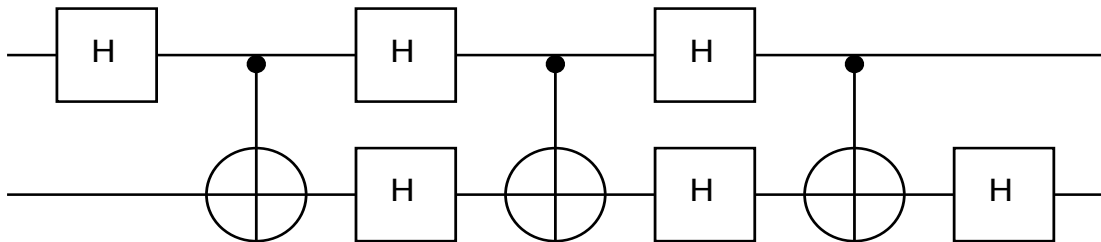


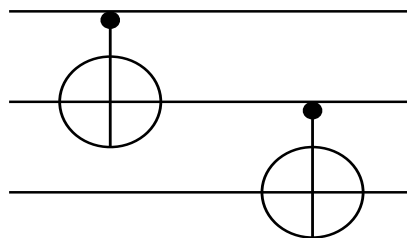
# Quantum Circuits

We have already seen the notation for quantum circuits, which is quite analogous to the notation for classical Boolean circuits.



Like classical *reversible* circuits, the number of quantum bits in must equal the number of quantum bits out. (There can be exceptions if we allow destructive measurements as part of a circuit; but we have seen that these can, in principle, always be postponed or eliminated.) Each *gate* in a quantum circuit stands for a unitary transformation on the affected bits, just as each classical gate stands for an invertible Boolean function. The unitary transformation acts as the identity on the other bits.

Unlike the classical case, however, the bits which enter a quantum circuit cannot be assumed to have individual states; in general, all of the bits may be in a joint, entangled state. Consider this circuit:

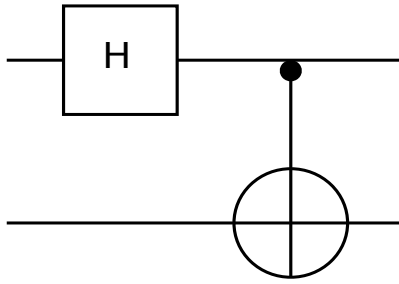


This unitary takes computational basis states to computational basis states. If the input state is  $|i\rangle \otimes |j\rangle \otimes |k\rangle$  the output state is  $|i\rangle \otimes |i \oplus j\rangle \otimes |i \oplus j \oplus k\rangle$ . If the input q-bits are in a general state, we use linearity: write the state in terms of the computational basis and apply the circuit term by term. E.g.,

$$\sqrt{1/2}(|0\rangle + |1\rangle)|00\rangle \rightarrow \sqrt{1/2}(|000\rangle + |111\rangle).$$

Note that a product state has become entangled; this is the Greenberger-Horne-Zeilinger (GHZ) state.

This procedure is more complicated if we include gates which do not preserve the computational basis. For example, consider this:



The Hadamard gate is applied only to the first q-bit. This means that the full unitary transformation is  $\hat{H} \otimes \hat{I}$  on both bits. We can see how the basis states are transformed:

$$|00\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$|01\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

$$|10\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

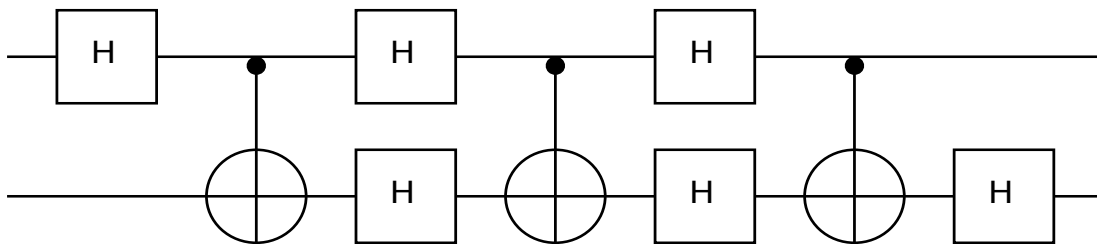
$$|11\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

This circuit takes computational basis states to Bell states. However, if we apply it to a general state, we must be careful to collect terms properly. For instance,

$$\begin{aligned}\sqrt{1/2}(|0\rangle + |1\rangle)|0\rangle &= \sqrt{1/2}(|00\rangle + |10\rangle) \\ \rightarrow (1/2)(|00\rangle + |11\rangle + |00\rangle - |11\rangle) &= |00\rangle.\end{aligned}$$

In this case, a product state is taken to a product state, in spite of the fact that this circuit can produce entanglement when applied to a basis state.

