Lecture 5: Computation using circuits

Rajat Mittal

IIT Kanpur

These notes will focus on the circuit model of computation. Even though we have seen a few quantum circuits, we will introduce these concepts formally (starting with classical circuits). Finally, we will take a few examples of quantum circuits to become more familiar with them.

1 Classical circuits

To understand asymptotic speedups for natural problems, we need to understand the model of computation. We will assume that the reader is familiar with the very basic concepts in computation, like asymptotic notation and importance of Turing machine. If not, please look at the third chapter of Nielsen and Chuang [4] or the book by Arora and Barak [1].

Classically, computation is generally described by what can or cannot be performed on a Turing machine. Deutsch gave the concept of quantum Turing machine in 1985. Though, researchers in quantum computing prefer another equivalent model (to Turing machines) known as circuit model. We will introduce circuit model of quantum computation in this chapter, i.e., computation through quantum circuits. Let us look at classical circuits first.

We again assume that the reader is familiar with boolean circuits used to compute functions (circuits constructed from AND, OR and NOT gates). A circuit computes some function $f : \{0,1\}^n \to \{0,1\}^m$. In general, a circuit is a collection of gates connected by wires to transfer the information. A gate is a function from $\{0,1\}^k$ to $\{0,1\}^l$.

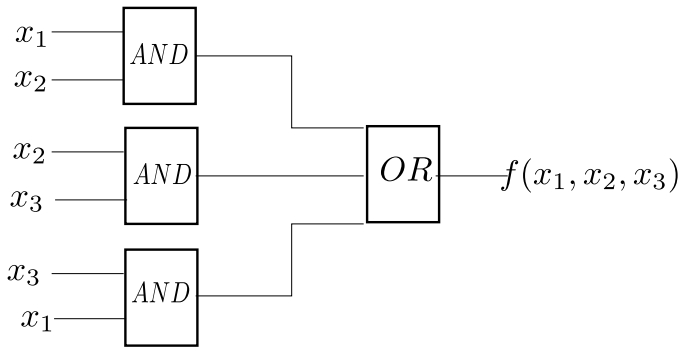


Fig. 1. A circuit computing a simple function. This one processes input from left to right, you might have seen one where it might go from bottom to top.

Exercise 1. Read about $AND, OR, Parity, Majority$ functions on n bits.

Exercise 2. What function does the above circuit compute?

Note 1. If there is a cycle in the circuit then the corresponding wire can have conflicting values. So, we assume that there are no cycles in the circuit. In other words, we can think of it as a directed acyclic graph. We will draw the circuits from left to right, there won't be any wires running from right to left to avoid cycles in the circuit.

Exercise 3. Draw a circuit for NAND on three bits using AND, OR (on two bits) and NOT (on one bit).

In case of classical circuits, we allow an operation called *FANOUT*. This FANOUT operation has one input and two output; it makes two copies of the given input. You can easily show that copying is *not* a unitary operation, and hence *not* possible for quantum circuits.

What functions should we allow as part of a circuit? Clearly, if we allow arbitrary functions in the circuit then we can calculate arbitrary functions. Practically, it makes sense to assume some standard gate set and ask what functions can be computed from that gate set. As you know, one of the famous gate set is AND, OR and NOT.

Using the truth table, you can show that any function can be computed using the gate set of AND, OR and NOT (assignment). AND, OR on multiple bits can be broken down into AND and OR on two bits. Hence, we can have circuit for any function using NOT gate (on one bit) and AND, OR (on two bits). Even the OR gate can be computed with other two gates (De Morgan's rule), so AND and NOT are enough to perform any computation. Such a set of gates, which can compute any function, is called a *universal* gate set. The complexity of a circuit is counted in terms of the gates used from a particular gate set.

Exercise 4. Show that the gate set $\{NAND\}$ is universal.

Exercise 5. The complexity can differ if we use different universal gate sets, why does it not mattter?

We also allow creation of *ancilla* bits, these are extra bits required to do the calculation (workspace) and generally assumed to start in the state 0 ($|0\rangle$ for quantum circuits) of the required dimension. You can think of them as temporary variables in usual computation. Ancilla bits are important in quantum circuits specially because we need the computation to be reversible. Though, they need to be handled carefully, because they could get entangled with the output. More about that when we learn about quantum circuits.

1.1 Uniformity

We mentioned that the Turing machine model is equivalent to the circuit model. What does that mean? At a first look, Turing machines compute languages and circuits compute functions.

Exercise 6. What is the difference?

In case of languages, the input size is not fixed. For a language, the set of accepted strings is some subset of $\{0,1\}^*$. Alternatively, in case of circuits, the input size is fixed and hence we cannot talk about arbitrary size of strings.

One possible resolution is to have circuits for every length of input. That means, for every n there is a circuit C_n which accepts all strings of length n (or say less than n) and nothing else. Then, we can say that the circuit family $\{C_n\}$ computes the language.

Exercise 7. What function is the circuit computing in terms of the language?

