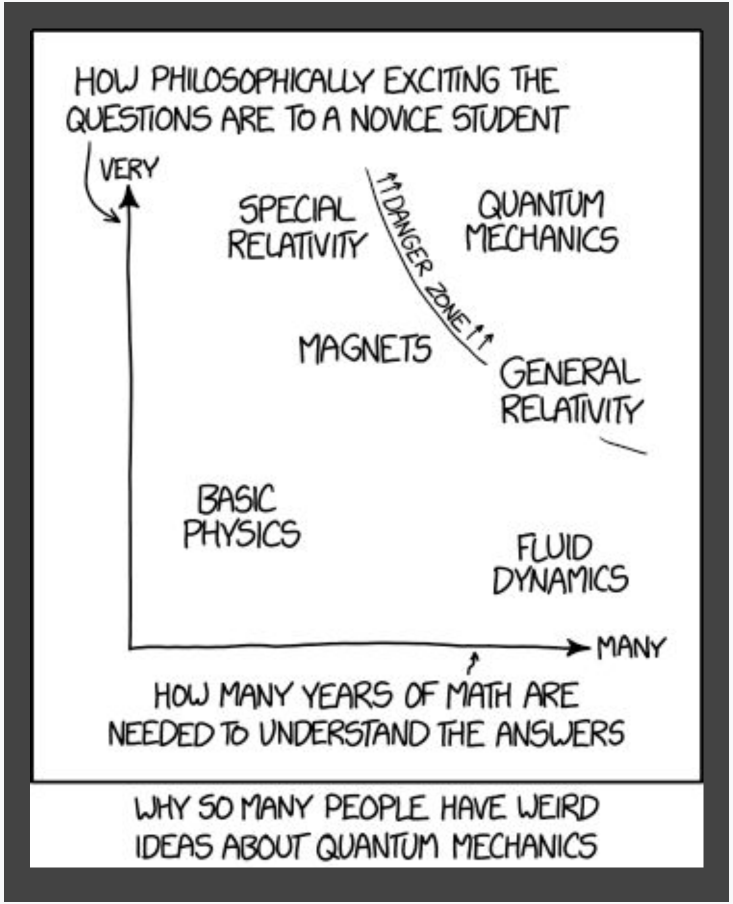
Verifying that the measurement of a quantum computation comes from a quantum computer

(not from a quantum simulator on a classical computer)

Richard Kent



Source: https://www.xkcd.com/1861/

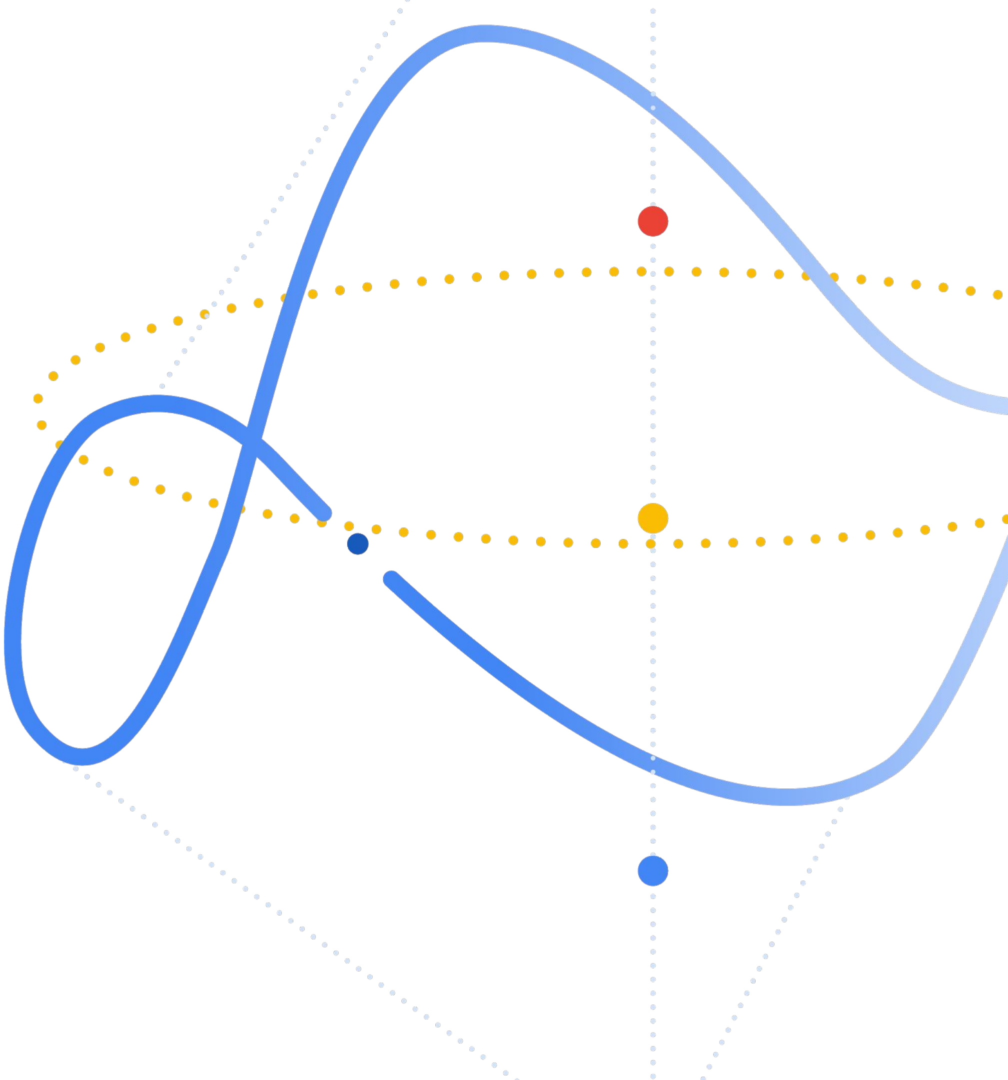
- Resources required to simulate quantum hardware increases exponentially with the number of qubits.
- Technologies currently available to the general public
  - Quantum computers to send jobs to
  - Quantum simulators

