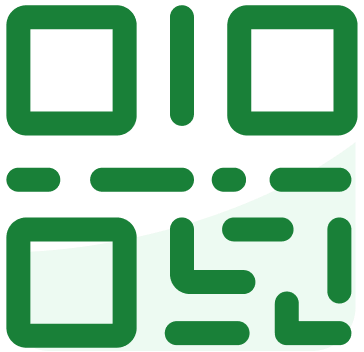
slido



Join at slido.com
#4180915

ⓘ Start presenting to display the joining instructions on this slide.

Introduction to Quantum Programming

양자정보 겨울학교

2024.1.14 - 1.19

성균관대학교 김준기

slido

Join at slido.com
#4180915

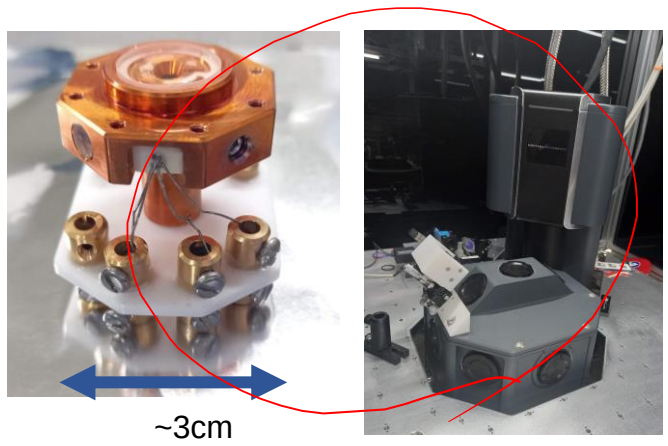
ⓘ Start presenting to display the joining instructions on this slide.

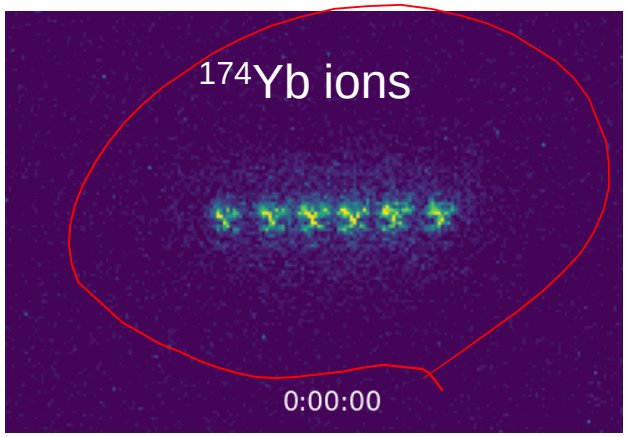
Speaker : Prof. Junki Kim

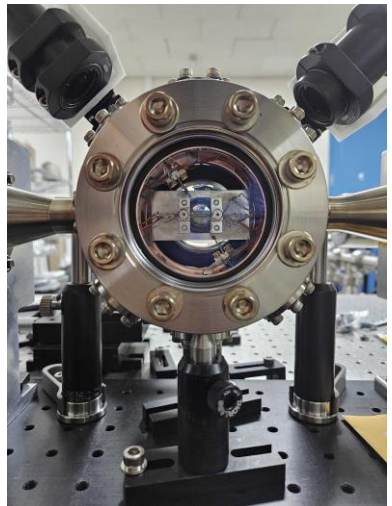
- Quantum Engineering with Trapped Ions (QuETI) Lab. @ SKKU

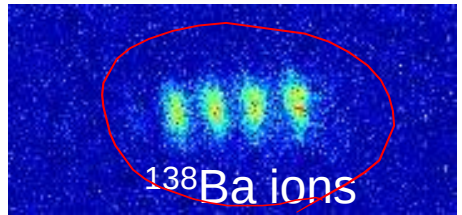
- Research Topics

  - Full-stack Ion-trap Quantum Computing
  - Quantum engineering for scalable & versatile QI platform
    - Ion-photon quantum interface
    - High-dimensional quantum information processor



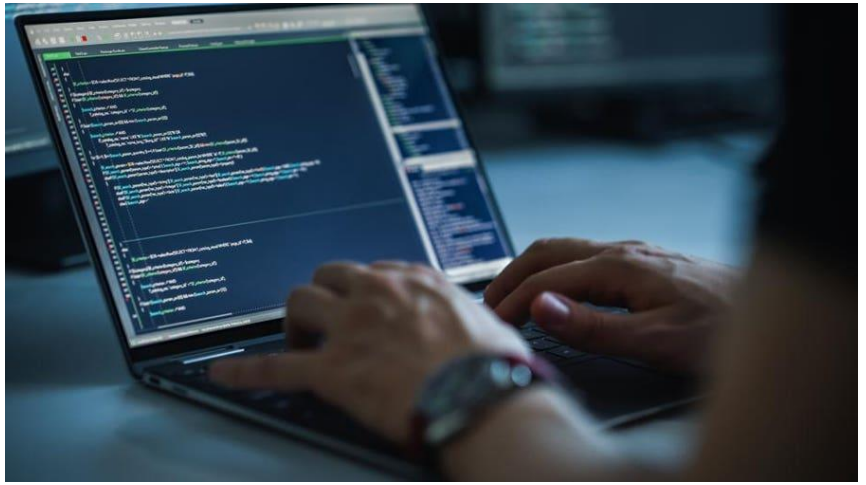






Quantum Programming?

- Programming : the composition of sequences of ***instructions***, that computers can follow to perform tasks.
  - Classical computers use **bits** (0 or 1).
  - Bits follows **Boolean algebra**
- Quantum programming – set of ***quantum instructions***
  - Quantum computers use **qubits**
  - *Instructions for qubits? Algebra for qubits?*

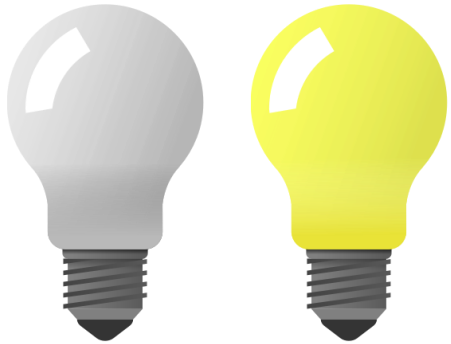


Contents

- Classical computing – logic circuits
- Quantum information basics – Qubits
- Quantum gates and Quantum circuits
- Quantum computing architectures

Bits

- The minimal unit of information
- Ex) Head/Tail of coins, Lamp on/off

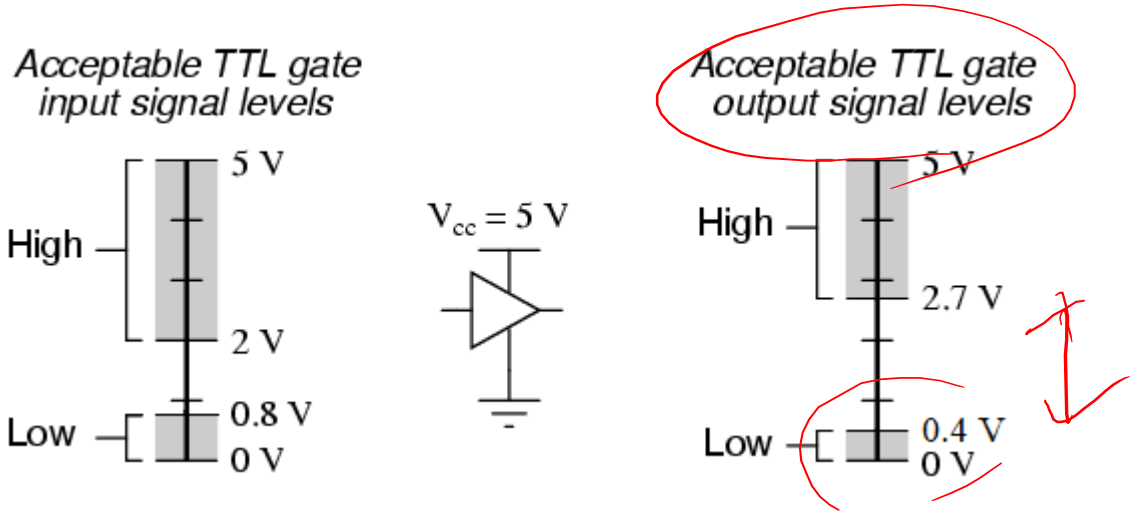


- Number of possible states with N bits : 2^N
- 1 byte = 8 bits

Other than bits?

- Dice (Cubic) : 6^N
- DNA : A,G,C,T = 4 Nucleobase

- Why bits? : Easy to physically realize (no intermediate state), Resilient to noise.
- ex) TTL (Transistor-transistor logic)



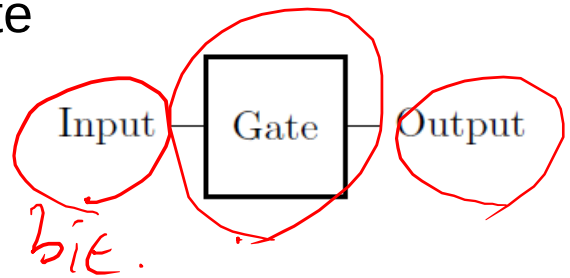
Encoding informations

- ASCII
  - Use 7 bits to encode various characters and signals
  - 65 = 100 0001 (2) = “A”
  - 48 = 011 0000 (2) = “0”
  - 95 bit string → character/number/special characters - printable, 32 bit string → Control character

- UTF-8 / Unicode
  - Up to 32 bits (4-byte) : $2^{32} = 4{,}294{,}967{,}296$ states
  - In use : 1,112,064
  - All other language characters (including Korean) need unicode

Logic gate ← Instruction

- Computers store information in bit format.
- Computer calculation = bit calculation
- Single bit gate



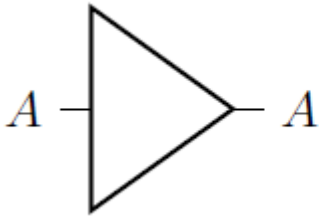
Input	Output
0	?
1	?

0 1

0 1

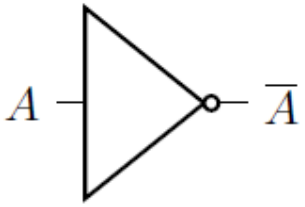
Single bit gate

- Identity gate



A	A
0	0
1	1

- NOT gate

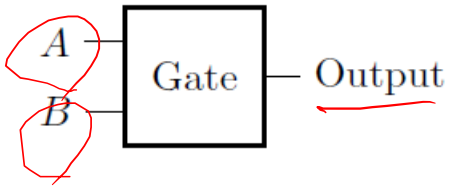


A	$\overline{A}$
0	1
1	0

- Always 0 gates , Always 1 gates

Two-bit gate

- Input bits = 2, output bit = 1→ $2^4 = 16$ cases

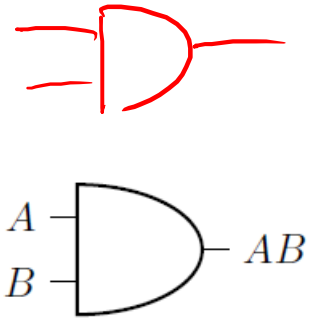


A	B	Output
0	0	?
0	1	?
1	0	?
1	1	?

- Notable two-bit gates : AND, OR, XOR, NAND, NOR

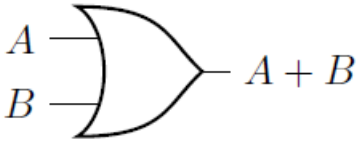
Two-bit gates

- AND gate



A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

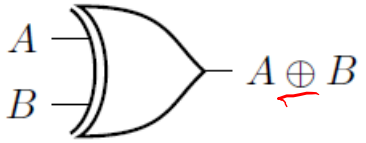
- OR gate



A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Two-bit gates

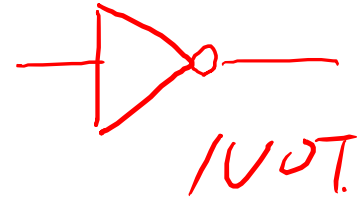
- XOR gate

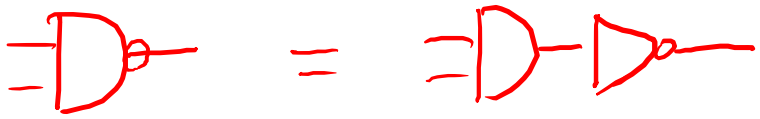


A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

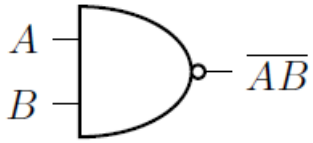
- Modulo 2 addition : $\oplus$, $1 + 1\ (mod\ 2) = 2\ (mod\ 2) = 0\ (mod\ 2)$

