

Quantum Circuits and Algorithms

G. Byrd - ASPLOS Tutorial - Apr 14, 2019

Quantum State (qubit)

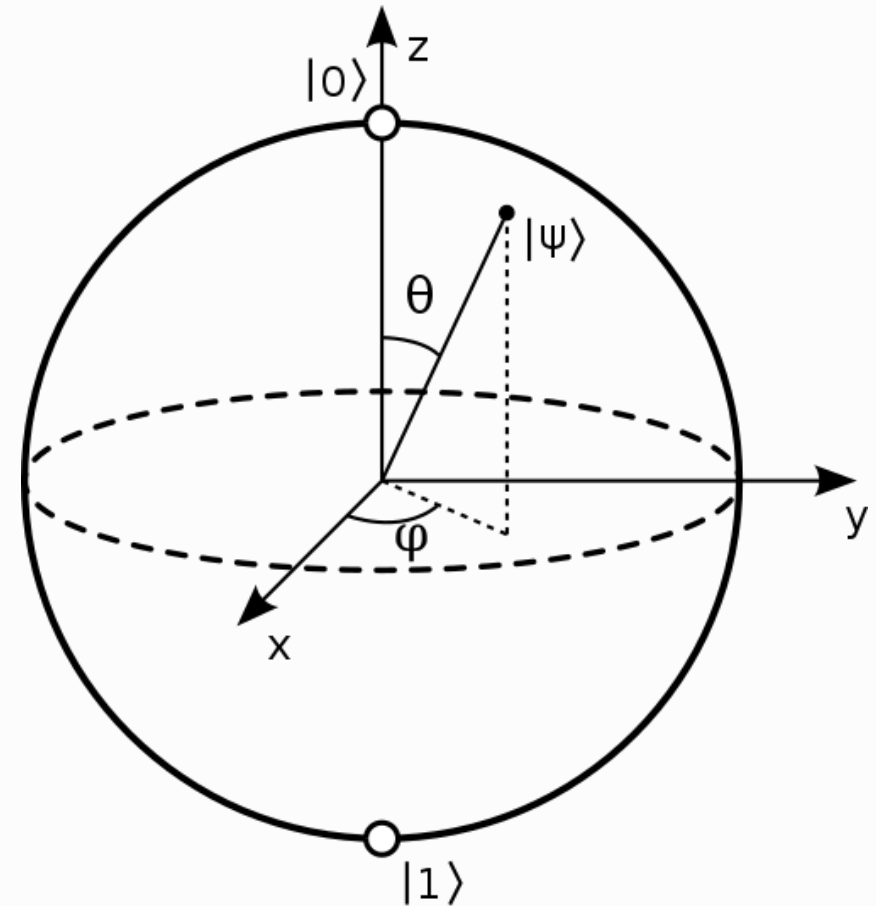
Mathematically represented as a **vector**, or a point on the surface of the Bloch sphere:

$$|\psi\rangle = \underbrace{\cos\left(\frac{\theta}{2}\right)}_{\alpha} |0\rangle + e^{i\varphi} \underbrace{\sin\left(\frac{\theta}{2}\right)}_{\beta} |1\rangle \quad |\alpha|^2 + |\beta|^2 = 1$$

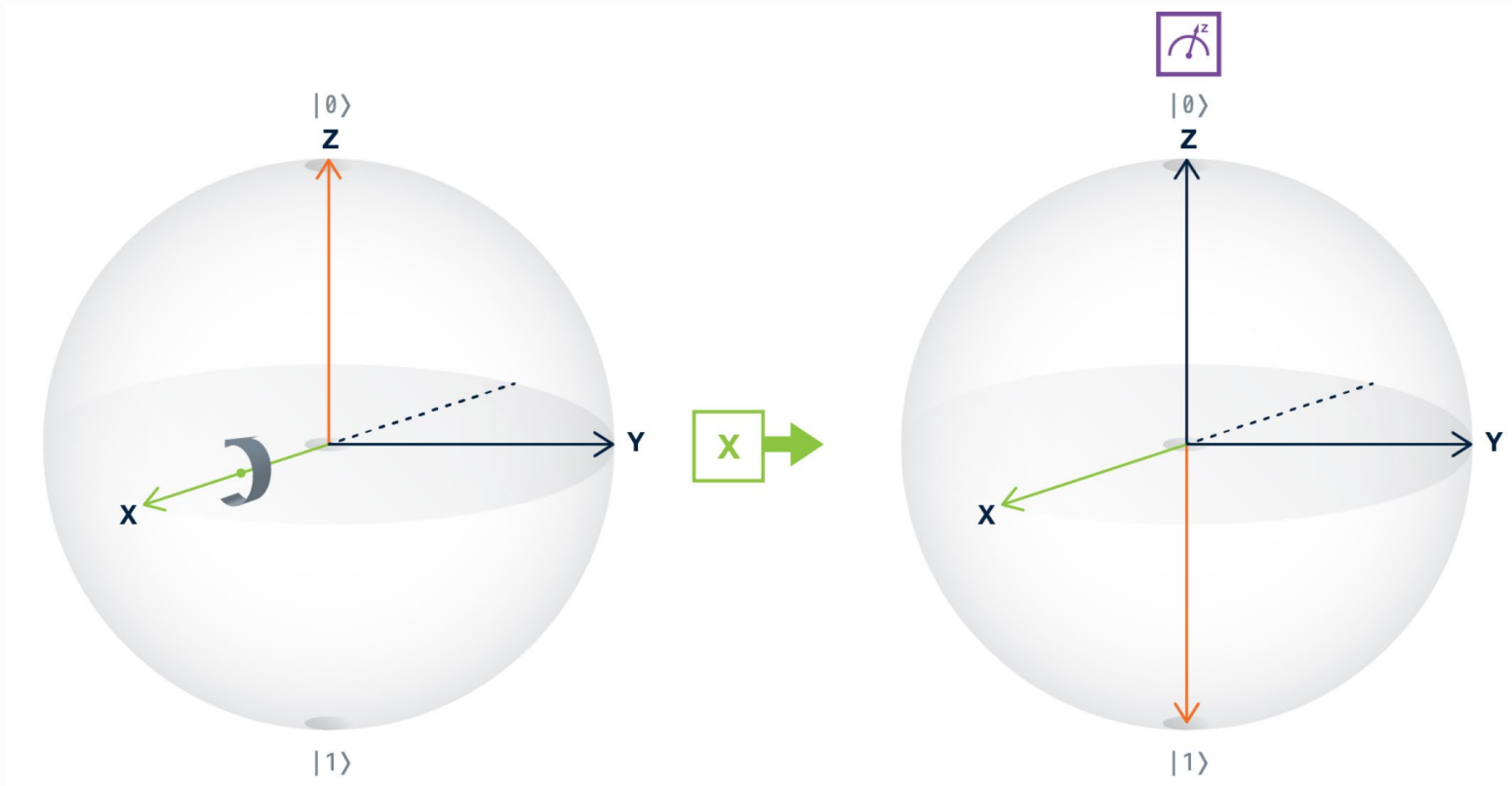
Measurement = projection of state to a basis vector
(changes the state – superposition is destroyed)

Quantum gate is a transformation from one qubit state to another.
Single-qubit gate = rotation around Bloch sphere. **Reversible**.
Represented by a **matrix** (unitary, ...) acting on the **vector**.

NOTE: There are many possible basis vector sets – any antipodal points on the Bloch sphere are orthogonal. “Standard” basis is $\{|0\rangle, |1\rangle\}$.



X Gate: NOT

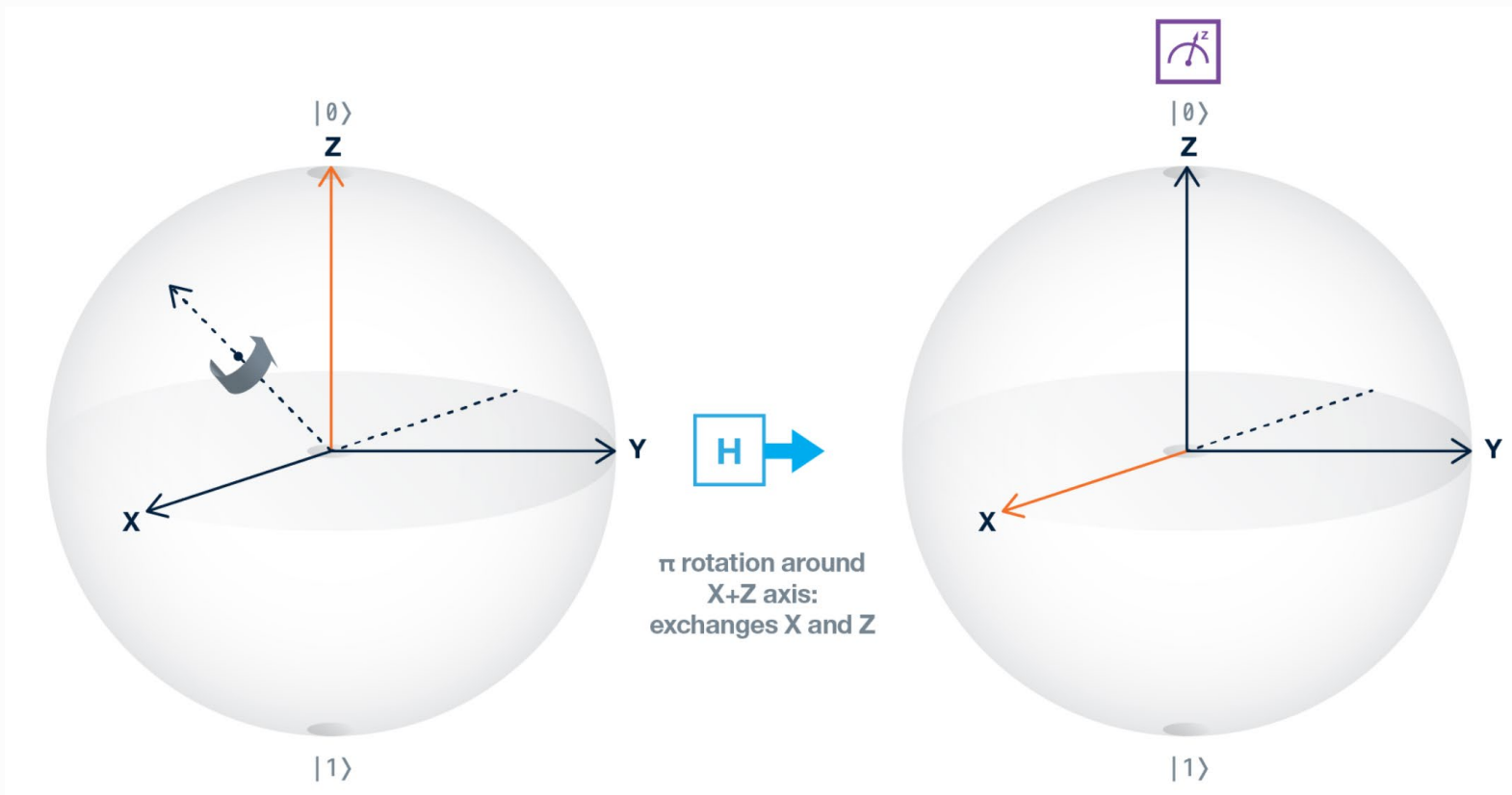


Start	End
$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$
$\alpha 0\rangle + \beta 1\rangle$	$\beta 0\rangle + \alpha 1\rangle$

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

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Hadamard (H) Gate: Superposition

