TECHNOLOGY-DEPENDENT QUANTUM LOGIC SYNTHESIS AND COMPILATION

Approved by:

TECHNOLOGY-DEPENDENT QUANTUM LOGIC SYNTHESIS AND COMPILATION

A Dissertation Presented to the Graduate Faculty of the

Lyle School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Electrical Engineering

by

Kaitlin N. Smith

(B.S., EE, Southern Methodist University, 2014)
(B.S., Mathematics, Southern Methodist University, 2014)
(M.S., EE, Southern Methodist University, 2015)

December 21, 2019

ACKNOWLEDGMENTS

I am grateful for the many people in my life who made the completion of this dissertation possible. First, I would like to thank Dr. Mitch Thornton for introducing me to the field of quantum computation and for directing me during my graduate studies. I would also like to thank my committee for supporting my research and for all of the suggestions and guidance that helped me to develop my skills as a scientist.

To my family and friends: thank you for being there. You have no idea how much your constant encouragement, advice, and love have meant to me over the years as I completed this degree.

To my Mom and Dad: thank you for always being my biggest fans and for always believing in me. You have both taught me so much, and have given me the courage to chase my dreams. I love you.

Smith , Kaitlin N. B.S., EE, Southern Methodist University, 2014
B.S., Mathematics, Southern Methodist University, 2014
M.S., EE, Southern Methodist University, 2015

Technology-dependent Quantum Logic Synthesis and Compilation

Advisor: Dr. Mitchell Thornton - Committee Chairman
Doctor of Philosophy degree conferred December 21, 2019
Dissertation completed September XX, 2019

The models and rules of quantum computation and quantum information processing (QIP) differ greatly from those that govern classical computation, and these differences have caused the implementation of quantum processing devices with a variety of new technologies. Many platforms have been developed in parallel, but at the time of writing, one method of quantum computing has not shown to be superior to the rest. Because of the variation that exists between quantum platforms, even between those of the same technology, there must be a way to automatically synthesize technology-independent quantum designs into forms that are capable of physical realization on a quantum computer (QC) with specific operating parameters. Additionally, results of synthesis must be formally verified to confirm that output technology-dependent specifications are logically identical to their original, technology-independent forms. The first contribution of this work to the field of quantum computing is the creation of such a methodology. Quantum technology mapping and optimization for machines with fixed coupling maps and libraries of gates can be performed with an automatic quantum compiler, and the development and test of this compiler will be explored in this dissertation. Furthermore, this compiler can be considered in a more general context to be a synthesis tool for QIP circuits in a specific realization technology, many of which are capable of implementing systems where the radix of computation, r, is greater than two. As a result of this ability, the second contribution of this work is the presentation of architectures for higher-dimensional quantum entanglement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ALD	Atomic Layer Deposition
CNOT	Controlled-NOT
CPU	Central Processing Unit
CTR	Connectivity Tree Reroute
CZ	Controlled-phase
EPR	Einstein, Podolsky, and Rosen
ESOP	Exclusive Sum of Products
FIB	Focused Ion Beam
GHZ	Greenberger–Horne–Zeilinger
HBr	Hydrogen Bromine
ICP	Inductively Coupled Plasma
MQW	Multi-quantum Well
NISQ	Noisy Intermediate-scale Quantum
OAM	Orbital Angular Momentum
QASM	Quantum Assembly Language
QC	Quantum Computer
QFT	Quantum Fourier Transform
QIP	Quantum Information Processing
QKD	Quantum Key Distribution
QMDD	Quantum Multiple-valued Decision Diagram
QPIC	Quantum Photonic Integrated Circuits

QPU	Quantum Processing Unit
QUIL	Quantum Instruction Language
SDK	Software Development Kit
ZDD	Zero-surpressed Decision Diagram

Chapter 1

Introduction

Modern digital circuitry can evaluate complex equations quickly and with precision, allowing for rapid rates of data creation and processing. Computers that are modeled with classical switching theory have revolutionized problem solving, but due to their nature, these devices are not ideal for every calculation without large overhead with respect to time or physical resources. The quantum computing model, however, has potential to solve some of these difficult problems with greater efficiency due to a fundamentally different underlying model of computation.

The field of quantum computation is one of great potential. Theoretical work has proven that a quantum computer (QC) can complete tasks such as searching large data sets and simulating quantum behavior at a rate much faster than what is currently possible. In addition, QCs have demonstrated other computing advantages, such as the ability to encode state spaces where the basis dimension is greater than two. Unfortunately, physical QC development is not as advanced as the state of quantum theory and algorithms. Currently, there are many contenders for what will eventually be the standard quantum computing platform or technology.

Because of the variety that exists in quantum implementations and devices, techniques are needed that map quantum circuits, especially those that generate important phenomena such as quantum entanglement, to physical technology platforms that contain varied gate libraries and qubit layout topologies. Thus, methods for quantum logic synthesis, or hardware compilation, are required.

1.1. Classical Computation and Limitations

The electronic computers that are widely available today are referred to here as classical computers. These devices process information in discrete units of information, modeled as bits, that have a value of either one or zero indicating an asserted or a deasserted state, respectively. Over the last half century, classical computers have consistently improved in their processing power. This advance in technology has helped simplify many complex calculations, such as calculus problems, and has allowed the development of classical algorithms that support the analysis of system reliability (Smith et al., 2017), computer security (Taylor et al., 2017), poynomial decomposition (Smith and Thornton, 2019h), and many others. There are, however, still several problems that are impractical to solve on a classical machine due to their spatial or temporal complexities. Additionally, because of the bistable nature of transistors that are standard in electronic technology, the majority of classical computation is based on an implementation of a radix-2 model where the bit takes a single value of $\mathbb{B} \in \{0, 1\}$. Due to the very nature of classical computing, the Turing machine model guarantees that theoretical complexities can not be overcome. Examples of problems that are currently difficult for classical computers to solve involve searching large state spaces and factoring numbers (DiVincenzo, 2010). Even if classical computers could continue the trend of increased performance, although the momentum of computational power is slowing due to the inability to continue to scale down the feature size of transistors, complexity issues would still exist because some problems remain intractable. The abilities of classical computers will always exhibit deficits due to the Turing model, leaving many intractable problems unsolved. Therefore, alternative computing methods, such as quantum methods, should be investigated.

1.2. Contribution

The models and rules of quantum computation and quantum information processing (QIP) differ greatly from those that govern classical computation, and these differences have caused quantum device implementation with a variety of new technologies. Many platforms have been developed in parallel, but at the time of writing, one method of quantum computing has not shown to be superior to the rest. Because of the variation that exists between quantum platforms, even between those of the same technology, there must be a way to automatically synthesize technology-independent quantum designs into forms that are capable of physical realization on a QC with specific operating parameters. Additionally, results of synthesis should be verified to confirm that output technology-dependent specifications are logically identical to their original, technology-independent forms. The first contribution of this work to the field of quantum computation is the creation of such a methodology. Quantum technology mapping and optimization for machines with fixed coupling maps and libraries of gates can be performed with an automatic quantum compiler. The development and test of compilation algorithms will be explored in this dissertation. Compilation can be considered in a more general context to be a synthesis tool for QIP circuits in a specific realization technology, and many of these technologies are capable of implementing systems where the radix of computation, r, is greater than two. As a result of this ability, the second contribution of this work is the presentation of architectures for higher-dimensional quantum entanglement.

The contributions above have been included in (Smith and Thornton, 2017, 2018, 2019a,b,c,d,e,f; Smith et al., 2019). Other publications and works completed during my PhD studies include (Smith and Thornton, 2015, 2019g,h; Smith et al., 2017, 2018a,b,c; Taylor et al., 2017).

Chapter 2

Quantum Information

2.1. The Qubit

In classical computation, units of information are stored in strings of bits. Each bit can have a logic value of $\mathbb{B} \in \{0, 1\}$. According to switching theory, Boolean logic gates are applied to bit values in order to cause change over time. The mathematical model of Boolean algebra is used to create and manipulate meaningful data from strings of bits, and this information must be represented physically in computing systems such as in the form of voltage or current in a wire or as light in a fiber optic cable. The classical states of 0 and 1 are most frequently realized within electronic devices as either a low or high voltage, respectively, when using positive logic or as a high or low voltage, respectively, when using negative logic.

In QIP, the unit of information is the quantum bit, or qubit. The qubit stores information by holding values such as $|0\rangle$ or $|1\rangle$ which are a set of orthonormal basis states in Dirac notation (Dirac, 1958) that represent the two-dimensional column vectors of

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.1}$$

Although similarities between the qubit and the classical bit exist, the qubit may represent an infinite number of states while in a superposition of a set of basis states. For example, the qubit $|\Psi\rangle$ may equal either $|0\rangle$ or $|1\rangle$, or it may take a value that is a linear combination of both basis states in the form of

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{2.2}$$

In Eqn. 2.2, the probability amplitudes α and β are complex numbers, c, that take the form of $c = a + bi$. Here, i is an imaginary number where $i^2 = -1$. The state of a qubit can be visualized using the geometry of the Bloch sphere (Bloch, 1946; Nielsen and Chuang, 2010). On the Bloch sphere, pictured in Fig. 2.1, $|0\rangle$ is found at the north pole, $\hat{z}$ whereas $|1\rangle$ is found at the south pole, $-\hat{z}$. The states $|0\rangle$ and $|1\rangle$ are referred to as the computational basis. A qubit may take any value on the surface of the Bloch sphere that represents a linear combination of $|0\rangle$ and $|1\rangle$.

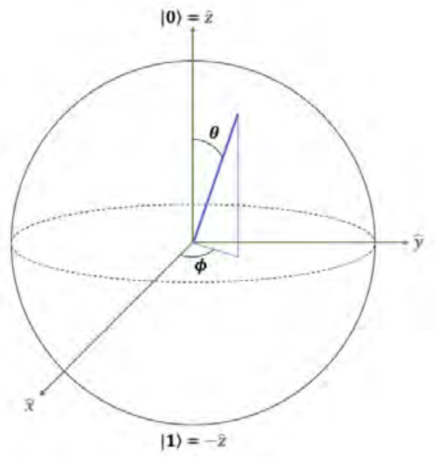


Figure 2.1. The Bloch sphere

The magnitude of qubit probability amplitudes must sum to unity, thus, the Bloch sphere has a radius of one. In Eqn 2.2, the probability that $|\Psi\rangle = |0\rangle$ is equivalent to $\alpha^*\alpha = |\alpha|^2$ and the probability that $|\Psi\rangle = |1\rangle$ is $\beta^*\beta = |\beta|^2$ where * indicates a complex conjugate and $|\alpha|^2 + |\beta|^2 = 1$. Once a qubit is measured, it collapses into a basis state and its quantum properties are lost.

2.2. Physical Quantum Implementations

Qubits have been successfully realized in many different mediums. In microscopic realizations, the size of the particles acting as qubits are on the atomic scale. Because these particles are so small, they exhibit the quantum characteristics needed by the qubit to exist in states of superposition. Of these different particle types, none have proven to be a superior information carrier. Some examples of popular qubit representations within quantum particle systems are photons in optical cavities, photons in microwave cavities, ions in ion traps, spin in electrons, and charge in quantum dots (Nielsen and Chuang, 2010). Qubits can also be realized in larger, mesoscopic systems like with electric charge in solid-state superconducting circuits (Koch et al., 2007).

At the time of writing, quantum technology is in the noisy intermediate-scale quantum (NISQ) era (Preskill, 2018). The QCs available are of modest size, but a significant limiting factor with these devices is the high probability of an accidental measurement of qubit state that causes the system to collapse. This collapse, usually caused by an unintended interaction between a qubit and its environment, is called decoherence (Nielsen and Chuang, 2010). Scientists have observed through experimentation that some quantum systems are more resilient to decoherence than others. For example, the quantum coherence time for a photon within an optical cavity is approximately 10^{-5} second whereas the quantum coherence time for an indium atom within an ion trap is approximately 10^{-1} second (Nielsen and Chuang, 2010). More recent implementations with transmons, charge-based qubits used by companies such as IBM and Rigetti, have demonstrated coherence times of approximately 10^{-4} second (Devoret and Schoelkopf, 2013).

There are two types of qubits in QIP: stationary qubits and “flying” qubits. Stationary qubits are used to perform calculations in fixed locations such as within a integrated circuit. Because stationary qubits are used for computation, they must be implemented with technologies that allow qubits to easily interact with each other. Flying qubits, on the other hand, are used for the transmission of quantum information. This type of qubit must be implemented with technology resistant to decoherence so the quantum state is stable as it travels.

2.2.1. Transmons

The transmon QC is a popular platform of the NISQ era. The technology was first developed in 2007 at Yale University, and current work with these devices have demonstrated very promising coherence times of around 100 μs (Devoret and Schoelkopf, 2013; Koch et al., 2007). One- and two-qubit quantum state transformations can be executed with high fidelity (Tripathi et al., 2019), and as a result, many companies and research groups, such as IBM and Rigetti, have invested resources to continue to develop transmon technology. For example, IBM Q includes quantum machines with 5 or more qubits that can be programmed to run user-generated quantum circuits. A discussion of the IBM quantum machines can be found in Section 3.4.1 while information about the Rigetti devices can be found in Section 3.4.2.

Transmons are a physical quantum realization that are categorized as superconducting charge qubits. Superconducting qubits in general use microscopic phenomena within mesoscopic devices, such as electric charge in a circuit for the case of the transmon, to realize quantum information. To better understand how a transmon works, its components must be understood. A Cooper Pair Box, a capacitive shunt, and a transmission line resonator are the main components that comprise a transmon with the Cooper Pair Box being the element that contains the charge qubit. A Josephson Junction, two superconducting materials separated by a very thin insulator, along with a Cooper pair, two electrons bonded at low energy levels, form the Cooper Pair Box (Bader, 2013).

As indicated by the name "superconducting charge qubit," electric charge provides the representation of qubit basis states for the transmon. The transmission line resonator on the device provides a means of interacting with the qubit. For example, by applying specific electric fields to the resonator, a single qubit operation can be performed. Multiple transmon qubits can be coupled together if additional connecting resonators are added, allowing for multi-qubit transformations to occur. More details about the transmon architecture and the available operations can be found in (Bader, 2013; Devoret and Schoelkopf, 2013; Koch et al., 2007).

2.2.2. Photonics

Photonic computing is a promising area that provides some benefits with respect to security such as resilience to side channel attacks. There have been recent efforts to implement photonic computing using the classical Turing machine model (Singh et al., 2014; Smith and Thornton, 2015), however, the potential of quantum computing and the ability to use photons as qubit state carriers is considered by many in the field to be a more valuable use of photonics.

Photonic implementations could be considered one of the more successful physical quantum realizations. Its weaknesses due to the nature of light, however, have prohibited photonic devices from becoming the standard platform for quantum logic. In a photonic QC, qubits are realized with photons. A photonic qubit is characterized by having a long coherence time that can be demonstrated experimentally to travel lengthy distances at room temperature (Myers and Laflamme, 2006). This long coherence time is caused by the photon's resistance to coupling with other elements in its environment (Kok et al., 2007). While the photonic qubit's failure to interact with other objects is an advantage in terms of maintaining state, it creates difficulties in situations where qubits must interact in multi-qubit gates such as with the controlled-**NOT** (**CNOT**) or controlled-phase (**CZ**) operations. Single-qubit operations are easily implemented and deterministic with photonics, but multi-qubit gates are currently probabilistic in outcome. For example, the first quantum photonics gate with a control, a conditional phase flip gate, was demonstrated to have a success probability of $\frac{1}{16}$ in 2001 (Knill et al., 2001). These results were improved in 2003 by (O'Brien et al., 2003) when a photonic **CNOT** device was shown to operate with a success rate of $\frac{1}{9}$. Currently, controlled quantum operations implemented with photonic devices are proven to only have a fidelity of $\frac{1}{4}$ in ideal operating conditions (Eisert, 2005).

A physical quantum implementation must have a technique for encoding qubit state. Although other encoding methods exist with quantum photonics, the two main methods for representing the qubit are with photon polarization and location. In the first method, photon polarization acts as the information carrier where two orthogonal polarization angles of light represent a set of quantum basis states. Typically with the polarization-encoded qubit, horizontal polarization represents $|0\rangle$ and vertical polarization represents $|1\rangle$ (Kok et al., 2007). The second method of photonic qubit realization uses what is known as dual-rail representation. Dual-rail representation is a location-based means of representing quantum information. Whenever using location to represent qubit state, it is most common that the top rail represents $|0\rangle$ and the bottom rail represents $|1\rangle$ on quantum circuit diagrams (Kok et al., 2007; Myers and Laflamme, 2006). Converting between polarization-encoded and dual-rail encoded qubits is a relatively easy process that is frequently done in photonic quantum circuitry. To convert between the two forms of photonic qubits, both a polarizing beam splitter and a half wave plate can be used (Myers and Laflamme, 2006; O'Brien et al., 2003). The schematic representing the photonic transformation from a polarization encoding to a location encoding can be seen in Fig. 2.2.

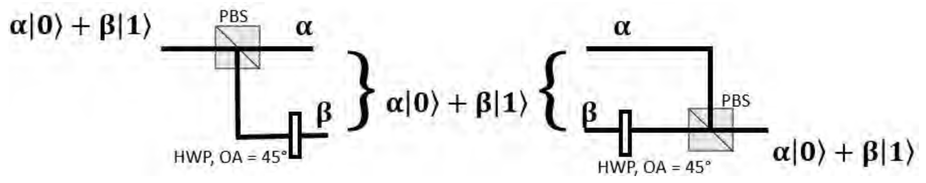


Figure 2.2. Photonic transformation between polarization and dual-rail encoding schemes

Orbital angular momentum (OAM) states of light are also used to encode qubit state (García-Escartín and Chamorro-Posada, 2008).

Choosing a qubit encoding scheme for a photonic quantum implementation depends on the qubit's task. For example, since fewer communication lines are needed for the polarization-encoded qubit, this representation may be more suitable for long-haul qubit transmissions. Whenever photonic quantum computations are performed, however, dual-rail representation is most frequently used. Additionally, dual-rail encoding is easier to implement on quantum photonic integrated circuits (QPICs). More information about photonic quantum operators for both polarization and location encoding methods can be found in Section 3.4.3.

2.3. The Superposition Principle

In Section 2.1, the concept of superposition for a single qubit was introduced. Because of superposition, if $|\Psi\rangle$ and $|\Phi\rangle$ are two quantum basis states for a qubit, any linear combination of these two states, $\alpha|\Psi\rangle + \beta|\Phi\rangle$ is also a valid state of the system where $|\alpha|^2 + |\beta|^2 = 1$. The superposition principle, however, is one that does not only pertain to a single qubit. Quantum networks composed of multiple qubits can be in states of superposition as well. For example, if there are two quantum systems, and these two systems have their quantum state held in the vectors $|x\rangle = \alpha_1|\Psi\rangle + \beta_1|\Phi\rangle = \begin{bmatrix}\alpha_1 \\ \beta_1\end{bmatrix}$ and $|y\rangle = \alpha_2|\Psi\rangle + \beta_2|\Phi\rangle = \begin{bmatrix}\alpha_2 \\ \beta_2\end{bmatrix}$, respectively. If these two systems were to be combined, the vector $|xy\rangle$ would be formed by the tensor product of two original state vectors:

$$|xy\rangle = |x\rangle|y\rangle = |x\rangle \otimes |y\rangle = \begin{bmatrix}\alpha_1 \\ \beta_1\end{bmatrix} \otimes \begin{bmatrix}\alpha_2 \\ \beta_2\end{bmatrix} = \begin{bmatrix}\alpha_1\alpha_2 \\ \alpha_1\beta_2 \\ \beta_1\alpha_2 \\ \beta_1\beta_2\end{bmatrix}$$

Due to the superposition principle, any linear function of the possible basis states of the multi-qubit system is also a valid state, as long as the inner product, or the dot product, of the vector formed from the qubits' combined state equals unity.

2.4. The Wavefunction and Quantum Computing

Quantum mechanics is commonly viewed under the perspective of the Schrödinger picture. When looking through this lens, state vectors evolve in time and operators are constant with respect to time. In other words, operators act on a wavefunction, Ψ, a mathematical description of the quantum points of interest of a system, causing change. The wavefunction includes complex-valued amplitudes, and the probabilities for the possible outcomes from measurement of the system can be derived from solving $|\Psi|^2 = \Psi^*\Psi$ (Griffiths, 1995). Upon measurement, the wavefunction collapses into a basis state of the system. The wavefunction is a solution of the time- and position-dependent Schrödinger equation

$$i\hbar\frac{\partial\Psi(x,t)}{\partial t} = \hat{H}\Psi(x,t). \tag{2.3}$$

In the equation above, $i = \sqrt{-1}$, and $\hbar$ is the reduced Planck constant. $\hat{H}$ is the Hamiltonian operator,

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x), \tag{2.4}$$

that represents the total energy for a system where where m represents mass and $V(x)$ represent potential as a function of position. The differential equation of Eqn. 2.3 can be solved to derive an expression for the wavefunction. To begin, the right side of the Eqn. 2.3 will be expanded using Eqn. 2.4 to form

$$i\hbar\frac{\partial\Psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\Psi(x,t)}{\partial x^2} + V(x)\Psi(x,t). \tag{2.5}$$

A function that describes how $\Psi(x,0)$ evolves into $\Psi(x,t)$ is desired, but this is difficult to find while the Schrödinger equation contains partial differential equations. To put Eqn. 2.5 into a simpler form, a separation of variables technique will be applied to find a solution in the form of $\Psi(x,t) = F(t)\Psi(x)$ where the time and position components of $\Psi(x,t)$ are separated into two different equations, respectively, that intersect under multiplication. The reason why $\Psi(x,t)$ is written as two functions rather than one is because it allows difficult partial derivatives to become total derivatives. Writing Eqn. 2.5 with $\Psi(x,t) = F(t)\Psi(x)$ gives

$$i\hbar\Psi(x)\frac{dF(t)}{dt} = \left(-\frac{\hbar^2}{2m}\frac{d^2\Psi(x)F(t)}{dx^2} + V(x)\Psi(x)F(t)\right),$$

$$i\hbar\Psi(x)\frac{dF(t)}{dt} = F(t)\left(-\frac{\hbar^2}{2m}\frac{d^2\Psi(x)}{dx^2} + V(x)\Psi(x)\right),$$

which can be reduced using Eqn. 2.4 to

$$i\hbar\Psi(x)\frac{dF(t)}{dt} = F(t)\hat{H}\Psi(x).$$

So that time- and position-dependent functions are grouped. Next, both sides of the equation will be divided by $F(t)\Psi(x)$ to give

$$i\hbar\frac{1}{F(t)}\frac{dF(t)}{dt} = \frac{1}{\Psi(x)}\hat{H}\Psi(x) = E. \tag{2.6}$$

Note how Eqn. 2.6 has the time- and position-dependent parts separated by the equals sign. Since changes to either x or t would cause only the right or left side of the equation to change, respectively, they need to be related by a term that is not a function of either variable. This term is a constant, E, that allows states of definite energy, or an eigenvectors of the Hamiltonian, to be expressed. Two new equations surface from the manipulation of Eqn. 2.6:

$$i\hbar\frac{dF(t)}{dt} = EF(t), \tag{2.7}$$

and

$$\hat{H}\Psi(x) = E\Psi(x). \tag{2.8}$$

Eqn. 2.7 is time-dependent and Eqn. 2.8 is time-independent and is instead dependent on position. Now the wavefunction can be solved while keeping the value for E in both Eqn. 2.7 and Eqn. 2.8 equal. Solving the differential equation in Eqn. 2.7 allows

$$i\hbar\frac{dF(t)}{dt} = EF(t),$$

to become

$$F(t) = F(0)e^{-iEt/\hbar}.$$

Now that the time-dependent Eqn. 2.7 has been solved, the variables can be recombined to create the wavefunction

$$\Psi(x,t) = F(t)\Psi(t) = F(0)e^{-iEt/\hbar}\Psi_E(x),$$

thus,

$$\Psi_E(x,t) = e^{-iEt/\hbar}\Psi_E(x, t = 0). \tag{2.9}$$

The subscript E is used to denote that $\Psi(x)$ is associated with the same state of definite energy as that of $F(t)$.

In quantum computing, the state of the wavefunction Ψ is written using dirac notation and is embodied by the qubit. For instance, the qubit $|\Psi\rangle$ can be in a state of either $|0\rangle$ or $|1\rangle$, or in a superposition of both simultaneously. After measurement, it collapses into one of the basis states. The qubit is transformed by quantum gates that are represented collectively by $\mathbf{U}$ such that

$$\mathbf{U} = e^{-iEt/\hbar} = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}. \tag{2.10}$$

$\mathbf{U}$ is a unitary operator. When $\mathbf{U}$ is substituted into Eqn. 2.9, the following form of the wavefunction,

$$|\Psi_E(x,t)\rangle = \mathbf{U}\,|\Psi_E(x,t=0)\rangle\,, \tag{2.11}$$

results. In Eqn. 2.11, $\mathbf{U}$ allows $|\Psi_E(x,t=0)\rangle$ to transform into $|\Psi_E(x,t)\rangle$ over time.

2.5. Quantum Operations

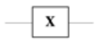
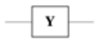
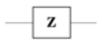
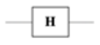
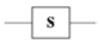
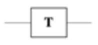
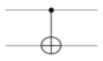
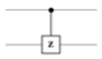
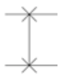
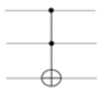
Quantum operators transform qubit state to implement quantum computation. If a quantum algorithm were to be thought of as a circuit, the quantum operators would be the gates. These operators are represented by a unique transfer function matrix of size $2^n \times 2^n$ where n is the number of qubits that the operation transforms. The transformation matrix for a quantum operator, $\mathbf{U}$, is always unitary, so the following characteristics are observed:

- $\mathbf{U}^\dagger\mathbf{U} = \mathbf{U}\mathbf{U}^\dagger = \mathbf{I}_n$,
- $\mathbf{U}^{-1} = \mathbf{U}^\dagger$,
- $\text{Rank}(\mathbf{U}) = n$,
- $|\mathbf{U}| = 1$.

In the identities above, the symbol † indicates a complex-conjugate transpose. Gate transformations can take place on single or multiple qubits. Some of the most common one- and two-qubit operations are included in Table 2.1.

The transformations described in Table 2.1 are frequently used quantum operations, and controlled variations of these gates can also be defined. For example, the **CNOT** operation is the controlled Pauli-**X**, or controlled **NOT**, operation where a control qubit determines the enable operation on a target qubit. Additional controls can be added onto the **CNOT**. The transformation matrix and symbol for the Toffoli operation, an operator that acts as a controlled-**CNOT**, can be seen in Table. 2.1. Adding more control qubits onto a Toffoli gate results in an n-qubit generalized Toffoli where $m = n - 1$ qubits act as controls and the n^{th} qubit is the target.

Table 2.1. Common single- and multi-qubit quantum operators

Operator	**Symbol**	**Transfer Matrix**
Pauli-X (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
CZ		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Quantum operators are combined to form quantum circuits, and quantum circuits can be described in a variety of ways. Some of the most popular techniques include drawing the circuits as graphs, like the one seen in Fig. 2.3, or describing them as a netlist with Quantum Assembly Language (QASM) or Quantum Instruction Language (Quil).

Understanding how information is transformed in a quantum circuit requires some knowledge of linear algebra and tensor products. As seen in Table. 2.1, quantum operators are represented by transformation matrices. To determine the resulting quantum state, $|\Psi_{out}\rangle$, after gate transformation, $\mathbf{U}$, the calculation

$$|\Psi_{out}\rangle = \mathbf{U}\,|\Psi_{in}\rangle \tag{2.12}$$

must be completed. To determine $|\Psi_{in}\rangle$, the input qubit values are combined via tensor product. Consider the quantum circuit pictured in Fig. 2.3. In this circuit, two qubits, $|a\rangle$ and $|b\rangle$, are each represented by a single horizontal line, but these horizontal lines should not be confused with conductors like those in electrical circuit schematics. Reading from left to right on the graph, qubit state evolves as time progresses and they pass though the **CNOT** gate. Together at the input they form the value of $|\Psi_{in}\rangle$ and

$$|\Psi_{in}\rangle = |a_{in}\rangle \otimes |b_{in}\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle\,.$$

Determining $|\Psi_{out}\rangle$ requires Eqn. 2.12 to be used, and in this case, **CNOT** will take the place of generalized transformation matrix $\mathbf{U}$ to generate

$$|\Psi_{out}\rangle = \mathbf{U}\,|\Psi_{in}\rangle = \mathbf{CNOT}\,|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle$$

.

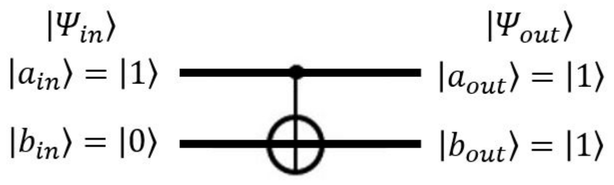

$|\Psi_{in}\rangle$
$|a_{in}\rangle = |1\rangle$
$|b_{in}\rangle = |0\rangle$
$|\Psi_{out}\rangle$
$|a_{out}\rangle = |1\rangle$
$|b_{out}\rangle = |1\rangle$

Figure 2.3. Quantum circuit example 1

Quantum operators are not limited to radix-2, or base-2, operations. Higher-order gates act on quantum digits or "qudits" that are characterized by three or more basis states. An example radix-4 Chrestenson gate can be found in (Smith et al., 2018a,b) and in Appendix A. Methods and operators for generating higher-radix quantum entanglement are found in (Smith and Thornton, 2019a,c) and within Chapter 6 and Chapter 7.

2.6. Requirements for Quantum Computation

Quantum computation holds the key to unlocking the mystery of nature as QCs are the ideal devices for the simulation of physics (Feynman, 1982). Performing quantum computations, however, requires more than simply realizing a qubit in a physical medium. For example, while it is important for a system to demonstrate quantum characteristics to hold qubits, these qubits must be able to evolve and interact with each other in order to represent computation. In (Deutsch, 1985), a work that many scientists accept as the fundamental model for quantum computation, the requirements for a quantum computer are described in detail. According to this paper, a grouping of n qubits that are successfully able to act as a quantum computer must demonstrate the following:

- Qubits must be initialized to a known state, such as $|0\rangle$ or $|1\rangle$.
- Qubits must be measurable, causing their collapse into a basis state.
- A qubit (or set of qubits) has the ability to evolve through a universal quantum gate or set of gates, $\mathbf{U}$, in a series that represents a unitary transformation (i.e. operations are reversible because $\mathbf{U}\mathbf{U}^{\dagger}=\mathbf{U}^{\dagger}\ \mathbf{U}=\mathbf{I}$).
- Qubits maintain their current quantum state if the aforementioned actions do not occur.

The requirements above list the basic necessities for the evaluation of a quantum algorithm. They describe a system where the outputs of the circuitry are dependent only on the current inputs, or the original qubits that were initialized to a known state. If viewed from a classical computing perspective, the requirements outlined in (Deutsch, 1985) allow us to realize combinational quantum logic circuits where the output is a function of the current circuit input only. These theoretical concepts for quantum computing have been expanded since their original introduction in 1985.

In (DiVincenzo, 2010), the essential operational characteristics for a quantum computer are described in greater detail along with the requirements for communicating in quantum networks. In addition to the above necessities for quantum algorithm execution from (Deutsch, 1985), (DiVincenzo, 2010) states that a practical quantum computer will need to be built using technology that is scalable and is capable of long coherence times to maintain qubit state during computation. For quantum computation between machines, qubits must be transmitted in a controlled manner, and those transmitted qubits, known as "flying qubits," must interact with stationary qubits in order to produce meaningful information (DiVincenzo, 2010).

2.7. Entanglement

Entanglement is one of the most significant quantum phenomena. It describes how two or more qubits can interact in such a way where they become a composite system that is no longer separable. In other words, none of the member qubits can be described independently from the qubit group once the group is in an entangled state. From a mathematical point of view, if $|\Psi\rangle$ is an entangled quantum state, it cannot be expressed as a product $|x\rangle \otimes |y\rangle$ of its component systems. The following four states are examples of two-qubit entangled states:

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \tag{2.13}$$

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \tag{2.14}$$

$$|\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \tag{2.15}$$

$$|\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \tag{2.16}$$

The states listed in the equations above are known as the Bell states. In some texts, they are also referred to as EPR states, or pairs, after Einstein, Podolsky, and Rosen and their groundbreaking paper (Einstein et al., 1935). The Bell states represent how two qubits may be maximally entangled, or each of the possible outcomes of an observation are equally likely. A Hadamard gate followed by a **CNOT** gate can be used to create Bell state entanglement. The Bell state generating circuit is pictured in Fig. 2.4. Entanglement between two qubits is not limited to just the Bell states, however. Other arbitrary entangled pairs are possible where the probability amplitudes are unequal.

One of the significant properties of an entangled qubit pair is that knowledge of a single qubit in the set gives insight to the rest of the member qubits. For example, consider a

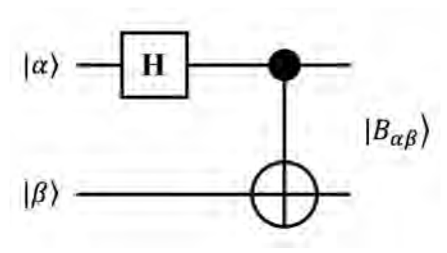


Figure 2.4. Bell state generator

two-qubit system in the Bell state of $|\beta_{00}\rangle$. Before measurement, the system has an equal probability of being in either state $|00\rangle$ or state $|11\rangle$, respectively. During computation, the first qubit is measured, and it collapses into a basis state. If the first qubit is measured to be a $|0\rangle$, the second qubit must also be $|0\rangle$ because the two qubits in the system were entangled. Likewise, if the first qubit is measured to be a $|1\rangle$, the second qubit must also be $|1\rangle$.

