Reversible Computing Foundations in Quantum Computation: From Classical Logic to Quantum Algorithms

Keval Barvaliya

Kean University, Union, New Jersey 07083, USA
kevalbarvaliya007@gmail.com

Abstract. Reversible computing provides a foundational framework for understanding quantum computation. This paper first examines classical reversible logic gates, including the CNOT and Toffoli gates, and demonstrates how they can simulate arbitrary Boolean functions. We then extend these concepts to quantum gates and circuits, where all operations are inherently unitary and reversible. Quantum algorithms such as the Deutsch–Jozsa algorithm, phase estimation, and Shor's factoring algorithm are discussed to illustrate how reversibility enables exponential computational speedups. Finally, current challenges and future directions in quantum hardware and algorithm design are presented. Highlighting the essential role of reversible computing in the realisation of practical quantum computers.

Keywords: Quantum computation, Reversible computing, Quantum algorithms, Shor's algorithm, Quantum Fourier transform, Quantum gates

1 Introduction

Quantum computation exploits quantum-mechanical phenomena such as superposition and entanglement to perform computations that are infeasible for classical computers. A defining characteristic of quantum computation is the requirement that all operations be unitary and therefore reversible. This constraint directly connects quantum computing to the principles of classical reversible computation.

2 Qubits and Quantum Measurement

2.1 Mathematical Foundation of Qubits

A qubit is essentially the quantum analog of a classical computer's bit. Quantum entanglement and superposition enables quantum computation to be feasible. The qubit is a two-state quantum system [1], which is quite different from classical systems. Quantum information's basic element consists of a 2-level quantum

S. Bhalerao et al. (eds.), *Proceedings of the 2nd International Conference on Recent Advancement and Modernization in Sustainable Intelligent Technologies & Applications (RAMSITA-2026)*, Advances in Intelligent Systems Research 207,
https://doi.org/10.2991/978-94-6239-678-4_33

mechanical system. A qubit can be interpreted as a vector in the complex 2D Hilbert space of unit magnitude C^2. The basis states are

Inside the matrix $\mathbf{M}$, the vector denoted by $\mathbf{e}_0$ is one that has its first [2].

The state of a qubit can be described as a superposition. Satisfactory. The quantum computer's state is c0e0+c1e1, is the basis on which you make measurements.

2.2 Quantum Measurement and Collapse

A vector in Equation (1) defines the superposition of one qubit. Separation of linear combinations with complex coefficients. Let us also define the basis vectors $\mathbf{e}$.

When measured in the computational basis, any state will force the state of the qubit to be a basis state. In the same way, any measurement "against" any basis has to force the state of the qubit to be one of the vectors in that basis [3]. Imagine we have a Hermitian operator T whose eigenvectors are orthonormal. T being Hermitian, this can happen. result in a spectral undecomposable scheme of the type

$$T =$$

Again, measure S on the joint state

$$|\phi\rangle = \sum_i c_i |i\rangle |a_i\rangle.$$

Recall that the same principle applies to any Hermitian operator.

2.3 Multiple Qubits and Entanglement

For multiple-qubit systems, the tensor product space is present. Systems of n qubits then use

$$\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2,$$

repeated n times. Mostly, entanglement is an essential feature of such systems.

3 Reversible Classical Computing

3.1 The Need for Reversibility in Quantum Systems

The reversal of unitary processes is possible. Quantum computation was born out of physical limitations. The internal physics of the quantum system limits this fact.

We employ bits in classical computing. The operations AND, OR, and Unit are irreversible operations [4]. The NOT operation is reversible in general terms. Use different output state for every input. An undoable execution is called reversible operation. 1

The reversibility condition seems to be unsuitable for arbitrary computations since almost all (except NOT) of the essential classical logic gates (NOT, AND, OR, COPY) are non-reversible. Thus, reversible classical computing cannot function before quantum computing [5].

