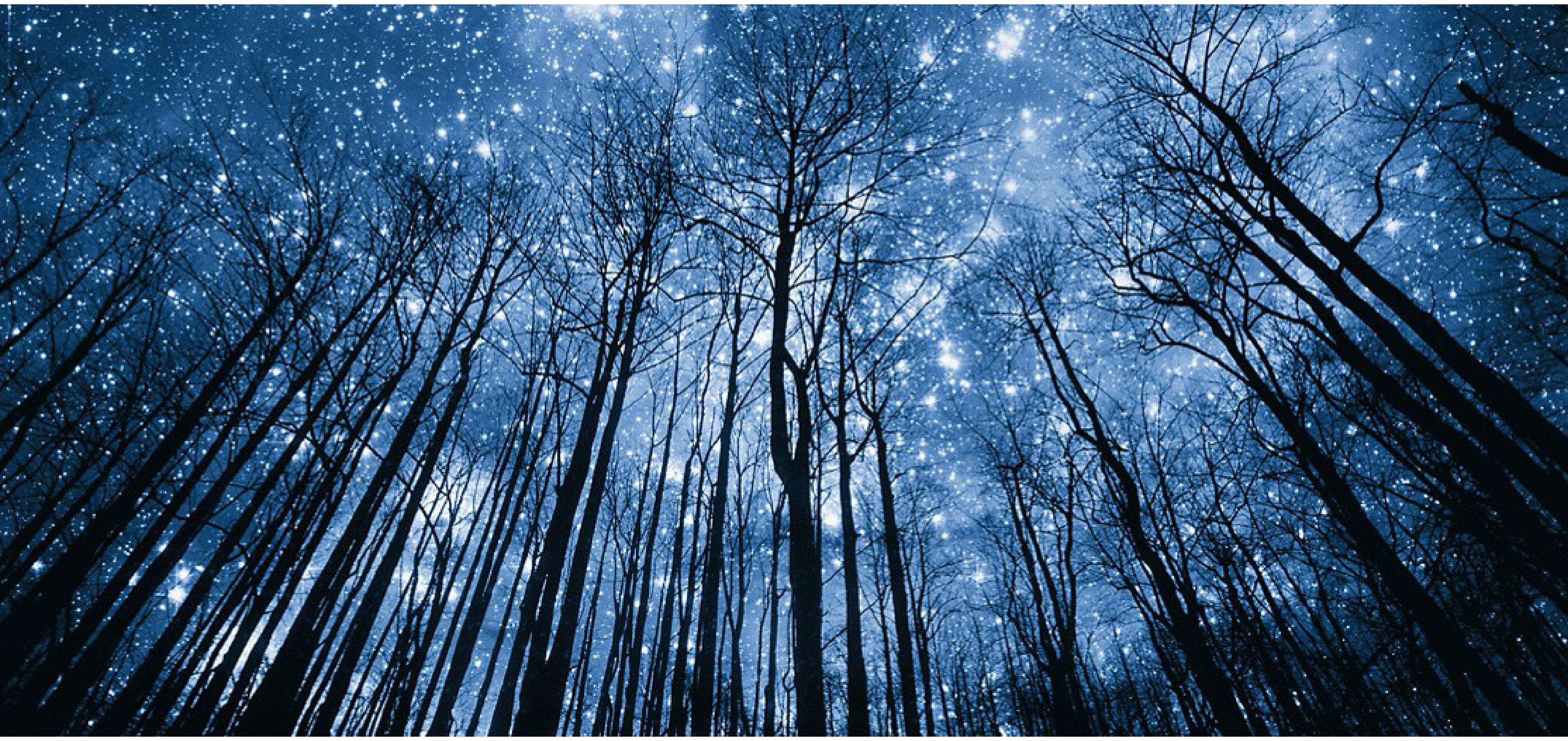
SWINGING BETWEEN CLASSICAL AND QUANTUM BITS



Rajasekhar Nannapaneni
Senior Principal Site Reliability Engineer
Dell Technologies
Rajasekhar.nannapaneni@dell.com

The Dell Technologies Proven Professional Certification program validates a wide range of skills and competencies across multiple technologies and products.

From Associate, entry-level courses to Expert-level, experience-based exams, all professionals in or looking to begin a career in IT benefit from industry-leading training and certification paths from one of the world's most trusted technology partners.

Proven Professional certifications include:

- Cloud
- Converged/Hyperconverged Infrastructure
- Data Protection
- Data Science
- Networking
- Security
- Servers
- Storage
- Enterprise Architect

Courses are offered to meet different learning styles and schedules, including self-paced On Demand, remote-based Virtual Instructor-Led and in-person Classrooms.

Whether you are an experienced IT professional or just getting started, Dell Technologies Proven Professional certifications are designed to clearly signal proficiency to colleagues and employers.

Learn more at www.dell.com/certification

Table of Contents

Abstract

Computing is one of mankind's most important technological advancements. The evolution of computing in a way has driven human progress over the years. Information processing has always been the primary objective for this evolution. The pervasiveness of information further drives the need for this advancement.

A variety of requirements use cases exist for information processing, from large enterprises to personal computing and from centralized to distributed edge setups. Several architecture models – from client-server to cloud computing – have evolved over the years for serving various purposes. While the architecture model's aim to focus on where or how information processing happens, it is equally important to understand what type of information processing is required for a specific use case.

As the nature of applications and their requirements change, one needs flexibility to switch between Classical and Quantum Computing. This article emphasizes the process of toggling between classical data and quantum data to solve complex and next generation problems such as Quantum Cryptography.

List of Figures

List of Abbreviations

Qubit	Quantum Bit
cNOT	Controlled NOT Gate
2-D	2 Dimensional
H gate	Hadamard Gate
X gate	Pauli X Gate
Z gate	Pauli Z Gate
P Time	Polynomial Time
NP Time	Non-Deterministic Polynomial Time
NP Complete	Non-Deterministic Polynomial Complete
EXP Time	Exponential Time
BQP Time	Bounded-Error Quantum Polynomial Time

Introduction

Computing has evolved exponentially since the mid-20th century led by computers that became faster, more powerful, and smaller.

This evolution is reaching its physical barrier. As the processor and other associated computer chips are approach the size of an atom restricts further advancements in processor development. A computer processor contains modules of logic gates which in turn contains transistors.

A transistor functions as a switch which can either open or close for information flow. Information consists of 0's and 1's (binary) bits and the action of the transistor results in the flow of these bits. Logic gates are formed with a combination of multiple transistors organized in a logical way to result in a specific outcome (like an adder). An arrangement of various logic gates can be combined to form modules in a computer which can perform complex math.

As transistor size shrinks to the scale of a few atoms, physical limitations governed by quantum physics result in a process called Quantum Tunneling where the electrons/holes in a transistor transfer to the other side of the junction.

Classical physics stops making sense when the size of the particles and its associated properties are at the scale of planks constant (10^{-34}). This makes things tricky for information processing and thus becomes a barrier for technology progress. To address this problem, we are trying to use these limitations caused by quantum physics to their advantage by building quantum computers.

A quantum computer represents an evolution of computing which can harness the nature to do computation. It's not called an advancement due to its smaller size; rather, it is but due to its capability to solve interesting problems.

Using a quantum computer may not be efficient for use cases such as addition of numbers and weather forecasting. But for other, more complex use cases, a quantum computer could do things in seconds that, as far as we know, would take any existing computer longer than the age of the universe.

Even for the problems solved by using a quantum computer, we would still require classical computer to process the outputs as that will be the optimal way. Quantum Computing in its true sense does not replace classical computing; it complements it.

Before we explore the use cases that could be efficiently solved using quantum computers, let's consider some of the foundational concepts of quantum computing.

Qubits

In classical computers, information processing happens via binary classical bits which can be represented using various descriptions:

- Mathematical description - 0's and 1's
- Physical description – voltage and/or magnetic domain

In quantum computers, information processing happens using qubits. A qubit is the simplest quantum system. Qubits can be realized in different ways such as using atoms, photons, electrons, etc. and this formulates the physical description.

A Qubit has 2 special states |0> and |1> where "|>" is called the ket notation.

