Quantum Software Development

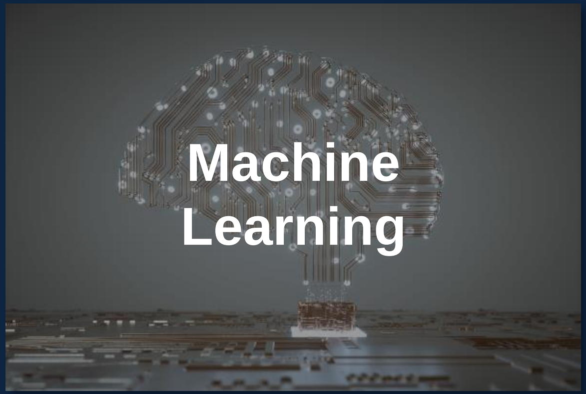
Lecture 7: Basic Quantum Algorithms, Hybrid Algorithms

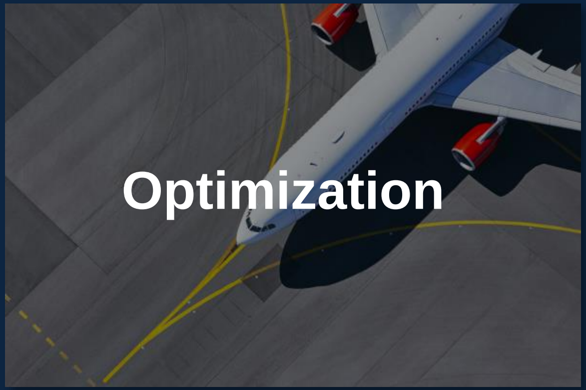
March 6, 2024

Tufts University | School of Engineering
MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®

Basic Quantum Algorithms

Quantum algorithms use interference to provide a computational advantage for solving certain problems.





The Deutsch-Jozsa problem is designed to be hard for classical computers but easy for quantum computers.

Suppose you're given a black-box function f that outputs 0 or 1 based on a binary input, and you are guaranteed that it is either:

- **Constant** – it outputs the same value for all possible input combinations, or
- **Balanced** – it outputs 0 for exactly half of the input combinations and 1 for the other half.

How do you determine which one it is?

How quickly could you perform the necessarily computation on a classical computer?

