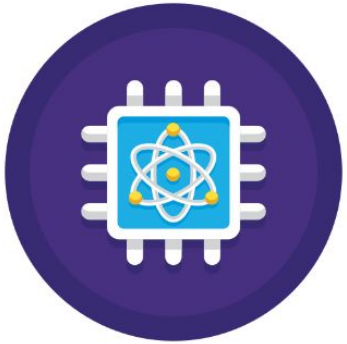
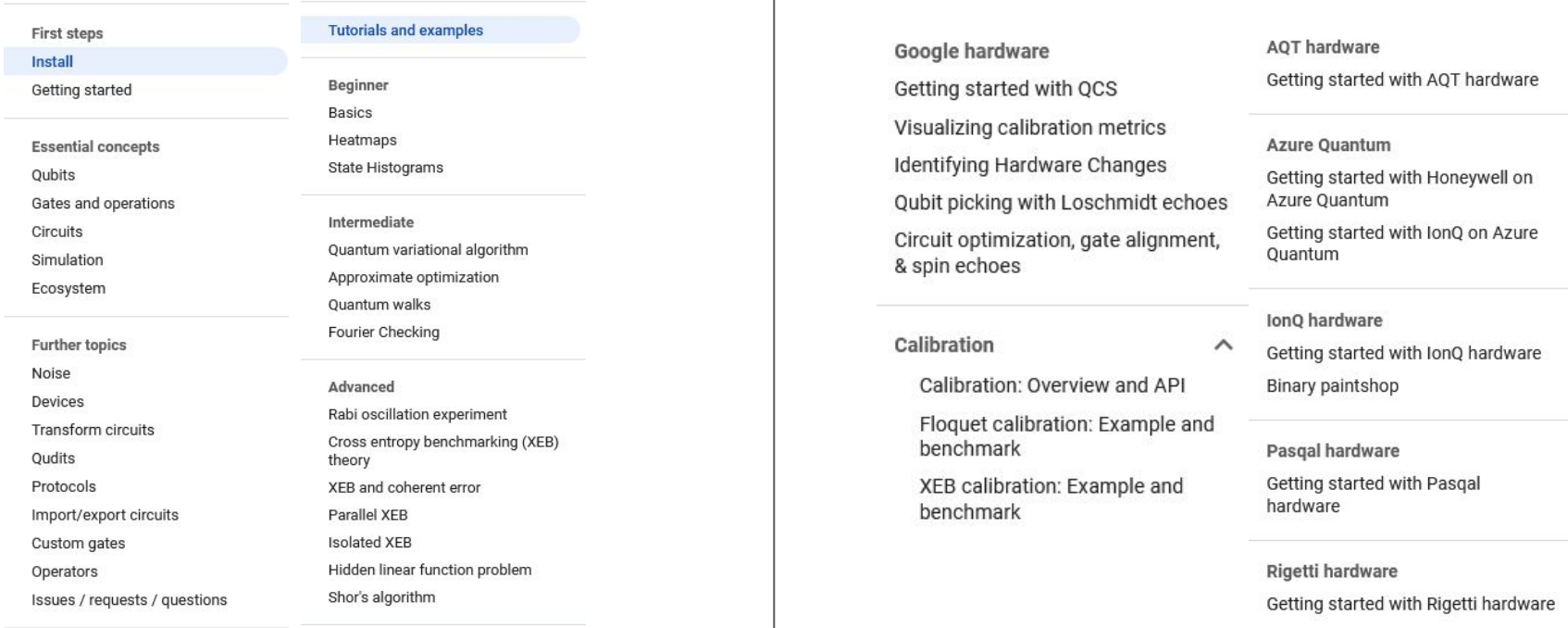
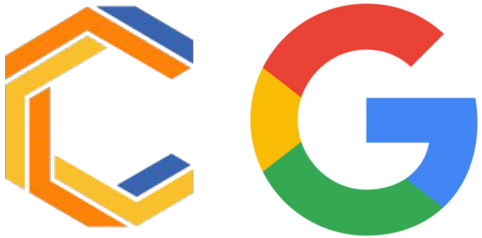
Cirq Quantum Framework

Masters of Advanced Cybersecurity,
Cezar Crăciunoiu & Dragoș Argint
24 may 2022

Google + 


First steps
Install
Getting started
Essential concepts
Qubits
Gates and operations
Circuits
Simulation
Ecosystem
Further topics
Noise
Devices
Transform circuits
Qudits
Protocols
Import/export circuits
Custom gates
Operators
Issues / requests / questions
Tutorials and examples
Beginner
Basics
Heatmaps
State Histograms
Intermediate
Quantum variational algorithm
Approximate optimization
Quantum walks
Fourier Checking
Advanced
Rabi oscillation experiment
Cross entropy benchmarking (XEB) theory
XEB and coherent error
Parallel XEB
Isolated XEB
Hidden linear function problem
Shor's algorithm
Google hardware
Getting started with QCS
Visualizing calibration metrics
Identifying Hardware Changes
Qubit picking with Loschmidt echoes
Circuit optimization, gate alignment, & spin echoes
Calibration
Calibration: Overview and API
Floquet calibration: Example and benchmark
XEB calibration: Example and benchmark
AQT hardware
Getting started with AQT hardware
Azure Quantum
Getting started with Honeywell on Azure Quantum
Getting started with IonQ on Azure Quantum
IonQ hardware
Getting started with IonQ hardware
Binary paintshop
Pasqal hardware
Getting started with Pasqal hardware
Rigetti hardware
Getting started with Rigetti hardware

- Qubits on a grid
- Measurements really slow
- Measurements only at the end
- Limited set of gates
- Limited at ~400ms long circuits
- Limited qubits for real world-compatible circuits (23q/54q/72q)