Quantum State

A quantum state of a qubit is a vector in 2-D complex vector space. A general quantum state can be formed from a linear combination of |0> and |1>.

Superposition – Superposition mathematically can be thought of as another word for linear combination.

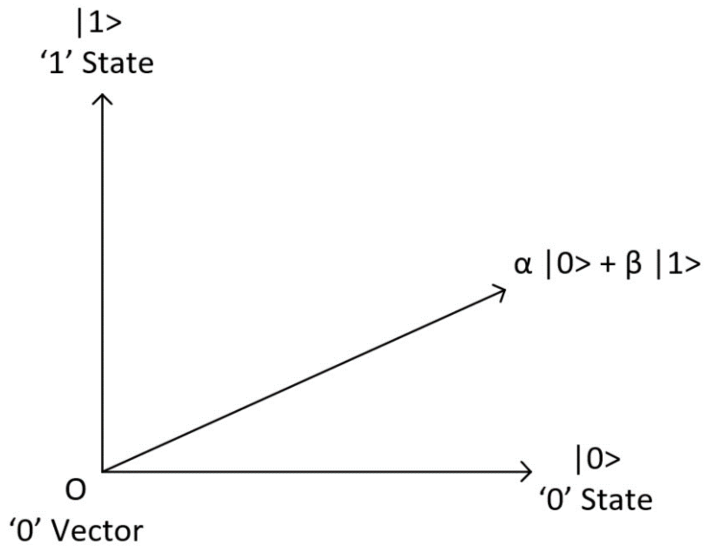


Figure 1: 2-D Complex Vector Space

In Figure 1, quantum state α |0> + β |1> is a linear combination, in other words a state in superposition and the amplitudes α and β are the coefficients, in other words the probabilities of states |0> and |1>, respectively. A quantum state is a vector with certain constraints that the vector should be of unit length.

$$\alpha^2 + \beta^2 = 1\ (Normalization\ requirement\ as\ sum\ of\ all\ probabilities\ should\ be\ equal\ to\ 1)$$

So, a quantum state of a qubit is a vector of unit length in a 2-D complex vector space as shown in Figure 1. The quantum state can be represented using | ψ> = α |0> + β |1> which can also be represented in matrix form $\begin{pmatrix}\alpha \\ \beta\end{pmatrix}$ Where,

$$|0> = \begin{bmatrix}1 \\ 0\end{bmatrix}$$
$$|1> = \begin{bmatrix}0 \\ 1\end{bmatrix}$$

These quantum states |0> and |1> are called computational basis states.

Quantum Logic Gates

Quantum logic gate is a way of manipulating quantum information or manipulating the quantum state of a qubit or a set of qubits. They are the basic building blocks of quantum computation.

Quantum Circuit

A quantum circuit is a representation of the quantum gates and their combinations in a sequential arrangement like classical circuits for modelling quantum computation.

Quantum Wire is a fundamental quantum circuit which is analogous to the classical circuit wires which seems obvious for information flow. However, unlike classical circuit wires, quantum circuit wire is very challenging to implement practically as the quantum state of a qubit that traverses through this wire is very fragile and is easily disturbed. Figure 2 gives a representational view of quantum wire which allows transmission of quantum state of a qubit.

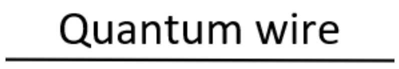


Figure 2: Quantum Wire

Quantum circuits for quantum gates will be represented in upcoming sections of this article where quantum wires are an inherent part of those circuits.

Quantum NOT Gate

Quantum NOT gate is a single qubit gate. A generalization of the classical NOT gate, The quantum NOT gate is a single qubit gate which has one input and one output. The quantum circuit representation of the quantum NOT gate is shown in Figure 3.

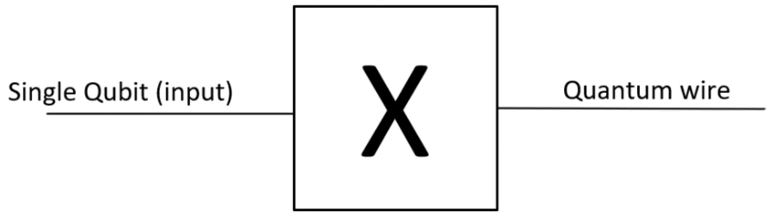


Figure 3: Quantum NOT gate

The mathematical representation of quantum NOT gate is in the matrix form shown below.

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

When a quantum NOT gate is applied to a quantum state |0>, the output would be switched to quantum state |1> like classical NOT gate.

$$X\,|0> = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1>$$

Similarly, when a quantum NOT gate is applied to a quantum state |1>, the output would be switched to quantum state |0> like classical NOT gate.

$$X\,|1> = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0>$$

It’s interesting to note that when a quantum NOT gate is applied on a superposition state, the gate is applied linearly on individual qubits and switches the state as expected like a classical NOT gate operation.

$$X\,\{\alpha\,|0> + \beta\,|1>\} = \alpha\,|1> + \beta\,|0>$$

It is important to verify that the quantum state traversing via NOT gate is remaining at unit length; in other words, the quantum NOT gate is a Unitary Matrix. This can be verified by applying quantum NOT gate twice sequentially on any given quantum state $|\psi>$.

$$|\psi> \text{--------->} X|\psi> \text{-------->} XX|\psi> \text{--------->} |\psi>$$

In matrix form, when quantum NOT gate is applied twice the resultant matrix is an identity matrix indicating no change in the length or magnitude of the quantum state.

$$XX = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \text{ (Identity Matrix)}$$

The same applies for any state in superposition that the resultant qubit is always of unit length and in quantum circuit representation. Figure 4 shows that applying quantum NOT gate twice is equivalent to not applying any gate on that qubit.

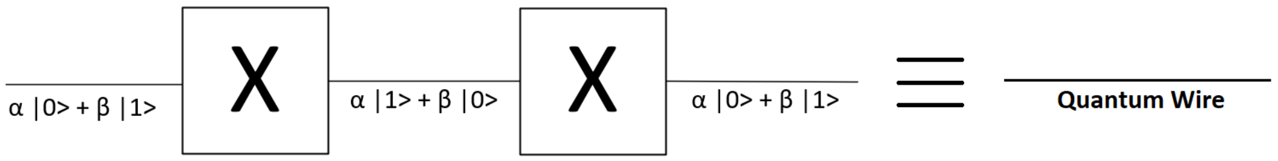


Figure 4: Quantum NOT applied twice on a qubit

Quantum Hadamard Gate

The quantum Hadamard gate is another single qubit gate which takes single qubit as input and results in a superposition quantum state. The quantum Hadamard gate is a single qubit gate which has one input and one output. The quantum circuit representation of the quantum Hadamard gate is shown in Figure 5.

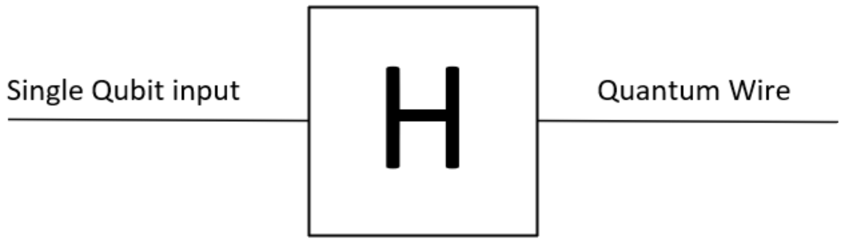


Figure 5: Quantum Hadamard Gate

The mathematical representation of quantum Hadamard gate is in the matrix form shown below.

