Calculation and Minimisation of Garbage Bits in Quantum Reversible Circuits

A thesis
submitted in partial fulfillment of the requirements
for the award of degree
of

Master of Engineering
in
Software Engineering

Under the Supervision of
Mr. Amardeep Singh
Senior Lecturer
CSED, TIET, Patiala.

Submitted By
Rajeev Tanwar
(Roll No. 8033115)

Computer Science & Engineering Department
Thapar Institute of Engineering & Technology
(Deemed University), Patiala-147004

May 2005

Abstract

Quantum Computation and Quantum Information is the study of information processing task that can be accomplished using Quantum mechanical systems. It is relatively new and emerging area in the field of computing that taught us to think physically about computation. This approach yields many new and exciting capabilities for information processing and communication. Quantum Computing will be a total change in how the computer will operate and function. The explorations in this field may one day result in information processing devices with capabilities far beyond today's computing and communication systems.

Reversible Computation is a fundamental aspect necessary for Quantum Computation. The overall advantage of making the computation reversible on a Classical Computer is that heat dissipation is reduced and new techniques like Quantum Computing can be approached with this foundational change in the way processing is done. Such combination of Classical and Quantum Computers would be indispensable, since our perception is classical.

The primary objective of this thesis is to gain insight into the Reversible Computation and its use in Quantum Computing and to study Quantum Information Processing. However the main task that was identified to be pursued as a focal issue around which the concepts of Reversible and Quantum Computation were to be built is:

Determining the function of increase in the number of "ancilla" bits and "garbage" bits in reversible computing in comparison with conventional performance with respect to the implementation of basic circuits. Also to find out how reversible circuit can be optimised in terms of number of gates, garbage bits and logical complexity.

Declaration

I hereby certify that the work which is being presented in the thesis entitled, **"Calculation and Minimisation of Garbage Bits in Quantum Reversible Circuits",** in partial fulfillment of the requirements for the award degree of Master of Engineering in Software Engineering at Computer Science and Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Mr. Amardeep Singh.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other University.

Rajeev Tanwar

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Mr. Amardeep Singh)
Senior Lecturer,
Computer Science and Engineering Department,
Thapar Institute of Engineering and Technology,
Patiala- 147004.

Countersigned by

(Mr. R.S. Salaria)
Assistant Professor & Head,
Computer Science and Engineering Department,
Thapar Institute of Engineering and Technology,
Patiala - 147004.

(Dr. D. S. Bawa)
Dean of Academic Affairs,
Thapar Institute of Engineering
and Technology,
Patiala - 147004.

Acknowledgement

It is my proud privilege to express my regards and sincere gratitude to **Mr. Amardeep Singh, Senior Lecturer, Computer Science and Engineering Department, TIET, Patiala,** for his valuable suggestions and expert guidance.

I also take the opportunity to thank **Mr. R. S. Salaria, Assistant Professor & HOD, Computer Science and Engineering Department, TIET, Patiala,** and the entire faculty and staff for their help, inspiration and moral support, which went a long way in successful completion of my thesis work.

Other than this the Internet has been my constant source of information on the ongoing research in the field of Quantum and Reversible Computing, because without such a medium it would have been almost impossible for amateur enthusiasts in this field to update themselves and benefit from it.

The existence of researchers and authors are an obvious asset to the scientific world whose efforts are used and multiplied by those having common interests, which leads to astounding growth due to the diverse ability of human beings which compliment each other to deliver. This once again reiterates the unity in diversity in every aspect.

To God Almighty whose grace is the reason for the good that I do and the achievement of which is the purpose of my life.

Rajeev Tanwar
Roll No. 8033115
M.E. II Yr
Software Engineering

Contents

List of Figures

List of Tables

Organization of Thesis

The primary objective of the thesis entitled," **Calculation and Minimisation of Garbage Bits in Quantum Reversible Circuits**", is to study Reversible Computation and its use in its use in Quantum Computing and Quantum Information Processing. Organization of the work is described as under:

The **First Chapter** gives introduction to Quantum Computing and provides an insight into how it differs from other computational models and the promises it holds. Quantum Computation and Quantum Information provide a useful series of challenges at varied levels of difficulty for people devising methods to better manipulate Quantum Systems, and simulate the development of new techniques.

The **Second Chapter** of the thesis is devoted to the concept of Reversible Computing. It explains theory behind reversible computing and its need in current scenario. The work on classical, reversible computation has laid the foundation for the development of Quantum mechanical computers. Reversible design methods might give rise to better Quantum circuit construction, resulting in much more powerful computers and computations.

The **Third Chapter** explains the Principles of Conservative Reversible Logic and Reversible Logic Gates. However, the main emphasis would be on Fredkin Gate and its universality. The gate is used in **Chapter 4** to design various reversible circuits and calculate the number of ancilla bits and garbage bits for respective circuits.

The main resource over head involved in reversible circuit is the use of garbage bits and ancilla bits to attain reversibility but they are hard to avoid in reversible cicuits.**Fifth Chapter** discusses the need for garbage bits.

The **Sixth Chapter** gives comparative findings on reversible circuits against classical circuits. Based on these findings the next Chapter opens up with how reversible circuit can be optimised in terms of number of gates, garbage bits and logical complexity.

Finally the work done has been concluded along with future scope.

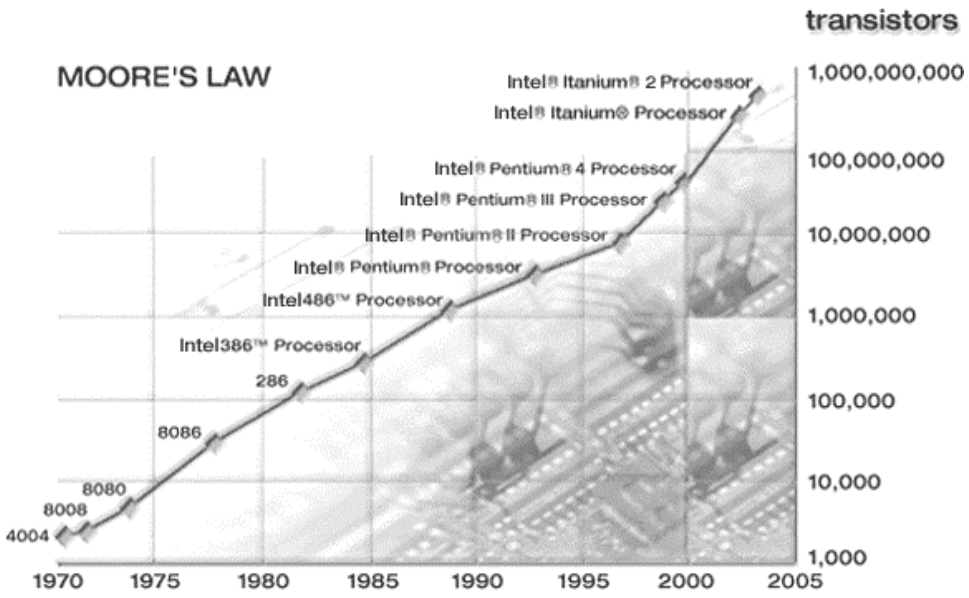
Chapter 1

Quantum Computing

1.1 Will Moore's Law fail?

Computer hardware has grown in power at an amazing pace ever since. The growth was codified by Intel co-founder Gordon Moore in 1965 in terms of **Moore's Law**, which says-

"The number of transistors on a given piece of silicon would double every couple of years". Hence computer power will double for constant cost roughly once every two years.



Source: www.Intel.com

Figure 1.1: Moore's Law

Moore's law has approximately held true in the decades since the 1960s. Nevertheless, most observers expect that this dream run will end some time during the first two decades of the twenty-first century because conventional approaches to the fabrication of computer technology will begin to run up against fundamental difficulty of size. Quantum

effects will begin to interfere in the functioning of electronic devices as they are made smaller and smaller. Now the question is will Moore's law fail? [5] The answer is no because one possible solution to the problem posed by the eventual failure of Moore's Law is to move to a different computing paradigm. One such paradigm is provided by the theory of Quantum Computation, which is based on the idea of using quantum mechanics to perform computations, instead of classical physics.

"How can we simulate the quantum mechanics? Can you do it with a new kind of computer - a quantum computer? It is not a Turing machine, but a machine of a different kind." - R P Feynman, 1981.

Quantum computers offer an essential speed advantage over Classical computers. This speed advantage is so significant that many researches believe that no conceivable amount of progress in classical computations would be able to overcome the gap between the power of a Classical computer and the power of a Quantum computer. Quantum computing will be a total change in how the computer will operate and function.

Figure 1.2 shows a survey made by Keyes in 1988.The number of dopant impurities required for logic in the bases of bipolar transistors is plotted against the year. This plot may be thought of as showing the number of electrons required to store a single bit of information. An extrapolation of the plot suggests that we might be within the reach of atomic-scale computations within the next two decades.

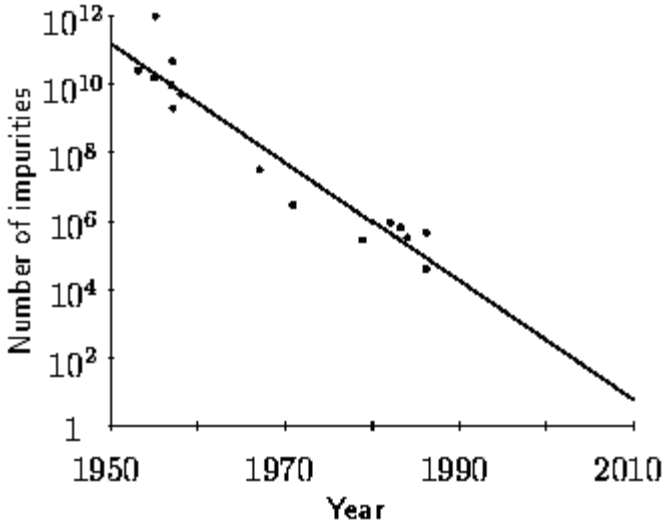


Figure 1.2: Plot showing the number of dopant impurities involved in logic in bipolar transistors with year (Keyes)

We can take advantage of the novel quantum features to devise new types of software as well as hardware. It is hoped that the explorations in this field may one day result in information processing devices with capabilities far beyond today's computing and communication systems.

1.2 Introduction to Quantum Computing

1.2.1 Quantum Computation and Quantum Information

Quantum Computation and Quantum Information is the study of the information processing tasks that can be accomplished using Quantum mechanical systems. Quantum mechanics is a mathematical framework or set of rules for the construction of physical theories. Quantum computation taught us to think physically about computation, and this approach yields many new and exciting capabilities for information processing and communication. In the broadest terms, any physical theory, not just Quantum mechanics, may be used as the basis for a theory of information processing and communication. One of the messages of Quantum computation and information is that new tools are available for those problems that are relatively more difficult or impossible to solve on Classical computers. Quantum computing believes that what is computable and what is not computable is limited by the Laws of physics.

Traditional computer science is based on Boolean logic and algorithms. Its basic variable is a bit with two possible values, 0 or 1. These values are represented in the computer as stable saturated states off or on. Quantum mechanics offer a new set of rules that go beyond this classical paradigm. The basic variable is now a qubit, represented as a vector in a two dimensional complex Hilbert space. $|0\rangle$and $|1\rangle$ form a basis in this space. The logic that can be implemented with such qubits is quite distinct from Boolean logic, and this is what has made Quantum computing exciting by opening new possibilities.

A related historical strand contributing to the development of quantum computation and quantum information is the interest, dating to the 1970s, of obtaining complete control over single Quantum systems [5]. Applications of Quantum mechanics prior to the 1970s typically involved a gross level of control over a bulk sample containing an enormous

number of Quantum mechanical systems, none of them directly accessible. Since the 1970s many techniques for controlling single Quantum systems have been developed. For example, methods have been developed for trapping a single atom in an 'atom trap', isolating it from the rest of the world and allow us to probe many different aspects of its behavior with incredible precision. Quantum computation and Quantum information provide a useful series of challenges at varied levels of difficulty for people devising methods to better manipulate single quantum systems, and simulate the development of new experimental techniques. The ability to control single quantum systems is essential if we are to harness the power of quantum mechanics for applications to Quantum computation and Quantum information.

Despite this intense interest, efforts to build Quantum information processing systems have resulted in modest success to date. Small Quantum computers, capable of doing dozens of operations on a few qubits represent the state of the art in Quantum computation. However, it remains a great challenge to physicists and engineers to develop techniques for making large-scale Quantum information processing a reality.

1.2.2 Elementary quantum Mechanics

Contrary to classical computing we perform computations using quantum states which follow properties of Quantum mechanics. Changes occurring to a Quantum state can be described using the language of Quantum computation. Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a Quantum computer is built from a Quantum circuit containing wires and elementary Quantum gates to carry around and manipulate the Quantum information.

1.2.2.1 Quantum State and Qubit

A Quantum state can be described by a state vector living in a complex vector space, also called the state space. Such a complex vector space with inner product is called a Hilbert space [17].

The bit is the fundamental concept of classical computation and classical information. Quantum computation and Quantum information are built upon an analogous concept, the quantum bit, or qubit [3,11] for short. We can describe qubits as mathematical object with certain specific properties. Qubits, like bits, can be realized as actual physical systems. The beauty of treating qubits as mathematical objects is that it gives us the freedom to construct a general theory of Quantum computation and Quantum information, which does not depend upon a specific system for its realization.

Just as a classical bit has a state either 0 or 1 – a qubit also has a state. Two possible states for a qubit are the states $|0\rangle$ and $|1\rangle$, which correspond to the states 0 and 1 for a classical bit. Notation like '$|\ \rangle$' is called the Dirac Notation**,** and it is a standard notation for states in Quantum mechanics. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$. It is also possible to form linear combination of states, often called Superposition [3,5,12]:

$$|\Psi\rangle=\alpha|0\rangle+\beta|1\rangle \qquad (1)$$

in which a and b are complex numbers satisfying the normalization condition

$$|\alpha|^2+|\beta|^2=1 \qquad (2)$$

The numbers 'α' and 'β' are complex numbers, although for many purposes not much is lost by thinking of them as real numbers. Put another way, the state of a qubit is a vector in a two-dimensional complex vector space. The special states $|0\rangle$ and $|1\rangle$ are known as computational basis states, and form an orthonormal basis $\{|0\rangle, |1\rangle\}$ for this vector space. These two basis states can be written as vectors:

It is useful to use a matrix representation of qubits. A single qubit is then written as

$$\alpha|0\rangle+\beta|1\rangle\equiv\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \qquad (3)$$

We can examine a bit to determine whether it is in the state 0 or 1. For example, computers do this all the time when they retrieve the contents of their memory. Rather remarkably, we cannot examine a qubit to determine its Quantum state, that is, the values of 'α 'and 'β'. Instead, Quantum mechanics tells us that we can only acquire much more restricted information about the Quantum state. When we measure a qubit we get either

the result 0, with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$. Thus while the classical bit can only be 0 or 1 at an instant in time, the qubit can be both. If $\alpha = 0$ or $\beta = 0$, the qubit can be represented by a classical bit. Naturally, $|\alpha|^2+|\beta|^2=1$, since the probabilities must sum to one. Geometrically, we can interpret this as the condition that the qubit's state be normalized to length 1. Thus, in general a qubit's state is a unit vector in a two-dimensional complex vector space.

The ability of a qubit to be in a superposition state runs counter to our 'common sense' understanding of the physical world around us. A classical bit is like a coin; either heads or tails up. For imperfect coins, there may be intermediate states like having it balanced on an edge, but those can be disregarded in the ideal case. By contrast, a qubit can exist in a continuum of states between $|0\rangle$ and $|1\rangle$ until it is observed [17]. Let us emphasize again that when a qubit is measured, it only ever gives '0' or '1' as the measurement result probabilistically. For example, a qubit can be in the state

$$1/\sqrt{2}|0\rangle+1/\sqrt{2}|1\rangle \qquad (4)$$

which, when measured, gives the result 0 fifty percent ($|1/\sqrt{2}|^2$) of the time, and the result 1 fifty percent of the time. This state is sometimes denoted $|+\rangle$.

We can now combine such individual qubits to higher dimensional quantum states. Two quantum mechanical systems are combined using the tensor product ($\otimes$).This symbol also stands for `combined with', and since the qubit states are independent, this is simply multiplication.

For example we can write a system of two qubits $|\psi\rangle=\alpha_1|0\rangle+\beta_1|1\rangle$ and $|\Phi\rangle=\alpha_2|0\rangle+\beta_2|1\rangle$

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \otimes \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \alpha_1\beta_1 \\ \alpha_1\beta_2 \\ \alpha_2\beta_1 \\ \alpha_2\beta_2 \end{pmatrix} \qquad (5)$$

Instead of $|\psi\rangle \otimes |\phi\rangle$, we will also use the shorthand notations $|\psi\rangle|\Phi\rangle$=and $|\psi\ \Phi\rangle$.Formally the state $|\psi\ \Phi\rangle$ is called a product state. As a simple example consider the 2 qubit state.

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (6)$$

Keeping with the classical analogy, we can build a register of L qubits by ordering them such that the total state of the register ψ reads

$$|\psi\rangle = (\alpha_1|0\rangle_1 + \beta_1 |1\rangle_1)\,(\alpha_2|0\rangle_2 + \beta_2 |1\rangle_2) \otimes \ldots\ldots \otimes (\alpha_L|0\rangle_L + \beta_L |1\rangle_L) \quad (7)$$

Notice that the expansion of Eq. 7 contains states ranging from $|0\rangle_1|0\rangle_2 \ldots|0\rangle_L$ to $|1\rangle_1|1\rangle_2\ldots|1\rangle_L$, each of which can be identified with the binary numbers ranging from 0 to 2^L-1. For example, if we let x = 5, then $|x\rangle \equiv |5\rangle \equiv |1\rangle_1|0\rangle_2|1\rangle_3 \equiv |101\rangle$. If we denote C_x as the certain combination of the complex numbers α_i and β_i that multiplies state $|x\rangle$, then the total state is equivalent to

$$|\Psi\rangle = \sum_{x=0}^{2^L-1} C_x|x\rangle \quad (8)$$

The measurement of every qubit would necessarily yield only one of the eigenvectors of $|\Psi\rangle$. Since the ai's and bi's were originally normalized,

$$\sum_{x=0}^{2^L-1} |C_x|^2 = 1 \quad (9)$$

An array of L qubits in the computational basis is written as

$$\sum_{x=0}^{2^L-1} C_x |x\rangle \equiv \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{2^L-1} \end{pmatrix} \quad (10)$$

As an example, suppose we have 2 qubits q_1 and q_2, independently prepared such that

$$q_1 = \sqrt{3/4}|0\rangle - \sqrt{1/4}|1\rangle \text{ and } q_2 = \sqrt{1/6}|0\rangle + i\sqrt{5/6}|1\rangle \quad (11)$$

$$
\begin{aligned}
|\psi\rangle &= \left(\sqrt{3/4}|0\rangle - \sqrt{1/4}|1\rangle\right)\left(\sqrt{1/6}|0\rangle + i\sqrt{5/6}|1\rangle\right) \\
&= \sqrt{1/8}|0\rangle|0\rangle + i\sqrt{5/8}|0\rangle|1\rangle - \sqrt{1/24}|1\rangle|0\rangle - i\sqrt{5/24}|1\rangle|1\rangle \\
&= \sqrt{1/8}|00\rangle + i\sqrt{5/8}|01\rangle - \sqrt{1/24}|10\rangle - i\sqrt{5/24}|11\rangle \\
&= \begin{pmatrix} \sqrt{1/8} \\ i\sqrt{5/8} \\ -\sqrt{1/24} \\ -i\sqrt{5/24} \end{pmatrix}
\end{aligned} \tag{12}
$$

The equality in Eq.12 is in the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

A vector written in the form $|.\rangle$ is called a ket. Its so-called dual, $\langle\psi|$, is defined as the conjugate transpose of $|\psi\rangle$. This way of writing a vector is referred to as bra. Together they form the bra(c) ket $\langle\psi|\psi\rangle$ [17]. This denotes the inner product between two state vectors $|\psi\rangle$ and $|\phi\rangle$ given by $\langle\psi|\phi\rangle$.It becomes clear that n qubits can have a state space of dimension C^{2n}, which means the state space grows exponentially with the number of qubits.

