Quantum Circuits

EE599-001 & EE699-010, Spring 2026

Hank Dietz

http://aggregate.org/hankd/

University of Kentucky

Why Quantum Algorithms?

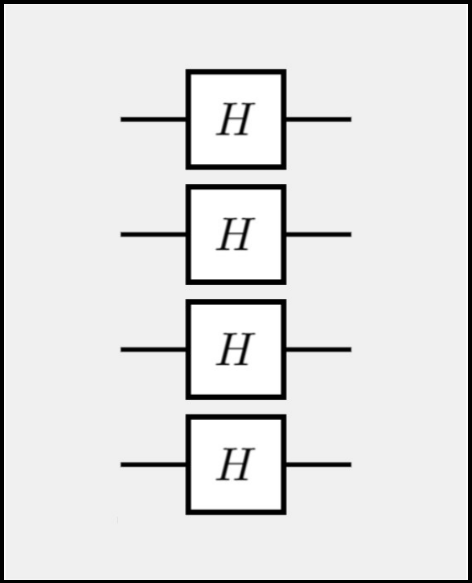
- Find a solution faster:
  - Exponentially parallel execution
  - Quantum operations reduce O() complexity
- Reduce memory size required:
  Holding 2^n *n*-bit values in *n* qubits
- Reduce power consumed per computation:
  - Parallel computation without parallel HW
  - Reduced O() complexity reduces operations

Why Quantum Algorithms?

- Find a solution faster:
  - Exponentially parallel execution
  - Quantum operations reduce O() complexity
- Reduce memory size required:
  Holding 2^n n-bit values in n qubits
- Reduce power consumed per computation:
  - Parallel computation without parallel HW
  - Reduced O() complexity, fewer operations

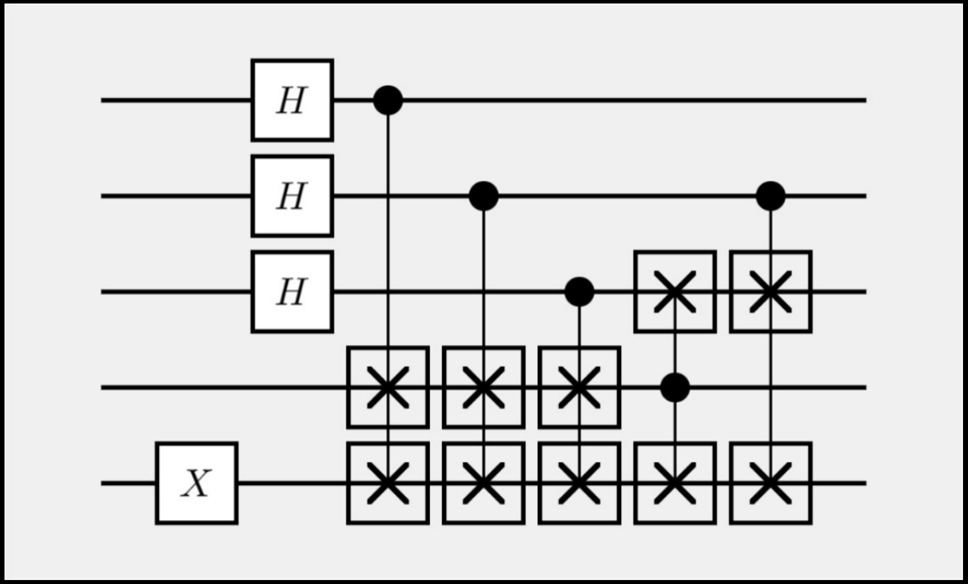
Parallel Evaluation

- Perform operation on all data values
  - Measure a randomly-selected result
  - **MuqcsCraft Random 4-bit value**



Parallel Evaluation

- Perform operation on all data values
  - Report a randomly-selected result
  - **MuqcsCraft FA**



Query Model of Computation

- Input isn't data, but a function
  - Goal is finding a property of the function by making queries against it
  - Function is treated as an **oracle**
  - Function is computable using binary logic
- The function is mapped into a unitary gate ***U_f***
  - ***U_f*** inputs |x>|y> and outputs |x>|y^f(x)>
  - f(x) can be 0 or 1; *not necessarily* entangled
  - ***U_f*** does not have to be simple...