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

When Hadamard gate is applied on the quantum state 0, the quantum state is switched to superposition quantum state as shown below.

$$H\,|0> = H\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{|0> + |1>}{\sqrt{2}}$$

When Hadamard gate is applied on the quantum state 0, the quantum state is switched to superposition quantum state as shown below:

$$H\,|1> = H\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} = \frac{|0> - |1>}{\sqrt{2}}$$

When Hadamard gate is applied to a superposition state, the output below will be obtained.

$$H\,\{\alpha\,|0> + \beta\,|1>\} \text{--->} \alpha\,(\frac{|0>+|1>}{\sqrt{2}}) + \beta\,(\frac{|0>-|1>}{\sqrt{2}}) = \frac{\alpha+\beta}{\sqrt{2}}\,|0> + \frac{\alpha-\beta}{\sqrt{2}}\,|1>$$

It is important to verify that the quantum state traversing via Hadamard gate is remaining at unit length; in other words the quantum Hadamard gate is a Unitary Matrix. This can be verified by applying quantum Hadamard gate twice sequentially on any given quantum state $|\psi>$.

$$|\psi> \text{--------->} H|\psi> \text{-------->} HH|\psi> \text{------->} |\psi>$$

In matrix form, when quantum Hadamard gate is applied twice the resultant matrix is an identity matrix indicating no change in the length or magnitude of the quantum state.

$$\text{HH} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \text{I (Identity Matrix)}$$

Quantum computation of applying Hadamard gate twice on a qubit at state 0 on quantum circuit level will be:

$$\text{H } |0> = \frac{|0>+|1>}{\sqrt{2}}$$

$$\text{H } \{\frac{|0>+|1>}{\sqrt{2}}\} = \frac{1}{\sqrt{2}}\{\frac{|0>+|1>}{\sqrt{2}} + \frac{|0>-|1>}{\sqrt{2}}\} = |0>$$

Quantum computation of applying Hadamard gate twice on a qubit at state 0 on quantum circuit level will be:

$$\text{H } |1> = \frac{|0>+|1>}{\sqrt{2}}$$

$$\text{H } \{\frac{|0>-|1>}{\sqrt{2}}\} = \frac{1}{\sqrt{2}}\{\frac{|0>+|1>}{\sqrt{2}} - \frac{|0>-|1>}{\sqrt{2}}\} = |1>$$

The same applies for any state in superposition that the resultant qubit is always of unit length and in quantum circuit representation. Figure 6 shows that applying quantum Hadamard gate twice is equivalent to not applying any gate on that qubit.

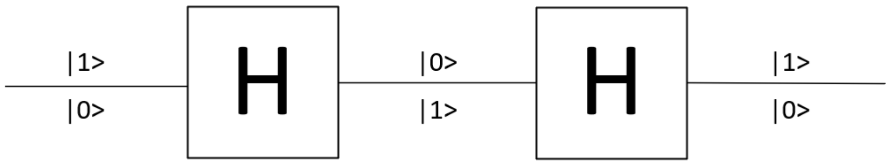


Figure 6: Quantum Hadamard gate applied twice on a qubit

Quantum gates – like Hadamard gates – expand the range of states it is possible for a computer to be in. Allowing such an expansion of the range of states creates the possibility to take shortcuts which may help perform computations faster.

More Single Qubit Quantum Gates

Another single qubit gate is shown below which has real parameters θ and φ. We can expect phase change in computational basis states |0> and |1>.

$$\begin{bmatrix}e^{i\theta} & 0\\0 & e^{i\phi}\end{bmatrix}$$

This matrix is also unitary which can be shown by self-multiplying the matrix by itself to result in identity matrix.

Applying this gate on quantum state |0> gives:

$$\begin{bmatrix}e^{i\theta} & 0\\0 & e^{i\phi}\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = e^{i\theta}\,|0>$$

Applying this gate on quantum state |1> gives:

$$\begin{bmatrix} e^{i\theta} & 0 \\ 0 & e^{i\phi} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = e^{i\theta}\ |1>$$

For a general quantum state (α |0> + β |1>), the output would be (α $e^{i\theta}$|0> + β $e^{i\phi}$|1>)

These phase factors (θ and φ) don't make any difference to the absolute value of α and β. So, the state (α |0> + β |1>) is indistinguishable from the state (α $e^{i\theta}$|0> + β $e^{i\phi}$|1>).

Since the input and output states are not distinguishable when seen from the absolute value point of view, application of this gate on a quantum state can been seen different when viewed from quantum circuit way. As an example, when the real parameters are set to θ = 0, φ = π and this gate is applied with an input $\frac{|0>+|1>}{\sqrt{2}}$ then,

$$\begin{bmatrix} e^{i\theta} & 0 \\ 0 & e^{i\phi} \end{bmatrix}\left(\frac{|0>+|1>}{\sqrt{2}}\right) = \frac{|0>-|1>}{\sqrt{2}} \text{ is the output}$$

So, the states $\frac{|0>+|1>}{\sqrt{2}}$ and $\frac{|0>-|1>}{\sqrt{2}}$ though having same absolute value, differ with each other when applied with Hadamard gate giving different outputs |0> and |1> respectively.

Hence, they are different and gates of the form $\begin{bmatrix} e^{i\theta} & 0 \\ 0 & e^{i\phi} \end{bmatrix}$ can have a practical impact on the outcomes of our computations.

There are several other single qubit gates such as:

- Rotation gate: $\begin{bmatrix} Cos\theta & -Sin\theta \\ Sin\theta & Cos\theta \end{bmatrix}$
- Pauli X gate: $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
- Pauli Y gate: $\begin{bmatrix} 0 & -i \\ \mathrm{i} & 0 \end{bmatrix}$
- Pauli Z gate: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

All these quantum gates are unitary which means if the matrix representation of each of these gates is multiplied by itself, it results in an Identity matrix.

Controlled-NOT Gate (Multi Qubit Quantum Gate)

Controlled-NOT gate is one of the 2 qubit quantum gates out there which has 2 qubits as inputs. One of the qubits is called the control qubit which decides how the outcome would vary while the other is called the target qubit as shown in Figure 7.

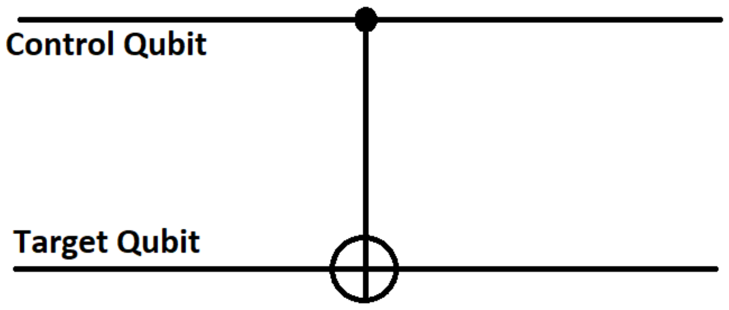


Figure 7: Controlled-Not gate

We now have 4 computational basis states, and they can form to be in superposition as follows:

$$\alpha\,|00> +\beta\,|01> +\gamma\,|10> +\delta\,|11>$$

For absolute value normalization of quantum state, $\alpha^2 + \beta^2 + \gamma^2 + \delta^2 = 1$ should be met.

In a quantum controlled-Not gate, the target qubit remains as is until the control qubit is in the quantum state of |0>. The moment the control bit is in the quantum state of |1>, the target qubit is flipped. If we represent this logic as a truth table:

|00>	------>	|00>
|01>	------>	|01>
|10>	------>	|11>
|11>	------>	|10>

This above logic can be summarized as:

|xy> ------> |x y⊕x>

The matrix representation of the controlled-Not gate is:

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

The 3-qubit controlled NOT gate shown in Figure 8 can be summarized as:

|xyz> ------> |x y z⊕y>

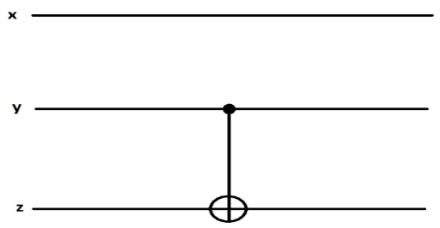


Figure 8: 3-qubit cNOT gate

Quantum Entanglement

Interesting things happen when quantum logic gates are arranged in certain ways. Consider a circuit as shown in Figure 9 where the input |00> is applied to Hadamard gate which then is applied to a cNOT gate.

