COMPUTER-AIDED-DESIGN METHODS FOR

EMERGING QUANTUM COMPUTING

TECHNOLOGIES

Approved by:

_____

Dr. Mitchell A. Thornton (Chair & Dissertation Director)

_____

Dr. Hesham El-Rewini

_____

Dr. Fatih Kocan

_____

Dr. D. Michael Miller (Univ. of Victoria, BC, Canada)

_____

Dr. Sukumaran Nair

_____

Dr. Stephen A. Szygenda

COMPUTER-AIDED-DESIGN METHODS FOR EMERGING QUANTUM COMPUTING TECHNOLOGIES

A Dissertation Presented to Graduate Faculty of the

School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Computer Engineering

By

David Dov Yehuda Feinstein

(B.S.E.E., Technion-Israel Institute of Technology, 1977)
(M.S.E.C.E., University of Houston, 1992)

May 17, 2008

ACKNOWLEDGEMENTS

I was very fortunate to have Prof. Thornton as my teacher, advisor and mentor. His superb classes on verification and CAD methods quickly brought me to the frontier of digital logic research. Prof. Thornton has the unique requirement that his graduate class projects must be submitted in the form of conference papers. This required me to write six practice conference papers during his classes, an experience that turns out to be invaluable for the success of my research. Prof. Thornton inspired me to publish quality work extensively with great care for details. He encouraged me to attend many international and national conferences, during which I met his many and diverse collaborators, all in the forefront of digital logic research. Coming from a similar background in the computer industry, Prof. Thornton was very supportive in my efforts to achieve excellence in studies and research while still running my high tech company.

It was a great honor to work with and to learn so much from Prof. D. Michael Miller while producing many of the papers at the core of this dissertation. A long time collaborator of Prof. Thornton, Prof. Miller graciously agreed to serve in my committee.

Many thanks to Prof. El-Rewini, Prof. Szygenda, Prof. Nair and Prof. Kocan for their time, support, valuable comments, and enthusiasm to serve in my committee. I benefited immensely from taking their graduate classes at SMU.

To accomplish a Ph.D. in computer engineering at the age of 53 while running a high tech company like INNOVENTIONS Inc required tremendous support from all my dedicated employees. Special thanks go to my secretary Monika Frausto for grammatically reviewing all my research work. Scott LaRoche and Jose Nunez had to work much harder to cover for the extra time I spent on this research.

The support of my family and many friends was crucial throughout the four years of my PhD "adventure". My wife Orna patiently waited for me to come home at 9:00 pm every day, knowing that I may still spend several more hours of study at home. With her love and support I was better able to succeed in this research. My son Oren and my daughter Donna were great fans of my work and encouraged me to reach the tough finish line. Oren's achievement in obtaining a B.S.E.E. from UT Austin with 4.0 GPA challenged me to accomplish the same with my graduate classes at SMU. My late father, Benjamin Feinstein, encouraged me to be an intellectual and he instilled in me the love of books in early childhood.

I am blessed with many supportive friends. Prof. Çetin Koç, my mentor during my MSEE studies, was instrumental in my choice of SMU. Prof. Nachum Dafny was a great inspiration on the importance of getting a PhD. Ron Avni, Menachem Zucker, David Prusman, Avi Alon, Shay Korentayer, Yadin David, Isaac Peretz, and Moti and Ora Tenenhaus followed my progress with much enthusiasm. Last but not least, I would like to thank my life time friend, Avery More, for his great support and encouragement.

Feinstein, David Dov Yehuda

B.S.E.E., Technion - IIT, 1977
M.S.E.C.E., University of Houston, 1992

Computer Aided Design Methods for
Emerging Quantum Computing
Technologies

Advisor: Professor Mitchell A. Thornton

Doctor of Philosophy conferred May 17, 2008

Dissertation completed April 18, 2008

Emerging quantum computing technologies are poised to replace standard CMOS logic when the exponential size reduction reaches sub-atomic dimensions. Quantum circuits are reversible and therefore promise the potential of computation without energy loss. This research considers *computer aided design* (CAD) methods for all major aspects of quantum computing circuit design including logic synthesis, simulation, verification, and testing.

The technologies we investigate include *quantum cell automata* (QCA) and general *quantum circuits* (QC), with significantly more emphasis on the later. The recently introduced *quantum multi-valued decision diagram* (QMDD) provides an efficient method to represent and simulate quantum (and other classical reversible) circuits. A major contribution of this dissertation is the development of a sift-like minimization as well as structure metrics based minimization techniques for QMDD. We have used the enhanced QMDD to efficiently simulate quantum circuits as well as quantum vectors.

Our early investigation of reversible logic is concerned with a virtual implementation that uses a direct translation of relatively complex binary functions into circuits composed of Fredkin reversible gates. This allowed us to project size and speed complexity of complex reversible circuits, although this direct translation approach fails to minimize garbage inputs and outputs. To achieve proper garbage minimization, one must synthesize reversible logic using gate cascades with appropriate optimization methods embedded that attempt to minimize the total number of lines in the circuit. To that end, we developed a novel QMDD-based tool for cascade logic synthesis that utilizes the QMDD minimized variable order for lexicographical synthesis with garbage minimization included as an optimization criterion.

We developed a synthesis tool that investigates the QCA native 3-input majority gates ability to implement complex logic circuits. In particular, we explore the benefit of transforming the logic description into *exclusive sum of products* (ESOP) forms prior to implementation in the majority gates.

We survey recent efforts in establishing the foundation for QC testing and fault tolerant QC. We project the potential use of random tests as well as *built in self test* (BIST) techniques for future QC. A major contribution of this dissertation is the investigation of partially redundant reversible logic. Detection of partially redundant logic within any design, reversible or irreversible, has ramifications for logic synthesis, for design verification, and for *design for test* (DFT) issues.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

DEDICATION

To my dear wife Orna Feinstein,

to my good friend Avery More, and

in memory of my father Benjamin Feinstein (1921-1986)

CHAPTER 1

INTRODUCTION

Quantum mechanics is the thought revolution of the 20th century. The Einstein, Podolsky, and Rosen experiment created a paradox when a pair of two entangled photons go in different directions and undergo polarization measurements. The pair seems to affect the polarization of each other while they are separated by a long distance [EPR35]. The EPR experiment suggested the possibility of hidden variables of quantum particles that inherently cannot be measured. The Heisenberg uncertainty principle further complicated the intuitive interpretation of the emerging quantum mechanics field [Heis30]. The Copenhagen interpretation during the 1930s, led by Niels Bohr, promoted the notion that a particle property may be considered real only when it is actually measured [Waer67, Omne99]. Bell's theorem proved that no hidden variable theory that preserves locality and determinisms can explain the prediction of quantum mechanics for an entangled photon pair [Bell64]. The field of quantum mechanics has seen tremendous developments since that time, but it is still entangled with magnificent wonders and phenomenon that defy our intuition.

Quantum mechanics was bridged to computing when Richard Feynman demonstrated in 1982 that various quantum mechanics effects cannot be simulated

properly on classical computers [Feyn82]. This observation brought forth the idea that perhaps we can obtain a new type of efficient computation if the computer itself is made to use quantum effects natively. Feynman and David Deutsch independently conceived the notion of the quantum computer in 1985 [Feyn85, Deut85]. Deutsch and Jozsa followed with the observation that quantum systems exhibit exponential increase in parallelism when the system size is increased [DJ92]. In 1993, Bennett et al. invented the notion of *quantum teleportation* [BBC+93]. A critical breakthrough was Peter Shor's seminal paper of 1994 that illustrated how a quantum computer may be able to factor primes exponentially faster than classical computation [Shor94]. Since the difficulty of factoring a large number into primes lies at the heart of the RSA encryption algorithm, such a potential breakthrough immediately caught worldwide attention. Lev Grover's search algorithm in 1996 that breaks the classical barrier of *O(n)* with an $O(\sqrt{n})$ complexity further accelerated research interest in the new field of quantum computing [Grov96]. However, the prospects of better performance were not the only motivation for the immense interest we now see in quantum computing.

It has been projected that by 2018, mainstream CMOS technology will not be able to follow Moore's Law of speed and density doubling every 18 months [XCN+06]. So far, technologists have been able to overcome major hurdles such as lithographic limits, power constraints, electric field breakdown, and leakage currents. However, the accelerated shrinkage of circuit dimensions will eventually bring the size of basic circuit elements to the molecular and atomic scale where quantum phenomena rules prevail.

This is the boundary which spells the imminent demise of conventional CMOS technology and motivates us to investigate quantum technology.

Ray Kurzweil and other futurists noted that regardless of the hurdles, overall technology must continue to follow Moore's Law, causing new technologies to be invented [Kurz07]. And indeed, the projected demise of CMOS spawns the arrival of *quantum computing* (QC). We briefly describe the fundamentals of quantum computing in Chapter 2, where, in a somewhat simplistic approach, we include several technologies whose basic circuit elements must deal with individual atoms at the quantum phenomenon level.

1.1. Emerging Quantum Computing Technologies

Emerging quantum nanotechnologies have been developed in the hope of achieving a performance that goes beyond the end-of-the-curve limits of CMOS. *Quantum-dot cellular automata* (QCA), invented by Lent et al. in 1993 [LTP93, TLP93], is a very promising technology. QCA had become the topic of intense research due to its projected size density of several orders of magnitude smaller than CMOS. It also promises fast switching time in the range of 10^{-12} to 10^{-15} seconds, and extremely low power. Experimental circuits utilizing metal dots were demonstrated in the late 90s [AOSL98], followed by the recent development of molecular QCA circuits [QSL+03].

QCA is radically different from today's CMOS technology. Information in QCA propagates along a line of cells via Coulombic charge interactions as opposed to conventional electric current as used in CMOS devices. Unlike the gate versatility of CMOS, the QCA technology natively implements the 3-input majority gate as the basic

building block. QCA also requires a new multi-zone clock mechanism for proper data propagation, prompting electrical and magnetic based solutions. The 3-input majority gate can easily implement 2-input AND or OR gates provided the third input is set at the logic '0' or '1' state, respectively. However, to avoid the waste of fixed logic states in one third of the inputs, researchers have been investigating improved utilization of all three inputs of the majority gates [ZWWJ04, WSJ+04]. QCA fundamentals are further described in Chapter 2.3.

In classical irreversible logic design, the circuit's inputs are evaluated to provide the output results in a process that does not preserve input information once the output is computed. As a result, one cannot recreate the inputs from the output information. In the extreme case of a decision circuit, an *n*-bit input is processed to obtain a single bit yes/no result. Clearly there is no way that a one bit result can re-construct an *n*-bit input. In contrast, reversible logic circuits have the main property that the circuit's input information can be fully reconstructed at any time from the output [NC00].

Landauer's principle states that each erasure of one bit of information dissipates at least $K_B T \ln 2$ energy, where K_B is the Boltzmann's constant and T is the temperature [L61]. Pioneering work by Bennet [Benn73], Fredkin and Toffoli [FT82], and Feynman [F85] illustrated the potential of reversible logic to create circuits with theoretical zero internal power dissipation. Reversible logic circuits are modeled as bijective functions that allow the full reconstruction of the circuit's input information from the output, thus eliminating the energy loss with the absence of erased bits. While Landauer's erasure energy is currently many orders of magnitude below today's CMOS

leakage current, it will play a major role in the future, posing a limiting factor for QCA and other futuristic *irreversible* logic circuits. As a result, the common consensus among computer scientists and theoretical physicists is that the long term quantum technology is likely to be based on reversible logic.

In addition to the theoretical elimination of energy loss, it seems that future implementation of quantum logic circuits based on optical or other quantum effects will necessarily be reversible for another reason - they impose the characteristic fan-in and fan-out of *one* due to the principle of conservation of matter.

We would like to emphasize that there are currently no substantial implementations of reversible QC circuits, although interesting small experimental circuits have been reported extensively [NC00, Hirv04]. An excellent recent survey of the architectural implementation of quantum computing technologies appears in [VMO06]. In contrast, logic reversibility can be synthesized and simulated on classical computers, as demonstrated in this research as well as in the large body of published relevant research [Cabe04].

To fulfill the bijective relation between input and output, reversible logic circuits must have the same number of outputs as inputs (or have disallowed patterns on the largest set). To illustrate this initial discussion of reversible logic, Fig. 1.1 previews the Toffoli reversible logic gate that is commonly used for quantum computing and will be formally described in Section 2.2. Note that this gate has three inputs and three outputs, and the specification shows that every output set bijectively relates to a specific input set. The transformation matrix indicates the permutation relation between the 8

combinations of inputs (2^3) to the 8 combination of outputs (2^3). For example, the first column indicates the mapping $(a,b,c) \rightarrow (a',b',c')$: 000→000, while the seventh column indicates the mapping 110→111. We will see in Chapters 2 and 3 that transformation matrices describing all quantum circuits are *unitary* and can consist of more complex mappings.

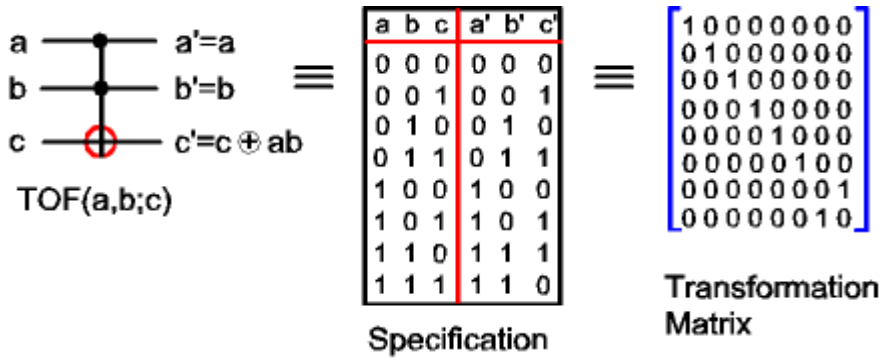


Figure 1.1. Transformation Matrix Representation of the Toffoli Gate

The requirement of a bijective relation between inputs and outputs can easily lead to a large number of unused (garbage) outputs for circuits with more inputs than outputs, and poses an immediate challenge for logic synthesis in order to minimize the garbage outputs. One can readily imagine the exponential explosion of the transformation matrix size with increasing numbers of lines.

Although general QC is materially different than QCA, we believe that there is some interesting commonality to allow their mutual inclusion in the present research. Both technologies are so radically different than CMOS that they require major new design paradigms to deal with circuit simulation, synthesis, and testing. The different

approaches taken with each technology provides added insight to the uniqueness of quantum computing.

1.2. Quantum Computing Computer Aided Design Tools

Following the development of classical digital logic, emerging QC technologies rely heavily on *computer aided design* (CAD) tools. For this dissertation we have developed several CAD tools that encompass the main targets of synthesis, simulation, testing and verification. Fig. 1.2 provides an overview of the QC CAD tools.

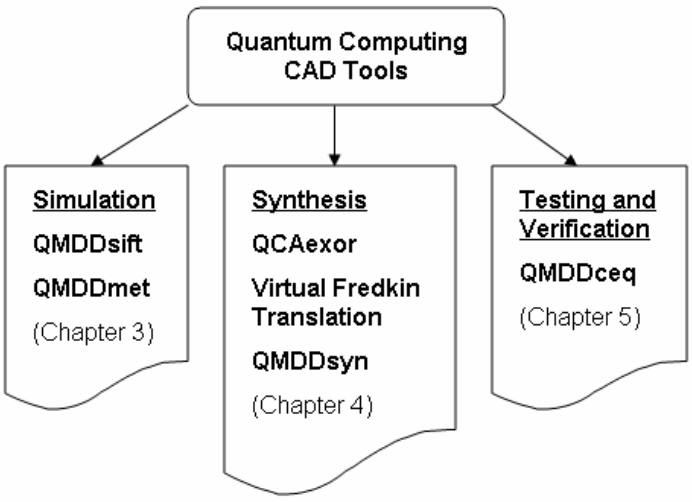


Figure 1.2. Overview of the Quantum Computing CAD Tools

Several of our tools are based on the *quantum multi-valued decision diagram* (QMDD) originally proposed by Miller and Thornton to simulate and specify the transformation matrix of reversible logic circuits in a compact form [MT06, MTG06]. Like all decision diagrams, QMDD suffers from the exponential size explosion of the data structure so that size minimization becomes crucial. In this dissertation, we have

further developed the *QMDDsift* package that includes a sift-like minimization [MFT07, MFT07b]. Further minimization improvements were achieved with the *QMDDmet* tool that guides the minimization process based on the structure metrics [FTM08b]. We have used various implementations of the QMDD packages to simulate QC benchmark circuits, as well as simulation of vector multiplication [GTFM07].

The *QCAexor* logic synthesis tool for QCA using *exclusive-OR sum of products* (ESOP) minimization is detailed in [FT07]. The major efforts in this dissertation target QC and classical reversible logic synthesis. In our initial work on reversible logic synthesis, we developed the *Virtual Fredkin Translation* tool [FTN07]. Using *Altera's Quartus II* common synthesis tool, we demonstrate how to virtually implement relatively complex logic circuits in reversible Fredkin gates. This approach, as discussed in Section 4.2, allowed us to gain a preliminary glimpse into the size and speed complexities of reversible logic. However, since this approach cannot control the exploding number of garbage outputs it is not intended as a replacement for other approaches to synthesize cascades of gates.

The state-of-the-art of gates cascade synthesis is currently limited to twenty inputs or less [AP05]. The *QMDDsyn* tool improves lexicographic synthesis of reversible logic by employing improved variable ordering as obtained by the QMDD package.

Quantum logic testing and verification is quite different from standard logic. It was shown by Agrawal that reversible logic circuits are easier to test since a single fault detection implies the ability for multiple fault detection [Agra81]. On the other hand,

testing of QC bring us to the problem of measurement in the quantum mechanics sense. The QC decoherence problem (described in Chapter 2) mandates the extensive use of QC fault tolerance techniques. There is also the unique issue of probabilistic testing versus deterministic testing.

The QMDDceq tool is useful for both *design for test* (DFT) and verification. It investigates the detection of partially redundant reversible logic. We have also developed a similar tool (called *CMBtest*) for classical irreversible logic. Compared results between reversible and irreversible logic enable us to gain new theoretical insight into reversible logic.

1.3. Impact and Contributions of this Research

We made several contributions to the major disciplines of CAD for QC, including simulation, synthesis, testing, and verification. In simulation, our work resulted in an improved QMDD package that can dynamically reduce its size and accommodate larger quantum circuits. Our work on data metrics based QMDD minimization represents a novel approach to *decision diagram* (DD) minimization, and may have application with other DD packages. We developed several approaches for reversible logic and for QC synthesis. We also introduce a new method to synthesize logic within the native 3-input majority gates of QCA. The QMDDceq tool for detecting partially redundant logic in reversible logic has ramifications for design verification, and for *design for test* (DFT) issues. Our comparative work on detection of partially redundant logic in irreversible logic can enhance traditional logic synthesis by eliminating wasted resources.

Our work on improving the QMDD and using it in various tools may prompt other researchers to further exploit the QMDD (or other DD tools for QC) for different CAD tasks beyond those explored in our research.

In addition to the experimental success of our various tools, several theoretical contributions have been made throughout this dissertation.

There is no doubt that the field of quantum computation is exploding. In fact, Adan Cabello's monumental bibliographic compendium on the foundation of quantum mechanics and quantum information details more than 11,000 publications [Cabe04]. This research further provides a contemporary survey of various theoretical and practical aspects of QC CAD. As we feel that QC logic synthesis is a crucial barrier in the development of QC, we made an effort to cover many different synthesis methods at some length, in addition to the disclosure of our original work. A good portion of the prior work we cite in this dissertation does not appear in the widely cited text book "Quantum Computation and Quantum Information" published by Nielsen and Chuang in 2000 [NC00]. We therefore hope that this dissertation will prove to be useful for other researchers who are new to this field.

1.4. Organization

The remainder of this dissertation is organized as follows. In Chapter 2 we discuss the fundamentals of the transition from quantum mechanics to quantum computing. Chapter 3 is devoted to the representation and simulation of QC and classical reversible logic. It introduces the QMDD package and details our novel minimization techniques. Chapter 4 discloses our contributions for reversible logic

synthesis as well as the *QCAexor* QCA synthesis tool. Our work on QC testing and verification is presented in Chapter 5.

Chapter 2 covers the fundamentals background for this research. We briefly survey the key ideas of quantum mechanics from the Stern-Gerlach experiment to entanglement. We then discuss the important issues of *measurement* and *decoherence* in view of the 3rd postulate of quantum mechanics. We then expand on the basic properties of reversible logic. With this theoretical foundation we proceed to outline the basic ideas of QC and quantum information as they evolved from quantum mechanics. Special attention is spent on quantum operations and elementary quantum gates since these are the building blocks used in our research. We conclude this chapter with a discussion of the QCA fundamental properties.

We open Chapter 3 with a discussion of classical *binary decision diagrams* (BDD) followed by a survey of QC simulation on computers using various BDDs. The novel QMDD simulation package is introduced followed by a detailed discussion of the sift-like minimization tool *QMDDsift* [MFT07, MFT07b]. We then disclose our recent work on QMDD minimization based on QMDD data structure metrics [FTM08b]. We conclude Chapter 3 with our simulation results of QC benchmark circuits from the Maslov web-page [Masl05] and briefly discuss the simulation of vector matrix multiplication [GTFM07].

Chapter 4 is dedicated to the synthesis of quantum circuits, where we start with our work on ESOP transformation to majority gates for QCA [FT07] and virtual implementation of reversible logic [FTN07]. We follow with a brief but comprehensive

survey of current approaches for reversible and QC synthesis. We close with a detailed discussion of our contribution to lexicographical synthesis using variable selection based on the *QMDDsyn* tool variable order.

We turn our attention to QC testing and verification in Chapter 5. We start with a survey of reversible logic testing techniques and various proposals to create a unified QC fault model. Our contribution for QC verification is then disclosed with a detailed description of the *QMDDceq* tool.

Finally, our conclusions and future research areas appear in Chapter 6. In view of the nascent nature of QC, it should be no surprise that we foresee substantial future areas of work in all the aspects of quantum computing CAD outlined in this research.

CHAPTER 2

FROM QUANTUM MECHANICS TO QUANTUM COMPUTING

Quantum computing developed in the past two decades from the core ideas of quantum mechanics that shook the physics world in the 1930s. In this chapter we delve into some of the major concepts of quantum mechanics and show how they evolved into today's foundation of quantum computing and quantum information. Fundamental ideas of reversible logic and QC that are crucial for this dissertation are introduced. We conclude this chapter with an introduction of QCA.

2.1. Key Ideas of Quantum Mechanics

We briefly discuss several key ideas that are crucial to better understand the evolution of quantum mechanics into the framework of quantum computing and quantum information in this section. Section 2.1.1 starts with some fundamental concepts of linear algebra. We then illustrate the famous Stern-Gerlach experiment that justifies the notion of qubits and superposition. Section 2.1.4 follows with the four postulates of quantum mechanics to illustrate the importance of the Hilbert space, the Hamiltonian operation, and measurements. The discussion of measurement in quantum mechanics is rather detailed in view of its importance for QC.

2.1.1. Linear Algebra Preliminaries

We briefly introduce a few of the concepts of linear algebra that are crucial to understand quantum circuits. More extensive background is available in [NC00, MM05, Saku92, Bohm01, Holl90, Dira58, Lawd67].

We start with the famous Dirac notation for complex vectors, which use a clever play on the pseudo-word "braket".

Definition 2.1: (Dirac Notation). A complex vector u which is represented by a single column vector is called a *ket* vector and is denoted as $|u\rangle$. A complex vector v which is represented by a single row vector is called a bra vector and is denoted as $\langle v|$.

Definition 2.2: An inner product produces a complex number while operating on two vectors $|u\rangle$ and $|v\rangle$. The vector space C^n has an inner product defined as

$$((u_1, u_2, u_3, ..., u_n), (v_1, v_2, v_3, ..., v_n)) \equiv \sum_i u_i^* v_i \tag{2.1}$$

in Dirac notation, the inner product of $|u\rangle$ and $|v\rangle$ is denoted as $\langle u|v\rangle$ where $\langle u|$ is formed as $|u^*\rangle^T$.

Definition 2.3: A vector space C^n that includes the inner product operation is called *Hilbert* space. The Hilbert space is commonly denoted $\mathcal{H}_n$.

We can span the Hilbert space $\mathcal{H}_n$ with the following $\{|0\rangle, |1\rangle, \ldots, |i\rangle, \ldots, |n\text{-}1\rangle\}$ set of basis vectors. These kets are defined as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, ..., |i\rangle = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix}, ..., |n-1\rangle = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{2.2}$$

The same Hilbert space $\mathcal{H}_n$ may be spanned by the $\{\langle 0|,\langle 1|,\ldots,\langle i|,\ldots\langle n\text{-}1|\}$ set of bra basis vectors.

A complex state vector ψ in a Hilbert space $\mathcal{H}_n$ is noted as

$$|\psi>=\alpha_0|0>+\alpha_1|1>+\ldots+\alpha_i|i>+\ldots+\alpha_{n-1}|n-1> \tag{2.3}$$

where $\alpha_0,\alpha_1,\ldots\alpha_i,\ldots\alpha_{n-1}$ are complex valued scalars. Each ket vector $|\psi\rangle$ has a dual bra vector $\langle\psi|$ which is defined as

$$\langle\psi|=\alpha_0^*\langle 0|+\alpha_1^*\langle 1|+\ldots+\alpha_i^*\langle i|+\ldots+\alpha_{n-1}^*\langle n-1| \tag{2.4}$$

Example 2.1: The following example demonstrates the *inner* and *outer* products of qubit states using the Dirac notation. Let $|\psi_1\rangle=(1, -i)^T$ and $|\psi_2\rangle=(e^{-i}, -1)^T$, then the inner product is

$\langle\psi_1|\psi_2\rangle=1\times e^{-i}+(-i)\times(-1)=e^{-i}+i$, and the outer product is

$$|\psi_1\rangle\langle\psi_2|=\begin{pmatrix}1\\-i\end{pmatrix}(e^{-i},-1)=\begin{pmatrix}e^{-i} & -1\\ -ie^{-i} & 1\end{pmatrix}.$$

Let the linear operator $\mathbf{A}^*$ be the complex conjugate of linear operator $\mathbf{A}$. For a linear operator A on a Hilbert space $\mathcal{H}_n$ there exists a Hermitian conjugate or adjoint such that $\mathbf{A}^+=(\mathbf{A}^T)^*$. The linear operator $\mathbf{A}^+$ acts on space $\mathbf{V}$ so that for all vectors $|v\rangle$, $|w\rangle\in\mathbf{V}$, we have

$$(|v\rangle, A|w\rangle) = (A^+|v\rangle,|w\rangle). \tag{2.5}$$

Definition 2.4: A matrix U is said to be ***unitary*** if $U^+U=UU^+=I$. Thus, a unitary matrix or unitary operator is reversible by definition.

Unitary operators preserve inner products between vectors so that $\langle x|y\rangle=\langle \mathbf{U}x|\mathbf{U}y\rangle$. Their determinant is unity and they preserve the norms, that is $\|x\| = \|\mathbf{U}x\|$. The rank of a unitary matrix $\mathbf{U}$ defined over C^n is n.

The importance of the unitary operation is illustrated in the second postulate of quantum mechanics which will be described in Section 2.1.3.

Definition 2.5: A $k \times k$ unitary matrix $\mathbf{U}$ is called *special unitary* and marked $\mathbf{U} \in \mathbf{SU}(k)$ if the determinant of $\mathbf{U}$ is equal to 1.

Lemma 2.1: Any special unitary 2×2 matrix U can be decomposed as follows:

$$\mathbf{U} = \begin{bmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{bmatrix} \times \begin{bmatrix} \cos(\theta/2) & \sin(\theta/2) \\ -\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \times \begin{bmatrix} e^{i\beta/2} & 0 \\ 0 & e^{-i\beta/2} \end{bmatrix} \tag{2.6}$$

for real valued θ, α, and β.

Proof: Since the rows and columns of a unitary matrix must be orthonormal, every 2×2 unitary matrix $\mathbf{U}$ can be written as

$$\begin{bmatrix} e^{i(\delta+\alpha/2+\beta/2)}\cos(\theta/2) & e^{i(\delta+\alpha/2-\beta/2)}\sin(\theta/2) \\ -e^{i(\delta-\alpha/2+\beta/2)}\sin(\theta/2) & e^{i(\delta-\alpha/2-\beta/2)}\cos(\theta/2) \end{bmatrix} \tag{2.7}$$

where δ is also real valued. For a special unitary matrix, the determinant must be 1 so that $e^{i\delta} = \pm 1$. The factorization in (2.6) then follows. □

Definition 2.6: The tensor product of $r \times s$ and $t \times u$ matrices, also called Kronecker product, is an $rt \times su$ matrix defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \ldots & a_{1s}\mathbf{B} \\ a_{21}\mathbf{B} & . & . & a_{2s}\mathbf{B} \\ . & . & . & . \\ a_{r1}\mathbf{B} & a_{r2}\mathbf{B} & \ldots & a_{rs}\mathbf{B} \end{bmatrix} \tag{2.8}$$

2.1.2. The Stern-Gerlach Experiment and the Idea of Superposition

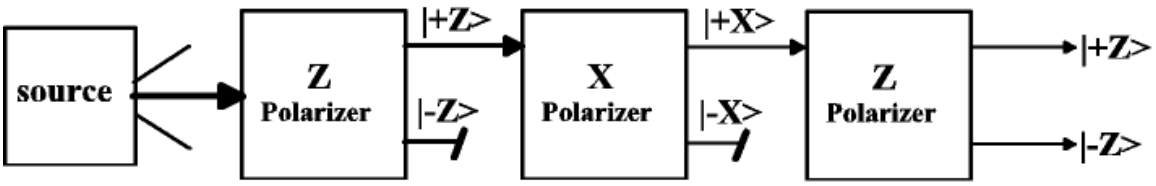


Figure 2.1. The Stern-Gerlach Experiment

The Stern-Gerlach experiment from 1922 posed one of the first paradoxes of quantum mechanics [SG22, Heis30]. The original experiment used an oven that heated silver atoms and the source particle beams were polarized using a magnetic field that operated on the particle spins. Later versions with hydrogen atoms as well as with polarized light conveyed the same paradoxical result. We assume that the source particle beams enter the first left *Z* polarizer of Fig. 2.1 and being polarized to components $|+Z\rangle$ and $|-Z\rangle$. The $|-Z\rangle$ beam is blocked and the $|+Z\rangle$ beam is sent to the second polarizer, which operates along the X basis.

The $|+Z\rangle$ is polarized again into two beams, $|+X\rangle$ and $|-X\rangle$. The results after the second *X* polarizer are indeed quite intuitive. The paradox appears on the third *Z* polarizer which seems to recreate the $|-Z\rangle$ by magic, since the $|-Z\rangle$ beam was clearly blocked after the first polarizer.

Definition 2.7: We define the *qubit* (quantum bit) model as representing the general state of a single quantum bit $|x\rangle$ with the vector:

$$|x\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.9}$$

where α and β are complex numbers satisfying the relation:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.10}$$

Extrapolating the notion of a qubit to the state of the beams in the Stern-Gerlach experiment, we can explain the paradox if we make the following assignments:

$$|0\rangle \rightarrow |+Z\rangle;\ |1\rangle \rightarrow |-Z\rangle;\ (|0\rangle+|1\rangle)/\sqrt{2} \rightarrow |+X\rangle; \text{ and } (|0\rangle-|1\rangle)/\sqrt{2} \rightarrow |-X\rangle. \tag{2.11}$$

We assume that the *Z* polarizer works on the basis $\{|0\rangle,|1\rangle\}$ and the *X* polarizer works on the basis $\{(|0\rangle+|1\rangle)/\sqrt{2}\ ,\ (|0\rangle-|1\rangle)/\sqrt{2}\ \}$. If we further assume that the state of the beam after the first *Z* polarizer is $|+Z\rangle = |0\rangle$ which equals to $(|+X\rangle + |-X\rangle)/\sqrt{2}$, it follows that the state after the second X polarizer includes a superposition of both $|0\rangle$ and $|1\rangle$, hence the recreation of the $|-Z\rangle = |1\rangle$ by the third polarizer.

2.1.3. The Four Postulates of Quantum Mechanics

While quantum mechanics deals with very complex phenomenon and provides results that may be regarded as counterintuitive, it is very interesting that the following four relatively simple postulates quite successfully define the foundation of this field.

