Chapter 1

Quantum Computing Basics and Concepts

1.1 Introduction

This book is for researchers and students of computational intelligence as well as for engineers interested in designing quantum algorithms in the circuit representation. The content of this book is presented as a set of design methods of quantum circuits with the focus on evolutionary algorithm; however some heuristic algborithms as well as a wide range of application of quantum circuits are provided.

The general idea behind this book is to represent every computational problem as a quantum circuit and then to use some classical synthesis approach to design the circuit. The goal of such approach is to describe and illustrate the use of classical design methods and their extension into quantum logic synthesis. The reason of using the circuit representation is that in classical logic synthesis various algorithms exist for the design of both combinatorial and sequential circuits and thus designing quantum algorithms in the circuit representation provides a good basis for comparison. Moreover the circuit representation is one that is the most explicit; at the same time it provides a good visual representation as well as it also allows a direct formalization and generalization of principles of both quantum computation and circuit design.

We know that Quantum Computation relies on quantum mechanics which is a mathematical model that describes the evolution of physical realization of computation and hence the computer itself. Several philosophically different but physically equivalent formulations have been found for quantum mechanics [Sty02]. In this book , we follow Schrödinger [Sch26] which describes the physical state of a quan-

tum system by a temporally evolving vector $|\phi\rangle$ in a complete complex inner product space H called a Hilbert space. The time evolution under the influence of a single term of the Hamiltonian is a single physical operation and in this book we will be designing and optimizing circuits at the level of such operations (pulses). (Hamiltonian is a physical state of a system which is observable corresponding to the total energy of the system. Hence it is bounded for finite dimensional spaces and in the case of infinite dimensional spaces, it is always unbounded and not defined everywhere).

The interesting fact about this book is the unified approach; in this book we use solely circuit representation (either direct such as wires and functions or more sophisticated representation such as a Reed-Muller form) to design logic circuuits, sequential machines or robot controllers for motion or machine learning. The target of all these circuits is to provide examples of application of quantum circuits and hopefully also show theri superiority over the classical circuits of the current technology.

Because this book is devoted to the computational aspects of designing quantum computers, quantum algorithms and quantum computational intelligence, one may ask "Why quantum computers are of interest and why are they more powerful than standard computers when used to solve problems in computational intelligence?" This is question is the main motivation for this introductory Chapter where the quantum computing is explained starting from its hisotrical context and ending in a description of quantum circuits and some of their properties.

1.2 Why quantum computing?

Quantum Mechanics (QM) describes the behavior and properties of elementary particles (EP) such as electrons or photons on the atomic and subatomic levels. Formulated in the first half of the 20th century mainly by Schrödinger [Sch26], Bohr [Boh08], Heisenberg [Cas] and Dirac [Dir95], it was only in the late 70's that quantum information processing systems has been proposed [Pop75, Ing76, Man80]. Even later, in the 80's of the last century it was Feynman who proposed the first physical realization of a Quantum Computer [Fey85]. In parallel to Feynman, Benioff [Ben82] also was one of the first researchers to formulate the principles of quantum computing and Deutsch proposed the first Quantum Algorithm [Deu85]. The reason that these concepts are becoming of interest to computer engineering community is mainly due to the Moore's law [Moo65]; that is: the number of transistors in a chip doubles every 18 months and the size of gates is constantly shrinking. Consequently problems such as heat dissipation and information loss are becoming very important for current and future technologies. Improving the scale of transistors ultimately leads to a technology working on the level of elementary particles

(EP) such as a single electron or photon. Since Moore's paper the progress led to the current 35 nm ($3.5 * 10^{-10}$m) circuit technology which considering the size of an atom (approximately 10^{-10}m) is relatively close to the atomic size. Consequently the exploration of QM and its related Quantum Computing becomes very important to the development of logic design of future devices and in consequence to the development of quantum algorithms, quantum CAD and quantum logic synthesis and architecture methodologies and theories. Because of their superior performance and specific problem-related attributes, quantum computers will be predominantly used in computational intelligence and robotics, and similarly to classical computers they will ultimately enter every area of technology and day-to-day life.

Despite the fact of being based on paradoxical principles, QM has found applications in almost all fields of scientific research and technology. Yet the most important theoretical and in the future also practical innovations were done in the field of Quantum computing, quantum information, and quantum circuits design [BBC$^+$95, SD96].

Although only theoretical concepts of implementation of complete quantum computer architectures have been proposed [BBC$^+$95, Fey85, Ben82, Deu85] the continuous progresses in technology will allow the construction of Quantum Computers in close future, perhaps in the interval of 10 to 50 years. Recent progress in implementation and architectures proove that this area is just at its begining and is gorwing. For instance the implementation of small quantum logic operations with trapped atoms or ions [BBC$^+$95, NC00, CZ95, DKK03, PW02] are the indication that this time-frame of close future can be potentially reduced to only a few years before the first fully quantum computer is constructed. The largest up to date implementation of quantum computer is the adiabatic computer by DWAVE [AOR$^+$02, AS04, vdPIG$^+$06, ALT08, HJL$^+$10]. Although up to now it is still an open issue whether the DWAVE computer is a proper quantum computer or not [], it provides consideerable speed up over classical computer in the SAT implementation and int the Random Number Generation []. In parallel to the adiabatic quantum computer, architectures for full quantum computers have been proposed [MOC02, SO02, MC]. In these proposals the quantum computations is implemented over a set of flying-photons that represents the degree of freedom of interactions between qubits. Such architectures however have not been implemented as of yet.

This chapter presents the basic concepts of quantum computing as well as the transition from quantum physics to quantum computing. We also introduce quantum computing models, necessary to understand our concepts of quantum logic, quantum computing and synthesis of quantum logic circuits. The Section 1.3 introduces some mathematical concepts and theories required for the understanding of quantum computing. Section 1.4 second section presents a historical overview of the

quantum mechanical theory and Section 1.5 presents the transition from quantum mechanics to quantum logic circuits and quantum computation.

1.3 Mathematical Preliminaries to Quantum Computing

According to [Dir84] each physical system is associated with a separate Hilbert space H. An H space is an inner product vector space where the unit-vectors are the possible states of the system. An inner product for a vector space is defined by the following formula:

$$\langle x, y\rangle = \sum_k x_k^* y_k \tag{1.1}$$

where x and y are two vectors defined on H and x^* denotes a complex conjugate of x. For quantum computation it is important to introduce the orthonormal basis on H, in particular considering the $\frac{1}{2}$-spin quantum system that is described by two orthonormal basis states. An orthonormal set of vectors M in H is such that every element of M is a unit vector (vector of length one) and any two distinct elements are orthogonal.

Example 1.3.0.1 Orthonormal basis set
An orthonormal basis set can be defined such as: $\{(1,0,0)^T, (0,1,0)^T, (0,0,1)^T\}$. In this space, a linear operator A represented by a matrix A transforms an input vector **v** to an output vector **w** such as $w = Av$.

1.3.1 Bra-Ket notation

One of the notations used in Quantum Computing is the bra-ket notation introduced by Dirac [Dir84]. Is it used to represent the operators and vectors; each expression has two parts, a bra and a ket. Each vector in the H space is a ket $|\Phi\rangle$ and its conjugate transpose is bra $\langle\Psi|$. The application of bra to ket results in the bra-ket notation $\langle|\rangle$. In the bra-ket notation, the inner product is represented by $\langle\psi_m|\psi_n\rangle = 1$, for $n = m$. By inverting the order and performing the ket-bra multipolication the outer product is obtained; it is given by $|\psi_m\rangle\langle\psi_n|$.

The information in quantum computation is represented by a qubit that in the Dirac notation can be written in the form of a characteristic equation. For instance a qubit with two possible orthonormal states $|0\rangle$ and $|1\rangle$ is described by eq. 1.2. The

deeper meaning of this equation will be explained in Section 1.5 of this chapter.

$$|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1.2}$$

1.3.2 Heisenberg Notation

In general, to describe basis states of a Quantum System, the Dirac notation is preferred to the vector based Heisenberg notation. This is mainly because the Dirac notation is much more practical than the Heisenberg notation for proving facts in Quantum Computing (Heisenberg notation is useful in computer calculations). However, the heinsenberg notation is much more explicit when one attempts to clearly explain the principles of quantum computations. Let the orthonormal quantum states be represented in the vector notation (Heisenberg notation) eq. 1.3.

$$\begin{aligned} |\uparrow\rangle = |0\rangle &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ |\downarrow\rangle = |1\rangle &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned} \tag{1.3}$$

1.3.3 Matrix Product

The multiplication of matrix A by vector **v** is defined be the following equation:

$$w[r] = \sum_{c} A[r,c] * v[c] \tag{1.4}$$

where r is the index of rows and c is the index of columns of the matrix. Such operator is bounded; it maps bounded sets to bounded sets.

From the equation (1.4) it follows that A is a projection, thus $\langle Av|v\rangle = ||Av||^2$ is called the *l2-norm* and measures the distance between the original vector **v** and the resulting vector A**v**. The A operator is called Hermitian if its hermitian conjugate $A^\dagger$ (conjugate transpose) satisfies $A^\dagger = A$ and a further extension of this property yields a *unitary* operator A. Such unitary operator is invertible and its inverse is given by its conjugate transpose $A^\dagger$ (also called Hermitian adjoint): $A^\dagger A = AA^\dagger = I$.

As will be seen in Section 1.5.2, all quantum events must be measured and all measurements are of a probabilistic nature. The inputs and the outputs to a quantum computational system are binary events (vectors) with probabilities in interval {0, 1} and the range of a projection is closed by A. The l2-norm of a projection of the vector **v** by A can be interpreted as a probability that a measurement will observe

the system in the state represented by A$\mathbf{v}$. The overall process of the input state being evolved and measured can be seen as a vector-matrix multiplication. The interested reader can find more information about the Hilbert space and quantum-probabilistic systems in [WG98, HSY+04, YHSP05].

In the above introduced dirac notation eq. 1.4 is rewriten to:

$$|w\rangle = A\,|v\rangle \tag{1.5}$$

Observe the introduction of the bra-ket notation considerably simplified eq. 1.4.

1.3.4 Kronecker Product

The combination of qubits into a multi-qubit system is mathematically given by the Kronecker multiplications; for a two-qubit system we obtain (using the Kronecker product [Gru99, Gra81, NC00]) the states represented in eq. 1.6:

$$\begin{aligned} |00\rangle &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad |10\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ |01\rangle &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad |11\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned} \tag{1.6}$$

Similarly for Operators, the Kronecker product exponentially increases the dimension of the space:

$$W \otimes H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{1.7}$$

This operation is shown in Figure 1.1.

Assume that qubit a (with possible states $|0\rangle$ and $|1\rangle$) is represented by $|\Psi_a\rangle = \alpha_a|0\rangle + \beta_a|1\rangle$ and qubit b is represented by $|\Psi_b\rangle = \alpha_b|0\rangle + \beta_b|1\rangle$. Each of them is represented by the superposition of their basis states, but put together the characteristic wave function of their combined states will be:

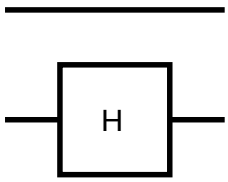


Figure 1.1: Circuit representing the $W \otimes H$ operation

$$|\Psi_a \Psi_b\rangle = \alpha_a\alpha_b|00\rangle + \alpha_a\beta_b|01\rangle + \beta_a\alpha_b|10\rangle + \beta_a\beta_b|11\rangle \tag{1.8}$$

with α_a and β_b being the complex amplitudes of states of each EP respectively. As shown before, the calculations of the composed state are achieved via the Kronecker multiplication operator. Hence come the quantum memories with extremely large capacities mentioned earlier and the requirement for efficient methods to calculate such large matrices.

1.3.5 Matrix Trace

A trace of a matrix is defined as $tra(U) = \sum_i D_{ii}$ and as it will be seen the concept of trace is used in the measurement operation in quantum computing. In particular it is required when dealing with ensemble systems [CFH97, NC00] and estimating their state. Such systems are represented by density matrices of the form:

$$\rho = \sum_{i}^{2^n} \alpha_i|\psi_i\rangle\langle\psi_i|\alpha_i^* = \sum_{i}^{2^n} p_i|\psi_i\rangle\langle\psi_i| \tag{1.9}$$

with $\sum_i^{2^n} p_i = 1$, α being the complex coefficient such that $|\alpha_i|^2 = p_i$.

The trace operator represents the possible observable states of a quantum system. Any quantum state $|\phi\rangle$ when observed collapses according to the applied measurement resulting in $\alpha|\phi\rangle \rightarrow p|\phi\rangle\langle\phi|$, with p being the probability of observing the state $|\phi\rangle$ from the set of all possible output states. Thus representing the overall state of a quantum system can be represented as the trace $\sum_{i=0}^{2^n} p_i|i\rangle\langle i|$ with p_i being the probability of observing the state $|i\rangle$.

1.4 Quantum Mechanics

1.4.1 Bohr Particle Model

The term "quantum" describes the fact that the EP's can be observed (measured) only in distinct energetic states and while moving from one state to another a quantified amount of energy is either emitted or absorbed. A closer look at the Bohr model of the atom will explain these notions even more. The example we are using here is based on the simplest of all atoms, the Hydrogen (H) atom. As all atoms, the Hydrogen atom (H) is composed of a nucleus and electrons orbiting around it, but H has only one electron (e). The electron can be only on orbits of certain allowed radii. When e is on the orbit that is closest to the nucleus then the atom is in the "ground state".

The electron can change orbits; going from a lower orbit to a higher one requires absorption of some energy and leaving an orbit for a lower one is characterized by emitting a quantum of energy from the electron. The energy levels that the electron can visit are characterized by the following equation:

$$E_n = (R_h)\left(\frac{1}{n^2}\right) \tag{1.10}$$

where R_h is the so-called *Rydberg constant* ($2.18 * 10^{-18} J$) and n is the *principle quantum number* corresponding to different allowed orbits of the electron. The difference of energy E associated with "orbits-jumping" can be expressed as the difference between the energy of the electron on the initial E_i and the final E_f orbit:

$$\Delta E = E_f - E_i \tag{1.11}$$

Max Planck has deduced that the energy of electrons comprising the electro-magnetic radiation is a function of frequency, from where his famous formula comes:

$$\Delta E = h\nu = -R_h\left(\frac{1}{n_f^2} - \frac{1}{n_i^2}\right) \tag{1.12}$$

where h is the *Planck constant* (6.63 x 10-34 Js) and v is the frequency of the emitted light (Figure 1.2).

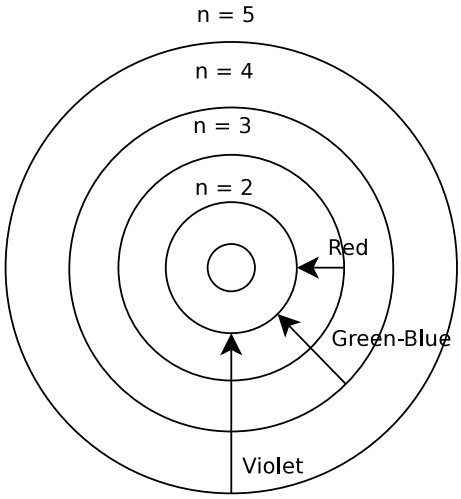


Figure 1.2: Bohr model of the atom (nucleus, orbiting electrons). Shown are light colors respective to the electron orbit transitions.

1.4.2 Quantum Model of Elementary Particle

This brief look into the physical background should be completed by the fact that Bohr's model of atom assumed that the electron is orbiting the nucleus similarly the Earth is orbiting around the Sun, which violates the Heisenberg uncertainty principle of Quantum Mechanics [Cas]. This principle states that the position and the momentum of an EP cannot be simultaneously determined with certainty. In particular in quantum mechanics, any elementary particle has the property that the root-mean-square deviation of the position x from the mean $\Delta x = \sqrt{\langle x^2 \rangle - \langle x \rangle^2}$ (where $\langle \cdot \rangle$ represents $x * x^*$) and root-mean-square of the momentum p from the mean $\Delta p = \sqrt{\langle p^2 \rangle - \langle p \rangle^2}$ multiplied together is never smaller than $\frac{\hbar}{2}$. This is also expressed by the commutator:

$$[x, p] = i\hbar \tag{1.13}$$

The introduction of these unusual properties was required to correctly describe the QM system (sometimes also referenced as a "failure" of the classical Bayesian statistics [You95]) and to allow predicting states of Physical Quantum Systems.

Example 1.4.2.1 Two-Slit experiment
In the two-slit experiment, the dual nature of EP was shown. The experiment consists of the emitter (device firing EP on a screen), of the screen with two holes and of the detector. Figure 1.5 illustrates the experimental setting. The system

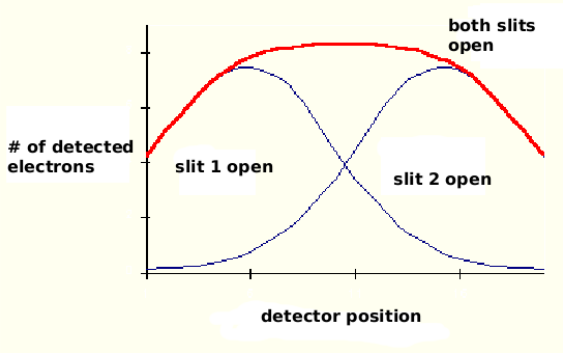


Figure 1.3: The measured number of electrons on the detector screen with top slit or the bottom slit open (thin lines). The expected probability when both slits are open (thick line).

is setup so that the electrons detected by the detector have to travel through the open holes (the screen is thick enough to stop the electrons completely). When only one of the two slits is open and the observer looks at (measures) the projection of fired particles on the detector, the distribution of their locations is proportional to a linear trajectory through the opened slit (photons behave like particle). The paradox shows up when both the slits are opened.

Figure 1.3 shows the detection screen and the number of electrons measured when either the top or the bottom slit is open. Two curves show the distribution of particles on the detector screen either with top or with the bottom slit open. The thick curve is an expectation of what should be the particle distribution with both slits opened based on the classical probability theory. What appears to be a classical probabilistic distribution of particles with only one of both slits open, is transformed to an interference pattern with both slits open (Figure 1.4), not obtainable using classical statistics.

When this measurement was made the problem was to interpret it and to decide whereas EP's travel in space on a straight line (as particles) or if they have wave properties. The problem was to determine how an EP (electron or photon) has the particle characteristics (mass and speed) when measured and could behave as a wave at the same time? The dual nature problem is solved by the supposition that the EP is a particle while the measurement is performed, and the EP behaves like a wave while not.

According to Figure 1.5 it is not possible to decide whereas a particle traveled through one, two or both slits simultaneously because the measurement does not allow determining it. If a measurement of particles is done on the screen, the result

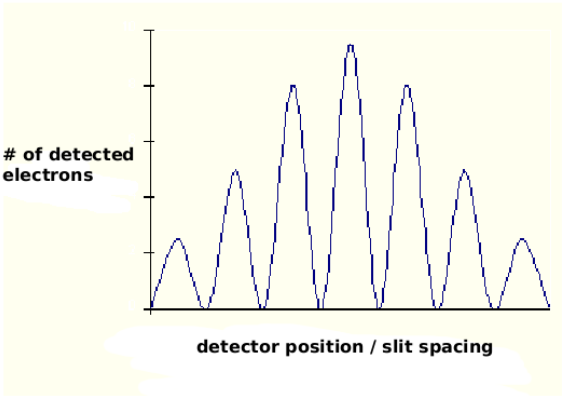


Figure 1.4: Results of measurement of particles position when both slits of the screen are open.

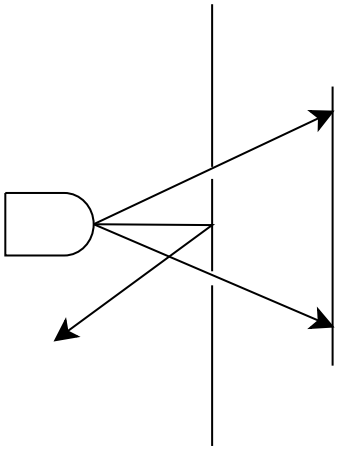


Figure 1.5: Schematic representation of the two-slit experiment. Left is the emitter and on the right is the detector (film). In the middle is the barrier with two holes.

will yield 50% of particles through left slit and 50% through the right slit. The consequence of these observations is the fact that while recording probabilities of detecting an electron in the interference pattern, the probabilities of observation of a given state can be smaller than in the standard Bayesian probabilistic model! This implies the following contradictory equation 1.14 [You95].

$$P(x) = P(x|slit_1) + P(x|slit_2) \leq P(x|slit_1) \tag{1.14}$$

where P(x) is the probability of measuring a particle on position x. The solution to this problem was the introduction of the concept of the complex probability amplitudes because such amplitudes can cancel each other. The system describe by eq. 1.14 is then mathematically a set of functions mapping real physical states from Hilbert space H into a complex space C:

$$\Psi : S \rightarrow C \tag{1.15}$$

where S is the physical space of states and C is complex vector space, respectively. As will be seen later, the functions from the set described by eq. 1.15 are the wave functions that represent non-trivial states of the quantum system. This state of a particle traveling through both slits is defined as the "superposed" state of the system. For one-particle system this superposition is a result of all its possible states that it is measured for. Its complex probability amplitude α is related to the classical probability p of measuring this system in a particular state by $|\alpha|^2 = p$. In a system of n particles (called also the quantum register), the system constitutes a superposition of $m * n$ states where m is the number of states of each elementary unit of this system.

1.4.3 Schrödinger equation

The general solution for quantum mechanical events is given by the Schrödinger equation:

$$i\hbar \frac{d|\psi\rangle}{dt} = H|\psi\rangle \tag{1.16}$$

with $\hbar$ being the Planck constant [Pla] and H being the Hamiltonian of the system. A Hamiltonian represents the observable corresponding to the total energy of the

system. In particular, the possible observable states are represented by the spectrum of the Hamiltonian [Dir84]. This general equation describes the natural evolution of a quantum system. The $\hbar$ constant can be absorbed into the Hamiltonian H. The Hamiltonian H can for example represent a particle that exists in the infinite one-dimensional potential such as the Simple Harmonic Oscillator (SHO) [MV76] model with Hamiltonian $H = \frac{p^2}{2m} + \frac{1}{2}m\omega^2x^2$ where p is the momentum, m is the mass, x is the position and ω is the angular velocity. The Schrödinger equation for SHO takes the form of:

$$\frac{-\hbar}{2m}\frac{d^2\psi_n(x)}{dx^2} + \frac{1}{2}\beta x^2\psi_n(x) = E\psi_n(x) \tag{1.17}$$

where $V(x) = \frac{1}{2}\beta x^2$ is the potential well with $\omega = \sqrt{\frac{\beta}{m}}$ (and β being the spring constant). In general the solution that one obtains when solving for physical systems is a solution to eq. 1.16 which is of the form:

$$|\psi(t)\rangle = e^{-iHt/\hbar}|\psi(0)\rangle \tag{1.18}$$

or more clearly as

$$|\psi(t)\rangle = e^{-in\omega t}|\psi(0)\rangle \tag{1.19}$$

with ω being the angular frequency and n being the index of distinct non-degenerate quantum states. From the eq. 1.19 and with respect to previous stipulations (or postulates of quantum mechanics) it can be observed that the resulting state is a distribution of corresponding probabilities p_n over the set of eigenstates n [MV76]. For more details, this particular problem has all solutions represented by Hermite polynomials [MV76, Wey32], however it is not the focus of this book.

1.4.4 Superposition of quantum states

The previous descriptions introduced a very unique phenomena: while a particle behaves as a wave it can be simulateneously in various basis states but when it is measured (as in the two-slit experiment) the basis state reveals it self a physically observable unique state. The state illustrated by eq. 1.19 is also called the quantum supperposition of states because a single physical particle contains a multitude of

observable staes that have a finite probability to collapse onto the orthonormal bases states. A more understandable explanation is to represent the quantum state as a state that is build of component observable states (with particular probabilities) and that when observed will be seen as one of the observables with the associated probability.

Let's illustrate the superposition phenomena by work done in [MMKW96]. In this experiment described is the creation of a "Schrödinger cat" state of an atom. The "Gedankenexperiment" of Schrödinger was to place a living cat into a superposition of being alive and dead. These superposed states in QM are described by a wave function. The cat state can be described in these terms as follows:

$$|\Psi\rangle = \frac{|\uparrow\rangle + |\downarrow\rangle}{\sqrt{2}} \tag{1.20}$$

where $|\uparrow\rangle$ and $\downarrow\rangle$ refer to the states of a living and dead cat, respectively. Once again this situation is not "realistic" in our macro-world, however appropriate for the QM. The interpretation of the general equation 1.20 is that for each measurement of the system described by it there is 50% chance to find the system in state $|\downarrow\rangle$ and a 50% chance to find the system in state $|\uparrow\rangle$. This can be formalized in Dirac's notation as:

$$\Psi = \alpha|\uparrow\rangle + \beta|\downarrow\rangle \tag{1.21}$$

$$= \frac{1}{\sqrt{2}}|\uparrow\rangle + \frac{1}{\sqrt{2}}|\downarrow\rangle \tag{1.22}$$

with

$$|\alpha|^2 + |\beta|^2 = 1 \tag{1.23}$$

Where α and β are in general complex amplitudes associated with each measured state, and $|\alpha|^2 = \alpha\alpha^*$ where $\alpha^* = (a + ib)^* = (a - ib)$ is a complex conjugate of α. Thus both states $|\uparrow\rangle$ and $|\downarrow\rangle$ will be observed when measured with probabilities $|\alpha^2|$ and $|\beta^2|$ respectively. This interpretation of the system defined by (1.22), shows that any quantum system can be represented by a wave function describing all possible states of the system (here we assume two orthonormal states $|\downarrow\rangle$ and $|\uparrow\rangle$) by using complex probabilities (here α and β). The complex probabilities are restricted only by the second equation in (1.23), and the observables of this quantum system are valued in the range of $\{0, 1\}$.

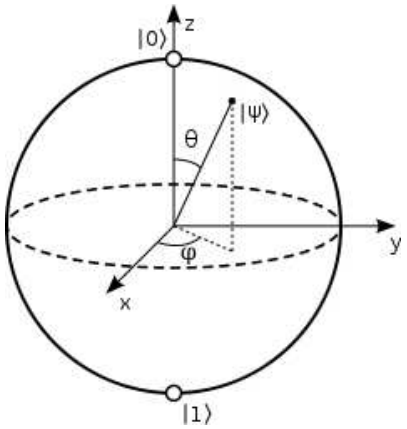


Figure 1.6: The Bloch sphere

The above representation is another notation of a Euler parametrized 3D rotation. The general state of a qubit rotation is given in eq. 1.24. Observe how demonstrated in this equation, a global phase $e^{i\rho}$ is visible, but in general it is ignored during the computation with qubits as it can be easily factored out and moreover, upon measurement it is completely destroyed.

$$|\psi\rangle = e^{i\rho}cos\frac{\theta}{2}|0\rangle + e^{i(\rho+\phi)}sin\frac{\theta}{2}|1\rangle = R(\rho,\phi,\theta) = e^{i\rho}\begin{pmatrix} cos\frac{\theta}{2} & -e^{-i\phi}sin\frac{\theta}{2} \\ e^{i\phi}sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{pmatrix} \tag{1.24}$$

The equation 1.24 is more clearly visualized on a sphere, commonly known as the Bloch sphere. This representation is usefull situated to show the state of a single qubit but does not allow to represent multiple qubits due to entanglement and the supperposition. The Bloch sphere is shown in Figure 1.6.

Now, stepping back to [You95], if instead of a cat we consider some of alkali-like ions such as 40Ca+, 24Mg+ or 198Hg+ which do not have a third electronic ground state available for the auxiliary level (thus can be directly used for quantum permutative computing) [MMKW96], the equations (1.22) and (1.23) would represent the wave function of an ion where $|\downarrow\rangle$ and $|\uparrow\rangle$ are two distinct energetic states.