Two-bit gates



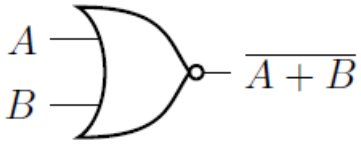


- NAND gate

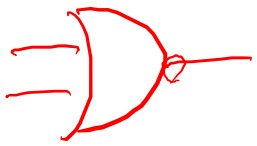


A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

- NOR gate

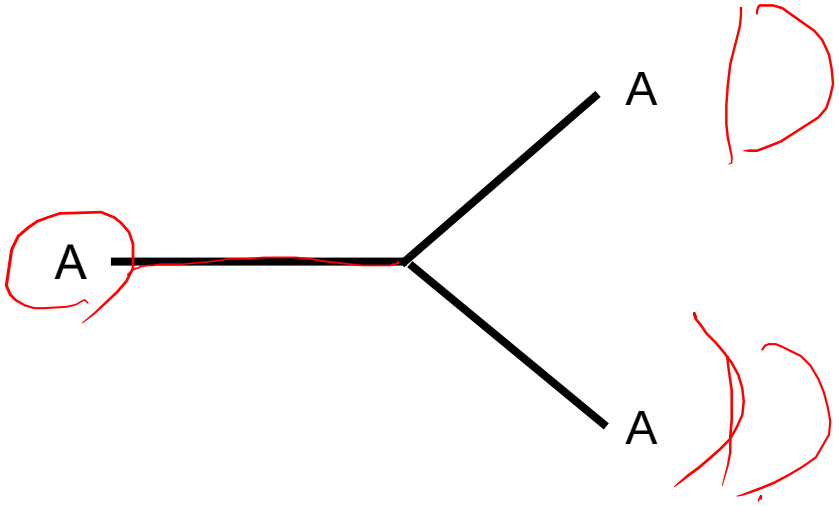


A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



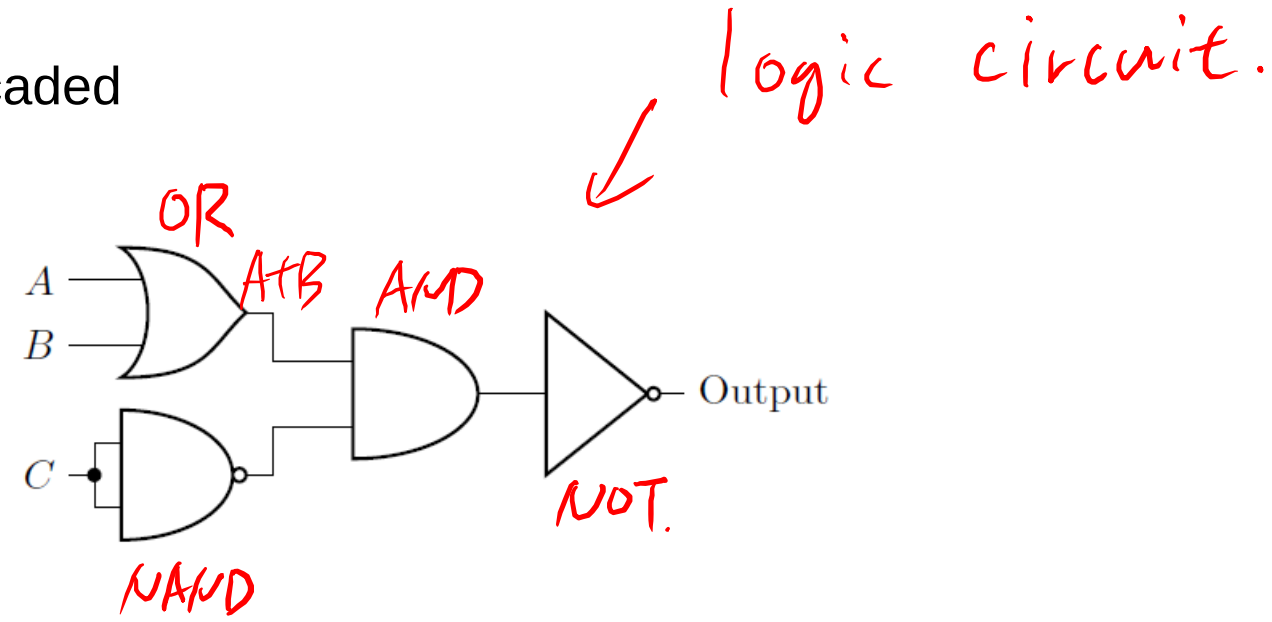
Fan out (copy)

- Copy information (Fan out)



Composite logic gates

- Multiple logic gates can be cascaded



Universal Logic gate

$f(x) = x^2 + 3x + 2$

- To realize arbitrary logical operation $f(A, B, C, \dots) = A'$, How many logic gates are required?
  - Possible input bitstrings 2^N → 2^N different logic gates required? Fortunately not!
  - arbitrary logic operations can be decomposed into AND, OR, NOT gates
  - → {AND, OR, NOT} is 'Universal logic gate set'

A	B	C	Output
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

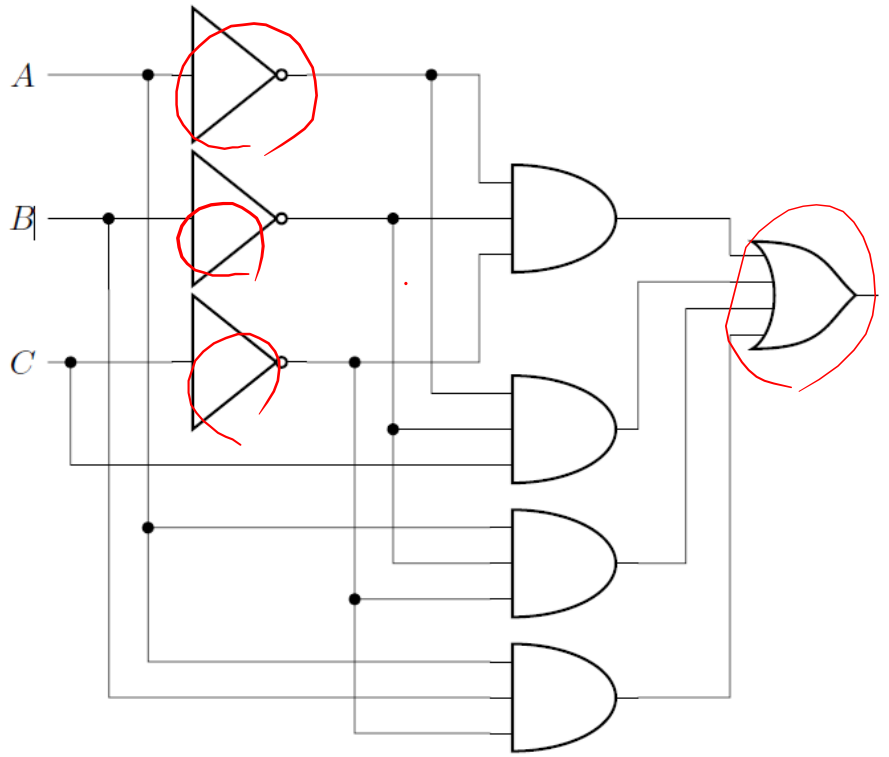
$\bar{A}\cdot\bar{B}\cdot\bar{C}$ → 1 only if

$+ \bar{A}\cdot\bar{B}\cdot C$ → 1 "

$+ A\cdot\bar{B}\cdot\bar{C}$

$+ AB\bar{C}$

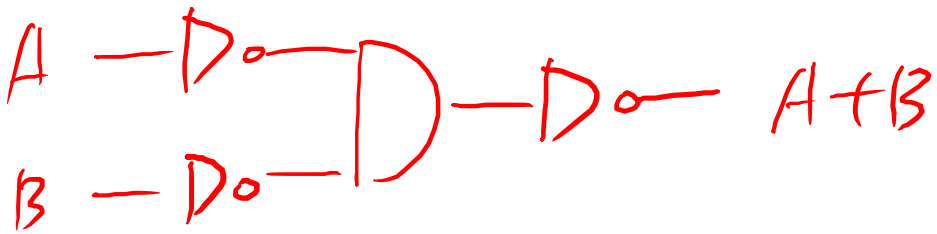
↳ 진리표 (truth table)



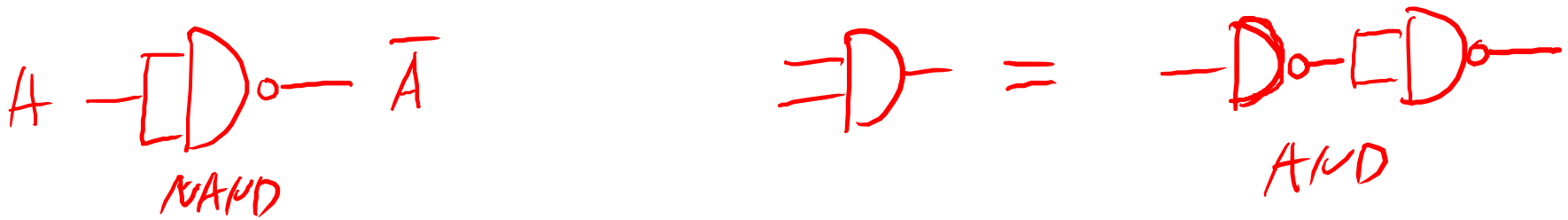
AND, NOT, OR

Universal Logic gate

- OR gate can be expressed by NOT and AND gates→ {AND, NOT} is universal gate set



- AND, NOT gates can be expressed by NAND gate → NAND gate is universal logic gate



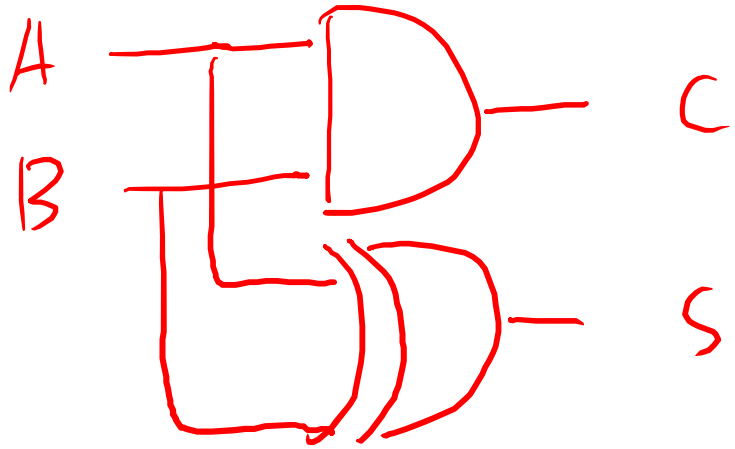
- NOR gate is universal logic gate as well

Example) Half-Adder

- A+B = ?
  - 0 + 0 = 00
  - 0 + 1 = 01
  - 1 + 0 = 01
  - 1 + 1 = 10

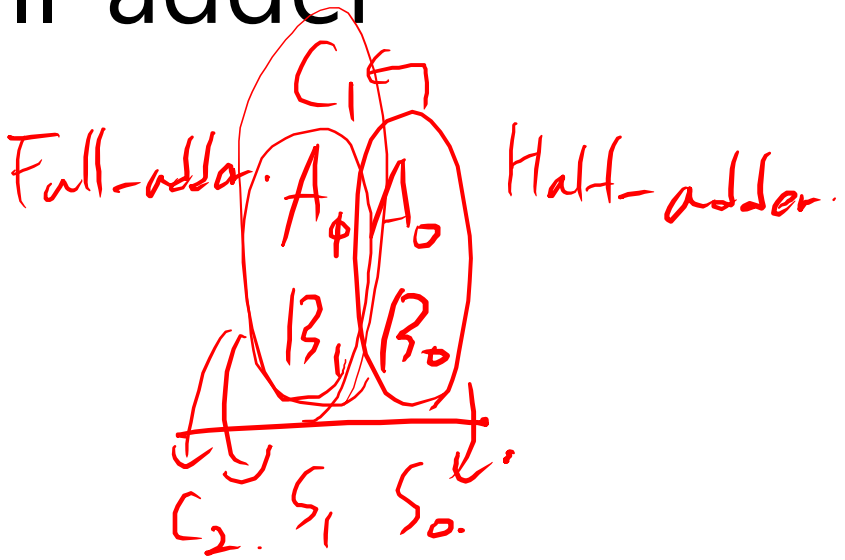
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

AND XOR

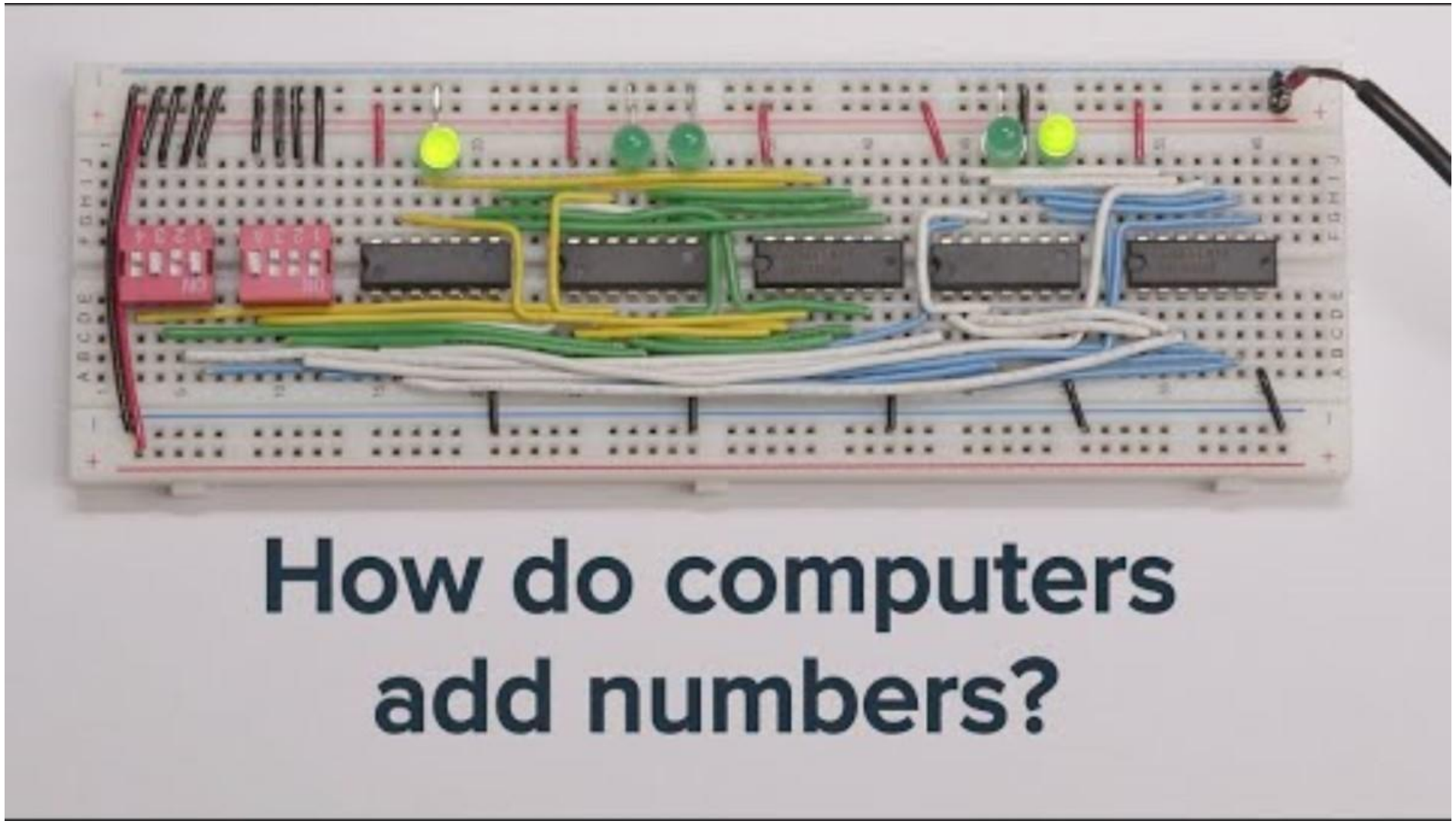


Example) Full-adder

- $A_1A_0 + B_1B_0 = ?$ $X_2X_1X_0$
  - $A_0 + B_0 = C_1S_0$: Half-adder
  - $A_1 + B_1 + C_1 = C_2S_1$: Full-adder

