QUANTUM COMPUTING PyQ

Q. Describe the physical conditions required for quantum computing

1. Robustly Representing Quantum Information

Quantum information is encoded in **qubits**, which can exist in a superposition of 0 and 1. To **robustly represent** this information:

- **Qubits** must have:
  - **Long coherence time** (time before quantum information decays).
  - **Low error rates** due to environmental noise (decoherence).
- Physical realizations:
  - **Superconducting qubits**: circuits with Josephson junctions.
  - **Trapped ions**: internal states of individual ions in electromagnetic fields.
  - **Spin qubits**: electron spins in silicon or diamond.
- **Isolation & Cooling**: Near absolute-zero temps are needed to prevent noise.
- **Error Correction**: Since no physical qubit is perfect, **quantum error correction** (QEC) is used to encode information redundantly across many qubits.

Quantum computation is based on transformation of quantum states. Quantum bits are two-level quantum systems, and as the simplest elementary building blocks for a quantum computer, they provide a convenient labeling for pairs of states and their physical realizations. Thus, for example, the four states of a spin-3/2 particle, $|m = +3/2\rangle, |m = +1/2\rangle, |m = -1/2\rangle, |m = -3/2\rangle$, could be used to represent two qubits.

For the purpose of computation, the crucial realization is that the set of accessible states should be *finite*. The position x of a particle along a one-dimensional line is not generally a good set of states for computation, even though the particle may be in a quantum state $|x\rangle$, or even some superposition $\sum_x c_x|x\rangle$. This is because x has a continuous range of possibilities, and the Hilbert space has infinite size, so that in the absence of noise the information capacity is infinite. For example, in a perfect world, the entire texts of Shakespeare could be stored in (and retrieved from) the infinite number of digits in the binary fraction $x = 0.010111011001\ldots$. This is clearly unrealistic; what happens in reality is that the presence of noise reduces the number of distinguishable states to a finite number.

In fact, it is generally desirable to have some aspect of symmetry dictate the finiteness of the state space, in order to minimize decoherence. For example, a spin-1/2 particle lives in a Hilbert space spanned by the $|\uparrow\rangle$ and $|\downarrow\rangle$ states; the spin state cannot be anything outside this two-dimensional space, and thus is a nearly ideal quantum bit when well isolated.

If the choice of representation is poor, then decoherence will result. For example, as described in Box 7.1, a particle in a finite square well which is just deep enough to contain two bound states would make a mediocre quantum bit, because transitions from the bound states to the continuum of unbound states would be possible. These would lead to decoherence since they could destroy qubit superposition states. For single qubits, the figure of merit is the minimum lifetime of arbitrary superposition states; a good measure, used for spin states and atomic systems, is T_2, the ('transverse') relaxation time of states such as $(|0\rangle + |1\rangle)/\sqrt{2}$. Note that T_1, the ('longitudinal') relaxation time of the higher energy $|1\rangle$ state, is just a *classical* state lifetime, which is usually longer than T_2.

2. Performing a Universal Family of Unitary Transformations

To compute, we apply **unitary operations** (quantum gates) that evolve the state of qubits. A **universal set** of quantum gates must be able to approximate **any unitary transformation** on qubits.

- Common universal sets:
  - **Single-qubit rotations**: e.g., Hadamard (H), Pauli-X/Y/Z, phase gates.
  - **Two-qubit gates**: e.g., CNOT or CZ (Controlled-Z), which introduce entanglement.
- These gates must be:
  - **Precise** and **fast** (faster than decoherence time).
  - **Scalable** to many qubits.
- Implementation:
  - In superconducting systems: microwave pulses control the gates.
  - In trapped ions: lasers perform gate operations by changing energy levels.

Closed quantum systems evolve unitarily as determined by their Hamiltonians, but to perform quantum computation one must be able to control the Hamiltonian to effect an *arbitrary* selection from a universal family of unitary transformations (as described in Section 4.5). For example, a single spin might evolve under the Hamiltonian $H = P_x(t)X + P_y(t)Y$, where $P_{\{x,y\}}$ are classically controllable parameters. From Exercise 4.10, we know that by manipulating P_x and P_y appropriately, one can perform arbitrary single spin rotations.

According to the theorems of Section 4.5, any unitary transform can be composed from single spin operations and controlled-NOT gates, and thus realization of those two kinds of quantum logic gates are natural goals for experimental quantum computation. However, implicitly required also is the ability to address individual qubits, and to apply these gates to select qubits or pairs of qubits. This is not simple to accomplish in many physical systems. For example, in an ion trap, one can direct a laser at one of many individual ions to selectively excite it, but only as long as the ions are spatially separated by a wavelength or more.

Two important figures of merit for unitary transforms are the minimum achievable fidelity $\mathcal{F}$ (Chapter 9), and the maximum time t_{op} required to perform elementary operations such as single spin rotations or a controlled-NOT gate.

3. Preparing a Fiducial Input State

Before computation starts, qubits must be initialized to a **well-defined starting state** — typically $|0\rangle$.

- Methods:
  - **Passive cooling**: Qubits naturally relax to ground state at low temperatures.
  - **Active reset**: Fast techniques to force a qubit into a known state (e.g., measurement-based reset or optical pumping).
- Importance:
  - Ensures that the quantum algorithm starts from a **predictable and reliable state**.
  - Reduces noise from random initial conditions.

One of the most important requirements for being able to perform a useful computation, even classically, is to be able to prepare the desired input. If one has a box which can perform perfect computations, what use is it if numbers cannot be input? With classical machines, establishing a definite input state is rarely a difficulty – one merely sets some switches in the desired configuration and that defines the input state. However, with quantum systems this can be very difficult, depending on the realization of qubits.

Note that it is only necessary to be able to (repeatedly) produce one specific quantum state with high fidelity, since a unitary transform can turn it into any other desired input state. For example, being able to put n spins into the $|00\ldots0\rangle$ state is good enough. The fact that they may not stay there for very long due to thermal heating is a problem with the choice of representation.

Two figures of merit are relevant to input state preparation: the minimum fidelity with which the initial state can be prepared in a given state ρ_{in}, and the entropy of ρ_{in}. The entropy is important because, for example, it is very easy to prepare the state $\rho_{\text{in}} = I/2^n$ with high fidelity, but that is a useless state for quantum computation, since it is invariant under unitary transforms! Ideally, the input state is a pure state, with zero entropy. Generally, input states with non-zero entropy reduce the accessibility of the answer from the output result.

4. Measuring the Output Result

Quantum measurements collapse a superposition to one of the basis states (e.g.,$|0\rangle$ or $|1\rangle$), giving a classical output.

- Properties:
  - **Projective measurement**: Forces the quantum system into one of the possible basis states.
  - **Repeatability**: Quantum algorithms often require repeated measurements to estimate probabilities/statistics.
- Technology:
  - In superconducting qubits: **dispersive readout** using resonators.
  - In trapped ions: **fluorescence detection** using lasers to count photons.
- Challenge:
  - Measurement must be **high-fidelity** (accurate) and **non-destructive** if used mid-computation.