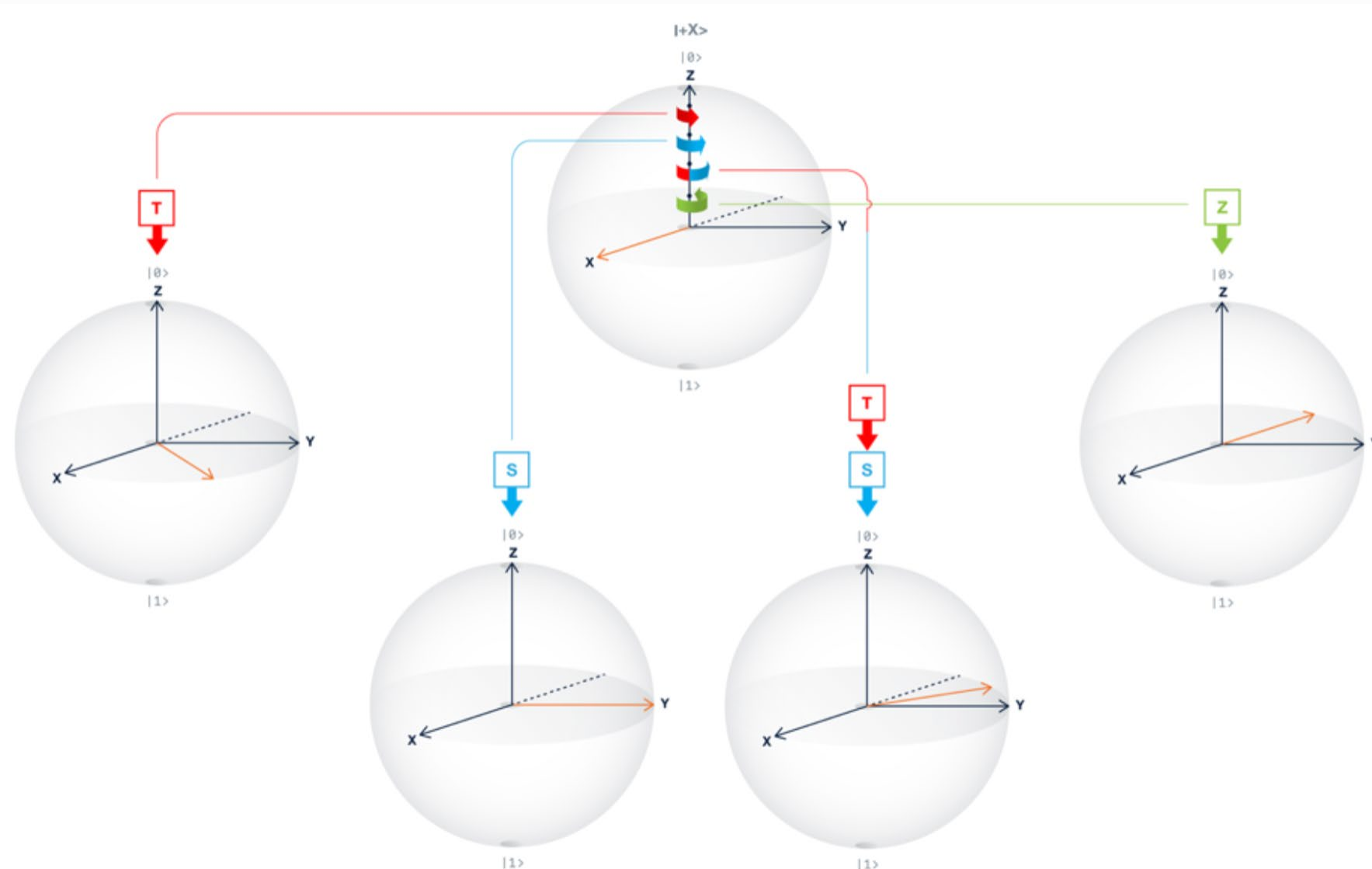


Start	End	AKA
$ 0\rangle$	$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$	$ +\rangle$
$ 1\rangle$	$\frac{1}{\sqrt{2}}(0\rangle - 1\rangle)$	$ -\rangle$

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

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Phase: Z, S, T



Rotations around the Z axis

$$T = \pi/4$$




















$$S = \pi/2$$

$$Z = \pi$$

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$



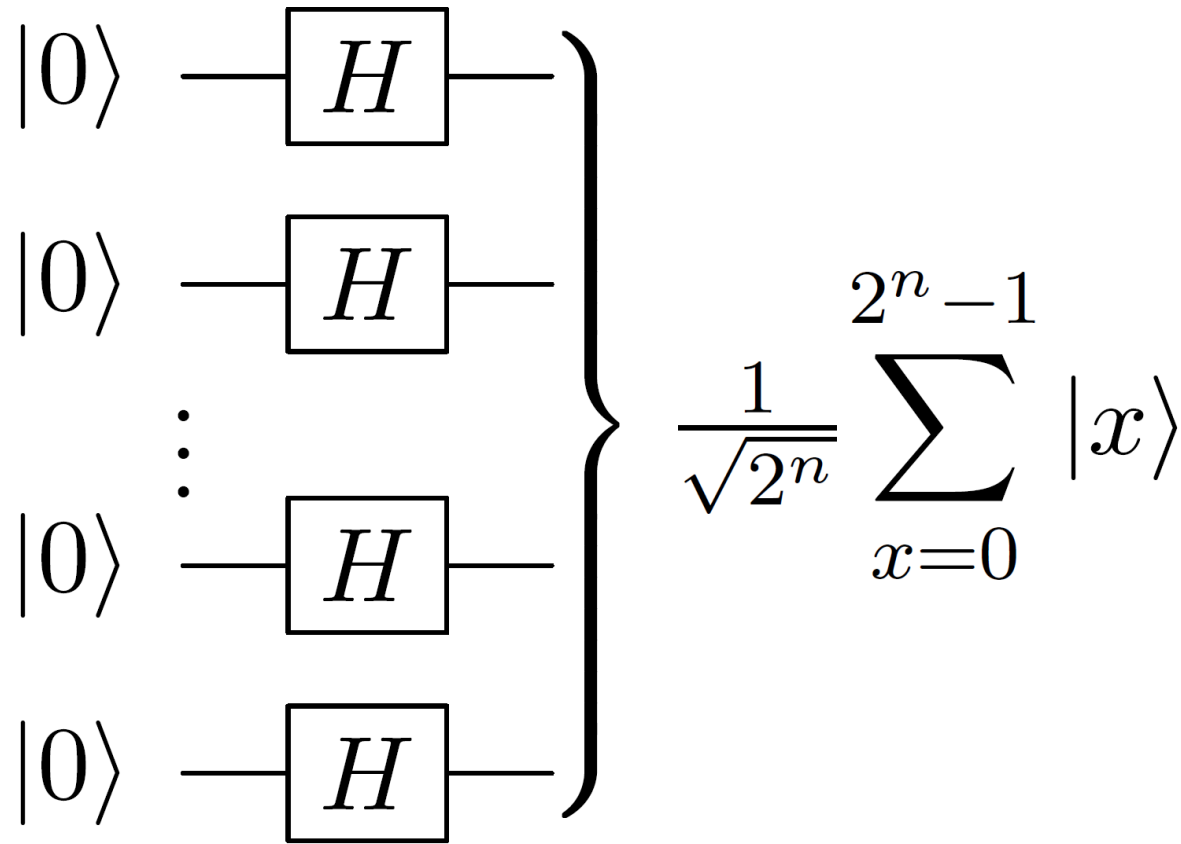
Phase: Z, S, T

Gate sequence	Rotation around Z	Probability of 0	Probability of 1
  	0	1.0	0
   	$\pi/4$	0.85	0.15
   	$\pi/2$	0.50	0.50
    	$3\pi/4$	0.15	0.85
   	π	0	1



Create +X state Change qubit phase Measure in superposition basis

Walsh-Hadamard Transform

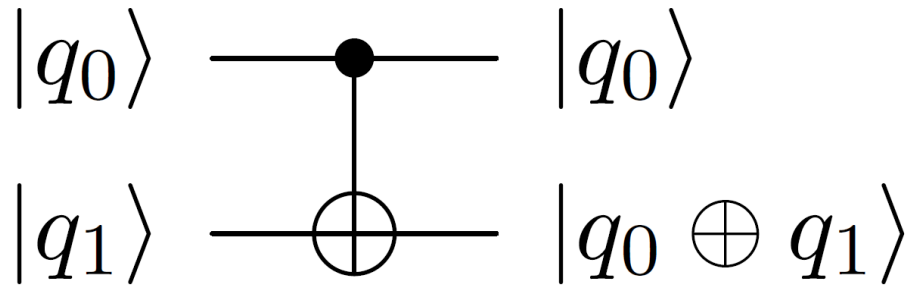


Used in the setup phase of algorithms, to create a **superposition of all inputs**.

Transformations occur on all components of the superposition. This is the source of **quantum parallelism**.

Two-qubit Gate: CNOT

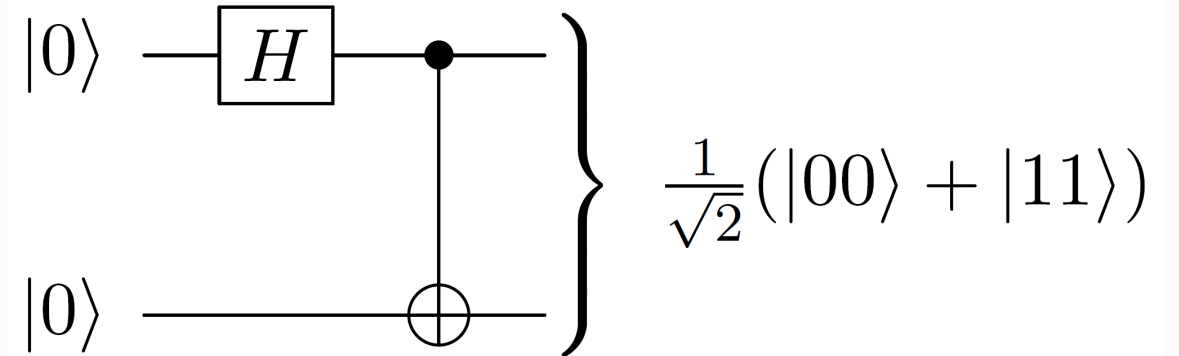
CNOT = controlled-NOT



Start	End
$ 00\rangle$	$ 00\rangle$
$ 01\rangle$	$ 11\rangle$
$ 10\rangle$	$ 10\rangle$
$ 11\rangle$	$ 01\rangle$

Be careful about notions of “control” and “target.”
More about this later...

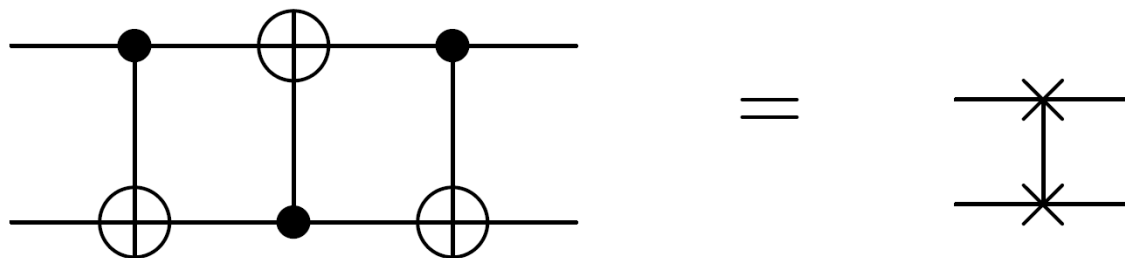
Entanglement: Bell Pair



There is no tensor product
 $|a\rangle \otimes |b\rangle$
that corresponds to this state.