It is important to be able to translate a quantum circuit into the proper sequence of unitaries. The unitary appropriate to the gate is applied to the affected bits, while the identity is performed on all the other bits. If we take our first example:

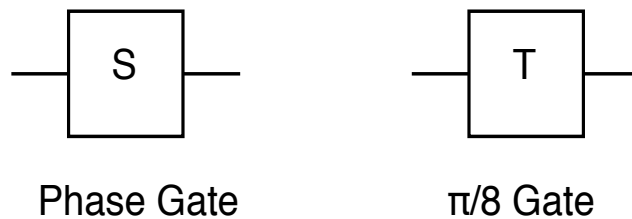
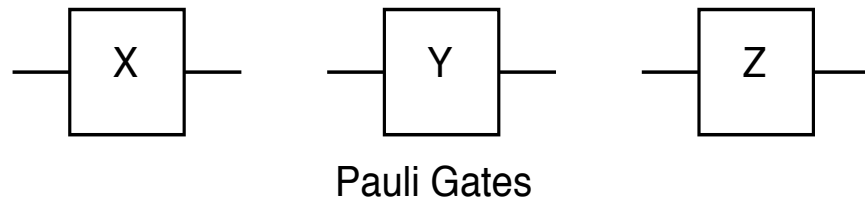


This is the same as the sequence of unitaries:

$$\begin{aligned}
 & (\hat{I} \otimes \hat{U}_H) \hat{U}_{\text{CNOT}} (\hat{U}_H \otimes \hat{U}_H) \hat{U}_{\text{CNOT}} \\
 & \quad \times (\hat{U}_H \otimes \hat{U}_H) \hat{U}_{\text{CNOT}} (\hat{U}_H \otimes \hat{I})
 \end{aligned}$$

Note that time in the circuit goes from left to right, but the corresponding unitary operators must be multiplied together from right to left, so that the earliest gates are applied to the state first.

We have so far seen the CNOT and the Hadamard gate. What are some other quantum gates? There is no standard list, but here are some common ones:



The Pauli gates we know; the other two are

$$\hat{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad \hat{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix},$$

Note that  $\hat{T}$  is the square-root of  $\hat{S}$ ,  $\hat{S} = \hat{T}^2$ , and  $\hat{S}$  is the square-root of  $\hat{Z}$ ,  $\hat{Z} = \hat{S}^2 = \hat{T}^4$ .

There are also common two-bit and three-bit gates. We have already seen the controlled-NOT and controlled-Z gates; there is also the SWAP gate which exchanges two bits (often indicated simply by crossing the wires in the diagram). Among three-bit gates, we have seen the *Toffoli* gate in its classical incarnation; quantum mechanically it exchanges  $|110\rangle \leftrightarrow |111\rangle$ . Another gate one sometimes encounters is the *Fredkin* gate, or controlled-SWAP: it exchanges  $|110\rangle \leftrightarrow |101\rangle$ .

We will also often see the controlled-U gate: an  $(n + 1)$ -bit gate, which applies the unitary operator  $\hat{U}$  on  $n$  bits if another control bit is in state  $|1\rangle$ , and not otherwise:  $|0\rangle\langle 0| \otimes \hat{I} + |1\rangle\langle 1| \otimes \hat{U}$ . Often this gate represents a large circuit. Other such sub-circuits are indicated by a block in which  $n$  q-bits enter and exit, having undergone some unitary transformation  $\hat{U}$ .

## Calculating functions

A common component of many classical and quantum algorithms is calculating a function. In this case, we consider the  $n$  bits of input to represent an  $n$ -bit integer  $x$  in binary notation  $x_{n-1} \cdots x_1 x_0$ . We will often use a shorthand notation in which  $|x\rangle$  represents a state of  $n$  q-bits:  $|x\rangle = |x_{n-1}\rangle \otimes \cdots \otimes |x_1\rangle \otimes |x_0\rangle$ . Such an  $n$ -bit composite system is called a *quantum register*. The output is the value of the function  $f(x)$ , which we assume to be an  $m$ -bit integer in binary notation.