Table 1: Comparison of Classical Bits and Quantum Qubits

Property	**Classical Bit**	**Quantum Qubit**
State Space	$\{0, 1\}$	$\mathbb{C}^2$, $\|c_0\|^2 + \|c_1\|^2 = 1$
State Representation	Discrete values	Continuous superposition
Measurement	Deterministic	Probabilistic collapse
Multiple Systems	Cartesian product	Tensor product
Information	Copyable (trivial)	No-cloning theorem applies
Operation	Irreversible gates allowed	Reversible (unitary) gates only

3.2 Reversible Gates and Boolean Function Simulation

The output has the same length as a gate's input. Only the output can be observed, but the input of the gate can be determined uniquely. In that case, the gate is reversible. The gates are said to simulate a Boolean function $f(x_1 \dots x_n)$ if they can build a circuit taking $x_1..x_n$ along with some ancilla bits and give $f(x_1..x_n)$ output along with some garbage bits [6].

The CNOT (controlled-NOT) gate is a fundamental reversible gate with the operation:

$$\text{CNOT}(x, y) = (x, x \oplus y), \tag{1}$$

where $\oplus$ denotes addition modulo 2.

In this case, the first bit acts as a controller that decides the status of the second bit, where a '1' means that the second bit is flipped, else nothing occurs to the second bit [7]. CNOT gate can behave like NOT gate by making the control bit equal to 1.

The Toffoli gate (CCNOT) extends this concept with two control bits:

$$\text{Toffoli}(x, y, z) = (x, y, z \oplus xy). \tag{2}$$

The Toffoli gate leaves the third bit unchanged whenever the first two bits are set to 1. Toffoli gate is universal which means we are able to build any Boolean function using only Toffoli gates [8].

3.3 Implementing Classical Functions with Reversible Gates

To implement the AND function using Toffoli gates, we use the configuration:

$$(x, y, 0) \rightarrow (x, y, xy) \tag{3}$$

where the third input is initialised to 0. For the NOT function, we can use:

$$\text{CNOT}(x, y) = (x, x \oplus y) \tag{4}$$

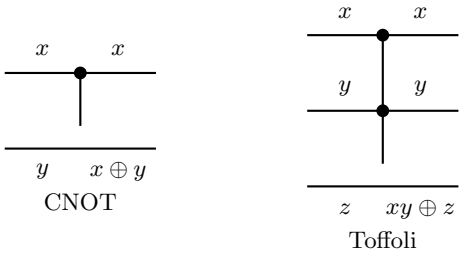

Fig. 1: Reversible logic gates: CNOT (left) and Toffoli (right)

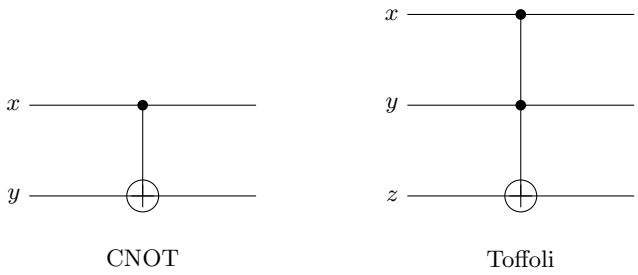

Fig. 2: Reversible logic gates: CNOT (left) and Toffoli (right)

As shown in Equation (4), the CNOT gate completes a reversible transformation.

Since every Boolean circuit can use NOT AND to simulate any combination of Boolean functions, then given enough permission to add some ancilla or garbage bits, almost every Boolean function can be implemented with reversible gates [9].

We can transform any classical computation into a reversible process by adding a few bits, as the theorem states. This indicates that we can also perform any quantum computation reversibly [10]. Nonetheless, it doesn't mention the quantity of additional bits we may have to add.

4 Quantum Gates and Circuits

Changing Classical Logic Gates into Quantum Logic Gates.