> What measurement capability is required for quantum computation? For the purpose of the present discussion, let us think of measurement as a process of coupling one or more qubits to a classical system such that after some interval of time, the state of the qubits is indicated by the state of the classical system. For example, a qubit state $a|0\rangle + b|1\rangle$, represented by the ground and excited states of a two-level atom, might be measured by pumping the excited state and looking for fluorescence. If an electrometer indicates that fluorescence had been detected by a photomultiplier tube, then the qubit would collapse into the $|1\rangle$ state; this would happen with probability $|b|^2$. Otherwise, the electrometer would detect no charge, and the qubit would collapse into the $|0\rangle$ state.
>
> An important characteristic of the measurement process for quantum computation is the wavefunction collapse which describes what happens when a projective measurement is performed (Section 2.2.5). The output from a good quantum algorithm is a superposition state which gives a useful answer with high probability when measured. For example, one step in Shor's quantum factoring algorithm is to find an integer r from the measurement result, which is an integer close to qc/r, where q is the dimension of a Hilbert space. The output state is actually in a nearly uniform superposition of all possible values of c, but a measurement collapses this into a single, random integer, thus allowing r to be determined with high probability (using a continued fraction expansion, as was described in Chapter 5).

Q. Show mathematically that if message rate exceeds channel capacity, reliable communication is not possible over a classical channel.

Reliable decoding of unreliable messages

Converse of Shannon's main Theorem

$M < 2^{nC}$ ← Shannon.

message rate $= \frac{\log M}{n} = C$.

channel capacity = ●↗

Let $M = 2^{n(C+\epsilon)}$; violation $\epsilon \to 0$

Analysing channel behaviour over long sequency

$A^n \to$ consecutive I/P symbols

$B^n \to$ consecutive o/p symbols

For n^{th} extension

$$H(A^n)\, H(A^n|B^n) \leq nC \quad -③$$

$p = \frac{1}{m}$ = prob of choosing 1 out of m messages

$$H(A^n) = \sum_{n} \frac{1}{M} \log M = \log M = n(C+\epsilon) \quad -●④$$

using ③ and ④

$$n(C+\epsilon) - nC = n\epsilon \leq H(A^n|B^n)$$

(17)

Fano's bound

$$n\epsilon \leq H(A^n|B^n) \leq H(P_E) + P_E \log(q-1)$$

$H(P_E) \leq 1$; $q-1 < q = M$.

$\Rightarrow n\epsilon \leq 1 + P_E \log M$

$\Rightarrow n\epsilon \leq 1 + P_E(nC + n\epsilon)$

$\Rightarrow P_E(nC + n\epsilon) \geq n\epsilon - 1$

$\Rightarrow P_E = \frac{n\epsilon - 1}{nC + n\epsilon}$

$\Rightarrow P_E = \frac{\epsilon - \frac{1}{n}}{C+\epsilon} \geq \frac{\epsilon}{C}$; bounded away .. doesn't vanish

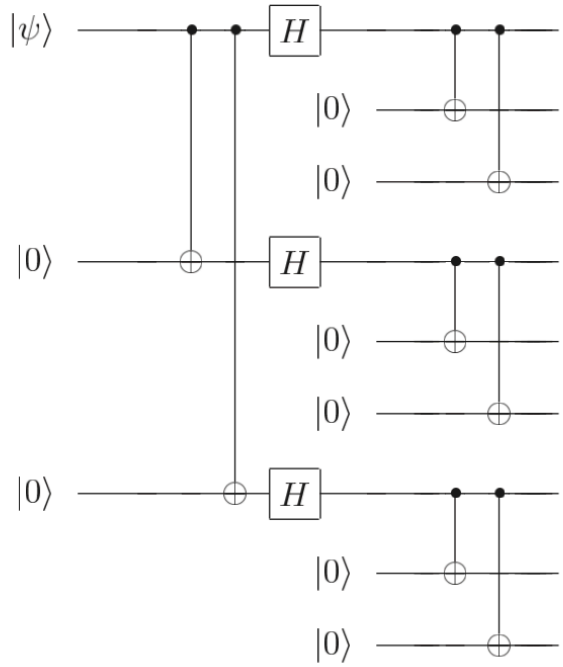
So, message rate exceeding channel capacity doesn't ensure reliable decoding.

Q. Discuss about a coding scheme that ensures correction of both bit flip as well as phase flip error in quantum communication.

10.2 The Shor code

There is a simple quantum code which can protect against the effects of an *arbitrary* error on a single qubit! The code is known as the *Shor code*, after its inventor. The code is a combination of the three qubit phase flip and bit flip codes. We first encode the qubit using the phase flip code: $|0\rangle \rightarrow |+++\rangle, |1\rangle \rightarrow |---\rangle$. Next, we encode each of these qubits using the three qubit bit flip code: $|+\rangle$ is encoded as $(|000\rangle + |111\rangle)\sqrt{2}$ and $|-\rangle$ is encoded as $(|000\rangle - |111\rangle)\sqrt{2}$. The result is a nine qubit code, with codewords given by:

$$|0\rangle \rightarrow |0_L\rangle \equiv \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}$$
$$|1\rangle \rightarrow |1_L\rangle \equiv \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}. \quad (10.13)$$



The quantum circuit encoding the Shor code is shown in Figure 10.4. As described above, the first part of the circuit encodes the qubit using the three qubit phase flip code;

comparison with Figure 10.3 shows that the circuits are identical. The second part of the circuit encodes each of these three qubits using the bit flip code, using three copies of the bit flip code encoding circuit, Figure 10.2. This method of encoding using a hierarchy of levels is known as *concatenation*. It's a great trick for obtaining new codes from old, and we use it again later to prove some important results about quantum error-correction.

The Shor code is able to protect against phase flip and bit flip errors on any qubit. To see this, suppose a bit flip occurs on the first qubit. As for the bit flip code, we perform a measurement of $Z_1 Z_2$ comparing the first two qubits, and find that they are different. We conclude that a bit flip error occurred on the first or second qubit. Next we compare the second and third qubit by performing a measurement of $Z_2 Z_3$. We find that they are the same, so it could not have been the second qubit which flipped. We conclude that the first qubit must have flipped, and recover from the error by flipping the first qubit again, back to its original state. In a similar way we can detect and recover from the effects of bit flip errors on any of the nine qubits in the code.