Deutsch's Problem

- Data isn't the input; a function f(a) is
- There are 4 possible functions of one qubit:

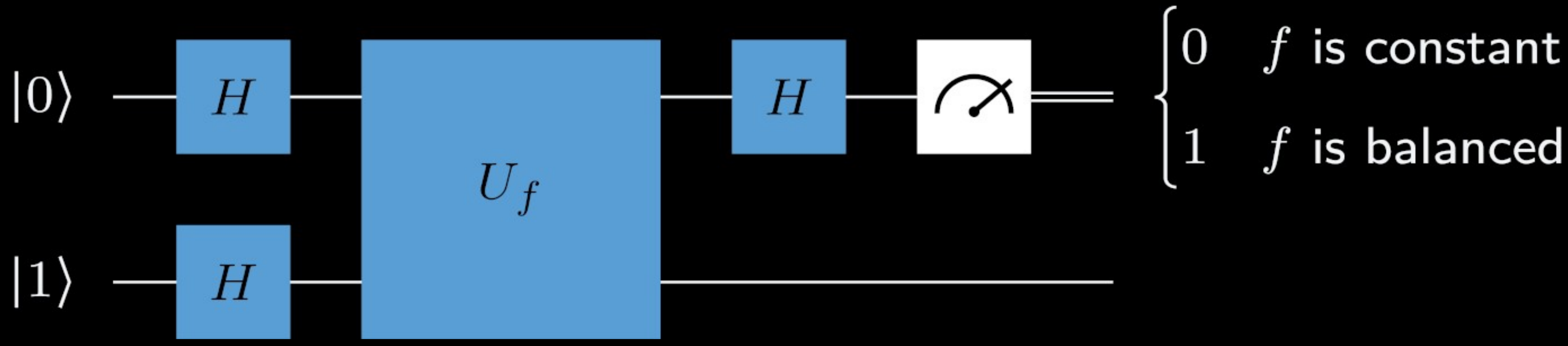
a	$f_0(a)$	$f_1(a)$	$f_2(a)$	$f_3(a)$
0	0	0	1	1
1	0	1	0	1

- A function f(a) can be:
  - **Constant**: always same output: $f_0(a)$, $f_3(a)$
  - **Balanced**: {0,1} equiprobable: $f_1(a)$, $f_2(a)$
  - Neither: *well, not in this case…*

Conventional Solution

- Query f(a) twice: f(0) and f(1)
  that is testing all possible values...
- Is f(0) == f(1)?
  - Yes: function is **Constant**
  - No: function is **Balanced**

Deutsch's Algorithm



- X input passes through ***Uf*** unchanged (generally necessary to make ***Uf*** reversible)
- Y input is **XOR**ed with ***f(X)*** in ***Uf***, but Y is 180º out of phase with X (due to |1>)
- Final H(X) completes **phase kickback**

Deutsch's Algorithm

a	$f_0(a)$	$f_1(a)$	$f_2(a)$	$f_3(a)$
0	0	0	1	1
1	0	1	0	1

- ***U_{f0}*** inputs |x>|y> and outputs |x>|y^f_0(x)>
  - $f_0(x)$ is 0
  - ***U_{f0}*** is |x>|y^0> which is just |x>|y>
  - Thus, ***U_{f0}*** is no gates at all!

- **MuqcsCraft Uf0**, entangled **MuqcsCraft Uf0**

Deutsch's Algorithm

a	$f_0(a)$	$f_1(a)$	$f_2(a)$	$f_3(a)$
0	0	0	1	1
1	0	1	0	1

- ***U_{f1}*** inputs |x>|y> and outputs |x>|y^f_1(x)>
  - f_1(x) is x
  - ***U_{f1}*** is |x>|y^x>
  - Thus, ***U_{f1}*** is a **CNOT x, y** gate
- **MuqcsCraft Uf1**

Deutsch's Algorithm

a	$f_0(a)$	$f_1(a)$	$f_2(a)$	$f_3(a)$
0	0	0	1	1
1	0	1	0	1

- U_{f2} inputs |x>|y> and outputs |x>|y^f_2(x)>
  - $f_2(x)$ is ~x
  - U_{f2} is |x>|y^~x> which is just |x>|~(y^x)>
  - Thus, U_{f2} is a **CNOT x, y** , **NOT y** gate
- **MuqcsCraft Uf2**

Deutsch's Algorithm

a	$f_0(a)$	$f_1(a)$	$f_2(a)$	$f_3(a)$
0	0	0	1	1
1	0	1	0	1

- U_{f3} inputs |x>|y> and outputs $|x>|y \wedge f_3(x)>$
  - $f_3(x)$ is 1
  - U_{f3} is |x>|y^1> which is just |x>|~y>
  - Thus, U_{f3} is a **NOT y** gate
- **MuqcsCraft Uf3**