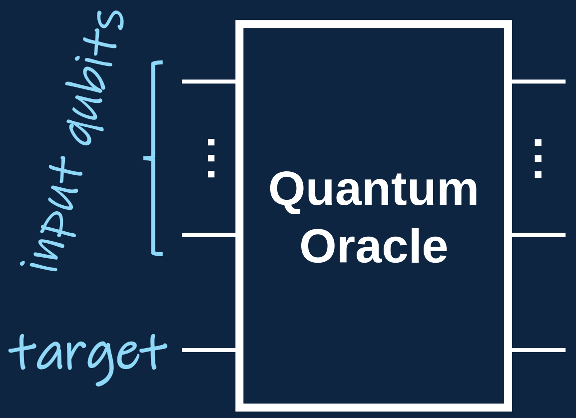
Constant Example

x	$f(x)$
00	1
01	1
10	1
11	1

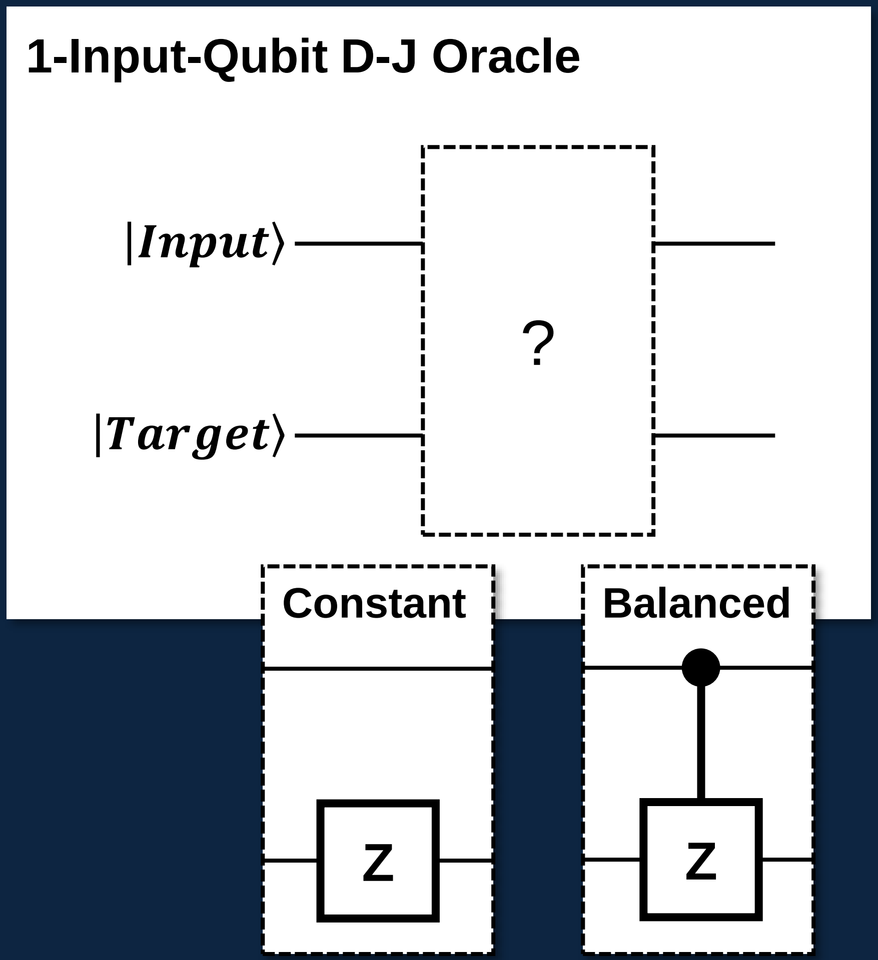
Balanced Example

x	$f(x)$
00	0
01	0
10	1
11	1

For a QC to solve the D-J problem, the black-box function must be provided as a quantum oracle.



A typical quantum oracle phase-flips the target based on the input.



Constant

$\|x\rangle$	$(-1)^{f(x)}$
$\|0\rangle$	-1
$\|1\rangle$	-1

Balanced

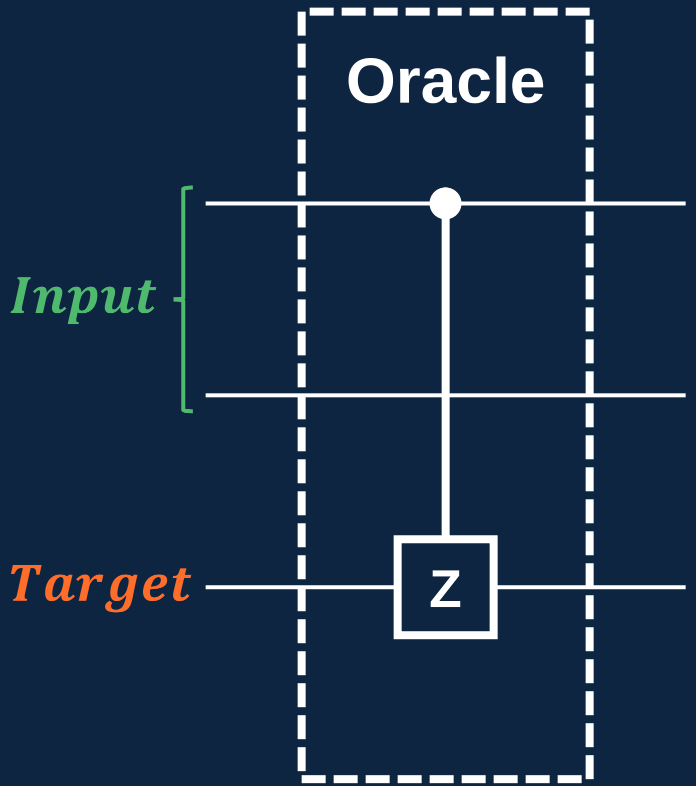
$\|x\rangle$	$(-1)^{f(x)}$
$\|0\rangle$	1
$\|1\rangle$	-1

What does a balanced oracle look like for 2 input qubits?

Balanced Example

x	$f(x)$
00	0
01	0
10	1
11	1

$\|x\rangle$	$(-1)^{f(x)}$
$\|00\rangle$	1
$\|01\rangle$	1
$\|10\rangle$	-1
$\|11\rangle$	-1



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

$$\downarrow$$

$$\frac{1}{\sqrt{2}}(|10,0\rangle - |10,1\rangle)$$

$f(10) = 1$

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

$$\downarrow$$

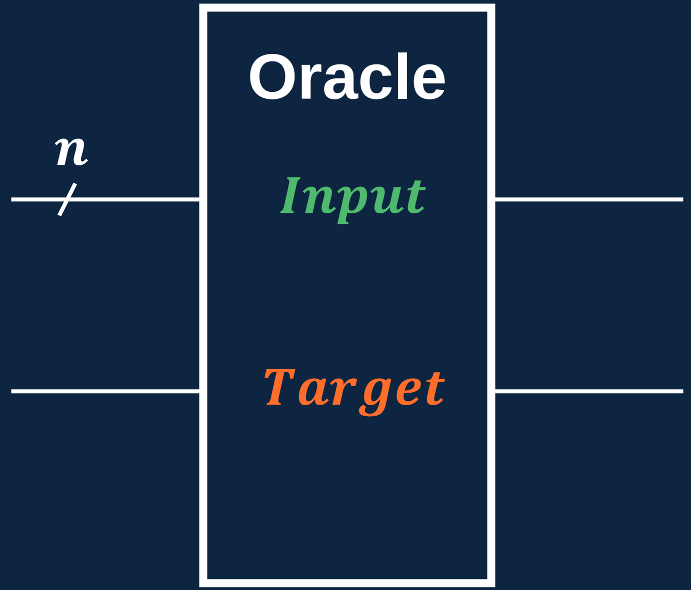
$$\frac{1}{\sqrt{2}}(|00,1\rangle - |10,1\rangle)$$

$f(00) \neq f(10)$

What happens when a uniform superposition is input into the oracle, with the target qubit a $|1\rangle$?

$$\frac{1}{\sqrt{N}}\left(|0\rangle + |1\rangle + \cdots + |N-1\rangle\right)$$

$$|1\rangle$$



$$\frac{1}{\sqrt{N}}\begin{pmatrix} (-1)^{f(0)}|0,1\rangle + \\ (-1)^{f(1)}|1,1\rangle + \\ \ldots + \\ (-1)^{f(N-1)}|N-1,1\rangle \end{pmatrix}$$

$$= \sum_{x=0}^{N-1} (-1)^{f(x)}|x\rangle \otimes |1\rangle$$

Due to phase kickback, the oracle flips the phase of the input terms where $f(x) = 1$.