As mentioned earlier in this section (and as will also be seen in Section 1.5), operations on single trapped ions have been already implemented [MMK$^+$95, MMKW96, MML$^+$98, WMI$^+$05]. This technique consists in trapping in an Electro-Magnetic field one or more ions and using laser beams setting these ions into certain states. Applying a well-determined sequence of laser pulses on particular ions in the trap, one can achieve such gates as NOT (an inverter of classical logic design). Again the setting of ions is represented by jumps of electron on their orbits and emitting or absorbing quanta of energy. It also allows realizing one-qubit, two-qubit or even three

qubit gates. However, no large-scale quantum computer was yet experimentally created using ion-traps.

1.5 From Quantum Mechanics to Quantum Logic

In the Section 1.4, the example of implementation of a quantum computer was mentioned: the trapped ions in the EM field interacting with laser beams. This has for consequence the changes of states of the ions such as moving from ground state to an excited state by an electron jump from a lower orbit to a higher one. But electrons are more complicated than just quantum energy collectors and their wave functions are more complex because they depend on three parameters beside the principle quantum numbers; the orbital quantum number "l", the magnetic quantum number "m" and the spin "S". While parameters l and m determine the angular dependence, the S determines the internal electron rotation. For our explanation it is only important to know that the S number is often used to represent basis states in a Quantum Computation because the hydrogen atom can have only two values $\pm\frac{1}{2}$ of spin. Consequently, using spin rotations for the basis implies to work directly with the two-valued (binary)quantum logic. Now, considering the spin S value as the basis, a quantum operator on an atom will result in a rotation of the electron spin. Consequently all single qubit operations can be expressed as rotations by certain angles [NC00].

Moreover to build a quantum computer a multi-qubit system (also called quantum register) requires to be defined and analyzed. Beside the quantum register definition addtional operations such as measurement have to be explained.

1.5.1 Multi-Qubit System

To illustrate these important properties let's have a look on a more complicated system with two quantum particles a and b represented by $|\psi_a\rangle = \alpha_0|0\rangle + \beta_a|1\rangle$ and $|\psi_b\rangle = \alpha_b|0\rangle + \beta_b|1\rangle$ respectively. For such a system the problem space increases exponentially and is represented using the Kronecker product [Gru99].

$$|\psi_a\rangle \otimes |\psi_b\rangle = \begin{bmatrix} \alpha_0 \\ \beta_0 \end{bmatrix} \otimes \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{bmatrix} \tag{1.25}$$

Thus the resulting system is represented by $|\psi_a\psi_b\rangle = \alpha_a\alpha_b|00\rangle + \alpha_a\beta_b|01\rangle + \beta_a\alpha_b|10\rangle + \beta_a\beta_b|11\rangle$ (1.8) where the double coefficients obey the unity(completeness) rule and

each of their square powers represents the probability to measure the corresponding state. The superposition means that the quantum system is or can be in any or all the states at the same time. This superposition potentially gives the massive parallel computational power to quantum computing.

Anoterh property of the multi-qubit register is the multi-qubit superposition. Assume a system with n qubits. A classical register represents 2^n distinct states while the quantum system is an arbitrary superposition of these 2^n states. Practically it means that while not measured the system can be in one, two, three ... or all of the 2^n states at the same time!

1.5.2 Simple Projective Measurement

As the states of qubits are vectors with complex coefficients, the real (Boolean) logic state (also called the observable) is obtained by measuring the system and observing the result. The measurement process projects the measured qubit onto the set of real valued observables. In other words a quantum system is described by a set of *filters*, each letting through and capturing a particular state. As will be seen later in Chapter **??**, for logic design this implies that not only one can measure for a set of observables to design a function but also one can use various sets of observables in order to obtain different functions.

Definition 1.5.1 (Projective measurement)**.** *The measurement of a single binary qubit is described by the overall probability of observing both orthonormal states given by* $p(0) + p(1) = 1$*. Thus, in a more general way, any (*$\frac{1}{2}$ *- spin) quantum system is described by:*

$$p(m) = \sum_n \langle \Psi | M_n^\dagger M_n | \Psi \rangle \tag{1.26}$$

where $p(m)$ *is the probability to measure value* m*,* Ψ *is the wave representation of the circuit and* M_n *is the measurement operator for the value* n*. From the above it is simple to derive the complete description of a single-qubit circuit as:*

$$p(0) + p(1) = \langle \Psi | M_0^\dagger M_0 | \Psi \rangle + \langle \Psi | M_1^\dagger M_1 | \Psi \rangle = 1 \tag{1.27}$$

where

$$M_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } M_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{1.28}$$

and where $M_0^\dagger$ and $M_1^\dagger$ are respective Hermitian conjugates of M_0 and M_1. Because

$$M_0^\dagger M_0 + M_1^\dagger M_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \tag{1.29}$$

, and the measurement given by $M = \sum_k M_k = M_0 + M_1$ (representing all possible outcome values) describes the quantum system completely. □

Similarly to the single qubit measurement the two qubit measurement operation can be designed from single qubit measurments. To start, one can look at the single qubit measurements on a two qubit register. Equation 1.30 shows the measurement of a two qubit system for the $|01\rangle$ state.

$$M_{|01\rangle} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.30}$$

Similarly, to the described trace operation in Section 1.3 one can measure only a sub-part of a quantum register. Thus to measure a single qubit of a two qubit system, the un-measured qubit is transformed using an identy operation and thus:

$$M_{|1-\rangle} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.31}$$

Eq. 1.31 shows that similarly to the trace operation, in order to obtain from a two qubit system a single qubit state, the other qubit has to be measured. The result of this oepration is that starting from a quantum state $|ab\rangle$ and measuring then qubit b, the result is the quantum state of the qubit a.

Example 1.5.2.1 Entanglement
Assume the above two-particle vector is transformed using the quantum circuit from Figure 1.7.

This circuit executes first a Hadamard transform on the top qubit and then a Controlled Not operation with the bottom qubit as the target. Depending on the initial state of the quantum register the output will be either $|\phi\rangle = |ab\rangle = \alpha_a\alpha_b|00\rangle \pm |\beta_a\beta_b|11\rangle$ or $|\phi\rangle = |ab\rangle = \alpha_a\beta_b|01\rangle \pm |\beta_a\alpha_b|10\rangle$. Thus it is not possible to estimate with 100% probability the initial state of the quantum register.

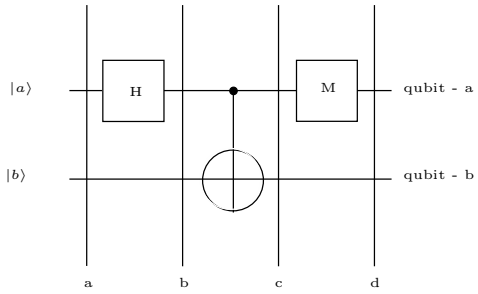


Figure 1.7: EPR producing circuit

Let $|ab\rangle = |00\rangle$ at level a (Figure 1.7). The first step is to apply the $[H]$ gate on the qubit-a and the resulting state at level b of the circuit is

$$\begin{aligned} |ab\rangle &\rightarrow (H \otimes W)|ab\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \end{aligned} \tag{1.32}$$

Next the application of the CNOT gate results in:

$$|ab\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.33}$$

For an output 0 (on the qubit-a), the projective measurement of the first (topmost) qubit (qubit-a on Figure 1.7) on this stage would collapse the global state (with a single measurement) to the state $|00\rangle$:

$$|ab\rangle \rightarrow \frac{M_0|ab\rangle}{\sqrt{\langle ab|M_0^\dagger M_0|ab\rangle}} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T = |00\rangle \tag{1.34}$$

with

$$M_{0-}|ab\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1.35}$$

and

$$\begin{aligned} \sqrt{\langle ab|M_{0-}^{\dagger} M_{0-}|ab\rangle} &= \sqrt{\frac{1}{2}} \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \end{aligned} \tag{1.36}$$

Observe that because the measurement operators are positive-definite the product $M_{0-}^{\dagger} M_{0-} = M_{0-}$. Note that we naturally extended the single qubit measurement operators from eq. 1.28 to multi qubit measurement such as M_{00} or M_{01}. This is formally possible despite the fact that as a result of the contemporary measurement technology, only single qubit measurements are allowed and to detect multiple qubit states, synchronous single-qubit measurements must be executed [LBA+08].

Similarly, the probability of measuring output on the qubit-a in state $|0\rangle$ is:

$$\begin{aligned} p(0) &= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\ &= \frac{1}{2} \end{aligned} \tag{1.37}$$

If one would look to the output of the measurement on the second qubit (qubit-b), the probability for obtaining $|0\rangle$ or $|1\rangle$ is in this case the following:

$$
\begin{aligned}
p(0) &= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\
&= p(1) = \frac{1}{2}
\end{aligned}
\tag{1.38}
$$

Thus the expectation values for measuring both values 0 or 1 on each qubit independently are $\frac{1}{2}$.

If however one looks on the second and non-measured qubit (if the qubit-a is measured, it is the qubit-b, and vice versa) and calculates the output probabilities, the output is contradictory to the expectations given by standard probabilistic distribution such as a coin toss $q = 1 - p$. To see this let's start in the state

$$
\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}
\tag{1.39}
$$

and measure the qubit-a and obtain a result. In this case assume the result of the measurement is given by:

$$
|\Psi\rangle \rightarrow \frac{M_0|\Psi\rangle}{\sqrt{\langle\Psi|M_0^\dagger M_0|\Psi\rangle}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\tag{1.40}
$$

Then measuring the second qubit (qubit-b) will not affect the system because the measurement of the qubit-a has collapsed the whole system into a single basis state:

$$
|\Psi\rangle \xrightarrow{M} |00\rangle
\tag{1.41}
$$

The probability for obtaining a $|1\rangle$ on the qubit-b is thus 0 and the measurement on qubit-b (after having measured qubit-a) has no effect on the system at all. This non-locality paradox was first described by Einstein-Podolsky-Rosen work [EPR35] and is known as the EPR paradox.

(1.42)
$$|\uparrow\uparrow\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}},\ |\uparrow\downarrow\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}},\ |\downarrow\uparrow\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}},\ |\downarrow\downarrow\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

Equation 1.42 shows the so-called Bell states. These states are the example of entangled basis states that can be used for quantum computation. Observe, that mathematically, the entangled states are such that they cannot be factorized in simpler terms. For example, the state $\frac{(|00\rangle+|01\rangle)}{\sqrt{2}} \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle$ and thus is factorizable. However, the states as those introduced in eq. 1.33 cannot be factorized in such a manner and are thus entangled; physically implying that they are related through measurement or observation. This particular phenomenon is one of the most powerful in quantum mechanics and quantum computing, as it allows together with superposition the speedup of solutions to certain types of problems. □

1.5.3 Density Matrix and POVM

When dealing with measurement and representation of systems that are not completely known it is useful to represent the system using the Density Matrix representation [NC00]. A quantum system $|\psi\rangle$ spanning a Hilbert space on basis states $|0\rangle$ and $|1\rangle$, with p_i the probability of observing value i can be represented as:

(1.43)
$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

With $|\psi_i\rangle\langle\psi_i|$ being the outer-product, ρ is the density matrix and p_i is the probability of observing the given collapsed state $|\psi_i\rangle\langle\psi_i|$. For example, for a system

described by $|\phi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$ we have:

$$\begin{aligned}\frac{|00\rangle\langle 00|}{2} + \frac{|01\rangle\langle 01|}{2} &= \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\0\end{bmatrix} \otimes \begin{bmatrix}\frac{1}{\sqrt{2}} & 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0\\\frac{1}{\sqrt{2}}\\0\\0\end{bmatrix} \otimes \begin{bmatrix}0 & \frac{1}{\sqrt{2}} & 0 & 0\end{bmatrix} \\ &= \begin{bmatrix}\frac{1}{2} & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\0 & \frac{1}{2} & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix} \\ &= \begin{bmatrix}\frac{1}{2} & 0 & 0 & 0\\0 & \frac{1}{2} & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix}\end{aligned} \tag{1.44}$$

To make a complete description one needs to determine precisely the probability of observation of the states $|00\rangle\langle 00|$ and $|01\rangle\langle 01|$. Thus density matrix corresponding to the system from eq. 1.44, is shown in eq. 1.45.

$$\rho = \begin{bmatrix}\frac{1}{2} & 0 & 0 & 0\\0 & \frac{1}{2} & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix} \tag{1.45}$$

Let M_k be a measurement operator for the quantum state $|k\rangle$ such as $M_{00} = |00\rangle\langle 00|$, the density matrix is calculated using the trace:

$$\begin{aligned}&\sum_i p_i \langle\psi_i|M_k^\dagger M_k|\psi_i\rangle \\ &\qquad = \sum_i p_i tra(M_k^\dagger M_K|\psi_i\rangle\langle\psi_i|) \\ &\qquad = tra(M_K^\dagger M_K \rho)\end{aligned} \tag{1.46}$$

The final state of the system in the post-measurement state $|\psi_k\rangle$ is described by applying the given operator to the quantum state. Thus one of the measurement operators from $M = \sum_k M_k$ is used and the final state is represented as:

$$|\psi_k\rangle = \frac{M_k|\psi\rangle}{\sqrt{\langle\psi|M_k^\dagger M_k|\psi\rangle}} \tag{1.47}$$

In density matrix notation this can be expanded to:

$$\rho_k = \sum_i p_i \frac{M_k|\psi_i\rangle\langle\psi_i|M_k^\dagger}{\sqrt{\langle\psi_i|M_k^\dagger M_k|\psi_i\rangle}} = \frac{M_k \rho M_k^\dagger}{tra(M_K^\dagger M_K \rho)} \tag{1.48}$$

The density matrix is also useful to describe quantum ensemble systems (quantum system constructed from multiple independent subsystems). Such system represents a set of prepared states. For instance, a set of prepared states such that 50% of them is in $|0\rangle\langle 0|$ and 50% in $|1\rangle\langle 1|$ is said to be in a mixed state (a statistical average). The density matrix of such state is given in eq. 1.49.

$$\rho_R = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \tag{1.49}$$

In contrast, a system in a state $|\phi\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)$ is said to be in a pure state (eq. 1.50).

$$\begin{aligned} \rho_Q = |\phi\rangle\langle\phi| =& \frac{1}{4}[|00\rangle\langle 00| + |00\rangle\langle 01| + |00\rangle\langle 10| + |00\rangle\langle 11|] \\ &+ \frac{1}{4}[|01\rangle\langle 00| + |01\rangle\langle 01| + |01\rangle\langle 10| + |01\rangle\langle 11|] \\ &+ \frac{1}{4}[|10\rangle\langle 00| + |10\rangle\langle 01| + |10\rangle\langle 10| + |10\rangle\langle 11|] \\ &+ \frac{1}{4}[|11\rangle\langle 00| + |11\rangle\langle 01| + |11\rangle\langle 10| + |11\rangle\langle 11|] \end{aligned} \tag{1.50}$$

The trace $tra(\rho^2)$ is also a measure for whether the system is in pure state or in a mixed quantum state. It can be seen, that given ρ_R, the trace $tra(\rho_R^2) < 1$ (the system is in mixed state) while $tra(\rho_Q^2) = 1$ because it is a pure quantum state [NC00]. In other words, a pure state is a quantum state that can be represented by a single ket vector (a probability of 1) while a mixed state is a statistical distribution of states. Thus a pure state has a density matrix with a single probability $p = 1$.

This distinction can be observed from the logic point of view; quantum operators corresponding to boolean reversible functions will be represented as permutation matrices while quantum operators with probabilistic outcomes will be represented by unitary operators having coefficients (possibly complex) $(\sqrt{|\alpha|^2 = 0}) < |\alpha| < (\sqrt{|\alpha|^2 = 1})$. A similar observation can be made for a density matrix; a density matrix such that $tra(\rho^2) = 1$ will represent a permutative reversible logic function.

Example 1.5.3.1
Permutative and non-Permutative matrices Let $f(a, b, c)$ be a reversible logic function defined by a $f = a \oplus b \oplus c$ with logic table shown in Table 1.1. Observe that

Table 1.1: Logic function $f(a, b, c) = a \oplus b \oplus c$

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

this reversible function can be written as a matrix and the input-output pair as a vector. For three variable fucntion an input vector will have 2^3 coefficients, each representing the presence or the absence of the minterm in the input state. For isntance the input logic state 011 = [00010000] and the logic state 010 = [00100000]. The permutative matrix representing the function $f(a, b, c)$ is

$$f(a,b,c) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.51}$$

Such matrix is called permutative matrix because the operation it performs is a permutation of the inputs.

So far in this chapter we have been using the Projective Measurement (illustrated in examples above), however other types of measurement are also possible. Another type of measurement (much more realistic) is the Positive valued-operator measurement (POVM). The importance of POVM is due to the fact that the projective measurement is not well suited to measure for states in non orthonormal bases. Moreover POVM, unlike the projective measurement case, does not always allow to determine the complete state of the system after a measurement (contrary to the case of projective measurement operator $M = \sum_k p_k M_k$) but rather allows to make prediction about the probabilities of the different possible measurement outcomes represented by the density matrix. This is because if one desires to measure two non orthonormal states and with the condition that every quantum system is completely described by $M = \sum_k p_k M_k$, then because the two desired states do not span the

complete underlying Hilbert space, there is a state that when measured nothing can be said about the observed result. This measurement is much more realistic.

Example 1.5.3.2 Projective Measurement of non-orthonormal states
For example assume we have a quantum system $|\Phi\rangle$ in a Hilbert space spanned by 2 orthonormal basis states $|0\rangle$ and $|1\rangle$. In this setting the projective measurement allows to determine completely the state of the system. Now assume that our basis states are not orthonormal, and we use for measurement the states $|0\rangle$ and $\frac{(|0\rangle+|1\rangle)}{\sqrt{2}}$ [NC00]. The projective measurement operators for these states are shown in equation 1.52.

$$\rho_0 = |0\rangle\langle 0| = \begin{pmatrix} 10 \\ 00 \end{pmatrix} \quad \text{and} \quad \rho_a = \frac{1}{2}(|0\rangle + |1\rangle)(\langle 0| + \langle 1|) = \frac{1}{2}\begin{pmatrix} 11 \\ 11 \end{pmatrix} \tag{1.52}$$

Looking now to the outcome of the possible measurements by using $|\psi\rangle = \frac{M|\psi\rangle}{\langle\psi|M^\dagger M|\psi\rangle}$, for a initial state $|0\rangle$ we obtain:

$$\frac{\rho_0|0\rangle}{\sqrt{\langle 0|\rho_0^\dagger \rho_0|0\rangle}} = 1 \quad \text{and} \quad \frac{\rho_1|0\rangle}{\sqrt{\langle 0|\rho_1^\dagger \rho_1|0\rangle}} = \frac{1}{2} \tag{1.53}$$

The obtained measurement results in value 1 when measuring using the right measurement basis. However, in the case when the result of measurement is probabilistic (observing state $|0\rangle$ and state $|1\rangle$ with equal probability), the state prior to the measurement cannot be determined with certainty as it could be either state of them. □

As can be seen in the above example, the projective measurement defined is underdimensioned to capture all the features of this system. One solution to this problem is to create projective measurement in a higher dimensional space. The other solution is to use the POVM measurement. That means, that for a quantum system that does not have orthonormal bases, to predict the outcome of the measurement is still possible .

Definition 1.5.2 (POVM)**.** *A POVM is a set of Hermitian positive operators such that* $\sum_i |u_i\rangle\langle u_i| = I$ *as only requirement.* □

Example 1.5.3.3 Constructing a POVM
Let $|u_i\rangle$ be a set of non-normalized quantum states such that:

$$\begin{aligned} |u_0\rangle &= \frac{\sqrt{2}}{\sqrt{3}}|1\rangle \\ |u_1\rangle &= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{6}}|1\rangle \end{aligned} \tag{1.54}$$

The non-normalized quantum states (eq 1.54) and similarly to eq. 1.53 the probabilities of outcomes for these states are given by $|\langle u_i|\Phi\rangle|^2$. This means that when the outcome of our measurement generates an eigenvalue of 1, the state of the system is well detected. When the output is probabilistic (after measurement) the initial state of the system cannot be determined with certainty. With such initializations of states $|u_i\rangle$, it is possible to measure for more states that is possible in the orthonormal basis set. Thus let represent the states $|u_i\rangle$ as corresponding density matrices D_i of the observable:

$$
\begin{aligned}
D_0 &= |u_0\rangle\langle u_0| = \frac{2}{3}|1\rangle\langle 1| = \frac{2}{3}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \\
D_1 &= |u_1\rangle\langle u_1| = (\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{6}}|1\rangle)(\frac{1}{\sqrt{2}}\langle 0| - \frac{1}{\sqrt{6}}\langle 1|) = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2\sqrt{3}} \\ -\frac{1}{2\sqrt{3}} & \frac{1}{6} \end{pmatrix} \\
D_2 &= I - D_0 - D_1
\end{aligned}
\tag{1.55}
$$

If the POVM operators from eq. 1.55 are used to detect the desired states from eq. 1.54, it is easy to see that both states $|u_0\rangle$ and $|u_1\rangle$ results in observing the state being detected by D_0 and D_1 with equal probability independently from the initial state (similarly to example 1.5.3.2).

To solve this, one can construct such POVM operators that each observable D is orthogonal to u:

$$
\begin{aligned}
D_{0^+} &= \frac{2}{3}|0\rangle\langle 0| = \frac{2}{3}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\
D_{1^+} &= (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{6}}|1\rangle)(\frac{1}{\sqrt{2}}\langle 0| + \frac{1}{\sqrt{6}}\langle 1|) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2\sqrt{3}} \\ \frac{1}{2\sqrt{3}} & \frac{1}{6} \end{pmatrix} \\
D_2 &= I - D_0 - D_1
\end{aligned}
\tag{1.56}
$$

The observables D_{0^+} and D_{1^+} now allow to distinguish between states u_0 and u_1. For instance, when D_{0^+} is observed the initial state must have been u_1 ($D_{0^+}|u_0\rangle = \{\}$) and when the observable D_{1^+} was observed the prior state must have been u_0 ($D_{1^+}|u_1\rangle = \{\}$). This fact is summarized and verified by simple mathematical operations as shown in eq. 1.57

$$\begin{aligned} \langle u_0|D_{0^+}^\dagger D_{0^+}|u_0\rangle &= 0 \\ \langle u_1|D_{1^+}^\dagger D_{1^+}|u_1\rangle &= 0 \\ \text{and} & \\ \langle u_0|D_{1^+}^\dagger D_{1^+}|u_0\rangle &= 1 \\ \langle u_1|D_{0^+}^\dagger D_{0^+}|u_1\rangle &= 1 \end{aligned} \tag{1.57}$$

The D_2 represents the uncertainty of the missed measurement and thus each time the D_2 is observed, nothing cannot be said about the system. In other words, for two given non orthonormal states, one can construct a POVM operators such that for each desired state expressed by D_n $p(D_n) = 1, p(D_m) = 0, \forall m \neq n$.

Thus, well designed POVM operators allows to distinguish between non-orthonormal states for the price of obtaining no information at all about the system in some cases. It can be noticed that using the POVM operators one will not obtain any information about the state of the system before the measurement. However, POVM operators allow to determine the probabilities of outcome (quantum state) with certainty.

Chapter 2

Quantum Logic Synthesis and Principles of Quantum Circuits Design

2.1 Introduction

Logic minimization is a well known area of computer engineering and in this book various new research aspects related to search, automated synthesis and minimization of quantum circuits are discussed. In Quantum Logic Synthesis, the methods used are directly related to the representations that are being applied. For these representations different approaches are used while synthesizing FSM's, Logic Circuits, Behaviors or Quantum Cellular Automata. For instance within evolutionary approaches, to synthesize a FSM using evolutionary approach, the most prominent method includes the Genetic Programming [Koz92, Koz94] while the synthesis of boolean logic functions or circuits has mainly been done using the Genetic Algorithm. Algorithmic methods such as composition or spectral synthesis [SBM05a,SBM05b,Mil02,MMD06,PARK$^+$01,KPK02,GAJ06,FTR07,WGMD09, SZSS10, PLKK10] have been used as well.

In this chapter introduced are concepts of quantum logic synthesis with respect to quantum primitives and their costs. We describe a general methodology for the synthesis of quantum circuits. Various heuristics are studied on the functional level in order to demonstrate logic synthesis methods used for Machine Learning (Chapter **??**). The described concepts introduce the cost of quantum gates used in our synthesis methods and in particular we analyze the quantum inductive bias on the logic synthesis of circuits that can be used in the control of behavioral robots using inductive machine learning.

2.2 Previous research on automated synthesis of quantum circuits

The search for smaller, cheaper and ideally optimum circuits in quantum and reversible logic led to a set of gates and circuits commonly used as universal minimal primitives for logic synthesis [BBC+95, Per00, SD96, HSY+04]. There are several properties that are being searched for and some of them are: universality, low realization cost, technology specificity and good synthesis properties. In general, the goal is a sum of the mentioned sub-goals with a various degree of importance for every single one of these goals. However, depending on the complexity of the defined problem, it is also required to precisely specify the partial goals and explore them individually.

It was shown by [DiV95, DiV95, SD96, MML+98, Per00] that all gates (quantum circuits) with more than 1 qubit could be build using only one-qubit and selected two-qubit primitives. A big challenge is to build the basic possible gates such as Fredkin [SD96, LPG+04] or Toffoli having the smallest cost for a given technology. According to the description of quantum logic in Chapter 1, it is clear that logic synthesis of quantum circuits consists in finding compositions of primitive gates such that their resultant matrix is equal to the specification unitary matrix. This problem can be seen in an analogy to designing classical logic circuits from basic logic gates using a specification in form of a Karnaugh Map (KMap) [DM94]. As was shown in [LPG+04] the synthesis of quantum circuits is a non-monotonic process and consequently it is hard to use automated techniques to quantum circuit synthesis without relying on some heuristics. Also as can be implied from matrices representing gates or circuits, their dimensionality grows exponentially with the number of qubits. For example a circuit with 3 qubits will be represented by a matrix of 2^3 by 2^3 (64 elements) while a circuit with 5 qubits will have a matrix of size 2^5 by 2^5 (1024 elements). Each element of such a matrix is in general a complex number and consequently the calculation of the matrix may in the worst case demand also an exponential time. Moreover, in quantum logic synthesis all circuits can be composed in infinitely many ways using quantum gates and without adding more qubits. In other words, a circuit given by a Unitary transformation U, can be realized either from a minimum number of gates or can be realized in infinitely many circuits of various costs; the more component gates available as the input set, the more solutions to the synthesis are possible. Thus the problem of minimization in Quantum Logic Synthesis is not only a problem of exponentially expanding the solution space with the size of the circuit but also that of finding the minimal set of gates that would allow a potentially minimal solution.

Without any constraints, the synthesis problem described in the previous para-

graph, is NP; the process of synthesizing a circuit with k-quantum gates can be seen as the problem of subset-sum (knapsack) [GJ79, CLRS01] problem. To see this, it is enough to consider an initial finite-size set of quantum gates and the problem is to ask whether yes or not there is a circuit with k-gates implementing function f? This description is analogous to the Knapsack problem. In particular depending on technology, the challenge is to build any universal gate using only one-qubit and two-qubit primitives.

Most of known quantum circuits synthesis techniques are either for a small number of qubits only, for a small number of gates or for certain specific constrained logic families of functions (such as reversible or linear functions). The most common Quantum Logic Synthesis (QLS) approaches are used for the design of purely quantum permutative (reversible) logic circuits [MD03,LPG$^+$04,LP02,YHSP05,YSPH05, MDM05, SBM05a, SBM05b, MDM07, HSY$^+$06, WGMD09, PLKK10, **?**]. The synthesis of the reversible circuits can be further split into two main subcategories; one approach to the reversible logic design relies heavily on the usage of the ancilla bits [MD03,MDM05,WGMD09], the second approach designs reversible logic circuits only on the minimal number of qubits [MP02, LPG$^+$04, YHSP05, FTR07, **?**, LSKed]. The general strategy separating these two mainstreams of reversible logic synthesis is that a large number of ancilla qubits can potentially reduce the number of the required gates to synthesise a circuit at a price of the ancilla bits [MWD10].