Quantum entanglement is an important phenomenon that is a critical component of most quantum computation and communications algorithms. The ability to experimentally demonstrate entanglement is significant because this phenomenon enables quantum computing algorithms that exhibit a computational advantage as compared to their classical counterparts. Another very important application of entanglement is that it allows for the implementation of ultra-secure quantum communications protocols. For example, although the original BB84 quantum key distribution (QKD) protocol only relies on superposition (Bennett Ch and Brassard, 1984), entanglement is necessary for many BB84 derivatives (Ekert, 1991; Enzer et al., 2002). Additionally, quantum factoring of composite prime numbers (Shor, 1994), quantum radar (Lanzagorta, 2011), quantum teleportation (Bennett et al., 1993), and many other applications depend upon and exploit the properties of entanglement in their implementation. The entire concept of many QIS systems such as teleportation, quantum communication channels, and others are based on the property of entanglement. Recently, the well-known recent Chinese experiments based upon their Micius satellite demonstrated that a quantum channel could be created between the earth and space. The Micius experiments utilized quantum entanglement generators as a key function (Yin et al., 2017) for their impenetrable communication network.

Chapter 3

Quantum Logic Synthesis Considerations

3.1. No-Cloning Theorem

Another unique characteristic quantum computation that further distinguishes it from the classical computation model is the inability copy information. This property is especially apparent whenever qubits are in a superimposed state. A classical bit encoded as a voltage level can ideally fan out onto multiple branches as connections are added in parallel. Due to the no-cloning theorem, a similar action cannot be performed on a qubit that results in the creation of two qubits with identical value (Nielsen and Chuang, 2010).

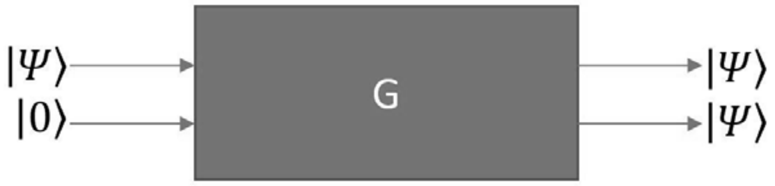


Figure 3.1. Proposed qubit copying gate, **G**

The no-cloning theorem can be proven using a contradiction. Assume that there exists a generalized cloning gate, **G**, that transforms any steady state, such as $|0\rangle$, into a copy of a qubit, $|\Psi\rangle$. The transformation **G** is represented by a unitary transformation matrix as is required for quantum gates. A block diagram describing the function of the proposed gate **G** can be seen in Fig. 3.1. With **G**, two orthogonal quantum basis states are cloned, $\mathbf{G}(|\Psi 0\rangle) = |\Psi\Psi\rangle$ and $\mathbf{G}(|\Phi 0\rangle) = |\Phi\Phi\rangle$. Attempts to copy a state that is in a superposition of these basis states, however, is not as successful. The qubit $|\beta\rangle = \frac{1}{\sqrt{2}}(|\Psi\rangle + |\Phi\rangle)$ will be the example state that will undergo the cloning transform:

Anticipated cloned value of $|\beta\beta\rangle$:

$$|\beta\beta\rangle = \frac{1}{\sqrt{2}}(|\Psi\rangle + |\Phi\rangle) \otimes \frac{1}{\sqrt{2}}(|\Psi\rangle + |\Phi\rangle) = \frac{1}{2}[|\Psi\Psi\rangle + |\Psi\Phi\rangle + |\Phi\Psi\rangle + |\Phi\Phi\rangle]$$

Actual value of $|\beta\beta\rangle$ after applying cloning transform, $\mathbf{G}$:

$$\mathbf{G}(|\beta 0\rangle) = \mathbf{G}\left(\frac{1}{\sqrt{2}}(|\Psi\rangle + |\Phi\rangle)|0\rangle\right) = \frac{1}{\sqrt{2}}\left[\mathbf{G}(|\Psi 0\rangle) + \mathbf{G}(|\Phi 0\rangle)\right] = \frac{1}{\sqrt{2}}\left[|\Psi\Psi\rangle + |\Phi\Phi\rangle\right]$$

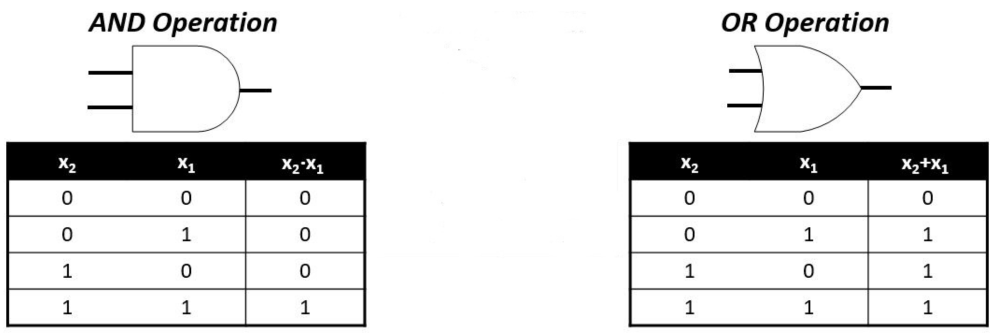
The anticipated cloned value of $|\beta\beta\rangle$ using $\mathbf{G}$ and the actual value are unequal. This contradiction proves the no-cloning theorem since it shows that $\mathbf{G}$ cannot exist.

The limitations of the no-cloning theorem on the qubit have severe implications in terms of how quantum algorithms store information. Because of the inability to copy a qubit, quantum memory will differ greatly from a classical memory. An example of a proposed method to implement quantum storage using ring oscillator structures can be found in (Smith et al., 2018c). While a classical bit can theoretically be read from memory as many times as necessary while it occupies a memory address, stored qubits are available for one use only as measurement causes superposition to collapse into a basis state. It is reasonable to conclude that to agree with the no-cloning theorem, any sort of space that once held quantum information will be invalid after a qubit is retrieved from storage for use if no sort of regeneration of the qubit occurs. The no-cloning theorem is just one of many examples of qubit properties that change the way engineers must think about information storage while developing quantum designs and performing compilation procedures. Quantum information is highly sensitive and cannot be copied, and this characteristic must be taken into consideration whenever transforming a quantum algorithm into a QC executable.

3.2. Reversible Logic

Reversible logic is a type of logic where information can travel bidirectionally without loss. In a reversible circuit, a combination of inputs is sent through a function, $f_{REV}(x_1, x_2, ..., x_n)$, to produce a set of outputs, $[y_1, y_2, ..., y_m]$. If these outputs are sent back through the inverse of the function, $f_{REV}^{-1}(y_1, y_2, ..., y_m)$, the original inputs, $[x_1, x_2, ..., x_n]$, are generated. Boolean operations such as the AND and OR operations are not inherently reversible. The truth tables and symbols for these operations can be found in Fig. 3.2. It is apparent while examining these Boolean functions that the outputs are easily derived from the inputs, but information cannot travel in the reverse direction with the same clarity. For example, the output for the AND function is simple to derive with a set of inputs. The function is $x_2 \cdot x_1 = 0$ whenever the input variables are $x_2x_1 = 00, 01, 10$ and is $x_2 \cdot x_1 = 1$ whenever $x_2x_1 = 11$. Since there are three possible combinations of inputs that allow the AND operation to equal zero, however, there is no way to know with certainty what combination is present at the input of the gate if only the output of zero is known. This is caused by an irretrievable loss of information that occurs as soon as the input signals produce an output from the Boolean logic gate. The loss of information is directly related to total energy dissipated by the circuit according to Landauer's principle where each bit of erased data costs at minimum $kT \ln(2)$ Joules in energy dissipation (Landauer, 1961). Therefore, as a computer loses less information during calculations, it improves in efficiency.

According to the work in (Bennett, 1973), any irreversible function can be made reversible without drastically increasing its spatial and temporal complexity during computation. A motivating reason to convert functions into a reversible form would be to minimize the amount of required power during operation if a physically reversible medium is available. To make an arbitrary switching function a reversible function, it must be transformed into a bijection in which it displays the characteristics of being one-to-one and onto (Fazel et al., 2007). When a function is one-to-one, each element in the codomain, or the target set of the function, is the image of at most one element in the domain, or the set of input argument values. When a function is onto, each element in the codomain is the image of at least one element in the domain.



x_2	x_1	$x_2 \cdot x_1$
0	0	0
0	1	0
1	0	0
1	1	1

x_2	x_1	$x_2 + x_1$
0	0	0
0	1	1
1	0	1
1	1	1

Figure 3.2. Boolean AND and OR operation symbols and truth tables

Although reversible logic was a concept that existed before quantum logic was popularized, reversible forms of classical circuits could be thought of as a subset of quantum circuits as all quantum circuits are inherently reversible. This is caused by unitary operators where $\mathbf{U}^\dagger = \mathbf{U}^{-1}$. Because of this property, a quantum transformation followed by its adjoint on a vector of qubits results in the original quantum state since $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$. For example, since $\mathbf{CNOT}$ is self-adjoint where $\mathbf{CNOT} = \mathbf{CNOT}^{-1}$, so

$$\mathbf{CNOT}^2 \left|10\right\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \left|10\right\rangle .$$

Algorithms exist that transform Boolean logic functions into a reversible form. This process requires the addition of ancilla input values and garbage output values since the amount of inputs and outputs must be equal in a reversible logic function. Once Boolean functions are reversible, they may be compiled into circuits that are executable on a quantum machine. An example of a reversible logic generator is the work discussed in (Fazel et al., 2007). This algorithm inputs a Boolean logic function in its exclusive sum of products, ESOP, form and converts it to a reversible Toffoli cascade. Another example of a reversible logic synthesis tool is RevKit (Soeken et al., 2012). This program includes ESOP transformation algorithms based on the work first described in (Fazel et al., 2007) as well as decision diagram based reversible logic synthesis techniques. Recent work has resulted in a methods that transform irreversible switching functions into reversible forms with a minimal number of ancillary information carriers (Gabrielson and Thornton, 2018a,b).

3.3. Gate Libraries and Coupling Constraints

There are various architectures for qubit representation currently competing to become the standardized quantum computing platform. This variety between machines, even between those using the same underlying technology, causes conflicts during the circuit design process. For example, several different transmon-based QCs have been developed by the companies IBM and Rigetti. Although these implementations represent quantum information with transmon technology, differences in architecture topology and gate libraries create compatibility issues which, in many cases, prohibit the use of designs originally mapped for one QC to properly run on another device. This challenge motivates the development of techniques to automatically decompose, map, and optimize quantum circuits to forms that are technologically dependent and physically executable.

Quantum devices have a small set of gates, often ones that are limited to single- and two-qubit transformations, that are physically executable. Because quantum algorithms are usually specified using high-level, multi-qubit operations, they must be simplified into primitive operations that are available in a gate library if the functions are to be physically realized on a real machine. Additionally, although current QCs contain a modest number of physical qubits, connections between qubits on these devices are sparse. Because of nearest neighbor coupling constraints, not all qubit combinations are going to be available for multi-qubit operations. Therefore, special techniques are required to map the decomposed operations of a generalized quantum circuit so that multi-qubit operations are executable on an architecture.

3.4. Current Physical Quantum Technology

3.4.1. IBM Q

IBM has developed QCs based on solid-state, superconducting circuit technology, and quantum information is realized with the charge-based transmon qubit (Chow et al., 2014a; Córcoles et al., 2015; Takita et al., 2017). The company has developed real quantum machines and a quantum simulator that the public can access and perform experiments on. The Python software development kit (SDK) *Qiskit* is used to implement QIP with their platform.

Circuits targeted for the IBM machines must consist of single-qubit operators that are within the gate set of $\mathbf{R_z}(\phi)$, $\mathbf{R_x}(\theta)$, $\mathbf{R_y}(\gamma)$. This includes the common transformations of Identity, Pauli-**X**, Pauli-**Y**, Pauli-**Z**, Hadamard (**H**), Phase (**S**), Phase† ($\mathbf{S}^\dagger$), $\pi/8$ (**T**), $\pi/8^\dagger$ ($\mathbf{T}^\dagger$), phase rotation (ϕ angle on Bloch sphere), and amplitude rotation (θ angle on Bloch sphere). The **CNOT** gate is the only available two-qubit gate on the IBM QCs, and its implementation is restricted to a specific coupling map that is set by the connectivity of the transmons on the device. The coupling map prevents arbitrary **CNOT** placement. Generalized quantum circuits must be redesigned so that **CNOT** gates are mapped to connected qubits. The coupling maps for the public IBM devices can be represented as dictionaries where `device` $= \{a_0 : [b_0], a_1 : [b_1], \ldots, a_{n-1} : [b_{n-1}]\}$. In these dictionaries, the keywords, a_i, are qubits that can act as **CNOT** controls and the paired list, b_i, indicate the qubit(s) that the **CNOT** control can target (IBM Q team, 2018a,b,c,d,e):

- `ibmqx2` = {0:[1,2], 1:[2], 3:[2,4], 4:[2]}
- `ibmqx3` ={0:[1], 1: [2], 2:[3], 3:[14], 4:[3,5], 6:[7,11], 7:[10], 8:[7], 9:[8,10], 11:[10],

12:[5,11,13], 13:[4,14], 15:[0,14]}

- `ibmqx4` = {1:[0], 2:[0,1], 3:[2,4] 4:[2]}
- `ibmqx5`= {1:[0,2], 2:[3], 3:[4,14], 5:[4], 6:[5,7,11], 7:[10], 8:[7], 9:[8,10], 11:[10], 12:[5,11,13], 13:[4,14], 15:[0,2,14]}
- `ibmq_16` ={1:[0,2], 2:[3], 4:[3,10], 5:[4,6,9], 6:[8], 7:[8], 9:[8,10], 11:[3,10,12], 12:[2], 13:[1,12]}

The IBM quantum simulator includes additional gates and qubits that are unrestricted by a coupling map. Information about select IBM machines can be found in Table 3.1. This table contains information about QC name, date of release, capacity, and coupling complexity.

QCs differ in size and layout, and these variations determine the extent that a general circuit must be modified for realization on a particular machine. To give designers better insight to the available qubit connections within a machine, a metric referred to as the "coupling complexity" was devised. The term complexity was chosen to describe the amount of topological interconnects between qubits on a QC. In Table 3.1, coupling complexity is the ratio of the number of allowable **CNOT** couplings found in the coupling map to the total number of two-qubit permutations for an IBM quantum machine calculated as

$$\text{IBM coup. complex.} = \frac{\text{total available couplings}}{\frac{q!}{(q-2)!}} \tag{3.1}$$

where q is the number of physical qubits on the machine. For example, the `ibmqx2` machine has 6 couplings on the coupling map and a total of 20 two-qubit permutations. Therefore,

$$\texttt{ibmqx2}\text{ coup. complex.} = \frac{\text{6 available couplings}}{\text{20 total coupling permutations}} = 0.3.$$

A coupling complexity close to one indicates that a high percentage of a quantum machine's qubits are coupled for arbitrary two-qubit **CNOT** operations. A coupling complexity close to zero indicates a low percentage of coupled qubits.

Table 3.1. IBM Q device details (* indicates a retired device) (IBM Q team, 2018a,b,c,d,e)

Name	**Release Date**	**Qubits Supported**	**Coupling Complexity**
ibmqx2 (Yorktown)	Jan. 2017	5	0.3
ibmqx3*	June 2017	16	0.0833
ibmqx4 (Tenerife)	Sept. 2017	5	0.3
ibmqx5* (Rueschlikon)	Sept. 2017	16	0.0917
ibmq_16 (Melbourne)	Sept. 2018	14	0.0989

3.4.2. Rigetti

Rigetti has developed three solid-state-circuit-based QCs named Agave, Aspen, and Acorn (Didier et al., 2018; Otterbach et al., 2017; Reagor et al., 2018). The Python library PyQuil as well as a software development kit (SDK) *Forest* are used for writing quantum algorithms, interacting with the quantum devices, and simulating quantum computing (Rigetti Computing, 2019b). To specify an algorithm for the Rigetti QCs, quantum operators must be specified in quantum instruction language, or Quil (Smith et al., 2016). Although Quil allows for the specification of algorithms using many of the standard quantum gates, for algorithm to be executable on a real quantum processing unit (QPU), a native gate set must be used. This set includes $\mathbf{R_z}(\phi)$ rotations, $\mathbf{R_x}(\theta)$ rotations rotations that are integer-multiples of $\pi/2$, and **CZ** (Rigetti Computing, 2019b).

Not every qubit pair can couple on a QPU, and this severely limits the total number of executable multi-qubit operations. The Rigetti devices demonstrate this operational constraint because the only available two-qubit gate, **CZ**, may only be implemented on adjacent qubits that are connected. Therefore, the device topology prevents the execution of arbitrary **CZ** operations in an algorithm. The device topology for the Rigetti QPUs can be represented as dictionaries where `device` $= \{a_0 : [b_0], a_1 : [b_1], \ldots, a_{n-1} : [b_{n-1}]\}$. In these dictionaries, the keywords, a_i, are qubits that can act as **CZ** controls and the paired list, b_i, indicates which qubit(s) that the **CZ** control can target (Rigetti Computing, 2019a):

- `Agave` = { 0:[1,7], 1:[0,2], 2:[1,3], 3:[2,4], 4:[3,5], 5:[4,6],6:[5,7], 7:[0,6] }
- `Aspen` ={ 0:[1,7], 1:[0,2,16], 2:[1,3,15], 3:[2,4], 4:[3,5], 5:[4,6], 6:[5,7], 7:[0,6], 10:[11,17], 11:[10,12], 12:[11,13], 13:[12,14], 14:[13,15], 15:[2,14,16], 16:[1,15,17], 17:[10,16] }
- `Acorn` = {0:[5,6], 1:[6,7], 2:[7,8], 4:[9], 5:[0,10], 6:[0,1,11], 7:[1,2,12], 8:[2,13], 9:[4,14], 10:[5,15,16], 11:[6,16,17], 12:[7,17,18], 13:[8,18,19], 14:[9,19], 15:[10], 16:[10,11], 17:[11,12], 18:[12,13], 19:[13,14] }

Information about the Rigetti machines that details QC name, date of release, capacity, and coupling complexity can be found in Table 3.2. An interesting observation is that the **CZ** gate is bidirectional in the sense that the transformation matrix is equivalent if the control and target qubits are interchanged. Because of the bidirectional nature of the **CZ** gate, Rigetti coupling complexity is calculated with combinations rather than permutations as

$$\text{Rigetti coup. complex.} = \frac{\text{total available couplings}}{\frac{q!}{2(q-2)!}}. \tag{3.2}$$

3.4.3. Quantum with Photonic Devices

Photons are a great medium for representing qubit state. They are stable particles in the sense that they do not couple easily with their environment. Additionally, photons are not spatially stationary particles. Since they resist decoherence, photons retain quantum information for long periods of time at room temperature. This property makes them suitable

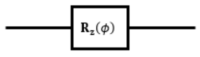
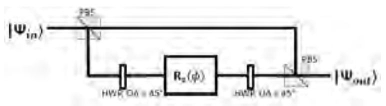
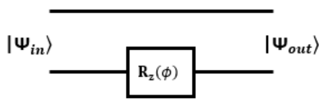
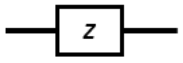
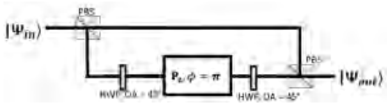
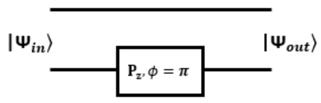
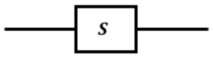
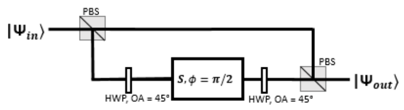
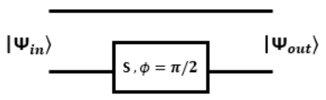
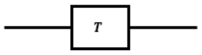
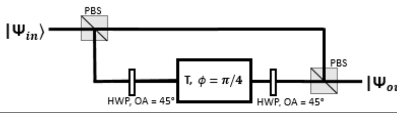
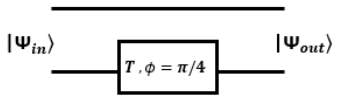
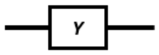
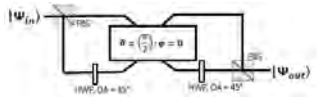
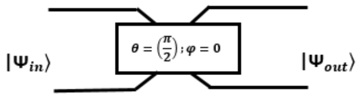
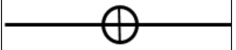
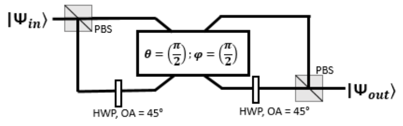
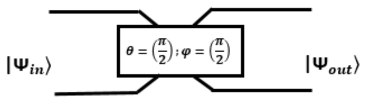
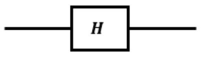
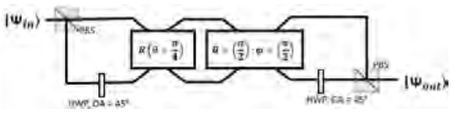
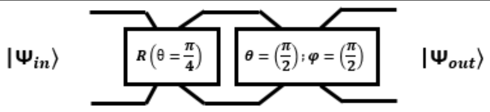
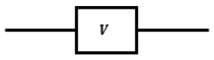
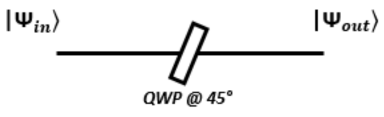
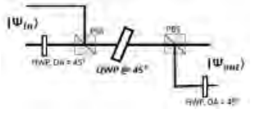
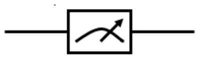
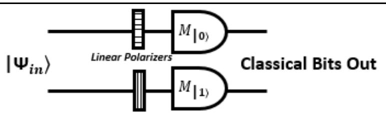
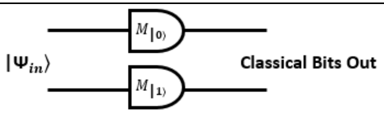
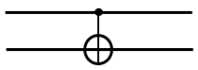
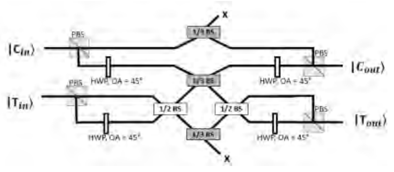
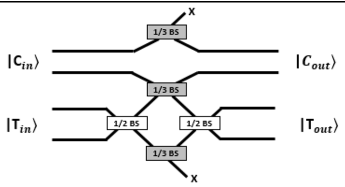
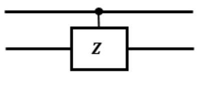
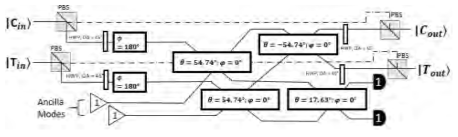
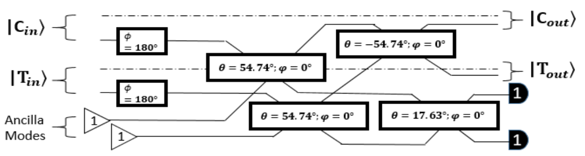
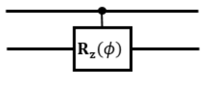
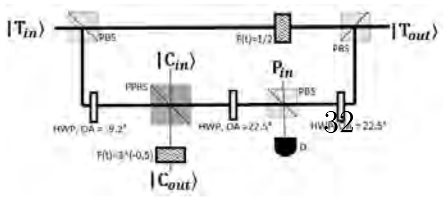
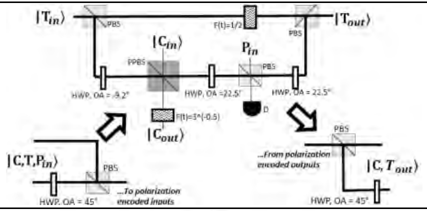
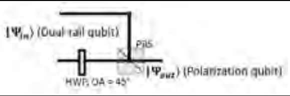
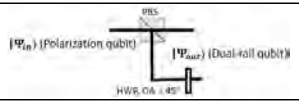
Table 3.2. Rigetti device details (* indicates a retired device) (Rigetti Computing, 2019a)

Name	**Release Date**	**Qubits Supported**	**Coupling Complexity**
Aspen	Nov. 2018	16	0.15
Agave*	June 2018	8	0.2857
Acorn*	Dec. 2017	19	0.1345

as flying qubits. The photon's resistance to decoherence, however, causes the implementation of multi-qubit gates to be difficult. Because photons fail to easily interact with each other, multi-qubit gates act in a probabilistic rather than in a deterministic manner.

A popular methodology for implementing a photonic universal quantum computer is with the KLM protocol (Knill et al., 2001). This protocol uses linear photonics which are devices that transform light in a linear fashion. Examples of linear photonic devices are lenses, mirrors, wave plates, phase shifters and beam splitters. Examples of nonlinear photonic devices include materials that demonstrate the higher-ordered Kerr effect where refractive index of a material is a function of the applied electric field. Because linear devices are used, multi-qubit operations have probabilistic outputs. If a KLM protocol quantum computer is used, it is critical to incorporate error detection and correction in order to repair data after multi-qubit interactions occur. Photonic qubits can either be polarization encoded or dual-rail encoded with the KLM protocol. Table 3.3 has descriptions of photonic quantum operators that operate according to KLM protocol. These components are constructed using elements such as wave plates, beam splitters, and phase shifters. This table of photonic operator implementations, however, is not all inclusive; alternative realizations for photonic gates also exist. Table 3.3 was formed using the information from references (Knill et al., 2001; Knill, 2002; Lemr et al., 2015; Myers and Laflamme, 2006; Nielsen and Chuang, 2010; O'Brien et al., 2003).

Table 3.3. Photonic quantum operators

Operator	**Symbol**	**Polarization-encoded Implementation**	**Dual-rail Implementation**
Z-axis Rotation (Phase Shift) $\mathbf{R_z(\phi)} = \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix}$ **Deterministic**			
Pauli-Z $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ **Deterministic**			
Phase (S) $S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ **Deterministic**			
π/8 (T) $T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$ **Deterministic**			
Pauli-Y $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ **Deterministic**			
NOT (Pauli-X) $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ **Deterministic**			
Hadamard $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ **Deterministic**			
$\sqrt{NOT}$ $V = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}$ $V^{\dagger} = \frac{1}{2}\begin{bmatrix} 1-i & 1+i \\ 1+i & 1-i \end{bmatrix}$ **Deterministic**			
Measurement $M_{\vert 0\rangle} = [1 \quad 0]$ $M_{\vert 1\rangle} = [0 \quad 1]$ **Deterministic**			
CNOT *CNOT=* $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ **Probabilistic, P = 1/9**			
Controlled-Z *CZ=* $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$ **Probabilistic, P = 1/16**			
Tunable Controlled Phase Gate $CR_z(\phi)=$ $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix}$ **Probabilistic, P = 1/48**			
Implementation Conversion			

3.5. Quantum Cost

Whenever engineers think of cost, usually measures for reducing power consumption, delay, and area of a circuit come to mind. Classical computation has advanced to the point where certain parameters can be tuned during the design process to optimize one or more of these metrics. It is often the case, however, that all three of these circuit characteristics cannot be improved simultaneously because measures made to improve one property can negatively impact another.

Quantum engineers are still working towards building a reliable and scalable QC during the NISQ era, so designers have less freedom with implementation parameters. Since the main goal for researchers is to allow quantum algorithms to run on physical devices, the reduction of instances where qubit state could decohere is of high importance. With current implementations, quantum state eventually decoheres after time, but its liklihood of decoherence increases as a qubit undergoes more transformations. Additionally, circuit depth and gate volume is a concern as devices have limited execution times and must be periodically recalibrated. Since each transformation requires a finite amount of time, whenever thinking about reducing cost for a quantum circuit, reducing the total number of operations performed on a qubit, especially those that implement multi-qubit operations, is of high priority. All arbitrary n-qubit gates are capible of being decomposed into the set of all single qubit gates as well as the **CNOT** or **CZ** gate (Barenco et al., 1995). Because of this, one can conclude that an important measurement of cost for quantum circuit is the total number of multi-qubit operations it requires.

Cost functions need to be tunable during quantum logic synthesis and compilation so that key circuit features can be optimized. It is expected that each particular technologically-dependent quantum cell library will be characterized and annotated with custom cost functions for use during synthesis depending on if metrics such as qubit fidelity, operator fidelity, or decoherence times are of focus. In this work, the quantum cost function for the IBM back-end was defined as

$$q_{cost} = 0.5 \times t + 0.25 \times c + a \tag{3.3}$$

however optimization methods allow for any cost function to be used.
In Eqn. 3.3, t is the count of all $\mathbf{T}$ and $\mathbf{T}^\dagger$ gates, c is the count of **CNOT** gates, and a is the total gate count, or gate volume, for a circuit. **T** gates are given an additional cost of 0.5 as compared to all other single qubit gates because of the operator's poor fidelity as compared to other single qubit operators in fault tolerant quantum implementations (Amy et al., 2014). **CNOT** gates are given a cost that is 0.25 more than single qubit gates, with the exception of **T**, because two-qubit operations for the transmon are characterized by higher error rates as compared to the other single qubit operations (Chow et al., 2014b). The IBM cost function was selected based upon what is commonly seen in the literature: fewer gates usually leads to smaller circuits with a lower probability for decoherence and fewer **T** gates improves reliability and results in greater fault-tolerance. A larger quantum cost indicates a higher likelihood of qubit decoherence and decreased fault tolerance. Quantum cost for a design is minimized by quantum logic design automation tools during the optimization process.

For the Rigetti back-end,

$$q_{cost} = 5c + 3h + 2y + x + z + s + t \tag{3.4}$$

was chosen as the function for quantum cost. In Eqn. 3.4, c is the count of **CZ** gates, h is the count of **H** gates, y is the count of **Y** gates, x is the count of **X** gates, s is the count of $\mathbf{S}$ and $\mathbf{S}^\dagger$ gates, and t is the count of $\mathbf{T}$ and $\mathbf{T}^\dagger$ gates for the technology-dependent circuit mapping generated by the tool. **X**, **Z**, **S**, and **T** are given the lowest weights in the cost function because these gates can be executed with single $\mathbf{R_x}$ or $\mathbf{R_z}$ gates native to the Rigetti library. **H** and **Y** gates are given weights of 3 and 2, respectively because $\mathbf{H} =$

$\mathbf{R_z}(\pi/2)\mathbf{R_x}(\pi/2)\mathbf{R_z}(\pi/2)$ and $\mathbf{Y} = \mathbf{R_z}(\pi)\mathbf{R_x}(\pi)$ are the transformations used to decompose the single qubit gates into the Rigetti gate set. Finally, **CZ** is given a weight five times that of the gates that are purely **X** and **Z** rotations function because as a two-qubit gate, it has on average longer execution times with a lower fidelity.

3.6. Quantum Multiple-valued Decision Diagrams

An important aspect for technology-dependent quantum logic synthesis is formal verification. In this work, outputs of synthesis and compilation undergo equivalence checking to confirm that all transformations have occurred without introducing additional error. As previously mentioned, n-qubit quantum logic gates or operators are functionally described using unitary matrices of size $2^n \times 2^n$. The size of the transfer matrix describing an entire quantum circuit thus grows exponentially as the number of qubits in the function increases. Data structures have been developed that allow these matrices to be represented in a compact form. For example, the Quantum Multiple-valued Decision Diagram (QMDD) represents quantum transfer matrices efficiently in the form of a directed acyclic graph. The QMDD was first introduced in (Miller and Thornton, 2006) and is further described in (Niemann et al., 2016b).

QMDDs are a collection of nodes, or vertices, and directed edges. Non-terminal vertices represent qubits and have four outgoing edges that serve as one of the four quadrants in a quantum transformation matrix. From left to right, the four edges leaving a non-terminal node represent the sub matrices of $\mathbf{U}_{00}$, $\mathbf{U}_{01}$, $\mathbf{U}_{10}$, and $\mathbf{U}_{11}$ for the quantum transformation matrix $\mathbf{U}$. Since redundancy in the graph is removed and each qubit variable only appears once, the QMDD representing a quantum function becomes compact in size. An example of the **CNOT** operation in the form of a QMDD is shown in Fig. 3.3. Here, x_0 and x_1 are used to encode the binary encoded decimal values for the row and column indices in the operation transformation matrix. Dashed lines are included in this illustration to make submatrix values more apparent. The variable order is $x_0 \rightarrow x_1$, so x_0 acts as the initial vertex and x_1 vertices appear afterwards along the path for submatrices that are not equal to the constant zero.

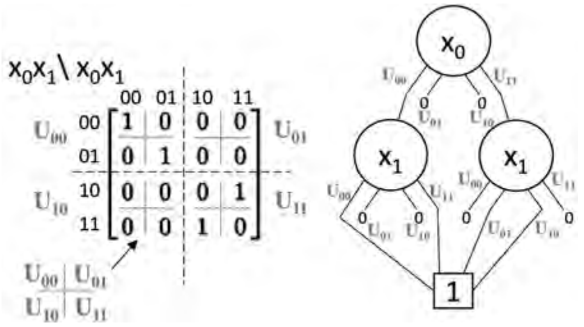


Figure 3.3. Representation of **CNOT** operation as a QMDD.

The QMDD representation of a quantum function is canonical with respect to a fixed variable order due to the reduction rules described in (Miller and Thornton, 2006). Even if two circuits with the same transformation matrix are described using different operators, their QMDDs will be equal as long as the variable order used to construct the QMDDs is identical. This concept is used to perform formal verification within the automated quantum logic synthesis tool. An equivalence check between the original technology-independent quantum circuit and the resulting technology-dependent mapped circuit is ensured by requiring the QMDDs to match. If the realizations are indeed identical, the reduction rules for the QMDDs will cause the two designs to share the same graph in memory. This is important in the current era of NISQ devices since errors can be attributed to decoherence since the executable has been formally verified.