When the Hadamard transform is applied on the output, the interference pattern is distinct for each oracle type.

bitwise dot product

$$\frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & -1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & -1 & \cdots & 1 \end{bmatrix} \cdot \frac{1}{\sqrt{N}}\begin{bmatrix} (-1)^{f(0)} \\ (-1)^{f(1)} \\ \vdots \\ (-1)^{f(N-1)} \end{bmatrix} = \frac{1}{N}\left(\boxed{\sum_{x=0}^{N-1} (-1)^{f(x)}} (-1)^{x \cdot 0} |0\rangle \right) + \cdots$$

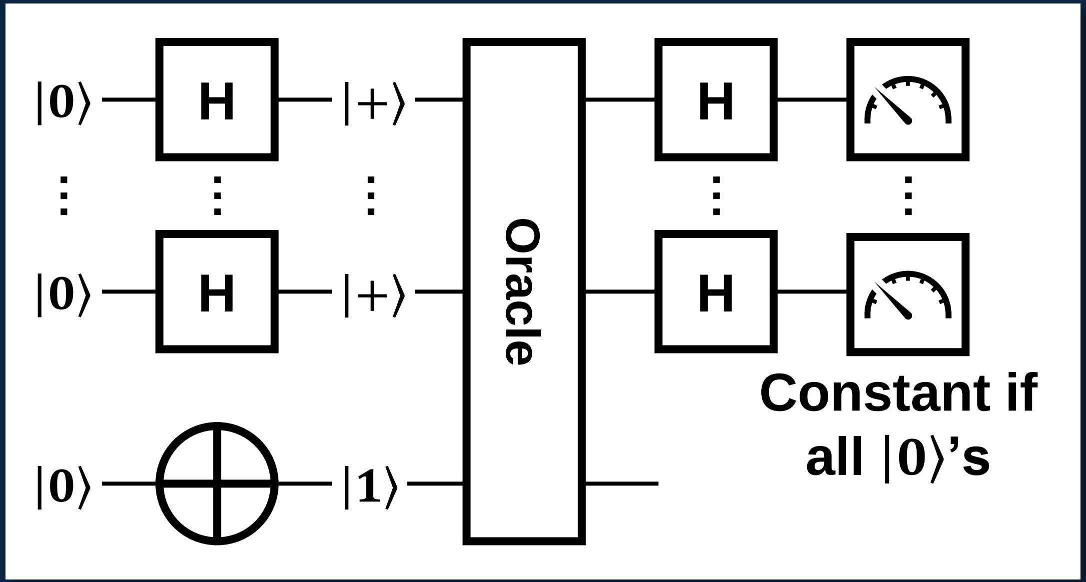
$$= \begin{cases} 0: & \text{if balanced} \\ N \text{ or } (-N): & \text{if constant} \end{cases}$$

A constant oracle always causes constructive interference on the $|0\rangle$ term.
A balanced oracle always causes destructive interference on the $|0\rangle$ term.

MITRE

The Deutsch-Jozsa algorithm demonstrates the potential of quantum computation.

1. Allocate n input qubits and $\mathbf{1}$ target qubit
2. Apply H to each input and X to target
3. Apply the oracle under test
4. Apply H to each input (again)
5. Measure the input qubits
6. If all $|\mathbf{0}\rangle$'s are measured, the oracle is constant; otherwise, it is balanced



The quantum solution reduces the computational complexity from $O\left(2^{n-1}\right)$ to $O(1)$!!

How could a quantum computer be used to solve the Bernstein-Vazirani problem?

Suppose you're given a black-box function f that outputs the bitwise dot product of the input x and some secret bitstring s.

How do you find out what s is?

How quickly could you perform the necessary computation on a classical computer?

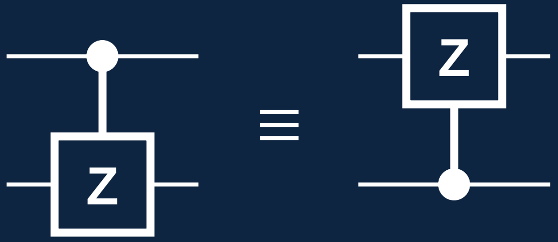
How quickly could you perform the necessary computation on a quantum computer if f is provided as a quantum oracle?

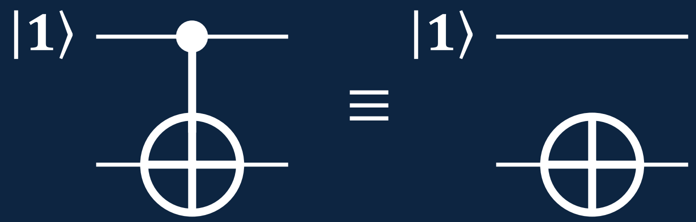
Hint: Try the same setup as the Deutsch-Jozsa algorithm.