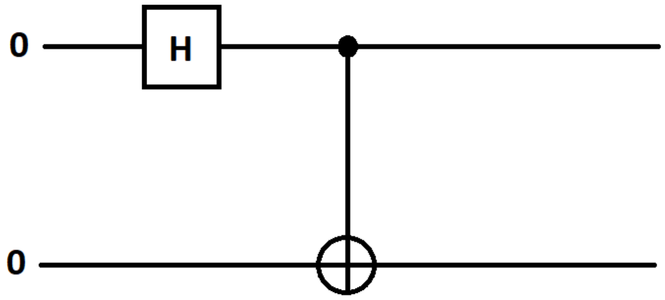


Figure 9: Hadamard gate applied to cNOT gate

When the Hadamard gate is applied to the input quantum state |00>:

$$|00> \text{ transforms into } \frac{|00> + |10>}{\sqrt{2}}$$

$$\text{Applying cNOT to } \frac{|00> + |10>}{\sqrt{2}} \text{ results in } \frac{|00> + |11>}{\sqrt{2}}$$

This output $\frac{|00> + |11>}{\sqrt{2}}$ is a non-classical state (also called an entangled state or Bell State). There is no simple interpretation of this state as a classical state of two qubits.

Thus, a single qubit gate and cNOT gate can result in an interesting output which has special use in information processing.

Measuring a Qubit

Qubits can be in any quantum state, be it quantum computational basis states |0> or |1> or in an intermediate superposition quantum state in a 2-D complex vector space. To know which quantum state (ψ) a given qubit (α |0> + β |1>) is in, we need to measure the qubit to observe its quantum state.

In quantum mechanics, the term quantum state (ψ) used thus far in this article based on mathematical description is considered as wave function ψ(x) which is related to probability distribution of observable quantities (x) like position/energy/momentum of a particle.

ψ(x) is a complex function and has value as a complex number which can never really be observable. As an example, a position of value (5+6i) meters is something non-existent in physical world.

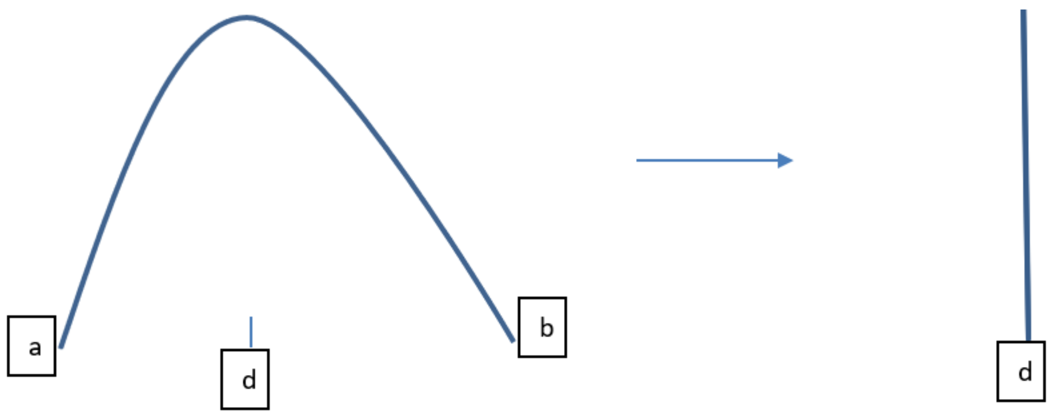


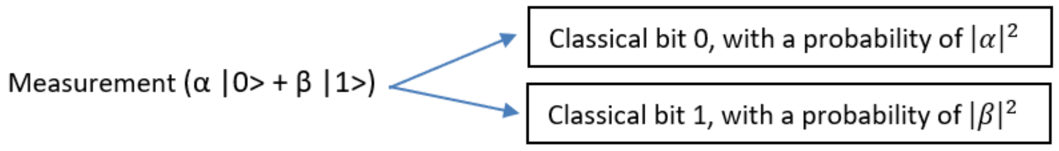
Figure 10: Wave function before and after measurement

Mathematically to determine the probability of finding the particle between positions ‘a’ and ‘b’ is given by integral of probability distribution of this state of the system.

$$P(particle\ between\ a\ \&\ b) = \int_b^a |\psi(x)|^2\, dx$$

The wave function collapses after the measurement to the position of the particle as shown in Figure 10. Similarly, the quantum state of a qubit in mathematical description is not directly observable and it collapses to quantum computational basis states |0> or |1> after measurement.

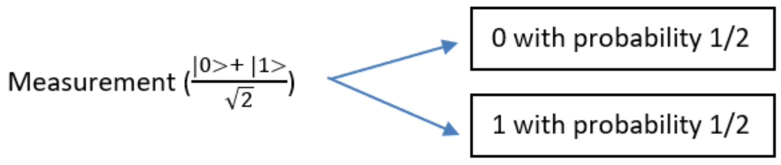
So, when a qubit (α |0> + β |1>) is given and we perform measurement on it in computational basis, it gives us a classical bit 0 or 1.


Measurement (α |0> + β |1>)
Classical bit 0, with a probability of $|\alpha|^2$
Classical bit 1, with a probability of $|\beta|^2$

The measurement in the computational basis for a given qubit disturbs the state of the quantum system. After the measurement, qubit is either in the computational basis state 0 or 1 and the coefficients α and β disappear.

α and β are complex numbers and are like hidden information; we can’t get full information about them and they are gone after the measurement. A qubit can’t store an infinite amount of classical information.

As an example:



Measurement follows normalization to maintain the rule that sum of all probabilities is equal to one. The circuit notation for measurement is shown in Figure 11.

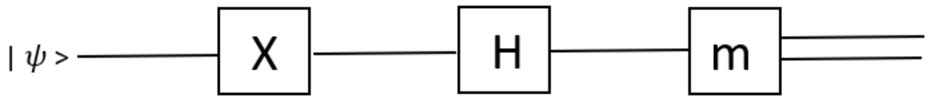


Figure 11: Circuit Notation for Measurement

Classical information is sent via quantum circuits for processing and when the measurement is performed on the processed quantum bits, the result is classical bits as shown in Figure 12.

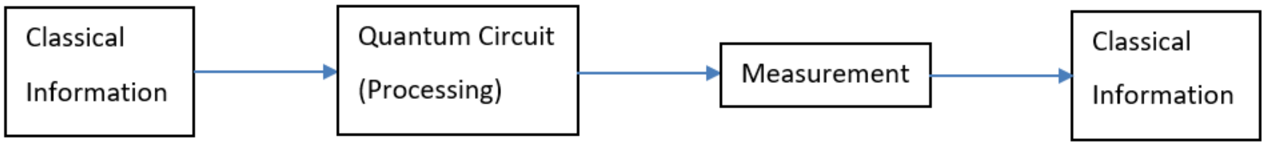


Figure 12: Classical to Quantum to Classical Information Flow

Universality in Computation

Universality with respect to computation is the ability of the compute system to compute any given function. That is, given a function F, the compute system should be able to compute F(x) for all values of x.

Classical Computation

Classical computers combine logic gates like AND & NOT gates together to compute any function. Hence, the classical NOT & AND gates are called universal gates. There are other logic gates like NAND gate in a classical computer which is also called universal gates. Thus, the concept of universality in classical computers is achieved using universal gates and/or a combination of those logic gates.

Quantum Computation

In the case of quantum computation, the analogous universal gates could be cNOT and single qubit gates which can be used to build up any unitary operation on 'n' qubits.