At this point we can see the enormous complexity involved: classically the state of a register of 300 qubits would require 2^{300}-1 $\approx 2*10^{90}$ complex numbers to describe completely, since each qubit has 2 complex numbers attributed to it and an overall phase can be factored out. Even if we could store a complex number for every particle in the universe, it would be impossible to classically store this state.

Despite this strangeness, qubits are decidedly real; their existence and behavior extensively validated by experiments, and many different physical systems can be used to realize qubits. They can be realized as the two different polarizations of a photon [7,22]; as the alignment of a nuclear spin in a uniform magnetic field; or as two states of an electron orbiting a single atom such as shown in the Figure 1.3.

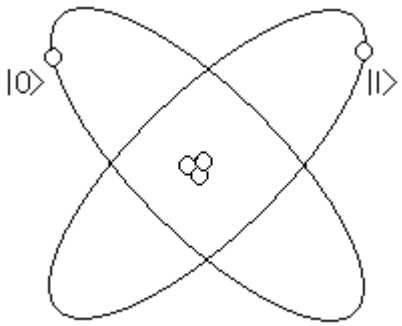


Figure 1.3: Qubit represented by two electronic levels in an atom

In the atom model, the electron can exist in either the so-called 'ground' or 'excited' states, which we'll call $|0\rangle$ and $|1\rangle$, respectively. By shining on the atom, with appropriate energy and for an appropriate length of time, it is possible to move the electron from the $|0\rangle$ state to the $|1\rangle$ state and vice versa. But more interestingly, by reducing the time we shine the light, an electron initially in the state $|0\rangle$ can be moved 'halfway' between $|0\rangle$ and $|1\rangle$, into the $|+\rangle$ state. One picture useful in thinking about qubits is the following geometric representation. Because $|a|^2+|b|^2=1$, we may rewrite equation (1) as

$$|\Psi\rangle=e^{i\gamma}(\text{Cos}\,(\theta/2)\,|0\rangle+e^{i\varphi}\text{Sin}\,(\theta/2)\,|1\rangle), \qquad (13)$$

where θ, φ and γ are real numbers. We can ignore the factor of $e^{i\gamma}$ out the front, because it has no observable effects, and for that reason we can effectively write

$$|\Psi\rangle= (\text{Cos}\,(\theta/2)\,|0\rangle+e^{i\varphi}\text{Sin}\,(\theta/2)\,|1\rangle). \qquad (14)$$

The numbers θ and φ define a point on the unit three-dimensional sphere, as shown in the Figure 1.4. This sphere is often called the Bloch Sphere**;** it provides a useful means of visualizing the state of a single qubit, and often serves as an excellent test bed for ideas about Quantum Computation and Quantum Information [22]. Many of the operations on single qubit can be neatly described within the Bloch Sphere picture. However, it must be kept in mind that this intuition is limited because there is no simple generalization of the Bloch Sphere known for multiple qubits.

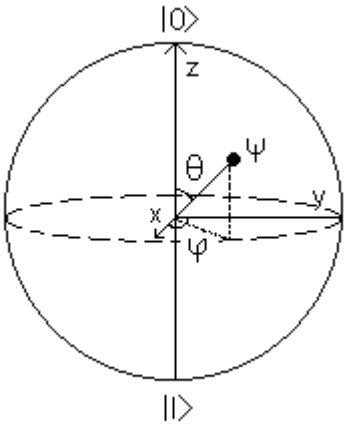


Figure 1.4: Bloch sphere representation of a qubit

How much information is represented by a qubit? There are an infinite number of points on the unit sphere, so that in principle one could store an entire text of Shakespeare in the infinite binary expansion of θ. However, this conclusion turns out to be misleading, because of the behavior of a qubit when observed. Measurement of a qubit gives only either 0 or 1. Furthermore, measurement changes the state of a qubit, collapsing it from its superposition of $|0\rangle$ and $|1\rangle$ to the specific state consistent with the measurement result. For example, if measurement of $|+\rangle$ gives 0, then the post-measurement state of the qubit will be $|0\rangle$. Why does this type of collapse occur? Nobody knows. This behavior is simply one of the fundamental postulates of Quantum mechanics.

1.2.2.2 Logic Gates for Quantum Bits

Arbitrary Logic gates may be constructed for quantum bits. We start by considering various one-bit unitary operations and a single two-bit XOR operation. Combinations of these are sufficient to construct a Toffoli gate for Quantum bits or indeed any unitary operation on a finite number of bits. Refer other Quantum gates in **Appendix I**.

Start with a single quantum bit. If we represent the states $\left|\downarrow\right\rangle$ and $\left|\uparrow\right\rangle$ (i.e., $|0\rangle$ and $|1\rangle$) as the vectors $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$, respectively, then the most general unitary transformation corresponds to a 2*2 matrix of the form:

$$U_\theta \equiv \begin{pmatrix} e^{i(\delta+\sigma+\tau)}\cos\left(\theta/2\right) & e^{-i(\delta+\sigma-\tau)}\sin\left(\theta/2\right) \\ -e^{i(\delta-\sigma+\tau)}\sin\left(\theta/2\right) & e^{i(\delta-\sigma-\tau)}\cos\left(\theta/2\right) \end{pmatrix}, \tag{15}$$

where we typically take $\delta = \sigma = \tau = 0$ [10]. Using this operator we can flip bits via:

$$U_\pi|0\rangle = -|1\rangle \text{ , and } U_\pi|1\rangle = -|0\rangle \tag{16}$$

The extraneous sign represents a phase factor that does not affect the logical operation of the gates and may be removed if we wish, now or at a later stage. Such one-qubit computation is illustrated schematically as a quantum circuit in Figure 1.5 below.

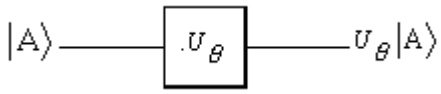


Figure 1.5: One-qubit quantum circuit

Schematic of the Quantum circuit diagram for a one-bit gate. The line represents a single quantum bit (such as a spin -1/2 particle). Initially, this bit has a state described by; after it has ``passed" through this circuit it comes out in the state.

Another important one-bit gate is $U_{-\pi/2}$ which maps a spin-down particle

$$U_{-\pi/2}|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right), \tag{17}$$

to an equal superposition of down and up. Consider a string of K spin-1/2 particles initially spin-down. If we apply this gate independently to each particle we obtain a superposition of every possible bit-string of length K:

$$|0\rangle \to \frac{1}{\sqrt{q}}\sum_{\alpha=0}^{q-1}|a\rangle, \tag{18}$$

where $q=2^k$. Our computer is now in a superposition of an exponentially large number of integers a from 0 to 2^k-1.Suppose we could now construct a unitary operation which maps a pair of bit-strings $|a;0\rangle$ into the pair $|a; f(a)\rangle$ for some function $f(a)$. Then such a unitary operator acting on the superposition of states [17].

$$\frac{1}{\sqrt{q}}\sum_{\alpha=0}^{q-1}|a;0\rangle \rightarrow \frac{1}{\sqrt{q}}\sum_{\alpha=0}^{q-1}|a; f(a)\rangle, \tag{19}$$

would compute $f(a)$ in parallel an exponentially large number of times for the various inputs a.

To see how such unitary operators may be constructed from a few elementary ones we must also consider the XOR gate [1,20]. Writing the two-particle basis states as the vectors .Refer equation (6)

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \quad |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix},$$
$$|10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \quad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}, \tag{20}$$

we may represent the XOR gate as a unitary operator

$$U_{XOR} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix} \tag{21}$$

Here the first particle acts as a conditional gate to flip the state of the second particle. It is easy to check that the state of the second particle corresponds to the action of the XOR gate. The quantum circuit for an XOR gate is illustrated in Figure 1.6 below. This circuit is equivalent to the elementary instruction

$$\text{If } (|A\rangle = 1\ )\ |B\rangle = \text{NOT}\ |B\rangle \tag{22}$$

which may be thought of as example of quantum computer code. The ket-brackets $|\ \rangle$ are reminders that we are dealing with quantum rather than classical bits. The XOR gate allows us to move information around.

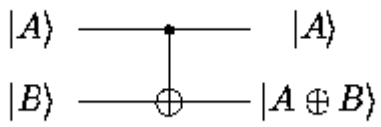


Figure 1.6: Quantum circuit diagram for an XOR gate.

The lower bit $|B\rangle$ is flipped whenever the upper bit $|A\rangle$ is set. It can be termed as "Controlled-NOT" or C-NOT operation.

$$CNOT : |A\rangle|B\rangle = |A\rangle|B \oplus A\rangle, \tag{23}$$

$$\begin{aligned} CNOT &: |0\rangle|B\rangle = |0\rangle|B\rangle \\ CNOT &: |1\rangle|B\rangle = |1\rangle|1 - B\rangle \end{aligned} \tag{24}$$

where $A \in \{0,1\}$ is the control, $B \in \{0,1\}$ is the target, and $\oplus$ denotes addition modulo 2. The above logic can be used to swap a pair of qubits.

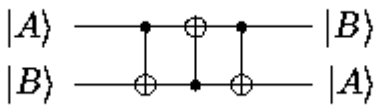


Figure 1.7: Circuit for swapping a pair of qubits

1.2.3 Properties of Quantum Computer

The main differences between quantum and classical computers are three quantum properties that would make a quantum computer "less than a machine than a force of nature". These three properties are as follows:

1.2.3.1 Superposition

Two bits in a classical computer can take values 00,01,10,11.Two qubits could represent each of these values, however two qubits can also be put into a superposition[3,5,12] of any or all these states. Thus 2 qubits can represent the numbers 0, 1, 2 and 3 simultaneously. To represent the superposition states of a qubit we usually use the following notation. We write $|0\rangle + |1\rangle$ to represent particular superposition of $|0\rangle$ and $|1\rangle$.For a polarized light, $|0\rangle + |1\rangle$ represent light polarized halfway -45 degrees between the horizontal and the vertical. For light polarized at other angles we need to add the two components in different amounts. This is achieved by multiplying each term by a factor or amplitude. By varying the amplitudes of 2 states $|0\rangle$and$|1\rangle$ we can create infinite number of superpositions.

The general state represent a superposition in which the amplitudes are a and b

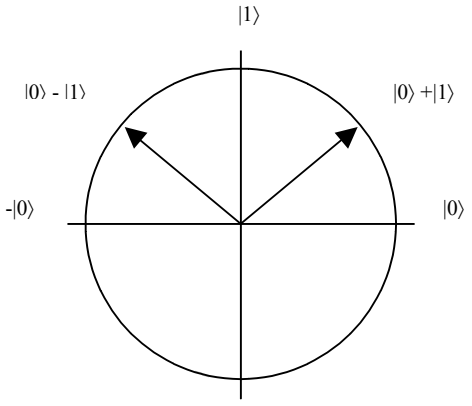


Figure 1.8: Vector representation of qubit states

1.2.3.2 Entanglement

Without entanglement quantum system would be impossible. Entanglement [7,12] is the ability of Quantum systems to exhibit correlations between states within a superposition. If we have two qubits each in superposition of 0 and 1, the qubits are said to be entangled if the measurement of one qubit is always correlated with the result of the measurement of the other qubit. According to some physicists it is specifically these entangled superpositions that open up extraordinary new possibilities in information processing

If we had superposition of numbers from 0 to 7 stored in 3 qubit register (register are groups of bits or qubits) and performed a complicated series of operations on these numbers to do some mathematical calculations. Reading off an answer would merely tell us one possible answer but not the number that generated it. Quantum entanglement though enables us to link quantum registers so that whenever an answer appears in one register, we can always look in the other to find out what number generated it. Without Quantum entanglement, the Quantum computer would be like the papers with answers of unknown questions.

1.2.3.3 Interference

To exploit the power of Quantum computation we need to use phenomenon called Interference [12]. It helps to overcome a severe restriction imposed by Quantum mechanics. We are not actually allowed to look at each and every answer individually. Because looking at Quantum information disturbs it and in Quantum computation only one result would survive in non Quantum form i.e. Interference.

Interference arises from the fact that the wavelike aspects of Quantum particles can overlap one another to cause unusual and distinctive patterns of behavior. In Quantum computation, we use interference to read a new result that depends mathematically on all those intermediate results without revealing what any of them are.

1.3 Model Quantum Computer and Quantum Code

In this section we describe a simple model for a Quantum computer based on a Classical computer instructing a machine to manipulate a set of spins. This model has some intrinsic limitations, which make designing algorithms in a high-level language somewhat tricky. We discuss some of the rules for writing such Quantum computer code as a high-level language and give an example.

Consider the following model for the operation of a Quantum computer: Several thousand spin- particles (or two-level systems) are initially in some well defined state, such as spin-down. A classical machine takes single spins or pairs of spins and entangles them (performing an elementary one-bit operation or the two-bit XOR gate); see Figure1.9 a, b and c. These stages are repeated on different pairs of spins according to the instructions of a conventional computer program. Since the spins are entangled, we must not look at the spins at intermediate stages: We must keep the quantum superposition intact. Furthermore, nothing else may interfere with the spins, which could destroy their orientation or interrupt their unitary evolution. Once this well-defined cycle of manipulation is complete the orientations of the spins are measured (Figure1.9d). This set of measured orientations is the output of the computation.

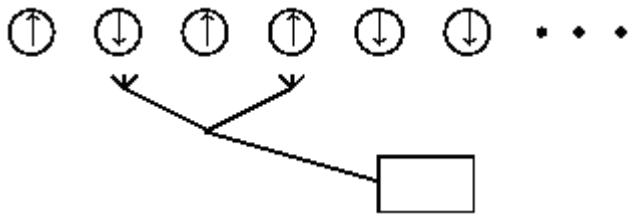


Figure 1.9 (a)

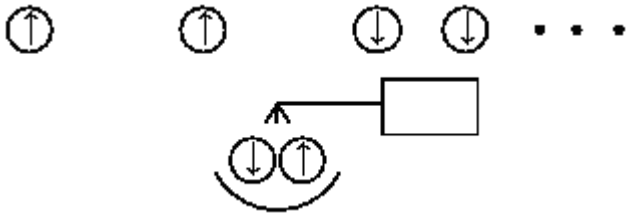


Figure 1.9 (b)

Figure 1.9 (c)

Figure 1.9 (d)

Figure 1.9: Model quantum computer as pictured by Shor

Initially all particles are spin-down. Stage a) a classical machine takes a single or pair of spins and in stage b) it performs a selected one-bit or two-bit operation; in stage c) the ``entangled'' particles are returned to their original locations. These three stages are repeated many times in accord with the instructions given by an ordinary classical computer. When this cycle is complete stage d) consists of measuring the state of the particles (leaving them in some particular bit-string); this bit-string is the result of the computation.

Given this paradigm for a Quantum computer, what might its high-level language (its computer code) look like? The most serious difficulty that must be dealt with is that the ,quantum information is manipulated by a conventional computer in a completely blind manner---without any access to the values of this Quantum information. This means that the program cannot utilize ``shortcuts'' conditional on the value of a Quantum variable (or register or bit). For example, loops must be iterated through exactly the same number of

times independent of the values of the Quantum variables. Similarly, conditional branches around large pieces of code must be broken down into repeated conditions for each step. In addition, each instruction performed upon the Quantum bits must be logically reversible. Thus, ordinary assignments of a value to a variable, such as $|a\rangle = n$, are not legal and must instead be performed as increments on an initially zeroed variable, such as $|a\rangle = |a\rangle + n$.

An example of such code that could run on this machine might look like this [21]:

Do 10 k = 1, worstdiv
$|a\rangle = |a\rangle - n$
if ($|a\rangle$ >= 0) $|q\rangle = |q\rangle + 1$
10 continue
Do 20 k = 1, worstdiv
if (k > $|q\rangle$) $|a\rangle = |a\rangle + n$
20 continue

This code fragment could be used to calculate the quotient and the remainder, placed in and , respectively, for the division of by n; the constant worstdiv is the worst-case number of times the loop must be traversed. Here is initially zero. Each instruction here is either a conventional computer instruction or one involving some Quantum variables. The former are direct instructions for the external computer, while the latter must be interpreted as a sequence of manipulations to be performed upon the Quantum bits. As it stands, this code is not reversible (neither is it very efficient), e.g., the label 10 gives no specification of which routes might be used to get to it. It can, however, be easily rewritten.

1.4 Application of Quantum Computing

- **Quantum Communication:** Many research groups are working on Quantum communication systems. They allow a sender and receiver to agree on a code without ever meeting in person. With quantum teleportation one cannot possibly intercept a signal path and extract information because it's possible to transmit information without a signal path. The uncertainty principle, an inescapable property of the quantum world, ensures that if an eves dropper tries to monitor the

signal in transit it will be disturbed in such a way that the sender and receiver are alerted. Ultra secure communication is also possible by super-dense information coding. Quantum bits can be used to communicate more information per bit than the same number of classical bits.

- **Quantum Cryptography:** The expected capabilities of Quantum computation promise great improvements in the world of cryptography. Ironically the same technology also poses current cryptography techniques a world of problems. The implications of Shor's factoring algorithm on the world of cryptography is staggering. The ability to break the RSA coding system will render almost all current channels of communication insecure.
- **Improved error correction and error detection:** Through similar processes that support ultra-secure and super-dense communication, existing bit streams can be made more robust and secure by improving error correction and detection. Recovering information from a noisy transmission path will be lucrative and useful.
- **Artificial Intelligence:** The theories of Quantum computation suggest that every physical object, even the Universe, in some sense a Quantum computer. If this is the case, then according to Turing's work which says that all computers are functionally equivalent, computers should be able to model every physical process. Ultimately this suggests that computers will be capable of simulating conscious rational thought. These theories provoke a minefield of philosophical debate, but maybe the Quantum computer will be the key to achieving true artificial intelligence.
- **Molecular simulations:** Classical computers couldn't simulate quantum effects without an exponential slow down. A Quantum computer can simulate physical processes of Quantum effects in real time. Molecular simulations of chemical interactions will allow chemists and pharmacists to learn more about how their products interact. Further, they will be able to determine how their products interact with biological processes; how a drug may interact with human a person's metabolism or disease. Pharmaceutical research offers a big market to such applications.