Overview

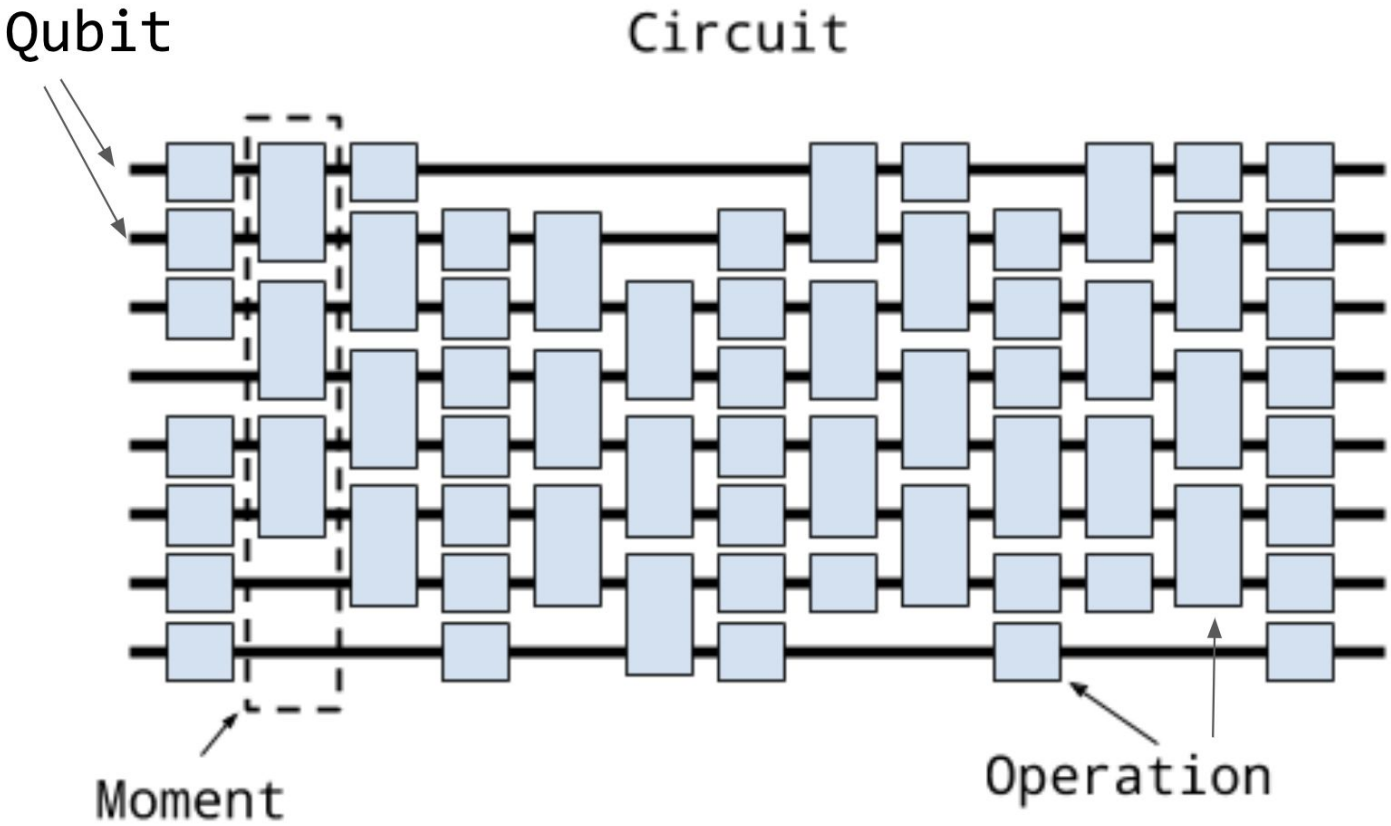
ID	Title	Path	Type
0	Cirq Setup	0-setup	Setup
1	Basic Quantum gates	1-gates	Basic
2	Superdense-Coding Implementation	2-superdense	Intermediate
3	Deutsch Implementation	3-deutsch	Intermediate
4	Quantum Linear Equation Solving	4-hhl	Advanced
5	Quantum MNIST Classification	5-mnist	Advanced
6	Cirq-Specific Tools	6-bonus	Bonus

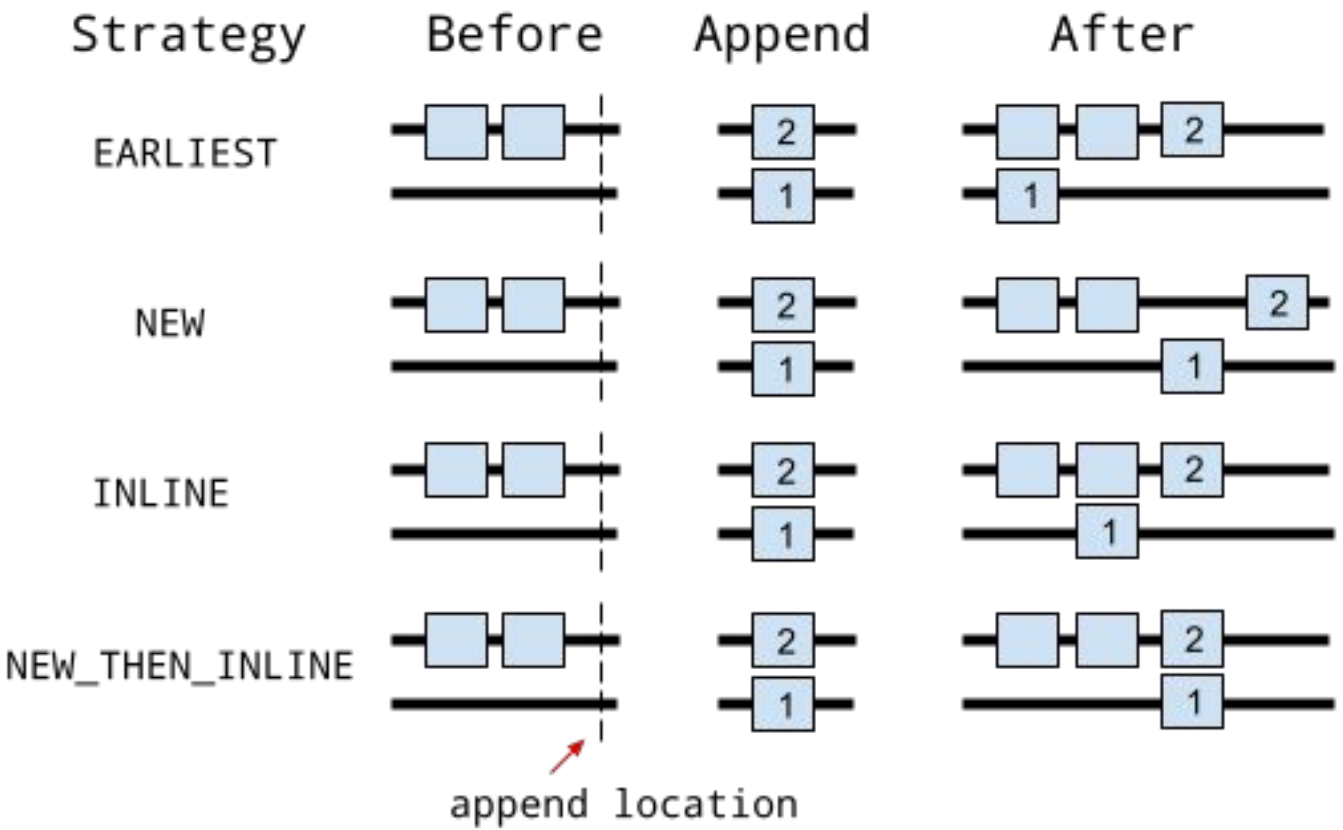
0. Cirq Setup

Everything starts with Github: https://github.com/craciunoiuc/IQC-lab-presentation

