truth Table

x	y	z	x'	y'	z'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

Table 5.5. rMAJ^{-1} Operation Truth Table

x	y	z	x'	y'	z'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	0	1	1

gates comprise the symmetric group S(2^n) of permutations on 2^n items. For any finite group, the inverse G^{-1} of any reversible gate G can be formed by a certain power r of the original gate; in group theory, the exponent r is called the order of the element G. In this case, since the permutation comprises a 2-cycle and a 3-cycle, the order r of the permutation is LCM(2,3)=6, meaning that rMAJ^6 = I, and so the exponent r to invert the gate is 6-1 = 5.

Note, however, that rMAJ by itself is not equal to its own inverse rMAJ^{-1}, due to the presence of the 3-cycle. The truth table of its inverse rMAJ^{-1} can be seen in table 5.5

This can be described in Boolean algebra form, derived from Karnaugh maps, as:

$$\begin{aligned} x' &= x\neg z + x\neg y + \neg xyz \\ y' &= xz + x\neg y + \neg xy\neg z \\ z' &= x\neg z + xy + \neg x\neg yz \end{aligned} \tag{5.4}$$

Notice that these equations are very different in form from those that define MAJ!

This illustrates the general phenomenon that computing the inverse of a function may require a very different algorithm from computing the original function.

As another example, consider the difference between algorithms for multiplying ordered pairs of k-bit primes and for factoring the resulting products. Much of modern cryptography depends on the fact that the inverse of a function such as this is much harder to compute than the original function, using known algorithms. Functions whose inverse is exponentially more difficult to compute are called "one-way." However, we should note that a non-expanding function that is composed entirely of reversible gate operations can never really be a one-way function, since one can always run the original gate sequence backwards with identical time complexity. So, for example, if there is a polynomial-time implementation of non-expanding multiplication of prime number sequences using reversible gates, then FACTOR is in P. Anyway, that is a tangent.

Now, we can reduce the total number of gates required to implement equations (4) from 26 to 18 (3 NOT, 9 AND, and 6 OR) through common subexpression elimination:

$$
\begin{aligned}
t_1 &= \neg x \\
t_2 &= \neg y \\
t_3 &= \neg z \\
t_4 &= x t_3 = x\neg z \\
t_5 &= x t_2 = x\neg y \\
t_6 &= t_1 y = \neg x y \\
x' &= t_4 + t_5 + t_6 z = x\neg z + x\neg y + \neg x y z \\
y' &= xz + t_5 + t_6 t_3 = xz + x\neg y + \neg x y \neg z \\
z' &= t_4 + xy + t_1 t_2 z = x\neg z + xy + \neg x \neg y z
\end{aligned}
\tag{5.5}
$$

Even simpler logic for this function might exist that takes advantage of the MAJ gate more directly, rather than always through its use in AND and OR. However, we will not take the time to investigate that here.

Now, rMAJ itself can be implemented directly from its equations (1). In QDCA, XOR requires 3 MAJ gates and a NOT gate, while AND uses 1 MAJ gate, so the equations (1) translate to 8 MAJ gates altogether and 2 NOT gates, while equations (5) involve 13 MAJ gates and 3 NOT gates.

Let's go ahead and draw the Boolean circuits for rMAJ and rMAJ^{-1}. They can be seen in figures 5.6 and 5.7.

We have several options for clocking this circuit. We could use 3-phase wave clocking with extra garbage outputs (not shown), in which case the four stages form a chain 1 clock cycle deep. Or, we could use retractile clocking, in which case the initiation interval is 10 ticks long, where a tick is the time to latch or unlatch a stage. After the stage (not shown) that is biasing the pipeline input (PI) stage is latched, The 10 ticks are: (1) Latch PI. (2) Latch stage 1. (3) Latch stage 2. (4) Latch stage

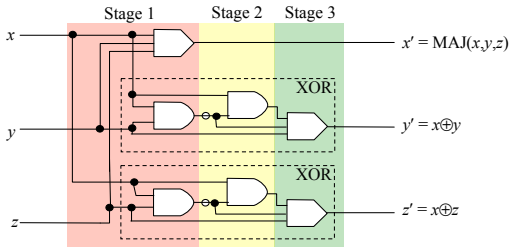


Figure 5.6. Implementaion of rMAJ using QDCA primitives. The wedge-shaped icon represents a QDCA majority gate. The AND gate icon represents a QDCA majority gate with one input tied to a constant 0 (e.g., a cell with trapped charges). The bubble represents a QDCA bidirectional NOT gate. Note this circuit is 3 logic levels deep if you count the MAJ gates, but don't count the NOTs.

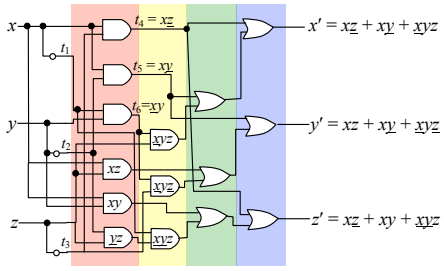


Figure 5.7. Implementaion of rMAJ^{-1} Again, the AND and OR icons represent majority gates with constant inputs. As with rMAJ, the logic can be implemented with 4 stages of logic in which there is only 1 level of gates per stage, not counting the inverter bubbles as gates. The timing options for this structure are therefore identical to the ones for rMAJ discussed in the caption of Figure 5.6.

3. (5) Latch PO, the pipeline output register. The data is held there until someone else unlatches it, which we won't discuss in this sequence. (6) Unlatch stage 3. (7) Unlatch stage 2. (8) Unlatch stage 1. (9) Unlatch PI. (10) Unlatch other circuit (not shown) that is biasing PI, and maybe simultaneously re-latch a new input to it from another path. After all this, PI is now ready to accept a new input and the cycle can begin again. The throughput is $\frac{1}{f}^{th}$ of the 3-phase wave-clocked approach. If we wished, since each stage computes an invertible (if expanding) function of its inputs,

we could insert additional pipeline registers in the middle to improve throughput, but this would also add more interconnects. Also, there is no guarantee that the reverse of the forward stages can be implemented easily in a single level of logic.

Chapter 6

Architectural Approaches

The QDCA design work done prior to 2006 dealt with irreversible designs. This work included proof-of-concept layouts of a complete (though simple) microprocessor and an accompanying memory; an FPGA cell; several other memory cell designs; and a processing-in-memory like execution model for QDCA. While this work was all irreversible logic, it revealed several design principles that can also be exploited by designers interested in reversibility. The basic principles revealed include:

- Layout=timing: Because there is a limited number of QDCA cells that can be controlled by a single clocking wire (determined by the specific QDCA implementation and clocking circuit implementation), there is a more direct connection between physical space and the time for a QDCA signal to travel across it. While conventional circuit designers worry about things like clock skew, the problem in QDCA is much more explicit.
- Processing-in-wire: Unlike CMOS devices, the logic and communication are done by the same device. Because of this, the strict separation between combinational logic units and wires does not need to be maintained.
- Fine-grained Pipelining: The natural implication of the first two items is the proclivity of QDCA toward fine-grained pipelining. Because communication across a distance requires several clocking "zones" and because the wire can contain the logic, this naturally leads to implementing circuits with fine-grained pipelining.

The work that lead to the identification of these three principles assumed the following:

- Zone floorplanning
- Zone-type clock signal
- Four phase clock
- Data on QDCA layer cannot influence the clock layer

- Physical crossovers available
- Both 90 degree cells and 45 degree cells available

Changing these assumptions does not necessarily invalidate the principles, but different assumptions may lead to the identification of more domain specific design principles as well as illuminating true QDCA-wide design principles. For instance, the fine-grained pipelining that is so obvious under this set of assumptions is less obvious when Bennett clocking, for instance, is assumed.

Obviously, a system can be designed with many components each clocked in a different manner according to the constraints of the component. In order to ensure the reversibility of the system up to the desired level, the interfaces between components needs to be considered carefully to eliminate any unintended erasure of information. Four design examples are discussed below. The retractile cascade fully reversible pipeline was first explored by Younis and Knight for "traditional" reversible computing. There are a few extra challenges for QDCA that are discussed. The mirror circuit fully reversible pipeline is another traditional reversible approach that that tends to be less space efficient but improves time efficiency. The collapsed Bennett model was first explored by Murphy and DeBenedictis. It is a potential hardware implementation of Bennett's 1989 algorithm. Finally, a partially reversible pipeline is discussed that maximizes throughput while selectively using reversibility to minimize dissipation.

Retractile Cascade Fully Reversible Pipeline

This is the reversible pipelining method that was invented by Younis and Knight '93 for application in a CMOS technology context. As we'll see, it is easily adapted to the QDCA technology paradigm.

The general picture of the bidirectional-retractile pipeline architecture is as can be seen in figure 6.1. Notice that the computational sections are Bennett clocked, meaning any circuit design style will be reversible.

In QDCA, if the wire lengths are short and there are not too many levels of logic per stage, then it should be possible to combine some of these steps together, and obtain a 4-step cycle, as shown in figure 6.2. However, if there are several levels of logic per pipeline stage, then it may not be safe to depend on the entire pipeline stage simultaneously and adiabatically converging to the exact desired state. If the number of steps in the cycle are held constant, one could expect that the adiabaticity would be reduced at a given frequency if the pipeline stage is deeper, due to the increased intrinsic propagation delay along the depth of the circuit. Multiple clocking stages can be introduced into the computation segment of the pipeline, though.