- **True randomness:** Classical computers cannot generate true random numbers. Quantum computers can generate true randomness and give more veracity to programs that need randomness in processing. Randomness plays a significant part of applications with heavy reliance on statistical approaches, such as simulations, code making, randomized algorithms for problem solving, and for stock market predictions
- **Quantum Algorithms:** Besides all the above applications of quantum computing there are various popular Quantum Algorithms currently being used like for Searching (especially algorithmic searching) Grover's algorithm and Shor's algorithm for Factorising large numbers very rapidly. Also Grover and Durr-Hoyer algorithm for estimating median and mean respectively. Quantum algorithms are also used for simulating Quantum mechanical systems efficiently.

1.5 Recent Development in Quantum Computing

- On 18 April 1998, Neil Gershenfeld of MIT and Isaac Chuang of IBM Almaden Research Center announced the construction of the first Quantum computer. It was composed of two qubits and was manipulated using a "thimbleful of chloroform" placed in an NMR (nuclear magnetic resonance) chamber. Small as it was, this computer successfully demonstrated a Quantum search algorithm devised by Lov Grover of Bell Laboratories.
- A year later Gershenfeld and Chuang expanded their computer to 3-qubit version. They again ran a demonstration of the Grover algorithm.
- On 15 August 2000, Isaac Chuang, leading a team of IBM Almaden Research Center, Standford University, and the University of Calgary, announced the construction of a 5-qubit computer. It was again NMR based, but this time using flourine atoms and demonstrating "order finding," an aspect of the much celebrated code-cracking capabilities of quantum computers.
- On 19 December 2001, Isaac Chuang's team once again set a new milestone by running Shor's factoring algorithm on a 7-qubit Quantum computer. The approach was the same that was used over a year ago to create a 5-qubit computer, but this time the molecule was made up of five flourine atoms plus two

carbon atoms. It is speculated that the NMR approach may be able to produce one more milestone before losing steam and giving up the lead to computers built using quantum dots or Josephson Junctions.

- April 2002**,** Researchers at Harvard University have demonstrated the ability to transfer Quantum Information from light into the spin state of atoms and then, nondestructively, read that information back out again into light. This result was achieved by impinging rubidium atoms with laser beams. What's more, it was shown that when a numerical operation was performed (via magnetic pulses) on the information while it was stored in the spins of the rubidium atoms, when the signal was read back out again, the phase of the light was predictably shifted in response to that operation. The main focus of the Harvard team is Quantum cryptography, but the same technology could also serve to create effective RAM for a Quantum computer.

1.6 Future Prospects

To date this is the factoring algorithm is the only one displaying exponential speed-up. This algorithm was applied to a traditional computer-science problem, factoring, only by recognizing a deeper structure within that problem. This requirement appears to be a general one: Quantum parallelism will only yield an exponential speedup in problems whose structure avoids the need to try exponentially many solutions [2,8].

Thus, a brute force approach to some of the hardest computational questions, known as NP-complete problems, will not succeed with the aid of quantum parallelism. Any progress for such problems will require finding a deeper structure within them. Instead, Quantum computers are likely to be most useful for simulating or manipulating small Quantum systems [12].

Even within apparently small atomic-scale systems, Quantum computation runs on the enormous size of Hilbert space. Quantum computation involves building a trajectory from a standard initial state to a complex final state. The difficulty would be in keeping to this trajectory. To fail is to be lost in Hilbert space [17]. The largest problem is hypersensitivity to perturbations, shifting the computational trajectory randomly from its

path. Such perturbations come from an unintentional coupling to external noise. It is too soon to predict the gravity of this problem. It appears, though, that there is no fundamental limit to how well we can isolate a Quantum system. Currently, several implementations are being considered by theoreticians and experimentalists worldwide, like the ion-trap method, solid state Quantum dots, NMR method, Neutral atom and optical methods.

So we may be at a stage where we can speculate whether or not the increase in power or complexity of Quantum computers will follow the same trend as silicon based classical computers (Moore's Law). An analogous trend for Quantum computers would be that the number of qubits will double every eighteen months. However, it is still too early to draw any firm conclusions.

Extending this concept to multi-qubyte systems, it can be seen that there is potential for computational efficiency exponentially beyond anything possible with Classical computers.

Chapter 2
Reversible Computing

A computation is called reversible if its inputs can always be deduced from its outputs. It's only irreversible computations that must dissipate heat.
One of the most important computational resources is energy. In principle computation both classical and Quantum can be done without expending any energy. The energy consumption in computation turns out to be deeply linked to the reversibility of the computation.

One of the biggest problems with miniaturizing conventional computers is the difficulty of dissipated heat. As early as 1961 Landauer studied the physical limitations placed on computation from dissipation [9]. Surprisingly, he was able to show that almost all operations required in computation could be performed in a reversible manner, thus dissipating no heat! The first condition for any deterministic device to be reversible is that its input and output be uniquely retrievable from each other. This is called logical reversibility. If, in addition to being logically reversible, a device can actually run backwards then it is called physically reversible and the second law of thermodynamics guarantees that it dissipates no heat.

2.1 Landauer's Principle

Today's computers erase (overwrite) bit of information every time they perform a logic operation. These logic operations are therefore called "irreversible". They dissipate energy into the environment with every bit erased. The amount of energy dissipated into the environment is at least k_bTln2, where k_b is a universal constant known as the Boltzmann's constant, and T is the temperature of the environment of the computer. [9] For T = 300 Kelvins (room temperature), this is about 2.9 x 10^-21 joules. This is roughly the kinetic energy of a single air molecule at room temperature.

If we are to continue the revolution in computer hardware performance we must continue to reduce the energy dissipated by each logic operation. Today, because we are dissipating much more than *kT*, we can do this by improving conventional methods, i.e., by improving the efficiency with which we erase information.

An alternative is to use logic operations that do not erase information. These are called *reversible* logic [9] operations, and in principle they can dissipate arbitrarily little heat. As the energy dissipated per irreversible logic operation approaches the fundamental limit of ln 2 x *kT*, the use of reversible operations is likely to become more attractive. If current trends continue this should occur sometime in the 2010 to 2020 timeframe. If we are to reduce energy dissipation per logic operation below ln 2 x *kT* we will be *forced* to use reversible logic.

Charles Bennett showed that for power not be dissipated in the circuit it is necessary that arbitrary circuit can be build from reversible gates and that there are no unavoidable energy consumption requirements per step in a computer.

2.2 Heat Dissipation

Physical limitations placed on computation by heat dissipation were studied for many years [9].It is one of the major problems with the miniaturization of Classical computers and constant cooling of all components is required. This is achieved by the thermic coupling of the circuits to a heat reservoir like e.g. the surrounding air.

For a Quantum computer, cooling by heat coupling is no option since its logical state is directly represented by the common Quantum state of its registers. Any heat coupling would necessarily result in the entanglement [12] of this state with the outside world and destroy the coherence of the computation. The second law of thermodynamics postulates that any non-reversible state change of a system must dissipate heat. Many common logical operations like AND, OR or resetting a bit to 0 or 1 are non-reversible is the sense that the input cannot be calculated from the output. Therefore, these operations cannot directly be implemented in a Quantum computer.

As we pack more and more logic elements into smaller and smaller volumes and clock them at higher and higher frequencies, we dissipate more and more heat. This creates at least three problems:

- Energy costs money.
- Portable systems exhaust their batteries.
- Systems overheat.

Reversible computation leads to reduce heat dissipation, thereby allowing:

- Higher densities
- Higher speed

2.3 Illustration of reversible computing

Reversible circuits or gates are those which have one to one mapping between the vectors of the inputs and the outputs. Consequently they have the same number of inputs and outputs. Thus the vector of the input states can be always reconstructed from the vector of the output states. Feedback and Fanout not allowed in these combinations [18].

Figure 2.1 shows such an associations.

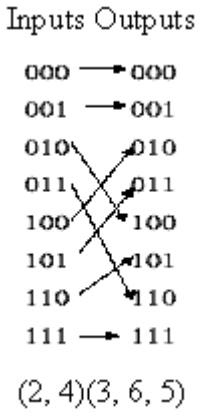


Figure 2.1: Input Output Mapping In Reversible Circuit

Original

```
int x=0, y=1;

f()
{
  if (x>y)
    y += x;
  else
    x += y;
}
```

Reversible

```
int x=0, y=1;
bit b;

f'()
{
  b = (x>y);
  if (b)
    y += x;
  else
    x += y;
}
```

Reverse

```
int x=0, y=1;
bit b;

rf'()
{
  if (b)
    y -= x;
  else
    x -= y;
}
```

Note: If S is the state, then we need rf'(f'(S))==S. However we don't care about the result of f'(rf'(S)).

Figure 2.2: Reversible C Programs

Note that we use an extra variable which apparently stores the operation which done on the variables hence keeping track of it. This is used in reversing the operation.

2.4 Billiard Ball Model

A major step toward Fredkin's goal of finding a reversible physical basis for computation was his discovery of the billiard-ball model of computation [14,10].

This takes advantage of the fact that a collision between two classical hard spheres ("balls") diverts each one from the path it would have followed had the other been absent; thus a collision can be thought of as a two-input, four output logic function whose outputs, for inputs A and B ,are, A and B ,B and not A, A and not B, A and B respectively.

]

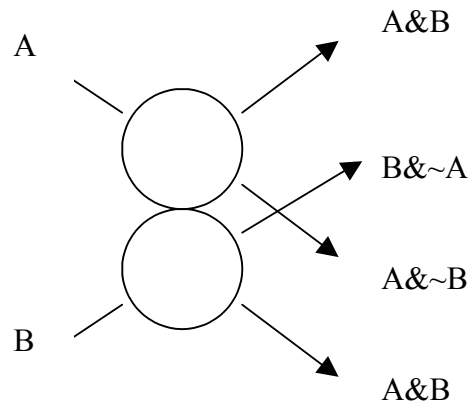


Figure 2.3: Use of a billiard ball collision to realize a two-input, four output logic function

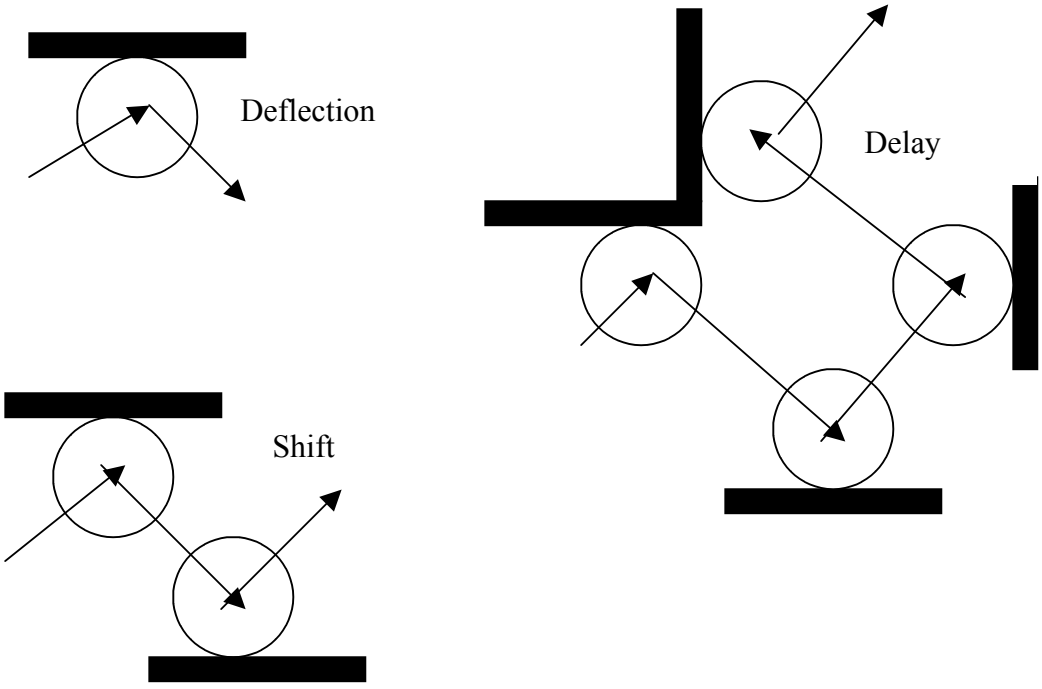


Figure 2.4: Few examples of how operations are realized using the billiard ball model

Figure 2.5, e.g., shows a classical billiard ball mechanism based on the ideas of Fredkin that uses one billiard ball (dark) to test the presence of another (light) without disturbing the dynamical state of the latter [18]. The apparatus consists of a number of fixed mirrors

(dark rectangles) which reflect the billiards balls. First assume that the dark ball is absent.then a light ball injected into the apparatus at X will follow the closed diamond shaped trajectory, ABCDEFA forever representing the value 1; conversely, the absence of the light ball (i.e. no balls in the apparatus at all) represents the value 0.The goal of the measurement is to inject another ball (dark color) into the apparatus in such a way that it tests whether the light ball is present without altering the light ball's state. By injecting the dark ball at Y at the appropriate time, the light ball(if present) is diverted from, but then returned to its original path(following BGD instead of BCD),while the dark ball leaves the apparatus at M if the light ball was present and at N if it was absent.

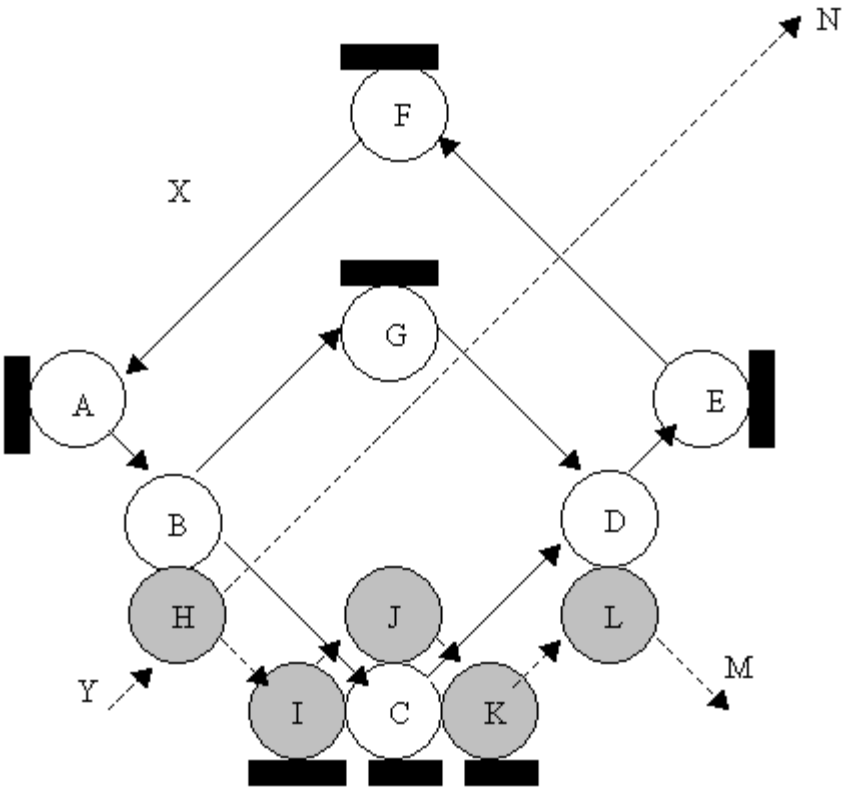


Figure 2.5: Reversible Measurement in Billiard ball model of Computation

Fredkin showed that, with the addition of "mirrors" to redirect the balls, such collisions can simulate any conservative logic function, and therefore any ordinary logic function. This implies that a model computer built entirely of billiard balls and mirrors gives a

beautiful concrete realization of the principles of reversible computation. Furthermore this model of computation turns out to be universal in the sense that it can be used to simulate an arbitrary computation in the standard circuit model of computation.

Of course, if a billiard ball computer were ever built it would be highly unstable. A billiard ball rolling frictionlessly over a smooth surface is easily knocked off course by small perturbations. The billiard ball model of computation depends on perfect operation, and the absence of external perturbations such as those caused by thermal noise. Periodic corrections can be performed, but the information gained by doing this would have to be erased, requiring work to be performed. Expenditure of energy thus serves the purpose of reducing this susceptibility to noise, which is necessary for practical, real-world computational machine.

So we saw that in the computation process, energy is dissipated whenever information is destroyed. The lesson is: to dissipate less energy, don't discard information. One way to do this would be to build logic circuits that can run in reverse. That is, the circuits (and the computers using them) would return to their original states at the end of the computation cycle after having performed the required calculations. This idea has been promoted by Rolf Landauer and Charles Bennett of IBM, who spoke at the March APS Meeting in Seattle. In principle such circuits, performing reversible operations, would dissipate less energy than irreversible circuits (operating with no regard for the retention or destruction of information) performing irreversible operations. Ralph Merkle of Xerox said that the use of reversible circuitry, entailing also the use of reversible software, was not yet a priority for computer architects but would be early in the next century when the problems of heat dissipation (requiring large heat sinks) and energy consumption become more pressing.

2.5 Future Prospects

Today the use of reversible logic operations can be a useful heuristic in the design of systems that use very little power. To achieve a completely reversible system (which

erases no bits at all) is very difficult. Systems that perform some operations in a reversible fashion can dissipate less energy and might prove competitive today.

The work on classical, reversible computation has laid the foundation for the development of Quantum mechanical computers. On a Quantum computer, programs are executed by unitary evolution of an input that is given by the state of the system. Since all unitary operators $\mathbf{U}$ are invertible with $\mathbf{U}^{+} = \mathbf{U}^{-}$ [10], we can always "Uncompute" (reverse the computation) on a Quantum computer. Therefore research on reversible logic is beneficial to the development of future Quantum technologies: reversible design methods might give rise to methods of Quantum circuit construction, resulting in much more powerful computers and computations.

Chapter 3
Logic Gates

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

3.1 Basic logical gates

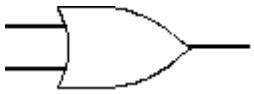
There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.The **AND GATE** is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator. The following illustration and table show the circuit symbol and logic combinations for an AND gate. (In the symbol, the input terminals are at left and the output terminal is at right.) The output is "true" when both inputs are "true." Otherwise, the output is "false."

AND gate

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

Table 3.1: AND gate

The OR gate gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false."

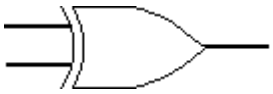


OR gate

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Table 3.2: OR gate

The XOR (exclusive-OR) gate acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.

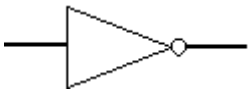


XOR gate

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table 3.3: XOR gate

A logical inverter, sometimes called a NOT gate to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state.



Inverter or NOT gate

Input	Output
1	0
0	1

Table 3.4: NOT gate

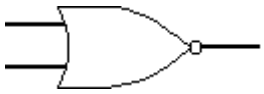
The NAND gate operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

NAND gate

Input 1	Input 2	Output
0	0	1
0	1	1
1	0	1
1	1	0

Table 3.5: NAND gate

The NOR gate is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false."

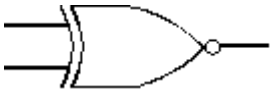


NOR gate

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	0

Table 3.6: NOR gate

The XNOR (exclusive-NOR) gate is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same, and "false" if the inputs are different.