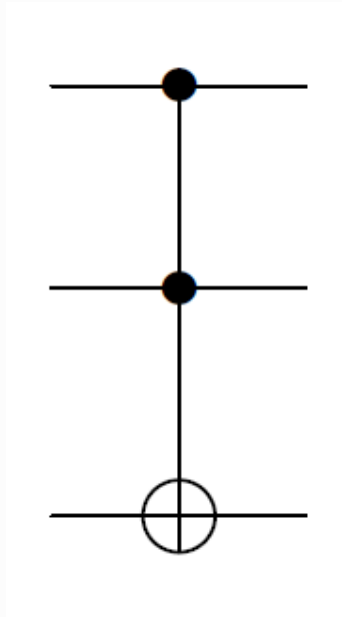
Other Two-Bit Gates (IBM Qiskit)

- controlled **Pauli** gates (X, Y, Z) – controlled X is CNOT
- controlled **Hadamard** gate
- controlled **rotation** gates (Rx, Ry, Rz)
- controlled **phase** gate (u1)
- controlled **u3** gate
- **swap** gate

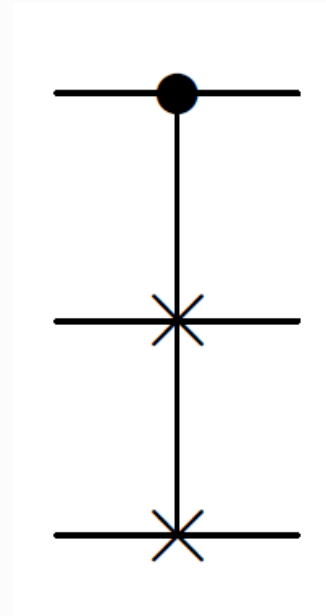


Three-qubit Gates

Toffoli: controlled CNOT

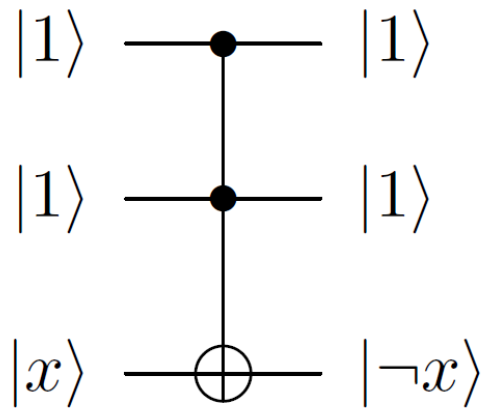


Fredkin: controlled swap

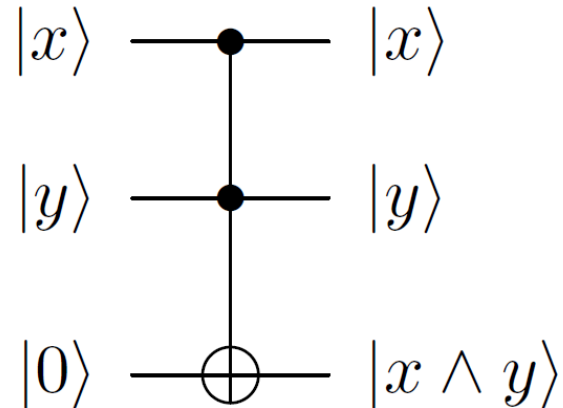


These are not implemented directly on the IBM Q. They are built from 1- and 2-qubit gates.

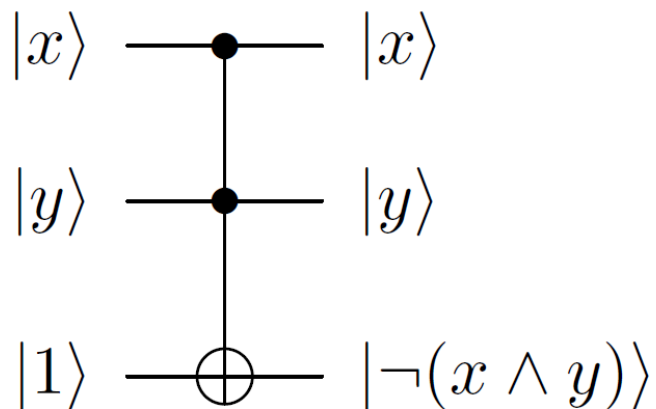
Toffoli: Reversible Classic Gates



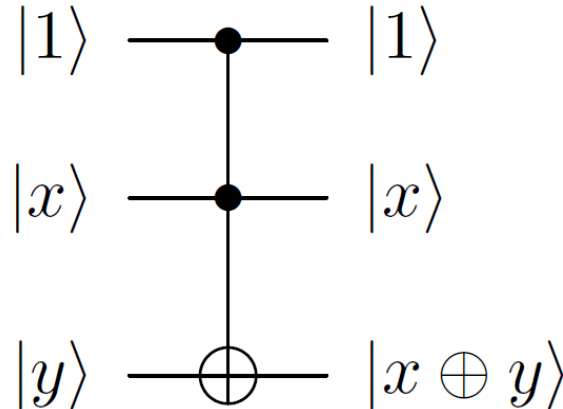
NOT



AND

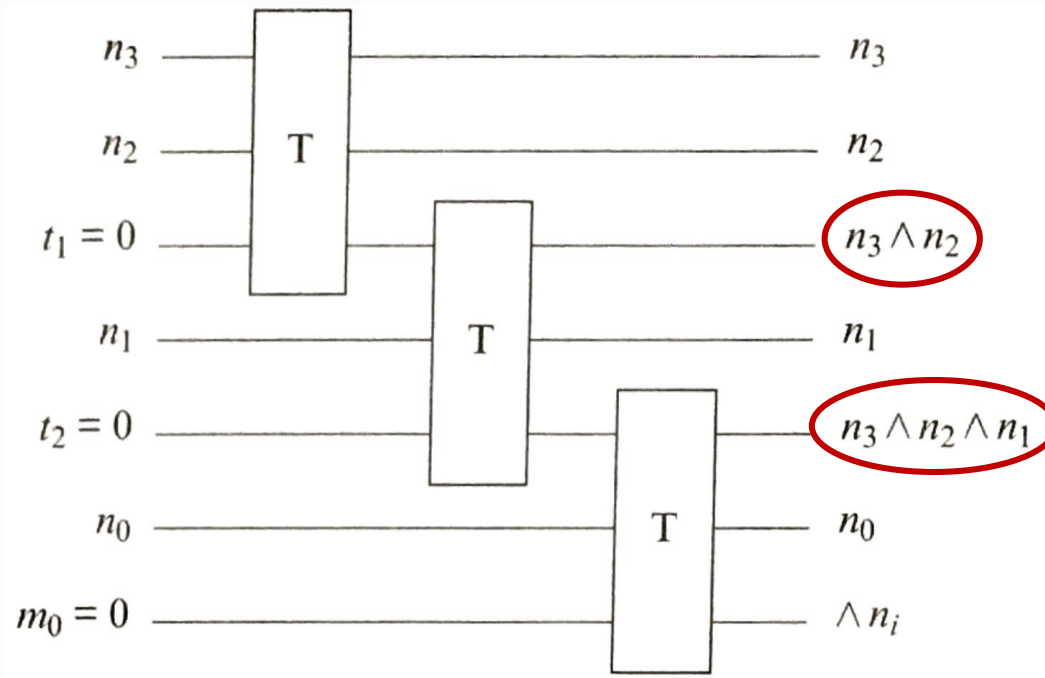


NAND

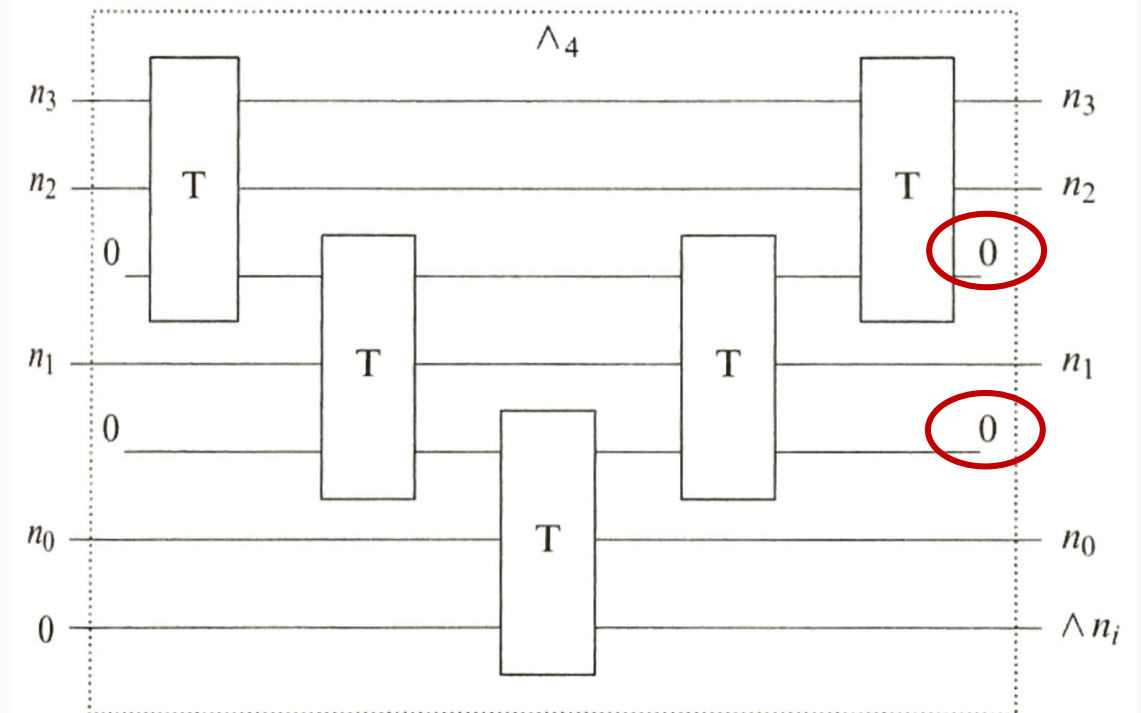


XOR

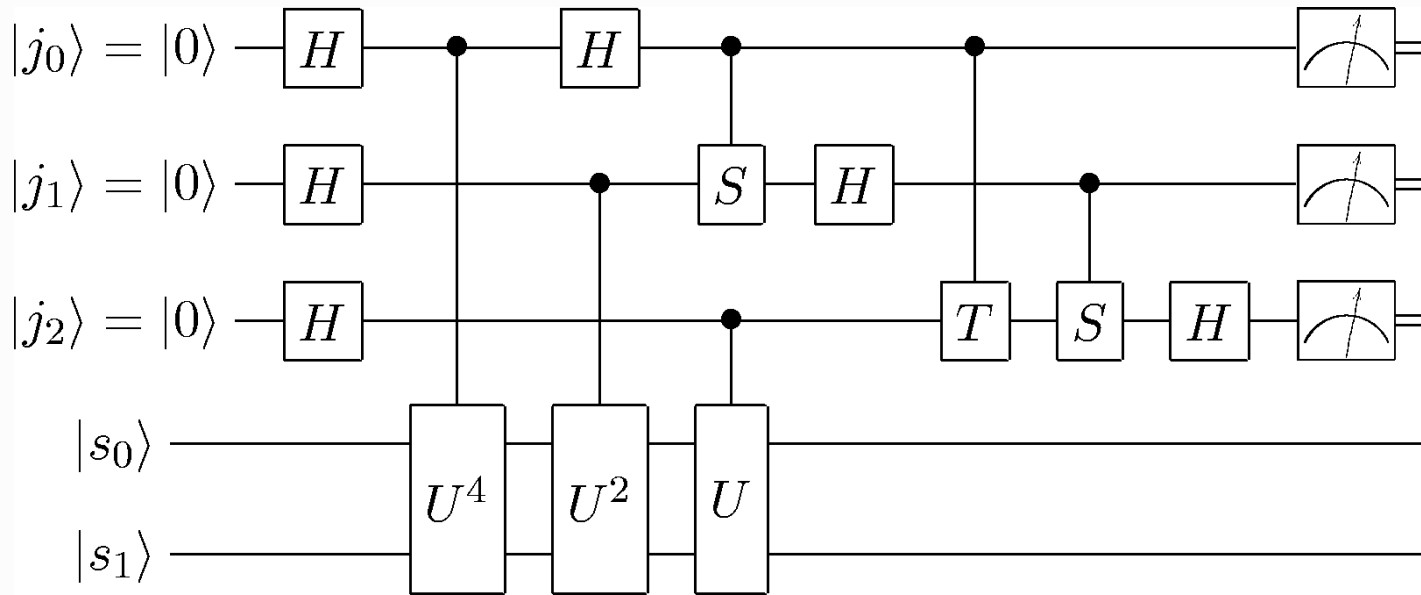
Reusing Temporary Bits



These bits are no longer zero, and can't be reused if this feeds into additional computation. Can't just "reset" them, because that's not reversible. Need to **uncompute** to reclaim them.