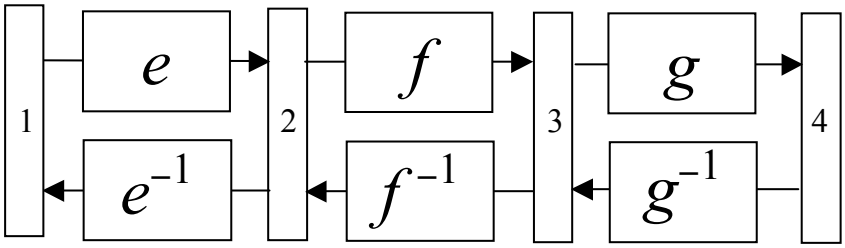


Figure 6.1. Bidirectional-retractile scheme for reversible pipelining invented by Younis and Knight. The vertical rectangles represent pipeline registers. The labels e, f, and g represent reversible functions; this particular pipeline is intended for computing the overall function $g \circ f \circ e$, or its inverse, depending on which direction it is operated in. The boxes represent retractile circuits for computing the functions shown, where the direction from inputs to outputs is indicated by the arrows. The normal sequence of operation is as follows. Assume that block e has just produced its output, which has been latched into pipeline stage 2. Now, the complete cycle until new input arrives on stage 2 is as follows. (1) f is operated in the forwards direction, and meanwhile, e is retracted, and e^{-1} is operated. (2) f's output is latched into stage 3, and meanwhile, the contents of stage 1 are unlatched reversibly under control of e^{-1}. (3) g is operated in the forwards direction, while f is retracted, reversibly clearing its contents, and f^{-1} is operated, resupplying an image of stage 2's contents, and e^{-1} is retracted. (4) The stage 2 contents are unlatched reversibly under control of f^{-1}, and the stage 4 contents are latched. Now stage 2 is empty and stage 4 contains valid data. Meanwhile, stage 1 is being written with a new valid input. (5) e is charged, g discharged, g^{-1} charged, (6) stage 2 is charged, stage 3 discharged. After this we are back to the initial conditions and can begin a new cycle.

and output latched)

In effect, the pipeline stages themselves can be internally pipelined. This leads to some slightly more complicated equations than the standard pipelining throughput and delay equations. However, it does not necessarily dramatically increase the number of clocking signals required.

Each pipeline stage contains the pipeline in table 6.1. The "compute release" (i.e.

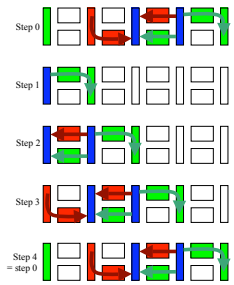


Figure 6.2. Four-step timing sequence for bidirectional pipeline. Notice the latency is only 1 tick per stage. This design requires 12 distinct clocks (12 types of clocking regions). For this design to work, it must be possible to adiabatically charge up the logic region and the pipeline register within a single transition time.

Table 6.1. Pipelining within a Pipe Stage

Stage	Latch	Compute	Drive	Compute Release Uncompute	Drive	Uncompute Release
Delay	L	Cn	D	Cn	D	Cn
Step	1	2	3	4	5	6

the Bennett clock is retracting across the compute stage) and "uncompute" (i.e. the Bennett clock is csacading across the uncompute stage) can be overlapped at time four since the effects are controlled by different clocking wires.

Figure 6.3 shows an example of a fully reversible retractile cascade pipeline with four clocking wires for each compute stage. Notice that the uncompute stage of any pipeline stage i uses the same clocking signals as the compute stage of pipeline stage $i + 1$.

Throughput

The initial delay (the time to fill the pipeline) can be described by the following equation:

$$t_{fill} = (3Cn + 2D + L)S \tag{6.1}$$

Table 6.2. Symbol Meanings for Retractile Cascade Pipeline Equations

Symbol	Meaning
C	Delay through a single computation stage
D	Time a stage needs to be locked in order to successfully drive a neighboring stage.
L	Time required to latch a signal
S	Number of pipeline stages in the system
n	Number of clocking wires in a computation stage

Notice that in general C = D = L = time for a signal to be either latched or released (whichever is greater).

where n is the number of compute regions in one pipe stage (i.e. the number of wires used to control the Bennett clocked region), C is the delay for one compute section to switch from unlatched to latched, D is the time needed to allow the result of the compute stage to drive the memory stage, L is the time for the memory latch to switch, and S is the total number of pipeline stages.

Throughput depends on when the next input enters the pipeline. If it is done at the soonest possible moment, the throughput will be:

$$throughput = \frac{1}{2Cn + D} \tag{6.2}$$

However, this will may result in a larger number of unique clocking signals being required. If the input is held so clocking signals can be reused, the throughput equation changes to:

$$throughput = \frac{1}{3Cn + 2D + L} \tag{6.3}$$

Number of Unique Clock Signals Required

The number of uniquely clocked pipeline stages can be determined by finding the least common denominator of the number of cycles for one stage of the pipeline to

begin driving the next (n+1, where n is the number of computing wires and 1 allows for the time needed for the next latch to be driven) and the number of cycles for an input to move completely through one stage of the pipeline ($3n + 2 + d$, where n is the number of computing wires and 2 allows for the 2 driving stages needed over the course of computation in one stage, and d is the number of cycles delayed between the completion of the processing of one input and inserting new data into the pipeline).

The least common denominator can be determined by multiplying the two numbers together and dividing by the greatest common factor. The greatest common factor can be found using the Euclidean algorithm.

The total number of unique clocking signals required can be calculated by:

$$signals = (1 + lcd(n + 1, 3n + 2 + d)) * (1 + n), \tag{6.4}$$

where $lcd()$ is the least common denominator function, and the other symbols are consistent with the above discussion. The $1 + n$ is the number of clocking signals per pipeline stage, where this 1 is the number of wires needed for the latch associated with the pipe stage, and n is the number of wires in the compute stage. The wires in the uncompute stage are repeat signals of the next stage and should not be counted twice. However, the final uncompute stage has nothing to overlap with. The 1 in $1 + lcd(n + 1, 3n + 2 + d)$ accounts for these wires as well as the latch for the final output that is not considered to be part of any other stage.

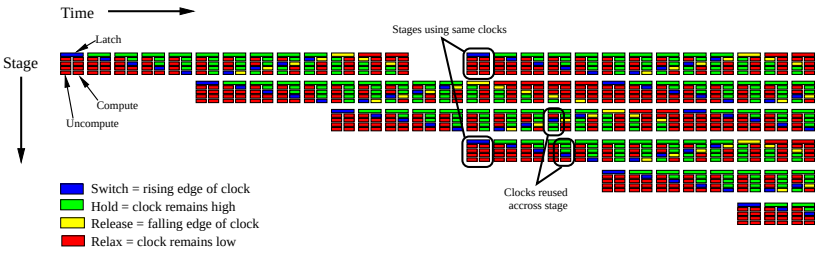


Figure 6.3. Clocking signals needed for a pipeline with a compute phase with four clocking wires. Notice that sets of clocking signals can be repeated in whole between stages (e.g. at time 16 between stages 1 and 4) and across stages such as between the uncompute section of stage i and the compute section of stage i+1

Mirror Circuit Fully Reversible Pipeline

For the mirror circuit approach, reversibility comes from the circuit rather than the clock. One approach is to simply transform the irreversible gates into reversible "expanding" gates by fanning out their inputs and replicating them at the output. The structure shown in figure 6.4 is a portion of a pipeline for computing an iterated sequence of maps:

$$\begin{array}{ccccccccc} & & e & & f & & g & & h & \\ w & \rightarrow & x & \rightarrow & y & \rightarrow & z & \rightarrow & a \end{array} \tag{6.5}$$

Each of the functions e, f, g, h represents a different invertible map, each of which may be either an expanding non-onto function (with more outputs than inputs), a non-expanding bijective function (with the same number of inputs and outputs), or a conditionally-reversible contracting partial function (with more inputs than outputs). Thus, the form of the functions $efgh$ is completely unrestricted, apart from the condition that they be invertible, and that (if they are only partial functions, in other words only conditionally reversible) their preconditions are satisfied by the actual data.

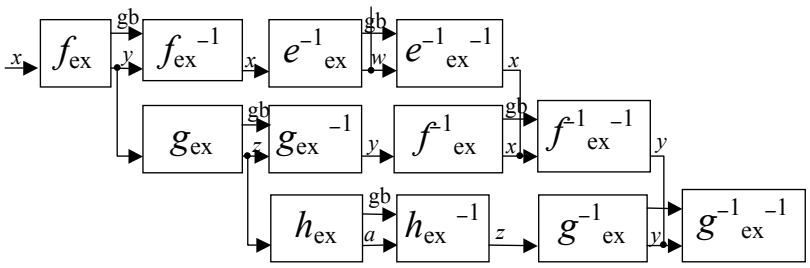


Figure 6.4. Reversible pipeline based on mirror circuits. The structure is optimized to minimize latency.

Each of these functions f gets implemented in pipelined fashion by embedding it into an expanding function f_{ex} (generally with more outputs than f) which is composed directly from the immediately-available pipelineable reversible primitives, some of which may be expanding operations (e.g., AND with both inputs copied, or MAJ with all 3 inputs copied). The result is that we get both the desired output $y = f(x)$ and also some garbage data ("gbg" in the figure). To get rid of the garbage data, we construct the inverse f_{ex}^{-1} of f_{ex} by using the mirror-image circuit, leaving us with just x and y. All this is particularly easy to do in QDCA due to the fact that its reversible gates can be input-consuming, and information can flow through them in either direction, depending on the clocking sequence.