XNOR gate

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	1

Table 3.7: XNOR gate

Using combinations of logic gates, complex operations can be performed. In theory, there is no limit to the number of gates that can be arrayed together in a single device. But in practice, there is a limit to the number of gates that can be packed into a given physical

space. Arrays of logic gates are found in digital integrated circuits (ICs). As IC technology advances, the required physical volume for each individual logic gate decreases and digital devices of the same or smaller size become capable of performing ever-more-complicated operations at ever-increasing speeds.

3.2 Reversible Logic Gates

A reasonable computation is one that may be written in terms of some (possibly large) Boolean expression, and any Boolean expression may be constructed out of a fixed set of logic gates. Such a set (e.g., AND, OR and NOT) is called universal. In fact we can get by with only two gates, such as AND and NOT or OR and NOT. Alternatively, we may replace some of these primitive gates by others, such as the exclusive-OR (called XOR); then AND and XOR form a universal set. Any machine, which can build up arbitrary combinations of logic gates from a universal set is then a universal computer.

Which of the above gates is reversible? Since AND, OR, and XOR are many-to-one operations, they are not logically reversible. Before we discuss how these logic gates may be made reversible we consider some non-standard gates that we shall require.

3.2.1 Fanout and Erase

Although the above gates are sufficient for the mathematics of logic, they are not sufficient to build a practical machine. A useful computer will also require the Fanout and Erase gates.

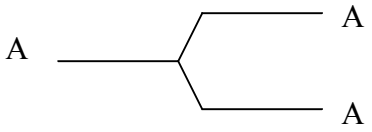


Figure 3.1(a): Fanout

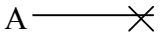


Figure 3.1(b): Erase

Two non-standard gates that are required to build a computer, in addition to a universal set of logic gates, are: Figure 3.1(a) the FANOUT gate which duplicates an input A and Figure3.1 (b) the ERASE gate which deletes its input.

First consider the FANOUT gate: Is it reversible? Certainly no information has been destroyed so it is at least logically reversible. Landauer showed that it could also be physically reversible.

Let us now consider the ERASE operation, which is required to ``clean out'' the computer's memory periodically. One type of erasure can be performed reversibly: If we have a backup copy of some information, we can erase further copies by uncomputing the FANOUT gate. The difficulty arises when we wish to erase our last copy, referred to here as the primitive ERASE.

Consider a single bit represented by a pair of equally probable classical states of some particle. To erase the information about the particle's state we must irreversibly compress phase-space by a factor of two. If we allowed this compressed phase-space to adiabatically expand, at temperature T, to its original size, we could obtain an amount of work equal to $K_b T \ln 2$ (where K_b is Boltzmann's constant). Landauer concluded, based on simple models and more general arguments about the compression of phase-space, that the erasure of a bit of information at temperature T requires the dissipation of at least heat $K_b T \ln 2$ [9]

3.2.2 Principles of Conservative Reversible Logic

A gate is reversible if the gate’s inputs and outputs have a one-to-one correspondence, i.e. there is a distinct output assignment for each distinct input [27]. Therefore, a reversible gate’s inputs can be uniquely determined from its outputs. A ramification of this definition is that reversible logic gates must have an equal number of inputs and outputs.

If logic gates are classified according to the number of inputs and outputs, a gate with m inputs and n outputs is a (m,n) or m*n logic gate, where reversible gates must have m=n. Furthermore, a reversible gate’s output vector is a permutation of the numbers 0 to 2^n-1.

A gate is balanced if its output equals one for exactly one-half of its inputs. If a gate is not balanced, it is said to be unbalanced. Reversible gates are balanced. The only Non

trivial reversible logic gate in classical digital logic is the (1,1) reversible gate, i.e. the inverter. This gate is very important since it does not introduce garbage outputs.

A gate not possessing the reversibility characteristic is said to be irreversible. All multiple-input single output logic gates in classical digital logic are irreversible, e.g. AND, OR, XOR, etc.

A circuit without constants on inputs which includes only reversible gates realizes on all outputs only balanced functions, therefore it can realize non-balanced functions only with garbage outputs. An additional constraint of reversible logic is that the fanout of every signal, including primary inputs, must be one. Currently, logic functions constructed with reversible logic gates are designed in an ad hoc fashion. By definition, all reversible gates have an inverse, i.e. a gate that "undoes" the logic function. A logic gate is said to be self-invertible if the gate is equal to its own inverse. In some works, the terms symmetric or dual are used, and the ambiguous term invertible is also common. For example, a gate is self-invertible if, for every input x, G(G(x)), or, equivalently, $G^2(x)=x$. Obviously, the classic digital inverter is self-invertible.

A gate is linear when all its outputs are linear functions of input variables and it is one through gate if one input variable is also output. A gate is zero-preserving if the all zeros input produces all zeros outputs. A gate is one-preserving if the all ones input produces all ones outputs. A gate is said to be controlled if one or more of its inputs causes a function to be performed, conditionally, on its other inputs. The inputs that determine whether the action is taken or not are called the gate's control signals. A gate is conservative if the Hamming weight (number of logical ones) of its input equals the Hamming weight of its output. A conservative reversible gate is a gate that is both conservative and reversible simultaneously.

A consequence of a gate's reversibility and conservability is that conservative reversible gates are zero preserving and ones-preserving. A conservative reversible logic gate effectively permutes its inputs to form its outputs. Conservative gates are not necessarily

reversible as several inputs with equal Hamming weights can be mapped to a single output of the same Hamming weight.

For example, consider the two-input, two-output logic gate where one output is the logical AND of the inputs, and the other output is the logical OR of the inputs. This logic gate is conservative and irreversible. Magnetic bubble circuits initiated much research on conservative, but irreversible, logic in the 1970s [15,24,25].

3.2.3 The Fredkin Gate

The Fredkin gate is a fundamental concept in reversible and quantum reversible computing. The Fredkin gate can be built from the Billiard Ball Model as described before. Indeed the properties of the Fredkin gate provides an informative overview of the general principles of reversible logic gates and circuits. Few of the other reversible gates are Feynman gate (also called CNOT), Toffoli gate and Peres gate. Refer Appendix II. The following text explains Fredkin Gate.

The Fredkin gate has three input bits and three output bits. We refer to as p, q, r, where r is the control bit. The value of r is unchanged by the action of the Fredkin gate. The control bit controls what happens to the other two bits. If r is the set to 0, then p and q are left alone. If r is set to 1 then p and q are swapped. It is easy to see that the Fredkin gate is reversible.

The schema of the Fredkin gate can be illustrated as below:

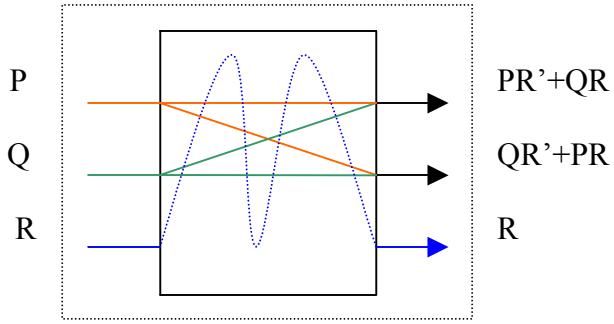


Figure 3.2: Fredkin gate

In this gate the input signals P and Q are routed to the same or exchanged output ports depending on the value of control signal R. Where R'= NOT R.

Other reversible gates like CNOT and Toffoli gates can be constructed from Fredkin gates and vice versa. The Fredkin configured CNOT and Toffoli is explained later.
One can also notice that the sums of the inputs to the gate and outputs from the gate are the same.
Similarly 4*4 Fredkin gates can also be illustrated as follows

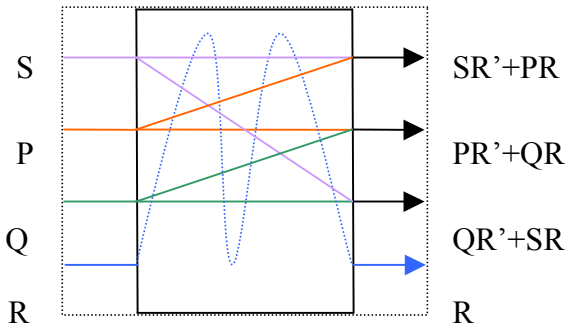


Figure 3.3: 4*4 Fredkin gate

Similarly n*n Fredkin gates can be generalized as follows:

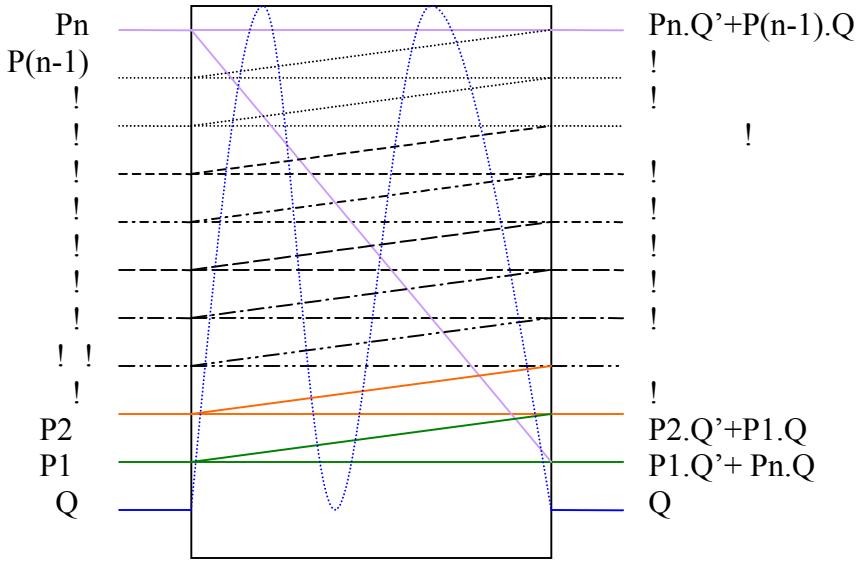


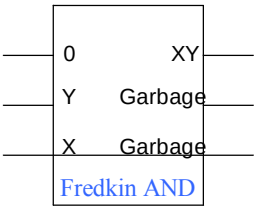
Figure 3.4: n*n Fredkin gate

In addition to its reversibility, the Fredkin gate also has the interesting property that the number of 1s is conserved between the input and the output. In terms of the billiard ball computer, this corresponds to the number of billiard balls going into the Fredkin gate being equal to the number coming out. Thus, it is sometimes referred to as being a conservative reversible logic gate. This is interesting as all laws of natue with the exception of quantum measurement are reversible, and the conservation property analogous to the conservation of mass, energy, etc.

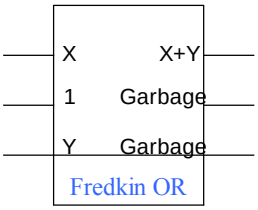
So to simulate these classical gates using Fredkin gate, we make use of two ideas. First we allow the input of ancilla bits in specially prepared states. Second the output of the Fredkin gate contains extraneous garbage not needed for the remainder of the computation. These ancilla and garbage are not directly important to the computation, but their importance lies in the fact that they make the computing reversible. Indeed the irreversibility of the classical AND, OR gates may be viewed as a consequence of the ancilla and garbage bit being hidden.

3.3 Universality

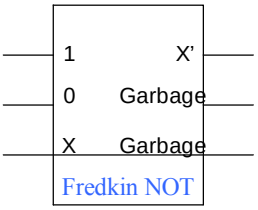
The Fredkin gate is a universal gate. The Fredkin gate can be used to simulate AND, NOT, OR, Crossover and Fanout functions, and thus can be cascaded to simulate any classical circuit whatsoever. Every Boolean function can be build from 3 * 3 Fredkin gates.



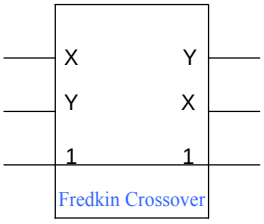
Fredkin AND performs typical AND operation. Additionally uses one ancilla bit ‘0’ and generates two garbage bits.



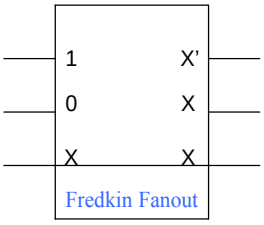
Fredkin OR performs typical OR operation. Additionally uses one ancilla bit ‘1’ and generates two garbage bits.



Fredkin NOT performs typical NOT operation. Additionally uses two ancilla bits '1' and '0' and generates two garbage bits.



Fredkin Crossover performs a crossover operation (which means its replaces the 'X' with 'Y' and vice versa). Additionally uses one ancilla bit and generates one garbage bit.



Fredkin fanout performs a fanout operation (which means generates more of the same input). Additionally uses two ancilla bits '1' and '0' and generates one garbage bit which is an inverse of the input.

Figure 3.5: Representation of Fredkin gates

In Fredkin gates we are considering which is the 3*3 gates

Number of ancilla bits needed = 1

Number of garbage bits produced =2

From the observation of the configurations possible with the Fredkin gates we find that given any classical circuit computing a function f(x), we can built a reversible circuit made entirely of Fredkin gates, which on input of x, together with some ancilla bits in standard state, computes f(x) with some garbage output g(x). So (x,a) gives (f(x),g(x)).

Thus the basic definition of universality, ie being able to compute any function with the Fredkin gate has been satisfied.

Chapter 4
Logic Circuits Using Fredkin Gate

4.1 Adders

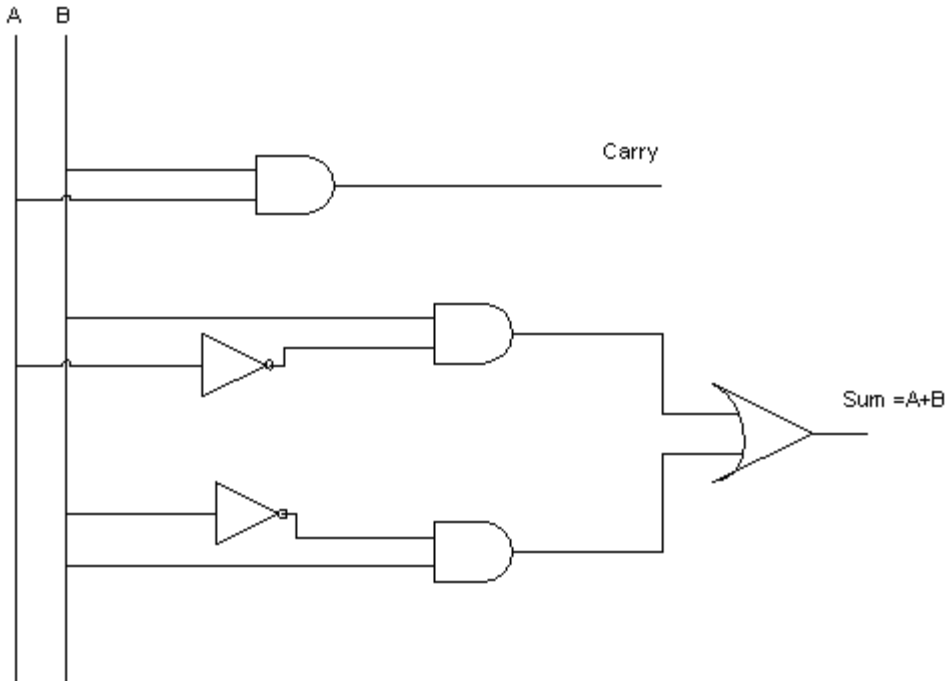


Figure 4.1: Half Adder using conventional Boolean gates

It can be easily noticed from our definitions of Fredkin gates, that we can replace the elements of this circuit ie. the irreversible logic gates by reversible Fredkin gates. Therefore the number of fundamental gates of any circuit will be the same. No new construction have to be done apart from the use of ancilla bits and the generation of the garbage bits which have to be suitably taken care of.