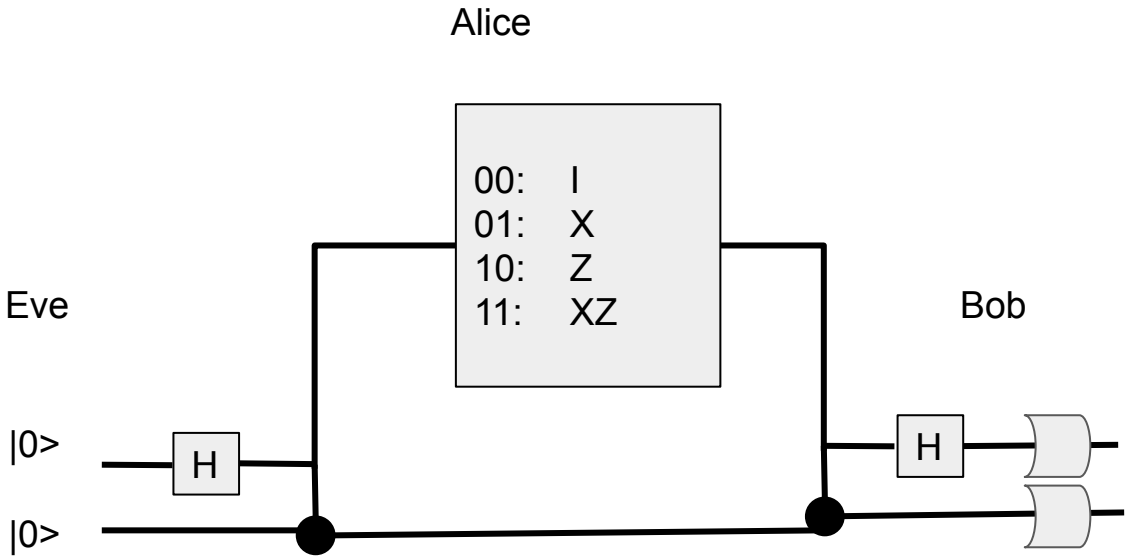
Clone the repository and let the fun begin!

