QUANTUM PHYSICS ON A QUANTUM COMPUTER*

Richard Brower Boston University

MINI-LECTURE SERIES ON QUANTUM COMPUTING AND QUANTUM INFORMATION SCIENCE,

Jefferson Laboratory March 11, 2020

U.S. DEPARTMENT OF ENERGY

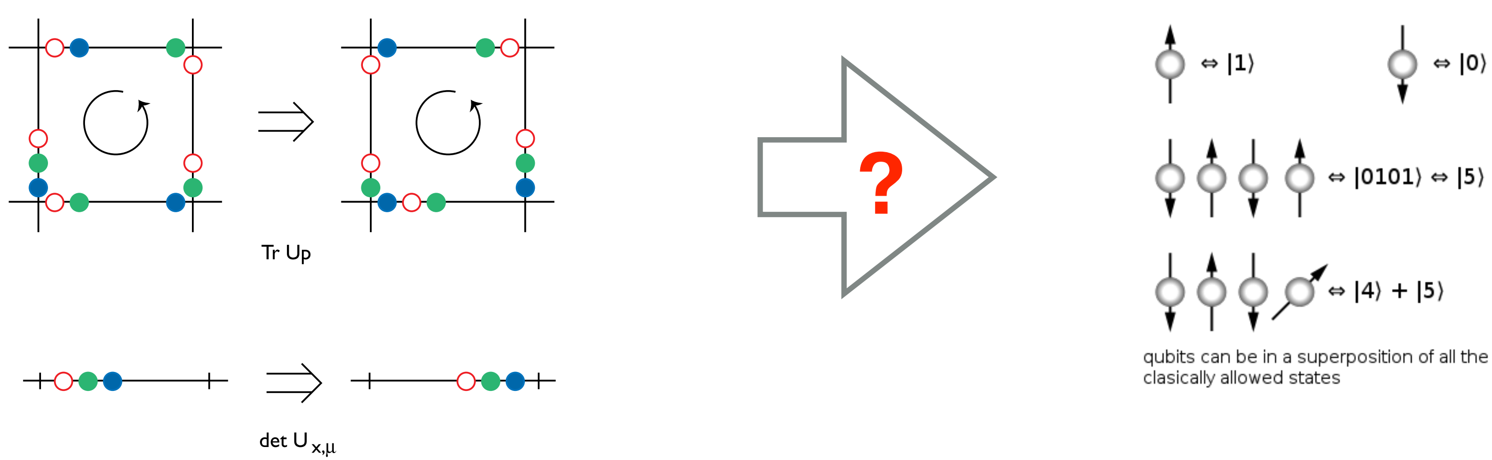
*D-theory: The accidental discovery of a Quantum Algorithm for Lattice Gauge theories: Circa 1998:

R. C. Brower, S. Chandrasekharan, U-J Wise , QCD as quantum link model, Phys. Rev D 60 (1999).

R. C. Brower, S. Chandrasekharan, U-J Wiese, D-theory: Field quantization … discrete variable Nucl. Phys. B (2004)

D-theory :QCD Abacus*

Fermionic Qubit Algorithm ?



Bravyi and Kitaev "Fermionic Quantum Computation" (2002)

R. C. Brower, The QCD Abacus: APCTP-ICPT Conference, Seoul, Korea, May (1997)

Richard Feynman

On quantum physics and computer simulation

"There is plenty of room to make [computers] smaller. . . . nothing that I can see in the physical laws . . . says the computer elements cannot be made enormously smaller than they are now. In fact, there may be certain advantages. —1959"

60 years ago!

"trying to find a computer simulation of physics seems to me to be an excellent program to follow out. . . . the real use of it would be with quantum mechanics. . . . Nature isn't classical . . . and if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy. —1981"

Feynman's Unfinished Legacy

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

"...The question is, If we wrote a Hamiltonian which *involved only* these [Pauli] operators, locally coupled to corresponding operators on the other space-time points, could we imitate every quantum mechanical *system* which is discrete and has a finite number of degrees of freedom? *I know, almost certainly,* that we could do that for any quantum mechan*ical system* which involves Bose particles. I'm no sure whether Fermi *particles* could be described by such a system. So, I leave that open..."

Richard P. Feynman

(Simulating Physics with Computers (1982))