3.7. Zero-supressed Decision Diagrams

Zero-supressed decision diagrams (ZDDs) are excellent data structures for representing sparse sets. Since coupling constraints force limited connectivity between qubits on a device, it was hypothesized that ZDDs could effectively hold the sparse set that represent valid mapping permutations of algorithm qubits to physical qubits.

Given a set of variables $X = \{x_1, \dots, x_n\}$, a ZDD is a directed acyclic graph with nonterminal vertices N and two terminal vertices $\top$ and $\bot$ (Knuth, 2011; Minato, 1993). Each non-terminal vertex $v \in N$ is associated with a variable $V(v) \in \{1, \dots, n\}$ and two successor nodes $HI(v), LO(v) \in N \cup \{\top, \bot\}$. The nodes on a path follow a fixed variable order on the way to a terminal node. There exists $HI(v) \in \{\top, \bot\}$ or $V(HI(v)) > V(v)$ for all v.[1] The same applies to $LO(v)$.

Each vertex in the ZDD represents a finite family of finite subsets over X where families of sets are canonical up to order of the sets and repetition. The terminal node $\bot$ represents the empty family, $\emptyset$, and the terminal node $\top$ represents the unit family which is the set containing the empty set $\{\emptyset\}$. Each non-terminal v represents the subset

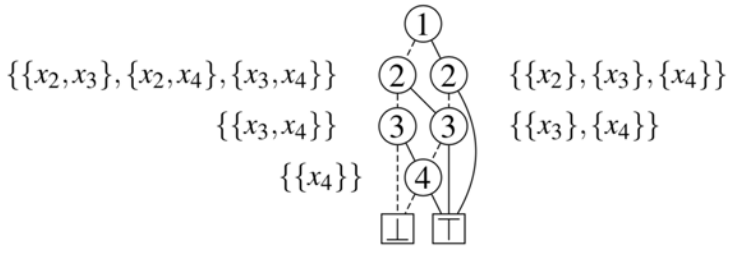
$$LO(v) \cup \{S \cup \{x_{V(v)}\} \mid S \in HI(v)\}. \tag{3.5}$$

A ZDD is reduced if there are no two vertices that represent the same sets. This implies that in a reduced ZDD there cannot be a vertex v with $HI(v) = \bot$, since such a vertex represents the set $LO(v)$. For the sake of convenience, ϵ_x denotes the elementary family $\{\{x\}\}$ for each $x \in X$. Finally, $\wp$ refers to the ZDD that represents the universal family of all subsets of X.

$|f|$ denotes the number of sets in a family f. $Z(f)$ denotes the number of nodes, including the terminal nodes, of the reduced ZDD for f. It should be noted that because of the reduction rules associated with ZDDs, the data structure is a canonical representation of a function with respect to a fixed variable order.

Given two ZDDs f and g, the following list of operations is part of what is called a ZDD

[1]To simplify the presentation, it is assumed the variable ordering $1 < 2 < \cdots < n$. In practice, any permutation of this order can be used.



$\{\{x_2,x_3\},\{x_2,x_4\},\{x_3,x_4\}\}$
$\{\{x_3,x_4\}\}$
$\{\{x_4\}\}$
1
2
2
3
3
4
⊥
⊤
$\{\{x_2\},\{x_3\},\{x_4\}\}$
$\{\{x_3\},\{x_4\}\}$

Figure 3.4. A ZDD representing the family of sets $\{\{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_4\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_3, x_4\}\}$. All internal non-terminal nodes are annotated with the sets they represent. Dashed edges indicate LO and solid edges indicate HI.

family algebra. Each operation can be efficiently implemented using ZDDs.

$$f \cup g = \{\alpha \mid \alpha \in f \text{ or } \alpha \in g\} \qquad \textit{union}$$

$$f \cap g = \{\alpha \mid \alpha \in f \text{ and } \alpha \in g\} \qquad \textit{intersection}$$

$$f \setminus g = \{\alpha \mid \alpha \in f \text{ and } \alpha \notin g\} \qquad \textit{difference}$$

$$f \sqcup g = \{\alpha \cup \beta \mid \alpha \in f \text{ and } \beta \in g\} \qquad \textit{join}$$

$$f \sqcap g = \{\alpha \cap \beta \mid \alpha \in f \text{ and } \beta \in g\} \qquad \textit{meet}$$

$$f \searrow g = \{\alpha \in f \mid \beta \in g \text{ implies } \alpha \not\supseteq \beta\} \qquad \textit{nonsupersets}$$

Finally, if f represents the family $\epsilon_{x'_1} \cup \cdots \cup \epsilon_{x'_l}$ for some subset $\{x'_1, \ldots, x'_l\} = X' \subseteq X$, then

$$\binom{f}{k}$$

is the ZDD that represents the family $\binom{X'}{k}$.

Note that the nonsupersets operation can be described in terms of the others: $f \searrow g = f \setminus (f \sqcup g)$. However, it may be more efficient to implement the operation explicitly in a ZDD package. For a detailed description of how ZDDs are represented in memory and how the ZDD family algebra operations are implemented, refer to the literature (Knuth, 2011).

Chapter 4

Technology Mapping Algorithms

4.1. Connectivity Tree Reroute

A significant drawback of solid-state QCs is that the stationary qubits are limited to certain multi-qubit operations, like those described in coupling maps, due to the layout and properties of the physical device. This limitation, however, extends to other implementations because qubits must be physically connected, in close proximity, and have the appropriate operational characteristics to realize a multi-qubit operation such as the **CNOT** or other controlled operations in any physical realization. For this reason, a generic reroute algorithm capable of implementing multi-qubit operations on uncoupled qubits for any architecture specified by a coupling map is essential for the technology-dependent quantum logic synthesis tool described here. Tree data structures assist in finding the shortest **SWAP** route for **CNOT** execution.

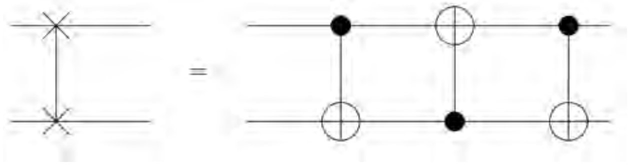


Figure 4.1. Implementation of **SWAP** operation using **CNOT**.

The connectivity tree reroute algorithm (CTR) was implemented in the quantum synthesis tool to automatically reroute **CNOT** operations that are not supported by a coupling map. In this method, a tree structure based on the coupling map for the selected QC determines the shortest **SWAP** path that the control qubit travels to reposition for a **CNOT** operation. As shown in Fig. 4.1, a **SWAP** is implemented with **CNOT** gates among physically connected qubits, as indicated by the coupling map, causing the interchange of quantum information. **SWAP** operations continue to move the control qubit until the desired **CNOT** operation can execute on the specified target. After the **CNOT** operation executes, the control qubit traverses the **SWAP** path in reverse to return to its original position in order to preserve the original assignment of qubits in the circuit.

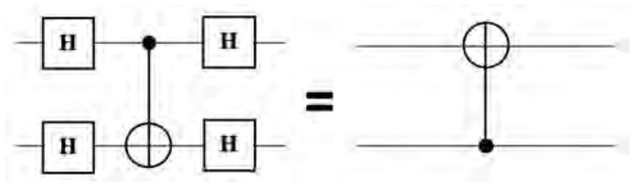
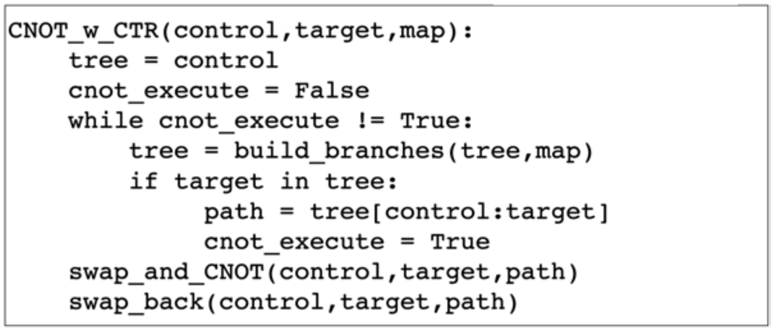


Figure 4.2. **CNOT** orientation reversal.

To find the **SWAP** path, CTR builds a tree data structure. The tree root node is the control qubit, and edges leading to other qubit nodes are generated according to the available coupling map configurations. Because of the transformation in Fig. 4.2, direction of the natively available **CNOT** operation does not matter when building the connectivity tree. In other words, if a qubit $|\psi_0\rangle$ can act as either a control or target in a **CNOT** operation with qubit $|\psi_1\rangle$, the two qubits will be connected with an edge to form a potential **SWAP** path. The tree describes all possible paths that the qubit can take, until the shortest path to a position coupled with the target qubit is found. If a node is reached that is already represented in the tree, the branch is terminated. Pseudocode describing the CTR algorithm is found in Fig. 4.3.

An example of the CTR algorithm in action can be seen in Fig. 4.4. In this illustration, the desired operation is a **CNOT** with q5 as the control and q10 as the target on the 16-qubit `ibmqx3` machine. According to the `ibmqx3` coupling map, q5 and q10 cannot natively perform a **CNOT** operation together. After implementing CTR, however, the operation is performed after two **SWAP** operations of first, q5 with q12 and then second, q12 with q11.



```
CNOT_w_CTR(control,target,map):
    tree = control
    cnot_execute = False
    while cnot_execute != True:
        tree = build_branches(tree,map)
        if target in tree:
             path = tree[control:target]
             cnot_execute = True
    swap_and_CNOT(control,target,path)
    swap_back(control,target,path)
```

Figure 4.3. Pseudocode CTR algorithm.

Since a connection between q11 and q10 exists on the coupling map, q11 acts as a control for q10. After the desired **CNOT** operation executes, the information of the control qubit transitions back to its original position, q5, on the QC.

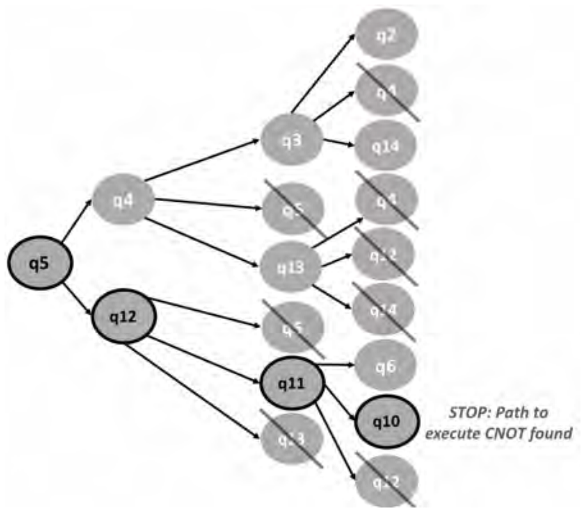


Figure 4.4. CTR implementation on the `ibmqx3` machine for a **CNOT** with q5 as control and q10 as target.

4.2. Zero-suppressed Decision Diagram Technology Mapping

Another method to tackle the problem of having limited connectivity on a quantum machine involves assigning the qubits in a quantum circuit to their ideal placement on a chip before performing any gate decomposition or transformation processes. Finding this ideal assignment permutation, however, is an intractable problem because of the large combinational search space. Zero-surpressed decision diagrams (ZDDs) can be used in mapping algorithms to combat combinational complexity as they are efficient at representing sparse sets.

4.2.1. Problem Formulation with ZDD Mapping

The quantum circuit that will be mapped to a quantum device is modeled as a set of pseudo qubits $V = \{v_1, \dots, v_n\}$ and an ordered sequence of two-qubit gates $G_j = g_1, \dots, g_n$, with $g_i \in \binom{V}{2}$. The one-qubit gates in the circuit can be ignored in this case since the coupling constraints of the device do not affect their mapping. As a note the direction of a gate (e.g., the position of control and target in a **CNOT**) is not taken into account as unidirectional gates may be reversed, as seen in Fig. 4.2, with single qubit operations.

A quantum device is modeled by an undirected graph (P, E), where $P = \{p_1, \dots, p_m\}$ is a set of physical qubits and an edge $\{p, q\} \in E \subseteq \binom{P}{2}$ states that a 2-qubit operation can be executed using the two physical qubits p and q. The goal is to find a mapping $\varphi : V \to P$ of pseudo qubits into physical ones, such that all two-qubit operations in the circuit are executed on adjacent qubits according to the device's coupling constraints. It may not be possible to find such mapping for the input circuit. However, by adding **SWAP** gates to reorder pseudo qubits, a mapping can be achieved. A **SWAP** gate is a two-qubit operation that can either be implemented with **CNOT** or single-qubit rotations and **CZ**.

The aim is to use a small number of **SWAP** gates when transforming an initial circuit to a circuit that can be mapped into a target device. Finding the globally optimum mapping and a transformed circuit using the fewest number of **SWAP** gates is a computationally complex and time-consuming task (Botea et al., 2018). To address the problem of finding maximal partitions for a circuit using ZDDs, partitions are used where

- there exists a (V, G_j) that is a subgraph of (P, E) for $1 \leq j \leq l$,
- there is not a $(V, G_j \cup \{g_{e_j+1}\})$ that is a subgraph of (P, E) for $1 \leq j < l$.

Each partition has an associated set of mappings of pseudo to physical qubits $\Phi_j = \{\varphi : V \mapsto P\}$ where φ is a subgraph isomorphism of (V, G_j) to (P, E). If partition G_j starting at gate g_b cannot be extended, **SWAP** operations are inserted to merge the last gate of the partition, g_e with the adjacent gate, g_{e+1}, in the circuit. These **SWAP** operations, referred to as *layers*, exchange information on the adjacent physical qubits of the device and are executable in parallel within a single time cycle. The best **SWAP** layer is chosen according to a scoring metric. Once selected, the **SWAP** layer merges the gate g_i with g_{i+1} by inserting **SWAP** gates before g_{i+1}, extending G_j.

Ideally, a single partition will cover the entire circuit, providing a set of mappings that assign pseudo qubits to physical qubits on the device. In the case that multiple partitions exist that are fully extended with inserted **SWAP** *layers*, a mapping for the circuit is selected using the largest, and therefore maximal, partition. A maximal partition within a quantum circuit is the largest subset of gates G_j that can be covered by a set of mappings of pseudo to physical qubits, Φ_j. Maximal partitions are fully expanded with parallel **SWAP** operations that exchange information between physical qubits.

4.2.2. Finding Maximal Partitions

In this section, how to use ZDDs and ZDD operations to find a maximal partition that starts in some gate g_i is described. The ZDDs are defined over the nm variables vp for each $v \in V$ and each $p \in P$. Each ZDD represents a family of finite subsets, and each subset α represents a partial mapping $\varphi : V \to P$, where $\varphi(v) = p$, if and only if $vp \in \alpha$ where α is a mapping.

First, some general sets are defined, which are used throughout the following operations. It is sufficient to initialize these sets once at the beginning of the algorithm. The set

$$from(v) = \bigcup_{p \in P} \epsilon_{vp} \tag{4.1}$$

contains all singleton mappings $v \mapsto p$ for some $v \in V$. Analogously, the set

$$to(p) = \bigcup_{v \in V} \epsilon_{vp} \tag{4.2}$$

contains all singleton mappings $v \mapsto p$ for some $p \in P$. Using the set in Eqn. 4.2, two other helpful sets, *valid* and *bad*, are defined using the ZDD family algebra operations. The set *valid* contains all two-element partial mappings that are feasible with respect to the coupling constraints of the device such that

$$valid = \bigcup_{\{p,q\} \in E} to(p) \sqcup to(q). \tag{4.3}$$

The set *bad* contains all two element sets of illegal partial mappings, because they either contain an element with two images or two elements which map to the same image as

$$bad = \bigcup_{v \in V} \binom{from(v)}{2} \cup \bigcup_{p \in P} \binom{to(p)}{2}. \tag{4.4}$$

Last, the set *map*(i) is defined to represent all possible mappings of the pseudo qubits in gate $g_i = \{v, w\}$ as

$$map(i) = (from(v) \sqcup from(w)) \cap valid. \tag{4.5}$$

In other words, all possible mappings of v with all possible mappings of w are joined before the resulting two-element subsets to those which are valid with respect to the target device are restricted. Finally, the possible mappings of two consecutive gates g_i and g_{i+1} can be computed using

$$(map(i) \sqcup map(i+1)) \searrow bad. \tag{4.6}$$

In some instances, Eqn. 4.6 results in the empty set, $\emptyset$, whenever the mappings of two consecutive gates are combined. In this case, **SWAP** procedures must be performed on physical qubits to exchange pseudo qubit information and extend the partition G_j.

```
Data: Gate sequence $g_1, \ldots, g_k$, and device $(P, E)$
Result: partitions $G_j$ with begin and end indexes $b_j, e_j$; all possible mappings $\Phi_j$
Set $j \leftarrow 1, b_j \leftarrow 1, m \leftarrow map(1)$;
for $i = 2, \ldots, k$ do
    Set $m' \leftarrow (m \sqcup map(i)) \setminus bad$;
    if $m' = \emptyset$ then
        for layer in layers do
            calculate scores;
        end
        if max score $\neq 0$ then
            insert SWAP circuit;
            update topology;
            Set $m(\text{max score})' \leftarrow (m \sqcup map(i)) \setminus bad$;
            Set $m \leftarrow m'$;
        else
            Set $e_j \leftarrow i - 1, \Phi_j \leftarrow m$;
            Set $j \leftarrow j + 1, b_j \leftarrow i, m \leftarrow map(i)$;
        end
    else
        Set $m \leftarrow m'$;
    end
end
Set $e_j \leftarrow k, \Phi_j \leftarrow m$;
```

Figure 4.5. Algorithm: Find maximal partitions.

Finding an ideal **SWAP** circuit during quantum circuit mapping is an intractable problem that works in an exponentially-growing state space. In this method, the search space is narrowed by focusing on the implementation of all **SWAP** circuits that can be executed in parallel during a single time cycle. This particular set of **SWAP** circuits is called *layers*, and it is a combination set rather than permutation set of ordered **SWAP** operations. ZDDs are a good data structure for combinatorial set representation, so a ZDD is implemented to represent the *layers* set. Acceptable **SWAP** operations are determined by the topology of the device, and the sets of operations that can be executed simultaneously within a time cycle are desired. ZDDs are used to create a set of all "good" **SWAP** circuits, which are those that interact with at least one qubit in the image of φ and the depth of the circuit is one. In other words, a **SWAP** circuit may only contain multiple **SWAP** gates as long as qubits between gates are not shared. The ZDDs for this task differ from those used to enumerate all mappings, and they do not share any variables. The **SWAP** circuit ZDDs are defined over the $|E|$ variables e for each $e \in E$ since **SWAP** gates can only be placed on certain edges connecting physical qubits according to the quantum device operational characteristics. The ZDD base, or the 1-terminal node, is initialized with the following ZDD. For each $p \in P$, the ZDD

$$edges(p) = \bigcup \{\epsilon_e \mid e \in E \text{ s.t. } p \in e\} \tag{4.7}$$

contains all **SWAP** operations that interact with qubit p. All possible subsets of **SWAP** gates that can be executed in parallel (i.e., in depth 1) are described by the ZDD

$$layers = \wp \searrow \bigcup_{p \in P} \binom{edges(p)}{2}. \tag{4.8}$$

Not all combinations of **SWAP** gates in *layers* may be useful for extending a partition of gates. For example, some **SWAP** circuits may allow the partition G_j to extend further and have greater depth while others provide more mapping options within Φ_j. As a result, a scoring function is calculated for each member of *layers* to determine the optimal **SWAP** decision every time it is desired to extend a circuit partition. The function of

$$score = (A\alpha + B\beta)\,\frac{\gamma}{C} \tag{4.9}$$

is used to select the optimal **SWAP** circuit according priority weights set by the user and feature counts of the circuit being mapped. In Eqn. 4.9, α is depth weight and A is depth count where depth count in this case describes how many additional two-qubit gates the current partition could cover until the end of the partition is reached if a specific **SWAP** circuit is implemented. The variable β is map weight and B is map count that describes how many maps of pseudo to physical qubits, $\varphi : V \to P$, will be available in Φ_j if a specific **SWAP** circuit is implemented. Finally γ is **SWAP** weight and and C is **SWAP** operation count of the implemented **SWAP** circuit. In Eqn. 4.9, **SWAP** count has an inverse relationship with a layer's *score* as lower overall gate counts, or gate volume, in a technology-mapped implementation are preferred. The parameter γ, however, can be adjusted to make the cost of an additional **SWAP** operations less severe. The weights of this function can be tuned to prioritize the growth of the partition with respect to either gate coverage or available mappings if a **SWAP** circuit layer in *layers* is implemented. In the case that *score* for each layer is zero, the current partition cannot be extended and a new partition must start to continue to map the circuit.

The algorithm in Fig. 4.5 implements ZDD data structures and the aforementioned ZDD operations to compute maximal partitions of quantum circuits starting from the first gate. A counter j indicates the current partition number as the algorithm parses through the operators in the network. In m, a set of mappings are stored and updated as the current partition increases in size. These maps are eventually stored in Φ_j. For each gate i, there is an attempt to extend m by adding the gate using *map*(i), and storing the resulting mappings in m'. If m' is empty, *layers* will be used to determine if a **SWAP** operation can be implemented in order to increase the current partition. The *scores* for all of the sets in *layers* are calculated, and the **SWAP** circuit with the largest *score* is used to extend the partition. After the **SWAP** is implemented, the topology of the device is updated, maximum *score* m' is calculated, and m is updated with m'. If the maximum *score* is zero, then the then the current partition ends at $i - 1$, and a new partition $j + 1$ starts at gate i.

4.2.3. ZDD mapping in the Quantum Compilation Flow

The algorithm discussed in the Section 4.2.2 implements the mapping of pseudo qubits in a quantum circuit specification to physical qubits on a real device. While the mapping procedure is essential for for quantum compilation, additional optimization steps can further improve the technology-mapped logic. For this reason, the incorporation of the ZDD mapping techniques into a larger logic synthesis flow is proposed. In this procedure, mapping would occur after a circuit has been decomposed into one- and two-qubit operators and before a specification is compiled by a device's custom compilation tool. Completing synthesis with available compilation tools allows the opportunity to take advantage of existing optimization algorithms while the operators of the circuit are transformed into a platform's native gate library. Additionally, if the maximal partition found by the ZDD mapper does not cover the entire circuit, the native compiler of the technology platform is required to ensure that the placement of the two-qubit operators does not violate the coupling constraints of the device. It should be noted that although the ZDD mapping algorithm was evaluated using superconducting qubits as a target platform, the techniques described here are applicable to other quantum technologies that have coupling restrictions.

4.2.4. Experimental Results

A subset of benchmarks from (Amy, 2019) were selected to evaluate the ZDD mapping methods. These benchmarks contain a variety of reversible arithmetic and quantum algorithms such as a Grover's algorithm oracle, a demonstration of the Quantum Fourier Transform (QFT), and various Toffoli implementations. The benchmark set was chosen for experimentation because they are functions commonly seen in quantum synthisis literature and are publicly available for use. These benchmarks are originally specified in a .qc file format with a gate set that contains physically unrealizable multi-qubit gates. Thus, the specifications are transformed into the Clifford+T library of single- and two-qubit gates using the "phasefold" pass of the *Feynman* toolkit (Amy, 2019) . After this procedure, the benchmarks are in a technology-independent form that consists of elementary gates. Mapping to a target quantum device may now begin.

A ring topology was chosen as the target device during synthesis. A ring structure was chosen because this type of architecture is seen in commercial QPUs like in Rigetti's `Agave` and `Aspen` quantum machines. Additionally, the benefit of this structure is that it allows for a homogeneous testing environment that is flexible in size while using benchmarks that vary in number of pseudo qubits. Each benchmark was targeted to a device that contained n physical qubits where n is the number of pseudo qubits in a quantum algorithm. In these devices, all qubits are connected to their two adjacent neighbors, and the connections are bidirectional with respect to the placement of the two-qubit **CNOT** gate. Once the circuit and topology are selected, the algorithm of Fig. 4.5 is applied to map the pseudo qubits in the design to physical qubits. The scoring operation of Eqn. 4.9 that chooses between the **SWAP** circuits in *layers* to extend the partition is set to zero, one, and one for the depth, map, and **SWAP** weights, respectively. If the benchmark can be covered by an entire partition during the application of the algorithm of Fig. 4.5, then the resulting specification is in a fully technology-mapped form and therefore compatible with the available connections of the target device. If multiple partitions are needed for a benchmark, the resulting specification is mapped using a permutation of the largest partition, making additional **SWAP** operations required for the design to be fully compatible with the target technology. This additional circuit modification is accomplished by the custom compilers that are provided with the Rigetti and IBM SDKs. Compiling the ZDD mapped circuits into the selected device topology with the available Rigetti and IBM compilers is the final procedure in the mapping flow. This step also ensures that all algorithm operations are translated into gates appropriate for the target device. Note that such a translation cannot lead to any further violations of the coupling constraints. After the final compilation step, the circuits are ready for execution on their respective platform since they are in a technology-mapped and optimized form. Details about the benchmarks along with experimental results of the mapping and compilation procedures can be found in Table 4.1.

In Table 4.1, details about gate depth, gate volume, and two-qubit gate count have been included for the ZDD mapped and compiled circuits. Metrics for the benchmarks after transformation with just the ZDD mapping procedures are also shown for reference. It should be noted that circuits transformed with only the ZDD mapper can include additional **SWAP** circuitry to expand the mapping partitions, and only if the maximal partition covers the entire benchmark is the resulting circuit fully mapped for the target technology. Because it is often the case that multiple partitions cover a benchmark and the maximum partition must be chosen for pseudo to physical qubit mapping, transformation by the native compiler is required. This synthesis procedure also confirms that the benchmark circuits use the appropriate gate library for the targeted technology. The benchmarks were compiled with and without preprocessing the circuit with the ZDD mapper. Circuits that improved in metrics for a particular device and benchmark whenever ZDD mapping was implemented are emphasized. On average, benchmarks mapped to a Rigetti ring topology saw a decrease of around 10% with respect to gate depth, gate volume, and two-qubit gate count whenever ZDD mapping was included in technology-dependent logic synthesis flow before compilation. The IBM-compiled circuits, however, only saw an average decrease of 2.3% in gate depth, gate volume, and two-qubit gate count whenever ZDD mapping was used. Individual improvements in circuit metrics of up to approximately a 50% decrease was seen in gate volume on the Rigetti devices and up to approximately a 44% decrease was seen in depth on the IBM devices. These findings demonstrate the potential that ZDD mapping techniques have with respect to finding more optimal solutions whenever generating technology-dependent forms of quantum circuits.

Table 4.1. Gate depth, gate volume, and two-qubit metrics of benchmarks after zdd mapping, IBM compiling, and Rigetti compiling. Values that decreased whenever ZDD mapping was implemented before compilation have been emphasized.

Benchmark Name	No. Qubits		Original ZDD Mapped	Original Rigetti Compiled	Original IBM Compiled	ZDD Mapped/ Rigetti Compiled	ZDD Mapped/ IBM Compiled
barenco_tof_3	5	*depth:*	64	118	62	98 *(-16.95%)*	84 *(+35.48%)*
		vol.:	95	446	180	221 *(-50.45%)*	165 *(-8.33%)*
		2q gates:	73	68	67	58 *(-14.71%)*	63 *(-5.97%)*
barenco_tof_4	7	*depth:*	94	230	131	155 *(-32.6%)*	130 *(-0.76%)*
		vol:	190	763	462	449 *(-41.15%)*	335 *(-27.49%)*
		2q gates:	152	123	177	117 *(-4.88%)*	132 *(-25.42%)*
barenco_tof_5	9	*depth:*	94	231	121	155 *(-32.9%)*	130 *(+7.44%)*
		vol.:	285	1136	528	682 *(-39.96%)*	505 *(-4.36%)*
		2q gates:	231	184	201	177 *(-3.8%)*	201 *(+0%)*
gf2^4_mult	12	*depth:*	46	337	251	361 *(+7.12%)*	354 *(+41.03%)*
		vol:	232	2319	1450	2593 *(+11.82%)*	1511 *(+4.21%)*
		2q gates:	145	363	557	430 *(+18.46%)*	587 *(+5.39%)*
gf2^5_mult	15	*depth:*	64	422	259	504 *(+19.43%)*	342 *(+32.05%)*
		vol:	363	3747	2212	4510 *(+20.36%)*	2351 *(+6.28%)*
		2q gates:	230	596	842	775 *(+30.03 %)*	910 *(+8.07%)*
grover_5	9	*depth:*	210	968	989	872 *(-9.92%)*	552 *(-44.19%)*
		vol:	777	4857	2909	4484 *(-7.68%)*	2590 *(-10.97%)*
		2q gates:	441	781	1096	739 *(-5.38%)*	1011 *(-7.76%)*
hwb6	7	*depth:*	113	449	269	432 *(-3.79%)*	290 *(+7.81 %)*
		vol:	303	2032	1049	2027 *(-0.25%)*	1101 *(+4.96%)*
		2q gates:	185	332	404	338 *(+1.81%)*	422 *(+4.46%)*
mod_mult _55	9	*depth:*	49	189	123	177 *(-6.35 %)*	144 *(+17.07 %)*
		vol:	155	978	500	850 *(-13.09 %)*	469 *(-6.2%)*
		2q gates:	88	151	193	143 *(-5.3 %)*	176 *(-8.81%)*
mod_5 _4	5	*depth:*	60	115	94	95 *(-17.4%)*	92 *(-2.13%)*
		vol:	121	459	229	308 *(-32.9%)*	239 *(+4.37%)*
		2q gates:	98	73	88	79 *(+8.21%)*	92 *(+4.55%)*
qft _4	5	*depth:*	142	162	155	137 *(-15.43%)*	105 *(-32.26%)*
		vol:	247	447	322	433 *(-3.13%)*	293 *(-9.01%)*
		2q gates:	120	79	126	92 *(+16.46%)*	114 *(-9.52%)*
tof_3	5	*depth:*	39	98	62	72 *(-26.53%)*	61 *(-1.61%)*
		vol.:	75	309	145	195 *(-36.89%)*	135 *(-6.9%)*
		2q gates:	54	47	53	45 *(-4.26%)*	52 *(-1.89%)*
tof _4	7	*depth:*	46	117	98	88 *(-24.79%)*	62 *(-36.73%)*
		vol:	125	505	326	327 *(-35.25%)*	218*(-33.12%)*
		2q gates:	92	80	121	75 *(-6.25%)*	84 *(-30.58%)*
tof _5	9	*depth:*	46	118	68	89 *(-24.58%)*	62 *(-8.82%)*
		vol:	175	707	335	459 *(-35.08%)*	308 *(-8.06%)*
		2q gates:	130	112	132	106 *(-5.36%)*	118 *(-10.61%)*
vbe _adder _3	10	*depth:*	67	216	165	197 *(-8.8%)*	232 *(+40.61%)*
		vol:	162	1244	765	1131 *(-9.08%)*	835 *(+9.15%)*
		2q gates:	122	190	294	195 *(+2.63%)*	329 *(+11.9%)*

Chapter 5

Formally-verified Synthesis Methods and Experiments

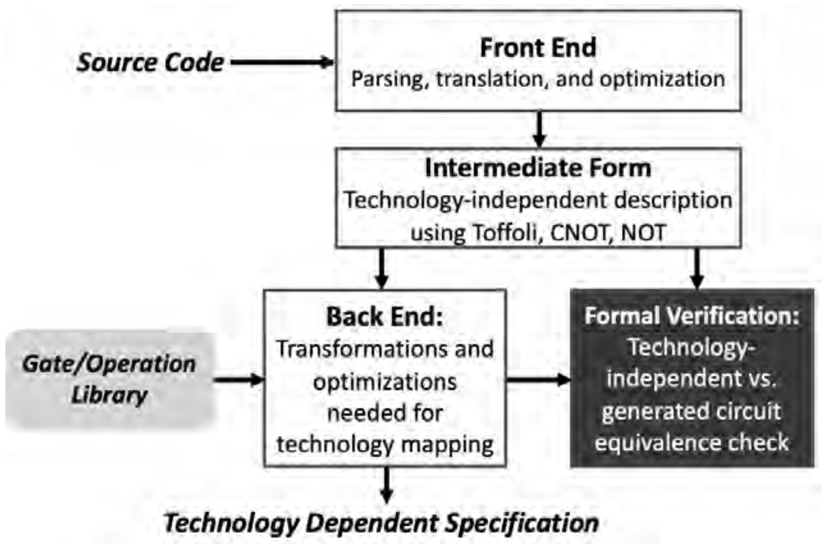

Source Code
Front End
Parsing, translation, and optimization
Intermediate Form
Technology-independent description using Toffoli, CNOT, NOT
Gate/Operation Library
Back End:
Transformations and optimizations needed for technology mapping
Formal Verification:
Technology-independent vs. generated circuit equivalence check
Technology Dependent Specification

Figure 5.1. Synthesis and compilation tool architecture.

A complete flow chart describing the architecture for a quantum logic synthesis and compilation tool is depicted in Fig. 5.1. This figure illustrates how information is processed to create a final implementation-specific quantum circuit. The technology-dependent synthesis and compilation techniques described in this work were developed into a prototype that acts as the back-end of a quantum logic synthesis tool. This tool has been named Mustang-Q.

With respect to the formally-verified technology-dependent outputs, if the target technology is IBM, the final specification is in the form of QASM. Rigetti devices, on the other hand, are described with Quil. Quantum compilation targeting both families of quantum devices will be described in this section.