Quantum Circuit



Measurement.

Double line represents classical bit.

Standard Circuit Model

- CNOT plus all single-bit transformations
- Measurement in the standard basis

Any quantum transformation can be realized in terms of the basic gates of the standard circuit model.

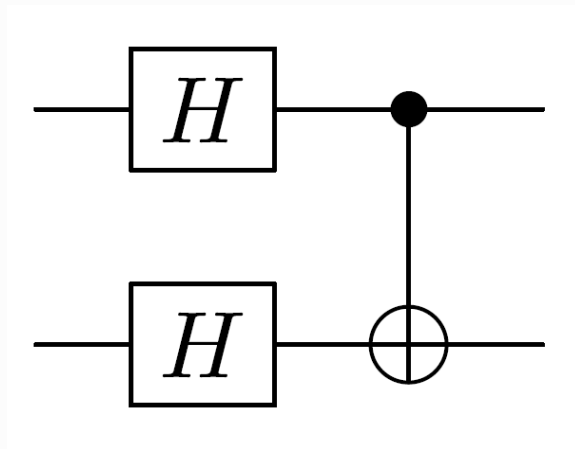
Time flows left to right.

Quantum gates (operators) are applied sequentially to qubit states, with result shown on the right.

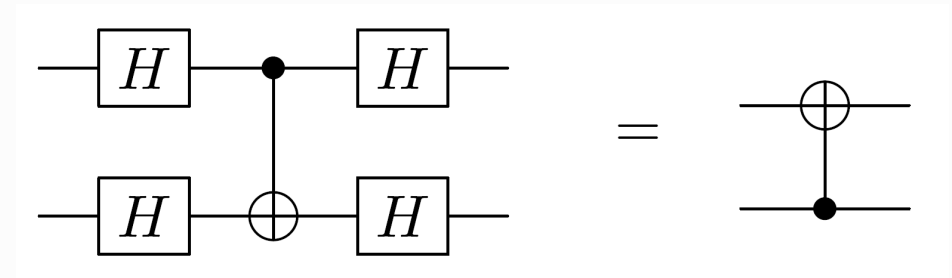
Caution 1: Notion of Control

The notions of **control** and **target** bit is a carryover from the classical gate, and should not be taken too literally. Do not conclude that the control bit is never changed.

Consider the CNOT gate operating in the Hadamard basis:



Start	End
$ ++\rangle$	$ ++\rangle$
$ +-\rangle$	$ --\rangle$
$ -+\rangle$	$ -+\rangle$
$ --\rangle$	$ +-\rangle$

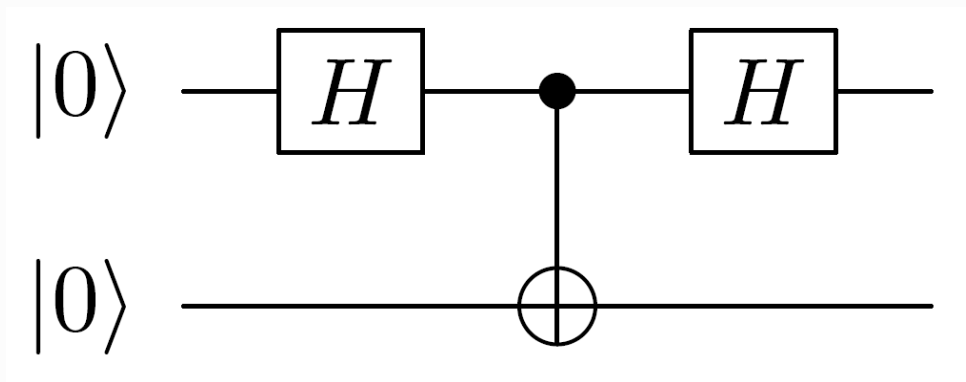


In this case, it's the first qubit that changes.

Caution 2: Reading Circuit Diagram

The graphical representation of a circuit can be misleading. Must “do the math” and figure out exactly what transformation is happening, even if all qubits are in the standard basis.

What is the output of the following circuit?

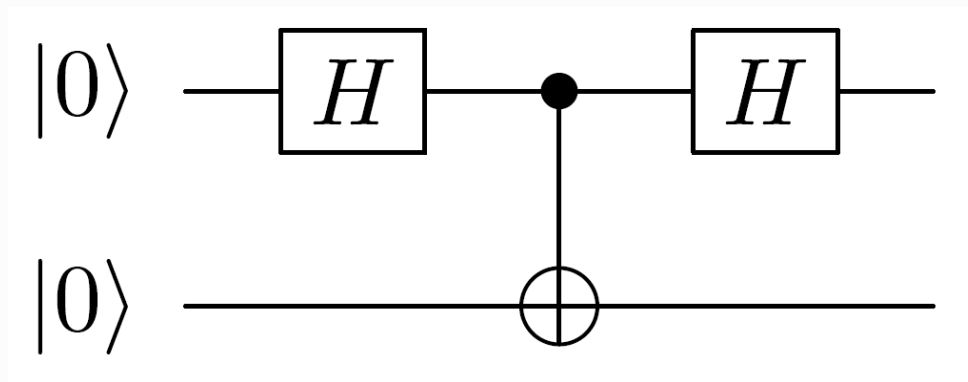


Because the H gate is its own inverse, you might think that the first qubit will be unchanged. But the output is $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)$ – not obvious from the diagram.

Caution 2: Reading Circuit Diagram

The graphical representation of a circuit can be misleading. Must “do the math” and figure out exactly what transformation is happening, even if all qubits are in the standard basis.

What is the output of the following circuit?



After H:

$$\frac{1}{\sqrt{2}} (|00\rangle + |10\rangle)$$

After CNOT:

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

After H:

$$\frac{1}{2} ((|00\rangle + |10\rangle) + (|01\rangle - |11\rangle))$$

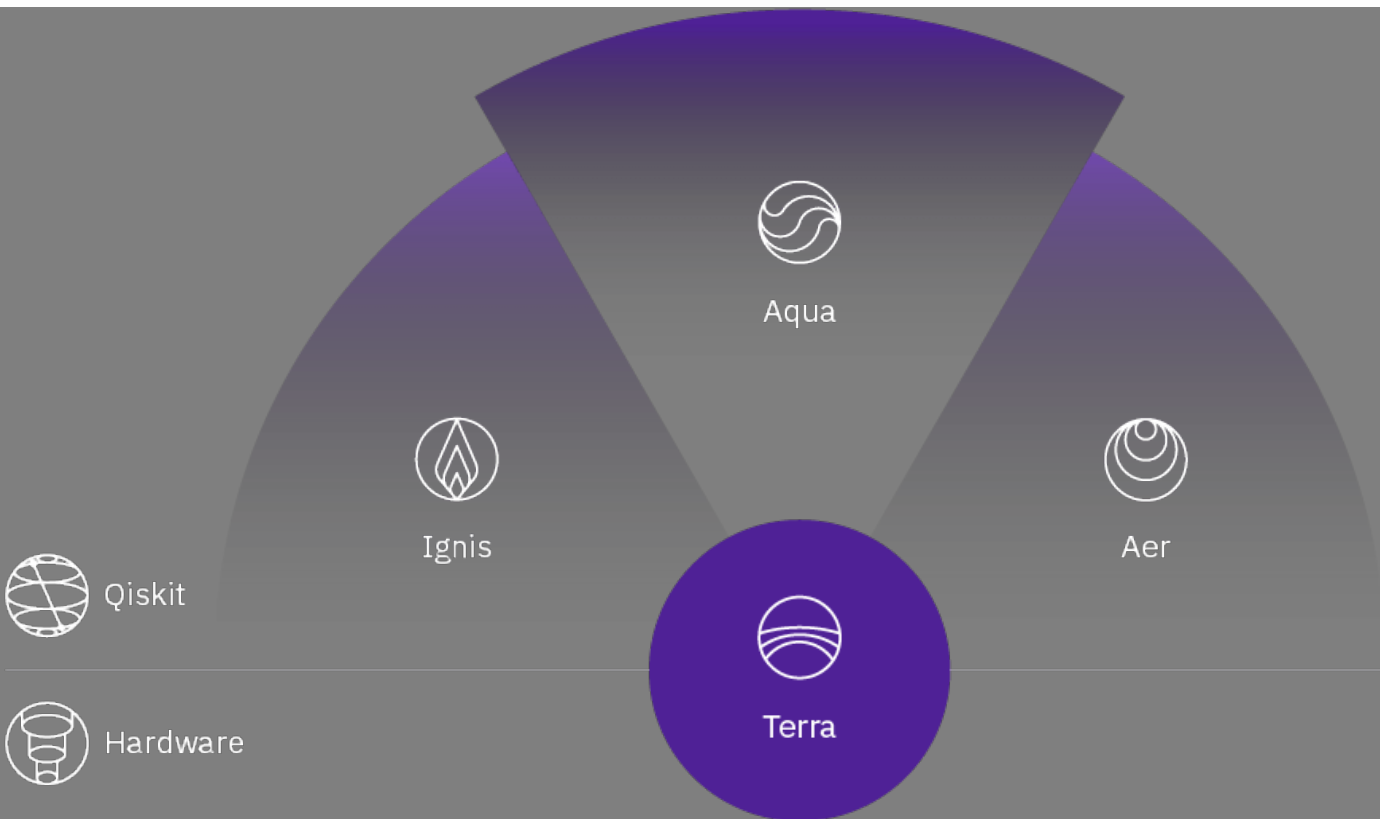
Because the H gate is its own inverse, you might think that the first qubit will be unchanged.

But the output is $\frac{1}{2} (|00\rangle + |01\rangle + |10\rangle - |11\rangle)$ – not obvious from the diagram.



Intro to Qiskit

Qiskit = IBM QC Dev Platform



- **Terra:** Composing programs using circuits and pulses
- **Aqua:** Building algorithms and applications
- **Aer:** Simulators, emulators, and debuggers
- **Ignis:** Addressing errors and noise

Qiskit Terra

- Build
 - Create circuit out of registers, gates
- Compile
 - Translate to QASM, then to backend instructions
- Execute
 - Backends = simulators (Aer), hardware

Building a Circuit

QuantumRegister

- Collection of qubits
- Indexed to reference individual qubit: `q[0]`

ClassicalRegister

- Collection of bits
- Used as the receiver of measurements on qubits

QuantumCircuit

Starts with set of registers

Add gates specifying registers/qubits as arguments

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit

qreg = QuantumRegister(3) # a 3-qubit register
creg = ClassicalRegister(3) # a 3-bit classical register
qc = QuantumCircuit(qreg,creg) # create a circuit

qc.measure(qreg,creg) # measure all qubits in qr, put results in cr
```

Basic Gates

Quantum Gate	...on qubits	...on register
X (NOT)	<code>qc.x(qreg[0])</code>	<code>qc.x(qreg)</code>
Hadamard	<code>qc.h(qreg[0])</code>	<code>qc.h(qreg)</code>
CNOT	<code>qc.cx(qreg[0], qreg[1])</code>	--
Toffoli	<code>qc.ccx(qreg[0], qreg[1], qreg[2])</code>	--
Phase shift	<code>qc.u1(angle, qreg[0])</code>	<code>qc.u1(angle, qreg)</code>
Swap	<code>qc.swap(qreg[0], qreg[1])</code>	--
Measure (not a gate)	<code>qc.measure(qreg[0], creg[0])</code>	<code>qc.measure(qreg)</code>
Reset (not a gate)	<code>qc.reset(qreg[0])</code>	<code>qc.reset(qreg)</code>

Other Circuit Operations

Operation	Description
<code>qc.barrier()</code>	Completes operations before proceeding. Can specify registers, qubits.
<code>qc.add(<i>regs</i>)</code>	Add register(s) to circuit.
<code>qc.combine(<i>circuit</i>)</code>	Appends circuit (if compatible). Creates new circuit (<code>qc + circuit</code>) and returns it.
<code>qc.extend(<i>circuit</i>)</code>	Appends circuit (if compatible). Modifies <code>qc</code> .
<code>qc.qasm()</code>	Returns a string containing the QASM representation of circuit.

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
```

```
q = QuantumRegister(2)
```

```
c = ClassicalRegister(2)
```