Theoretical

International J. of Theoretical Physics Vol 21, now 6./7, 1982

Los Alam

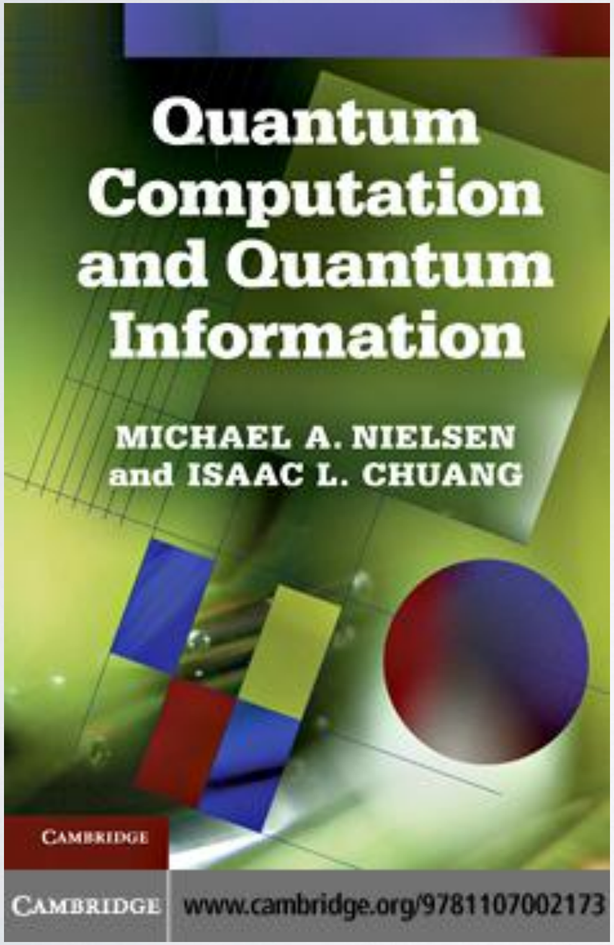
OUTLINE:

Quantum Computing for Quantum Field Theory

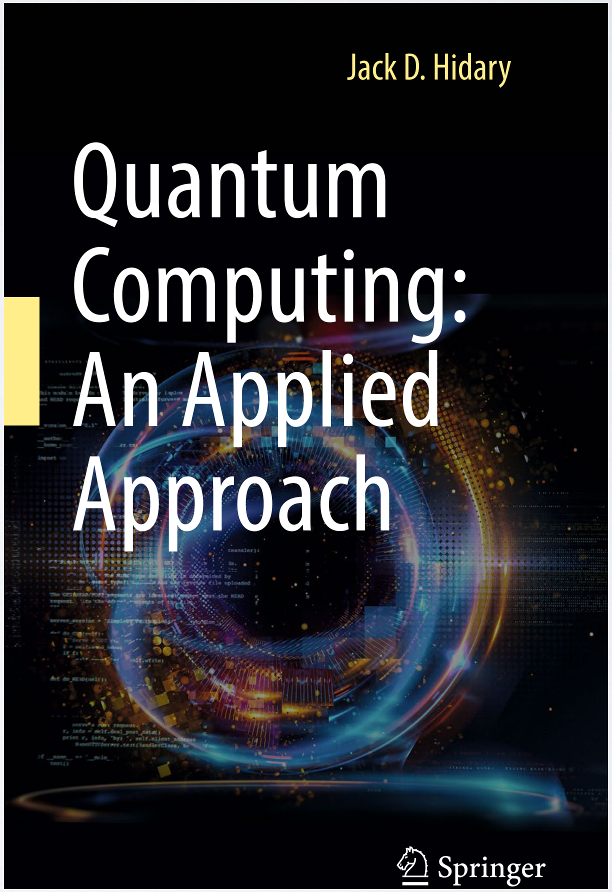
1. What is a Computer?
2. What is Quantum ? (Qubits and all that)
3. Universal vs Efficient/Fault Tol Quantum Computing ? (FFT)
4. What is Quantum Link Lattice Field Theory: Universality?
5. Example for NISQ (Noisy Intermediate Scale Quantum) ?
6. Programming (virtually) the IBM Q ?