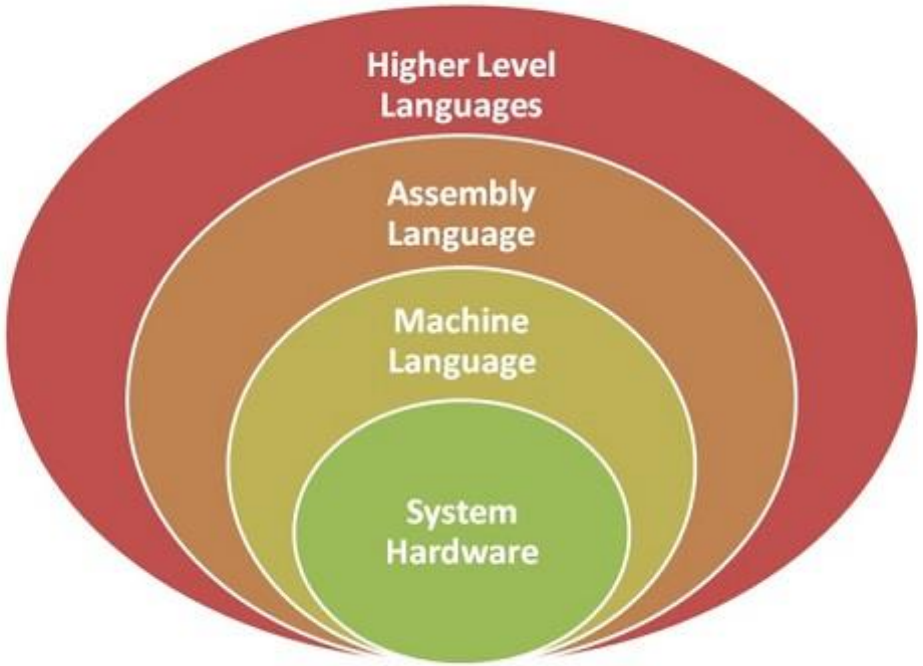


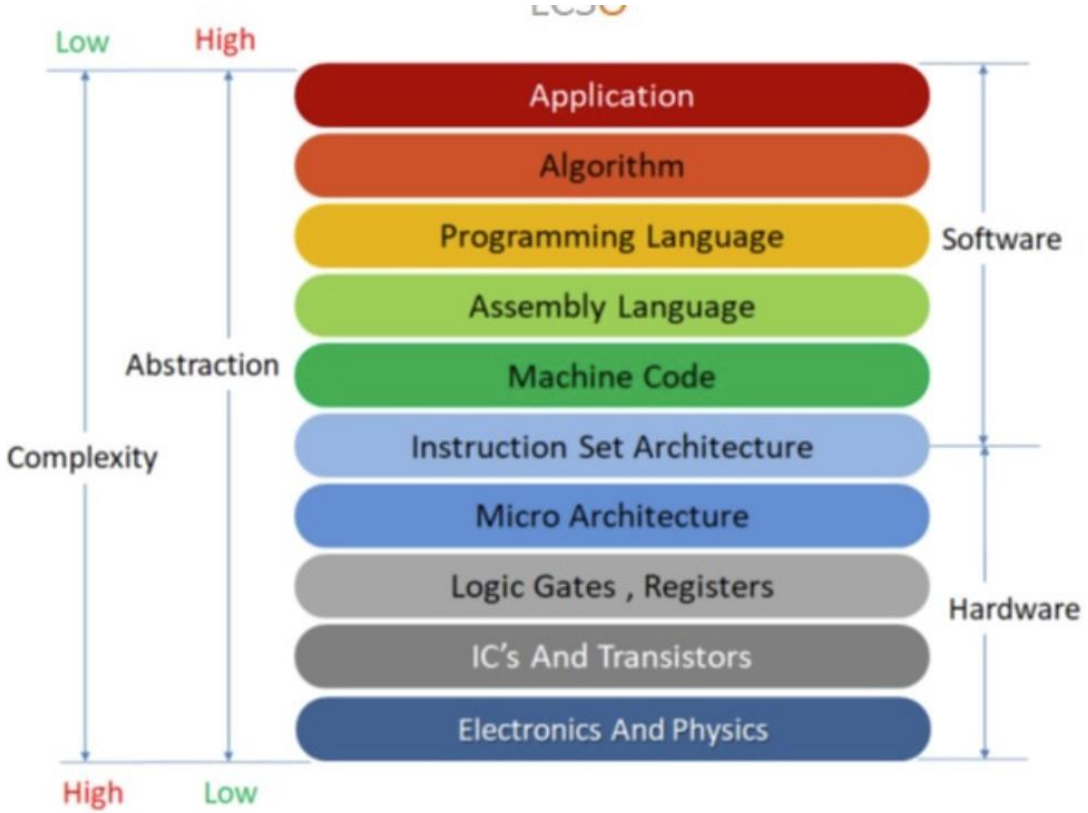
Example) IC circuit version



Program to hardware

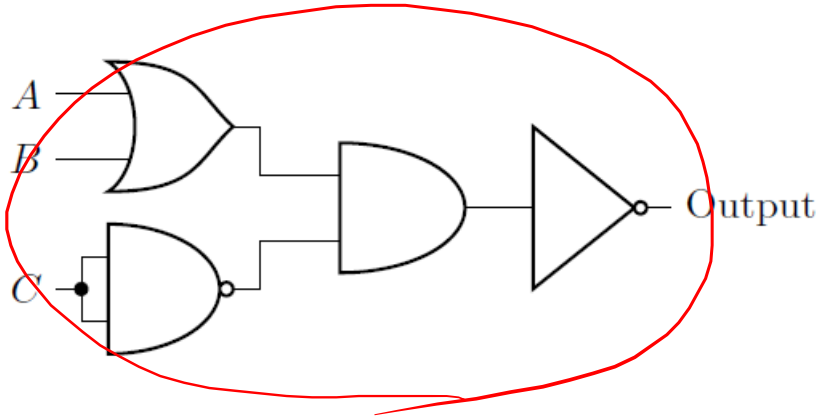
- Modern programming paradigm has many abstraction layers.
- Some programing language still works in very low level - e.g.) Verilog for FPGA coding





Classical computing : summary

- What computers do at very low level → Logic gates



- Algebras for digital computers → Boolean algebra

- High-level programming language should be translated into low-level machine language to be executed.

- *What if our information units DO NOT follow Boolean algebra?*

slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.

Qubits

- Qubit = Quantum bit

Dirac notation

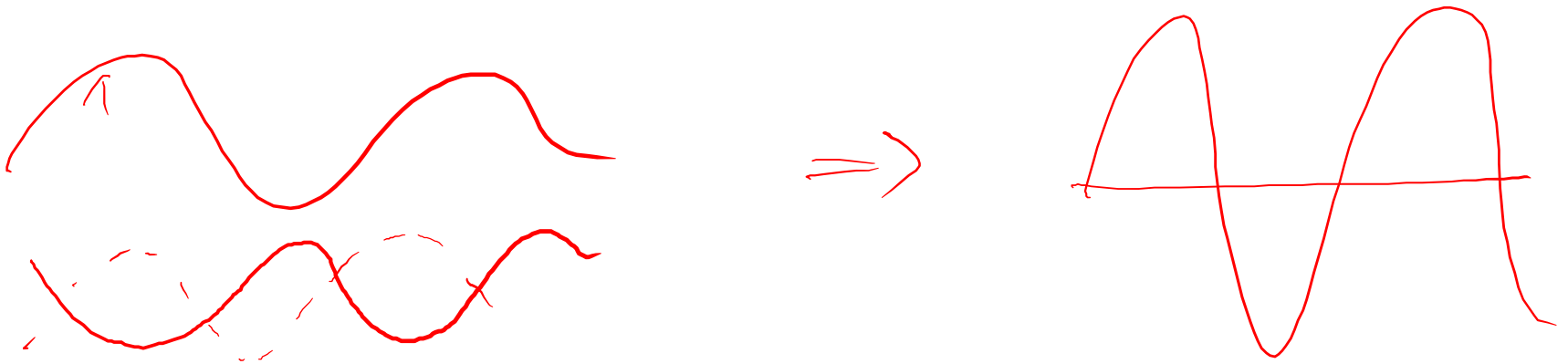
↓

- Classical bit : 0 or 1 vs. Quantum bit : $|0\rangle$, $|1\rangle$

Rules for Qubits (1) : Superposition

중첩.

- Superposition is general physical phenomena (ex. Constructive/destructive interference of sound wave)



- Superposition in QM : allow multiple state vectors with different measurement outcomes!

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

$\alpha, \beta \in \mathbb{C}$

복소수.

Not only 0 or 1

- Qubit state can be described by numerous superposition state other than just 0 and 1

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

- One simple rule for a legal (single) qubit state: $|\alpha|^2 + |\beta|^2 = 1$

$\alpha = a + bi$

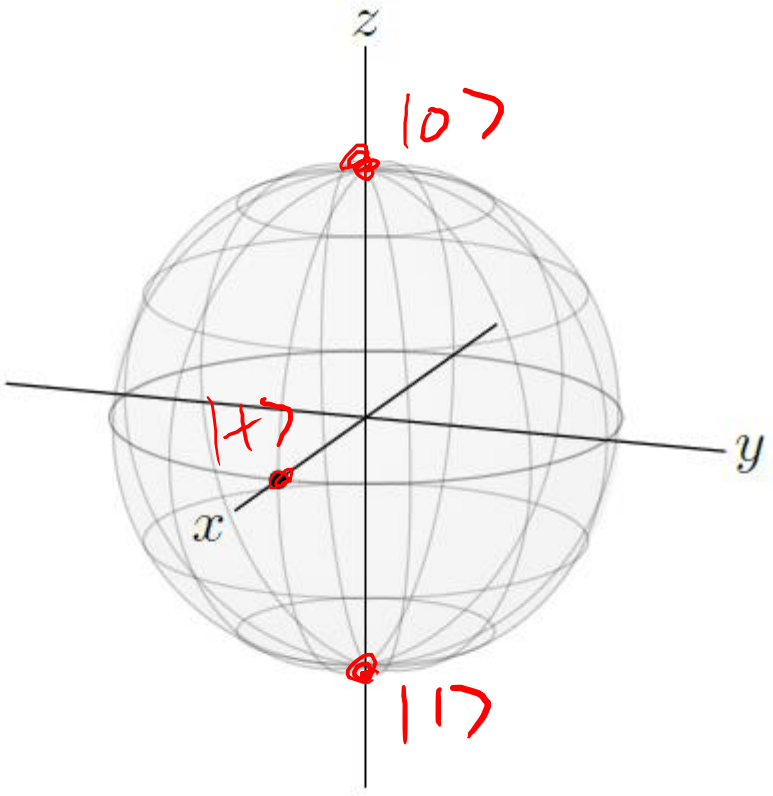
$|\alpha|^2 = a^2 + b^2$

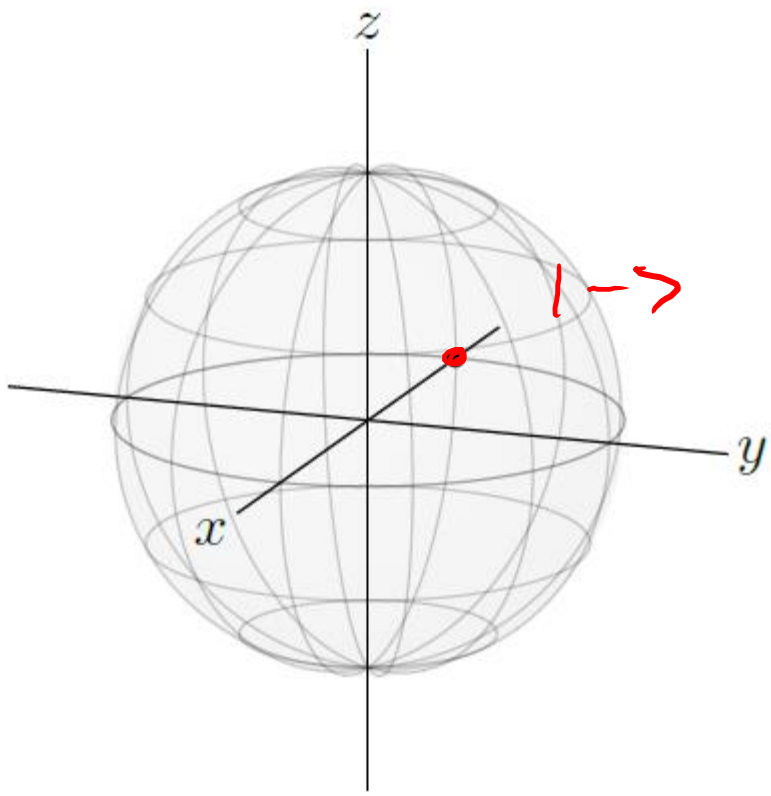
Not only 0 or 1

$\alpha = \beta = \frac{1}{\sqrt{2}}$

$\alpha = \frac{1}{\sqrt{2}}, \beta = -\frac{1}{\sqrt{2}}$.