$s = 110$ Example

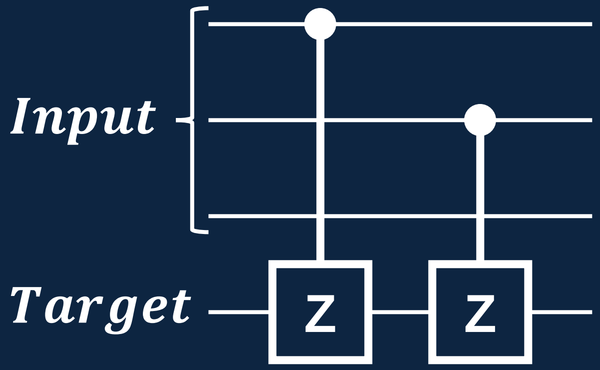
x	$f(x) = x \cdot s$
000	0
001	0
010	1
011	1
100	1
101	1
110	0
111	0

Two quantum circuits are equivalent if they implement the same matrix transformation.



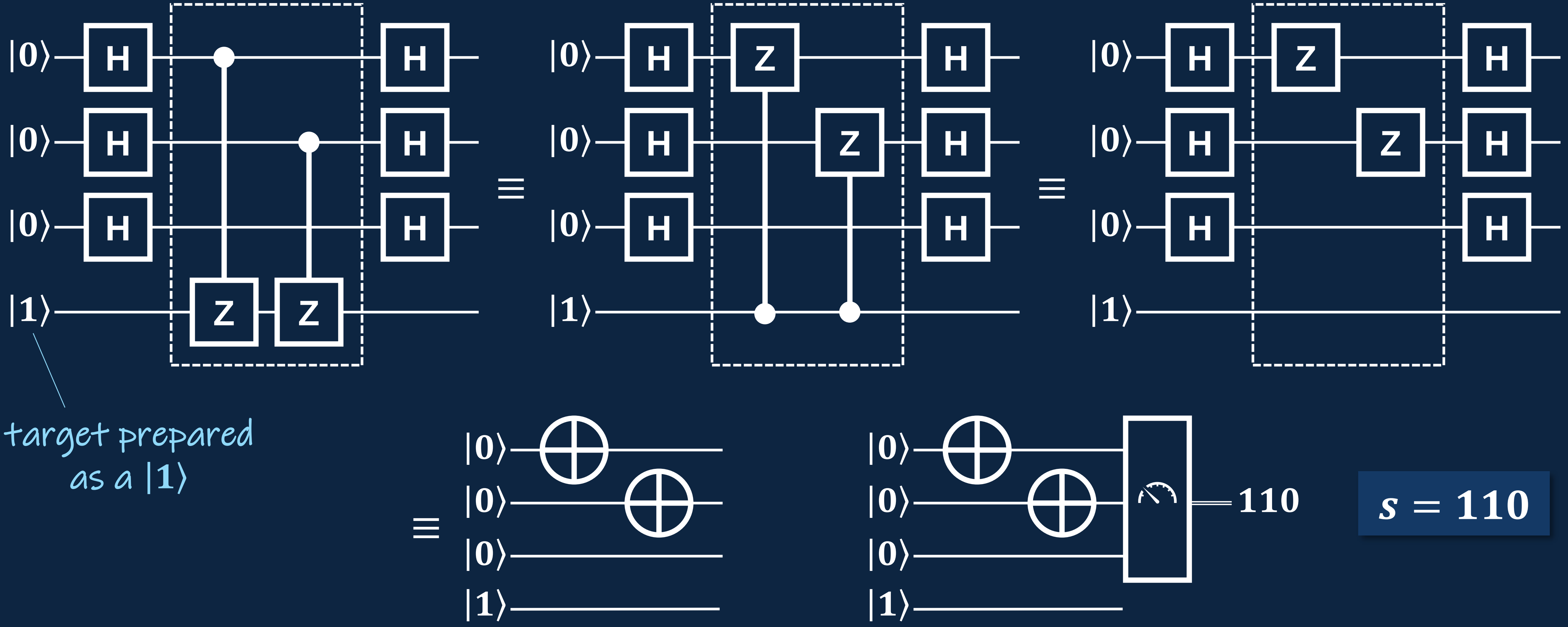


B-V phase-flip oracle, $s = 110$



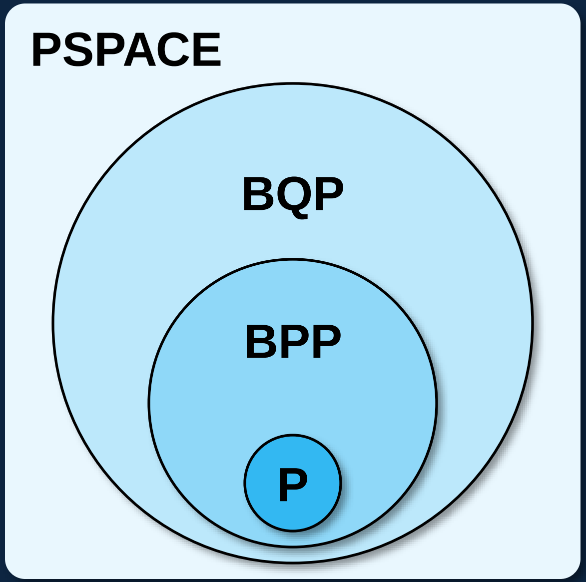
By definition, all oracle implementations are equivalent circuits.