GOOD QC REFERENCES

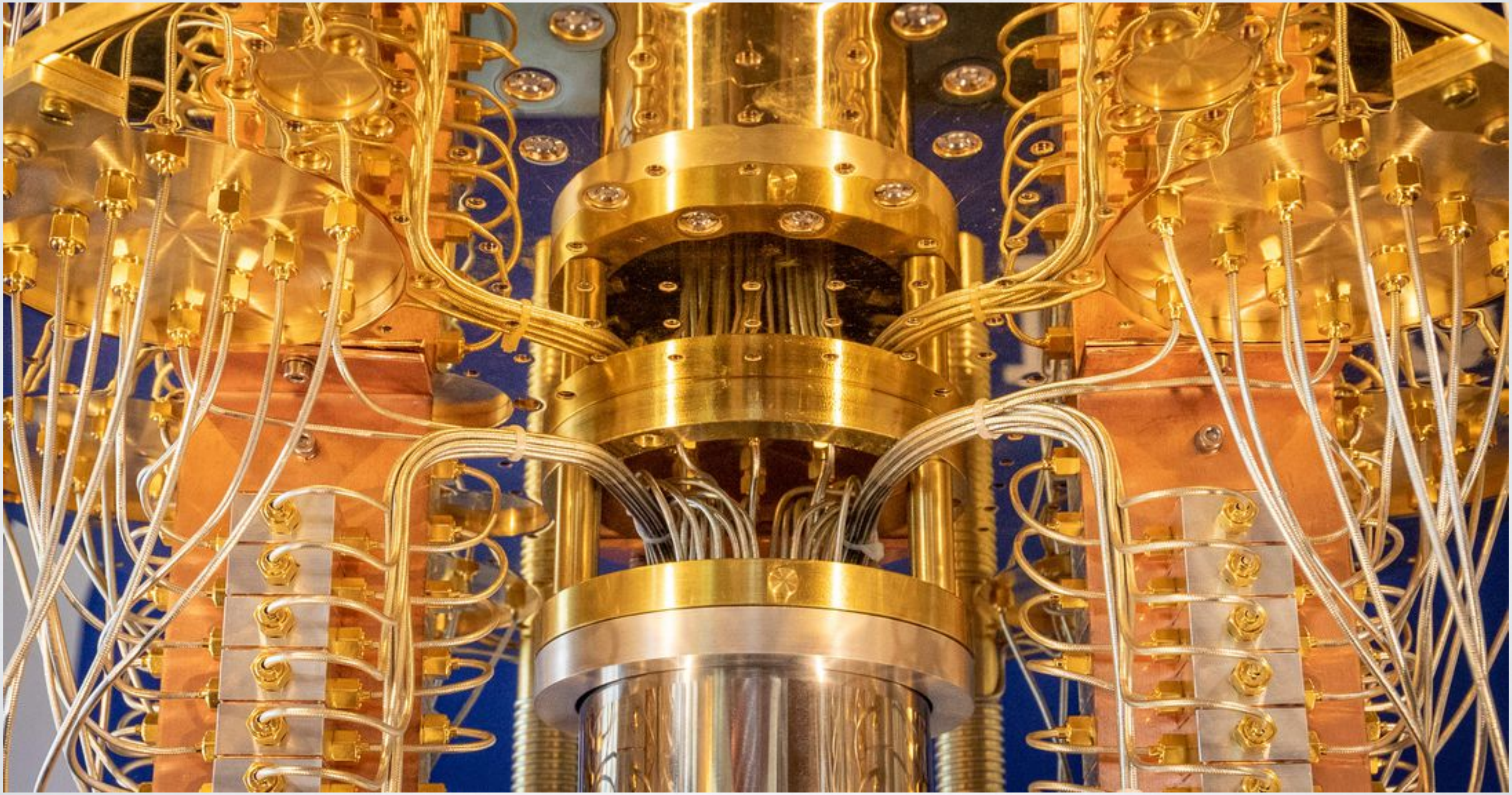
Help you Read this Book!