A more general QLS for arbitrary quantum circuits was performed fro much smaller number of qubits [Yab00, Rub01, LPG$^+$03, LSKed]. This approach was in general more experimental up to now due to the fact that there is potentially an infinite number of quantum gates that can be used for the QLS. In these approaches a single algorithm - a genetic algorithm - was used to design or optimized a quantum circuit.

Thus despite some already reported results from the QLS approach there is no general method to synthesize larger than 2-qubit quantum circuits using quantum non-permutative primitives. Some of the methods are adapted from Reversible logic synthesis and have been used mainly for synthesis using the CNT set of gates (NOT, Feynman and Toffoli) or similar libraries not allowing to use the entire power offered by the quantum circuits and quantum logic. There exists also a small set of various new libraries of gates for quantum logic synthesis [BBC$^+$95, SD96, LP02, YSPH05, LPK10]. Among these approaches also exists methods using the so-called Multi-Controlled Toffoli (MCT) gates as the unique synthesis component gate [MMD03, MDM05, MDM07, WGMD09, PLKK10] where the function designed as a circuit is sollely based from the Toffoli gates. Closer to the quantum hardware implementation is for instance the approach proposed in [SBM05a] where the synthesis of the reversible gates is done using the so called-quantum multi-plexer. However there is no proof that any of them has the minimal

quantum realization cost with respect to all circuits that can be build for the given functional specification. Thus it is still an open issue to find out which set of gates will allow to generate a least costly circuit (in the number of gates and in the number of ancilla bits) for various technologies.

2.3 Quantum Gates and Quantum Logic Circuits

2.3.1 Single-qubit Quantum Gates

We are now concerned with matrix representation of operators. The first class of important quantum operators are the one-qubit operators realized in the quantum circuit as the one-qubit (quantum) gates. Some of their matrix representations can be seen in equation 2.1.

$$\begin{aligned} &a) \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad b) \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad c) \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ &d) \quad H = \tfrac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad e) \quad V = \tfrac{(1+i)}{2} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix} \quad f) \quad Phase = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \end{aligned} \tag{2.1}$$

Each matrix of an Operator has its inputs from the top (from left to right) and the outputs on the side (from top to bottom). Thus taking a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and an unitary operator H (eq. 2.1d) the result of computation is represented in equation 2.2.

$$H|\Psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{\alpha+\beta}{\sqrt{2}} \\ \frac{\alpha-\beta}{\sqrt{2}} \end{bmatrix} \tag{2.2}$$

To understand which particular quantum logic operation is executed for each output, the operation from eq 2.2 is broken down to each of the possible input states in eq. 2.3. The first equation from eq. 2.3 shows that when the input state is $|0\rangle$ it is transformed so that when it is then observed the outcome will be 50% of observing the state $|0\rangle$ and 50% of observing the state $|1\rangle$. It is similar for the input state $|1\rangle$, because $p(1) = \frac{1}{2}(\langle 0| + \langle 1|)M_1^\dagger M_1(|0\rangle + |1\rangle) = \frac{1}{2}$.

$$
\begin{aligned}
&|0\rangle \rightsquigarrow \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\\
&\text{and}\\
&|1\rangle \rightsquigarrow \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)
\end{aligned}
\tag{2.3}
$$

Observe that while the NOT gate has a unitary matrix that is a permutation matrix, some other gates like the Hadamard and the V gate have unitary matrices that are not permutative matrices. The Hadamard gate is very well known because it is used to create a superposition of states. An example of creating one qubit in a superposition is given in equation (2.3) where for each input either state $|0\rangle$ or $|1\rangle$ the output state $|0\rangle$ or $|1\rangle$ can be measured with a probability of $\frac{1}{2}$.

Observe that the Square-root-of-Not is a unitary transformation creating a complex superposed state (eq. 2.4).

$$
\begin{aligned}
&\sqrt{X} = V \quad \text{and} \quad \sqrt{X} = V^\dagger\\
&\text{where } V = \frac{1+i}{2}\begin{bmatrix}1 & -i\\ -i & 1\end{bmatrix}\\
&V^\dagger = \frac{1+i}{2}\begin{bmatrix}-i & 1\\ 1 & -i\end{bmatrix}
\end{aligned}
\tag{2.4}
$$

The V gate has two interesting properties $V \cdot V = V^\dagger \cdot V^\dagger = NOT = X$ and $V^\dagger \cdot V = V \cdot V^\dagger = I$ and as is shown later, this gate is used to construct the well-known cheapest universal quantum gates.

In this book we will be using the single-qubit gates that are commonly used in papers on quantum synthesis . They are : NOT (Pauli rotation X, denoted also in literature by σ_x), Hadamard, $\pi/8$, and S (eq. 2.5).

$$
S(\psi) = \begin{bmatrix}1 & 0\\ 0 & e^{i\psi}\end{bmatrix} \tag{2.5}
$$

We will also use Pauli rotations X, Y and Z or arbitrary angle rotations with respect to axes X, Y and Z of the Bloch sphere (as in Figure 1.6) and some their special cases for fixed angles which are multiples of 45° . We will use also two new gates; pseudo-Hadamard h and its adjoint pseudo-Hadamard gate h^{-1} (eq. 2.6), because they are used to build many quantum gates, both permutative (pseudo-binary) and general-purpose-quantum gates (called also truly quantum gates) that are most useful in synthesis [JM98, JHM98].

$$h = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \quad h^{-1} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{2.6}$$

Some additional gates are also listed in equation 2.7. In equation 2.7 symbols X, Y, and Z are the defined earlier Pauli spin matrices and $P(\psi)$, $X(\psi)$, $Y(\psi)$, and $Z(\psi)$ are the corresponding 2*2 matrices of arbitrary parameterized angle rotations by angle ψ. The rotations $X(\psi)$, $Y(\psi)$, and $Z(\psi)$ can be explained as rotations with respect to angles X, Y and Z, respectively, as illustrated on the Bloch sphere [NC00]. P is a phase rotation by $\psi/2$ [Lom03].

$$\begin{aligned} T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{j\pi}{4}} \end{bmatrix}, \qquad P(\phi) = e^{\frac{j\phi}{2}} I, \qquad X(\phi) = cos\frac{\phi}{2} I - jsin\frac{\phi}{2} X, \\ X(\phi) = cos\frac{\phi}{2} I - jsin\frac{\phi}{2} Y, \qquad Z(\phi) = cos\frac{\phi}{2} I - jsin\frac{\phi}{2} Z \end{aligned} \tag{2.7}$$

Let us now try to find, by matrix/vector multiplication, all possible states that can be created by applying all possible serial combinations of gates V and $V^\dagger$ to states $|0\rangle$, $|1\rangle$, and all states created from these basis states (Figure 2.1). A qubit $|0\rangle$, given to a "square root of NOT" gate (Figure 2.1a) gives a state denoted by $|V_0\rangle$. After measurement this state gives $|0\rangle$ and $|1\rangle$ with equal probabilities $\frac{1}{2}$. Similarly all other possible cases are calculated in Figure 2.1b - h. As we see, after obtaining states $|0\rangle$, $|1\rangle$, $|V_0\rangle$ and $|V_1\rangle$ the system is closed and no more states are generated. Therefore the subset of (complex, continuous) quantum space of states is restricted with these gates to a set of states that can be described by a four-valued algebra with values $\{|0\rangle, |1\rangle, |V_0\rangle, |V_1\rangle\}$.

2.3.2 Multi-qubit and Controlled Quantum Gates

$$U = \begin{array}{c} \\ 00 \leftarrow \\ 01 \leftarrow \\ 10 \leftarrow \\ 11 \leftarrow \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 01 & 10 & 11 \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \tag{2.8}$$

The second class of quantum gates includes the Controlled-U gates. Schematic representation of such gates can be seen in Figure 2.2. Gates in Figure 2.2a - Figure

(a) $|0\rangle$ — V — $|V_0\rangle$ $\Leftrightarrow$ $\frac{1}{2}\begin{bmatrix}1+j & 1-j\\ 1-j & 1+j\end{bmatrix}\times\begin{bmatrix}1\\0\end{bmatrix}=\frac{1}{2}\begin{bmatrix}1+j\\1-j\end{bmatrix}$

(b) $|V_0\rangle$ — V — $|1\rangle$ $\Leftrightarrow \frac{1}{2}\begin{bmatrix}1+j & 1-j\\ 1-j & 1+j\end{bmatrix}\times\begin{bmatrix}1+j\\1-j\end{bmatrix}=\frac{1}{2}\begin{bmatrix}(1+j^2)+(1-j^2)\\(1-j^2)+(1-j^2)\end{bmatrix}$

$$=\frac{1}{2}\begin{bmatrix}2(1+j^2)\\2(1-j^2)\end{bmatrix}=\frac{1}{2}\begin{bmatrix}(1-1)\\(1+1)\end{bmatrix}=\begin{bmatrix}0\\1\end{bmatrix}$$

(c) $|V_0\rangle \xrightarrow{V} |1\rangle$ (d) $|V_1\rangle \xrightarrow{V} |0\rangle$ (e) $|0\rangle \xrightarrow{V^\dagger} |V_1\rangle$

(f) $|V_0\rangle \xrightarrow{V^\dagger} |0\rangle$ (g) $|1\rangle \xrightarrow{V^\dagger} |V_0\rangle$ (h) $|V_1\rangle \xrightarrow{V^\dagger} |1\rangle$

Figure 2.1: Calculating all possible superposition states that can be obtained from basis states and using V and $V^\dagger$ gates

2.2c represent the general structures for single-qubit-controlled single-qubit, two-qubit-controlled and single-qubit and two-qubit-controlled and two-qubit quantum gates respectively. The reason for calling these gates *Controlled* is the fact that they are based on two operations: first there is one or more control bits and second there is a unitary transformation similar to the matrices from equation 2.1 that is controlled. For instance the Feynman gate is a Controlled NOT gate and has two input qubits a and b as can be seen in Figure 2.2 and shown with input and output minterms in 2.8 (minterm being a product term of given values for all input variables). Thus qubits controlling the gate are called the control qubits and the qubits on which the unitary transform is applied to are called the target qubits.

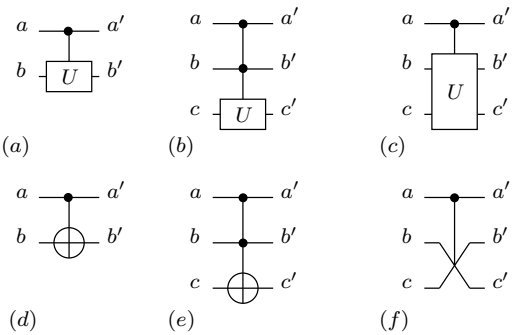


Figure 2.2: Schematic representation of Controlled-U gates: a) general structure of single-qubit controlled U gate (control qubit a, target qubit b), b) two-qubit controlled single-qubit operation, c) single-qubit controlled two-qubit target quantum gate, d) Feynman (CNOT), e) Toffoli (CCNOT), f) Fredkin. a, b, c are input qubits and a', b' and c' are respective outputs.

Figures 2.2d - Figure 2.2f represent special cases where the controlled unitary operator is Not, Not and Swap, respectively. The respective unitary matrices are in equations 2.8, 2.9a and 2.9b.

Equation 2.8 shows that if the input state is for instance $|00\rangle$ (from the top) the output is given by $U|00\rangle = p_{00}|00\rangle = 1 * |00\rangle$. Similarly for all other possible input /output combinations.

$$(2.9) \quad (a) \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} \end{bmatrix} \quad (b) \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

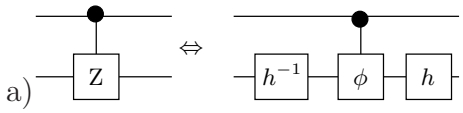
The *Controlled-U* gate means that while the controlled qubit a is equal to 0 the qubits on output of both wires are the same as they were before entering the gate (a' = a, b' = b). Now if qubit a equals to 1, the result is a' = a and b' = ¬b according to the matrix in equation (2.1.a). It can be easily verified that the CCNOT (Toffoli) gate is just a Feynman gate with one more control qubit and the Fredkin gate is a controlled swap as shown in Figure 2.2.

A closer look at equations (2.8 and 2.9) gives more explanation about what is described in eq. 2.8: CNOT, eq. 2.9a : Toffoli and eq. 2.9b : Fredkin gates. For instance, equation 2.8 shows that while the system is in states $|00\rangle$ and $|01\rangle$ the output of the circuit is a copy of the input. For the inputs $|10\rangle$ and $|11\rangle$ the second output is inverted and it can be seen that the right-lower corner of the matrix (in bold fonts) is the NOT gate.

The second type of multi-qubit gates are such gates that are not controlled-U gates. There is essentially only one type of such gates. This important gate is the SWAP gate and its derivatives. The SWAP gate, as its name indicates swaps two neighboring qubits. Matrix of a SWAP gate is shown in eq. 2.10

$$(2.10) \qquad SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

To summarize, quantum gates can be divided into two major groups: one-qubit gates and controlled-U gates. Most of gates are represented by permutation matrices and



b) $h \times \Phi \times h^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \times \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

c)

$\phi = 0$	I
$\phi = 90°$	V
$\phi = -90$	$V^\dagger$
$\phi = 180$	X

d)

$\phi = 0$	I
$\phi = 90°$	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = h$
$\phi = -90$	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = h^{-1}$
$\phi = 180$	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

Figure 2.3: (a) Controlled-Z gate realized with controlled-phi gate surrounded by pseudo-hadamard gates, (b) Calculation of unitary matrix for lower qubit of this gate, (c) Various gates realized by ϕ for angles 0°, 90°, −90° and 180° in X rotations.The ϕ gate realizes identity, Square-root-of-NOT, its adjoint and Inverter, (d) some gates realized by Y rotations.

the gates that cannot be represented by permutation matrices create a superposition of states. Unitary matrices are linear operators modifying complex amplitudes of the input state and thus they affect the probability of measurement of each basis state.

2.3.3 Constructing Quantum Circuits

A quantum gate operating in parallel with another quantum gate will increase the dimensions of the quantum logic system represented in matrix form. This is due to application of the Kronecker (tensor) product of matrices to the system. Kronecker Matrix Multiplication is responsible for the growth of qubit states such that N bits correspond to a superposition of r^N states, whereas in other digital systems, N bits correspond to a single state at a time. The number r denotes the base (radix) of logic, being 2 for the binary logic and 3 for the ternary logic. The Kronecker Product of two one-qubit gates is illustrated below:

A quantum gate in series with another quantum gate will retain the dimensions of the quantum logic system. The resultant matrix is calculated by multiplying the operator matrices in a reverse order. This is a standard multiplication operation on matrices.

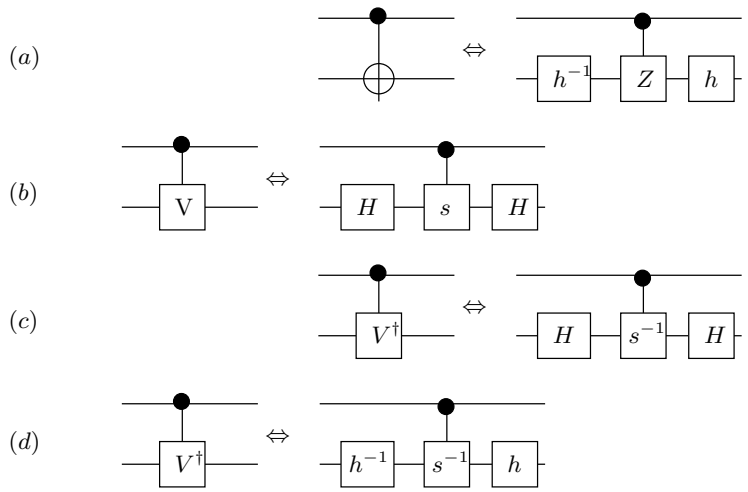


Figure 2.4: (a) CNOT realized with controlled-Z and pseudo-hadamard gates. Symbol h stands for pseudo-hadamard gate and symbol h-1 for inverse pseudo-hadamard gate. (b) CV realized with Controlled-S and Hadamard gates, (c) $CV^\dagger$ realized with controlled-S-1 and Hadamards, (d) $CV^\dagger$ realized with controlled-S-1 and pseudo-hadamards. Observe that this realization requires less pulses than its equivalent from Figure 2.6c

A quantum circuit can be easily analyzed. A parallel connection of gates corresponds to the Kronecker Product (the Tensor Product) of unitary matrices of respective gates. The serial connection of gates corresponds to the matrix multiplication (in reverse order) of the matrices of these gates. One can thus easily check that the equivalence transformations from Figure 2.4, 2.5a and 2.6b are correct. All verifications of quantum equivalence transformations can be done by multiplying and comparing respective unitary matrices.

Figure 2.3a presents the controlled general phase gate used together with a pseudo-Hadamard and its inverse gate. Figure 2.3b has the symbolic unitary matrix when the control signal is $|1\rangle$. By substituting various values of angles, $0°$, $90°$, $-90°$, $180°$ the unitary matrices are created which are next combined with the pseudo-Hadamard matrices, as in Figure 2.3a. This leads to the table from Figure 2.3c that demonstrates that by changing the angle the gate from Figure 2.3a can work as a 2-qubit identity, controlled-V, controlled-$V^\dagger$ and CNOT. Actually this gate can be used as a controlled root of various degrees. Figure 2.3d illustrates unitary matrices for various angles of Y. This figure demonstrates therefore the usefulness of Y and Z rotations to create gates. Unitary matrices for some useful 2-qubit gates are presented in Figure 2.7.

There are two methods of designing and drawing quantum/reversible permutative circuits.

(a)

$$Y(\phi) = cos\frac{\phi}{2} - isin\frac{\phi}{2} = cos\frac{\phi}{2}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - isin\frac{\phi}{2}\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} =$$
$$\begin{bmatrix}cos\frac{\phi}{2} & 0\\ 0 & cos\frac{\phi}{2}\end{bmatrix} - \begin{bmatrix}0 & -i^2sin\frac{\phi}{2}\\ i^2sin\frac{\phi}{2} & 0\end{bmatrix} = \begin{bmatrix}cos\frac{\phi}{2} & -sin\frac{\phi}{2}\\ sin\frac{\phi}{2} & cos\frac{\phi}{2}\end{bmatrix}$$

Figure 2.5: Example how to calculate unitary matrices of generalized rotations from general matrix formulas

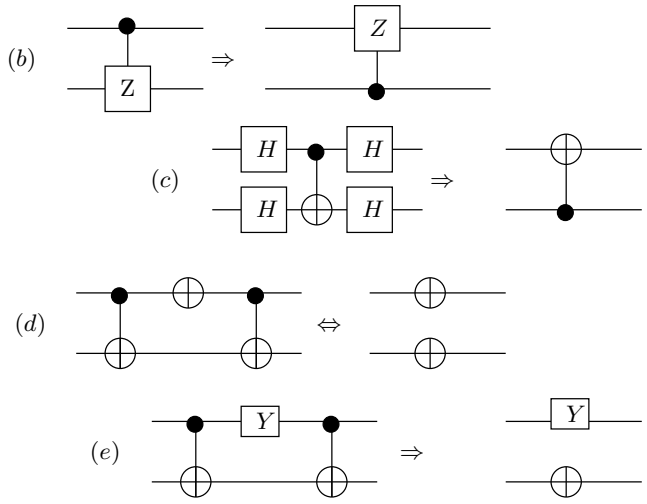
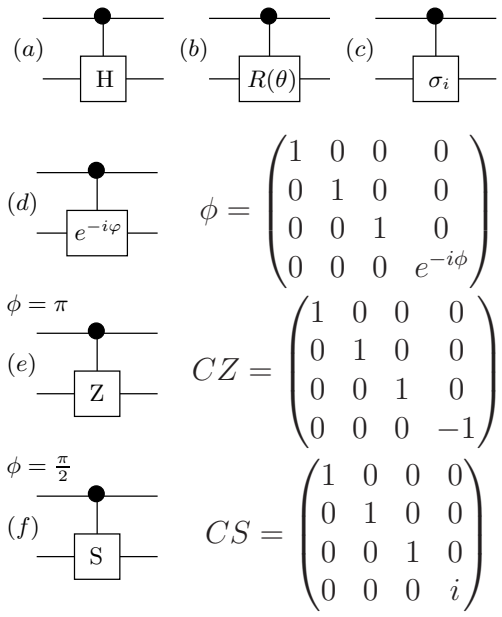


Figure 2.6: (b) Equivalent transformation of Z gate, (c) equivalent transformation of CNOT and Hadamard gates, (d) CNOT and NOT transformation, (e) CNOTs and Pauli Y transformation.

In the first method one draws a circuit from gates and connects these gates by standard wires. This method is similar to classical circuit design, but the used gates are reversible or quantum. The rules to design a reversible circuit using this approach are the following: (1) no loops allowed in the circuit and no loops internal to gates, (2) fan-out of every gate is one. These rules preserve the reversible characteristic of gates thus the resulting circuit is also completely reversible. When circuits are drawn, the gates are placed on a 2-dimensional space and the connections between them are routed. Every crossing of two wires in the schematics is replaced with the quantum Swap gate making the schematics planar, which means, no more two wires intersect in it. Also, it is in most cases needed to add ancilla bits initialized to constants. This method is not practical with respect to the required small width of quantum registers in modern quantum technologies. The schematics is thus rewrit-



$$\phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{pmatrix}$$

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

$$CS = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}$$

Figure 2.7: Controlled gates. (a) Controlled Hadamard gate, (b) Controlled Rotation with respect to angle θ. This symbol applies to any angle type, particularly X, Y and Z. Additional symbol is used to denote the angle type, (c) symbol of Pauli rotation where subscript $i = X, Y, Z$, (d) controlled phase and its unitary matrix, (e) Controlled Z and its unitary matrix, (f) controlled phase gate and its unitary matrix.

ten to a quantum array notation used in this book. It is relatively easy to transform a quantum array to its corresponding unitary matrix, as will be illustrated in the sequel. The approaches that use this first design method of reversible circuits are closer to those of the classical digital CAD where the design stages of logic synthesis and physical (geometrical) design are separated. They are intuitive and allow to create large quantum networks without resorting to their unitary matrices. Unfortunately they are not formal and thus not much is published on these design methods. Using this methodology can lead to circuit with very wide quantum registers.

The second design method for quantum circuits is to synthesize directly the quantum array of a circuit that was initially specified by a unitary matrix (or a set of functions for desired outputs). This method is executed without involving additional graph-based or equation-based representations. The synthesis can be conducted by one of two approaches:

- composing matrices of elementary gates in series or in parallel until the matrix of the entire circuit becomes the same as the specification matrix [LPG+03, LPMP02, MMD03, MDM05, Rub01, WGMD09, PLKK10],

- decomposing the specification matrix of the entire circuit to parallel and serial connections of unitary matrices until all matrices correspond to matrices of elementary gates directly realizable in a given technology [SBM05a, KP06, HSY$^+$04, HSY$^+$06, PLSK11]. A more mathematical method was used by Yang et al. [YHSP05, YSPW05] to analyze group theoretical properties of quantum unitary operators and deduce the set of universal quantum gates. In a similar matehmatical manner, Hung et al. [HSY$^+$06] used the reconstructibility analysis to desing quantum circuits.

The above methods were all exact in the sense that the circuit realizes the intended quantum operator in an exact way (no error). In another synthesis variant, called the approximate synthesis, it is not required that the circuit specification matrix and the matrix of composed gates are exactly the same. They can differ by small allowed values or/and differ in some matrix coordinates only [SBM05a, Luk09].

In Quantum Logic Synthesis some of the difficulties are the lack of general model for synthesis, heuristics are not well known and until recently there were no counterparts in quantum logic of such familiar notions of classical logic CAD as KMaps [1], prime implicants or reductions to covering/coloring combinatorial approaches. Therefore to explore Quantum Gates and QLS most authors turned to evolutionary algorithms as the fast prototyping methods for quantum arrays [WG98, Rub00, LP02, LPG$^+$03]. These approaches seem to be good for introductory investigations of the solution space and its properties, with the hope that by analyzing solutions the researchers will learn more about the search space and ultimately create more efficient algorithms based on the acquired knowledge.

2.4 Function Classification for Logic Synthesis

Before discussing various techniques in QLS, let us characterize the problem space that we are studying (Figure 2.8). On the top are the reversible/quantum synthesizable logic functions. It is assumed that these functions are either reversible by default or are made reversible (by adding ancilla bits). The completely specified logic functions represent a class of synthesis problems that is well known in electronics industry and various approaches have already been applied and explored in CAD tools. In this book the focus is mainly on the incompletely specified functions, as defined by Definition 2.4.3. The interest in these functions is mainly based on the facts that a) - incomplete specifications allow to search for novel circuit realizations of functions or automata, b) they can be synthesized for various learning biases (Chapter **??**) and c) they are used in inductive machine learning (Chapter **??**).

[1] KMaps were first used in quantum circuit synthesis in [LP07]

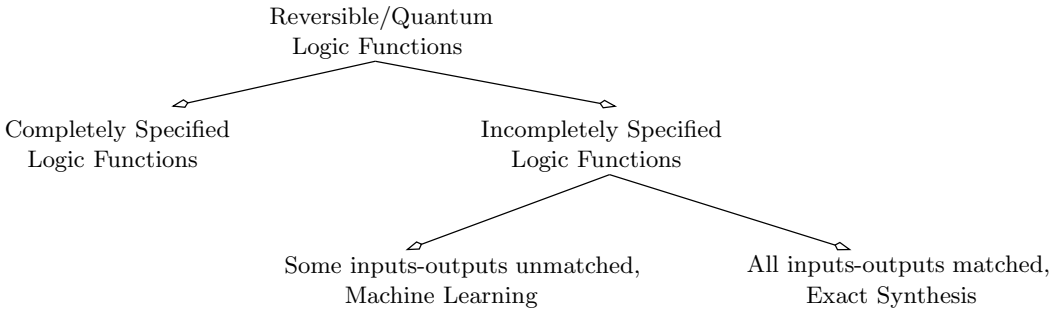


Figure 2.8: Schema representing the tree of relations between synthesis approaches for completely specified and incompletely specified quantum/reversible functions as used in this book.

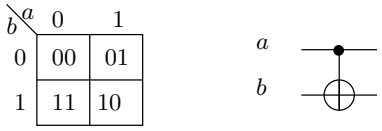


Figure 2.9: a) Complete Karnaugh map of the CNOT Gate from 2.9b

2.4.1 Quantum Karnaugh Maps and Function Definitions

As we know, every circuit can be realized using CV, $CV^{\dagger}$ and CNOT gates plus one-qubit gates such as V or Hadamard. When analyzing such circuits it is important to use the familiar Karnaugh maps (KMaps) in a new way. The user has to learn how to overlap groups in the map - this way new circuits and even new types of circuits have been invented in our PSU group [LPG+03, LP05a, LP07]. These maps allow to find patterns in Boolean, multiple-valued, multiple-valued-input-binary-output fuzzy and quantum functions. All synthesis methods in classical logic are based on patterns: the special classes of functions (such as the symmetrical or unate functions) have their specific patterns in KMaps. Therefore, being able to find new types of patterns and use these patterns in synthesis is very important when one wants to create new logic synthesis methods for new types of logic.

The Karnaugh map is derived from the truth table in a relatively simple process. The Karnaugh map of the CNOT gate is illustrated in Figure 2.9.

The arrangement of bits on the K-Map's rows and columns are in a sequence known as Gray code, where each value is only one bit change away from the preceding value. In this case, the order is 0,1. The sequence is 00,01,11,10, as it is for all two-bit Karnaugh maps (an example is in Figure 2.10), and so on. In a Karnaugh map, each possible bit combination of a and b is listed, with cells representing every single

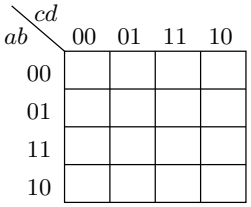


Figure 2.10: Skeleton of the 4 bit Karnaugh maps

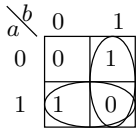


Figure 2.11: Groups in partial Karnaugh map of CNOT. Overlap of the groups represents 0 ($A \oplus A = 0$). Thus function is $\bar{a}b \oplus a\bar{b} = a \oplus b$.

possible input/output combination. Use the truth table to put the correct output in each cell. We will notice that the Karnaugh map for 2 inputs registers x and y as the outputs (Figure 2.9a). Now we make it y Karnaugh map (Figure 2.11) and synthesize from it (other output is trivial).