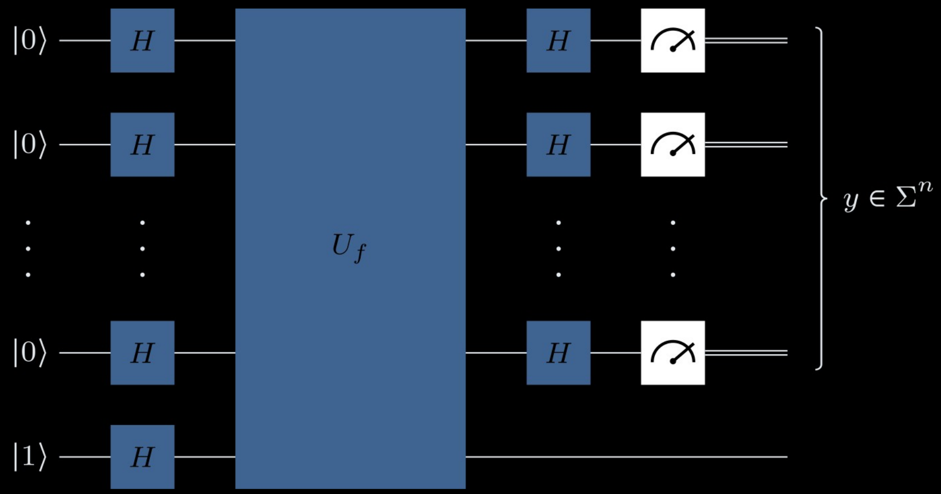
Deutsch-Jozsa Problem

- Extends Deutsch Problem to operate on a function with *k* inputs and one output
- Distinguishes **constant** from **balanced**, but "don't care" about result if it is neither (i.e., if **promise** is not kept)
- Random functions are unlikely to be either constant or balanced; e.g., **AND** X0, X1

Conventional Solution

- Evaluate f() for 2 to $2^{k-1}+1$ random inputs
- If any two evals don't return the same value, it must be balanced and can stop early
- To get a statistical answer, can stop after *n* evaluations that were all the same
  - If f() is constant, answer is correct
  - If f() is balanced, probability of error is 2^{-n+1}

Deutsch-Jozsa Algorithm



- Output 0 for constant, 1 for balanced, but 1 is if **ANY** measurement is 1 (**OR** *reduction*)
- **OR** reduction takes O(*k*) operations, but ***U_f*** is evaluated just once, with 2^k parallelism

Deutsch-Jozsa Algorithm

X1	X0	0	NOT X0	AND
0	0	0	1	0
0	1	0	0	0
1	0	0	1	0
1	1	0	0	1

- f(X1,X0)=0 is constant: **MuqcsCraft DJ0**
- f(X1,X0)=~X1 is balanced: **MuqcsCraft DJNOT**
- AND(X1,X0) is neither: **MuqcsCraft DJAND**

Bernstein-Vazirani Problem

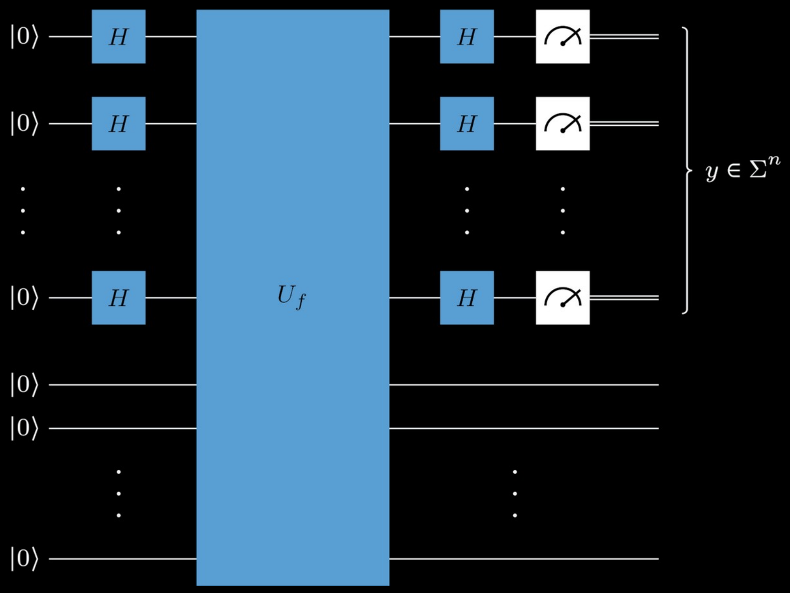
- Sometimes called **Fourier sampling** problem
- Given the promise that there exists a vector s such that f(x)=s·x for all x, find s