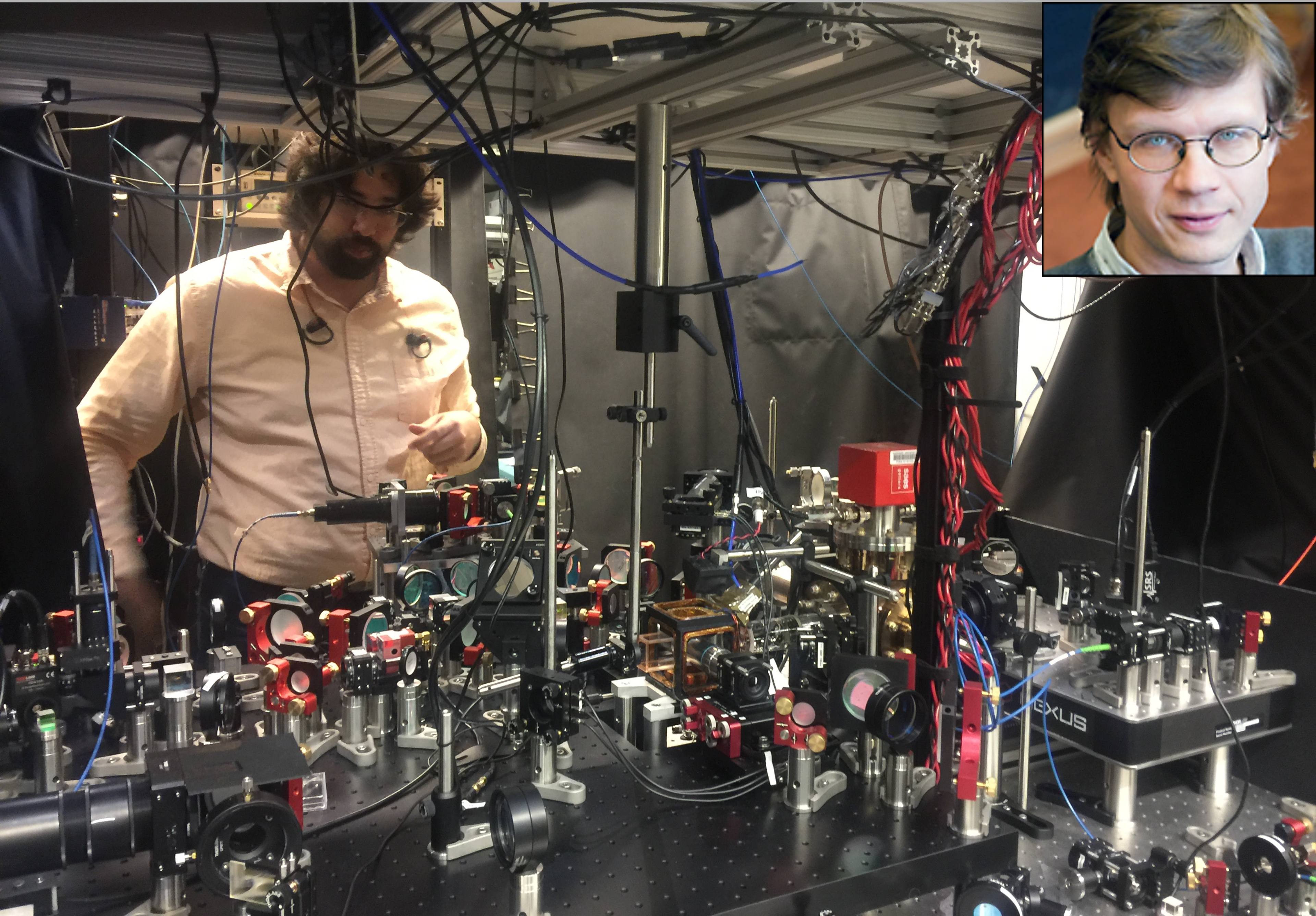
Simple Pedagogical Overview



IBM's new 53-qubit quantum computer is its biggest yet



Mikhail Lukin's Lab Harvard Trapped Ion

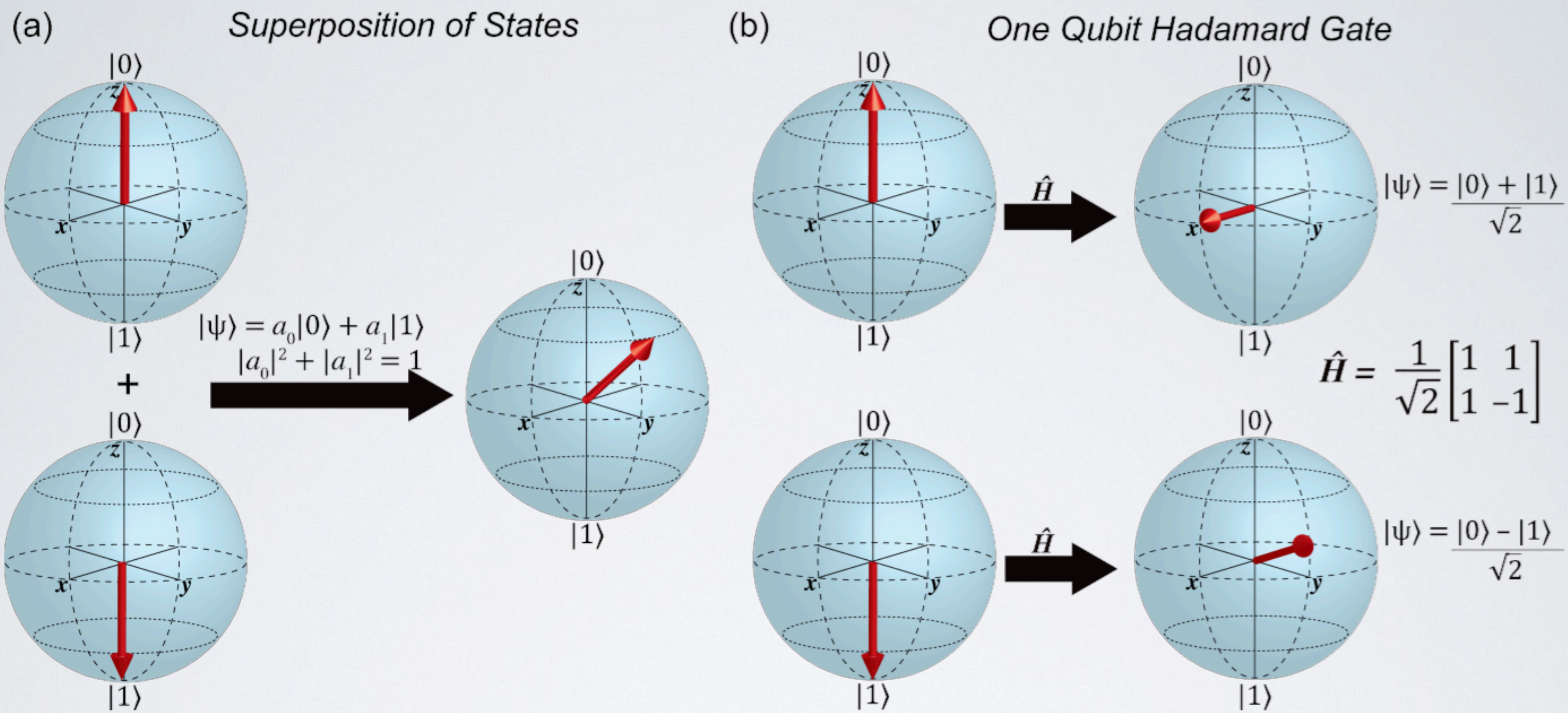


TODAY

Oak Ridge National Laboratory's 200 petaflop supercomputer

“Lattice Gauge Theory Machine” 200,000,000,000,000,000 Floats/sec
9,216 IBM POWER9 CPUs and 27648 NVIDIA GPUs
Each GPU has 5120 Cores and total of 580,608,000,000,000 transistors

