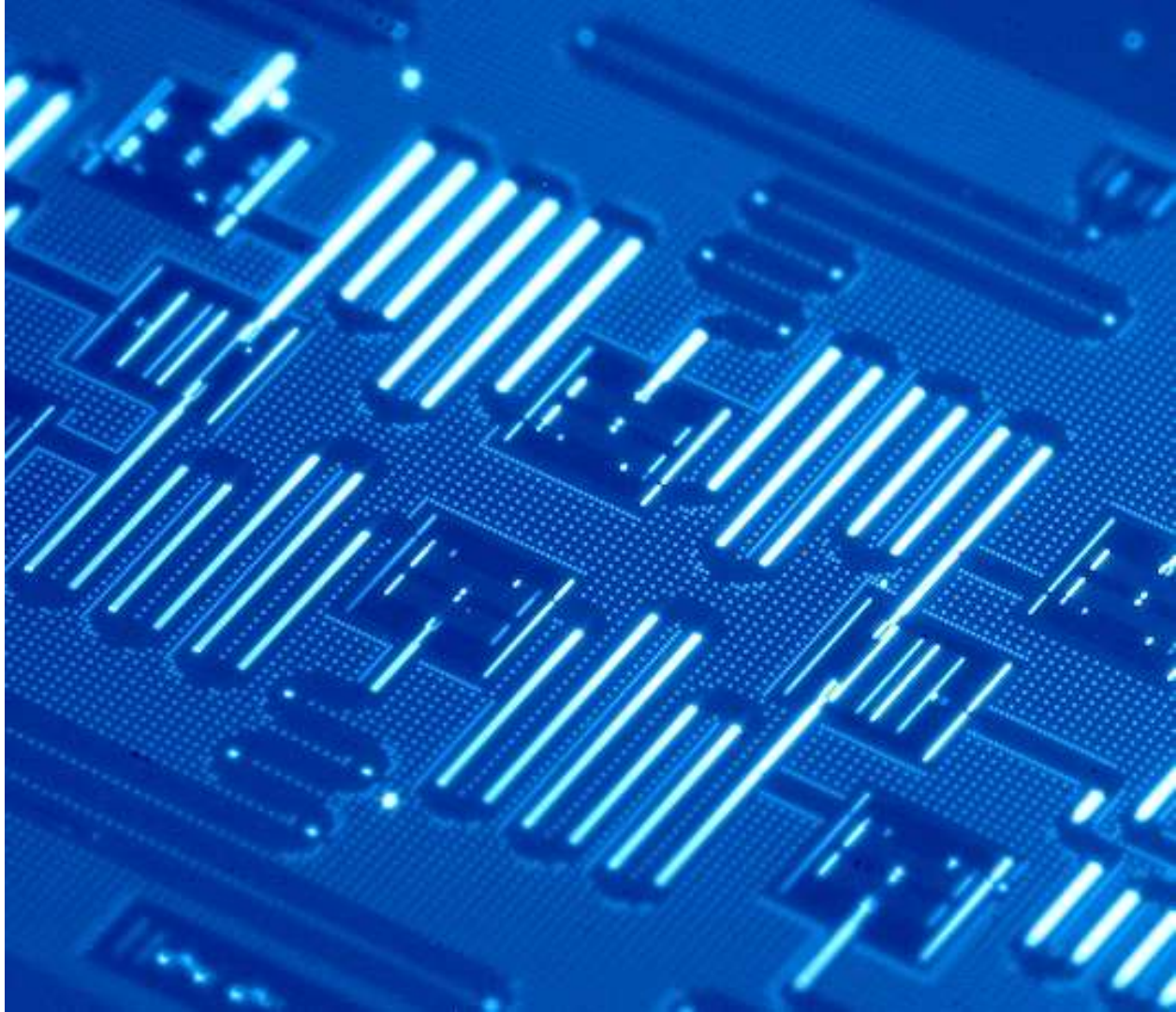
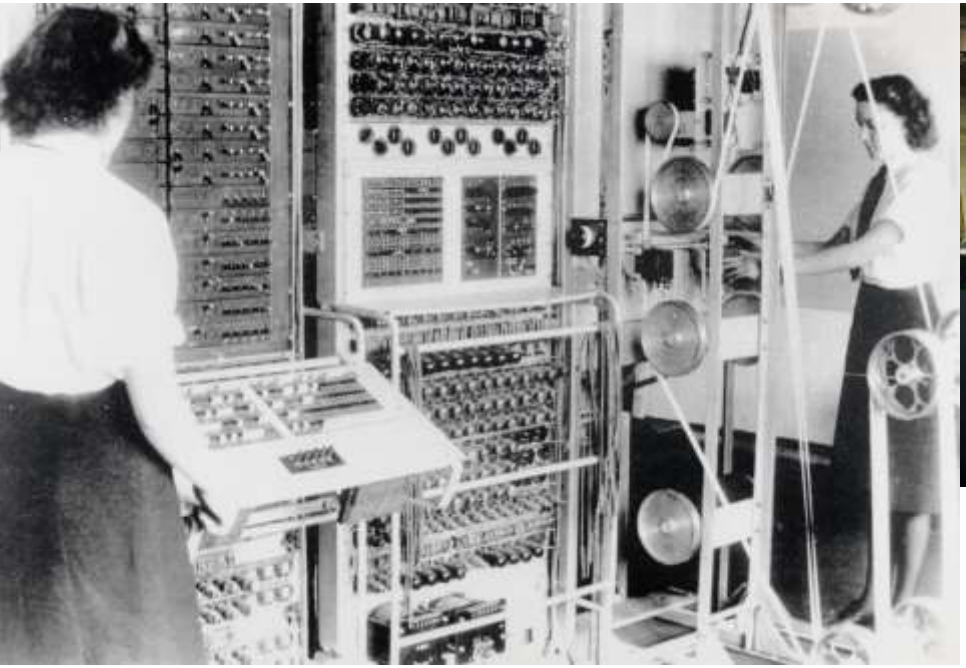