After obtaining y, we would like to get rid of x. This is done later on by constructing the inverse f^{-1} of the function f and embedding it into an expanding circuit, f_{ex}^{-1}. Note that $f_e^{-1}x$ is in general a totally different circuit from f_{ex}^{-1}! The difference is that f_{ex}^{-1} takes y and produces x plus garbage, whereas f_{ex}^{-1} takes y plus garbage and produces x. There is in general no simple, straightforward transformation from one of these circuits to the other! In fact, if f is one-way, then f_{ex}^{-1} will be an exponentially larger circuit than f_{ex}^{-1}.

After obtaining a new copy of x plus garbage, we then decompute both copies of x, and the garbage by running them through $f_{ex}^{-1^{-1}}$, which is simply the mirror-image circuit of f_{ex}^{-1}. We are now left with only y, which can be taken through the next function in the desired sequence.

To reduce the latency, notice that in figure 6.4 we have interleaved things so that we can begin the next function g as soon as we have obtained y for the first time. After running g_{ex} and its mirror g_{ex}^{-1}, we use y as input to f_{ex}^{-1}, producing the garbage that is required in order for us to be able to use $f_{ex}^{-1^{-1}}$ to decompute the copy of x that is output from the $e_{ex}^{-1}/e_{ex}^{-1^{-1}}$ sequence which is used to decompute w, which is the original input to the previous stage e_{ex} (not shown) that produced x originally. This elegant structure emerged from a discussion between Mike Frank and Erik DeBenedictis.

The advantages of this structure are as follows: The input-output latency is almost as low as possible; the only time overhead here is perhaps the extra propagation delay required for the signals to travel longer distances necessitated by the hardware overheads compared to an irreversible solution. The throughput is as high as possible since data can be propagated through the pipeline in a minimal 3-phase wave pattern; the only lower bound on the wavelength is the cell size, or the minimum width of the clocking wires. The clocking pattern is columnar and highly regular. Only 3 clocks are needed if crossovers are provided at the hardware level. The number of crossovers required is not especially large.

The only significant disadvantage of this design style is that it requires somewhat more than 4 times the hardware of an irreversible solution, and about twice the hardware of the bidirectional-retractile approach. This is due to the use of the mirrored, inverted, and mirrored-inverted versions of each function, in addition to the original function. This is in contrast to the retractile approach where the mirrored versions of the circuits reuse the same hardware and merely retract the clocking sequence.

The total number of gate-operations performed is about the same in the two techniques, since they both require roughly four gate operations for each gate in the original circuit (do expanding function, undo it, do expanding inverse function, undo it). However, one should keep in mind that for either design style, if the original function has a one-way characteristic, then the overhead to implement the inverse functions can be significant, and in such cases, it may be preferred to operate irreversibly, or to keep around the original inputs as garbage, even if the original function was invertible and thus garbage-free operation was possible in principle.

One can think of the mirror-pipelined circuit shown in figure 6.4 as an "unrolled" version of the same operation sequence that is shown in figure 6.2. Similarly to the situation with loop unrolling for microprocessors, more redundant code (here, circuitry) is required, but the parallelizability of the code is improved, and the achievable throughput is greater.

Collapsed Bennett

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. Charles 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, uncalculate the intermediate result so it does not have to be stored. Bennett's algorithm is an optimal ordering for the computing and uncomputing the segments. Figure 6.5 shows an example of the order of computations and uncomputations for an algorithm broken into eight segments.

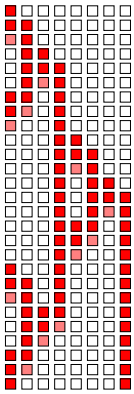


Figure 6.5. Bennett's algorithm divides an algorithm into stages (8 stages in this example) and selectively computes and decomputes them to store the least amount of data necessary to maintain the reversibility of an irreversible algorithm.

Erik DeBenedictis introduced the beginnings of a potentially implementable model that collapses Bennett's reversible algorithm tree into a single level of 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). I further developed this model and designed a simple ripple-carry adder to demonstrate the

proposed operation.

A schematic of the components can be seen in figure 6.6. 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 can be similarly disabled.

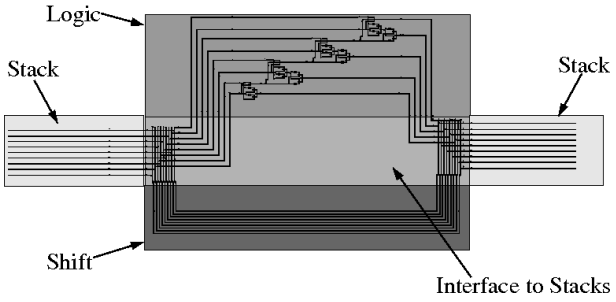


Figure 6.6. The regions of the collapsed Bennett layout include two stacks, a logic or computational area, a shift area that allows data to be transferred between the stacks, and an interface between the stacks and the logic and shift regions.

The collapsed Bennett model operates as follows:

1. Initial input begins at the top of the left stack
2. A cascade shaped Bennett clock moves across the logic section (starting at the left, the clock goes high and stays high as the clock front travels to the right).
3. Result is latched at the top of the right stack
4. The cascade is retracted, decomputing the logic (leaving the results latched)
5. A set of results is shifted to prepare for the next stage

   a) Right stack is popped and the value is pushed onto the left stack (via the shifter) and is ready to be the input in the next cycle

   b) Left stack is popped and the value is pushed onto the right stack (via the shifter) and is ready to be decomputed in the next cycle
6. This process is repeated with the shifting determined by Bennett's 1989 algorithm [2]

A simple adder is shown in this section to illustrate the operation of the layout and demonstrate the required interfaces between the computation and storage. One can imagine sandwiching an entire processor between two stacks in this manner for a general purpose reversible processor.

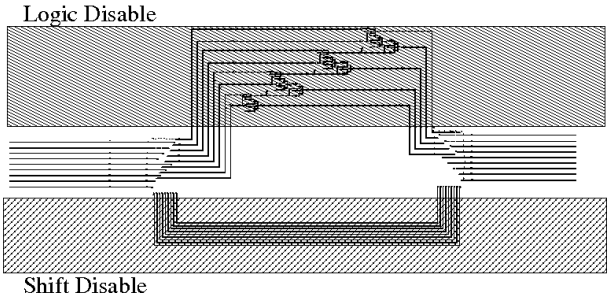


Figure 6.7. There are two disable sections in this layout. The top area disables the logic, while the bottom area disables the shift. While disabled, the QDCA cells have no value and do not contribute to the computation of any nearby cells.

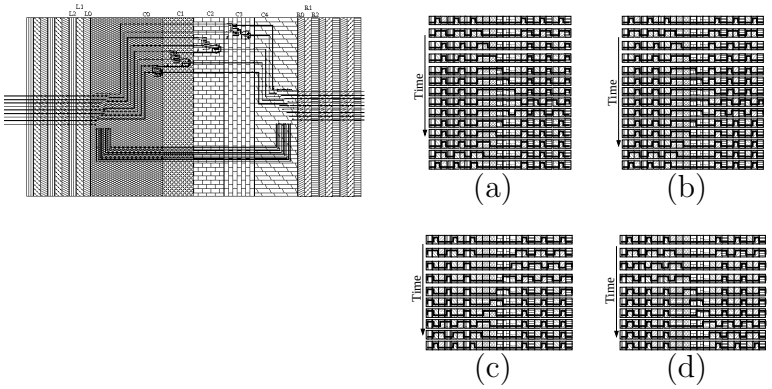


Figure 6.8. Clocking signals required for four modes of operation of collapsed Bennett clocking layout: (a) Compute, (b) UnCompute, (c) Shift Left, (d) Shift Right

Partially Reversible Pipeline

This section reports the proposed approach to obtain high performances in power consumption and throughput. The design is divided into computational and memory stages, the computational stages are clocked with the Bennett scheme and do not dissipate power whereas the memory stages are used to introduce multiple stages in the circuit and therefore to increase the computation speed with pipelining. We consider a circuit as being partitioned into N stages where each stage has i_j inputs and o_j outputs. Obviously $i_j = o_{(j-1)}$.

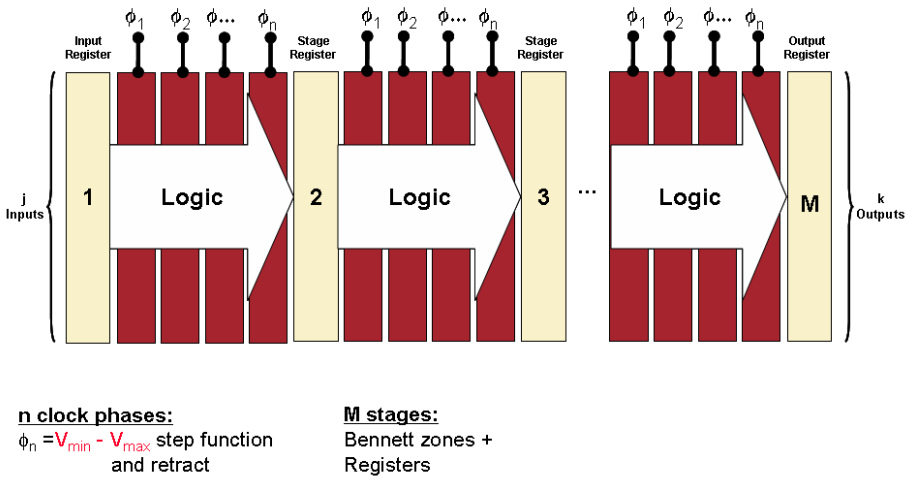


Figure 6.9. Proposed pipelined approach: Top view

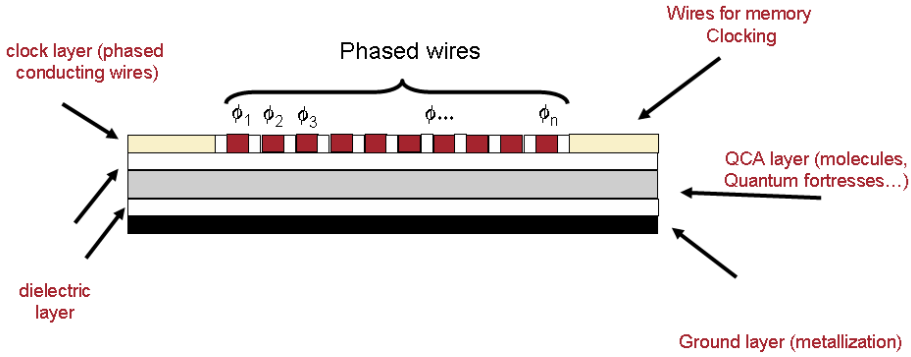


Figure 6.10. Proposed pipelined approach: Cross Section

Computation Stages

In this section we describe the clocking scheme for the combinational parts of the circuit. Before describing the adopted clocking scheme we provide some background on the clock distribution techniques and clocking schemes proposed for QDCA.