WHAT IS A QUBIT



$$|\psi_{block}\rangle = e^{i\gamma}\big(\cos\frac{\theta}{2}|0\rangle + ie^{i\phi}\sin\frac{\theta}{2}|1\rangle\big)$$

$$|\psi\rangle = e^{i(\theta/2)\hat{n}\cdot\vec{\sigma}}|0\rangle = (\cos\theta/2 + i\hat{n}\cdot\vec{\sigma}\sin\theta/2)|0\rangle$$

Math Stuff: U(2) = U(1) x SU(2)^*

$$U = e^{i\phi} e^{i(\theta/2)\hat{n}\cdot\vec{\sigma}} = e^{i\phi}[\cos(\theta/2) + i\,\hat{n}\cdot\vec{\sigma}\;\sin(\theta/2)]$$

$$U = e^{i\phi}\begin{bmatrix} \cos\theta/2 + in_z\sin\theta/2 & i(n_x - in_y)\sin\theta/2 \\ i(n_x + in_y)\sin\theta/2 & \cos\theta/2 - in_z\sin\theta/2 \end{bmatrix}$$

THEREFORE:

$$U|0\rangle = e^{i\phi}[\cos(\theta/2) + in_z\sin(\theta/2)]|0\rangle + (in_x + n_y)\sin(\theta/2)|1\rangle$$

$$\alpha\alpha^* + \beta\beta^* = 1 \implies \cos^2(\theta) + \hat{n}\cdot\hat{n}\sin^2(\theta) = 1$$

---

* OR $U = e^{i\phi}[k_0\sigma_0 + i\,\vec{k}\cdot\vec{\sigma}] \implies k_\mu k_\mu = 1 \implies \mathbb{S}^1 \otimes \mathbb{S}^3$

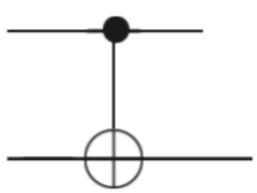
UNIVERSAL GATE SET

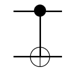
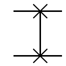
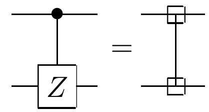
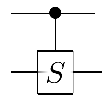
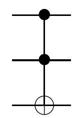
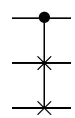
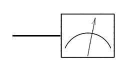
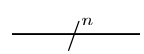
	Name	Symbol	Matrix	
*	Hadamard	$-\boxed{H}-$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	
	Pauli-X	$-\boxed{X}-$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	NOT
	Pauli-Y	$-\boxed{Y}-$	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	
	Pauli-Z	$-\boxed{Z}-$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	
	Phase	$-\boxed{S}-$	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	
*	$\pi/8$	$-\boxed{T}-$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	

Figure 4.2. Names, symbols, and unitary matrices for the common single qubit gates.

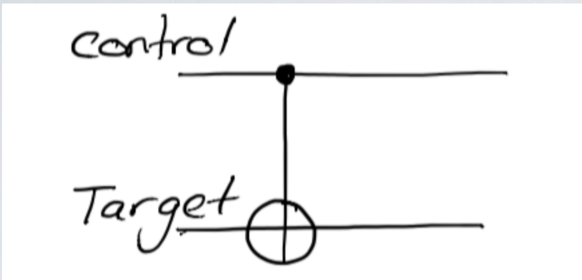
CNOT

* $$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$



controlled-NOT		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
swap		$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
controlled-Z		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
controlled-phase		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&i\end{bmatrix}$
Toffoli		$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$
Fredkin (controlled-swap)		$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\end{bmatrix}$
measurement		Projection onto $\|0\rangle$ and $\|1\rangle$
qubit		wire carrying a single qubit (time goes left to right)
classical bit		wire carrying a single classical bit
n qubits		wire carrying n qubits