If  $f(x)$  is an *invertible* function, then  $m = n$  and the function must be a permutation of the numbers  $0, \dots, 2^n - 1$ . In this case there is a unitary transformation  $\hat{U}_f$  such that  $\hat{U}_f|x\rangle = |f(x)\rangle$ .

$$\hat{U}_f \equiv \sum_{x=0}^{2^n-1} |f(x)\rangle\langle x|.$$

If  $f$  is not invertible (which is usually the case), there is no such  $\hat{U}_f$ . What do we do then? Instead of using a single register to hold both the input and the output, we have two separate registers, and define a unitary transformation

$$\hat{U}_f(|x\rangle \otimes |y\rangle) \equiv |x\rangle \otimes |y \oplus f(x)\rangle,$$

in which  $\oplus$  is bitwise-XOR. This transformation is clearly invertible. We now apply  $\hat{U}_f$  to the input state  $|x\rangle \otimes |0\rangle$  and get  $|x\rangle \otimes |f(x)\rangle$  as the output.

Most quantum algorithms contain a unitary transformation like this. Of course, to carry it out we actually need to construct a circuit to perform  $\hat{U}$ . If the function  $f(x)$  has a classical circuit, we can make this reversible and translate it directly into a quantum circuit.

## Scratch bits and ancillas

Just as in classical circuits, in some cases a circuit can be made more efficient by the use of extra bits, called “scratch bits,” which are reset to zero at the end of the calculation. For instance, calculating a function  $f(x)$  may be more conveniently done by first calculating some function  $p(x)$ , and then a function  $g(p)$ :  $f(x) = g(p(x))$ . We call  $p(x)$  a *partial* result. We now have three registers: input, output, and scratch, and define two unitary transformations  $\hat{U}_g$  and  $\hat{U}_p$ :

$$\hat{U}_p(|x\rangle|y\rangle|z\rangle) = |x\rangle|y\rangle|z \oplus p(x)\rangle$$

$$\hat{U}_g(|x\rangle|y\rangle|z\rangle) = |x\rangle|y \oplus g(z)\rangle|z\rangle$$

We see  $\hat{U}_g\hat{U}_p(|x\rangle|0\rangle|0\rangle) = |x\rangle|f(x)\rangle|p(x)\rangle$ . The scratch space can be re-used by inverting  $\hat{U}_p$ :  $\hat{U}_p^\dagger\hat{U}_g\hat{U}_p(|x\rangle|0\rangle|0\rangle) = |x\rangle|f(x)\rangle|0\rangle$ .

How do we do  $\hat{U}_p^\dagger$ ? If a unitary is produced by a sequence of gates  $\hat{U} = \hat{G}_N \cdots \hat{G}_1$ , then obviously  $\hat{U}^\dagger = \hat{G}_1^\dagger \cdots \hat{G}_N^\dagger$ . Many common gates are their own inverses; for instance, the Hadamard, X, Y, Z, CNOT, SWAP, Toffoli and Fredkin gates. A circuit with only these gates can be inverted by it running backwards. If it involves other gates (such as the phase or  $\pi/8$  gates), one must explicitly include their inverses. (E.g.,  $\hat{S}^\dagger = \hat{S}^3$ .)

In the case of  $\hat{U}_p^\dagger$ , the case is even simpler. Since a second bitwise XOR undoes the first, in fact  $\hat{U}_p$  is its own inverse. In this case, our sequence can be  $\hat{U}_p \hat{U}_g \hat{U}_p$ .

In addition to scratch bits, some extra ancilla bits may be necessary to make the circuit reversible, just as in the classical case. E.g., with ancillas the Toffoli gate is sufficient to do any classical gate. Ancillas are also useful for postponed or indirect measurements.

## Oracles

Just as in classical computation, the idea of an *oracle* is useful in quantum computation. In the classical case, an oracle was a “black box” which input an  $n$ -bit number  $x$  and output a function  $f(x)$ . (We can make oracles reversible, just like any other classical circuit; for instance, it could input  $x$  and  $y$  and output  $x$  and  $y \oplus x$ .)