We use the distribution mechanism introduced in [15] where a E field generated on a layer of metallic wires above (or below) the QDCA layer controls the tunneling inside of the single QDCA cells: since the cells are not directly connected to the clocking circuitry, this approach has the major advantage of allowing very high level of integration for the QDCA cells that can be implemented with molecules. Moreover the continuous transition of the E field on the leading edge of the wave reduces the possibility of kink. The traveling E field of [15] is generated by providing each of the wires with a voltage phase shifted from the neighbor of a $\Delta\phi = \frac{\pi}{2}$ having a conducting (ground) layer on the other side of the QDCA layer. [15] shows that the E field generated with such circuit can assume a sinusoidal shape thus allowing for the so called wave clocking i.e. the E field or more accurately, the component of the vector $\vec{E}$ in the z direction E_z, in the QDCA layer is described by the wave equation [21]:

$$E_z(x,t) = E_0 cos(\kappa x - \omega t)$$

With the wave clocking the switching of the cells occurs only on the edge of the wave therefore implying a clear directionality in the propagation and virtually a null probability of kink. This traveling wave is also referred to as "computational wave" [5] and is a space continuous implementation of the classic four phases- four zones clocking scheme introduced in [22]. The Landauer clocking [42] is another name of this scheme and it refers to the fact that the traveling wave performs always a copy function of the data before deleting it (as for example in the QDCA wire).

The maximum performance in terms of speed is related to the maximum applicable clock speed and is a consequence of the physics of the tunneling between quantum dots. In order to maintain the adiabatic solution of the Scrhödinger equation it is required that the switch time t^* for the E field on a cell to be higher of the tunneling speed between quantum dots [22] consequently the fastest applicable clock period to a cell is

$$T_l = 2t^*$$

and therefore $\omega \leq \omega_0 = \frac{2\pi}{2t^*}$.

The constraint on the maximum applicable period will be used in a later section to measure the throughput of Landauer clocking for the case studied, generally for o outputs $Tr = \frac{o}{2t^*}$. Moreover the traveling E_z wave is also characterized by its phase velocity

$$v = \frac{\omega}{\kappa} = \frac{\lambda}{2t^*}$$

, therefore for a given t^*, v and λ are inversely proportional: a faster traveling wave implies a a lesser populated pipeline (i.e. a smaller number of data present in the

pipeline at a given time t) and vice versa.

In this paper we use the wave clocking distribution mechanism of [15] to reproduce the Bennett clocking scheme in the computational stages of the pipeline therefore our computational wave [5] is split in Bennett clocked zones [20] [21].

The Bennett scheme has two steps, in the first step it performs the computation on the inputs and propagates to the outputs without ever deleting the intermediate results until the outputs are generated and latched, in the second step the clock "backs off" i.e. the release of the cells starts from the outputs and goes back to the inputs with a eventual release of the whole circuit. The second step is what can be called the de-computation as in this stage all the intermediate results latched in the QDCA cells are released, this process does not delete any information as it is easy to show that any released cell has always at least one cell on its left that carries its same information, therefore when the cell is released there is always a copy of its information, this copy process is what saves the information to be thermalized as shown in [42].

Circuits implemented with Bennett clocking therefore do not dissipate energy provided that the input and output are copied while the speed of computation is a consequence of the time required to the signals to propagate back and fort in the circuit.

Moreover circuits clocked with the Bennett scheme have also an important advantage that the circuit does not require any modification to the layout to avoid deleting the information included in the inputs as it would happen if the inputs needed to be propagated to the outputs as shown in Figure 6.11. Figure 6.11 also shows that if the computation is done in an irreversible way i.e. without preserving the information of the inputs, the power dissipated when losing a bit of information is almost equal to the kink energy $E_{diss} \simeq E_k \gg KTln2$. The value of the dissipated energy is obtained from the non equilibrium equation i.e. a set of first-order differential equations for the coherence vector of QDCA cells in contact with the thermal environment [21].

The Bennett scheme can be implemented through the wave clocking by applying suitable signals ($\Phi_1, ...\Phi_n$ shown in Figure 6.13) to the buried wires in order to have the propagation of a computational wave that keeps the the cell locked and therefore the intermediate results until the output are saved in the memory stage and after releases the cells in the opposite direction following the two steps procedure discussed above. The temporal sequence of the clock value throughout the Bennett stage is shown in Figure 6.12 where T represents the time period of the stage.

The signals applied to each buried clocking wire are shown in figure 6.13; as can be seen the phase Φ_1 of stage $j+1$ releases the information contained in it while the memory stage is switching and the the phase Φ_n of stage j is in hold phase. This should allow an asymmetric interaction of the memory cell that should allow to propagate always the value on the stage j while not deleting the information in stage $j+1$ when this has the same value on the one in stage j.

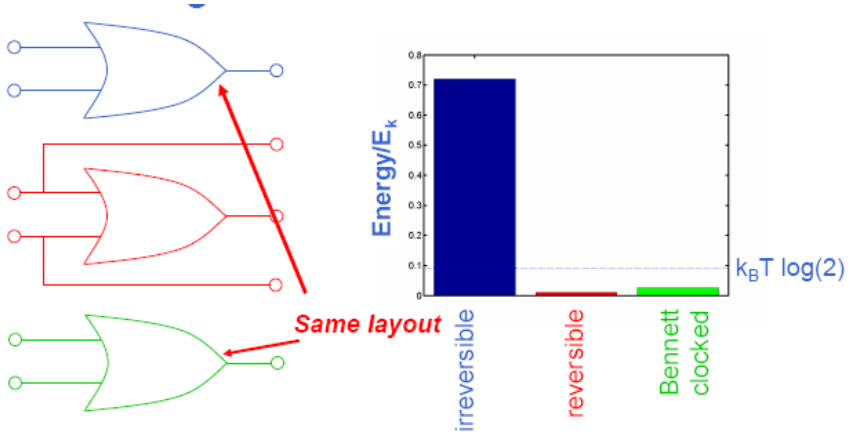


Figure 6.11. Advantages of Bennett clocking: area and power consumption

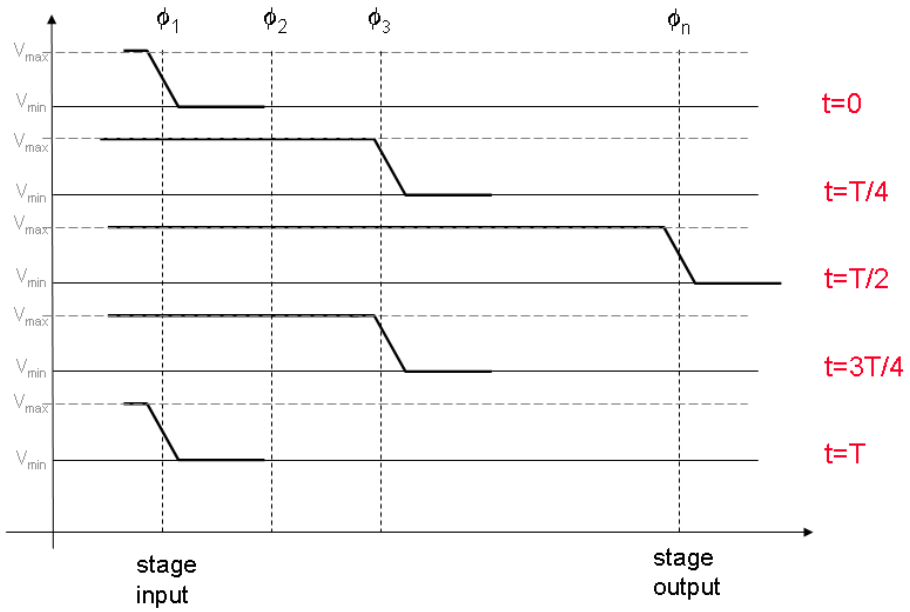


Figure 6.12. Clocking wave for the Bennett scheme

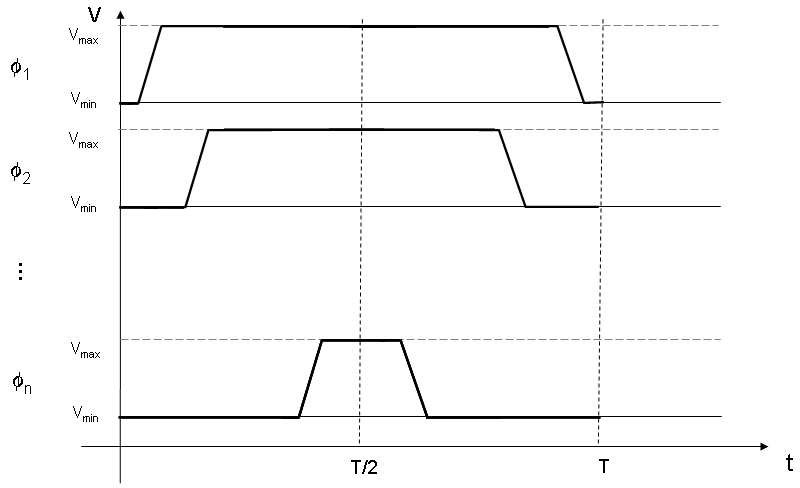


Figure 6.13. Clock signal to the buried wires

With this clocking scheme data are output from the stage at every period $t = T$ that is required to propagate the wave forward to the outputs and back to the inputs. An approximate analytical expression of $E_z(x,t)$ is the following

$$E_z(x,t) = E_0\left(1 - u_0\left(\frac{x}{\lambda_c} - tri_T(t - \frac{T}{2})\right)\right)$$

where u_0 is the Heaviside step function tri_T is the triangular function of width T and λ_c is the width of the Bennett-clocked region. The use of u_0 represents an approximation on the use of smooth transitions of duration t^*.