Postulate 1 – The Hilbert Space of a Closed System *S*

A closed system *S* takes place in the quantum mechanics sense in a complex Hilbert space called the state space of *S*. As defined in Section 2.1.1, the Hilbert space is a complex vector space that includes the inner product. *S* is completely specified by the unit vector $|\psi\rangle$. Two states are equivalent *even* if they differ by a phase. Thus, $|\psi\rangle$ is equivalent to $|\psi'\rangle$ if, and only if,

$$|\psi'\rangle = e^{i\theta}|\psi\rangle \tag{2.12}$$

Postulate 2 –The Change of the State of a Closed Quantum System *S*

A closed quantum system changes in time along a unitary transformation **U**. The unitary operator **U** depends only on the times t_1 and t_2 to describe the time evolution

$$|\psi_{t_2}\rangle = \mathbf{U}|\psi_{t_1}\rangle. \tag{2.13}$$

Werner Heisenberg first formulated this relation in 1925. A few months later, M. Born and P. Jordan recognized the use of matrix algebra for this formulation [Wear67, Heis30]. Alternatively, we can describe the time evolution of the system *S* by the Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}|\psi\rangle = \mathbf{H}|\psi\rangle \tag{2.14}$$

where $\hbar$ is Planck's constant, and **H** is the fixed self-adjoint (Hamiltonian) operator of the system *S*. We note that a unitary operation is always reversible since the inverse of a unitary matrix is equivalent to its complex conjugate. Dirac and Von Neumann independently showed that equation 2.14 and 2.15 are equivalent [Vonn95, Dira58, Lawd67].

Postulate 3 – Quantum Measurements

Quantum measurements are described by a collection of measurement operators $\{M_m\}$. Each operator, called an observable, has the spectral decomposition

$$M_m = \sum_m mP_m \tag{2.15}$$

where P_m is the projector onto the eigenspace of eigenvalue *m*.

All the eigenvalues *m* of *M* describe possible results of the measurement. The probability of getting the result *m* while measuring $|\psi\rangle$ is

$$p(m) = \langle \psi | M_m^+ M_m | \psi \rangle \tag{2.16}$$

This reduces $|\psi\rangle$ to the post measurement state of $|\psi'\rangle$

$$|\psi'\rangle = \frac{M_m |\psi\rangle}{\sqrt{\langle \psi | M_m^+ M_m | \psi \rangle}} \tag{2.17}$$

Postulate 4 – Composite of Quantum States

The composite state space **H** of a group of states $\{|\psi_1\rangle, |\psi_2\rangle, |\psi_3\rangle, ..., |\psi_n\rangle\}$ is the tensor product of these states. If the subsystems are in the states $|\psi_i\rangle$, then the joint state of the entire composite system is

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle \otimes ... \otimes |\psi_n\rangle \tag{2.18}$$

2.1.4. Entanglement

A composite system $|\Psi\rangle$ of the pair $|x_i\rangle$ and $|y_j\rangle$ can be represented as

$$|\Psi\rangle = \sum_i \sum_j \alpha_{ij} |x_i, y_j\rangle \quad \text{with} \quad \sum_i \sum_j |\alpha_{ij}|^2 = 1 \tag{2.19}$$

Definition 2.8: If the coefficients α_{ij} in (2.16) can be written as $\alpha_{ij} = \beta_i \gamma_j$, the system is in a *decomposeable* state. Otherwise, the system is in an *entangled* state.

For example, the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is entangled, while the state

$\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ is decomposable.

The property of entanglement is crucial in many of the important algorithms that portray the major benefits of QC, including superdense coding and quantum teleportation [NC00, KLM07].

2.1.5. Measurement and Decoherence in Quantum Mechanics

Measurement in quantum mechanics cause any state held at superposition to collapse to an observable result. *Decoherence* is the phenomenon by which a quantum state held at superposition tends to "decay" into the basis states, resulting in a loss of the quantum state information (similar to noise in classical computing and communication theory).

In the following discussion we show that the measurement of an observable of a quantum system comprised of a single qubit will always yield an eigenvalue of the Hermitian matrix A representing the measurement operator.

We first need to prove the following fundamental Theorem:

Theorem 2.1: The eigenvalues $\lambda_i, \lambda_j, \ldots$ of a Hermitian operator **A** are real, and the eigenkets $\zeta_i, \zeta_j, \ldots$ of **A** corresponding to different eigenvalues $\lambda_i, \lambda_j, \ldots$ are orthogonal.

Proof: Following [Saku94], by definition of eigenvalues and eigenvectors

$$\mathbf{A} \mid \zeta_i \rangle = \lambda_i \mid \zeta_i \rangle \tag{2.20}$$

and because of the Hermitian properties,

$$\langle \zeta_j \mid \mathbf{A} = \lambda_j^* \langle \zeta_j \mid \tag{2.21}$$

By left multiplication of both sides of equation (2.14) by $\langle \zeta_j |$, right multiplication of both sides of equation (2) by $| \zeta_j \rangle$, and subtracting the two results, we get

$$(\lambda_j - \lambda_j^*)\langle \zeta_j | \zeta_i \rangle = 0 \tag{2.22}$$

First, let us assume that λ_j, λ_i are the same eigenvalue. Since $| \zeta_i \rangle$ is not the null ket, we must have $\lambda_j = \lambda_j^*$, which means that the **eigenvalues are real**.

If the eigenvalues are different, then since they are real, the first multiplicand of (2.22) cannot disappear. Thus we obtain the **orthogonality property**

$$\langle \zeta_j | \zeta_i \rangle = 0, \quad (\zeta_j \neq \zeta_j) \tag{2.23}$$

This completes the proof. □

Before the measurement of the observable **A**, we can represent our system $|\psi\rangle$ (not necessarily just the single qubit of the question) by the following linear combination:

$$| \psi \rangle = \sum_{\lambda} c_\lambda | \lambda \rangle \tag{2.24}$$

We now multiply on the left with $\langle\lambda|$ and by applying the orthogonality property from theorem 2.1, we find that for all λ,

$$c_\lambda = \langle \lambda | \psi \rangle \tag{2.25}$$

and consequently, equation (2.17) becomes

$$| \psi \rangle = \sum_{\lambda} | \lambda \rangle \langle \lambda | \psi \rangle \tag{2.26}$$

When the measurement is performed, the system is "thrown to" one of the eigenstates of the observable A, say $|\lambda\rangle$. This is what Dirac meant in his 1958 book regarding the postulate that "a measurement always causes the system to jump into an eigenstate of the dynamical variable that is being measured." In this sense, the result of the measurement yields one of the eigenvalues of the observable being measured. Since the observables are the quantum analogue of dynamical variables (e.g. position, spin, linear and angular momentum, energy, polarization), the theorem stating that the eigenvalues of the Hermitian operator **are real** is crucial. In other words, we cannot observe "complex" outcomes of measurements.

We note, of course, that if $|\psi\rangle$ is already in the state $|\lambda\rangle$, the measurement by **A** yields the result λ with certainty.

However, postulate 3 of quantum mechanics states that the probability of the system being thrown to or collapsed to a particular eigenstate $|\lambda\rangle$ is

$$P_\lambda = \left|\langle \lambda \,|\, \psi \rangle\right|^2 \tag{2.27}$$

This postulate insures us that the probability conforms to the requirements that each eigenvalue of **A** has a nonnegative probability. It also provides

$$\sum_\lambda P_\lambda = 1 \tag{2.28}$$

Since we deal here with a single qubit, the Hermitian observable **A** has two eigenvalues, λ_1 and λ_2 each with a probability as shown in (2.24). It should be noted that it is possible to have the original qubit already prepared in the eigenstate $|\zeta_1\rangle$. In this

case, the same observable **A** will result in the eigenvalue λ_1 with certainty. This is similar to the case of measurements in repeated tests [MM05].

The measurement problem is one the key concepts of quantum mechanics. An interesting discussion relates to the famous Schrödinger's Cat Paradox [Schr35]. The single quantum bit has the basis states of ALIVE and DEAD, and while the cat is in the box, its state $|\psi\rangle$ is the superposition of

$$|\psi\rangle = \alpha_1 |\, ALIVE\rangle + \alpha_2 |\, DEAD\rangle \tag{2.29}$$

A common superposition with 50/50 chance for the cat being alive is

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}(|\, ALIVE\rangle + |\, DEAD\rangle) \tag{2.30}$$

The measurement observable **A** of this qubit is performed by simply opening the box in which the "superpositioned" cat is enclosed. When the box is opened and a measurement is made, the system is "thrown to" (or collapses to) only one of the eigenvalues of the observable **A** – *ALIVE* or *DEAD*. Clearly, the observable **A** has the form

$$\mathbf{A} = \begin{pmatrix} ALIVE & 0 \\ 0 & DEAD \end{pmatrix} \quad \text{or} \quad \mathbf{A} = \begin{pmatrix} DEAD & 0 \\ 0 & ALIVE \end{pmatrix} \tag{2.31}$$

This observable was the basis of a huge debate at the heart of quantum physics: "What mechanism converts the probabilities of live/dead into such a sharp outcome?"

Briefly, there were the *Copenhagen Interpretation* (QC deals only with the probabilities of the observable quantities – all other quantities are just meta-physical), the *Quantum Decoherence* (the macroscopic nature of the measurement apparatus

allows physicists to distinguish the fuzzy boundary between quantum world and actual world) and other philosophies (non-scientific like "conscious collapse" etc.) [Bohm01, BCS04, Hirv04, MM05, Omne99].

A very detailed and modern discussion of the measurement problem and decoherence appears in Schlosshauer [Schl05]. Another extensive discussion appears in Chapter 3 of the original class notes from 1997 of John Preskill of Caltech [Pres97].

2.2. Reversible Logic

Quantum computation is reversible in nature. In this section we present the foundation of reversible logic. We show how reversible logic can be regarded from the point of view of group symmetry. Efficient reversible circuits that minimize garbage inputs and outputs are built in the form of cascades of gates that are defined in Section 2.2.3. We also discuss simple implementations of common logic functions that may be obtained directly from Toffoli and Fredkin gates.

2.2.1. Defining Reversible Logic

Definition 2.9: A gate or a circuit is logically *reversible* if it maps each input pattern to a unique output pattern. For classical reversible logic, the mapping is a permutation matrix. For quantum circuits, the gate or circuit operation can be described by a unitary transformation matrix. Another way to describe the notion of logically reversible circuits is to note that the functions describing the behavior of the circuits are bijective.

Reversible gates or circuits can be designed with a different number of inputs and outputs [Alra04b]. In such cases, not all the patterns of the largest set of outputs or inputs are used. However, in the main research stream and in the remainder of this dissertation we are concerned only with $n \times n$ reversible circuits with equal number (n) of inputs and outputs. We should note inputs/outputs of reversible circuits are commonly referred to as "variables" or "lines" in this dissertation and in the literature [MM05, NC00].

Bennett showed that reversible gates can theoretically result in binary circuits that are completely free from energy loss [Benn73]. The concept of reversibility has been extended to *multiple-valued logic* (MVL) circuits [MDM04, Alra04b]. Binary quantum logic gates and circuits are inherently both logically and physically reversible [NC00]. Non-binary quantum logic circuits are logically reversible.

A variety of basic reversible gates have been proposed in the past few decades. In Chapter 1 we illustrated a 3-variable Toffoli gate in Fig. 1.1. We can easily extend the 3-variable Toffoli gate to an arbitrary n-variable Toffoli gate as shown in Fig. 2.2. It has n-1 control lines (filled circles) and one target line (open circle). For each gate, the value on the target line is negated if, and only if, all the n-1 control lines are set at '1'. We denote a n-variable Toffoli gate as TOF(x_0,x_1,…,x_{i-1},x_{i+1},…,x_{n-1};x_i), where the target line x_i is separated by a semicolon from the control lines.

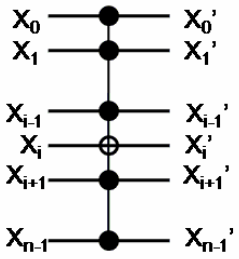


Figure 2.2. n-variable Toffoli Gate

A common alternate notation for this gate is $[t,C]$ where t is the target line and C comprises the set of control lines. With this notation, the gate of Fig. 2.2 is denoted $[x_i, \{x_0,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_{n-1}\}]$.

A library of universal gates is a set of gates that can implement any arbitrary quantum circuit. We encounter classical reversible libraries like the NCT (not, control not, Toffoli), NCTF (NCT with Fredkin gate) and NCTFS (NCTF with the swap gate).

2.2.2. Symmetry in Reversible Logic

Let $\boldsymbol{\rho}$: $B^n \rightarrow B^n$ be a binary logic circuit with n inputs and outputs, where $B = \{0, 1\}$.

If $\langle X_1,\ldots, X_n \rangle \in B^n$ *and* $\langle Y_1,\ldots, Y_n \rangle \in B^n$ be the input and output vectors respectively, then there are 2^n different assignments for the input vectors. A binary logic circuit $\boldsymbol{\rho}$ is reversible if, and only if, it is a one-to-one and onto function. Such a function is called a bijective function.

A binary reversible logic circuit with n inputs and n outputs is also called an n-qubit binary reversible gate. There are a total of $(2^n)!$ different n-qubit binary reversible circuits.

A bijective mapping s: $M \rightarrow M$ can be written as

$$s = \begin{pmatrix} x_1, x_2, x_3, ..., x_n \\ x_{s1}, x_{s2}, x_{s3}, ..., x_{sn} \end{pmatrix} \tag{2.32}$$

and is called a *permutation*.

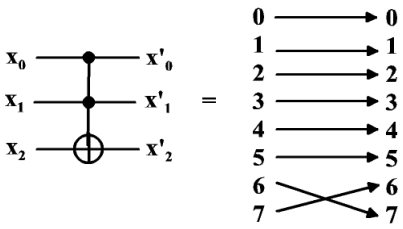


Figure 2.3. Toffoli Gate as a cycle(6,7) Operation

All the possible permutations form a group under the composition of mappings and is denoted by S_k [GFX06]. In Fig. 2.3 we demonstrate how the Toffoli $T(x_0,x_1;x_2)$ gate is equivalent to *cycle(6,7)* operation when considered as a permutation.

2.2.3. Reversible Gates Cascade

Reversible circuits are synthesized in the form of a cascade of gates. Fig. 2.4 shows the benchmark circuit "hbw5" which is the 5-variable hidden weighted bit function from the Maslov web page [Masl05]. The circuit uses 5 lines that represent 5 inputs and 5 outputs. Each vertical line depicts a controlled NOT gate (if the gate operates on only two lines) or generalized n-input Toffoli gates.

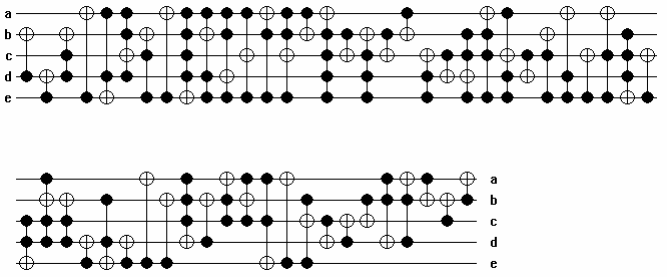


Figure 2.4. The 5-variable "hwb5" Cascade of Gates

This cascade example includes 54 gates illustrating the inherent complexity of quantum circuits. The transformation matrix of the entire circuit is computed by repeated matrix multiplication of the individual gates and the tensor product among the individual lines which results in an overall circuit transfer matrix of dimension 32×32 ($2^5 \times 2^5$).

Definition 2.10: An *n*-variable reversible gate cascade is composed of adjacent reversible gates that operate on the same *n* variables represented by horizontal lines across the circuit. Each gate may be connected to one or more of the lines and must be extended via tensor product to affect all *n* lines.

Cascading of two reversible gates is equivalent to the multiplication of the two permutation matrices representing the corresponding gates. We demonstrate the computation of the transformation matrix on the smaller cascade of four gates shown in Fig. 2.5.

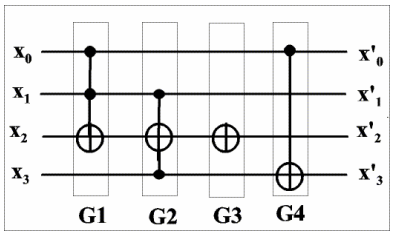


Figure 2.5. A Cascade of 4 Gates with 4 Variables

Each gate is part of a stage of the cascade which must include the gate and all the unconnected lines running through it. For example, gate *G3,* is a reversible NOT gate with a 2×2 transformation matrix. However, in view of the 4 variables that the circuit depends upon, it must have a 16×16 transformation matrix. We thus extend the reversible NOT operation on variable x_2 into a 16×16 transformation matrix using the tensor product with the 2×2 identity matrix $\mathbf{I}_2$

$$G4 = \mathbf{I}_2 \otimes \mathbf{I}_2 \otimes \mathbf{NOT} \otimes \mathbf{I}_2 \tag{2.33}$$

The overall transformation matrix **C** of this cascade is obtained by multiplying the transformation matrices of the 4 gates in reverse order [MM05, NC00]. Thus,

$$\mathbf{C} = G4 \times G3 \times G2 \times G1 \tag{2.34}$$

2.2.4. Logic Function Realization with Classical Reversible Gates

Many more classical reversible gates have been introduced including the Feynman (Control NOT), the Fredkin gate and the swap gate [FT82]. We will re-visit these gates as well as quantum reversible gates in the next Section. The Fredkin gate is defined in Fig. 2.6. Using only Toffoli and Fredkin gates, we show in Fig. 2.3 the

implementation of common classical logic operations (NAND, AND, NOT, and fan-out duplication). It is clear that once the NAND gate or the equivalent set of AND and NOT gates are achievable, any arbitrary logic function can be implemented. Note that these implementations require the extra cost of auxiliary inputs (also called ancilla inputs), which are crucial to the operation of the gates. For example, the Fredkin NOT and Fan-out implementation uses only one input for the input variable *x*, while the remaining gate inputs become ancillary inputs, initialized to either 0 or 1.

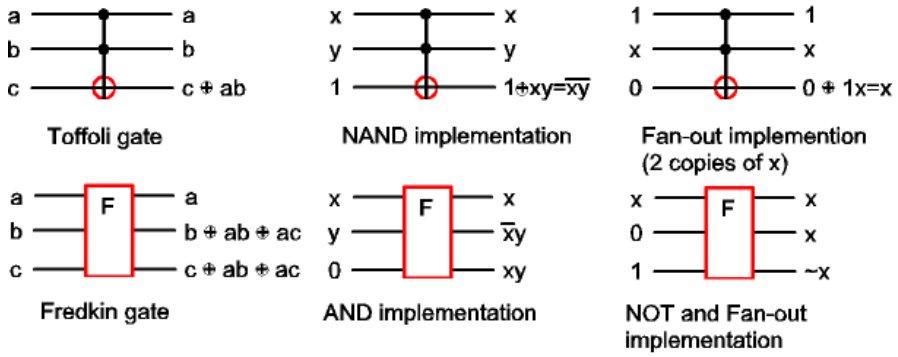


Figure 2.6. Logic Functions by Toffoli and Fredkin Gates

Since the number of outputs for reversible logic gates must equal the number of inputs, we find that reversible circuits can create a large number of unused or "garbage" outputs. Therefore, a major constraint in the synthesis, specification, and simulation of quantum circuits is the need to minimize the garbage outputs and the ancillary inputs.

2.3. Quantum Computation and Quantum Information

The pioneering works of Feynman, Deutsch, Sozja, Shor, and Grover have spawned an immense interest in quantum computation and quantum information [RP00]. In Section 2.3.1 we briefly discuss their work to justify our interest in QC. We then describe the building blocks of quantum computation, starting with quantum bits (qubits) and quantum registers, and following with the basic quantum operations and gates. We conclude this section with the important Beck-Zeilinger-Bernstein-Bertani decomposition Theorem.

2.3.1. Quantum Computing Algorithms

The promise of QC was staked on the success of four important algorithms: *dense coding*, *quantum teleportation*, *quantum prime factoring*, and *quantum search*. The first two algorithms depend on the entanglement phenomenon. Quantum prime factoring and quantum search depends on the ultimate parallelism due to the superposition phenomenon. Both phenomena have no counterparts in classical computing.

In dense coding, a quantum communication channel allows for sending exponentially more information using entangled qubits rather than classical bits. For example, one entangled qubit has the information content of two classical bits [KLM07].

The important *no-cloning theorem* states that an unknown quantum state cannot be copied [NC00]. Bennett et al. discovered that quantum teleportation enables the

transmission of quantum states from point A to point B, using entangled pairs of qubits [BBC+93]. However, the original quantum state is destroyed in the process, thus complying with the no-cloning theorem.

Peter Shor's prime factorization uses repeated reduction of the factoring problem into an order finding problem and phase estimation problem. The solution for the later problem is based on the *quantum fast Fourier transform* (QFFT) and illustrates the inherent parallelism of QC [Shor94]. The Deutsch and Sozja algorithm also relates to QFFT [DJ92].

The quantum search algorithm is also based on exploiting the parallelism of superposition and was proposed by Grover [Grov96]. A quantum register with the search data is initialized and then transformed into a superposition by the Walsh-Hadamard transform (see Section 2.3.5). A repeated Grover's iteration step which utilizes a special oracle is performed to "amplify" the searched items. This algorithm breaks the classical barrier of time complexity from $O(n)$ to $O(\sqrt{n})$ when searching a dataset of n items.

An important group of algorithms deals with simulating quantum systems as suggested by Feynman [Feyn85]. Ultimately, a future quantum computer will be the best choice for simulation of quantum systems. In the meantime, results such as our contribution in simulation (Chapter 3) illustrate the potentials and pitfalls when using classical computers to simulate quantum systems.

2.3.2. Quantum Bits and Registers

The basic computing element in quantum computing is the *qubit*– the quantum bit that was defined in section 2.1.2. A qubit exists in a Hilbert space $\mathcal{H}_2$ where it represents the superimposed state of the smallest unit of computational data. The simplest basis states that span $\mathcal{H}_2$ are

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.35}$$

Equation (2.7) showed that the qubit quantum state in the basis $\{|0\rangle, |1\rangle\}$ can be described as $|x\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers satisfying the relation $|\alpha|^2 + |\beta|^2 = 1$.

Before we measure the qubit state, it still represents a superposition of $|0\rangle$ and $|1\rangle$. Once we actually measure the quantum state, we obtain the state $|0\rangle$ with a probability of $|\alpha|^2$ and state $|1\rangle$ with probability $|\beta|^2$. Clearly, this superposition and the dependence on measurement is significantly different than the familiar classical computing where determinism ensures that '1' is a '1' and '0' is '0'.

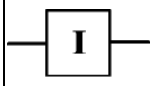
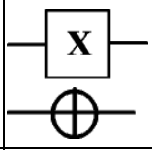
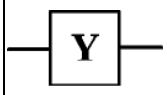
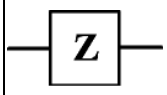
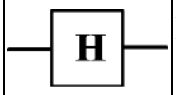
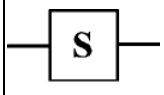
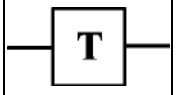
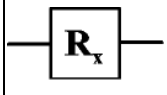
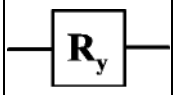
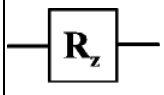
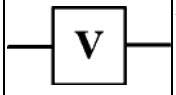
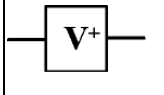
A system of two qubits comprises a four-dimensional Hilbert space $\mathcal{H}_4$ with the standard basis set of $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Obviously, other orthonormal basis sets can be used. This system is a *quantum register of two qubits*. Larger *quantum registers* are composed using the tensor product according to the 4th postulate of quantum mechanics (2.18). *The superposition of the entire basis set in a quantum register is the source for*

the parallelism potential of QC. There is no parallel for this phenomenon in classical logic.

2.3.3. Quantum Operations on a Single Qubit

The quantum operation on a qubit must be reversible and preserve the norm and the inner product. Therefore they are represented by 2×2 unitary matrices. Although there is an infinite number of 2×2 unitary matrices, we list some important single qubit operations in Table 2.1.

Table 2.1. Single Qubit Operations

Name	Unitary matrix	Symbol	Name	Unitary matrix	Symbol
Pauli $\sigma_0 \equiv I$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$		Pauli $\sigma_1 \equiv \sigma_x \equiv X$ NOT	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	
Pauli $\sigma_2 \equiv \sigma_y \equiv Y$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$		Pauli $\sigma_3 \equiv \sigma_z \equiv Z$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	
Hadamard	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$		Phase	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	
$\pi/8$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$		$R_x(\theta)$	$\begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$	
$R_y(\theta)$	$\begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$		$R_z(\theta)$	$\begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$	
V Square root of NOT	$\frac{1+i}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$		$V^+ = V^{-1}$	$\frac{1-i}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$	

2.3.4. Elementary Quantum Gates and Quantum Circuits

Feynman proposed the general notation for a controlled operation shown in Fig. 2.7 as an extension of a qubit operation that works on more than one variable [Feyn85]. This is similar to the *n*-variable Toffoli gates we have shown in Fig. 2.2, except that we use any general single qubit operation specified by the 2×2 *unitary matrix,* **U***,* instead of the NOT operation used in classical reversible logic. Again, the control lines with dots are transferred without a change. The lower target line performs the unitary operation **U** only when all the control lines are set to $|1\rangle$. There is no limit on the number of control lines.

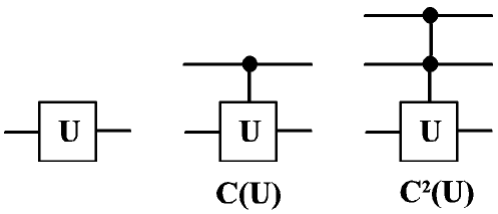


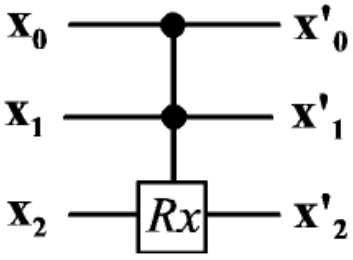
Figure 2.7. The General Controlled Operation

An alternate notation for the control operation is $\wedge_{n-1}(\mathbf{U})$ as used in the seminal paper on elementary quantum gates by Barenco et al. [BBC+95]. Unlike the *n*-variable Toffoli gate shown in Fig. 2.2, the $\wedge_{n-1}(\mathbf{U})$ notation is always normalized so that the control lines precede the target line. The transformation matrix for an *n*-variable $\wedge_{n-1}(U)$ is the $2^n \times 2^n$ matrix

$$\wedge_n(U) = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & ..... & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & u_{00} & u_{01} \\ & & & & & & u_{10} & u_{11} \end{pmatrix} \tag{2.36}$$

Where the single qubit operation **U** is $\begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$.

A variety of basic reversible gates that are realizable using quantum devices include the Deutsch-Toffoli, Controlled-NOT, Controlled-V, and controlled V^+ and many more [KSV02]. Fig. 2.8 illustrates the Deutsch-Toffoli gate $\wedge_2(R_x)$ which was shown to be a universal gate for quantum computing [Deut89]. We note that many other sets of quantum universal gate sets have been proposed.



$$\equiv \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

Figure 2.8. The Deutsch-Toffoli Universal Gate

The Deutsch-Toffoli gate is a two-level unitary matrix. The two-level unitary matrix is formally defined as follows.

Definition 2.11: A two-level unitary matrix transforms only two or fewer vector components. Therefore, a $n \times n$ two-level unitary matrix V that implements the unitary operation $\mathbf{U} = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$ is an identity matrix $\mathbf{I}_n$ with two or fewer diagonal '1's replaced by the unitary components of **U**. For example, equation (2.38) is an 8*x*8 two-level unitary matrix, while a 4*x*4 two-level unitary matrix V, may appear as

$$\mathbf{V} = \begin{bmatrix} u_{00} & 0 & u_{01} & 0 \\ 0 & 1 & 0 & 0 \\ u_{10} & 0 & u_{11} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.37}$$

Quantum circuits are represented as gate cascades and are of a form in a similar way to Figure 2.5 where the large variety of single qubit operations can replace the Toffoli NOT operator. We should note that a quantum gate cascade is actually a quantum register with control and time flowing from left to right [Cybe01]. It is for this reason that the overall circuit transformation matrix is formed as a product of gate component transformation matrices from right to left.

When synthesizing quantum and classical reversible circuits, it is often convenience to have a standardized cost measure that can help in the comparison of competing implementations. The following definition of the quantum cost of a circuit is used throughout this dissertation [Masl03]

Definition 2.12: The quantum cost of a circuit *C* is the number of elementary quantum operations that are required to implement the function of *C*.

The number of elementary quantum operations is usually determined in accordance with Barenco et al. interpretation of elementary quantum gates [BBC+95].

2.3.5. The Walsh-Hadamard Transformation

The Hadamard single bit operation that was described in Table 2.1 is very important in QC. When applied to a qubit initialized to state $|0\rangle$, it creates the evenly distributed superposition

$$\mathbf{H}\,|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,. \tag{2.38}$$

When similar transformation is applied to a quantum register of n-qubits, it creates a superposition of all the numbers from 0 to 2^n-1. Such a transformation is called a Walsh-Hadamard transform and is defined using Dirac notation as follows:

$$H \otimes H \otimes \ldots \otimes H\,|00..0\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \ldots \otimes (|0\rangle + |1\rangle) =$$

$$= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{2.39}$$

An alternative recursive definition for the Walsh-Hadamard transformation matrix is

$$W_1 = H\,, \quad W_n = H \otimes W_{n-1} \tag{2.40}$$

The Walsh-Hadamard ability to create arbitrary large qubit register superposition states makes it crucial for QC and we will revisit it in the next chapter.

2.3.6. The Beck-Zeilinger-Bernstein-Bertani Theorem

The beam splitter is an important component for quantum optical experiments. The function of a beam splitter can be represented by a two-level unitary matrix. Beck, Zeilinger, Bernstein and Bertani proved the following important theorem in their quest to show that any arbitrary finite dimension unitary matrix can be implemented in the laboratory by beam splitters [BZBB94]. Expressed in the context of two-level unitary matrices, the theorem is stated as follows:

Theorem 2.2: An arbitrary $k \times k$ unity matrix **U** can be decomposed into a product of two-level unitary matrices so that

$$\mathbf{U}=\mathbf{V}_1\mathbf{V}_2....\mathbf{V}_n \tag{2.41}$$

and

$$n \le k(k-1)/2 \tag{2.42}$$

Proof: (By construction.) We first reduce **U** into a $k \times k$ matrix $\mathbf{U}_1$ that has 1 on its first diagonal element and 0s elsewhere on the first column and first row. We perform this task by multiplying **U** by up to k-1 two level unitary matrices, each designed to achieve one 0 on the first column and first row. If **U** already has 0s in the first column and/or column, the required number of two level unitary matrices is reduced accordingly. In the next step, we reduce $\mathbf{U}_1$ into a $k \times k$ matrix $\mathbf{U}_2$ which now has 1s on the first two elements of the diagonal with 0s elsewhere on the first and second column and row. This step requires up to k-2 multiplications of $\mathbf{U}_1$ by two-level unitary matrices.