In the quantum case, an oracle is a black box which takes  $n$  q-bits as input and performs a unitary transformation  $\hat{U}$  on them. For instance, it could input two quantum registers, and perform the transformation  $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ . However, there are other possibilities as well; for instance, it could do a transformation

$$|x\rangle \rightarrow (-1)^{f(x)}|x\rangle,$$

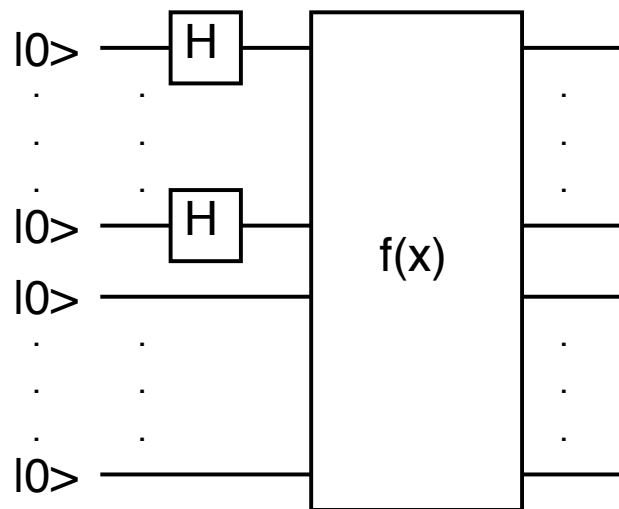
performing a phase shift conditioned on the value of the function  $f(x)$ .

Computations involving oracles usually center on determining some property of the function  $f(x)$ , or finding a value of  $x$  which has some particular value  $f(x)$ . In some cases, the fact that  $f(x)$  is determined by a black box is important; if the function  $f(x)$  were known, the problem would be solved. In other cases, however, this property is sufficiently nonobvious that knowledge of  $f(x)$  makes no difference. In these cases, the difference between an oracle problem and an ordinary computation is somewhat blurred.

Just as in the classical case, we call each invocation of an oracle a *query*, and the number of queries needed to complete the circuit the *query complexity*.

# Quantum Parallelism

So far, things don't look very different between classical and quantum circuits. What things can we do with quantum bits that we can't with classical? Consider this circuit:



The first  $n$  bits undergo Hadamard gates:

$$|0\rangle^{\otimes n} \rightarrow [(|0\rangle + |1\rangle)/\sqrt{2}]^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle.$$

This produces an equally-weighted superposition of all  $|x\rangle$ .

We then see what happens in the whole circuit. The input register is put in a superposition of all inputs, and together with the output register is passed to the oracle.

$$\begin{aligned}
 |0\rangle^{\otimes n} \otimes |0\rangle^{\otimes m} &\rightarrow \left( \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \right) \otimes |0\rangle^{\otimes m} \\
 &\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle
 \end{aligned}$$

By a single call to the oracle, we have computed  $f(x)$  for *every possible input*  $x$ ! There is no classical analogue of this process, which Deutsch called *quantum parallelism*.

It would seem that this capability should make quantum computers far more powerful than classical computers. But are they?

By  $n$  Hadamard gates and a single oracle query, we have calculated  $f(x)$  for every possible value of  $x$ ; they are all contained in this massive superposition. But does it do us any good? What happens if we try to read the values out?

We do this by measuring the output register in the computational basis. If we do this, we get the result  $f(x) = y$  with probability

$$p_y = \sum_{x, f(x)=y} 1/2^n.$$

We could do equally well by choosing a single  $x$  at random and calculating  $f(x)$ . This reflects a general principle: we cannot get more information out of measuring  $m$  q-bits than  $m$  classical bits.

Let us see why this is so. We learned before that the *Shannon entropy* bounds how much we can learn from a measurement:

$$S = - \sum_j p_j \log_2 p_j,$$

where  $j$  labels the outcomes. For a projective measurement of an  $D$ -dimensional system, there are at most  $D$  distinct outcomes, for which the maximum value of  $S$  is  $\log_2 D$ . In the case of  $m$  q-bits,  $D = 2^m$ , so one can gain at most  $m$  bits of information.

(Generalized measurements are more complicated; we can make  $S$  as large as we like. But this excess information represents *pure randomness*; one cannot learn more than with a complete projective measurement.)

To learn  $f(x)$  for all values of  $x$  would require  $m2^n$  bits of information. Clearly this is out of the question.