WHAT IS CNOT ? UNITARY (GENERALIZED) XOR



c = 1 (true) activates Not t *

$$|c\rangle \otimes |t\rangle \equiv |ct\rangle$$

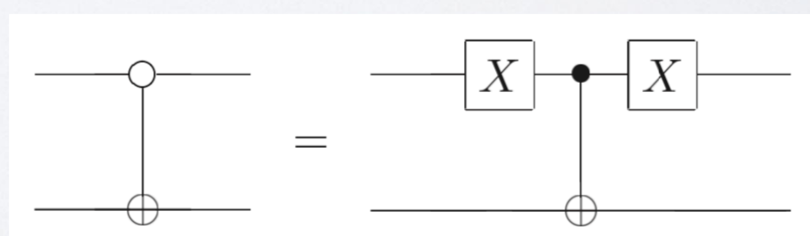
XOR / Mod 2 add

Cnot is $\quad |c\rangle|t\rangle \to |c\rangle|c \oplus t\rangle$

$$\begin{array}{cc} & \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\ CNOT = & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = Cnot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

WARNING
NC has a NotCnot with 0



UNIVERSAL QUANTUM COMPUTER

involving only those gates. We now describe three universality constructions for quantum computation. These constructions build upon each other, and culminate in a proof that any unitary operation can be approximated to arbitrary accuracy using Hadamard, phase, CNOT, and $\pi/8$ gates. You may wonder why the phase gate appears in this list, since it can be constructed from two $\pi/8$ gates; it is included because of its natural role in the fault-tolerant constructions described in Chapter 10.

The first construction shows that an arbitrary unitary operator may be expressed *exactly* as a product of unitary operators that each acts non-trivially only on a subspace spanned by two computational basis states. The second construction combines the first construction with the results of the previous section to show that an arbitrary unitary operator may be expressed *exactly* using single qubit and CNOT gates. The third construction combines the second construction with a proof that single qubit operation may be approximated to arbitrary accuracy using the Hadamard, phase, and $\pi/8$ gates. This in turn implies that any unitary operation can be approximated to arbitrary accuracy using Hadamard, phase, CNOT, and $\pi/8$ gates.

Our constructions say little about efficiency – how many (polynomially or exponentially many) gates must be composed in order to create a given unitary transform. In Section 4.5.4 we show that there *exist* unitary transforms which require exponentially many gates to approximate. Of course, the goal of quantum computation is to find interesting families of unitary transformations that *can* be performed efficiently.

Universal Quantum Computing

$$|\Psi(t)\rangle = U(t)|\Psi(t) = U(t)|\Psi(t) = e^{-itH}|\Psi(0)\rangle$$

I. Any dxd unitary U(d) is product of 2x2 U(2) unitary by GAUSSIAN ELIMINATION!

$$2^n(2^n - 1)/2 = 2^{n-1}(2^n - 1) \quad \text{Qubit Rotations}$$

II. General U(2) rotation: Gray coding with ***cnot's*** *and one Qubit* rotation

$$n^2 \quad \texttt{cnot} \text{ gates} + \text{one rotation}$$

III. But Solovay-Kitaev theorem approx U(2) on single Qubit can be approximated by Hadamard and pi/8 gates

$$\text{No of gates } \in O(\log^c(1/\epsilon) \quad c \simeq 2$$

Gaussian Elimination: Rotate one Qubit at a time!

Label: $|s_n\rangle \otimes \cdots \otimes |s_2\rangle \otimes |s_1\rangle$

$$i = 0, 1, \cdots 2^n \rightarrow 000, 001, 010, 011, 100, 101, 110, 111, \cdots 2^n - 1$$

$$UX = b \implies u_{ij}x_j = b_i$$

$$u_{11}x_1 + u_{12}x_2 + u_{13}x_3 + u_{14}x_4 + \cdots = b_1$$
$$u_{21}x_1 + u_{22}x_2 + u_{23}x_3 + u_{24}x_4 + \cdots = b_2$$
$$u_{31}x_1 + u_{32}x_2 + u_{33}x_3 + u_{34}x_4 + \cdots = b_3$$
$$\cdots$$