This process continues recursively until **U** is fully reduced to the identity matrix by

$$\mathbf{V}_1\mathbf{V}_2...\mathbf{V}_n\mathbf{U}=\mathrm{I} \tag{2.43}$$

Since all the matrices on the left side of (2.42) are unitary, we obtain the desired decomposition

$$\mathbf{U} = \mathbf{V}_1^{+}\mathbf{V}_2^{+}...\mathbf{V}_n^{+} \tag{2.44}$$

The total number n of two level matrices is smaller than

$$n \le (k-1)+(k-2)+...+1 = k(k-1)/2 \tag{2.45}$$

This completes the proof. □

Example 2.2**:** We are going to use the BZBB decomposition process to investigate a phenomenon of *skipped variables* occurring in our QMDD tools to be described in Chapter 3. Therefore, it is important to demonstrate this decomposition with the following 4*x*4 unitary matrix **U** (taken from [NC00] exercise 4.37).

$$\mathbf{U} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5i & -0.5 & -0.5i \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5i & -0.5 & 0.5i \end{bmatrix} \tag{2.46}$$

We first calculate the two-level unitary V_1 that will produce '0' in the second element of the first column (a_{21}) to obtain

$$\mathbf{V}_1 = \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ 0.7071 & -0.7071 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \quad \mathbf{V}_1\mathbf{U} = \begin{bmatrix} 0.7071 & 0.3536+0.3536i & 0 & 0.3536-0.3536i \\ 0 & 0.3536-0.3536i & 0.7071 & 0.3536+0.3536i \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5i & -0.5 & 0.5i \end{bmatrix} \tag{2.47}$$

The two-level unitary V_2 should produce '0' in a_{31} to obtain

$$\mathbf{V}_2 = \begin{bmatrix} 0.8165 & 0 & 0.5774 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5774 & 0 & -0.8165 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 0.866 & 0.2887i & 0.2887 & -0.2887i \\ 0 & 0.3536 - 0.3536i & 0.7071 & 0.3536 + 0.3536i \\ 0 & 0.6124 + 0.2041i & -.4083 & 0.6124 - 0.2041i \\ 0.5 & -0.5i & -0.5 & 0.5i \end{bmatrix} \quad (2.48)$$

The two-level unitary V_3 should produce '0' in a_{41} to obtain

$$\mathbf{V}_3 = \begin{bmatrix} 0.866 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0 & 0.866 \end{bmatrix} \rightarrow \mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.3536 - 0.3536i & 0.7071 & 0.3536 + 0.3536i \\ 0 & 0.6124 + 0.2041i & -0.4083 & 0.6124 - 0.2041i \\ 0 & 0.5774i & 0.5774 & -0.5774i \end{bmatrix} \quad (2.49)$$

The two-level unitary V_4 should produce '0' in a_{32} to obtain

$$\mathbf{V}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.433 + 0.433i & 0.75 - 0.25i & 0 \\ 0 & 0.75 + 0.25i & 0.433 + 0.433i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \mathbf{V}_4\mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.8165 & 0.4082i & 0.4082 \\ 0 & 0 & 0.7071 & 0.7071i \\ 0 & 0.5774i & 0.5774 & -0.5774i \end{bmatrix} \quad (2.50)$$

The two-level unitary V_5 should produce '0' in a_{42} to obtain

$$\mathbf{V}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.8165 & 0 & -0.5774i \\ 0 & 0 & 1 & 0 \\ 0 & 0.5774i & 0 & 0.8165 \end{bmatrix} \rightarrow \mathbf{V}_5\mathbf{V}_4\mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071i \\ 0 & 0 & -0.7071 & 0.7071i \end{bmatrix} \quad (2.51)$$

The decomposition in the right side of (2.51) is already in the form of a final two-level matrix. Therefore, the two-level unitary matrix V_6 is essentially the simple inverse of the last decomposition to achieve the identity matrix.

$$\mathbf{V}_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0 & -0.7071i & -0.7071i \end{bmatrix} \rightarrow \mathbf{V}_6\mathbf{V}_5\mathbf{V}_4\mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0. & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.52)$$

The required decomposition of U is therefore

$$\mathbf{U} = \mathbf{V}_1^+\mathbf{V}_2^+\mathbf{V}_3^+\mathbf{V}_4^+\mathbf{V}_5^+\mathbf{V}_6^+ \quad (2.53)$$

as each $\mathbf{V}_i^+, 1 \le i \le 6$ is a two-level unitary operation. It is important to note that we have actually run this example (using *Maple 11*) at the high precision of 8 decimals. We use rounding here for conciseness. □

Cybenko pointed out the similarity of this construction to the classical triangularization or QR factorization of a general matrix. Since we start with a unitary matrix, one can expect the factorized upper triangular matrix to be diagonal [Cybe01]. Equation (2.43) indeed shows that the BZBB decomposition process reaches the identity matrix. Cybenko treats the general unitary matrix of $n \times n$ as a depiction of the n^{th} dimension on which Schrödinger's unitary time evolution equation operates. Accordingly, he denotes a two-level unitary matrix as a 2D unitary operation which performs the Givens rotation operation.

2.3.7. Decoherence and Fault Tolerance for Quantum Circuits

QC reliance on extensive data communication is further impacted by the **decoherence** problem. Decoherence is the tendency of a quantum state held at superposition to "decay" into the basis states, resulting in a loss of the quantum state information (like noise in classical computing). Isailovic et al. showed that decoherence causes an extremely large error rate of single quantum gate of 10^{-3} [IPWK06]. While near future enhancements are projected to improve this failure rate to 10^{-8}, it is still many orders of magnitude below the standard CMOS gate's error rate of 10^{-19}. Therefore, QC must rely extensively on *quantum error correction codes* (QECC) for proper operation [Stea96]. All current and proposed implementations for quantum

computing typically use [*n,k,d*] quantum error correcting codes, where a total of *n* qubits are used to encode *k* qubits of data, with distance *d*. This code is capable of correcting the error set $\{E_i\}$ if and only if

$$PE_i^{+}E_jP = \alpha_{ij}P \tag{2.54}$$

where α is an Hermitian matrix with complex elements and *P* is a projector into the code [NC00, Niel98].

After an error is detected, a properly designed fault-tolerant QC circuit must activate its error recovery means. Such a QC circuit can tolerate a single-fault failure probability *p* in the circuit's gates provided the measurement result reported has probability of error of $O(p^2)$ [AB97].

The important **threshold theorem** for quantum computation states that an arbitrarily complex QC circuit can work reliably, provided the error probability *p* of each individual gate is below a certain *threshold* [NC00]. While the QC circuit still requires some reasonable overhead in size for QECC and other noise avoidance circuits, this theorem is heralded by QC researchers as the light at the end of the tunnel that leads to the emergence of real quantum computers.

2.4. Quantum Cellular Array Fundamentals

In this section we introduce the basic structure of the *quantum cellular array* (QCA). We also discuss the QCA native 3-input majority gates ability to implement the basic logic functions.

2.4.1. The Quantum Cellular Array Basic Cell and Wires

The basic QCA cell can be schematically depicted as a square substrate with four quantum dots located at the four corners [LTP93]. Each cell has only two electrons so that they tend to be in the two bistable arrangements shown in Fig. 2.9.a. The electrons cannot occupy two adjacent dots due to the columbic repulsion between them. We arbitrarily select the first cell as logic 1 encoding, leaving the second cell to encode logic 0.

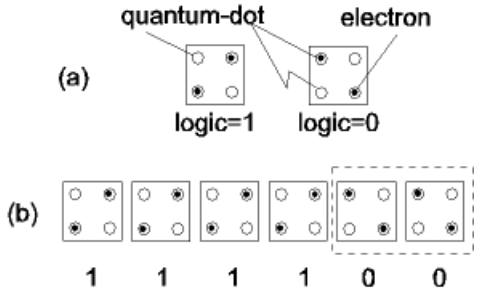


Figure 2.9. Basic QCA Cell (a) and (b) QCA Wires

The same columbic repulsion aligns the data within the cells when they are arranged along a wire as shown in Fig. 2.9.b. When two electrons occupy quantum dots that are near one another, the electrostatic Coulombic repulsion between them causes one of the electrons to move to another quantum dot further in distance by using the quantum effect of "tunneling". Here we assume that information flows from left to right and the cells in the dotted box are next to respond. Various clocking mechanisms, as described in [WJ06] are used to enforce the proper or desired direction of data

propagation. It should be noted that since no current is flowing between the cells, QCA is essentially a very low power technology (even though current is used, and hence power is dissipated, in some implementations for the clocking zones [XCN+06]).

2.4.2. The Native Gates of QCA

The native gate for QCA is created when five cells are placed adjacently in the geometric configuration as shown in Fig. 2.9. The result is a logical majority gate with three inputs and one output. This three input majority gate performs the Boolean operation described by the binary function

$$M(a,b,c) = ab + bc + ac \; . \tag{2.55}$$

Basic two variable AND and OR functions can be readily implemented by M(a,b,0) and M(a,b,1), respectively. However, the majority gate is not a universal gate since it is unable to implement the NOT function. Therefore, majority logic gates must be combined with inverter circuits in order to implement any possible arbitrary logic function.

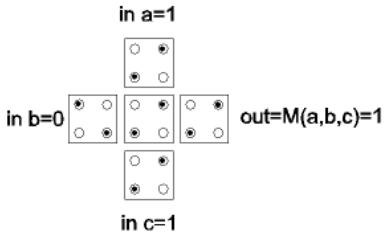


Figure 2.10. QCA 3-input Majority Gate

Fig. 2.11 shows a fork inverter. The output cell is affected by both end cells of the fork to insure reliable operations. Other inverters that use 45^0 and 90^0 cells orientations have been developed [HMS+05].

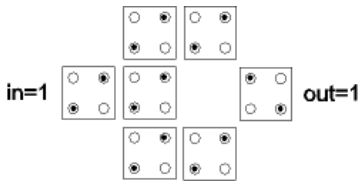


Figure 2.11. QCA Fork Inverter

Recently, the QCADesigner synthesis and simulation tool was developed by Walus et al. [WDJB04, WJ06, WSJ+04], allowing convenient experimentation with virtual QCA circuits.

CHAPTER 3

REPRESENTATION AND SIMULATION OF QUANTUM CIRCUITS

Richard Feynman's demonstration that some quantum mechanics effects cannot be simulated properly on classical computers prompted the initial interest in quantum computing and quantum information [Feyn82]. However, since quantum computers are not yet available, we must find ways to use classical computers based on conventional deterministic logic operations to represent and simulate quantum circuits. We begin by discussing the challenge of quantum circuit representation and simulation and briefly survey the previous work. We introduce the *quantum multi-valued decision diagram* (QMDD) in some detail in Section 3.2. We then describe our contribution to QMDD minimization in sections 3.3 and 3.4. We conclude this chapter with our work on QC simulation.

3.1. Quantum Simulation on Classical Computers

It is quite clear that classical computers cannot fully simulate quantum phenomena since it often relates to an infinite series of evolution matrices [Omne99]. However extensive research has been expended with some degree of success to simulate specific aspects of quantum networks. Quantum programming languages like Q-gol, qGCL, Quantum C, and QCL have been proposed in the past decade and have been

successfully used in various QC simulation environments [Omer03]. A reader interested in more details on such QC simulators may consult Julia Wallace's survey at [Wall01]. In this dissertation we mainly consider the use of decision diagrams for QC simulation.

3.1.1. Classical Binary Decision Diagrams (BDD)

Binary decision diagrams (BDD) were developed by Akers [Aker78] and can be considered to be an exploitation of the Shannon expansion theorem for representing complex Boolean functions as a compact directed graph [Shan48]. In 1986, Bryant's seminal paper [Brya86] introduced *reduced order binary decision diagrams* (ROBDD) that reduce the size of a BDD and create a canonic representation of the switching function utilizing:

- Redundant node removal.
- Sharing of isomorphic graphs.

All modern BDD packages (like the popular F. Somenzi CUDD package [Some95]) are based on Bryant's ROBDD so that mentioning BDD inherently implies a ROBDD. Depending on the switching function that is being represented, BDDs can suffer from memory overflow as the number of variables grows. Various heuristics for variable ordering and reordering have been developed to reduce the size complexity of BDDs [FS90, FMK91, ISY91, PS95, Rude93]. The BDD size minimization for incompletely specified functions was shown to be NP-complete by Sauerhoff and Wegener [SW94]. It was later shown that the problem is also NP-Complete for the general ROBDD by Bollig and Wegener [BW96].

3.1.2. Early Quantum Circuit Simulation Tool Using BDDs

In 1999, Greve created a *quantum decision diagram* (QDD) tool based on Jorn Lind-Nielsen's BuDDy package [Grev99]. Greve was able to successfully simulate Shor's quantum factoring algorithm using the QDD structure he developed. The program was able to factor a 16 bit number in about 8 minutes on a Pentium 200MHz computer with 64M of RAM

Ablayev et al. [AGK01] proposed the use of a quantum circuit BDD in 2001. Homemister et al. [HW05] described the QBDD proposed by Ablayev as a node having four outgoing edges, each marked with the amplitude for choosing the particular edge.

3.1.3. The QuIDDPro Tool

Viamontes et al. [VRM+03] suggested the *Quantum Information Decision Diagram* (QuIDD) as an application of the *Algebraic Decision Diagram* (ADD) to quantum computing [BFGH+93]. QuIDD matrices and QuIDD vectors allow for matrix-vector computation to be performed efficiently within the QuIDD structure. Similar to the general ROBDD, a new QuIDD may be constructed recursively using the APPLY operation on other QuIDDs. The QuIDDPro package is implemented using the well-known CUDD package for manipulation of BDDs developed at the University of Colorado [Some95].

3.1.4. The Binary Control Signal Constraint

A common crucial issue for quantum circuit simulation is whether we should constrain the control lines to take only the "binary" values $|0\rangle$ and $|1\rangle$ or to allow the

control lines the freedom of using superposition values. In their paper on quantum synthesis using the *quantum decision diagram* (QDD) structure, Abdollahi and Pedram dubbed this issue as the *binary control signal constraint* [AP05]. They pointed out that it is the practice of many researchers to adopt this constraint in quantum logic synthesis. Further, they stipulated that relaxing this constraint will not improve the optimality of the synthesis result.

As described in the previous chapter, many quantum algorithm circuits clearly require the control inputs to have superposition values. We therefore believe that this crucial issue must be further addressed. We re-visit this issue in Section 3.3.4 to show that relaxing this constraint results in a unique phenomenon of skipped variables in our QMDD tools.

3.2. The QMDD Data Structure

Miller and Thornton proposed the QMDD structure to simulate and specify reversible logic circuits in a compact form [MT06, MTG06]. As discussed in section 2.2.3, a reversible circuit with n variables (that is, a circuit with n inputs and n outputs) requires a transformation matrix of dimension $r^n \times r^n$, where r is the radix. Such transformation matrices quickly explode in size [FMY97]. However, this transformation matrix exhibits a great degree of regularity that motivated the idea for the development of the QMDD data structure.

3.2.1. Properties of the QMDD

A transformation matrix M of dimension $r^n \times r^n$ can be partitioned as:

$$M = \begin{bmatrix} M_0 & M_1 & \cdots & M_{r-1} \\ M_r & M_{r+1} & \cdots & M_{2r-1} \\ \vdots & \vdots & \ddots & \vdots \\ M_{r^2-r} & M_{r^2-r+1} & \cdots & M_{r^2-1} \end{bmatrix} \tag{3.1}$$

where each M_i element is a matrix of dimension $r^{n-1} \times r^{n-1}$. A QMDD applies this partitioning analogous to how a reduced ordered binary decision diagram (*ROBDD*) [Brya86] recursively applies Shannon decompositions. In a similar manner to ROBDDs, a QMDD adheres to a fixed variable ordering and common substructures (submatrices) are shared. A QMDD has a single terminal vertex with value 1, and each edge in the QMDD, including the edge pointing to the start vertex, has an associated complex-valued multiplicative weight. For the binary case with radix $r = 2$, a representation matrix M of dimension $2^n \times 2^n$ can be readily decomposed into four matrices $M_0,..,M_3$ so that:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_0 & \mathbf{M}_1 \\ \mathbf{M}_2 & \mathbf{M}_3 \end{bmatrix} \tag{3.2}$$

Definition 3.1: A *quantum multiple-valued decision diagram* (QMDD) [MT06] is a directed acyclic graph with the following properties:

• There is a single *terminal* vertex (or node) annotated with value 1. The terminal vertex has no outgoing edges.

• There is a number of non-terminal vertices (or nodes) each labeled by an r^2-valued selection variable. Each non-terminal vertex has a plurality of r^2 outgoing edges designated $e_0, e_1, ..., e_{r^2-1}$.

• One vertex is the *start* vertex and has a single incoming edge that itself has no source vertex.

• Every edge in the QMDD, including the one leading to the start vertex, has an associated complex-valued weight. An edge with weight of 0 must point to the terminal vertex. This is required to ensure uniqueness of the representation of each distinct matrix.

• The selection variables are *ordered* (assume with no loss of generality the ordering $x_0, x_1, ..., x_{n-1}$) and the QMDD satisfies the following two rules:

– Each selection variable appears at most once on each path from the start vertex to the terminal vertex.

– An edge from a non-terminal vertex labeled x_i points to a nonterminal vertex labeled x_j , $j < i$ or to the terminal vertex. Hence x_0 is closest to the terminal and x_{n-1} labels the start vertex.

• No non-terminal vertex is redundant, i.e. no non-terminal vertex has its r^2 outgoing edges all with the same weight and pointing to a common vertex.

• Each non-terminal vertex has outgoing edges with normalized complex weight values. Such a vertex is referred to as being normalized.

• Non-terminal vertices are unique so that no two non-terminal vertices labeled by the same x_i can have the same set of outgoing edges with the same destinations and weights.

Since each row and column of a unitary matrix must be orthonormal, we note that the magnitude of each element of such a matrix must be equal to or smaller than 1. This allows the following vertex normalization rule for QMDD to be formulated.

Definition 3.2: A QMDD vertex is normalized if its outgoing edges are such that the largest weight on any edge exiting a vertex is 1.

Since there is no natural ordering for complex numbers, we determine that the complex number $r_a e^{i\theta_a}$ to be greater than $r_b e^{i\theta_b}$ if $r_a > r_b$ or in the case of $r_a = r_b$, if $\theta_a < \theta_b$.

3.2.2. QMDD Representation is Canonic

Like Bryant's ROBDD [Brya86], the QMDD is useful due to its ***canonicity***.

Theorem 3.1: An r^n x r^n complex valued matrix M has a unique (up to variable reordering or relabeling) QMDD representation [MT06].

Proof: The proof is by induction on n.

$n = 0$**:** In this case, M is a single element. The QMDD representation consists of the terminal vertex which is also the start vertex and no nonterminal vertices. The weight on the edge leading to the terminal (start) vertex is the value in M. This is clearly a unique representation.

$n > 0$**:** Assume the result holds for all $r^{n-1} \times r^{n-1}$ matrices. Consider the r^2-partitioning of a matrix M of dimension $r^n \times r^n$. Since each sub-matrix M_i has

dimension $r^{n-1} \times r^{n-1}$, by the inductive hypothesis, the QMDD for each M_i is unique. Let *sv*, which is labeled x_{n-1}, denote the start vertex for the QMDD representing M. Initially, equate each outgoing edge e_k, $0 \leq k \leq r^n - 1$ from *sv* to the edge pointing to the start vertex in the QMDD representation of M_i, and give the edge leading to *sv* weight 1.

Normalizing *sv* as defined above, begins by finding the smallest *k* such that the weight w_k on edge e_k from *sv* is nonzero. If $wk = 1$, *sv* is already normalized. If $w_k \neq 0$, divide all the nonzero weights on outgoing edges from *sv* by w_k and set the weight on the edge leading to *sv* to w_k.

Since the QMDD for each M_i is unique up to variable reordering or relabeling and the normalization process ensures *sv* and its associated edge weights are unique, the QMDD for M is unique up to variable reordering or relabeling [MFT07b]. □

3.2.3. A QMDD Example

To familiarize the reader with the QMDD representation and operation, we provide a detailed example of the construction of a QMDD for a Controlled-V gate directly from its transformation matrix in Fig. 3.1. The general Controlled-U operation and the V gate are detailed in section 2.3.

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} \\ 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} \end{array}\right]$$

Figure 3.1. Transformation Matrix of the Controlled-V Gate

The construction of the Controlled-V QMDD representation (in radix 2) is detailed in Fig. 3.2. In step 1 we construct the start node for the first decision variable X_2 which decomposes the original 8×8 transformation matrix into four 4×4 matrices. For illustration purposes, we depict the resulting decomposed matrices near each QMDD node throughout Fig. 3.2. Note that there is an incoming edge with a weight of one pointing to the start node. We initially set the weights along the four edges from the start node to be one. Redundant nodes are nodes having all their edges pointing to the same node with the same weight on each edge. Node N3 and N4 illustrate the efficiency of the QMDD presentation as these redundant nodes are immediately eliminated in step 2 by connecting their incoming edge directly to the terminal node with weight 0. As we encounter a lot of all-zero matrices in the process, we use a short stub with value 0 to indicate an edge with zero weight to the terminal node.

In step 2 we further decompose the matrices of step 1 by applying the second decision variable X_2. The 4×4 matrices associated with nodes N2 and N5 are further decomposed to the eight 2×2 matrices along nodes N6 to N13. We can see that nodes N7 and N8 represent the zero matrix and they are eliminated in step 3.

A QMDD requires all the non-terminal nodes to be unique so that nodes having all their edges pointing to the same nodes with the same relative weights must be merged into a single node. In the figures, redundant nodes can be identified if they represent the same decomposed matrices. Redundant QMDD nodes (N10, N13) and (N11, N12) are merged in step 3.

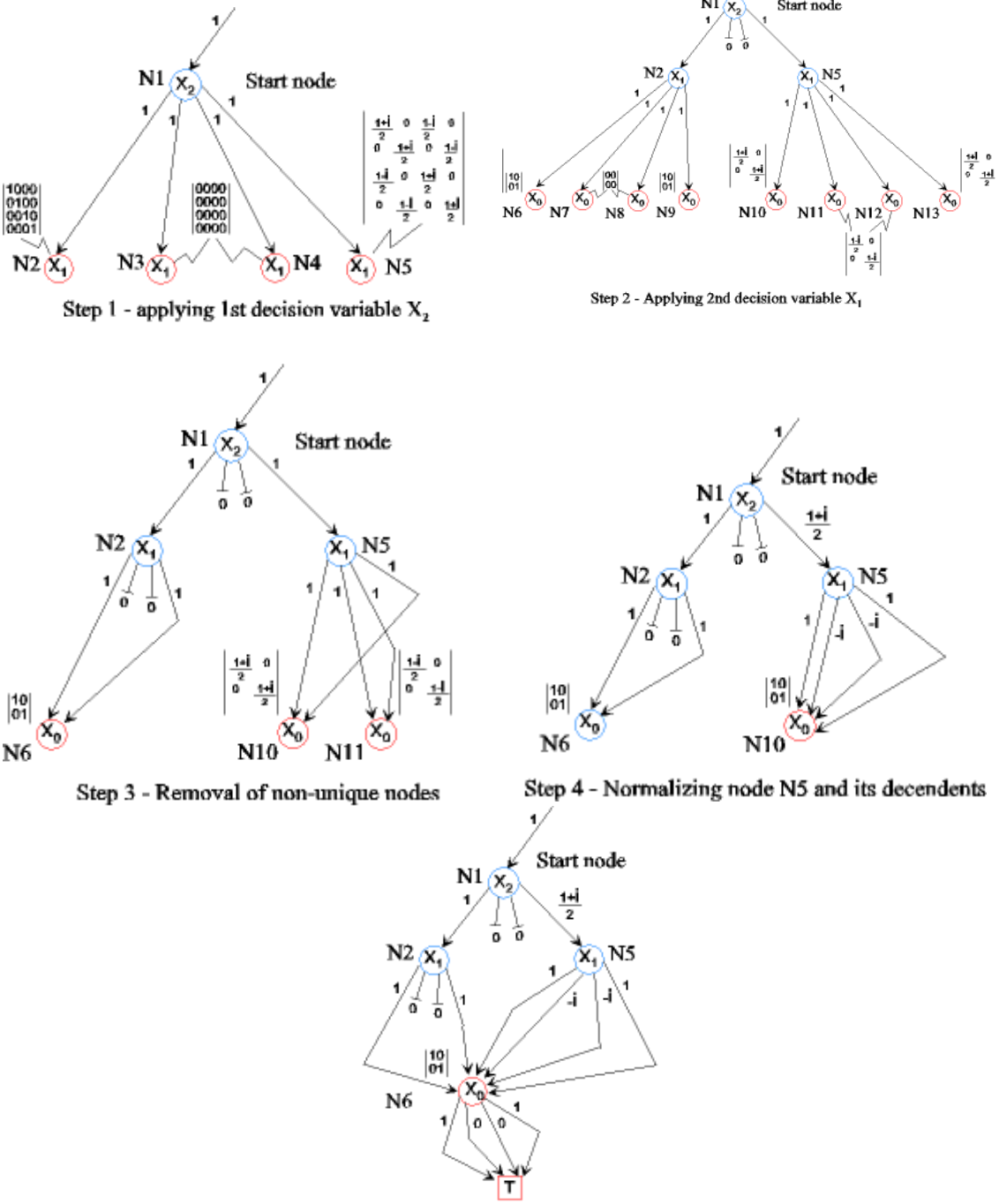


Figure 3.2. Detailed QMDD Construction for the Controlled-V Gate

Each non-terminal QMDD node is normalized to ensure canonicity as illustrated in step 4. Normalization is accomplished by assigning a weight of one to the first non-zero edge of the node. There must be such an edge since if all the edges of the nodes are annotated with a zero-valued weight, the node is redundant, and all the incoming edges that point to it should point to the terminal node with zero weight.

To make the weight of the first non-zero weighted edge of N5 equal to one, we divide all the other edges with the current weight of $(i+1)/2$ and multiply the incoming edge with this weight. The weights for the edges from node N5 now become $\{1, -i, -i, 1\}$. Note that normalization is preserved at the start node, as its first edge pointing to node N2 remains with a weight of one. Since nodes N10 and N11 represent the same decomposed 2×2 identity matrix, redundant node N11 is merged into node N10.

In the final step 5, we merge the non-unique node N10 into node N6. Node N6 is the third decision variable X_3 and it is decomposed into primitive 1×1 matrices that directly connect to the terminal node T.

The final QMDD must uniquely resolve to the original transformation matrix. This is verified by following the path from the start node to the terminal node as determined by the variable assignments. The normalized weights along each path are multiplied to achieve the corresponding values in the transformation matrix.

3.2.4. QMDD Construction and Matrix Operations

The QMDD representing the transformation matrix of a circuit is constructed by reading a quantum gate cascade description of the circuit from a netlist file. As each gate is parsed, it is converted to a QMDD representation of that particular gate. During the process of parsing the circuit file, these QMDD representations of the gates are multiplied together to form the complete QMDD of the circuit. Because QMDDs are directed graphs, multiplication is achieved through a graph traversal algorithm.

QMDD algorithms have been developed to perform the following operations on two matrices A and B, where both A and B are QMDDs and accessed by the edges to their start nodes:

- Matrix addition A+B.
- Matrix multiplication A+B.
- Tensor products $\mathbf{A} \otimes \mathbf{B}$.

For details, please see [MT06]. Using these operations, the QMDD software can efficiently simulate the gate transformation in a one pass process without backtracking or recursion.

3.3. QMDD Sift-Like Minimization and Manipulation

In this section we describe an important contribution of this research that relates to QMDD sift-like minimization [MFT07, MFT07b]. As shown in Section 3.1.1,

classical BDDs use dynamic variable reordering to minimize the BDD size. The QMDD seems to benefit equally from the implementation of dynamic reordering.

3.3.1. Dynamic Variables Re-ordering Using Sifting

We first examine the dynamic variable reordering technique commonly referred to as sifting in classical BDDs, where local variable swap operations are exploited allowing for a technique to be implemented that iteratively works on a small local subset of BDD nodes [FMK91, ISY91, YMSS06]. The sifting method was first proposed by Rudell where a single variable is sifted (moved) within a local group until minimal size is achieved for this particular local group of nodes [Rude93]. Panda and Somenzi extended Rudell's work to group sifting, but this approach (and many other heuristics) can just find a local minima [PS95]. The problem of finding the best variable ordering has been proven to be NP-Hard and just improving a given variable order is NP-Complete, with currently known optimal algorithms requiring exponential time complexities of $O(n^2 3^n)$ [AGK01].

The key observation used in Rudell's sifting method is that the swapping of two adjacent variables in a given ordering can be accomplished with a simple exchange of two pointer values and avoids the need to traverse the entire BDD. Fig. 3.3 illustrates this local complexity in the trivial case when we swap node i and i+1, with node i+1 following after node i in the original variable order. *Since an advantage of ROBDDs is the elimination of the need to keep back pointers*, node i+1 can be simply placed in the new order before node i (that is, performing the swap) without any change.

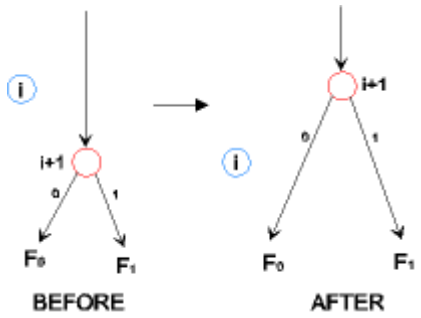


Figure 3.3. Local Complexity in a Trivial BDD Swap

The more general case is shown in Fig. 3.4 where the replacement of nodes i and i+1 causes a swap of the cofactors F_{10} and F_{01}, as shown in red. Using the common if-then-else tuple notation, the local swap operation is represented by replacing the original function $f=(x_i, F_1, F_0)$ pointing to node i with a new function g pointing to node i+1 so that:

$$g = (x_{i+1},(x_i,F_{11},F_{01}),\ (x_i,F_{10},F_{00})) \tag{3.3}$$

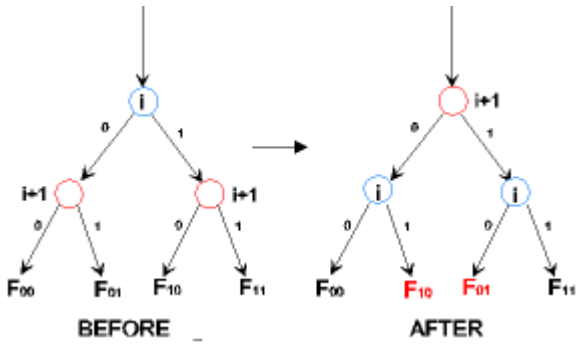


Figure 3.4. The General Case of Local Swap Affects the Cofactors

3.3.2. QMDDs without Skipped Variables

Definition 3.3: A QMDD variable i is *skipped* if a non-terminal vertex of decision variable i+1 has a non-zero weighted edge that points to a non-terminal vertex of decision variable i-1 *skipping the intermediate variable i in the overall ordering*.

A very efficient local variable swap in a QMDD can be implemented if there are no skipped variables. Local swap of skipped variables is performed in accordance with the procedure shown in [MD03]. In this section we prove that QMDD representing classical reversible circuits and binary quantum controlled-NOT gates do not have skipped variables. In the following section, we show that quantum cascades may have skipped variables.

We begin with Lemma 3.1 that outlines the requirement for a skipped variable in a QMDD structure.

Lemma 3.1: In order for an intermediate vertex representing decision variable i to be skipped in a QMDD, at least one of the $r \times r$ decomposed sub-matrices for the decision variable i+1 matrix must be further decomposed by r^2 identical sub-matrices.

Proof: The $r^i \times r^i$ sub-matrix of decision variable i+1 with r^2 identical sub-matrix decompositions represents a vertex that has all outgoing edges pointing to identical subtrees. By Definition 3.1, a non-terminal vertex is redundant if all r^2 edges point to the same vertex with the same weight. It is easy to see that the condition in Lemma 3.1 causes the intermediate vertex i to be redundant, since the identical matrices pointed to by all edges are represented by the same vertex due to the uniqueness requirement of all the QMDD vertices. This same vertex represents a matrix of size

r^{i-1} x r^{i-1}, which represents decision variable i+1. We thus show that the condition in Lemma 3.1 causes a vertex of decision variable i+1 to skip directly to decision variable i-1. □

Fig. 3.5 illustrates the condition of Lemma 3.1 for the construction of a non-zero weighted edge that skips all the nodes representing variable X_i in a simplified reversible circuit. Matrix elements a, b, c, d, e, and f can have complex values. Notice that the higher index variables are closer to the start vertex while variable 0 is closer to the terminal node.

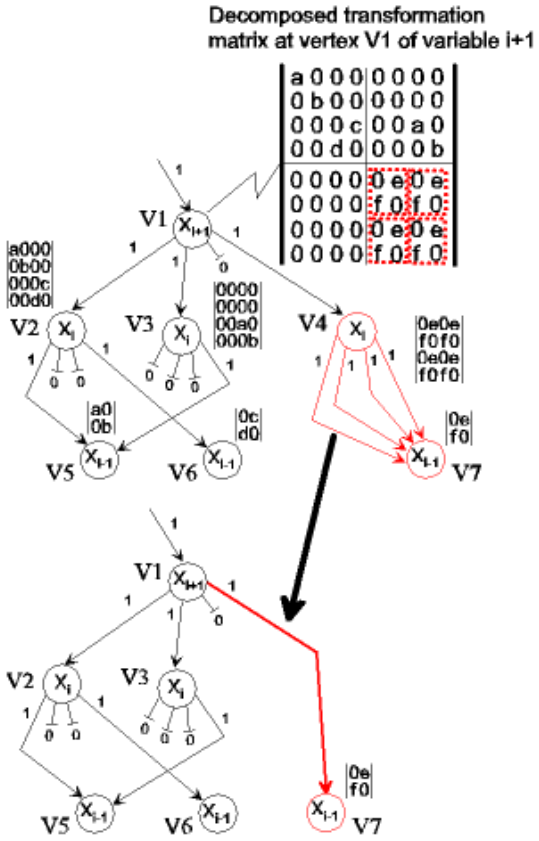


Figure 3.5. The Condition for a QMDD Skipped Variable

We now show that a classical logic reversible circuit does not have skipped variables.