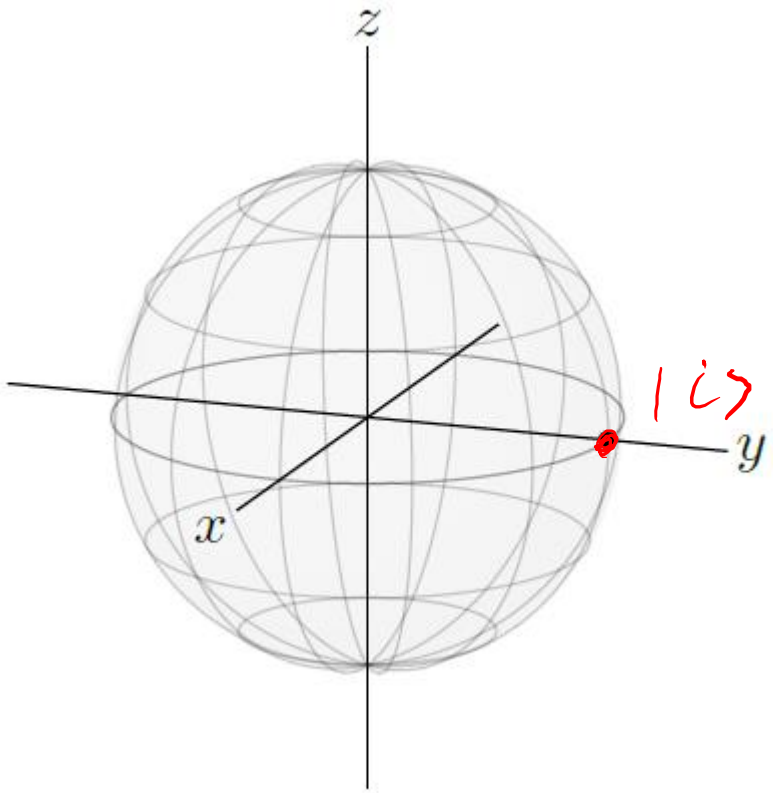
- $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

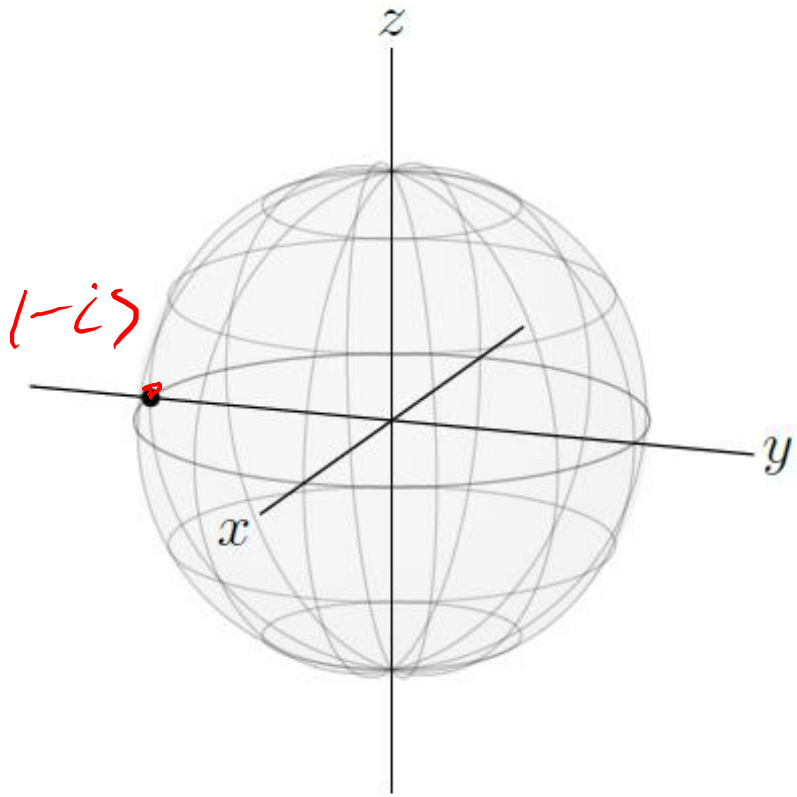




Not only 0 or 1

- $|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle), |-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$



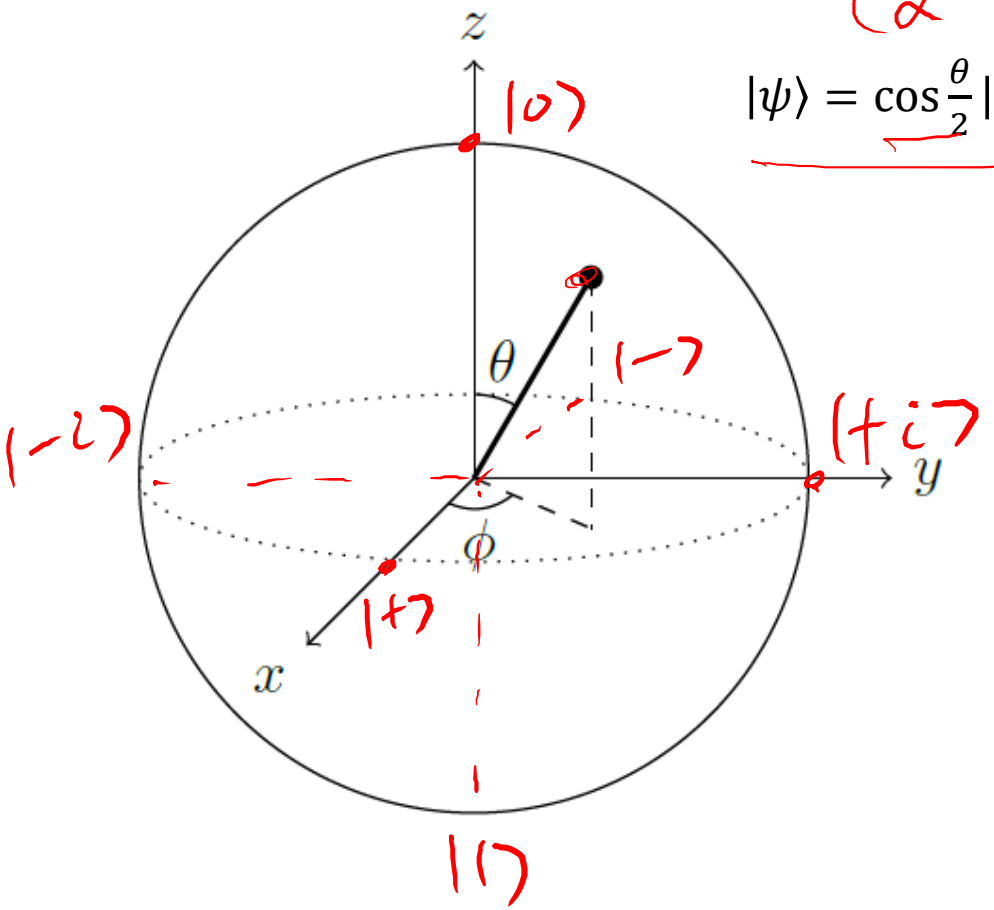


Not only 0 or 1

- Arbitrary single qubit states can be mapped on to Bloch sphere

$(\alpha, \beta) \longleftrightarrow (\theta, \varphi)$

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi}|1\rangle$$



Vector representation of qubits

ket

- $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow |\psi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$

- $|0\rangle = \begin{pmatrix}1\\ 0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\ 1\end{pmatrix}$

$|+\rangle = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$

bra

- $\langle\psi| = (\alpha^* \quad \beta^*)$

Distance between qubit states

- How similar are two qubit states?

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$|\psi'\rangle = \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}$$

$$|\langle\psi|\psi'\rangle|^2 = \left|\begin{pmatrix} \alpha^* & \beta^* \end{pmatrix}\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}\right|^2 = |\alpha^*\alpha' + \beta^*\beta'|^2$$

$0 \leq \quad \leq 1$

Fidelity

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\langle 0|1\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0.$$

Rules for Qubits (2) : Measurement

<|+|-> = 0

- Quantum measurement on qubits

0, 1

|x>, |x>

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

2가지

|+>, |->

$P_+ = |\langle +|\psi\rangle|^2$

$P_- = |\langle -|\psi\rangle|^2$

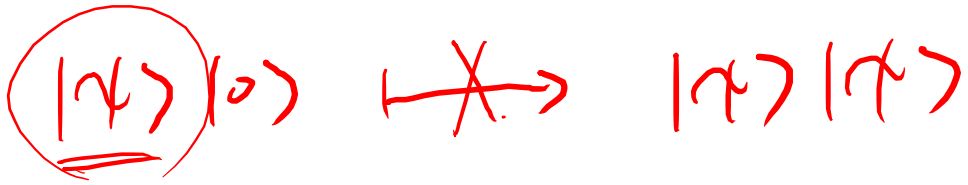
i) $P_0 = |\alpha|^2$, $P_1 = |\beta|^2$ 확률을 갖는다.

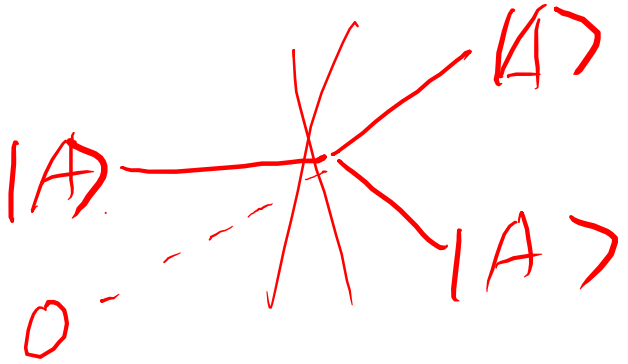
ii) 랜덤하게.

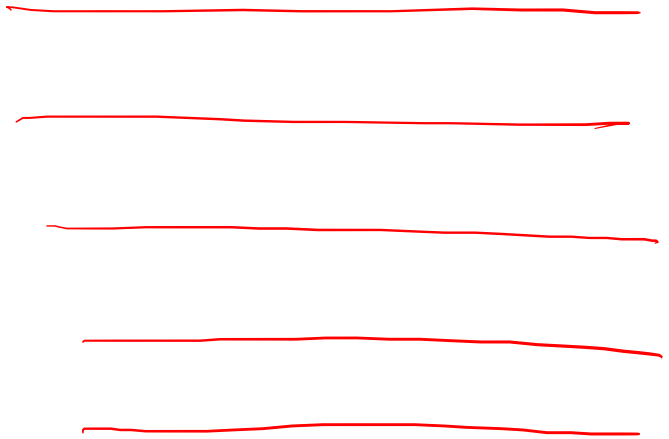
iii) $|\psi\rangle$ ⟼ $|0\rangle$ / $|1\rangle$ 둘 중 하나로.

Rules for Qubits (3) : No-cloning Theorem

- Quantum information cannot be copied (no fan-out)







Rules for Qubits (4) : More than one qubit

- Tensor product
  - $|0\rangle \otimes |0\rangle = |0\rangle|0\rangle = |00\rangle$: Both qubits are in 0 state. Tensor product
  - Basis for two qubits : $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$
  - $|\psi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle \rightarrow |c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$
- Other relevant notations
  - Binary to decimal : $|110\rangle = |6\rangle$ (little endian)
  - Powers : $|0\rangle^{\otimes n} = |00 \ldots 0\rangle$

Multi-qubit measurement

- Reminder : measuring $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$? → ? $P_0 = |\alpha|^2$, $P_1 = |\beta|^2$

- Multi-qubit states : $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|01\rangle + \frac{\sqrt{3}}{4}|10\rangle + \frac{1}{4}|11\rangle.$

C_{00}

$\downarrow$ $\frac{1}{2}$ $\frac{1}{4}$ $\frac{3}{16}$ $\frac{4}{16}$

00 01 10 11

Sequential Single-qubit Measurements

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|01\rangle + \frac{\sqrt{3}}{4}|10\rangle + \frac{1}{4}|11\rangle.$$

Entanglement

- Product state

- Entangled state

Bell states

$$\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right),$$

$$\left|\Phi^{-}\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right),$$

$$\left|\Psi^{+}\right\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right),$$

$$\left|\Psi^{-}\right\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right).$$

Qubits : summary

- Rules for qubits:

    1. Superposition : $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
    2. Measurement
    3. No-cloning theorem
    4. Multi-qubit representation

- Algebra for qubits : Linear algebra

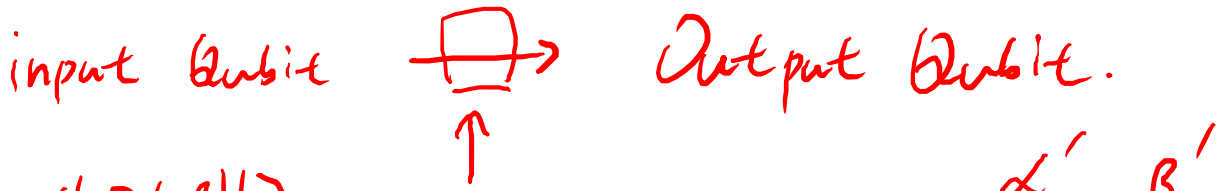
slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.

Quantum gate

- Quantum gates transfer one qubit state to other (c.f. logic gates)



A	$\bar{A}$
0	1
1	0

$\|A\rangle$	output.
$\|0\rangle$	$\|\psi_0\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$
$\|1\rangle$	$\|\psi_1\rangle = \begin{pmatrix}\alpha'\\ \beta'\end{pmatrix}$

- Quantum gates on superposition states

$$G|0\rangle = |\psi_0\rangle$$
$$G|1\rangle = |\psi_1\rangle$$

$$G(\alpha|0\rangle + \beta|1\rangle) = \alpha(G|0\rangle) + \beta(G|1\rangle) = \alpha|\psi_0\rangle + \beta|\psi_1\rangle$$

Single qubit quantum gates

- Identity gate

$$I|0\rangle = |0\rangle,$$
$$I|1\rangle = |1\rangle.$$

- Pauli X gate (NOT gate)

$$X|0\rangle = |1\rangle,$$
$$X|1\rangle = |0\rangle.$$

$X|-\rangle = X\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}|1\rangle - \frac{1}{\sqrt{2}}|0\rangle = -|-\rangle$

Single qubit quantum gate

- Pauli Y gate

$$Y|0\rangle = i|1\rangle,$$
$$Y|1\rangle = -i|0\rangle.$$

in	out
$\|0\rangle$	$i\|1\rangle$
$\|1\rangle$	$-i\|0\rangle$

- Pauli Z gate

$$Z|0\rangle = |0\rangle,$$
$$Z|1\rangle = -|1\rangle.$$

in	out
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-\|1\rangle$

$$Z|+\rangle = Z\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = |-\rangle$$

Single qubit quantum gate

- Phase gate (S gate)

$Z = S^2$

$$S|0\rangle = |0\rangle,$$
$$S|1\rangle = i|1\rangle.$$

$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$i\|1\rangle$

- T gate (Magic gate)

$S = T^2$

$$T|0\rangle = |0\rangle,$$
$$T|1\rangle = e^{i\pi/4}|1\rangle.$$

$\sqrt{i}$

$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$e^{i\pi/4}\|1\rangle$

$e^{i\theta} = \cos\theta + i\sin\theta$

Matrix representation of Single qubit gate

- Results of single qubit gates should be also legal single qubit states

$$G|0\rangle = g_{00}|0\rangle + g_{01}|1\rangle$$

$$G|1\rangle = g_{10}|0\rangle + g_{11}|1\rangle$$

$$G(\alpha|0\rangle + \beta|1\rangle) = (g_{00}\alpha + g_{01}\beta)|0\rangle + (g_{10}\alpha + g_{11}\beta)|1\rangle = \alpha'|0\rangle + \beta'|1\rangle$$

$$\begin{pmatrix}\alpha'\\ \beta'\end{pmatrix} = \begin{pmatrix}g_{00} & g_{01}\\ g_{10} & g_{11}\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

output qubit, 2X2 matrix, input qubit.