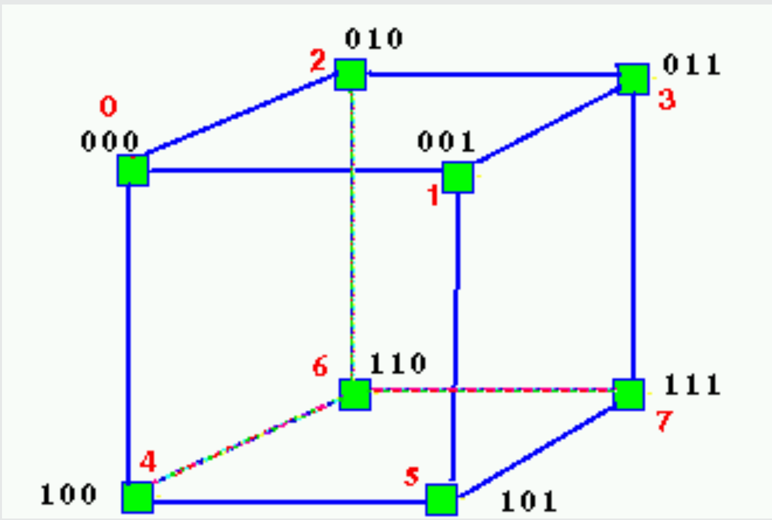
Now multiply $U_1UX = U_1b$ where

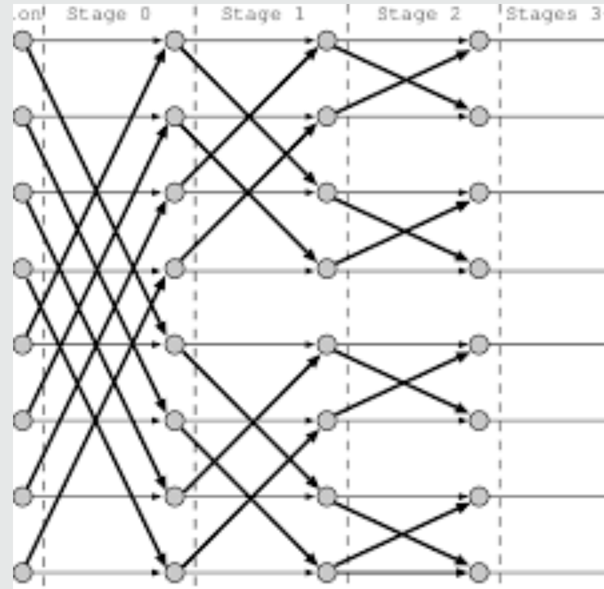
$$U_1 = \begin{bmatrix} \alpha & \beta & 0 & 0 & 0 & 0 & \cdots \\ -\beta^* & \alpha* & 0 & 0 & 0 & \cdots & \\ 0 & 0 & 1 & 0 & 0 & \cdots & \\ 0 & 0 & 0 & 1 & 0 & \cdots & \\ 0 & 0 & 0 & 0 & 1 & \cdots & \\ \cdots & & & & & & \end{bmatrix}$$

$U \rightarrow U_1U$ such that $u_{21}^{(new)} = -\beta^* u_{11} + \alpha^* u_{12} = 0$

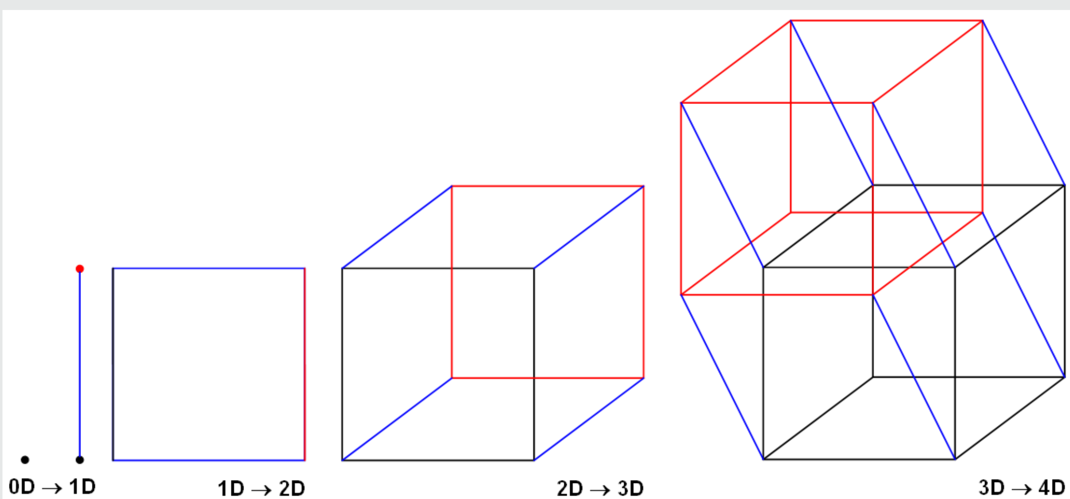
Hypercube - Gray Codes - FFT and all that

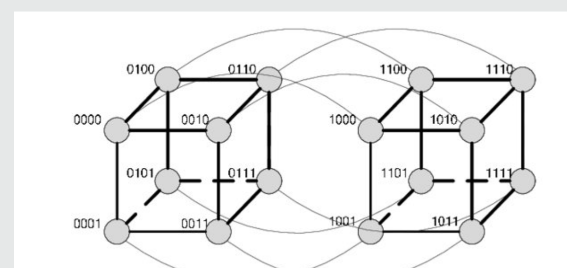
- Gray Coding: Adjacent Vertices Differ by one bit