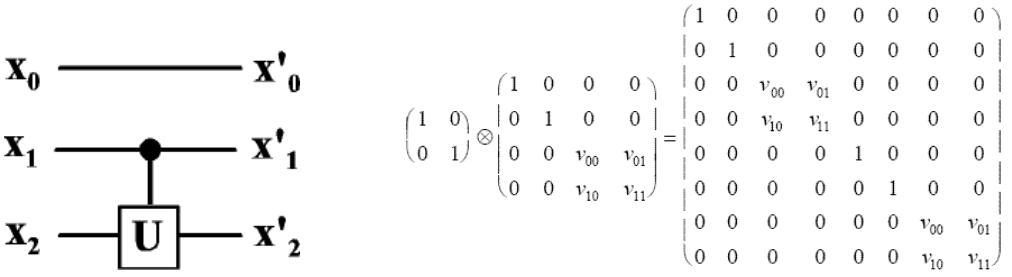
Theorem 3.2: A QMDD for a matrix representing a binary or multiple-valued reversible circuit has no edges that skip variables except for edges that point to the terminal vertex and have weight 0.

Proof: It is clear from the definition of a QMDD, that an edge annotated with a nonzero weight that skips a variable means the corresponding matrix has a sub-matrix of adjacent non-zero entries of dimension $r^k \times r^k$ for some $k > 0$. However, the matrix representing a binary or multiple-valued reversible function is a permutation matrix since reversible functions are bijections and thus have only '0' and '1' entries with a single '1' in each row and column. It follows that the QMDD for such a circuit can have no edges with a nonzero weight that skips variables. □

Theorem 3.3: A QMDD for a matrix representing a two-level unitary matrix or controlled unitary operation has no edges that skip variables except for edges that point to the terminal vertex and have weight of zero. Furthermore, even if a controlled unitary operation of m-variables operates on an n-variable circuit, $m<n$, the transformation matrix that describes this gate does not have skipped variables.

Proof: It is easy to see by reviewing expressions (2.36) and (2.37) that a controlled operation gate or a two-level unitary matrix can have only one '1' or a couple of the unitary complex elements V_{xx} in each row or column. Therefore, the condition of Lemma 3.1 for the existence of skipped variables cannot be met.

When an *m*-variable controlled-U gate is part of an *n*-variable circuit, $n>m$, it must leave *n-m* lines of the circuit unconnected. According to the 4th postulate of quantum mechanics, the unitary transformation matrix for this gate in the context of the *n*-variable circuit must be represented by the tensor (Kronecker) product of the gate matrix with identity matrices [MM05]. The order and size depends on the exact circuit architecture and an example is shown in Fig 3.6.



$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & v_{00} & v_{01} \\ 0 & 0 & v_{10} & v_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & v_{00} & v_{01} & 0 & 0 & 0 & 0 \\ 0 & 0 & v_{10} & v_{11} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & v_{00} & v_{01} \\ 0 & 0 & 0 & 0 & 0 & 0 & v_{10} & v_{11} \end{pmatrix}$$

Figure 3.6. Transformation Matrix for a 2-variable Gate in a 3-variable Circuit

The property of the tensor product ensures that each row and each column can have only one '1' or two of the unitary complex elements V_{xx}. Again, the condition of Lemma 3.1 cannot be met and the circuit does not have skipped variables. □

3.3.3. Skipped QMDD Variables Appear in QC Cascades

In this section we consider the possibility of skipped variables in an arbitrary QC cascade. Since any unitary matrix can be decomposed into a cascade of gates in view of Theorem 2.2, we need to consider skipped variables in a general unitary matrix. When keeping the *binary control signal constraint* (Section 3.1.4), we were so far

unable to find quantum circuits that will cause skipped variables in QMDD representation (see the Conjecture in [MFT07b]).

However, we do find skipped QMDD variables when we *relax* this constraint. As an example of the skipped variable phenomena, quantum circuits that have been proposed for quantum error correction codes (Chapter 5) or the quantum search algorithms (Section 2.3.1) exhibit this property. We found that a very important unitary matrix that is used in the Grover sorting algorithm for *inversion about the average* produces skipped variables when represented as a QMDD.

Definition 3.4: We will refer to an $N \times N$ matrix D with all diagonal elements equal to $\frac{2}{N} - 1$ and all the remaining elements equal to $\frac{2}{N}$ as $D_I(N)$ matrices.

Equation (3.3) illustrates the structure of $D_I(N)$.

$$D_I(N) = \begin{pmatrix} \frac{2}{N}-1 & \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N}-1 & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} & \frac{2}{N}-1 & \dots & \frac{2}{N} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{2}{N} & \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N}-1 \end{pmatrix} \tag{3.4}$$

Lemma 3.2: If N is an integer such that $N>1$, then $D_I(N)$ matrix is unitary. If m is an integer such that $m>1$, then a QMDD representing $D_I(2^m)$ must have skipped variables. Further, the QMDD must have at least one vertex with a non-zero weight that skips m-1 variables.

Proof: It is easy to show that $D_I(2^m)\, D_I(2^m)^* = I$, and therefore, $D_I(2^m)$ is unitary and represents a valid quantum state transformation. When $n = 2^m$, where m is an

integer and $m>1$, $D_I(2^m)$ can be represented by a binary (radix $r = 2$) QMDD. To prove that $D_I(2^m)$ has a skipped variable, consider the decomposition of the matrix into 4 sub-matrices at each decision variable during the QMDD build up. For $m=2$, $D_I(4)$ is the 4×4 unitary matrix

$$D_I(4) = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

The first decision variable x_1 decomposes $D_I(4)$ into four 2×2 sub-matrices according to (3.1). Clearly these sub-matrices are not equal. The next and final decision variable x_0 further decomposes each of the four 2×2 sub-matrices into four 1×1 sub-matrices. With this example, it is easy to show that the condition of Lemma 3.1 exists within the second and third sub-matrices that are

$$S_1 = S_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

It is easy to show by induction that matrix $D_I(2^m)$ where $m>1$ results in a QMDD containing skipped variables that actually skip over m-1 variables. This proves the case for $m=2$ that there is a non-zero weighted edge that skips m-1=1 variables.

For the m+1 case, we note that increasing m by 1 adds one decision variable and quadruples the size of the transformation matrix from that of m. Since all elements except the diagonal elements are the same, we have a skipped variable which skips one more variable than the case for m. □

Fig. 3.8 illustrates the skipped variables that arise in the $D_I(8)$ unitary matrix. For conciseness, we present the transformation matrices with $a = \frac{2}{N} = \frac{1}{4}$ and with the diagonal elements $b = \frac{2}{N} - 1 = \frac{1}{4} - 1$. The left part of the drawing shows the creation of the QMDD prior to the elimination of the redundant vertices. The resultant QMDD is depicted on the right side of the drawing. The red edges skip two decision variables while the blue edges skip one decision variable.

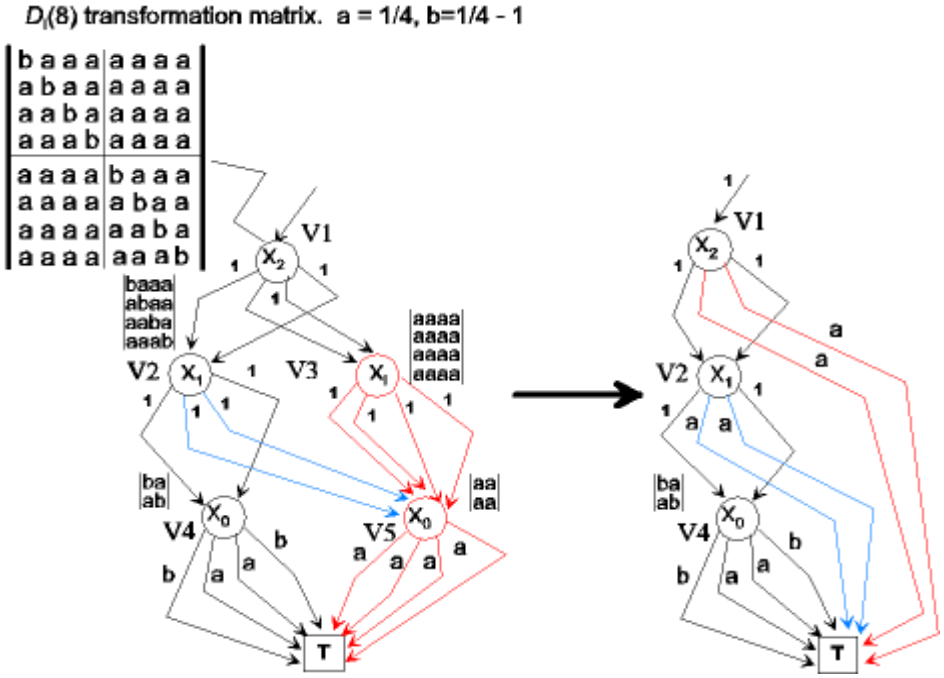


Figure 3.7. QMDD with One and Two Skipped Variables

Corollary 3.1: An $N \times N$ matrix with all elements equal to $\frac{2}{N}$ except for the cross diagonal elements which are equal to $\frac{2}{N} - 1$, causes skipped variables to occur in

the corresponding QMDD representation similar to the $D_I(N)$ matrix. We call such a matrix a *reversed* $D_I(N)$ matrix.

Proof: The proof is very similar to Lemma 3.2 in view of the symmetry of both matrices. □

The simple circuit HCH, shown in Fig. 3.8 is used in the stabilizer circuit for QC error correction codes. The HCH circuit is represented by the transformation matrix of (3.4) that is a *reversed* $D_I(4)$. Therefore, by Corollary 3.1, the QMDD representing this circuit must have at least one non-zero weight edge that skips one variable (shown in blue).

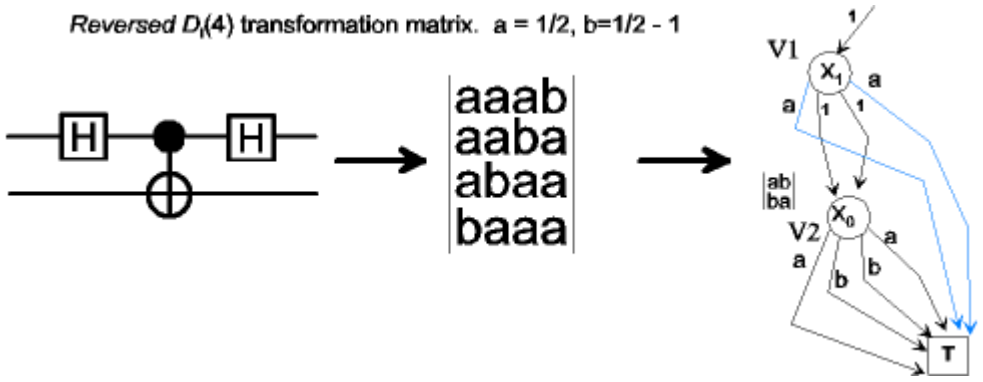


Figure 3.8. The HCH Circuit

We decompose the HCH circuit using the Beck-Zeilinger-Bernstein-Bertani Theorem of Section 2.3.6. We follow steps (2.47)-(2.53) so that we can compare this matrix decomposition to the example of Section 2.3.6 that did not create skipped variables.

$$\mathbf{U} = \mathbf{HCH} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \tag{3.5}$$

We first calculate the two-level unitary V_1 that will produce '0' in the second element of the first column (a_{21}) to obtain

$$\mathbf{V}_1 = \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ 0.7071 & -0.7071 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \quad \mathbf{V}_1\mathbf{U} = \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \tag{3.6}$$

The two-level unitary V_2 should produce '0' in a_{31} to obtain

$$\mathbf{V}_2 = \begin{bmatrix} 0.8165 & 0 & 0.5774 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5774 & 0 & -.8165 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \quad \mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 0.866 & 0.2887 & 0.2887 & 0.2887 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0.8165 & -0.4082 & -0.4082 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \tag{3.7}$$

The two-level unitary V_3 should produce '0' in a_{41} to obtain

$$\mathbf{V}_3 = \begin{bmatrix} 0.866 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.5 & 0 & 0 & -0.866 \end{bmatrix} \rightarrow \quad \mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0.8165 & -0.4082 & -0.4082 \\ 0 & -0.5774 & -0.5774 & -0.5774 \end{bmatrix} \tag{3.8}$$

The two-level unitary V_4 should produce '0' in a_{32} to obtain

$$\mathbf{V}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \quad \mathbf{V}_4\mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.8165 & -0.4082 & -0.4082 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & -0.5774 & -0.5774 & -0.5774 \end{bmatrix} \tag{3.9}$$

The two-level unitary V_5 should produce '0' in a_{42} to obtain

$$\mathbf{V}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.8165 & 0 & -0.5774 \\ 0 & 0 & 1 & 0 \\ 0 & -0.5774 & 0 & -0.8165 \end{bmatrix} \rightarrow \quad \mathbf{V}_5\mathbf{V}_4\mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0 & 0.7071 & 0.7071 \end{bmatrix} \tag{3.10}$$

The decomposition in the right side of (2.51) is already in the form of a final two-level matrix. Therefore, the two-level unitary matrix V_6 is essentially a simple inverse of the last decomposition to achieve the identity matrix in the decomposition.

$$\mathbf{V}_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \\ 0 & 0 & -0.7071 & 0.7071 \end{bmatrix} \rightarrow \mathbf{V}_6\mathbf{V}_5\mathbf{V}_4\mathbf{V}_3\mathbf{V}_2\mathbf{V}_1\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{I} \tag{3.11}$$

The required decomposition of **U** is therefore

$$\mathbf{U} = \mathbf{V}_1^+\mathbf{V}_2^+\mathbf{V}_3^+\mathbf{V}_4^+\mathbf{V}_5^+\mathbf{V}_6^+ \tag{3.12}$$

Consider the transition between (3.6) to (3.7) to the comparable transition in Chapter 2 between (2.48) and (2.49). In this step the algorithm attempts to convert matrix element a_{41} to '0'. Notice that in the HCH case, this step also sets the diagonal element a_{22} to '0'. Our intuition is summarized in the following conjectures [FTM08d]:

Conjecture 3.1: A QMDD representation of a unitary matrix that has any of its diagonal elements changing to '0' during the BZBB decomposition steps must exhibit skipped variables.

Conjecture 3.2: A QMDD representation of a unitary matrix has skipped variable only if we relax the binary control signal constraint.

3.3.4. Local Variable Swap for QMDD with no Skipped Variables

The local variable swap is the basic operation in dynamic variable re-ordering algorithms for decision diagrams. An extremely efficient algorithm results for QMDD with no skipped variables.

Algorithm 3.1: QMDD Local Adjacent Variable Pair Swap

Input: QMDD ***f*** representing a reversible quantum circuit with k variables and n nodes. i, a variable to swap, $1<i<k$. t is the number of all nodes representing variable i in QMDD ***f***.

Output: QMDD ***g*** with ***n'*** nodes, which is QMDD ***f*** with swapped variables i and i-1.

Initialization: Reset queue *q1* and an array of t lists L_x.

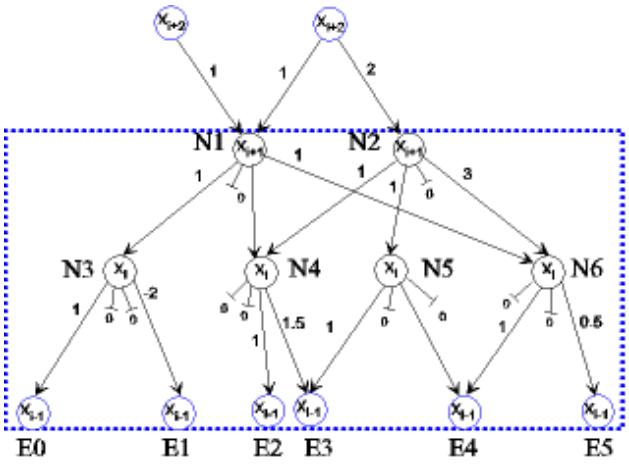
1. Enter pointers to all the t nodes N_x ,$0<x<t$, representing variable i into the queue *q1*.
2. While queue *q1* is not empty.
3. Get a node pointer p to N_x from *q1*.
4. Do a DFS search from the node N_x pointed by p and create a list L_x of all the paths and local weight from node N_x to all the reachable nodes representing variable i-2. Each such path includes exactly two edges m and l so that $0<=m,l<4$, e_m is the edge from the i node N_x to an i-1 node, and e_l is the edge from the i-1 node to the i-2 node. Each path is marked as Nx_{ml}. It is stored in

pair with the local weight Nx_w for each path, which is computed as e_m x e_l. Each pair must be associated with the node *Ej* representing variable *i*-2 reachable by the path.

5. Delete all nodes currently representing variable *i*-1.
6. For (*x*=0*; x<t; x*++):
    [1] Change N_x variable to *i*-1.
    [2] Build a new set of nodes of variable *i* by reversing the paths in L_x from Nx_{ml} to Nx_{lm} and connecting the second edge to the appropriate E_j node of variable *i*-2. Follow QMDD construction rules to eliminate redundant and non-unique new nodes. Use normalization to assign new edges weight to equal the original path weight.
7. Return the post-swap QMDD as ***g***, and report its total number of nodes ***n'***. □

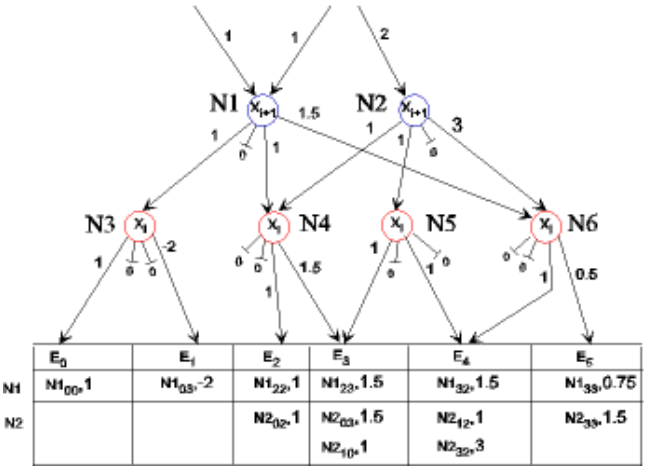
This algorithm may be best explained in conjunction with a detailed graphic example. A portion of our example QMDD showing all the nodes for variables X_{i+2}, X_{i+1}, X_i, and X_{i-1} is shown in Fig. 3.9a. Our goal is to locally swap adjacent variables X_{i+1} and X_i, which are bounded by the blue bracket. Constant local complexity is achieved if the swap can be done completely within the bracket. This is where the lack of non-zero weighted edges with skip variables assures us that we do not need to change the incoming edges from the X_{i+2} top nodes. Our local swap method provides new *locally determined edges* to connect to the six nodes of variable X_{i-1} which are marked *E0* to *E5*.

(a) Before local swap of variables X_{i+1} and X_i



(b) Mapping of all local paths and weights

Edges from variables X_{i+2}



	E_0	E_1	E_2	E_3	E_4	E_5
N1	$N1_{00}$,1	$N1_{03}$,-2	$N1_{22}$,1	$N1_{23}$,1.5	$N1_{32}$,1.5	$N1_{33}$,0.75
N2			$N2_{02}$,1	$N2_{03}$,1.5 $N2_{10}$,1	$N2_{12}$,1 $N2_{32}$,3	$N2_{33}$,1.5

(c) Creating the local swap and weights normalization

Edges from variables X_{i+2}

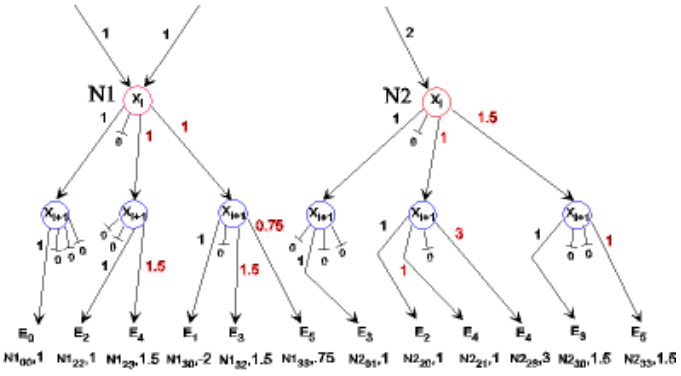


Figure 3.9. QMDD Local Variable Swap Example

(d) After local swap of variables X_{i+1} and X_i

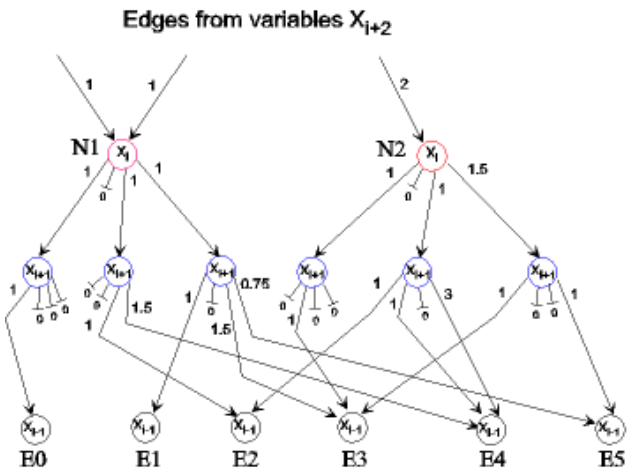


Figure 3.9. (Cont'd) QMDD Local Variable Swap Example

3.3.5. The QMDD Sifting Procedure

The QMDD sifting procedure repeatedly uses the local variable swap algorithm in a similar way to the sifting procedure for classical BDDs [Rude93].

Algorithm 3.2: QMDD Sifting Procedure

i) Select a variable α that labels the most vertices in the QMDD. In the event of a tie, choose the variable closest to the terminal vertex.

ii) Sift α to the bottom (closest to the terminal vertex) of the QMDD by a sequence of adjacent variable interchanges.

iii) Sift α to the top of the QMDD by a sequence of adjacent variable interchanges.

iv) During steps (ii) and (iii) a record is kept of the position of α that yields the smallest vertex count in the QMDD, so now sift α back down to that position.

v) Repeat steps (i) to (iv) until each variable has been sifted into its *best* position noting that once a variable is selected for sifting, it is not selected a second time.

3.3.6. QMDD Minimization Results with Sifting

Table 3.1 shows the experimental minimization results with binary reversible benchmark circuits [Masl05]. The results show the sifting method can be quite effective, but at times, it can be computationally expensive with little benefit. More results regarding our experimentation with ternary circuits can be found in the work reported in [MFT07].

Table 3.1. Experimental Results – Binary Circuits

name	type	lines	gates	vertices before sifting	time to build QMDD (ms)	vertices after sifting	time to sift QMDD (ms)	vertex count reduction by sifting	max. vertices during sifting
5mod5	nct	6	17	28	0	16	8	43.9%	28
6symd2	nct	10	20	247	7	170	42	31.2%	478
9symd2	nct	12	28	229	7	184	50	19.7%	558
c2	nct	35	116	150	30	136	289	9.3%	504
c2	qc	35	305	150	220	136	241	9.3%	504
c3-17	nct	3	6	10	0	10	4	0.0%	11
c410184	nct	14	46	39	0	33	43	15.4%	86
c410184	qc	14	74	39	0	33	52	15.4%	86
cyc17-3	nct	20	48	236	10	42	131	82.2%	418
ham3	nct	3	5	10	0	10	0	0.0%	10
ham15	nct	15	132	4522	140	2638	488	42.7%	13878
hwb4	nct	4	11	22	0	20	4	9.1%	22
hwb4	qc	4	21	22	0	20	8	9.1%	22
hwb7	nct	7	289	179	30	155	16	13.4%	181
hwb8	nct	8	614	343	120	280	28	18.4%	351
hwb9	nct	9	1541	683	640	520	40	23.9%	690
hwb10	nct	10	3595	1331	2920	960	59	27.9%	1347
hwb11	nct	11	9314	2639	14290	1730	130	34.4%	2685
hwb12	nct	12	18393	5167	53350	3185	265	38.4%	5254
rd84d1	nct	15	28	3588	27	396	117	89.0%	4415

In Table 3.2 we compare our minimized results to those obtained with the QuIDDPro tool [VRM+03]. Since the QMDD has four edges from each vertex and thus more efficient connectivity, we see that even before sifting, the QMDD requires significantly less vertices than the QuIDDPro tool.

Table 3.2. Size Comparison between QMDD and QuIDDPro

name	lines	gates	vertices before sifting	vertices after sifting	time to sift QMDD (ms)	vertex count reduction by sifting	QuIDDPro vertices	time to build BDD (ms)
5mod5	6	17	28	16	8	43.9%	45	77
6symd2	10	20	247	170	42	31.2%	299	117
9symd2	12	28	229	184	50	19.7%	445	194
c2	35	116	150	136	289	9.3%	348	1546
c2	35	305	150	136	241	9.3%	348	10245
c3-17	3	6	10	10	4	0.0%	21	22
c410184	14	46	39	33	43	15.4%	86	274
c410184	14	74	39	33	52	15.4%	86	492
cyc17-3	20	48	236	42	131	82.2%	584	880
ham3	3	5	10	10	0	0.0%	21	21
ham15	15	132	4522	2638	488	42.7%	7547	2051
hwb4	4	11	22	20	4	9.1%	45	43
hwb4	4	21	22	20	8	9.1%	45	83
hwb7	7	289	179	155	16	13.4%	351	2117
hwb8	8	614	343	280	28	18.4%	684	6254
hwb9	9	1541	683	520	40	23.9%	1349	24566
hwb10	10	3595	1331	960	59	27.9%	2650	98704
hwb11	11	9314	2639	1730	130	34.4%	5223	478555
hwb12	12	18393	5167	3185	265	38.4%	10283	1722050
rd84d1	15	28	3588	396	117	89.0%	1252	255

3.4. Data Structure Metrics of Quantum Multiple-valued Decision Diagrams

In this section we consider the use of data structure metrics for the analysis and minimization of QMDD. We should recall that a QMDD has r^2 transition edges from

each vertex, where r is the radix. Since the transformation matrices representing quantum circuits are often sparse, many QMDD edges point directly to the terminal node with zero weight. This creates a complex inter-connectivity among the QMDD vertices that allows QMDD to represent relatively intricate circuits with a small number of vertices. We were motivated to examine the frequency of non-zero weight edges as a potential indicator that improves the minimization of the QMDD.

The data structure metrics proposed in this research allow us to better understand this inter-connectivity. They include the ratio of non-zero weight edges to the number of vertices for the entire QMDD as well as a distribution of the similar ratio measured for each variable. We propose a set of heuristics based on the values of these metrics to guide *dynamic variable ordering* (DVO) resulting in an improvement in minimization for many benchmark circuits.

In our previous work, we have developed a DVO sifting algorithm that utilized efficient adjacent variable interchange [MFT07b]. The sifting algorithm, adapted for QMDD from Rudell's binary ROBDD sifting approach [Rude93], achieved significant size reductions on some benchmark circuits of reversible/quantum circuits of n-variables while considering only $O(n^2)$ of the $n!$ possible variable orderings.

In this work we offer additional heuristic algorithms for QMDD minimization. The new heuristics achieve improved minimization for some benchmark circuits over our previous sifting algorithm. Since such DVO minimization efforts consume processing time, it is beneficial to be able to predict in advance whether a certain QMDD structure can be minimized [YMSS06]. We propose heuristics, based on our

new data structure metrics that provide effective prediction in this regard for most of the benchmark circuits we tested.

3.4.1. Exploring QMDD Data Structure Metrics

In this section, we demonstrate the proposed QMDD data structure metrics with the help of the simple benchmark file "C3_17.nct" shown in Fig. 3.10. Fig. 3.11 shows the QMDD for benchmark file "c3_17.nct" of Fig. 3.10.

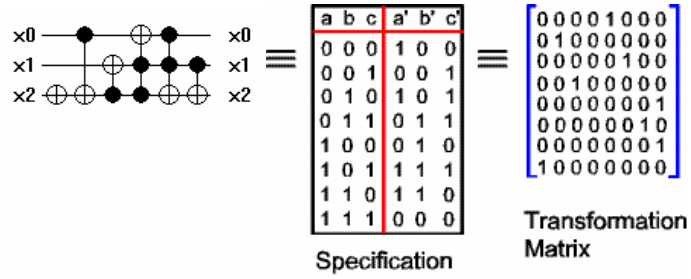


Figure 3.10. 3-variable Benchmark Circuit "c3_17"

The unique table used by the QMDD software continuously maintains and tracks *Active[i]*, the number of active vertices for each variable *i,* $0 \leq i \leq n$, where *n* is the number of variables. This is an important auxiliary QMDD data structure that we have used extensively in our previous work. We define new QMDD data structure metrics as follows:

Definition 3.5: Let α be the ratio of the total number of edges with non-zero weight to the total number of vertices in the entire QMDD. Similarly, let $\alpha[i]$ be the

ratio of the number of edges with non-zero weight that emerge from all the vertices of variable *i* to the number of vertices of the same variable. It follows that,

$$\alpha = \frac{\sum_{0}^{n-1} Active[i] \times \alpha[i]}{\sum_{0}^{n-1} Active[i]} \tag{3.13}$$

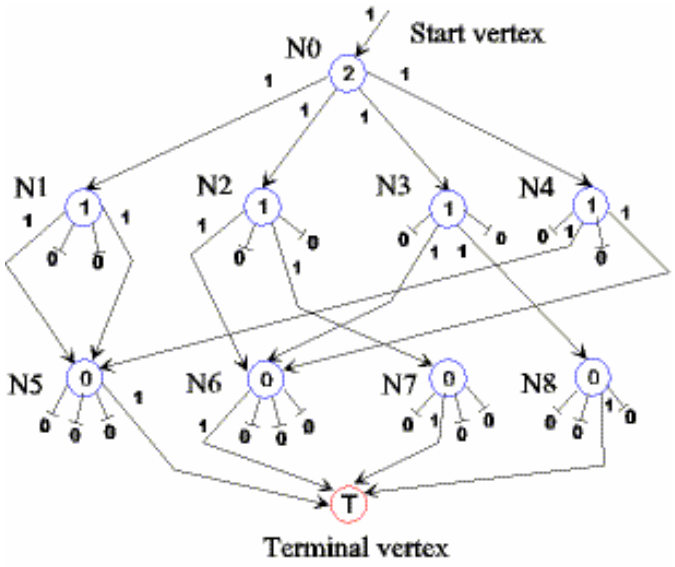


Figure 3.11. The QMDD for Benchmark Circuit "c3_17"

Since each nonterminal QMDD vertex has r^2 edges, $1 \leq \alpha \leq r^2$ and $1 \leq \alpha[i] \leq r^2$, where *r* is the QMDD radix. For example, computing $\alpha[1]$ for the QMDD of Fig. 3.11, we have a total of four vertices and eight non-zero weight edges, thus $\alpha[1] = 8/4 = 2.00$. We observe that while vertex *N1* has two edges with non-zero weight, they both connect *N1* to *N5*, essentially connecting the vertex to only one *unique* vertex. We therefore distinguish between the number of edges with non-zero weight and the number of unique *connections* that reach different vertices as follows:

Definition 3.6: Let β be the ratio of the total number of *unique connections with non-zero weight* to the total number of vertices in the entire QMDD. Similarly, let $\beta[i]$ be the ratio of the number of non-zero weight edges that emerge from all the vertices of variable i to the number of vertices of the same variable. As above, $1 \leq \alpha \leq r^2$ and $1 \leq \beta[i] \leq r^2$. It follows that

$$\beta = \frac{\sum_0^{n-1} Active[i] \times \beta[i]}{\sum_0^{n-1} Active[i]} \tag{3.14}$$

For variable 1 in Fig. 3.11, we have a total of four vertices and seven unique connections with non-zero weight, thus $\beta[1] = 7/4 = 1.75$. The relation between α and β is given by the following lemma.