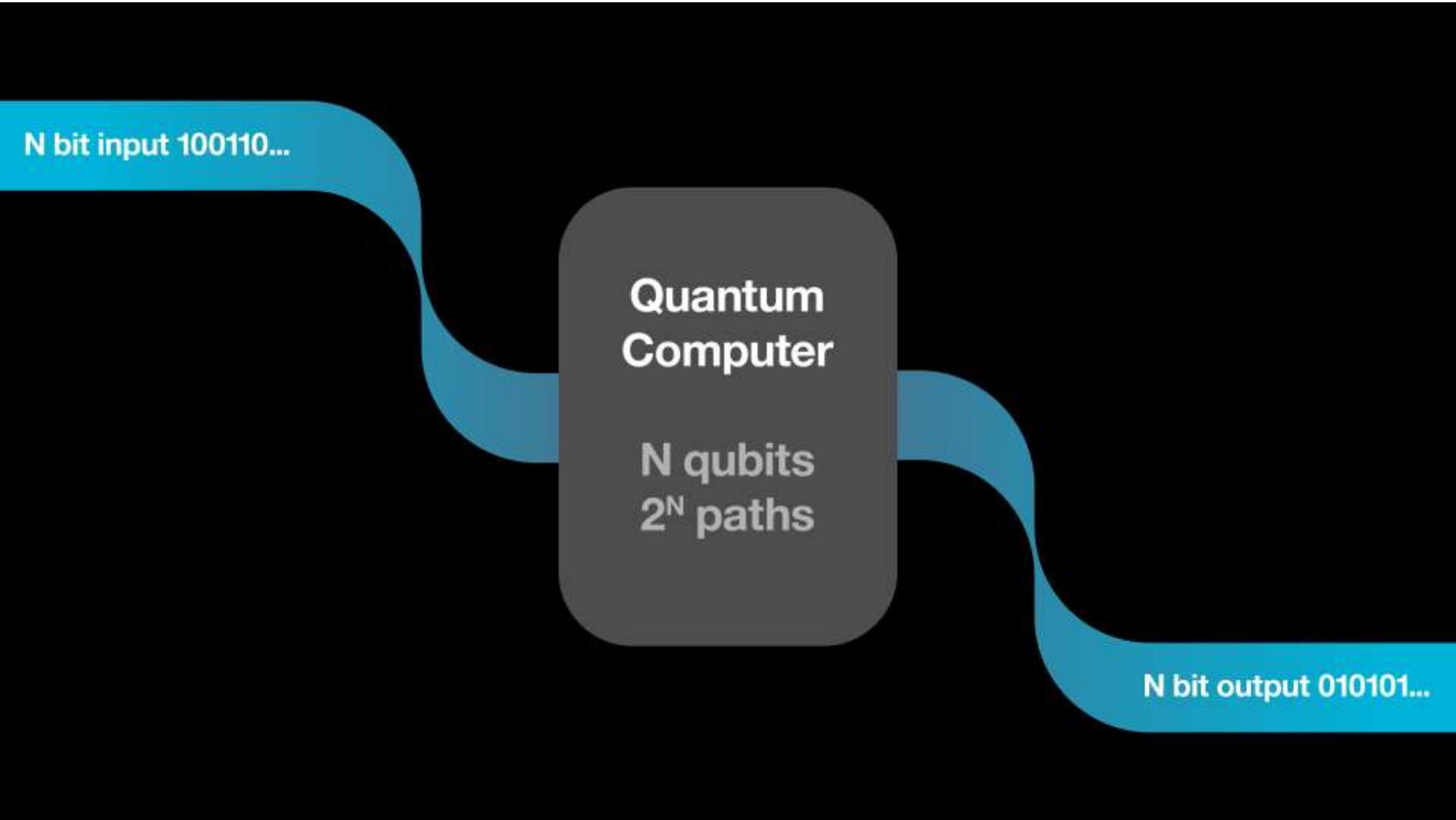
Quantum Computing