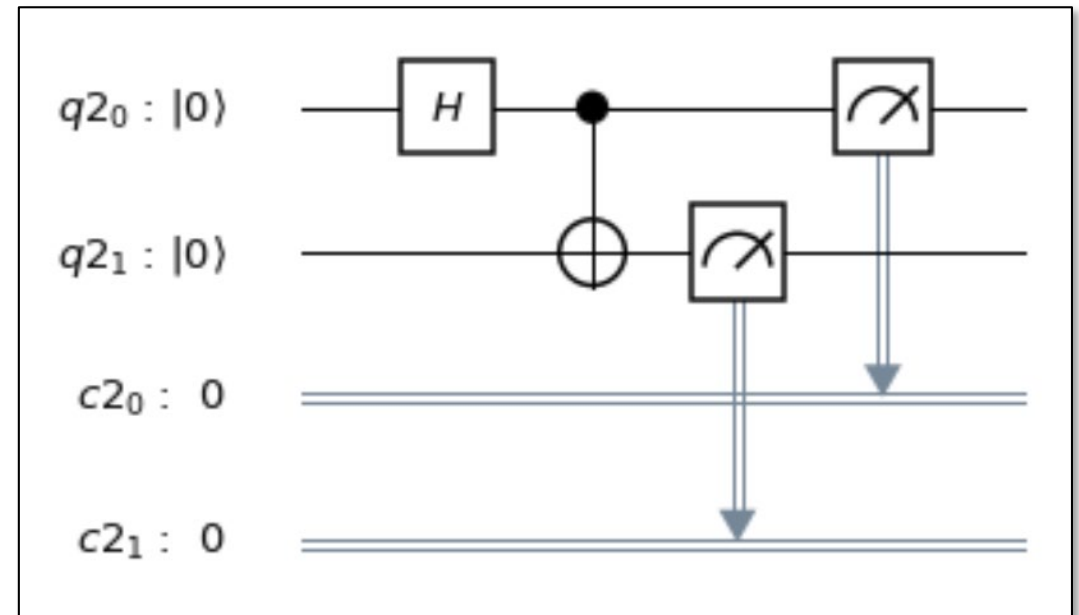
```
qc = QuantumCircuit(q, c)
```

```
qc.h(q[0])          # Hadamard on first qubit
```

```
qc.cx(q[0],q[1])    # CNOT to entangle
```

```
# creates a Bell state
```

```
qc.measure(q,c)
```



Compiling and Running

- Provider
 - Facilitates access to a selection of backends
 - Aer Provider
 - simulators, running locally on your machine
 - IBM Q Provider
 - hardware, remote simulator
- Backend
 - Runs a compiled program (Qobj) and reports result
- Job
 - The result of an execution
 - Asynchronous – query job to see status
 - Get result when complete

Backends

- To compile/execute a circuit, must specify a backend.
- Simulators:
 - Local (Aer):
 - `qasm_simulator` – emulates a machine with/without noise, multi-shot
 - `statevector_simulator` – single shot, returns state vector
 - `unitary_simulator` – returns unitary matrix represented by circuit
 - IBMQ: `ibmq_qasm_simulator`
- Hardware:
 - IBMQ provider – to be discussed later

Job Operations

Operation	Description
<code>job.status()</code>	Returns current status.
<code>job.done()</code>	Returns True if done, False if not.
<code>job.id()</code>	Identifier (remote provider only)
<code>job.result()</code>	Results from completed job.
<code>job.result().get_counts()</code>	Instances of various measured states, e.g. { '111': 512, '000': 512 }
<code>job_monitor(job)</code>	Loop that waits for job to complete, periodically printing the job status.

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit import Aer, execute
from qiskit.tools.visualization import plot_histogram

# ... deleted circuit building commands...
qc.measure(q,c)

backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=512) # shots default = 1024
result = job.result()
print(result.get_counts())
plot_histogram(result.get_counts())
```

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit import Aer, execute
from qiskit.tools.visualization import plot_histogram
```

```
# ... deleted circuit building commands...
```

```
qc.measure(q,c)
```

{'00': 269, '11': 243}

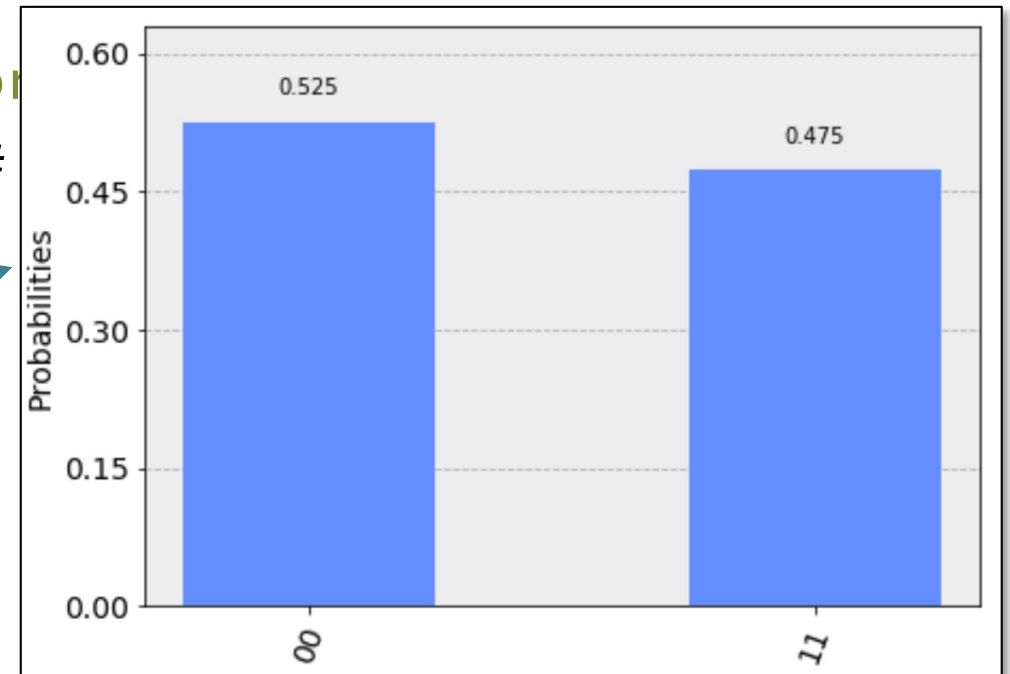
```
backend = Aer.get_backend('qasm_simulator')
```

```
job = execute(qc, backend, shots=512) #
```

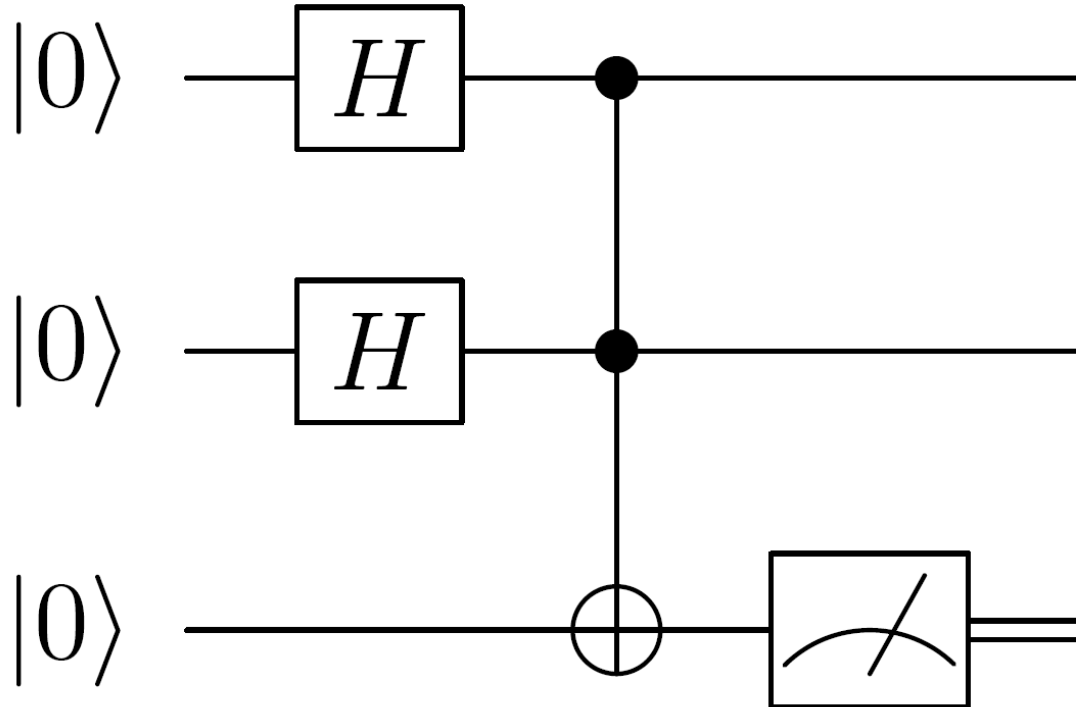
```
result = job.result()
```

```
print(result.get_counts())
```

```
plot_histogram(result.get_counts())
```