Lemma 3.3: Let $\alpha, \beta, \alpha[i],$ and $\beta[i]$ be the data structure metrics of a QMDD with n variables as defined above. Hence, (i) $1 \leq \beta[i] \leq \alpha[i] \leq r^2$ for all i, 0<i<n-1, and (ii) $1 \leq \beta \leq \alpha \leq r^2$.

Proof: Each vertex representing variable i must have at least one edge with a non-zero weight, since a QMDD vertex with all edges having zero weight is redundant and would be removed. This edge creates a unique connection to a nonterminal vertex, thus $1 \leq \beta[i]$ must be true. We observe that the number of unique connections for the vertices of variable i cannot exceed the number of edges with non-zero weight by Definition 3.6. Thus $\beta[i] \leq \alpha[i]$ must be true.

Let *Active[i]* denote the number of active vertices for variable *i*, and *E* the total number of the QMDD's edges with non-zero weight and *C* the total number of unique connections. Then

$$E = \sum_0^{n-1} Active[i] \times \alpha[i] \tag{3.15}$$

and

$$C = \sum_0^{n-1} Active[i] \times \beta[i] \tag{3.16}$$

Since we have proven that $\beta[i] \le \alpha[i]$, it follows that $C \le E$ and

$$\beta = \frac{C}{\sum_0^{n-1} Active[i]} \le \frac{E}{\sum_0^{n-1} Active[i]} = \alpha. \tag{3.17}$$

Clearly, $\alpha[i] \le r^2$, with the equality occurring when all edges have non-zero weights. The proof for (ii) follows similar arguments. □

Listing the metrics for QMDD metrics in tabular form provides a histogram-like display of the structure of the QMDD which, we will for convenience, refer to as a distribution.

We demonstrate in Table 3.3 the distribution of the metrics for the QMDD in Fig. 3.11. The bottom line of the distribution provides the overall α, β and number of vertices for the entire QMDD. Note that the sum of *Active[i]* does not include the terminal node. The distribution lists the variables according to the *current variable order* of the QMDD. In Table 3.3, variable 2 labels the start node and variable 0 is the closest variable to the terminal node. As a result, variable *n*-1, the start node, always has *Active[n-1]*=1. The last row of the distribution shows the overall values for the

QMDD. For the Active[i] column, this is the sum (total of nodes), and for the $\alpha[i], \beta[i]$ columns, the last row depicts the overall α, β.

Table 3.3. Metric Values for Benchmark Circuit "3_17.nct"

variable	Active[i]	$\alpha[i]$	$\beta[i]$
2	1	4.00	4.00
1	4	2.00	1.75
0	4	1.00	1.00
Overall	9	1.78	1.67

BDD researchers have observed that the numbers of vertices for each variable (arranged by the variable order) in general exhibit a pear-shaped pattern [PS95]. We have observed that while most QMDD exhibit similar pear shape distributions of the metrics, many distributions exhibit quite a flat pattern, where most variables have roughly the same *Active[i]*. This allows us to classify QMDD metric distributions as "FLAT" or "PEAR", as demonstrated in the two examples in Table 3.4. The left distribution of "hwb12" exhibits the typical pear-like pattern commonly observed in classical BDDs. The right distribution corresponding to the benchmark circuit "cycle10_2*" is of type flat as variables 7 to 2 have similar numbers of vertices in the range [15,18]. We later show that different minimization heuristics apply based on this classification.

The $\alpha[i]$ and $\beta[i]$ distributions provide additional insight that is explored in the following Section. It should be noted that the QMDD unique table provides an efficient means to compute the Active[i] as well as the $\alpha[i]$ and $\beta[i]$ distributions incurring very little processing overhead.

Table 3.4. Histograms for "hwb12" and "cycle10_2*"

var	act[i]	$\alpha[i]$	$\beta[i]$	var	act[i]	$\alpha[i]$	$\beta[i]$
11	1	4.00	4.00	11	1	2.00	2.00
10	4	4.00	4.00	10	2	4.00	3.00
9	16	4.00	4.00	9	6	1.67	1.67
8	64	4.00	4.00	8	10	2.00	1.60
7	256	3.91	3.91	7	16	1.44	1.19
6	990	2.84	2.84	6	16	1.44	1.19
5	2258	1.37	1.37	5	16	2.00	1.56
4	1174	1.17	1.17	4	18	1.50	1.17
3	304	1.16	1.16	3	18	2.00	1.00
2	76	1.16	1.16	2	15	1.20	1.00
1	19	1.21	1.21	1	9	1.33	1.00
0	4	1.00	1.00	0	3	1.33	1.00
Overall	5167	1.76	1.76	Overall	131	1.65	1.24

3.4.2. Improving QMDD Minimization with the Data Structure Metrics

The inability of our sifting algorithm (as well as any other similar heuristic algorithm) to achieve consistent positive results with all the benchmark circuits, is of course, due to the fact that it examines only n^2 ordering possibilities out of $n!$ possible orderings. To achieve improved minimization, common binary BDD packages offer a rich choice of reordering heuristics. These heuristics can be grouped into sifting algorithms (that include also group, iterative, and symmetric sifting), random algorithms, window algorithms, and other special algorithms (including simulated annealing and genetic evolutionary algorithms) [YMSS06].

While all of these heuristics can be extended for QMDD, we concentrate here on heuristics primarily based on sifting and random approaches. Our experimentation with the group sifting approach that exploits the affinity among variables achieved limited success. One reason is that benchmark circuits with sufficiently large numbers of

variables are currently not available. Furthermore, it seems that since reversible circuits are maximally connected [Agra81, FTM08], they tend to display a strong group behavior as a whole, making attempts to identify subgroups more difficult or redundant.

Our random algorithm performs a set of random variable position changes without regard to structure size between repeated applications of the sifting algorithm. This process breaks apart the initial grouping of variables allowing the sifting algorithm to find a better ordering. A set of heuristics inferred from the data structure metrics is then used to determine when the process should stop.

Algorithm 3.3: QMDD Minimization Procedure

i) Apply the sifting algorithm for initial reordering.

ii) Use metrics-based heuristics to predict if a better variable ordering is likely. If prediction is negative – stop the procedure.

iii) Perform a random set (or a specially designed fixed set) of variable rearrangements *without regard* to structure size.

iv) Repeat steps (i) to (iii). Stop the process if this step is repeated more than *k* times.

We present a sample set of the data structure metrics heuristics that have been investigated. Like all heuristics, they obtain different levels of success as described in the following section, hence the *k* limit in step *iv*.

Sample Heuristics:

Heuristic 1: For pear-type QMDDs representing various benchmark circuits, if the sifting algorithm provides moderate or no improvement and if the pattern of the of $\alpha[i]$ and $\beta[i]$ distribution is not changed by sifting, then no significant improvement is likely to be achieved.

Heuristic 2: For flat-type QMDDs representing various benchmark circuits, no significant improvement is likely if the first half of the variables exhibit $\alpha[i] \approx 2.00$.

Heuristic 3: Appearance of $\alpha[i]$ or $\beta[i]$ equal to 2 in the first two variables (of the variable ordering) of a pear-type benchmark circuit suggests benefit for further reduction attempts, as the QMDD may become flat.

Heuristic 4: Benchmark circuits with an $\alpha[i]$ *distribution* that is not monotonically decreasing (along variable order from the start vertex) resist minimization.

Heuristic 5: If reordering of a flat-type QMDD representing a benchmark circuit changes its distribution shape to pear-type, the total number of vertices will usually increase.

3.4.3. Experimental Improvements with the Data Structure Metrics

We summarize our results for a broad set of benchmark circuits in Table 3.5. The second column indicates the histogram type (PEAR or FLAT) as described in Section 3. The last column indicates the improvement (if any) of the new algorithm over our previous sifting algorithm.

Table 3.5. Improvements in QMDD Minimization

Name	Hist type	vars	gates	Initial QMDD vertices	Previous sifting reduction	New algorithm reduction	Improve-ment
5mod5	Pear	6	17	28	42.86%	46.43%	8.33%
6symd2	Pear	10	20	247	50.20%	56.68%	12.91%
9symd2	Pear	12	28	229	19.65%	48.19%	145.24%
mod5adders	Pear	6	21	38	0.00%	0.00%	0.00%
0406142c2	Flat	35	116	150	9.33%	9.33%	0.00%
0410184	Flat	14	46	39	15.38%	15.38%	0.00%
0410184.qc	Flat	14	74	39	15.38%	15.38%	0.00%
cycle17_3	Flat	20	48	236	57.20%	82.20%	43.71%
cycle17_3r	Flat	20	48	236	82.20%	82.20%	0.00%
cycle10_2	Flat	12	19	67	40.30%	62.69%	55.56%
ham7	Pear	7	23	130	3.08%	24.62%	699.35%
ham15	Pear	15	132	4521	49.56%	72.62%	46.53%
ham15m	Pear	15	132	1774	19.17%	36.30%	89.36%
ham15r	Pear	15	132	4522	60.59%	73.33%	21.03%
hwb7	Pear	7	289	179	13.41%	13.41%	0.00%
hwb8	Pear	8	614	343	18.37%	18.37%	0.00%
hwb9	Pear	9	1541	683	23.87%	23.87%	0.00%
hwb10	Pear	10	3631	1331	27.87%	29.30%	5.13%
hwb11	Pear	11	9314	2639	34.44%	34.44%	0.00%
hwb12	Pear	12	18393	5167	38.36%	39.40%	2.71%
rd84d1	Pear	15	28	3588	92.64%	92.64%	0.00%
rd73d2	Pear	15	28	3588	14.00%	40.76%	191.14%
rd53d1	Pear	7	12	26	19.32%	19.32%	0.00%

We have seen significant improvements with the benchmark circuits “9symd2”, “cycle10_2”, “ham7”, “ham15m” and “rd73d2”. We recorded the QMDD histogram during the entire minimization process. An interesting observation for most benchmark cirucits is that the initial QMDD construction process produces a monotonically decreasing $\alpha[i]$ distribution. On the other hand, the $\beta[i]$ distribution may exhibit fluctuations, although the values are generally decreasing in view of Lemma 1. The only observed violation of the monotonic decrease of $\alpha[i]$ appears with benchmark circuit “mod5adders”. Since this circuit seems to resist minimization, we use this finding in Heuristic 4.

The “hidden weighted bit” benchmark files (“hwb7”, “hwb8”, “hwb9”, “hwb10”, “hwb11”, and “hwb12”) worked very well with Heuristic 1. In its’ reversed application, it showed particularly good improvements for “rd73d2”, “ham15r”, “ham15”.

Heuristic 2 appears to be effective, however, a large number of FLAT type benchmark circuits are needed to validate it more extensively. A similar situation occurs with Heuristic 3.

To further evaluate Heuristic 5, we subjected the benchmark circuit “cycle10_2” to several random variable reorderings, and captured distributions we marked as random0, random1 and random2. We compared the *active[i]* (number of vertices of variable *i*) of these distributions with the minimized distributions obtained by our sifting algorithm and the new algorithm described here. We show our results in Fig. 3.12, where we also include the original ordering distribution. Each series in the graph illustrates the *active[i]* for each variable. The variable number is assigned from the start node (1) to the last variable (12) as traversed along the QMDD variable order.

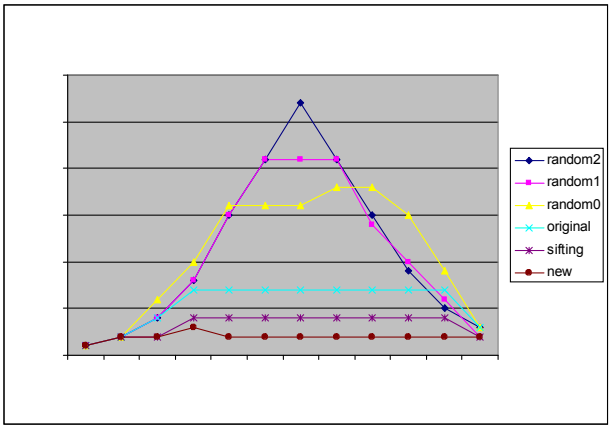


Figure 3.12. Benchmark Circuit “cycle10_2” Changes from FLAT to PEAR

The total number of vertices for random2, random1, and random0 are 132, 126, and 131, respectively. From Table 3.5, the total number for the original variable order is 67, and for the minimized sifting and new algorithm it is 40 and 25, respectively. We can see that the shape changes from FLAT to PEAR for random2 and random1. Clearly, these variable orders cause the QMDD to use a much higher number of vertices than in our minimized variable orders. Therefore, a minimized flat benchmark circuit must remain flat during further minimizations.

These preliminary minimization results show significant improvement versus application of the sifting algorithm without using the metrics, although no improvement occurred for some benchmark circuits.

3.5. Using QMDD for QC Simulation

Wallace defined quantum computer simulators as computer programs executed on a classical machine to emulate the behavior of a quantum computer [Wall01]. Quantum circuit simulation is the prediction of the output of a circuit given a specific input stimulus. In this dissertation we are concerned with the quantum circuit simulation obtained with our various QMDD tools.

We primarily look at the ability of our tools to simulate the Maslov quantum circuit benchmark circuits [Masl05]. The QMDD essentially represents the transformation matrix of the circuit that needs to be simulated. If **M** is the

transformation matrix, then the prediction of the output pattern O in relation to the input stimuli I is given by

$$O = \mathbf{M} \times I \tag{3.18}$$

We have formulated, implemented, and compared two approaches for quantum logic circuit simulation based on QMDDs. The first approach is performed by explicit multiplication using QMDD representations of the circuit and input vector. The second approach is performed through implicit multiplication where the QMDD representing the circuit undergoes a guided traversal based upon the input vector. These methods were implemented and compared in terms of memory usage and runtime as shown in Table 3.6. [GTFM07].

Table 3.6. Simulation of Implicit and Explicit QMDD Vector Multiplication

Benchmark circuit	Number of qubits	Number of gates	Number of Input vectors	Average Implicit Multiplication Runtime (seconds)	Average QMDD size for Implicit Multiplication	Average Explicit Multiplication Runtime (seconds)	Average QMDD size for Explicit Multiplication
Ham3	3	5	8	0.00005	11	0.00004	20
3_17	3	6	8	0.00005	12	0.00007	21
Rd32	4	4	16	0.00006	18	0.00006	28
Mod5adders	6	21	64	0.00007	46	0.00007	51
5mod5tc	6	17	64	0.00007	38	0.00006	52
Hwb7	7	289	128	0.00016	220	0.00014	309
Rd53	7	30	128	0.00008	127	0.00010	188
Hwb9	9	1541	512	0.00084	851	0.00086	1176
Hwb11	11	9314	2048	0.01028	3312	0.00988	4557
0410184.nct	14	46	16384	0.00207	75	0.00009	105
Ham15a	15	132	328	0.99800	36617	1.00677	49594
Cycle17_3	20	48	1048	0.12637	255	0.00026	277
Mod1408576	40	210	-	Timeout	-	Timeout	-

In most of the benchmark circuits reported in Table 3.6 the average runtime for the implicit multiplication approach is slightly slower than the average runtime for the explicit multiplication approach (the exception is “Cycle17_3”). From average QMDD

size point of view, the implicit multiplication approach consistently requires smaller QMDD sizes [GTFM07]. This result indicates the implicit approach is better in terms of memory requirements while the explicit approach generally has more desirable runtimes.

The ability of the QMDD tools to simulate the output of the benchmark circuit files was used to create complete specifications in terms of the transformation matrix from the Maslov benchmark circuits. These specification files were used extensively in our reversible circuit synthesis work described in Section 4.4.

During simulation, any BDD based package periodically uses dynamic variable reordering to minimize the data structure and to perform appropriate garbage collection. The effect of the size and time requirements of dynamic variable reordering is discussed in detail in Sections 3.3 and 3.4.

CHAPTER 4

SYNTHESIS OF QUANTUM AND CLASSICAL REVERSIBLE CIRCUITS

Synthesis is the process of generating a circuit that implements a design specification. In this section we introduce several synthesis tools that have been developed in this research. Section 4.1 describes the *QCAexor* tool for synthesizing QCA circuits using ESOP minimization [FT07]. Section 4.2 discloses a method for virtual implementation of complex circuits in classical reversible logic using a commercially available synthesis tool [FTN07]. A comprehensive yet brief survey is provided in Section 4.3 regarding different approaches for synthesizing quantum reversible circuits using gate cascades. In Section 4.4 the *QMDDsyn* tool is described for augmenting lexicographical synthesis of reversible logic with improved variable ordering.

4.1. ESOP Techniques for Majority Gate Based Logic Synthesis for QCA

QCA synthesis is different than quantum circuit synthesis since the QCA architecture is not logically reversible. We investigate how to use the QCA native 3-input majority gates in an efficient way to synthesize logic circuits. In particular, the Exclusive-OR Sum of Products (ESOP) forms of logic description are transformed into an implementation composed of 3-input majority gates. It is well known that ESOP

logic minimization can often (but not always) produce better results than the more well known inclusive-OR Sum of Products (SOP) minimization form [FK00, FTY87, MP01, S99, SB90, SMA01, BHMS84, Thor95]. Better results are quantified in terms of area minimization which implies the resulting circuit description is composed of as few cubes as possible and that each cube covers as large a number of minterms as possible.

4.1.1. Multi-input XOR and AND/OR Implementation in QCA

A 2-input XOR circuit can be formed using three 3-input majority gates as shown in Fig. 4.1. The circuit performs the well known function

$$a \oplus b = \bar{a}\, b + a\, \bar{b} \tag{4.1}$$

All these majority gates are of the *first kind*, since they have one input at a fixed logic state, in accordance with [M63]. The schematic diagram of this gate is also shown in Fig. 4.1.

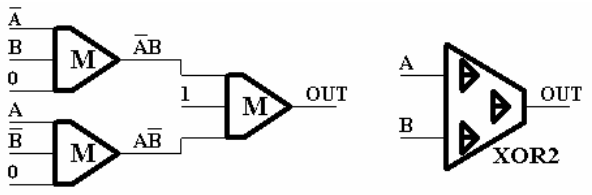


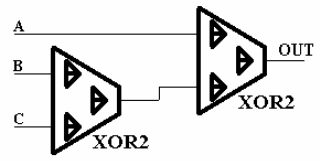
Figure 4.1. 2-input XOR Gate Implementation

A naïve implementation of a 3-input XOR gate can be achieved by connecting two XOR2 gates as shown in Fig. 4.2a. This implementation is inefficient as it consumes a total of 6 majority gates of the *first kind*. A better implementation of the

XOR3 gate is illustrated in Fig. 4.2b. Here we efficiently exploit only 3 majority gates of the *second kind*, namely, majority gates which use circuit variables in all of their 3 inputs. We can see that the **XOR2** gate is equivalent to the minimized **XOR3** gate of Fig. 4.2b with one input set to '0', performing

$$a \oplus b = a \oplus b + 0 \tag{4.2}$$

(a)



(b)

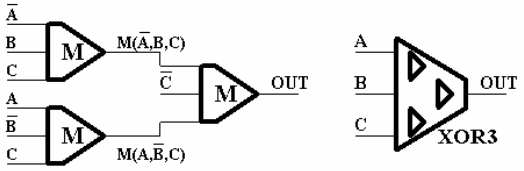


Figure 4.2. 3-input XOR Gate Implementation

In Fig. 4.3 we explore the architecture of XOR gates with 4 and 9 inputs (for more examples please refer to [FT07]). It seems that only **XOR3**, **XOR5**, **XOR7** and **XOR9** are relatively efficient in the sense that they exclusively use majority gates of the *second kind*. We assume, following [M63], that once all inputs of the 3-input majority gate are used we have a relatively efficient implementation. However, it is still possible that the techniques shown in [ZWWJ04, ZJ06], combined with factoring or decomposition, may still improve our results. However, our construction provides a reasonable base line estimation of the complexity of the general *k*-input XOR QCA implementation.

Based on the construction for n-input QCA XOR gates, the number of levels d_{xor} is:

$$d_{xor} = 2 \times \lceil \log_3 n \rceil \tag{4.3}$$

The number of 3-input majority gates m_{xor} is

$$m_{xor} = 3 \times \left\lfloor \frac{n}{2} \right\rfloor \tag{4.4}$$

Each XOR3 element requires 3 inverters. The total number of inverters i_{xor} is

$$i = \begin{cases} m & if \quad n \quad is \quad odd \\ m-1 & if \quad n \quad is \quad even \end{cases} \tag{4.5}$$

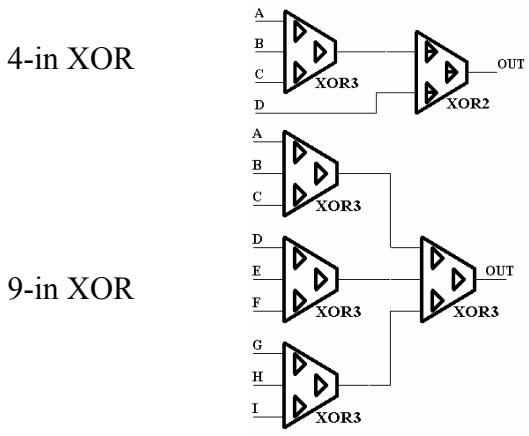


Figure 4.3. Implementation of multi-input XOR Gates

Clearly, the complexity of XOR implementation is O($\log_3 n$) delay and O(n) in size. For example, an **XOR10** will use 14 majority gates, 13 inverters, and have a delay of four. We intuitively construct multi-input AND or OR gates using majority gates of

the *first kind* arranged in a binary tree like architecture. Fig. 4.4 illustrates the construction of a 9-input AND/OR gate. Note that setting *S* to 0 selects the AND functions, while *S*=1 selects the OR function.

Based on this construction, the number of levels d_{and} is:

$$d_{and} = \lceil \log_2 n \rceil \tag{4.6}$$

The number of 3-input majority gates *m* is

$$m_{and} = n - 1 \tag{4.7}$$

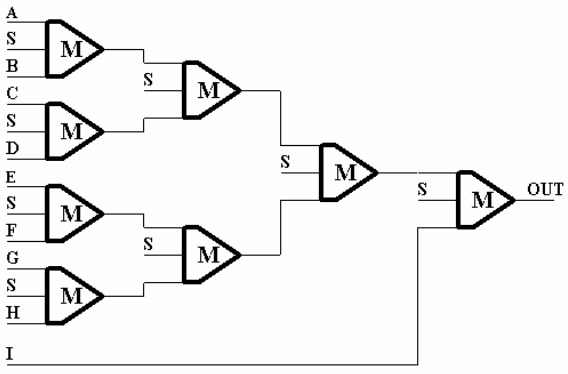


Figure 4.4. Implementation of a 9-input AND or OR Gates

The OR and AND functions have the same size and delay since the only difference is the value of S. Notice that the OR/AND construction follows the same asymptotic complexity (O($\log_2 n$) delay and O(n) size) as the XOR construction. However, it is clear that even before considering the extra cost of inverters for the XOR construction, the AND/OR is significantly smaller.

4.1.2. The QCAexor Tool

An overview of the QCAexor tool is shown in Fig. 4.5. In the pre-processing step, the benchmark files in the *.pla format are minimized by ESPRESSO [BHMS84] for SOP cube list. The resulting files are minimized using EXORCISM4 to produce corresponding ESOP cube lists in a *.esop file. Size comparisons between benchmark files in the *.pla and *.esop formats are reported in [MP01].

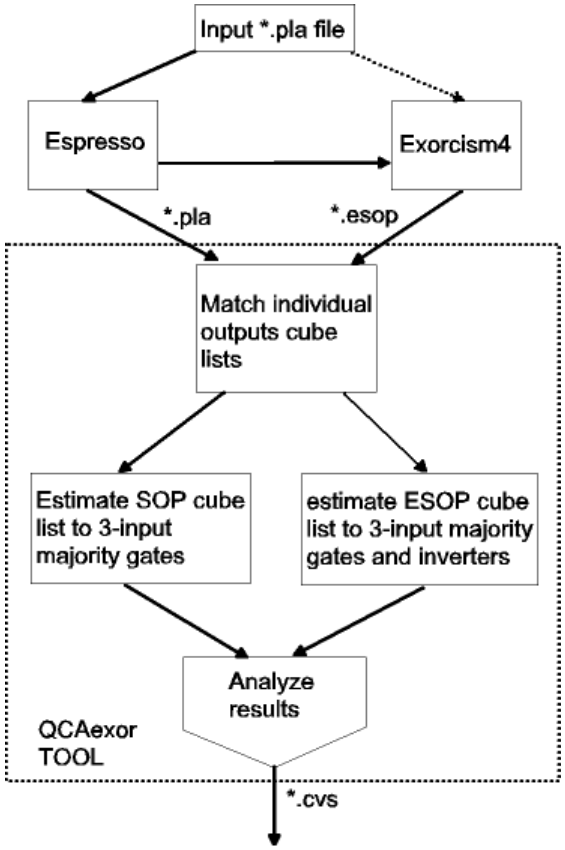


Figure 4.5. An Overview of the QCAexor Tool

The QCAexor tool accepts both the *.pla and *.esop forms of circuit netlist files. The tool determines equivalent individual circuit outputs and processes each separately.

QCAexor estimates the size of the SOP cube lists when implemented as majority gate netlists. Similarly, it estimates the size and delay of the ESOP cube list. The tool also computes the numbers of inverters required by each XOR gate implementation since inverters are expensive resources for QCA technology. After this analysis is performed, QCAexor saves the comparison results in a comma delimited file, *.cvs.

The implementations based on the subcircuits described in section 4.1.1 require the ESOP cube list to be implemented by a single multi-input XOR function and several multi-input AND functions. Similarly, the SOP cube list uses the simpler OR implementation. Therefore, it is expected that ESOP synthesis may show a gain over SOP synthesis only when the number of cubes (that is, the size of the XOR) and their sizes are smaller. For this reason it is useful to compare each output in the ON-set list rather than the entire circuit (which is the common practice when considering standard CMOS logic [MP01]).

4.1.3. Experimental Results in ESOP vs. SOP QCA Synthesis

A large number of the standard PLA benchmark circuits were synthesized and representative results are given in Table 4.1. The first column depicts the circuit output investigated in each row of the table. Columns 2-4 show the results for that output using the SOP specification data, and columns 5-8 show the results for the ESOP realizations. The second and fifth columns show the number of product terms associated with the output. The third and sixth columns show the total size in terms of required majority gates. For the ESOP implementation, column 7 shows the extra

number of inverters required by the XOR gate in the second level of the circuit. We show the maximum delay, as measured by majority gate levels, in columns 4 and 8.

The GAIN ESOP column shows the *size* benefit (or loss) when an output is implemented in ESOP instead of SOP form, and it is computed as:

$$GAIN = 100\% \ (SOPsize – ESOPsize – ESOPPinverters)/SOP\ size \qquad (4.8)$$

A negative gain result indicates that the SOP form results in smaller sized circuits than the ESOP form. Table 4.2 shows the results of the benchmark circuit *ALU4.PLA*. The ESPRESSO minimization of the 14 inputs/8 outputs circuit in Table 4.1 required 575 cubes, while the EXROCISM4 minimization required 426 cubes. The large negative results for outputs 0, 2 and 3 validates our approach to investigate each output individually as these negative results would clearly mask the benefit of ESOP implementation for outputs 1, 4, 5, and 6.

Table 4.1. Comparison for ALU4.PLa - 14 inputs/8 outputs

Out	**SOP prds**	**SOP size**	**SOP delay**	**ESOP prds**	**ESOP size**	**ESOP invs**	**ESOP delay**	**GAIN ESOP**
0	16	67	7	31	218	45	7	-292%
1	12	51	6	5	15	6	6	58%
2	50	247	9	51	466	75	9	-119%
3	72	423	10	130	1327	195	10	-259%
4	186	1594	12	53	481	78	12	64%
5	90	614	10	25	161	36	10	67%
6	36	206	9	9	37	12	9	76%
7	182	1927	12	210	2183	315	12	-29%

We summarize our results with various benchmark circuits in Table 4.2. Column 4 indicates the total number of cubes in the ESPRESSO minimization of the circuit, while column 5 shows the minimization results with EXORCISM4. In column 6 we show how many of the outputs can be implemented based on ESOP minimization with a gain over the results with SOP minimization. We also show the best gain and the worst loss achieved during the processing of all the outputs individually.

Table 4.2. Potential Gains in ESOP Synthesis for QCA

Benchmark circuit	**Inputs**	**Outputs**	**ESPR. Minim.**	**EX-4 Minim.**	**ESOP GAINS**	**Best gain**	**Worst loss**
5xp1.pla	7	10	65	31	6 of 10	61%	-100%
dc2.pla	8	7	39	32	3 of 7	18%	-64%
alu1.pla	12	8	19	16	0 of 8	0%	-150%
seq.pla	41	35	336	246	6 of 35	74%	-333%
vg2.pla	25	8	110	184	0 of 8	none	-303%
alu4.pla	14	8	575	426	4 of 8	76%	-259%
9sym.pla	9	1	86	51	1 of 1	8%	none
c8.pla	28	18	79	50	4 of 18	31%	-166%
z5xp1.pla	7	10	65	33	9 of 10	89%	-20%
tial.pla	14	8	581	428	4 of 4	76%	-317%
x6dn	39	5	82	95	0 of 5	none	-120%
vda	17	39	93	87	4 of 39	12%	-2820%

It is important to note that QCAexor was able to perform the entire analysis for all benchmark circuits in a fraction of a second, so this negligible amount of time is not added to the results.

4.2. Virtual Implementation of Reversible Logic using Direct Translation

We investigated a virtual implementation of reversible circuits that use a direct translation of complex binary functions into circuits composed of Fredkin reversible gates [FTN07]. We note in section 2.2.4 that the Fredkin gate can realize any arbitrary Boolean function. We investigate the Brent-Kung Prefix Parallel Adder (PPA) as an example for a complex yet regular logic circuit [BK82, LF80, Zimm98, EL04].

Since a classical logic n-bit PPA requires $2n$ inputs and n outputs (neglecting the carry-in and the carry-out bits), a reversible counterpart will require at least $2n$ variables or lines. As shown in the next section, current reversible logic synthesis based on gate cascading reported various size limits of 8-25 variables [MMD03]. In fact, very few benchmark circuits for reversible circuits with more than 25 variables are listed in the Maslov's benchmark circuits page [Masl05]. Therefore, it is fair to assume that 32 and 16 bit adders are beyond the scope of current reversible logic synthesis based on gate cascading. We follow a direct translation approach similar to the work in [BTS+02]. This results in an output VHDL file that is appropriate for *Field Programmable Gate Array* (FPGA) implementation. The emphasis is on circuit regularity as opposed to minimization of the number of garbage outputs and ancillary inputs. We recognized in

advance that our results will include a large number of such garbage outputs and ancillary inputs, and are therefore useful only for the purpose of virtual implementation in our current investigation. Our implementation maintains direct correlation between the sub-modules of the reversible result to their standard logic counterpart. We use the Altera Corporation's Quartus II software, a comprehensive suite of tools for the synthesis, simulation, and implementation of complex logic circuits [Altera].

We first modeled the basic Fredkin gate using CMOS transmission gates as shown in Fig. 4.6. As VHDL implementation of the transmission gate is not a trivial matter [Leun89], we found that the approximation of a one-way transmission gate using the built-in Altera "TRI" primitive is useful for our application.

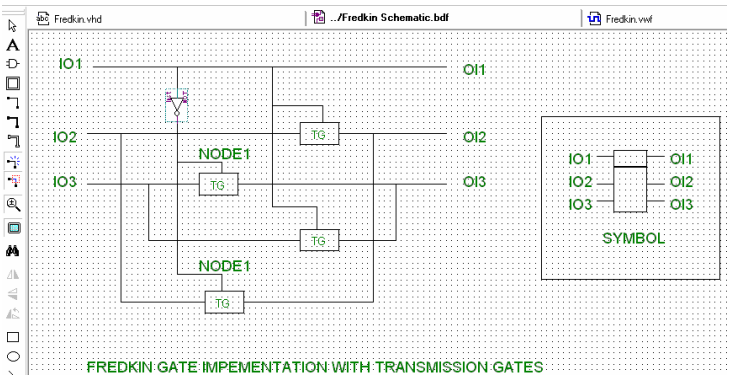


Figure 4.6. VHDL Model of a Fredkin Gate

We use Zimmerman's comprehensive parameterized VHDL library for the implementation and investigation of various PPA architectures [Zimm98]. We modified the Zimmerman VHDL files to create the Brent-Kung PPA using the Fredkin gate model. We used direct translation by replacing each original standard logic

equation with a corresponding Fredkin implementation corresponding to the mappings of Fig. 4.6. The resulting files were then used as input to the *Quartus II* tool for synthesis and simulation. The simulation tool was quite useful for debugging translation errors occurring during the design decomposition into Fredkin gates.

