Introduction to Quantum Computing

Parallelizing Data Processing Algorithms

Bhattaraprot Bhabhatsatam, Ph.D.

Lecture content by Bhattaraprot,. · Visual design assisted by Claude

WHAT WE'LL COVER TODAY

01
Classical Computing Limits

Why binary machines hit a wall

02
The Quantum Leap

Feynman, Deutsch & the big idea

03
What is a Quantum Computer?

Qubits, superposition, entanglement

04
Qubits & Math Foundations

Bra-ket, Bloch sphere, wavefunctions

05
Quantum Gates & Circuits

X, H, CNOT gates & Grover's algorithm

06
Current State (NISQ)

Where we are & what's next

SECTION 01

Classical Computing & Its Limits

Classical Computing vs Quantum Computing

Binary vs Superposition

Classical Computing: Hardware Evolution (1906–Present)

1

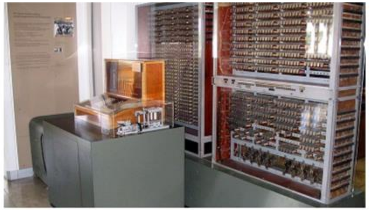
Electro-mechanical

1906–1945

Relays & gears
~5 Hz clock

The first Electric Digital Computer was designed and built by Konrad Zuse in Germany (1941).

It used 2600 electrical relays as 0/1 switches. The clock speed was about 5 Hz.



Replica of the Zuse Z3. Deutsches Museum. Munich.

2

Vacuum Tubes

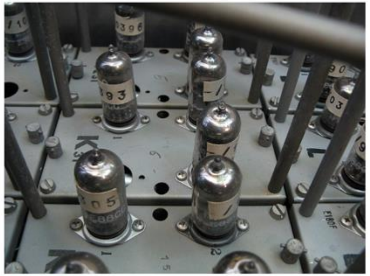
1945–1955

Gen 1 computers
0.5–1 MHz

First generation Computers (1945-1950) used vacuum tubes as binary switches.

Vacuum tubes are much faster than electrical relays.

The clock speed of these computers was between 500 KHz and 1 Mhz.



3

Transistors

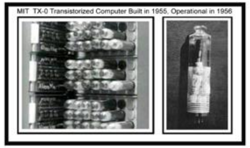
1955–1965

Gen 2 — faster,
smaller, cooler

Second Generation Computers

Second generation Computers (1950-1960) used transistors as binary 0/1 switches.

Transistors are much faster than vacuum tubes.



4

Integrated Circuits

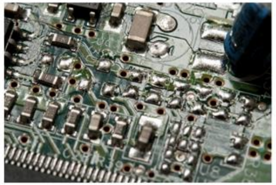
1960–now

Gen 3 → modern
CPUs & GPUs

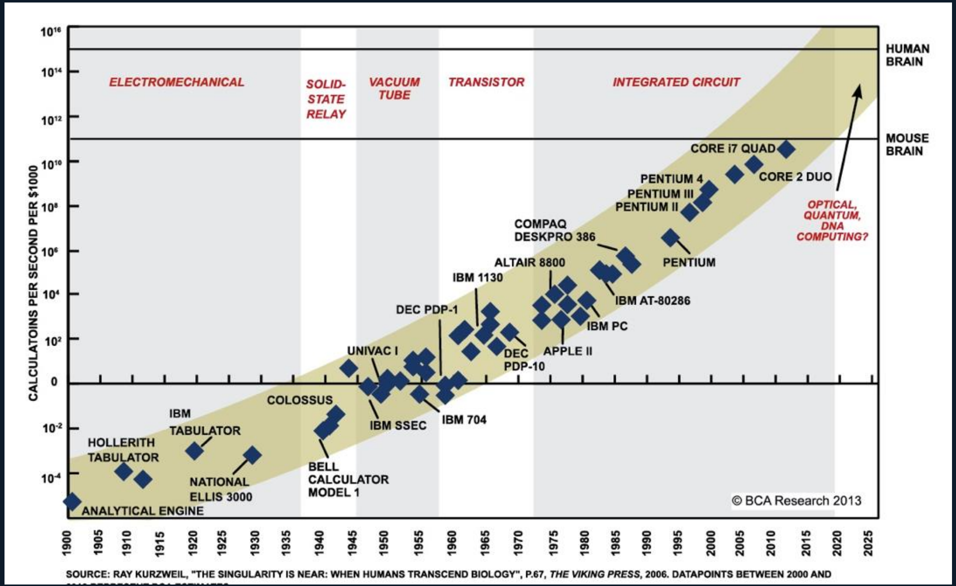
Third Generation Computers

Third generation Computers (1960) used integrated circuits as binary switches.