Number of garbage bits generated = 12

Number of ancilla bits used = 8

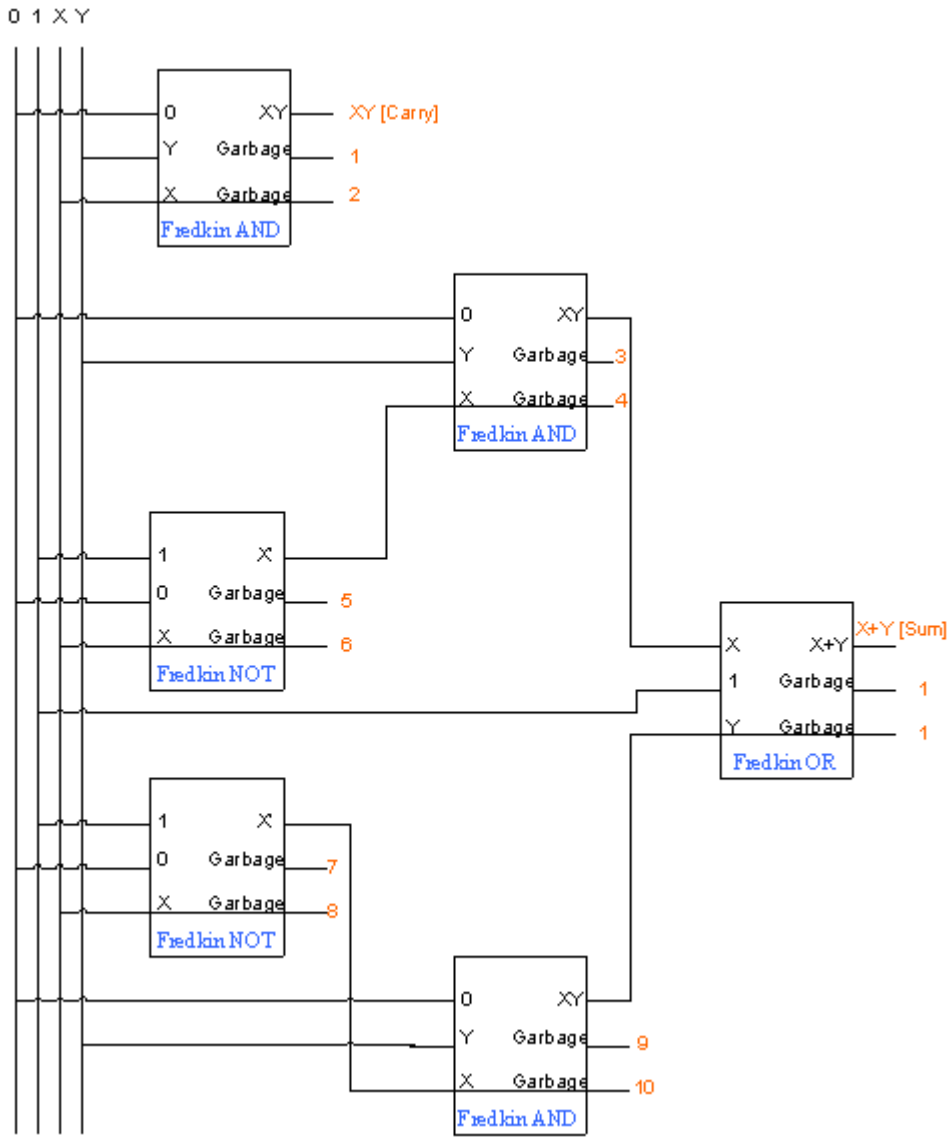


Figure 4.2: Half Adder configured using Fredkin gate

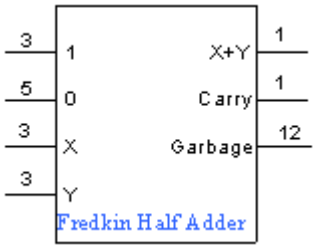


Figure 4.3: Representation of Half Adder configured through Fredkin gates

4.2 Full Adder

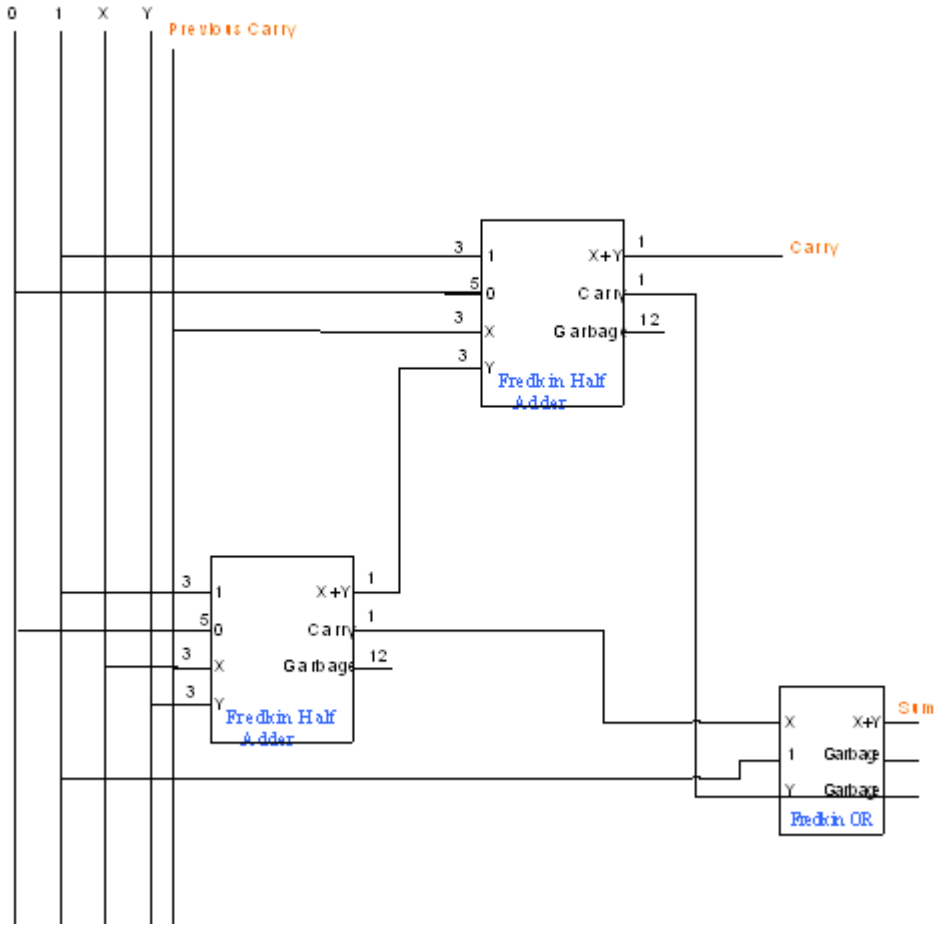


Figure 4.4: Full Adder configured using Fredkin gates

Number of garbage bits generated = 25

Number of ancilla bits used = 17

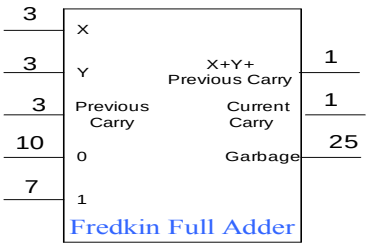


Figure 4.5: Representation of Full Adder configured through Fredkin gates

4.3 Parallel Adder

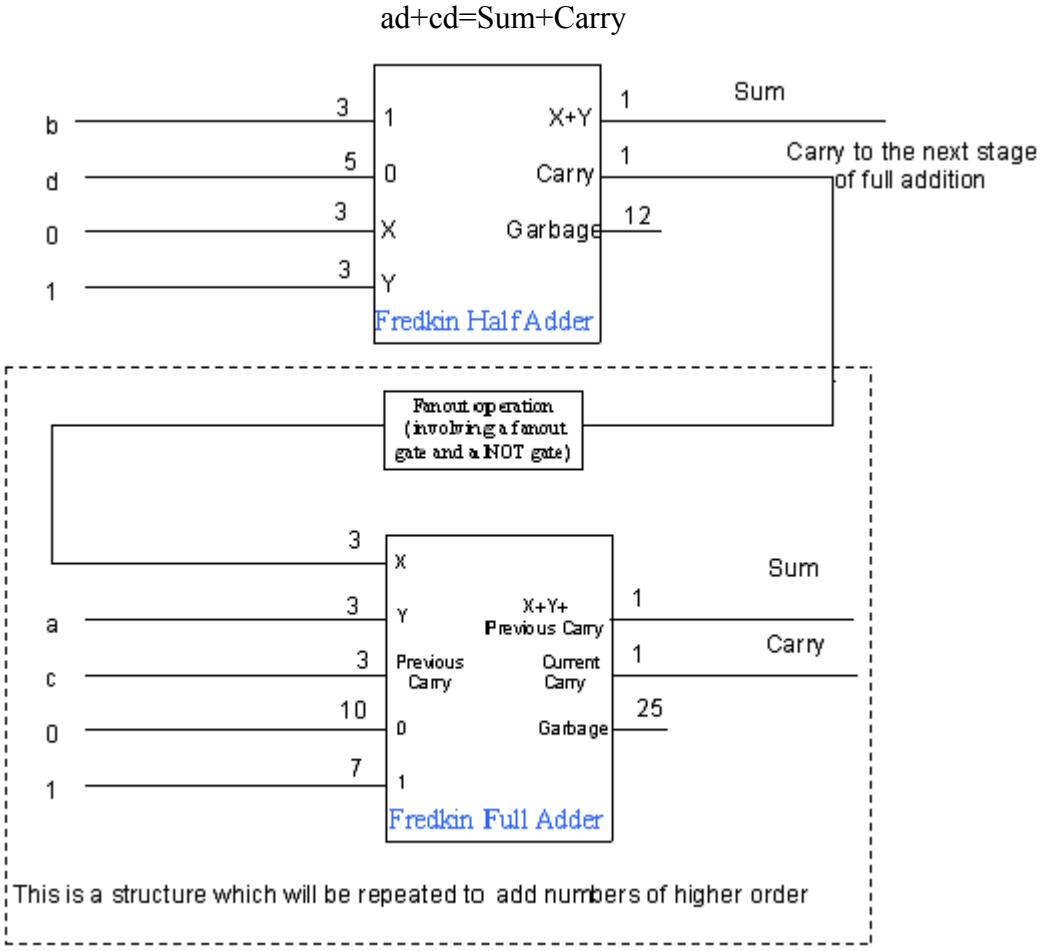


Figure 4.6: Parallel Adder configured using Fredkin gates

Number of garbage bits generated = 40 (3 from fanout operation)

Number of ancilla bits used = 29 (4 from fanout operation)

Therefore for two digit addition

Number of ancilla bits used = 29

For every digit more 21 ancilla bits are added

Therefore for three digit addition

Number of ancilla bits used = 46

However for single digit addition

Number of ancilla bits used = 8

Therefore generalizing the findings mathematically, we get

$N_a = 8 + 21*(n-1)$

Where N_a is the number of ancilla bits used

n is number of digits in the operand of highest value in the addition operation

$N_g = 12 + 28*(n-1)$

Where N_g is the number of garbage bits generated

n is the number of digits in the operand of highest value in the addition operation.

4.4 Multipliers

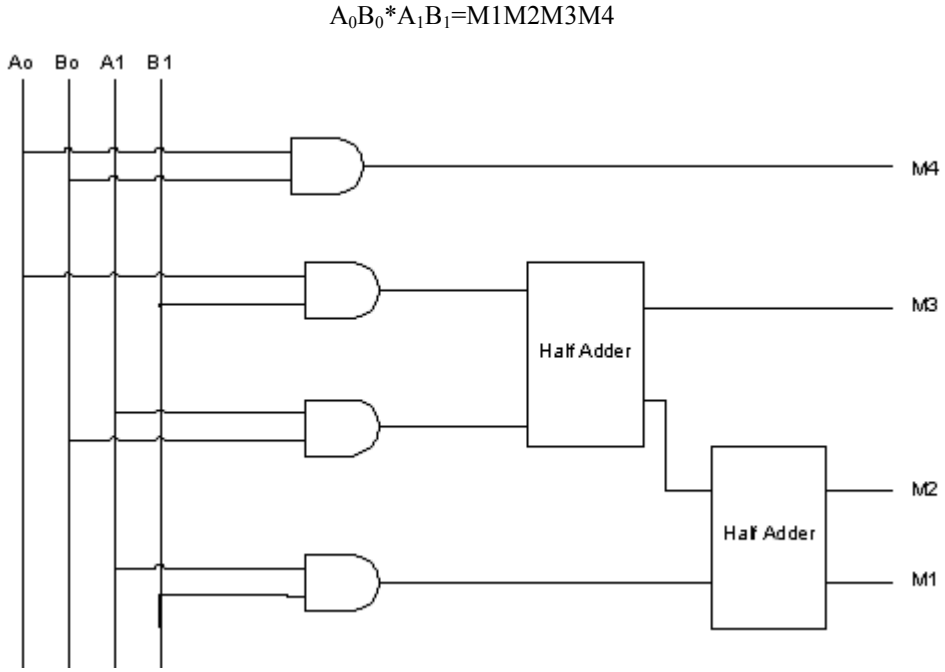


Figure 4.7: Observation of Schema of 2*2 Multiplier

$A_0B_0*A_1B_1=M1M2M3M4$

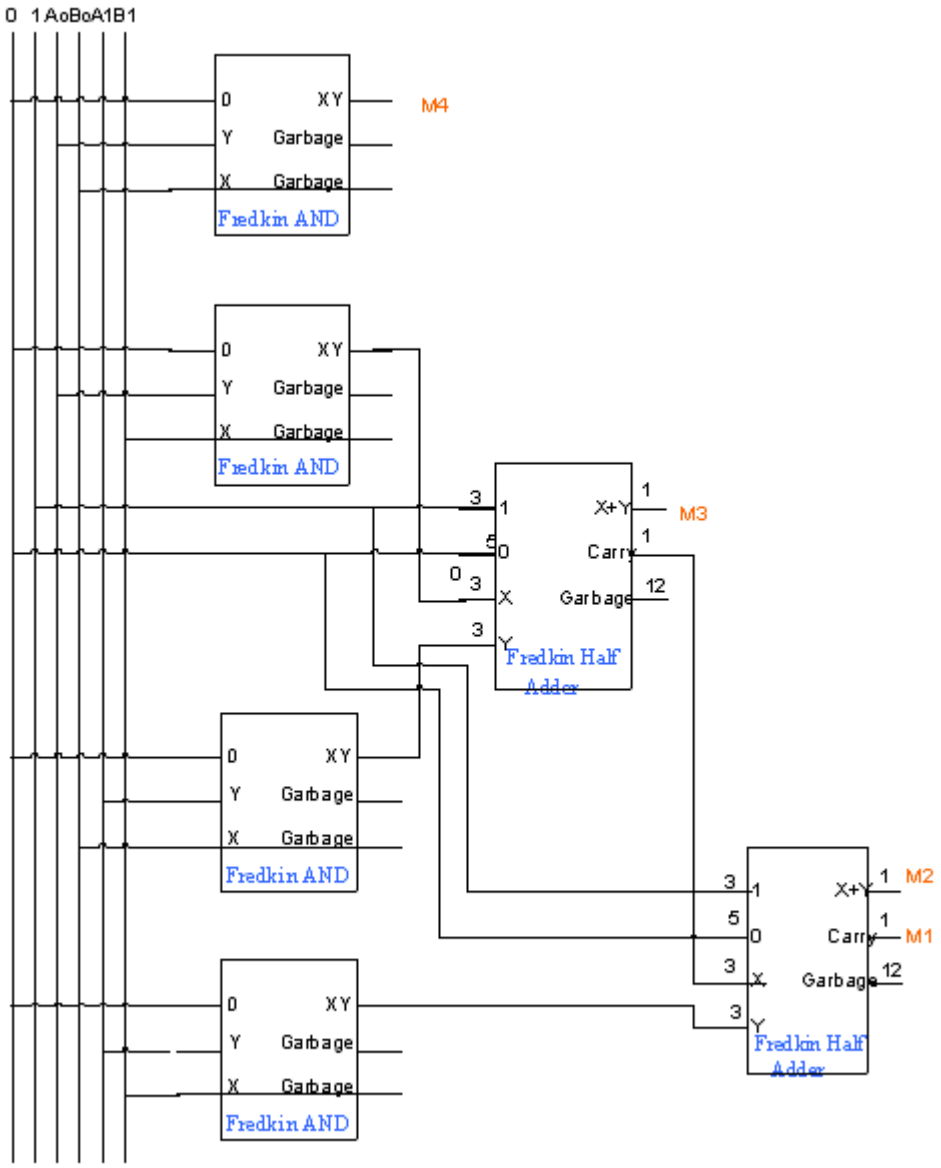


Figure 4.8: 2-Bit Multiplier configured through Fredkin gate

Number of garbage bits generated = 32

Number of ancilla bits used = 20

Similarly we could implement a 4*3 multiplier, we shall merely consider its schema and use it for the generalization of a formula for the bits used and generated.

$A_0A_1\ A_2 * B_0B_1\ B_2B_3 = M1M2M3M4M5M6M7$

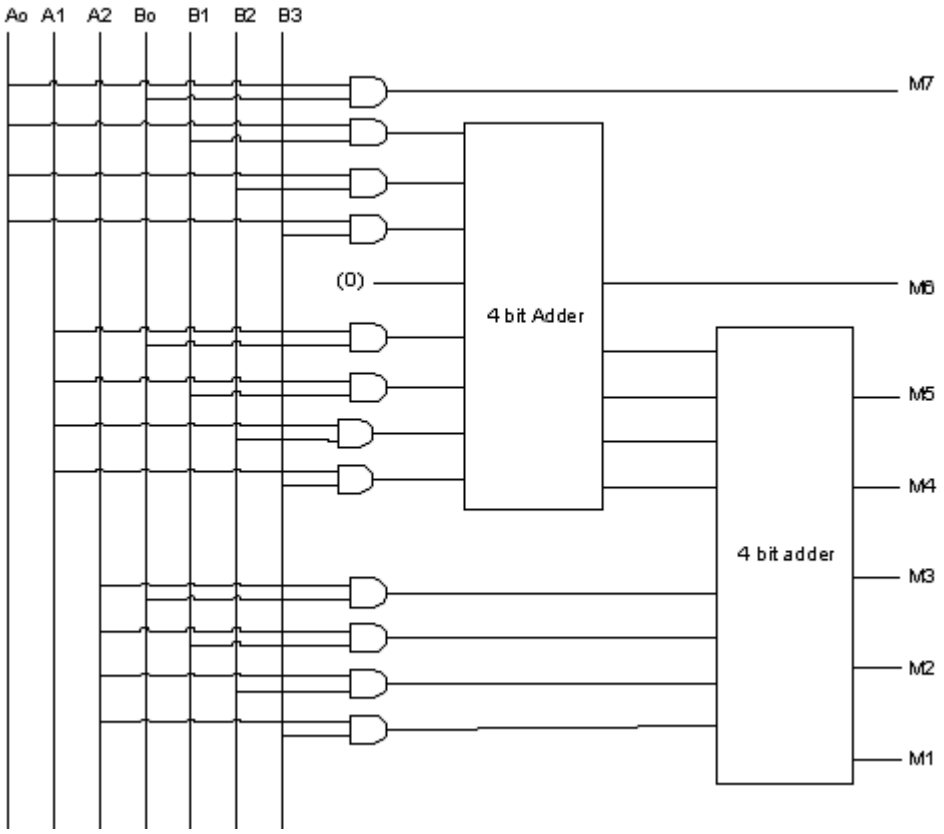


Figure 4.9: Observation of a 4*3 Multiplier

From the observation of the 4*3 multiplier, we can construct a general m*n multiplier as below:

A0…..Am*B0…..Bn=M1……M(m+n)
Assume: n>m
Nh=n
Nl=m

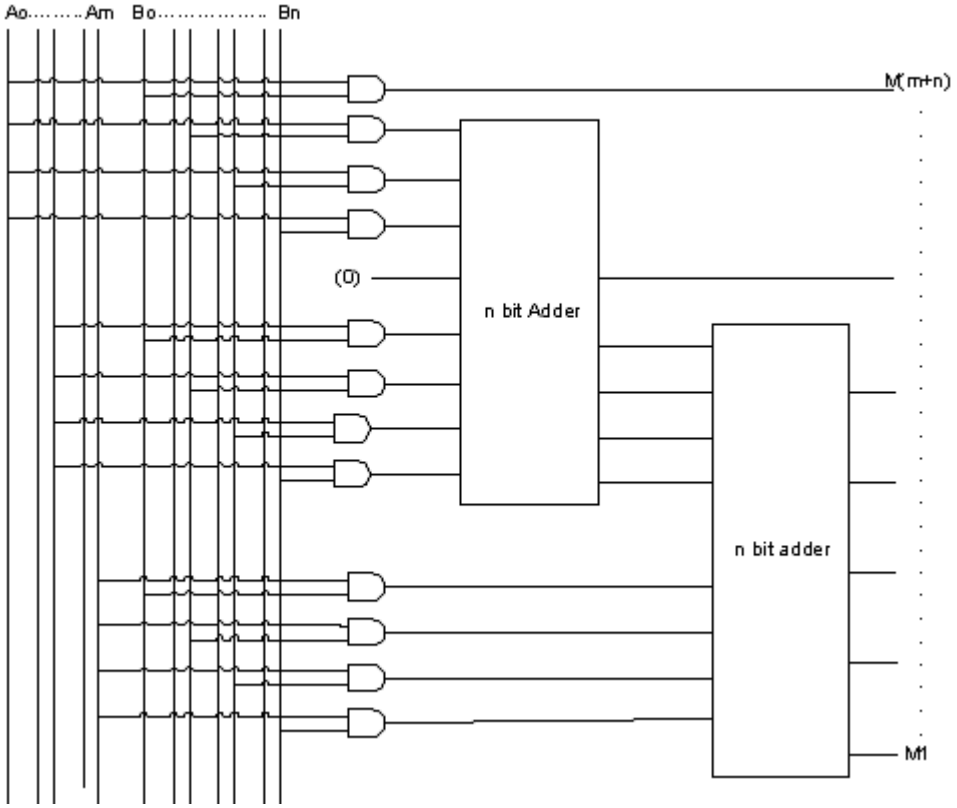


Figure 4.10: Observation of a m*n multiplier

From observing the construction of the n*m multiplier we find that we can generalize and formulate the number of ancilla bits used and garbage bits generated.