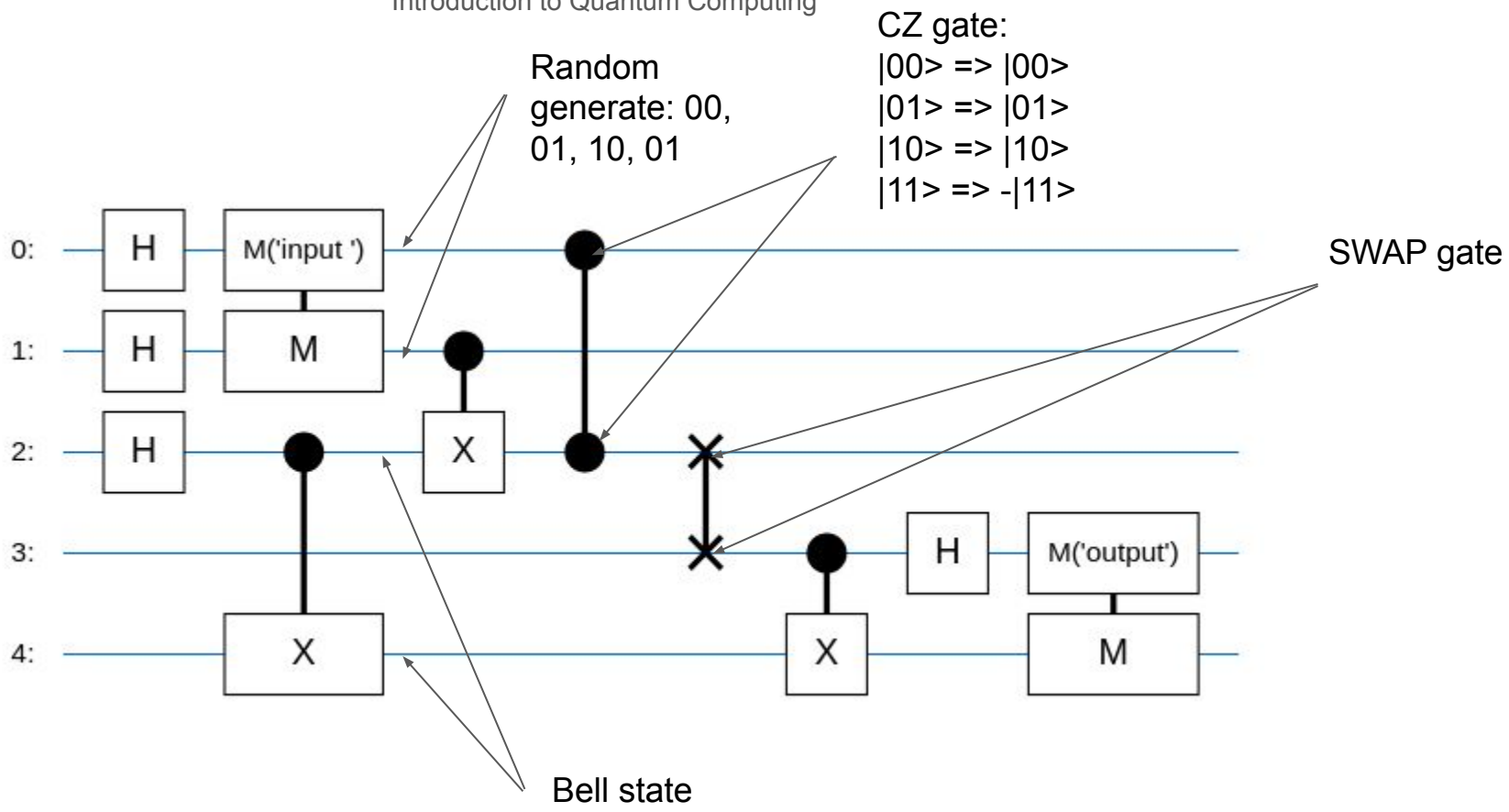
1. Basic Code In Cirq

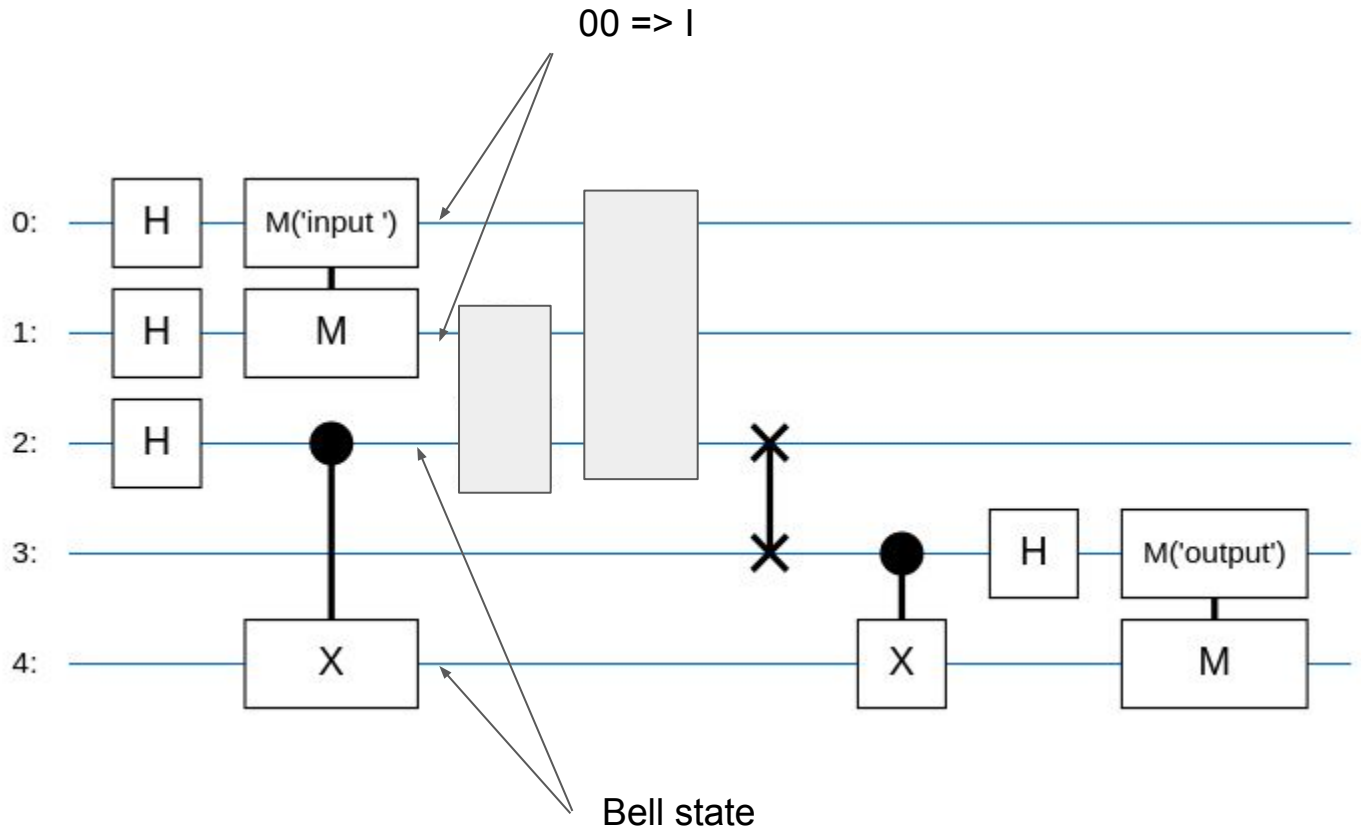

Qubit
Circuit
Moment
Operation

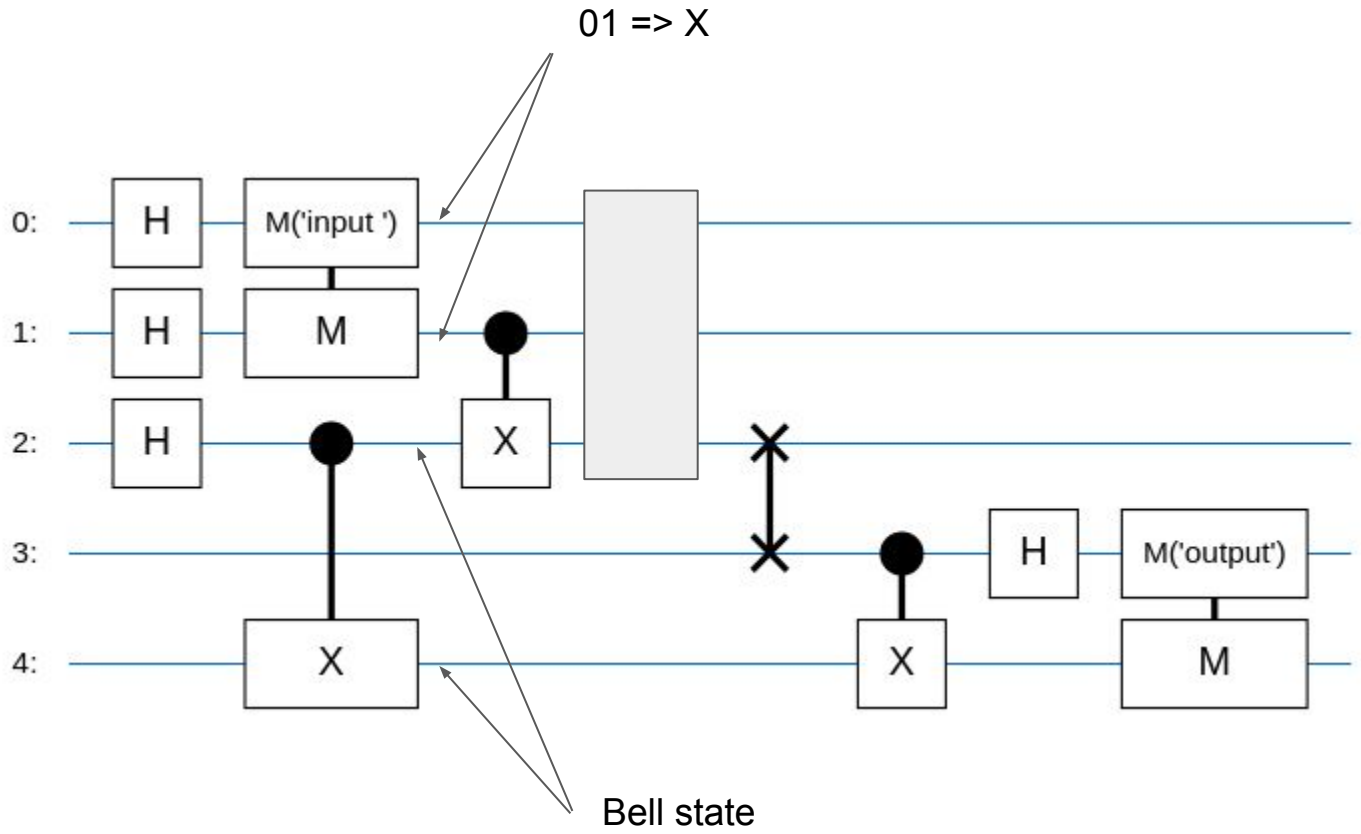

Strategy
Before
Append
After
EARLIEST
NEW
INLINE
NEW_THEN_INLINE
2
1
append location