This definition turns out to be very strong. Such family of circuits can potentially calculate by having the output hard coded for particular n's.

To take an example, let S be the set of strings s for which s_i is 1 iff i-th Turing machine (in some list) halts. Can we compute the language represented by set S? This is an undecidable problem for a Turing machine. Though, there is just one string of a particular length and a very small circuit for a particular length exists. In other words, the above definition of circuits will be able to compute an undecidable language!

The previous example shows that we should not be allowed to have very different circuits for different size of strings. This restriction, to avoid having arbitrary circuits for a particular size, is known as *uniformity*.

A circuit family is called uniform, if there exists a Turing machine which can generate circuit C_n, given an input n using bounded resources. Here, bound on resources is determined by the complexity class of interest. This circuit C_n should accept only and all the strings in the language of length less than n. Then, we say that this uniform circuit family $\{C_n\}$ computes the language.

Turing machine model and uniform circuit family is equivalent in the sense that whatever language can be computed in one model can be computed in another model too. This equivalence holds for quantum Turing machine and circuits too.

We haven't looked at the matter of efficiency. In general, an algorithm is considered *efficient* on a Turing machine if the output is determined in polynomial time. For the circuit family $\{C_n\}$ to be efficient,

- the Turing machine outputting circuits C_n (uniformity) should do so in polynomial time,
- the circuits should be of polynomial size in n (number of standard gates).

The first condition is mostly implicit in our algorithms; this is because circuits for different sizes are similar in our constructions. While designing and reading about algorithms, we focus on the second constraint for efficiency.

Let us see how quantum circuits are similar to classical circuits (and different from classical circuits).

2 Quantum circuits

Before defining quantum circuits, let us take a look at the basic entities of quantum circuit introduced earlier.

Qubits: The fundamental unit of computation is a bit on a classical computer. A bit can be in two states, either in 0 or in 1. The analog of a bit on a quantum computer is called a *qubit*. A qubit can be in the state $|0\rangle$ or $|1\rangle$ (corresponding to the classical states) or even in a superposition of these states. Specifically, the state of a qubit is

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

Where α, β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

Exercise 8. Why do we have a mathematical equation for a qubit. Shouldn't it be described by a physical object?

The states $|0\rangle$ and $|1\rangle$ are called the basis states and the above equation states: any linear combination of the basis states with *length* 1 is a valid state. While talking about the length of a state, we view it as a vector. Remember that this corresponds to the first postulate of quantum mechanics.

Importantly, it is not possible to figure out α, β given a physical qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. We are only allowed to *measure* the qubit. Let us take an example of a measurement. If we measure in the standard basis, with basis states $|0\rangle$ and $|1\rangle$, then with $|\alpha|^2$ probability we will observe state $|0\rangle$ and with $|\beta|^2$ probability we will get state $|1\rangle$. This provides more justification as to why the norm of a state should be 1.

Like in the case of classical computing, we use multiple qubits to store the data in a quantum computer. What are the possible states of two qubits?

The basis states should be $|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle$ and $|1\rangle|1\rangle$. We will identify the state $|0\rangle|0\rangle$ with the state $|00\rangle$ and similarly for other basis states. As before, we will say that any linear combination of these states will be a valid state.

$$|\psi\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle.$$

The length of this state should be 1, so $\sum_i |\alpha_i|^2 = 1$.

Operations: We have defined states of different parts of the quantum computer. The next step would be, what kind of operations can we perform on this computer? In other words, we are searching for the analog of NOT/AND/NAND/OR gates on a quantum computer.

For simplicity, we will start with single qubit gates, gates which act on only one qubit. We can define the quantum NOT gate which takes $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$.

Exercise 9. Is this description sufficient?

Since our state space is big, any linear combination of $|0\rangle$ and $|1\rangle$, we need to define the action on the complete state space. Using the postulates of quantum mechanics, this is done linearly. So for a quantum NOT gate, $\alpha|0\rangle + \beta|1\rangle$ goes to $\alpha|1\rangle + \beta|0\rangle$.

Note 2. This means that to specify a quantum gate, we only need to mention its action on the basis elements.

Exercise 10. What are the other bases for the state of a quantum bit?

Linearity implies that any quantum gate on a single qubit can be written as a 2×2 matrix (if there are n qubits, then $2^n \times 2^n$ matrix). Since any state has unit norm, these matrices should be *unitary*. The matrix representation of quantum NOT gate is pretty simple and it is equal to Pauli X operator.

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

There are infinite possible single qubit gates (unitary 2×2 matrix) as compared to finitely many classical gates. Can we realize each of them on a quantum computer? Should we build a quantum computer which has all these gates?

Thankfully, It turns out that there are elementary gates which can be composed to perform any possible quantum gate approximately (even for the case of multiple qubit gates). Whether this can be done efficiently, that is a question of great interest.

Few other interesting gates we saw were *Hadamard* and *controlled NOT (CNOT)* gate.

Exercise 11. What do you think CCNOT gate should be? Write the matrix representation of CCNOT gate.