We cope in a similar manner with phase flips on the qubits. Suppose a phase flip occurs on the first qubit. Such a phase flip flips the sign of the first block of qubits, changing $|000\rangle + |111\rangle$ to $|000\rangle - |111\rangle$, and vice versa. Indeed, a phase flip on *any* of the first three qubits has this effect, and the error-correction procedure we describe works for any of these three possible errors. Syndrome measurement begins by comparing the sign of the first and second blocks of three qubits, just as syndrome measurement for the phase flip code began by comparing the sign of the first and second qubits. For example, $(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)$ has the same sign $(-)$ in both blocks of qubits, while $(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)$ has different signs. When a phase flip occurs on any of the first three qubits we find that the signs of the first and second blocks are different. The second and final stage of syndrome measurement is to compare the sign of the second and third blocks of qubits. We find that these are the same, and conclude that the phase must have flipped in the first block of three qubits. We recover from this by flipping the sign in the first block of three qubits back to its original value. We can recover from a phase flip on any of the nine qubits in a similar manner.

Suppose both bit and phase flip errors occur on the first qubit, that is, the operator $Z_1 X_1$ is applied to that qubit. Then it is easy to see that the procedure for detecting a bit flip error will detect a bit flip on the first qubit, and correct it, and the procedure for detecting a phase flip error will detect a phase flip on the first block of three qubits, and correct it. Thus, the Shor code also enables the correction of combined bit and phase flip errors on a single qubit.

Q. Describe an algorithm that uses QFT and number theory to perform factorisation of a number into its two prime factors and give an illustrative example

Procedure:

1. If N is even, return the factor 2.
2. Determine whether $N = a^b$ for integers $a \geq 1$ and $b \geq 2$, and if so return the factor a (uses the classical algorithm of Exercise 5.17).
3. Randomly choose x in the range 1 to $N-1$. If $\gcd(x, N) > 1$ then return the factor $\gcd(x, N)$.
4. Use the order-finding subroutine to find the order r of x modulo N.
5. If r is even and $x^{r/2} \neq -1 \pmod{N}$ then compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$, and test to see if one of these is a non-trivial factor, returning that factor if so. Otherwise, the algorithm fails.

Box 5.4: Factoring 15 quantum-mechanically

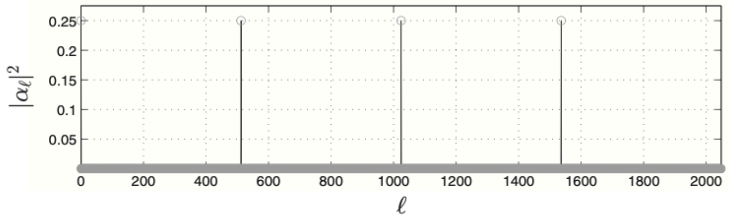
The use of order-finding, phase estimation, and continued fraction expansions in the quantum factoring algorithm is illustrated by applying it to factor $N = 15$. First, we choose a random number which has no common factors with N; suppose we choose $x = 7$. Next, we compute the order r of x with respect to N, using the quantum order-finding algorithm: begin with the state $|0\rangle|0\rangle$ and create the state

$$\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle|0\rangle = \frac{1}{\sqrt{2^t}} \left[|0\rangle + |1\rangle + |2\rangle + \cdots + |2^t - 1\rangle\right] |0\rangle \tag{5.61}$$

by applying $t = 11$ Hadamard transforms to the first register. Choosing this value of t ensures an error probability ϵ of at most $1/4$. Next, compute $f(k) = x^k \bmod N$, leaving the result in the second register,

$$\begin{aligned} &\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle|x^k \bmod N\rangle \qquad (5.62) \\ &= \frac{1}{\sqrt{2^t}} \left[|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + \cdots\right] . \end{aligned}$$

We now apply the inverse Fourier transform $FT^\dagger$ to the first register and measure it. One way of analyzing the distribution of outcomes obtained is to calculate the reduced density matrix for the first register, and apply $FT^\dagger$ to it, and calculate the measurement statistics. However, since no further operation is applied to the second register, we can instead apply the principle of implicit measurement (Section 4.4) and assume that the second register is measured, obtaining a *random* result from 1, 7, 4, or 13. Suppose we get 4 (any of the results works); this means the state input to $FT^\dagger$ would have been $\sqrt{\frac{4}{2^t}} \left[|2\rangle + |6\rangle + |10\rangle + |14\rangle + \cdots\right]$. After applying $FT^\dagger$ we obtain some state $\sum_\ell \alpha_\ell |\ell\rangle$, with the probability distribution



shown for $2^t = 2048$. The final measurement therefore gives either 0, 512, 1024, or 1536, each with probability almost exactly $1/4$. Suppose we obtain $\ell = 1536$ from the measurement; computing the continued fraction expansion thus gives $1536/2048 = 1/(1 + (1/3))$, so that $3/4$ occurs as a convergent in the expansion, giving $r = 4$ as the order of $x = 7$. By chance, r is even, and moreover, $x^{r/2} \bmod N = 7^2 \bmod 15 = 4 \neq -1 \bmod 15$, so the algorithm works: computing the greatest common divisor $\gcd(x^2 - 1, 15) = 3$ and $\gcd(x^2 + 1, 15) = 5$ tells us that $15 = 3 \times 5$.

Q. Show that the QFT resembles unitary transform and provides drastic complexity improvement over the classical fast Fourier transform.

The product representation (5.4) makes it easy to derive an efficient circuit for the quantum Fourier transform. Such a circuit is shown in Figure 5.1. The gate R_k denotes the unitary transformation

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}. \tag{5.11}$$

To see that the pictured circuit computes the quantum Fourier transform, consider what happens when the state $|j_1 \ldots j_n\rangle$ is input. Applying the Hadamard gate to the first bit produces the state

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right) |j_2 \ldots j_n\rangle, \tag{5.12}$$

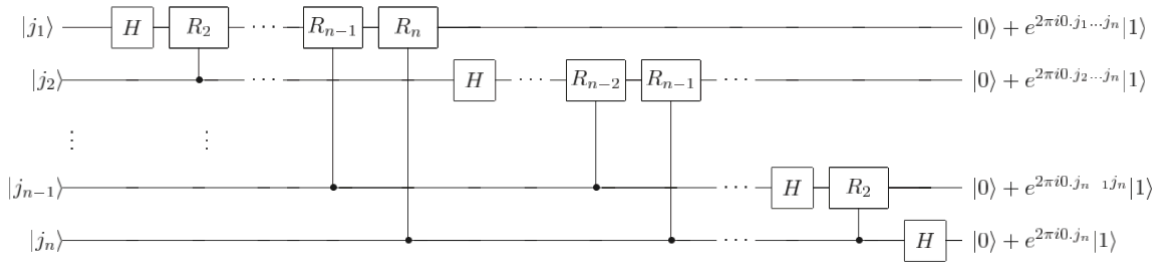


Figure 5.1. Efficient circuit for the quantum Fourier transform. This circuit is easily derived from the product representation (5.4) for the quantum Fourier transform. Not shown are swap gates at the end of the circuit which reverse the order of the qubits, or normalization factors of $1/\sqrt{2}$ in the output.