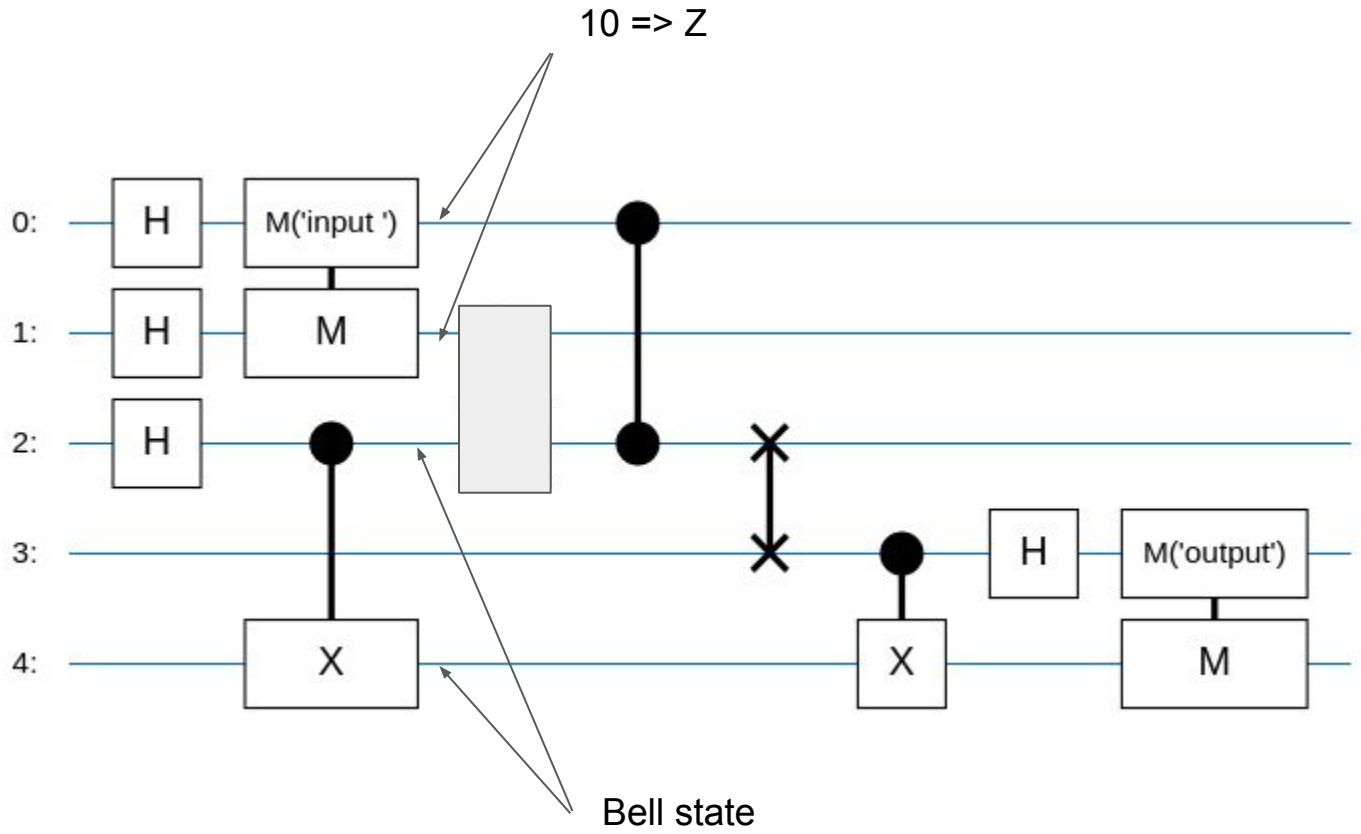
2. Superdense Coding

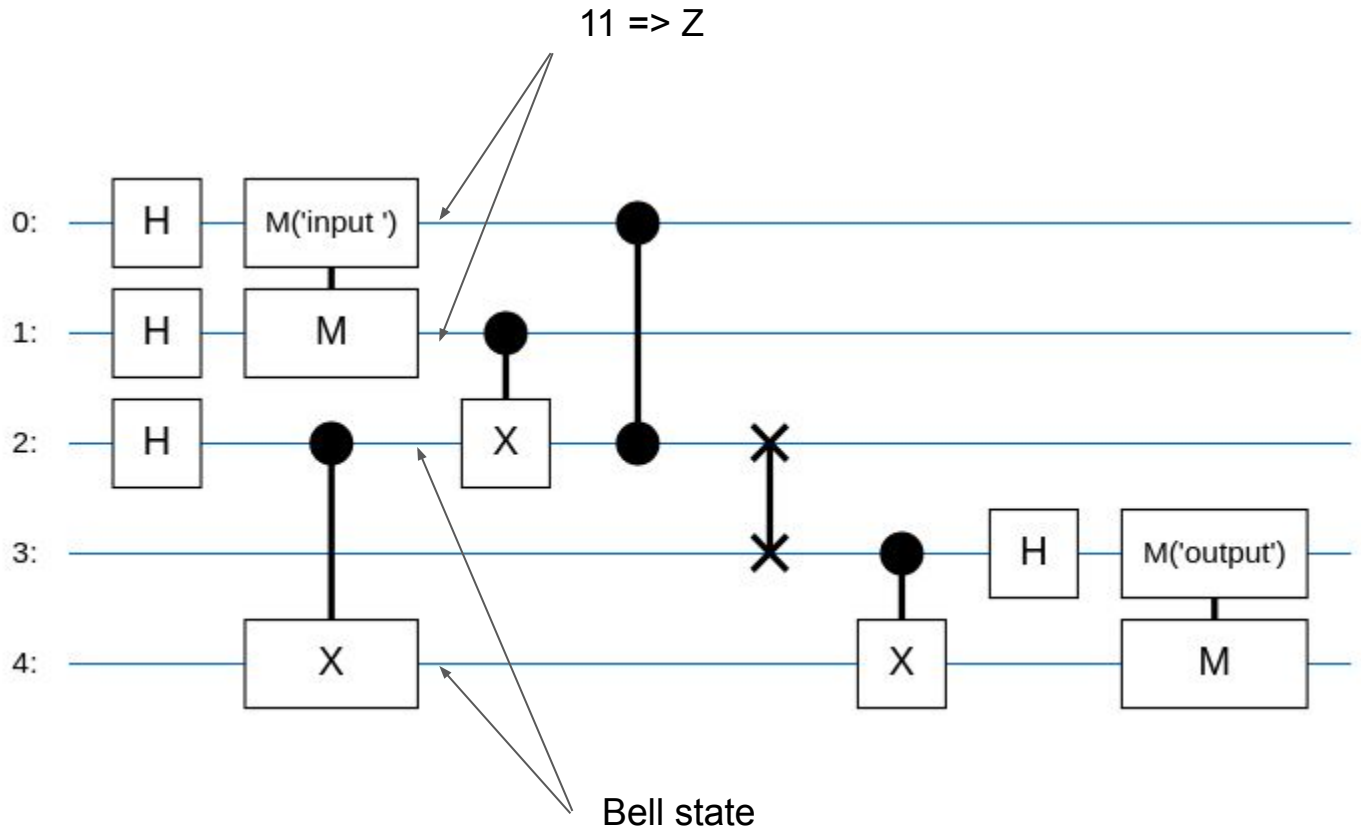

Alice
00: I
01: X
10: Z
11: XZ
Eve
Bob
|0>
|0>
H
H


Random generate: 00, 01, 10, 01
CZ gate:
|00> => |00>
|01> => |01>
|10> => |10>
|11> => -|11>
SWAP gate
0:
1:
2:
3:
4:
H
H
H
M('input ')
M
X
X
H
M('output')
X
M
Bell state