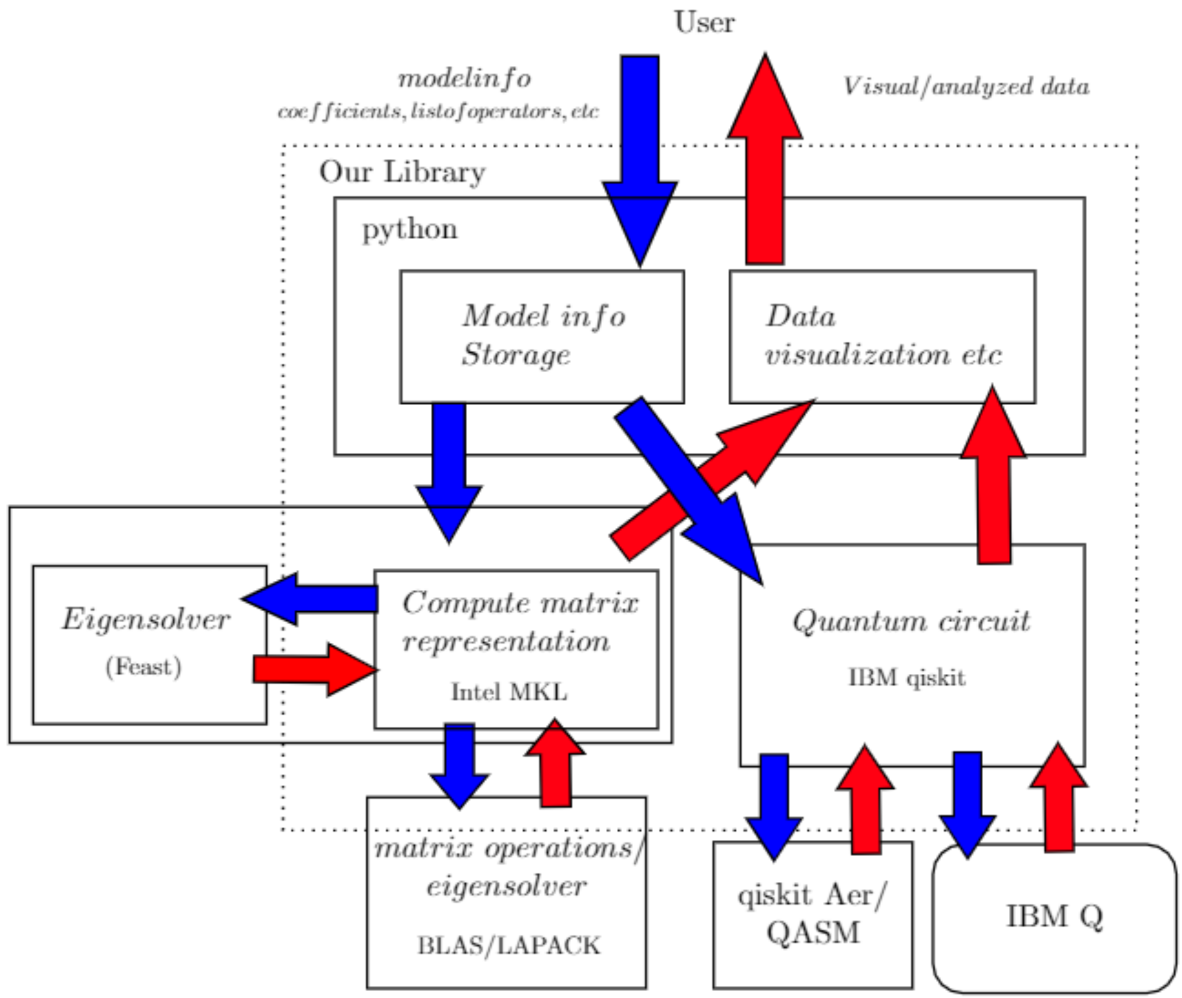


Texi Cab distance = Hamming Distance.





The architecture of the library:



IBM-Q Links for Excellent Python Tutorials, compiler and simulators

https://medium.com/quantum1net/richard-feynman-and-the-birth-of-quantum-computing-6fe4a0f5fcc7

FERMIONIC QUBIT GATES

Each Fermion $a^\dagger a + a a^\dagger = 1$ $aa = a^\dagger a^\dagger = 0$

On Each qubit $a^\dagger(\alpha|1\rangle + \beta|0\rangle) = \beta|1\rangle$ $a(\alpha|1\rangle + \beta|0\rangle) = \alpha|0\rangle$

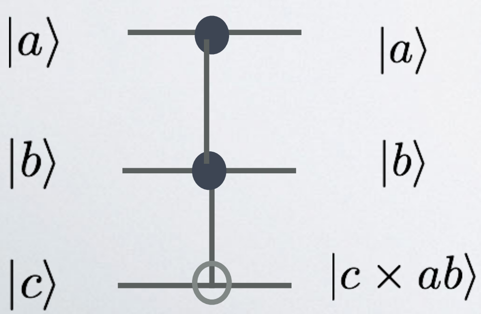
$$a^\dagger + a \implies X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$(a + a^\dagger)^2 - 1 \implies H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$(a^\dagger + a)/i \implies Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

$$(2a^\dagger a - 1) \implies Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

ContrNOT



$$(1 - a^\dagger a b^\dagger b) + a^\dagger a b^\dagger b(c^\dagger + c) \implies \text{Toffoli}$$

Classical Reversible Computing Conserving Energy R. Landauer IBM J journal of Research and Development, vol. 5, pp. 183-191, 1961