since $e^{2\pi i 0.j_1} = -1$ when $j_1 = 1$, and is $+1$ otherwise. Applying the controlled-R_2 gate produces the state

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right) |j_2 \ldots j_n\rangle. \tag{5.13}$$

We continue applying the controlled-R_3, R_4 through R_n gates, each of which adds an extra bit to the phase of the co-efficient of the first $|1\rangle$. At the end of this procedure we have the state

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right) |j_2 \ldots j_n\rangle. \tag{5.14}$$

Next, we perform a similar procedure on the second qubit. The Hadamard gate puts us in the state

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right) |j_3 \ldots j_n\rangle, \tag{5.15}$$

and the controlled-R_2 through R_{n-1} gates yield the state

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle\right) |j_3 \ldots j_n\rangle. \tag{5.16}$$

We continue in this fashion for each qubit, giving a final state

$$\frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle\right) \ldots \left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right). \tag{5.17}$$

Swap operations (see Section 1.3.4 for a description of the circuit), omitted from Figure 5.1 for clarity, are then used to reverse the order of the qubits. After the swap

operations, the state of the qubits is

$$\frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \ldots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right) . \quad (5.18)$$

Comparing with Equation (5.4) we see that this is the desired output from the quantum Fourier transform. This construction also proves that the quantum Fourier transform is unitary, since each gate in the circuit is unitary. An explicit example showing a circuit for the quantum Fourier transform on three qubits is given in Box 5.1.

How many gates does this circuit use? We start by doing a Hadamard gate and $n-1$ conditional rotations on the first qubit – a total of n gates. This is followed by a Hadamard gate and $n-2$ conditional rotations on the second qubit, for a total of $n+(n-1)$ gates. Continuing in this way, we see that $n+(n-1)+\cdots+1 = n(n+1)/2$ gates are required, plus the gates involved in the swaps. At most $n/2$ swaps are required, and each swap can be accomplished using three controlled-NOT gates. Therefore, this circuit provides a $\Theta(n^2)$ algorithm for performing the quantum Fourier transform.

In contrast, the best classical algorithms for computing the discrete Fourier transform on 2^n elements are algorithms such as the *Fast Fourier Transform (FFT)*, which compute the discrete Fourier transform using $\Theta(n2^n)$ gates. That is, it requires exponentially more operations to compute the Fourier transform on a classical computer than it does to implement the quantum Fourier transform on a quantum computer.

Q. Quantum Search Algorithm based on Grover iterations

6.1 The quantum search algorithm

Let us begin by setting the stage for the search algorithm in terms of an *oracle*, similar to that encountered in Section 3.1.1. This allows us to present a very general description of the search procedure, and a geometric way to visualize its action and see how it performs.

6.1.1 The oracle

Suppose we wish to search through a search space of N elements. Rather than search the elements directly, we concentrate on the *index* to those elements, which is just a number in the range 0 to $N-1$. For convenience we assume $N = 2^n$, so the index can be stored in n bits, and that the search problem has exactly M solutions, with $1 \leq M \leq N$. A particular instance of the search problem can conveniently be represented by a function f, which takes as input an integer x, in the range 0 to $N-1$. By definition, $f(x) = 1$ if x is a solution to the search problem, and $f(x) = 0$ if x is not a solution to the search problem.

Suppose we are supplied with a quantum *oracle* – a black box whose internal workings we discuss later, but which are not important at this stage – with the ability to *recognize* solutions to the search problem. This recognition is signalled by making use of an *oracle qubit*. More precisely, the oracle is a unitary operator, O, defined by its action on the computational basis:

$$|x\rangle|q\rangle \xrightarrow{O} |x\rangle|q \oplus f(x)\rangle \,, \tag{6.1}$$

where $|x\rangle$ is the index register, $\oplus$ denotes addition modulo 2, and the oracle qubit $|q\rangle$ is a single qubit which is flipped if $f(x) = 1$, and is unchanged otherwise. We can check whether x is a solution to our search problem by preparing $|x\rangle|0\rangle$, applying the oracle, and checking to see if the oracle qubit has been flipped to $|1\rangle$.

In the quantum search algorithm it is useful to apply the oracle with the oracle qubit initially in the state $(|0\rangle - |1\rangle)/\sqrt{2}$, just as was done in the Deutsch–Jozsa algorithm of Section 1.4.4. If x is not a solution to the search problem, applying the oracle to the state $|x\rangle(|0\rangle - |1\rangle)/\sqrt{2}$ does not change the state. On the other hand, if x is a solution to the search problem, then $|0\rangle$ and $|1\rangle$ are interchanged by the action of the oracle, giving a final state $-|x\rangle(|0\rangle - |1\rangle)/\sqrt{2}$. The action of the oracle is thus:

$$|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \xrightarrow{O} (-1)^{f(x)}|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) . \tag{6.2}$$

Notice that the state of the oracle qubit is not changed. It turns out that this remains $(|0\rangle - |1\rangle)/\sqrt{2}$ throughout the quantum search algorithm, and can therefore be omitted from further discussion of the algorithm, simplifying our description.

With this convention, the action of the oracle may be written:

$$|x\rangle \xrightarrow{O} (-1)^{f(x)}|x\rangle \,. \tag{6.3}$$

We say that the oracle *marks* the solutions to the search problem, by shifting the phase of the solution. For an N item search problem with M solutions, it turns out that we need only apply the search oracle $O(\sqrt{N/M})$ times in order to obtain a solution, on a quantum computer.

6.1.2 The procedure

Schematically, the search algorithm operates as shown in Figure 6.1. The algorithm proper makes use of a single n qubit register. The internal workings of the oracle, including the possibility of it needing extra work qubits, are not important to the description of the quantum search algorithm proper. The goal of the algorithm is to find a solution to the search problem, using the smallest possible number of applications of the oracle.

The algorithm begins with the computer in the state $|0\rangle^{\otimes n}$. The Hadamard transform is used to put the computer in the equal superposition state,

$$|\psi\rangle = \frac{1}{N^{1/2}} \sum_{x=0}^{N-1} |x\rangle \, . \tag{6.4}$$

The quantum search algorithm then consists of repeated application of a quantum subroutine, know as the *Grover iteration* or *Grover operator*, which we denote G. The Grover iteration, whose quantum circuit is illustrated in Figure 6.2, may be broken up into four steps:

(1) Apply the oracle O.
(2) Apply the Hadamard transform $H^{\otimes n}$.
(3) Perform a conditional phase shift on the computer, with every computational basis state except $|0\rangle$ receiving a phase shift of -1,

$$|x\rangle \rightarrow -(-1)^{\delta_{x0}}|x\rangle. \tag{6.5}$$

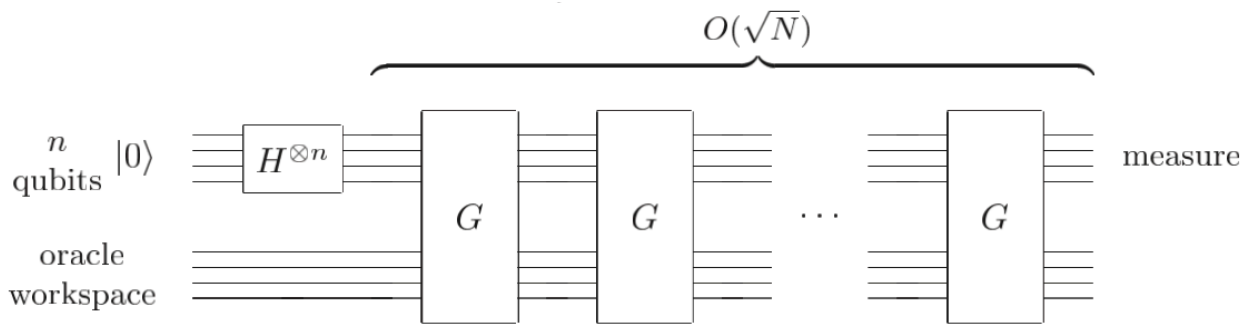
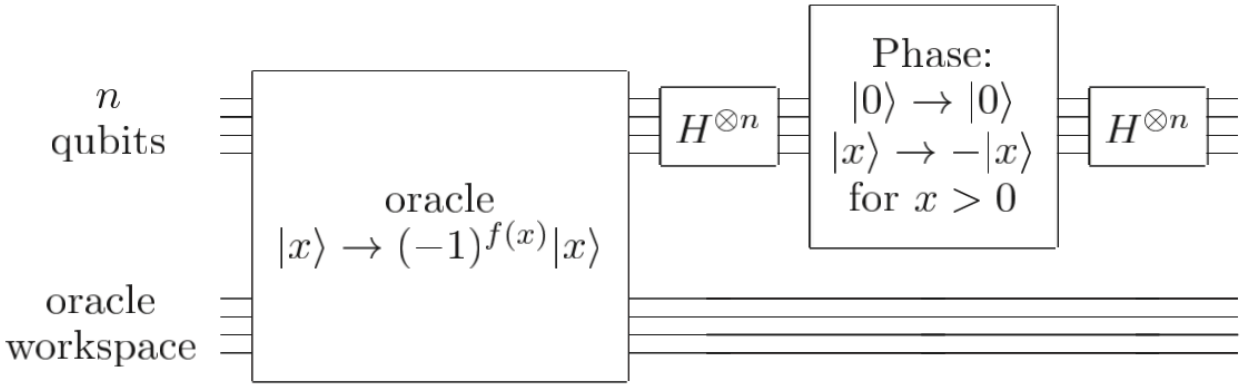


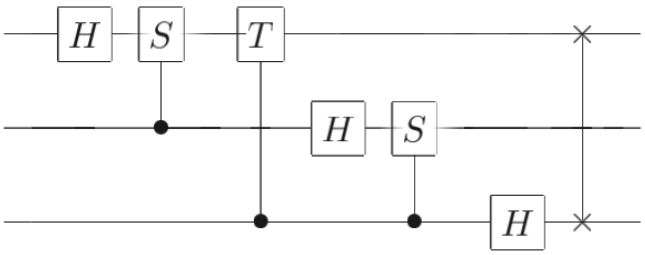
Figure 6.1. Schematic circuit for the quantum search algorithm. The oracle may employ work qubits for its implementation, but the analysis of the quantum search algorithm involves only the n qubit register.



Q. 3 qubit QFT

Box 5.1: Three qubit quantum Fourier transform

For concreteness it may help to look at the explicit circuit for the three qubit quantum Fourier transform:



Recall that S and T are the phase and $\pi/8$ gates (see page xxiii). As a matrix the quantum Fourier transform in this instance may be written out explicitly, using $\omega = e^{2\pi i/8} = \sqrt{i}$, as

$$\frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix}. \quad (5.19)$$

The Quantum Fourier Transform (QFT) is a **unitary transformation** that acts on the amplitudes of a quantum state, analogous to the classical Discrete Fourier Transform (DFT). For an n-qubit quantum register, it maps computational basis states to quantum superpositions weighted by complex roots of unity.

Let $|x\rangle$ be a computational basis state of n qubits, where $x \in \{0, 1, \ldots, 2^n - 1\}$. The QFT maps this state as follows:

$$\mathrm{QFT}(|x\rangle) = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i x k / 2^n} |k\rangle$$

This transformation is linear and unitary, and applies to arbitrary superpositions as well by linearity.

Specialization to 3 Qubits

Let $n = 3$. Then, the computational basis consists of 8 states: $|000\rangle$ through $|111\rangle$, which correspond to decimal values 0 through 7.

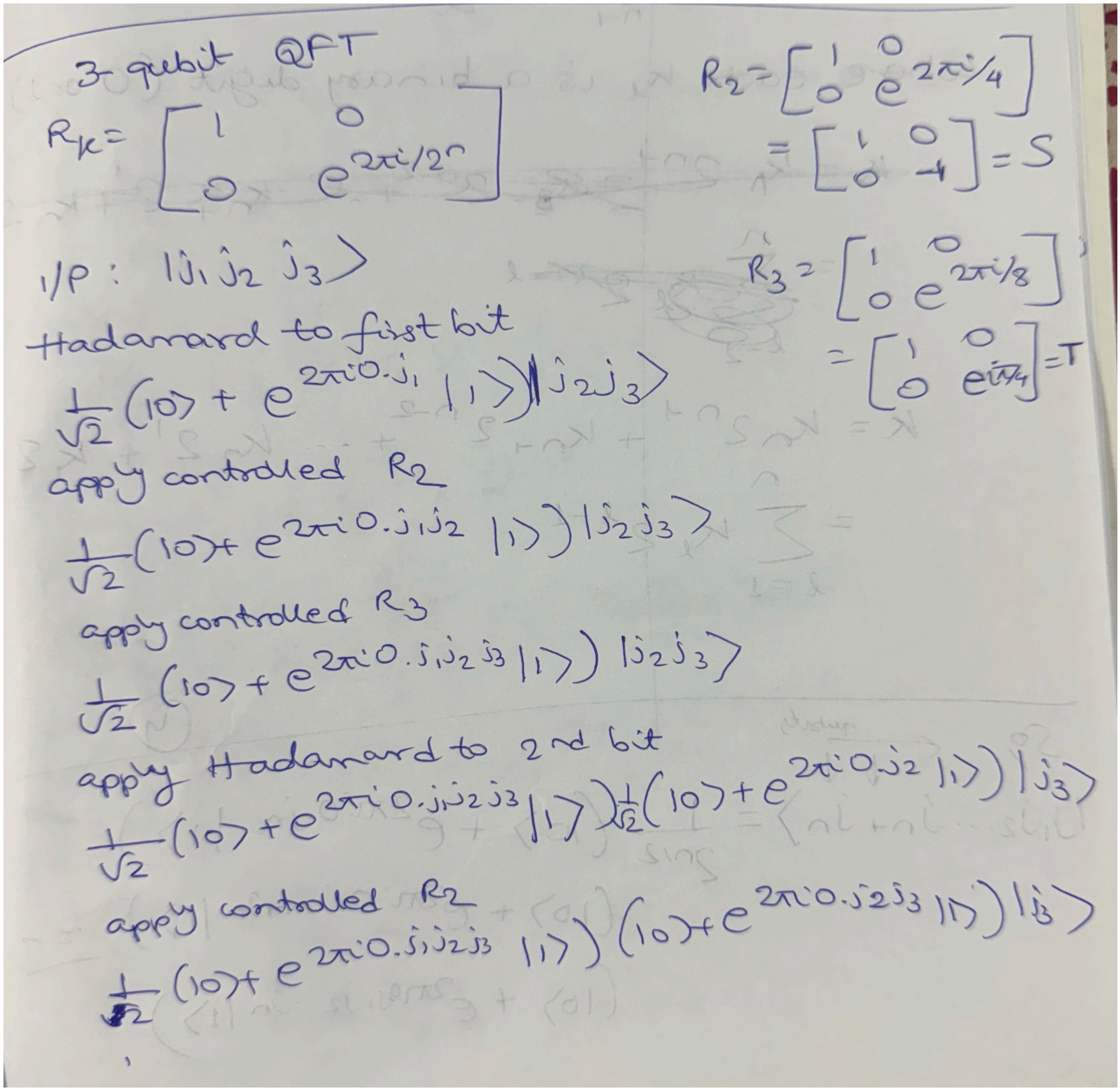
The QFT on 3 qubits transforms:

$$|x\rangle \mapsto \frac{1}{\sqrt{8}} \sum_{k=0}^{7} e^{2\pi i x k/8} |k\rangle$$

for any $x \in \{0, 1, \ldots, 7\}$.

This means:

- Input $|0\rangle$ maps to an **equal superposition** of all 8 basis states.
- Input $|1\rangle$ maps to a **superposition with progressive phase shifts**, corresponding to the 8th roots of unity.



apply Hadamard to 3rd qubit

$$\frac{1}{2\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle\right)$$
$$\left(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle\right)$$
$$\left(|0\rangle + e^{2\pi i 0.j_3}|1\rangle\right)$$

Swapping 1st & 3rd qubit

$$\frac{1}{2\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_3}|1\rangle\right)$$
$$\left(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle\right)$$
$$\left(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle\right)$$

Q. Discuss about properties of quantum computing that make it unique

4. Measurement Collapse

Definition

Quantum measurement is a fundamentally probabilistic process. When a quantum system is measured, the act of measurement **collapses the quantum state** (superposition) to a specific classical outcome, with a probability given by the square of the amplitude of that outcome.

Formally, for a qubit in state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

A measurement in the computational basis (standard basis) yields:

- **Outcome |0⟩** with probability $|\alpha|^2$
- **Outcome |1⟩** with probability $|\beta|^2$

Post-measurement, the system **collapses** to the measured state, i.e., all superposition is lost.

Mathematical Framework

Measurement is described by a set of **projection operators** (or more generally, Positive Operator-Valued Measures - POVMs). In the simplest case:

$$P_0 = |0\rangle\langle 0|, \quad P_1 = |1\rangle\langle 1|$$

Given a state $|\psi\rangle$, the probability of outcome i is:

$$p(i) = \langle\psi|P_i|\psi\rangle$$

After measuring and getting result i, the system collapses to:

$$\frac{P_i|\psi\rangle}{\sqrt{p(i)}}$$

5. No-Cloning Theorem

Definition

The **No-Cloning Theorem** states that it is **impossible to create an exact copy** of an **arbitrary unknown quantum state**. That is, there is **no universal quantum operation** U such that:

$$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle \quad \forall |\psi\rangle$$

Proof Sketch

Assume such a universal cloning unitary U exists:

$$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle$$
$$U(|\phi\rangle \otimes |0\rangle) = |\phi\rangle \otimes |\phi\rangle$$

Take the inner product:

$$\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^2 \Rightarrow \langle\psi|\phi\rangle(1 - \langle\psi|\phi\rangle) = 0$$

Which only holds if $\langle\psi|\phi\rangle = 0$ or 1 — i.e., only orthogonal states can be cloned.

Implications

- **Quantum information is private**: An unknown quantum state cannot be duplicated, securing **quantum cryptographic protocols**.
- **No backup**: You cannot "save" a quantum state. Once it's gone (e.g., by measurement or decoherence), it's gone.
- **Restricts classical copying logic**: Many classical algorithms involve copying intermediate results. In quantum computing, you must instead **preserve entanglement** and use **unitary transformations**.

Use Cases

- **Quantum cryptography**: Prevents an eavesdropper from cloning a quantum key in protocols like BB84.
- **Quantum teleportation**: Teleportation transfers quantum state without copying — an original is destroyed when a copy is created at the destination.

6. Quantum Parallelism

Definition

Quantum parallelism refers to the ability of a quantum system to **evaluate a function on many inputs simultaneously** due to superposition.

Suppose we want to evaluate a function $f : \{0,1\}^n \to \{0,1\}$. On a quantum computer, we prepare:

$$\sum_{x \in \{0,1\}^n} |x\rangle \otimes |0\rangle$$

Applying the unitary $U_f : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$ yields:

$$\sum_x |x\rangle|f(x)\rangle$$

The function $f(x)$ is **evaluated on all 2^n inputs simultaneously**.

Important Clarification

- We **cannot directly read** all $f(x)$ values — measurement gives **one output**.
- **Interference must be engineered** to amplify correct answers and cancel incorrect ones — this is what algorithms like **Shor's** and **Grover's** exploit.

Implications

- **Exponential data processing**: One quantum operation can act on an exponentially large number of classical states.
- **No free lunch**: Quantum parallelism alone is not enough — we need **constructive interference** to extract useful information.
- **Algorithm design challenge**: You must build **circuit patterns** (unitary operations) that shape the final state distribution to concentrate amplitude on useful answers.

Examples in Algorithms

- **Deutsch–Jozsa Algorithm**: Uses parallelism to determine if a function is constant or balanced with a single query.
- **Simon's Problem**: Finds a secret string with exponentially fewer queries than classical algorithms.
- **Shor's Algorithm**: Applies quantum Fourier transform to periodic superpositions resulting from parallel function evaluation.