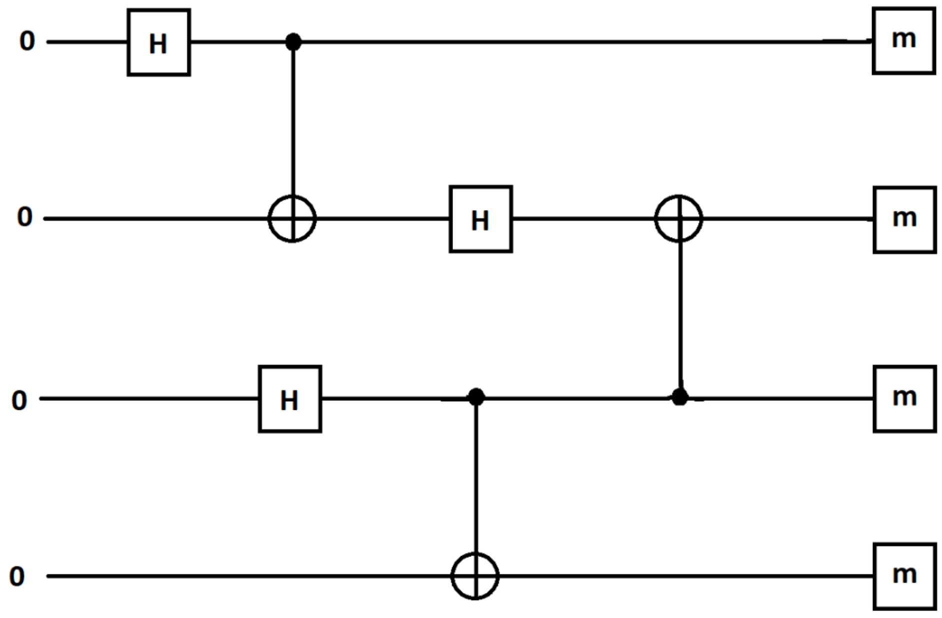


Figure 13: Quantum Universal Computation

A sample quantum circuit to demonstrate quantum computation is illustrated in Figure 13. The following is the high-level computational steps:

- Start in a computational basis state as 4 inputs |0> |0> |0> |0> ~= |0000>
- Apply a sequence of cNOT's and single Qubit gates
- Measure in the computational basis

- Pr(0….0) = |amplitude|^2
- This can be considered as formulating computation of any function with inputs (x)
  - |x,0> -----------> |x, F(x)>

For every x->F(x) in classical computing there is an analogous equivalent |x,0> ------->|x,F(x)>.

3.1 Problem Complexity in Classical Computing

Alan Turing has tabled an idea of a universal computing machine where the operation of the machine can be written in terms of a program. The purpose of this universal computing machine is to compute any computable sequence. To perform computations, we realized and developed systems that could have a lot of space and memory. Having space and memory turned out to be a smaller problem than the time required to perform the computation sequence for complex problems as shown in Figure 14.

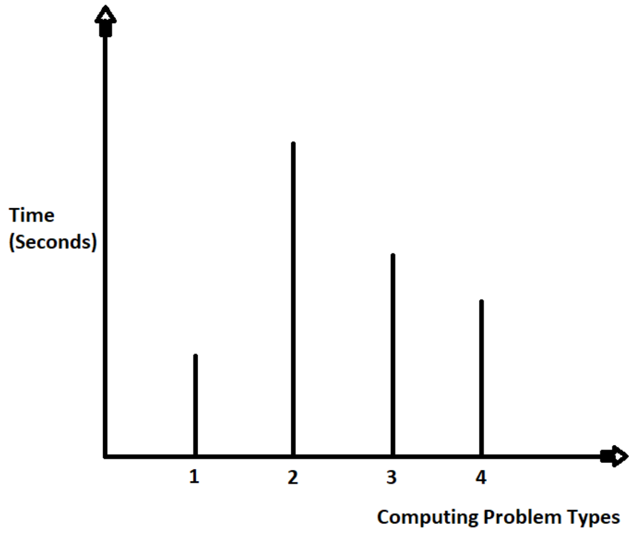


Figure 14: Computing Problem Types Vs Time

Dr John Nash during his work in break codes suggested that we should not look at the absolute time required for computing a problem (example: time required to perform subtraction of two 3-digit numbers) but instead look at how the computation grows relative to input size as shown in Figure 15. (example: How long does it take to subtract two 3-digit numbers Vs subtracting 4-digit numbers).

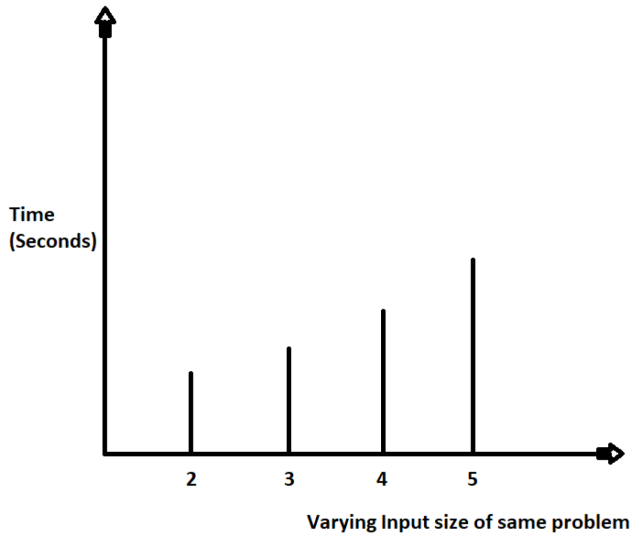


Figure 15: Input size vs Time of a computation problem

Later, it was thought that instead of measuring them with respect to time (Seconds) we should instead use the number of machine operations. Number of steps in other words can be considered as number of state transitions.

The growth of each problem can be drawn as a curve based on number of steps a problem would take on any machine as the input size grows. As shown in Figure 16, we can relate Problem 1 to Addition which grows more slowly than Problem 2 which can be thought as a Multiplication operation.

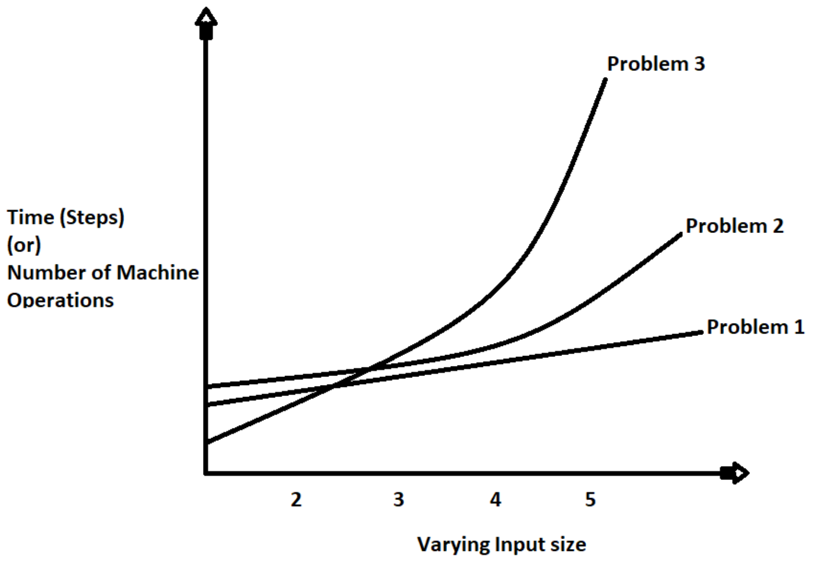


Figure 16: Input Size Vs Time Steps for computational problems

3.1.1 Linear Time vs Polynomial Time Problems

The shape of these growth curves became a meaningful way to classify a given problem. One important reason the growth increases more quickly for some problems verses others is due to the presence of loops in the algorithm run by the system. So, the number of steps in the loop depend on the size of the input. This is where the number of time steps linearly grow with input size and often called as a linear growth problem.

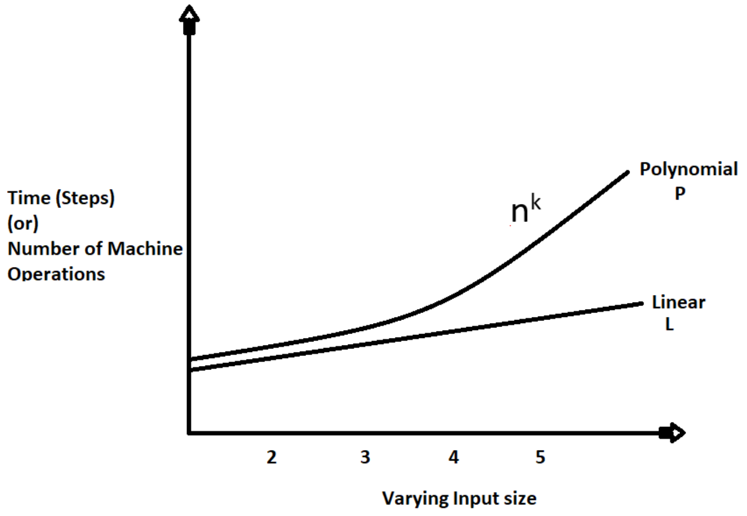


Figure 17: Linear Vs Polynomial Complexity

More complex problems would have loops inside another loop which are called nested loops. If the loop is nested twice, we get a growth curve of n^2 and if the loop is nested thrice, the growth curve resulted is n^3. These types of problems are called polynomial time because the growth of the problem can be described using n^k where n is the input size and k is the number of nested loops or depth of the computational problem. Figure 17 illustrates the growth curves of linear vs polynomial problems.