Our research methodology was to compare the size and the delay simulation results between reversible and standard logic versions of the Brent-Kung PPA of various operand sizes. An interesting pitfall of our approach quickly appeared when we tried to place and route the resulting synthesis of both reversible and classic logic versions into Altera's FPGA chips. Our motivation was to use the number of *Logic Element* (LE) units reported at the end of device compilation as a measure of size. Fig. 4.7 shows the Place and Route map result when a reversible 16-bit Brent-Kung PPA is implemented in an Altera EP20K30E FPGA device. Each small square within the chip map represents one LE unit.

When we placed and routed reversible and standard logic versions of same size adders, the optimization and minimization phase of the *Quartus II* tool inadvertently converts the reversible logic files back to the standard logic implementation. The solution was to use the *Quartus II* 's register transfer level (RTL) viewer which shows the non-optimized VHDL files and to manually count the number of Fredkin gates. An example of the RTL Viewer output for the 8-bit Brent-Kung PPA is shown in Fig. 4.8.

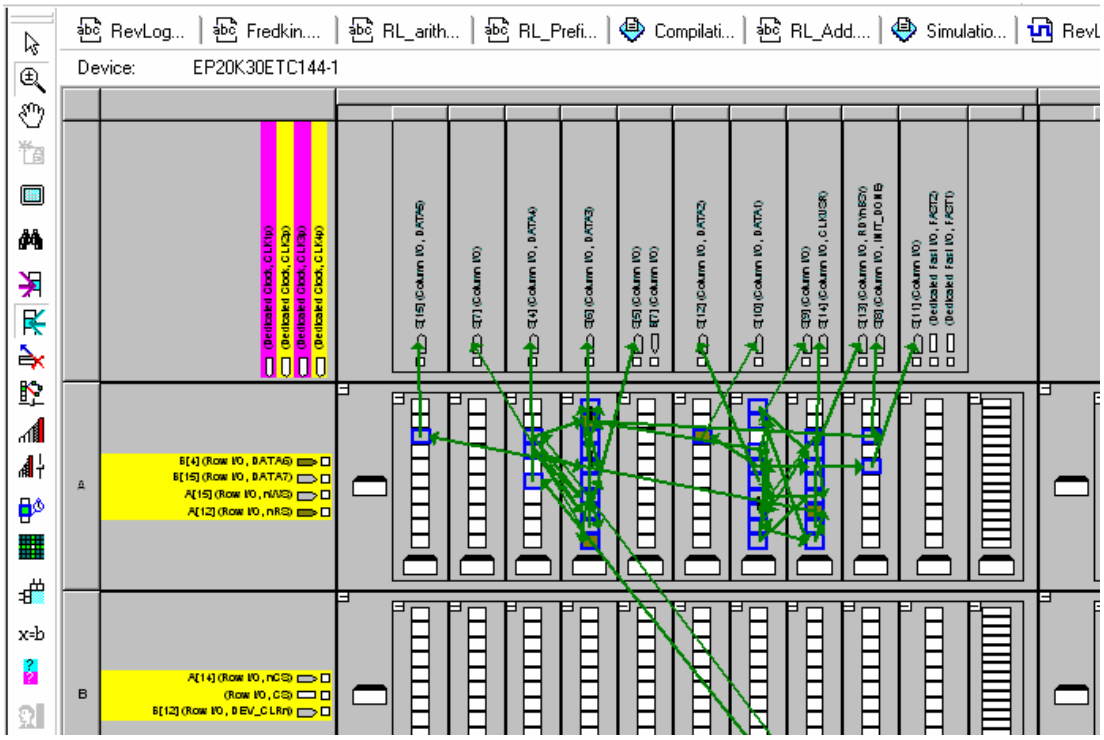


Figure 4.7. Place and Route Map of the 16-bit Brent-Kung PPA

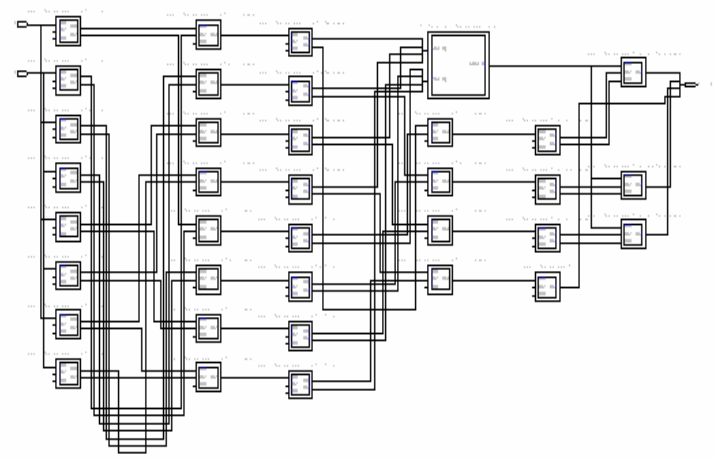


Figure 4.8. The RTL Viewer Output for the 8-bit Brent-Kung PPA

We compare our virtual reversible logic implementation of the Brent and Kung PPA with the standard logic implementation in Fig. 4.9. Our results indicate that our virtual reversible logic implementation still follows the $O(\log_2 n)$ delay and $O(n)$ cost of the standard logic implementation. Due to the complexity of the Fredkin gate as compared to a standard logic gate, the cost of the reversible logic implementation is roughly four times that of the standard logic implementation. However, this is because the target technology used in the Altera tool is classical CMOS circuitry and not quantum logic technology.

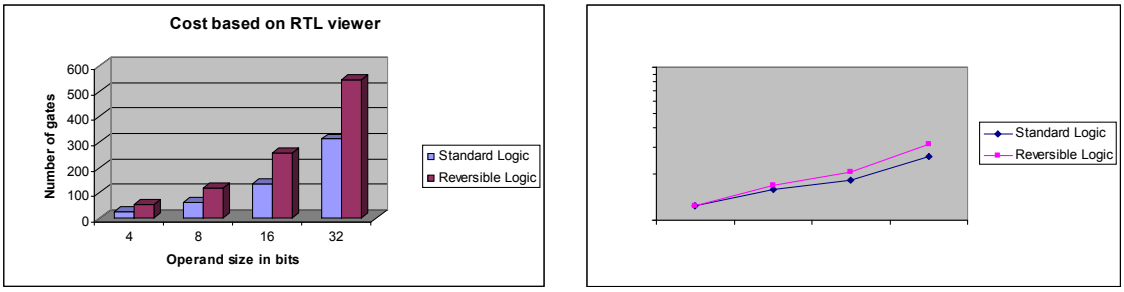


Figure 4.9. The Size and Delay Comparison of Classical and Reversible PPAs

We could not compare power consumption estimates since the *Quartus II* tool Place and Route process converts the reversible implementation back to the standard irreversible logic implementation.

4.3. Survey of Synthesis Methods for Reversible Gate Cascades

Many recent papers and several books deal with synthesis of quantum reversible circuits. In this section we describe recent results on this topic in order provide a survey

of the state-of-the-art. Most synthesis approaches utilize a cascade of gates netlist as described in Definition 2.10 as input and attempt to minimize the number of gates. Due to the relative absence of don't-care assignments in reversible logic, the synthesis usually start with a complete specification of the circuit. In classical reversible logic, the specification is a permutation that transforms the input to the output in a bijective manner. In quantum logic the specification is typically a unitary matrix that describes the transformation that the target circuit is to provide.

4.3.1. Exhaustive Approaches for Synthesis

In the exhaustive approach, the synthesis algorithm attempts all the possible choices of gates and their connections in a gate cascade until the circuit is able to achieve the desired permutation [Kern00]. While this approach provides optimal solutions, it is useful for only a very small number of inputs due to the explosive search space size. Typically, exhaustive synthesis techniques are used on small benchmark circuits to determine the optimal solution in order to test the performance of other more efficient but non-optimal synthesis approaches based on heuristics.

Shende et al. noted that even for small sized circuits of just three inputs, the exhaustive algorithm requires hours to complete [SPMH02, SPMH03]. For example, using the CNT library (Section 2.2.1), a 3-input circuit creates 8!=40,320 permutations. They explored the use of branch-and-bound algorithms to create a library of optimal circuits while reducing the search space. They utilized the property that any portion of an optimal circuit is also optimal. However, the runtime of their algorithm for finding optimal circuits is still exponential with respect to the number of variables.

4.3.2. Synthesis Using Formal Verification Methods

Formal methods that heuristically deal with pruning the search space have been applied to reversible logic synthesis by Hung et al. [HSY+06]. They used nonpermutative quantum gates with symbolic reachability analysis in order to achieve optimal synthesis, or suboptimal synthesis with some speed up. In effect, they reduce the synthesis problem of reversible logic into a multiple-value logic synthesis problem.

Groβe et al. developed an algorithm to synthesize reversible functions using generalized Toffoli gates [GCD06, GCDD07, WG07]. They formulate the synthesis of *d* Toffoli gates as a sequence of Boolean Satisfiability (SAT) instances, where satisfiability indicates that a circuit of *d* gates exists. As they iterate their algorithm with increasing *d* size, they are able to guarantee minimality.

4.3.3. Template Based Synthesis

Since the problem of quantum circuit synthesis is intractable, various synthesis techniques tend to produce circuits that contain unduly complex sub-circuits that cannot be easily minimized. A common minimization technique is to attempt to match different sections of circuit to simpler stored equivalent sub-circuits called templates. The process of template matching can be accomplished using transformation rules. Therefore, template matching for circuit simplification can be regarded as performing a local optimization.

A set of six local transformation rules to reduce the cost of circuits comprised of *n*-variable Toffoli gates was developed by Iwama et al. [IKY02]. Using the notation of [*t*,*C*] described in Section 2.2.1, their set includes the following rules:

- $[t_1, C_1] \bullet [t_1, C_1] \Leftrightarrow \varepsilon$, namely two identical CNOT circuits in series cancel each other.
- $[t_1, C_1] \bullet [t_2, C_2] \Leftrightarrow [t_2, C_2] \bullet [t_1, C_1]$ only if $t_1 \notin C_2$ and $t_2 \notin C_1$. That means that gates order is *not important* if the gates are independent.
- $[t_1, C_1] \bullet [t_2, C_2] \Leftrightarrow [t_2, C_2] \bullet [t_1, C_1] \bullet [t_1, C_1 \cup C_2 - \{t_2\}]$, if $t_1 \notin C_2$ and $t_2 \in C_1$. This transformation allows us to make a sequence of the gate B after A equivalent to the sequence of gate A after B, provided that the target of gate B is part of gate A's control bits.
- $[t_1, C_1] \bullet [t_2, C_2] \Leftrightarrow [t_2, C_1 \cup C_2 - \{t_1\}] \bullet [t_2, C_2] \bullet [t_1, C_1]$, if $t_1 \in C_2$ and $t_2 \notin C_1$. This transformation is the dual of the previous one.
- $[t_1, \{c_1\}] \bullet [t_2, C_2 \cup \{c_1\}] \Leftrightarrow [t_1, \{c_1\}] \bullet [t_2, C_2 \cup \{t_2\}]$, if $t_1 > n+1$ and there is no Toffoli gate with target on t_1 before the gate $[t_1, \{c_1\}]$.
- $[t, C] \Leftrightarrow \varepsilon$ if the gate has an auxiliary control bit that remains at |0>.

They also proved that any quantum circuit *S* can be transformed into the canonical form as specified in [IKY02] by using only these six transformation rules.

Template matching based synthesis was reported in Maslov et al. [MDM03]. They noted that although the template set was quite small, the potential for reduction in the synthesized circuit size is quite significant. Figure 4.5 illustrates the MMD template

for 2×2 and 3×3 circuits composed of CNOT and Toffoli gates for post synthesis simplification, as reproduced here from their paper [MMD03].

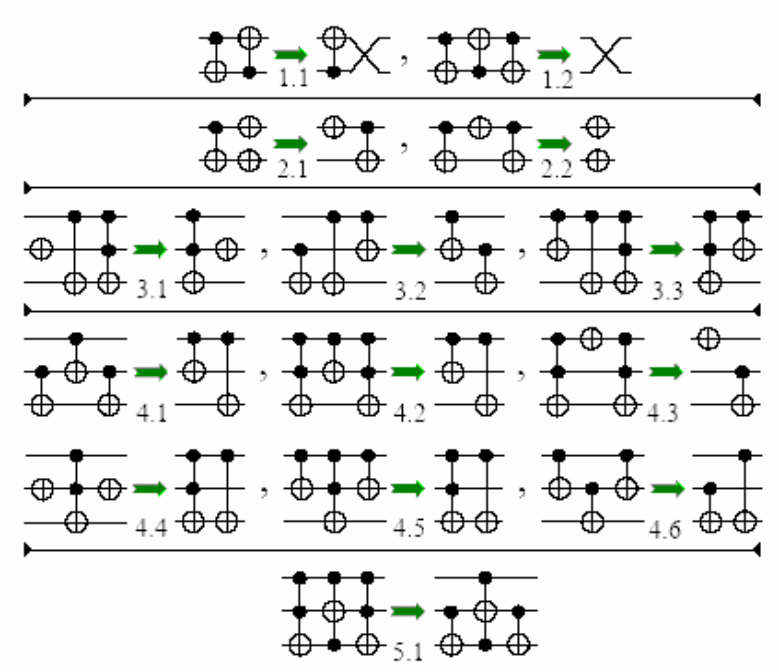


Figure 4.10. MMD's Templates for 2x2 and 3x3 Quantum Circuits

4.3.4. Lexicographic Based Reversible Logic Synthesis

In this section we describe non-search based reversible logic synthesis techniques that proceed in a lexicographic order consistent with the function specification.

Definition 4.1 A *Complete specification* of a reversible circuit lists all output permutations of the function in a lexicographic order as represented by the function's truth table.

In 2003, Miller, Maslov, and Dueck proposed the well known "MMD" reversible logic synthesis, a non-search based lexicographic algorithm [MMD03]. The algorithm inspects the complete specification in a lexicographic order and for each input it implements a series of one or more gates that translate the input pattern into the output pattern. The gates used by the MDD algorithm are from the NCTSF library. The key idea is that once an input pattern is transformed into the output pattern, this output must not be changed, hence the lexicographic order. An interesting aspect of the algorithm is that the transformation can simultaneously proceed from the inputs to the outputs and from the outputs to the inputs.

At each point in the lexicographic scan of an $n \times n$ complete specification, the algorithm heuristically selects the required translation gates without a search. This results in a relatively fast algorithm which completes the synthesis of an $n \times n$ function in time $O(n2^n)$. The algorithm has two variants:

- The uni-directional "naïve" algorithm that adds gates while permuting from the output specification to the input; and
- The bi-directional algorithm that advantageously synthesizes from both directions.

Both algorithms are guaranteed to terminate with a valid circuit although the generated circuit is not necessarily optimal. Therefore, the MMD algorithm employs a

post synthesis smoothing process which is based on template matching [IKY02, MDM03]. Template matching often requires multiple consideration of the same area of the circuit as smaller portions are being modified. This emphasizes the importance of providing smaller circuits to the template matching step.

Another non-search based reversible logic synthesis algorithm was recently introduced by Saeedi et al. [SSZ07]. Their algorithm scans a circuit specification lexicographically, and iteratively exchanges pairs of minterms that are out of order in each step. In this dissertation we contribute an improvement for lexicographic synthesizers as described in Section 4.4.

4.3.5. Group Theory and Symmetry Based Synthesis

De Vos et al. investigated the structure of the group R_w that contains all reversible gates and their cascades with w inputs and outputs [DRS02]. They created a chain of maximal subgroups and found that the subgroup of control gates C_w (which includes Toffoli gates) is particularly useful to generate such chains. The control gates C_w subgroup form one of the Sylow 2-subgroups of the entire R_w group.

Galois field sum of products (GFSOP) is used in [KPK03] to synthesize multiple-valued reversible circuits efficiently using a generalization of ternary Toffoli and other ternary gates. They proposed a local mirror technique whereby a garbage output is converted to a constant (e.g. $|0\rangle$) which is used as an input to the next stage of

the cascade. The circuit is minimized in a gate cascade that utilizes local mirrors, efficient variable ordering, and product ordering..

Symmetric function detection can help simplify the synthesis of reversible logic as described in Perkowski et al. [PKB01]. They devised a 2×2 net regular structure that is easier for reversible logic synthesis of symmetric functions. They extend the method to other Boolean functions using the fact that non-symmetric functions can become symmetric by repeating selected inputs. For this approach, the best improvement in terms of garbage output minimization was achieved with symmetric functions and with incompletely specified functions comprising many outputs. Such incompletely specified functions could have their "don't-cares" exploited to cause them to exhibit more symmetry. Maslov also reported improvement in the cost of the realized circuits when he used a dynamic programming algorithm to realize symmetric functions in reversible logic [Masl02].

In [GFX06] the authors use symmetric and group theory methods to efficiently construct *n*-qubit reversible functions. Observing the relations between the input and output patterns, they found a realization using CNOT gates that improves upon the size and time complexities of previous algorithms. Still, these complexities grow exponentially with circuit's size.

4.3.6. Spectral and Reed-Muller Transformation Techniques

Miller developed a synthesis method for reversible logic using the Rademacher-Walsh spectral transform and two-place decompositions of Boolean functions [Mill02,

MD03b]. He first defined a table of the logic computation of NOT, AND, OR and EXOR in the spectral domain. The reversible logic specification is then transformed to the spectral domain. Within the spectral domain, the algorithm attempts to identify single Feynman, Toffoli and Fredkin gates that may be applied to specific inputs. The spectral NOT, AND, OR, and EXOR are then applied to achieve the overall synthesis.

Permutation-based fast transforms for multiple-valued quantum circuit synthesis were developed by Al-Rabadi [Alra04]. The algorithm uses two planes, the first for reversible permutation butterfly circuit in the spectral plane, and the second for the actual quantum gates that perform the actual additions and multiplications.

Agrawal and Jha describe a heuristic-based tool called RMRLS for the synthesis of reversible logic using the positive-polarity Reed-Muller decomposition at each stage [AJ04]. A priority queue based search tree attempts to find the best factors at each stage. The Positive Polarity Reed-Muller expansion of a function is uniquely represented in the form:

$$f(x_1, x_2, ..., x_n) = a_0 \oplus a_1 x_1 \oplus ... \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus ... \oplus a_{n-1,n} x_{n-1} x_n \oplus .... \\ \oplus a_{2...1n} x_1 x_2 .... x_n \tag{4.9}$$

where $a_i \in \{0,1\}$ and all x_i are un-complemented for positive polarity. They outlined several examples which provided the same results obtained by other researchers in a shorter synthesis time. In a later version of that work [GAJ06], the authors were able to synthesize most randomly generated reversible functions with up to 25 gates and 16 variables.

4.3.7. Synthesis Approaches Based on Decision Diagrams

Kerntopf [Kern04] offered a PPRM heuristic algorithm for reversible logic synthesis using a complexity measure based on shared binary decision diagrams (BDD) with complemented edges. This replaces complexity measure that relates to the complexity of the specification as provided by the circuit truth table or its PPRM forms. Since the algorithm produces several implementations, the user can globally select the best among them. Experimental results in this paper also relate to only small circuits with few inputs/outputs and with up to 14 gates.

Abdollahi and Pedram developed a synthesis framework for quantum logic that uses the basic rotation and controlled rotation operators [AP05]. We have already described their *binary control signal constraint* issue in Section 3.1.4. They produced a canonical representation tool called *quantum decision diagrams* (QDD) that apply recursive unitary functional bi-decomposition of the original graph created when the circuit specification is loaded onto the tool. Like QMDD and classical logic BDDs, the QDD tool recursively implements the quantum APPLY operation to combine to QDD nodes. They detailed two variants of their synthesis algorithm, the *q-factor* and the modified *q-factor*, and show promising synthesis results.

4.4. Using QMDD Minimization for Efficient Synthesis

An immediate problem with a lexicographic order is the fact that it may lead the synthesis into an excessively non-optimal circuit implementation. In our research, we

were motivated to improve the synthesis by adjusting the lexicographic order based on the best dynamic variable reordering obtained by the QMDD minimization technique described in Chapter 3.

In classical irreversible logic, Fujita et al. used the structural shape of the circuit to select the best initial variable order for the implementation of the binary decision diagram [FMK91]. They minimized the number of wire crossings and gave priority to fan-outs encountered during their depth-first transversal from the outputs to the inputs of the circuit. We found that minimizing the QMDD representation of the specification and using the corresponding variable order significantly improves the lexicographic reversible logic synthesizer [FTM08c].

4.4.1. Circuit Complexity versus QMDD Metrics

In this section we investigate the relation between the reversible circuit size complexity and the size metrics of the QMDD that represent the circuit.

Lemma 4.1: Two equivalent reversible circuits composed of a different number of gates result in the same QMDD for a given variable ordering.

Proof: Directly from the canonic property of QMDD [MT06], and following the discussion in [FTM08b].

Assume that circuit *C1* comprised of *s* gates and circuit *C2* comprised of *t* gates are two distinct implementations of the same function specification, where $t>s$. Since Lemma 1 indicates that *C1* and *C2* are represented by the same QMDD, the QMDD size metrics cannot be used as an absolute measure of circuit complexity. Reversible

circuits may include partially redundant logic that artificially increases the size as shown in [FTM08b]. Furthermore, the following lemma illustrates that an arbitrarily large section of a synthesized circuit may represent the functionality of an identity sub-circuit that is not reflected in the size of the QMDD, but that unnecessarily increases the size of the gate cascade.

Lemma 4.2: There exists an $n \times n$ trivial identity reversible circuit cascade with any arbitrary even number of gates.

Proof: Let $m=2k$ be the number of gates, where k is an integer so that $k \geq 0$. For $k=0$, the circuits merely passes all the lines and therefore is a trivial identity circuit. For $k=1$, we make a cascade of two identical arbitrary gates. It is easy to see that such a circuit is a trivial identity reversible circuit. For $k>1$, we first build a cascade C of k arbitrary gates. We then build another cascade C' using exactly the same gates but in a *reversed order*. Let A and A' be the transformation matrices of C and C' respectively. Since C is equivalent to reversing a cascade so that the outputs exchange with the inputs, $A'=A^T$. It then follows that $A' \times A = A^T \times A = I$, which completes the proof. □

Example 4.1: We illustrate the implication of Lemma 4.2 in the inability of the QMDD size to predict the size of the reversible circuit with the example in Fig. 4.11. We simply add 10 gates to the left of circuit syn_3v_rnd1, comprising two groups of five gates. The second group is identical but in reverse order of the first group in accordance with Lemma 4.2. We have noted the quantum cost of each circuit based on Definition 2.11 and on the work of Maslov [Masl03].

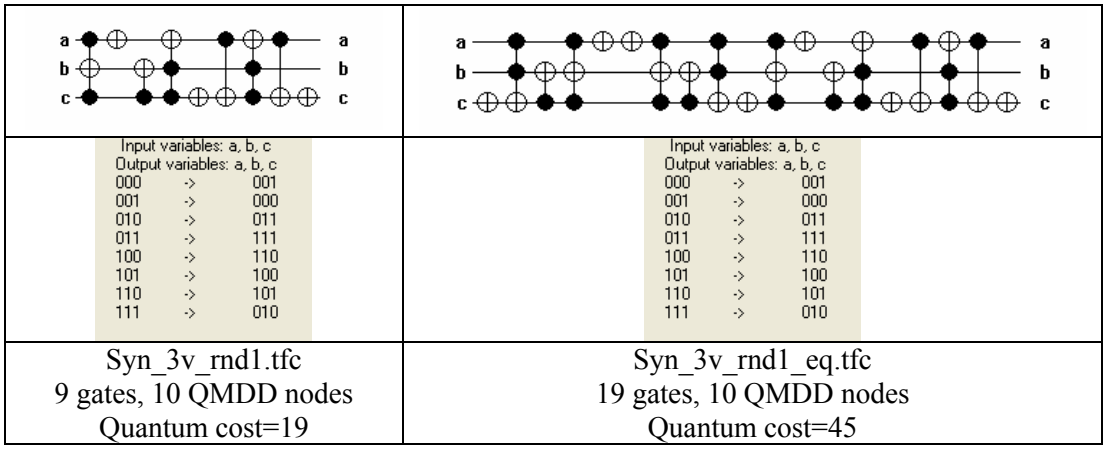


Figure 4.11. Functionally Equivalent Circuits of Different Sizes

We now consider the relation between circuit size and QMDD size in more detail. An identity $n \times n$ circuit (one that maps each input pattern to the same output pattern) requires only n non-terminal nodes in its QMDD representation.

Since a binary QMDD has four edges exiting from each node, the upper limit for QMDD size is reached if each edge points to a different node in a unique way. This condition leads to a geometric series with size equal to

$$S_n = \sum_{i=1}^{n} 4^{i-1} \tag{4.10}$$

However, for classical reversible logic, the transformation matrix represented by the QMDD has only a single '1' element in each row and each column [MFT07]. This immense sparsity leads to much smaller QMDD size as shown in the following experimental results.

Table 4.3 shows the maximum and minimum theoretical limits for the QMDD size as related to the number of variables in the circuit. The first column depicts the number of variables, and the largest QMDD size predicted by equation (4.10) is shown in the third column.

Table 4.3. Maximal and Minimal QMDD' Size vs. Circuit Size.

Variables	Minimum QMDD size	Maximum QMDD size
2	2	5
3	3	21
4	4	85
5	5	341
6	6	1365
7	7	5461
8	8	21845

Our experimental results are shown in Table 4.4. We explored circuits of 5, 6, and 7 variables, with number of gates ranging up to 100. For each circuit with a given number of gates, we show the quantum cost (see Definition 2.11) and the number of nodes in the QMDD representing the circuit. The results include published benchmark circuits as well as randomly generated circuits. We tried to match a published benchmark circuit to the required number of gates (depicted in each row) even if the benchmark circuit size is slightly different.

The results in Table 4.4 that use published benchmark circuits are marked with a file identifier as follows: 1 – "mod5d1" (8 gates), 2 – "hwb5tc" (55 gates), 3 – "2of5d1" (18 gates), 4 – "rd53d1" (12 gates), and 5-ham7tc (23 gates).

Table 4.4. Maximal and Minimal QMDD Size versus Circuit Size.

Gates	5 -vars		6-vars		7 -vars	
	Quant. cost	**QMDD size**	**Quant. cost**	**QMDD size**	**Quant. cost**	**QMDD size**
0 (Ident)	0	5	0	6	0	7
1	5	13	1	11	5	17
10	24 (1)	19 (1)	158	42	120 (4)	26 (4)
20	208	36	158 (3)	56 (3)	81 (5)	130 (5)
30	310	41	494	82	836	137
40	424	46	696	77	968	141
50	510	44	830	84	1340	134
60	313 (2)	47 (2)	1016	83	1712	153
70	682	42	1186	72	1968	149
80	808	46	1388	82	2344	140
90	918	42	1388	82	2716	164
100	1036	45	1824	83	3047	176

The results demonstrate that the QMDD size does not follow the circuit size once the circuit size passes 20-40 gates. In contrast, the quantum cost does follow the circuit size, as expected. For example, for 5-variable, a circuit with 90 gates is represented by a QMDD with 42 nodes, less than the 46-node QMDD that represents the circuit with 40 gates. These experimental results illustrate the outcome of Lemma 4.2, as well as the work on partially redundant reversible logic reported in [FTM08].

Due to the large degree of sparsity commonly present in most transformation matrices, Table 4.4 shows that the size range is significantly smaller than the maximum size predicted by (4.10). This finding corresponds to the fact that most edges in a QMDD are essentially zero-weighted, pointing to the terminal node (see Section 3.4).

Fig. 4.12 demonstrates how the size of QMDD for random circuits of six variables relates to the number of gates. A circuit composed of no gates (i.e. one that is only a quantum register), the functionality is described by the identity transformation matrix and the QMDD size is the minimal 6 nodes. As we increase the number of

gates, the number of nodes grows while the QMDD data structure implements the deviation of the transformation matrix from the simple identity. However, this growth is neither monotonic nor linear. With larger numbers of gates, we see much less correlation in QMDD size as compared to circuit size. Our experiments also show that adding and removing NOT gates often has little incremental effect on the resulting QMDD size.

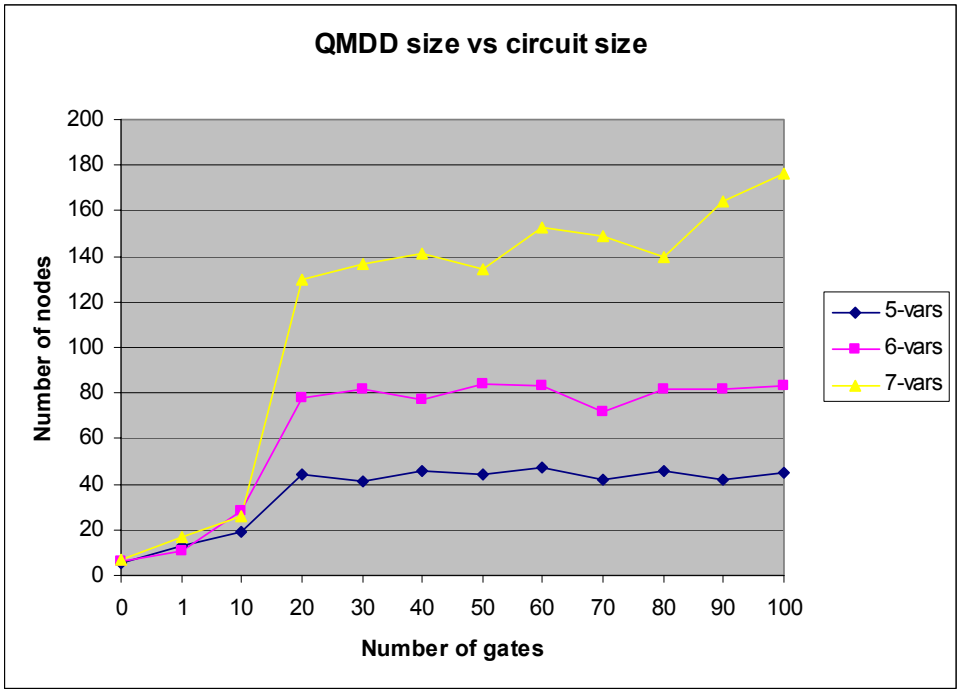


Figure 4.12. QMDD Size for the Circuits of Table 4.4.

4.4.2. Lexicographic Synthesis Approach

We have seen in Section 4.3.4 that a reversible logic circuit lexicographic synthesizer can be implemented using the time consuming step of template matching reduction in order to minimize the size of the resulting circuit. We propose the "QMDDsyn" framework to improve such template-matching based synthesizers using the best variable order obtained when the QMDD that represents the function specification is minimized. The key idea is to delay the time consuming process of template matching until it can be provided with a smaller circuit to optimize. Fig. 4.13 illustrates this process.

We first synthesize the complete specification lexicographically to obtain the baseline circuit *C1*. The circuit is then converted into a QMDD representation and a sifting minimization using *dynamic variable ordering* (DVO) is performed [FMT07b, YMS06]. While the QMDD can be built directly from the complete specification, we must have circuit *C1* in order to gauge the efficiency of the proposed framework's results.

The full specification is reordered based on the variable order that minimized the QMDD. The re-ordered specification is used to lexicographically synthesize circuit *C2*. Since we use the bidirectional version of the MMD algorithm, the framework must also re-order the complete specification in the reversed variable order of the minimized QMDD. This reversed re-ordered specification is used to synthesize circuit *C3*.