Integrated circuits are much faster than transistors.



Classical Computer Historical Evolution (1906–Present)
The Binary Foundations of Computing

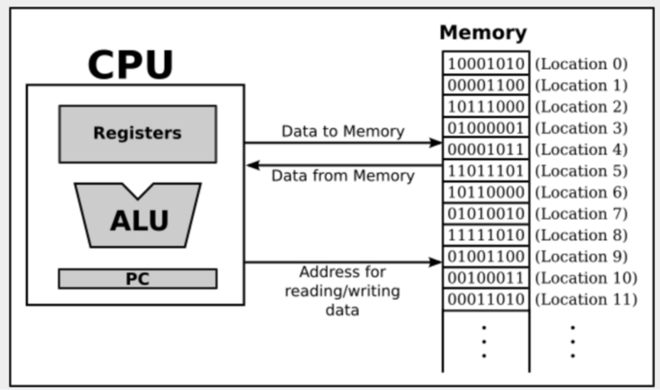


- Vacuum tubes → Transistors → Integrated Circuits
- Fundamental unit of information: binary (0s and 1s)

Classical Computing: Hardware Evolution (1906–Present)

1 2 3 4

- **Binary at the Core**: Despite hardware evolution (vacuum tubes → transistors → ICs), the **fundamental unit** of data remains the **bit (0 or 1)**
- **Most modern computing tasks** still rely on binary arithmetic and logic



https://math.hws.edu/javanotes-swing/c1/s1.html

Classical Computing: Hardware Evolution (1906–Present)