5.1. IBM

5.1.1. Methodology

Technology-dependent synthesis for the IBM machines followed the procedures outlined in Fig. 5.1. Initially, the original circuit is parsed in as source code. Various file formats are supported for the input specification depending on the type of logic. If the input circuit is in the form of a classical switching function, the front-end will handle the initial parsing and translation of the specification into a reversible cascade of **NOT**, **CNOT**, Toffoli, and generalized Toffoli operators using the algorithm in (Fazel et al., 2007). The front-end result is a reversible representation of the input circuit that is technology-independent. Reversible code generated by the front-end as well as input circuits already implemented with quantum logic and specified in a quantum instruction language (*i.e.* a .qasm, .qc, or .real file format) are then processed by the back-end of the design tool. The back-end, the component of focus in this work, performs transformations and optimizations needed for technology mapping to a specific physical quantum machine. Multi-qubit gate decomposition algorithms, such as those given by Barenco et al. in (Barenco et al., 1995), are representative of some of the transformations implemented in the back-end. However, additional optimizations were devised and implemented to accommodate for device coupling maps that limit qubit connections for two-qubit operations.

The back-end generates technology-dependent QASM specifications based upon two distinct objectives. The first objective is to produce QASM that conforms to the user-specified target QC's architectural constraints such as a fixed **CNOT** coupling map. The second objective is to determine a mapping that minimizes quantum cost. The quantum cost function is defined in Eqn. 3.3. Results in this section are targeted to the IBM family of QCs as well as machines inspired by the architecture of the IBM QCs, but custom transmon devices with different coupling maps can be added to the tool to provide additional targets during synthesis. While targeting the IBM machines, the quantum logic synthesis tool implements the following mapping and optimization procedures:

1. **CNOT** operations placed in directions opposite of what is available in coupling map may be reversed (Nielsen and Chuang, 2010). **CNOT** reversal can be seen in Fig. 4.2.

2. **CNOT** operations on qubits not coupled directly or in reverse on the coupling map are rerouted with CTR.

3. Generalized Toffoli gates are decomposed into Toffoli cascades using (Barenco et al., 1995).

4. Toffoli operations are decomposed into one- and two-qubit operators supported by the transmon technology library using a transform from (Nielsen and Chuang, 2010).

5. Local optimizations based on removing partitions of gates that equal the identity function are implemented recursively until technology library cost function cannot be further reduced.

6. Local optimizations based on removing partitions of gates that can be minimized with an logically identical circuit identity are implemented recursively until technology library cost function cannot be further reduced.

It should be noted that all **SWAP** operations will have a maximum gate count of 7, including four **H** operations and three **CNOT** operations, due to unidirectional transmon **CNOT** operations and the identity pictured in Fig. 4.2.

After all synthesis and optimization procedures are complete, formal verification is invoked by the compiler. QMDDs are used in equivalence tests to formally verify all technology-dependent compiler outputs. During this process, the original technology-independent specification is compared to the generated technology-dependent specification by building the QMDD data structures. Since the QMDDs share isomorphic subgraphs, the pointers to the original and technology-mapped specification will match if the two designs are functionally identical. The final step of formal verification is critical as it is important that the algorithm's logic is unchanged by the synthesis tool's transformations and optimizations.

5.1.2. Experimental Results

Back-end algorithms of the synthesis and compilation tool responsible for mapping and optimizing algorithms for real QC architectures were developed in Python. The tool's purpose is to synthesize technology-dependent algorithms for execution on actual QCs using classical computing methods. Design automation, including formal verification, was performed on a laptop running macOS 10.13.6 with an Intel i5 processor at 2.9 GHz and 8 GB of RAM. Although results in this section include the devices in Section 3.4.1 as well as an example 96-qubit IBM-inspired layout, additional architectures can be targeted for synthesis by adding the desired topology coupling map to the device library of the tool.

The first set of benchmarks used during experimentation were obtained from reference (rev, 2017). The technology-independent specifications, titled "Optimal Single-target Gates," range from 3 to 6 qubits in size. These circuits were chosen as benchmarks because they act as essential components for quantum logic synthesis based on lookup-table approaches (Soeken et al., 2018). Complex reversible and quantum circuits decompose into these functions, and in turn, the single-target gates can be decomposed into one and two qubit operations. These benchmarks were input into the synthesis tool as technology-independent .qc files that contained single qubit operations and **CNOT** gates. When mapped to the simulator, the logic is referred to as technology-independent because is not restricted by the layout of a physical device. The simulator synthesis resulted in technology-independent circuits that match what is featured in (rev, 2017) with respect to **T** counts and total gate counts because these generated circuits, like the original benchmarks, have no restrictions with respect to where multi-qubit operations are placed. Cost function-based optimization did not reduce the gate counts or the total Eqn. 3.3 cost of the benchmarks whenever they were mapped to the simulator as the benchmark circuits are already in their most compact and optimal form when qubit connections are unrestricted.

In reality, quantum circuit designers must be careful with gate placement on real QCs due to architectural constraints. These architectural constraints limit what qubit pairs can couple for multi-qubit transformations, and permitted qubit connections for a QC are described in the form of a coupling map. When the single-target gate benchmarks are mapped to real devices, unsupported gate placements must be decomposed or rerouted with **SWAP** operations, as described in Section 4.1, that cause the circuits to expand. It was noted that QCs with a lower coupling complexity usually required more gates to achieve a technology-dependent mapping. After mapping finishes, the resulting circuit may be optimized using built-in local optimizers. Synthesis results of the "Optimal Single-target Gates" mapped to the IBM devices can be found in Table 5.7. This table includes both pre- and post-optimization metrics for **T**-count, total gate count, and cost calculated using Eqn. 3.3 for each of the generated designs. Technology-independent (i.e. the original, unmapped) metrics for **T**-count, total gate count, and cost for the benchmarks for unoptimized and optimized mappings were included as well for comparison. A comparison can be made here because these benchmark circuits are already fully optimized in a gate library suitable for IBM that includes one- and two-qubit gates when they are not constrained by any sort of connection restrictions. When these circuits are mapped to a real machine, however, they expand in size because the circuits must be reconfigured according to the coupling map to be executable. After original mapping, optimizations help reduce the overall gate counts and cost.

A few observations can be made from analyzing Table 5.7. First, some designs were not synthesizable, as indicated by N/A, if the target QC was too small for a circuit (i.e. 5 qubit machines cannot support a circuit with n qubits where $n > 5$). Second, technology mapping processes during compilation caused circuits in most cases to expand. This expansion, sometimes of order 10^1 in size, was caused by the need to reroute **CNOT** operations to

Table 5.1. Results of compilation using benchmarks from (rev, 2017) mapped to IBM devices.

Ftn.	# Qubits	*T* gates	Tech. Ind. (unopt. == opt.)	In form of (unoptimized mapping T-count / gates / cost) (optimized mapping T-count / gates / cost)									
				ibmqx2		ibmqx3		ibmqx4		ibmqx5		ibmq_16	
#1	3	7	7/17/22.25	***7/33/38.25***	***7/29/34.25***	***7/61/69.25***	***7/59/67.25***	***7/29/34.25***	***7/25/30.25***	***7/57/65.25***	***7/55/63.25***	***7/57/65.25***	***7/53/60.75***
#3	3	0	0/3/3.25	0/3/3.25	0/3/3.25	0/17/18.75	0/17/18.75	0/7/7.25	0/7/7.25	0/17/18.75	0/17/18.75	0/17/18.75	0/17/18.75
#01	5	15	15/51/63.75	***15/215/239.75***	***15/213/237.75***	***15/433/480.25***	***15/427/473.75***	***15/207/231.75***	***15/205/229.75***	***15/397/444.25***	***15/389/435.25***	***15/425/472.25***	***15/419/465.75***
#03	4	7	7/20/25.25	7/96/107.25	7/96/107.25	7/116/130.25	7/116/130.25	7/92/103.25	7/92/103.25	7/112/126.25	7/112/126.25	7/112/126.25	7/112/126.25
#07	5	16	16/60/75	***16/272/302***	***16/264/294***	***16/552/612***	***16/536/594.5***	***16/260/290***	***16/252/282***	***16/504/564***	***16/490/548***	***16/532/592***	***16/518/576.5***
#0f	4	0	0/3/3.25	0/21/22.75	0/21/22.75	0/35/38.25	0/35/38.25	0/21/22.75	0/21/22.75	0/35/38.25	0/35/38.25	0/31/34.25	0/31/34.25
#17	4	7	7/43/51.75	***7/215/235.75***	***7/203/223.75***	***7/289/320.25***	***7/271/300.25***	***7/203/223.75***	***7/197/217.75***	***7/273/304.25***	***7/257/286.25***	***7/273/304.25***	***7/257/286.25***
#0001	6	40	40/186/233	N/A		***40/2928/3240.5***	***40/2496/2769***	N/A		***40/2840/3152.5***	***40/2434/2706.5***	***40/2876/3188.5***	***40/2456/2725.5***
#0003	6	15	15/66/83	N/A		***15/900/996.5***	***15/824/913.5***	N/A		***15/880/976.5***	***15/792/880***	***15/904/1000.5***	***15/812/900***
#0007	6	47	47/246/304.25	N/A		***47/3116/3445.75***	***47/2728/3021.75***	N/A		***47/3096/3425.75***	***47/2652/2940.75***	***47/3144/3473.75***	***47/2688/2976.25***
#000f	5	7	7/21/27.5	***7/177/195.5***	***7/153/170***	***7/323/358***	***7/291/323***	***7/153/171.5***	***7/135/152***	***7/303/338***	***7/275/307***	***7/319/354***	***7/287/319***
#0017	6	23	23/129/159	N/A		***23/1851/2050.5***	***23/1683/1864.5***	N/A		***23/1855/2054.5***	***23/1667/1849***	***23/1899/2098.5***	***23/1745/1930***
#001f	6	43	43/194/244.5	N/A		***43/2864/3171***	***43/2474/2744***	N/A		***43/2772/3079***	***43/2434/2707***	***43/2828/3135***	***43/2454/2724***
#003f	6	16	16/73/92.25	N/A		***16/1103/1219.75***	***16/1077/1192.75***	N/A		***16/1075/1191.75***	***16/1051/1167.75***	***16/1115/1231.75***	***16/1087/1203.75***
#007f	6	40	40/189/238.5	N/A		***40/2791/3091***	***40/2593/2876***	N/A		***40/2727/3027***	***40/2503/2780***	***40/2787/3087***	***40/2531/2806.5***
#00ff	5	0	0/3/3.25	0/21/22.75	0/21/22.75	0/45/49.75	0/45/49.75	0/21/22.75	0/21/22.75	0/45/49.75	0/45/49.75	0/49/53.75	0/49/53.75
#0117	6	79	79/401/498	N/A		***79/5167/5718.5***	***79/4631/5132***	N/A		***79/5119/5670.5***	***79/4483/4973***	***79/5159/5710.5***	***79/4471/4956***
#011f	6	27	27/136/169.5	N/A		***27/1886/2086***	***27/1662/1839.5***	N/A		***27/1818/2018***	***27/1628/1807.5***	***27/1842/2042***	***27/1622/1799***
#013f	6	48	48/240/299.5	N/A		***48/3060/3389.5***	***48/2698/2993.5***	N/A		***48/3116/3445.5***	***48/2750/3046.5***	***48/3148/3477.5***	***48/2780/3078.5***
#017f	6	80	80/359/455	N/A		***80/5025/5566.5***	***80/4595/5097***	N/A		***80/4953/5494.5***	***80/4415/4904***	***80/5033/5574.5***	***80/4451/4936.5***
#033f	5	7	7/49/60.75	***7/391/428.25***	***7/299/329.25***	***7/887/978.25***	***7/747/824.25***	***7/347/384.25***	***7/277/307.75***	***7/839/930.25***	***7/691/767.25***	***7/879/970.25***	***7/727/802.75***
#0356	5	12	12/42/54.75	***12/248/274.25***	***12/224/249.25***	***12/444/491.25***	***12/388/430.25***	***12/220/246.25***	***12/208/234.25***	***12/412/459.25***	***12/372/415.25***	***12/416/463.25***	***12/362/404.25***
#0357	6	61	61/266/336.5	N/A		***61/3944/4367***	***61/3418/3788.5***	N/A		***61/3860/4283***	***61/3276/3639.5***	***61/3944/4367***	***61/3296/3653***
#035f	6	23	23/107/135.5	N/A		***23/1327/1469.5***	***23/1193/1322***	N/A		***23/1327/1469.5***	***23/1161/1288***	***23/1343/1485.5***	***23/1191/1319.5***

qubits where they could execute. Fortunately, out of the 94 output technology-dependent designs, 74, or approximately 79%, were improved when optimization algorithms based on minimizing the transmon cost function were implemented. Designs that improved in cost post-optimization are emphasized in Table 5.7. Information about the post-optimization cost function improvement for the technology-dependent "Optimal Single-target Gates" is found in Table 5.2. The greatest average percent decrease in cost was about 8.5% for the circuits mapped to the `ibmq_16` QC. The average post-optimization improvement for all of the technology-dependent forms of the reference (rev, 2017) benchmarks was approximately 7%.

The second set of benchmarks tested on the automatic quantum logic synthesis tool was a small set of Toffoli cascades from (rev). Toffoli cascade circuits are widely used to describe technology-independent reversible and quantum logic algorithms in the form of **NOT**, **CNOT**, Toffoli, and generalized Toffoli operations. These circuits were used with the tool to demonstrate the Toffoli and general Toffoli decomposition techniques. In their

Table 5.2. Percent decrease of (rev, 2017) benchmark cost after optimization.

Funct.	ibmqx2	ibmqx3	ibmqx4	ibmqx5	ibmq_16
#1	10.46	2.89	11.68	3.07	6.90
#3	0.00	0.00	0.00	0.00	0.00
#01	0.83	1.35	0.86	2.03	1.38
#03	0.00	0.00	0.00	0.00	0.00
#07	2.65	2.86	2.76	2.84	2.62
#0f	0.00	0.00	0.00	0.00	0.00
#17	5.09	6.25	2.68	5.92	5.92
#0001	N/A	14.55	N/A	14.15	14.52
#0003	N/A	8.33	N/A	9.88	10.04
#0007	N/A	12.31	N/A	14.16	14.32
#000f	13.04	9.78	11.37	9.17	9.89
#0017	N/A	9.07	N/A	10.00	8.03
#001f	N/A	13.47	N/A	12.08	13.11
#003f	N/A	2.21	N/A	2.01	2.27
#007f	N/A	6.96	N/A	8.16	9.09
#00ff	0.00	0.00	0.00	0.00	0.00
#0117	N/A	10.26	N/A	12.30	13.21
#011f	N/A	11.82	N/A	10.43	11.90
#013f	N/A	11.68	N/A	11.58	11.47
#017f	N/A	8.43	N/A	10.75	11.44
#033f	23.12	15.74	19.91	17.52	17.26
#0356	9.12	12.42	4.87	9.58	12.74
#0357	N/A	13.25	N/A	15.02	16.35
#035f	N/A	10.04	N/A	12.35	11.17
Average	***5.85***	***7.65***	***4.92***	***8.04***	***8.48***

original form, these technology-independent Toffoli cascades include larger, multi-qubit gates that are not supported by the IBM transmon gate library. The benchmarks may seem small because of their gate count, but they are actually complex functions to implement because of the required multi-qubit interaction. Synthesis results of the Toffoli cascade benchmarks mapped to the IBM devices can be found in Table 5.8. It should be noted that these results do not include a column for technology-independent, unoptimized and optimized mappings as seen in Table 5.7 because the Toffoli gate is not present in the IBM gate library and is therefore not a technology-ready gate, even when connections between the qubits on the quantum device are disregarded.

In Table 5.8, circuits that could not synthesize because they were too large for an architecture were once again indicated by N/A. These experiments demonstrated that Toffoli decomposition followed by mapping procedures caused circuits to expand in gate count up to orders of magnitude of 10^2 their original size. After optimization, 100% of the mapped Toffoli cascades in Table 5.8 decreased in size. Information detailing the post-optimization improvement of the Toffoli cascade benchmarks is found in Table 5.4. The greatest average percent decrease in cost was nearly 30% for the circuits mapped to the `ibmqx3` machine. The average post-optimization improvement for all of the reference (rev, 2017) benchmarks was approximately 17.4%.

Most technology-dependent specifications in Table 5.7 and Table 5.8 were generated in approximately 10^{-2} seconds, but a few of the larger benchmarks with more multi-qubit gates required a few seconds with none exceeding 5 seconds for synthesis procedures. All outputs were confirmed to be the same function as their original technology-independent description by building the QMDD data structure for each design and testing for equivalence.

Table 5.3. Results of compilation using benchmarks from (rev) mapped to IBM devices.

Ftn.	**# Qubits**	**Largest Gate**	**Gate Count**	***In form of (unoptimized mapping T-count / gates / cost) (optimized mapping T-count / gates / cost)*** **ibmqx2**		**ibmqx3**		**ibmqx4**		**ibmqx5**		**ibmq_16**	
3_17_14	3	toffoli	6	*14/142/160.25*	*14/138/156.25*	*14/142/160.25*	*14/138/156.25*	*14/114/132.25*	*14/112/130.25*	*14/142/160.25*	*14/138/156.25*	*14/142/160.25*	*14/138/156.25*
fred6	3	toffoli	3	*21/164/188*	*21/162/186*	*21/148/172*	*21/138/162.0*	*21/164/188*	*21/154/178*	*21/148/172*	*21/138/162*	*21/148/172*	*21/138/162*
4_49_17	4	toffoli	12	*35/357/402.75*	*35/299/340.25*	*35/567/635.25*	*35/515/578.25*	*35/325/370.75*	*35/271/312.25*	*35/535/603.25*	*35/471/533.25*	*35/567/635.25*	*35/515/578.25*
4gt12-v0_88	5	T5	5	N/A		*70/2882/3211.5*	*70/1348/1515.5*	N/A		*70/2382/2657.5*	*70/1372/1542.5*	*70/2218/2472.5*	*70/1554/1741*
4gt13-v1_93	5	T4	4	*28/377/423*	*28/365/411*	*28/2665/2960*	*28/587/660*	*28/357/403*	*28/345/391*	*28/2181/2422*	*28/631/710*	*28/1685/1869*	*28/863/961.5*

The size of quantum architectures is anticipated to increase, so it is important that design tools are able to scale. To test the synthesis and compilation tool, a 96-qubit transmon-based QC architecture was designed and loaded into the tool. The qubits of the machine ranged from q0-q95, and the coupling map, pictured in Fig. 5.2, was inspired by the `ibmqx5` machine.

Table 5.4. Percent decrease of (rev) benchmark cost after optimization.

Funct.	ibmqx2	ibmqx3	ibmqx4	ibmqx5	ibmq_16
3_17_14	2.50	2.50	1.51	2.50	2.50
fred6	1.06	5.81	5.32	5.81	5.81
4_49_17	15.52	8.97	15.78	11.60	8.97
4gt12-v0_88	N/A	52.81	N/A	41.96	29.59
4gt13-v1_93	2.84	77.70	2.98	70.69	48.56
Average	***5.48***	***29.56***	***6.40***	***26.51***	***19.08***

Benchmarks containing more qubits were needed for the larger machine. Previous experimentation found that Toffoli circuits decomposed into large designs. Additionally, those that included generalized Toffoli gates with more controls had a greater final gate volume. With this in mind, benchmarks with the generalized Toffoli gates $\mathbf{T}_6$, $\mathbf{T}_7$, $\mathbf{T}_8$, $\mathbf{T}_9$, and $\mathbf{T}_{10}$ were designed for implementation on the 96-qubit architecture. For each of these Toffoli operators, the subscript indicates the total number of qubits that are included in the multi-qubit operation. Each circuit contained a cascade of four gates of each type, and they were placed on the QC in such a manner that the gates shared at least a single qubit with another. Information about the contents of the third set of benchmarks can be found in Table 5.5. In this table, the controls and target for each $\mathbf{T}_n$ gate in the cascade are described.

All of the Table 5.5 benchmarks were mapped to the 96-qubit example QC of Fig. 5.2. Although each circuit originally included four gates, the designs greatly increased in gate volume to accommodate to the Fig. 5.2 coupling map as well as the one- and two-qubit transmon gate library. Data concerning pre- and post-optimization $\mathbf{T}$-counts, gate counts, and cost is included in Table 5.6. A column for technology-independent data for unoptimized and optimized mapping was also omitted from this table for the same reasons as with Table 5.8. The purpose of Table 5.6 is to not only demonstrate the generalized Toffoli decomposition capabilities of our tool, but to also demonstrate that the tool is scalable. In Table 5.6, optimization drastically improved the overall cost on the larger machine. On average, the large Toffoli cascade benchmarks improved in cost by 39.5%. Most of the resulting

Table 5.5. 96-qubit QC benchmark details.

Name	Gates	Controls	Target
T6_b	1: T6	q1, q2, q3, q4, q5	q25
	2: T6	q21, q22, q23, q24, q25	q45
	3: T6	q41, q42, q43, q44, q45	q65
	4: T6	q61, q62, q63, q64, q65	q85
T7_b	1: T7	q1, q2, q3, q4, q5, q6	q25
	2: T7	q21, q22, q23, q24, q25, q26	q45
	3: T7	q41, q42, q43, q44, q45, q46	q65
	4: T7	q61, q62, q63, q64, q65, q66	q85
T8_b	1: T8	q1, q2, q3, q4, q5, q6, q7	q25
	2: T8	q21, q22, q23, q24, q25, q26, q27	q45
	3: T8	q41, q42, q43, q44, q45, q46, q47	q65
	4: T8	q61, q62, q63, q64, q65, q66, q67	q85
T9_b	1: T9	q1, q2, q3, q4, q5, q6, q7, q8	q25
	2: T9	q21, q22, q23, q24, q25, q26, q27, q28	q45
	3: T9	q41, q42, q43, q44, q45, q46, q47, q48	q65
	4: T9	q61, q62, q63, q64, q65, q66, q67, q68	q85
T10_b	1: T10	q1, q2, q3, q4, q5, q6, q7, q8, q9	q25
	2: T10	q21, q22, q23, q24, q25, q26, q27, q28, q29	q45
	3: T10	q41, q42, q43, q44, q45, q46, q47, q48, q49	q65
	4: T10	q61, q62, q63, q64, q65, q66, q67, q68, q69	q85

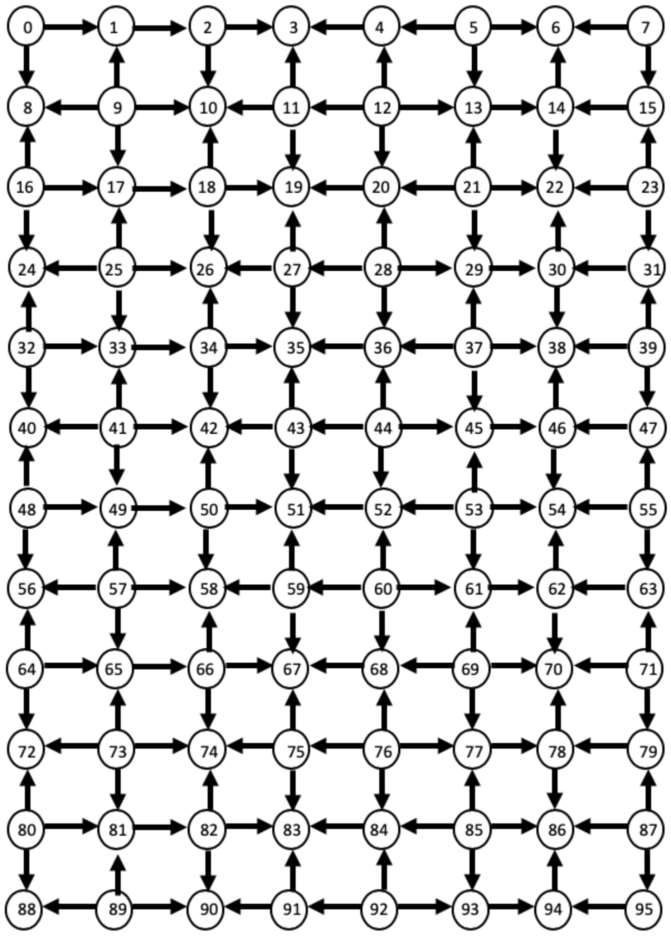


Figure 5.2. Proposed 96-qubit machine used for experimentation.

Table 5.5 technology-dependent circuits took under a second to generate, with the largest taking approximately 6.5 seconds. All of the output designs were verified for accuracy using the QMDD equivalence test.

5.2. Rigetti

5.2.1. Methodology

The prototype of this work is capible of transforming technology-independent quantum

Table 5.6. 96-qubit QC benchmark compilation results.

Name	**Unoptimized**	**Optimized**	**Percent Cost**
	(T-count / gates / cost)	**(T-count / gates / cost)**	**Decrease**
T6_b	336/17312/19268	336/10156/11359	41.05
T7_b	448/20112/22400	448/12234/13694	38.87
T8_b	560/21264/23728	560/13134/14746	37.85
T9_b	672/17696/19784	672/11544/13002	34.28
T10_b	784/17792/19960	784/9518/10846	45.66
Average			***39.54***

circuits into technology-dependent Quil specifications for execution on a Rigetti machine. Just as when IBM is the target technology, the original circuit is parsed in as source code, and it contains a variety of operators that may or may not be supported by the target architecture. The tool then performs transformations and optimizations needed for technology mapping to a specific Rigetti QC. Multi-qubit gate decomposition algorithms, such as those given by Barenco et al. in (Barenco et al., 1995), are representative of some of the transformations implemented in this tool. Additional optimizations and algorithms were also developed to accommodate for QC topological differences that limit qubit connections for two-qubit operations.

As previously described, a significant drawback of solid-state qubit QCs is that the stationary qubits are limited to certain multi-qubit operations due to the layout and physical properties of the device. For this reason, CTR was implemented for completing multi-qubit operations over uncoupled qubits.

The **CNOT** gate is commonly used in technology-independent circuits, and as shown in Fig. 4.1, three of the operators in series can form a **SWAP** operation among physically connected qubits. Due to the simplicity of the **SWAP** implementation with **CNOT**, the

CTR algorithm in the quantum synthesis tool uses **CNOT** to automatically reroute two-qubit operations that are not supported by a coupling map. Of course, as **CNOT** is not a supported gate in the Rigetti library, these gates are eventually converted into **CZ** gates using the transformation in Fig. 5.3 after reroute operations have completed.

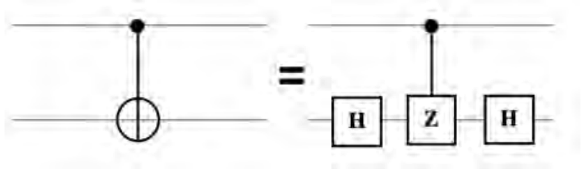


Figure 5.3. **CNOT** to **CZ** transformation.

Technology-dependent Quil specifications are generated based upon two distinct objectives. The first objective is to produce a quantum algorithm that conforms to the user-specified target QC's architectural constraints. The second objective is to determine a mapping that minimizes quantum cost. The quantum cost function is defined in Eqn. 3.4, and the quantum logic synthesis tool implements the following mapping and optimization procedures:

1. Generalized Toffoli gates are decomposed into Toffoli cascades using (Barenco et al., 1995).
2. Toffoli operations are decomposed into one and two-qubit operators using transforms from (Nielsen and Chuang, 2010), unsupported two-qubit gate placements are rerouted, and **CNOT** to **CZ** transformation occurs.
3. Local optimizations based on removing circuit partitions that equal the identity function are implemented recursively until the cost function cannot be further reduced.
4. One-qubit operators are decomposed into gates that are native to the Rigetti operator library.

After all synthesis and optimization procedures are complete, the optimized technology-dependent specification is formally verified by performing an equivalence checking test using QMDDs.

5.2.2. Experimental Results

Quantum logic synthesis involving the mapping and optimization of algorithms for the Rigetti QCs was completed with technology-independent benchmarks from a library titled "Optimal Single-target Gates," from (rev, 2017). These benchmarks were input into the synthesis tool as technology-independent .qc files that contained single-qubit and **CNOT** operations. Because of arbitrary single-qubit gates and two-qubit gate placement within the benchmarks, transformation is required before the algorithms can be executed on the Rigetti QPUs. The benchmarks of (rev, 2017) range from 3 to 6 qubits in size, and in the interest of testing only the more complex and non-trival designs, the 6 qubit "Optimal Single-target Gates" were used in experimentation.

When the single-target gate benchmarks were mapped to QCs, unsupported gate placements were decomposed or rerouted with swapping operations that cause the circuits to expand. After mapping finished, the resulting circuit could be optimized using built-in local optimizers. Synthesis results of the "Optimal Single-target Gates" mapped to the Rigetti QCs can be found in Table 5.7. This table includes both pre- and post-optimization metrics for **CZ** operation count and cost calculated using Eqn. 3.4 for each of the generated designs.

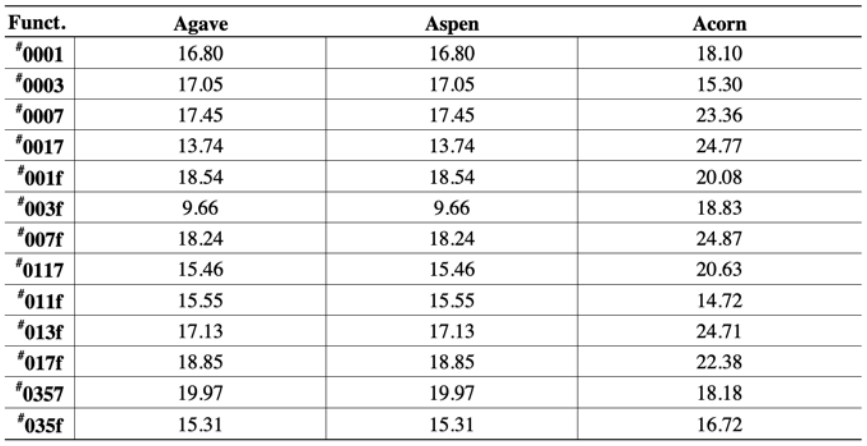
Referring to Table 5.7, the synthesis results for `Agave` match those for `Aspen`. This occurs since these devices have a similar style of ring topology. `Acorn`, characterized by a grid topology, yields different synthesis results. Information about cost improvement whenever synthesis includes optimization processes is found in Table 5.8. On average, when optimization was performed on a technology-dependent circuit, cost improved by approximately 17.7%. This improvement in cost is significant because a lower cost indicates that a circuit uses fewer operations and thus executes faster with a decreased probability of de-

Table 5.7. Metrics for (**CZ** count/cost) after synthesis using benchmarks from (rev, 2017) targeting the Rigetti QCs.

	In form of (unoptimized mapping) (optimized mapping)		
Funct.	**Agave**	**Aspen**	**Acorn**
#0001	(1026/11392) (942/9478)	(1026/11392) (942/9478)	(2070/22876) (1860/18736)
#0003	(296/3296) (270/2734)	(296/3296) (270/2734)	(404/4484) (376/3798)
#0007	(1141/12696) (1041/10480)	(1141/12696) (1041/10480)	(3103/34278) (2613/26272)
#0017	(692/7683) (656/6627)	(692/7683) (656/6627)	(1256/13887) (1036/10447)
#001f	(974/10820) (874/8814)	(974/10820) (874/8814)	(1844/20390) (1618/16296)
#003f	(399/4429) (395/4001)	(399/4429) (395/4001)	(453/5023) (403/4077)
#007f	(964/10703) (870/8751)	(964/10703) (870/8751)	(1936/21395) (1600/16073)
#0117	(1856/20641) (1742/17449)	(1856/20641) (1742/17449)	(5426/59911) (4736/47551)
#011f	(674/7486) (626/6322)	(674/7486) (626/6322)	(1670/18442) (1558/15728)
#013f	(1174/13050) (1076/10814)	(1174/13050) (1076/10814)	(2878/31794) (2382/23938)
#017f	(1778/19745) (1594/16023)	(1778/19745) (1594/16023)	(3704/40931) (3168/31771)
#0357	(1390/15432) (1232/12350)	(1390/15432) (1232/12350)	(3088/34110) (2778/27910)
#035f	(488/5423) (454/4593)	(488/5423) (454/4593)	(1670/18425) (1522/15345)

coherence. All outputs of the synthesis runs were verified to be equivalent to their original technology-independent specifications using QMDDs.

Table 5.8. Percent decrease in cost from unoptimized to optimized synthesis targeting the rigetti QCs.



Funct.	Agave	Aspen	Acorn
#0001	16.80	16.80	18.10
#0003	17.05	17.05	15.30
#0007	17.45	17.45	23.36
#0017	13.74	13.74	24.77
#001f	18.54	18.54	20.08
#003f	9.66	9.66	18.83
#007f	18.24	18.24	24.87
#0117	15.46	15.46	20.63
#011f	15.55	15.55	14.72
#013f	17.13	17.13	24.71
#017f	18.85	18.85	22.38
#0357	19.97	19.97	18.18
#035f	15.31	15.31	16.72

Chapter 6

Higher Dimensioned Quantum Logic Synthesis

Conventional information processing technology is overwhelmingly based upon radix-2, or binary, switching algebras, and the most commonly used measure of information is the "bit." It is well-known, however, that higher-radix systems offer more information content per fundamental representational unit, or "digit" (Miller and Thornton, 2007). More precisely, an information processing system based upon a radix-r system allows for $log_2(r)$ bits of information to be represented per digit. It is disputed which base of computation is most ideal as computing resources have to be taken into consideration along with information bandwidth, but even the introduction of another logic level in classical computing schemes to implement ternary allows for the simplification of decision tree processes. For example, rather than in a comparison of x and y that results in the binary answer to "is less than" followed by an answer to "is equal," ternary allows an answer of x "is less," "is equal," and "is greater" than y in a single step (Hayes, 2001). Despite this higher-radix advantage, the rapid size decrease in transistors has caused the binary radix to continue to prevail since transistor-based information processing circuits are predominantly in the form of voltage-mode devices. The increasingly smaller feature sizes and the corresponding and necessarily smaller rail voltage levels result in noise margins that cause higher-valued radices to be impractical in modern electronic information processing circuits. This happens because the benefits afforded by increasing the overall number of small transistors per unit of area in conventional electronic central processing units (CPUs) outweighs those that could potentially be realized through the use of larger transistors that are enabled to switch among multiple voltage levels in a higher-radix implementation. For instance, an $r = 3$ ternary system would require voltages corresponding to the digits $\{0, 1, 2\}$. From a practical point of view, there would actually need to be voltage ranges specified, commonly characterized as "noise margins," that define how much a particular voltage can vary from a specified nominal voltage that represents a particular valuation of information. Thus, implementing higher-radix systems in conventional electronics is theoretically possible, but it is rarely done in practice because the advantages of using extremely small, and therefore more transistors per unit area, acting as binary switches outweighs the advantages that would be gained by using larger transistors, and hence fewer per unit area, that implement switching among r different voltages representing a higher-valued non-binary radix system. Smaller transistors require smaller rail voltages to operate properly and subdividing these small rail-to-rail voltage intervals into more than two discrete ranges would result in noise margins that are impractical to implement. As a consequence, radix-2 or binary logic dominates in modern electronic devices under the classical computational models.