$G(G^*)^T = I$

$G^\dagger$

Unitary

Single qubit quantum gate

- Superposition of Quantum gates : $G = G_1 + G_2$

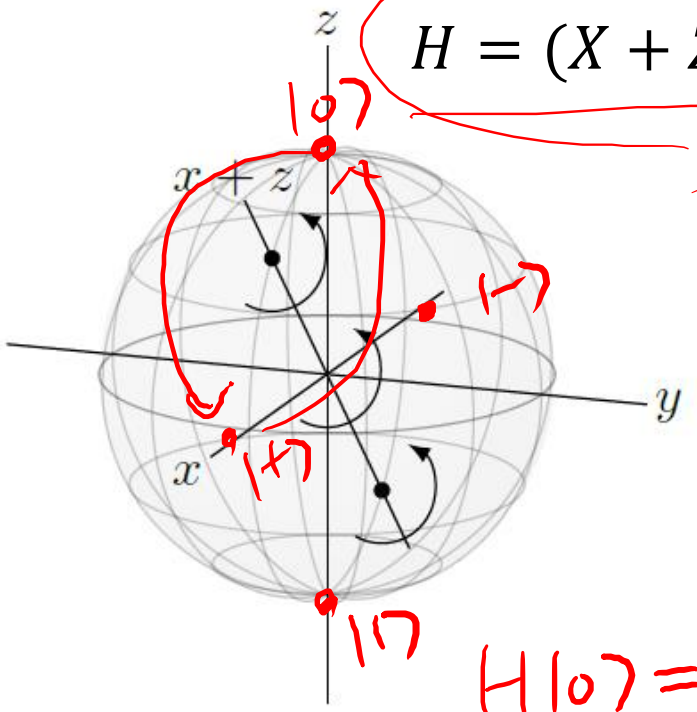
$|0\rangle, |1\rangle \longmapsto \alpha|0\rangle + \beta|1\rangle$

$G_1, G_2 \longmapsto aG_1 + bG_2$

$(aG_1 + bG_2)|\psi\rangle = a(G_1|\psi\rangle) + b(G_2|\psi\rangle)$

Single qubit quantum gate

- Hadamard gate (H gate)



$$H = (X + Z)/\sqrt{2}$$

in	out
$\|0\rangle$	$\|+\rangle$
$\|1\rangle$	$\|-\rangle$

$H|-\rangle = |1\rangle$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

$H|0\rangle = \frac{1}{\sqrt{2}}(X+Z)|0\rangle = \frac{1}{\sqrt{2}}(X|0\rangle + Z|0\rangle) = \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) = |+\rangle$

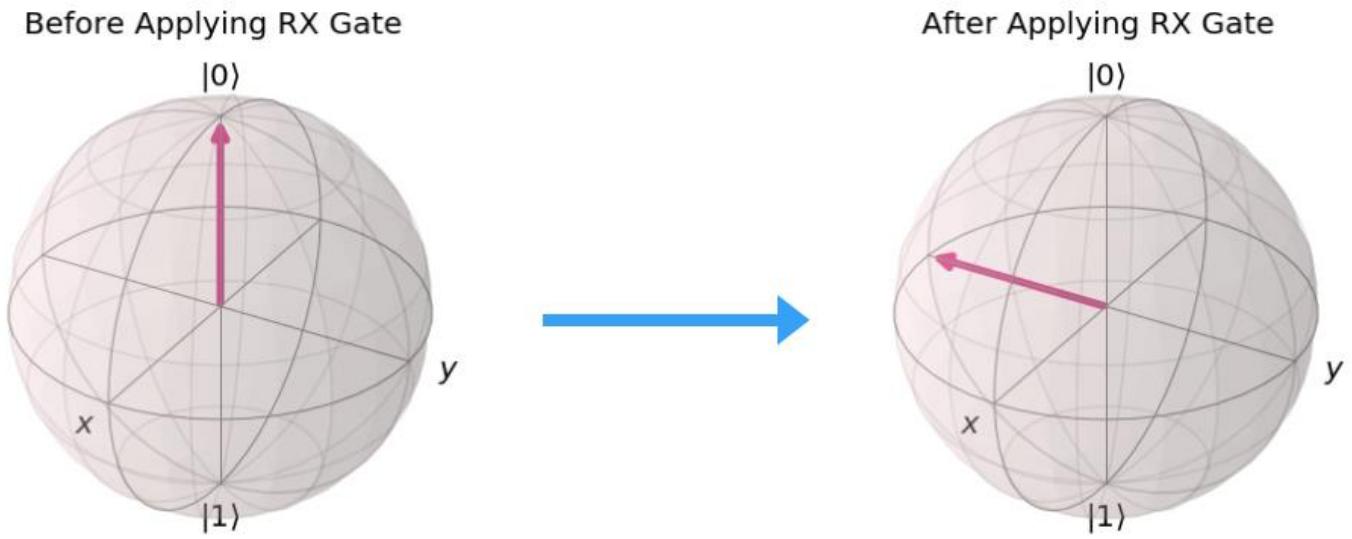
$H|+\rangle = H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}|+\rangle + \frac{1}{\sqrt{2}}|-\rangle = |0\rangle$

https://algassert.com/quirk

https://algassert.com/quirk

Single qubit quantum gate

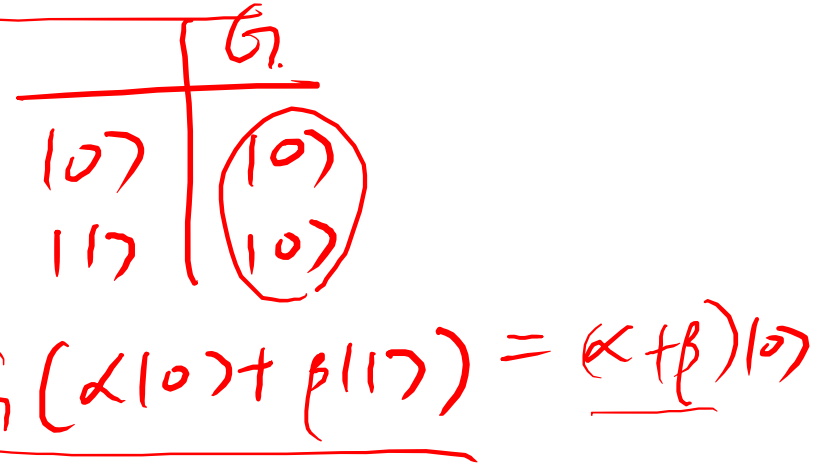
- Rotation X (Y, Z) : $R_x(\theta) = \cos\frac{\theta}{2} I + i \sin\frac{\theta}{2} X$



- Any single qubit gate can be represented as rotation on the Bloch sphere

State initialization and measurement

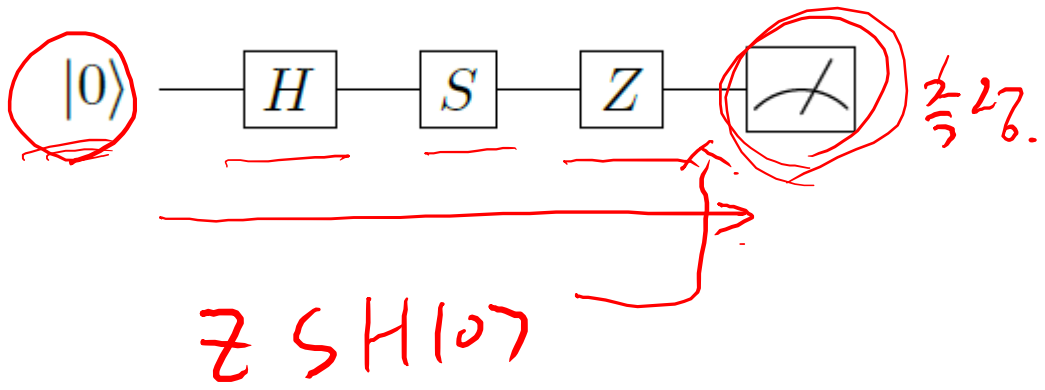
- State initialization : reset qubit state into $|0\rangle$



- State measurement : “collapse” qubit states into $|0\rangle$ or $|1\rangle$

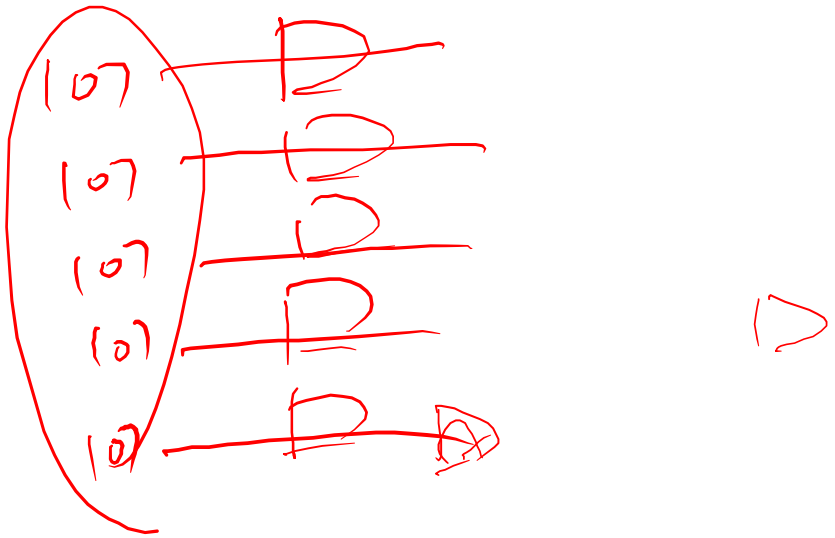
Quantum Circuit

- Alike classical logic circuit, quantum circuits consist of qubits and quantum gates.
- One difference is measurement operator, which did not exist in logic circuit.



Quantum gates on multi-qubits

- Single-qubit quantum gates on multi-qubit states



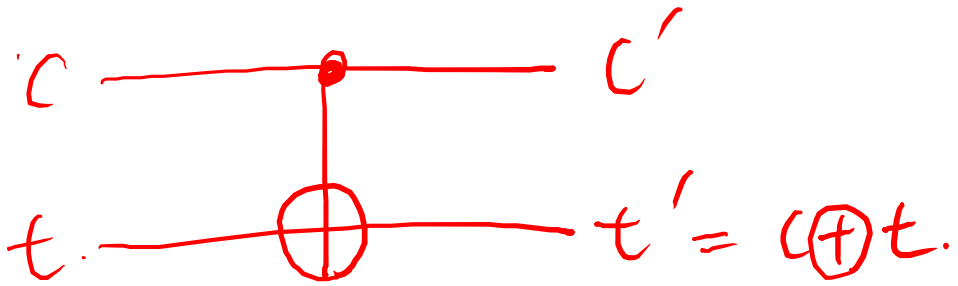
Multi-qubit quantum gate

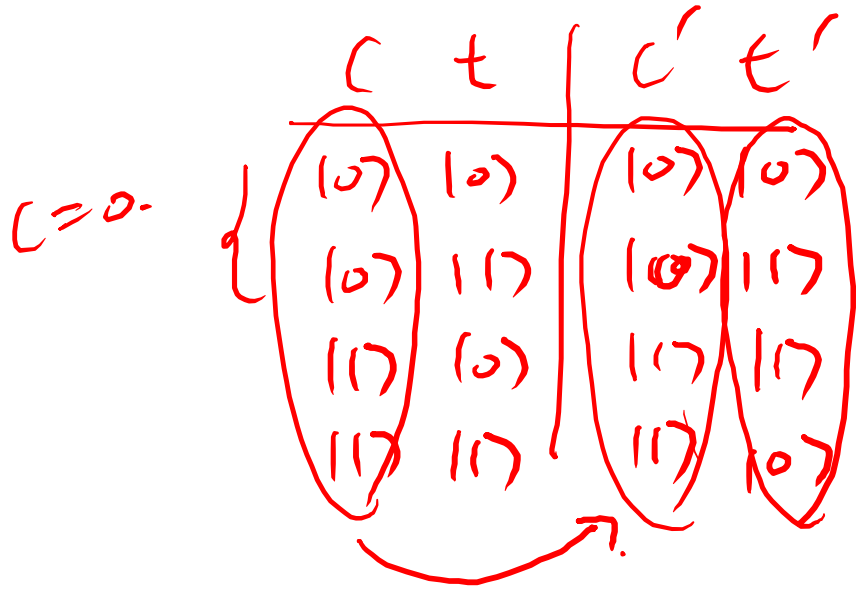
- Multi-qubit gate : Quantum gate acting on multiple qubits
    - c.f.) multi-bit logic gate : AND / OR

- Reversible, Unitary
- Can generate/eliminate quantum entanglement! (single qubit gates X)