Applying the H-transform before and after the B-V oracle results in an equivalent circuit that exposes s.



MITRE

In computational complexity theory, the class of tractable problems for quantum computers is called BQP.



PSPACE = Polynomial space (memory)

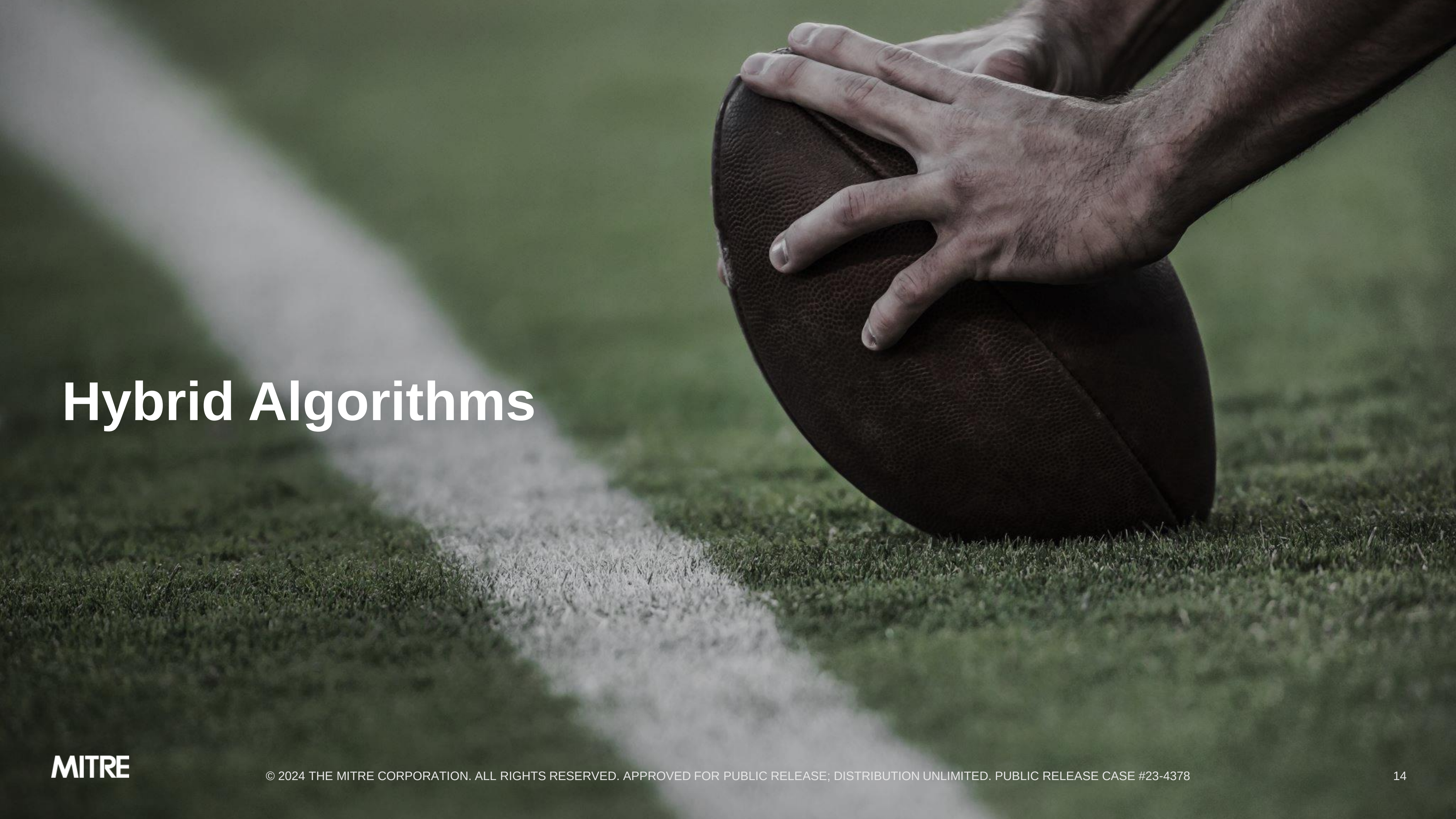
BQP = Bounded-error Quantum Polynomial time