As discussed above, to keep adiabaticity the switch time on a cell must be at least t^* therefore, considering d the lateral size of a QDCA cell, and $N = \lambda_c/d$ the width of the Bennett-clocked region in number of cells, we have

$$T_b = \frac{2\lambda}{d}t^* = 2Nt^*$$

Memory Stages

The memory stages are a single buffer register used to separate the different stages of the pipeline, their implementation is rather straightforward as they could in principle be implemented with a single vertical row of QDCA cells or the minimum number of cells related to the achievable pitch of the clocking wires. The memory stages provide the inputs to the Bennett clocked zones and latch their outputs; as shown in Figure 6.14 the clocking signal is sinusoidal with the same period T of the Bennett clocking scheme. As discussed above when describing the clocking scheme of the computation stages, an asymmetric interaction on the memory cells should provide a left to right

directionality of the information propagation while also avoiding the deletion of the information coming from the right when it represents the same binary value of the one coming from the left. A sketch of the propagation in the two opposite directions is shown in Figure 6.15 where two opposite values are interacting on the memory cells in the middle. Since the cell on the left locks its value (attains the hold phase) earlier than the one on the right, the Coulombic interaction (quadripole moment) on the memory cell is stronger and therefore it should cause the memory cell to assume its value. It is evident that the proposed asymmetric interaction requires a very accurate synchronization of the clocks, that could be beyond the reach of a feasible implementation: therefore if the overlapping of the switch zone of the memory and the relax zone of the stage $j+1$ would show too many difficulties the waveforms of figure 6.13 and 6.14 could be modified to avoid such overlapping. Obviously this modification would imply that the decomputation of data in the stage $j+1$ would always delete all the inputs and not only those that are different from the outputs of the stage j.

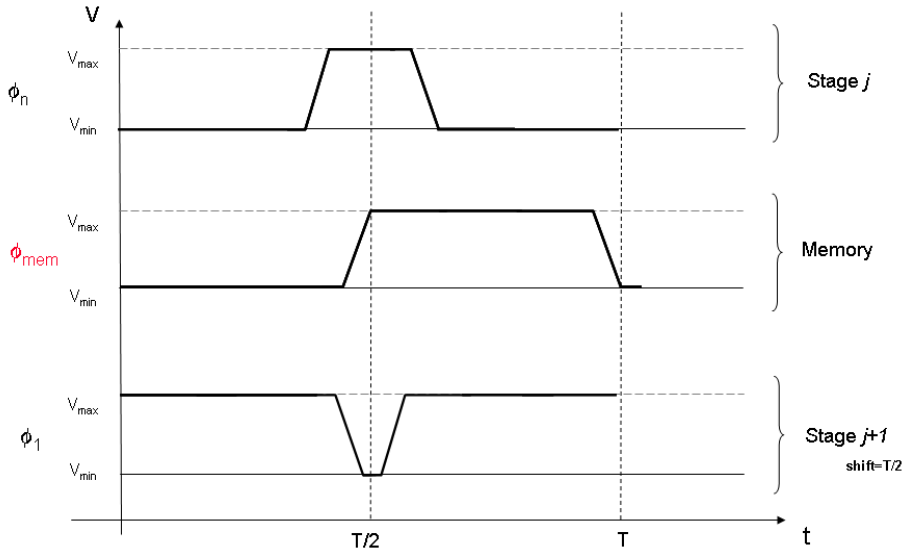


Figure 6.14. Clocking signal for the memory zones

Performance evaluation

The performances of the proposed solution are evaluated both in terms of throughput and power consumption The following considerations apply:

- the stages allow introducing pipelining and therefore increase the throughput
- the number of stages increments the power consumption as it increases the amount of discarded information

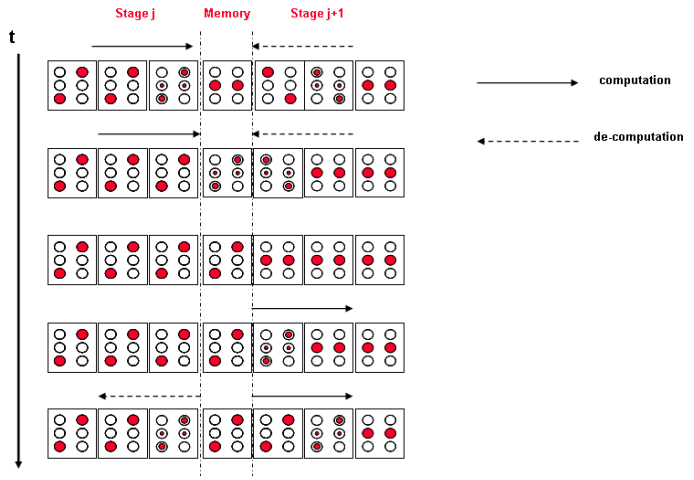


Figure 6.15. Asymmetric interaction on the memory cell

Therefore the allocation of the amount of stages is the outcome of a trade off among computing performances and power consumption.

For what is related to the throughput we already showed in a previous section that the period of a Bennett stage is $T_b = 2Nt^*$, therefore for a pipelined combinatorial circuit (no feedback loops are considered) composed of M stages and o outputs the throughput is

$$T_r = \frac{o}{T_b} = \frac{o}{2Nt^*}$$

whereas, as shown in Figure 6.16 the initial latency is proportional to $T_b/2$:

$$L_b = \frac{MT_b}{2}$$

For the same M staged pipeline the power consumption as a function of time $P(t)$ can be expressed as follows:

$$P(t) = E_{diss} \cdot \sum_{j=0}^{\infty} \sum_{i=0}^{M-1} K_i(t)\delta(t - \frac{jT_b}{2}) \tag{6.6}$$

where $K_i(t)$ is the number of inputs on stage i that change value at time t, E_diss is the energy dissipated (thermalized) when a bit is deleted on the stage registers. The time varying value of $K_i(t)$ accounts for the random time variability of the data in the pipeline on the memory stage i and it could be considered being in average equal to half of the bits stored in the memory. The power dissipation of a circuit is therefore spatially localized on the memory stages and is a time varying function composed of

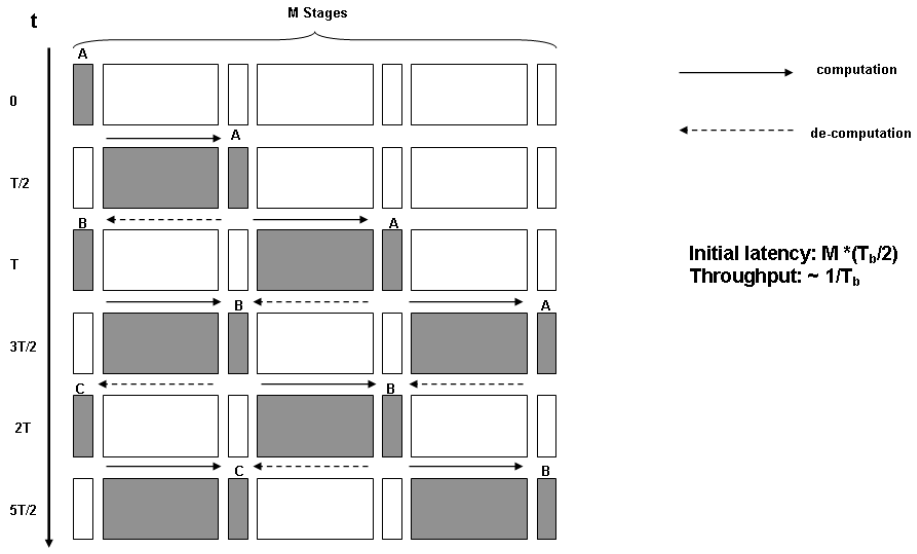


Figure 6.16. Pipelined stages with Bennett clocking

a train of pulses accounting for the dissipation occurring at the discrete time instants $t = jT/2$ (with j integer) on the memory stages, Figure 6.17 shows a possible shape of the P(t) not related to a specific circuit implementation. Note also that, as can be seen in Figure 6.16 at each $t = jT/2$ the power dissipation occurs only on $\lfloor M/2 \rfloor$ i.e. at the same the deletion of data occurs only in that half of the memory stages where the computing and the decomputing waves meet. Consequently, at a given time $t = nT/2$ the actual number of coefficients $K_i(t) \neq 0$ is $\lfloor M/2 \rfloor$.