Let N_l be the number of digits of the lesser operand

N_h be the number of digits of the greater operand

N_{fa} be the number of full adders used

From the observations

Number of AND gates used = $N_l * N_h$

Number of Adders used = $N_l - 1$

{ $N_l = 2$ if Half adders are used

$N_l > 2$ if Full adders of N_h order ar used }

Every AND gates generates 2 garbage bits

Every Full Adder gives {12 garbage bits if $N_l = 2$

$12 + 25*(N_{fa} - 1)$ garbage bits if $N_l > 2$}

Every AND gate used 1 ancilla bit

Every Full Adder uses {8 ancilla bits if $N_l = 2$

$8 + 17*(N_{fa} - 1)$ ancilla bits if $N_l > 2$}

Therefore, in a m*n multiplier

If $N_l > 2$

Number of garbage bits generated = $2*(N_l * N_h) + (N_l - 1)*12$

Number of Ancilla bits used = $(N_l * N_h) + (N_l - 1)*8$

If $N_l > 2$

Number of garbage bits generated = $2*(N_l * N_h) + (N_l - 1)*(12 + 25*(N_{fa} - 1))$

Number of Ancilla bits used = $(N_l * N_h) + (N_l - 1)* (8 + 17*(N_{fa} - 1))$

4.5 Subractor

Rather than construct a separate piece of hardware, we can use our adder circuit to implement subtraction. By simple rules of mathematics, a + b = a + (-b), therefore if we can generate the value of -*B* we can feed this input in to our adder circuit to produce the required function. Generating the 2's complement form of a number is simple. It just involves inverting each bit, and adding 1 to the result.

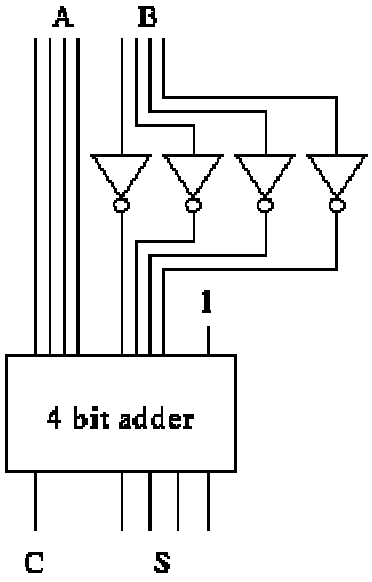


Figure 4.11: Subtractor

From the observation of the above construction, we can formulate the number of ancilla bits used and garbage bits generated.

4.6 Divider

The block diagram for implementation of binary division is shown below. The M-register holds the divisor and remains unchanged throughout the division procedure. The double length register formed by A and Q registers initially hold the double length dividend. The outputs of A and M register is connected to the inputs of the parallel adder/subtracter, the output of which are connected to the input of the A register. The signs of the divisor and dividend initially determine the adder/subtract control. The signs of the divisor and remainder formed at the outputs of the adder/subtracter then determine the quotient digit which is entered into the Q register. This is carried out at a time when the entire contents of A and Q registers are shifted one bit position to the left. In the following steps of the procedure the signs of the divisor and the remainder determine the add/subtract control and the quotient digit. The quotient digits are therefore formed one digit at a time starting from the most significant digit progressing to the least significant magnitude digit.

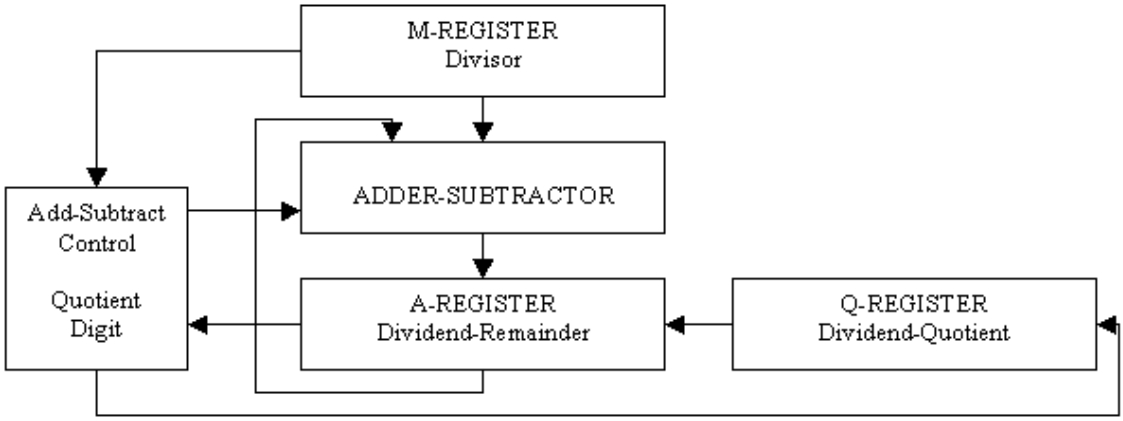


Figure 4.12: Block diagram of an arithmetic unit for binary division

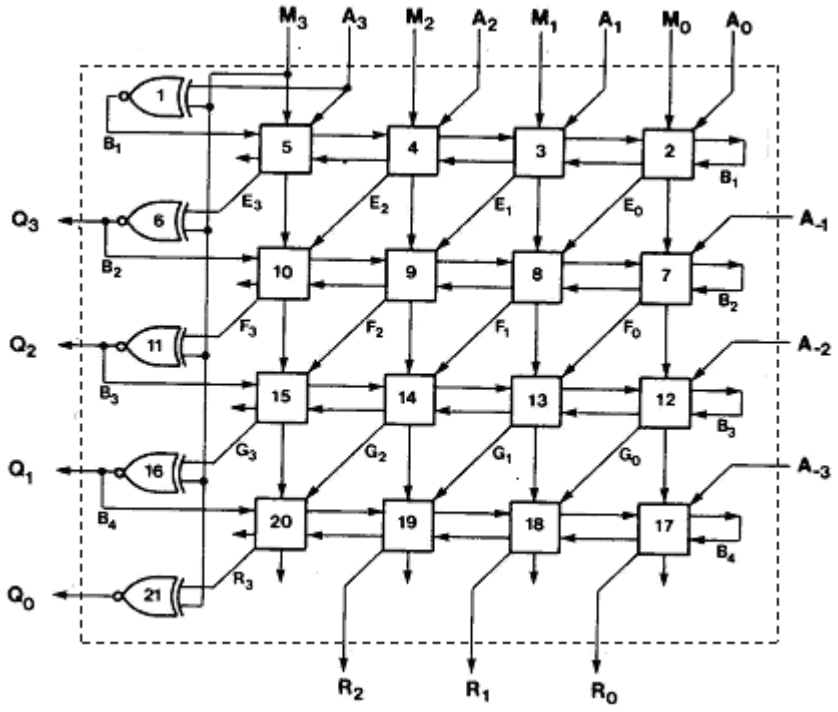


Figure 4.13: Circuit for an iterative array for non-restoring, two's complement division fractions

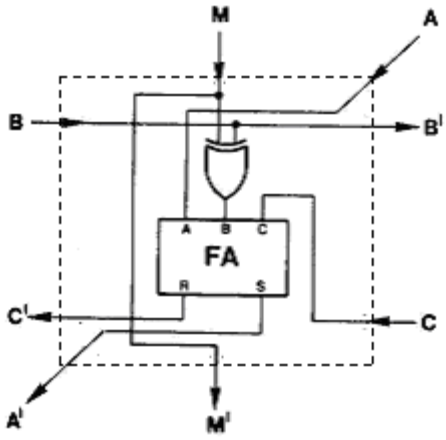


Figure 4.14: Circuit of basic cell used in the above diagram

The XOR gates when realized with the help of primary basic gates comprises of 2 NOT gates, 2 AND gates and an OR gate. We can easily compute the number of garbage bits generated and the ancilla bits used to make this a reversible XOR.

Number of ancilla bits used: $Na_{XOR} = 2*(Na_{NOT}) + 2*(Na_{AND}) + Na_{OR}$

$= 2*2 + 2*1 + 1$

$= 7$

Number of garbage bits generated = $2*(Ng_{NOT}) + 2*(Ng_{AND}) + Ng_{OR}$

$= 2*2 + 2*2 + 2$

$= 10$

Full Adder uses 17 ancilla bits and generates 25 garbage bits.

Refer to section on 'Logic gates configured through Fredkin gates' for details.

Therefore a basic cell used in the divider circuit:

Number of ancilla bits used: Na = 7+ 17 = 24

Number of garbage bits generated: Ng = 10 + 25 =35

For the divider shown in the above diagram, we find:

Na = 24 * 16 + 7 * 5 = 419

Ng = 35 * 16 + 10 *5 = 610

ie.

Na = Na(cell) * (no. of cells) + Na(XOR) * (no. of XOR's)

Ng = Ng(cell) * (no. of cells) + Ng(XOR) * (no. of XOR's)

Where

Na(cell) = Na(XOR) + Na(Full adder)

Ng(cell) = Ng(XOR) + Ng(Full adder)

4.7 CNOT gate

The controlled NOT gate is a gate that acts on two qubits. The first qubit is called a control qubit, and the second qubit is the data qubit. If the control qubit is $|0\rangle$ then the data qubit is left alone. If the control qubit is $|1\rangle$ the data qubit is flipped. The gate, for which we are going to use the $\oplus$ symbol, can be therefore described by the following equations:

$$\begin{aligned} \oplus: \quad & |0\rangle|0\rangle \rightarrow |0\rangle|0\rangle \\ \oplus: \quad & |0\rangle|1\rangle \rightarrow |0\rangle|1\rangle \\ \oplus: \quad & |1\rangle|0\rangle \rightarrow |1\rangle|1\rangle \\ \oplus: \quad & |1\rangle|1\rangle \rightarrow |1\rangle|0\rangle \end{aligned} \tag{25}$$

or, using the computational basis:

$$\begin{aligned} \oplus: \quad & |0\rangle \rightarrow |0\rangle \\ \oplus: \quad & |1\rangle \rightarrow |1\rangle \\ \oplus: \quad & |2\rangle \rightarrow |3\rangle \\ \oplus: \quad & |3\rangle \rightarrow |2\rangle \end{aligned} \tag{26}$$

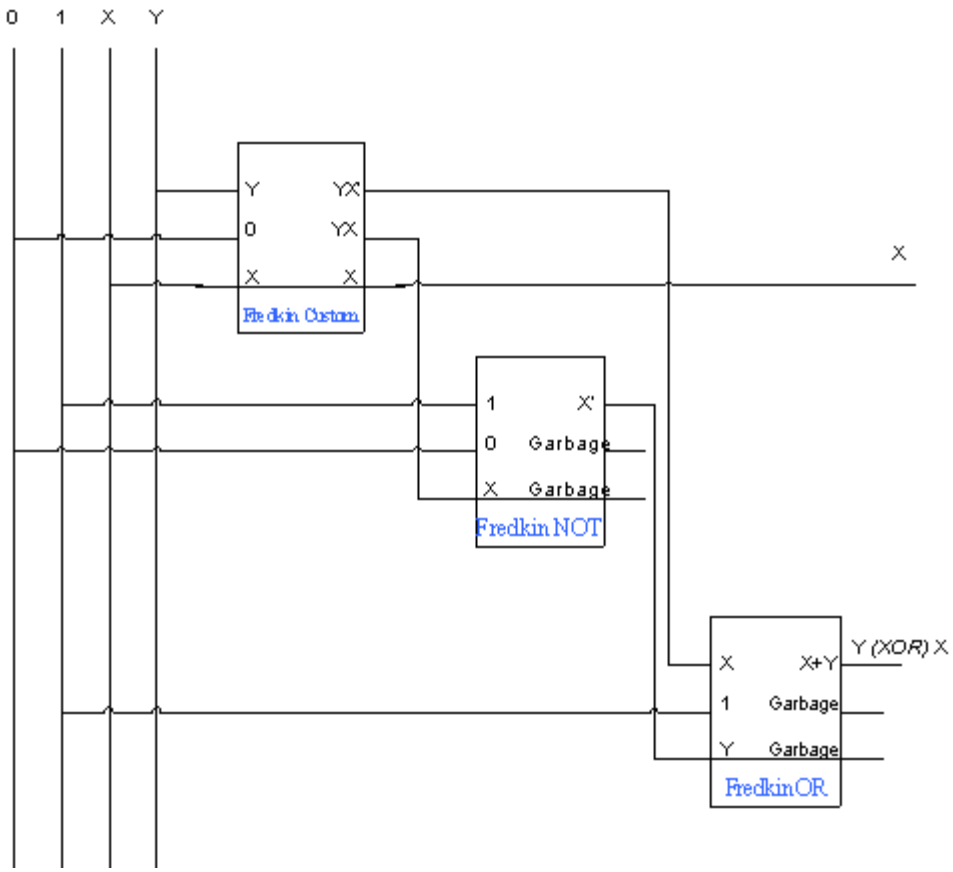


Figure 4.15: Implementing CNOT gates using Fredkin gates

Number of Ancilla bits used = 4

Number of garbage bits generated = 4

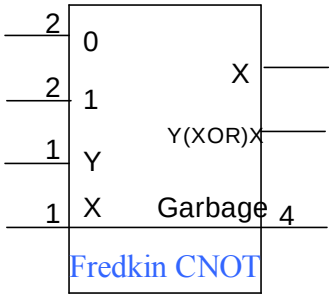


Figure 4.16: Representation of the Fredkin configured CNOT gate

4.8 Toffoli gate

The Toffoli gate is a *controlled-controlled-NOT* **gate and its** diagramatic representation is as follows:

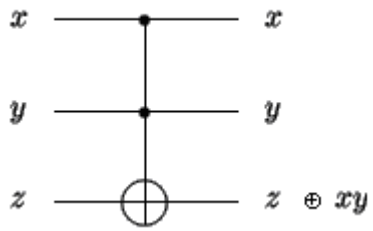


Figure 4.17: Circuit for Toffoli gate

This gate flips *z* if *x* and *y* are both 1 and leaves *z* alone if they aren't. *x* and *y* themselves remain unchanged. The nonlinearity of the gate is evident in the formula that describes the bottom output: $z \oplus xy$.

The Toffoli gate can be used as a *universal* gate for Boolean logic if

1. fixed input bits can be provided on some inputs,
2. some output bits can be ignored.

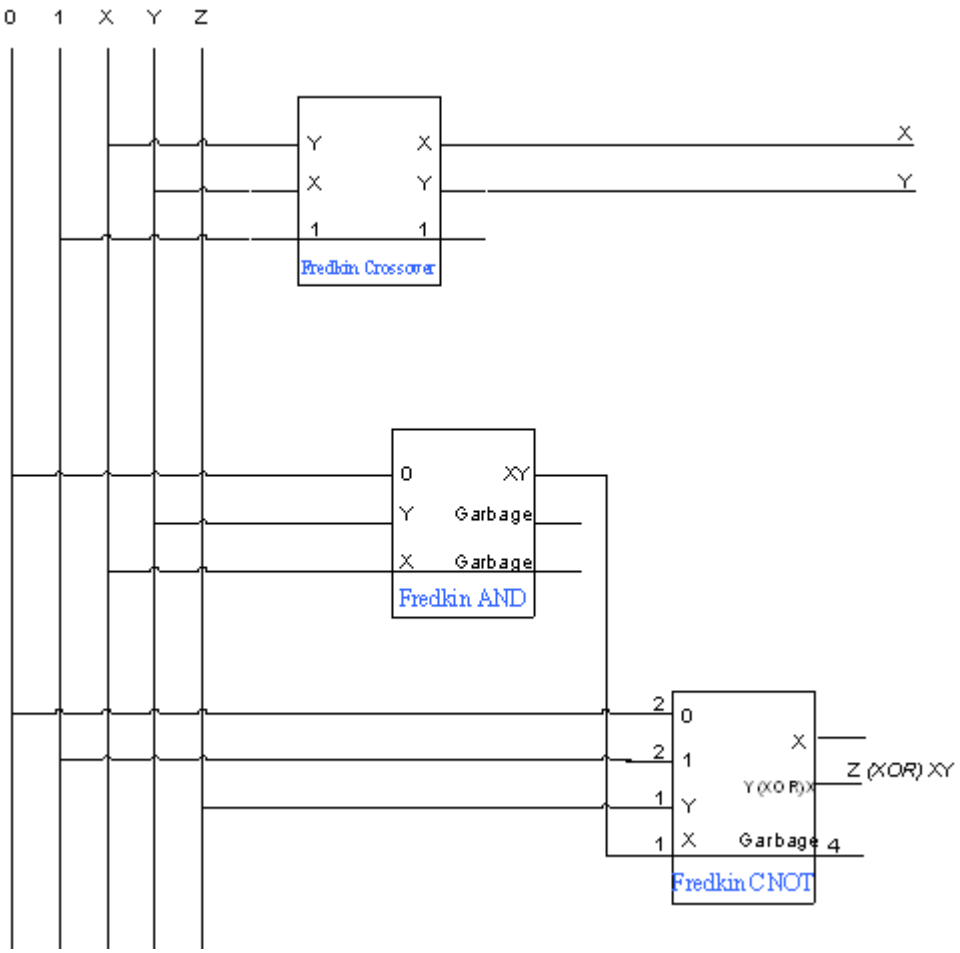


Figure 4.18: Implementation of Toffoli gate using Fredkin gate

Number of ancilla bits used = 6

Number of garbage bits generated = 8.

Chapter 5
Reversible Garbage Collection

5.1 Why is Garbage Collection Needed?

The Reversible computing thus demonstrated unfortunately produces unwanted garbage bits. With some modification it turns out to be possible to perform the computation so that any garbage bits produced are in a standard state.

This construction is crucial for Quantum computation, because garbage bits whose value depends upon the input will destroy the interference properties crucial to Quantum computation. To understand how this works it is convenient to assume that the ancilla bits a all start out as 0s, with NOT gates being added where necessary to turn the ancilla 0s into 1s. It will also be convenient to assume that the classical Controlled-NOT gate is available, which can be thought of as a reversible copying gate or FANOUT, which leaves no garbage at the output. As shown in the previous chapter all gates can be constructed through Fredkin gates.

Fortunately, the primitive ERASE is not absolutely essential in computation. To see why, consider what is required to compute arbitrary functions using reversible logic (where the primitive ERASE is forbidden). Landauer showed how any function could be made one-to-one by keeping a copy of the input.

One way of performing reversible logic is by the Toffoli gate. The output of this gate may be decomposed into various gates:

$$B \oplus (AC) = \begin{cases} AC, & for \quad B = 0 (AND) \\ A \oplus B, & for \quad C = 1 (XOR) \\ \overline{B}, & for \quad A = C = 1 (NOT) \\ A, & for \quad B = 0, C = 1 (FANOUT) \end{cases} \tag{27}$$

where AB represents an AND gate, $A \oplus B$ represents an XOR gate and $\overline{B}$ represents a NOT gate. We see that this gate is universal, because it performs AND, XOR, NOT or FANOUT depending on its inputs. A combination of many such gates could then be used for any computation and would still be reversible.

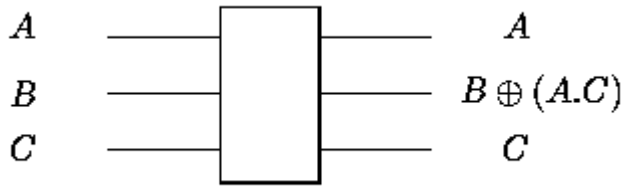


Figure 5.1: Three-input three-output universal reversible Toffoli gate.

This gate is clearly reversible since a second application of it retrieves the original input.