1

Electro-mechanical

1906–1945

Relays & gears
~5 Hz clock

2

Vacuum Tubes

1945–1955

Gen 1 computers
0.5–1 MHz

3

Transistors

1955–1965

Gen 2 — faster, smaller, cooler

4

Integrated Circuits

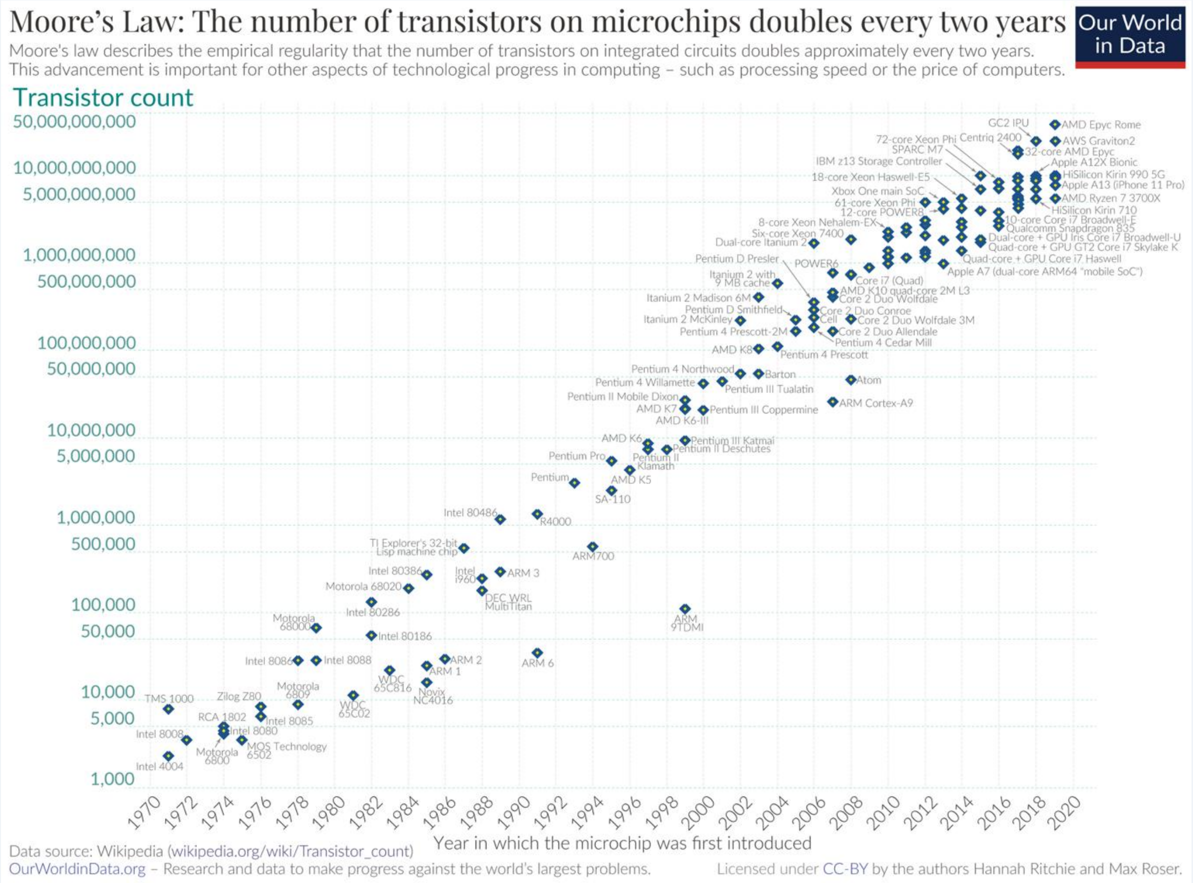
1960–now

Gen 3 → modern CPUs & GPUs

Key Insight

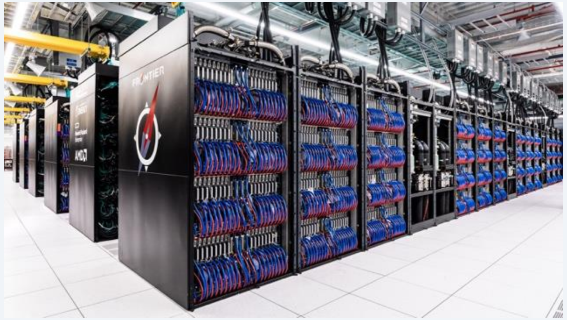
Despite four generations of radical hardware change, the fundamental unit of information remains the same: the bit (0 or 1). Everything is built on binary logic.

Classical Computing: Number of transistors



Classical Computing: Boosting Performance

- **AI & Big Data Era**
  - Large-scale, complex algorithms (machine learning, deep learning)
  - Demanding applications (big data analytics, optimization tasks)
  - Prime factorization and other computationally hard problems



A view of Frontier, an exascale-class machine at Oak Ridge National Laboratory.

Boosting Classical Performance: Three Strategies

⚡ Clock Speed

- Increase processor frequency
- Physical & heat limits (~3–5 GHz wall)
- Moore's Law: transistors double ~2 yrs

🖥 Hardware Parallelism

- Multi-core CPUs, GPUs, TPUs
- Distributed clusters & supercomputers
- Frontier (Oak Ridge): 1.1 exaFLOP/s

🔄 Algorithmic Parallelism

- Divide-and-conquer algorithms
- MapReduce, parallel data pipelines
- Specialized optimisation heuristics

⚠ Still All Binary:

All three strategies still rely on 0s and 1s. Certain problems — prime factorisation, molecular simulation, large-scale optimisation — remain exponentially hard, regardless of hardware improvements.

Problems Classical Computers Struggle to Solve

Prime Factorisation

The backbone of modern encryption (RSA). Factoring a 2048-bit number would take classical supercomputers millions of years.

Large-scale Optimisation

Travelling salesman, protein folding, logistics scheduling. Search space grows exponentially — brute force is infeasible.

Quantum Chemistry

Simulating molecular orbitals (Hartree-Fock, DFT) for drug discovery. Even HPC systems can't fully model large molecules.

Astrophysical N-body

Galaxy formation, dark matter clustering. Billions of particles each interacting — full-precision models take months on HPC.

These are not just engineering challenges — they are fundamentally hard problems that require an entirely different computing paradigm.

Problems Classical Computers Struggle to Solve

Georgescu, I. M., Ashhab, S., & Nori, F. (2014). *Quantum Simulation*. *Reviews of Modern Physics*.
Aspuru-Guzik, A., Dutoi, A. D., Love, P. J., & Head-Gordon, M. (2005). *Simulated Quantum Computation of Molecular Energies*. *Science*.
Springel, V. (2005). *The Cosmological Simulation Code GADGET-2*. *Monthly Notices of the Royal Astronomical Society*.
Johnson, D. S., & McGeoch, L. A. (Eds.). (2001). *Local Search in Combinatorial Optimization*. *Princeton University Press*.