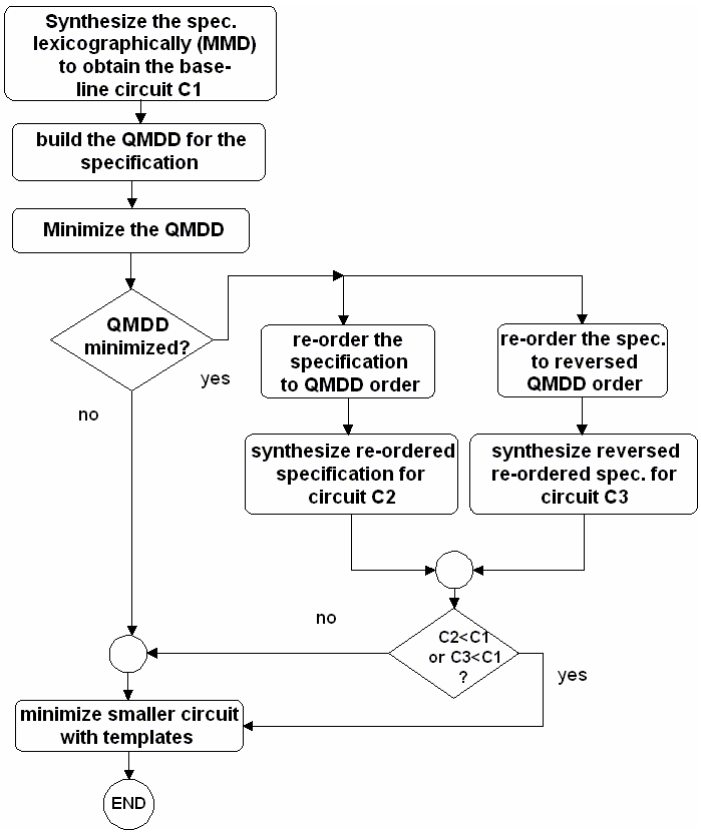


Figure 4.13. The "QMDDsyn" Framework

The framework then selects the smaller of circuits *C1*, *C2*, and *C3* for the final step of template matching minimization. Since the QMDD minimization may provide several variable orders that produce the same minimized QMDD, the process of Fig. 4.13 is repeated to obtain the smaller circuit.

Example 4.2: In the following example the QMDDsyn framework demonstrates the synthesis of the 3-variable function specification {5,3,4,7,2,6,1,0}. This specification list the order of the output minterms as related to ascending input

minterms. We first use the MMD unidirectional algorithm to produce circuit C1 of 12 gates with quantum cost of 30 as shown in Fig. 4.14a.

While this initial circuit may not be optimal, it is an exact implementation of the specification. Using the standard variable order of {*a,b,c*}, *C1* is represented by a QMDD with 10 non-terminal nodes. Using the minimization techniques reported in [MFT07b, FTM08b], the QMDD is reduced to 9 non-terminal nodes with variable order of {*b,a,c*}.

The specification is reordered by first applying the new variable order as shown on the left truth table of Table 4.5. The results need to be ordered lexicographically according to the input as shown in the right truth table of Table 4.5.

Table 4.5. Re-ordering the Specification for Example 4.2.

b a c	**b'a'c'**
000	011
010	110
100	001
110	111
001	100
011	101
101	010
111	000

Ordering lexicographically →

b a c	**b'a'c'**
000	011
001	100
010	110
011	101
100	001
101	010
110	111
111	000

The MMD unidirectional algorithm is applied on the reordered specification {3,4,6,5,1,2,7,0} to produce circuit *C2* of 7 gates with quantum cost of 11 as shown in Fig. 4.14b. The lines of *C2* are rearranged according to the variable order by moving the top line to the bottom while still keeping the same gates' connections. The final circuit C3 is shown on Fig. 4.14c.

The QMDDsyn framework verifies that circuit C3 is equivalent to C1 as they are both represented by the same QMDD in view of Theorem 3.1. This example illustrates an overall size reduction in terms of gates of 7/12 (42%) as well as size reduction in terms of quantum cost of 11/30 (53%). In this example there was no need to produce C3 since we selected the unidirectional MMD algorithm. In the following section, we exclusively use the better bidirectional MMD algorithm for the lexicographic synthesis step.

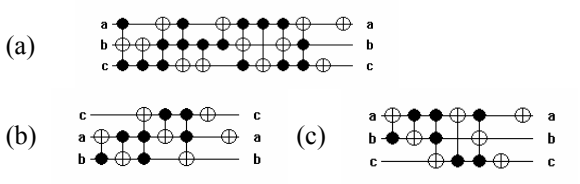


Figure 4.14. Minimization of the Circuit of Example 4.2

4.4.3. The QMDDsyn Framework Synthesis Results

Table 4.6 illustrates the preliminary experimental results obtained by the QMDDsyn framework [FTM08c]. We use the complete specification from the published benchmark circuits [Masl05] as well as the specification obtained from the random circuits in Table 4.4. We should note that we merely convert the benchmark circuits into complete function specification regardless of the garbage and ancilla assignments that are included with the benchmark files. Therefore, the lexicographic algorithm cannot be expected to produce smaller size then the benchmark circuits. The benchmark circuits were made with significant don't-care assignment that allowed their

designers to minimize the number of gates. When we converted those files to a complete specification, there are no don't-care assignment in these specification. However, this does not affect the ability to compare the lexicographic synthesis results as we use complete specifications for both processes (with and without variable re-order).

For each specification, we list the number of variables in column 2 and the QMDD size in non-terminal nodes in column 3. The minimization results are shown in columns 4 to 5. Column 6 lists the variable order obtained during the QMDD minimization. Column 7 shows the quantum cost of the baseline circuit *C1*. Column 8 displays the quantum cost of circuit *C2* obtained with the variable reorder of column 6. Column 9 lists the quantum cost of circuit *C3* obtained with the *reversed* variable reorder. In column 10 there is a comparison quantum cost for circuit *C4* that is synthesized with a random variable order.

The quantum cost improvement obtained when using the QMDD variable reorder is depicted in column 11. Negative percentage indicates that the QMDDsyn obtain a smaller circuit with the variable reorder. We should note that the smaller of *C2* and *C3* is compared to the baseline *C1*, as discussed in Section 4.4.2.

The results show significant overall improvement in the quantum cost using the approach of QMDDsyn. There is generally a good correlation between the level of QMDD size reduction and the improvement in the quantum cost.

As is observed in Table 2, the QMDD size grows rapidly from the number of variables (for trivial circuit) to a certain fluctuating range. Smaller QMDD size

indicates less permutation by the specification. It seems from our results that specifications with smaller QMDD sizes (i.e. less permutation) are likely to produce better quantum cost reduction on the QMDDsyn framework.

Table 4.6. QMDDsyn Synthesis Results

Specification Source	Vars	QMDD nodes	Minim nodes	QMDD reduction (%)	Variable reorder	Circuit C1 (w/o reorder) Quantum cost	Circuit C2 (with reorder) Quantum cost	Circuit C3 (reversed reorder) Quantum cost	Circuit C4 (random reorder) Quantum cost	Reordered Quantum cost change (%)
4_49tc1	4	21	20	4.76%	0 2 1 3	135	126	169	199	-6.66%
hwb4tc	4	22	20	9.09%	0 2 1 3	96	87	87	88	-9.35%
hwb5tc	5	47	38	19.15%	0 2 3 1 4	488	487	518	536	0%
rand5v1	5	36	32	11.11%	2 0 3 1 4	441	312	287	200	-35.60%
rand5v2	5	44	34	22.73%	2 1 0 3 4	567	370	409	464	-34.74%
rand5v3	5	41	37	9.76%	2 3 0 1 4	421	475	360	466	-14.48%
5mod5	6	28	15	46.43%	5 3 1 4 0 2	185	185	185	185	0%
2of5d1	6	56	49	12.50%	3 1 0 4 5 2	525	481	507	766	-8.38%
rand6v1	6	64	42	34.38%	3 1 0 4 2 5	1114	619	661	845	-44.43%
rand6v2	6	78	67	14.10%	3 1 0 2 4 5	1518	1643	1460	1941	-3.82%
rand6v3	6	82	72	12.00%	2 1 0 3 4 5	2218	1753	2121	1951	-20.96%
rand6v4	6	77	69	10.39%	2 1 0 3 4 5	1668	1584	1767	1861	-5.03%
rd53d1	7	26	21	19.23%	2 1 0 3 4 5 6	176	188	120	139	-31.81%
rand7v1	7	95	56	41.00%	2 4 3 0 6 1 5	2997	3109	1744	2413	-41.81%
rand7v2	7	123	75	39.02%	3 4 2 0 5 16	6902	6159	4170	4987	-39.58%
rand7v4	7	141	100	29.08%	4 2 0 3 5 1 6	5028	4640	7099	6999	-7.71%
rand7v5	7	134	111	17.16%	3 2 0 5 4 1 6	6390	5770	7162	7558	-9.70%
rand7v6	7	176	147	16.48%	2 3 0 4 1 5 6	7075	7792	7609	6981	+7.54%
rand8v1	8	167	60	64.07%	2 0 5 4 6 1 3 7	2933	1136	8081	8510	-61.26%
rand8v2	8	242	150	38.02%	2 5 0 3 4 1 6 7	22968	19227	23270	22388	-16.29%
rand8v3	8	313	210	32.91%	3 5 2 0 1 6 4 7	20951	17094	13585	23105	-35.18%
rand9v1	9	114	59	48.25%	0 4 2 3 7 5 1 8 6	14242	11680	4930	22970	-65.38%
rand9v2	9	242	186	23.14%	2 0 3 6 4 7 5 1 8	55200	56569	29422	55512	-46.69%
rand9v3	9	368	337	8.42%	5 2 0 3 4 7 6 1 8	63009	55924	62431	60341	-11.24%

Since this approach is based on a heuristic observation, it does not work in all cases, and for several specifications (“hwb4tc”, “rand5v1”, “rand7v6”), the selection of a random variable order obtained a better quantum cost reduction. There is an 8% quantum cost deterioration for “rand7v6”. The file “5mod5tc” resists any size change

apparently due to its unique construction with 5 garbage outputs and only one controlled output.

The run times for the QMDDsyn framework are not shown in Table 4.6 since they are not substantially different than the "MMD" run times, except that the synthesis process is repeated two times to evaluate *C2* and *C3*.

Comparing our results to those obtained by Fujita et al. with classical irreversible logic circuits, it seems that heuristics for reversible logic are likely to have less dramatic improvements due to the inherent maximal connectivity associated with each output.

CHAPTER 5

TESTING AND VERIFICATION OF QUANTUM CIRCUITS

Relatively little work has been accomplished in the area of reversible logic testing, and in particular, quantum circuit testing [FTN07]. Most verification work so far has been limited to symbolic circuit equivalence checking. This is not surprising since quantum circuits have not yet been implemented in a meaningful way. However, theoreticians have already created an interesting foundation. In this chapter we consider the differences between testing of general reversible circuits and testing future QC. Since QCA is different from QC we briefly discuss several QCA testing issues. We conclude this chapter with our investigation of partially redundant logic detection in both reversible and irreversible logic circuits. This work relates to verification and *design for test* (DFT) aspects of QC.

5.1. Testing of Reversible Logic

Agrawal investigated the probability of fault detection in logic circuits using the principle of maximum entropy from general thermodynamics [Agra81]. Based on the principle of maximum entropy, he demonstrated that one can obtain a high level of confidence in the test if the output entropy of the system is maximized.

Definition 5.1: A maximized output is achieved when the output information in changed bits/pattern equals the input information.

Let the inputs of a circuit with n inputs and m outputs be connected to an information source (or a pattern generator). Let the input information be H_i changed bits/pattern and the output information be H_o changed bits/pattern. It is easy to show that $H_o \leq H_i$. The limiting case with $H_o = H_i$ is referred to as maximized output. It is easy to show that $n=m$ is a necessary condition for maximized output.

Since an $n \times n$ reversible logic circuits constitute a bijective relation between the inputs and outputs, they must have maximized outputs. Thus, based on Agrawal analysis it is should be easier to detect faults in reversible logic circuits.

Patel et al. [PHM04] showed that reversible circuits require fewer test vectors for multiple fault detection based on the stuck-at model as compared to classical circuits. They found that in reversible logic, the number of test vectors grows logarithmically in both the number of inputs/output and the number of gates.

Moreover, they demonstrated a critical and unique feature for reversible logic testing – *any test-set that can detect all single faults in a reversible logic circuit will also detect all multiple stuck-at faults*. This feat cannot be achieved in irreversible logic where multiple stuck-at faults are significantly more difficult to cover than single stuck-at faults. They further developed practical test-set generation algorithms.

5.2. Towards a Unified QC Fault Model

In general, only a few of the many fault models of conventional logic can be immediately extended to quantum logic. For example, quantum logic has not reached the real technological level that allows the use of **behavioral faults** that deal with the highest level of design [VMO06]. Similarly, the concept of delay faults and defect faults as applied to conventional logic currently do no analog within the QL domain [BA01]. In contrast, logic-level fault models that deal with a netlist of quantum gate interconnections can be readily adapted to QC. Logic-level faults are commonly referred to as *register transfer level* RTL faults**.**

The most popular RTL fault model is the stuck-at-fault. In conventional logic, a "stuck-at" fault is modeled by assigning a fixed logic value to net of the circuit under test. Perkowski et al. [PBL05] proposed two fault models, the first for binary permutative QC, and the second for general quantum gates. In the first model, they define multiple-valued states for the wires, including $\{|0\rangle, |1\rangle, |V_0\rangle$, and $|V_1\rangle\}$ which are observed to determine if any state of the wire is stuck at a given value. The main difference between their two fault models is the need for *probabilistic* tests whenever the outputs use complex values.

Conventional logic often defines stuck-at faults using particular gate fault models [MZ00, Micz03]. Polian et al. promoted the use of a gate fault model for QC over the stuck-at model [PFBH05]. They derived a family of logical fault models for

reversible circuits composed of *k-input controlled-NOT* (*k*-CNOT) gates. Their model relates to the single missing-gate fault model (MGF). Our work disclosed in Section 5.6 is related to this MGF-like fault model.

In conventional logic, the stuck-open and stuck-short faults are used to model specific problems for a single transistor within the construction of a full CMOS gate. Specific device faults similar to these have been discussed for primitive QC devices like Si n*uclear magnetic resonance* (NMR), solution NMR, quantum-dot charge, scalable ion trap, Josephson junction charge, and the Kane solid-state NMR [VMO06]. However, they have been formally developed in a proper fault model formalism as of yet.

An interesting and unique aspect of quantum circuit testing is the need to distinguish between probabilistic tests and deterministic tests. Clearly, the deterministic tests required multiple measurements, preferably in different bases.

Leakage faults, which are very important in conventional digital logic, have been treated sparsely in the QC literature [Pres97]. We assume that each qubit operates in complete isolation within its two dimensional Hilbert space $\mathcal{H}_2$ so that in response to an error, the qubit can either become entangled with the environment or rotated unpredictably within the two dimensional space. The leakage in quantum circuits occurs when the qubit leaks out of this two-dimensional Hilbert space into a larger space. The testing of QC leakage faults is a difficult open issue. We note that faults manifested as rotations can be modeled as the unintentional insertion of a single qubit Pauli rotation gate and that faults causing only phase changes in the qubit may be

ignored since the only information carrying portion of the qubit is the direction it points to in the Hilbert space.

It is possible that more than one fault (*single-fault* model) can materialize at the same time, thus leading to the *multiple-fault* model [BA00, MZ00, Micz03].

The test generation problem for irreversible, classical circuits was shown to be an *intractable* problem by Ibarra and Sahni as early as 1975 [IS75]. Agrawal noted in the early 80s that fault detection is improved when the output content of the tested circuit is maximized [Agra81]. Since reversible logic circuits have *maximized outputs*, it should be easier to detect faults in QC reversible logic circuits as compared to conventional logic.

Biamonte and Perkowski demonstrated that quantum circuits can only use some of the test methods developed for reversible circuits [BP04]. In [Haye05] it is stated that QC testing may be more difficult than conventional digital testing.

Due to the no-cloning theorem of QC, quantum computers can be modeled as physically moving the qubits from the storage area to the quantum processing area and back to storage during every computation cycle [CLM07]. A suitable QC transmission channel fault model is adapted from the conventional binary symmetric channel fault model. It assumes the same probability p for a qubit to flip from $|0\rangle$ to $|1\rangle$ or from $|1\rangle$ to $|0\rangle$ [NC00, Niel98]. The binary symmetric channel fault model is shown in Fig. 5.1.

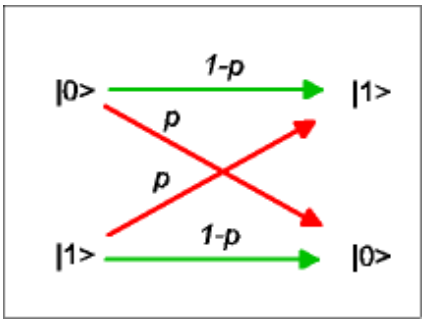


Figure 5.1. The Binary Symmetry Channel Fault Model

To survive the challenges of transportation and decoherence, QC must rely extensively on *quantum error correction codes* QECC for proper operation.

5.3. Technology Specific Defect Characterization for QCA Testing

The individual QCA cell by itself is reversible in view of its quantum nature. However, QCA circuits are not reversible since the QCA cells are subjected to electrical or magnetic fields. This results directly from the first two postulates of quantum mechanics that requires a closed system for the reversibility of the quantum state time evolution, as discussed in Chapter 2. Therefore, QCA testing is different than reversible QC testing. QCA have been successfully implemented, thus designers are concerned with real defect issues similar to those encountered in CMOS testing.

Tahoori et al. [TJML04] compared QCA testing with conventional CMOS-based designs. They developed a testing technique that requires only a constant number of

test vectors to achieve 100% fault coverage. They further explored a design-for-test technique which generates a reduced test set with excellent coverage.

Gupta et al. [GJL06] discovered that the stuck-at fault test set of a QCA circuit is not guaranteed to detect all implementation related defects. They generate additional test vectors to target bridging faults in QCA interconnects. They include a comprehensive analysis of their framework on MCNC benchmark circuits that use majority gates as primitives.

5.4. Partially Redundant Reversible and Irreversible Logic Detection

A gate or segment of logic is redundant if it can be removed from a circuit without affecting the functionality of the circuit. A gate or segment of logic is partially redundant if it can be modified without affecting the functionality of the circuit. Detection of partially redundant logic within any design, reversible or irreversible, has ramifications for logic synthesis, for design verification, and for *design for test* (*DFT*) issues. In this section we contrast the ability of reversible logic and irreversible classical logic verification tools to detect partial redundancies by circuit modifications [FTM08]. Generally, one would expect that a given circuit would not be equivalent with a version of itself having undergone some structural modification. And yet, it is well known that for irreversible (classical) logic circuits, partially redundant logic or internal don't-care cones allow for such structural modifications to be undetected since such replacements do not affect overall functionality due to internal don't-care

conditions. We were motivated to investigate whether similar partially redundant logic instances are likely to be found with emerging quantum and reversible logic technologies.

In logic synthesis, partially redundant logic may use valuable circuit resources and its removal can provide more efficient circuits. While in irreversible logic, don't-cares are often exploited by synthesis tools to reduce the circuit time/size complexity. This partially redundant logic can also inhibit the effectiveness of structural verification of the design, or even mask some potential design flaws. For DFT, partially redundant logic can create numerous observability and controllability obstacles.

While several researchers have recently dealt with the testability issues of reversible logic, to the best of our knowledge, this work is the first investigation of reversible logic verification for structural modification detection using equivalence checking. To facilitate our research, we have developed similar symbolic equivalence checking programs for irreversible classical logic (CMBtest) and reversible logic (DQMMceq) that automatically compare a benchmark file with a controlled modified version. This topic is very important with respect to the development of synthesis optimization tools, test–set generators, and the understanding of functional equivalence tools.

5.4.1. Partially Redundant Logic in Irreversible Logic

Partial redundancy in irreversible logic arises from don't-care conditions internal to the circuit. Simulation based functional validation uses a set of vectors that are applied to the design and compared to the expected results. Since a design

containing partially redundant logic may respond correctly to a set of test vectors, various approaches have been suggested that use verification tools to pinpoint and remove such logic. Ratchev et al. use the technique of forced error detection which is based on allowing the synthesis tool to invert one randomly selected logic signal in the netlist [RHBA03]. They found that while not all the single injected errors are detected, the probability of error detection increases with the number of such insertions. Huang et al. developed a tool that can detect don't-care logic while performing static property checking using ATPG and binary decision diagram (BDD) techniques [HYTC00]. They demonstrated that their don't-care cases result from user-defined don't-care assignments, over/under flow in variable index assignments of an array construct, and the default section of a Verilog case statement. While don't-care assignments often result in reduced circuit size during synthesis, it may create redundant logic that cannot be tested and wastes circuit resources.

5.4.2. Partially Redundant Logic in Reversible Logic

Definition 5.2 The *trivial identity gate* is a reversible gate that implements the identity transformation matrix I (a diagonal matrix with all non-zero entries equal 1). Any reversible gate with a transformation matrix that differs from the identity matrix is a *non-trivial* gate.

Identity gates are trivial in the sense that they do not change the circuit transformation matrix and can be ignored in a functional sense. We consider only non-trivial gates when discussing reversible cascades.

5.4.3. Circuit Modification Strategy

We have developed two tools for detecting partially redundant logic. The "QMDDceq" for reversible logic uses a QMDD-based simulation tool, while the "CMBtest" tool for irreversible logic is based on the well-known CUDD package [Some95].

Both tools are used to perform repeated equivalence checking between a benchmark circuit and a copy of the circuit with controlled circuit modifications. Exhaustive structural tests that scan all the gates of the benchmark circuit are possible only for small circuits. Larger circuits require random selection approaches. Fig. 5.2 shows the operational flow of our tools.

The gate modifications are quite different for irreversible and reversible logic gates and will be described separately.

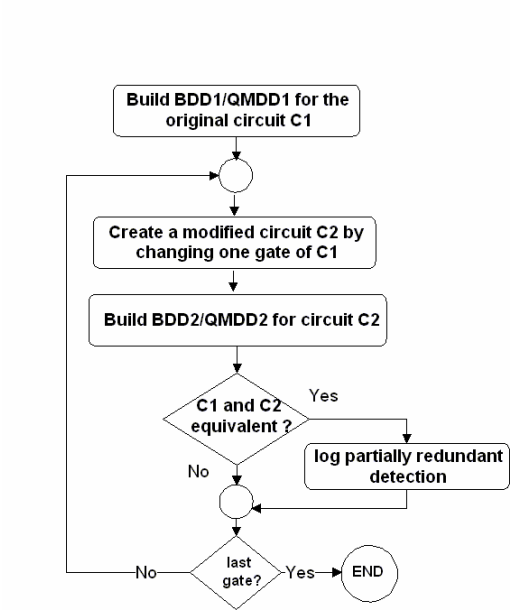


Figure 5.2. Detection of Partially Redundant Logic

5.4.4. Gate Modification for Irreversible Logic

The ISCAS85 combinational benchmark circuits are comprised of a non-homogeneous mixture of AND, NAND, OR, NOR, XOR, XNOR, BUFF, and NOT gates of various input sizes. The program first constructs BDD representing the benchmark circuit using the CUDD package. Next, a copy of the benchmark circuit with only one of the gates modified in accordance with the user selected replacement categories of Table 5.1 is created.

Table 5.1. Gate Replacements for Irreversible Logic

Replacement category	Allowed gate type changes
Negation	AND↔NAND; OR↔NOR; XOR↔XNOR.
Moderate1	AND→NOR; NAND→XOR;OR→NAND; NOR→XNOR; XOR→OR; XNOR→AND.
Moderate2	AND→OR; NAND→NOR; OR→XOR; NOR→AND; XOR→NAND; NOR→NOR.

The number of inputs remains fixed when a gate type is replaced. Clearly, the negation replacements are the most severe and negated gates tend to be more easily detected by the equivalence checking tools. Therefore, such hidden replacements in the benchmark circuits cannot exist unless they are buried inside a partially redundant logic cone.

5.4.5. Gate Replacement for Reversible Logic

We now prove several lemmas that illustrate the major differences between reversible and irreversible circuits.

Lemma 5.1: Two reversible circuits (cascades) that are different by exactly one (non-trivial) gate cannot be equivalent.

Proof: Let us assume that the two cascades C_1 and C_2 each consist of n gates, represented by the transformation matrices $G_1, G_2, \ldots, G_k, .., G_n$ and $G_1, G_2, \ldots, G'_k, .., G_n$ respectively. Let $G_k \neq G'_k$ be the only different gates in these circuits. Then by (2.34),

$$C_1 = G_n \times G_{n-1} \times \ldots \times G_{k+1} \times G_k \times G_{k-1} \times \ldots \times G_2 \times G_1 \tag{5.3}$$

and

$$C_2 = G_n \times G_{n-1} \times \ldots \times G_{k+1} \times G'_k \times G_{k-1} \times \ldots \times G_2 \times G_1 \tag{5.4}$$

and the inequality $C1 \neq C2$ follows from the associative rule of matrix multiplication. □

Lemma 5.1 immediately illustrates a major difference between reversible logic and irreversible logic since modifying a single gate is guaranteed to be detected. This requires the QMDDceq to modify two or more gates during each iteration to create partially redundant logic.

Lemma 5.2: *The removal of a single gate from one cascade of a pair of identical reversible cascades ensures their non-equivalence.*

Proof: The proof follows directly from Lemma 5.1 by substituting the removed gate with the identity gate. □

Lemma 5.3: If a pair of equivalent cascades C_1 and C_2 remain equivalent after the removal of a contiguous set of l gates from C_2, then the removed set comprises a cascade representing the identity matrix.

Proof: It is easy to see that after removing l gates from C_2, the remaining equivalence condition $C1=C2$, namely

$$C_1 = G_n \times \ldots \times G_k \times \ldots \times G_{k-l+1} \times \ldots \times G_1 = G_n \times \ldots \times G_{k+1} \times G_{k-l} \times \ldots \times G_1 = C_2 \tag{5.5}$$

can be satisfied only if

$$G_k \times G_{k-1} \times \ldots \times G_{k-l+1} = \mathbf{I} \tag{5.6}$$

due to the properties of matrix multiplication. □

The synthesis tools for reversible logic often employ template matching for circuit reductions [IKY02, MMD03]. Obviously, the condition of Lemma 5.3 should be easily detected and removed from the benchmark circuits by the synthesis tool. Theorem 5.1 can now be proven that sets the limit for partially redundant logic in reversible circuits.

Theorem 5.1: A reversible logic cascade C of n non-trivial gates may contain redundant logic comprising 2 to n gates.

Proof: By Lemma 5.1, redundant logic cannot be comprised of a single non-trivial gate. By Lemma 5.3, we can provide an example of a redundant logic section of arbitrary size. □

It is the prime motivation of this portion of the research to find hidden replacements in reversible logic when non-contiguous sets of gates are modified.

We now consider various types of gate modifications for reversible logic circuits. A severe gate modification is achieved when we replace the type of gate altogether (e.g. TOF(x_0,x_1;x_2) ↔ V^+(x_0;x_2)). The deletion of a gate is essentially the replacement of the current gate type with the identity gate. A more moderate gate modification is achieved when changing the structure of a controlled gate with multiple lines like the Toffoli gate. We must take into account that changing the order of control lines does not change the gate. For example,

$$\text{TOF}(a,b,c;d) = \text{TOF}(a,c,b;d) \tag{5.7}$$

since only the order of the control lines has been changed. On the other hand,

$$\text{TOF(a,b,c;d)} \neq \text{TOF(a,d,c;b)} \tag{5.8}$$

represents a change since the target line has been modified, as has one of the control lines. Also,

$$\text{TOF(a,b,c;d)} \neq \text{TOF(a,b;d)} \neq \text{TOF(a,b,c,e;d)} \tag{5.9}$$

since we are changing the number of control lines.

5.4.6. Irreversible Benchmark Circuits Results

We have run the various gate replacements on a number of ISCAS85 benchmark circuits as shown in Table 5.2. The table lists how many hidden replacements were detected using the Negation, Moderate1 and Moderate2 replacement rules. We found that only one negation replacement is hidden in the benchmark circuit C2670.

On the other hand, there are many hidden replacements for the moderate gate replacements. Some of these may indicate the presence of partially redundant logic. Other hidden replacements exist because the benchmark circuits implement a large

number of don't-care logic cones. These don't-cares cones were conveniently implemented in NAND and OR gates, but, as is apparent in the results, XOR and XNOR would have performed as well.

Files with the suffix "nr" denote a later revision of the previously published benchmark circuits with their redundancy removed. We note that the hidden negation replacement in C2670 was identified and fixed in C2670nr. We can see that the newer versions are smaller in size and that the overall number of hidden replacements is generally reduced. In some cases, the percent reduction in hidden replacements is larger than the benchmark circuit size reduction.

Table 5.2. Hidden Replacements in Irreversible Logic

Benchmark circuit	gates/levels	Using Negation	Using Moderate1	Using Moderate2
C432	159 / 7	0	24	18
C432nr	156 / 7	0	24	18
C880	382/ 12	0	57	20
C1355	545/ 15	0	416	2
C1355nr	545/ 15	0	416	2
C1908	879 / 17	0	327	0
C1908nr	877 / 17	0	326	0
C2670	1192 / 15	1	271	84
C2670nr	960 / 13	0	201	47
C3540	1668 / 15	0	292	117
C3540nr	1691 / 15	0	252	67
C5315	2306 / 25	0	475	184
C5315nr	2297 / 25	0	472	164
C7552	3511 / 24	0	1059	244
C7552nr	3396 / 24	0	955	186

5.4.7. Reversible Logic Benchmark Circuits Results

Table 5.3 shows our results for a number of the reversible logic benchmark circuits from Maslov's website [Masl05]. These circuits include quantum as well as classical reversible logic benchmark circuits of generally "well synthesized" circuits. As expected, all the gate changes were promptly detected by the QMDDceq program. For larger files, we were able to run only random gate replacement selections, as is shown in the 5th column. For the smaller benchmark circuits, we were able to run exhaustive gate modifications for sets of 2 to 6 gates as shown in the 6th column. As illustrated in the experimental results of [MFT07b], the time to parse the netlist into a QMDD data structure representation depends on the circuit's structure. Therefore, the exhaustive test for some smaller benchmark circuits requires a prohibitively long amount of runtime.

Although no partially redundant logic was detected in these benchmark circuits, we should stress that the random gate selections tackle only a small portion of the search space. We cannot guarantee that some other gate selection changes will not reveal partially redundant logic in these benchmark circuits. Yet it is clear from comparing Table 5.3 and Table 5.2 that redundant logic in reversible circuits is less likely to occur than in irreversible classical logic.

Table 5.3. No Hidden Replacements in Reversible Logic

name	type	lines	gates	random	exhaustive
hwb4	qc	4	21	none	2,3,4,5,6
c410184	qc	14	74	none	2
c2	qc	35	305	none	
ham3	nct	3	5	none	2,3,4
c3-17	nct	3	6	none	2,3,4,5
hwb4	nct	4	11	none	2,3,4,5,6
5mod5	nct	6	17	none	2,3,4,5
hwb7	nct	7	289	none	
hwb8	nct	8	614	none	
hwb9	nct	9	1541	none	
6symd2	nct	10	20	none	2,3
hwb10	nct	10	3595	none	
hwb11	nct	11	9314	none	
9symd2	nct	12	28	none	2
c410184	nct	14	46	none	2
rd84d1	nct	15	28	none	
ham15	nct	15	132	none	
cyc17-3	nct	20	48	none	2
c2	nct	35	116	none	

5.4.8. Results with Randomly Generated Reversible Circuits

In Lemma 5.3 we have shown that a trivial set of reversible gates implementing the identity transformation matrix can create an arbitrarily large redundant logic sequence. We were curious to determine if potential redundant logic in reversible circuits can appear in the form of non-continuous gates. For that purpose, we have generated several random reversible logic circuits without using template matching (as

described in [MMD03]) to optimize our design. Fig. 5.3 demonstrates one of our findings with a circuit called "random4".

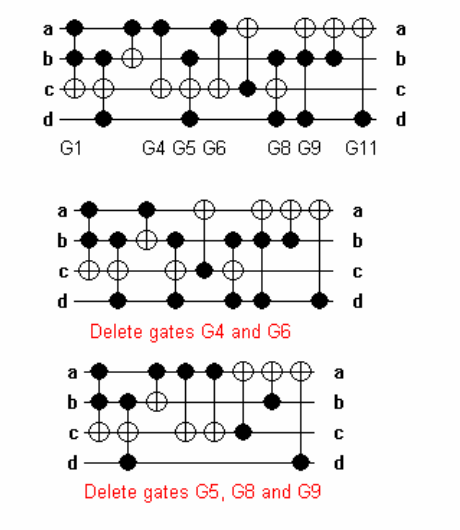


Figure 5.3. Results with File "random4"

This reversible circuit (shown in Fig. 5.3 using Maslov's RCviewer tool [Masl05]) of 11 gates captures hidden replacements using the exhaustive gate deletion. For exhaustive two gate deletion, we found several hidden replacements such as the pair {G4, G6}. For exhaustive three gate deletion, we found a hidden replacement by deleting gates {G5, G8, G9}. We note that the template matching techniques described in Section 4.3.3 were not used in the process shown in Fig 5.3. It is obvious that in the latter case, the 4th and 5th remaining gates would also be deleted if template matching is employed (or with exhaustive deletion of five gates).