Classical mechanics characterizes physical systems and how these systems interact with their environment and each other on a large, often human-perceivable scale. If systems in nature are heavily magnified, materials eventually break down to a discrete set of particles that are influenced by discrete packets of energy. Relating mechanics to computation, QCs are built of quantum mechanical elements in order to obey quantum mechanical laws. Due to the nature of the technology being considered in QIS, the limitations that have caused conventional electronic information processing systems to remain in the radix-2 or binary realm, such as continuous values during measurement, are not necessarily relevant. Since quantum mechanics utilizes quantized energy levels and particles, the use of higher-radix systems for the representation and processing of information in QIS is potentially viable and offers advantages in terms of the amount of information that can be represented per system element.

The additional levels for basis representation exist for many technologies. Examples of non-binary qudit-based QIP realizations based upon the photon include OAM (Gibson et al., 2004), time-energy (Zhong et al., 2015), frequency (Lukens and Lougovski, 2017), time-phase (Islam, 2018b), and location. Qudits implemented with superconducting solid-state technology such as transmon circuits have also been reported (Liu et al., 2017) where existing higher-energy levels are used for logic encoding. To accommodate to higher dimensioned quantum systems, especially those that are compatable with radix-2 technology, methodologies for qudit control and readout have been developed (Randall et al., 2015, 2018).

Although higher-radix quantum implementations have been researched and demonstrated, preexisting familiarity with computation techniques using a binary basis often causes additional energy levels beyond two to be left unused. Expanding the dimension of quantum computation, however, offers advantages in terms of the amount of information that can be represented per system element often without the strain of many additional resources. The compression involving the number of radix-r qudits, M, required to express the information of N qubits is described as

$$M = \frac{N}{log_2\left(r\right)}. \tag{6.1}$$

Increasing the radix of a system greatly increases computation and communication bandwidth. Although including more logic levels provides a certain level of computational advantage, an increase in dimension does increase system complexity because of the added opportunity for introducing error (Gokhale et al., 2019). This presents the question of determining the best radix for quantum computation. This value will be heavily influenced by the number of clearly defined and manipulatable logic levels for a particular technology platform. Regardless of the added opportunity for error with current technology, QIS research groups have eagerly written about the advantages of implementing qudits rather than qubits in QCs that could lead to more powerful quantum computation (Choi, 2017).

An important consideration when exploring the common information-theoretic justifications for using multiple-valued logic (MVL), such as enhanced information representation efficiencies in wired and wireless transmission channels, is that, in QIS, no matter what the radix, information content is much different than classical information due to superposition. As a result of superposition, the Shannon models of information representation for discrete information, or switching algebras, fall apart since, in QIS, multi-radix means a higher-dimensioned Hilbert space. This means that the the cardinality of the basis vector set increases. Higher radices in discrete switching theory have higher resolution in each discrete dimension of the Boolean, or Post, vector spaces, but the vector spaces themselves do not increase in dimensionality. Fig. 6.1 illustrates this difference by showing diagrams of the discrete switching algebra spaces for conventional information models and the Hilbert vector spaces for QIS models. This observation provides motivation for the development of QIS systems that utilize quantum digits, or "qudits," rather than the more commonly considered quantum bits, or "qubits." Because higher-radix systems are under consideration, the preparation of entangled qudit states must be investigated.

The technology-dependent synthesis techniques described in this work primarily focus on systems that are base-2. These algorithms, however, are implemented in a modular manner to accommodate for higher dimension QC compilation whenever devices and technology libraries become available. Although a complete, physically-implementable gate library for quantum MVL does not yet exist, methods and operators for synthesizing circuits that generate higher-radix superposition and entanglement were studied in preparation for when quantum devices become more robust.

6.1. Qudit Information

The most commonly used physical systems for quantum information are elements that have two distinct basis states. Due to the fact that the carriers have two distinct basis states, they carry information that is mathematically represented as a quantum bit, or qubit, by assigning each of the basis states to one of two orthonormal vectors. Quantum information can also be represented with carriers that exhibit quantum mechanical behavior over a non-binary basis set (Choi, 2017). Mathematically, a higher-radix system can be characterized

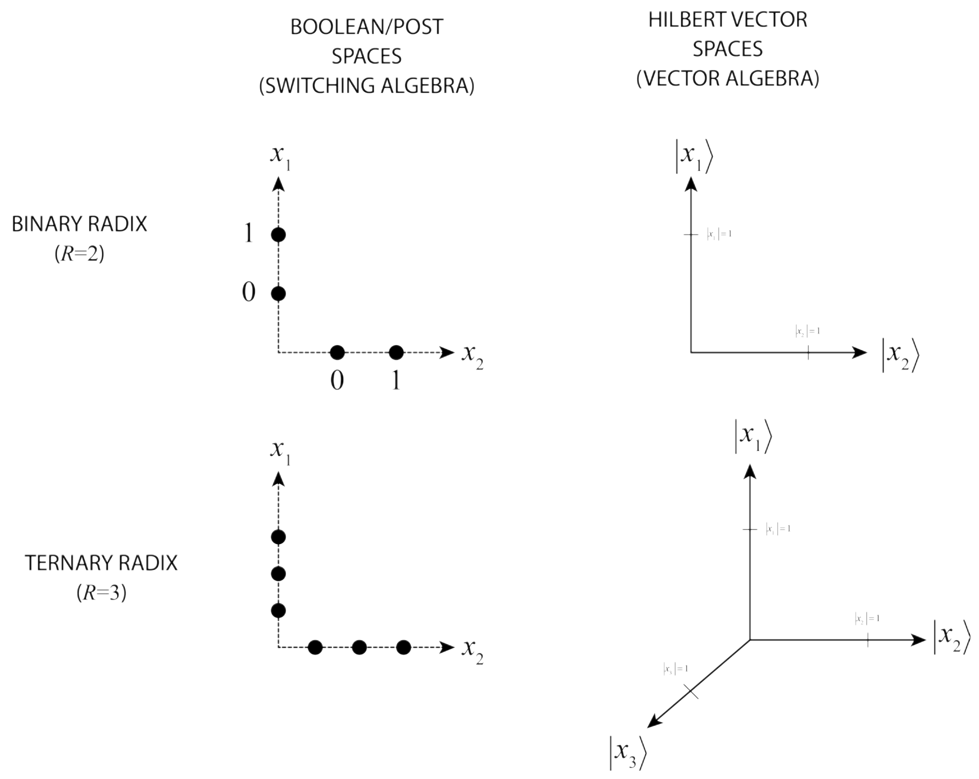


Figure 6.1. Comparison of vector spaces for $r = 2, 3$.

with a set of basis vectors that span a Hilbert vector space of dimension $r > 2$ in terms of qudits rather than the binary ($r = 2$) case of qubits. An example application for higher-radix QIP is QKD (Almeida et al., 2005; Islam, 2018a; Rohit and Srinivas, 2016).

Qubits are the most commonly implemented units of information in QIP. As a result, methods for automatically generating arbitrary radix-2 quantum state have been considered, yet the general problem of logic synthesis to produce a cascade of known operators remains a research problem. In terms of generating arbitrary quantum states, a recent method for the binary case is given in (Niemann et al., 2016a), yet the result is in terms of controlled rotation gates with arbitrary angles of rotation and not in terms of actual operators that are known to be fabricated in some technology.

As an example of higher-radix QIP, a radix-3 system consists of qudits expressed mathematically as a linear combination of three orthonormal basis vectors, $|0_3\rangle$, $|1_3\rangle$, and $|2_3\rangle$ where the subscripts indicate the value of r to avoid consusion while discussing systems of different values of r. In this case, the basis vectors span a three-dimensional Hilbert space. A general radix-3 qudit, $|\phi_3\rangle$ may be expressed mathematically as $|\phi_3\rangle = a_0 |0_3\rangle + a_1 |1_3\rangle + a_2 |2_3\rangle$. The set of radix-3 computational basis vectors are explicitly denoted as $|0_3\rangle = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$, $|1_3\rangle = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}}$, and $|2_3\rangle = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$. A quantum information system comprised of two radix-3 qudits is formulated in the same manner as that of a multi-qubit system through use of the tensor product.

In general, the form of a single qudit for an arbitrary radix, r, is given by

$$|\phi_r\rangle = \sum_{i=0}^{r-1} a_i |i_r\rangle . \tag{6.2}$$

Additionally, it is also the case that the a_i in Eqn. 6.3 are complex-valued quantities that satisfy

$$\sum_{i=0}^{r-1} |a_i|^2 = 1. \tag{6.3}$$

6.2. Qudit Superposition

A QIP system of radix-r qudits can also exhibit superposition. Superposition enables the processing of multiple valuations of information in a single quantum computation. Mathematically, superposition is expressed by the presence of two or more basis vector coefficients, seen as a_i in Eqn. 6.3, having non-zero values.

When the probability amplitudes are all non-zero and the square of their magnitudes are the same value, the qudit is said to be maximally superimposed or is in maximal superposition with respect to some basis set. Practically, this means that the qudit is equally likely to be measured as being in a state that is equivalent to any of the possible basis vectors. Multiple qubits or qudits may demonstrate states of maximal superposition. Achieving maximal superposition is an important operation and is one that is typically achieved as one of the very first operations in many quantum computing algorithms or processing flows.

6.2.1. The Hadamard Gate

The Hadamard operator, pictured in Table 2.1 causes a qubit originally in a basis state to evolve into a maximally superimposed state. It is denoted by the unitary transformation matrix $\mathbf{H}$. Each column, or row, vector comprising $\mathbf{H}$ is a discretized Walsh function with a scalar normalization factor of $\frac{1}{\sqrt{2^n}}$ where n is the order of the matrix that corresponds to the number of qubits comprising the quantum system. Since the Walsh functions are orthogonal and the Hadamard transform matrix includes a scalar normalization factor, the overall $\mathbf{H}$ matrix is comprised of an orthonormal column, or row, space. The first-order Hadamard matrix, $\mathbf{H}$, is expressed as

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{6.4}$$

$\mathbf{H}$ evolves a qubit, $|\phi_2\rangle$, initially in a basis state, into a state of maximal superposition so that it has equal probability of being observed, or measured, as either $|0_2\rangle$ or $|1_2\rangle$. As an example, consider the qubit $|\phi_2\rangle = |0_2\rangle$ that is evolved in time through application of a Hadamard operation, $|\phi_2\rangle = \mathbf{H}\,|0_2\rangle = (1/\sqrt{2})(|0\rangle + |1\rangle)$.

6.2.2. The Chrestenson Gate

Quantum operators exist for other computational bases, such as radix-3 and above, that achieve equal superposition among the corresponding basis states. These operators are referred to as “Chrestenson” gates. Since the Chrestenson operators can be formed for any radix $r > 2$, we denote them as $\mathbf{C}_r$ to indicate the radix, and alternatively, that $\mathbf{C}_r$ is a square matrix of dimension $r \times r$. Chrestenson gates are characterized by transformation matrices

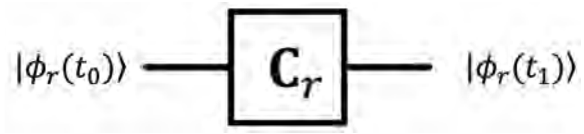


Figure 6.2. Radix-r Chrestenson gate, $\mathbf{C}_r$ evolving $|\phi_r\rangle$.

that may be derived using the discrete Fourier transform over Abelian groups. The general theory of the discrete Fourier transform over Abelian groups, referred to as the Chrestenson transform, can be found in references (Chrestenson et al., 1955; Vilenkin, 1947). Many useful applications of Chrestenson transforms in QIS have been demonstrated (Zilic and Radecka, 2002).

The graphical representation of the application of a Chrestenson gate, $\mathbf{C}_r$, on a radix-r qudit at time t_0 is in Fig. 6.2 and illustrates the operation $|\phi_r(t_1)\rangle = \mathbf{C}_r\,|\phi_r(t_0)\rangle$. Because the Chrestenson operator is a generalized version of the Hadamard operator wherein the radix, r, is an integer greater than two, the resulting transformation matrix for a single qudit is square with dimension $r \times r$. The Chrestenson transform can likewise be applied to a collection of n qudits with a resulting transformation matrix of dimension $r^n \times r^n$. The corresponding transformation matrix can be formed using the tensor product of n Chrestenson transformation matrices of dimension $r \times r$ and is denoted as

$$\mathbf{C}_r^n = \bigotimes_{i=1}^{n} \mathbf{C}_r = \mathbf{C}_r \otimes \mathbf{C}_r \otimes \cdots \otimes \mathbf{C}_r. \tag{6.5}$$

The structure of the Chrestenson transform matrix is in the form of a Vandermonde matrix where each row vector consists of component, w_k, raised to an integral power j. The components within a Chrestenson transform matrix, w_k, are one of the r^{th} roots of unity raised to some integral power (Chrestenson et al., 1955; Zilic and Radecka, 2007). The r^{th} roots of unity may be geometrically envisioned as r points that lie upon the unit circle in the complex plane that are equiangular and that always include the $(1, 0)$ point denoted as w_0

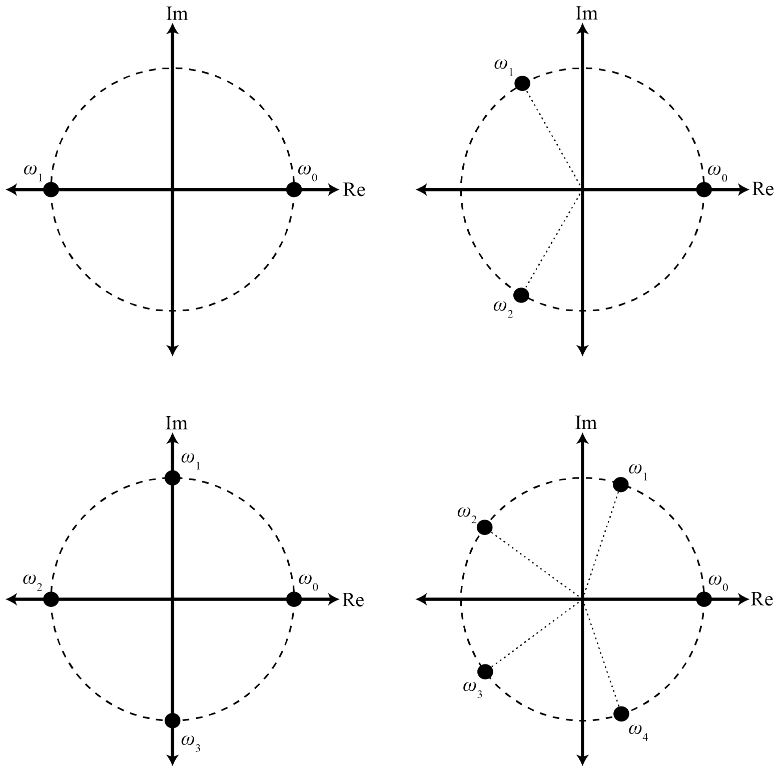


Figure 6.3. Roots of unity in the complex plane for $r = 2, 3, 4$, and 5.

along the positive real axis. In general, for radix-r, the roots of unity of interest are denoted as w_k where $k = 0, 1, \ldots, (r-1)$ and satisfy $(w_k)^r = 1$ as roots of one. Fig. 6.3 contains the plots for the r^{th} roots of unity for $r = 2, 3, 4$, and 5. The closed form representation of the r^{th} roots of unity is

$$w_k = e^{i\frac{2\pi}{r}\times k}. \tag{6.6}$$

Regarding notation, each element of the matrix is some form of w_k^j where j is determined by the column index and k is determined by the row index. In this indexing scheme, the indices j and k begin with $j = k = 0$ and increase to $j = k = (r-1)$. It is observed that, for the case $r = 2$, the Hadamard matrix results. Thus, the Chrestenson transform matrices can be considered as generalizations of the Hadamard transform for higher-dimensioned systems.

The generalized Chrestenson transform matrix, $\mathbf{C}_r$, is

$$\mathbf{C}_r = \frac{1}{\sqrt{r}} \begin{bmatrix} w_0^0 & w_0^1 & \dots & w_0^{(r-1)} \\ w_1^0 & w_1^1 & \dots & w_1^{(r-1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{(r-1)}^0 & w_{(r-1)}^1 & \dots & w_{(r-1)}^{(r-1)} \end{bmatrix}. \tag{6.7}$$

The transformation matrix of Eqn. 6.7 is comprised of a set of normalized orthogonal Chrestenson functions as the column or row vectors (Chrestenson et al., 1955). As is similar to the Hadamard gate acting on a qubit, the radix-r Chrestenson gate evolves a radix-r qudit into a state of maximal superposition when the qudit is initialized to a basis state. Using Eqn. 6.7, radix-3 the Chrestenson transformation matrix is calculated as

$$\mathbf{C}_3 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{\frac{i2\pi}{3}\times 1} & e^{\frac{i2\pi}{3}\times 2} \\ 1 & e^{\frac{i2\pi}{3}\times 2} & e^{\frac{i2\pi}{3}\times 4} \end{bmatrix} \tag{6.8}$$

while the radix-4 the Chrestenson transformation matrix is calculated as

$$\mathbf{C_4} = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}. \tag{6.9}$$

The Chrestenson gate has been physically implemented. An example implementation of the radix-4 Chrestenson gate can be found in references (Smith et al., 2018a,b) and in Appendix A.

6.3. Single Qudit Basis Permutation

Many single qubit operators can be generalized into a higher-radix form to evolve qudit state. An example of such an operator is the radix-2 **X** operation, or **NOT** operation, that performs a Pauli-**X** rotation on a qubit. The quantum gate or operator for the Pauli-**X** is represented with the transformation matrix

$$\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{6.10}$$

Mathematically, the Pauli-**X** operation can be considered a modulo-2 addition-by-one operation as it transforms a qubit $|0_2\rangle$ to be $|((0+1)\text{mod } 2)_2\rangle = |1_2\rangle$ and $|1_2\rangle$ to $|((1+1)\text{mod } 2)_2\rangle = |0_2\rangle$. In the case where $|\phi_2\rangle$ is in a state of superposition, $|\phi_2(t_0)\rangle = a_0|0_2\rangle + a_1|1_2\rangle$, the **X** operation exchanges the probability amplitude coefficients of the quantum state yielding the qubit $|\phi_2(t_1)\rangle = a_1|0_2\rangle + a_0|1_2\rangle$. Thus, the Pauli-**X** gate can be understood as a basis permutation operation due to modulo-k addition with respect to modulus $r = 2$.

The single qudit modulo-addition operations are denoted as $\mathbf{M}_k$ for operators that cause a modulo-k addition with respect to modulus r as was used in (Thornton et al., 2008). These modulo-addition operators that cause a change of basis are also referred to in the literature as Heisenberg-Weyl operators (Bertlmann and Krammer, 2008). Using the $\mathbf{M}_k$ notation, the Pauli-**X** operator for qubits is $\mathbf{M}_1$. As a note, the modulo-0 operation for any r is equal to the identity function, or $\mathbf{M}_0 = \mathbf{I}_r$ where $\mathbf{I}_r$ is the $r \times r$ identity matrix. Therefore, the operation of $\mathbf{M}_0$ is considered trivial since no change of basis occurs for the probability amplitudes within the quantum state. The notation of implementing $\mathbf{M}_0$ as the identity transformation matrix may be used within equations rather than $\mathbf{I}_r$ to show patterns within qudit transformation functions.

To demonstrate the non-trivial single qudit modulo-addition operations in the ternary, $r = 3$, case, consider the transformation matrices

$$\mathbf{M}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \tag{6.11}$$

The $\mathbf{M}_1$ operator causes the evolutions $|0_3\rangle \rightarrow |1_3\rangle$, $|1_3\rangle \rightarrow |2_3\rangle$, and $|2_3\rangle \rightarrow |0_3\rangle$ to occur. Likewise $\mathbf{M}_2$ results in $|0_3\rangle \rightarrow |2_3\rangle$, $|1_3\rangle \rightarrow |0_3\rangle$, and $|2_3\rangle \rightarrow |1_3\rangle$.

For higher-dimensional systems with radix-r, $r > 2$, there are $r - 1$ different single non-trivial $\mathbf{M}_k$ operators. Thus, an $r = 4$ quantum system would have a total of three non-trivial modulo-addition operations. The non-trivial $\mathbf{M}_k$ gates are

$$\mathbf{M}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{M}_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{6.12}$$

The radix-4 modulo-addition operation $\mathbf{M}_1$ causes the evolutions $|0_4\rangle \rightarrow |1_4\rangle$, $|1_4\rangle \rightarrow |2_4\rangle$, $|2_4\rangle \rightarrow |3_4\rangle$, and $|3_4\rangle \rightarrow |0_4\rangle$ to occur. $\mathbf{M}_2$ results in $|0_4\rangle \rightarrow |2_4\rangle$, $|1_4\rangle \rightarrow |3_4\rangle$, $|2_4\rangle \rightarrow |0_4\rangle$, and $|3_4\rangle \rightarrow |1_4\rangle$ to occur. Finally, $\mathbf{M}_3$ results in $|0_4\rangle \rightarrow |3_4\rangle$, $|1_4\rangle \rightarrow |0_4\rangle$, $|2_4\rangle \rightarrow |1_4\rangle$, and $|3_4\rangle \rightarrow |2_4\rangle$.

6.4. Controlled Qudit Operators

Single qudit operators can all be implemented in controlled variations to make multi-qudit quantum gates. For instance, the $\mathbf{M}_k$ operation discussed in Section 6.3 can be modified to execute on a target qudit if and only if a control qudit has a probability amplitude for a specific basis state. For example, the controlled version of the **X** gate is the "controlled-**X**" or "controlled-NOT" gate denoted as **CNOT**. The controlled-NOT gate may also be referred to by the somewhat unconventional name of "controlled-modulo-add by one" gate as it acts as a controlled $\mathbf{M}_1$ operation. The $\mathbf{C}_{NOT}$ gate is defined as

$$\mathbf{C}_{NOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6.13}$$

and causes a Pauli-**X** operation on a target qubit if the control has a probability amplitude for $|1_2\rangle$.

In the case of radix-2 systems, only two different modulo-2 additions are possible since there are two computational basis vectors, $|0_2\rangle$ and $|1_2\rangle$. Furthermore, one of these is the trivial case of modulo-2 addition-by-zero that results in the identity transformation matrix and is not of interest. Thus, there is only one significant single-qubit modulo-2 addition operation, the Pauli-**X** operation. Likewise, there is only one two-qubit controlled-modulo-addition operation of interest, the controlled-**X** or **CNOT**. For the sake of completeness, it is noted that the single $r = 2$ controlled modulo-addition operation of interest, **CNOT**, could be considered to represent two different operations since the control qubit may cause the target Pauli-**X** operation to occur when the control qubit is either $|1_2\rangle$ or $|0_2\rangle$. Most past work in binary QIS consider only the single **CNOT** operator wherein the target is activated when the control is $|1_2\rangle$. If the other case is of interest, it is represented as the **CNOT** with the control qubit passing through a pair of Pauli-**X** operations, one before and one after the **CNOT**, to cause the target to activate when the control has value $|0_2\rangle$. This is generally the case to account for common calculations of quantum cost. However, for consistency, these two cases for $r = 2$ are considered since the generalized controlled modulo-addition operations for qudits where $r > 2$ that are discussed later do consider different values of the control qudits that activate the target operation.

A controlled modulo-addition-by-k gate is a two-qudit gate specified as $\mathbf{A}_{h,k}$. The target modulo-addition-by-k operation occurs on the target qudit when, and only when, the control qudit has the appropriate value as specified by the gate as h. The k value defines the modulo addition operation to occur on the target. A single controlled modulo-addition operator is permissible and non-trivial when the control value that activates the target modulo-addition operator is any basis state $|h\rangle$ where $h \in \{0, \dots, (r-1)\}$ and the target modulo-addition-by-k operation is restricted to non-zero values of k where $k \in \{1, \dots, (r-1)\}$. Since the modulo-r addition-by-zero operation results in the radix-r identity transformation matrix, $\mathbf{I}_r$, controlled modulo-r addition-by-zero gates are considered trivial. Controlled modulo-addition gates with multiple control and addition-by-k values are composite controlled-mod-add operators.

For higher-dimensional systems with radix-r, $r > 2$, there are $r-1$ different single non-trivial qudit modulo-r additions and, thus, $r-1$ controlled-modulo-addition operations of interest with respect to modulus r for a single basis control value. Considering all combinations of control values, r, as well as the different moduli, $r-1$, there are a total of $r^2 - r$ different and non-trivial controlled-modulo-addition operators with all possible control values. Since the controlled-modulo-addition transformation matrices, $\mathbf{A}_{h,k}$, operate over two qudits of arbitrary radix r, they are of dimension $r^2 \times r^2$.

In general, the radix-r controlled modulo-addition-k matrix, $\mathbf{A}_{h,k}$, where h and k have a single value, is in the form

$$\mathbf{A}_{h,k} = \begin{bmatrix} \mathbf{D}_0 & \mathbf{0}_r & \cdots & \cdots & \cdots & \cdots & \mathbf{0}_r \\ \mathbf{0}_r & \mathbf{D}_1 & \mathbf{0}_r & \cdots & \cdots & \cdots & \mathbf{0}_r \\ \vdots & \mathbf{0}_r & \ddots & \mathbf{0}_r & \cdots & \cdots & \mathbf{0}_r \\ \vdots & \vdots & \mathbf{0}_r & \mathbf{D}_j & \mathbf{0}_r & \cdots & \mathbf{0}_r \\ \vdots & \vdots & \vdots & \mathbf{0}_r & \ddots & \mathbf{0}_r & \vdots \\ \vdots & \vdots & \vdots & \vdots & \mathbf{0}_r & \ddots & \mathbf{0}_r \\ \mathbf{0}_r & \mathbf{0}_r & \mathbf{0}_r & \mathbf{0}_r & \cdots & \mathbf{0}_r & \mathbf{D}_{(r-1)} \end{bmatrix}, \quad \mathbf{D}_i = \begin{cases} \mathbf{M}_0 = \mathbf{I}_r, & i \neq h \\ \mathbf{M}_k, & i = h. \end{cases} \tag{6.14}$$

The $\mathbf{A}_{h,k}$ matrix is banded with $r-1$ non-zero super- and sub-diagonals. In Eqn. 6.14, each submatrix along the diagonal is denoted as $\mathbf{D}_i$ and is of dimension $r \times r$. The two-qudit controlled variation of the modulo-add gate, $\mathbf{A}_{h,k}$, only allows the modulo-addition by k operation to occur on the target whenever the control qudit is in state, $|h_r\rangle$. The control qudit can, in general, be in a superimposed state.

Considering radix-3 for an example, there are a total of six non-trivial variations of the controlled modulo-add gate. These are denoted as $\mathbf{A}_{0,1}$, $\mathbf{A}_{0,2}$, $\mathbf{A}_{1,1}$, $\mathbf{A}_{1,2}$, $\mathbf{A}_{2,1}$, and $\mathbf{A}_{2,2}$. The transformation matrices for these operations are

$$\mathbf{A}_{0,1} = \left[\begin{array}{ccc:ccc:ccc} 0&0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ \hdashline 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ \hdashline 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1 \end{array}\right], \mathbf{A}_{0,2} = \left[\begin{array}{ccc:ccc:ccc} 0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0\\ \hdashline 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ \hdashline 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1 \end{array}\right], \qquad (6.15)$$

$$\mathbf{A}_{1,1} = \left[\begin{array}{ccc:ccc:ccc} 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0\\ \hdashline 0&0&0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ \hdashline 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1 \end{array}\right], \mathbf{A}_{1,2} = \left[\begin{array}{ccc:ccc:ccc} 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0\\ \hdashline 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ \hdashline 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1 \end{array}\right], \qquad (6.16)$$

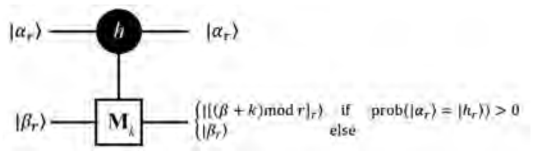

$|\alpha_r\rangle$
h
$|\alpha_r\rangle$
$|\beta_r\rangle$
$\mathbf{M}_k$
$\begin{cases} |[(\beta + k) \bmod r]_r\rangle & \text{if} \quad \text{prob}(|\alpha_r\rangle = |h_r\rangle) > 0 \\ |\beta_r\rangle & \text{else} \end{cases}$

Figure 6.4. Symbol of the controlled modulo-add gate, $\mathbf{A}_{h,k}$.

$$\mathbf{A}_{2,1} = \left[\begin{array}{ccc:ccc:ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right], \mathbf{A}_{2,2} = \left[\begin{array}{ccc:ccc:ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right]. \tag{6.17}$$

After examining $\mathbf{A}_{0,1}$ from Eqn. 6.15, it is observed that the operation only allows $\mathbf{M}_1$ to execute on the target if the control qudit has a value of $|0_3\rangle$. If the control qudit is either $|1_3\rangle$ or $|2_3\rangle$, the target qudit remains unchanged. If another logic level is added and a radix-4 QIP system is considered, the available controlled-modulo add operations become $\mathbf{A}_{0,1}$, $\mathbf{A}_{0,2}$, $\mathbf{A}_{0,3}$, $\mathbf{A}_{1,1}$, $\mathbf{A}_{1,2}$, $\mathbf{A}_{1,3}$, $\mathbf{A}_{2,1}$, $\mathbf{A}_{2,2}$, $\mathbf{A}_{2,3}$, $\mathbf{A}_{3,1}$, $\mathbf{A}_{3,2}$, and $\mathbf{A}_{3,3}$. The transformation matrices for these two-qudit operators can be derived with Eqn. 6.14.

The symbol for the generalized circuit for controlled modulo-add gate can be seen in Fig. 6.4. In this generalized symbol, the h for the control qudit $|\alpha_r\rangle$ is the active control level that may have a value from the set $\{0, 1, \ldots, (r-1)\}$. Likewise, the target qudit $|\beta_r\rangle$ will be transformed by $\mathbf{M}_k$ where k has a value from the set $\{1, 2, \ldots, (r-1)\}$.

Chapter 7

Higher Dimensioned Entanglement Generators

The benefits of QIS depend on entanglement, and without entanglement coupled with superposition, QIS offers no benefit in comparison to Turing machines. There are very few known and useable QC algorithms that do not employ entanglement in some form or fashion because quantum entanglement is an enabling property in many algorithms and communications schemes. For example, many implementations of QKD are dependent on the ability to generate entangled states. Furthermore, entanglement is a key phenomenon that is present inside many well-known binary quantum algorithms such as in the QFT, Deutsch-Jouza, Simon's, Grover's, Shor's algorithms. Virtually none of the currently known binary QIS methods can be extended to higher radices without entanglement generators. In addition, entanglement is a key ingredient in all of the logical qubit and qudit formulation methods. Thus, for higher-radix QIP to become viable, it is important to determine methods to generate entanglement for higher-radix systems. Here, a focus will be placed on generalizations of the binary bipartite entanglement generators such as Bell generators and the multi-qubit Greenberger–Horne–Zeilinger (GHZ) generators for the higher-radix case. Included results provide a description of how higher-radix entanglement generators can be constructed from realizable physical components including the Chrestenson and controlled-modulo-addition operators.

Properties of maximum qudit entanglement have been studied in (Enríquez et al., 2016; Thew et al., 2002). Qudit entangled states have also been experimentally demonstrated (Kues et al., 2017). However, a general methodology for the synthesis of a state preparation algorithm to yield a maximally entangled state has not been clearly defined until (Smith and

Thornton, 2019a,c). In this section, the single- and multi-qudit operators required to entangle quantum states will be presented. The structure of the bipartite binary entanglement generator provides inspiration for the development of this theory. The end result of this section is a generalized circuit structure that can be implemented in QIP algorithms to create varying degrees of higher-radix entanglement. To demonstrate the methods, $r = 3$ systems will be used. Additional examples of entanglement generators for $r = 4$ are included in reference (Smith and Thornton, 2019c).

7.1. Partial Entanglement of Qudit Pairs

The the maximally entangled states for two qubits are known as the Bell states, and as seen in Section 2.7, entangled qubit pairs are generated with the Bell state generator. When preparing entangled quantum states, there are two key transformations that must occur in series. First, one of the quantum elements must be placed into a state of maximal superposition with a radix-r Chrestenson gate. Next, the two quantum elements must interact together via controlled modulo-addition operation or operations.