Quantum logic gates are unitary matrices that operate on qubit states. A classical reversible gate simulates quantum gates [11]. Furthermore, a subspace entanglement could also be generated among the states of the system which are interacting with them. The behaviour of quantum gates is interesting. Quantum gates could enable classical gates to effectuate [12].

The quantum CNOT gate performs on basis elements as anticipated.

Figure 2 illustrates the basic reversible logic gates.

But when applied to superposition states, it can create entanglement. For example:

$$U_{CNOT}\left(\frac{1}{\sqrt{2}}(\mathbf{e}_0 - \mathbf{e}_1) \otimes \mathbf{e}_1\right) = \frac{1}{\sqrt{2}} U_{CNOT}(\mathbf{e}_{01} - \mathbf{e}_{11}) \tag{5}$$

$$= \frac{1}{\sqrt{2}}(\mathbf{e}_{01} - \mathbf{e}_{10}) \tag{6}$$

which is an entangled singlet state.

4.1 The Hadamard Gate and Superposition

The Hadamard gate is one of the most fundamental quantum gates, defined by the matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{7}$$

Its action on basis states creates uniform superpositions:

$$H\mathbf{e}_0 = \frac{1}{\sqrt{2}}(\mathbf{e}_0 + \mathbf{e}_1), \quad H\mathbf{e}_1 = \frac{1}{\sqrt{2}}(\mathbf{e}_0 - \mathbf{e}_1) \tag{8}$$

When applied to multiple qubits, the Hadamard transform generates superpositions of all possible computational basis states. For n qubits initialized to $\mathbf{e}_{0\ldots0}$:

$$H^{\otimes n}\mathbf{e}_{0\ldots0} = \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} \mathbf{e}_y \tag{9}$$

This ability to create uniform superpositions forms the basis for quantum parallelism in many algorithms.

4.2 Quantum Circuit Design Principles

A series of quantum gates applied to qubits make up quantum circuits. Quantum circuits, unlike classical circuits, must be reversible throughout the computation. We use the standard method for non-invertible functions $f(x)$:

$$U_f(\mathbf{x} \otimes \mathbf{y}) = \mathbf{x} \otimes (\mathbf{y} \oplus f(\mathbf{x})) \tag{10}$$

In this case, $\mathbf{y}$ consists of ancilla bits initialised to the appropriate values, and the output includes the result and garbage bits. 2

The complete computation is reversible, while only the required function or result is computed. This means that using Bennett's technique, one can uncompute garbage bits to clean the workspace.

Table 2: Key Quantum Gates and Their Properties

Gate	**Matrix Representation**	**Input Qubits**	**Primary Function**
Pauli-X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	1	Bit flip (quantum NOT)
Hadamard	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	1	Creates superposition
CNOT	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	2	Controlled negation
Toffoli	8×8 permutation matrix	3	Doubly-controlled NOT
Phase	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	1	Phase shift

5 Deutsch-Jozsa Algorithm

5.1 Problem Formulation and Classical Complexity

Recently the experts did a lot of work, while there was a lot of interest among researchers in skills. Either the function is constant or it is a balanced function [13]. A balanced boolean function has equal number of zeros and ones. Constant refers to all 0 or all 1.

Figuring out whether a black box is always one value or equally likely to be both. The issue will show a constant function (which is also not observable, it is not learnable with XFXLE). One can't necessarily find the function with the XFXLE calls [14].

Traditionally, at least $2^{n-1} + 1$ queries are needed.

Implementation of Quantum Circuit

Using the circuit in Figure 3, A quantum resolution offers solution using only a single evaluation. This is how the algorithm functions.

The last Hadamard transformation on the first n qubits gives.

$$\frac{1}{2^n} \sum_{z,y \in \{0,1\}^n} (-1)^{y \cdot z + f(y)} |z\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{11}$$

where $y \cdot z = y_1 z_1 + \cdots + y_n z_n \pmod 2$.