Definition of a quantum circuit We saw that classical circuits are made up of classical gates and wires which carried bits. Quantum circuits are analogous; they have quantum circuits with wires which carry quantum states or qubits. Through the postulates, we know that the quantum gates are unitary. So, quantum circuits will have unitary gates and measurements.

For example, look at the circuit given in Fig. 2. Notice that the wires run from left to right and there is no cycle. There are two inputs (x_1 and x_2) and two ancilla bits (y_1 and y_2). We have applied multiple gates, for example, Hadamard on the first qubit and CNOT's. A multi qubit unitary (whose description might be given separately, think of a subroutine) is applied on all 4 qubits. Finally, the output is obtained on the third qubit. The *meter* symbol represents measurement on the third qubit.

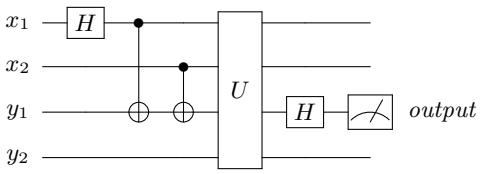


Fig. 2. An example of a quantum circuit

In general, it is sufficient to give the action of a circuit/gate on a basis. Choosing standard basis, it is enough to specify the action on classical bits. The action on other states can be inferred by linearity.

Exercise 12. What will be the action of *CNOT* gate on state $\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle$ (first qubit is the control qubit)?

You should convince yourself that the answer is $\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|11\rangle + \alpha_4|10\rangle$.

3 Examples of a quantum circuit

We start by taking a few examples of a quantum circuit.

Swapping two quantum bits: We are given two qubits and we want to swap their states (obviously, we are not allowed to physically change their places).

Exercise 13. How will you do it classically?

The easiest way, classically, is to have a temporary variable and then swap the states. The problem is, we are not allowed to copy states in quantum computation. There is a trick to achieve the swap classically, without using copy. Given that x and y are two bits, notice the following set of operations.

$$x = x \oplus y$$
$$y = y \oplus x$$
$$x = x \oplus y$$

Convince yourself that the following operation swaps the two bits. Notice that the procedure above requires x (and y) to be bits, so that $\oplus$ operation is defined. We want to achieve the same thing quantumly.

Exercise 14. What is the quantum equivalent of $\oplus$ gate?

The quantum equivalent of classical XOR gate is the quantum CNOT gate. So, our circuit will instead consist of three CNOT's.

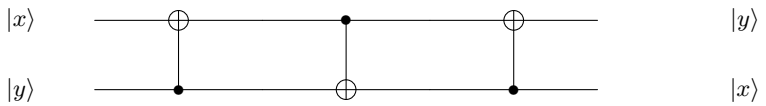


Fig. 3. Circuit for swapping two quantum bits

Again, check that the circuit in Fig. 7 works for all classical values of x and y. The linearity of quantum mechanics ensures that this circuit works for all states.

Exercise 15. Take two arbitrary quantum states and show that Fig. 7 performs swap through direct calculation.

3.1 Addition of two binary strings:

Let us try to construct a more involved quantum circuit, for the addition of two binary strings. Notice that the problem is classical, we just need a quantum circuit for it.

Exercise 16. Make sure that you can add two numbers given in binary representation (WITHOUT converting them into decimal representation). What is the sum of 10110 and 10011? You should get 101001.

Like the case of decimal representation, the trick is to add the numbers coordinate-wise and move the carry forward. For example, let a=1011 and b=1101, then

$$\begin{array}{rr} carry & 1\,1\,1\,1 \\ \hline a = & 1\,0\,1\,1 \\ b = & 1\,1\,0\,1 \\ \hline ans = 1\,1\,0\,0\,0\,0 & \end{array}$$

Looking at this procedure, one of the main part (subroutine) seems to be the addition of two bits. There will be two output, one bit goes to the answer and other is the carry. Such a circuit is called a *half adder*. To make it precise, here is the output on all 4 possible classical inputs (the most significant bit is the carry).

$$\begin{aligned} 00 &\to 00 \\ 01 &\to 01 \\ 10 &\to 01 \\ 11 &\to 10 \end{aligned}$$

It is not very difficult to extend the half adder circuit to addition of two full binary strings.

Exercise 17. Can you construct a circuit to do addition of two 2-bit strings, given half adder as a blackbox? Can you generalize it to n-bit strings?

Try to finish the circuit by yourself. If not, look at Fig. 5. For the quantum case, answer needs to be given on two ancilla qubits. A quantum half adder will have two inputs and two ancilla bits, for the sake of clarity, Fig. 5 only shows two inputs bits on the left hand side and two ancilla bits on the right hand side.

Our remaining task in this section is to figure out a circuit for half adder. We will make it *bit by bit*. For the first bit (one that goes to the answer), it is 1 if and only if exactly one of the two input bits is 1.

Exercise 18. Do you know the name of this gate in the classical world?