Controlled- quantum gates

- Controlled NOT or CNOT

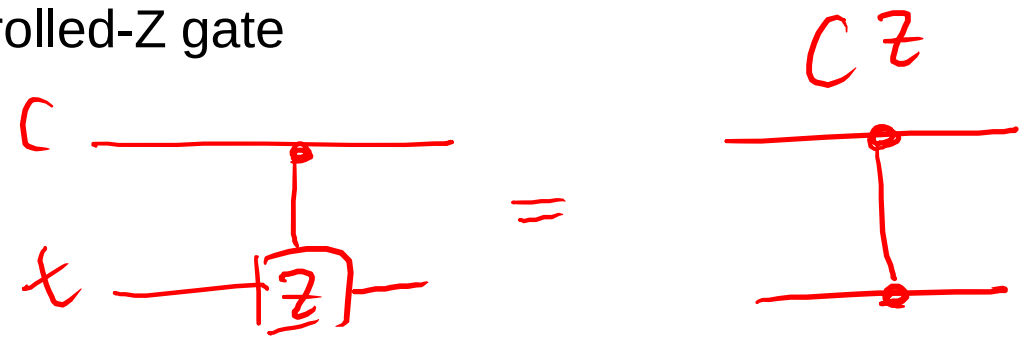




- CNOT vs. XOR

Controlled- quantum gates

- Controlled-Z gate

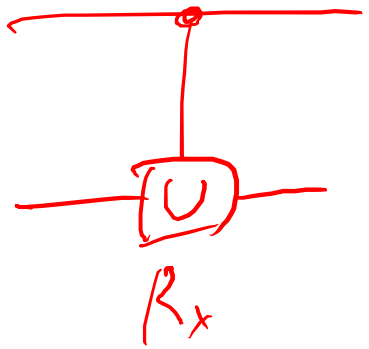


ct	$c't'$
$\|0\rangle\|0\rangle$	$\|0\rangle\|0\rangle$
$\|0\rangle\|1\rangle$	$\|0\rangle\|1\rangle$
$\|1\rangle\|0\rangle$	$\|1\rangle\|0\rangle$
$\|1\rangle\|1\rangle$	$-\|1\rangle\|1\rangle$

$|1\rangle \otimes (Z|1\rangle) = |1\rangle \otimes (-|1\rangle)$

$= -|1\rangle \otimes |1\rangle$

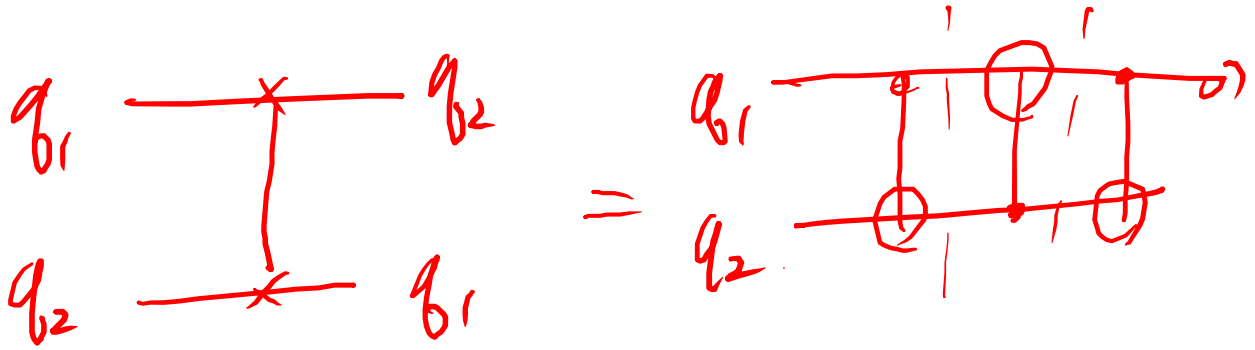
- Controlled-U gate



ct	$c't'$
$\|0\rangle\|0\rangle$	$\|0\rangle\|0\rangle$
$\|0\rangle\|1\rangle$	$\|0\rangle\|1\rangle$
$\|1\rangle\|0\rangle$	$\|1\rangle \otimes (U\|0\rangle)$
$\|1\rangle\|1\rangle$	$\|1\rangle \otimes (U\|1\rangle)$

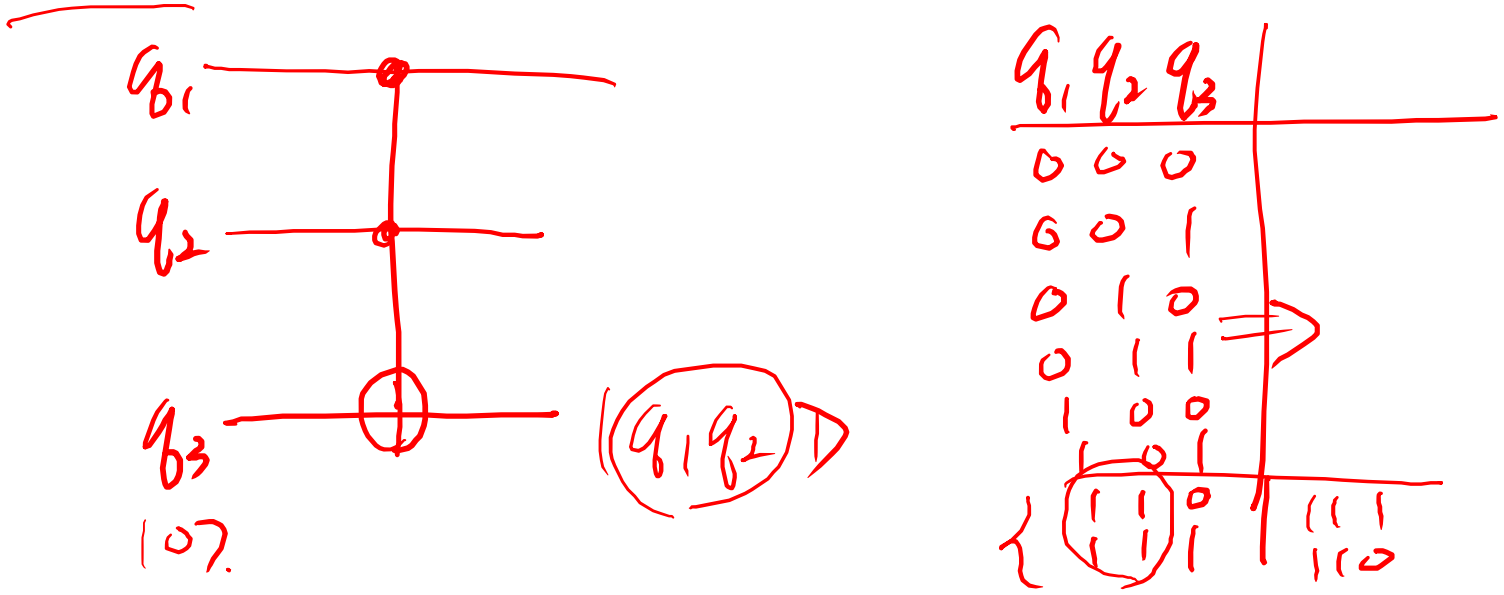
Other multi-qubit gates

- SWAP gate



- Toffoli gate (Quantum AND)

CCX



Universality of quantum gates

- One can build arbitrary quantum gates with finite set of quantum gates
- {CNOT, all single qubit gates}
- {CNOT, H, T}
- {Toffoli, H, S}

Entangling qubits with quantum gates

- Bell state generation

- Greenberger–Horne–Zeilinger (GHZ) state

Quantum Gates & Circuits : Summary

- Quantum circuit is a sequence of abstracted quantum instructions.
  - Qubit initialization
  - Single qubit gates
  - Multi-qubit gates
  - Qubit measurements

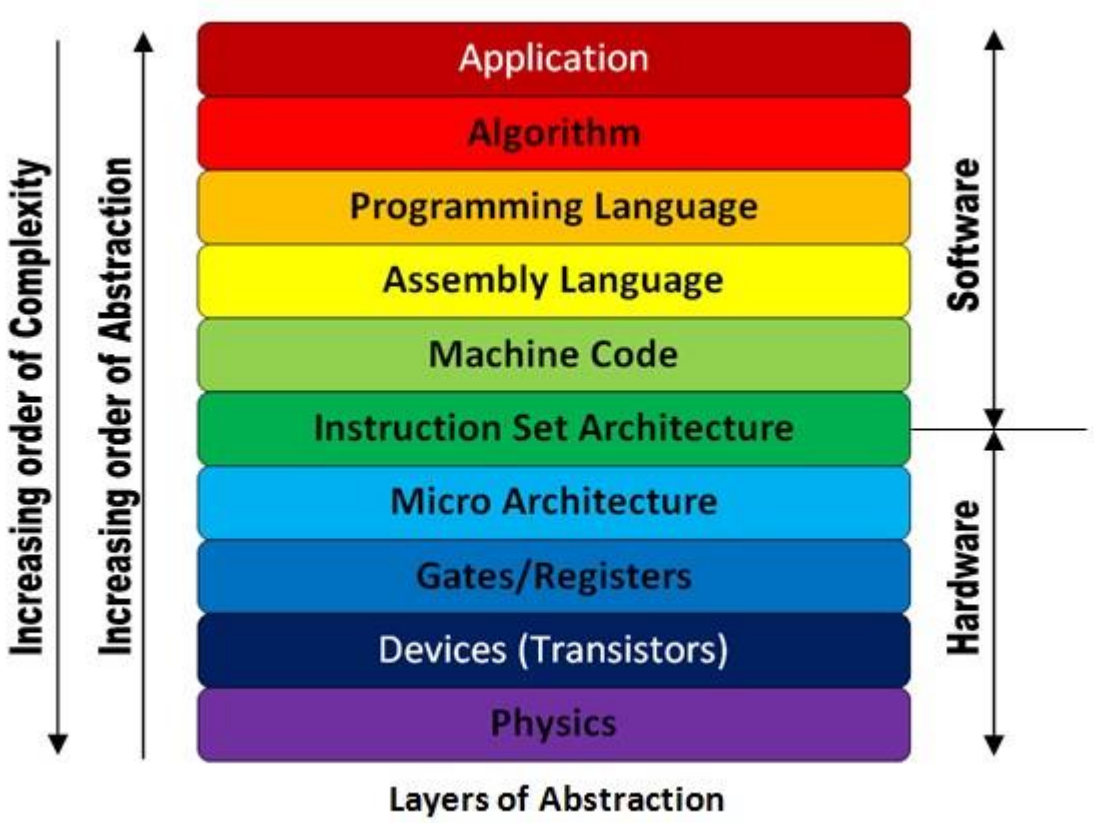
Quiz!

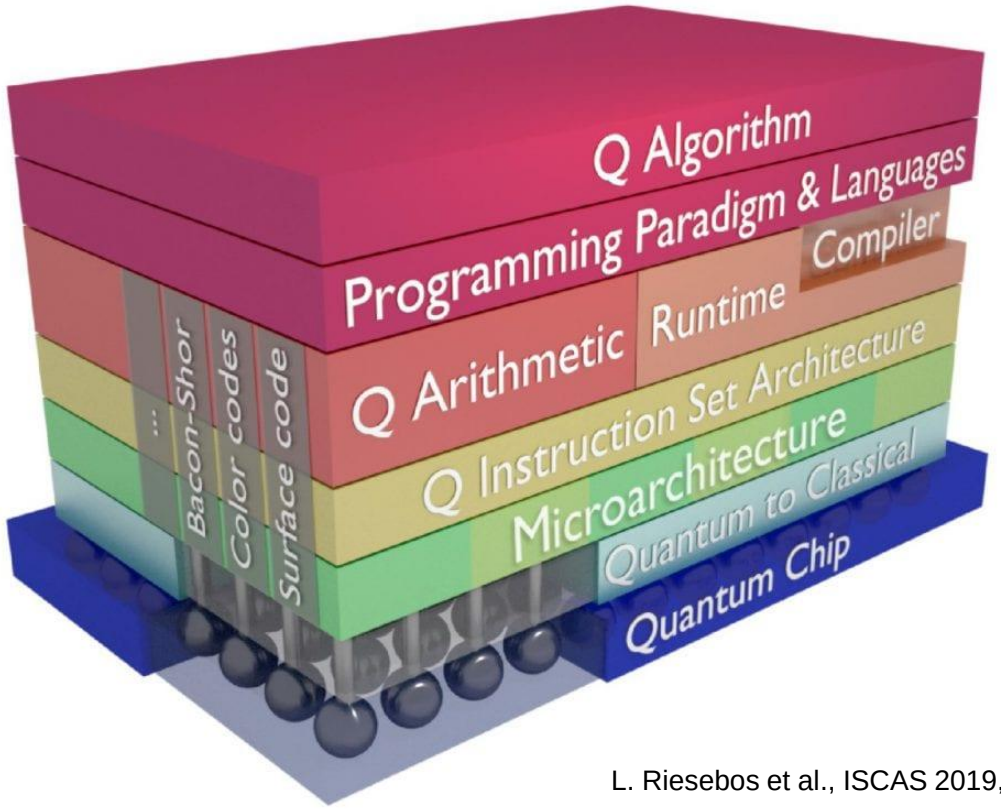
slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.