Transforming qubits with the Bell state generator creates maximal entanglement. The resulting Bell states are entangled in the sense that the wavefunction vector does not contain groups of separable basis states. Although they do not appear in binary systems, states of partial entanglement exist when $r > 2$. A partially entangled state is one wherein some subset of the basis states are entangled and the remaining set or sets are not. As a system's radix increases in size, greater degrees partial entanglement are possible.

Using the structure of the well-known radix-2 Bell state generator as motivation, quantum generator circuits can be formulated as a cascade that includes a radix-r Chrestenson operator followed by a controlled-modulo add-operator. This set of operations evolves a pair of $r > 2$ qudits originally in a basis state into a partially, but not maximally, entangled qudit pair. The general form of a particular radix-r partial entanglement generator is shown in Fig. 7.1:a. There exist $r^2 - r$ different partial entanglement generators with a single controlled

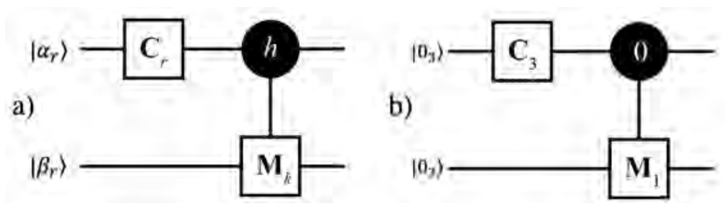


Figure 7.1. a) General circuit for radix-r two-qudit partial entanglement generator. b) Specific example circuit for radix-3 two-qudit partial entanglement generator.

gate as any non-trivial $\mathbf{A}_{h,k}$ operator can be used.

To illustrate the concept of partial entanglement, consider the case where $r = 3$ and a controlled-modulo add operator of the form $\mathbf{A}_{0,1}$ is utilized. Furthermore, assume that the initial quantum state of the radix-3 qudit pair is $|\alpha\beta_3\rangle = |00_3\rangle$. The specific partial entanglement generator is shown in Fig. 7.1:b. The resulting partially entangled quantum state arising from the evolution of $|00_3\rangle$ through the radix-3 circuit of Fig. 7.1:b is calculated as $|\alpha\beta_3\rangle = \mathbf{T}_{par}\,|00_3\rangle$ where $\mathbf{T}_{par}$ is the transfer matrix of the partial entanglement generator. The partially entangled state is calculated as

$$\begin{aligned}\mathbf{T}_{par}\,|00_3\rangle &= \mathbf{A}_{0,1}(\mathbf{C}_3 \otimes \mathbf{I}_3)\,|00_3\rangle \\ &= \frac{1}{\sqrt{3}}\left(|01_3\rangle + |10_3\rangle + |20_3\rangle\right) \\ &= \frac{1}{\sqrt{3}}\big[\,|01_3\rangle + (|1_3\rangle + |2_3\rangle) \otimes |0_3\rangle\,\big].\end{aligned} \tag{7.1}$$

Since the value $|0_3\rangle$ can be factored out of two of the three basis components in the evolved quantum state, the state is not fully entangled. However, this state is referenced as "partially entangled" since the basis $|01_3\rangle$ is present. $|01_3\rangle$ can be considered an entangled basis state within the partially entangled output state in Eqn. 7.1 because measurement of either $|\alpha_3\rangle$ or $|\beta_3\rangle$ gives insight to the state of the other qudit. Thus, there is a probability of $\frac{1}{3}$ that an observation or measurement of the evolved state will be this entangled state. However, if the evolved form of qudit $|\beta_3\rangle$ is observed to be $|0_3\rangle$, then the evolved qudit $|\alpha_3\rangle$ may be either

$|1_3\rangle$ or $|2_3\rangle$ with equal likelihood, thus violating the definition of maximal entanglement. Mathematically, partial entanglement is present due to the fact that $|0_3\rangle$ can be factored out of two of the components of the evolved state. In contrast, a maximally entangled state is one where no such factoring of the basis values within the quantum state vector is possible.

Because the radix-3 quantum logic has three basis states that can act as active control values for the controlled-mod-add operators and there are two non-trivial modulo-add gates (*i.e.*, $\mathbf{A}_{h,1}$ and $\mathbf{A}_{h,2}$ for $h \in \{0, 1, 2\}$), there are six different circuits that could be used to create partial entanglement. The evolved states resulting from the radix-3 partial entanglement circuit can be seen in Table 7.1 where $|0_3\rangle$ is the control basis value, Table 7.2 where $|1_3\rangle$ is the control basis value, and Table 7.3 where $|2_3\rangle$ is the control basis value. In these tables, the control level allows either the modulo-add by one or modulo-add by two function to act upon the target qudit. All of the states provided in these tables are only partially entangled because a qudit can be factored out of a subset of the final quantum state's basis values, violating the definition of maximal entanglement.

7.2. Maximal Entanglement Generators for Qudit Pairs

Many radix-2 QIP algorithms begin with initializing the qubits in a ground or other basis state followed by placing them into states of full and maximal entanglement. This is accomplished by using a quantum circuit that includes the Hadamard gate and the **CNOT**, or radix-2 controlled-modulo-one, operator. If a QC is a higher-radix device, then the analogous operation would be instantiated. That is, to first initialize all qudits into basis states and then to immediately perform a Chrestenson operation to evolve the control qudit into a

Table 7.1. Outputs of radix-3 partial entanglement generator circuit with $|0_3\rangle$ as control level

	Two-Qudit Gate in Generator	
Input	$\mathbf{A}_{0,1}$	$\mathbf{A}_{0,2}$
$\|00_3\rangle$	$\frac{1}{\sqrt{3}}(\|01_3\rangle + \|10_3\rangle + \|20_3\rangle)$	$\frac{1}{\sqrt{3}}(\|02_3\rangle + \|10_3\rangle + \|20_3\rangle)$
$\|01_3\rangle$	$\frac{1}{\sqrt{3}}(\|02_3\rangle + \|11_3\rangle + \|21_3\rangle)$	$\frac{1}{\sqrt{3}}(\|00_3\rangle + \|11_3\rangle + \|21_3\rangle)$
$\|02_3\rangle$	$\frac{1}{\sqrt{3}}(\|00_3\rangle + \|12_3\rangle + \|22_3\rangle)$	$\frac{1}{\sqrt{3}}(\|01_3\rangle + \|12_3\rangle + \|22_3\rangle)$
$\|10_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|20_3\rangle\right)$
$\|11_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|21_3\rangle\right)$
$\|12_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|22_3\rangle\right)$
$\|20_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|20_3\rangle\right)$
$\|21_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|21_3\rangle\right)$
$\|22_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|22_3\rangle\right)$

Table 7.2. Outputs of radix-3 partial entanglement generator circuit with $|1_3\rangle$ as control level

	Two-Qudit Gate in Generator	
Input	$\mathbf{A}_{1,1}$	$\mathbf{A}_{1,2}$
$\|00_3\rangle$	$\frac{1}{\sqrt{3}}(\|00_3\rangle + \|11_3\rangle + \|20_3\rangle)$	$\frac{1}{\sqrt{3}}(\|00_3\rangle + \|12_3\rangle + \|20_3\rangle)$
$\|01_3\rangle$	$\frac{1}{\sqrt{3}}(\|01_3\rangle + \|12_3\rangle + \|21_3\rangle)$	$\frac{1}{\sqrt{3}}(\|01_3\rangle + \|10_3\rangle + \|21_3\rangle)$
$\|02_3\rangle$	$\frac{1}{\sqrt{3}}(\|02_3\rangle + \|10_3\rangle + \|22_3\rangle)$	$\frac{1}{\sqrt{3}}(\|02_3\rangle + \|11_3\rangle + \|22_3\rangle)$
$\|10_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|20_3\rangle\right)$
$\|11_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|21_3\rangle\right)$
$\|12_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|22_3\rangle\right)$
$\|20_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|20_3\rangle\right)$
$\|21_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|21_3\rangle\right)$
$\|22_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|22_3\rangle\right)$

state of maximal superposition. Next, specific controlled operations may be implemented to evolve the qudit pair into partial or maximal entanglement. Augmentations must be made to the circuit in Fig. 7.1:a in order to produce maximally entangled radix-r qudit pairs. A pair of radix-3 qudits can become maximally entangled when two controlled modulo-add operations are utilized rather than a single operation.

Fully entangling a pair of radix-3 qudits requires one additional controlled operation in the entanglement generator as compared to what is necessary for partial entanglement. The two controlled gates needed for full entanglement must have different target activation values, h, and they must have two different modulo-add by k operations on the target.

An example radix-3 full entanglement generator for two qudits can be seen in Fig. 7.2. The transformation matrix for the controlled operations in the generator is derived by combining the single-control modulo-add transformation functions, $\mathbf{A}_{1,1}$ in series with $\mathbf{A}_{2,2}$ using a matrix product. For example,

$$\mathbf{A}_{(1,2),(1,2)} = \mathbf{A}_{2,2} \times \mathbf{A}_{1,1} = \left[\begin{array}{ccc:ccc:ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] = \begin{bmatrix} \mathbf{M}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M}_2 \end{bmatrix} \tag{7.2}$$

is created by combining $\mathbf{A}_{2,2}$ with $\mathbf{A}_{1,1}$, and it describes the evolution associated with the entire controlled portion of Fig. 7.2. Since combining the controlled-mod-add operations is commutative, the matrix products $\mathbf{A}_{2,2} \times \mathbf{A}_{1,1} = \mathbf{A}_{(1,2),(1,2)}$, $\mathbf{A}_{2,1} \times \mathbf{A}_{1,2} = \mathbf{A}_{(1,2),(2,1)}$, $\mathbf{A}_{2,2} \times \mathbf{A}_{0,1} = \mathbf{A}_{(0,2),(1,2)}$, $\mathbf{A}_{2,1} \times \mathbf{A}_{0,2} = \mathbf{A}_{(0,2),(2,1)}$, $\mathbf{A}_{1,2} \times \mathbf{A}_{0,1} = \mathbf{A}_{(0,1),(1,2)}$, and $\mathbf{A}_{1,1} \times \mathbf{A}_{0,2} =$

$\mathbf{A}_{(0,1),(2,1)}$ describe all of the unique transformation functions that can be used as the multi-qudit gates of a radix-3 maximal entanglement generator for two qudits. All of the outputs created by these six different maximal entanglement generators are provided in Tables 7.4, 7.5, and 7.6. Table 7.4 contains information when $|1_3\rangle$ and $|2_3\rangle$ are the active control values, Table 7.5 contains information when $|0_3\rangle$ and $|2_3\rangle$ are the active controls, and Table 7.6 contains information when $|0_3\rangle$ and $|1_3\rangle$ are the active controls. If these tables are examined, it is clear that they contain maximally entangled quantum states since each qudit cannot be described independently from the pair. For example, the evolved state resulting from an initial state of $|00_3\rangle$ where the entanglement generator $\mathbf{T}_{max}$ from Fig. 7.2 containing the controlled modulo-add gate $\mathbf{A}_{(1,2),(1,2)} = \mathbf{A}_{2,2} \times \mathbf{A}_{1,1}$ is provided in Table 7.4. The output qudit state corresponding to the input $|00_3\rangle$ for this maximum entanglement generator would be calculated as

$$\mathbf{T}_{max} |00_3\rangle = \mathbf{A}_{(1,2),(1,2)}(\mathbf{C_3} \otimes \mathbf{I_3}) |00_3\rangle = \frac{1}{\sqrt{3}} \left(|00_3\rangle + |11_3\rangle + |22_3\rangle\right). \tag{7.3}$$

This generated state cannot be mathematically factored and can only be described as a summation of entangled basis states. Since the output of the two-qudit maximal entanglement generator agrees with the definition of maximal entanglement, the pair of radix-3 qudits are maximally entangled.

Fig. 7.2 illustrates the previously described maximal entanglement generator. In Fig. 7.2, the two controlled-mod-add gates are shown in two ways: as a single symbol and as to two separate symbols. Although these two graphical depictions are identical, it may be advantageous to implement them as a single operation in certain technologies. Additionally,

Table 7.3. Outputs of radix-3 partial entanglement generator circuit with $\lvert 2_3\rangle$ as control level

	Two-Qudit Gate in Generator	
Input	$\mathbf{A}_{2,1}$	$\mathbf{A}_{2,2}$
$\lvert 00_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \lvert 10_3\rangle + \lvert 21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \lvert 10_3\rangle + \lvert 22_3\rangle\right)$
$\lvert 01_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \lvert 11_3\rangle + \lvert 22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \lvert 11_3\rangle + \lvert 20_3\rangle\right)$
$\lvert 02_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \lvert 12_3\rangle + \lvert 20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \lvert 12_3\rangle + \lvert 21_3\rangle\right)$
$\lvert 10_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 22_3\rangle\right)$
$\lvert 11_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 20_3\rangle\right)$
$\lvert 12_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 21_3\rangle\right)$
$\lvert 20_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 22_3\rangle\right)$
$\lvert 21_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 20_3\rangle\right)$
$\lvert 22_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 21_3\rangle\right)$

Table 7.4. Outputs of radix-3 maximal entanglement generator circuit with $\lvert 1_3\rangle$ and $\lvert 2_3\rangle$ as control levels

	Two-Qudit Gate in Generator	
Input	$\mathbf{A}_{2,2} \times \mathbf{A}_{1,1} = \mathbf{A}_{(1,2),(1,2)}$	$\mathbf{A}_{2,1} \times \mathbf{A}_{1,2} = \mathbf{A}_{(1,2),(2,1)}$
$\lvert 00_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \lvert 11_3\rangle + \lvert 22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \lvert 12_3\rangle + \lvert 21_3\rangle\right)$
$\lvert 01_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \lvert 12_3\rangle + \lvert 20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \lvert 10_3\rangle + \lvert 22_3\rangle\right)$
$\lvert 02_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \lvert 10_3\rangle + \lvert 21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \lvert 11_3\rangle + \lvert 20_3\rangle\right)$
$\lvert 10_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 21_3\rangle\right)$
$\lvert 11_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 22_3\rangle\right)$
$\lvert 12_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 20_3\rangle\right)$
$\lvert 20_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 21_3\rangle\right)$
$\lvert 21_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 22_3\rangle\right)$
$\lvert 22_3\rangle$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\lvert 02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\lvert 11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\lvert 20_3\rangle\right)$

Table 7.5. Outputs of radix-3 maximal entanglement generator circuit with $|0_3\rangle$ and $|2_3\rangle$ as control levels

	Two-Qudit Gate in Generator	
Input	$\mathbf{A}_{2,2} \times \mathbf{A}_{0,1} = \mathbf{A}_{(0,2),(1,2)}$	$\mathbf{A}_{2,1} \times \mathbf{A}_{0,2} = \mathbf{A}_{(0,2),(2,1)}$
$\|00_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \|10_3\rangle + \|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \|10_3\rangle + \|21_3\rangle\right)$
$\|01_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \|11_3\rangle + \|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \|11_3\rangle + \|22_3\rangle\right)$
$\|02_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \|12_3\rangle + \|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \|12_3\rangle + \|20_3\rangle\right)$
$\|10_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|21_3\rangle\right)$
$\|11_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|22_3\rangle\right)$
$\|12_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|20_3\rangle\right)$
$\|20_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|21_3\rangle\right)$
$\|21_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|22_3\rangle\right)$
$\|22_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|20_3\rangle\right)$

Table 7.6. Outputs of radix-3 maximal entanglement generator circuit with $|0_3\rangle$ and $|1_3\rangle$ as control levels

	Two-Qudit Gate in Generator	
Input	$\mathbf{A}_{1,2} \times \mathbf{A}_{0,1} = \mathbf{A}_{(0,1),(1,2)}$	$\mathbf{A}_{1,1} \times \mathbf{A}_{0,2} = \mathbf{A}_{(0,1),(2,1)}$
$\|00_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \|12_3\rangle + \|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \|11_3\rangle + \|20_3\rangle\right)$
$\|01_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \|10_3\rangle + \|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \|12_3\rangle + \|21_3\rangle\right)$
$\|02_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \|11_3\rangle + \|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \|10_3\rangle + \|22_3\rangle\right)$
$\|10_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|20_3\rangle\right)$
$\|11_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|21_3\rangle\right)$
$\|12_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|22_3\rangle\right)$
$\|20_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|20_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|20_3\rangle\right)$
$\|21_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|02_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|21_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|12_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|21_3\rangle\right)$
$\|22_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|00_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|11_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|22_3\rangle\right)$	$\frac{1}{\sqrt{3}}\left(\|01_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\,\|10_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\,\|22_3\rangle\right)$

combining the controlled-mod-add gates into a single symbol allows for more concise circuit diagrams. The composite and single transformation matrix in Eqn. 7.2 characterizes the controlled portion, $\mathbf{A}_{(1,2),(1,2)}$, of Fig. 7.2. As previously discussed for a radix-3 two-qudit maximal entanglement generator, there are 6 different composite controlled-mod-add operators that can be implemented. These are the $\mathbf{A}_{(0,1),(1,2)}$, $\mathbf{A}_{(0,1),(2,1)}$, $\mathbf{A}_{(1,2),(1,2)}$, $\mathbf{A}_{(1,2),(2,1)}$, $\mathbf{A}_{(0,2),(1,2)}$, and $\mathbf{A}_{(0,2),(1,2)}$ operators.

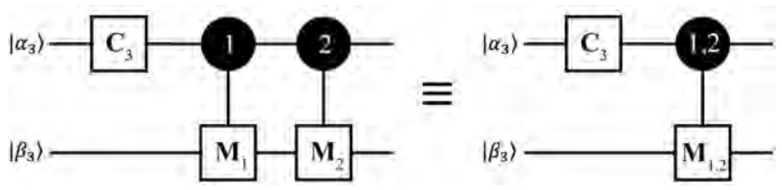


Figure 7.2. Radix-3 two-qudit maximal entanglement generator implemented with $\mathbf{A}_{1,1}$ and $\mathbf{A}_{2,2}$ that form the composite gate $\mathbf{A}_{(1,2),(1,2)}$.

Observation of the radix-3 composite controlled-mod-add operators reveals a pattern. A radix-3 maximal entanglement generator for two qudits can be formulated as a cascade of a single qudit Chrestenson operator, $\mathbf{C}_3$, and a composite controlled modulo-add operator, $\mathbf{A}_{(h_1,h_2),(k_1,k_2)}$. The qudit that is transformed by the Chrestenson operator acts as the control qudit . The generalized form of the composite controlled modulo-add operator, $\mathbf{A}_{(h_1,h_2),(k_1,k_2)}$, for $r = 3$ is defined as

$$\mathbf{A}_{(h_1,h_2),(k_1,k_2)} = \left[\begin{array}{c:c:c} \mathbf{D}_0 & \mathbf{0}_3 & \mathbf{0}_3 \\ \hdashline \mathbf{0}_3 & \mathbf{D}_1 & \mathbf{0}_3 \\ \hdashline \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{D}_2 \end{array}\right], \qquad \mathbf{D}_i = \begin{cases} \mathbf{M}_0, & i \neq h_1, h_2 \\ \mathbf{M}_{k_1}, & i = h_1 \\ \mathbf{M}_{k_2}, & i = h_2 \end{cases} \tag{7.4}$$

It is assumed that the entanglement generator evolves the quantum state $|\theta\phi_3(t_0)\rangle = |(i,j)_3\rangle$ where $(i,j) = \{0,1,2\} \times \{0,1,2\}$. That is, the original quantum state is initialized to a basis state. It is further assumed without loss of generality that the control qudit is

$|\theta_3\rangle$ and the target qudit is $|\phi_3\rangle$ when referring to the state evolution due to a controlled modulo-add operator.

It is assumed that the control qudit $|\theta_3\rangle$ is in a state of maximal superposition before it is applied to the controlled modulo-addition operators as is consistent with the architecture of an entanglement generator. Thus, the control qudit $|\theta_3\rangle$ is mathematically in the form

$$|\theta_3\rangle = \frac{1}{\sqrt{3}}(|0_3\rangle + |1_3\rangle + |2_3\rangle)$$

.

with each basis multiplied by a radix-3 root of unity raised to some integral power depending on the control qudit's original value. Therefore, due to the control value h_1 of a controlled modulo-add operator, partial entanglement occurs between the $|(h_1)_3\rangle$ component of the control qudit $|\theta_3\rangle$ and the target qudit $|\phi_3\rangle$ due to the operation of $\mathbf{A}_{h_1,k_1}$ wherein the $|[(h_1),((k_1+\phi) \mod 3)]_3\rangle$ term becomes entangled only as is shown in the results of Tables 7.1, 7.2, 7.3. Thus, the remaining $r-1=3-1=2$ terms are not entangled.

To entangle the remaining $r-1=2$ terms, it is necessary to apply another $\mathbf{A}_{h_2,k_2}$ operator such that $h_2 \neq h_1$ and $k_2 \neq k_1$ to ensure that non-entangled elements of the partially entangled quantum state are evolved. The result of this evolution causes the $|[(h_2),((k_2+\phi) \mod 3)]_3\rangle$ term to become entangled. Because two of the $r=3$ terms are now entangled, the only possibility for the third term is for it to be in a fully entangled state also to satisfy Born's rule in which the probability of observing a quantum system in a particular state is equal to the magnitude squared of the probability amplitude of a specific basis state in the wavefunction (Born, 1926).

Another result concerns the required number of controlled modulo-addition operators for a general radix-r maximal entanglement generator for two qudits where $r>2$. It is required that $r-1$ unique and permissible controlled-mod-add operators be utilized in a general maximal entanglement generator for two qudits wherein the controlling qudit is provided with a maximally superimposed state. To prove this statement, consider that it is well-known that a Bell state generator can be formed with a single Hadamard gate and a single **CNOT** gate that can be considered to be a controlled modulo-addition-1 gate. This result indicates that maximal entanglement can be achieved for a system where $r = 2$ and the initial quantum state is a basis state when applied to the Bell state generator with $r-1 = 2-1 = 1$ controlled modulo-addition operators.

For a radix-3 system, $r - 1 = 3 - 1 = 2$ controlled-modulo-add operators are required to achieve a state of maximal entanglement in an entanglement generator of the form wherein the control qudit $|\theta_3\rangle$ is in a state of maximal superposition and the target qudit $|\phi_3\rangle$ is initialized to a basis state.

By induction, it is the case that a maximal state generator for qudits of radix-r, require the use of $r - 1$ controlled modulo-addition operators wherein the $r - 1$ controlled modulo-addition operators utilize control values that are mutually exclusive from the set $\{0, 1, 2, \ldots, (r - 1)\}$. Conditions for the the target modulo-addition operators, $\mathbf{M}_j$ are described as a result of the requirements for $A_{h,k}$ gates in a quantum entanglement circuit. A given radix-r, two-qudit maximal entanglement generator is comprised in part of $r - 1$ controlled modulo-addition operators wherein the operators are all non-trivial and permissible and furthermore wherein none of the operators are identical. Consider the group G where the group elements are all $r \times r$ permutation matrices $\mathbf{M}_i$ for $i \in \{0, 1, \ldots, (r - 1)\}$ and the group operator is direct matrix multiplication. Because G is a group, closure holds, thus $\mathbf{M}_i \times \mathbf{M}_i = \mathbf{M}_{(i+i)(\text{mod } r)} = \mathbf{M}_j$, where, $j = (i + i)(\text{mod } r)$.

Consider an attempted maximal entanglement generator comprising in part $r - 1$ controlled modulo-addition operators wherein two of the $r - 1$ operators are identical and of the form $\mathbf{A}_{h,k}$. Since the $\mathbf{M}_i$ permutation matrices in group G are identical to the modulo-addition-by-i transformation matrices in a controlled modulo-addition operator, this result indicates that the presence of two of the same $\mathbf{A}_{h,k}$ operators in a set of $r - 1$ operators comprising an attempted maximal entanglement generator are equivalent to a set of $r - 2$ operators wherein the two identical $\mathbf{A}_{h,k}$ operators are equivalent to a single $\mathbf{A}_{h,2k(\text{mod } r)}$

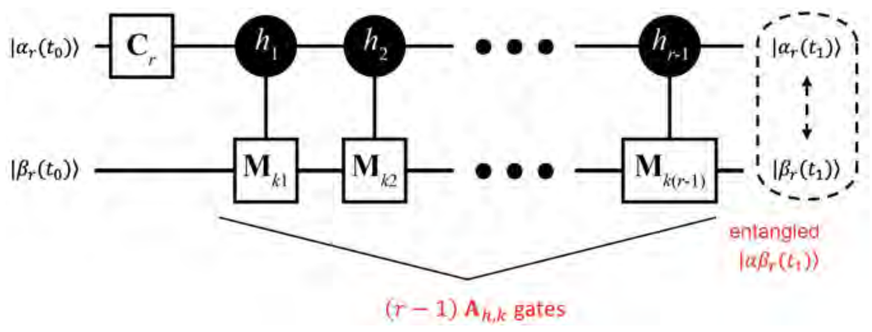


Figure 7.3. Generalized maximal entanglement circuit for a radix-r qudit pair.

operator.

It is proven in (Smith and Thornton, 2019a) that $r-1$ unique and permissible controlled-mod-add operators be utilized within a maximum entanglement generator. This requirement is violated when two identical controlled modulo-addition operators are present in the set of size $r-1$ since two identical operators of the form $\mathbf{A}_{h,k}$ are equivalent to a single operator of the form $\mathbf{A}_{h,2k(\text{mod } r)}$.

The composite controlled mod-add operators in a radix-r maximal entanglement generators are a cascade of $r-1$ permissible and unique controlled-mod-add operators of the form $\mathbf{A}_{h,k}$ where $h \in \{0, \ldots, (r-1)\}$ and $k \in \{1, \ldots, (r-1)\}$. This result leads to the definition of the structure of a radix-r, two-qudit maximal entanglement generator. A radix-r maximal entanglement generator for a radix-r qudit pair can be formed as a series of qudit evolutions in time wherein the first evolution is that resulting from the application of a radix-r Chrestenson gate to the first qudit. Then, the resulting evolved qudit controls the $r-1$ control inputs of the $r-1$ permissible and unique controlled modulo-addition gates where the second qudit acts as the target qudit on the $r-1$ controlled modulo-addition gates. To verify this structure, consider a radix-r qudit initialized to a basis state is evolved to a state of maximal superposition when a radix-r Chrestenson transform is applied. Thus, a qudit in a basis state is maximally superimposed after it is evolved via a $\mathbf{C}_r$ Chrestenson gate.

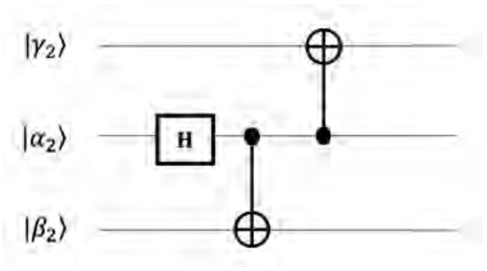


Figure 7.4. Three-qubit GHZ state generator.

A generalized diagram of a radix-r maximal entanglement generator for two qudits is given in Fig. 7.3.

7.3. Maximal Entanglement of Qudit Groups

The Bell state generator for radix-2 quantum logic can be expanded with an additional $\mathbf{A}_{1,1} = \mathbf{CNOT}$ operator and qubit to produce GHZ states. These states, introduced in (Greenberger et al., 1989), are examples of entanglement that involve three or more qubits. A circuit structure for entangling three qubits is pictured in Fig 7.4. To demonstrate how three-qubit entanglement is generated, consider the transformation of $|000_2\rangle$ by the circuit in Fig. 7.4 to produce the fully entangled state of

$$\begin{aligned}\mathbf{GHZ}\,|000_2\rangle &= (rev(\mathbf{CNOT}) \otimes \mathbf{I_2})(\mathbf{I_2} \otimes \mathbf{CNOT})\,|000_2\rangle \\ &= \frac{1}{\sqrt{2}}(|000_2\rangle + |111_2\rangle).\end{aligned} \tag{7.5}$$

As a note, reversing the orientation of an operator causes an interchange of columns in the transformation matrix. A quantum operator $\mathbf{U}$ in its reversed orientation is indicated by $rev(\mathbf{U})$. This notation is seen in Eqn. 7.5.

The higher-radix maximal entanglement generator in Fig. 7.3 is capable of entangling three qudits if minor modifications are made. An example of a $r = 3$, three-qudit maximal entanglement generator is pictured in Fig. 7.5. Many versions of the three-qudit generator

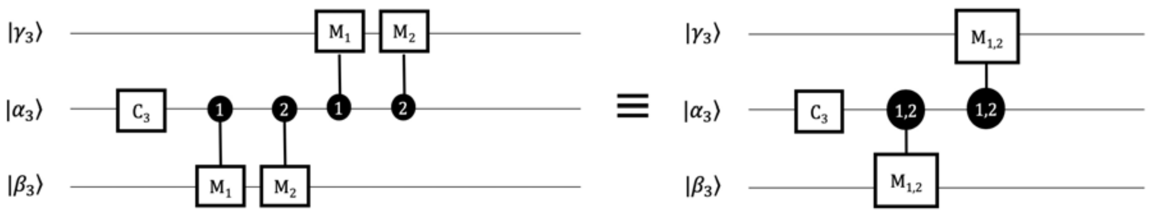


Figure 7.5. Radix-3 three-qudit maximal entanglement generator implemented with two instances of $\mathbf{A}_{1,1} \times \mathbf{A}_{2,2} = \mathbf{A}_{(1,2),(1,2)}$.

can be created depending on the set of permissible controlled modulo-addition operators combined in a composite form to act on each target qudit. By applying an additional set of $\mathbf{A}_{h,k}$ operators to an added qudit, the radix-3, two-qudit maximum entanglement generator of Fig. 7.2 becomes the radix-3, three-qudit maximum entanglement generator pictured in Fig. 7.5. In Fig. 7.5 after the $\mathbf{C}_3$ gate acts on $|\alpha_3\rangle$, the $\mathbf{A}_{h,k}$ operators acting on $|\alpha_3\rangle$ and $|\beta_3\rangle$ are the same with respect to controls and modulo-adds as the operators acting on $|\alpha_3\rangle$ and $|\gamma_3\rangle$. The second set of gates acting on $|\alpha_3\rangle$ and $|\gamma_3\rangle$, however, are in a reversed orientation with the target qudit on the top qudit, $|\gamma_3\rangle$, and control qudit on the bottom qudit, $|\alpha_3\rangle$. This combination of gates causes $|\alpha_3\rangle$, $|\beta_3\rangle$, and $|\gamma_3\rangle$ to become entangled, as can be seen when the state $|\gamma\alpha\beta_3\rangle = |000_3\rangle$ passes through the three-qudit maximal entanglement generator to create the output

$$\begin{aligned}\mathbf{T}_{max}\,|000_3\rangle &= (rev(\mathbf{A}_{(1,2),(1,2)}) \otimes \mathbf{I_3})(\mathbf{I_3} \otimes \mathbf{A}_{(1,2),(1,2)})(\mathbf{I_3} \otimes \mathbf{C_3} \otimes \mathbf{I_3})\,|000_3\rangle \\ &= \frac{1}{\sqrt{3}}\left(|000_3\rangle + |111_3\rangle + |222_3\rangle\right).\end{aligned} \tag{7.6}$$

All of the three-qudit maximal entanglement generator outputs for the circuit in Fig. 7.5 are included in Table 7.7.

The generalized radix-r entanglement circuit that acts on n qudits includes an additional

Table 7.7. Outputs of radix-3 three-qudit maximal entanglement generator circuit in Fig. 7.5

Input	**Output**
$\|000_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|000_3\rangle + \|111_3\rangle + \|222_3\rangle\right)$
$\|001_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|001_3\rangle + \|112_3\rangle + \|220_3\rangle\right)$
$\|002_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|002_3\rangle + \|110_3\rangle + \|221_1\rangle\right)$
$\|010_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|000_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|111_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|222_3\rangle\right)$
$\|011_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|001_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|112_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|220_3\rangle\right)$
$\|012_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|002_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|110_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|221_3\rangle\right)$
$\|020_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|000_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|111_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|222_3\rangle\right)$
$\|021_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|001_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|112_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|220_3\rangle\right)$
$\|022_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|002_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|110_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|221_3\rangle\right)$
$\|100_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|022_3\rangle + \|100_3\rangle + \|211_3\rangle\right)$
$\|101_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|020_3\rangle + \|101_3\rangle + \|212_3\rangle\right)$
$\|102_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|021_3\rangle + \|102_3\rangle + \|210_3\rangle\right)$
$\|110_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1-i\sqrt{3})\|022_3\rangle + \|100_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|211_3\rangle\right)$
$\|111_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1-i\sqrt{3})\|020_3\rangle + \|101_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|212_3\rangle\right)$
$\|112_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1-i\sqrt{3})\|021_3\rangle + \|102_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|210_3\rangle\right)$
$\|120_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1+i\sqrt{3})\|022_3\rangle + \|100_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|211_3\rangle\right)$
$\|121_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1+i\sqrt{3})\|020_3\rangle + \|101_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|212_3\rangle\right)$
$\|122_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1+i\sqrt{3})\|021_3\rangle + \|102_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|210_3\rangle\right)$
$\|200_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|011_3\rangle + \|122_3\rangle + \|200_3\rangle\right)$
$\|201_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|012_3\rangle + \|120_3\rangle + \|201_3\rangle\right)$
$\|202_3\rangle$	$\frac{1}{\sqrt{3}}\left(\|010_3\rangle + \|121_3\rangle + \|202_3\rangle\right)$
$\|210_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1+i\sqrt{3})\|011_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|122_3\rangle + \|200_3\rangle\right)$
$\|211_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1+i\sqrt{3})\|012_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|120_3\rangle + \|201_3\rangle\right)$
$\|212_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1+i\sqrt{3})\|010_3\rangle + \frac{1}{2}(-1-i\sqrt{3})\|121_3\rangle + \|202_3\rangle\right)$
$\|220_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1-i\sqrt{3})\|011_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|122_3\rangle + \|200_3\rangle\right)$
$\|221_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1-i\sqrt{3})\|012_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|120_3\rangle + \|201_3\rangle\right)$
$\|222_3\rangle$	$\frac{1}{\sqrt{3}}\left(\frac{1}{2}(-1-i\sqrt{3})\|010_3\rangle + \frac{1}{2}(-1+i\sqrt{3})\|121_3\rangle + \|202_3\rangle\right)$

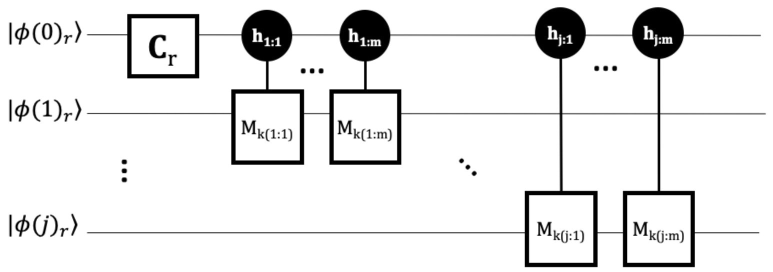


Figure 7.6. Generalized structure of circuit needed for radix-r maximal entanglement among n qudits where $j = n - 1$ and $m = r - 1$.

cascades of $\mathbf{A}_{h,k}$ operators that act on each introduced target. In these cascades, $(\mathbf{A}_{h,k})_i$, the rules for h and k values, as defined earlier, are followed. Each group presented as $(\mathbf{A}_{h,k})_i$ is treated as an independent set where h and k values are appropriate and must not repeat. An illustration of an n qudit maximal entanglement generator structure is included in Fig. 7.6. This type of generator could be considered a higher-radix generalization of the circuitry needed for the preparation of GHZ states whenever the number of involved qudits, n, is greater than 2.