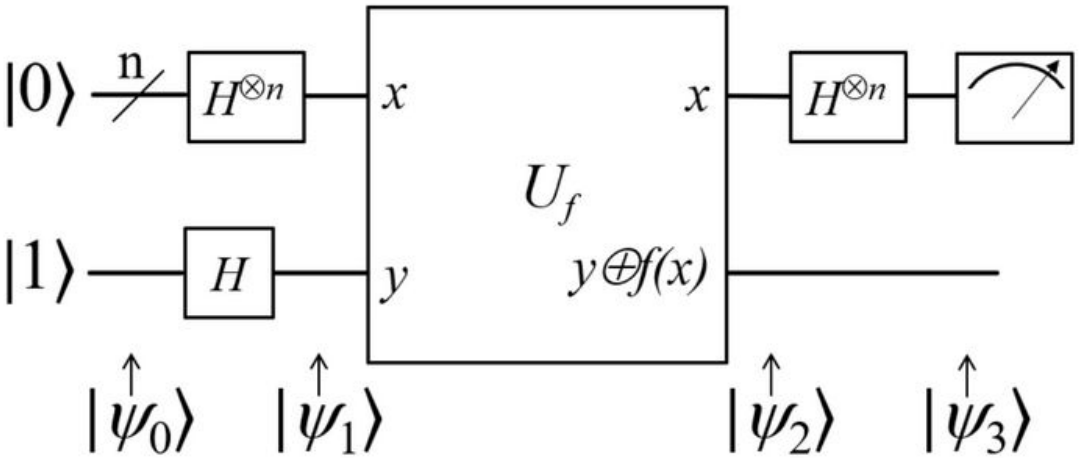

00 => I
0:
1:
2:
3:
4:
H
H
H
M('input ')
M
X
X
H
M('output')
M
Bell state


01 => X
0:
1:
2:
3:
4:
H
H
H
M('input ')
M
X
X
H
M('output')
X
M
Bell state


10 => Z
0:
1:
2:
3:
4:
H
H
H
M('input ')
M
X
X
H
M('output')
M
Bell state

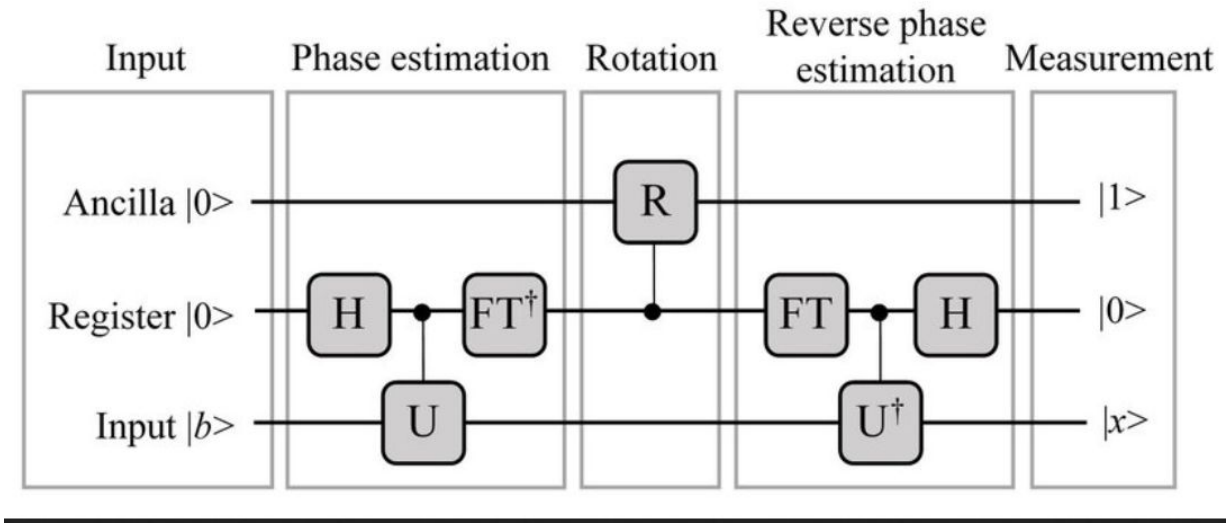

11 => Z
0:
1:
2:
3:
4:
H
H
H
M('input ')
M
X
X
H
M('output')
M
Bell state

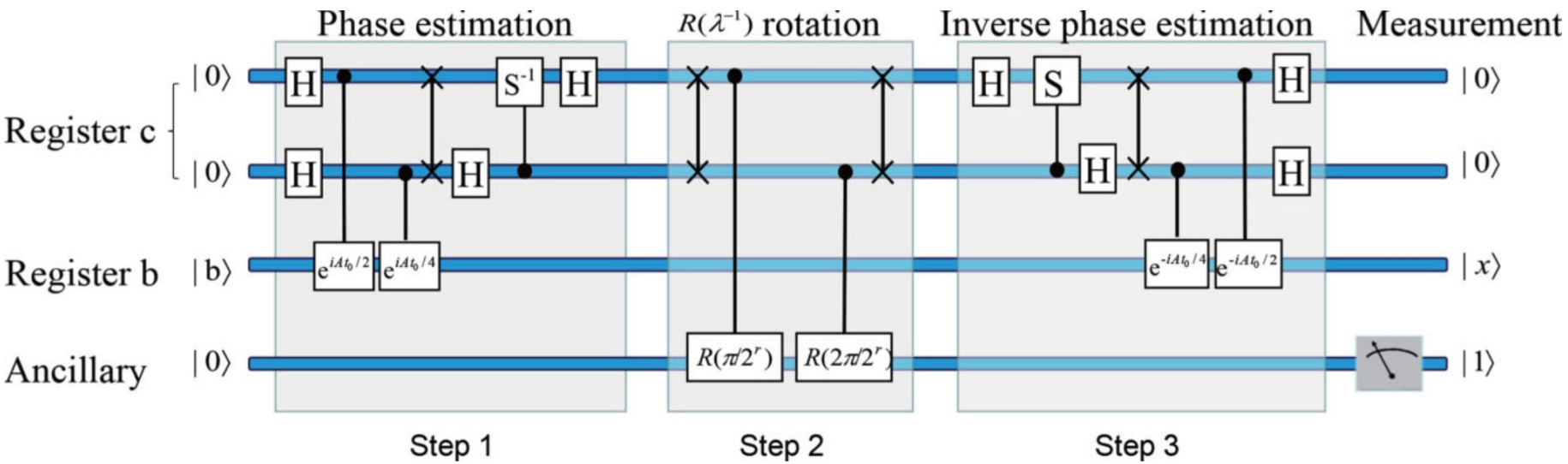
3. Deutsch Algorithm

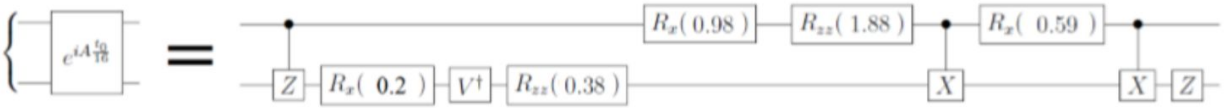

|0⟩
n
H⊗n
x
x
H⊗n
Uf
|1⟩
H
y
y⊕f(x)
|ψ0⟩
|ψ1⟩
|ψ2⟩
|ψ3⟩

To the drawing board!