N bit input 100110...
Quantum Computer
N qubits
2^N paths
N bit output 010101...

Types of universal quantum computers

Universal fault-tolerant quantum computer

The holy grail of quantum information science. Allows one to run useful quantum algorithms which achieve exponential speed ups over their classical counterparts. However the over head of quantum error correction estimates 1M-5M qubits

Approximate quantum computer

A quantum device which does not need fault tolerance, with the goal of demonstrating a useful application by interacting with a classical computing system, e.g. quantum chemistry, optimization. Estimate 1K-5K qubits

Quantum Advantage

Quantum advantage is an idea that before any useful quantum computer is built it may be possible to demonstrate a special purpose quantum device or application whose output cannot be simulated as fast using existing classical computers. Estimate 50-100 qubits

IBM Quantum Experience

Since launch

- >45,000 users
- >300,000 unique quantum circuits
- 15+ scientific publications
- 10+ professors committing to using IBM Quantum Experience for quantum course education

www.research.ibm.com/quantum

Defining the quantum assembly language (QASM)

Statement	Description	Example
`IBMQASM 2.0;`	Denotes a file in IBM QASM 2.0 format[a]	`IBMQASM 2.0;`
`qreg name[size];`	Declare a named register of qubits	`qreg q[5];`
`reg name[size];`	Declare a named register of bits	`reg c[5];`
`include "filename";`	Open and parse another source file	`include "mygates.incl";`
`gate name(params) qargs { body }`	Declare a unitary gate subroutine	(see text)
`# comment text`	Comment a line of text	`# oops!`
`U(theta,phi,lambda) qubit|qreg;`	Apply built-in single qubit gate(s)[b]	`U(pi/2,2*pi/3,0) q[0];`
`CX qubit|qreg,qubit|qreg;`	Apply built-in CNOT gate(s)	`CX q[0],q[1];`
`measure qubit|qreg -> bit|reg;`	Make measurement(s) in Z basis	`measure q -> c;`
`reset qubit|qreg;`	Prepare qubit(s) in $\|0\rangle$	`reset q[0];`
`gatename(params) qargs;`	Apply a user-defined unitary gate	`crz(pi/2) q[1],q[0];`
`if(reg==int) qop;`	Conditionally apply quantum operation	`if(c==5) CX q[0],q[1];`
`barrier qargs;`	Prevent optimization across this source line	`barrier q[0],q[1];`

[a] This must appear as the first line of the file.

[b] The parameters `theta`, `phi`, and `lambda` are given by *parameter expressions*; see text.