3.1.2 Polynomial Time vs Exponential Time Problems

It turns out most of the computational problems that need to be solved require polynomial time. Sorting of numbers is one such example that requires polynomial time. There is another type of computational problem which poses a problem, called exponential computational problems. These type problems grow with input size; that is, the number of nested loops require is of the order of input size. As an example, if 3-digit input size problems require 3 nested loops then incrementing the input size to 4 requires 4 nested loops. The growth curve of these problems grows more sharply than the polynomial type of computational problems. Thus, these types of problems can be represented by k^n where n is the input size and k are the number of nested loops.

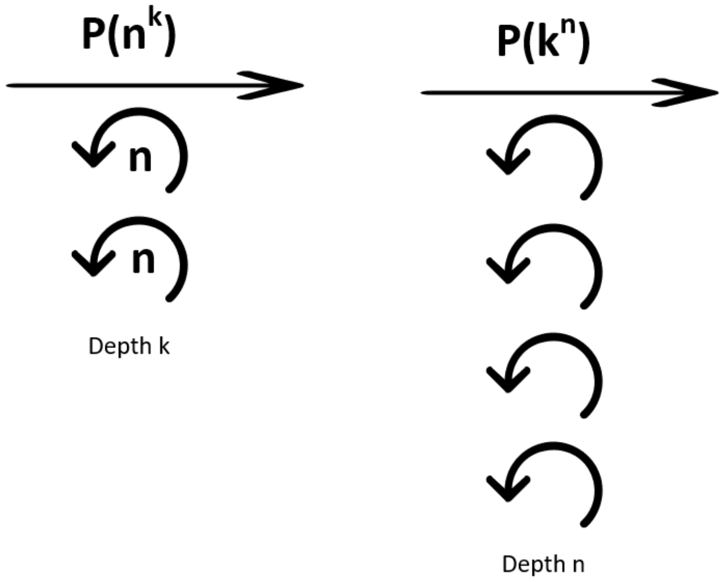


Figure 18: Nested Loops in Polynomial vs Exponential time problems

The key difference between polynomial time and exponential time computational problems is that in the polynomial time problems $P(n^k)$, the number of loops (k) in the algorithm doesn't change when the input size (n) grows as shown in Figure 18. Increasing the input size (n) simply adds to the number of steps within each loop but in an exponential time problem EXP (k^n), the number of nest loops (k) increases as the input size (n) grows.

The exponential computational problems quickly become intractable with classical computing due to increase in number of operations or number of nested loops as the input size increases. Hence, we can say that polynomial time problems are practical and exponential problems are not practical.

3.1.3 Polynomial Time vs Non-Deterministic Polynomial Time Problems

It is desirable to develop algorithms that could solve exponential time problems in polynomial time. If this is accomplished, one can say that the complexity of an exponential time category problem is reduced to a polynomial time category problem. Many exponential problems such as finding prime factorization of a number are irreducible to polynomial time.

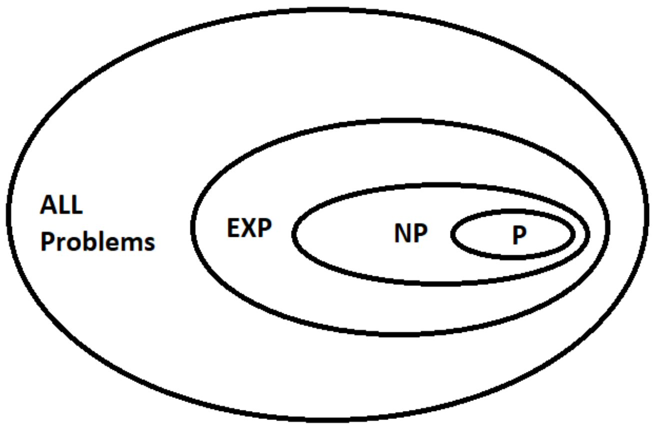


Figure 19: Categories of the Problems based on complexity

The next set of problems are difficult to solve but easy to verify. These sets of problems at first appear very difficult to solve but once the answer is known, the solution appears obvious. Many decision-based problems fall into this category and these problems are called Non-deterministic Polynomial time (NP) problems which are theoretically possible to solve in polynomial time. This is practically possible only if we have a non-deterministic computing which processes the algorithm of decision making in parallel.

NP is solvable in polynomial time using a Non-Deterministic machine as shown in Figure 19. Since Non-Deterministic machines are non-existent, NP problems remain as exponential time problems. So, P ≠ NP.

3.1.4 NP vs NP-Complete Problems

There are certain problems in the NP category which share structural similarity and can be reduced to each other. This set of interconnected problems are called NP-Complete problems. If any one of the NP Complete problems is solved in polynomial time, all the interconnected problems will get solved making P = NP. However, this wasn't the case so far.

3.2 Problem Complexity in Quantum Computing

Quantum complexity theory is a subfield of the world of computational complexity theory which deals with the categorization of algorithms, sorting them into bins according to how well they run on computers.

The categorization is based on how much harder it is to solve the problem as the problem gets larger. The P box shown in Figure 20 in green is easy to solve with a classical computer, but anything outside it means we don't have efficient classical algorithms to solve them and factoring large numbers is one of these.

But the BQP box shown in Figure 20 is efficient for a quantum computer, but not a classical computer. These are the problems better solved by quantum computers than classical computers.

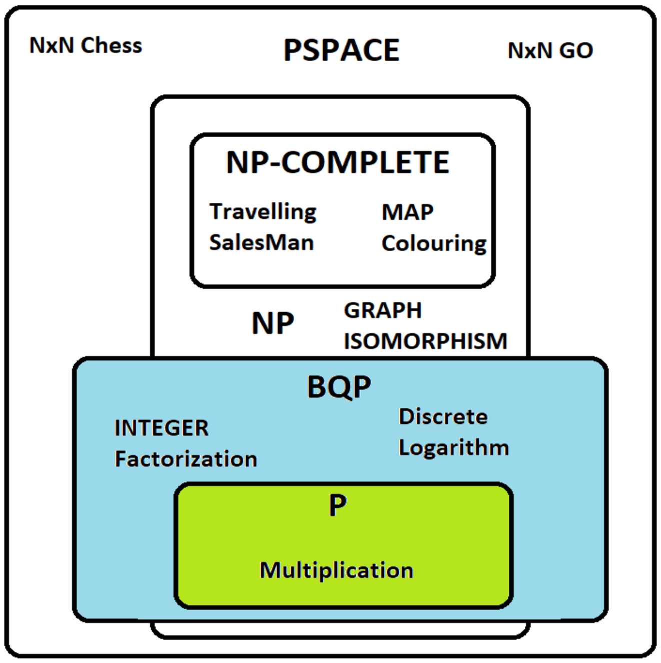


Figure 20: Quantum Complexity

There are a number of problems that you can solve on a quantum computer that are considered intractable on classical computers.

There are many quantum algorithms, such as Shor's algorithm. If you have two large numbers and multiply them together there is a very fast, efficient, classical algorithm for finding the answer.

However, It is a lot more difficult if you start with the answer and then ask, what are the original numbers that multiply together to make this number. This is known as factorization, and these numbers are called factors. The reason finding them is so hard is because the search space of possible factors is so large. Unfortunately, there is no efficient classical algorithm for finding the factors of large numbers.