As noted by Landauer, this procedure leads to an immediate problem because of the absence of the primitive ERASE. The more gates we employ, the more ``junk" bits we accumulate: At each gate we must save input bits in order to preserve reversibility. In other words a computer built out of reversible logic instead of conventional, irreversible logic gates would behave like

$$f : a \to (a, j(a), f(a)), \tag{28}$$

with many extra junk bits $j(a)$.

5.2 According to Bennett

Bennett solved this problem by showing that the junk bits could be *reversibly* erased at intermediate steps with minimal run-time and memory costs [9]. Bennett's solution may be understood in terms of the following procedure:

$$
\begin{aligned}
&f : a \to (a, j(a), f(a)) \\
&FANOUT : (a, j(a), f(a)) \to (a, j(a), f(a), f(a)) \\
&f\uparrow : (a, j(a), f(a), f(a)) \to (a, f(a)),
\end{aligned}
\tag{29}
$$

where $f\uparrow$ denotes uncomputing f, as opposed to computing f^{-1}. First, f is computed, producing both junk bits and the desired output. Then the FANOUT gate is applied to duplicate the output. Finally, we uncompute the original function f by running its computation backwards. This procedure removes the junk bits and the original output. The duplicate, however, remains!

5.3 According to Nielsen-Chuang

With the additional NOT gates appended at the beginning of the circuit, the action of computation may be written as(x,0) = (x, f(x), g(x)). We could also have added CNOT gates to the beginning of the circuit, in order to create a copy of x which is not changed during the subsequent computation. With this modification, the action of the circuit can be written as (x, 0, 0) = (x, f(x), g(x)).

This is a useful way of writing the action of the reversible circuit, because it allows an idea known as uncomputation to be used to get rid of the garbage bits, for a small cost in the running time of the computation. The idea is the following. Suppose we start with the four register computer in the state (x, 0, 0, y). The second register is used to store the result of the computation, and the third register is used to provide workspace for the computation, that is the garbage bits g(x). The use of the fourth register is described below, and we assume it starts in an arbitrary state y.

We begin as before, by applying a reversible circuit to compute f, resulting in the state (x, f(x), g(x), y). Next we use the CNOTs to add to the result f(x) bitwise to the fourth register leaving the machine in a state (x, f(x), g(x), y$*$f(x)). However all the steps used to compute f(x) were reversible and did not attract the fourth register, so by applying the reverse of the circuit used to compute f, we come to the state (x, 0, 0, y$*$f(x)). Typically

we omit the ancilla 0s from the description of the function evaluation, and just write the action of the circuit as $(x,y) = (x, y^{*}f(x))$[17].

5.4 Resource Overhead

To analyse what resource overhead is involved in doing reversible computation by this method, we need to count the number of extra ancilla bits needed in a reversible circuit, and compare the gate counts with classical methods. It is clear that the number of gate in a reversible circuit is the same as in an irreversible circuit to within the constant factor which represents the number of Fredkin gates needed to simulate a single element of the irreversible circuit, and an additional factor of two for uncomputation, with an overhead for the extra CNOT operations used in reversible computation which is linear in the number of the bits involved in the circuit. Similarly the number of ancilla bits required scales at most linearly with the number of gates in the irreversible circuit, since each element in the irreversible circuit can be simulated using a constant number of ancilla bits. Hence the complexities such as P or NP are also the same whichever mode of computation is used.

Chapter 6

Comparative Findings on Reversible Against Classical Circuits

The following are the additional bits which will be used in order to achieve reversible computing

6.1 Basic Gates

- **AND:**

  Number of ancilla bits needed = 1

  Number of garbage bits generated = 2

- **OR:**

  Number of ancilla bits needed = 1

  Number of garbage bits generated = 2

- **NOT:**

  Number of ancilla bits needed = 2

  Number of garbage bits generated = 2

Since the basic gates require extra bits and generate extra bits, the circuits built with them will have corresponding increase in those bits.

6.2 Logic Circuits

- **Adders**

  Half adder:

  Number of garbage bits generated = 12

  Number of ancilla bits used = 8

Full Adder:

Number of garbage bits generated = 25

Number of ancilla bits used = 17

Arbitrary addition (Paralell Adder):

$N_a = 8 + 21*(n-1)$

Where N_a is the number of ancilla bits used

n is number of digits in the operand of highest value in the addition operation

$N_g = 12 + 28*(n-1)$

Where N_g is the number of garbage bits generated

n is the number of digits in the operand of highest value in the addition operation

- **Multipliers**

  2*2 Multiplier:

  Number of garbage bits generated = 32

  Number of ancilla bits used = 20

  m*n Multiplier:

  Let N_l be the number of digits of the lesser operand

  N_h be the number of digits of the greater operand

  N_{fa} be the number of full adders used

  If $N_l > 2$

  Number of garbage bits generated = $2*(N_l * N_h) + (N_l - 1)*12$

  Number of Ancilla bits used = $(N_l * N_h) + (N_l - 1)*8$

  If $N_l > 2$

  Number of garbage bits generated = $2*(N_l * N_h) + (N_l - 1)*(12 + 25*(N_{fa} -1))$

  Number of Ancilla bits used = $(N_l * N_h) + (N_l - 1)* (8 + 17*(N_{fa} -1))$

- **CNOT gate:**

  Number of Ancilla bits used = 4

  Number of garbage bits generated = 4

- **Toffoli gate:**

  Number of ancilla bits used = 6

  Number of garbage bits generated = 8

Hence we find that the garbage bits generated need to be destroyed, otherwise it would occupy unnecessary memory, and may overload the system. The destruction of the garbage bits as we saw in the previous section on 'Reversible garbage collection', needs 2 stages of operation, hence apart from the number of number of stages of operation in the classical circuit, we need two more operations for every garbage bit.

Though it is true that production of garbage bits are disadvantageous thereby increasing numbers of steps to compute, the, overall advantage of making the computation reversible is that heat dissipation is reduced and new techniques like Quantum computing can be approached with this foundational change in the way processing is done.

Chapter 7

Optimising Reversible Circuit

The main resource over head involved in reversible circuit is the use of garbage bits and ancilla bits to attain reversibility. If every time that we start a new computation we supply a circuit with fresh constants and throw away the garbage (i.e., treat the garbage signals as thermal modes), then we will have dissipated energy. To avoid throwing away that waste, we might consider reprocessing garbage signals so as to use them as known inputs to a subsequent computation. In order to do so we must explicitly figure out their dependence on the argument; but this will require in general a second computation as complicated as the one that generated the garbage signals to begin with. Thus, while we strive to maintain and use the garbage generated by the original computation, the new computation is likely to generate additional garbage which in general will depend on the argument in an even more complex way.

As garbage and ancilla bits can't be avoided in achieving reversibility so the best approach would to design the reversible circuit keeping the garbage bits minimum..

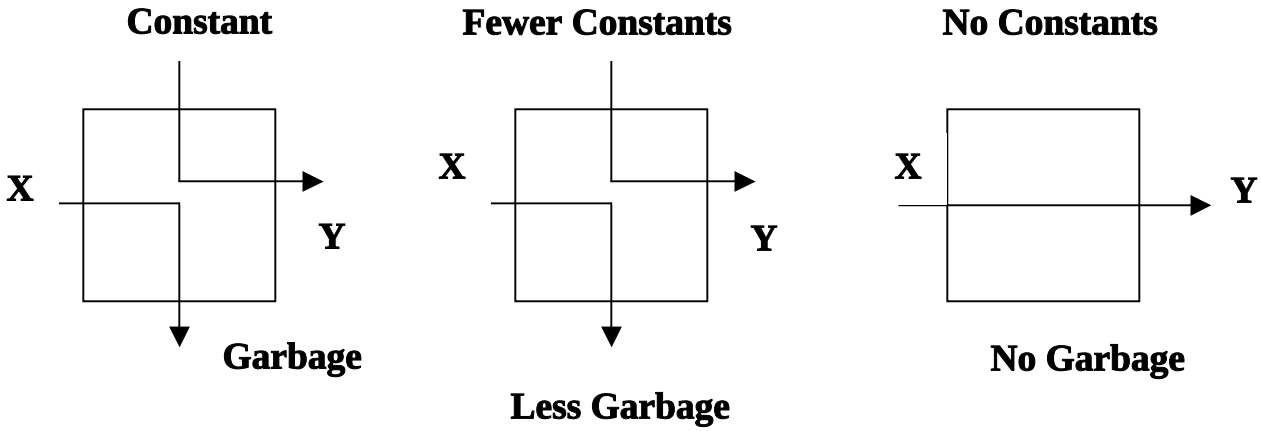


Figure 7.1: Comparison of Reversible Circuits on the basis of Input and Output

7.1 Efficient Reversible Circuit Synthesis

The main constraints of reversible logic are as follows [16]:

a) The fan-out of every signal, including primary inputs, is one.

b) The graph of the reversible circuit must be a directed acyclic graph, that is, there will not exist any loops of gates and internal loops in gate.

c) Many of the practical functions are themselves irreversible and need to make reversible before implementing them with reversible gates.

The main guidelines for efficient reversible logic synthesis are as follows:

a) Use as many outputs of every gate as possible, and thus minimize the garbage outputs

b) Do not create more constant inputs to gates unless necessary i.e. use constant bits efficiently.

c) Avoid leading output signals of gates to more than one input, because each such fan out of two requires adding one copying circuit [16].

d) Select appropriate reversible gates and use as less number as possible to reduce complexity of circuit. When number of gates is reduced the circuit requires less timing and less logical operations. If possible customise the gates according to requirement.

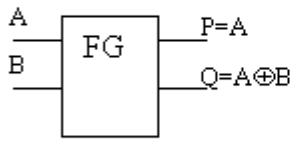
f) Perform data flow analysis and based on that analysis, connect and position the gates. Though it will turn out to be complex for large circuits and a well designed algorithm is required. But for small circuits it can be done manually.

g) Compare the possible circuit configurations on the basis of number of gates, number of garbage outputs and complexity of the circuit to select optimal circuit.

Steps to follow

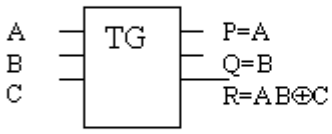
Step 1: If (Current_Circuit is Better than Best_Circuit) then goto Step 2.
Step 2: If Current_Circuit is optimal then END;
else
Best_Circuit=Current_Circuit;
Step 3: Repeat steps 4, 5, 6 for count1=1 to Total_Gates
Step 4: Repeat steps 5, 6 for count2=1 to Total_Gate_combinations
Step 5: Find Best Combination of gates using total Gate and current combination.
Step 6: Using Best Combination of Gates find Current _Circuit; goto Step 1

So, the reversible circuit can be optimised in terms of number of gates and number of garbage bits.This is shown with the help three full adder circuits using the following gates.



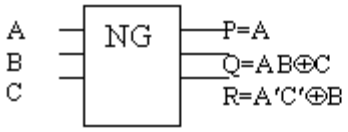
A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Figure 7.2: Feynman Gate with truth table



A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Figure 7.3: Toffoli Gate with truth table



A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	0	0

Figure 7.4: Customised New Gate

The Full Adder circuits are as follows:

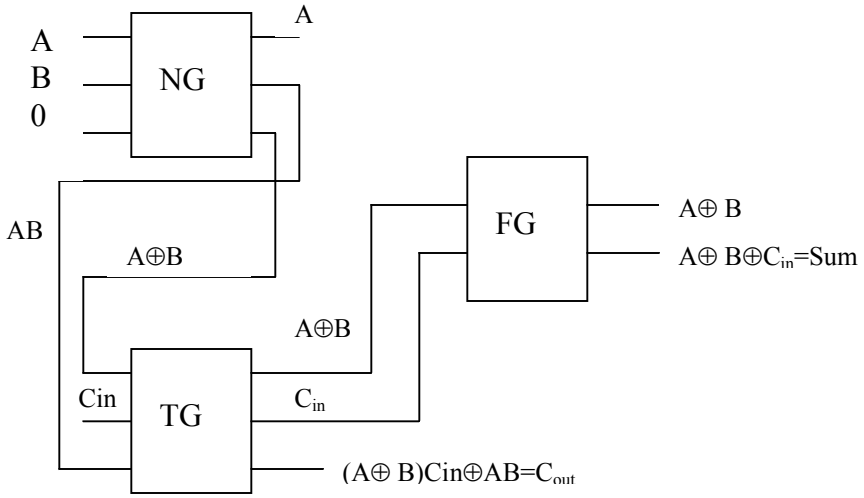


Figure 7.5: Full Adder Circuit_1

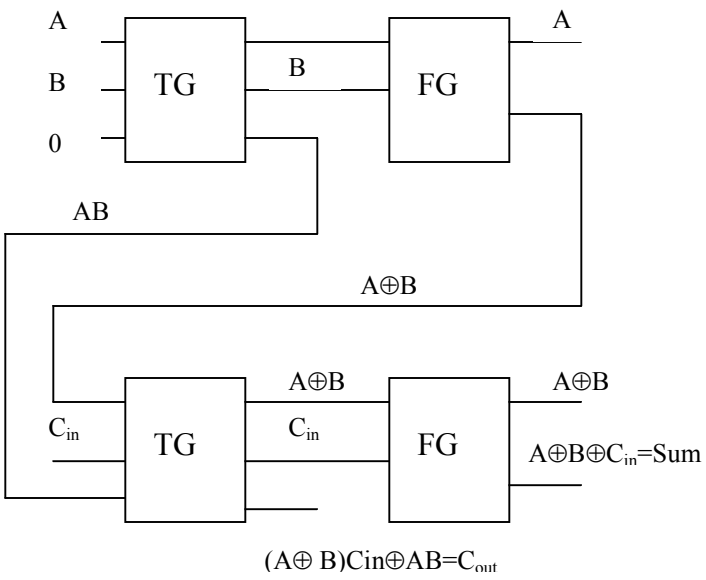


Figure 7.6: Full Adder Circuit_2

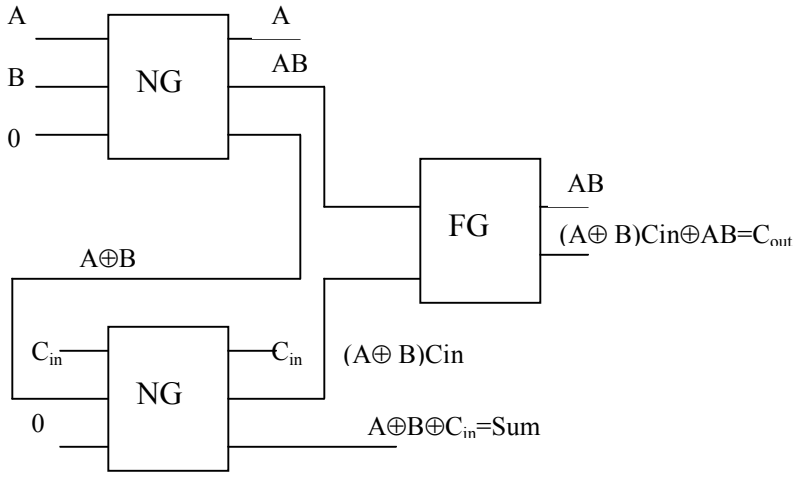


Figure 7.7: Full Adder Circuit_3

7.1.1 Comparison of Reversible Full Adder Circuits

	No of Gates	No. of Garbage Outputs	Unit clock Cycle
Full Adder Circuit_1	3	2	3
Full Adder Circuit_2	4	2	4
Full Adder Circuit_3	3	3	3

Table 7.1: Comparison of reversible Full Adder Circuits

Let
U=Unit clock cycle
α=A two-input EX-OR gate calculation
β=A two-input AND gate calculation
χ=A NOT calculation
N_E= No of Error corrections

T_C and T_L are the total clock cycle and total logical calculation respectively. So for

Circuit_1: $T_C = (3+N_E)U$
$T_L = 4\alpha+3\beta+3\chi$

Circuit_2: $T_C=(4+N_E)U$
$T_L=4\alpha+2\beta$

Circuit_3: $T_{C=}(3+ N_E)U$
$T_{L=}5\alpha+4\beta+6\chi$

The following results were obtained after VHDL simulation of respective circuits:

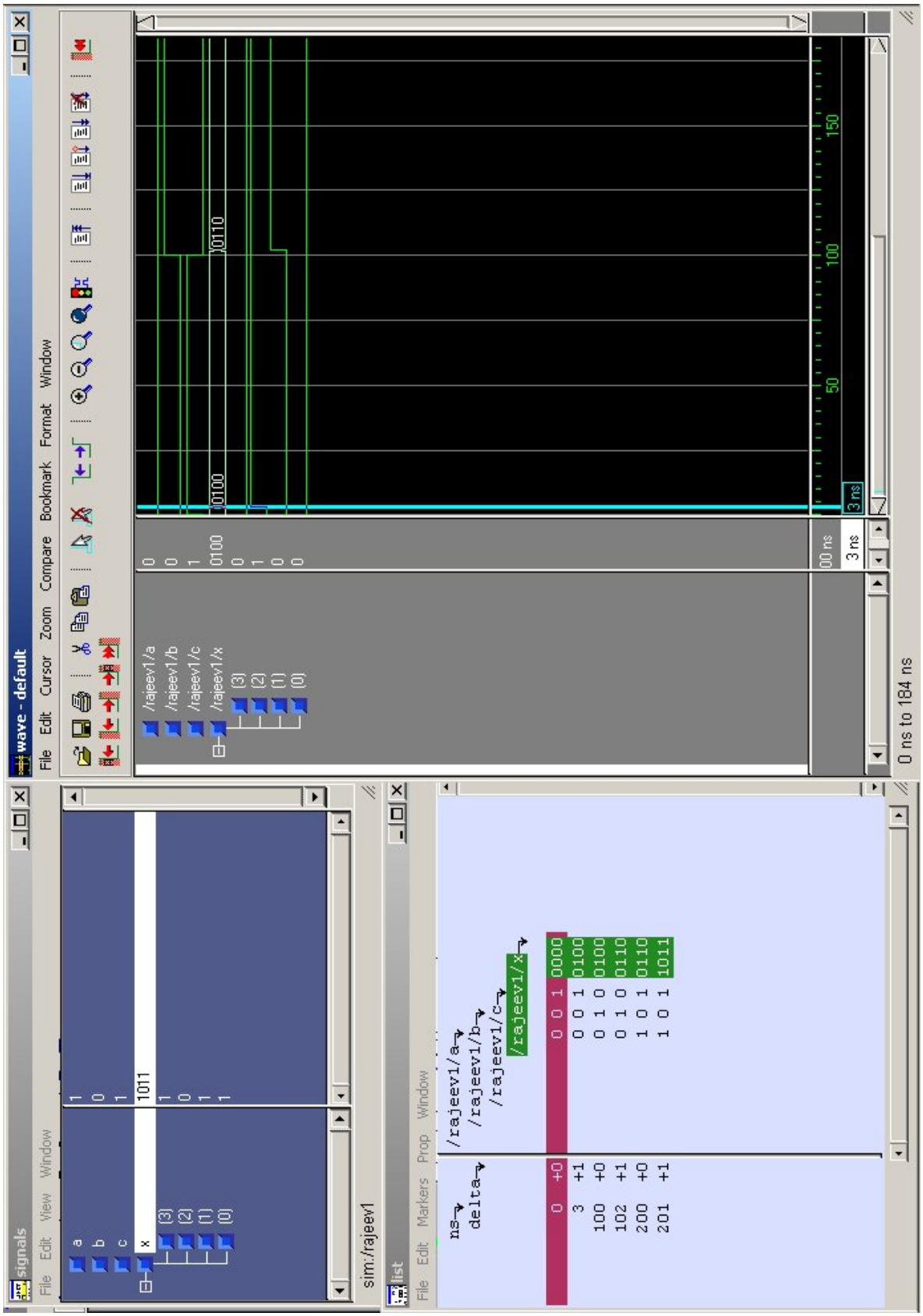
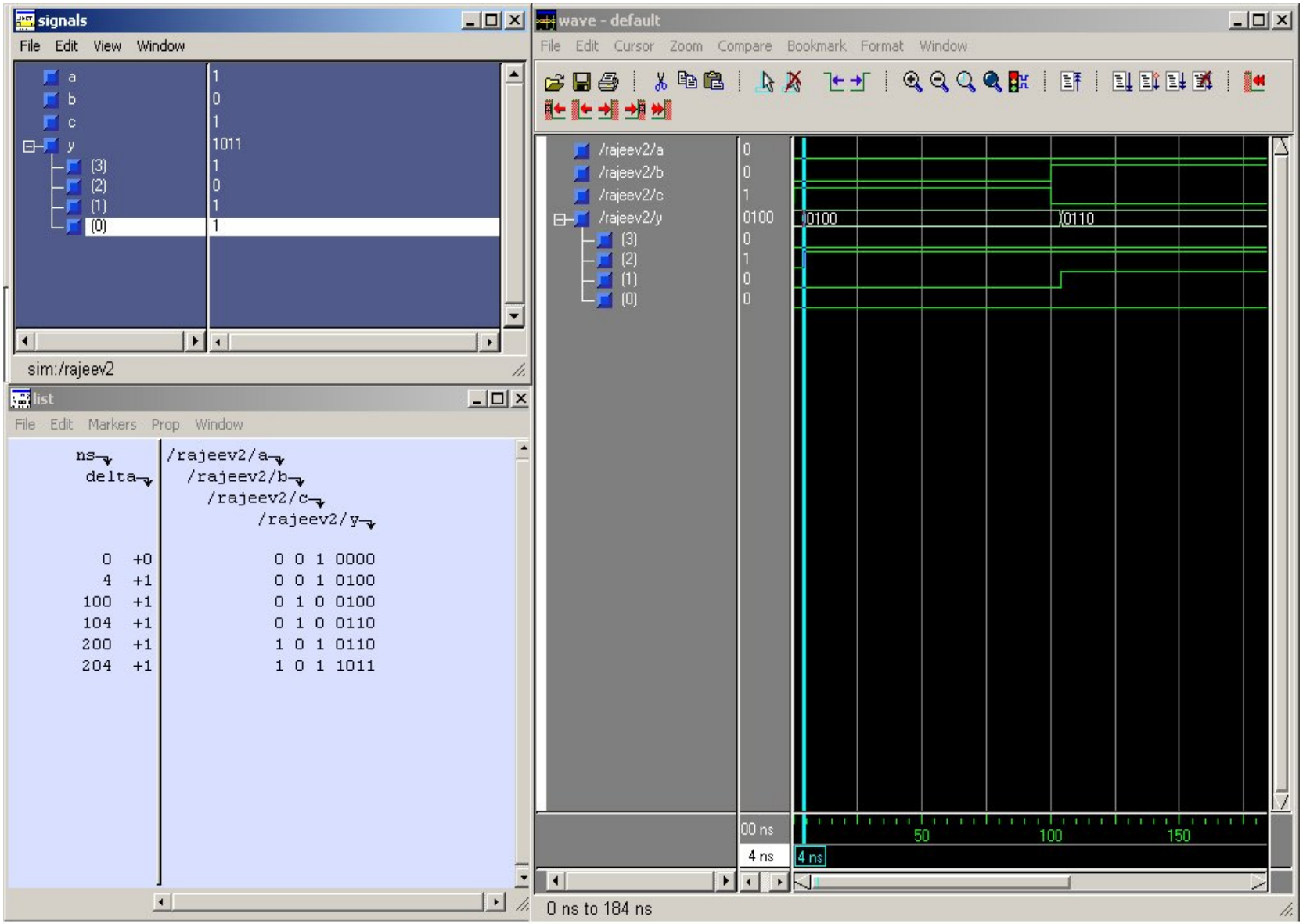


Figure 7.8: VHDL Simulation of Full Adder Circuit_1


signals
File Edit View Window
a 1
b 0
c 1
y 1011
(3) 1
(2) 0
(1) 1
(0) 1
sim:/rajeev2
list
File Edit Markers Prop Window
ns
delta
/rajeev2/a
/rajeev2/b
/rajeev2/c
/rajeev2/y
0 +0 0 0 1 0000
4 +1 0 0 1 0100
100 +1 0 1 0 0100
104 +1 0 1 0 0110
200 +1 1 0 1 0110
204 +1 1 0 1 1011
wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/rajeev2/a 0
/rajeev2/b 0
/rajeev2/c 1
/rajeev2/y 0100
(3) 0
(2) 1
(1) 0
(0) 0
0100
0110
00 ns
4 ns
4 ns
50
100
150
0 ns to 184 ns

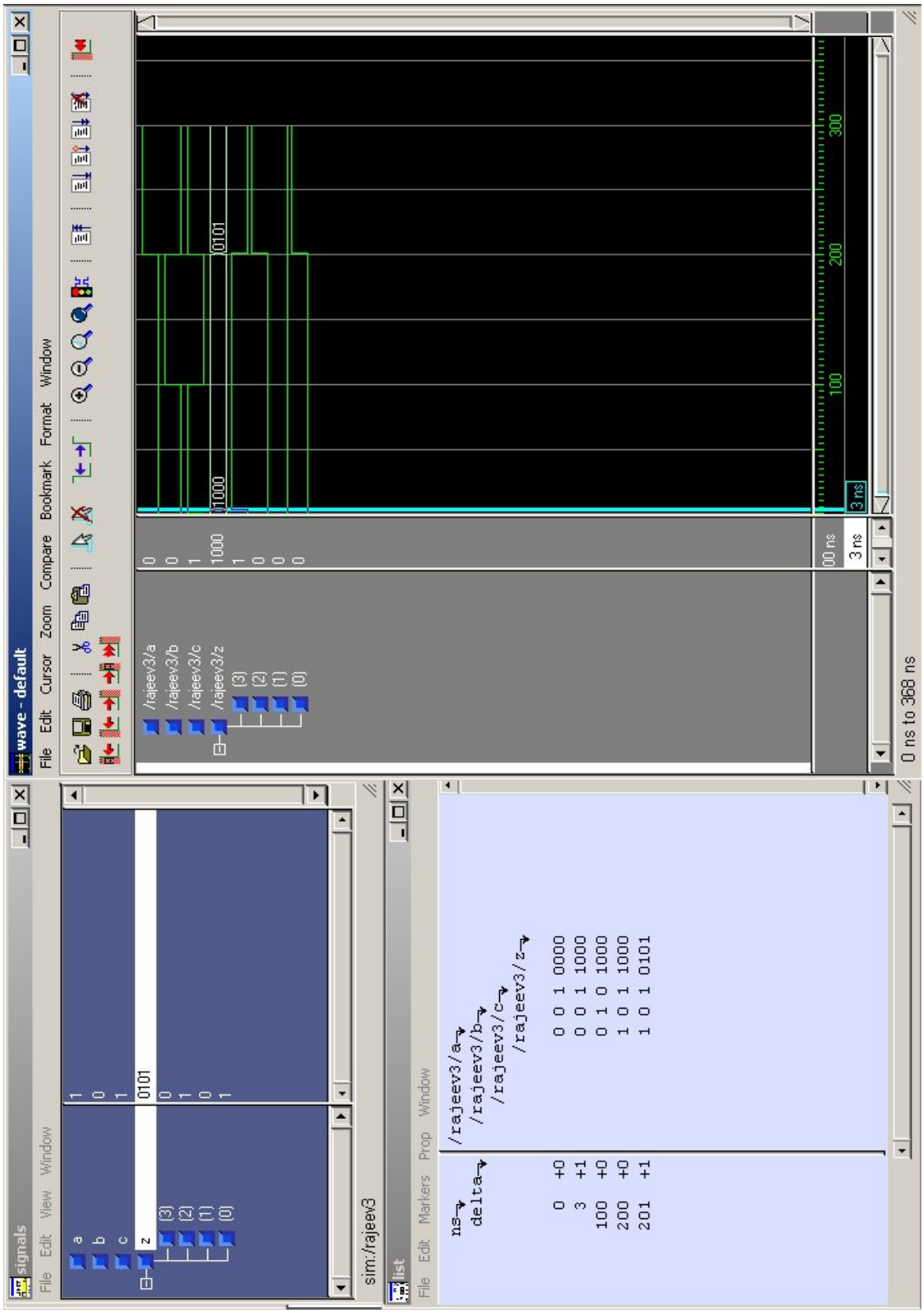


Figure 7.10: VHDL Simulation of Full Adder Circuit_3

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

There are three key ideas that were observed:

Firstly, reversibility stems from keeping track of every bit of information; irreversibility occurs only when the information is lost or erased.

Second, by doing computation reversibly, we obviate the need for energy expenditure during computation. All computations can be done, in principle, for zero cost in energy.

Third, reversible computation can be done efficiently, without the production of garbage bits or by minimising them whose value depends upon the input to the computation and type of reversible gates used and their interconnections.

So computation can be made dissipation free by making it reversible, although in practice energy dissipation is required for system stability and immunity from noise.

Moreover since the physical world is reversible, one can argue that complexity classes based upon reversible models of computation are more natural than complexity classes based upon irreversible models.

And from the view of Quantum computation and Quantum information, reversible computation is enormously important. To harness the full power of Quantum computation, any classical subroutines in Quantum computation must be performed reversibly and without the production of garbage bits depending on the classical input.

This completes the discussion of the construction of classical, reversible computers. To conclude, reversibility does not bar the logical design of computing machines. Therefore these ideas are mapped to Quantum systems.

8.2 Future Scope

Although the future of Quantum computing looks promising, we have only just taken our first steps to actually realizing a Quantum computer.

Quantum computers are presently being tested and worked. Researchers around the world are racing to achieve a practical system, a task which some scientist think is futile. Whether or not such systems will prove practical remains to be seen. David Deutsch-one of the ground breaking scientists in the world of Quantum computing-said himself that perhaps 'their most profound effect may prove to be theoretical'. However it was Einstein who said:

"If you can think about it, it will be possible now or in the future".

When Quantum computers will emerge, they will be considered as the superior computational device, and perhaps make today's computers obsolete. The obstacles are being solved that are getting us closer to the final product.

If we are able to design a Classical Reversible Computer, Quantum Computation and Quantum Algorithms can very well be simulated to produce error free results. Hence the main challenge would be to handle complex circuits on classical computer and designing better reversible circuits. Designing algorithms for that computer also seem to be an exciting field of reaserch.

References

[1] A.Barenco, C.H.Bennett, R.Cleve, D.P.DiVincenzo, N.Margolus, P.Shor, T.Sleator, J.Smolin and H.Weinfurter,"*Elementary gates for quantum computation*", Physical reviews. A, 1995.

[2] Andrew Chi-Chih Yao, "*Quantum Circuit Complexity*", Proceedings of 34th IEEE Annual symposium on Foundations of Computer Science,2004.

[3] A.Barenco,A.Ekert,A.Sanperaand,C.Machiavello,"*A short introduction to quantum computation*", by from *La Recherche",* November 1996.

[4] Andrei B. Khlopotine ,Marek Perkowski ,Pawel Kerntopf,"*Reversible Logic Synthesis by Iterative Compositions",* Portland Quantum Logic Group.

[5] Bernhard Omer*," A procedural formalism for quantum computing"*, Technical university of Vienna, Homepage: http://tph.tuwein.ac.at/~oemer.

[6] Barry A. Cipra *,"Quantum Computation: A New Spin On Complexity Theory"*, SIAM News, Volume 31, Number 8,Jan 2003.

[7] B. Kane, "*A Silicon-Based Nuclear Spin Quantum Computer*", Nature, vol. 393, 1998.

[8] C.H.Bennett,E.Bernstein,G.Brassard,U.V.Vazirani,"*Strengths and weaknesses of quantum computing*", SIAM Journal on Computing Volume 26, Issue 5,October 1997.

[9] C.H. Bennett, "*Logical reversibility of computation*", IBM Journal of Research and Development, pp. 525-532,17,1973.

[10] Charles H. Bennett,"*Notes on the History of Reversible Computation*", IBM Journal of Research and Development 32(1):16-23,Jan 1988.

[11] David Deutsch and Artur Ekert, "*Machines Logic and Quantum Physics*",Centre for Quantum Computation, Clarendon Laboratory, University of Oxford,Parks Road, Oxford, OX1 3PU, U.K., arXiv:math.HO/9911150 v1.

[12] D. P. DiVincenzo, ``*Quantum computation*,"IEEE Trans. Computers, vol. 25,No.6,1985.

[13] David Deutsch and Artur Ekert "*A short introduction to quantum information, including teleportation*", SIAM News, Volume 33, Number 7,Jan 2003.

[14] E.Fredkin and T.Toffoli,"*Conservative logic*",International Journal of Theoretical Physics,21,219-253,1982.

[15] J.W. Bruce, M.A. Thornton, L. Shivakumaraiah, P.S. Kokate, and X. Li,*"Efficient Adder Circuits Based on a Conservative Reversible Logic Gate"*,IEEE Computer Society Annual Symposium on VLSI, Pittsburgh, PA April 25-26, 2002.

[16] Gerhard W. Dueck and Dmitri Maslov,"*Reversible Function Synthesis with Minimum Garbage Outputs*",IEEE Computer Society,March 2004.

[17] Michael A. Nielsen and Issac L. Chuang ,"*Quantum Computation and Quantum Information*", Cambridge University Press,2001.

[18] Marek Perkowski and Pawel Kerntopf,*"Reversible and Quantums Logic Fundamentals from the Logic Synthesis Point View"* Portland Quantum Logic Group Tutorial 1.

[19] Mark Oskin, Frederic T. Chong, and Isaac Chuang,"*A Practical Architecture for Reliable Quantum Computers*", IEEE Comp. Society ,Jan.2002.

[20] Professor Samuel L. Braunstein,"*Notes*",Bangor University, UK, http://www.sees.bangor.ac.uk/~schmuel.

[21] P.W.Shor,"*Proceedings 35th Annual Symposium on the Foundations of Computer Science"*, IEEE Computer Society Press, Los Alamitos, California, 1994.

[22] P. Dirac,"*The principles of Quantum Mechanics*", Clarendon Press Oxford, United Kingdom,1987.

[23] Pawel Kerntopf ,"*Synthesis of Multipurpose Reversible Logic Gates*",Proceedings of the Euromicro Symposium on Digital System Design (DSD'02),0-7695-1790-/02 © IEEE, 2002.

[24] R.C. Minnick, P.T. Bailey, R.M. Sandfort, and W.L. Semon, *"Cascade realizations of magnet bubble logic using a small set of primitives"*,IEEE Trans. Computers, pp. 215-217, 1975.

[25] T. Sasao and K. Kinoshita, *"Conservative logic elements and their universality"*, IEEE Trans. Computers, Vol. 28, No. 9, pp. 682-685,1979.

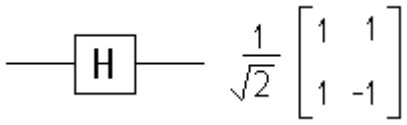
[26] Tycho Sleator and Harald Weinfurter,*"Realizable Universal Quantum Gates",*Physical Reviews letters Vol 74,No 20,1995.

[27] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes," *Reversible logic circuit synthesis*",In ICCAD, San Jose, California, USA, Nov,10-14 ,2002.

Appendix I

Frequently used Quantum gates and circuit symbols.Certain schematic symbols are often used to denote unitary transforms, which are useful in the design of quantum circuits. Many of these are shown below:

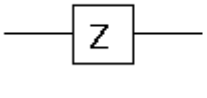
Hadamard



Pauli-X

Pauli-Y

Pauli-Z

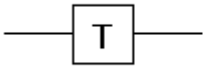


$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Phase

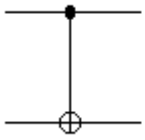
$$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

π/8



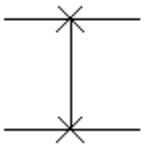
$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

Controlled-NOT



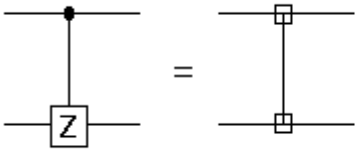
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Swap



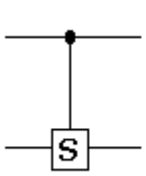
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Controlled-Z



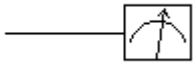
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Controlled-phase



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}$$

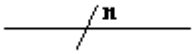
Measurement



Qubit

Classical bit

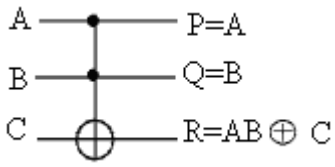
n qubits



Appendix II

Various Reversible Gates with their respective circuit representation

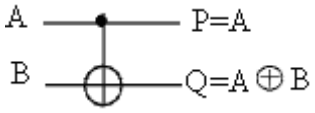
Toffoli gate (also called CNOT gate)



A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	1	0
0	1	1		0	1	1
1	0	0		1	0	0
1	0	1		1	0	1
1	1	0		1	1	1
1	1	1		1	1	0

P=A, Q=B, R=AB⊕ C

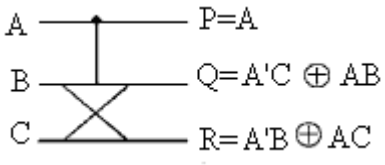
Feynman gate



A	B		P	Q
0	0		0	0
0	1		0	1
1	0		1	1
1	1		1	0

P=A, Q=A⊕ B
If B=0 then P=Q=A (in this case it works as a copying gate)

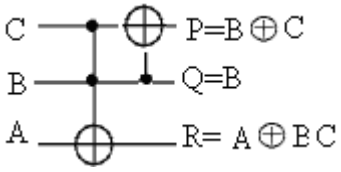
Fredkin gate



A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	1	0
0	1	0		0	0	1
0	1	1		0	1	1
1	0	0		1	0	0
1	0	1		1	0	1
1	1	0		1	1	0
1	1	1		1	1	1

P=A
Q=A’C+AB=C⊕AB⊕AC
R=A’B+AC=B⊕AB⊕AC

Peres gate



A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		1	0	0
0	1	0		1	1	0
0	1	1		0	1	1
1	0	0		0	0	1
1	0	1		1	0	1
1	1	0		1	1	1
1	1	1		0	1	0

P=B⊕ C
Q=B
R=A⊕BC

Paper Communicated

- Rajeev Tanwar,Amardeep Singh, ”*Comparative Findings on Reversible Against Classical Circuits*”, in Eighteenth Annual Conference on Neural Information Processing Systems ,NIPS’05 to be held in Vancouver, Canada on December 13-18th ,2005(communicated).