It is called an *XOR* operation. What is the quantum equivalent? You have seen that gate before, it is the *CNOT* gate (with *XOR* computed in date qubit). This would have worked if we wanted the answer qubit as one of the input qubits. What if we want to compute answer (and carry) on an ancilla qubit starting in $|0\rangle$ state?

Exercise 19. How can you compute the answer in an ancilla qubit?

Since the ancilla is set to 0, you can take *XOR* with the first qubit and then the second qubit. In other words, use two *CNOT* gates.

We are left with the task of computing the carry bit. It is 1 if and only if both the qubits are 1. This is known as *AND* gate classically.

Exercise 20. Use Toffoli (CCNOT) gate to create a circuit for *AND* gate.

The final circuit will look like the following figure.

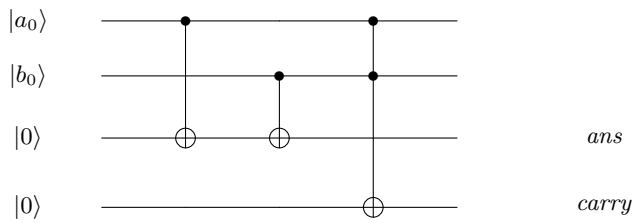


Fig. 4. Circuit for half adder

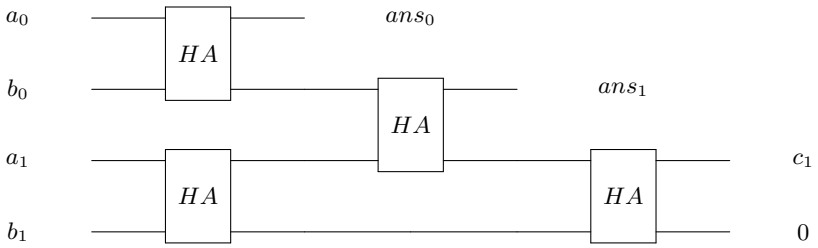


Fig. 5. Circuit for addition using half adder (HA). The top bit of HA is the answer bit and the bottom one is the carry.

3.2 Elitzur-Vaidman as a quantum ciruit

We saw earlier that the beam splitters can be thought of as Hadamard gates and the bomb itself can be thought of as a measurement (Figure 6).

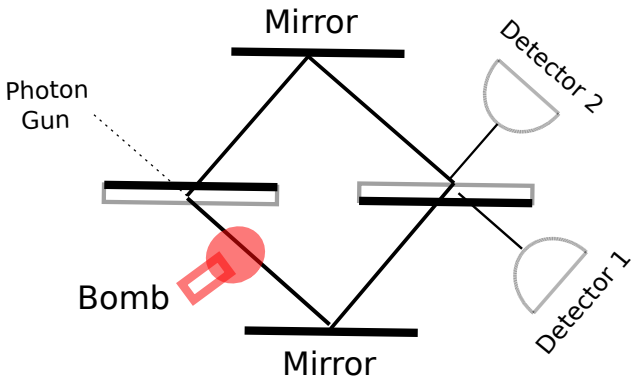
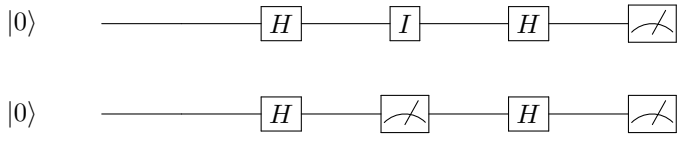
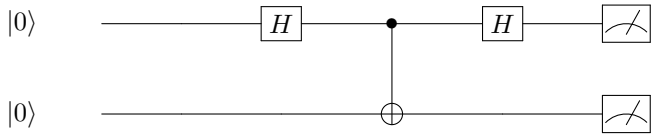


Fig. 6. Elitzur Vaidman bomb tester: how to test a bomb without exploding it.

In other words the circuit looks like (for no bomb and with bomb respectively) this.



From the principle of deferred measurement, we can move the intermediate measurement in the end for the bomb case.



Notice that the CNOT and measurement on the second register will be replaced by identity if the bomb doesn't work.

You can easily verify the properties of Elitzur-Vaidman bomb testing. If the controlled operation is Identity, we will always obtain $|0\rangle$ (H is its own inverse). If the controlled operation is NOT, then there are two cases depending upon the state of the second qubit.

If second qubit is in $|1\rangle$, that means CNOT was applied and the bomb has blown. On the other hand if second qubit is $|0\rangle$, in the first qubit we get $|0\rangle$ and $|1\rangle$ with equal probability. If we get $|1\rangle$ in the first register, then we know that bomb is real and it has *NOT* exploded.

Exercise 21. Write the quantum state explicitly and check the above assertions.

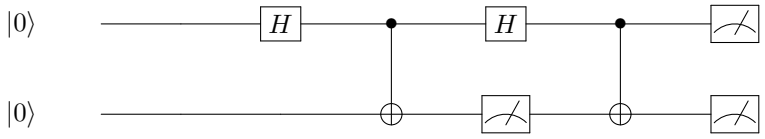
The probability of not blowing the bomb and getting the correct answer is 1/4, you might not be happy about it.