The representation used to specify quantum functions from the physical point of view is not the most appropriate when designing quantum circuits. In standard approach to Logic Design the function $f(a, b, c)$ is specified as a KMap or a LUT (Look-Up-Table or Truth Table) (Table. 2.1).

Observe that for the single output function in Table 2.1, the output is balanced; exactly half of the output values are 0 and half of them are 1.

Definition 2.4.1 (Reversible Functions)**.** *Let $f(.)$ be a completely defined function on $\{0,1\}^{\otimes n}$ such that for every input vector $i_j \subset I$ (I being the set of all possible*

Table 2.1: a). - K-map , b) - LUT

a).

ab \ c	0	1
00	0	1
01	1	0
11	1	0
01	0	1

b).

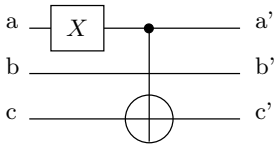
abc	f
000	0
001	1
010	1
011	0
100	0
101	1
110	1
111	0

input vectors defined over the binary vector of width n) it holds that

$$f(i_j) = o_j \text{ such that } \forall i_j, i_k \subset I; o_j \neq o_k \tag{2.11}$$

Eq. 2.11 represents a one-to-one mapping between input and output vectors defining a reversible function. The importance of reversible functions in quantum computation comes from the fact that every quantum function is reversible up to the measurement. The most obvious example of this phenomenon is the entanglement, where the unitary matrix representing the transformation U is a one-to-one mapping, but once measured this property is lost (Example 1.5.2.1). □

Example 2.4.1.1 Reversible Function
The circuit in Figure 2.12a represents the function shown in a K-map Figure 2.12b).



(a)

ab \ c	0	1
00	$\|101\rangle$	$\|100\rangle$
01	$\|111\rangle$	$\|110\rangle$
11	$\|010\rangle$	$\|011\rangle$
10	$\|000\rangle$	$\|001\rangle$

(b)

Figure 2.12: Example of representation of a quantum circuit using a quantum K-map. X is the Pauli X rotation or an Inverter.

The Table in Figure 2.12b is read as follows: for each logic input combination to the circuit represented as a minterm, the input state is transformed into another quantum state. For instance the quantum state $|100\rangle$ is transformed to state $|000\rangle$ which is denoted as $|100\rangle \rightarrow |000\rangle$.

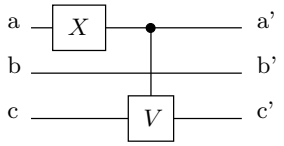
The representation from the above example is however not appropriate for quantum circuits, as Quantum Unitary operations can yield either deterministic, probabilistic or entangled states. In particular it is important to notice that the entangled states are different from the simple probabilistic states, and thus they require also a special notation.

Definition 2.4.2 (Quantum-Reversible Function)**.** *Let U be a Unitary transformation in a Complex Hilbert space H^d, such that for every input state $\psi_j \subset I$ (I being the set of all possible binary input vectors width n of the quantum register) it holds that*

$$U|\psi_j\rangle = |\psi_m\rangle, \text{ such that } \forall |\psi_j\rangle, |\psi_k\rangle \subset I; |\psi_m\rangle\langle\psi_n| = 0 \tag{2.12}$$

□

To be more precise, the K-map can be written for output states before the measurement. In such case, the complex coefficients can be written along the output states (Figure 2.13b).



(a)

ab \ c	0	1
00	$\|10\rangle\|G_0\rangle$	$\|10\rangle\|G_1\rangle$
01	$\|11\rangle\|G_0\rangle$	$\|11\rangle\|G_1\rangle$
11	$\|010\rangle$	$\|011\rangle$
10	$\|000\rangle$	$\|001\rangle$

(b)

Figure 2.13: Example of a quantum circuit (a) having a non-permutative unitary matrix (using a quantum K-map (b)). The V is the $\sqrt{X}$ gate. Symbols $|G_0\rangle$ and $|G_1\rangle$ are explained in text.

In this case all the quantum complex coefficients are visible. The G_0 and G_1 are two quantum states given by

$$|G_0\rangle = \frac{(0.5 + 0.5i)|0\rangle + (0.5 - 0.5i)|1\rangle}{2} \tag{2.13}$$

and

$$|G_1\rangle = \frac{(0.5 - 0.5i)|0\rangle + (0.5 + 0.5i)|1\rangle}{2} \tag{2.14}$$

respectively. These quantum states are collapsed to observable binary states once measured. It is also possible to set up POV measurement allowing to introduce new output values, otherwise not available in classical computing.

Beside KMaps and LUT's a common representation in classical logic is the BDD (Binary Decision Diagram). A BDD is a canonical representation of a given function that is obtained after the minimization of a Decision Tree allowing for more compact representation of data and functions. In quantum logic synthesis, there are already known decision diagram representations. The most common is Quantum Multi-valued Decision Diagram (QMDD) [MM06], however, because QMDD is optimized for structured quantum logic circuits we do not use them because in general our algorithm searches the space of all quantum circuits.

Classical, Reversible and Quantum functions can also be specified only for a subset of input values. In that case they are called Incompletely specified functions. Such specifications of functions are very useful in machine learning, where only a subset of input-output pairs of values is known. This is due to the fact that learning tasks

in Machine Learning are dealing with real-world problems that are in general not completely known or understood.

Definition 2.4.3 (Incompletely Specified Function)**.** *An incompletely specified function is a set K of input-output pairs $(|\psi_j\rangle, |\psi_l\rangle)$ such that $|K| < 2^{|N|}$ for any boolean function. For instance, a 3×1 incompletely specified function is shown in Table 2.2.*

Table 2.2: K-map of an incompletely specified 3×1 reversible quantum function before measurement

c / ab	0	1
00	$\|0\rangle$	$\|1\rangle$
01	$-$	$\|1\rangle$
11	$-$	$\|1\rangle$
01	$\|0\rangle$	$-$

□

Definition 2.4.4. *Cares and Don't Cares An Boolean incompletely function F is a mapping $F : B^n \rightarrow B^*$ where $B^* \in \{0, 1, -\}$. The values 0 and 1 are refered to as cares and represents a well defined function output. The $-$ is refered to as don't care and represents the fact that such output is not defined.*

Definition 2.4.5 (Reversible Function Prototype)**.** *An incompletely specified function is a reversible-function prototype (specified by K) if and only if there exists an Unitary transformation in a Complex Hilbert space H^d, $U(.)$ such that for every input state $\psi_j \subset I$ (I being the set of all possible binary input vectors width n of the quantum register)*

$$U|\psi_j\rangle = |\psi_m\rangle, \forall |\psi_j\rangle \subset K \subset I; |\psi_m\rangle = |\psi_l\rangle \tag{2.15}$$

This means that an incompletely specified function is a set of such input-output states (logic values) that must be realizable by a reversible logic function (a permutative unitary matrix). Thus, such incompletely specified function is realizable as a reversible function. □

To represent incompletely specified functions the KMaps or LUTs can be used. However, as it will be seen later, there are various ways how to understand and represent the unknown values that are also called don't cares. A common way of representing such unknown values is shown in Table 2.2, where the symbol '-' represents the don't care value.

Another method of representation is the shortened form. For instance the function defined in Table 2.2 can also be represented as a vector of output values in the natural

order of the input values combinations(minterms): $f = [f(0), f(1), \ldots, f(7)] = [0, 1, -, 1, 0, -, -, 1]$. To guarantee reversibility one can also use the output values of the whole circuit, thus a three-bit circuit will have outputs in the range $[0, 7]$. The above vector representation can be transformed to $f = [0, 1, ---, 5, 6, ---, --- , 3]$ and a possible result will have to contain exactly one of the possible output values. For instance $f = [0, 1, ---, 5, 6, ---, ---, 3] \Rightarrow [0, 1, 4, 5, 6, 7, 2, 3]$.

Finally, the don't cares can specify a whole minterm or only a part of it. For instance, a 3×3 incompletely specified reversible function shown in Table 2.3 shows an example of don't cares present in some output states but only on some selected qubits.

Table 2.3: K-map of an incompletely specified 3×3 reversible quantum function before measurement with don't cares within single minterms

c / ab	0	1
00	$\|010\rangle$	$\|001\rangle$
01	$\|-10\rangle$	$\|1-0\rangle$
11	$\|-0-\rangle$	$\|1--\rangle$
01	$\|--0\rangle$	$\|--1\rangle$

It will be shown later in this book that it is possible to design quantum-reversible circuits that not only satisfy the above criteria, but also have probabilistic outputs (on certain or all qubits) along with the deterministic ones. This is done by either specifying the output probabilities of output states or by designing particular measurement operators detecting the specific quantum states.

For instance. Let the output be able to take symbolic quantum state values such as $\{0, 1, V_0, V_1\}$. In order to distinguish by measurement between states V_0 (eq. 2.16) and V_1 (eq. 2.17), one would use a set of basis states to create projective measurement operators. These basis states can be mixed with measurement operators for states $|0\rangle$ and $|1\rangle$ to create POV measurement:

$$|V_0\rangle = \frac{1+i}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} * \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1+i}{2}\begin{bmatrix} 1 \\ -i \end{bmatrix} \tag{2.16}$$

and

$$D_0 = |V_0\rangle\langle V_0| = \frac{1+i}{2}\begin{bmatrix} 1 \\ -i \end{bmatrix} * \frac{1-i}{2}\begin{bmatrix} -i & 1 \end{bmatrix} = \frac{1}{2}\begin{pmatrix} i & 1 \\ 1 & -i \end{pmatrix} \tag{2.17}$$

$$|V_1\rangle = \frac{1+i}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} * \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1+i}{2}\begin{bmatrix} -i \\ 1 \end{bmatrix} \tag{2.18}$$

and

$$D_1 = |V_1\rangle\langle V_1| = \frac{1+i}{2}\begin{bmatrix}-i\\1\end{bmatrix} * \frac{1-i}{2}\begin{bmatrix}1 & -i\end{bmatrix} = \frac{1}{2}\begin{pmatrix}-i & 1\\1 & i\end{pmatrix} \tag{2.19}$$

. It can be easily verified that a qubit in state $|V_0\rangle$ is detected with the result of measurement D_1 and the quantum state $|V_1\rangle$ is detected as the result of measurement D_0 (see definition 1.5.2). These operations are explained graphically in Figure 2.14.

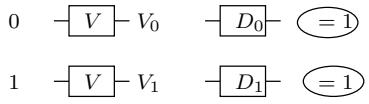


Figure 2.14: Schematic representation of detecting the quantum states V_0 and V_1 using the measurement operator specified by the density matrices D_0 and D_1.

Thus, a quantum reversible function can be generated such that the outputs include both binary basis states as well as quantum states such as those detected by D_0 and D_1. Observe, that this approach is different than measuring quantum states and obtaining a probability distributions of orthonormal states $|0\rangle$ and $|1\rangle$. Moreover, it is a natural extension of the presented ideas that other states such as those based on measurement in circuits that use gates V, $\sqrt{V}$, $\sqrt[4]{V}$, etc can be used to allocate the don't cares in incompletely specified reversible functions.

2.4.2 Circuit Identities and Optimizing Transformations

The reduction of quantum circuits uses, among others the well-known rule [NC00]: $[A, B] = AB - BA$, $AB - BA = 0 \rightarrow AB = BA$.

This reduction rule is illustrated in the quantum circuit from Figure 2.15 which means, that one can shift left or right pulses or gates for which the above rule holds.

The reduction algorithm uses the following commutation rules(Equations 2.20 - 2.23):

$$[R_{i\alpha}, R_{i\alpha'}] = 0 \text{ for } i \neq j \tag{2.20}$$

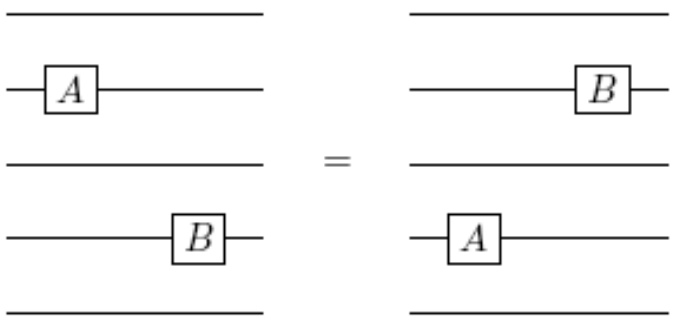


Figure 2.15: Graphical illustration of the rule $[A, B] = 0$

$$
\begin{aligned}
R_{ix}(\pm\pi)R_{iy}(\psi) &= R_{iy}(-\psi)R_{ix}(\pm\pi) \\
R_{ix}(\psi)R_{iy}(\pm\pi) &= R_{iy}(\pm\pi)R_{ix}(-\psi) \\
R_{ix}(\pm\frac{\pi}{2})R_{iy}(\psi) &= R_{iz}(\pm\psi)R_{ix}(\pm\frac{\pi}{2}) \\
R_{ix}(\pm\frac{\pi}{2})R_{iz}(\psi) &= R_{iy}(\pm\psi)R_{ix}(\pm\frac{\pi}{2}) \\
R_{ix}(\psi)R_{iy}(\pm\frac{\pi}{2}) &= R_{iz}(\pm\frac{\pi}{2})R_{ix}(\pm\psi) \\
R_{ix}(\psi)R_{iz}(\pm\frac{\pi}{2}) &= R_{iz}(\pm\frac{\pi}{2})R_{iy}(\pm\psi)
\end{aligned}
\tag{2.21}
$$

and the relations generated by the cyclic permutation of x y z.

$$[J_{ij}, J_{i'j'}] = 0 \tag{2.22}$$

$$[J_{ij}, R_{i'z}] = 0 \tag{2.23}$$

Graphically, these rules are represented as in Figure 2.16. If necessary, more rules can be added to the program, and/or can be made usable only in one direction (only from left to right or from right to left).

2.4.2.1 Realization of Single Qubit Gates

The most frequently used single-qubit gates in quantum algorithms are the NOT (N) (also known as Pauli-X, or X [NC97]), Hadamard(H), and phase(P) (also known as S [NC00]) gates. These gates are the special cases of the single-qubit rotation operations and are implemented by the rotation pulses as shown in Figure 2.17.

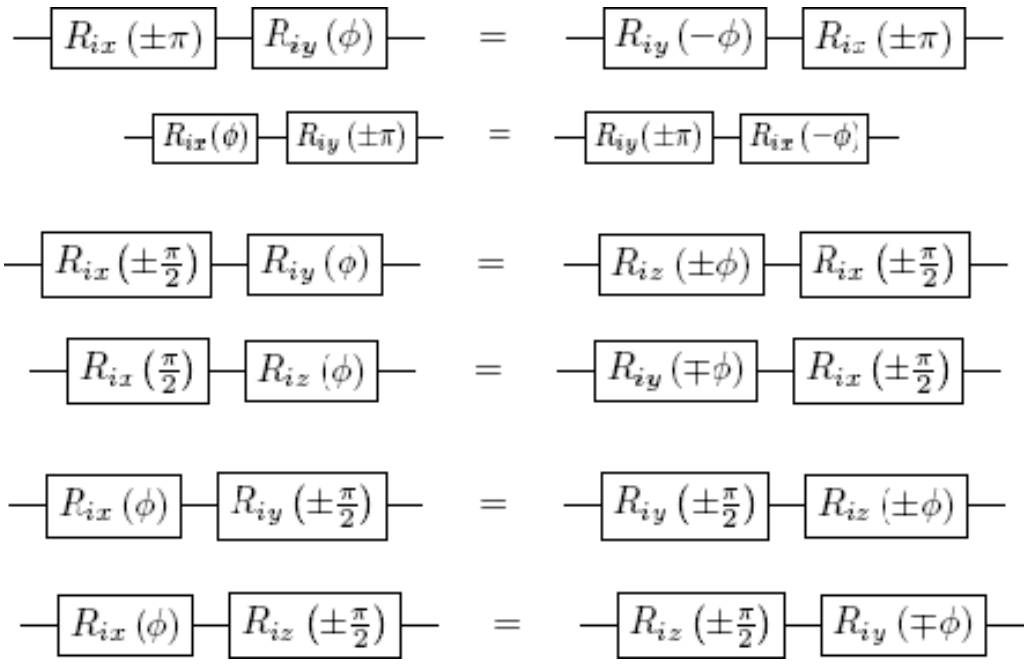


Figure 2.16: Graphical illustration of some commutation rules for quantum algebra that are used in the tree search-based pulse-level circuit minimization algorithm [IKY02, Lom03, CM04, LKBP06]

$$N = iR_x(\pi) = i\begin{bmatrix} cos\left(\frac{\pi}{2}\right) & -isin\left(\frac{\pi}{2}\right) \\ -isin\left(\frac{\pi}{2}\right) & cos\left(\frac{\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad = \text{—}\oplus\text{—}$$

$$H = iR_y\left(\frac{\pi}{2}\right)R_z(\pi) = i\begin{bmatrix} cos\left(\frac{\pi}{4}\right) & -isin\left(\frac{\pi}{4}\right) \\ isin\left(\frac{\pi}{4}\right) & cos\left(\frac{\pi}{4}\right) \end{bmatrix}\begin{bmatrix} e^{-i\frac{\pi}{2}} & 0 \\ 0 & e^{-i\frac{\pi}{2}} \end{bmatrix} = \left(\frac{1}{\sqrt{2}}\right)\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad = \text{—}\boxed{H}\text{—}$$

$$S = e^{i\frac{\phi}{2}}R_z(\phi) = e^{i\frac{\phi}{2}}\begin{bmatrix} e^{-i\frac{\pi}{2}} & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \quad = \text{—}\boxed{S}\text{—}$$

Figure 2.17: (a) Calculation of matrix for Pauli X rotation, (b) calculation of matrix for Hadamard gate, (c) Calculation of matrix for S gate.

Therefore, the costs of Gates N and P are said to be 1, and that of H is 2, as in our model of a quantum circuit the quantum cost is the number of elementary poulses (gates) (see Figure 2.18). It is worthwhile to note that gates with the same number of input qubits can have and usually have very different costs in practice. The pulse sequence of a gate is not unique in general.

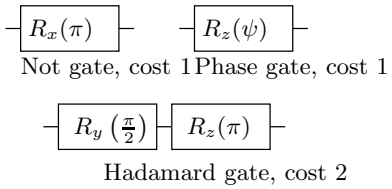


Figure 2.18: Quantum gates realized on the pulse level, they are decomposed to elementary rotations with respect to axes x, y and z.

It is also worthwhile to note the fact that the N, H and P gates are implemented up to overall phase. We illustrate an example of this fact for the N gate below in Figure 2.19. Let us denote a NOT gate such that it is correct to overall phase, by doing this we have the equations from Figure 2.19.

$$N = R_x(\pi) = \begin{bmatrix} cos\left(\frac{\pi}{2}\right) & -isin\left(\frac{\pi}{2}\right) \\ -isin\left(\frac{\pi}{2}\right) & cos\left(\frac{\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} 0 & -i \\ -i & 0 \end{bmatrix} = -i \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figure 2.19: Calculation of unitary matrix for inverter. The value of $-i = e^{-i\frac{\pi}{2}}$ is the phase that is lost in every quantum measurement.

The concepts of rotations and phase can be illustrated using the Bloch sphere, [NC00].

Before continuing in the multi-qubit and multi-gate representation of the quantum circuit it is important to observe that the circuits represented in this book use the forward order. In general, a quantum pqrticle evolving in time entails the fact that when it exists a unitary operation it will be in the state that was generated as the last one. For instance a quantum sate $|0\rangle$ going through a series of single qubit operations A, B and C will exit the sysem in the state directly generated by the C operator. this however means that when one does the mathematical computation the matrices must be multiplied in the reverse order, notably as $C * B * A$. This order is refered to as the reverse-order. Thus when analyzing the circuits in the next section the reader should take into account that the circuits are in the forward order and thus the matrices must be multiplied form the last gate in the circuit to the first one.

2.4.2.2 Realization of Two-Qubit Gates

The most frequently used two-qubit gates are the CNOT and SWAP gates. A possible pulse sequences for the CNOT gate is given in Equation 2.24. It represents a pulse sequence for CNOT gate (accurate to phase, where i is the target bit).

$$NOT_{ij} = R_{iz}\left(\frac{\pi}{2}\right)R_{jz}\left(\frac{\pi}{2}\right)R_{jy}\left(\frac{\pi}{2}\right)J_{ij}\left(-\frac{\pi}{2}\right)R_{jy}\left(-\frac{\pi}{2}\right) \tag{2.24}$$

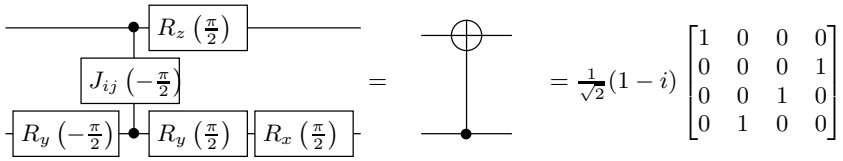


Figure 2.20: Representation of the CNOT Gate with EXOR up.

Most equations were verified by us using Matlab and simulation results are presented for some examples to encourage the reader to do the same when he will be designing quantum circuits and will need a verification.

Matlab simulation of Figure 2.20 is shown in eq. 2.25

(2.25)

$$CNOT = \begin{matrix}0.7071-0.7071i & 0 & 0 & 0\\ 0 & 0 & 0 & 0.7071-0.7071i\\ 0 & 0 & 0.7071-0.7071i & 0\\ 0 & 0.7071-0.7071i & 0 & 0\end{matrix}$$

The CNOT from Figure 2.21 is decomposed to pulses in eq. 2.26.

(2.26)

$$CNOT = R_{1y}\left(\frac{\pi}{2}\right)R_{2z}\left(\frac{-\pi}{2}\right)R_{1z}\left(\frac{-\pi}{2}\right)J_{12}\left(\frac{\pi}{2}\right)R_{1y}\left(\frac{-\pi}{2}\right) = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$$

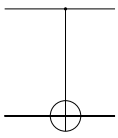


Figure 2.21: CNOT gate with EXOR down.

Step by step Matlab simulation of Figure 2.21 is shown in eq 2.27 to 2.32

(2.27)
$$R1 = \begin{pmatrix} 0.7071 - 0.7071i & 0 & 0 & 0 \\ 0 & 0.7071 - 0.7071i & 0 & 0 \\ 0 & 0 & 0.7071 + 0.7071i & 0 \\ 0 & 0 & 0 & 0.7071 + 0.7071i \end{pmatrix}$$

(2.28)
$$R2 = \begin{pmatrix} 0.7071 & 0 - 0.7071i & 0 & 0 \\ 0 - 0.7071i & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & 0 - 0.7071i \\ 0 & 0 & 0 - 0.7071i & 0.7071 \end{pmatrix}$$

(2.29)
$$R3 = \begin{pmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0 & 0.7071 & 0.7071 \end{pmatrix}$$

(2.30)
$$R4 = \begin{pmatrix} 0.7071 + 0.7071i & 0 & 0 & 0 \\ 0 & 0.7071 - 0.7071i & 0 & 0 \\ 0 & 0 & 0.7071 - 0.7071i & 0 \\ 0 & 0 & 0 & 0.7071 + 0.7071i \end{pmatrix}$$

(2.31)
$$R5 = \begin{pmatrix} 0.7071 & 0.7071 & 0 & 0 \\ -0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \\ 0 & 0 & -0.7071 & 0.7071 \end{pmatrix}$$

(2.32)
$$CNOT = \begin{pmatrix} 0.7071 - 0.7071i & 0 & 0 & 0 \\ 0 & 0.7071 - 0.7071i & 0 & 0 \\ 0 & 0 & 0 & 0.7071 - 0.7071i \\ 0 & 0 & 0.7071 - 0.7071i & 0 \end{pmatrix}$$

Simulation (eq 2.27 to 2.32) shows R1, R2, R3, R4 and R5 which are the Pauli Matrices from Figure 2.24 and CNOT results from the Equation 2.25

In Figure 2.21 the upper qubit is the control and lower qubit is target respectively. As shown by eq. 2.26, the cost of a CNOT gate is 5 pulses.

Another frequently used controlled gate is the controlled-V where V2 is equivalent to a NOT gate. The cost of this gate is also 5 because it can be implemented by Equation 2.33 (Pulse sequence for Controlled V gate (accurate to phase, where the target qubit is on the bottom fo the circuit)). The circuit corresponding to the equation 2.33 is shown in Figure 2.22.

$$CV = R_{2y}\left(\frac{\pi}{2}\right) R_{1z}\left(\frac{\pi}{4}\right) R_{2z}\left(\frac{\pi}{4}\right) J_{12}\left(\frac{-\pi}{4}\right) R_{2y}\left(\frac{-\pi}{2}\right) \tag{2.33}$$

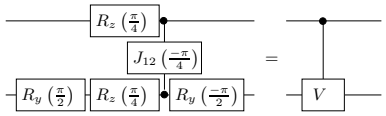


Figure 2.22: Controlled-V gate realized with 5 pulses.

Once the pulse sequences of the CNOT, controlled-V, and single-qubit gates are known, the pulse sequence for the other multi-qubit gates can be obtained if the gate is decomposed to a series of these basic gates.

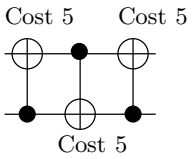


Figure 2.23: SWAP Gate comprised of 3 CNOT gates. The cost of the SWAP should be then 35 = 15 but it is lower thanks to local optimizations based on quantum algebra.