  Note that: $s \cdot x = (s_0 \& x_0) \wedge (s_1 \& x_1) \ldots$
- Uses Deutsch-Jozsa Algorithm
  - Measurements are weights for s
  - No conventional postprocessing (no **ANY**)
  - For example: **MuqcsCraft DJNOT**

Simon's Problem

- For a function with *n* inputs and *m* outputs
- Find v such that f(x)==f(y) implies either:
  - x^s==y
  - x==y (in which case s=0^n)
- Requires promise that s exists…

Simon’s Algorithm



- The *n* inputs are at the top and *m* outputs are at the bottom – without phase kickback
- Measurement does not directly give s…

Simon's Algorithm

- Each run of the quantum algorithm gives a randomly-selected *n*-bit vector y; collect these into a binary matrix *M* with *n* columns and *k* rows (one for each quantum algorithm run)
- M s, for a column vector of s, should equal 0; thus, classical Gaussian elimination can be used to solve for s
- Classical queries only eliminate one possible s per pair of queries producing different values, so **$\geq 2^{n/2-1}-1$ queries** are generally needed

Qiskit

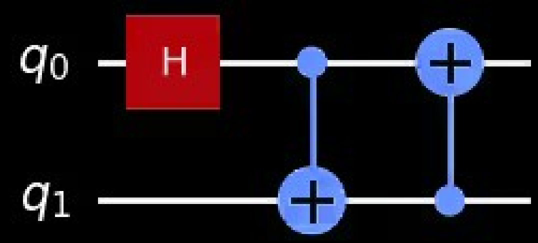
- **Qiskit** is the most popular quantum circuit SDK
  - Open source software from IBM Quantum
  - Python, but includes hardware interfaces
  - Qiskit ecosystem collects related projects

    `https://www.ibm.com/quantum/ecosystem`

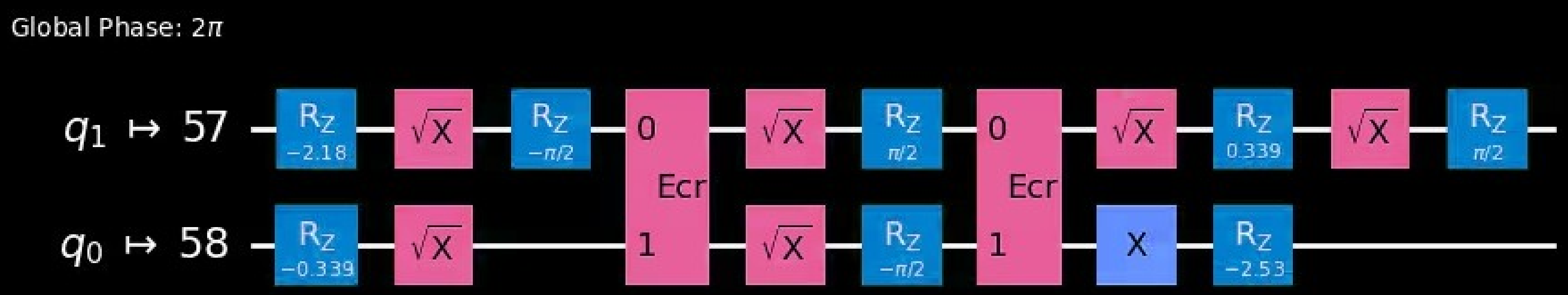
- Install Qiskit on a local machine
  `https://quantum.cloud.ibm.com/docs/en/guides/install-qiskit#local`

- IBM Quantum Experience was drag-and-drop, WWW versions are now Jupyter Notebooks
  `https://quantum.cloud.ibm.com/docs/en/guides/online-lab-environments`

Transpilation



- **Transpilation** is compilation
  - Allocating specific qubits
  - Translating to supported primitive operations
  - Applys optimization and scheduling passes



Qiskit philosophy

- Qiskit is a library in conventional python code
- Qiskit constructs a data structure representing each quantum circuit and calls functions to perform actions on that data structure
- IBM's Qiskit in the classroom
  https://quantum.cloud.ibm.com/learning/en/modules/quantum-mechanics/get-started-with-qiskit
- Let's look at the "Hello World" example
  https://quantum.cloud.ibm.com/docs/en/guides/hello-world