There is a smart way to make this probability arbitrarily close to 1, without blowing the bomb. As a first step let us make the probability of *false negative*, saying that the bomb is dud when it is real to be zero. In a sense, we will move this probability to blowing the bomb.

The main observation is that if the measurement on the second register outputs 0, the state in the first register becomes $|0\rangle$. If we had measured it right here, we would have correctly said that bomb is real. Unfortunately, we need to apply a Hadamard due to *dud* case, which leads to a 1/4 probability that we answer dud when the bomb was infact real.

Exercise 22. Can you think of an extra operator, after the 2nd Hadamard, which will make this false negative probabiity go to exploding the bomb?

The idea is very simple, we place the bomb once more in the path. You can easily check that when the second register is in $|1\rangle$ (the false negative case), it will lead to exploding the bomb. If the bomb is dud, this does not change anything. In other words, our circuit becomes,

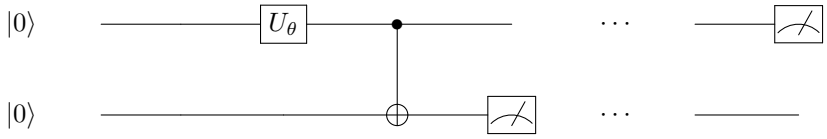


Remember that the CNOT and measurement (on the second register) will be replaced by controlled Identity (which is Identity) in case the bomb is dud. For this new circuit, we always output dud if the bomb is a dud. If the bomb is real, it explodes with probability 3/4 and we answer it is real (without exploding it) with probability 1/4.

Our task is very clear now, decrease the probabiity of bomb exploding. When is the bomb exploding?

In this case, after the first Hadamard the state becomes $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and the $|1\rangle$ part destroys the bomb (with half the probability). The idea is to use operator U_θ which takes $|0\rangle$ to $\cos(\theta)|0\rangle + \sin(\theta)|1\rangle$ with a very small θ, that way the bomb goes off with very small probability (bomb going off means we measure the second register and got $|1\rangle$).

This is going to be the first part of the circuit if the bomb is real (uptil the first measurement of the second qubit). (If the bomb is dud, then CNOT and measurement of second register will be removed.)

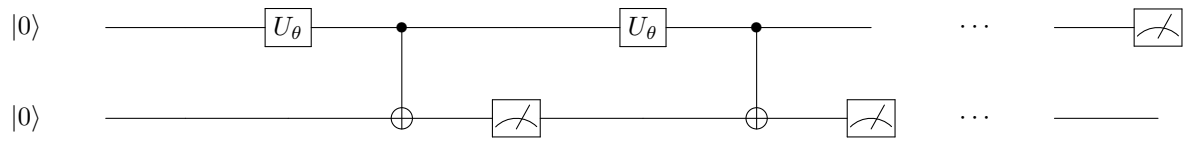


Note 3. We will learn how to implement U_θ in the next section.

Remember that if the bomb is real and does not explode then we measure $|0\rangle$ in the second register, that means the first register will get into state $|0\rangle$ again.

We will keep repeating this process until the bomb explodes or number of repetition become $k = \frac{\pi}{2\theta}$.

Exercise 23. Check that this implies that the circuit will output 1 if the bomb is dud with probability 1.



The probability of bomb exploding needs to be small. Intuitively, this will happen because angle moves with amount θ $\sin(\theta)$, but the probability of exploding is only $\sin^2(\theta)$. Let us analyze all the cases in detail.

In case we don't have a bomb, the repeated rotation takes the first register to state $|1\rangle$ and then we can declare the bomb to be dud.

In case we have the bomb, we might measure 0 in the first register (the first register gets reset every time to $|0\rangle$). Then we can declare that the bomb is real and has not exploded. The troublesome case is that the bomb is real and it explodes. The total probability of exploding is bounded by $k\sin^2(\theta)$. Since $\sin(\theta) \approx 1/k$, this probability goes to 0 with smaller θ.

Exercise 24. Can it happen that we measure $|1\rangle$ and the bomb is real?

The main analysis works because after k repetitions, probability of exploding increase much more slowly than the angle of rotation ($k\theta^2$ as compared to $k\theta$). So we can *quickly* go to $|1\rangle$ state in case of dud, keeping probability of exploding small.

Note 4. Similar idea will be used in Grover search.

Please refer to the following wiki article for details [5].

4 Extra reading: quantum teleportation

This section is motivated by the reference [2]. They have a nice explanation of swapping and teleportation.

Let us look at another application of the fourth postulate, specifically entanglement, called *quantum teleportation*. You might have noticed that a qubit is much harder to describe than a classical bit, i.e., a qubit takes two complex numbers to describe but a bit just needs a binary value.

Suppose two parties, Alice and Bob, want to exchange a qubit. Unfortunately, they only have access to a classical communication channel and not a quantum channel. It seems that it should not be possible to exchange a qubit as that requires infinite bits of information to describe.