The SWAP gate is decomposed of three CNOT gates as shown in Figure 2.23. The pulse sequence of the SWAP gate obtained by replacing each CNOT gate (EXOR up) and EXOR down CNOT with sequence from Figure 2.21 is given in Equation 2.34. It has cost 15.

$$\begin{aligned} CV =& R_{2y}\left(\frac{\pi}{2}\right) R_{1z}\left(\frac{-\pi}{2}\right) R_{2z}\left(\frac{-\pi}{2}\right) J_{12}\left(\frac{\pi}{2}\right) R_{2y}\left(\frac{-\pi}{2}\right) \\ &\times R_{1y}\left(\frac{\pi}{2}\right) R_{2z}\left(\frac{-\pi}{2}\right) R_{1z}\left(\frac{-\pi}{2}\right) J_{12}\left(\frac{\pi}{2}\right) R_{1y}\left(\frac{-\pi}{2}\right) \\ &\times R_{2y}\left(\frac{\pi}{2}\right) R_{1z}\left(\frac{-\pi}{2}\right) R_{2z}\left(\frac{-\pi}{2}\right) J_{12}\left(\frac{\pi}{2}\right) R_{2y}\left(\frac{-\pi}{2}\right) \end{aligned} \tag{2.34}$$

Using the algorithm from [LKBP06] it can be shown that Equation 2.34 can be reduced to Equation 2.35, and from Equation 2.35, the cost of the SWAP gate is 11. The circuit corresponding to Equation 2.35 is shown in Figure 2.24.

$$
\begin{aligned}
CV =& R_{2y}\left(\frac{\pi}{2}\right) R_{2z}\left(\frac{-3\pi}{2}\right) R_{1z}\left(\frac{-3\pi}{2}\right) J_{12}\left(\frac{\pi}{2}\right) \\
&\times R_{2y}\left(\frac{\pi}{2}\right) R_{1y}\left(\frac{-\pi}{2}\right) J_{12}\left(\frac{\pi}{2}\right) R_{1x}\left(\frac{\pi}{2}\right) \\
&\times R_{2x}\left(\frac{-\pi}{2}\right) J_{12}\left(\frac{\pi}{2}\right) R_{2y}\left(\frac{-\pi}{2}\right)
\end{aligned} \tag{2.35}
$$

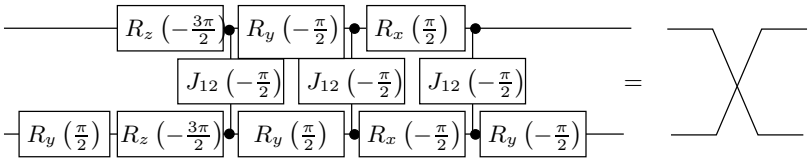


Figure 2.24: Swap Gate with 11 Pulses.

$$
R_x^1(\phi) = \begin{bmatrix} cos\frac{\phi}{2} & 0 & -isin\frac{\phi}{2} & 0 \\ 0 & cos\frac{\phi}{2} & 0 & -isin\frac{\phi}{2} \\ -isin\frac{\phi}{2} & 0 & cos\frac{\phi}{2} & 0 \\ 0 & -isin\frac{\phi}{2} & 0 & cos\frac{\phi}{2} \end{bmatrix} \quad R_x^2(\phi) = \begin{bmatrix} cos\frac{\phi}{2} & -isin\frac{\phi}{2} & 0 & 0 \\ -isin\frac{\phi}{2} & cos\frac{\phi}{2} & 0 & 0 \\ 0 & 0 & cos\frac{\phi}{2} & -isin\frac{\phi}{2} \\ 0 & 0 & -isin\frac{\phi}{2} & cos\frac{\phi}{2} \end{bmatrix}
$$

$$
R_y^1(\phi) = \begin{bmatrix} cos\frac{\phi}{2} & 0 & -sin\frac{\phi}{2} & 0 \\ 0 & cos\frac{\phi}{2} & 0 & -sin\frac{\phi}{2} \\ -sin\frac{\phi}{2} & 0 & cos\frac{\phi}{2} & 0 \\ 0 & -sin\frac{\phi}{2} & 0 & cos\frac{\phi}{2} \end{bmatrix} \quad R_y^2(\phi) = \begin{bmatrix} cos\frac{\phi}{2} & -sin\frac{\phi}{2} & 0 & 0 \\ -sin\frac{\phi}{2} & cos\frac{\phi}{2} & 0 & 0 \\ 0 & 0 & cos\frac{\phi}{2} & -sin\frac{\phi}{2} \\ 0 & 0 & -sin\frac{\phi}{2} & cos\frac{\phi}{2} \end{bmatrix}
$$

$$
R_z^1(\phi) = e^{-i\frac{\phi}{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\phi} & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix} \quad R_z^2(\phi) = e^{-i\frac{\phi}{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\phi} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix}
$$

$$
J_{12}(\phi) = e^{-i\frac{\phi}{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\phi} & 0 & 0 \\ 0 & 0 & e^{i\phi} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Figure 2.25: Two-Qubit Rotation Operations.

The Rotations matrices for two-qubit gates are given in Figure 2.25. They can be easily used to verify some of the calculations from this chapter.

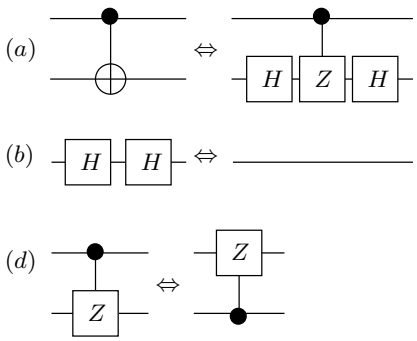


Figure 2.26: (a) The Controlled-NOT gate realised by controlled-Z gate surrounded by Hadamard gates, (b) two serially connected Hadamard gate are together equal to a quantum wire and (c) for controlled Z we can interchange the control qubit and the target qubit in the control-Z gate.

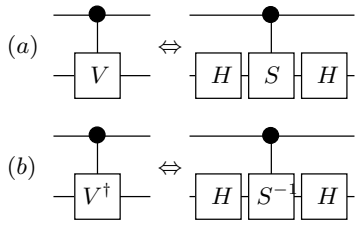


Figure 2.27: Construction of CV and $CV^\dagger$ from Hadamard gate , Phase gate(S) and its inverse(S-1).

The two-qubit gates can often be realized as a combination of or as a result of transformation of a group of other quantum gates. Examples of such transfomartions are shown in Figures 2.26, 2.27 and 2.28. The transformations from Figure 2.26 shows examples of usage the Controlled-Z and the Hadamard gates, the Figure 2.27 illustrates transformations and equivalences using the Hadamard, CNOT, Controlled-V and Controlled-S gates and the Figure 2.28 shows transformations for combinations of the Hadamard gate with Pauli-X, Pauli-Y and Pauli-Z rotations as well as with the CNOT gate.

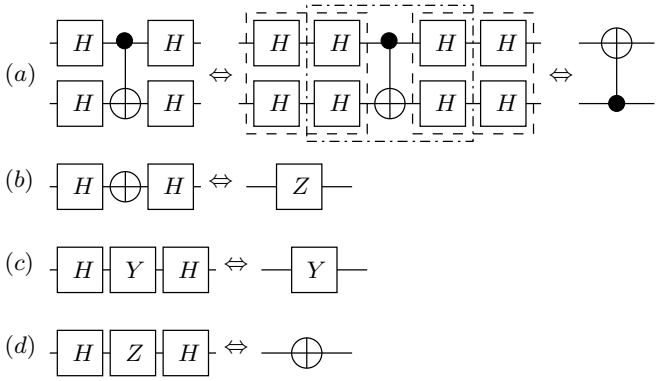


Figure 2.28: (a) Example of transformation for Feynman gate surrounded by Hadamard gates, (b)Hadamard gate used as serial connection creates Z gate, (c)Y gate surrounded by Hadamard creates Y gate, (d) Z gate surrounded by Hadamard gates creates NOT gate. These rules can be used to prove the correctness of the Grover Algorithm.

2.4.2.3 Realization of Three-Qubit Gates

The most frequently used three-qubit gates are Toffoli and Fredkin gates, the Miller gate [Mil02] and Peres gate [Per85] are also used. The circuit diagrams of these four gates are shown in Figure 2.29. The Peres gate is the cheapest gate found among those familiar in the universal set of reversible logic gates. It is just like a Toffoli gate but without the last CNOT gate, as shown in Figure 2.29(a).

The pulse sequence of the Toffoli gate reduced from the circuit in Figure 2.29b is composed of 15 pulses and contains 5 interaction terms. However, the equivalent sequence of this gate analyzed by the geometric algebra method presented in [CFH97] is composed of 13 pulses and contains 6 interaction terms. The sequence we listed in Table 2.4 Toffoli gate is the one with the lower cost. This case indicates that there is at least one quantum circuit for the Toffoli gate more efficient than shown in Figure 2.29b, a possibility also exists that the sequences listed in the table can be reduced further. Although the cost of the Toffoli gate given in Table 2.4

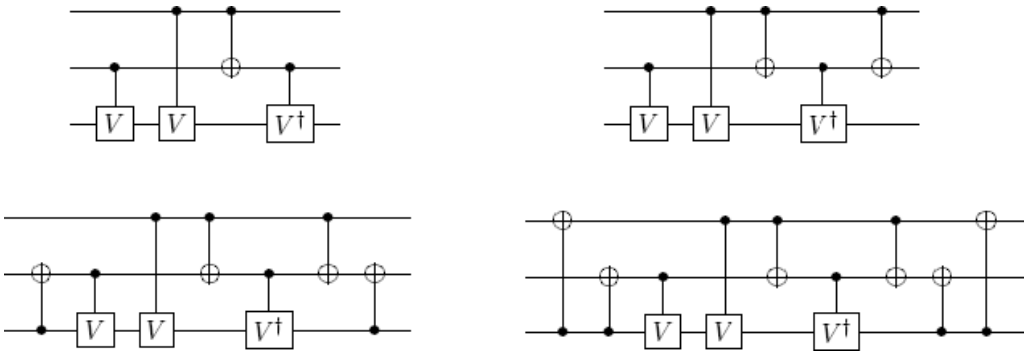


Figure 2.29: (a)The Peres Gate, (b) The Toffoli Gate, (c)The Fredkin Gate, (d) The Miller Gate.

is lower than the gate shown in Figure 2.29b, the gate from Figure 2.29b is practically cheaper than using the method explained in [LKBP06]. It is also possible that equivalent sequences can have a different number of interaction terms because $R_{iz}(\pi)R_{jz}(\pi)J_{ij}(\pi)$ is equal to the identity operation. The minimized Peres gate on the level of pulses is shown in Figure 2.30.

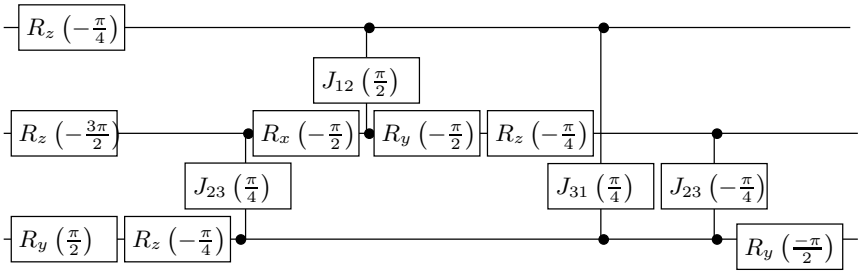


Figure 2.30: Peres Gate with 12 pulses.

The circuit diagram for the "pulse-level" realization of 3×3 Toffoli gate is shown in Figure 2.31. This is perhaps the exact minimum pulse-level realization. This fact has been confirmed by our exhaustive search software. If such search would be completed the cheapest universal gate for quantum computing (most likely Peres gate) would be proved. An interesting future project is also to find the cheapest realization of the fundamental Toffoli gate.

The circuit for the minimized "pulse-level" Fredkin gate is given in Fig. 2.32 and the circuit for the minimized Miller gate is given in Figure 2.33.

Example 2.4.2.1
To explain the fundament of our exhaustive search we can analyze and visualize the Miller gate's pulse level optimization. This is graphically represented on Figures 2.34 through Figure 2.36.

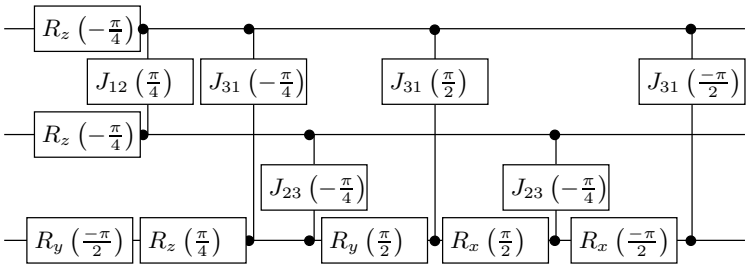


Figure 2.31: The Toffoli gates with 13 pulses.

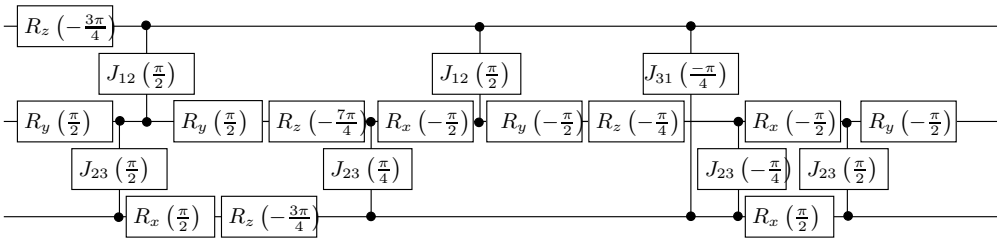


Figure 2.32: The Fredkin Gate with 19 pulses.

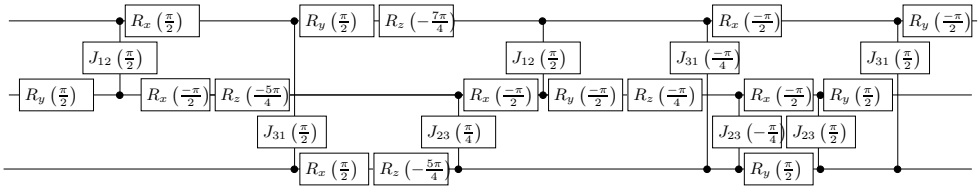


Figure 2.33: The Miller Gate with 24 pulses.

First the following are used in this circuit model. NMR Hamiltonian:

$$H = \sum_{k}(\omega_k I_{kz} + \omega_1(t)[I_{kx}cos\psi(t) + I_{ky}sin\psi(t)]) + \sum_{j,k}\pi J_{jk}2I_{jz}I_{kz} \tag{2.36}$$

Preferred single-qubit operations:

1. Rotation of qubit k by 90° and 180° about the x axis.

$$I_{kx}\left(\frac{\pi}{2}\right) \equiv exp\left(-i\frac{\pi}{2}I_{kx}\right) \tag{2.37}$$

$$I_{kx}\left(\pi\right) \equiv exp\left(-i\pi I_{kx}\right) \tag{2.38}$$

2. Rotation of qubit k by 90° and 180° about the y axis.

$$I_{ky}\left(\frac{\pi}{2}\right) \equiv exp\left(-i\frac{\pi}{2}I_{ky}\right) \tag{2.39}$$

$$I_{ky}\left(\pi\right) \equiv exp\left(-i\pi I_{ky}\right) \tag{2.40}$$

3. Rotation of qubit k by θ about the z axis.

$$I_{kz}\left(\theta\right) \equiv exp\left(-i\theta I_{kz}\right) \tag{2.41}$$

Preferred two-qubit operations

1. Rotations of the states of two-qubit j and k by θ through the evolution by the coupling term $2I_{jk}I_{kz}$.

$$J_{jk}\left(\theta\right) \equiv exp\left(-i\theta 2I_{jk}I_{kz}\right) \tag{2.42}$$

Any single-qubit rotation can be accomplished in three steps, known as Euler rotations. The Euler rotations are composed of two z-rotations and one y-rotation. We prefer 90° or 180° y-rotations and the y-rotations in arbitrary angles can be decomposed into two 90° x-rotations and z-rotation.

These figures can be compared with the macro-level specification of the Miller gate using 22 quantum gates from Figure 2.29d.

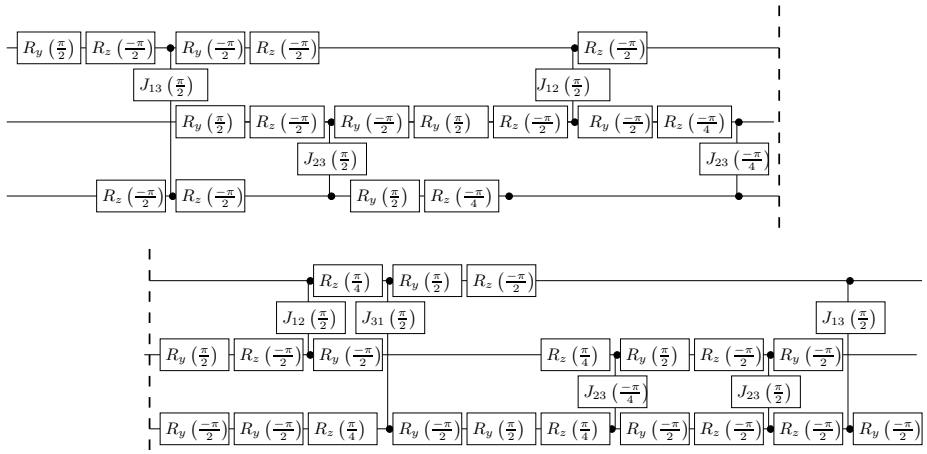


Figure 2.34: Miller Gate realized with 45 pulses from Equation 2.2.6.8.

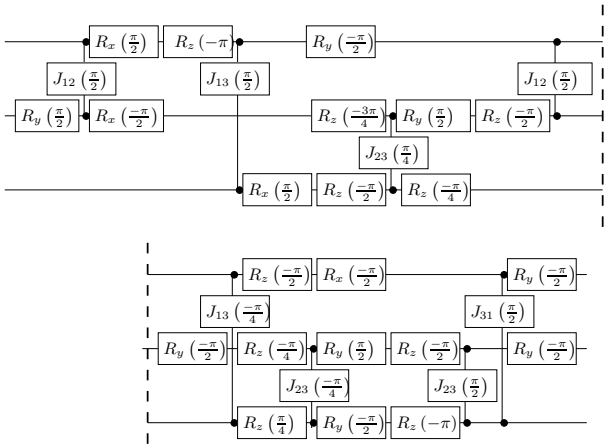


Figure 2.35: Miller Gate realized with 30 pulses from Equation 2.2.6.10.

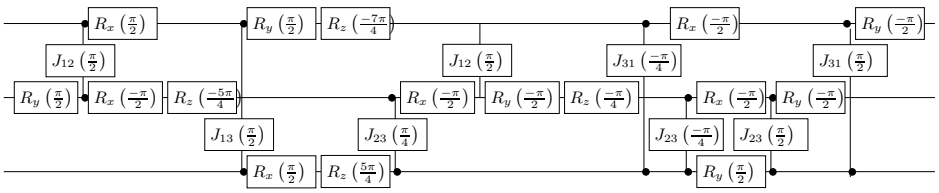


Figure 2.36: Optimal Miller Gate Realized with 24 pulses from Equation 2.2.6.11.

2.4.2.4 Large gates and gates for the "neighbor-only" technology

Example 2.4.2.2
In some quantum technologies such as "Linear ion trap" every qubit can communicate only with its neighbors above and below; this increases the cost of gates. If we have a wire that is "going through" the Feynman gate (Figure 2.37b), what should we do? We have to create a sequence of Feynman gates realizing SWAPs (Figure 2.37). The realization of Toffoli gate itself in the neighbor-only technology is shown in Figure 2.38. Again the SWAP gates should be transformed as in Figure 2.37a.

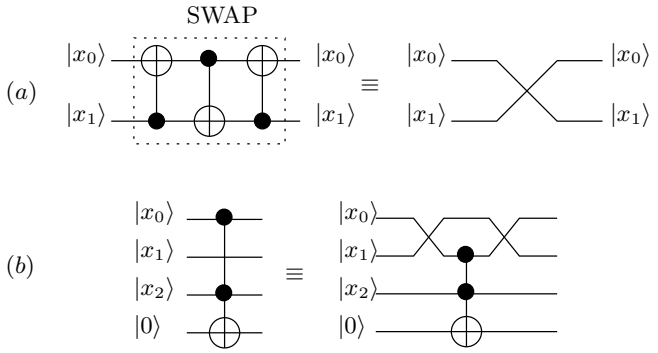


Figure 2.37: Transforming a 3×3 Toffoli gate with qubit $|x_1\rangle$ going through. (a) the SWAP gate, (b) the transformation of the Toffoli gate by surrounding it with two SWAP gates. Each of these SWAP gates is next transformed as in Figure 2.37a.

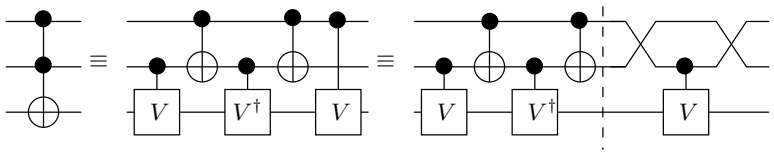


Figure 2.38: Realization of Toffoli gate in the technology that allows interactions only between neighbor qubits.

Example 2.4.2.3

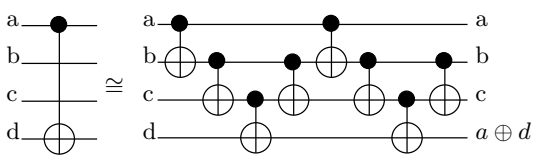


Figure 2.39: Transformation of "big CNOT" gate in the "neighbors only" quantum Technology. This is a Feynman gate with two-qubit wires "going through" it.

Example 2.4.2.4
A CNOT gate with many qubit wires "going through" can be realized as shown in Figure 2.39. Please note the Boolean equations used in the verification process. As we see from these simple examples, the "neighbor-only" technologies increase very substantially the costs of gates and circuits. These effects were entirely not taken into account by the previous researchers thus the claimed by them "minimal circuits" are in fact very far from the minimum when one calculates their costs on "pulse level" rather than "abstract mathematical gate level" (like n-input Toffoli). This is why the concept of affine gates was created [**?**, **?**, **?**]. For this is reason in some variants of GA (Chapter 4 and 5) we take the *neighbor only* constraint of linear Ion-Trap.

2.4.3 Quantum gates and circuits on the level of pulses in Quantum technologies such as NMR and ion traps.

2.4.3.1 NMR-based Quantum Logic Gates

The NMR (Nuclear Magnetic Resonance) technology approach to quantum computing [Moo65, PW02, DKK03] is the most advanced quantum realization technology used so far, mainly because it was used to implement the Shor algorithm [Sho94] with 7 qubits [NC00]. Yet other technologies such as Ion trap [DiV95], Josephson Junction [DiV95] or cavity QED [BZ00] are being used. The NMR quantum computing has been reviewed in details in [PW02,DKK03] and for this book it is important that it was so far the NMR computer that allowed the most advanced algorithm (7 qubit logic operation) to be practically realized and analyzed in details. Thus it is based on this technology that the constraints of the synthesis are be established in next chapters for the cost and function evaluation. Some prior work on synthesis has been also already published [**?**] and few simple cost functions have been used.

For the NMR-constrained logic synthesis the conditions are:

- Single qubit operations: rotations R_x, R_y, R_z for various degrees of rotation θ. With each unitary rotation (R_x, R_y, R_z) represented in equation 2.43.

$$
\begin{aligned}
R_x(\theta) &= e^{-i\theta X/2} = cos\frac{\theta}{2}I - isin\frac{\theta}{2}X = \begin{pmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \\
R_y(\theta) &= e^{-i\theta Y/2} = cos\frac{\theta}{2}I - isin\frac{\theta}{2}Y = \begin{pmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \\
R_z(\theta) &= e^{-i\theta Z/2} = cos\frac{\theta}{2}I - isin\frac{\theta}{2}Z = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}
\end{aligned}
\tag{2.43}
$$

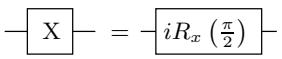


Figure 2.40: Single pulse Logic gate - NOT

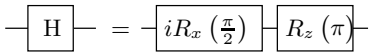


Figure 2.41: Two pulses Logic gate - Hadamard

- Two-qubit operation; depending on approach the Interaction operator is used as I_{zz} or I_{xy} for various rotations θ

Thus a quantum circuit realized in NMR will be exclusively built from single-qubit rotations about three axes *x,y,z* and from the two-neighbor-qubit operation of interaction allowing to realize such primitives as CNOT or SWAP gates. Examples of gates realized using NMR quantum primitives are shown in Figure 2.40 to Figure 2.43.

Also, the synthesis using the NMR computing model using EM pulses, is common to other technologies such as Ion Trap [CZ95, PW02] or Josephson Junction [BZ00]. Thus the cost model used here can be applied to synthesize circuits in various technologies, all of these technologies having the possibility to express the implemented logic as a sequence of EM pulses.

We are building large quantum matrices of algorithms from small quantum matrices of gates (pulses) that are realizable in some selected quantum technologies. In this section we will concentrate on realization of quantum circuits in two most advanced as of 2007 quantum realization technologies: that of liquid state nuclear magnetic resonance (NMR) [CFH97, GC97, JM98, JHM98] and ion traps [LBMW03, Pau90, Ste97, WMI+98, WBB+02, WH04].

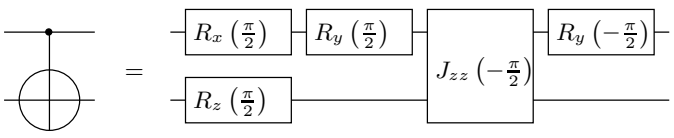


Figure 2.42: Detailed Realization of Feynman gate with five EM pulses.

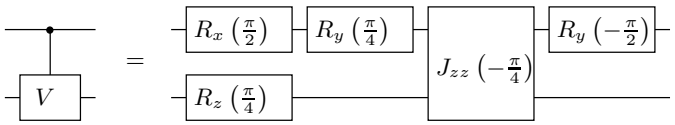


Figure 2.43: Five pulses Logic gate - Controlled-V

2.4.3.2 The quantum gates on the level of electromagnetic pulses. The fundaments.

The total calculation time in quantum computation depends on the number of basic gates in the series and the number of physical operations required for a quantum system to implement each gate. Let us denote a series of physical operations as a sequence of electromagnetic pulses distinguishing it from the series of basic gates, as the physical operations are either the time evolution of finite duration under the influence of an externally applied magnetic field, or interactions between qubits. In quantum computation, the calculation time is a very precious resource due to the finite coherence time of a quantum system. Therefore, it is important to know the cost of gates for the successful implementation of an algorithm, and thus for the future design of a practical quantum computer. Once the pulse sequences for the single-qubit and two-qubit gates are obtained, the total pulse sequence for a circuit is given by replacing each elementary gate by the corresponding pulse sequence. The pulse sequence of more complicated circuits with larger numbers of input qubits can be obtained in the same way, that is, by finding the quantum circuits composed of simpler gates and replacing each gate by the corresponding pulse sequence. In paper [LKBP06] the costs of gates were calculated in terms of numbers of basic pulses. The software used there calculated the cost of each gate by reducing the number of pulses in the sequence using the commutation rules of the pulse operations using nave greedy search algorithm. We demonstrate that these results can be improved by using the new heuristic search algorithm that will be developed in this chapter.

The optimized circuits presented in [LKBP06] are not necessarily minimal, since the heuristic algorithm that found them has no way of knowing if the solutions found are local or global minima. Therefore, they may not be the true minimal costs of gates, and the authors claim only to provide the upper bounds as the worst case. To evaluate the quality of their heuristic algorithm we develop exhaustive search to be used in comparison of small problems.

The new approach in quantum circuits synthesis introduced in [LKBP06] differed from the previous publications [SD96, SPIH03, Mil02, MD03] which optimized the quantum circuit at higher levels of abstraction. It is still rare to see papers in the literature that would optimize on the level of pulses, but this is in some chapters

of this book. This is partially possible thanks to our software which is intended to perform hierarchical top-down synthesis from various levels of specification. In one synthesis variant, the software will modify the initial non-optimal design by shifting gates left and right in the circuit and applying quantum logic identities, analogously to [LKBP06, Lom03], but calculating the combined cost of the operations that are necessary to build arbitrary quantum circuits instead of the total gate cost (gate number). The approach from [LKBP06] was next extended to larger circuits, but with a smaller number of transformations [MD03], the so-called "template matching approach". In next chapters we present software that operates on larger circuits and with a larger, user-defined numbers of operations.

The most important result from [LKBP06] is a table of realizations of useful gates and their costs, given in Table 2.4

The basic quantum gates that are used in quantum circuits are Inverter (NOT, Pauli X rotation), Hadamard, Toffoli, Feynman, CV (controlled square root of NOT) and $CV^{\dagger}$ (controlled square root of NOT Adjoint gate). *These gates are truly quantum and universal* . Their subset $\{NOT, CV, CV^{\dagger}\}$ allows creating all permutative binary quantum gates (circuits) by their compositions.

2.5 Quantum-Based Synthesis: useful quantum circuits synthesis problems

As we discussed in Example 1.5.2.1 the EPR circuit [NC00, Gru99] composed of a Hadamard gate and a Feynman gate realizes entanglement. In an extended circuit the Hadamard gate can be controlled, which means that when controlled with signal $|0\rangle$, the EPR circuit changes to a single Feynman gate and the entanglement is removed, thus the circuit's behavior becomes deterministic. Similarly the controlled Hadamard and Controlled Square-Root-of-Not (CV) gates can be used as sources of superposition and randomness. Such circuits find applications as possible robot controllers [RFW+07, LP07] where randomness of robot behavior is useful.

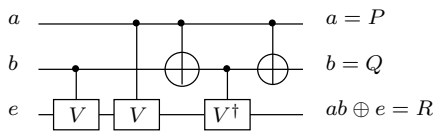


Figure 2.44: Toffoli gate realized using 2×2 controlled quantum gates. When used as a quantum robot controller, signals a, b and e can come from touch, sound or other sensors and outputs P, Q and R through measurement units go to motors or other actuators.