SECTION 02

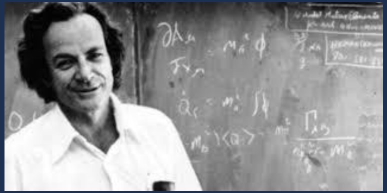
The Origins of Quantum Computing

The Founding Visions

Richard Feynman — 1981

Key Claim: Classical computers cannot efficiently simulate quantum systems.
Nature is quantum mechanical — so our computers should be too.

Feynman, R.P. "Simulating Physics with Computers."
Int. J. Theor. Phys. 21, 467–488 (1982).



David Deutsch — 1985

Key Contribution: Formalised the concept of a Universal Quantum Computer — the Quantum Turing Machine — providing the theoretical foundation for quantum computation.

Deutsch, D. "Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer."
Proc. R. Soc. Lond. A 400, 97–117 (1985).

SECTION 03

What Is a Quantum Computer?

https://quantumai.google/quantumcomputer

- **A New Computational Paradigm**
  - Utilizes **quantum mechanical phenomena** (superposition, entanglement) to **encode** and **process** information
- **Not All of Quantum Mechanics**
  - Focuses on the computational aspects rather than the full breadth of quantum theory
- **Departure from Binary Processing**
  - Moves beyond bits (0 and 1) to **qubits** that can represent multiple states simultaneously

Classical vs. Quantum Computing: The Key Differences

CLASSICAL COMPUTER

- Built on **transistors** tiny electronic switches, either ON (1) or OFF (0)
- The CPU flips billions of these switches every second
- **Registers** hold data currently being processed
- **Accumulator** stores intermediate values during calculation
- **ALU** (Arithmetic Logic Unit) performs all math and logic operations.
- Deterministic logic
- AND, OR, NOT gates
- Can parallelize, but stays binary
- Everything runs on streams of binary **0s and 1s**

VS

QUANTUM COMPUTER

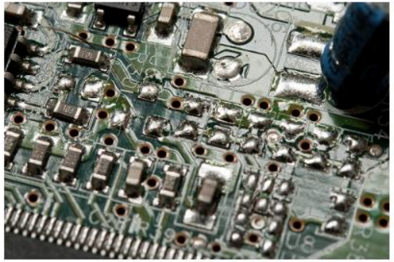
- Probabilistic — quantum amplitudes
- Superposition, entanglement, interference
- Quantum parallelism via wavefunction
- Excels at specific hard problems
- Throws out the transistor model entirely
- Uses qubits instead of bits
- Unit of info: qubit ($|0\rangle$, $|1\rangle$, or both!)
- A qubit is not a switch; it holds 0 and 1 simultaneously (superposition)
- Processes all possible states at the same time, not one by one
- When measured, the correct answer emerges from the quantum state

Classical vs. Quantum Computing: Assembly Language Level

Third Generation Computers

Third generation Computers (1960) used integrated circuits as binary switches.

Integrated circuits are much faster than transistors.

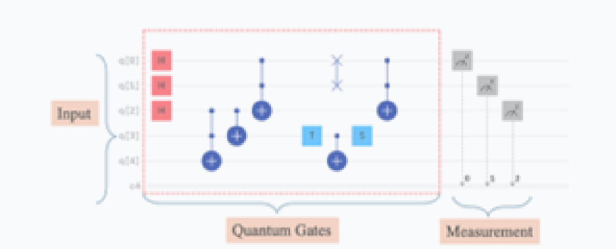


```
; ----------------------------------------------------------------------------------------
; Writes "Hello, World" to the console using only system calls. Runs on 64-bit Linux only.
; To assemble and run:
;
;     nasm -felf64 hello.asm && ld hello.o && ./a.out
; ----------------------------------------------------------------------------------------

          global    _start

          section   .text
_start:   mov       rax, 1                  ; system call for write
          mov       rdi, 1                  ; file handle 1 is stdout
          mov       rsi, message            ; address of string to output
          mov       rdx, 13                 ; number of bytes
          syscall                           ; invoke operating system to do the write
          mov       rax, 60                 ; system call for exit
          xor       rdi, rdi                ; exit code 0
          syscall                           ; invoke operating system to exit

          section   .data
message:  db        "Hello, World", 10      ; note the newline at the end
```




```python
from qiskit import QuantumCircuit, execute, Aer
import math

def create_interference_circuit():
    # Create a quantum circuit with 3 qubits
    qc = QuantumCircuit(3)

    # Initialize qubits in superposition with different phases
    qc.h(0)
    qc.h(1)
    qc.h(2)
    qc.p(math.pi/4, 0)  # Phase for constructive interference
    qc.p(math.pi/2, 1)  # Different phase
    qc.p(math.pi, 2)    # Phase for destructive interference

    # Create entanglement and additional interference
    qc.cx(0, 1)  # CNOT gate for entanglement
    qc.cx(1, 2)  # Further entanglement
    qc.p(math.pi/2, 2)  # Additional phase shift

    # Add measurements
    qc.measure_all()

    return qc
```