In other words, to transfer a quantum bit, we should have some form of quantum communication. Surprisingly, using entanglement and classical communication, we can transfer a qubit without using any form of quantum communication!! That means, Alice and Bob need access to a classical channel and share a pair of entangled states. This pair does not (should not too) depend upon the qubit to be transferred. This protocol is called *quantum teleportation*.

Looking at it from a different perspective, suppose Alice and Bob have quantum computers but *don't* have a channel which can transfer quantum bits. Using entanglement, we can transfer quantum bits from one party to another with the help of only classical communication. This protocol is called the quantum teleportation protocol.

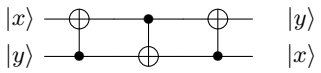


Fig. 7. Circuit for swapping two quantum bits

4.1 Protocol for teleporting a quantum bit:

Notice that teleportation is like swapping, it is a *one-sided* swap. Remember the circuit for swap.

For teleportation, we need to transfer a qubit from Alice to Bob. The problem is, one qubit is with Alice and one with Bob. So, it is not possible to apply CNOT on these two qubits.

Exercise 25. Show that the first CNOT is redundant if we start with $|0\rangle$ state on Bob's part.

So, we want to get the following circuit.

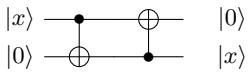


Fig. 8. Simplified circuit when Bob's qubit is $|0\rangle$

The idea is to use an intermediate qubit and use that to transfer the state of Alice to state of Bob. This will only require CNOT's between Alice's qubit and intermediate qubit, or between intermediate qubit and Bob's qubit (this trick is classical).

Exercise 26. Find the classical circuit which takes $x, y, z \to x, y, z \oplus x$ using XOR operations where no gate can be used between x and z.

Hint: Use 4 XOR gates

The question remains, who will have the intermediate qubit. It is kept with Alice, and CNOTs between intermediate and Bob's qubit is simulated using entanglement and classical communication. The circuit becomes

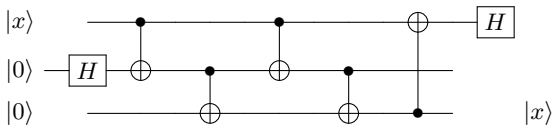


Fig. 9. Introducing the intermediate qubit

Notice that a Hadamard gate at the beginning of second qubit (the initial state of second qubit is not important) and at the end of the third qubit does not change anything. The last two operations can be changed by switching from CNOT to CZ gate. We switch to CZ because control and target can be switched for CZ without affecting anything. You will show in the assignment that CNOT can be changed to CZ by moving Hadamard.

Please convince yourself that the circuit below is equivalent.

Once again the first CNOT can be switched (using Hadamard) and then CZ can be removed (control is $|0\rangle$).

Note the last two gates have only control qubits on Alice's part, they can be simulated using measurement and then sending the measurement results to Bob (by Alice). Also, The first Hadamard gate and the first gate between Alice's intermediate qubit and Bob's qubit is equivalent to having entangled qubits (it does not depend upon the state to be transferred). The full protocol is given below.

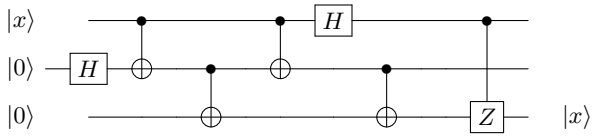


Fig. 10. Switching between CNOT and CZ

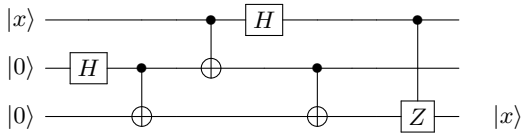


Fig. 11. Switching between CNOT and CZ again

As mentioned before, this protocol requires the use of entanglement. Alice and Bob can meet before and keep one part (qubit) of the Bell state with each of them. Suppose, Alice wants to transfer state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to Bob. Notice that we have chosen a completely arbitrary state to transfer to Bob.

Suppose the state Alice wants to transfer, $|\psi\rangle$, is the first qubit and her part of Bell state is the second qubit. Alice applies CNOT gate to these two qubits. Remember, CNOT gate is a 2-qubit gate, which applies NOT gate to the second qubit if and only if the first qubit is in state $|1\rangle$.

Exercise 27. Show that CNOT is unitary operator.

Then she applies Hadamard gate to her first qubit.

Exercise 28. What is the state of three qubits now?

It can be shown that the resulting state is,

$$\frac{1}{2}\left(|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)\right).$$

Now Alice measures her two qubits and sends them to Bob.

Exercise 29. Convince yourself that Bob can recover $|\psi\rangle$ using Pauli operators.

This completes the quantum teleportation. Alice is able to transfer one quantum bit using two classical bits of communication.

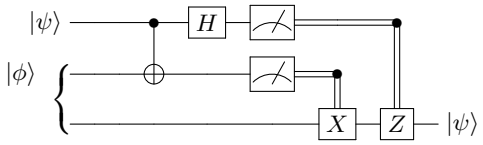


Fig. 12. Final protocol (here $|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is the Bell state).