4. HHL Algorithm


Input
Phase estimation
Rotation
Reverse phase estimation
Measurement
Ancilla |0>
Register |0>
Input |b>
H
FT†
U
R
FT
H
U†
|1>
|0>
|x>

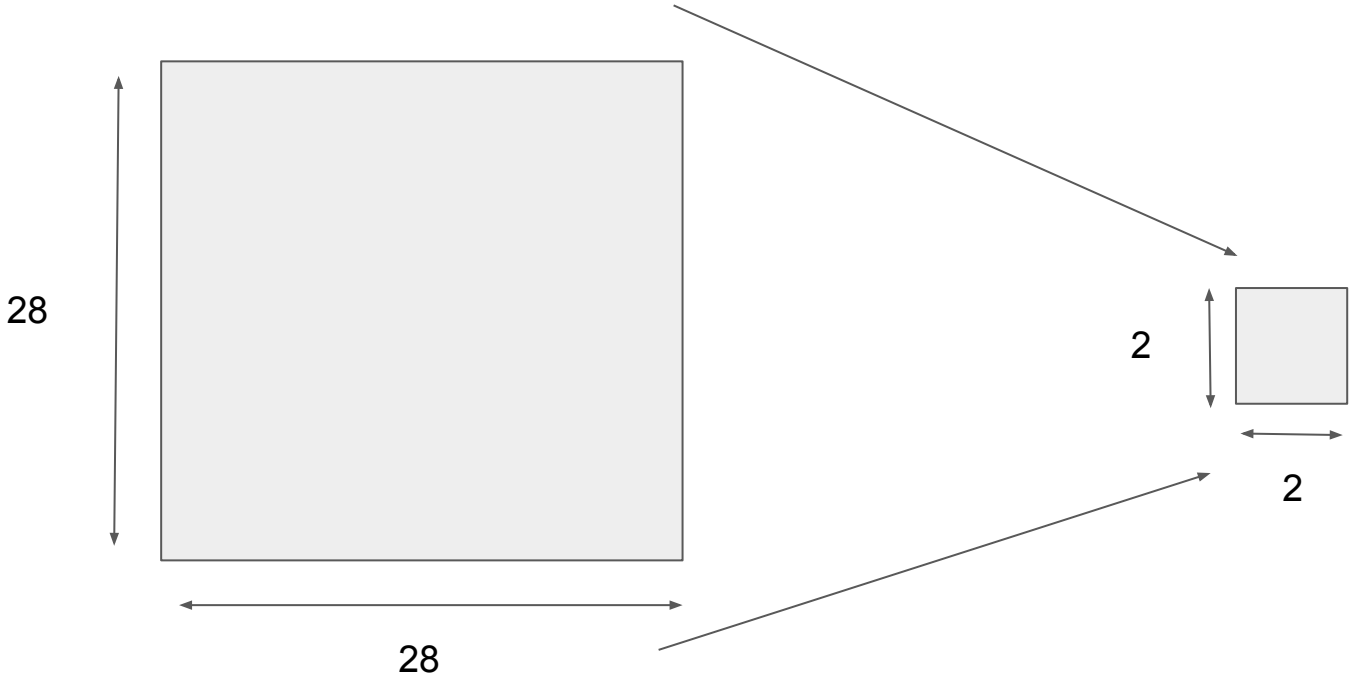

Phase estimation
$R(\lambda^{-1})$ rotation
Inverse phase estimation
Measurement
Register c
Register b
Ancillary
$|0\rangle$
$|0\rangle$
$|b\rangle$
$|0\rangle$
H
H
S^{-1}
H
$e^{iAt_0/2}$
$e^{iAt_0/4}$
$R(\pi/2^r)$
$R(2\pi/2^r)$
H
S
H
H
$e^{-iAt_0/4}$
$e^{-iAt_0/2}$
H
$|0\rangle$
$|0\rangle$
$|x\rangle$
$|1\rangle$
Step 1
Step 2
Step 3


$e^{iA\frac{t_0}{16}}$
=
$R_x(0.98)$
$R_{zz}(1.88)$
$R_x(0.59)$
Z
$R_x(0.2)$
$V^\dagger$
$R_{zz}(0.38)$
X
X
Z

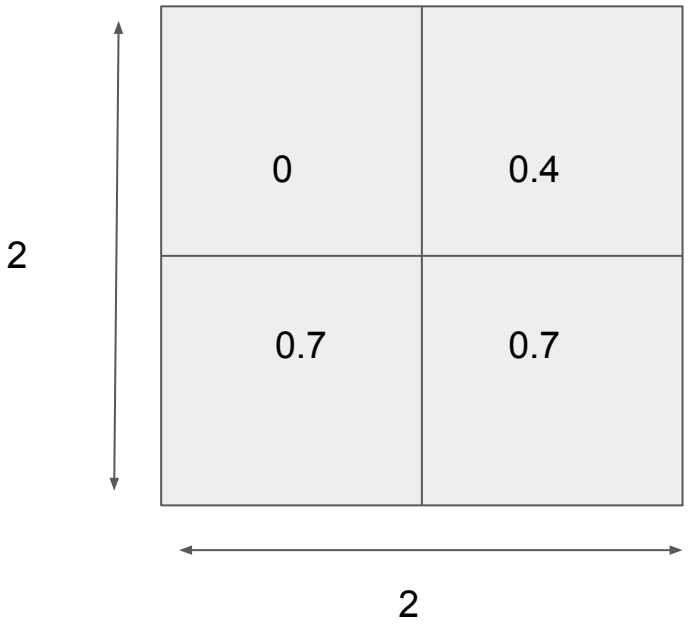
To the drawing board!