Computing architecture : Classical vs. Quantum



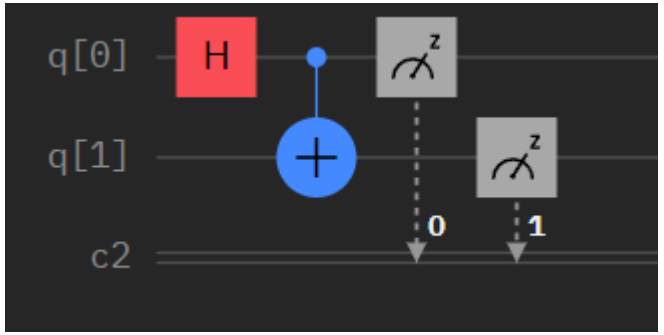


L. Riesebos et al., ISCAS 2019, pp. 1-4

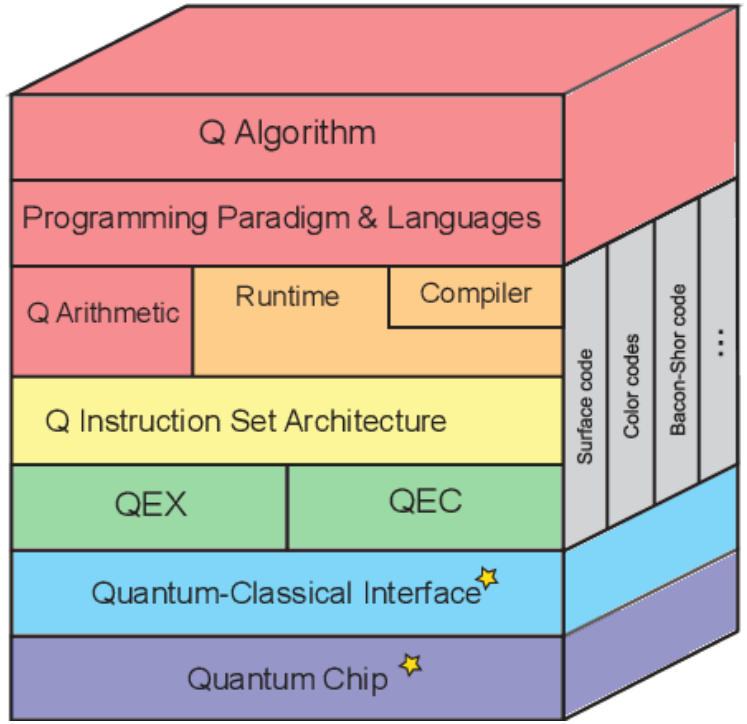
- Abstracted instructions for classical computer : Logic circuits
- Abstracted instructions for quantum computer : Quantum circuits

Quantum assembly language

- Programming languages designed for describing quantum instructions (circuits)
- OpenQASM, cQASM, Quil, Blackbird, Jaqal, etc.
- Mostly platform-agnostic (for gate-base QC)




```
1   OPENQASM 2.0;
2   include "qelib1.inc";
3
4   qreg q[2];
5   creg c[2];
6   h q[0];
7   cx q[0], q[1];
8   measure q[0] -> c[0];
9   measure q[1] -> c[1];
10
```



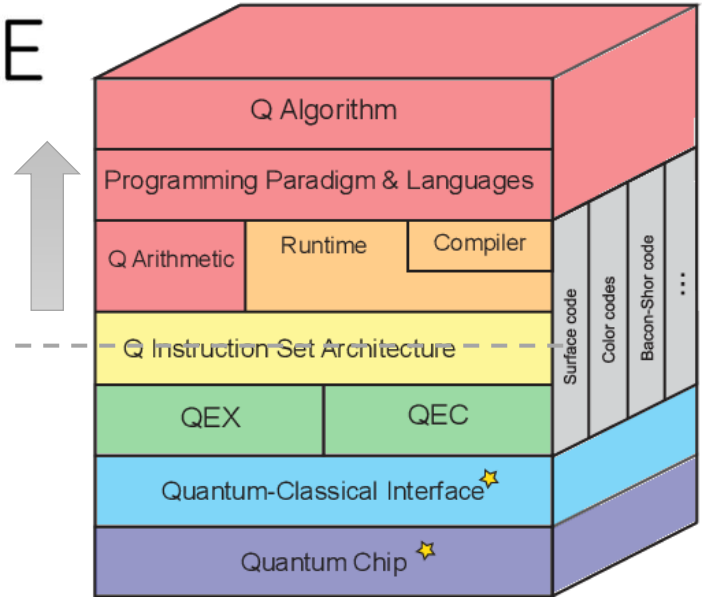
Toward higher level : Quantum SDKs

- Quantum SDK (Software Development Kits) helps users to write quantum programs (= quantum circuits)
- They provides useful features like:
  - Circuit optimizations
  - Gate decomposition
  - Simulation of quantum circuits on classical computers
  - Submit jobs to actual quantum computers on network
- **Mostly used in this winter school!**
- Not much abstraction has been made for quantum variables or functions. (Qint? Qfloat?)

Qiskit

Cirq

PENNYLANE



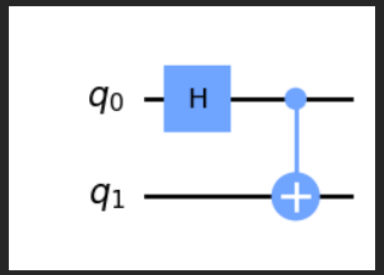
IBM Qiskit

- Open-source quantum SDK made by IBM researchers and collaborators.
  - Documentations : https://docs.quantum.ibm.com/
- Two main parts:
  - QuantumCircuit
    - Container for quantum circuit description
    - Pretty easy to use. See API reference.
  - Backend & Job
    - Backend objects represent quantum computer or simulator, which can execute QuantumCircuit contents
    - Each submission returns a Job object to track result asynchronously.

[1]:

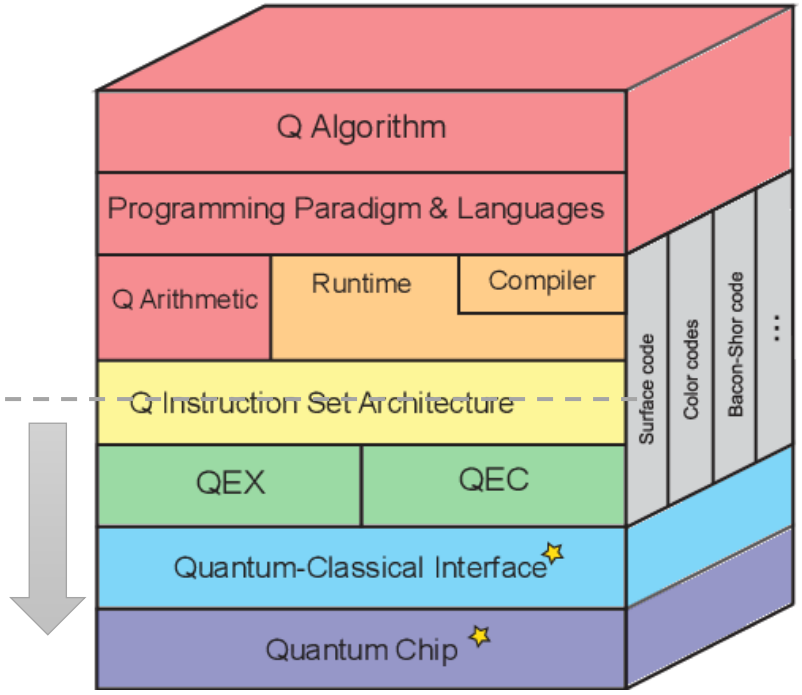
```
from qiskit import QuantumCircuit

# Create a new circuit with two qubits (first argument) and two classical
# bits (second argument)
qc = QuantumCircuit(2)

# Add a Hadamard gate to qubit 0
qc.h(0)

# Perform a controlled-X gate on qubit 1, controlled by qubit 0
qc.cx(0, 1)

# Return a drawing of the circuit using MatPlotLib ("mpl"). This is the
# last line of the cell, so the drawing appears in the cell output.
# Remove the "mpl" argument to get a text drawing.
qc.draw("mpl")
```

Output:



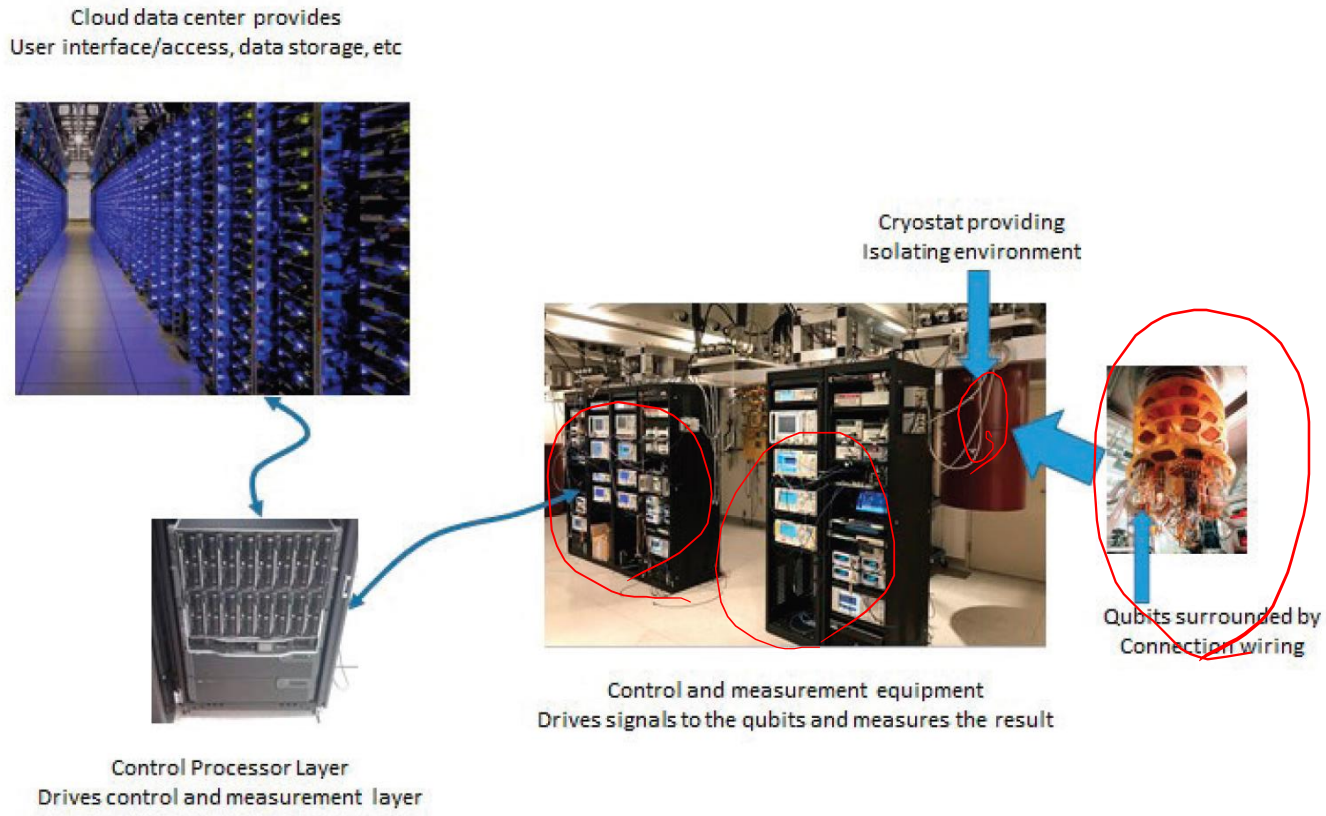
Toward lower level : Control stack + Hardware

- To execute quantum program, instructions should be translated into actual physical sequence onto qubits.
  - Qubit mapping
  - Topological constraints resolving
  - Physical-gate decomposition
  - Physical-level optimization
- Not only hardware work, but also extensive software automation & optimization (like OS).



arXiv:2011.00028v1

Toward lower level : Control stack + Hardware



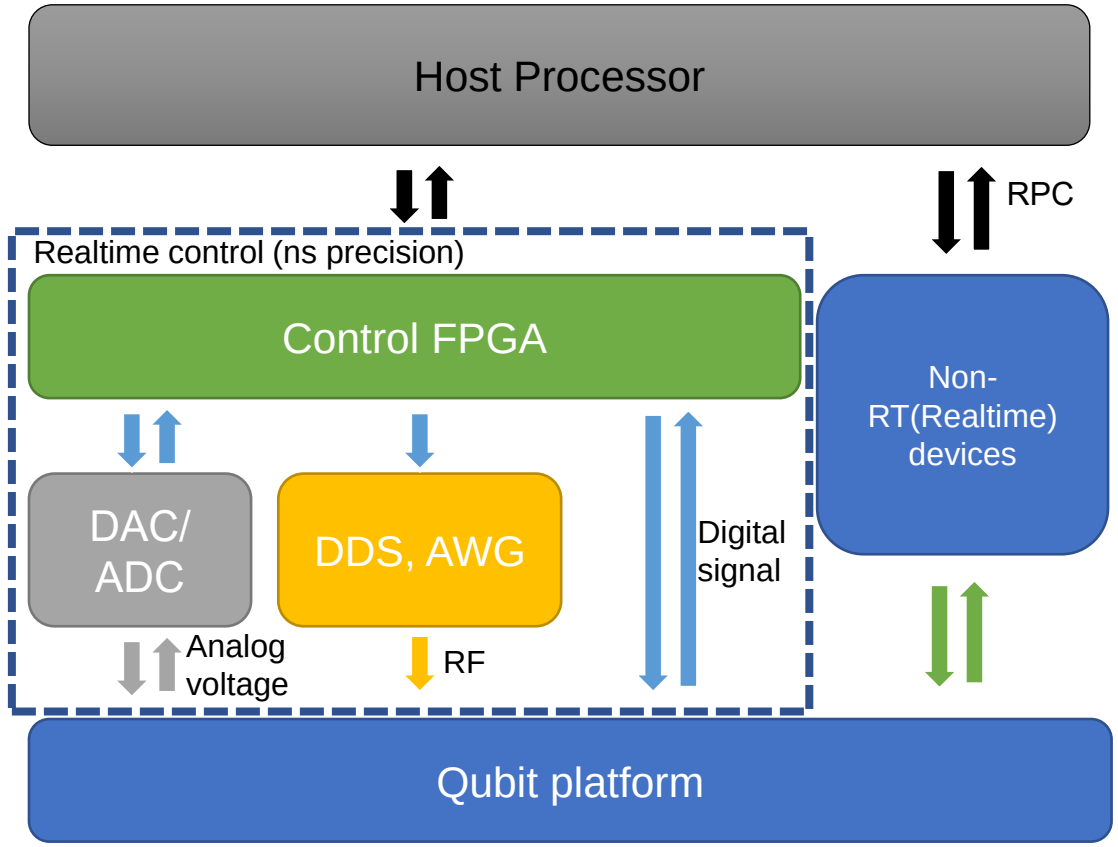
IBM Quantum Two

- Making qubits is hard, but controlling them very well is much harder.
- Calibration, exact control, tight scheduling, error mitigation, etc.