Exercise 30. We said that we can transfer and not copy the quantum bit. Why?

In quantum computing we can't copy qubits, this is known as *no-cloning* theorem.

5 Properties of quantum circuits

5.1 Gates of a quantum circuit

The main difference is the kinds of gates applied in a quantum circuit, remember that they have to be unitary. The simplest unitary gates are single qubit gates. We have already looked at Pauli matrices (X, Y, Z) and Hadamard gate.

Exercise 31. Do you remember the definition of these gates?

It is sometimes easier to remember gates by their action instead of their matrix representation. The action needs to be specified only on the basis states.

Pauli X is like a NOT gate, it interchanges states $|0\rangle$ and $|1\rangle$. Pauli Z puts a phase of -1 if the state is $|1\rangle$ and does nothing on state $|0\rangle$. Pauli Y can be obtained by the relation $Y = iXZ$. The Hadamard gate moves $|0\rangle$ to $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ states.

Another important gate is the phase gate, which puts a phase of i if the qubit is $|1\rangle$ (it is the square root of Z gate). The unitary matrix S for the phase gate is given by,

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

Another important gate is $\frac{\pi}{8}$ gate. It is the square root of phase gate.

Exercise 32. Show that the following gate is a square root of the phase gate.

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$$

Exercise 33. Suppose A is the Hermitian matrix and x is a real number, s.t., $A^2 = I$. Show that $e^{iAx} = cos(x)I + isin(x)A$.

Exercise 34. Calculate $R_X(\theta) = e^{-iX\theta/2}, R_Y(\theta) = e^{-iY\theta/2}, R_Z(\theta) = e^{-iZ\theta/2}$. It can be shown [4] that any single qubit unitary operation can be decomposed in terms of these rotations and a global phase shift.

What about gates on more than one qubit? We need an important class of operators called controlled operators. We have already seen $CNOT$ gate. In terms of computational states, the action of $CNOT$ can be written as,

$$|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus x\rangle.$$

It turns out that controlled version of any single qubit gate (e.g. CZ gate) can be implemented using $CNOT$ and other single qubit gates. Actually, any unitary can be performed given $CNOT$ and all single qubit gates [4].

We can also define controlled operations where we have multiple control qubits. The Toffoli gate is an important example with two control qubits. It is also called $CCNOT$ gate. It applies an X gate (NOT) iff both control qubits are set to $|1\rangle$.

Exercise 35. Write the matrix representation for Toffoli gate.

Another important part of the quantum circuits is measurement. We state two properties of measurement without proof, interested readers can refer to Nielsen and Chuang [4].

1. In general, there can be measurements in between and the classical information can be used to control further elements in the later part of the circuit. Even if we replace the classically controlled operations by quantumly controlled operators, the final output is the same. To summarize, we can assume that the measurements are done at the end of the circuit.

2. We can assume that all un-terminated wires at the end are measured. It does not affect the measurement statistics of the remaining system.

Like the case of classical circuits, we know of combinations of gates which are universal for quantum circuits (though the meaning of universality is slightly different). For our purpose it is enough to assume that we can use all the standard gates mentioned till now in our circuits without worrying about the universality. Also, it is possible to simulate any classical circuit using a quantum circuit (without much change in size). So, we can assume access to not only standard quantum gates but also any classical circuit (with equivalent cost). There is some more information in the two subsections (Section 5.2 and Section 5.3) below, interested reader can go through them (and Nielsen and Chuang's book [4]) for more details.

5.2 Universality

We saw that *AND*, *OR* and *NOT* gates were universal for classical circuits. Can we come up with universal gate sets for quantum circuits? It was mentioned that *CNOT* and all single qubit unitary can perform any unitary operation. Though, this does not seem to be a practical choice (because of the size of this gate set).

A universal set of gates G in quantum computation are those which can approximate any unitary operator with arbitrary accuracy. In other words, for any unitary U and for arbitrary accuracy, there is a circuit with gates from G implementing U with that accuracy.

It turns out that the set of Hadamard, *CNOT* and $\frac{\pi}{8}$ gates is universal in this sense [4]. Again, it does not tell us about the size of the circuit needed to compute U. We can only use those U's, which can be implemented using only *polynomial* number of implementations of *basic* gates (gates from universal set of families).

This brings up another question. There are multiple families of universal gate sets for quantum computation. What if some operation can be performed using polynomial applications of one gate set but not with another?

Exercise 36. Why is this not a problem in classical case?

Solovay-Kitaev theorem tells us: Given a universal gate set G, any 1 or 2 qubit unitary can be performed with accuracy ϵ using only *poly-log*$(\frac{1}{\epsilon})$ number of gates from G. Intuitively, any 1 or 2 qubit unitary can be simulated with constant (assuming accuracy to be constant) number of gates. So, it doesn't matter which set of universal gates we pick.