Implications in Quantum Computing

- **One-shot readout**: A quantum algorithm must structure interference and gate operations such that the correct output has **high probability** when measured.
- **No state peeking**: You **cannot inspect** a quantum state directly — you must design an algorithm to **extract information via interference**, then **collapse the state** via measurement.
- **Measurement affects future computation**: Once a qubit is measured, it is **irreversibly altered**. Further operations on that qubit act on a different state.
- **Basis matters**: Measurement can be performed in any orthonormal basis (via pre-rotation gates). Choosing the right **measurement basis** is often critical.

Use Cases

- **Quantum key distribution (QKD)**: Eavesdropping can be detected because **any measurement disturbs** the quantum state.
- **Quantum tomography**: Reconstructing quantum states requires repeated measurement on **many identically prepared states**, never just one.

Feature	Measurement Collapse	No-Cloning Theorem	Quantum Parallelism
Nature	Irreversible probabilistic process	Impossibility principle	Resource from superposition
Implication	Collapses quantum state	Cannot duplicate unknown states	Evaluate function on all inputs simultaneously
Benefit	Allows readout of final result	Secures quantum communication	Enables algorithmic speedup
Limitation	Destroys quantum state	No backup or copying of data	Need interference to extract output
Use Case	Output extraction, QKD	QKD, Teleportation, Secure protocols	Grover’s, Shor’s, Simon’s, DJ

Q. Is it possible to perform classical computation on quantum computer? - Explain schematically with proper illustration..

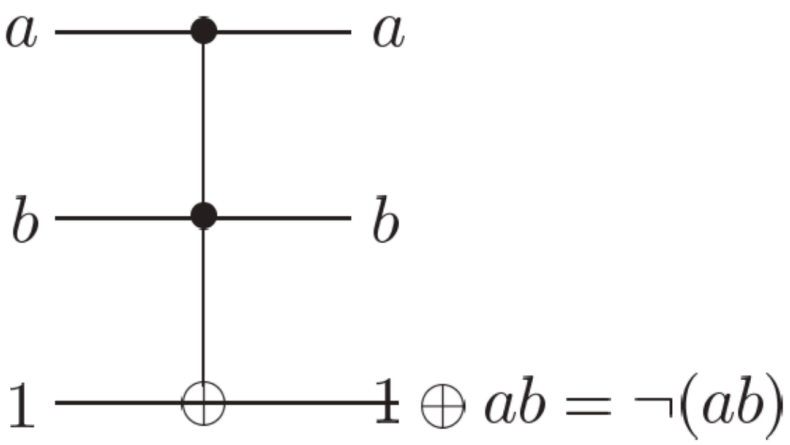
Unitary matrix CNOT gate explanation

- Basis for two qubit system $|00\rangle, |01\rangle, |10\rangle, |11\rangle$
- Transformations are
  $|00\rangle \rightarrow |00\rangle$,
  $|01\rangle \rightarrow |01\rangle$,
  $|10\rangle \rightarrow |11\rangle$,
  $|11\rangle \rightarrow |10\rangle$
- Hence CNOT can be expressed as
  $|00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11|$
- Another way to express it is $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$
- Vectors are
  $|00\rangle = (1000)^T; |01\rangle = (0100)^T; |10\rangle = (0010)^T; |11\rangle = (0001)^T$
- Hence the matrix for CNOT is $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Unitary matrix CCNOT - Toffoli gate

- Basis for three qubit system
  $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$
- Transformations only when first two bits are 1, third bit inverts.
  $|110\rangle \rightarrow |111\rangle$ and $|111\rangle \rightarrow |110\rangle$
- Hence CCNOT can be expressed as
  $(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I + |11\rangle\langle 11| \otimes X$
- This is the Toffoli gate

Classical computation - NAND using Toffoli gate



Non-deterministic Classical computation

- Classical computers that are able to generate random bits for computation.
- Prepare qubit in state $|0\rangle$ then pass it through Hadamard gate to produce $\frac{(|0\rangle+|1\rangle)}{\sqrt{2}}$
- Now if state is measured, outcome is $|0\rangle$ or $|1\rangle$ with probability 0.5 each.

Classical computation using Toffoli gate

- When both a and b are 1, c flips.
- Reversibility is ensured since $(a, b, c) \rightarrow (a, b, c \oplus ab) \rightarrow (a, b, c)$
- Now $c = 1$ implements NAND gate.
- Toffoli gate resembles 8×8 unitary matrix, hence it is valid quantum gate.
- Since NAND can be simulated, this is classical computation using quantum circuits.

Q. Software Development Facility for Quantum Algorithm Implementation: IBM Qiskit

1. Introduction to Qiskit

Qiskit (Quantum Information Science Kit) is an **open-source SDK** developed by IBM for working with quantum computers at multiple levels of abstraction. It enables developers, researchers, and students to:

- Write quantum algorithms
- Simulate them locally
- Execute them on real quantum devices via the IBM Quantum platform

Qiskit provides a **complete software stack** to support quantum application development, from circuit-level programming to high-level algorithm libraries.

2. Architecture and Components of Qiskit

Qiskit is modular and consists of **four primary elements**:

a. Qiskit Terra

- Core of Qiskit — provides basic tools for writing quantum circuits.
- Functions:
  - Circuit creation and manipulation
  - Compilation into low-level instructions
  - Optimization for specific quantum devices
- Supports quantum gates like X, H, CX, U, etc.
- Interface to **backends** (simulators and quantum hardware)

b. Qiskit Aer

- High-performance quantum **simulator**.
- Simulates realistic quantum noise, decoherence, and measurement errors.
- Useful for debugging and benchmarking quantum algorithms.

c. Qiskit Ignis

- Focused on **quantum error correction and noise characterization**.
- Tools for calibration, decoherence studies, and dynamical decoupling.

d. Qiskit Aqua (deprecated, migrated to other modules)

- Initially used for high-level quantum algorithms in chemistry, ML, optimization.
- Now migrated to:
  - **Qiskit Machine Learning**
  - **Qiskit Nature**
  - **Qiskit Optimization**
  - **Qiskit Finance**

3. Development Environment and Workflow

a. Language

- Qiskit is written in and used via **Python**.
- Compatible with Jupyter Notebooks, Python scripts, and interactive shells.

b. Setup

pip install qiskit

c. Example Development Workflow

Create a Quantum Circuit

```
from qiskit import QuantumCircuit
qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0, 1)
qc.measure([0, 1], [0, 1])
```

Simulate Locally

```
from qiskit import Aer, execute
simulator = Aer.get_backend('qasm_simulator')
job = execute(qc, simulator, shots=1000)
result = job.result()
print(result.get_counts(qc))
```

Run on Real Hardware

```
from qiskit import IBMQ
IBMQ.save_account('YOUR_API_KEY')
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_quito')
job = execute(qc, backend, shots=1000)
```

4. Visual Tools and Circuit Drawing

Qiskit provides tools for **visualization**:

- **Circuit Diagrams**:
  `qc.draw('mpl')  # or 'text'`
- **Bloch Sphere and State Vector** visualizations
- **Measurement Histograms**
- **Noise models and fidelity plots** in Aer and Ignis

5. Advanced Capabilities

a. Quantum Machine Learning

- With qiskit-machine-learning, build and train hybrid quantum-classical ML models.
- Example: Quantum Support Vector Machines, Variational Classifiers

b. Quantum Chemistry (Qiskit Nature)

- Simulates molecular structures using **VQE (Variational Quantum Eigensolver)**.
- Used in pharma and materials science.

c. Quantum Optimization

- Solve problems like **Max-Cut**, **TSP**, and **Portfolio Optimization** using **QAOA (Quantum Approximate Optimization Algorithm)**.

d. Quantum Finance

- Risk analysis, option pricing models using quantum amplitude estimation.