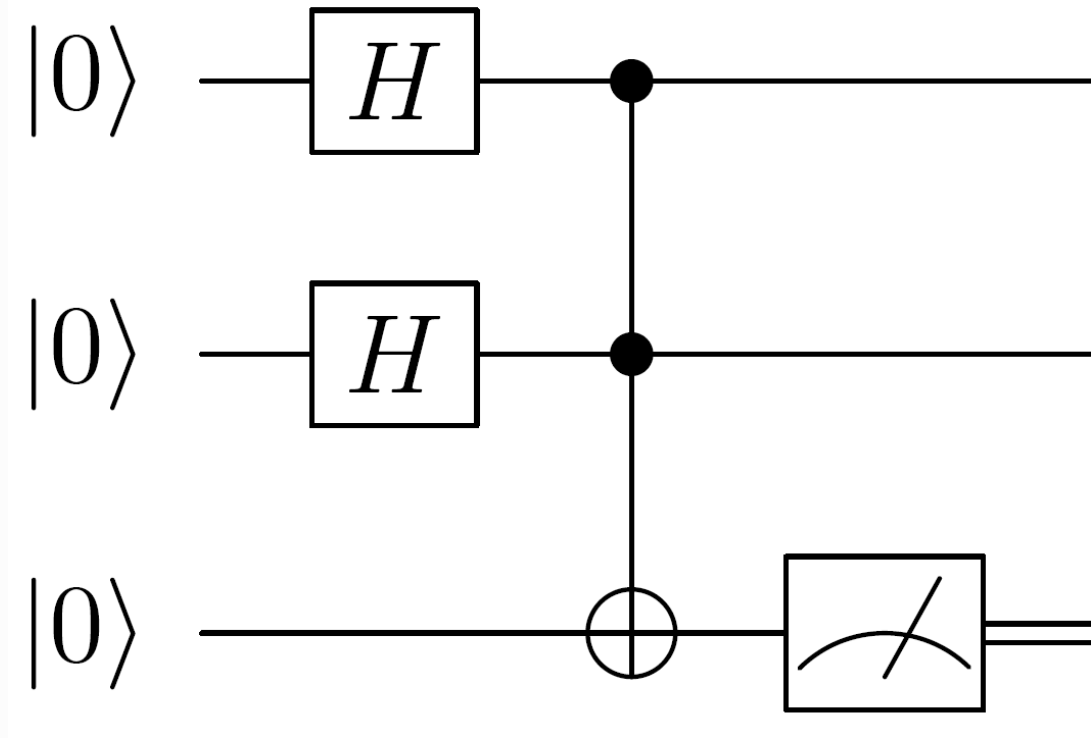
Example 2



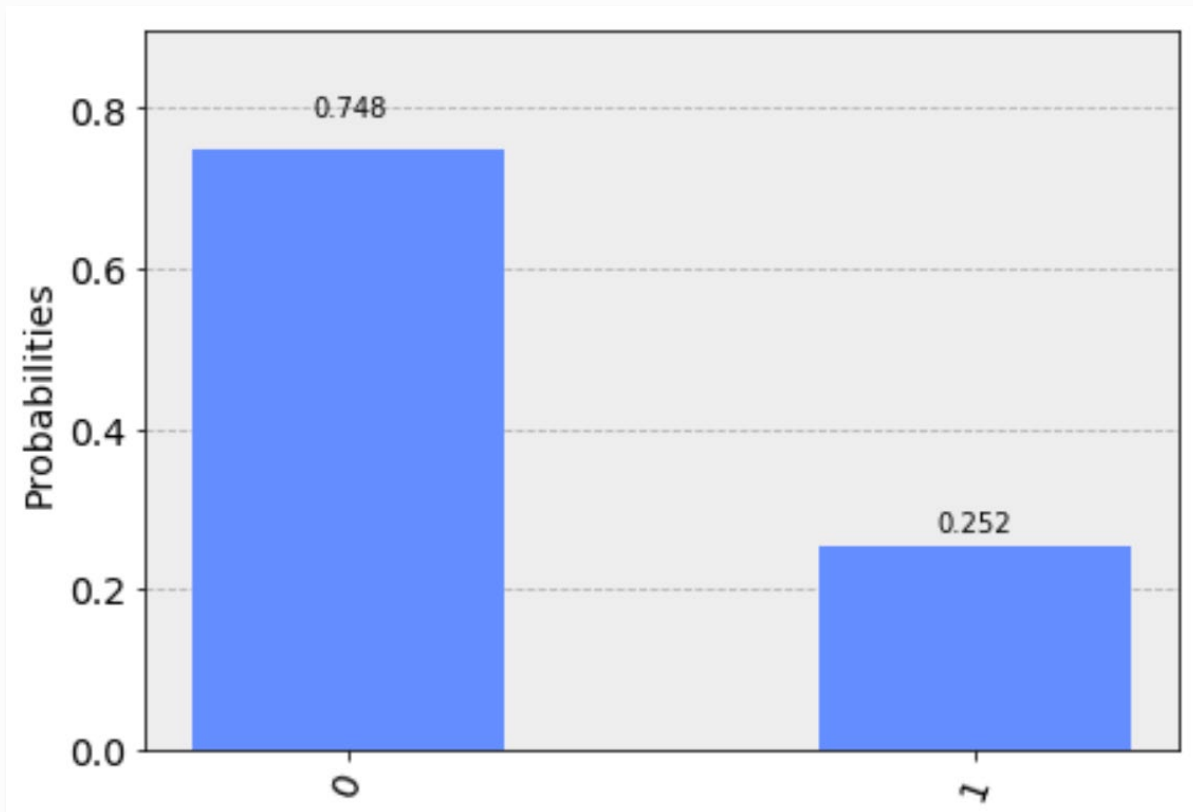
What is the result of this measurement?

After the Toffoli gate, are the qubits entangled?

Example 2



What is the result of this measurement?



Example 3

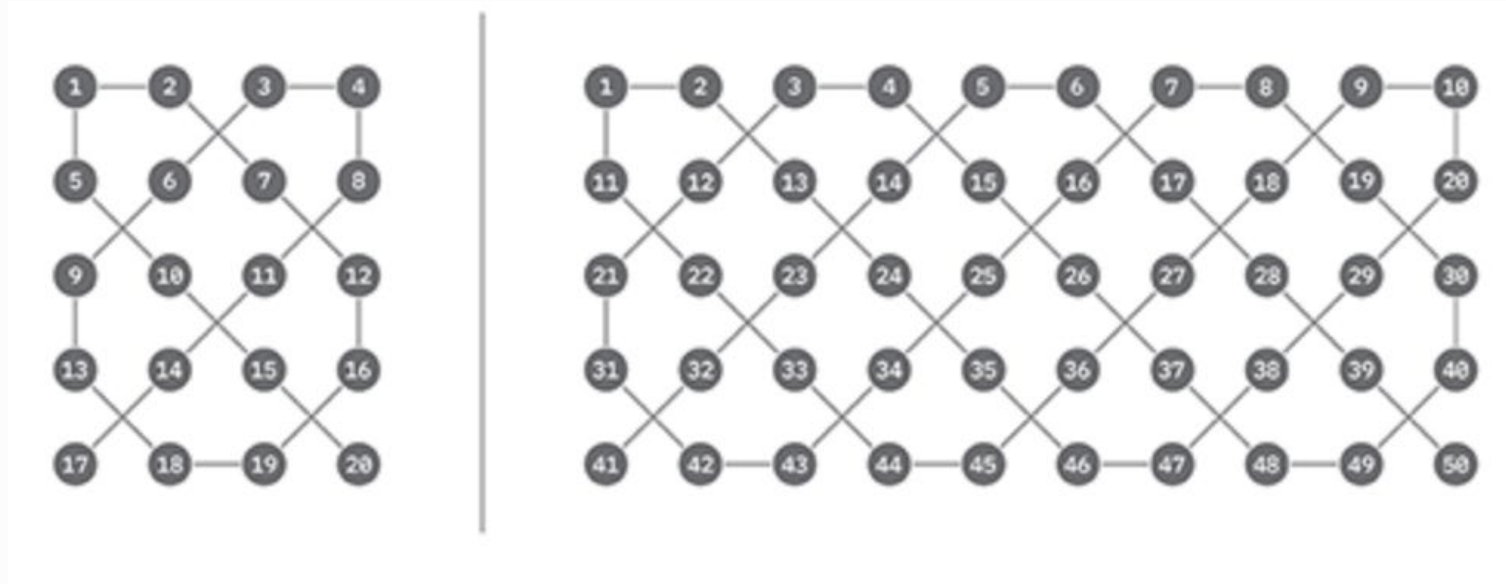
Implement a quantum circuit that checks whether two qubits are equal (in the computational basis).

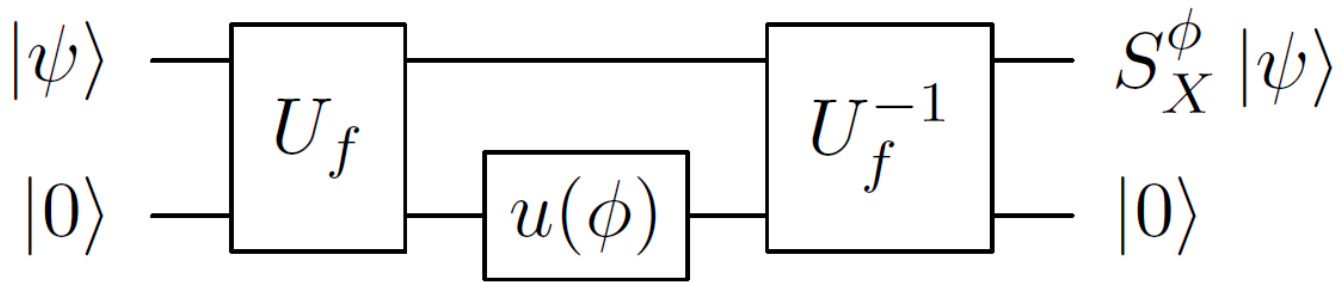
Use qiskit to demonstrate that your circuit works.

Qiskit Summary

- Create quantum and classical registers.
- Create quantum circuit, adding registers.
- Add gates and measurements to circuits.
- Choose backend from provider.
- Execute circuit – compiles circuit to match specifics of backend.
- Get results from job.

Limited Connectivity





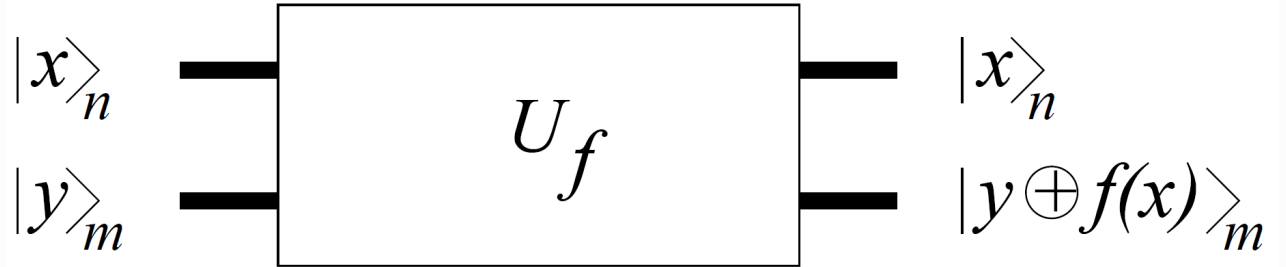
Quantum Algorithms

$$S_X^\phi: \sum_{x=0}^{N-1} a_x |x\rangle \rightarrow \sum_{x \in X} a_x e^{i\phi} |x\rangle + \sum_{x \notin X} a_x |x\rangle$$

Quantum Parallelism

A typical transformation U_f :

$$U_f: |x, 0\rangle \rightarrow |x, f(x)\rangle$$



When this acts on a superposition,
it acts on each element of the superposition:

$$U_f: \sum_x a_x |x, 0\rangle \rightarrow \sum_x a_x |x, f(x)\rangle$$

But if you measure $|x, f(x)\rangle$, you're only going to get one value.
So have to do other things to make this useful.

Quantum Algorithm Strategies

Create superposition of states (quantum parallelism)

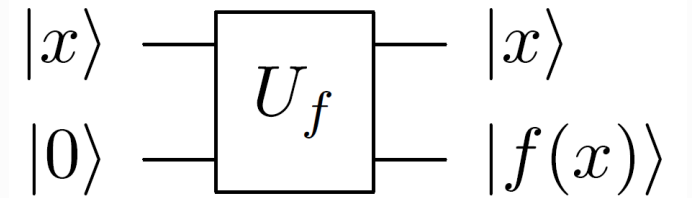
Apply transforms that **amplify** desirable values and **diminish** unwanted values

- Measure to get desired value with high probability.
 - Typically, execute many times (“shots”) to identify high-probability value(s).
- Repeat calculation to learn about relationships among values.
- Measurements can yield information about the properties of values

Deutsch's Algorithm

Problem: Given a Boolean function $f: \mathbb{Z}_2 \rightarrow \mathbb{Z}_2$, determine whether f is constant.

Apply U_f to the input state $|+-\rangle$.



If $f(x)$ is constant, then output is $|+-\rangle$.

If not, output is $|- -\rangle$.

Apply Hadamard to first qubit and measure: 1 if constant, 0 if not.

(Details on next slides.)

Requires only a single call to black box U_f , while classical algorithm requires two calls.

$$\begin{aligned}
U_f |+\rangle |-\rangle &= U_f \left(\frac{1}{2} (|0\rangle + |1\rangle) (|0\rangle - |1\rangle) \right) \\
&= \frac{1}{2} (|0\rangle (|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle) + |1\rangle (|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle)) \\
&= \frac{1}{2} \sum_{x=0}^1 |x\rangle (|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle)
\end{aligned}$$

When $f(x) = 0$, this becomes $\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = |-\rangle$.

When $f(x) = 1$, this becomes $\frac{1}{\sqrt{2}} (|1\rangle - |0\rangle) = -|-\rangle$.