For this reason, we use this mathematical property for internet encryption: secure websites, emails, and bank accounts.

If you know these factors you can easily decrypt the information, but if you don't, you'd need to find them first which is intractable on the world's most powerful computers.

3.3 Quantum Cryptography

In Cryptography, the process of encryption – scrambling a message using a secret key – is usually fast to perform while the process of decryption which is unscrambling the message without knowing the secret key is slow or, in other words, an exponential time problem. If we have kept the encryption key secret, we will have an encryption system impossible to break.

3.3.1 Transporting 2 classical bits in single qubit

Quantum information processing allows someone to send two classical bits to another person using just a single qubit of communication. This procedure is known as superdense coding.

Consider the quantum circuit shown in Figure 21 to see how two classical bits can be embedded into a single qubit. In this procedure, the quantum circuit is virtually divided into 3 segments where each segment has visibility to only one person; Eve, Alice, and Bob respectively.

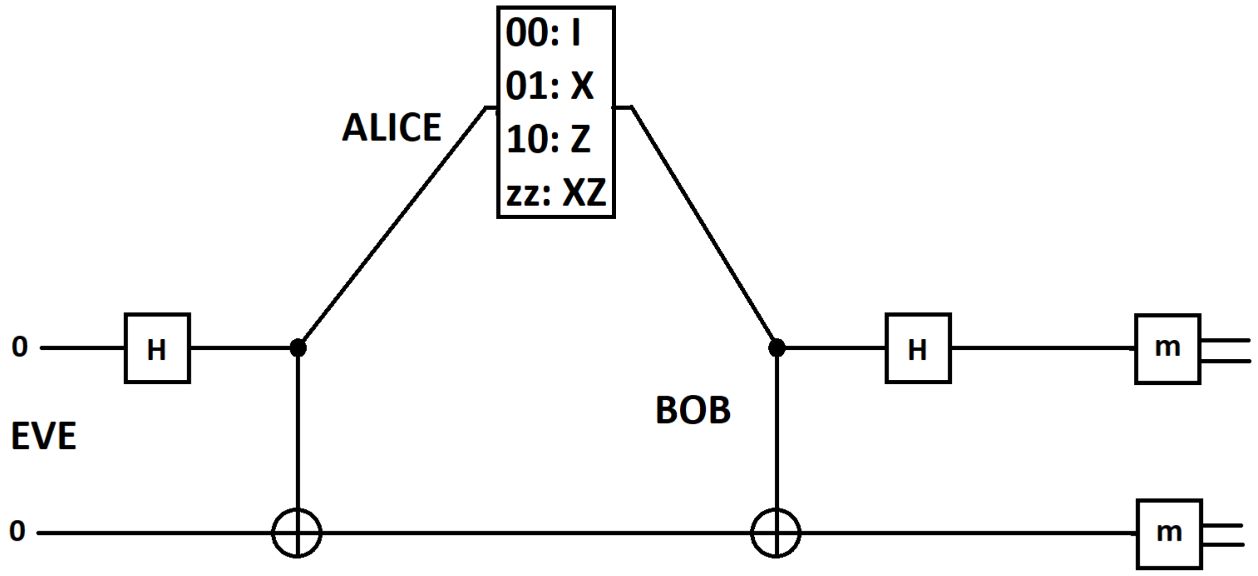


Figure 21: Superdense coding

In this superdense coding protocol, we shall use a special quantum state called Bell state. Bell state is an example of a special type of quantum state called entangled quantum state.

The Bell state $\frac{|00>+|11>}{\sqrt{2}}$ cannot be mentioned in terms of 1st and 2nd qubits separately.

Any state with the property that they cannot be decomposed into separate qubits is the entangled state. Bell state is just one of many such entangled states. These entangled states are different than classical states and are essential in the workings of the quantum computers.

Any quantum computation which never generates entangled quantum states can be efficiently simulated on a classical computer. The above also means that for a quantum algorithm to be much faster than a corresponding classical algorithm, it must necessarily use entangled state.

Eve has two quantum bits which are in |0> state as shown in Figure 21. As mentioned earlier, it is essential to prepare Bell state so that we could leverage the power of quantum computing and to do so there could be many ways to prepare a Bell state. In this scenario, we shall use Hadamard gate followed by cNOT gate to prepare Bell state as shown in Figure 22 which is an initial segment from Figure 21.

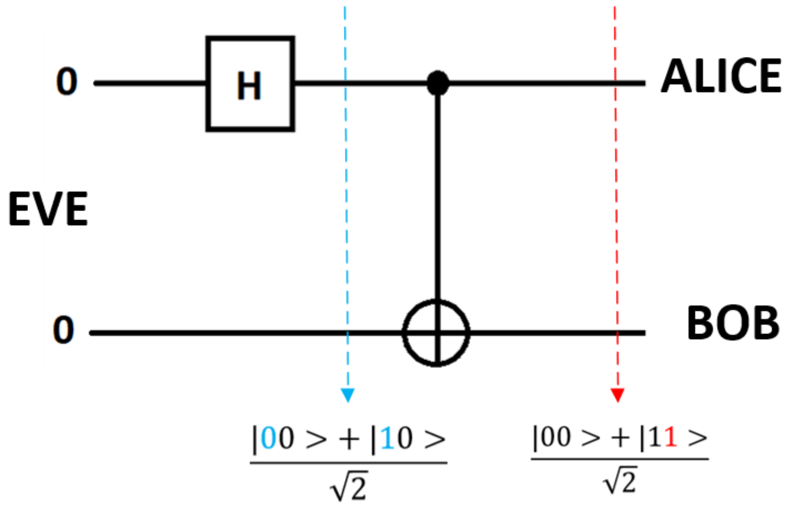

0
H
ALICE
EVE
0
BOB
$\frac{|00> + |10>}{\sqrt{2}}$
$\frac{|00> + |11>}{\sqrt{2}}$

Figure 22: Bell State preparation

The 2 qubits in state |0> with Eve are transformed into the quantum state $\frac{|00>+|10>}{\sqrt{2}}$ after passing through Hadamard gate. Then this intermediate quantum state is passed through cNOT gate to result in Bell state $\frac{|00>+|11>}{\sqrt{2}}$.

Once Eve prepares the Bell state, she sends one qubit to Alice and the other to Bob. Alice wants to communicate 2 classical bits over to Bob using 1 qubit she received from Eve. Alice wants to embed classical bits in a specific way such that only Bob understands them.

If Alice wants to send two classical bits as 00, then she doesn't want to change the quantum state of that single qubit she received from Eve. This communication can be specified by notation 00 : I where 00 are the classical bits and I is the Identity (or no change). In this case the quantum state of qubit sent by Alice to Bob will be $\frac{|00>+|11>}{\sqrt{2}}$.

If Alice wants to send two classical bits as 01, then she will apply a quantum NOT gate to change the quantum state of that single qubit she received from Eve. This communication can be specified by notation 01 : X where 01 are the classical bits and X represents the quantum NOT gate. In this case the quantum state of qubit sent by Alice to Bob will be $\frac{|10>+|01>}{\sqrt{2}}$.

If Alice wants to send two classical bits as 10, then she will apply a quantum Z gate to change the quantum state of that single qubit she received from Eve. This communication can be specified by notation 10 : Z where 10 are the classical bits and Z represents the quantum Z gate. In this case the quantum state of qubit sent by Alice to Bob will be $\frac{|00>-|11>}{\sqrt{2}}$.

If Alice wants to send two classical bits as 11, then she will apply a quantum X and Z gates to change the quantum state of that single qubit she received from Eve. This communication can be specified by notation 11 : XZ where 11 are the classical bits and XZ represents the quantum X and Z gates. In this case, the quantum state of qubit sent by Alice to Bob will be $\frac{|10>-|01>}{\sqrt{2}}$.