5.2 Analysis and Quantum Advantage

The probability of measuring $\mathbf{e}_{0\ldots0}$ is determined by the squared magnitude of its coefficient:

$$\left| \frac{1}{2^n} \sum_{y \in \{0,1\}^n} (-1)^{f(\mathbf{e}_y)} \right|^2 \tag{12}$$

If f is constant, this sum equals $pm2^n$, giving probability 1. If f is balanced, This means that when we take one classification of the positive and the negative results, we have complete cancellation of the two terms to yield probability 0. Thus we can get a deterministic scheme with just a single evaluation of the function to distinguish constant from balanced.

Even though the algorithm lacks practical significance, but it possesses an excellent theoretical value. The very first technique to exhibit an exponential separation between classical deterministic computation and quantum computation Presenting the technique of usage of Hadamard transform before oracle call and phase kickback. This method has become a standard technique for quantum algorithm design.

$$\begin{array}{cccc} \mathbf{e}_0 & \mathrm{H} & [\mathrm{wires}{=}2]\mathrm{U}_f & H \\ & \mathbf{e}_0 & \mathrm{H} & \mathrm{H} \\ & & \vdots \; \vdots & \vdots \; \vdots \\ & \mathbf{e}_0 & \mathrm{H} & \mathrm{H} \\ & \mathbf{e}_1 & \mathrm{H} & \end{array}$$

Fig. 3: Quantum circuit for the Deutsch-Jozsa algorithm

6 Phase Estimation and Quantum Fourier Transform

6.1 Phase Estimation Problem

In 1995, Kitaev showed that the quantum algorithm's phase estimation is an important subroutine. Consequently, phase estimation has a variety of uses in most quantum algorithms. If a unitary operator, U, acts on an eigenvector, v, then the action of such a unitary operator is given by U v. Thus, the circuit could be or would be much simpler. If n equals 0 or 1, it indicates that it would be an eigenvector of the circuit in any case. This problem is classical to find the function's period. As a result, it is likely the easiest such problem. It naturally arises to estimate the eigenvalues of some unitary operator in many quantum algorithms. QFT is employed when a problem-solving solution is defined with a fixed time period.

6.2 Quantum Fourier Transform

An n-qubit quantum Fourier transform is a linear operator defined by its action on basis states.

$$\mathrm{QFT}|j\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n} |k\rangle. \tag{13}$$

The inverse QFT is defined in the same way but with gate phases opposite that used in the forward QFT.

The integrated circuit above implements the algorithm of phase estimation for quantum gates.

The complete circuit for estimating an m-bit phase is given below.

The initial register is $\mathbf{e}_{0\ldots0}$. 2. Apply Hadamards to the m qubits of the first register. 3. The first register is operated on with Λ while the second register contains $\mathbf{v}$. 4. Perform Inverse QFT to the First Register. 5. Measure the first register.

6.3 Accuracy and Performance

If $\theta = j/2^m$ holds for $j=0,\ldots, 2^m - 1$, then one will measure exactly j with probability 1.

It is possible to exponentially decrease the probability of error with $m + 2$ bits of accuracy and sufficient repetitions. For almost all values of , the probability for obtaining the best m-bit approximation is at least 4/2. The efficient implementations of the QFT and the controlled Uoperations is essential for a phase estimation algorithm. Also, one can apply.

7 Shor's Factoring Algorithm

7.1 Algorithm Overview and Significance

Integer factorisation is one of the most famous quantum computing results. That is Shor's algorithm. In particular, the algorithms factor a given integer N efficiently, requiring polynomial time as a function of log N for this operation. This implies that a significant speedup over classical algorithms is suggested.

