Introduction to Quantum Computing

How I Learned to Stop Worrying and Love the Bomb

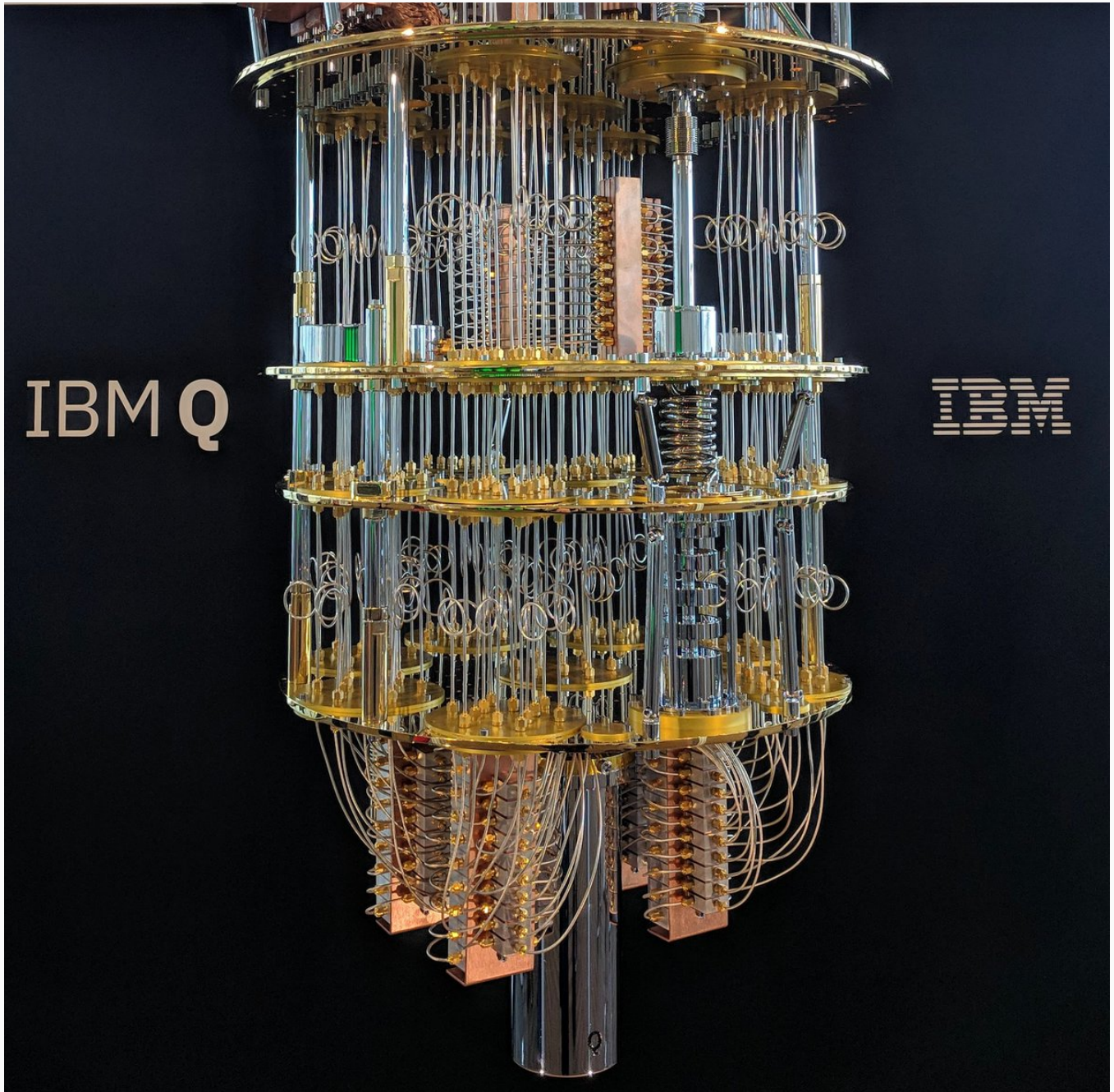
Dr. Omri Har-Shemesh

13. February 2019

Schmiede.ONE GmbH & Co. KG

Introduction

Ce n'est pas un lustre (This is not a Chandelier)