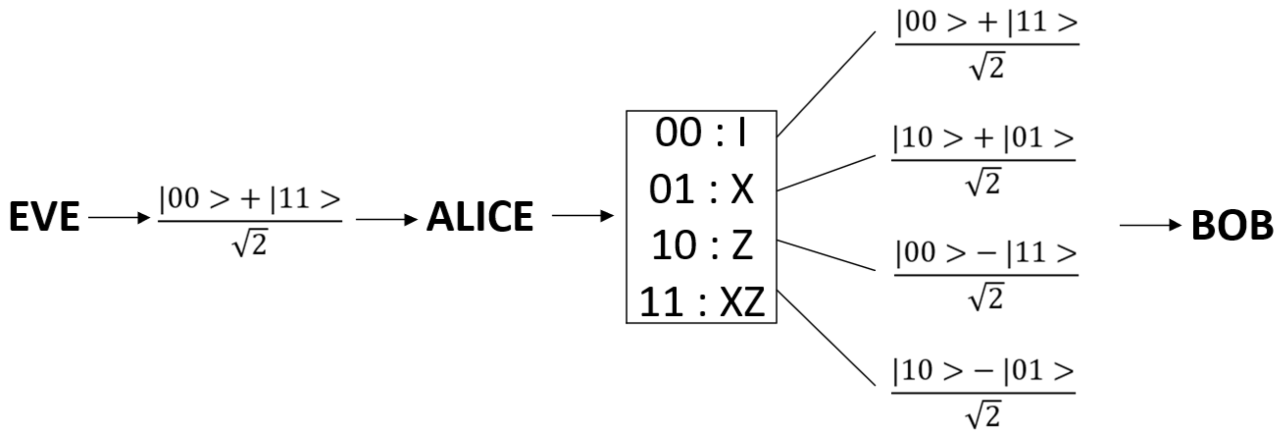

EVE
$\frac{|00>+|11>}{\sqrt{2}}$
ALICE
00 : I
01 : X
10 : Z
11 : XZ
$\frac{|00>+|11>}{\sqrt{2}}$
$\frac{|10>+|01>}{\sqrt{2}}$
$\frac{|00>-|11>}{\sqrt{2}}$
$\frac{|10>-|01>}{\sqrt{2}}$
BOB

Figure 23: Classical bits embedded into Quantum bits

Alice has encoded this classical information in that single qubit she received from Eve and is sent over to Bob as shown in Figure 23 which is a segment from Figure 21.

Bob receives 2 qubits, 1 from Eve and the other from Alice. Alice's qubit starts out entangled with Bob's qubit and it is with this entangled state that it's possible to store the two bits of classical information. If Alice's qubit is not entangled in the first place, then most information we can store on that 1 qubit is just 1 classical bit.

The qubits received by Bob are in one of the 4 quantum states (00+11, 10+01, 00-11 & 10-01). These 4 quantum states are called Bell states which are further processed by Bob to extract information sent by Alice.

For Bob to distinguish these 4 states, a decoding circuit is required, and this decoding circuit can have the quantum logic gates in the reverse order to that of the encoding circuit we used. That is the 2-qubit received by Bob can be first sent to a cNOT gate and then Hadamard gate.

So, if the cNOT gate is first applied to the 2 qubits received by Bob, then the outputs are as follows:

00+11 → cNOT gate → 00+10

10+01 → cNOT gate → 11+01

00-11 → cNOT gate → 00-10

10-01 → cNOT gate → 11-01

When Hadamard gate is applied to the output of cNOT gates then the following outputs can be observed:

00+10 →Hadamard gate→ $\frac{|00>+|10>}{\sqrt{2}} + \frac{|00>-|10>}{\sqrt{2}} = 00$

11+01 →Hadamard gate→ $\frac{|01>-|11>}{\sqrt{2}} + \frac{|01>+|11>}{\sqrt{2}} = 01$

00-10 →Hadamard gate→ $\frac{|00>+|10>}{\sqrt{2}} - \frac{|00>-|10>}{\sqrt{2}} = 10$

11-01 →Hadamard gate→ $\frac{|01>+|11>}{\sqrt{2}} - \frac{|01>-|11>}{\sqrt{2}} = -11$

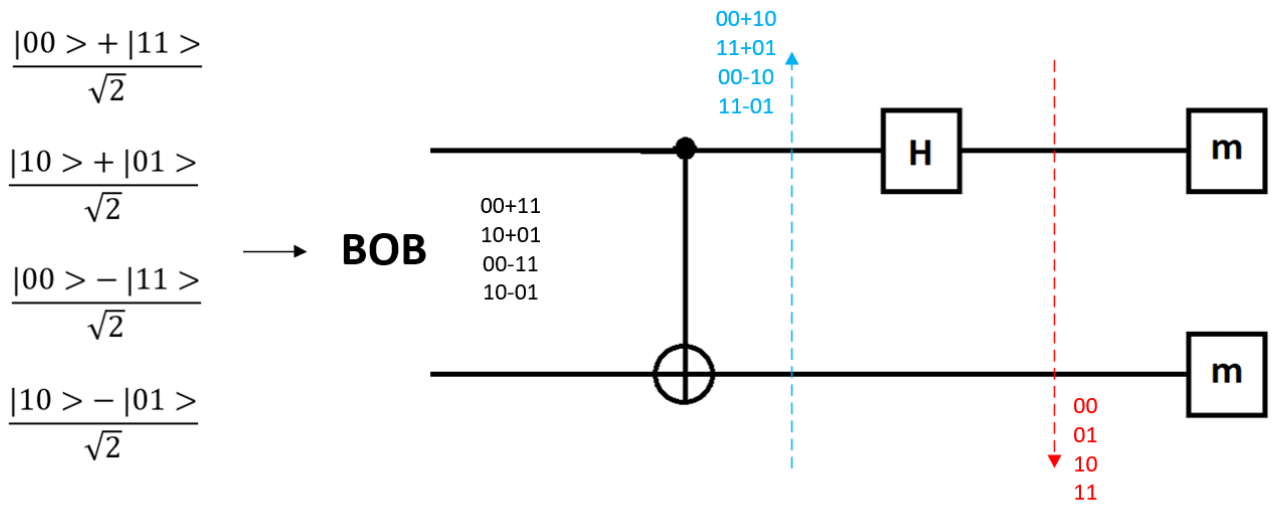

$\frac{|00> + |11>}{\sqrt{2}}$
$\frac{|10> + |01>}{\sqrt{2}}$
$\frac{|00> - |11>}{\sqrt{2}}$
$\frac{|10> - |01>}{\sqrt{2}}$
BOB
00+11
10+01
00-11
10-01
00+10
11+01
00-10
11-01
H
m
m
00
01
10
11

Figure 24: Decoding the classical bits from quantum bits

These outputs are measured as shown in Figure 24 and Bob would be able to decode the classical information sent by Alice. Hence, this procedure enables transmission of 2 classical bits using 1 quantum bit. In a way, this can be seen as encoding and decoding classical information using quantum bits. There are other ways such as Quantum Teleportation where the quantum properties can be leveraged to transmit information in 2 quantum bits via single classical bit.

Conclusion

Shor’s algorithm is an example which took advantage of the special features of quantum computers to create an algorithm that could solve integer factorization with much better scaling much than the best classical algorithm. The best classical algorithm is exponential, whereas Shor’s algorithm is polynomial which is a huge deal in the world of complexity theory and computer science in general because it makes an intractable problem into one that can be solved.

There is no conclusion to this topic but what’s interesting to note is that some of the problems or use cases which can’t be solved using classical computers can be solved using quantum computers by leveraging quantum properties to our advantage. During this process, the information toggles between quantum and classical bits thereby pointing out the need for coexistence of both classical and quantum computers. In other words, hybrid computing is going to be the means for the future of computing.

Bibliography

1) Nielsen, M. A., & Chuang, I. (2002). Quantum computation and quantum information.

2) The Map of Quantum Computing | Quantum Computers Explained. (2021, December 3). [Video]. YouTube.

3) Griffiths, D. J., & Schroeter, D. F. (2018). Introduction to quantum mechanics. Cambridge university press.

4) What is complexity theory? (P vs. NP explained visually). (2017, October 5). [Video]. YouTube.

5) McMahon, D. (2007). Quantum computing explained. John Wiley & Sons.

6) Hidary, J. D., & Hidary, J. D. (2019). Quantum computing: an applied approach (Vol. 1). Cham: Springer.