Simulating the probability distribution

Consider computations that have classically measurable results

- That is, a distribution of possible quantum states
- Can we verifiably simulate the probability distribution?
    - Must account for quantum noise
        - Knowing the resulting states of a particular computation, is it possible to simply emulate the probabilities of these outcomes?
        - If pseudorandomness is not enough, how to generate the entropy needed to simulate the distribution of states?

Quantum Programming

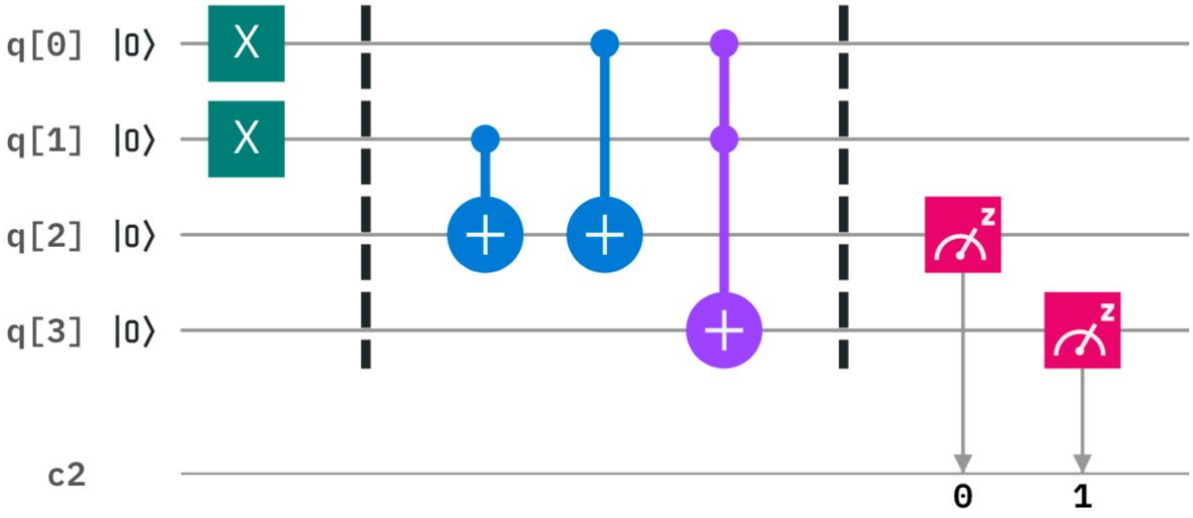


IBM Circuit Composer

- Build quantum circuits using quantum gates
- Run program on either
  - simulator
  - or one of IBM’s *real* quantum computers
- Output
  - Simulator
    - Ideal result
  - Quantum Computer
    - Imperfect results due to error / noise / decoherence

Getting started with quantum circuits

If you think quantum mechanics sounds challenging, you are not alone. All of our intuitions are based on day-to-day experiences, and so are better at understanding the behavior of balls and bananas than atoms or electrons. Though quantum objects can seem random and chaotic at first, they just follow a different set of rules. Once we know what those rules are, we can use them to create new and powerful technology. Quantum computing will be the most revolutionary example of this.