5. Quantum Neural Network

Fashion MNIST - dataset


28
28
2
2

if pixel > 0.5 => 1
else => 0



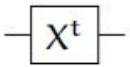

```
(1, 0): ———X———

(1, 1): ———X———
```

Parameterized gates

$$X^t = e^{-i\frac{\pi}{2}t(X-I)} = e^{i\frac{\pi}{2}t}R_x(\pi t)$$

$$= e^{i\frac{\pi}{2}t}\begin{bmatrix} \cos(\frac{\pi}{2}t) & -i\sin(\frac{\pi}{2}t) \\ -i\sin(\frac{\pi}{2}t) & \cos(\frac{\pi}{2}t) \end{bmatrix}$$



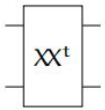

```
import sympy

q0 = cirq.LineQubit(0)
circuit = cirq.Circuit()
circuit.append(cirq.X(q0) ** sympy.Symbol('t'))
print(circuit)
```

```
0: ——X^t——
```

Parameterized gates

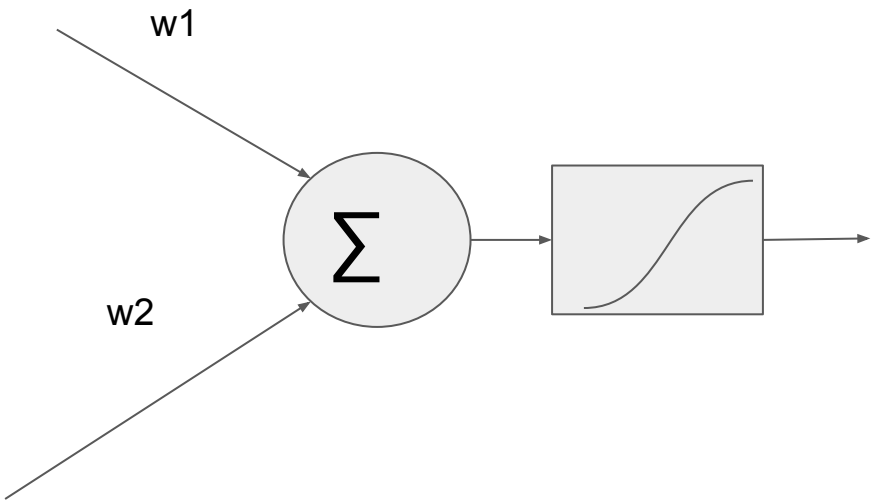
$$XX(t) = e^{-i\frac{\pi}{2}tX\otimes X}$$
$$= \begin{bmatrix} \cos(\frac{\pi}{2}t) & 0 & 0 & -i\sin(\frac{\pi}{2}t) \\ 0 & \cos(\frac{\pi}{2}t) & -i\sin(\frac{\pi}{2}t) & 0 \\ 0 & -i\sin(\frac{\pi}{2}t) & \cos(\frac{\pi}{2}t) & 0 \\ -i\sin(\frac{\pi}{2}t) & 0 & 0 & \cos(\frac{\pi}{2}t) \end{bmatrix}$$
$$= \mathrm{Can}(t, 0, 0)$$

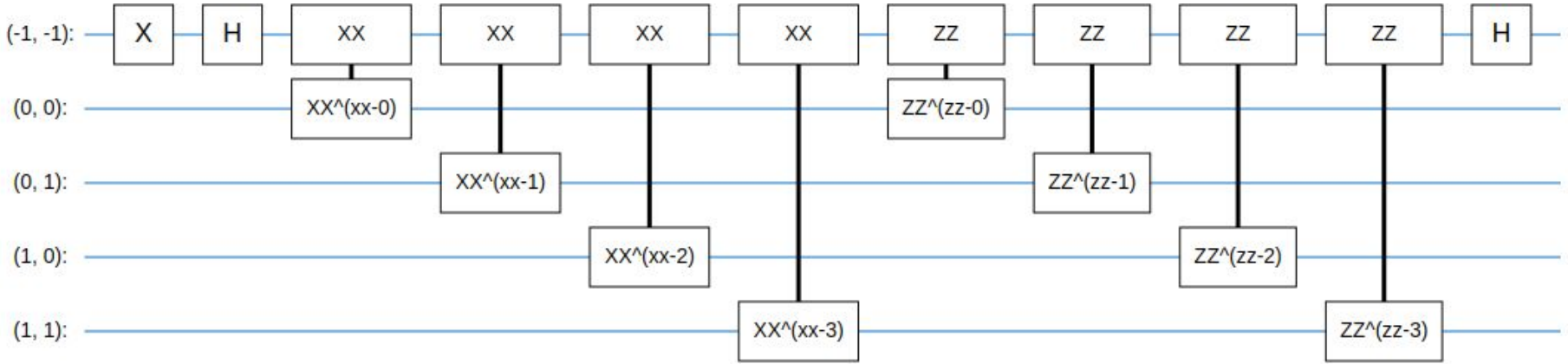



```
import sympy

q0, q1 = cirq.LineQubit.range(2)
circuit = cirq.Circuit()
circuit.append(cirq.XX(q0, q1) ** sympy.Symbol('t'))
print(circuit)
```

```
0: ──XX──────
     │
1: ──XX^t────
```

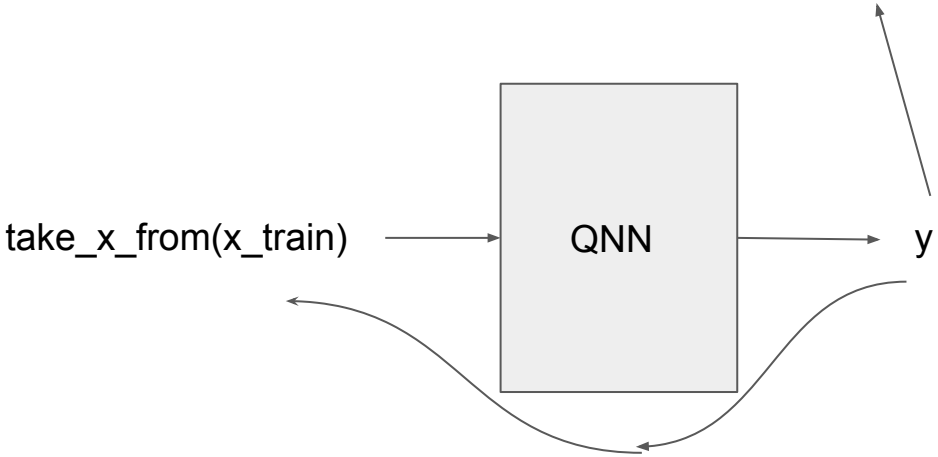

w1
w2
∑


(-1, -1):
(0, 0):
(0, 1):
(1, 0):
(1, 1):
X
H
XX
XX
XX
XX
ZZ
ZZ
ZZ
ZZ
H
XX^(xx-0)
XX^(xx-1)
XX^(xx-2)
XX^(xx-3)
ZZ^(zz-0)
ZZ^(zz-1)
ZZ^(zz-2)
ZZ^(zz-3)

Hinge loss

y_train:	1	0	0	1	0	…
y_train_hinge:	1	-1	-1	1	-1	…

$$l(y) = \max(0, 1-ty)$$



References

https://www.flaticon.com/free-icon/quantum-computing_1998708
https://quantumai.google
https://github.com/quantumlib/Cirq
http://cpb.iphy.ac.cn/article/2018/1924/cpb_27_2_020308/cpb_27_2_020308_f4.jpg
https://quantumai.google/cirq/tutorials/basics
https://quantumai.google/cirq/circuits
https://www.tensorflow.org/quantum/tutorials/mnist