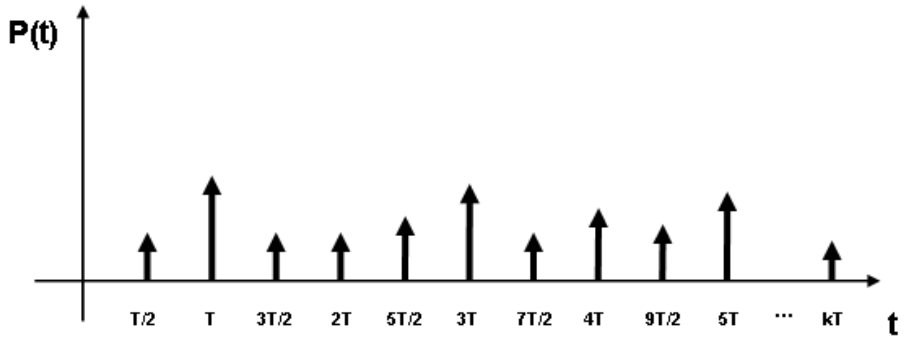


Figure 6.17. Possible shape of $P(t)$

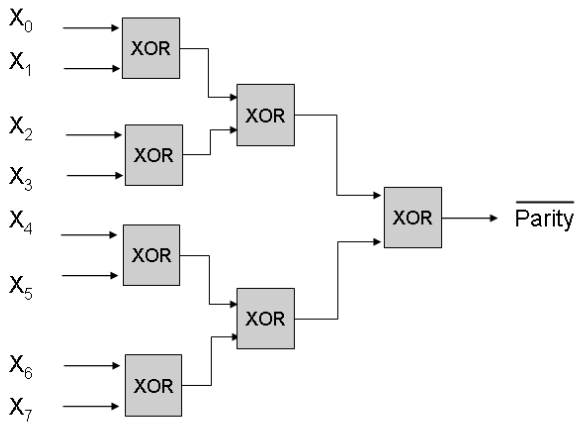


Figure 6.18. Case Study: XOR tree parity checker

Case Study: parity checker XOR tree

The size of the zones clocked with the Bennett scheme can vary from a minimum of two QDCA cells size (the one cell case would be degenerated into a Landauer scheme and the clock would be a traveling wave) to the size of the whole circuit (thus becoming a purely Bennett clocked circuit). As stated previously, we expect that by increasing the size of the zones the throughput and the power consumption would decrease.

It should be noticed that the contribution to power dissipation strongly depends on the circuit layout: a circuit composed of only wires and inverters and thus composed of only reversible building blocks, would have the best performances with Landauer clocking as no information would be deleted apart from the I/O whereas a circuit comprising Majority voters would require the introduction of a Bennett scheme to reduce the dissipation due to the deletion of information. To have some advantage in power dissipation a Bennett clocked stage should have a number of MVs big enough such that the number of bit of information that would be deleted in that stage using Landauer clocking is significantly higher than the number of inputs deleted of the Bennett stage. Note that the number of bits deleted in a stage is not necessarily equal to the MVs as shown in the next example.

We show a simple example of the proposed approach. An M stages binary tree composed of XOR gates generates the parity bit for $w = 2^M$ inputs. We report an analysis of throughput and power dissipation of the XOR based parity bit generator by using the previously introduced formulas with the simplification of the worst case scenario. The performances in throughput in this case can be evaluated as follows:

considering that only one output is generated at each T_l the throughput for the Landauer scheme is

$$Tr = \frac{1}{T_l}$$

. With the Bennett scheme the performances depend on the size chosen for each zone because this decides the value of $T_b = 2Nt^*$, in the following analysis we will consider a minimum value of N corresponding to the size (in the x direction) of an XOR gate and we will evaluate the performances as a function of integer multiples of N. Again since $o = 1$ the throughput is

$$Tr = \frac{1}{T_b}$$

For what is related to the analysis of power consumption, since the XOR function (XOR(A,B)= (NOT(A) AND B) OR (A AND NOT(B))) comprises two AND and one OR, as shown in Figure 6.19 there is no combination of the inputs that causes the presence of different inputs to the three gates at the same time, therefore in the worst case for each computed output the power dissipation is with Landauer clocking $2E_{diss}$. For Bennett clocking also the maximum amount of power dissipated in a XOR gate is $2E_{diss}$ since in the worst case (no overlapping of the waveforms on the memory stages) both inputs will be deleted.

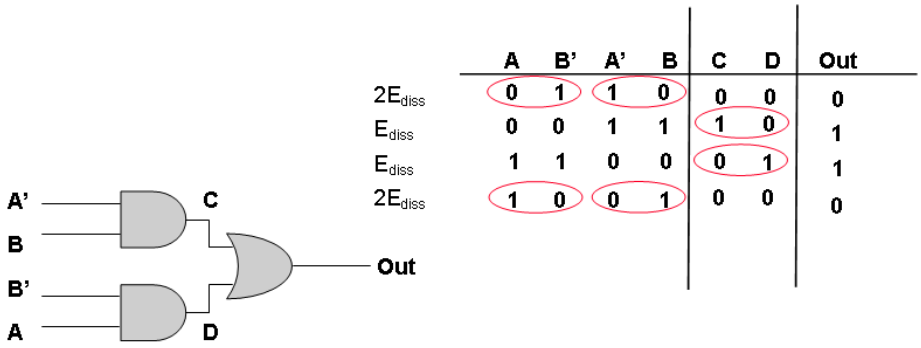


Figure 6.19. Dissipation in the XOR gate

To compare the power performances of a M stages binary XOR tree clocked with the Landauer and Bennett schemes also the following assumptions and definitions are used:

1. the dissipated energy of a thermalized bit of information is considered equal to the kink energy, i.e. $E_{diss} \simeq E_k$
2. the number of stages of the XOR tree is k;
3. the number of stages of the pipeline is M;

4. the number of stages of the XOR tree per pipeline stage is $c = \frac{k}{M}$

5. the kink energy value is $E_k = 3.14577 \cdot 10^{-}20$ Joule obtained for a molecular squared cell of lateral size $l = 1.5$nm [12] and relative permittivity $\epsilon_r = 1$ (no dielectric material between cells)

6. the values of dissipated energy are calculated over the respective period of computation for each scheme, the corresponding power values are considered averaged on the same period

7. since we are considering the worst case scenario, the value of $K_i(t)$ from equation 6.6 is non time dependent therefore the deleted information is always equal to the number of inputs of stage i

From the previous assumptions and from equation 6.6 the energy dissipated in a period T_b for Bennett clocked scheme in the XOR binary tree is calculated as follows:

$$\begin{aligned} E_B &= \int_0^{T_b} P_B(t)dt \\ &= 2E_k \int_0^{T_b} \sum_{k=0}^{\infty} \sum_{i=0}^{M-1} K_i(t)\delta(t - \frac{kT_b}{2})dt \\ &= 2E_k \sum_{i=0}^{M-1} K_i(T_b/2) + K_i(T_b) \\ &= 2E_k \sum_{i=0}^{M-1} 2^{ic} \\ &= 2E_k \frac{2^{cM} - 1}{2^c - 1} \end{aligned}$$

where the formula for the sum of a geometric progression of ratio 2^c has been used. Similarly the energy dissipated in a Landauer clocked binary tree is also the sum on the energy dissipated on the whole binary tree:

$$\begin{aligned} E_L &= \int_0^{T_l} P_L(t)dt \\ &= 2E_k \sum_{i=0}^{M-1} 2^i \\ &= 2E_k(2^M - 1) \end{aligned}$$

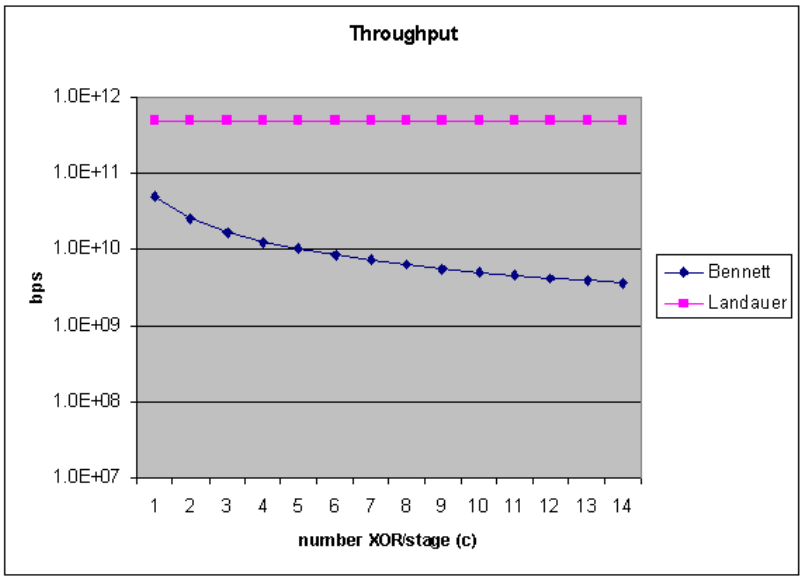


Figure 6.20. Throughput comparison

Figures 6.20 through 6.24 compare on different parameters a binary tree with M=14 stages as a function of the number of stages of the XOR tree per pipeline stage is c, obviously being c a parameter used only for Bennett scheme the values for Landauer scheme are always constant.Moreover, even though only the values of c that provide an integer M are physically significative, the extra points are used to show a better interpolation of the curves. when c=14 the Bennett scheme has no actual pipelining being the whole circuit in an only Bennett stage.