Table 2.4: Comonly used Quantum circuits realized in the Issing model for the NMR computer.

Gate Name	Pulse Representation	EM pulses Cost
NOT	$iR_x(\pi)$	1
Phase	$e^{i\frac{\phi}{2}}R_z(\phi)$	1
Hadamard	$iR_y(\frac{\pi}{2})R_z(\pi)$	2
CNOT	$e^{-i\frac{\pi}{4}}R_{2y}(\frac{\pi}{2})R_{1z}(-\frac{\pi}{2})R_{2z}(-\frac{\pi}{2})J_{12}(\frac{\pi}{2})R_{2y}(-\frac{\pi}{2})$	5
SWAP	$e^{-i\frac{3\pi}{4}}R_{2y}(\frac{\pi}{2})R_{2z}(-\frac{3\pi}{2})R_{1z}(-\frac{3\pi}{2})J_{12}(\frac{\pi}{2})R_{2y}(\frac{\pi}{2})$ $R_{1y}(-\frac{\pi}{2})J_{12}(\frac{\pi}{2})R_{1x}(\frac{\pi}{2})R_{2x}(-\frac{\pi}{2})J_{12}(\frac{\pi}{2})$ $R_{2y}(-\frac{\pi}{2})$	11
Peres	$e^{-i\frac{\pi}{8}}R_{3y}(\frac{\pi}{2})R_{3z}(\frac{\pi}{4})R_{2z}(-\frac{3\pi}{4})R_{1z}(-\frac{\pi}{4})J_{23}(\frac{\pi}{4})$ $R_{2x}(-\frac{\pi}{2})J_{12}(\frac{\pi}{2})R_{2y}(-\frac{\pi}{2})R_{2z}(\frac{\pi}{4})J_{31}(-\frac{\pi}{4})$ $J_{23}(-\frac{\pi}{4})R_{3y}(-\frac{\pi}{2})$	12
Toffoli	$e^{-i\frac{\pi}{8}}R_{1z}(-\frac{\pi}{4})R_{2z}(-\frac{\pi}{4})J_{12}(\frac{\pi}{4})R_{3y}(-\frac{\pi}{2})$ $R_{3z}(\frac{\pi}{4})J_{31}(-\frac{\pi}{4})J_{23}(-\frac{\pi}{4})R_{3y}(\frac{\pi}{2})J_{31}(\frac{\pi}{2})$ $R_{3x}(\frac{\pi}{2})J_{23}(-\frac{\pi}{4})R_{3x}(-\frac{\pi}{2})J_{31}(-\frac{\pi}{2})$	13
Fredkin	$e^{-i\frac{7\pi}{8}}R_{2y}(\frac{\pi}{2})R_{1z}(-\frac{3\pi}{4})J_{23}(\frac{\pi}{2})R_{3x}(\frac{\pi}{2})R_{3z}(-\frac{3\pi}{4})$ $J_{12}(\frac{\pi}{2})R_{2y}(\frac{\pi}{2})R_{2z}(-\frac{7\pi}{4})J_{23}(\frac{\pi}{4})R_{2x}(-\frac{\pi}{2})$ $J_{12}(\frac{\pi}{2})R_{2y}(-\frac{\pi}{2})R_{2z}(-\frac{\pi}{4})J_{31}(-\frac{\pi}{4})J_{23}(-\frac{\pi}{4})$ $R_{3x}(\frac{\pi}{2})R_{2x}(-\frac{\pi}{2})J_{23}(\frac{\pi}{2})R_{2y}(-\frac{\pi}{2})$	19
Miller	$e^{i\frac{7\pi}{8}}R_{2y}(\frac{\pi}{2})J_{12}(\frac{\pi}{2})R_{2x}(-\frac{\pi}{2})R_{2z}(-\frac{5\pi}{4})R_{1x}(\frac{\pi}{2})$ $J_{31}(\frac{\pi}{2})R_{1y}(\frac{\pi}{2})R_{1z}(-\frac{7\pi}{4})R_{3x}(\frac{\pi}{2})R_{3z}(-\frac{5\pi}{4})$ $J_{23}(\frac{\pi}{4})R_{2x}(-\frac{\pi}{2})J_{12}(\frac{\pi}{2})R_{2y}(-\frac{\pi}{2})R_{2z}(-\frac{\pi}{4})$ $J_{31}(-\frac{\pi}{4})J_{23}(-\frac{\pi}{4})R_{3y}(\frac{\pi}{2})R_{2x}(-\frac{\pi}{2})R_{1x}(-\frac{\pi}{2})$ $J_{23}(\frac{\pi}{2})R_{2y}(-\frac{\pi}{2})J_{31}(\frac{\pi}{2})R_{1y}(-\frac{\pi}{2})$	24

The realization of the reversible Toffoli gate (Fig. 2.44) using Controlled-NOT ($CNOT$), Controlled-V (CV) and Controlled-V† ($CV^\dagger$) gates [BBC$^+$95, HSY$^+$06, NC00] is another source of inspiration to create quantum circuits. Figure 2.44 shows that a deterministic behavior of a permutative (classical reversible) circuit is created using truly quantum gates (such as CV). These gates operate in Hilbert Space and create intermediate signals that are superposed [NC00]. (By truly quantum gates we understand those that their unitary matrices are not permutative).

If we would thus measure the data path signal in the lowest qubit in Fig. 2.44 in the middle of this circuit, after two CV gates controlled by inputs a and b respectively, the behavior would be deterministic for some input signal combinations and probabilistic for other combinations, leading to very interesting behaviors of robots such as Quantum Braitenberg Vehicles [RFW$^+$07] controlled by this circuit.

Even more complicated binary quantum circuits (with permutative unitary matrices) can be composed from gates that are the controlled Pauli X rotations by angles π/k where k is a power of two. This leads to gates such as NOT - 180° rotation, square-root-of-not - 90° rotation, fourth-order-root-of-not - 45° rotation, etc. Gates that rotate by $k * (2\pi/3)$ where k is an integer are used in ternary quantum logic with base states $|0\rangle$, $|1\rangle$, $|2\rangle$ [?, **?**]. These all rotation gates can be controlled by arbitrary quantum states [MMD06]. When the resultant signal in the data path bit (the controlled qubit) is an eigenvalue of the unitary transformation(s), the behavior is deterministic. When it is not, the behavior is probabilistic according to the rules of quantum measurement [NC00, Gru99]. This means that a system in a superposed state, when measured, collapses to one of the possible observables given by the measurement operator. This way, a circuit can be designed from a set of examples corresponding to the care minterms of a truth table. For instance, referring again to Quantum Braitenberg Vehicles, value 0 may correspond to sensor conditions when we want our robot to turn left, and value 1 to the true minterm of input variables (a positive example) when the robot should turn right. Based on his design goals the designer specifies examples of robot behaviors as input-output pairs. The software induces behaviors for all other input states that are possible.

2.5.1 Cost of quantum circuits

Various cost models have been used in QLS to date. Table 2.5 presents a summary of various costs arranged from the highest gate level to the lowest one (Pulses, Rotations). The variation of the costs is respective to the synthesis level and can be viewed as a transition cost from Reversible Logic Synthesis to QLS. At the top of Table 2.5 there is a High Level cost of quantum primitives that has a direct correspondence to reversible gates. In the bottom row there is the cost expressed in

unitary pulses, thus closest to the real quantum cost. Such gates are uniquely quantum and thus they offer simplifications that cannot be achieved on higher levels. In more details, the first column of the Table 2.5 represents the cost category/level, the second columns shows some of gates used on this level of logic synthesis. The third column gives the cost specifications, for instance the first row has a cost increasing with the number of controls per logic gate as well as with the number of used gates. Thus, the more controls a gate has the more expensive and difficult it is to realize. The fourth column shows how the overall cost grows; again in the first row it is shown that one can take single-qubit controlled-Not and two-qubit controlled-Not having the basic cost of one, while the cost grows with additional control bits.

Table 2.5: Cost models

Cost Name	Example Gates	Cost Specification	Cost Values
High Level	CNOT,CCNOT,etc	cost per control cost per gate	1,1,2,etc
Standard	W CNOT,CCNOT	cost per wire cost per control	1 2,3,etc.
Restricted	W CNOT,SWAP	cost per wire	1 2,6,etc.
Low Level	$R_x(\phi)$,I_{ZZ}	cost per pulse	1,2

Such a reduced cost can be observed for the SWAP gate. In general it is assumed that the SWAP gate is made from three Feynman gates, and thus according to Figure 2.42 it would have a cost of 5 x 3 = 15 EM pulses. However because of the nature of the pulses it is possible to combine consecutive pulses on the same axis of the Bloch sphere together and thus minimize the pulse cost to 11 [?]. In this model the cost is calculated according to the bottom line; for instance the $R_x(\pi)$ single-qubit rotation has the same cost as rotation $R_y(-\frac{3\pi}{4})$. Moreover, adjacent quantum gates located on the same qubits can as well be combined. This will be shown in Chapter **??**. For synthesis this implies that for a particular technology various rules will constrain the search and thus will allow to adjust the cost functions[2].

Definition 2.5.1 (Cost of a Quantum Circuit)**.** *cost* $= f(Qc_U) = \sum_j U$ *is the sum of the costs of each operation used for computation in the circuit.* □

Definition 2.5.1 shows a simple positive cost function monotonic in j.

Example 2.5.1.1 Quantum Logic Primitives
There already exists a popularly used library of quantum primitives that is applied

[2]Moving pulses and reducing the sequences of pulses is also considered a useful heuristics for quantum circuits synthesis.

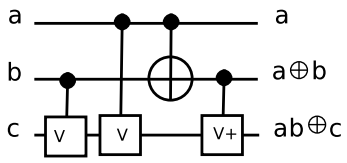


Figure 2.45: Structure of the Peres gate built from 2×2 quantum primitives

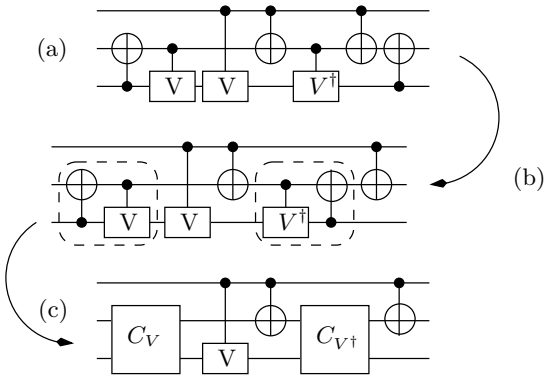


Figure 2.46: Structure and minimization stages to reduce the cost of the Fredkin gate. The top diagram shows the circuit non minimized. The middle diagram represents the circuit with permuted gates to allow forming larger blocks. The bottom diagram shows the minimized circuit for which the total quantum cost is calculated [LPG+04].

by the current quantum logic synthesis algorithms. These primitives are directly derived from atomic operations in quantum mechanics and constitute the basis for logic operations in Quantum computing. In general, these gates are from the following group: Wire, Inverter (Pauli X), Pauli Y, Pauli Z, Hadamard, Feynman, CV, $CV^\dagger$, Peres, Fredkin and Toffoli. In different technologies such as NMR or Ion Trap these gates have different costs and their optimal realizations are not yet established due to the fact that quantum computing is still only at its beginning. From the synthesis point of view, different approaches based on various parameters can be taken to calculate the "total quantum gate cost" of the quantum circuit which estimates the real realization cost of a quantum circuit, that is very much dependent on particular technology or even on particular equipment. We are thus interested only in the total gate cost which we will call the Quantum Cost for short.

The simplest method is to calculate the total number of 2-qubit primitives [SD96, LPG+04]. Using this method the Peres gate has the cost of 4 (see Figure 2.45), the Toffoli gate (see Figure 2.60) has the cost of 5, and the Fredkin gate has also the

cost of 5 (see Figure 2.46). □

2.5.2 The Size of Quantum circuits

Beside the cost, a quantum circuit can be described by a size. However, in order to allow precise definition of the size, first some required concepts need to be defined.

Definition 2.5.2 (Quantum Circuit Primitives)**.** *A quantum circuit primitive is the smallest unit of the synthesis process. As such, a quantum primitive does not represent a unique gate but rather the smallest unit used to build quantum circuits in a given synthesis model.* □

Definition 2.5.3 (Quantum Circuit Block)**.** *A block of Quantum Circuit is another Quantum Circuit or a Quantum Gate of arbitrary width but smaller than* n *(in which case the block becomes a segment). Also, a set of quantum gates is a block only if the contained gates are closed; i.e. the gates are fully enclosed inside of the block (no input or output vertically). Example of blocks of gates are shown in Figure 2.47.* □

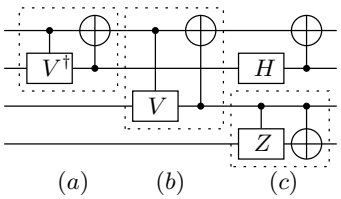


Figure 2.47: Example of Blocks of a Quantum Circuit.

With respect to the definition of block, a Segment of a Quantum circuit is defined by:

Definition 2.5.4 (Quantum Circuit Segment)**.** *A Segment of Quantum Circuit is another Quantum Circuit or a Quantum Gate of width* n *, such that it is built only by using Kronecker product between its component gates.* □

For instance, the three-qubit quantum circuit from Figure 2.48 can be separated into three segments, each of width three and each being built only by using the Kronecker product. Observe, that such definition then allows to build the quantum circuit by using standard matrix product and a set of segments.

The size of the circuit can now be simply defined as:

Definition 2.5.5 (Circuit size)**.** *The size of a Quantum Circuit is equal to the minimal number of segments that a circuit is described by.* □

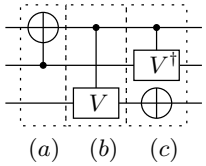


Figure 2.48: Example of Segmentation of a Quantum Circuit. In this case, the circuit is built as a serial composition of three parallel segments a, b and c.

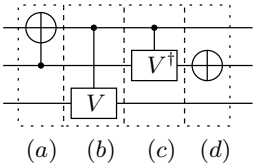


Figure 2.49: Example of Segmentation of a Quantum Circuit. In this case, the circuit is built from three parallel segments a, b, c and d.

For instance the circuit from Figure 2.48 has size 3 while the circuit from Figure 2.49 has size 4 despite of having the same cost.

Finally, with the above definitions the overall process of Quantum Logic Synthesis is described in Definition 2.5.6.

2.5.3 Quantum Logic Synthesis of Combinatorial Circuits

Definition 2.5.6 (Quantum Logic synthesis)**.** *The Quantum Logic Synthesis problem is to find the circuit that has the minimum value of the Quantum cost (whatever the definition of this cost) for a given truth table, K-map or other specification.*

Let $|\psi\rangle$ be the quantum register and let G be a set of single-qubit and two-qubit unitary operators on complex Hilbert space H^n. The process of synthesis of an appropriate quantum circuit finds the unitary matrix U such that the following relation:

$$U|\psi\rangle \rightarrow |\psi'\rangle \tag{2.44}$$

is satisfied for every defined care pair (ψ, ψ') of input-output. Therefore for each such pair the probability of obtaining the output state from the pair is 100%.

In another synthesis variant, the synthesis process can be expressed as a minimization of the given reversible or quantum-reversible function with respect to

- *the size of the circuit* s_{Qc} *(the number of elementary operators used),*
- *the width of the circuit* w_{Qc}*,*
- *and the error* e *with respect to the function to be designed as a circuit.*

Thus, the QLS process can be written as:

$$S_{H^N}(n, G) \xrightarrow{min(V(s,w,e))} Qc \tag{2.45}$$

where $V(s, w, e)$ *is the function to be minimized during the synthesis process. This function represents the overall cost of the circuit constructed using gates from set* G *(a size* s *of the circuit), with a width* w *of the circuit and with a given error* e*.* □

No synthesis methods are yet known that would formally consider the trade-off between the width of the quantum register and the quantum cost.

Constructing universal gates from smaller primitives is only one of several goals of quantum logic synthesis. As gates are represented by matrices, an infinity of combinations exists to represent any quantum gate. Also it follows logically that a gate can be easily invented by just creating a matrix for a particular function. However, as each gate is in fact one atom or a group of them, a quantum gate (matrix) must be in principle realized in accord with physical laws of EP's. But their costs may differ dramatically.

Reversible functions such as Toffoli or Fredkin are defined as gates and used as synthesis concepts for convenience, but all quantum gates with more than 2 qubits are practically not gates but circuits. This restriction is due to the fact that the state of the art quantum computers allow at present to build only one-qubit and two-qubit gates (for instance the rotation gates and the interaction gates J_{zz} in Figure 2.42).

The search techniques in quantum synthesis can be mainly split into two streams: algorithmic and heuristic. This is because there are still no tools available that would allow systematic and theory-grounded algorithm design research in this area. Results from manual or human-based heuristics can be found in [**?**, BBC$^+$95, Per00, SD96] while recent publications [LPG$^+$04, LP02, Rub00, WG98] show algorithmic rediscovery of the already known gates. Also some interesting circuits have been discovered and proved optimal by automated processes in [?, HSY$^+$04, YHSP05].

More relevant than in classical circuits synthesis, in quantum circuit synthesis, the technology influences the synthesis to a much higher extent by specifying the primitives to be used in the synthesis. This can be seen in the fact that various quantum

technologies use different sets of single-qubit rotations as well as different types of interaction gates (in this book we restrict our interest to the popular in NMR J_{zz} interaction gates). Thus every specific variant of quantum technology will specify to a much higher extent what type of gates and single-qubit primitives are available for logic synthesis. This is due to the fact that most of the quantum technologies are still only at the experimentation level and thus no standards for primitives, cost or technology-specific constraints have been established.

2.5.4 Quantum Circuits and Sequential Logic

2.5.4.1 Classical vs. Quantum Circuits representation

In classical logic design, one deals with physical elements (CMOS, Transistor Level) that need to have been given a certain input state, that propagates through the network of interconnected gates (logic elements) and generates output after the propagation delay τ. The output will be held as long as the input is held as well. This means that one can describe a reductionist classical circuit on a 2-dimensional plane. The Y axis describes the space (representing the width of the circuit) and allowing for parallel processing on the level of gates. The X axis represents time and space, as it represents the length of the circuit (propagation delay). This can be seen in Figure 2.50.

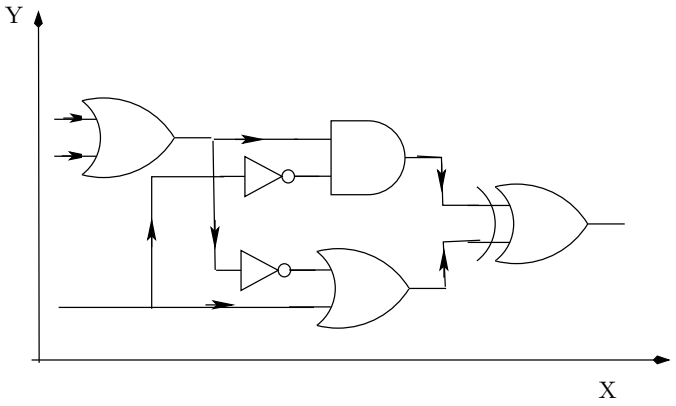


Figure 2.50: Space-time representation of a classical logic circuit. Time flows from left to right along the X axis (Y axis represents space dimension), unless a vertical interconnection is made in which case time also flows along the Y direction, following the direction of arrows.

In quantum circuits this representation must be however modified in order to be correct. Assume a quantum register of width w. While the width of a quantum circuit still corresponds to the Y axis from the classical circuits, the X dimension

does not mean the space component anymore. Unlike in the classical case, the logic operation might not be represented by a set of interconnections and logic elements, but instead it is represented by a set of localized EM pulses sequentially emitted on the same set of qubits (Figure 2.51).

Observe that this particular representation allows to describe parallelism and serial operation in quantum computing. In each time slice (called t_0, t_1, etc.) in Figure 2.51 a set of fully parallel operations on three qubits is represented (such as CNOT and U_1 executed in parallel in slice t_3). In contrast, each time slice is a serial operation with respect to the whole sequence representing the quantum circuit (this is illustrated by the sequence t_0, t_1, t_2, t_3 in Figure 2.51).

The validity of this particular representation is based on the NMR or the Ion Trap spin *state model* . For instance in solid-state quantum computing, the individual qubits can transmit information between neighbors allowing quantum synthesis to be effectively represented by a two dimensional space-time grid. Moreover, as will be discussed in details later, the computing procedure on a quantum circuit is a sequence of the following operations:

1. initialize the whole quantum register to a desired initial state
2. apply the transformation U
3. measure the desired qubits and observe the result.

Despite the different protocols of quantum computing, the main concept used in for this book is that while in classical logic there is an actual flow of particles through the gates, in quantum computing the gate is dynamically created by EM pulses that create a new quantum state. The state remains unchanged until either another logic operation is applied, the circuit is initialized, the circuit is measured or external events (noise, decoherence) perturb the quantum state.

2.5.4.2 Quantum Circuit, a natural register.

The conservation of the quantum state implies also the fact that a quantum circuit can be seen as a state machine in a particular state; i.e. after being initialized to $|\phi\rangle$ the state is now $U|\phi\rangle$, with U being the circuit unitary transformation obtained by the sequence of EM control pulses. This equivalence (between quantum circuit and state machine) is related to one of the main problems in FSM design; the state assignment problem. For instance, assume a FSM with states $Q = \{q_0, q_1, q_2, q_3, q_4\}$ and a state transition function $\delta(q, s) \rightarrow q'$, the problem is to find such state assignment that would minimize the functional logic (the state transition function).

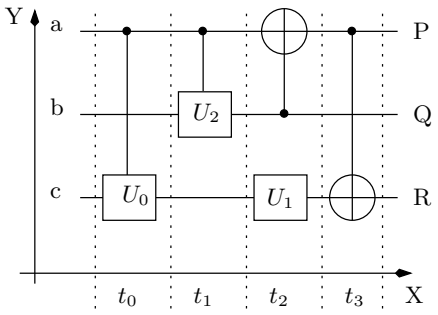


Figure 2.51: Space-time representation of a quantum logic circuit without measurement. Time flows from left to right along the X axis while space is represented by the Y axis. Remark that the physical size of this circuit is equal only to its width.

In this case, there are $5 \leq 2^3$ states the number of distinct state assignments is $\binom{8}{5} = 6720$. Therefore, synthesizing a single state machine with a randomly selected assignment does not directly addresses the problem of finding the best state assignment, but allows to synthesize machines with the same approach for both quantum circuits and finite state machines. If the state assignment is created as a byproduct of minimizing the logic circuit cost, as it will be done in our evolutionary approaches, this assignment is from definition good, without specifically using state-assignment methods.

Thus, a given quantum circuit can be considered as a quantum finite state machine (with determined state encoding), where each iteration of the unitary transformation U on $|\psi\rangle$, will generate the sequence of states given by $|\psi'\rangle = U|\phi\rangle$ starting in the initial state $|\psi_0\rangle$. For instance, in the quantum circuit from Figure 2.51, the top qubit (qubit a) can be considered as the FSM internal state qubit, the qubit b as the input, and qubit c as the output. In such a case, the computing procedure is described by the following QFSM realization algorithm:

1. initialize the whole quantum register to a desired initial state only once
2. apply the transformation U representing the desired function f
3. measure the output qubit(s) and observe the result.
4. initialize one input qubit (all next initializations of the qubit register affect only the input qubit.)
5. go to step 2

Example 2.5.4.1 Simple Quantum Gate as a Quantum Finite State Machine

Consider the Feynman gate described by the function shown in Table 2.6. Traditionally

Table 2.6: K-map of the Feynman gate

a \ b	0	1
0	$\|00\rangle$	$\|01\rangle$
1	$\|11\rangle$	$\|10\rangle$

the function described by this gate is a change of the values between two-qubits, however the implementation as a FSM allows a different point of view. In this case, assume that the FSM is constructed such that one of the qubits is the input (as described above, it is initialized at each computational step) and the second qubit represents the state. Moreover, a classical controller is required to control the overall functioning of the FSM implementation.

Figure 2.52b shows the CNOT gate built as a FSM; the qubit a represents the input qubit, the qubit b represents the state qubit. The whole setup is controlled by a classical computer, that can either initialize the quantum register, perform the quantum computation or measure the desired qubits. This setup allows to initialize the input qubit a while preserving the state qubit b. This is also shown in Figure 2.52a where the symbols I_0 or I_1 represent the initialization of the input qubit a and C represents the computation phase. I_0 initializes qubit a to state $|0\rangle$, I_1 initializes qubit a to state $|1\rangle$. During phase C the controls to the gates representing CNOT are given. Initialization of "state qubti b" is done only at the beginning of entire operation (Step 1). Observe that when not initialized, the machine will toggle between states $|11\rangle$ and $|10\rangle$.Thus the measurement of qubit b will result in random result while the measurement of qubit a will result in $|1\rangle$.

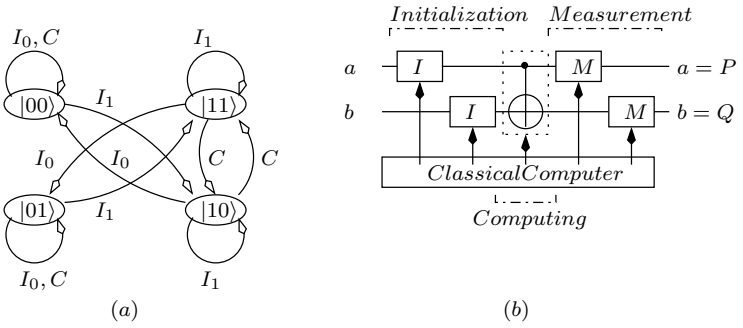


Figure 2.52: (a) - The state diagram for a FSM using a single Feynman Gate, (b) - schematic representation of a FSM built according to the protocol described in the text as QFSM Realization Algorithm.

□

As the final point of this section let us discuss the implications that the previous descriptions bear for the logic synthesis:

- Quantum circuits are represented as unitary matrices. The computation includes the initialization of the whole quantum register before computation to an arbitrary input state. This corresponds to setting the binary minterm as a state of the quantum register. The application of U generates the output for the given input state. Thus a quantum circuit is directly represented by the unitary transform U.
- Finite State Machines require a register to store the internal state; quantum computing allows to store in a natural way the complete state (the input signals, the internal state signals, the output signals) during the computation process. Thus using a Unitary transform U allows to represent one of many possible realizations of a given quantum FSM. This FSM has its behavior described by a set of states and a state transition and output function given by the unitary matrix U (both the states and the state transition function must be quantum realizable).
- Thus synthesizing Unitary transformations for circuit design, is a quantum equivalent of classical logic circuit synthesis methods of both combinational logic and state machines.
- The synthesis of a quantum circuit can be executed with respect to the observable output (after the measurement), with respect to the unitary matrix representing the quantum circuit (before the measurement), or with respect to new observable values on the circuit outputs (discovery of circuits generating novel output states) after the measurement.

2.6 Principles of Synthesis for NMR technology

2.6.1 V - Gate, T - Gate and the principle of "Level generalization"

The principle of creating permutative circuits using the V and the $V^{\dagger}$ gates (Section **??**) can be extended to SWAP gate. This is possible because SWAP gate has a NOT [3] sub-matrix as can be seen in equations 2.46 and 2.47. Equation 2.46 represents the SWAP gate and its Square-root, the T gate. The equation 2.47 represents the nSWAP (nS) gate and its Square-root.