Three Quantum Phenomena That Power Quantum Computing

〰️

Superposition

A qubit can be in a combination of $|0\rangle$ and $|1\rangle$ simultaneously until measured.

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

This lets quantum computers explore many possibilities at the same time.

🔗

Entanglement

Two or more qubits can be correlated such that measuring one instantly tells you the state of the other — no matter the distance.

Enables coordinated, high-dimensional computation.

🌊

Interference

Quantum amplitudes can add together (constructive) or cancel out (destructive).

Quantum algorithms exploit this to amplify correct answers and suppress wrong ones.

Types of Quantum Computing Architectures

Gate-Based

IBM, Google, Rigetti

Uses quantum gates (unitary matrices) applied to qubits. Most general-purpose. Basis for most quantum algorithms.

Quantum Annealer

D-Wave

Solves optimisation problems by cooling a system to its lowest energy state. Not universal.

Measurement-Based

Research / Photonics

Starts with a large entangled state (cluster state) and performs computation through sequential measurements.

Topological

Microsoft

Encodes qubits in topological states for inherent fault tolerance. Still largely experimental.

This course focuses on Gate-Based Quantum Computers — the most versatile and widely used platform.

Types of Quantum Computers

6 Major Architectural Approaches

Superconducting

Uses cooled superconducting circuits as qubits. Most common type. Used by IBM & Google.

Trapped Ion

Traps ions with electromagnetic fields. High fidelity. Used by IonQ & Honeywell.

Photonic

Uses photons (light particles) as qubits. Operates at room temperature.

Neutral Atom

Uses neutral atoms held by laser beams. Scalable architecture.

Topological

Uses exotic particles for error-resistant qubits. Microsoft's approach.

Quantum Annealing

Solves optimization problems. D-Wave's specialized architecture.

Quantum Computing Companies

Part 1: Industry Leaders

IBM Quantum

Superconducting

Pioneering quantum computing with 100+ qubit processors. Offers cloud access via IBM Quantum Network.

https://www.ibm.com/quantum

Google Quantum AI

Superconducting

Achieved 'quantum supremacy' in 2019 with Sycamore chip. Building fault-tolerant quantum computers.

https://quantumai.google

IonQ

Trapped Ion

Publicly traded pure-play quantum company. Offers cloud-based trapped-ion quantum computers.

https://ionq.com

D-Wave

Quantum Annealing

World's first commercial quantum computer company. Focuses on optimization problems via annealing.

https://www.dwavesys.com

Quantum Computing Companies

Part 2: Rising Players & Specialists

Quantinuum	Trapped Ion	Honeywell + Cambridge Quantum merger. Known for high-fidelity H-Series trapped-ion systems.	https://www.quantinuum.com
Rigetti Computing	Superconducting	Full-stack quantum computing company. Builds superconducting processors and offers Quantum Cloud Services.	https://www.rigetti.com
Microsoft Azure Quantum	Topological	Pursuing topological qubits for fault tolerance. Offers quantum services via Azure cloud platform.	https://azure.microsoft.com/quantum
PsiQuantum	Photonic	Building a million-qubit photonic quantum computer using silicon photonics chip manufacturing.	https://psiquantum.com