Although the theoretical discussion indicates that redundant logic may appear in reversible logic, the experimental results demonstrate that it is very uncommon, as

reversible logic cascades are maximally connected. In contrast, irreversible logic tends to contain don't-care cones that are more prone to result in redundant logic.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

This dissertation proposes several novel CAD tools for emerging quantum technologies. We survey the fundamentals of reversible logic, quantum computing and quantum computation and described the state of the art in simulation, synthesis, and testing tools. We combined in this research two emerging quantum technologies - QCA and general reversible quantum circuits. While the main difference between QCA and reversible QC is that QCA is not reversible, both technologies must deal with the probabilistic nature of quantum effects at the nano-scale.

6.1. Conclusions and Main Contributions

We have developed a sift-like minimization as well as structure metric based minimization techniques for the recently introduced *quantum multi-valued decision diagram* (QMDD), thus making it more efficient for use in CAD applications. Without such minimization, decision diagrams tend to grow exponentially, quickly exceeding the computer's memory resources. Our techniques, employed in the *QMDDmet* CAD tool, achieved size minimization of 20-93% on most benchmark circuits [MFT07, MFT07b, FTM08b].

We have used the enhanced QMDD to efficiently simulate quantum circuits and quantum vectors [GTFM07]. The tools further provide symbolic equivalence checking between quantum circuits.

We have investigated the problem of synthesizing QCA circuits. We developed the *QCAexor* CAD tool that demonstrated significant improvements for many benchmark circuits [FT07]. We investigated reversible logic synthesis using two approaches. We investigated a virtual implementation of reversible circuits that use a direct translation of complex binary functions into circuits composed of Fredkin reversible gates [FTN06]. While this approach does not result in cascades of gates, it provided some interesting insight into the area and time complexities of complex circuits. In investigation reversible logic synthesis techniques, we explored the synthesis of reversible cascades of gates. In this approach, we explored the use of QMDD minimization in the *QMDDsyn* CAD tool in lexicographical logic synthesis of quantum circuits, utilizing its ability to determine a good variable order [FTM08c].

The *QMDDceq* CAD tool investigates partially redundant reversible logic [FTM08]. Detection of partially redundant logic within any design, reversible or irreversible, has ramifications for logic synthesis, for design verification, and for *design for test* (DFT) issues. We have experimentally established that reversible logic is less likely to exhibit redundant logic than irreversible logic, since reversible logic cascades are maximally connected. Yet our theoretical analysis shows that redundant logic may still appear in reversible logic.

Overall, this dissertation contributed to the main disciplines of quantum computing CAD tools, including simulation, synthesis, and testing [FTM08e].

6.2. Areas of Future Research

The nascent nature of QC virtually guarantees that every topic discussed in this dissertation is likely to require significantly more research and refinement. In addition to the methods developed in this research, these research topics need a constant flow of new techniques in order to reach maturity. In the following subsections we describe several ideas we plan to investigate in the near future.

6.2.1 Limitations of Quantum Circuit Simulation on Classical Computers

In chapter 3 we have shown how the QMDD package can simulate quantum circuits on classical computers. We also surveyed recently proposed QC simulators by other researchers. And yet, we have noted that quantum computing emerged when Feynman observed that some quantum mechanics phenomenon cannot be accurately and fully simulated on a classical computer. Clearly, there are some fundamental limitations of classical computer-based simulators of QC which we have touched upon in our discussion of skipped variables in Section 3.2. We see a great prospect for future research along similar lines that try to unravel such fundamental limitations in the simulation of quantum effects.

6.2.2 Other Quantum Circuit Simulators

We plan to explore alternative methods for QC simulation that are based on decision diagrams or programming languages. The motivation for this research is to achieve better synthesis tool and overcome fundamental simulation limitations with a different tradeoffs than those provided by our current simulation package.

6.2.3 Direct Synthesis of Quantum Circuits from QMDD

In Section 4.4 we have explored the use of QMDD variable reordering minimization to improve the lexicographic synthesis of reversible circuits. A more profound challenge for our future research is to synthesize the circuit directly from the minimized QMDD data structure. The goal is to create an initial rough, non-optimal but fast representation of the circuit in QMDD, and then translate the minimized QMDD directly into a near-optimal circuit.

In classical BDDs the direct translation into a circuit can be readily done by replacing each vertex with a 2:1 multiplexer whose selector input takes the value of the variable represented by the vertex. Since QMDD actually decompose the overall transformation matrix of the circuit, there is no equivalent quantum circuit element that can be used instead like BDD's multiplexer for direct translation of the QMDD into a quantum circuit. An approach that works directly on the QMDD data structure by iteratively converting it into the representation of the identity matrix is one that warrants further investigation. The iterative steps in this process will constitute the required circuit synthesis technique.

We plan to further investigate and pursue improvements of the other QC synthesis methods described in Section 4.3. In particular we are interested in template based synthesis (Section 4.3.3) as well as symmetry based synthesis (Section 4.3.5).

6.2.4 Testing of Quantum Circuits

We are interested in further refinement of the unified fault model for quantum circuit testing. In particular, we wish to investigate various models that distinguish between probabilistic and deterministic quantum testing. Additional research work may be required to further integrate the *QMDDceq* tool into a future *design for test* (DFT) paradigm for quantum circuits as well as for fault tolerant quantum circuit design.

The QMDD simulation package can be used in QC testing for *automatic random pattern generation* (ATPG). This can be accomplished by simulating multiple vectors with the target circuit and analyzing the resulting fault coverage. We are also interested in future development of the *built-in-self-test* (BIST) paradigm for quantum circuits [FNT07].

REFERENCES

[AP05] A. Abdollahi and M. Pedram, "Efficient synthesis of quantum logic circuits by rotation-based quantum operators and unitary functional bi-decomposition", In *IWSL'05* on CD ROM 2005.

[AGK01] F. Ablayev, A. Gainutdinova, and M. Karpinski, "On the computational power of quantum branching programs", In *Proc. FCT 2001, Lecture Notes in Computer Science* 2138, 59–70, 2001.

[Agra81] V.D. Agrawal, "An information theoretic approach to digital fault testing", In *IEEE Trans. Comp.* Vol. 30, pp. 582-587, Aug. 1981.

[AJ04] A. Agrawal and N. K. Jha, "Synthesis of reversible logic", In *DATE'04,* 2004 on CD ROM.

[AB97] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error", In *Proc. of 29th ACM Symposium on Theory of computing* (*STOC'97*), May 1997.

[Aker78] S.B. Akers, "Binary Decision Diagrams", In *IEEE Trans. on Comp.*, C-27(6), June 1978.

[Alra04] A.N. Al-Rabadi, "Reversible fast permutation transforms for quantum circuit synthesis", In *ISMVL'04*, pp 81-86, May 2004.

[Alra04b] A.N. Al-Rabadi, *Reversible Logic Synthesis*, Spinger-Verlag, 2004.

[Altera] Altera Corporation, *Quartus II Software.* Available on-line: http://www.altera.com/products/software/products/quartus2/qts-index.html.

[AOSL98] I. Amlani, A.O. Orlov, G.L. Snider and C.S. Lent, "Demonstration of a functional quantum-dot cellular automata cell", *J. Vac. Sci. Technol.*, B, 16, 1998, pp. 3795-3799.

[BFGH+93] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic Decision Diagrams and their Applications", In *ICCAD'93*, 1993.

[BBC+95] A. Barenco, C. H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation", In *Phys. Rev. A*. Vol. 52, No. 5, pp. 3457-3467, 1995.

[BZBB94] M. Beck, A. Zeilinger, H.J. Bernstein, and P. Bertani, Experimental realization of any discrete unitary operator. In *Phys. Rev. Lett*., 73(1), pp. 58-61, 1994.

[BCS04] G. Benenti, G. Casati, and G. Strini, *Principle of Quantum Computation and Information: Volume I: Basic Concepts*, World Scientific, 2004.

[Benn73] C.H. Bennett, "Logical reversibility of computation", *IBM, J. Res. Dev.*, Vol. 17, No. 6, pp. 525-532, 1973.

[BBC+93] C.H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. Wootters, "Teleporting an unknown quantum state via dual classical and EPR channels", In *Phys. Rev. Lett.,* Vol. 70, pp. 1895-1899, 1993.

[Bell64] J. S. Bell, "On the Einstein Podolsky Rosen Paradox", In *Physics*, 1, 1964, pp. 195-200.

[BP04] J. Biamonte, and M. Perkowski, Testing a Quantum Computer, In *KIAS-KAIST 2004 Workshop on Quantum Information Science,* 2004. Quantum Phys. abstract, quant-ph/0409023.

[Bohm01] A. Bohm, *Quantum Mechanics Foundations and Applications*, 3rd Ed., Springer, 2001.

[BW96] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-Complete", In *IEEE Trans. on Computers*, Vol. 45, No. 9, pp. 993-1001, 1996.

[BHMS84] R.K. Brayton, G.D. Hachtel, C.T. McMullen and A.L.M. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Botson, 1984.

[BK82] R. P. Brent and H. T. Kung, "Regular layout for parallel adders", In *IEEE Trans. Comp*., Vol. C 31, no. 3, pp. 260-264, 1982.

[BTS+02] J.W. Bruce, M.A. Thornton, L. Shivakumaraiah, P.S. Kokate, and X. Li, "Efficient adder circuits based on a conservative reversible logic gate", In *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 83-88, April 2002.

[Brya86] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation", In *IEEE Trans. on Comp.*, Vol. C-35, No. 8, pp. 677-691, August 1986.

[BA01] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, Mixed-Signal VLSI Circuits*, Kluwer Academic 2nd Edition 2001.

[Cabe04] A. Cabello, "Bibliographic guide to the foundations of quantum mechanics and quantum information", available online at: arXiv:quant-ph/00120.

[CLM07] E. Chi, S. A. Lyon, M. Martonosi, "Core fusion and quantum: Tailoring quantum architectures to implementation style: a quantum computer for mobile and persistent qubits", In *ISCA '07*, pp. 198-209, June 2007.

[Cybe01] G. Cybenko, "Reducing quantum computations to elementary unitary operations", In *Computing in Science and Engineering Magazine*, pp. 27-32, March/April 2001.

[DRS02] A. De Vos, B. Raa and L. Storme, "Generating the group of reversible logic gates", In *J. Phys. A:Math. Gen.* 35(2002) pp. 7063-7078, 2002.

[Deut85] D. Deutsch, "Quantum theory, the Church-Turing Principle and the universal quantum computer", In *Proc. R. Soc. Lond. A*, 425:73, 1989.

[Deut89] D. Deutsch, "Quantum computational networks", In *Proc. R. Soc. Lond. A*, 400:97, 1985.

[DJ92] D. Deutsch and R Jozsa, "Rapid solution of problems by quantum computation", In *Proc. of the Royal Society of London Series A,* A439, pp. 553-558, 1992.

[Dira58] P. A. M. Dirac, *The principles of quantum mechanics,* 4th ed., Oxford University Press, Oxford, 1958.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?", In *Phys. Rev*, Vol. 47, pp. 777-780, 1935.

[EL04] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, Morgan Kaufmann, 2004.

[FK00] B.J. Falkowski and S. Kannurao, "Spectral theory of disjunctive decomposition for balanced boolean functions", *13th Int'l Conf on VLSI Design,* pp. 506-511, 2000.

[FNT07] D. Y. Feinstein, V.S.S. Nair, and M. A. Thornton, "Advances in Quantum Computing Fault Tolerance and Testing", In *Proc of High Assurance System Engineering Symposium,* pp. 369-370, 2007.

[FT07] D.Y. Feinstein and M. A. Thornton, "ESOP Transformation to Majority Gates for Quantum-dot Cellular Automata Logic Synthesis", In *Int'l Reed-Muller Workshop,* on CD ROM, 2007.

[FTK08] D. Y. Feinstein, M. A. Thornton, and F. Kocan, "Accelerated Temperature Tests by Internal Heating at frequencies above F_{max}", Submitted to *ITC'08*.

[FTK07] D. Y. Feinstein, M. A. Thornton, and F. Kocan, "System-on-chip power consumption refinement and analysis", In *DCAS'07*, 2007.

[FTM08] D. Y. Feinstein, M. A. Thornton, and D. M. Miller, "Partially Redundant Logic Detection Using Symbolic Equivalence Checking in Reversible and Irreversible Logic Circuits", in *DATE'08* pp 1378-1381, 2008.

[FTM08b] D. Y. Feinstein, M. A. Thornton, and D. M. Miller, "On the Data Structure Metrics of Quantum Multiple-valued Decision Diagrams", to appear in *ISMVL'08*.

[FTM08c] D. Y. Feinstein, M. A. Thornton, and D. M. Miller, "Lexicographic Reversible Logic Synthesis Guided by Dynamic Variables Reordering", submitted to *ICCAD'08*.

[FTM08d] D. Y. Feinstein, M. A. Thornton, and D. M. Miller, "Superposition manifestation as skipped variables in quantum computing decision diagrams", in preparation for *Physical Review Letters*.

[FTM08e] D. Y. Feinstein, M. A. Thornton, and D. M. Miller, "QMDD-based CAD tools for quantum and reversible logic", in preparation for IEEE *Transactions on CAD*.

[FTN07] D. Y. Feinstein, M. A. Thornton, and V.S.S. Nair, "Prefix Parallel Adder Virtual Implementation in Reversible Logic", In *IEEE Region 5 Conference April 2007* on CD ROM.

[Feyn85] R. P. Feynman, "Quantum mechanical computers," In *Optics News*, vol. 11, pp. 11–20, 1985.

[Feyn82] R. P. Feynman, "Simulating physics with computers," In *Int. J. Theor. Phys.,* Vol. 21, 6&7, pp. 467-488, 1982.

[FTY87] H. Fleisher, M. Tavel, J. Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms", In *IEEE Trans. Comp.* Vol. 36, No. 2, pp. 247-250, Feb. 1987.

[FT82] E. Fredkin and T. Toffoli, "Conservative Logic," In *Int, J. Theoretical Physics*, Vol. 21, Nos. 3/4, pp. 219-253, 1982.

[FS90] S.J. Friedman and K. J. Supowit, "Finding the Optimal Variable Ordering for Binary Decision Diagrams", In *IEEE Trans. on Comp.*, Vol. 39, Issue 5. pp.710-713, May 1990.

[FFK88] M Fujita, H. Fujisawa, and T. Kawato, "Evaluations and Improvements of a Boolean Comparison Method Based on Binary Decision Diagrams", In *ICCAD'88*, pp. 50-54, 1988.

[FMK91] M Fujita, Y. Matsunaga and T. Kakuda, "Logic synthesis: On variable ordering of binary decision diagrams for the application of multi-level logic synthesis", In *Proc. of the conference on European Design Automation*, pp. 50-54, Feb. 1991.

[FMY97] M Fujita, P.C. McGeer, and J. C.-Yang, "Multi-terminal binary decision diagrams: An efficient datastructure for matrix representation", In *Formal Methods Sys. Des.,* 10(2-3), pp. 149-169, 1997.

[GTFM07] D. Goodman, M. A. Thornton, D.Y. Feinstein**,** and D.M. Miller, "Quantum Logic Circuit Simulation Based on the QMDD Data Structure", In *Int'l Reed-Muller Workshop,* on CD ROM, 2007.

[Grev99] D. Greve, "QDD: A quantum computer emulation library", available on-line: http://thegreves.com/david/QDD/qdd.html, 1999.

[Grov96] L.K. Grover, "A fast quantum mechanical algorithm for database search", In *Proc. of 28th Symposium on the Theory of Computing*, pp. 212-219, 1996.

[GCD06] D. Groβe, X. Chen and R. Drechsler, "Exact Toffoli network synthesis of reversible logic using Boolean Satisfiability", In *Proceedings of the 2006 Dallas Circuits and Systems Workshop*, on CD ROM, 2006.

[GCDD07] D. Groβe, X. Chen, G.W. Dueck and R. Drechsler, "Exact SAT-based Toffoli network synthesis", *GLSVLSI '07*, pp. 96-101, 2007.

[GFX06] Y. Guowu, X. Fei, S. Xiaoyu, W.N.N. Hong, M.A. Perkowski, "A Constructive Algorithm for Reversible Logic Synthesis", In *IEEE Congress on Evolutionary Computation,* pp. 2416-2421, *2006.*

[GJL06] P. Gupta, N.K. Jha, and L. Lingappan, "Design methodologies for emerging technologies: Test generation for combinational quantum cellular automata (QCA) circuits", *Proceeding DATE'06*, pp. 311-316, March 2006.

[GAJ06] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits", In *IEEE Trans. on CAD*, Vol. 25, No. 11, pp. 2317-2330, 2006.

[Haye05] J.P. Hayes, "Faults and Tests in Quantum Circuits", In *14th Proc. of Asian Test Symposium*, Dec. 2005.

[Heis30] W. Heisenberg, *The Physical Principles of the Quantum Theory*, (Translated from German) University of Chicago Press 1930.

[Hirv04] M. Hirvenko, *Quantum Computing*, Springer-Verlag, 2nd Edition, 2004.

[Holl90] S. S. Holland, Jr., *Applied Analysis by the Hilbert Space Method*, Marcel Dekker, Inc. 1990.

[HW05] M. Homeister and St. Waack, "Quantum ordered binary decision diagrams with repeated tests", In *Proc. of the 7th Int'l Symposium on Representations and Methodology of Future Computing Technology*, Tokyo 2005.

[HYTC00] C-Y. Huang, B. Yang, H-C Tsai and K-T Cheng, "Static Property Checking Using ATPG vs. BDD Techniques", In *ITC'00*, pages 309-316, 2000.

[HSY+06] W.N.N. Hung, X. Song, G. Yang, J. Yang, and M Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis", In *IEEE Trans. on CAD*, Vol. 25, No. 9, pp. 1652-1663, Sept. 2006.

[HMS+05] J. Huang, M. Momenzadeh, L. Schiano, M. Ottavi, and F. Lombardi, "Tile-based QCA design using majority-like logic primitives", In *J. Emerging Tech. in Comp. Systems,* Vol. 1, No. 3, pp. 163-185, Oct. 2005.

[IS75] G.H. Ibarra and S.K. Sahni, Polynomially complete fault detection problems", In *IEEE Transc. Comp.*, Vol. C-24, No. 3, pp. 242-249, 1975.

[IPWK06] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiatowicz, "Interconnection Networks for Scalable Quantum Computers", In *ISCA '06*, pp. 1-12, May 2006.

[ISY91] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables", In *ICCAD'91*, pp. 472-475, 1991.

[IKY02] K. Iwama, Y. Kambayashi and S. Yamashita, “Transformation rules for designing CNOT-based quantum circuits”, In *DAC’02*, pp. 419-425, 2002.

[Kern00] P. Kerntopf, ”A comparison of logical efficiency of reversible and conventional gates”, In *Int’l Workshop Logic Synthesis*, pp. 261-269, 2000.

[Kern04] P. Kerntopf, “A new heuristic algorithm for reversible logic synthesis”, In *DAC’04*, pp. 835-837, 2004.

[KSV02] A.Y. Kitaev, A.H.Shen, and M.N. Vyalyi, *Classical and Quantum Computation*, (Translated from Russian by L.J. Senechal), American Mathematical Society, 2002.

[KPK03] M.H.A. Khan, M.A. Perkowski and P. Kenropf, “Multi-output Galois Field Sum of Products synthesis with new quantum cascades”, In *ISMVL’03*, pp. 146-153, May 2003.

[Kurz07] R. Kurzweil, “The web within us: when minds and machines become one”, *The Farfel Distinguished Lecture,* Univ. of Houston, April 4th 2007.

[LF80] R. Ladner and M. Fischer, “Parallel prefix computation", In *J. Assoc. Comp. Mach.*, Vol 27, pp. 831-838, 1980.

[Land61] R. Landauer, “Irreversibility and Heat Generation in the Computing Process*”,* In *IBM J. Research and Development*, vol. 3, pp. 183-191, July 1961.

[Lawd67] D.F. Lawden, *The Mathematical Principles of Quantum Mechanics*, Methuen & Co. Ltd. 1967, (Dover Publications reprint 2005).

[LTP93] C.S. Lent, P.D. Tougaw, and W. Porod, “Bistable saturation of in coupled quantum dot for quantum cellular automata”, In *Appl. Phys. Letters*, Vol. 62, No. 7, pp. 714-716, 1993.

[Leun89] S. S. Leung, “Behavioral modeling of transmission gates in VHDL”, In *DAC’89,* pp. 746-749, 1989.

[KLM07] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*, Oxford University Press 2007.

[MM05] D.C. Marinescu and G.M. Marinescu, *Approaching Quantum Computing,* Pearson Prentice Hall, 2005.

[Masl02] D. Maslov, "Dynamic programming algorithms as reversible circuits: symmetric function realization", 5 pages, 2002.

[Masl03] D. Maslov, *Reversible Logic Synthesis*, PhD Dissertation, 2003.

[Masl05] D. Maslov. *Reversible Logic Synthesis Benchmarks Page*. Online: http://www.cs.uvic.ca/~dmaslov/, Nov. 15, 2005.

[MDM03] D. Maslov, G.W. Dueck, and D.M. Miller, "Simplification of Toffoli networks via templates", In *16th Symp. on Integrated Circuits and System Design*, Sao Paulo, Brazil, On CD ROM, Sept. 2003.

[Micz03] A. Miczo, *Digital Logic Testing & Simulation*, 2nd Edition, John Wiley & Sons, 2003.

[Mill02] D. M. Miller, "Spectral and two-place decomposition techniques in Reversible Logic", In *Proc. IEEE Midwest Symp. Circuits and Systems*, August 2002.

[MD03] D. M. Miller and R. Drechsler. "Augmented sifting of multiple-valued decision diagrams", In *ISMVL'03*, pp. 275-282, 2003.

[MD03b] D.M. Miller and G. W. Dueck, "Spectral techniques for reversible logic synthesis", In *Proc. 6th International Symposium on Representations and Methodology of Future Computing Technology*, Trier, Germany, pp. 56-62, March 2003.

[MDM04] D. M. Miller, G.W. Dueck, and D. Maslov, "A Synthesis Method for MVL Reversible Logic," In *ISMVL'04*, Toronto, Canada, pp. 74-80, May 2004.

[MMD03] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis", In *DAC'03*, 318-323, June 2003.

[MT06] D.M. Miller and M.A. Thornton, "QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits", In *ISMVL'06*, on CD ROM, May 2006.

[MTG06] D.M. Miller, M.A. Thornton, and D. Goodman, "A Decision Diagram Package for Reversible and Quantum Circuits", *IEEE Congress on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pp. 8597-8604, July, 2006.

[MFT07] D. M. Miller, D. Y. Feinstein, and M. A. Thornton, "Variable Reordering and Sifting for QMDD", In *ISMVL'07,* on CD ROM, 2007.

[MFT07b] D. M. Miller, D. Y. Feinstein, and M. A. Thornton, "QMDD Minimization using Sifting for Variable Reordering", In *Journal of Multiple-valued Logic and Soft Computing*, Vol. 13, no. 4-6, pp. 537-552, 2007.

[MP01] A. Mischenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products", *5th Int'l Reed-Muller Workshop*, pp. 242-250, 2001.

[M63] F. Miyata, "Realization of arbitrary logical functions using majority element", *IEEE TEC*, Vol. EC-12, No. 3, pp. 183-191, 1963.

[MZ00] S. Mourad and Y. Zorian, *Principle of Testing Electronic Systems*, John Wiley & Sons, 2000.

[Niel98] M.A. Nielsen, *Quantum Information Theory*, PhD Dissertation, University of New Mexico, Dec. 1998, available online: www.arXiv:quant-ph/0011036v1 Nov. 2000.

[NC00] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Qunatum Information*, Cambridge University Press, 2000.

[Omer03] B. Omer, *Structure Quantum Programming*, PhD. Thesis, Institute of Information Systems, Technical University of Vienna, May 2003, available online: www. /tph.tuwien.ac.at/~oemer.

[Omne99] R. Omnes, *Understanding Quantum Mechanics*, Princeton University Press, 1999.

[PS95] S. Panda and F. Somenzi, "Who are the variables in your neighborhood", In *ICCAD'95*, pp. 1-4, December 1995.

[PHM04] K.N. Patel, J.P. Hayes, and I.L. Markov, "Fault testing for reversible logic", In *IEEE Trans. on CAD*, Vol. 23, No. 8, pp. 1220-1230, August 2004.

[PBL05] M.A. Perkowski, J. Biamonte, and M. Lukac, "Test generation and fault localization for quantum circuits", In *ISMVL'05*, pp 146-153, May 2005.

[PKB01] M.A. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, S. Xiaoyu, A. Al-Rabadi, L. Jezwiak, A. Coppola and B. Massey, "Regular realization of symmetric functions using reversible logic", In *Proc of Symposium on Digital Systems Design*, pp. 245-252, 2001.

[PFBH05] I. Polian, T. Fiehn, B. Becker, J. P. Hayes, "A Family of Logical Fault Models for Reversible Circuits" In *Proc. of 14th Asian Test Symposium*, pp. 422-427, Dec. 2005.

[Pres97] J. Preskill, *Fault-Tolerant Quantum Computation,* 58 pages, arXive e-print quant-ph/9712048, 1997.

[QSL+03] H. Qi, S. Sharma, Z.H. Li, G.L. Snider, A. Orlov, C.S. Lent and T.P. Fehlner, "Molecular quantum cellular automata cells", In *J. Am. Chem. Soc.*, 125, pp 15250-15259, 2003.

[RP00] E. Rieffel and W. Plack, "An introduction to quantum computing for non-physicists", In *ACM Computing Surveys*, Vol. 32, No. 3, pp. 300-335, Sept. 2000.

[RHBA03] B. Ratchev, M. Hutton, G. Baeckler and B. van Antwerpen, "Logic synthesis and mapping: Verifying the correctness of FPGA logic synthesis algorithms*"*, In *Proc. of eleventh international symposium on Field Programmable Gate Arrays*, February 2003.

[Rude93] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams", In *ICCAD'93*, pp. 42-47, November 1993.

[SSZ07] M. Saeedi, M. Seighi, and M.S. Zamani, "A novel synthesis algorithm for reversible circuits", In *ICCAD'07*, pp. 65-68, 2007.

[Saku94] J.J. Sakurai, *Modern Quantum Mechanics,* Addison-Wesley, 1994.

[S99] T. Sasao, *Switching Theory for Logic Synthesis,* Kluwer Academic Publishers, 1999.

[SB90] T. Sasao and Ph. W. Besslich, "On the complexity of MOD-2 sum PLA's", *IEEE Trans. on Comp.*, Vol. 34, No. 2, pp. 262-266, 1990.

[Schl05] M. Schlosshauer, "Decoherence, the measurement problem, and interpretations of quantum mechanics", arXiv:quant-ph/0312059v4, 2005.

[SW94] M. Sauerhoff and I. Wegener, "On the complexity of minimizing the OBDD size for incompletely specified functions", Forschungsbericht Nr. 560, Universität Dartmund, 1994.

[Schr35] E. Schrödinger, "The present situation in quantum mechanics", In *Naturwissenschaften*, (translated from German) Nov. 1935.

[Shan48] C.E. Shannon, "The synthesis of two-terminal switching circuits", In *Bell Systems Technical Journal*, 1948.

[Shor94] P. W. Shor, "Algorithms for quantum computation: Discrete log and factoring", In *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124-134, 1994.

[SPMH02] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits", In *ICCAD'02*, pp. 353-360, 2002.

[Some95] F. Somenzi, "The CUDD Package", University of Colorado at Boulder, 1995. Version 2.4.0 available on-line at: http://vlsi.colorado.edu/~fabio/.

[SPMH03] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits", *IEEE Trans. on CAD*, Vol. 22, No. 6, pp. 710-722, June 2003.

[SMA01] R.S. Stanković, C. Moraga and J.T. Astola, "Reed-Muller expressions in the previous decade", In *5th Int'l Reed-Muller Workshop*, 2001, pp. 7-26.

[Stea96] A. Steane, "Error correction codes in quantum theory", *Phys. Rev. Lett.*, 77:796, 1996.

[TJML04] M.B. Tahoori, H. Jing, M. Momenzadeh, and F. Lombardi, "Testing of quantum cellular automata", In *IEEE Trans. on Nanotechnology*, Vol. 3, no. 4, pp. 432-442, Dec. 2004.

[Thor95] M.A. Thornton, "Spectral based numerical methods for combinatorial logic synthesis", *PhD Thesis, Southern Methodist University*, 1995. Available on-line: http://engr.smu.edu/~mitch/publications.html.

[TLP93] P.D. Tougaw, C.S. Lent, and W. Porod, "Bistable saturation in coupled quantum dot cells", In *J. Appl. Phys*, Vol. 74, No. 5, pp. 3558-3566, 1993.

[VMO06] R. Van Meter and M. Oskin, "Architectural implications of quantum computing technologies", In *JETC*, Vol. 2 Issue 1.pp. 31-63, 2006.

[Vonn95] J. von Neumann, *Mathematical Foundations of Quantum Mechanics*, Princeton University Press, 1955.

[VRM+03] G.F. Viamontes, M. Rajagopalan, I.L. Markov, and J.P. Hayes, "Gate-level simulation of quantum circuits", In *ASP-DAC 2003,* pp. 295–301, 2003.

[Wall01] J. Wallace, "Quantum computer simulators", In *Journal of Computing Anticipatory Systems*, Vol. 10 pp. 230-245, 2001.

[WDJB04] K. Walus, T.J. Dysart, G.A. Jullien, and R.A. Budiman, "QCADesigner: a rapid design and Simulation tool for quantum-dot cellular automata", In *IEEE Trans. on Nanotechnology ,* Vol. 3, No. 1, pp. 26 – 31, March 2004.

[WJ06] K. Walus and G.A. Jullien, "Design tools for an emerging SoC technology: quantum-dot cellular automata", In *Proc. of the IEEE*, Vol. 94, No. 6, pp. 1225 – 1244, June 2006.

[WSJ+04] K. Walus, G. Schulhof, G., G.A. Jullien, R. Zhang, and W. Wang, "Circuit design based on majority gates for applications with quantum-dot cellular automata", In *38th Asilomar Conference on Signals, Systems and Computers*, Vol. 2, pp. 1354 - 1357, Nov. 2004.

[Wear67] B.L. van der Waerden, *Sources of Quantum Mechanics*, North-Holland Publishing Co. Ltd. 1967, (Dover Publications reprint 2007).

[WG07] R. Wille and D. Groβe, "Fast exact Toffoli network synthesis of reversible logic", In *ICCAD'07*, pp. 60-63, 2007.

[XCN+06] S.H. Xiaobo, M Crocker, M. Niemier, Y. Minjun and G. Bernstein, "PLAs in quantum-dot cellular automata", *Proc. of Emerging VLSI Technologies and Architectures (ISVLSI'06)*, 6 pp, March 2006.

[YMSS06] S. N. Yanushkevitch, D. M. Miller, V. P. Shmerko and R. S. Stankovic, *Decision Diagram Techniques for Micro- and Nanoelectronic Design*, CRC Taylor and Francis, 2006.

[ZWWJ04] R. Zhang, K. Walus, W. Wei, and G.A. Jullien, "A method of majority logic reduction for quantum cellular automata", In *IEEE Trans. on Nanotechnology*, Vol, 3, No. 4, pp. 443 – 450, Dec. 2004.

[ZJ06] R. Zhang and N.K. Jha, "Threshold/majority logic synthesis and concurrent error detection targeting nanoelectronic implementations", In *GLSVLSI'06* pp. 8-13, April 2006.

[Zimm98] R. Zimmermann, *Binary adder architectures for cell-based VLSI and their synthesis*, PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, Hartung-Gorre Verlag, 1998. Online: http://www.iis.ee.ethz.ch/~zimmi/.