6. IBM Quantum Ecosystem Integration

- Use IBM Qiskit with IBM's **Quantum Experience Cloud Platform**.
- Provides:
  - Free access to real quantum computers
  - GUI for circuit building
  - Backend selection (simulators, real hardware)
  - Job monitoring tools

7. Educational and Research Support

- **Qiskit Textbook**: Comprehensive resource teaching quantum computing with Qiskit.
- **IBM Quantum Lab**: Hosted Jupyter Notebooks to run Qiskit code.
- **Qiskit Advocates and Community**: Global support and events for learners.

8. Limitations and Considerations

- **Quantum hardware access** is shared, subject to queue delays.
- **Noise and decoherence** are non-negligible on current NISQ-era devices.
- **Limited qubit count** (typically < 50 for most public devices).
- Algorithms must be **error-resilient** and **optimized** for real use.

9. Summary Table

Feature	Qiskit Support
Language	Python
Simulation	Aer
Hardware Integration	IBM Quantum Devices
High-Level Algorithms	ML, Optimization, Chemistry, Finance
Visualization	Circuit, Bloch, Histograms
Education	Qiskit Textbook, Community

10. Conclusion

Qiskit offers a **comprehensive software development ecosystem** for quantum algorithm design, testing, simulation, and deployment on real quantum devices. It's ideal for researchers, developers, and students looking to explore quantum computing from theory to practical implementation.

Q. Quantum Key Distribution in Cryptography

The BB84 QKD protocol

1: Alice chooses $(4+\delta)n$ random data bits.
2: Alice chooses a random $(4+\delta)n$-bit string b. She encodes each data bit as $\{|0\rangle, |1\rangle\}$ if the corresponding bit of b is 0 or $\{|+\rangle, |-\rangle\}$ if b is 1.
3: Alice sends the resulting state to Bob.
4: Bob receives the $(4+\delta)n$ qubits, announces this fact, and measures each qubit in the X or Z basis at random.
5: Alice announces b.
6: Alice and Bob discard any bits where Bob measured a different basis than Alice prepared. With high probability, there are at least $2n$ bits left (if not, abort the protocol). They keep $2n$ bits.
7: Alice selects a subset of n bits that will to serve as a check on Eve's interference, and tells Bob which bits she selected.
8: Alice and Bob announce and compare the values of the n check bits. If more than an acceptable number disagree, they abort the protocol.
9: Alice and Bob perform information reconciliation and privacy amplification on the remaining n bits to obtain m shared key bits.

Eve's attack process

1 Eve intercepts every qubit Alice sends to Bob

2 Eve is unaware of the bases and chooses either:
- X : Measures $|+\rangle$ or $|-\rangle$
- Z : Measures $|0\rangle$ or $|1\rangle$

Correct Guess

- If Alice sent $|0\rangle$ (Z) and Eve measures in **Z**:
  - **Result**: Gets $|0\rangle$ → **No disturbance**.
- If Alice sent $|+\rangle$ (X) and Eve measures in **X**:
  - **Result**: Gets $|+\rangle$ → **No disturbance**.

Wrong Guess

- If Alice sent $|0\rangle$ (Z) but Eve measures in **X**:
  - $|0\rangle = (|+\rangle + |-\rangle)/\sqrt{2}$ → Eve gets $|+\rangle$ or $|-\rangle$ **randomly** (50% each).
- If Alice sent $|+\rangle$ (X) but Eve measures in **Z**:
  - $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ → Eve gets $|0\rangle$ or $|1\rangle$ **randomly**.

3 Re-encoding and forwarding

Eve **re-encodes** the qubit based on her measurement:

- If she measured $|0\rangle$ in Z-basis → Sends $|0\rangle$ to Bob.
- If she measured $|+\rangle$ in X-basis → Sends $|+\rangle$ to Bob.

Theoretical error calculation

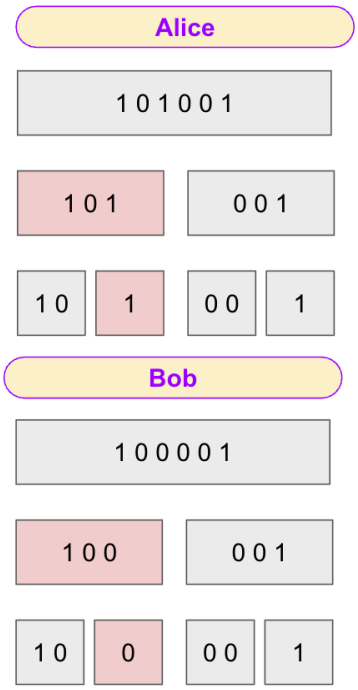
When Errors Occur:

Only when **these conditions** are met:

1. Eve picked the **wrong basis** (50% chance).
2. Her measurement **disturbed the state** (50% chance).

P = 0.5 × 0.5 = 0.25

1. Cascade error correction



Alice's Key: [1, 0, 1, 0, 0, 1]
Bob's Key: [1, 0, 0, 0, 0, 1] (Error at position 2)

1. **Pass 1 (Block Size = 3)**:
   - Block 1: Alice [1,0,1] (Parity=0), Bob [1,0,0] (Parity=1) → **Mismatch**.
     - Split into [1,0] and [0]:
       - Left half: Alice parity=1, Bob parity=1 → No error here.
       - Right half: Alice bit=1, Bob bit=0 → **Error found**.
     - Flip Bob's bit at position 2: [1,0,1,0,0,1].
2. **Pass 2 (Block Size = 1)**:
   - Check individual bits. No further errors detected.

Final Corrected Key: [1, 0, 1, 0, 0, 1] (Matches Alice's key).

2. Privacy Amplification

1. Choosing a Hash Function:

- Use a **random binary matrix** (Toeplitz) of size (final_length × input_length). Example for input 6 bits x 2 bits:

2. Multiply and XOR:

- For each row of the matrix:
  - Multiply each bit with the corresponding key bit.
  - Sum (XOR) all results to get 1 output bit.

3. Example:

- **Input Key**: [1, 0, 1, 0, 0, 1]

$$\text{Matrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Row 1:$1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 1$
  Row 2:$0 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$

- **Output Key**: [1, 1] (2 bits).