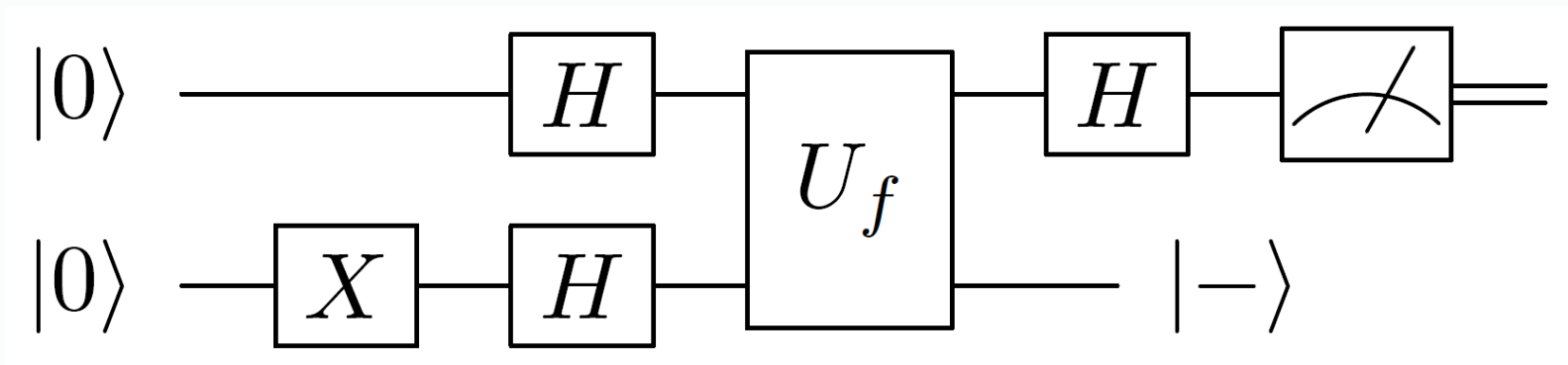
$$U_f |+\rangle |-\rangle = U_f \left(\frac{1}{\sqrt{2}} \sum_{x=0}^1 |x\rangle |-\rangle \right) = \frac{1}{\sqrt{2}} \sum_{x=0}^1 (-1)^{f(x)} |x\rangle |-\rangle$$

$$U_f |+\rangle |-\rangle = U_f \left(\frac{1}{\sqrt{2}} \sum_{x=0}^1 |x\rangle |-\rangle \right) = \frac{1}{\sqrt{2}} \sum_{x=0}^1 (-1)^{f(x)} |x\rangle |-\rangle$$

When $f(x)$ is constant, $(-1)^{f(x)}$ is a meaningless global phase, and the output is $|+\rangle |-\rangle$.

When $f(x)$ is not constant, then $(-1)^{f(x)}$ negates exactly one of the terms, so the output is $|-\rangle |-\rangle$.

By applying a Hadamard gate and measuring the first bit, we get 0 if constant and 1 if not constant.



Selective Phase Change

Problem: Change the phase of terms in a superposition $|\psi\rangle = \sum a_i |i\rangle$, depending on whether i is in a subset X of $\{0, 1, \dots, N - 1\}$ or not. More specifically, find an efficient implementation of the following transform:

$$S_X^\phi: \sum_{x=0}^{N-1} a_x |x\rangle \rightarrow \sum_{x \in X} a_x e^{i\phi} |x\rangle + \sum_{x \notin X} a_x |x\rangle$$

Requires an efficient implementation of U_f for the function $f(x)$ that tests for membership in X :

$$f(x) = \begin{cases} 1 & \text{if } x \in X \\ 0 & \text{otherwise} \end{cases}$$

This type of function is often called an **Oracle**. Used in "black box" algorithms.

First, apply U_f to $|\psi\rangle |0\rangle$.

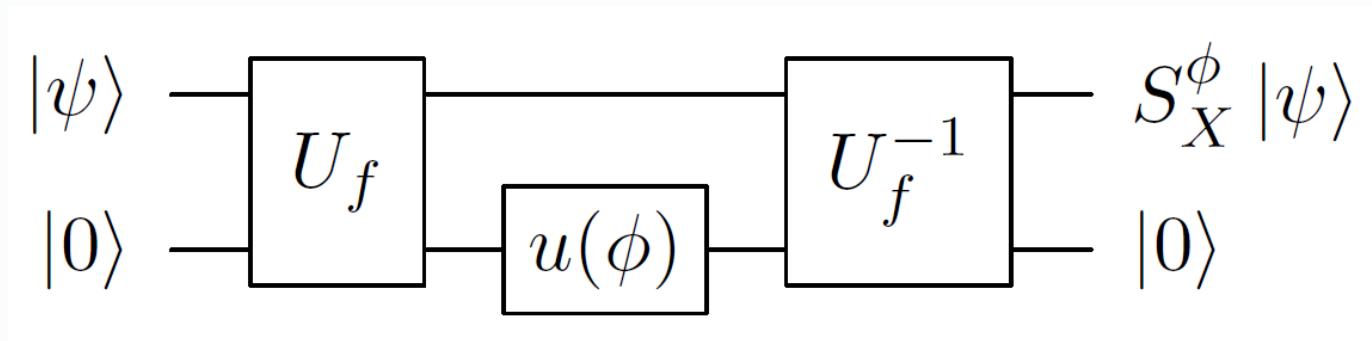
$$\begin{aligned} |\gamma\rangle &= U_f |\psi\rangle |0\rangle = \sum_x a_x |x, f(x)\rangle \\ &= \sum_{x \in X} a_x |x\rangle |1\rangle + \sum_{x \notin X} a_x |x\rangle |0\rangle \end{aligned}$$

Finally, uncompute using U_f^{-1} to remove any entanglement with the output bit.

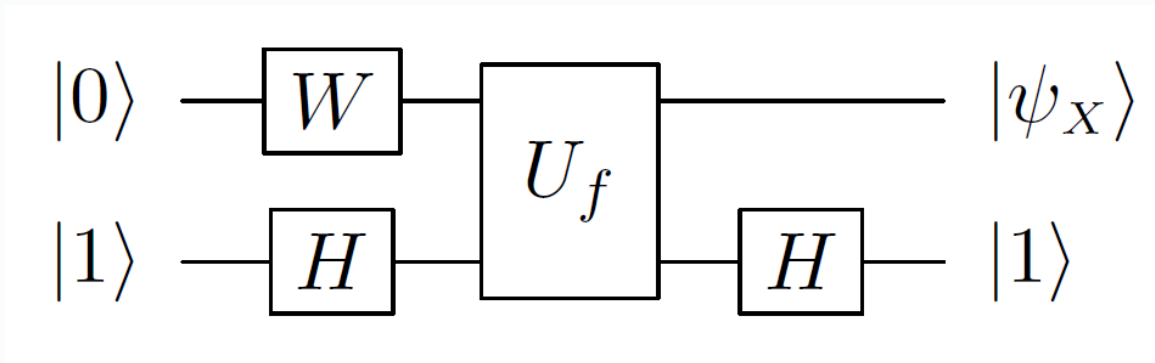


Now, apply the phase change gate $u(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$ to the output qubit. This has no effect on $|0\rangle$ and shifts the phase on $|1\rangle$.

$$\begin{aligned} (I^{\otimes n} \otimes u(\phi)) |\gamma\rangle &= \sum_{x \in X} a_x |x\rangle e^{i\phi} |1\rangle + \sum_{x \notin X} a_x |x\rangle |0\rangle \\ &= \sum_{x \in X} a_x e^{i\phi} |x\rangle |1\rangle + \sum_{x \notin X} a_x |x\rangle |0\rangle \\ &= \left(\sum_{x \in X} a_x e^{i\phi} |x\rangle + \sum_{x \notin X} a_x |x\rangle \right) |f(x)\rangle \\ &= (S_X^\phi |\psi\rangle) |f(x)\rangle \end{aligned}$$



Special case of π :



$$|\psi_x\rangle = \frac{1}{\sqrt{N}} \sum_x (-1)^{f(x)} |x\rangle$$

Qiskit Example

Create a circuit that generates an equal superposition of two qubits, except the sign (phase) is flipped when the two qubits are equal.

- Sign flip is phase change of π .
- Use the "equal" circuit from Example 3.

How do you see the phase shift in qiskit?

Deutsch-Josza Algorithm

Problem: Given an n -bit Boolean function (mapping n bits to 1) that is known to be either constant or balanced, determine whether it is balanced or constant. A function is “balanced” if an equal number of input values return 0 and 1.

Apply phase shift of π to negate elements where $f(x) = 1$.

Apply Walsh-Hadamard to the result.

For constant f , the final output is $|00 \dots 0\rangle$ with probability 1.

For balanced f , the final output is non-zero with probability 1.

(Details on next slides.)

Requires only a single call to black box U_f , while classical algorithm requires at least $2^{n-1} + 1$ calls.

Background: Hamming Distance

The Hamming distance $d_H(x, y)$ between two bit strings x and y is the number of bits in which the two strings differ.

The Hamming weight $d_H(x)$ of a bit string x is the number of 1 bits.

For two bit strings x and y , the operator $x \cdot y$ gives the number of common 1 bits.

Some interesting notes:

$$x \cdot y = d_H(x \wedge y)$$

$$\sum_{x=0}^{2^n-1} (-1)^{x \cdot x} = 0$$

$$\sum_{x=0}^{2^n-1} (-1)^{x \cdot y} = \begin{cases} 2^n & \text{if } y = 0 \\ 0 & \text{otherwise} \end{cases}$$

More on Walsh-Hadamard

$$W |0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

$$\begin{aligned} W |r\rangle &= (H \otimes \cdots \otimes H)(|r_{n-1}\rangle \otimes \cdots \otimes |r_0\rangle) \\ &= \frac{1}{\sqrt{2^n}}(|0\rangle + (-1)^{r_{n-1}} |1\rangle) \otimes \cdots \otimes (|0\rangle + (-1)^{r_0} |1\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} (-1)^{s_{n-1}r_{n-1}} |s_{n-1}\rangle \otimes \cdots \otimes (-1)^{s_0r_0} |s_0\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} (-1)^{s \cdot r} |s\rangle \end{aligned}$$

First, prepare a complete superposition, and then apply the phase shift algorithm to negate the terms corresponding to vectors $|x\rangle$ where $f(x) = 1$.

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{f(i)} |i\rangle$$

Next, apply the Walsh-Hadamard transform to obtain:

$$|\phi\rangle = \frac{1}{N} \sum_{i=0}^{N-1} \left((-1)^{f(i)} \sum_{j=0}^{N-1} (-1)^{i \cdot j} |j\rangle \right)$$

$$|\phi\rangle = \frac{1}{N} \sum_{i=0}^{N-1} \left((-1)^{f(i)} \sum_{j=0}^{N-1} (-1)^{i \cdot j} |j\rangle \right)$$

For constant f , the $(-1)^{f(i)} = (-1)^{f(0)}$ is simply a global phase, and the state $|\phi\rangle$ is $|0\rangle$:

$$\begin{aligned} |\phi\rangle &= (-1)^{f(0)} \frac{1}{N} \sum_j \left(\sum_i (-1)^{i \cdot j} \right) |j\rangle \\ &= (-1)^{f(0)} \frac{1}{N} \sum_i (-1)^{i \cdot 0} |0\rangle \\ &= (-1)^{f(0)} |0\rangle \end{aligned}$$

because $\sum_i (-1)^{i \cdot j} = 0$ for $j \neq 0$.

$$|\phi\rangle = \frac{1}{N} \sum_{i=0}^{N-1} \left((-1)^{f(i)} \sum_{j=0}^{N-1} (-1)^{i \cdot j} |j\rangle \right)$$

For balanced f ,

$$|\phi\rangle = \frac{1}{N} \sum_j \left(\sum_{i \in X_0} (-1)^{i \cdot j} - \sum_{i \notin X_0} (-1)^{i \cdot j} \right) |j\rangle, \text{ where } X_0 = \{x | f(x) = 0\}$$

In this case, when $j = 0$, the amplitude is zero.

Therefore, measuring $|\phi\rangle$ in the standard basis will return a non-zero j with probability 1.

Qiskit Example

Use the previous example (phase shift) to demonstrate Deutsch-Josza algorithm.