So it seems that, rather than being extraordinarily powerful, quantum parallelism buys you exactly nothing. But that, too, would be incorrect. One can acquire at most  $m + n$  bits of information by measuring the input and output registers; but by being more subtle, one can do much better than just getting a random value of  $x$  and  $f(x)$ . One could instead, learn something about the function  $f(x)$  *as a whole*. One could acquire up to  $n + m$  bits of information giving some *property* of  $f$ , which might be difficult to acquire classically without making many queries to the oracle.

What is an example of such a property?

## Deutsch's Problem

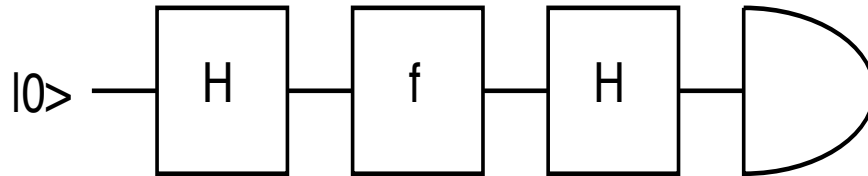
The simplest Boolean function  $f(x)$  is a function of a single bit. There are exactly four such functions. Here they are:

function	$f(0)$	$f(1)$
$f_0$	0	0
$f_1$	0	1
$f_2$	1	0
$f_3$	1	1

We see that  $f_0$  and  $f_3$  return the same thing regardless of the input  $x$ . We call these functions *constant*. By contrast,  $f_1$  and  $f_2$  return an equal number of 0's and 1's. We call these functions *balanced*.

Suppose we have a classical oracle which inputs  $x$  and returns  $f(x)$ . How many queries are needed to determine if  $f$  is constant or balanced? (Answer: two queries.)

Consider the following circuit:



where the oracle does the transformation

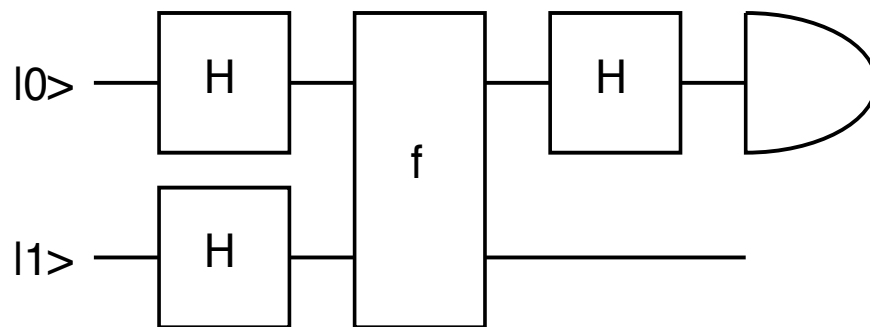
$$|x\rangle \rightarrow (-1)^{f(x)} |x\rangle.$$

We put the input q-bit into state  $(|0\rangle + |1\rangle)/\sqrt{2}$  and send it to the oracle. If  $f$  is *constant*, the state acquires an *overall* phase, and is returned to the state  $|0\rangle$  by the second Hadamard. If  $f$  is *balanced*, the two terms acquire a relative phase of  $-1$ , and the q-bit goes to  $|1\rangle$ . So a *single query* is sufficient to solve Deutsch's problem.

What if the oracle *isn't* the type that flips the phase of the state, but actually returns the value of  $f(x)$ :

$$|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle.$$

We can still solve Deutsch's problem with only a single query. Consider this circuit:



Before the oracle query, the bits are in state  $(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)/2$ . It is easy to check that applying the oracle gives  $|x\rangle(|0\rangle - |1\rangle) \rightarrow (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$ . So we again solve the problem by measuring the first bit *even though the value of  $f(x)$  was “put” in the second bit*. Remember, in QM the direction of information flow is basis dependent!

This was the first algorithm ever presented where a quantum computer outperformed a classical computer. This performance may not seem impressive—one query versus two—and the problem may be artificial. But the structure of the algorithm is worth noting:

1. The input and output registers start in computational basis states. ( $|0\rangle$  and  $|1\rangle$  in this case.)
2. By Hadamard gates, the input register is put in a superposition of all input values.
3. A unitary to evaluate  $f(x)$  is done on both registers.
4. The input register is transformed again, and measured in the computational basis.

We will encounter this structure again soon!

*Next time: universal quantum gates!*