One example : IBM Quantum (Superconducting)

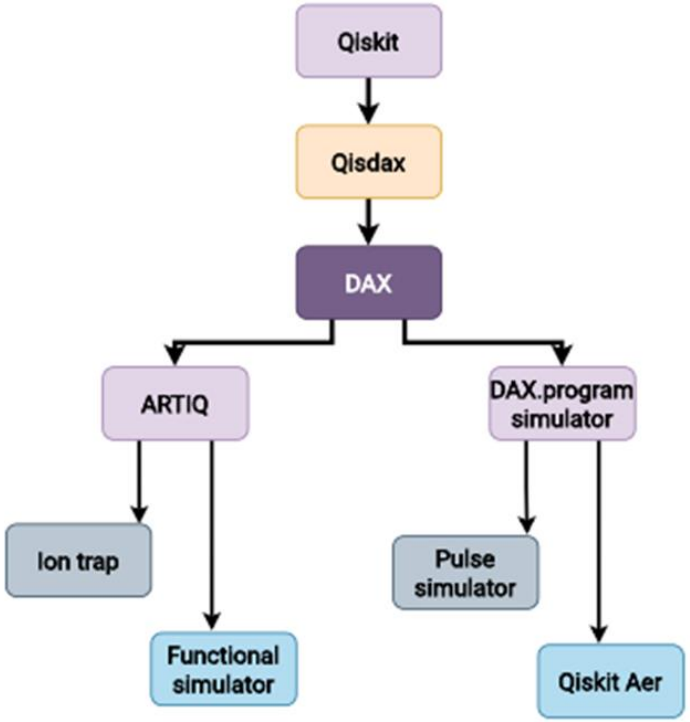
One example : Trapped-ion system

- Preparation stage
  - Check whether the trapped ion qubits are enough (if not, trap ions)
  - Check when the latest calibration was or whether it is correct (if not, re-calibrate them)
  - Compile control pulses (frequency, amplitude, pulse length) with calibration parameters
- Run stage
  - Initialize qubits states into $|0\rangle$ with a laser pulse
  - Apply control laser pulses with nano-second precision via FPGA
  - Perform qubit readout and store the result (image or photon counts) into classical register
  - (if possible) Check whether the whole process has gone well (no qubit lost, system drift, etc)
- Analyze stage
  - Retrieve stored register values and convert them into measurement results.



ARTIQ-DAX layer abstraction

QISDAX software stack



Qiskit

```python
from qiskit import QuantumCircuit, execute
from qiskit.visualization import plot_histogram
from qiskit.providers.dax import DAX

dax = DAX.get_provider()
backend = dax.get_backend('dax_artiq_device')
backend.load_config("resources.toml")

qc = QuantumCircuit(2,2)
qc.h(0)
qc.cx(0, 1)

qc.measure([0, 1], [0, 1])

dax_job = execute(qc, backend, shots=1000)
result = dax_job.result()
counts = result.get_counts()
print(f'Result: {counts}')

print(qc)
qc.draw(output='mpl').show()
plot_histogram(counts).show()
```

Automatic Transpile

ARTIQ Code(Dax)

```python
"""
Program generated on  by QisDax v.
"""

from dax.program import *


class QisDaxProgram(DaxProgram, Experiment):

    def build(self):
        self._num_iterations = 1000
        self._num_qubits = 2
        self.update_kernel_invariants('_num_iterations', '_num_qubits')
        self._classical_regs = set()

    def run(self):
        # Check number of qubits
        if self._num_qubits > self.q.num_qubits:
            raise RuntimeError('Circuit requires more qubits than the system supports')

        # Run the kernel
        self._run()

    @kernel
    def _run(self):
        t0 = self.core.get_rtio_counter_mu()
        self._qiskit_kernel()

        self.core.wait_until_mu(now_mu())
        t1 = self.core.get_rtio_counter_mu()
        self.set_dataset("dt", self.core.mu_to_seconds(t1 - t0))

    @kernel
    def _qiskit_kernel(self):
        with self.data_context:
            for _ in range(self._num_iterations):
                self.core.reset()
                self.q.prep_0_all()

                self.q.h(0)
                pass
                self.q.cnot(0,1)
                pass
                with parallel:
                    with sequential:
                        self.q.m_z(0)
                        pass
                        self.q.store_measurements([0])
                    with sequential:
                        self.q.m_z(1)
                        pass
                        self.q.store_measurements([1])
```

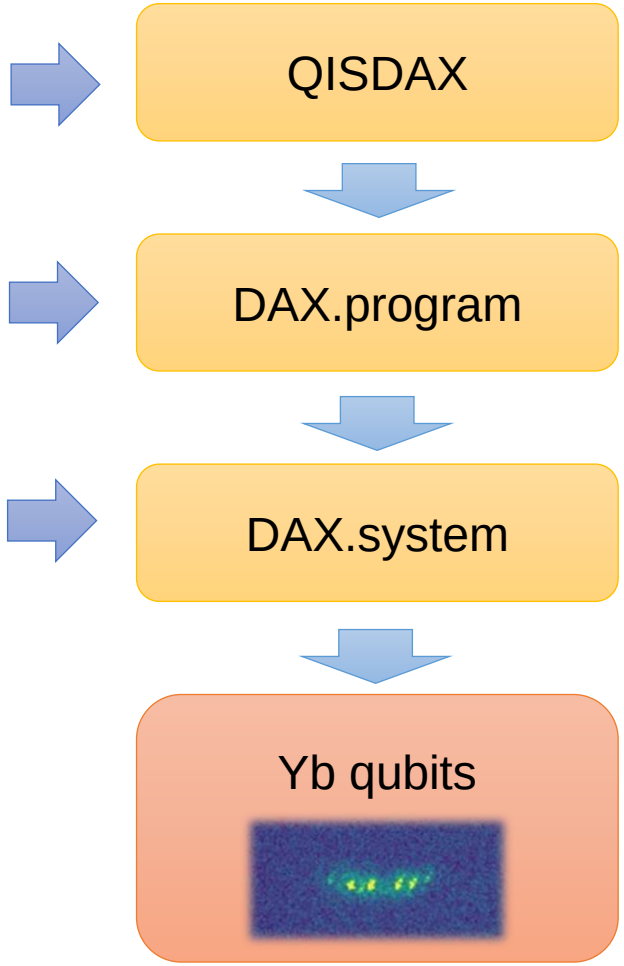
- Recent work by NC State University enables Qiskit-based quantum circuit execution (QISDAX)
- We tested the QISDAX with our system simulator

ARTIQ-DAX layer abstraction

Quantum Circuit (using Qiskit)

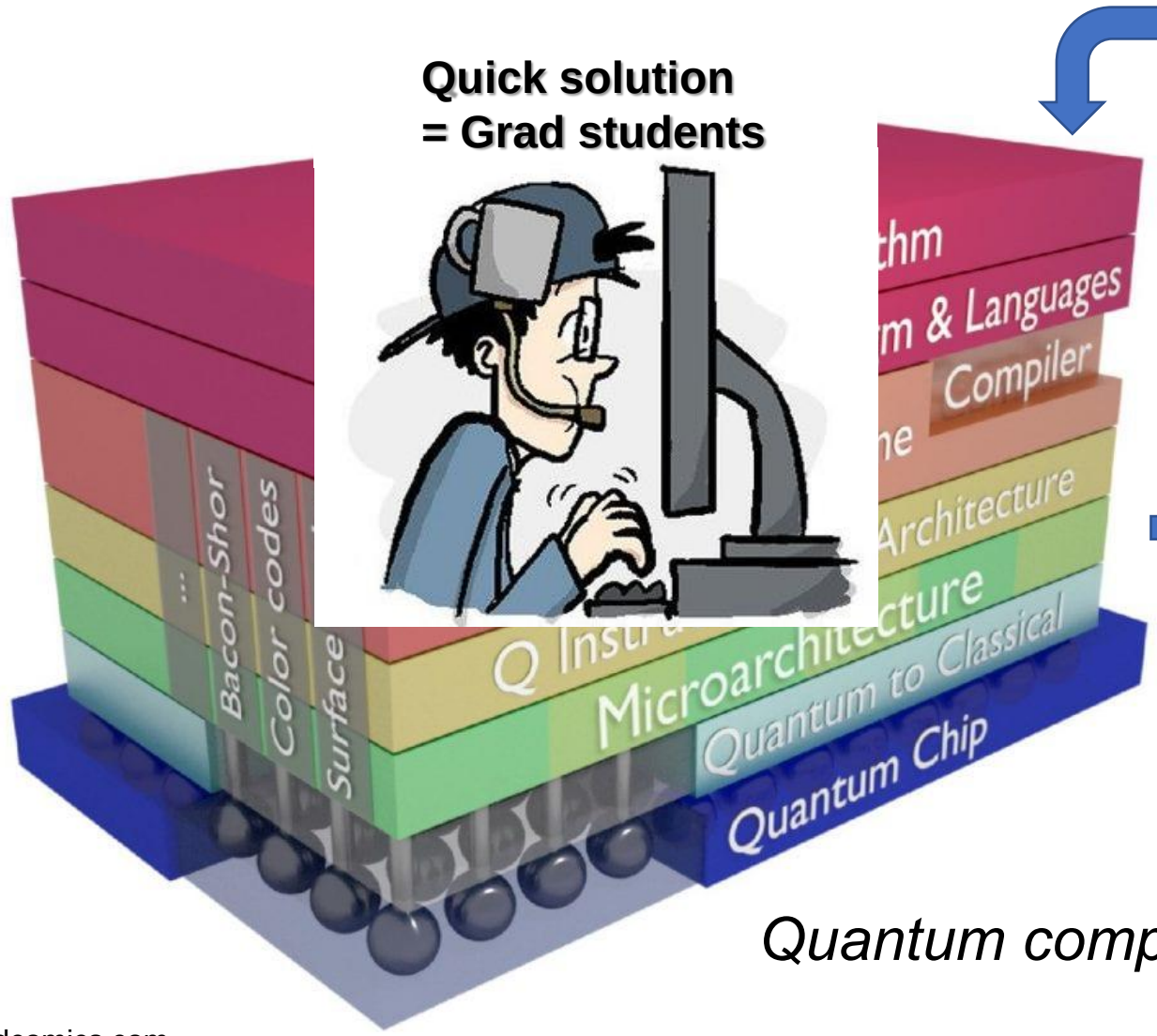
Hardware-specific Control (Pulse shaping, Composite pulse, etc.)

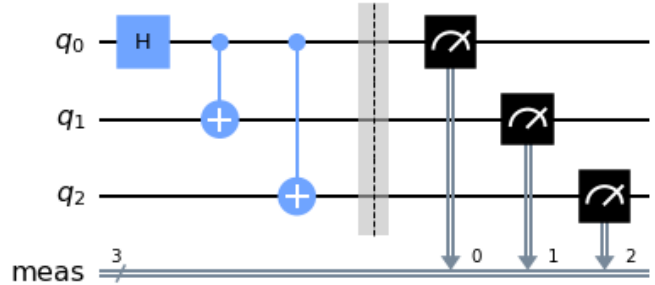
Full access to real hardware



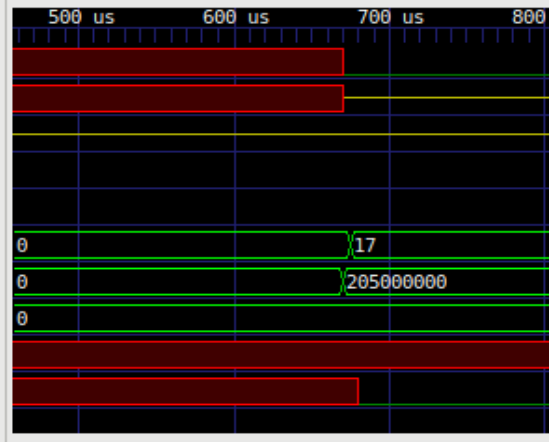
- Multiple abstraction layer defines various access levels to the ion-trap QC device
- One could remotely submit jobs to the device via QISDAX or DAX.program
- Only functional simulator available now. (system simulator could be useful tho!)

Unfolding quantum circuits






```
Signals
Time
      sensitivity =0
            value =z
            count =19
} pmt
detection_dds {
              att =17
             freq =205000000
            phase =0
             init =x
            value =1
```



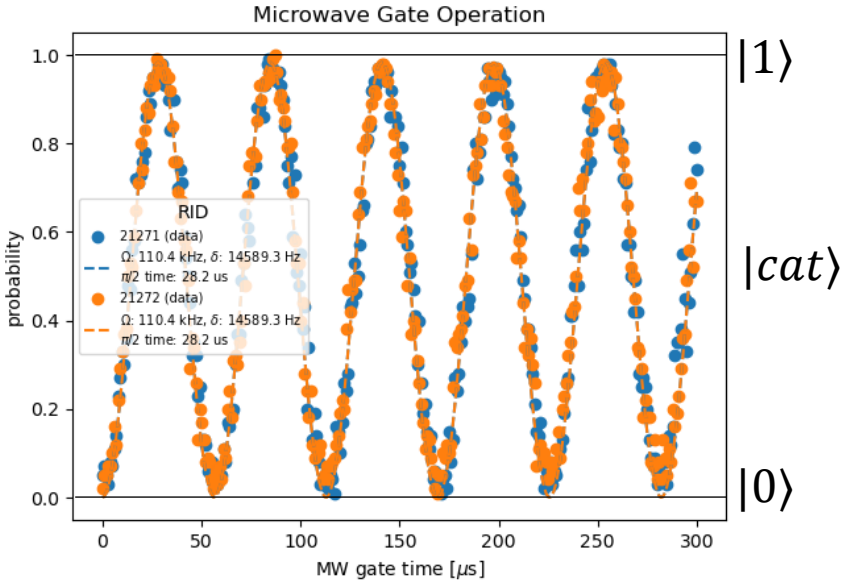
Hardware waveform

Quantum computing stack should be dynamic not static.

www.phdcomics.com

Quantum Engineering with Trapped Ions (QuETI) Lab @ SKKU

- Research Topic : Full-stack Ion-trap Quantum Computing
  - Build a system from bottom to top
  - Exciting research field for both quantum science and engineering!
  - Superposition ✔, Entangling operation comes soon!



Website: https://iontrap.skku.edu
Lab : 83292/83245
Office : 83212

We're looking for motivated undergrads & graduate students!
Contact info: Junki.kim.q@skku.edu

Quantum computing architecture : summary

- Programming language for quantum circuits : quantum assembly language
  - Mostly for gate-based quantum computers
- Quantum SDKs : handy utilities + functionalities for quantum coding
  - Not much abstraction made yet. The code is virtually the same with quantum circuit.
  - Useful to learn & code quantum algorithms
  - Examples : Qiskit, Cirq, Pennylane, etc.
- Hardware-dependent software stack : important and interesting engineering problem
  - Especially important for NISQ devices (limited performance)

slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.