Figure 6.20 shows a comparison of the throughput in the Bennett and Landauer scheme, as expected the Landauer scheme shows higher throughput and the gap between the performances increases with the increase of c.

Figure 6.21 shows the advantages of the Bennett scheme as a measure of the dissipated energy per period of computation (formulas reported above). The improvement in terms of power consumption becomes better with the increase of c however the power dissipated also with a pure Bennett scheme $c = 14$ does not become zero as the inputs to the whole circuit are still deleted every T_b.

Figure 6.22 compares the power dissipation computed as the ratio between the energy dissipated per period computation over the time length of a computation period T_l and T_b. The curves show again that the power dissipation for Bennett clocking improves with the increase of c.

Figure 6.23 instead tries to answer to the questions "what is the amount of operations (in this case output bits) per unit energy spent?" the answer also here show that Bennett clocking is more efficient for energy considerations.

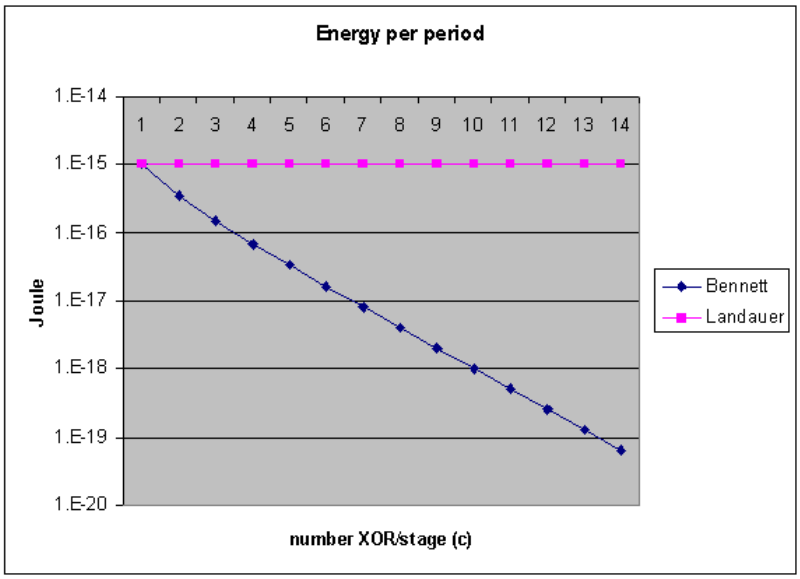


Figure 6.21. Comparison of Energy dissipation per period

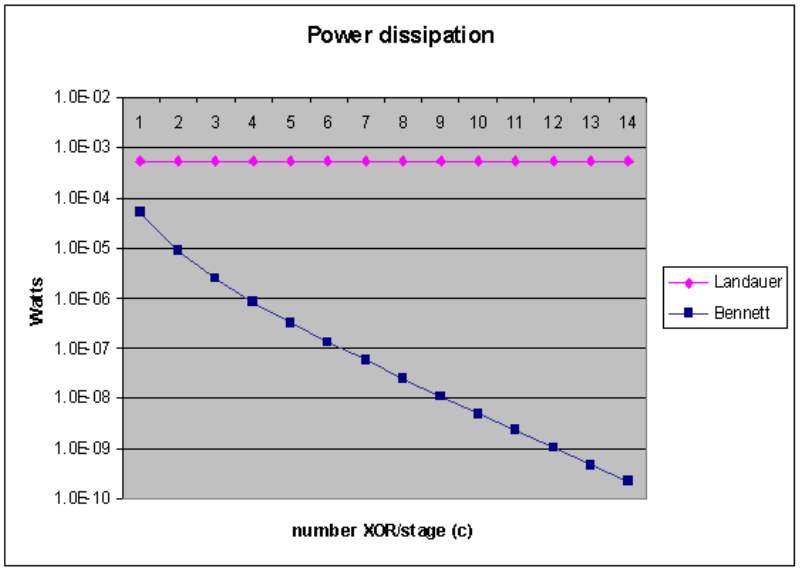


Figure 6.22. Comparison of Power dissipation

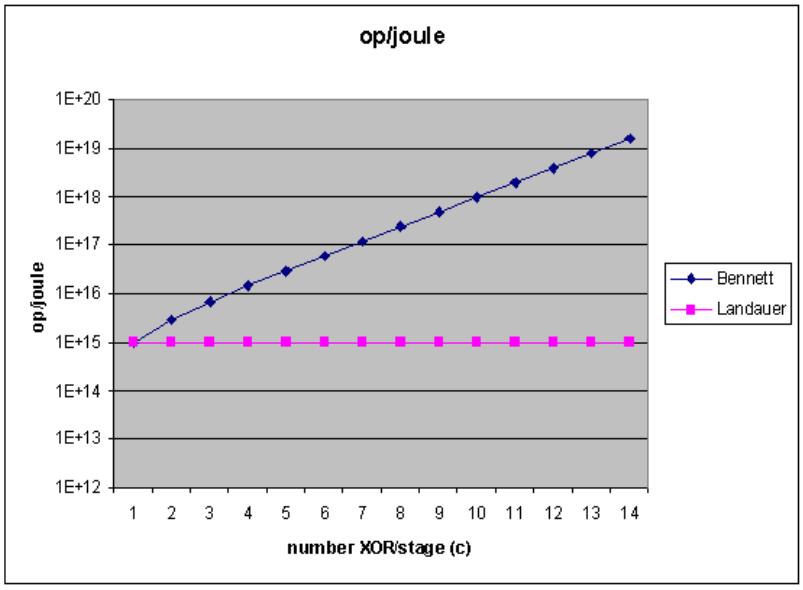


Figure 6.23. Comparison of operations per joule

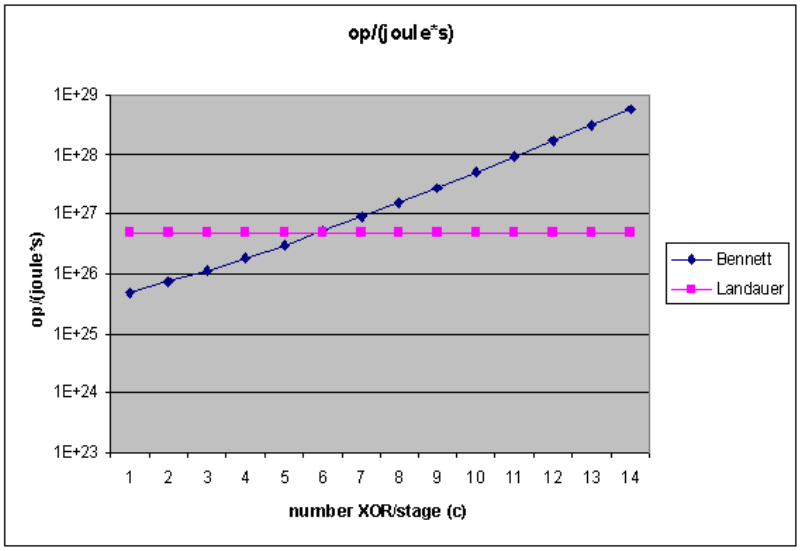


Figure 6.24. Comparison of operations per joule per second

Finally figure 6.24 addresses the question "Given a second of time and a Joule of energy, what is the amount of operations (output bits) obtained?" thus introducing also a time factor to the evaluation of the performances, in this case the result show an interesting intersection of the two curves introducing a watershed between the values of c for which Landauer clocking has better performances ($c \leq 6$) and the those for which Bennett clocking behaves better ($c > 6$). This result can be explained as follows: for low values of c the throughput advantages of using a pipelined approach with the Bennett scheme are not sufficient to overcome the penalty in terms of power dissipation, with the increase of the size of the pipeline stages (higher c) the advantages in terms of power dissipation have a bigger impact with respect to the reduction in performances.

Discussion

Three very different approaches to reversible system design. The reversible pipeline is erasure free at the cost of twice the circuit space. The Collapsed Bennett layout uses minimal additional area but incurs a time penalty. The partially reversible floorplan is relatively time and space efficient at the cost of power dissipation. The examples discussed here are clearly not an exhaustive list. However, they do present three very different approaches to the problem. The fully reversible pipeline approache attacks the erasure problem primarily with hardware. The partially reversible pipeline somewhat ignores the higher level problem and trades minimal dissipation for increased throughput. The Collapsed Bennett layout approaches the problem through a higher level algorithm. To design small, fast, low-power systems, one must be aware of all these approaches and be on the lookout for more.

Summary

Tables 6 and 6 summarize the set of the assumptions made for each architecture. This chapter presented several examples of very different approaches to architecture design for reversible QDCA systems. Which is the "best" will depend on the relative availablility of time, space, and power and the particular QDCA implementation.

Table 6.3. Summary of Assumptions, Part I

Architecture	Assumptions
Common for All Architectures	• Data on QDCA layer cannot influence clock • Physical crossovers available
Irreversible	• Zone floorplanning • Zone-type clock signal • Four phase clock • Both 90 degree and 45 degree cells available
Retractile Cascade	• Zone or abutting columnar floorplanning • Bennett clock signal • Multiple phase clock

Table 6.4. Summary of Assumptions, Part II

Architecture	Assumptions
Mirror Circuit	• Zone or abutting columnar floorplanning • Landauer clock signal • Multiple phase clock
Collapsed Bennett	• Zone or abutting columnar floorplanning • Bennett and pulse clock signals • Multiple phase clock
Partially Reversible Pipeline	• columnar floorplanning • Bennett clock signal and latch • Multiple phase clock

Chapter 7

Conclusion

This document is the first attempt to describe how to go about designing a QDCA system, and the first description of how to design reversible QDCA systems above the single gate level. The design space is substantial, and there are many questions that designers need to answer before beginning to design. This document attempts to make those questions clear and explicate the tradeoffs to be made while introducing techniques and tools that can be used to design power efficient QDCA systems.