BPP = Bounded-error Probabilistic Polynomial time

P = Polynomial time

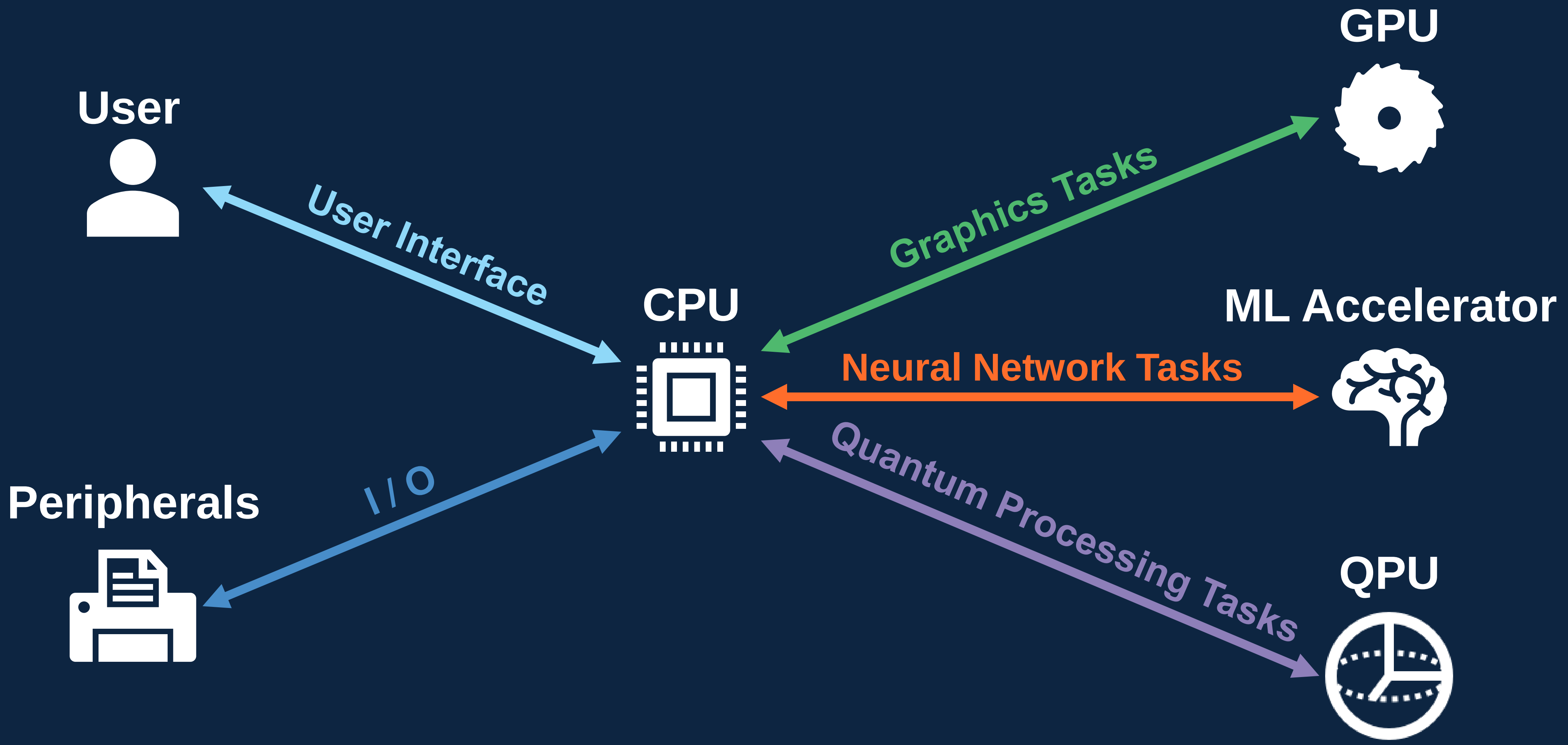
Bernstein-Vazirani shows BQP $\supseteq$ BPP.

The relationship between BQP and NP is an open problem.



Hybrid Algorithms

In computer architecture, a quantum computer is like a coprocessor or hardware accelerator.



Simon's problem can be solved efficiently with a hybrid algorithm, i.e., with a quantum subroutine.

Suppose you're given a black-box function f with input and output of n bits.

You are guaranteed that f is 2-to-1; for every possible output, there are exactly 2 inputs that produce it.

Also, the pairs of inputs that produce the same output, when XOR'd together, always produce the same value s. In other words, $f(x_1) = f(x_2) \Rightarrow x_1 \oplus x_2 = s$.

How do you find out what s is?

Left-shift-by-1, $n = 3$

$\boldsymbol{x}$	000	001	010	011	100	101	110	111
$\boldsymbol{f(x)}$	000	010	100	110	000	010	100	110

$\boldsymbol{s = 100}$

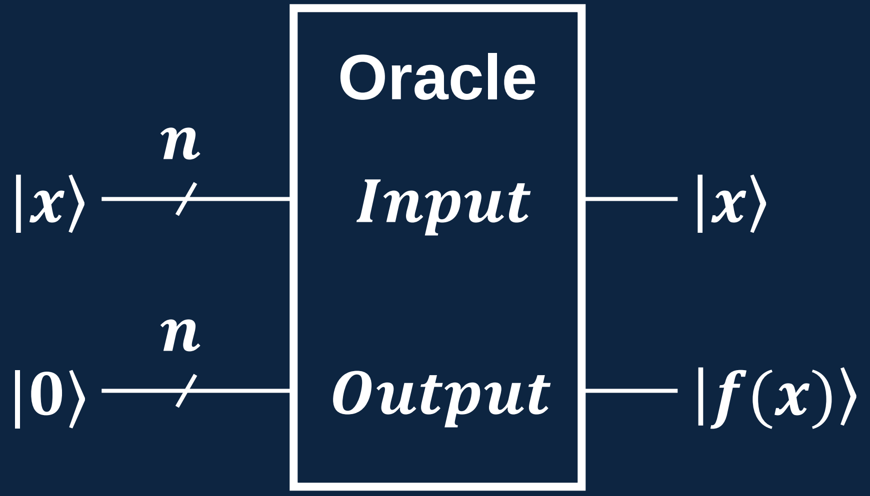
What is the secret string s for the function below?