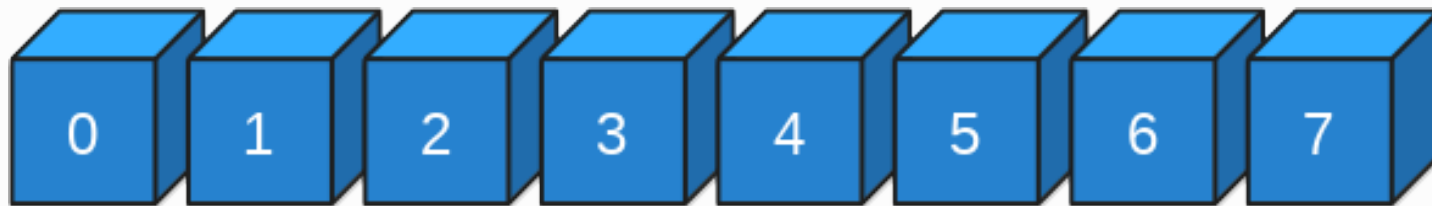
- isEqual is a balanced function

Summary

- Any efficient reversible classical circuit can be efficiently implemented as a quantum circuit.
 - Use inverse function to reduce space and unentangle temporary bits.
- For quantum advantage, add some non-classical operations.
 - E.g, phase change.
- Are these algorithms really useful?
 - Perhaps not directly, but they illustrate ways in which quantum computing may have an advantage over classical computing.

Grover's Algorithm

- Find one input x for which $f(x)$ is 1 (out of $N = 2^n$ possible inputs).
 - Which box has the prize?
- Classically, have to open all N boxes in the worst case.
- With quantum, can solve in \sqrt{N} steps.
- Let's see how...



Grover's Algorithm

- Two transforms:
 - S_X^π : change phase (by π) for the input for which $x \in X$
 - We know how to do this already.
 - Grover's diffusion operator
 - If $|s\rangle = W|0\rangle = \frac{1}{\sqrt{N}} \sum_x a_x |x\rangle$ (Walsh-Hadamard), then
 - $U_s = 2|s\rangle\langle s| - I = -W S_0^\pi W$, which performs
 - $\sum_x a_x |x\rangle \rightarrow \sum_x (2A - a_x) |x\rangle$
(rotates amplitudes around the mean).



Grover's Algorithm

- Do this \sqrt{N} times
- Measure
- Very likely to measure correct result
- From $O(N)$ to $O(\sqrt{N})$



Shor's Factoring Algorithm

- Given N is a product of two primes, p and q
 - We have N , want to find p and q
- For some number a that is not divisible by p or q , the following sequence repeats itself with a period r .

$$\{a^1 \bmod N, a^2 \bmod N, a^3 \bmod N, a^4 \bmod N, \dots\}$$

- As Euler discovered (~ 1760), r always divides $(p - 1)(q - 1)$.

Shor's Factoring Algorithm

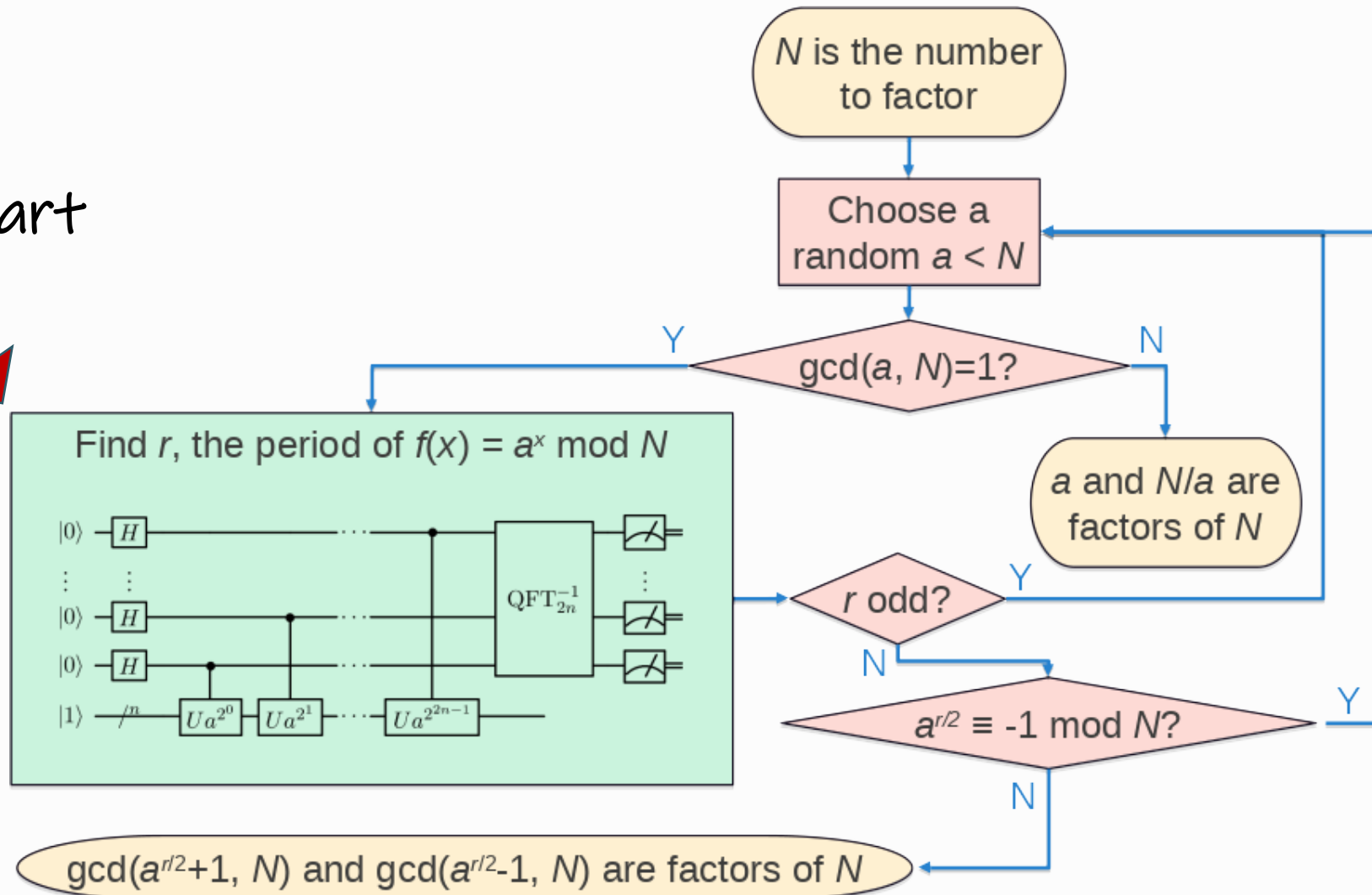
- Example: Let $N=15$ (3×5) and $a=2$.
 $\{2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8, 1, \dots\} \Rightarrow r = 4.$

Lo and behold, 4 divides $(3-1)(5-1) = 8$.

- If we can find r , it's pretty easy to find p and q .
- However, finding r is very challenging for a classical computer:
 $O(\exp(n^{1/3}))$
- Quantum: $O(n^2 \log n \log \log n)$

Shor's Factoring Algorithm

Quantum part



Backup Slides

Simon's Algorithm

Problem: Given a 2-to-1 function f , such that $f(x) = f(x \oplus a)$, find the hidden string a .

Create superposition $|x\rangle|f(x)\rangle$

Measure the right part, which projects the left state into $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle)$.

Apply Walsh-Hadamard. (Details next slide.)

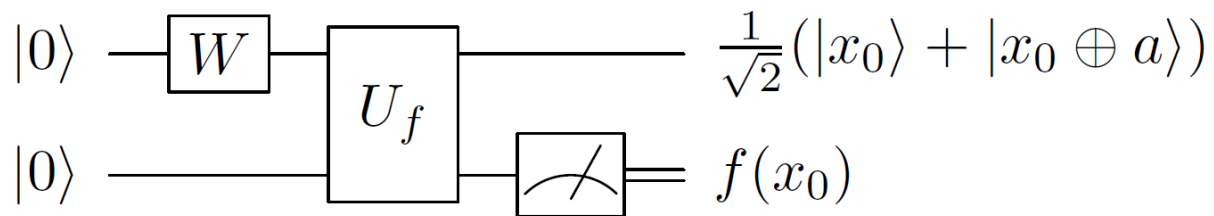
Measurement yields a random y such that $y \cdot a = 0 \pmod{2}$.

Computation is repeated until n independent equations – about $2n$ times.

Solve for a in $O(n^2)$ steps.

Requires $O(n)$ calls to U_f , followed by $O(n^2)$ steps to solve for a .

Classical approach requires $O(2^{n/2})$ calls to f .



$$\begin{aligned}
 W\left(\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle)\right) &= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2^n}} \sum_y ((-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus a) \cdot y}) |y\rangle \right) \\
 &= \frac{1}{\sqrt{2^{n+1}}} \sum_y (-1)^{x_0 \cdot y} (1 + (-1)^{a \cdot y}) |y\rangle \\
 &= \frac{2}{\sqrt{2^{n+1}}} \sum_{y \cdot a \text{ even}} (-1)^{x_0 \cdot y} |y\rangle
 \end{aligned}$$

Measurement yields random y such that $y \cdot a = 0 \pmod{2}$, so the unknown bits of a_i of a must satisfy this equation:

$$y_0 \cdot a_0 \oplus \cdots \oplus y_{n-1} \cdot a_{n-1} = 0$$

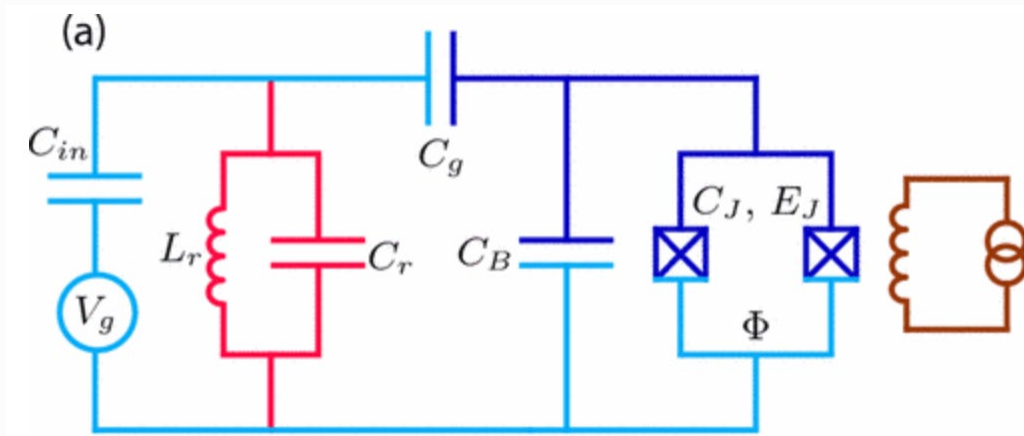
Computation is repeated until n linearly independent equations have been found. Each time, the resulting equation has at least a 50% probability of being linearly independent of the previous equations. After repeating $2n$ times, there is a 50% chance that n linearly independent equations have been found. These equations can be solved to find a in $O(n^2)$ steps.

A word about implementation...

Quotes from IBM Q material

The qubit we use is a **fixed-frequency superconducting transmon** qubit. It is a **Josephson-junction-based** qubit that is insensitive to charge noise.

The devices are made on silicon wafers with superconducting metals such as **niobium** and **aluminum**.



Koch, et al.
Phys. Rev. A **76**, 042319 –Oct 2007

Quantum gates are performed by sending **electromagnetic impulses at microwave frequencies** to the qubits through coaxial cables. These electromagnetic pulses have a particular **duration, frequency, and phase** that determine the **angle of rotation** of the qubit state around a particular axis of the Bloch sphere.

A word about implementation...

Quotes from IBM Q material

Two-qubit gates typically require tuning to calibrate the interaction between the two qubits during the gate duration, and minimizing the interaction at any other time. Since our qubits of choice are fixed-frequency transmons, we cannot tune the interaction by bringing them closer in frequency during the two-qubit gate. Instead, we exploit the **cross-resonance effect**, by **driving one of the qubits (called control) with a microwave pulse tuned at the frequency of the second qubit (called target)**. By doing this, we can actively increase the strength of the coupling between them. The nature of the cross-resonance effect also allows us to **perform rotations in the target qubit conditioned on the state of the control qubit**, a key characteristic of the CNOT operation required for a universal quantum gate set.