[3] Feynman, Fredkin and Toffoli gates are all a combination of the Control signal and the NOT target bit. Thus they do all have a NOT sub-matrix in their complete unitary matrices

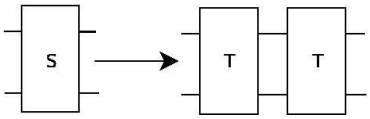


Figure 2.53: SWAP gate broken into two T gates

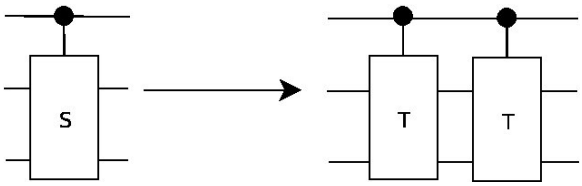


Figure 2.54: Fredkin gate broken into two Controlled-T (CT) gates

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \sqrt{S} = T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{2} & \frac{i-i}{2} & 0 \\ 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.46}$$

$$nS = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \rightarrow \sqrt{nS} = nT = \begin{pmatrix} \frac{1+i}{2} & 0 & 0 & \frac{1-i}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1-i}{2} & 0 & 0 & \frac{1+i}{2} \end{pmatrix} \tag{2.47}$$

The nS gate (introduced for the first time in [LP05b]) operation is opposite to the SWAP gate: while the SWAP gate exchanges the values of the qubits when they differ ($|01\rangle$ and $|10\rangle$) the nS gate swaps values when both qubits are equal ($|00\rangle$ and $|11\rangle$).

Both of these gates have the V sub-matrix and thus the T gate is a good candidate for logic synthesis similarly to the powerful CV gate. Again, the right side of equation 2.46 reveals the V core of the T gate. Compared to the CV gate, the matrix is shifted diagonally left-up. For the nT gate (square-root-of-nSWAP gate) the V core can be seen dispersed to the four corners of the matrix. From the above it can be concluded that the following identities hold true:

$$T \times T = S, \;\; nT \times nT = nS, \;\; T \times nT = nT \times T = I \tag{2.48}$$

The function realized by the Controlled-Swap (CS) is a conditional swapping of two target qubits. For instance let $|\phi\rangle = \frac{|110\rangle+|101\rangle}{\sqrt{2}}$ be an initial state and applying CS yields $CS|\phi\rangle \rightarrow \frac{|101\rangle+|110\rangle}{\sqrt{2}}$. The unitary matrix of the T (square-root-of-swap) is very similar to the matrix of the CV and thus can be expected to have similar powerful properties when used in synthesis. Thus, replacing the CV gates in the Peres gate by the CT gate gives a possible lead to explore such scaling properties (Figure 2.55). This is true for all gates from the Peres family (see below) and thus also for all families from Figure 2.56. For convenience this property of replacing gates with larger gates but preserving the relative structure of the circuit will be called the 'level-generalization'.

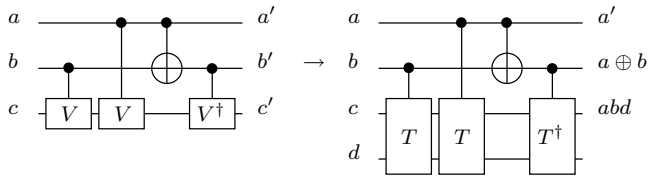


Figure 2.55: The level-generalization of Peres using CT. From a three-qubit circuit with (CV and $CNOT$ gates) at the left, the four qubit circuit (using $CNOT$ and CS) is related as its "Level-generalization". The logic equation of the generalized circuit is ab controlled SWAP gate.

2.6.2 Insertion principle

Similar to the previous property for synthesis, we introduce in this section the generalization of the concept of gate insertion. We will call our new principle the Peres-Toffoli-Fredkin (PTF) principle. The PTF principle just states that based on the simple transform required to go from Peres to Toffoli and Feynman, there can be more similar simple insertion or removal operations giving similar relation between different 'interesting' circuits. Using this PTF principle the exploration of the quantum circuit problem space is well situated to techniques exploring local groups (blocks) of gates and circuits. This is because the PTF principle can be algorithmically searched using simple exhaustive search. The PTF principle is shown in Figure 2.56. By adding more Feynman gates, several interesting circuits such as the Miller gate [LPMP02] can be directly created.

2.6.3 The Divide and Conquer Principle

The last property introduced here is the principle of "divide and conquer", explained on the following example. A single Fredkin gate is broken down into two

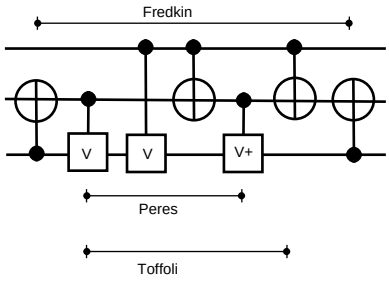


Figure 2.56: Peres, Toffoli and Fredkin gates illustrating simple search technique (PTF principle).

$Controlled - T$ gates and by simple adding of Feynman gates many new circuits are created. This example is illustrated in Figure 2.57. This new gate generation principle results from combining and generalizing two previous principles: the level-generalization and the PTF principle.

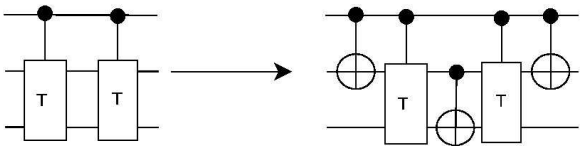


Figure 2.57: The "divide and conquer" circuit synthesis method

The above simple and powerful principles and tools to construct quantum circuits are sufficient to describe the low level of the GAEX tool (Chapter **??**). GAEX is an evolutionary-exhaustive-transformative software 'explorer' for the quantum circuit synthesis. This program is explained in the Section **??**.

2.6.4 The Gate-collapsing principle

The last important principle in the QLS is the so called gate-collapsing principle. It is based on the fact that as each quantum gate is represented by a unitary matrix and any neighboring gates that are on the same qubits can be collapsed into a single one. This is shown in Figure 2.58. The requirements for the gate collapsing principle are the following:

- gates must be neighbors (adjacent)
- gates must be defined on the same qubits
- gates must have the same width

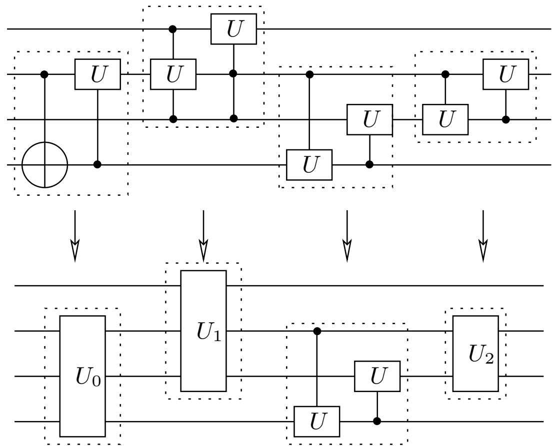


Figure 2.58: The gate-collapsing principle: quantum gates located on the same qubits and having no other gates in between can be *collapsed* into a single one.

The gate-collapsing principle can be also applied in specific cases when some of the above conditions are not fulfilled. Figure 2.59 shows two such cases (c) and (d). In order to collapse two non adjacent gates the following conditions must be fulfilled:

- moving gates left (or right) cannot change the value on any other control qubit (Figure 2.59a and 2.59b)
- moving gates left (or right) can be done if this gate movement affects only target qubit (Figure 2.59c and 2.59d)

In both Figures 2.58 and 2.59 the newly created boxes labelled U_x represent the gates resulting from the collpasing of such gates that result in a different gate when combined.

2.7 Examples of circuits obtained automatically for EM-pulses based quantum circuit technology using methods from sections 2.5 and 2.6

Now let us continue the discussion of the circuit decomposition into primitives started earlier in this chapter. Let us consider one practical example. The Toffoli or Fredkin gates introduced in section 2.5 are both universal quantum logic gates that are already well-known. They have been built in several quantum and reversible technologies. The problem discussed here is to find an optimal decomposition of the

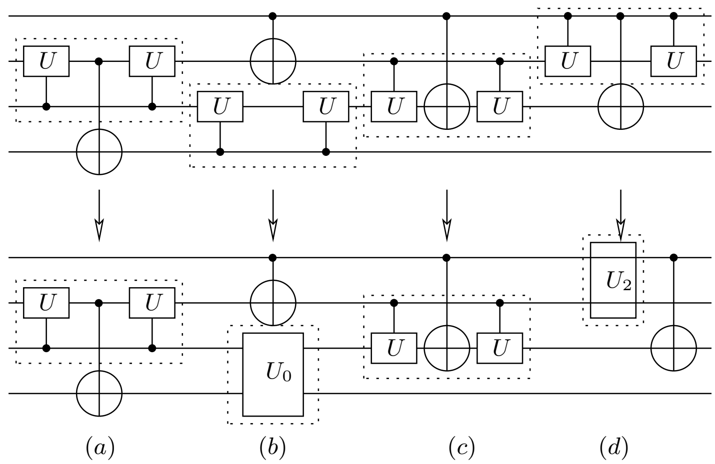


Figure 2.59: The gate-collapsing principle: quantum gates can also be collapsed if they can be moved and become neighbors without altering the controls of other quantum gates.

universal gates into smaller parts, especially into the directly realizable quantum primitives such as Feynman, NOT or Controlled-V (CV) gates. As mentioned earlier, the gates with one and two-qubits have costs directly dependent on the number of EM impulses. Thus using the result from Section 2.4.3.2, the individual costs for every single gate are the following: $W = 1$, $Phase = 1$, $H = 2$, $CNOT$,$CV = 5$, $Swap = 11$, $Peres = 12$, $Toffoli = 13$, $Fredkin = 19$. Figure 2.60 presents the well-known realization of Toffoli gate from [SD96]. There are five 2-qubit primitives here: CV_{23}, CV_{13}, $CNOT_{12}$, $CV_{23}^{\dagger}$, $CNOT_{12}$, and the cost is 5 * Cost($CNOT$) = 25. The subscript on each gate name signifies the wires that the gate is connected to. For instance on a three qubit circuit the gate $CV_{23}^{\dagger}$ is controlled by the second qubit and applies the conditional $V^{\dagger}$ transformation to the third qubit.

The circuit implementing Toffoli gate with the above cost is the solution with the smallest amount of used gates for the set of quantum gates consisting of CNOT and Controlled-V/V†. Different minimial circuit for the Toffoli gate is obtained when for instance the CNOT and the Controlled-Hadamard gate is used [LPK10, LBA^{+}08]. Thus a minimization of a circuit cost with respect to a known minimum will allow to both find circuits directly reducible to the ideal one or find circuits different than the ideal circuit. This will be presented later (chapter **??**) where we show circuits that realize the Toffoli gate with the same cost and the same component gates, as well as circuits realizing the Toffoli gate with a higher cost and with the realized function being the correct one. Observe that the transformations presented in these examples all minimize the cost of quantum circuits. Interestingly, these cost reducing methods allow to transform a circuit from being built using one set of quantum gates to another gate using a different set of quantum gates. Thus the cost

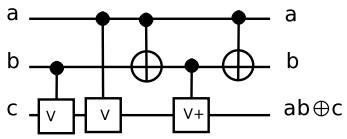


Figure 2.60: Toffoli circuit from 2×2 quantum primitives

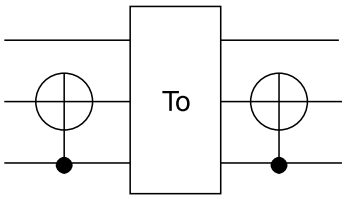


Figure 2.61: Toffoli-based Fredkin circuit from Feynman gates and a macro-Toffoli gate

minimization does not alter only the circuit structure but creates also new quantum gate primitives that can be used for QLS.

Using the PTF principle (Figure 2.56), we can realize the Fredkin gate from the Toffoli gate. The Fredkin gate can be synthesized using two Feynman gates and one Toffoli gate as in Figure 2.61. The cost of this gate is 2*5 + 25 = 35.

Substituting the Toffoli design from Figure 2.60 to Figure 2.61 we obtain the circuit from Figure 2.46a (top). Now we can apply an obvious EXOR-based transformation to transform this circuit to the circuit from Figure 2.46b (middle). This is done by shifting the last gate at right (Feynman with EXOR up) by one gate to the left. The reader can verify that this transformation did not change logic functions realized by any of the outputs. Observe that a cascade of two 2*2 gates is another 2*2 gate, so by combining a Feynman with EXOR-up gate (cost of 5), followed by controlled-V gate (cost of 5) we obtain a new gate CV with the cost of 5. Similarly gate $CV^{\dagger}$ with cost 5 is created (the unitary matrices of both CV and $CV^{\dagger}$ are shown in equation 2.49).

$$CV = [NOTC] \times [CV] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$CV^{\dagger} = [NOTC] \times [CV^{\dagger}] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.49}$$

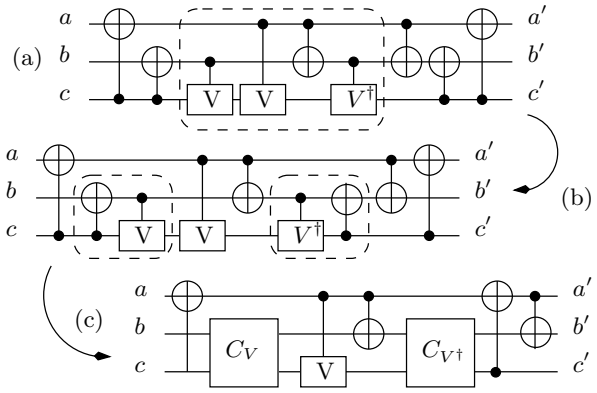


Figure 2.62: Stages of the minimization of the Miller Gate. Observe that by using the Peres macro and the collapsing principle the overall cost of the gate is reduced. The important fact is that the collapsed blocks C_V and $C_{V^\dagger}$ can be much less expensive in some technologies than in others.

This way, a circuit from Figure 2.46c (bottom) is obtained with the cost of 25. (This transformation is based on the method from [?] and the details of cost calculation of CV and $CV^\dagger$ are not necessary here). Thus, the cost of Toffoli gate is exactly the same as the cost of Fredkin gate, and not half of it, as was previously assumed and as may be suggested by classical binary equations of such gates.

Encouraged with the above observation, that sequences of gates on the same quantum wires have the cost of only single gate on these wires, we used the same method to calculate costs of other well-known gates. Let us now investigate a function of three majorities investigated first by Miller [Mil02, MD03, YZL03]. This gate is described by equations: P = ab $\oplus$ ac $\oplus$ bc, Q = $\bar{a}$b $\oplus$ $\bar{a}$c $\oplus$ bc, P = a$\bar{b}$ $\oplus$ ac $\oplus$ $\bar{b}$c. Where $\bar{a}$ is a negation of variable a. Function P is a standard majority and Q, R are majorities on negated input arguments a and b, respectively [YZL03]. We realized this function with quantum primitives, found it useful in other designs and thus worthy to be a stand-alone 3×3 quantum gate. We call it the Miller gate [YZL03] and we found a solution that is less expensive than that from [Mil02].

Our realization of the Miller gate requires 4 Feynman gates and a Toffoli gate [LPG$^+$03] (Figure 2.62a), which would suggest a cost of 4*5 + 25 = 45. Performing transformations as in Figure 2.62b, we obtain a solution with cost 35. Another solution obtained by the same method has cost 35 and is shown in Figure 2.62c. It is also based on simple EXOR transformation (x$\oplus$y) $\oplus$ z = (x$\oplus$z) $\oplus$ y applied to three rightmost Feynman gates from Figure 2.62a, with EXOR in the middle wire y. Again, the Miller gate, based on its binary logic equations, looks initially much more complicated than the Toffoli gate, but a closer inspection using quantum

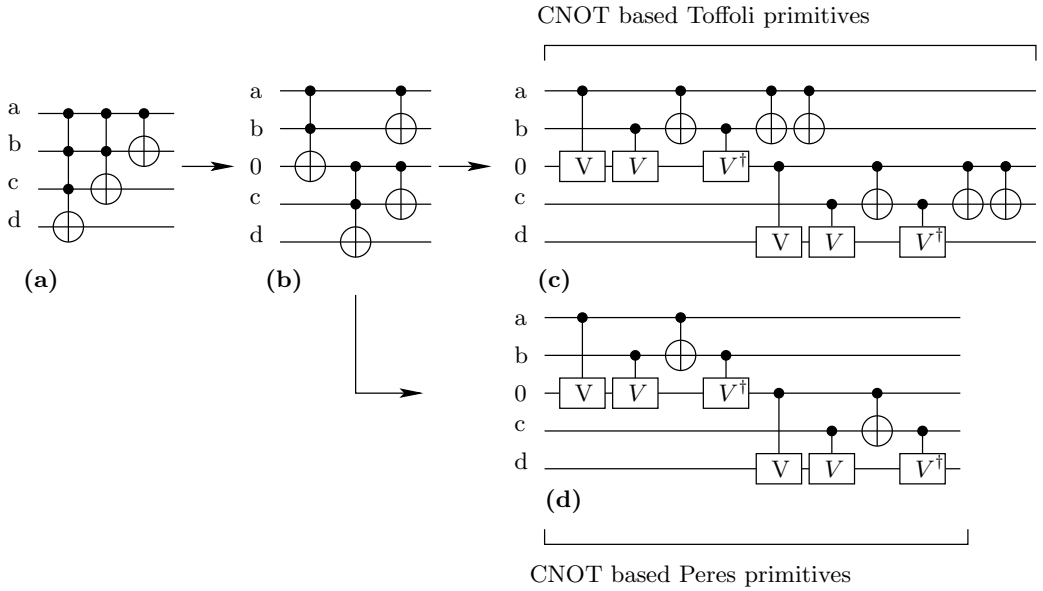


Figure 2.63: Example of comparison of synthesis using Toffoli and Peres primitives.

logic primitives proves that it is just slightly more expensive. Observe that these minimization methods are NMR technology related, as for instance in the quantum dot reversible technology with no ancilla bits the Miller gate is the least expensive gate. Also, remark that similar rules and software can be used for other quantum technologies when the basic gates are known.

Finally observe that the main reason for searching for novel quantum primitives is the minimization of large quantum circuits. For instance Figure 2.63 shows the difference of cost while building a larger circuit with Toffoli- and Peres-type primitives. Observe that when built with Toffoli primitives the resulting cost is 12 $2*2$ quantum gates while using the Peres primitives the cost is only 8 $2 * 2$ quantum gates.

2.7.1 Local, Quantum Gate-Optimizing Transformations

The transformations to optimize quantum circuits are grouped in 12 transformation sets. There are the following sets:

1. S1. 1-qubit transformations,
2. S2. 2-qubit transformations,
3. S3. 3-qubit transformations,

4. S4. 4-qubit transformations,
5. S5. n-qubit transformations,
6. S6. Ternary transformations,
7. S7. Mixed binary/ternary transformations,
8. S8. Macro-generations,
9. S9. Macro-cell creations,
10. S10. Peres Base transformations,
11. S11. Toffoli Base transformations,
12. S12. Controlled-V Base transformations,
13. S13. Input/Output permutting transformations.

Many transformations are shared between sets. In addition, in each of the above base sets, there are subsets to be chosen for any particular run of the optimizer program. Most of them are taken from [Mil02, MD03, DM03, IKY02, SPH02, KL00, Lom03] but some other are based on our research or general quantum literature. Different groups of transformations are used in various stages of circuit optimization.

The 1-qubit transformations are related to 1-qubit gates (Figures 2.64, 2.65, 2.66). They can precede and also follow the 2-qubit, 3-qubit and other transformations. The 2-qubit transformations are for 2-qubit circuits or 2-qubit subcircuits of larger quantum circuits, similarly the 3-qubit transformations are for 3-qubit circuits or subcircuits of larger circuits (Figure 2.67). The n-qubit transformations are general transformation patterns applicable to circuits with more than 3 qubits. They are less computationally efficient and they use internally transforms S1 - S4. Macro-generations are transformations that convert higher order gates such as Fredkin, Margolus, DeVos or Kerntopf gates to standard bases. Macro-cell creations from set S9 are inverse to those from set S8.

There are three standard bases of transformations: Toffoli Base, Peres Base, and Controlled-V Base. In Toffoli Base all permutation gates are converted to X (i.e. NOT), 3-qubit Toffoli and 2-qubit Feynman gates. This is the standard synthesis base used by all other authors in literature [Mil02, MD03, DM03, IKY02, SPH02, NC00]. The Peres Base has been introduced originally by us based on the observation of superiority of this base in NMR realizations (and perhaps other technologies as well). It includes only X, 3-qubit Peres and 2-qubit Feynman gates.

Controlled-V Base is another new base that is very useful to synthesize new low-cost permutation gates from truly quantum primitives of limited type. This base includes Controlled-V, Controlled-V†, V, V†, X and Feynman gates. In all bases the gates like Feynman, Toffoli, Controlled-V are stored in all possible permutations of quantum wires. Thus in 2-qubit base there are "Feynman EXOR up" and "Feynman EXOR down" gates and transformations respective to each of these gates. For simplification, in the tables below only some of the transformations are shown, for instance related only to "Feynman EXOR down" or "Toffoli EXOR down gates". Other transformations are completely analogous. The output permutting transformations lead in principle to a circuit that has an unitary matrix which is different from the original unitary matrix. Observe that each transformation can be applied forward or backward, so the software should have some mechanisms to avoid infinite loops of transformations. The matrix of the new circuit is the matrix of the original circuit with permuted output signals. In some applications the order of output functions is not important, so if the circuit is simplified by changing the output order, the output permutting transformation is applied.

In addition to operators defined earlier, we define now the following operators:

X,Y,Z defined earlier are Pauli spin matrices and $X(\Phi)$,$Y(\Phi)$, $Z(\Phi)$ are the corresponding angle-parameterized matrices, giving rotations on the Bloch sphere [NC00]. P is a phase rotation by $\Phi/2$ to help match identities automatically [Lom03].

The transformation software operates on sequences of symbols that represent gates and their parts. Symbol * is used to create sequences from subsequences. This symbol thus separates two serially connected gates or blocks. Numerically, it corresponds to standard matrix multiplication. Gate symbols within a parallel block may be separated by spaces, but it is necessary only if lack of space will lead to a confusion, otherwise space symbol can be omitted. So for instance symbols of macro-cells should be separated by spaces. There are four types of symbols: simple, rotational, parameterized and controlled. Simple symbols are just names (here - single characters, in software - character strings). The names of simple symbols (used also in other types of symbols) are the following: D - standard control point (a black dot in an array), E - control with negated input, F - control with negated output, G - control with negated input and output, X - Pauli-X, Y - Pauli-Y, Z - Pauli-Z, H - Hadamard, S - phase, T - $\Phi/8$ (although $\Phi/4$ appears in it). Symbols A, B and C are auxiliary symbols that can match several gate symbol definitions. They do not correspond to any particular gates but to groups of gates and are useful to decrease the set of rules and thus to speed-up transformations. Other simple symbols will be explained below. As we see, characters are used here not only for gates but also for parts of gates, such as D - standard control used in gates. Thus we can combine these symbols to create gate descriptions: DX (Feynman with EXOR down), XD (Feynman with EXOR up), DDX (Toffoli or Toffoli with EXOR

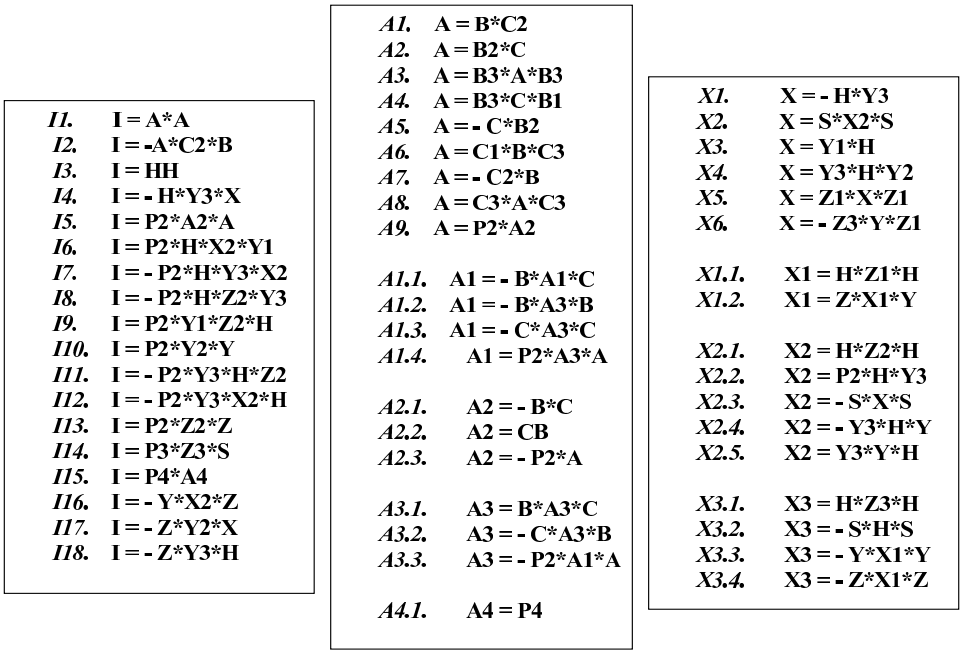


Figure 2.64: 1-qubit transformations for I, A and X groups.

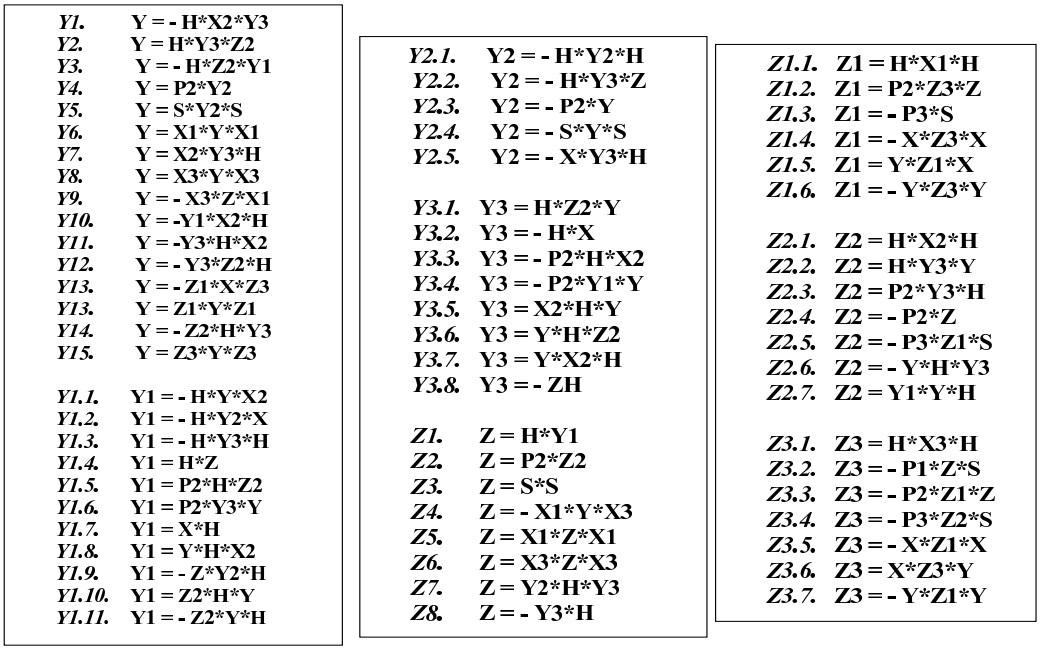


Figure 2.65: 1-qubit transformations for Y and Z groups.