x	000	001	010	011	100	101	110	111
$f(x)$	101	010	000	110	000	110	101	010

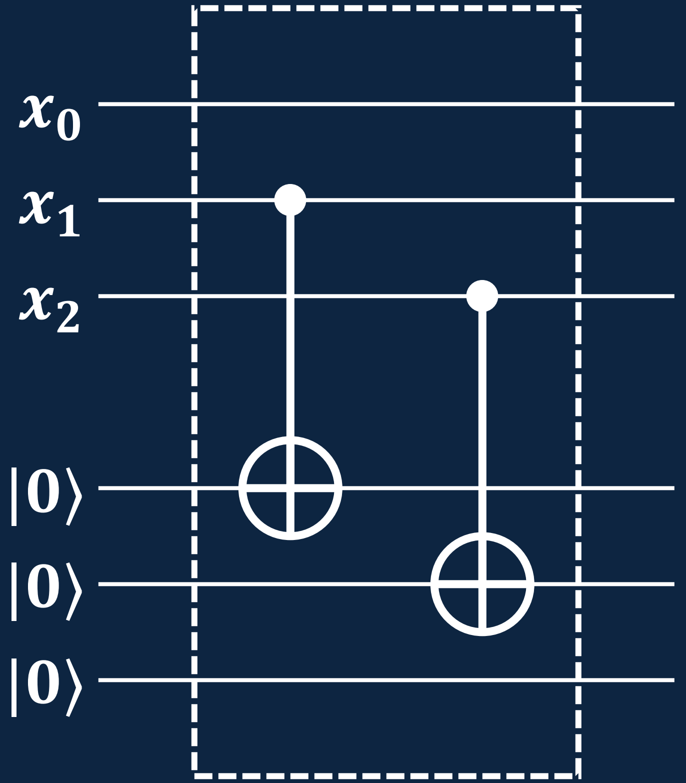
$$s = 000 \oplus 110 = 110$$

To compute s classically, we must find at least one pair of inputs that produce the same output. For an n-bit function, this is $O(2^{n-1})$.

A quantum oracle for Simon’s problem flips bit values in an output register based the input register value.



Left-shift-by-1, $n = 3$



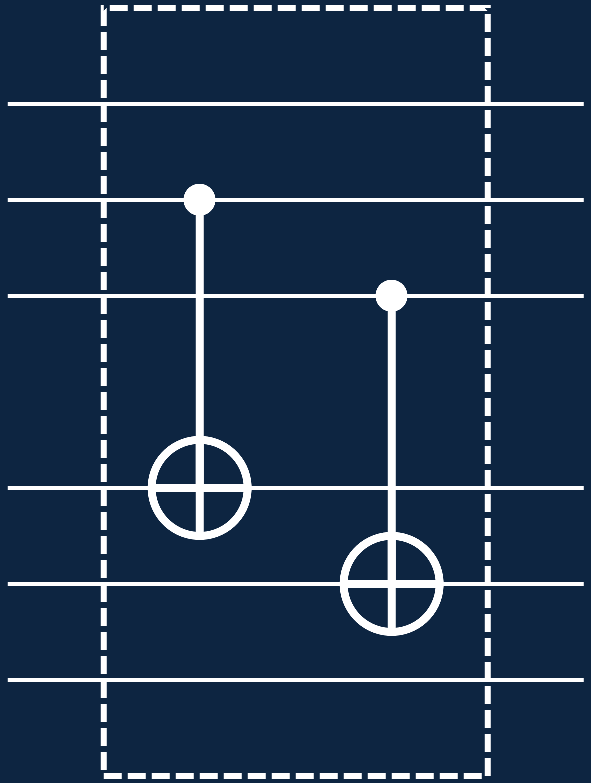
$\|x_0x_1x_2\rangle$	$\|f(x)\rangle$
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|010\rangle$
$\|010\rangle$	$\|100\rangle$
$\|011\rangle$	$\|110\rangle$
$\|100\rangle$	$\|000\rangle$
$\|101\rangle$	$\|010\rangle$
$\|110\rangle$	$\|100\rangle$
$\|111\rangle$	$\|110\rangle$

MITRE

How does the oracle transform a uniform superposition in the input register?

Left-shift-by-1, $n = 3$

$$\frac{1}{\sqrt{8}}\sum_{k=0}^{7}|k\rangle \qquad |0\rangle \quad |0\rangle \quad |0\rangle$$



$$\frac{1}{\sqrt{8}}\begin{pmatrix} |000,000\rangle + \\ |001,010\rangle + \\ |010,100\rangle + \\ |011,110\rangle + \\ |100,000\rangle + \\ |101,010\rangle + \\ |110,100\rangle + \\ |111,110\rangle \end{pmatrix} = \frac{1}{\sqrt{8}}\begin{pmatrix} (|000\rangle + |100\rangle) \otimes |000\rangle + \\ (|001\rangle + |101\rangle) \otimes |010\rangle + \\ (|010\rangle + |110\rangle) \otimes |100\rangle + \\ (|011\rangle + |111\rangle) \otimes |110\rangle \end{pmatrix}$$