7.3.1. Synthesis of Qudit Entanglement States

It is common knowledge that the Bell state generator can be implemented for radix-2 entangled state preparation. As the dimension of a quantum system grows in size, however, it becomes less intuitive what set of operators are required to prepare a entangled set of basis states where $r > 2$. In addition, it may be necessary to generate a specific entangled state using qudits that are initialized to fixed basis value. For example, many quantum technologies initialize quantum information into a ground state, $|000\ldots00_r\rangle$, that must be used as input to the state preparation generator circuit. For this reason, it is desirable to develop a synthesis technique that derives the gate cascade required to evolve an arbitrary quantum state into a targeted entangled state.

Here, we outline a methodology for determining the circuit structure needed to create a maximally-entangled, radix-r state with a desired set of basis states from a particular input of qudits. For radix-r, up to r qudits may become maximally entangled so that each qubit index in each linearly-combined basis has a unique value ranging from $[0 : r - 1]$. For this reason, the state preparation circuits generated by this methodology can include between two and r qudits. The entangled state generator algorithm, developed in Python as a prototype synthesis tool within Mustang-Q, outputs the necessary cascade of Chrestenson and controlled-modulo-add gates to act as the generator for the desired entangled quantum state distribution. The pseudocode that finds the required generator circuit to transform a fixed input basis state into an entangled state is included in Figure 7.7.

```
Data: Radix ($r$), qudits, input state ($input$), and
      desired entangled state ($basis$)
Result: Gate sequence, $G_i$, and associated controls, $c_i$,
        and targets, $t_i$
Set $G \leftarrow [\ ]$, $c \leftarrow [\ ]$, $t \leftarrow [\ ]$;
$G$.append(Chrestenson($radix$));
$c$.append(NULL);
$t$.append(0);
for $i : 1 \leq i < qudits$ do
    for item in basis do
        $h \leftarrow item[0]$;
        $k \leftarrow item[i] + (r - input[i])\%r$;
        if $k \neq 0$ then
            $G$.append(A($h, k$));
            $c$.append(0);
            $t$.append($i$);
    end
end
```

Figure 7.7. Algorithm: Find entangled state generator circuit.

Finding the entangled state generator circuit requires the input parameters of radix, number of qudits, input state, and desired entangled basis states. The input basis state is in the form of a list while the desired entangled state is a list of sublists that each represent a basis state in the linear combination. For instance, an input of $|\phi\theta_r\rangle = |00_3\rangle$ is passed as $input = [0,0]$ and an entangled state of $|\phi\theta_r\rangle = \frac{1}{\sqrt{3}}(|00_3\rangle + |11_3\rangle + |22_3\rangle)$ would be passed as $basis = [[0,0],[1,1],[2,2]]$. It should be noted that the probability amplitude for each of the entangled basis values is set by the column of the $\mathbf{C}_r$ matrix with an index equal to the control level in the input qudit state. Therefore, each basis will be multiplied by a radix-r root of unity along with the scale factor of $\frac{1}{\sqrt{r}}$. For example, if the output entangled state of

$$\begin{aligned}\mathbf{T}_{max}|00_3\rangle &= \mathbf{A}_{(2,2)}\mathbf{A}_{(1,1)}(\mathbf{C_3} \otimes \mathbf{I_3})|00_3\rangle \\ &= \frac{1}{\sqrt{3}}(|00_3\rangle + |11_3\rangle + |22_3\rangle)\end{aligned}$$

is examined where the superimposed control qudit is $|0_3\rangle$, it is clear that each of the basis states has a probability amplitude equal to an element from the 0^{th} index column of the $\mathbf{C}_3$ transform.

Following the structure pictured in Fig. 7.6, if n total qudits are to be entangled, the first qubit, indexed as $|\phi(0)_r\rangle$ will always be the superimposed qubit that acts as the control for the $\mathbf{A}_{h,k}$ gates. Therefore, the $\mathbf{C}_r$ operator will act on this qubit before any other operators. Next, $n-1$ cascades of $\mathbf{A}_{h,k}$ gates with appropriate h and k values needed to target each of the remaining qudits must be determined. As a note, to maximally entangle n qudits, $(n-1)(r-1)$ total $\mathbf{A}_{h,k}$ are required. These gates will always have qudit $|\phi(0)_r\rangle$ as the control qudit and each of the remaining qudits will act as a target for an $\mathbf{A}_{h,k}$ cascade to become entangled as a group. It is known that each cascade will contain $r-1$ gates.

Preparing the desired entangled state requires the generation of $n-1$ $\mathbf{A}_{h,k}$ cascades. Each qudit that will act as a target, the qudits ranging from $|\phi(1)_r\rangle$ to $|\phi(j)_r\rangle$, will be iterated

Table 7.8. Required generator circuit components for two-qudit maximally entangled state preparation

Input	**Basis States of Maximally Entangled Output**	$\mathbf{C}_r$ **Operator**	$\mathbf{A}_{h,k}$ **Operators** ($\|\phi(0)_r\rangle$ control and $\|\phi(1)_r\rangle$ target)
$\|00_2\rangle$	$\|00_2\rangle, \|11_2\rangle$	$\mathbf{C}_2$	$\mathbf{A}_{1,1}$
$\|11_2\rangle$			$\mathbf{A}_{0,1}$
$\|00_3\rangle$	$\|00_3\rangle, \|11_3\rangle, \|22_3\rangle$	$\mathbf{C}_3$	$\mathbf{A}_{1,1}$,$\mathbf{A}_{2,2}$
$\|11_3\rangle$			$\mathbf{A}_{0,2}$,$\mathbf{A}_{2,1}$
$\|22_3\rangle$			$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$
$\|00_4\rangle$	$\|00_4\rangle, \|11_4\rangle, \|22_4\rangle, \|33_4\rangle$	$\mathbf{C}_4$	$\mathbf{A}_{1,1}$,$\mathbf{A}_{2,2}$, $\mathbf{A}_{3,3}$
$\|11_4\rangle$			$\mathbf{A}_{0,3}$,$\mathbf{A}_{2,1}$,$\mathbf{A}_{3,2}$
$\|33_4\rangle$			$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,$\mathbf{A}_{2,3}$
$\|00_5\rangle$	$\|00_5\rangle, \|11_5\rangle, \|22_5\rangle, \|33_5\rangle, \|44_5\rangle$	$\mathbf{C}_5$	$\mathbf{A}_{1,1}$,$\mathbf{A}_{2,2}$, $\mathbf{A}_{3,3}$, $\mathbf{A}_{4,4}$
$\|22_5\rangle$			$\mathbf{A}_{0,3}$,$\mathbf{A}_{1,4}$,$\mathbf{A}_{3,1}$,$\mathbf{A}_{4,2}$
$\|44_5\rangle$			$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,$\mathbf{A}_{2,3}$, $\mathbf{A}_{3,4}$
$\|00_6\rangle$	$\|00_6\rangle, \|11_6\rangle, \|22_6\rangle, \|33_6\rangle, \|44_6\rangle, \|55_6\rangle$	$\mathbf{C}_6$	$\mathbf{A}_{1,1}$,$\mathbf{A}_{2,2}$, $\mathbf{A}_{3,3}$, $\mathbf{A}_{4,4}$, $\mathbf{A}_{5,5}$,
$\|22_6\rangle$			$\mathbf{A}_{0,4}$,$\mathbf{A}_{1,5}$,$\mathbf{A}_{3,1}$,$\mathbf{A}_{4,2}$,$\mathbf{A}_{5,3}$
$\|55_6\rangle$			$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,$\mathbf{A}_{2,3}$, $\mathbf{A}_{3,4}$, $\mathbf{A}_{4,5}$
$\|00_7\rangle$	$\|00_7\rangle, \|11_7\rangle, \|22_7\rangle, \|33_7\rangle, \|44_7\rangle, \|55_7\rangle, \|66_7\rangle$	$\mathbf{C}_7$	$\mathbf{A}_{1,1}$,$\mathbf{A}_{2,2}$, $\mathbf{A}_{3,3}$, $\mathbf{A}_{4,4}$, $\mathbf{A}_{5,5}$,$\mathbf{A}_{6,6}$
$\|33_7\rangle$			$\mathbf{A}_{0,4}$,$\mathbf{A}_{1,5}$,$\mathbf{A}_{2,6}$,$\mathbf{A}_{4,1}$,$\mathbf{A}_{5,2}$,$\mathbf{A}_{6,3}$
$\|66_7\rangle$			$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,$\mathbf{A}_{2,3}$, $\mathbf{A}_{3,4}$, $\mathbf{A}_{4,5}$,$\mathbf{A}_{5,6}$
$\|00_8\rangle$	$\|00_8\rangle, \|11_8\rangle, \|22_8\rangle, \|33_8\rangle, \|44_8\rangle, \|55_8\rangle, \|66_8\rangle, \|77_8\rangle$	$\mathbf{C}_8$	$\mathbf{A}_{1,1}$,$\mathbf{A}_{2,2}$, $\mathbf{A}_{3,3}$, $\mathbf{A}_{4,4}$, $\mathbf{A}_{5,5}$,$\mathbf{A}_{6,6}$,$\mathbf{A}_{7,7}$
$\|33_8\rangle$			$\mathbf{A}_{0,5}$,$\mathbf{A}_{1,6}$,$\mathbf{A}_{2,7}$,$\mathbf{A}_{4,1}$,$\mathbf{A}_{5,2}$,$\mathbf{A}_{6,3}$,$\mathbf{A}_{7,4}$
$\|77_8\rangle$			$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,$\mathbf{A}_{2,3}$, $\mathbf{A}_{3,4}$, $\mathbf{A}_{4,5}$,$\mathbf{A}_{5,6}$,$\mathbf{A}_{6,7}$
$\|00_9\rangle$	$\|00_9\rangle, \|11_9\rangle, \|22_9\rangle, \|33_9\rangle, \|44_9\rangle, \|55_9\rangle, \|66_9\rangle, \|77_9\rangle, \|88_9\rangle$	$\mathbf{C}_9$	$\mathbf{A}_{1,1}$,$\mathbf{A}_{2,2}$, $\mathbf{A}_{3,3}$, $\mathbf{A}_{4,4}$, $\mathbf{A}_{5,5}$,$\mathbf{A}_{6,6}$,$\mathbf{A}_{7,7}$,$\mathbf{A}_{8,8}$
$\|44_9\rangle$			$\mathbf{A}_{0,5}$,$\mathbf{A}_{1,6}$,$\mathbf{A}_{2,7}$,$\mathbf{A}_{3,8}$,$\mathbf{A}_{5,1}$,$\mathbf{A}_{6,2}$,$\mathbf{A}_{7,3}$,$\mathbf{A}_{8,4}$
$\|88_9\rangle$			$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,$\mathbf{A}_{2,3}$, $\mathbf{A}_{3,4}$, $\mathbf{A}_{4,5}$,$\mathbf{A}_{5,6}$,$\mathbf{A}_{6,7}$,$\mathbf{A}_{7,8}$

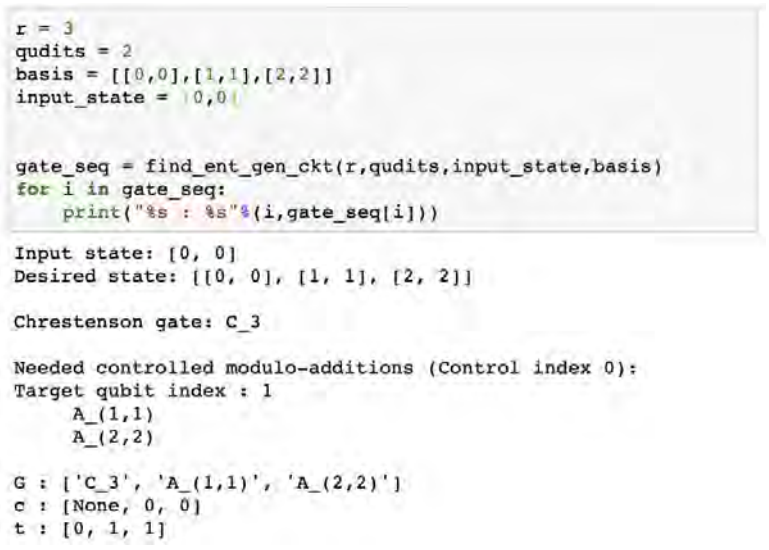
Table 7.9. Required generator circuit components for multi-qudit maximally entangled state preparation

Input	**Basis States of Maximally Entangled Output**	**$\mathbf{C}_r$ Operator**	**Target Qudit** ($\vert\phi(0)_r\rangle$ control)	**$\mathbf{A}_{h,k}$ Operators**
$\vert 012_4\rangle$	$\vert 000_4\rangle, \vert 111_4\rangle, \vert 222_4\rangle, \vert 333_4\rangle$	$\mathbf{C}_4$	$\vert\phi(1)_4\rangle$	$\mathbf{A}_{0,3}$,$\mathbf{A}_{2,1}$,$\mathbf{A}_{3,2}$
			$\vert\phi(2)_4\rangle$	$\mathbf{A}_{0,2}$,$\mathbf{A}_{1,3}$,,$\mathbf{A}_{3,1}$
$\vert 0123_4\rangle$	$\vert 0000_4\rangle, \vert 1111_4\rangle, \vert 2222_4\rangle, \vert 3333_4\rangle$	$\mathbf{C}_4$	$\vert\phi(1)_4\rangle$	$\mathbf{A}_{0,3}$,$\mathbf{A}_{2,1}$,$\mathbf{A}_{3,2}$
			$\vert\phi(2)_4\rangle$	$\mathbf{A}_{0,2}$,$\mathbf{A}_{1,3}$,,$\mathbf{A}_{3,1}$
			$\vert\phi(3)_4\rangle$	$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,,$\mathbf{A}_{2,3}$
$\vert 012_5\rangle$	$\vert 000_5\rangle, \vert 111_5\rangle, \vert 222_5\rangle, \vert 333_5\rangle, \vert 444_5\rangle$	$\mathbf{C}_5$	$\vert\phi(1)_5\rangle$	$\mathbf{A}_{0,4}$,$\mathbf{A}_{2,1}$,$\mathbf{A}_{3,2}$,$\mathbf{A}_{4,3}$
			$\vert\phi(2)_5\rangle$	$\mathbf{A}_{0,3}$,$\mathbf{A}_{1,4}$,$\mathbf{A}_{3,1}$,$\mathbf{A}_{4,2}$
$\vert 0123_5\rangle$	$\vert 0000_5\rangle, \vert 1111_5\rangle, \vert 2222_5\rangle, \vert 3333_5\rangle, \vert 4444_5\rangle$	$\mathbf{C}_5$	$\vert\phi(1)_5\rangle$	$\mathbf{A}_{0,4}$,$\mathbf{A}_{2,1}$,$\mathbf{A}_{3,2}$,$\mathbf{A}_{4,3}$
			$\vert\phi(2)_5\rangle$	$\mathbf{A}_{0,3}$,$\mathbf{A}_{1,4}$,$\mathbf{A}_{3,1}$,$\mathbf{A}_{4,2}$
			$\vert\phi(3)_5\rangle$	$\mathbf{A}_{0,2}$,$\mathbf{A}_{1,3}$,$\mathbf{A}_{2,4}$,$\mathbf{A}_{4,1}$
$\vert 01234_5\rangle$	$\vert 00000_5\rangle, \vert 11111_5\rangle, \vert 22222_5\rangle, \vert 33333_5\rangle, \vert 44444_5\rangle$	$\mathbf{C}_5$	$\vert\phi(1)_5\rangle$	$\mathbf{A}_{0,4}$,$\mathbf{A}_{2,1}$,$\mathbf{A}_{3,2}$,$\mathbf{A}_{4,3}$
			$\vert\phi(2)_5\rangle$	$\mathbf{A}_{0,3}$,$\mathbf{A}_{1,4}$,$\mathbf{A}_{3,1}$,$\mathbf{A}_{4,2}$
			$\vert\phi(3)_5\rangle$	$\mathbf{A}_{0,2}$,$\mathbf{A}_{1,3}$,$\mathbf{A}_{2,4}$,$\mathbf{A}_{4,1}$
			$\vert\phi(4)_5\rangle$	$\mathbf{A}_{0,1}$,$\mathbf{A}_{1,2}$,$\mathbf{A}_{2,3}$,$\mathbf{A}_{3,4}$

sequentially where $j = n - 1$. As seen in the algorithm pseudocode, value of h is derived by which basis in the output state becomes entangled. The value of k determines the extent of entanglement because it controls what modulo-add-k operation is implemented.

A demonstration of generator circuit synthesis is included in Fig. 7.8. In this example, the structure of the radix-3 maximal entanglement generator from Fig. 7.2 is determined using the parameters of the circuit input, $|00_3\rangle$, and the desired entangled quantum state basis values of $|00_3\rangle, |11_3\rangle, |22_3\rangle$. As mentioned previously, all of these basis states will have a probability magnitude of $\frac{1}{\sqrt{3}}$ because that is the value of each of the elements of the 0^{th} column of the $\mathbf{C}_3$ transform.

When generating a maximally-entangled qudit state, it may be necessary to begin with a set of qudits that are initialized to a particular basis before transformation procedures. Depending on the original quantum state, different cascades of $\mathbf{A}_{h,k}$ operations must be implemented to achieve a targeted entangled output. To address this scenario, additional



```
r = 3
qudits = 2
basis = [[0,0],[1,1],[2,2]]
input_state = [0,0]


gate_seq = find_ent_gen_ckt(r,qudits,input_state,basis)
for i in gate_seq:
    print("%s : %s"%(i,gate_seq[i]))
```

```
Input state: [0, 0]
Desired state: [[0, 0], [1, 1], [2, 2]]

Chrestenson gate: C_3

Needed controlled modulo-additions (Control index 0):
Target qubit index : 1
     A_(1,1)
     A_(2,2)

G : ['C_3', 'A_(1,1)', 'A_(2,2)']
c : [None, 0, 0]
t : [0, 1, 1]
```

Figure 7.8. Sample output of generator circuit synthesis to prepare $\frac{1}{\sqrt{3}}\left(|00_3\rangle + |11_3\rangle + |22_3\rangle\right)$ from ground state $|00_3\rangle$.

examples of synthesized circuit cascades for systems ranging from $r = 2$ to $r = 9$ can be found in Table 7.8 for the two-qudit case. This table includes sample input circuit values, found in column one, that target a set of maximally entangled basis states, found in column two. The required $\mathbf{C}_r$ operator is in column three and the synthesized $\mathbf{A}_{h,k}$ cascade needed to generate the desired linear combination of basis states is found in column four. Other input qudit and $\mathbf{A}_{h,k}$ operator combinations also generate the Table 7.8 entangled outputs, but Table 7.8 is not all inclusive in the interest of space. As a note, since only two qudits become entangled only $n-1 = 2-1 = 1$ cascade is required. More examples of synthesis are included in Table 7.9. In these results, entangled states including more than two qudits are analyzed for radix-4 and radix-5 systems. These circuit descriptions would be constructed using the form of Fig. 7.6.

Chapter 8

Conclusion

8.1. Summary

QIP has the potential to revolutionize modern computation. With a QC, many problems that are intractable with a classical computer can be solved in a tractable fashion. This improvement, however, is because the rules that govern quantum computation are vastly different than those associated with classical models as the Turing model does not apply to a QC. These differences in the computing paradigms, especially in information representation and transformation, require different approaches for logic synthesis and state initialization. This dissertation has the overarching theme of quantum logic synthesis and compilation, and the work is developed in two parts: technology-dependent mapping for available radix-2 QCs and the generation of higher-dimensional circuitry for entangled state preparation.

Various types of quantum technology has emerged in recent years. These NISQ-era devices, based on radix-2 number systems, are characterized by differing gate libraries and topological constraints. Operational characteristics such as native gate sets and qubit connectivity must be taken into consideration if a general quantum circuit is to be transformed into a technology-compatible form. To accomodate to this need, methodologies for formally-verified technology-dependent quantum logic synthesis were presented in this work. These algorithms are intended for design automation and were prototyped in order to map and optimize quantum circuits to various types of QCs, including transmon and photonic devices. The described synthesis tool shows great promise for simplifying the quantum algorithm and design process whenever a real QC architecture is targeted for use. In addition to the described methods targeting currently existing radix-2 devices, methods for sythesizing higher-dimensioned quantum circuitry were also proposed.

Higher-dimensioned quantum systems are feasible and they may provide opportunities for improvement over radix-2 QIP implementations. Along with presenting radix-2 quantum circuit transformation and optimization, a synthesis algorithm that generates a description of the circuitry required for higher-radix maximal entanglement was also described. These higher-dimension synthesis procedures depend on superposition and controlled change-of-base operations in order to produce entangled states, and these gates were generalized from the components of the well-known Bell state generator from binary qubit-based QIS.

8.2. Future Work

In this work, methods for quantum logic synthesis and compilation for the radix-2 and higher-dimensional cases were presented. There exist plans to expand the contributions within this dissertation in the future. For example, as new quantum technologies emerge, Mustang-Q will be expanded to include these platforms as back-end libraries. Each of these libraries will have associated cost functions and sets of native gate operations, so the research and development of new logic transformations and optimizations will be required.

Although the methods in this work include equivalence checking to verify results of synthesis, formal verification does not detect errored logic within the original quantum programs. Thus, a logical continuation of this work includes the development and integration algorithms for quantum debugging tools. Quantum debuggers would assist in pinpointing errors in quantum algorithms that are commonly introduced during the development process. Just as with compilation, methods for quantum program debugging will differ greatly from classical program debugging. For example, because of the inability to clone or observe qubit state without the collapes of superposition, classical debugging techniques cannot be immediately applied to QCs. Past discussions of quantum debuggers include (Chong et al., 2017; Huang and Martonosi, 2019).

As for the higher-radix entanglement generators, within the scope of quantum entanglement of qudit groups, there are many additional topics that can be investigated to build on the work described here. For instance, although maximal entanglement generator structures capable of outputting Bell- and GHZ-inspired states were considered and synthesis procedures were detailed, these methods can be generalized in the future so that they can output higher-radix quantum states inspired by W states where the number of basis states in the entanglement is greater than the system's radix, r.

Appendix A

The Radix-4 Chrestenson Gate

Using the fourth roots of unity, $w_0 = \exp[(i2\pi/4) * 0] = 1$, $w_1 = \exp[(i2\pi/4) * 1] = i$, $w_2 = \exp[(i2\pi/4) * 2] = -1$, and $w_3 = \exp[(i2\pi/4) * 3] = -i$, in Eqn. 6.7, the radix-4 Chrestenson gate transfer matrix becomes

$$\mathbf{C_4} = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} . \tag{A.1}$$

The radix-4 Chrestenson gate ($\mathbf{C_4}$), allows a radix-4 qudit originally in a basis to evolve into a quantum state of equal superposition. The following example shows how the radix-4 qudit $|a_4\rangle = |0_4\rangle$ evolves to $|b_4\rangle = \frac{1}{2}|0_4\rangle + \frac{1}{2}|1_4\rangle + \frac{1}{2}|2_4\rangle + \frac{1}{2}|3_4\rangle$, taking the value of the first column of the radix-4 Chrestenson matrix, after passing through the $\mathbf{C_4}$ transform

$$\mathbf{C_4}|a_4\rangle = |b_4\rangle ,$$

$$\mathbf{C_4}\left|0_4\right\rangle = \frac{1}{\sqrt{4}}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix},$$

$$\mathbf{C_4}\left|0_4\right\rangle = \frac{1}{2}[\left|0_4\right\rangle + \left|1_4\right\rangle + \left|2_4\right\rangle + \left|3_4\right\rangle].$$

If $\left|a_4\right\rangle = \left|3_4\right\rangle$, the radix-4 qudit would evolve to $\left|b_4\right\rangle = \frac{1}{2}\left|0_4\right\rangle - \frac{1}{2}i\left|1_4\right\rangle - \frac{1}{2}\left|2_4\right\rangle + \frac{1}{2}i\left|3_4\right\rangle$, taking the value of the last column of the $\mathbf{C_4}$ transformation matrix.

$$\mathbf{C_4}\left|3\right\rangle = \frac{1}{\sqrt{4}}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1\\-i\\-1\\i\end{bmatrix},$$

$$\mathbf{C_4}\left|3_4\right\rangle = \frac{1}{2}[\left|0_4\right\rangle - i\left|1_4\right\rangle - \left|2_4\right\rangle + i\left|3_4\right\rangle].$$

A.0.1. Quantum Optics

Optical quantum implementations are among the more successful physical realizations of quantum states. In these systems, orthogonal basis states can be encoded into photon OAM states, polarization, or location, and the state can easily evolve by passing through linear optical elements. The photon resists coupling to other objects in its environment, allowing it to maintain its quantum state and not decohere for long periods of time (Kok et al., 2007). Additionally, the ability to maintain coherence enables the photon to travel great distances at room temperature, making it a good candidate for long-haul quantum information transmission.

Although photons offer the benefit of state stability in QIP applications, their failure to interact with their surroundings prevents them from coupling with each other. Photon-to-photon interaction is difficult, limiting the development of reliable controlled multi-qubit or multi-qudit gate implementations. Without operations such as the radix-2 **CNOT** gate or the controlled-phase gate, a functional QC cannot exist.

It was once thought that photonic quantum computation was unachievable without non-linear optical elements, but the presentation of the KLM protocol in reference (Knill et al., 2001) improved the outlook for quantum optics. In that work, a methodology for implementing photonic multi-qubit operations using linear optics was introduced. These multi-qubit photonic gates, however, are unfortunately limited by probabilistic operation. Currently, the two-qubit **CNOT** operation can only work 1/4 of the time when implemented with linear optical elements in the optimal case (Eisert, 2005).

The subject of this section is a photonic radix-4 Chrestenson gate. Since this quantum operator is formed from linear optical elements and transforms a single qudit at a time, the gate is theoretically deterministic in nature.

A.1. The Four-port Coupler

The four-port coupler is an optical component introduced and described in reference (MacFarlane et al., 2004). This device is composed of four inputs and four outputs where the input and output are referred to by their orientation on the component of either W, N, E, or S. When a single beam is sent into one of the coupler inputs, the component routes a fraction of the original signal to each of the four outputs. This beam division is caused by the transmission and reflection of signals within the coupler. Each fraction of the input beam seen at an output corresponds to one of the following components of the original signal: a reflected component ρ, a transmitted component τ, a right-directed component α, and a

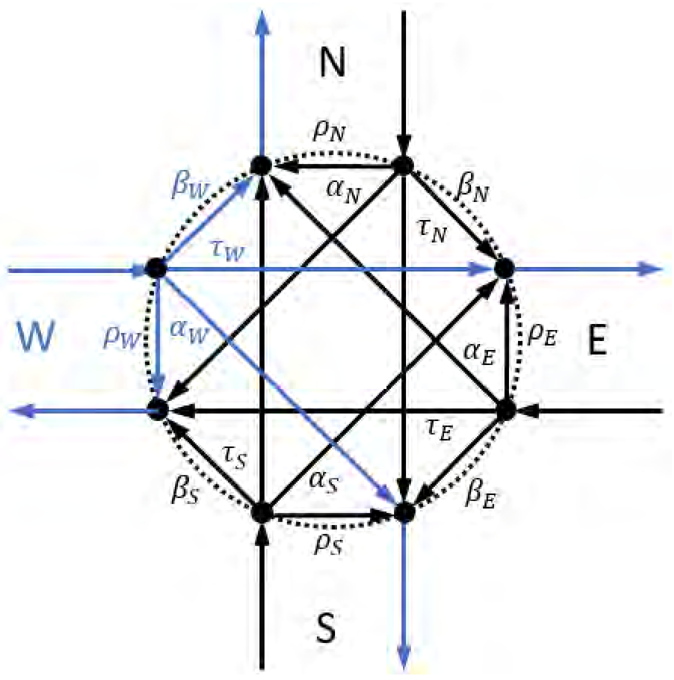


Figure A.1. Signal flow for four-port coupler with input at W.

left-directed component β. An illustration of signal flow of the four-port coupler can be seen in Fig. A.1. This image is recreated from a figure included in reference (MacFarlane et al., 2004).

Fig. A.1 demonstrates in blue a signal entering the four-port coupler from the W port and exiting the component from the W, N, E, and S ports. The output signals are generated by ρ_W, β_W, τ_W, and α_W, respectively. Whenever a single input enters the component, all four coupling coefficients are generated to produce four outputs. The coupling coefficients produced with a particular port input can be derived with the coupling coefficient matrix,

$$\begin{bmatrix} \rho_W & \alpha_N & \tau_E & \beta_S \\ \beta_W & \rho_N & \alpha_E & \tau_S \\ \tau_W & \beta_N & \rho_E & \alpha_S \\ \alpha_W & \tau_N & \beta_E & \rho_S \end{bmatrix}. \tag{A.2}$$

To produce the outputs, an input vector taking the form of $[WNES]^{\mathrm{T}}$ is multiplied by the matrix in Eqn. A.2 to create a column vector of coupling coefficients. The produced output column vector also takes the form of $[WNES]^{\mathrm{T}}$. The composition of the output vector in terms of coupling coefficients indicates what portion of the input signal contributes to an output from a port.

The four-port coupler does not consume nor dissipate any of the energy that is input into the component. Therefore, to conserve energy, all of the energy entering the element must be equal to the energy leaving the element. This concept leads to the creation of equations that act as conditions that must hold true for energy conservation. The 10 energy conservation equations of Eqns. A.3-A.12, first derived in reference (MacFarlane et al., 2004), use the coupling coefficients found in the matrix of Eqn. A.2. These equations are:

$$\rho_W^* \rho_W + \beta_W^* \beta_W + \tau_W^* \tau_W + \alpha_W^* \alpha_W = 1, \tag{A.3}$$

$$\rho_N^* \rho_N + \beta_N^* \beta_N + \tau_N^* \tau_N + \alpha_N^* \alpha_N = 1, \tag{A.4}$$

$$\rho_E^* \rho_E + \beta_E^* \beta_E + \tau_E^* \tau_E + \alpha_E^* \alpha_E = 1, \tag{A.5}$$

$$\rho_S^* \rho_S + \beta_S^* \beta_S + \tau_S^* \tau_S + \alpha_S^* \alpha_S = 1, \tag{A.6}$$

$$\rho_W^* \tau_E + \beta_W^* \alpha_E + \tau_W^* \rho_E + \alpha_W^* \beta_E = 0, \tag{A.7}$$

$$\alpha_N^* \beta_S + \rho_N^* \tau_S + \beta_N^* \alpha_S + \tau_N^* \rho_S = 0, \tag{A.8}$$

$$\rho_W^* \alpha_N + \beta_W^* \rho_N + \tau_W^* \beta_N + \alpha_W^* \tau_N = 0, \tag{A.9}$$

$$\alpha_N^* \tau_E + \rho_N^* \alpha_E + \beta_N^* \rho_E + \tau_N^* \beta_E = 0, \tag{A.10}$$

$$\tau_E^* \beta_S + \alpha_E^* \tau_S + \rho_E^* \alpha_S + \beta_E^* \rho_S = 0, \tag{A.11}$$

and

$$\rho_W^* \beta_S + \beta_W^* \tau_S + \tau_W^* \alpha_S + \alpha_W^* \rho_S = 0. \tag{A.12}$$

The first four conditions seen in Eqns. A.3-A.6 exist since the inner product of each produced field vector from a single input, W, N, E, and S, with itself must sum to one for energy conservation. The last six conditions seen in Eqns. A.7-A.12 exist due to energy conservation that occurs whenever two inputs are present in the component. Since the coefficient vectors are orthogonal, the inner product between the two produced coupling coefficient vectors corresponding to inputs at two different ports must equal zero. There are only 6 constraints produced from sending two inputs to the four-port coupler because the input combinations are commutative (i.e. $AB = BA$). The cases of three inputs and four inputs into the four-port coupler do not create additional constraints, so they are omitted (MacFarlane et al., 2004).