To summarize, we have a large class of gates at our disposal.

- All standard gates discussed till now. This includes Pauli gates (X,Y,Z), Hadamard, CNOT, CCNOT.
- All classical circuits with the same complexity up to a constant.
- Any 1 qubit gate or controlled 2 qubit gate can be implemented approximately (Solovay-Kitaev), this can be generalized to any constant qubit gate.
- Any subroutine given to us with the required complexity.

5.3 Extra reading: Classical circuit simulation

We have seen classical circuits as well as quantum circuits. It was mentioned before that anything which can be done on a classical computer can be simulated on a quantum computer. At a glance, it seems difficult; especially since *AND*, *OR* and *NOT* are irreversible, but quantum gates are constrained to be reversible.

It turns out that we can have reversible versions of *AND*, *OR* and *NOT* gates constructed from Toffoli gate. Just a reminder, a Toffoli gate is a *CCNOT* gate with the following action on basis elements,

$$a, b, c \rightarrow a, b, c \oplus ab.$$

Exercise 37. Show that the Toffoli gate is reversible.

You can easily show that the Toffoli gate can be used to simulate NAND and FANOUT gate. For example, inputting $a, b, 1$ will simulate the NAND gate. This is not enough, we need to simulate FANOUT gate too !

Exercise 38. Show that you can simulate FANOUT gate using Toffoli gate. Why is this not against *no cloning* theorem?

Hence, any arbitrary classical circuit can be simulated by an equivalent reversible circuit. A circuit from Toffoli gates can be thought of as a quantum circuit, by extending its action on all states using linearity. To conclude, any classical circuit can be simulated by a quantum circuit of equivalent circuit size (up to polynomial factors).

For the randomized computation, we also need to produce random bits. This can be done by applying Hadamard on $|0\rangle$ and then measuring the qubit in the standard basis. This shows that any randomized computation can also be efficiently simulated by a quantum circuit. A very good exposition of simulation of classical circuits by quantum circuits can be found in [3, Chapter 3].

Suppose, a classical black-box outputs $f(x)$ on an input x. This means, at the time of implementation of the black-box, there will be some classical circuit which will compute $f(x)$ using x. This circuit can be converted into a reversible circuit which acts as $(x, y) \to (x, y \oplus f(x))$. Hence, the quantum version of the blackbox will be the circuit that acts similarly on the basis states, and acts linearly on the other states.

This conversion is of great importance. Suppose we apply Hadamard on the input qubit x (required tensor product if the input is multiple qubits) with the second qubit $|0\rangle$. What is the output of the blackbox? You will answer this as part of the assignment.

Suppose x was only 1 bit then the output is,

$$\frac{1}{\sqrt{2}}|0, f(0)\rangle + |1, f(1)\rangle.$$

In some sense, we have computed the function on all possible inputs.

This aspect is known as *quantum parallelism*. The catch is, the only way to get information from this state is by a measurement. This implies, we can only get the output for one particular input.

Note 5. We can only compute $(x, y) \to (x, y \oplus f(x))$, that does not imply $x \to f(x)$.

6 Assignment

Exercise 39. A half-adder is a circuit which takes x, y as input and gives output $x \wedge y, x \oplus y$. Draw a circuit for half adder using AND, OR and NOT.

Exercise 40. Make a circuit which takes x, y, c as input (interpret c as carry) and output $c', x \oplus y \oplus c$ where c' is one if at least two of x, y, c are 1. Why is this circuit helpful.

Hint: You can use half adder from the previous exercise.

Exercise 41. Show that AND, OR and NOT form a universal gate set.

Exercise 42. Read about super-dense coding. What is the relation between teleportation and super-dense coding?

Exercise 43. What will be the action of the following circuit on the input $|++\rangle$ state.

Exercise 44. Give two circuits on the same number of qubits such that their action on all basis states is same but different on some quantum state.

Exercise 45. Write the matrix representation of the gate which takes $|00\rangle$ to $|00\rangle$, $|01\rangle$ to $|11\rangle$, $|10\rangle$ to $|10\rangle$ and $|11\rangle$ to $|01\rangle$. What gate is this? What is its action on $(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

Exercise 46. Show how to switch from CNOT to CZ gate by moving Hadamard. Hint: $Z = HXH$.

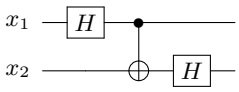


Fig. 13. A quantum circuit

References

1. S. Arora and B. Barak. *Computational Complexity: a modern approach*. Cambridge, 2009.
2. C. Gidney. From swapping to teleporting with simple circuit moves, 2016. `https://algassert.com/post/1628`.
3. D. Melkebeek. Quantum algorithms, 2023. `https://pages.cs.wisc.edu/~dieter/Courses/2023s-CS880/past-notes.html`.
4. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge, 2010.
5. Wikipedia. Elitzur-vaidman bomb tester, 2025. `https://en.wikipedia.org/wiki/Elitzur%E2%80%93Vaidman_bomb_tester`.