References

[1] Dominic Antonelli, Timothy Dysart, Danny Chen, Xiaobo Hu, Andrew Kahng, Peter M. Kogge, Richard C. Murphy, and Michael T. Niemier. Quantum dot cellular automata (qca) circuit partitioning: Problem modeling and solutions. In *41st Design Automation Conference (DAC)*, June 2004.

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

[3] Gary H. Bernstein, Islamshah Amlani, Alexei Orlov, Craig Lent, and Gregory Snider. Observation of switching in quantum-dot cellular automata cell. *Nanotechnology*, (10):166–173, 1999.

[4] Daniel Berzon and Terry Fountain. Computer memory structures using qca. Technical report, University College London, 1998.

[5] EP Blair and CS Lent. An architecture for molecular computing using quantum-dot cellular automata. *Nanotechnology, 2003. IEEE-NANO 2003. 2003 Third IEEE Conference on*, 1, 2003.

[6] SC Chan, KL Shepard, and PJ Restle. Design of resonant global clock distributions. *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 248–253, 2003.

[7] PE Cottrell and EM Buturla. VLSI wiring capacitance. *IBM Journal of Research and Development*, 29(3):277–288, 1985.

[8] E.P. DeBenedictis. Qca layout of a toffoli gate. Personal communication, July 2006.

[9] AJ Drake, KJ Nowka, TY Nguyen, JL Burns, and RB Brown. Resonant clocking using distributed parasitic capacitance. *Solid-State Circuits, IEEE Journal of*, 39(9):1520–1528, 2004.

[10] Timothy Dysart and Peter M. Kogge. Strategy and prototype tool for doing fault modeling in a nano-technology. In *IEEE Nano Conference*, August 2003.

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

[12] S.E. Frost, T.J. Dysart, P.M. Kogge, and C.S. Lent. Carbon nanotubes for quantum-dot cellular automata clocking. *Nanotechnology, 2004. 4th IEEE Conference on*, pages 171–173, 2004.

[13] 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 *First Workshop on Non-Silicon Computing*, 2002.

[14] S.E. Frost, Arun F. Rodrigues, Charles A. Giefer, and Peter M. Kogge. Bouncing threads: Merging a new execution model into a nanotechnology memory. In Asim Smailagic and Nagarajan Ranganathan, editors, *IEEE Computer Society Annual Symposium on VLSI: New Trends and Technologies for VLSI Systems Design*, pages 19–25. IEEE Computer Society, February 2003.

[15] K. Hennessy and C.S. Lent. Clocking of molecular quantum-dot cellular automata. In *JOURNAL OF VACUUM SCIENCE and TECHNOLOGY B*, volume 19(5), pages 1752–1755, 2001.

[16] K. Hirose and H. Yasuura. A bus delay reduction technique considering crosstalk. *Electronics & Communications in Japan, Part III: Fundamental Electronic Science(English translation of Denshi Tsushin Gakkai Ronbunshi)*, 85(1):24–31, 2002.

[17] X. Hu. Introduction to simple12 assembly and rtl. Dept. of Computer Science and Engineering, University of Notre Dame, 2004. Introduces S12 ISA and RTL in undergraduate computer architectour course(CSE 321).

[18] RK Kummamuru, AO Orlov, R. Ramasubramaniam, CS Lent, GH Bernstein, and GL Snider. Operation of a quantum-dot cellular automata (QCA) shift register and analysis of errors. *Electron Devices, IEEE Transactions on*, 50(9):1906–1913, 2003.

[19] R.K. Kummamuru, J. Timler, G. Toth, C.S. Lent, R. Ramasubramaniam, A. Orlov, and G. H. Bernstein. Power gain in a quantum-dot cellular automata latch. *Applied Physics Letters*, 81(7):1332–13334, August 2002.

[20] C.S. Lent, S.E. Frost, and P.M. Kogge. Reversible computation with quantum-dot cellular automata (QCA). *Proceedings of the 2nd conference on Computing frontiers*, pages 403–403, 2005.

[21] CS Lent, Liu M., and Lu Y. Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling. *Nanotechnology*, 17:4240–4251, 2006.

[22] C.S. Lent and P.D. Tougaw. A device architecture for computing with quantum dots. In *Proc. of the IEEE*, volume 85, pages 541–557, Mar 1997.

[23] C.S. Lent, P.D. Tougaw, and W. Porod. Quantum cellular automata: The physics of computing with arrays of quantum dot molecules. In *Proceedings of the Workshop on Physics and Computing, IEEE Computer Society Press*, 1994.

[24] Zhaohui Li and Thomas P. Fehlner. Molecular qca cells. 2. characterization of an unsymmetrical dinuclear mixed-valence complex bound to a au surface by an organic linker. *Inorganic Chemistry*, 42(18):5715–5721, 2003.

[25] M. Lieberman, S. Chellamma, B. Varughese, Y.L. Wang, C.S. Lent, G.H. Bernstein, G. Snider, and F.C. Peiris. Quantum-dot cellular automata at a molecular scale. In *Molecular Electronics II, Annals of the New York Academy of Sciences*, volume 960, pages 225–239, 2002.

[26] Gordon Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 19 1965.

[27] K. Nabors and J. White. FastCap: a multipole accelerated 3-D capacitance extraction program. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 10(11):1447–1459, 1991.

[28] Michael T. Niemier. Designing digital systems in quantum cellular automata. Master's thesis, University of Notre Dame, April 2000.

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

[30] Michael T. Niemier and Peter M. Kogge. Designing complex logic systems with qca devices. In *Great Lakes Symposium of VLSI*, March 1999.

[31] Michael T. Niemier and Peter M. Kogge. Logic-inwire: Using quantum dots to implement a microprocessor. In *International Conference on Electronics, Circuits, and Systems (ICECS '99)*, September 1999.

[32] Michael T. Niemier and Peter M. Kogge. Exploring and exploiting wire-level pipelining in emerging technologies. In *International Symposium of Computer Architecture*, pages 166–177, Sweden, July 2001. ISCA 2001.

[33] Michael T. Niemier and Peter M. Kogge. The 4-diamond circuit: A minimally complex nanoscale computational building block in qca. In *IEEE Symp. on VLSI (ISVLSI)*, pages 3–10, February 2004.

[34] Micheal T. Niemier and Peter M. Kogge. Logic-in-wire: Using quantum dots to implement really dense processing logic. In *Proceedings of the Third Petaflops Workshop, with Frontiers of Massively Parallel Processing*, February 1999.

[35] M.T. Niemier and P.M. Kogge. Problems in designing with qcas: layout=timing. In *International Journal of Circuit Theory and Applications*, volume 29(1), pages 49–62, Mar 2001.

[36] M.T. Niemier, A.F. Rodrigues, and P.M. Kogge. A potentially implementable fpga for quantum dot cellular automata. In *1st Workshop on Non-Silicon Computation (NSC-1), held in conjunction with 8th Int. Symp. on High Performance Computer Architecture (HPCA-8)*, Boston, MA, 2002.

[37] 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*, 74(19):2875–2877, May 1999.

[38] M. Ottavi, V. Vankamamidi, F. Lombardi, S. Pontarelli, and A. Salsano. Design of a qca memory with parallel read/serial write. In *IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design (ISVLSI'05)*, pages 292–294, 2005.

[39] Paul Rutten, Mickey Tauman, Hagai Bar-Lev, and Avner Sonnino. Is moore's law infinite? the economics of moore's law. In *Kellog TechVenture 2001 Anthology*. 2001.

[40] IH Tan, GL Snider, LD Chang, and EL Hu. A self-consistent solution of Schrodinger–Poisson equations using a nonuniform mesh. *Journal of Applied Physics*, 68(8):4071–4076, 2006.

[41] Scott Thompson, Mohsen Alavi, Makarem Hussein, Pauline Jacob, Chris Kenyon, Peter Moon, Matthew Prince, Sam Sivakumar, Sunit Tyagi, and Mark Bohr. 130 nm logic technology featuring 60nm transistors, low-k dielectrics, and cu interconnects. *Intel Technology Journal*, 6(2):5–13, May 16 2002.

[42] J. Timler and C. Lent. Maxwell's demon and quantum-dot cellular automata. *Journal of Applied Physics*, 94(2):1050, 2003.

[43] J. Timler and C.S. Lent. Power gain and dissipation in quantum-dot cellular automata. *Journal of Applied Physics*, 91(2):823–831, January 2002.

[44] P.D. Tougaw and C.S. Lent. Logical devices implemented using quantum cellular automata. In *Journal of Applied Physics*, volume 75(3), pages 1818–1825, 1994.

[45] K. Walus, A. Vetteth, G.A. Jullien, and V.S. Dimitrov. Ram design using quantum-dot cellular automata. In *Technical Proceedings of the 2003 Nanotechnology Conference and Trade Show*, volume 2, pages 160–163, 2003.

DISTRIBUTION:

1 S. Murphy
384 Fitzpatrick Hall
Notre Dame, IN 46556

1 M. Frank
2525 Pottsdamer St. Rm 341
Tallahassee, FL 32310

1 MS 1322
J. Aidun, 1435

1 MS 1319
N. Pundit, 1423

1 MS 1322
S. Dosanjh, 1420

2 MS 9018
Central Technical Files, 8944

2 MS 0899
Technical Library, 4536

1 MS 0123
D. Chavez, LDRD Office, 1011

Sandia National Laboratories