A.2. Physical Realizations of the Four-port Coupler

A macroscopic realization of a four-port coupler is shown in Fig. A.2. Whereas a popular implementation of a radix-2 Hadamard gate is an optical beam splitter, polarizing or not, the macroscopic four-port coupler is a unitary extension of a two-prism beam splitting cube. Here, the macroscopic four-port coupler is comprised of four right angle prisms, coated with an appropriate thin film, cemented together with care given to the precise mating of the four prism corners. This component has been used to demonstrate novel, four leg Michelson interferometers designed in reference (Sultana et al., 2009).

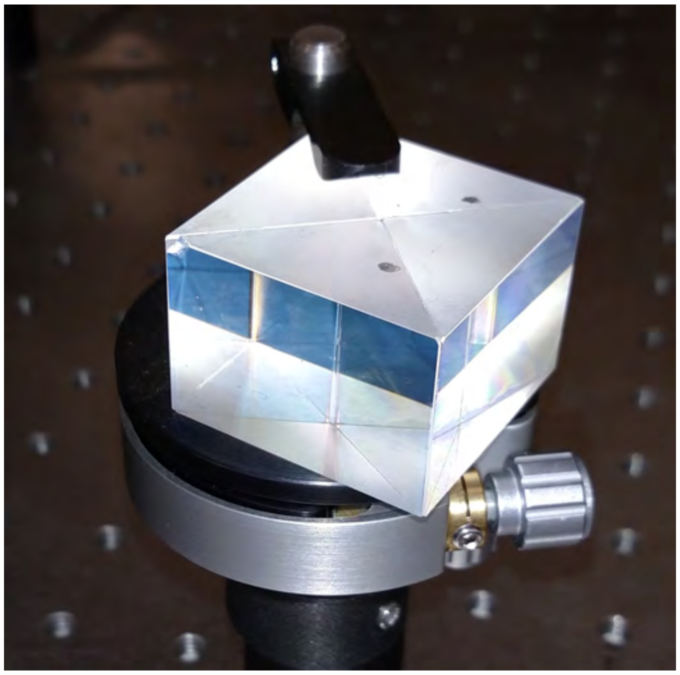


Figure A.2. Macroscopic realization of a four-port coupler.

Integrated photonic four-port couplers were previously demonstrated for applications in optical signal processing as part of a two-dimensional array of waveguides in a multi-quantum well (MQW) GaInAsP indium phosphide (InP) architecture (MacFarlane et al., 2011a,b). Fig. A.3 shows an electron micrograph of a coupler fabricated at the intersection of two ridge waveguides.

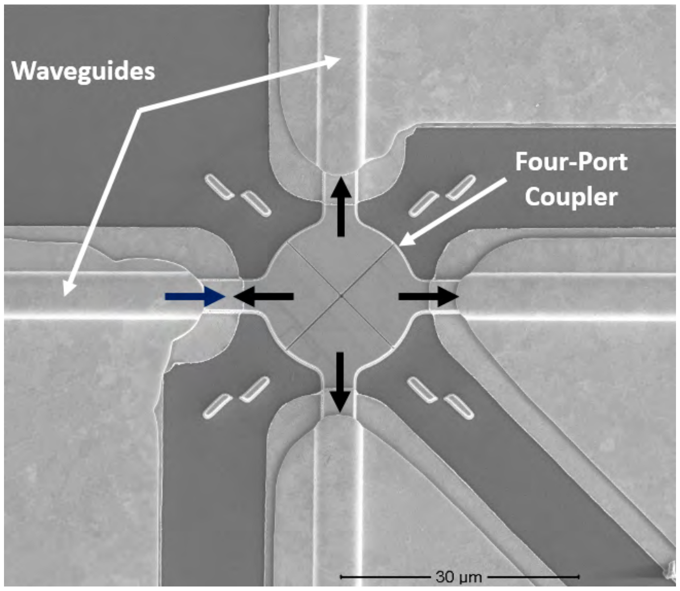


Figure A.3. Cross sectional scanning electron microscope image of a four-port coupler in MQW-InP.

The optical behavior of the four-port coupler depends on frustrated total internal reflection (Gale, 1972). The evanescent field of light incident on the coupler is transfered across the width of the component that may be an air gap or a thin slice of dielectric. Provided the barrier width is small enough, a part of the exponentially decaying optical power of the incident light is transmitted across while the remaining optical power is reflected. Thus, a fraction of light incident on a four-port coupler may be transmitted to the ongoing waveguide, reflected to both perpendicular waveguides, or reflected back into the originating waveguide. The fractions of light in outbound waveguides are determined by the refractive indices of the waveguide and coupler materials in addition to the width of the coupler.

A.2.1. Fabrication

Fabrication of the coupler was performed in several steps using nanoelectronic processing techniques. First, coupler regions of 180 nm widths and 7 μm lengths were defined by patterning a thin metallic chromium mask layer atop the waveguides by focused ion beam (FIB) lithography. Precision alignment and orientation of the coupler to the waveguides during FIB processing was achieved with alignment markers fabricated beforehand with the waveguides using conventional microelectronic processing steps. High aspect ratio trenches were then etched using a hydrogen bromine (HBr) based (Sultana et al., 2009) inductively coupled plasma (ICP) to a depth of 3.9 μm. This depth allows the coupler to fully cover optical modes confined to the quantum well region of the waveguides.

The optimal air gap width for 25% power on all output waveguides of about 90 nm was slightly smaller than the processing capability of the ICP dry etch tool for the required high-aspect ratio etch. Consequently, to meet this requirement for a wavelength of 1550 nm, alumina (Al_2O_3), with a refractive index of $n = 1.71$, was back-filled into the trench using atomic layer deposition (ALD). The resulting alumina-filled trench is shown in the composite cross-sectional transmission electron micrographs in Fig. A.4.

A.2.2. Characterization

A 1550 nm laser was coupled into the waveguides using a tapered lens fiber at one input port. The near-field modes of light were coupled out of the device and into another tapered lens fiber for optical power measurement to characterize the coupling efficiency of the four-port coupler. The measured average power coefficients were $\alpha = 0.156$, $\beta = 0.140$, $\rho = 0.302$, and $\tau = 0.220$ for a measured total average coupling efficiency of 82% for the four-port coupler (MacFarlane et al., 2011b).

A.3. Implementing Qudit Quantum Operations with the Coupler

It is known that the Hadamard gate meant for use with a quantum qubit can be constructed from a beam splitter (Kok et al., 2007). The radix-4 Chrestenson operation, an operation that acts on a quantum encoding using four basis states, transforms a radix-4 qudit, and the four-port coupler is a physical realization of this gate. In the realization of the radix-4 Chrestenson gate, the ports of the four-port coupler must be encoded in order to represent the four qudit basis states. Here, the following encoding has been chosen for the location-based scheme: port W is the $|0_4\rangle$ rail, port N is the $|1_4\rangle$ rail, port E is the $|2_4\rangle$ rail, and port S is the $|3_4\rangle$ rail.

The four-port coupler follows 10 energy conservation equations, Eqns. A.3-A.12, that are algebraically nonlinear. If the radix-4 Chrestenson matrix values are substituted for the values of the coupling coefficients in Eqn. A.2, the energy conservation constraints are satisfied and the following matrix is generated:

$$\begin{bmatrix} \rho_W = \frac{1}{2} & \alpha_N = \frac{1}{2} & \tau_E = \frac{1}{2} & \beta_S = \frac{1}{2} \\ \beta_W = \frac{1}{2} & \rho_N = \frac{1}{2}i & \alpha_E = -\frac{1}{2} & \tau_S = -\frac{1}{2}i \\ \tau_W = \frac{1}{2} & \beta_N = -\frac{1}{2} & \rho_E = \frac{1}{2} & \alpha_S = -\frac{1}{2} \\ \alpha_W = \frac{1}{2} & \tau_N = -\frac{1}{2}i & \beta_E = -\frac{1}{2} & \rho_S = \frac{1}{2}i \end{bmatrix}.$$

When a single photon, representing a qudit, is applied to one of the inputs the four-port coupler, either W, N, E, or S, energy is conserved and the radix-4 Chrestenson transform is achieved. The photon leaves the gate with equal superposition of all basis states. In other words, the photon has a 25% probability of being located in any of the output ports W, N, E, or S representing the basis states $|0_4\rangle$, $|1_4\rangle$, $|2_4\rangle$, or $|3_4\rangle$, respectively:

$$\rho_W^*\rho_W + \beta_W^*\beta_W + \tau_W^*\tau_W + \alpha_W^*\alpha_W = 1,$$
$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) = 1,$$
$$\rho_N^*\rho_N + \beta_N^*\beta_N + \tau_N^*\tau_N + \alpha_N^*\alpha_N = 1,$$
$$\left(-\frac{1}{2}i\right)\left(\frac{1}{2}i\right) + \left(-\frac{1}{2}\right)\left(-\frac{1}{2}\right) + \left(\frac{1}{2}i\right)\left(-\frac{1}{2}i\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) = 1,$$
$$\rho_E^*\rho_E + \beta_E^*\beta_E + \tau_E^*\tau_E + \alpha_E^*\alpha_E = 1,$$
$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(-\frac{1}{2}\right)\left(-\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(-\frac{1}{2}\right)\left(-\frac{1}{2}\right) = 1,$$

$$\rho_S^*\rho_S + \beta_S^*\beta_S + \tau_S^*\tau_S + \alpha_S^*\alpha_S = 1,$$
$$\left(-\frac{1}{2}i\right)\left(\frac{1}{2}i\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}i\right)\left(-\frac{1}{2}i\right) + \left(-\frac{1}{2}\right)\left(-\frac{1}{2}\right) = 1.$$

If two signals are input into the four-port coupler Chrestenson gate, the conservation of energy causes the inner product of the two produced vectors of coupling coefficients to be zero:

$$\rho_W^*\tau_E + \beta_W^*\alpha_E + \tau_W^*\rho_E + \alpha_W^*\beta_E = 0,$$
$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(-\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(-\frac{1}{2}\right) = 0,$$
$$\alpha_N^*\beta_S + \rho_N^*\tau_S + \beta_N^*\alpha_S + \tau_N^*\rho_S = 0,$$
$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(-\frac{1}{2}i\right)\left(-\frac{1}{2}i\right) + \left(-\frac{1}{2}\right)\left(-\frac{1}{2}\right) + \left(\frac{1}{2}i\right)\left(\frac{1}{2}i\right) = 0,$$
$$\rho_W^*\alpha_N + \beta_W^*\rho_N + \tau_W^*\beta_N + \alpha_W^*\tau_N = 0,$$
$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}i\right) + \left(\frac{1}{2}\right)\left(-\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(-\frac{1}{2}i\right),$$
$$\alpha_N^*\tau_E + \rho_N^*\alpha_E + \beta_N^*\rho_E + \tau_N^*\beta_E = 0,$$

$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)+\left(-\frac{1}{2}i\right)\left(-\frac{1}{2}\right)+\left(-\frac{1}{2}\right)\left(\frac{1}{2}\right)+\left(\frac{1}{2}i\right)\left(-\frac{1}{2}\right)=0,$$

$$\tau_E^*\beta_S+\alpha_E^*\tau_S+\rho_E^*\alpha_S+\beta_E^*\rho_S=0,$$

$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)+\left(-\frac{1}{2}\right)\left(-\frac{1}{2}i\right)+\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)+\left(-\frac{1}{2}\right)\left(\frac{1}{2}i\right)=0,$$

$$\rho_W^*\beta_S+\beta_W^*\tau_S+\tau_W^*\alpha_S+\alpha_W^*\rho_S=0,$$

$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)+\left(\frac{1}{2}\right)\left(-\frac{1}{2}i\right)+\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)+\left(\frac{1}{2}\right)\left(\frac{1}{2}i\right)=0.$$

Since these equations are satisfied with the elements of the derived radix-4 Chrestenson transform matrix, the four-port coupler proves to act as an effective radix-4 Chrestenson gate.

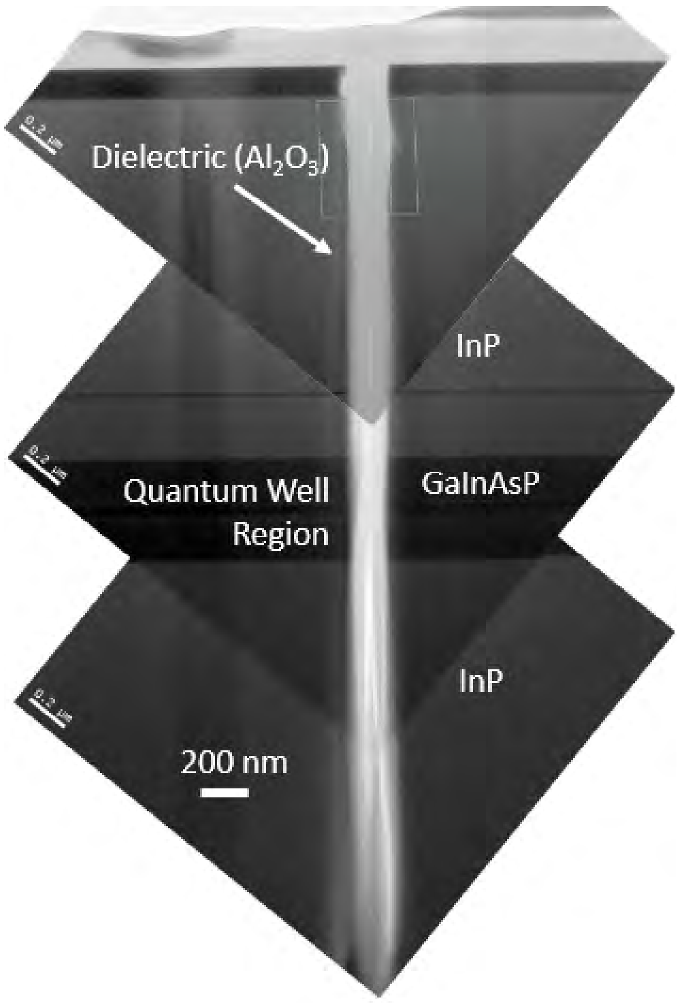


Figure A.4. Cross sectional transmission electron micrograph of a four-port coupler backfilled with alumina using atomic layer deposition.

REFERENCES

Revlib. URL http://www.revlib.org/.

Reversible logic synthesis and quantum computing benchmarks. http://quantumlib.stationq.com/, 2017.

MP Almeida, SP Walborn, and PH Ribeiro. Four-dimensional quantum key distribution using position-momentum and polarization correlations. *arXiv preprint quant-ph/0510087*, 2005.

Matthew Amy. Feynman, 2019. URL https://github.com/meamy/feynman.

Matthew Amy, Dmitri Maslov, and Michele Mosca. Polynomial-time t-depth optimization of clifford+ t circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(10):1476–1489, 2014.

Sam Bader. The transmon qubit. 2013.

Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical review A*, 52(5):3457, 1995.

Charles H Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973.

Charles H Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Physical review letters*, 70(13):1895, 1993.

H Bennett Ch and G Brassard. Quantum cryptography: public key distribution and coin tossing int. In *Conf. on Computers, Systems and Signal Processing (Bangalore, India, Dec. 1984)*, pages 175–9, 1984.

Reinhold A Bertlmann and Philipp Krammer. Bloch vectors for qudits. *Journal of Physics A: Mathematical and Theoretical*, 41(23):235303, 2008.

Felix Bloch. Nuclear induction. *Physical review*, 70(7-8):460, 1946.

Max Born. Quantum mechanics of collision processes. *Zeit fur Phys*, 38:803, 1926.

Adi Botea, Akihiro Kishimoto, and Radu Marinescu. On the complexity of quantum circuit compilation. pages 138–142, 2018. URL https://aaai.org/ocs/index.php/SOCS/SOCS18/paper/view/17959.

Charles Q. Choi. Qudits: The real future of quantum computing? *IEEE Spectrum Magazine*, 2017.

Frederic T Chong, Diana Franklin, and Margaret Martonosi. Programming languages and compiler design for realistic quantum hardware. *Nature*, 549(7671):180, 2017.

Jerry M Chow, Jay M Gambetta, Easwar Magesan, David W Abraham, Andrew W Cross, BR Johnson, Nicholas A Masluk, Colm A Ryan, John A Smolin, Srikanth J Srinivasan, et al. Implementing a strand of a scalable fault-tolerant quantum computing fabric. *Nature Communications*, 5, 2014a.

Jerry M Chow, Jay M Gambetta, Easwar Magesan, David W Abraham, Andrew W Cross, BR Johnson, Nicholas A Masluk, Colm A Ryan, John A Smolin, Srikanth J Srinivasan, et al. Implementing a strand of a scalable fault-tolerant quantum computing fabric. *Nature communications*, 5:4015, 2014b.

HE Chrestenson et al. A class of generalized walsh functions. *Pacific Journal of Mathematics*, 5(1):17–31, 1955.

Antonio D Córcoles, Easwar Magesan, Srikanth J Srinivasan, Andrew W Cross, Matthias Steffen, Jay M Gambetta, and Jerry M Chow. Demonstration of a quantum error detection code using a square lattice of four superconducting qubits. *Nature communications*, 6, 2015.

D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. In *Proceedings of the Royal Society of London A 400*, pages 97–117, 1985.

M. H. Devoret and R. J. Schoelkopf. Superconducting circuits for quantum information: An outlook. *Science*, 339:1169–1174, 2013.

Nicolas Didier, Eyob A Sete, Marcus P da Silva, and Chad Rigetti. Analytical modeling of parametrically modulated transmon qubits. *Physical Review A*, 97(2):022330, 2018.

Paul Adrien Maurice Dirac. *The principles of quantum mechanics*. Oxford university press, 1958.

David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 2010.

Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777, 1935.

Jens Eisert. Optimizing linear optics quantum gates. *Physical review letters*, 95(4):040502, 2005.

Artur K Ekert. Quantum cryptography based on bell's theorem. *Physical review letters*, 67 (6):661, 1991.

M Enríquez, I Wintrowicz, and Karol Życzkowski. Maximally entangled multipartite states: a brief survey. In *Journal of Physics: Conference Series*, volume 698, page 012003. IOP Publishing, 2016.

Daphna G Enzer, Phillip G Hadley, Richard J Hughes, Charles G Peterson, and Paul G Kwiat. Entangled-photon six-state quantum cryptography. *New Journal of Physics*, 4(1): 45, 2002.

K Fazel, MA Thornton, and JE Rice. Esop-based toffoli gate cascade generation. In *Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on*, pages 206–209. IEEE, 2007.

Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

Erik Gabrielson and Mitchell A Thornton. Minimizing ancilla and garbage qubits in reversible function specifications. Technical report, Southern Methodist University, Darwin Deason Institute for Cyber Security, 2018a.

Erik Gabrielson and Mitchell A Thornton. Minimizing ancilla and garbage qubits in reversible function specifications (to appear, poster). In *Southwest Quantum Information and Technology 20th Annual SQuInT Workshop (SQuInT)*, 2018b.

Douglas S Gale. Frustrated total internal reflection. *American Journal of Physics*, 40(7): 1038–1039, 1972.

Juan Carlos García-Escartín and Pedro Chamorro-Posada. Quantum multiplexing with the orbital angular momentum of light. *Physical Review A*, 78(6):062320, 2008.

Graham Gibson, Johannes Courtial, Miles J Padgett, Mikhail Vasnetsov, Valeriy Pas'ko, Stephen M Barnett, and Sonja Franke-Arnold. Free-space information transfer using light beams carrying orbital angular momentum. *Optics express*, 12(22):5448–5456, 2004.

Pranav Gokhale, Jonathan M Baker, Casey Duckering, Natalie C Brown, Kenneth R Brown, and Frederic T Chong. Asymptotic improvements to quantum circuits via qutrits. In *ACM/IEEE International Symposium on Computer Architecture (ISCA)*. IEEE, 2019.

Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72. Springer, 1989.

David J. Griffiths. *Introduction to Quantum Mechanics*. Prentice-Hall, Inc., 1995.

Brian Hayes. Computing science: Third base. *American scientist*, 89(6):490–494, 2001.

Yipeng Huang and Margaret Martonosi. Statistical assertions for validating patterns and finding bugs in quantum programs. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 541–553. ACM, 2019.

IBM Q team. IBM Q 5 Yorktown backend specification V1.1.0. https://ibm.biz/qiskit-yorktown, 2018a. Accessed: Dec. 2018.

IBM Q team. IBM Q 5 Tenerife backend specification V1.3.0. https://ibm.biz/qiskit-tenerife, 2018b. Accessed: Dec. 2018.

IBM Q team. IBM Q 16 Rueschlikon backend specification V1.1.0. https://ibm.biz/qiskit-rueschlikon, 2018c. Accessed: Dec. 2018.

IBM Q team. IBM Q 16 ibmqx3 backend specification V1.0.0. https://ibm.biz/qiskit-rueschlikon, 2018d. Accessed: Dec. 2018.

IBM Q team. IBM Q 16 Melbourne backend specification V1.1.0. https://ibm.biz/qiskit-melbourne, 2018e. Accessed: Dec. 2018.

Nurul Islam. *High-rate, high-dimensional quantum key distribution systems*. PhD thesis, 2018a.

Nurul T Islam. *High-Rate, High-Dimensional Quantum Key Distribution Systems*. Springer, 2018b.

E Knill, R Laflamme, and Milburn G. J. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, 2001.

Emanuel Knill. Quantum gates using linear optics and postselection. *Physical Review A*, 66 (5):052306, 2002.

Donald Ervin Knuth. *The Art of Computer Programming, Volume 4A*. Addison-Wesley, 2011.

Jens Koch, M Yu Terri, Jay Gambetta, Andrew A Houck, DI Schuster, J Majer, Alexandre Blais, Michel H Devoret, Steven M Girvin, and Robert J Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. *Physical Review A*, 76(4):042319, 2007.

P Kok, W. J. Munro, K Nemoto, C. Ralph, T, J. P. Dowling, and G. J. Milburn. Linear optical quantum computing with photonic qubits. *Reviews of Modern Physics*, 11, 2007.

Michael Kues, Christian Reimer, Piotr Roztocki, Luis Romero Cortés, Stefania Sciara, Benjamin Wetzel, Yanbing Zhang, Alfonso Cino, Sai T Chu, Brent E Little, et al. On-chip generation of high-dimensional entangled quantum states and their coherent control. *Nature*, 546(7660):622, 2017.

Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM journal of research and development*, 5(3):183–191, 1961.

Marco Lanzagorta. Quantum radar. *Synthesis Lectures on Quantum Computing*, 3(1):1–139, 2011.

Karel Lemr, Karol Bartkiewicz, and Antonín Černoch. Scheme for a linear-optical controlled-phase gate with programmable phase shift. *Journal of Optics*, 17(12):125202, 2015.

Tong Liu, Qi-Ping Su, Jin-Hu Yang, Yu Zhang, Shao-Jie Xiong, Jin-Ming Liu, and Chui-Ping Yang. Transferring arbitrary d-dimensional quantum states of a superconducting transmon qudit in circuit qed. *Scientific reports*, 7(1):7039, 2017.

Joseph M Lukens and Pavel Lougovski. Frequency-encoded photonic qubits for scalable quantum information processing. *Optica*, 4(1):8–16, 2017.

Duncan L MacFarlane, Jian Tong, Chintan Fafadia, Vishnupriya Govindan, L Roberts Hunt, and Issa Panahi. Extended lattice filters enabled by four-directional couplers. *Applied optics*, 43(33):6124–6133, 2004.

Duncan L MacFarlane, Marc P Christensen, Amr El Nagdi, Gary A Evans, Louis R Hunt, Nathan Huntoon, Jiyoung Kim, Tae W Kim, Jay Kirk, Tim P LaFave, et al. Experiment and theory of an active optical filter. *IEEE Journal of Quantum Electronics*, 48(3):307–317, 2011a.

Duncan L MacFarlane, Marc P Christensen, Ke Liu, Tim P LaFave, Gary A Evans, Nahid Sultana, TW Kim, Jiyoung Kim, Jay B Kirk, Nathan Huntoon, et al. Four-port nanophotonic frustrated total internal reflection coupler. *IEEE Photonics Technology Letters*, 24 (1):58–60, 2011b.

D Michael Miller and Mitchell A Thornton. Qmdd: A decision diagram structure for reversible and quantum circuits. In *Multiple-Valued Logic, 2006. ISMVL 2006. 36th International Symposium on*, pages 30–30. IEEE, 2006.

D Michael Miller and Mitchell A Thornton. Multiple valued logic: Concepts and representations. *Synthesis lectures on digital circuits and systems*, 2(1):1–127, 2007.

Shin-ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. pages 272–277, 1993. doi: 10.1145/157485.164890. URL https://doi.org/10.1145/157485.164890.

C. R. Myers and R. Laflamme. Linear optics quantum computation: an overview. In *Proceedings of the International School of Physics "Enrico Fermi"*, pages 45–93, 2006.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

Philipp Niemann, Rhitam Datta, and Robert Wille. Logic synthesis for quantum state generation. In *Multiple-Valued Logic (ISMVL), 2016 IEEE 46th International Symposium on*, pages 247–252. IEEE, 2016a.

Philipp Niemann, Robert Wille, David Michael Miller, Mitchell A Thornton, and Rolf Drechsler. Qmdds: Efficient quantum function representation and manipulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):86–99, 2016b.

J. L. O'Brien, G. J. Pryde, A. G. White, T. C. Ralph, and D. Branning. Demonstration of an all-optical quantum controlled-not gate. *Nature*, 426, 2003.

JS Otterbach, R Manenti, N Alidoust, A Bestwick, M Block, B Bloom, S Caldwell, N Didier, E Schuyler Fried, S Hong, et al. Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771*, 2017.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

J Randall, S Weidt, ED Standing, K Lake, SC Webster, DF Murgia, T Navickas, K Roth, and WK Hensinger. Efficient preparation and detection of microwave dressed-state qubits and qutrits with trapped ions. *Physical Review A*, 91(1):012322, 2015.

J Randall, AM Lawrence, SC Webster, S Weidt, NV Vitanov, and WK Hensinger. Generation of high-fidelity quantum control methods for multilevel systems. *Physical Review A*, 98 (4):043414, 2018.

Matthew Reagor, Christopher B Osborn, Nikolas Tezak, Alexa Staley, Guenevere Prawiroatmodjo, Michael Scheer, Nasser Alidoust, Eyob A Sete, Nicolas Didier, Marcus P da Silva, et al. Demonstration of universal parametric entangling gates on a multi-qubit lattice. *Science advances*, 4(2):eaao3603, 2018.

Rigetti Computing. QPU Specifications. https://rigetti.com/qpu, 2019a. Accessed: 5 Feb. 2019.

Rigetti Computing. pyQuil Documentation Release 2.4.0. https://media.readthedocs.org/pdf/pyquil/stable/pyquil.pdf, 2019b. Accessed: 5 Feb. 2019.

KR Rohit and Talabattula Srinivas. High dimensional quantum key distribution: Bb84 protocol using qudits. In *International Conference on Fibre Optics and Photonics*, pages Th3A–77. Optical Society of America, 2016.

Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In

Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on, pages 124–134. IEEE, 1994.

Pallavi Singh, Devendra Kr Tripathi, Shikha Jaiswal, and HK Dixit. All-optical logic gates: designs, classification, and comparison. *Advances in Optical Technologies*, 2014, 2014.

Kaitlin N Smith and Mitchell A Thornton. A multiple-valued logic synthesis tool for optical computing elements. In *Circuits and Systems Conference (DCAS), 2015 IEEE Dallas*, pages 1–4. IEEE, 2015.

Kaitlin N Smith and Mitchell A Thornton. Mustang-q: A technology dependent quantum logic synthesis and compilation tool (poster). In *Design Automation for Quantum Computers Workshop, IEEE International Conference on Computer Aided Design (ICCAD)*, 2017.

Kaitlin N Smith and Mitchell A Thornton. Automated mapping methods for the ibm transmon devices. In *International Workshop on Post-Binary ULSI Systems (ULSI-WS)*, 2018.

Kaitlin N Smith and Mitchell A Thornton. Higher dimension quantum entanglement generators. *Journal on Emerging Technologies in Computing (to appear)*, 2019a.

Kaitlin N Smith and Mitchell A Thornton. An open-source general compiler for quantum computers (poster). In *Free and Open Source Developers European Meeting (FOSDEM)*, 2019b.

Kaitlin N Smith and Mitchell A Thornton. Entanglement in higher-radix quantum systems. In *Symposium on Multiple-Valued Logic (ISMVL), 2019 IEEE International*, pages 162–167. IEEE, 2019c.

Kaitlin N Smith and Mitchell A Thornton. Entangled state preparation for non-binary quantum computing. In *International Conference on Rebooting Computing (ICRC)*. IEEE, 2019d.

Kaitlin N Smith and Mitchell A Thornton. A quantum computational compiler and design tool for technology-specific targets. In *International Symposium on Computer Architecture (ISCA)*. ACM, 2019e.

Kaitlin N Smith and Mitchell A Thornton. Quantum logic synthesis with formal verification. In *IEEE Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019f.

Kaitlin N Smith and Mitchell A Thornton. Fixed polarity pascal transforms with computer algebra applications. In *2019 Reed-Muller Workshop (RM 2019)*, 2019g.

Kaitlin N Smith and Mitchell A Thornton. Fixed polarity pascal transforms with symbolic computer algebra applications. In *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PACRIM)*. IEEE, 2019h.

Kaitlin N Smith, Michael A Taylor, Anna A Carroll, Theodore W Manikas, and Mitchell A Thornton. Automated markov-chain based analysis for large state spaces. In *Systems Conference (SysCon), 2017 Annual IEEE International*, pages 1–8. IEEE, 2017.

Kaitlin N Smith, Timothy P LaFave, Jr., Duncan L MacFarlane, and Mitchell A Thornton. Higher-radix chrestenson gates for optical quantum computation. *Journal of Applied Logics*, 5:1781–1798, 2018a.

Kaitlin N Smith, Timothy P LaFave, Jr., Duncan L MacFarlane, and Mitchell A Thornton. A radix-4 chrestenson gate for optical quantum computation. In *Symposium on Multiple-Valued Logic (ISMVL), 2018 IEEE International*, pages 260–265. IEEE, 2018b.

Kaitlin N Smith, Mitchell A Thornton, Duncan L MacFarlane, Timothy P LaFave, Jr., and William V Oxford. Single qubit quantum ring structures and applications (poster). In *Southwest Quantum Information and Technology 20th Annual SQuInT Workshop (SQuInT)*, 2018c.

Kaitlin N. Smith, Mathias Soeken, Bruno Schmitt, Giovanni De Micheli, and Mitchell A Thornton. Using zdds in the mapping of quantum circuits. In *Proceedings of 2019 Quantum Physics snd Logic (QPL)*, 2019.

Robert S Smith, Michael J Curtis, and William J Zeng. A practical quantum instruction set architecture. *arXiv preprint arXiv:1608.03355*, 2016.

Mathias Soeken, Stefan Frehse, Robert Wille, and Rolf Drechsler. Revkit: A toolkit for reversible circuit design. *Multiple-Valued Logic and Soft Computing*, 18(1):55–65, 2012.

Mathias Soeken, Martin Roetteler, Nathan Wiebe, and Giovanni De Micheli. Lut-based hierarchical reversible logic synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

N Sultana, Wei Zhou, Tim P LaFave Jr, and Duncan L MacFarlane. Hbr based inductively coupled plasma etching of high aspect ratio nanoscale trenches in inp: Considerations for photonic applications. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 27(6):2351–2356, 2009.

Maika Takita, Andrew W Cross, AD Córcoles, Jerry M Chow, and Jay M Gambetta. Experimental demonstration of fault-tolerant state preparation with superconducting qubits. *arXiv preprint arXiv:1705.09259*, 2017.

Michael A Taylor, Kaitlin N Smith, and Mitchell A Thornton. Sensor-based ransomware detection. In *Future Technologies Conference (FTC)*, pages 794–801, 2017.

RT Thew, Kae Nemoto, Andrew G White, and William J Munro. Qudit quantum-state tomography. *Physical Review A*, 66(1):012303, 2002.

Mitchell A Thornton, David W Matula, Laura Spenner, and D Michael Miller. Quantum logic implementation of unary arithmetic operations. In *Multiple Valued Logic, 2008. ISMVL 2008. 38th International Symposium on*, pages 202–207. IEEE, 2008.

Vinay Tripathi, Mostafa Khezri, and Alexander N. Korotkov. Operation and intrinsic error budget of a two-qubit cross-resonance gate. *Phys. Rev. A*, 100:012301, Jul 2019. doi: 10.1103/PhysRevA.100.012301. URL https://link.aps.org/doi/10.1103/PhysRevA.100.012301.

N Ya Vilenkin. Concerning a class of complete orthogonal systems. In *Dokl. Akad. Nauk SSSR, Ser. Math*, number 11, 1947.

Juan Yin, Yuan Cao, Yu-Huai Li, Sheng-Kai Liao, Liang Zhang, Ji-Gang Ren, Wen-Qi Cai, Wei-Yue Liu, Bo Li, Hui Dai, et al. Satellite-based entanglement distribution over 1200 kilometers. *Science*, 356(6343):1140–1144, 2017.

Tian Zhong, Hongchao Zhou, Robert D Horansky, Catherine Lee, Varun B Verma, Adriana E Lita, Alessandro Restelli, Joshua C Bienfang, Richard P Mirin, Thomas Gerrits, et al. Photon-efficient quantum key distribution using time–energy entanglement with high-dimensional encoding. *New Journal of Physics*, 17(2):022002, 2015.

Zeljko Zilic and Katarzyna Radecka. The role of super-fast transforms in speeding up quantum computations. In *Multiple-Valued Logic, 2002. ISMVL 2002. Proceedings 32nd IEEE International Symposium on*, pages 129–135. IEEE, 2002.

Zeljko Zilic and Katarzyna Radecka. Scaling and better approximating quantum fourier transform by higher radices. *IEEE Transactions on computers*, 56(2):202–207, 2007.