SECTION 04

Qubits, Wavefunctions & Mathematical Foundations

Representing a Qubit: Bra-Ket, Vectors & the Bloch Sphere

Mathematical Forms

General state:	$\|\psi\rangle = \alpha\|0\rangle + \beta\|1\rangle$
Normalisation:	$\|\alpha\|^2 + \|\beta\|^2 = 1$
Basis states:	$\|0\rangle = [1, 0]^T$ and $\|1\rangle = [0, 1]^T$
Superposition:	$\|\psi\rangle = (1/\sqrt{2})\|0\rangle + (1/\sqrt{2})\|1\rangle$
Bloch sphere:	$\|\psi\rangle = \cos(\theta/2)\|0\rangle + e^{-i\psi} \sin(\theta/2)\|1\rangle$
Probability:	$P(\|0\rangle) = \|\alpha\|^2$, $P(\|1\rangle) = \|\beta\|^2$

The Bloch Sphere

- A unit sphere that represents all possible single-qubit states geometrically.
- North pole (top) = |0⟩ state. South pole (bottom) = |1⟩ state.
- Any point on the surface = a valid superposition state.
- θ (theta) = polar angle — controls the probability mix of |0⟩ and |1⟩.
- ϕ (phi) = azimuthal angle — encodes the quantum phase.
- Quantum gates rotate the qubit's state vector on this sphere.

Scaling Up: From 1 Qubit to Many Qubits

State Space Growth

Qubits	Classical bits needed	States
1	2	2
2	4	4
3	8	8
10	1,024	1,024
30	8 billion	~1 billion
50	2^{50} bits	≈ 1 quadrillion
300	> atoms in universe	2^{300}

2-Qubit General State

$$|\psi\rangle = c_1|00\rangle + c_2|01\rangle + c_3|10\rangle + c_4|11\rangle$$

Where: $|c_1|^2+|c_2|^2+|c_3|^2+|c_4|^2 = 1$ (normalisation)

- 4 complex numbers describe 2 qubits simultaneously
- A gate applied to 1 qubit transforms the ENTIRE 4-component wavefunction
- This is fundamentally different from classical bitwise operations
- n qubits → 2^n amplitudes — exponential information density
- Measurement collapses superposition to ONE classical outcome

SECTION 05

Quantum Gates & Circuits

Common Single-Qubit Gates (Unitary Matrices)

I (Identity)

```
[1  0]
[0  1]
```

No operation — qubit unchanged

|0⟩→|0⟩ |1⟩→|1⟩

X (NOT Gate)

```
[0  1]
[1  0]
```

Flips qubit: |0⟩⟷|1⟩ (quantum NOT gate)

|0⟩→|1⟩ |1⟩→|0⟩

Z (Phase flip)

```
[1   0]
[0  -1]
```

Flips phase of |1⟩ but not |0⟩

|0⟩→|0⟩ |1⟩→-|1⟩

H (Hadamard)

```
1/√2·[1  1]
     [1 -1]
```

Creates superposition from a basis state

|0⟩→|+⟩ |1⟩→|-⟩

S (Phase S)

```
[1  0]
[0  i]
```

Rotates phase by π/2 (90°) around z-axis

Adds phase e^(iπ/2) to |1⟩

Y Gate

```
[0  -i]
[i   0]
```

Rotates π radians about the y-axis

|0⟩→i|1⟩ |1⟩→-i|0⟩

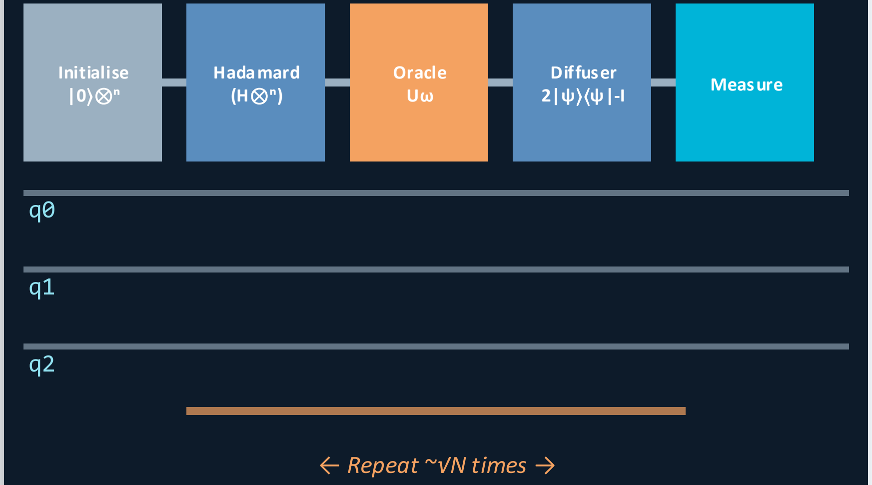
Quantum algorithms = sequences of gates chosen to amplify correct answers via interference.