Big big picture

- New computing paradigm

Big big picture

- New computing paradigm
- Uses the rules of quantum mechanics

Big big picture

- New computing paradigm
- Uses the rules of quantum mechanics
- **Might** be able to solve **some** problems exponentially faster

Big big picture

- New computing paradigm
- Uses the rules of quantum mechanics
- **Might** be able to solve **some** problems exponentially faster
- **Definitely** could simulate quantum systems better

Big big picture

- New computing paradigm
- Uses the rules of quantum mechanics
- **Might** be able to solve **some** problems exponentially faster
- **Definitely** could simulate quantum systems better
- Realizable in large scale?

Big big picture

- New computing paradigm
- Uses the rules of quantum mechanics
- **Might** be able to solve **some** problems exponentially faster
- **Definitely** could simulate quantum systems better
- Realizable in large scale?
- ¯_(ツ)_/¯

The Technical Part

“Shut up and calculate”

David Mermin

Overview

- Representing computation with linear algebra
- Qubits, superposition and quantum logic gates
- Simplest problem where a quantum computer outperforms a classical one
- Bonus: Quantum entanglement and quantum teleportation

Representing classical bits as vectors

One bit with value 0, also written as $|0\rangle$ (Dirac vector notation)

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

One bit with value 1, also written as $|1\rangle$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Review: matrix multiplication

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{pmatrix}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{pmatrix} aw + by & ax + bz \\ cw + dy & cx + dz \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

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

Operations on one classical bit (cbit)

Identity	$f(x) = x$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
Negation	$f(x) = \neg x$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
Constant-0	$f(x) = 0$	$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
Constant-1	$f(x) = 1$	$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Reversible computing

- Given the operation and the input, you can always infer the output.

Reversible computing

- Given the operation and the input, you can always infer the output.
  - For $Ax = b$, given b and A, you can uniquely find x.

Reversible computing

- Given the operation and the input, you can always infer the output.
  - For $Ax = b$, given b and A, you can uniquely find x.

- Permutations are reversible; erasing and overwriting are not

Reversible computing

- Given the operation and the input, you can always infer the output.
  - For $Ax = b$, given b and A, you can uniquely find x.

- Permutations are reversible; erasing and overwriting are not
  - Identity and negation are reversible.

Reversible computing

- Given the operation and the input, you can always infer the output.
  - For $Ax = b$, given b and A, you can uniquely find x.

- Permutations are reversible; erasing and overwriting are not
  - Identity and negation are reversible.
  - Constant-0 and Constant-1 are not reversible.

Reversible computing

- Given the operation and the input, you can always infer the output.
  - For $Ax = b$, given b and A, you can uniquely find x.

- Permutations are reversible; erasing and overwriting are not
  - Identity and negation are reversible.
  - Constant-0 and Constant-1 are not reversible.

- Quantum computers only use reversible operations.

Reversible computing

- Given the operation and the input, you can always infer the output.
  - For $Ax = b$, given b and A, you can uniquely find x.

- Permutations are reversible; erasing and overwriting are not
  - Identity and negation are reversible.
  - Constant-0 and Constant-1 are not reversible.

- Quantum computers only use reversible operations.
  - In fact, all quantum operations are their own inverse.