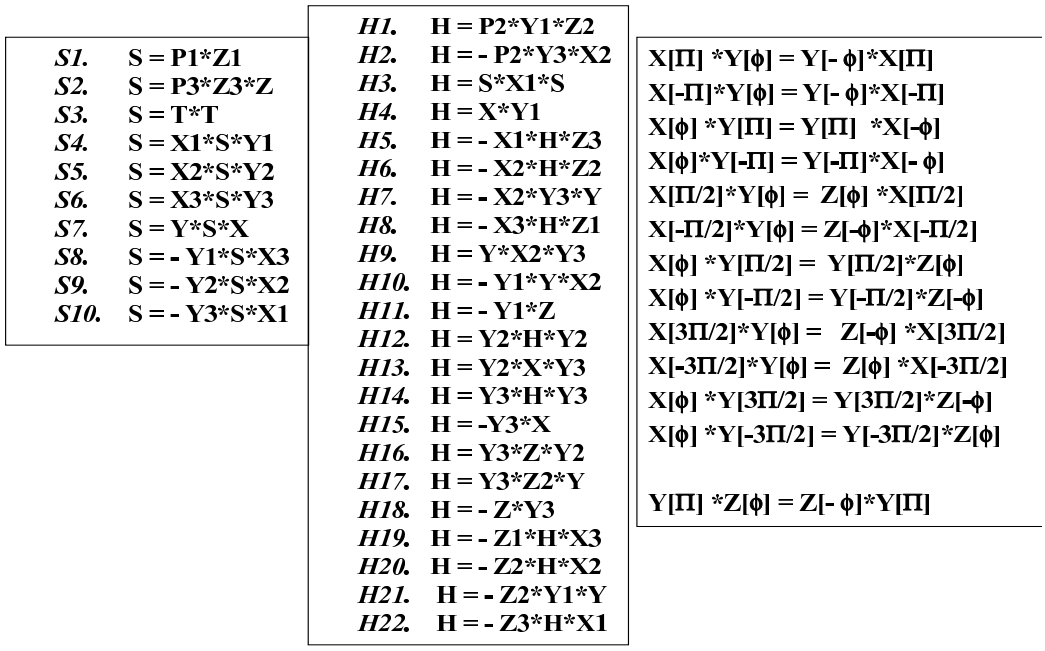


Figure 2.66: 1-qubit transformations for S, H and parameterized rotation groups.

down), DXD (Toffoli with EXOR in middle), XDD (Toffoli with EXOR up), and so on. Names of macro-cells are for instance: FE (Feyman), TOD (Toffoli with EXOR down), SW (swap), FRU (Fredkin controlled with upper wire), MA (Margolus), KED (Kerntopf with Shannon expansion in lowest wire), etc. Symbols like ϕ, ψ, π denote angles and other parameters. Parameterized symbols have the syntax:

simple_name $[parameter_1, \cdots, parameter_n]$, where $parameter_i$ are parameters. For instance, $X[4\Pi/8]$, $Y[3\Pi/2]$, $Z[\Phi]$, etc. Rotational symbols are composed of the (simple) rotation operator symbol such as X, Y, Z, or P, and a number. $X[\Phi_i]$, $Y[\Phi_i]$, $Z[\phi_i]$, $P[\Phi_i]$, where $\Phi_i = 4\Phi/8 * i$, $i = 1, 2, \ldots, 7$. We assume here that all rotational operators have period 4Π and equal identity when their argument is 0, thus the choices for Φ_I . In these operators the notation is like this, $X_r = X[4\Pi/8 * r]$, $r = 1, 2, \ldots, 7$, and so on for Y_r, Z_r and P_r. Controlled symbols have the syntax *simple_name [sequence of simple,rotational or parameterized symbols]*

Example 2.7.1.1
$D[X * X]$ is a symbol of a controlled gate that is created from two subsequent Feynman gates. Observe that $DX * CX = D[X * X] = D[I] = II$, which means that two controlled-NOT gates in sequence are replaced by two parallel quantum wires denoted by I I.

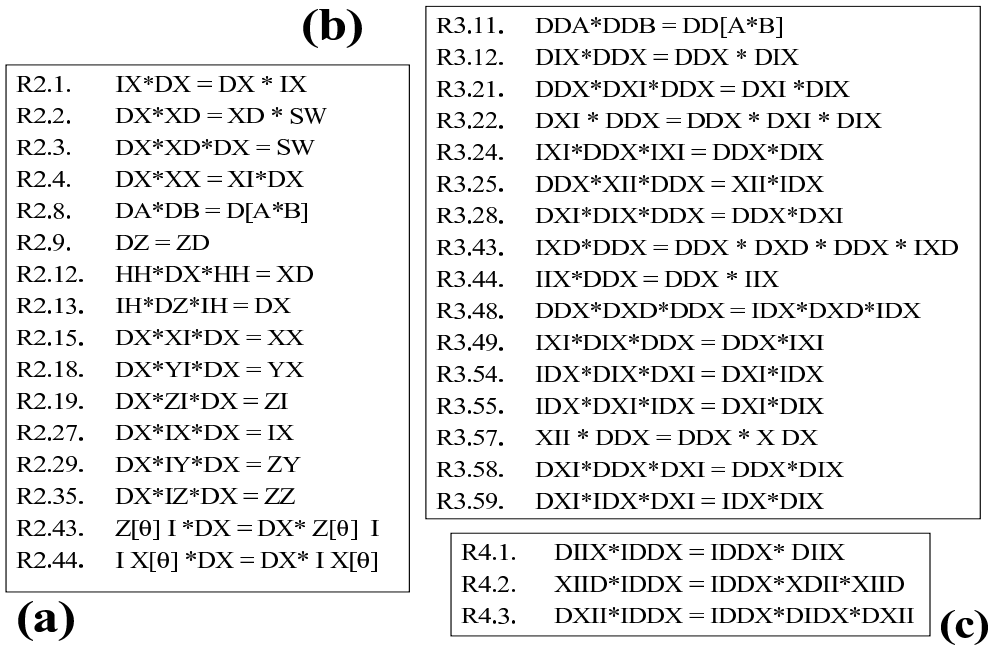


Figure 2.67: Examples of 2-qubit and 3-qubit transformations: (a) 2-qubit transformations, (b) 3-qubit transformations; observe a space between X and DX in rule R3.57 that signifies that D control the lower X, (c) 4-qubit transformations

Example 2.7.1.2
$D[S * T * H]$ is a sequence $S * T * H$ controlled by single-qubit.

Example 2.7.1.3
$DD[X1 * Y2 * Y2]$ is a sequence of rotational gates controlled by a logic AND of two-qubits (this is a generalization of a Toffoli gate).

Observe that controlled symbols are created only as a transitional step during the optimization process - such gates do not physically exist. Using the concept of controlled symbols, n-qubit circuits can be optimized using 1-qubit transformations without duplicating all the 1-qubit identity rules. Similarly the parameterized and rotational symbols allow the reduction and hierarchization of the set of rules, which causes more efficient and effective run of the optimization software.

The simplified algorithm SA for performing rule-based optimization of 3-qubit quantum arrays is the following:

1. 1.Apply all the 1-qubit transformations, until no more applications of such rules becomes possible.
2. 2.Apply all the 2-qubit transformations and 1-qubit transformations induced by them (for instance using the controlled symbols).
3. 3.Apply all the 3-qubit transformations until possible.
4. 4.Iterate steps 1,2 and 3 until no changes in the circuit.
5. 5.Apply inverse transformations that locally optimize the array.
6. 6.Repeat steps 1,2,3,4 until possible.
7. 7.Apply inverse transformations that do not worsen the cost of the array.
8. 8.Repeat steps 1,2,3,4 until possible.

Several similar variants of this heuristic algorithm can be created. In general, none of these versions gives a warranty of the optimal or even sub-optimal solution, as known from the theory of Post/Markov algorithms.

As an example, we present 1-qubit transformation algorithm A1q:

1. A1. Combine modulo-8 the same types of rotational operators P, X, Y, Z.
   - For instance $X2 * X3$ becomes $X5$, $X2 * X3 * X3$ becomes I, and $Y3 * Y3 * Y3$ becomes $I * Y1 = Y1$.

2. A2. Apply directly applicable rules that do not include symbols A and B.

3. A3. Iterate 1 and 2 until no more changes possible.

4. A4. Starting from the left of the sequence, find a grouping pattern such as $A2 = -BC$

5. A5. Substitute symbols X,Y,Z for A,B and C from the pattern.

6. A6. Apply in forward directions the standard simplifying transformations such as $I * A = A$

7. A7. Repeat steps $A1$ to $A6$ until possible.

Example 2.7.1.4
Given is an identity $Y = -X * Y * X$. Let us try to verify this identity using algorithm A1q. We have therefore to simplify the sequence $X * Y * X$. (A1) There are no patterns of the same rotational operators to combine, (A2) There are no directly applicable rules, (A4) We take pattern $-X * Y$ and match it with the rule $A2 = C * B$. This leads to $-Z2 * X$. (A1) no, (A2) no, (A3) no, (A4) we find pattern $A2 = CB$ which leads to $-Y * X * X$. (A6) $X * X$ is replaced with I, $Y * I$ is replaced with Y. No further optimization steps are possible, so the sequence was simplified to $-Y$, proving that $Y = -Y * Y * X$.

Example 2.7.1.5
Simplify HXH. (A2) Use rule $H = X * Y1$ twice. This leads to $X * Y1 * X * X * Y1$. (A2) Use rule $I = AA$ in reverse direction. This leads to $X * Y1 * Y1$. (A1) $Y1 * Y1$ is replaced by Y2. This leads to $X * Y2$. (A4) Use pattern $A = B * C2$. This leads to Z. No further optimization steps are possible. Thus we proved that $HXH = Z$. More optimization examples of algorithm SA will be given in the sequel.

In our runs of GA we look for solutions with the accuracy of: (1) permutation of quantum wires, (2) permutation of inputs, (3) permutation of outputs (transformation group S13). In addition we can also generate solution sets with accuracy of inverting input, output or input/output signals (the NPN classification equivalent circuits). Therefore, for each unitary matrix we generate therefore many logically equivalent solutions. We can generate solution sets also for the same set of Boolean functions, or for the same NPN classification class. One interesting aspect of such approach is that one can create new local equivalence transformations for circuits in each of these classes. Finding these transformations and applying them exhaustively to particularly interesting gates leads to levellized "onion-like" structures of gates, as the one shown in Figure 2.56. This Figure shows the layered structure of gates created by adding only Feynman gates to a seed composed of other gate types, in

this case a Peres gate. The additional gates created by the so-called PTF principle are the Toffoli, Fredkin and Miller gates. These transformations are used to find efficient realizations of new gates from known gate realizations.

Bibliography

[ALT08] M. H. S. Amin, P.J. Love, and C.J.S. Truncik. Thermally assisted adiabatic quantum computation. *Phys. Rev. Lett.*, 100:060503, 2008.

[AOR+02] M.H.S. Amin, A.N. Omelyanchouk, S.N. Rashkeev, M. Coury, and A.M. Zagoskin. Quasiclassical theory of spontaneous currents at surfaces and interfaces of d-wave superconductors. *Physica B*, 318:162, 2002.

[AS04] M.H.S. Amin and A.Y. Smirnov. Quasiparticle decoherence in d-wave superconducting qubits. *Phys. Rev. Lett.*, 92:017001, 2004.

[BBC+95] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and Weinfurter H. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995.

[Ben82] P. Benioff. Quantum mechanical Hamiltonian models of Turing machines. *Journal of Statistical Physics*, 29(3):515–546, 1982.

[Boh08] N. Bohr. *Niels Bohr Collected Works*. Springer, 2008.

[BZ00] A. Blais and A. M. Zagoskin. Operation of universal gates in a solid state quantum computer based on clean josephson junctions between d-wave superconductors. *Phys. Rev. A*, 61:042308, 2000.

[Cas] D. Cassidy. Werner heisenberg: A bibliography of his writings, 1922-1929, expanded edition.

[CFH97] D. G. Cory, A. F. Fahmy, and T. F. Havel. Ensemble quantum computing by NMR spectroscopy. *Proc. Natl. Acad. Sci. USA*, 94:1634–1639, 1997.

[CLRS01] T.H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *The subset-sum problem*, page 1043. MIT Press, McGraw-Hill, 2001.

[CM04] E. Curtis and Perkowski M. Transformation based algorithm for ternary reversible logic synthesis using universally controlled ternary gates, 2004.

[CZ95] J.I. Cirac and P. Zoller. Quantum computation with cold trapped ions. *Physical Review letters*, 74(20):4091, 1995.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.

[Dir84] P. A. M. Dirac. *The principles of quantum mechanics*. Clarendon, Oxford, 1984.

[Dir95] P.A.M. Dirac. *The collected works of P A M Dirac : 1924-1948, R H Dalitz (ed.)*. Cambridge, 1995.

[DiV95] P. DiVincenzo. Two-bit gate for quantum computation. *Physical Review A*, 50:1015, 1995.

[DKK03] L.M. Duan, A. Kuzmich, and H.J. Kimble. Cavity QED and quantum-information processing with 'hot' trapped atoms. *Physical Review A*, 67:032305, 2003.

[DM94] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[DM03] G.W Dueck and D. Maslov. Garbage in reversible designs of multiple-output functions. In *Proceedings of RM 2003*, pages 162–170, 2003.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47(10):777–780, May 1935.

[Fey85] R. P. Feynmann. Quantum mechanical computers. *Optic News*, 11:11, 1985.

[FTR07] K. Fazel, M.A. Thornton, and J.E. Rice. Esop-based toffoli gate cascade generation. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing,*, pages 206 – 209, 2007.

[GAJ06] P. Gupta, A. Agrawal, and N.K. Jha. An algorithm for synthesis of reversible logic circuits. *IEEE Trans. on CAD*, 25(11):2317–2330, 2006.

[GC97] N. A. Gershenfeld and I. L. Chuang. Bulk spin-resonance quantum computation. *Science*, 275(5298):350 – 356, 1997.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter Computers and Intractability: A Guide to the Theory of NP-Completeness, page 247. W. H. Freeman, 1979.

[Gra81] A. Graham. *Kronecker Products and Matrix Calculus With Applications*. Ellis Horwood Limited, Chichester, U.K., 1981.

[Gru99] J. Gruska. *Quantum computing*. Osborne/McGraw-Hill,U.S., 1999.

[HJL+10] R. Harris, M.W. Johnson, T. Lanting, A.J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, S. Rich, C.and Uchaikin, M.C. Thom, E.M. Chapple, J. Wang, B. Wilson, M.H.S. Amin, N. Dickson, K. Karimi, B. Macready, C.J.S. Truncik, and G. Rose. Experimental investigation of an eight qubit unit cell in a superconducting optimization processor. *Phys. Rev. B*, 82:024511, 2010.

[HSY+04] W.N.N. Hung, X. Song, G. Yang, J. Yang, and M Perkowski. Quantum logic synthesis by symbolic reachability analysis. In *Proceedings of DAC*, 2004.

[HSY+06] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M Perkowski. Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and systems*, 25(9):1652–1663, 2006.

[IKY02] K. Iwama, Y. Kambayashi, and S. Yamashita. Transformation rules for designing cnot-based quantum circuits. In *Proceedings of DAC 2002*, pages 419–424, New Orleans, Louisiana, 2002.

[Ing76] R.S. Ingarden. Quantum information theory. *Reports on Mathematical Physics*, 10(1):43–72, 1976.

[JHM98] A. Jones, R. Hansen, and M. Mosca. Quantum logic gates and nuclear magnetic resonance pulse sequences. *Journal of Magnetic Resonance*, 135(2):353–360, 1998.

[JM98] A. Jones and M. Mosca. Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer. *Journal of Chemical Physics*, 109:16481653, 1998.

[KL00] J. Kim and S. Lee, J-S.and Lee. Implementation of the refined deutsch-jozsa algorithm on a three-bit nmr quantum computer. *Physical Review A*, 62, 2000.

[Koz92] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[Koz94] J.R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, 1994.

[KP06] F. S. Khan and M. A. Perkowski. Synthesis of multi-qudit hybrid and d-valued quantum logic circuits by decomposition. *Theoretical Computer Science*, 3(367):336–346, 2006.

[KPK02] A. Khlopotine, M. Perkowski, and P. Kerntopf. Reversible logic synthesis by gate composition. In *Proceedings of IWLS*, pages 261–266, 2002.

[LBA+08] B. P. Lanyon, M. Barbieri, M. P. Almeida, T. Jennewein, T. C. Ralph, K. J. Resch, G. J. Pryde, J. L. O'Brien, A. Gilchrist, and A. G. White. Quantum computing using shortcuts through higher dimensions, April 2008.

[LBMW03] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland. Quantum dynamics of single trapped ions. *reviews of Modern Physics*, 75(1):281324, 2003.

[LKBP06] S. Lee, S.J. Kim, J. Biamonte, and M. Perkowski. The cost of quantum gate primitives. *Journal of Multiple-Valued Logic and Soft Computing*, 12(5-6):561–574, 2006.

[Lom03] Ch. Lomont. Quantum circuit identities, 16 July 2003.

[LP02] M. Lukac and M. Perkowski. Evolving quantum circuit using genetic algorithm. In *Proceedings of the 2002 NASA/DoD Conference on Evolvable hardware*, pages 177–185, 2002.

[LP05a] M. Lukac and M. Perkowski. Combining evolutionary and exhaustive search to find the least expensive quantum circuits. In *Proceedings of ULSI symposium*, 2005.

[LP05b] M. Lukac and M. Perkowski. Using exhaustive search for the discovery of a new family of optimum universal permutative binary quantum gates. In *Proceedings of International Workshop on Logic & Synthesis, Poster Session*, 2005.

[LP07] M. Lukac and M. Perkowski. Quantum mechanical model of emotional robot behaviors. In *Proceedings of the International Symposium on Multiple-Valued Logic*, 2007.

[LPG+03] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, C. H. Yu, K. Chung, H. Jee, B.-G. Kim, and Y.-D. Kim. Evolutionary approach to quantum reversible circuit synthesis. *Artif. Intell. Review.*, 20(3-4):361–417, 2003.

[LPG+04] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, C. H. Yu, K. Chung, H. Jee, B-G. Kim, and Y-D. Kim. Evolutionary approach to quantum and reversible circuits synthesis. In *Artificial Intelligence in Logic Design*, pages 201 – 257. Kluwer Academic Publisher, 2004.

[LPK10] M. Lukac, M. Perkowski, and M. Kameyama. Evolutionary quantum logic synthesis of boolean reversible logic circuits embedded in ternary quantum space using structural restrictions. In *Proceedings of the WCCI 2010*, 2010.

[LPMP02] M. Lukac, M. Pivtoraiko, A. Mishchenko, and M. Perkowski. Automated synthesis of generalized reversible cascades using genetic algorithms. In *Proceedings of Fifth Intern. Workshop on Boolean Problems*, pages 33–45, 2002.

[LSKed] M. Lukac, A. Sasaki, and M. Kameyama. Cellular automata based robotics architecture for behavioral decision making, to be published.

[Luk09] M. Lukac. *Quantum Logic Synthesis and Inductive Machine Learning, Ph.D. dissertation*. PhD thesis, Portland State University, 2009.

[Man80] Y. Manin. Computable and uncomputable. *Moscow: Sovetskoye Radio.*, -:–, 1980.

[MC] T.S. Metodi and F.T. Chong. *Quantum Computing for Computer Architects*.

[MD03] D. M. Miller and G.W. Dueck. Spectral techniques for reversible logic synthesis. In *Proc. RM*, pages 56–62, 2003.

[MDM05] D. Maslov, G. W. Dueck, and D. M. Miller. Synthesis of Fredkin-Toffoli reversible networks. *IEEE Transactions on VLSI*, 13(6):765–769, 2005.

[MDM07] D. Maslov, G. W. Dueck, and D. M. Miller. Techniques for the synthesis of reversible Toffoli networks. *ACM Trans. Des. Autom. Electron. Syst.*, 12(4):42, 2007.

[Mil02] D. M. Miller. Spectral and two-place decomposition techniques in reversible logic. In *Proc. Midwest Symposium on Circuits and Systems, on CD-ROM*, August 2002.

[MM06] D. M. Miller and A. T. Mitchell. QMDD: A decision diagram structure for reversible and quantum circuits. In *Proc. 2006 Int. Symposium on Multiple-Valued Logic*, 2006.

[MMD03] D.M. Miller, D. Maslov, and G.W. Dueck. A transformation based algorithm for reversible logic synthesis. In *Proceedings of DAC*, 2003.

[MMD06] D. M. Miller, D. Maslov, and G. W. Dueck. Synthesis of quantum multiple-valued circuits. *Journal of Multiple-Valued Logic and Soft Computing*, 12(5-6):431–450, 2006.

[MMK+95] C. Monroe, D. M. Meekhof, B. E. King, W.M. Itano, and D.J. Wineland. Demonstration of a fundamental quantum logic gate. *Physical Review Letters*, 75:4717–4717, 1995.

[MMKW96] C. Monroe, D. M. Meekhof, B. E. King, and D. J. Wineland. A schrödinger cat superposition state of an atom. *Science*, 272, 1996.

[MML+98] C. Monroe, B. E. Meekhof, D. M. ind King, D. Leibriefd, W. M. Itano, and D. J. Wineland. Manipulating the motion of a single trapped atom. *Acc. Chem. Res.*, 29(12):585590, 1998.

[MOC02] Frederic T. Chong Mark Oskin and Isaac Chuang. A practical architecture for reliable quantum computers. *IEEE Computer*, January 2002:79–87, 2002.

[Moo65] G.E. Moore. Cramming more components onto integrated circuits. In *Electronics, April 19*, 1965.

[MP02] A. Mischenko and M. Perkowski. Logic synthesis of reversible wave cascades. In *Proceedings of IWLS*, pages 197–202, 2002.

[MV76] P.M. Mathews and K. Venkatesan. *A textbook of quantum mechanics*. Tata McGraw-Hill, 1976.

[MWD10] M.D. Miller, R. Wille, and R. Drechsler. Reducing reversible circuit cost by adding lines. In *Proceedings of the ISMVL*, 2010.

[NC97] M. A. Nielsen and I. L. Chuang. Programmable quantum gate arrays. *Phys. Rev. Lett.*, 79:321 – 324, 1997.

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

[PARK+01] M. Perkowski, A. Al-Rabadi, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, M. Md. Azad Khan, A. Coppola, S. Yanushkevich, V. Shmerko, and L. Jozwiak. A general decomposition for reversible logic. In *Proc. RM'2001*, August 2001.

[Pau90] W. Paul. Electromagnetic traps for charged and neutral particles. *Rev. Mod. Phys.*, 62(3):531540, 1990.

[Per85] A. Peres. Reversible logic and quantum computers. *Phys. Rev. A*, 32(6):3266–3276, 1985.

[Per00] A. Peres. Reversible logic and quantum computers. *Physical review*, (32):3266–3276, 2000.

[Pla] M. Planck. Ueber das gesetz der energieverteilung im normalspectrum. *Ann. Physics*, 309(3):553–63. English translation: On the Law of Distribution of Energy in the Normal Spectrum.

[PLKK10] M. Perkowski, M. Lukac, P. Kerntopf, and M. Kameyama. Gpu library based approach to quantum logic synthesis. In *RC workshop*, 2010.

[PLSK11] P. Perkowski, M. Lukac, D. Shah, and M. Kameyama. Synthesis of quantum circuits in linear nearest neighbormodel using positive davio lattices. *facta Universitatis*, 24, 2011.

[Pop75] R.P Poplavskii. Thermodynamical models of information processing. *(in Russian). Uspekhi Fizicheskikh Nauk*, 115(3):465501, 1975.

[PW02] J. Pachos and H. Walther. Quantum computation with trapped ions in an optical cavity. *Physical Review Letters*, 89(18), 2002.

[RFW+07] A. Raghuvanshi, Y. Fan, M. Woyke, A. Kumar, and M. Perkowski. Quantum robots for teenagers. In *Proceedings of the International Symposium on Multiple-Valued Logic 2007*, 2007.

[Rub00] B. Rubinstein. *Evolving Quantum Circuits using Genetic Programming*, pages 325–334. Stanford University, 2000.

[Rub01] B.I.P. Rubinstein. Evolving quantum circuits using genetic programming. In *Congress on Evolutionary Computation (CEC2001)*, pages 114–121, 2001.

[SBM05a] V. V. Shende, S. S. Bullock, and I. L. Markov. A practical top-down approach to quantum circuit synthesis. In *Proceedings of Asia Pacific DAC*, 2005.

[SBM05b] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of quantum logic circuits. In *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*, pages 272–275, New York, NY, USA, 2005. ACM Press.

[Sch26] E. Schrödinger. Quantisierung als eigenwertproblem. *Annalen der Physik*, 79(361), 1926.

[SD96] J. Smolin and D. P. DiVincenzo. Five two-qubit gates are sufficient to implement the quantum fredkin gate. *Physical Review A*, 53(4):2855–2856, 1996.

[Sho94] P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35nd Annual Symposium on Foundations of Computer Science (Shafi Goldwasser, ed.)*, pages 124–134. IEEE Computer Society Press, 1994.

[SO02] Steven Swanson and Mark Oskin. Towards a universal building block of molecular and silicon computation. In *Workhop on Non-Silicon Computing, held in Conjunction with the International Symposium on High Performance Computer Architecture*, 2002.

[SPH02] V. V. Shende, I. L. Prasad, A.K.and Markov, and J.P. Hayes. Reversible logic circuit synthesis. In *Proceedings of 11th IEEE?ACM Intern. Workshop on Logic Synthesis (IWLS)*, pages 125–130, 2002.

[SPIH03] V.V. Shende, A.K. Prasad, Markov I.L., and J.P. Hayes. Synthesis of reversible logic circuits. 22(710), 2003.

[Ste97] A. Steane. The ion trap quantum information processor. 64(623), 1997.

[Sty02] D.F. et al. Styer. Nine formulations of quantum mechanics. 70(288), 2002.

[SZSS10] M. Saeedi, M. S. Zamani, M. Sedighi, and Z. Sasanian. Synthesis of Reversible Circuit Using Cycle-Based Approach. *ACM Journal of Emerging Technologies in Computing Systems*, 2010.

[vdPIG+06] S.H.W. van der Ploeg, A. Izmalkov, M. Grajcar, U. Huebner, S. Linzen, S. Uchaikin, Th. Wagner, A.Y. Smirnov, A.M. van den Brink, M.H.S. Amin, A.M. Zagoskin, E. Il'ichev, and H.-G. Meyer. Adiabatic quantum

computation with flux qubits, first experimental results. *IEEE Trans. App. Supercond.*, 17,:113, 2006.

[WBB+02] DJ Wineland, M. Barrett, J. Britton, J. Chiaverini, B. DeMarco, WM Itano, B. Jelenkovi'c, C. Langer, D. Leibfried, V. Meyer, et al. Quantum information processing with trapped ions, 2002.

[Wey32] H. Weyl. *The Theory of Groups and Quantum Mechanics,*. Dover Publications, 1932.

[WG98] C. Williams and A. Gray. Automated design of quantum circuits. In *in Proceedings of QCQC 1998*, pages 113–125, 1998.

[WGMD09] R. Wille, D. Große, D.M. Miller, and R. Dreschler. Equivalence checking of reversible circuits. In *Proceedings of the ISMVL*, 2009.

[WH04] D. Wineland and T. Heinrichs. Ion trap approaches to quantum information processing and quantum computing. *A Quantum Information Science and Technology Roadmap*, N/A, 2004.

[WMI+98] D. J. Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, , and D. M. Meekhof. Experimental issues in coherent quantum-state manipulation of trapped atomic ions. *Journal of Research of the National Institute of Standards and Technology*, 103(259):259, 1998.

[WMI+05] D. Wineland, C. Monroe, W. Itano, B. King, D. Leibfried, D. Meekhof, C. Myatt, and C. Wood. Experimental primer on the trapped ion quantum computer. In *Quantum Computing: Where Do We Want to Go Tomorrow? (ed S. L. Braunstein)*. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, FRG., 2005.

[Yab00] T. Yabuki. Genetic algorithms for quantum circuit design – evolving a simpler teleportation circuit –. In *In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, 2000.

[YHSP05] G. Yang, W.N.N. Hung, X. Song, and M. Perkowski. Majority-based reversible logic gates. *Theoretical Computer Science*, 334(1-3), 2005.

[You95] S. Youssef. Quantum mechanics as an exotic probability theory. In *Workshop on Maximum Entropy and Bayesian Methods, St.John's College, Santa Fe, New Mexico*, August 1995.

[YSPH05] G. Yang, X. Song, M. Perkowski, and W. N. N. Hung. The power of large pulse-optimized quantum libraries: Every 3-qubit reversible function can be realized with at most four levels. In *Proceedings of Proc. of IWLS*, 2005.

[YSPW05] G. Yang, X. Song, M. Perkowski, and J. Wu. Realizing ternary quantum switching networks without ancilla bits. *Journal of Physics A, Mathematical and General*, 38:9689–9697, 2005.

[YZL03] W.J. Yang, Y. Zhou, and K.T. Lau. Low power adiabatic programmable logic array with apdl-2. *ELECTRONICS LETTERS*, 39(21):2, 2003.