Putting It Together: Grover's Search Algorithm

What Grover's Algorithm Does

- Problem: Find one marked item in an unstructured database of N items.
- Classical approach: $O(N)$ — must check items one by one.
- Grover's approach: $O(\sqrt{N})$ — quadratic speedup via quantum interference.
- For N=1,000,000: classical needs ~1M checks; Grover needs ~1,000.
- Structure: Hadamard gates (superposition) → Oracle (marks target) → Diffuser (amplifies target) → Repeat ~√N times → Measure.

Circuit Structure (2 Iterations)



Applications: Database search, constraint optimisation, collision finding, cryptographic hash reversal.

SECTION 06

The Current State of Quantum Computing (NISQ Era)

The NISQ Era: Where We Are Now

NISQ = Noisy Intermediate-Scale Quantum

Today's devices have 50–1000+ qubits, but are noisy (error-prone) and lack full error-correction.

Today's Hardware

- IBM Quantum: 1000+ qubit Eagle/Heron processors
- Google Willow: demonstrated "quantum supremacy"
- IonQ, Quantinuum: trapped-ion systems
- Error rates: ~0.1–1% per gate operation

Key Limitations

- Qubits decohere quickly (microseconds to milliseconds)
- Gate errors accumulate in long circuits
- Limited connectivity between qubits
- No fault-tolerant error correction yet

What's Ahead

- Fault-tolerant quantum computers (logical qubits)
- Quantum error correction codes (surface codes)
- Hybrid classical-quantum algorithms (QAOA, VQE)
- Timescale: 5–15 years for broad utility

Key Takeaways

01

Classical computers are binary (bits). Despite massive hardware advances, they hit exponential walls on certain problems.

02

Quantum computers exploit superposition, entanglement, and interference to process information fundamentally differently.

03

A qubit's state is a complex wavefunction. Gates are unitary matrices that rotate state vectors on the Bloch sphere.

04

We are in the NISQ era — early-stage, noisy, analogous to 2nd-generation classical computers. But progress is rapid.

⟶ *Next: How quantum computers operate, programming qubits in Qiskit, and hands-on IBM Quantum Composer labs.*

Lab Activities: IBM Quantum Composer

Lab 2-1 — Exploring the IBM Quantum Composer

1. Go to https://quantum.ibm.com and create a free account.
2. Open the Quantum Composer tool from the dashboard.
3. Drag and drop quantum gates (start with H on a single qubit).
4. Observe the Bloch sphere and state vector update after each gate.
5. Run the circuit and read the measurement probability histogram.
6. Goal: Understand how |0⟩ changes as you apply H, X, Z, and S gates.

Lab 2-2 — Exploring Gates and Their Outputs

1. Start with qubit in |0⟩ and apply X gate — confirm |0⟩→|1⟩.
2. Apply H gate — confirm equal superposition (50%/50%).
3. Apply Z after H — observe phase flip on the Bloch sphere.
4. Try HXH — what do you notice? (This is equivalent to Z!)
5. Experiment with 2-qubit circuits using CNOT to create entanglement.
6. Goal: Build intuition for how gates compose and transform quantum states.

Recommended Reading & Resources

📐 Linear Algebra

- Gilbert Strang — Linear Algebra and Its Applications
- Khan Academy: Linear Algebra (free, online)
- 3Blue1Brown — The Essence of Linear Algebra (YouTube)
- Focus: vectors, matrices, eigenvalues, unitary matrices

⚛️ Quantum Mechanics

- Susskind & Hrabovsky — Quantum Mechanics: The Theoretical Minimum
- MIT OpenCourseWare: Intro to Quantum Mechanics (free)
- Feynman Lectures on Physics, Vol. III (free at Caltech)
- Focus: wavefunctions, measurement, operators

💻 Quantum Programming

- IBM Quantum Documentation — quantum.ibm.com/docs
- Qiskit Textbook — learn.qiskit.org (free, interactive)
- Nielsen & Chuang — Quantum Computation & Quantum Information
- Focus: gates, circuits, Qiskit Python SDK

Tip: Start with 3Blue1Brown's linear algebra series and the Qiskit Textbook — both are free and highly visual.

Questions?

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Bhattaraprot Bhabhatsatam, Ph.D.
Introduction to Quantum Computing — Session 1