What happens if we apply a Hadamard transform to the input register after applying the oracle?

$$H_{Input}^{\otimes 3} \cdot \frac{1}{\sqrt{8}} \begin{pmatrix} (|000\rangle + |100\rangle) \otimes |000\rangle + \\ (|001\rangle + |101\rangle) \otimes |010\rangle + \\ (|010\rangle + |110\rangle) \otimes |100\rangle + \\ (|011\rangle + |111\rangle) \otimes |110\rangle \end{pmatrix} = \frac{1}{\sqrt{8}} \begin{pmatrix} H^{\otimes 3}(|000\rangle + |100\rangle) \otimes |000\rangle + \\ H^{\otimes 3}(|001\rangle + |101\rangle) \otimes |010\rangle + \\ H^{\otimes 3}(|010\rangle + |110\rangle) \otimes |100\rangle + \\ H^{\otimes 3}(|011\rangle + |111\rangle) \otimes |110\rangle \end{pmatrix}$$

Same terms

$$H^{\otimes 3}(|000\rangle + |100\rangle) = \frac{1}{\sqrt{8}} \left(\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \right) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$H^{\otimes 3}(|001\rangle + |101\rangle) = \frac{1}{\sqrt{8}} \left(\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \right) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

After the Hadamard transform, the input register only contains values whose bitwise dot product with s is 0.

$$\frac{1}{\sqrt{8}}\begin{pmatrix} H^{\otimes 3}(|000\rangle + |100\rangle) \otimes |000\rangle + \\ H^{\otimes 3}(|001\rangle + |101\rangle) \otimes |010\rangle + \\ H^{\otimes 3}(|010\rangle + |110\rangle) \otimes |100\rangle + \\ H^{\otimes 3}(|011\rangle + |111\rangle) \otimes |110\rangle \end{pmatrix} = \frac{1}{\sqrt{16}}\begin{pmatrix} (|000\rangle + |001\rangle + |010\rangle + |011\rangle) \otimes |000\rangle + \\ (|000\rangle - |001\rangle + |010\rangle - |011\rangle) \otimes |010\rangle + \\ (|000\rangle + |001\rangle - |010\rangle - |011\rangle) \otimes |100\rangle + \\ (|000\rangle - |001\rangle - |010\rangle + |011\rangle) \otimes |110\rangle \end{pmatrix}$$

$$s = 100$$

$$000 \cdot 100 = 0$$

$$001 \cdot 100 = 0$$

$$010 \cdot 100 = 0$$

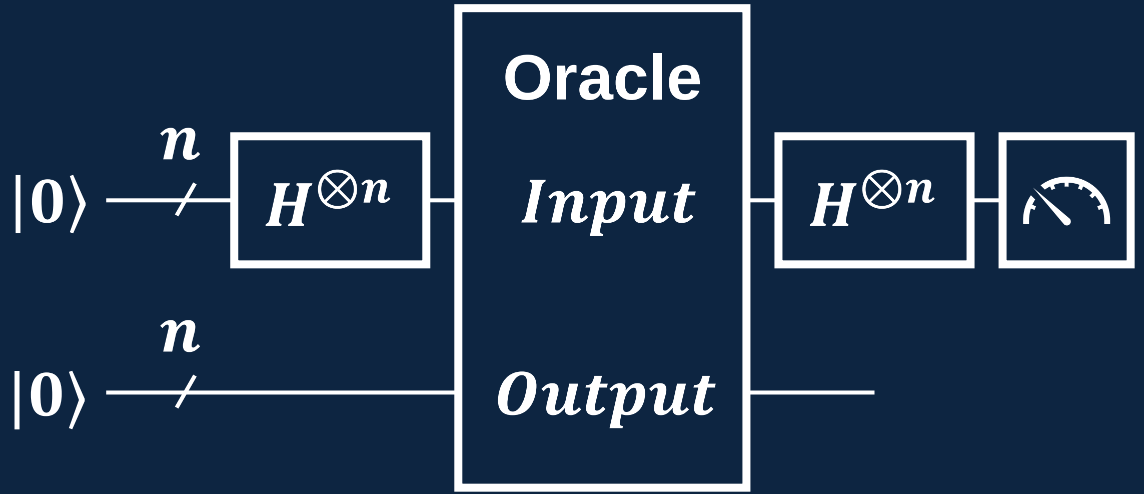
$$011 \cdot 100 = 0$$

If the input register is measured, we're guaranteed to get a value x such that $x \cdot s = 0$. With $n - 1$ linear independent x values, the system of equations can be solved for s.

Simon's Algorithm

Steps 1 and 2 are both $O(n)$.

1. Run the quantum subroutine until $n - 1$ linearly independent bitstrings are found:
   a. Apply H to each qubit in the input register.
   b. Apply the quantum oracle.
   c. Apply H to each qubit in the input register.
   d. Measure the input register.



2. We now have a system of $n - 1$ equations of the form $x \cdot s = 0$. Solve for s with mod-2 Gaussian elimination.

Try the quantum subroutine in Quirk.

- Go to https://algassert.com/quirk
- Build the quantum subroutine with the left-shift-by-1 oracle.
- How do you explain the results?