Review: tensor product of vectors

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \\ x_1 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} x_0 y_0 \\ x_0 y_1 \\ x_1 y_0 \\ x_1 y_1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \otimes \begin{pmatrix} z_0 \\ z_1 \end{pmatrix} = \begin{pmatrix} x_0 y_0 z_0 \\ x_0 y_0 z_1 \\ x_0 y_1 z_0 \\ x_0 y_1 z_1 \\ x_1 y_0 z_0 \\ x_1 y_0 z_1 \\ x_1 y_1 z_0 \\ x_1 y_1 z_1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Representing multiple cbits

$$|00\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \qquad |01\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \qquad |100\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$$

$$|10\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \qquad |11\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

- The tensor representation is called the **product state**.
- It can be **factored** back into the **individual state** representation.
- The product state of n bits is a vector of size 2^n.

Operations on multiple cbits: CNOT

- Takes two bits, one “control” bit and one“target” bit.
- If the “control” bit is set, flip the “target” bit, otherwise leave it.
- If most significant bit is control, and least-significant is target, then:

Operations on multiple cbits: CNOT

- Takes two bits, one “control” bit and one“target” bit.
- If the “control” bit is set, flip the “target” bit, otherwise leave it.
- If most significant bit is control, and least-significant is target, then:

$$
\begin{array}{ll}
|00\rangle & |00\rangle \\
|01\rangle & |01\rangle \\
|10\rangle & |10\rangle \\
|11\rangle & |11\rangle
\end{array}
$$

Operations on multiple cbits: CNOT

- Takes two bits, one “control” bit and one“target” bit.
- If the “control” bit is set, flip the “target” bit, otherwise leave it.
- If most significant bit is control, and least-significant is target, then:

$$
\begin{aligned}
|00\rangle &\longrightarrow |00\rangle \\
|01\rangle &\qquad\quad |01\rangle \\
|10\rangle &\qquad\quad |10\rangle \\
|11\rangle &\qquad\quad |11\rangle
\end{aligned}
$$

Operations on multiple cbits: CNOT

- Takes two bits, one “control” bit and one“target” bit.
- If the “control” bit is set, flip the “target” bit, otherwise leave it.
- If most significant bit is control, and least-significant is target, then:

$$|00\rangle \longrightarrow |00\rangle$$

$$|01\rangle \longrightarrow |01\rangle$$

$$|10\rangle \qquad\quad |10\rangle$$

$$|11\rangle \qquad\quad |11\rangle$$

Operations on multiple cbits: CNOT

- Takes two bits, one “control” bit and one“target” bit.
- If the “control” bit is set, flip the “target” bit, otherwise leave it.
- If most significant bit is control, and least-significant is target, then:

$$|00\rangle \longrightarrow |00\rangle$$

$$|01\rangle \longrightarrow |01\rangle$$

$$|10\rangle \qquad\quad |10\rangle$$

$$|11\rangle \qquad\quad |11\rangle$$

Operations on multiple cbits: CNOT

- Takes two bits, one “control” bit and one“target” bit.
- If the “control” bit is set, flip the “target” bit, otherwise leave it.
- If most significant bit is control, and least-significant is target, then:

$$|00\rangle \longrightarrow |00\rangle$$

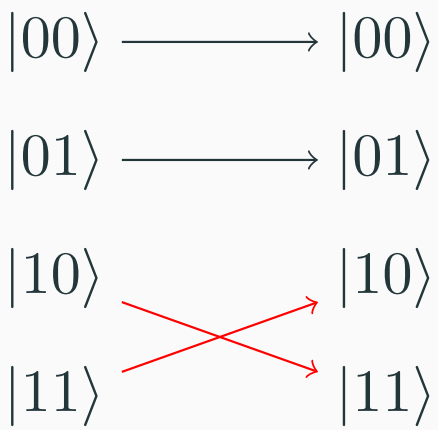
$$|01\rangle \longrightarrow |01\rangle$$

$$|10\rangle \longrightarrow |11\rangle$$

$$|11\rangle \longrightarrow |10\rangle$$

Operations on multiple cbits: CNOT

- Takes two bits, one “control” bit and one“target” bit.
- If the “control” bit is set, flip the “target” bit, otherwise leave it.
- If most significant bit is control, and least-significant is target, then:



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