The algorithm operates in this manner. 1. Use classical methods for an even or prime power N. 2. Choose a random value for $x < N$. 3. Use classical methods to find the GCD of x and N. If the output is either 1 or N, then don the output as a factor. 4. Apply a quantum order-finding algorithm to find the order r of x modulo N. 5. Per R. In the first place, the order-finding steps make use of the quantum Fourier transform. Determining decimal places in cuts. Shor's algorithm solves order finding, and reduction to factoring follows due to the mod group. Whenever x and N, the least positive integer r is called the order of x modulo N. The algorithm creates two multiples referred to as xr/2 – 1, N and xr/2 + 1, N. Unfavourable advice. N has 2 prime factors with high probability

of success. To observe this, note that the chances of satisfying the above random conditions are at least .378 when N has 2 distinct prime factors. Hence, the odds of success are rather high and rare.

7.2 Implementation Challenges and Current Status

According to Peter Shor, the 1994 [10]. Ever since its inception, the algorithm has been considered one of the most efficient algorithms. He found that a quantum computer could factor some integers in polynomial time, which is a remarkable discovery.

Most of the studies were on increasing the quality of the qubit and developments tailored to circuit implementations for error correction codes. Asking for the development of quantum engineering, the serious 2048-bit RSA project likely requires years before we produce sufficiently large fault-tolerant quantum computers able to factor 2048-bit RSA.

8 Conclusion

This article outlines how conventional reversible logic gates have now been quantum algorithms and all the way further in quantum computing. In this study, we demonstrate that the reversibility constraint, despite its superficial restrictive nature, will still allow for exponential speedups in the run-times of quantum algorithms – Deutsch–Jozsa versus Shor's factoring algorithm.

A notable perspective is that quantum computation is a broadening of reversible computing as it is done classically. To achieve a physical realization, it is necessary to satisfy reversibility and unitarity. However, the presence of hierarchical nature in reversibility and unitarity simultaneously conveys the resolution of quantum advantage.

Quantum hardware fidelity research, new algorithm research, cryptography research, and the quantum computing impact on future industries will one day be researched. According to the expert, as quantum technology progresses, reversible computing will be crucial for making realistic quantum computers.

References

1. M. A. Nielsen and I. L. Chuang, "Quantum Computation and Quantum Information: 10th Anniversary Edition," Cambridge University Press, 2010.
2. P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," SIAM Review, vol. 41, no. 2, pp. 303–332, 1999.
3. D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer," Proceedings of the Royal Society A, vol. 400, no. 1818, pp. 97–117, 1985.
4. C. H. Bennett, "Logical Reversibility of Computation," IBM Journal of Research and Development, vol. 17, no. 6, pp. 525–532, 1973.

5. T. Toffoli, "Reversible Computing," in International Colloquium on Automata, Languages, and Programming, pp. 632–644, Springer, 1980.
6. D. Deutsch and R. Jozsa, "Rapid Solution of Problems by Quantum Computation," Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, vol. 439, no. 1907, pp. 553–558, 1992.
7. R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum Algorithms Revisited," Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 454, no. 1969, pp. 339–354, 1998.
8. A. Y. Kitaev, "Quantum Measurements and the Abelian Stabilizer Problem," arXiv preprint quant-ph/9511026, 1995.
9. D. Coppersmith, "An Approximate Fourier Transform Useful in Quantum Factoring," IBM Research Report RC19642, 1994.
10. P. W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134, IEEE, 1994.
11. P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," SIAM Journal on Computing, vol. 26, no. 5, pp. 1484–1509, 1997.
12. S. J. van Enk, N. Lutkenhaus, and H. J. Kimble, "Experimental Procedures for Entanglement Verification," Physical Review A, vol. 75, no. 5, p. 052318, 2007.
13. A. Martin, B. George, and J. Smith, "Factoring 21 with Shor's Algorithm," Quantum Information & Computation, vol. 12, no. 1-2, pp. 61–70, 2012.
14. A. Einstein, B. Podolsky, and N. Rosen, "Can Quantum-Mechanical Description of Physical Reality be Considered Complete?" Physical Review, vol. 47, no. 10, p. 777, 1935.