A quantum circuit to calculate 1+1

[4]

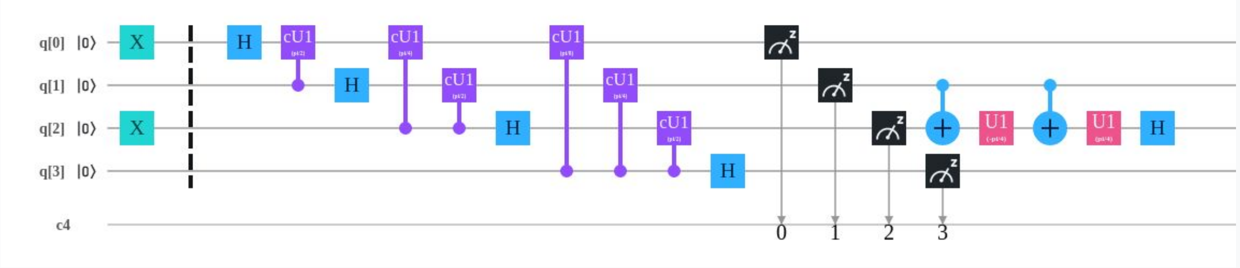
OpenQASM

Open Quantum Assembly Language

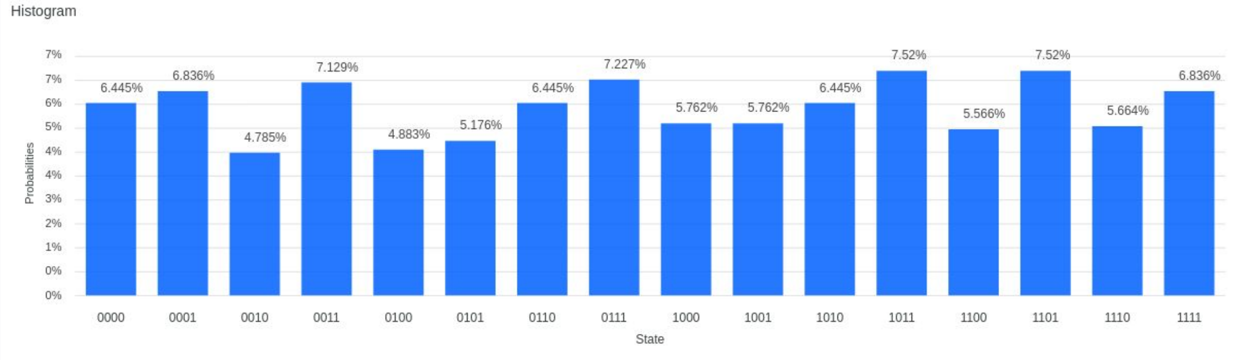
- Interface language for the Quantum Experience

Quantum Fourier Transform

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[4];
creg c[4];
x q[0];
x q[2];
barrier q;
h q[0];
cu1(pi/2) q[1],q[0];
h q[1];
cu1(pi/4) q[2],q[0];
cu1(pi/2) q[2],q[1];
h q[2];
cu1(pi/8) q[3],q[0];
cu1(pi/4) q[3],q[1];
cu1(pi/2) q[3],q[2];
h q[3];
measure q -> c;
cx q[1],q[2];
u1(-pi/4) q[2];
cx q[1],q[2];
u1(pi/4) q[2]; // end cu1
h q[2];
```



Gates in Circuit Composer



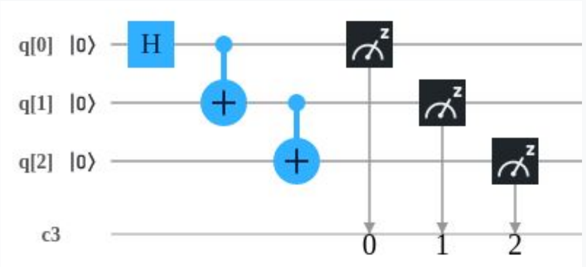
Probability Distribution

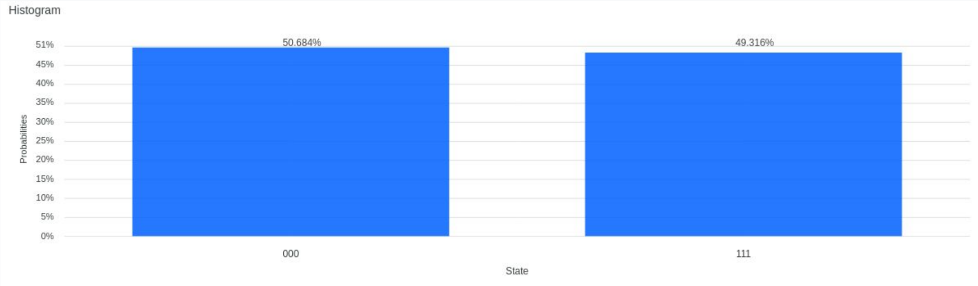
Greenberger–Horne–Zeilinger state

(GHZ)

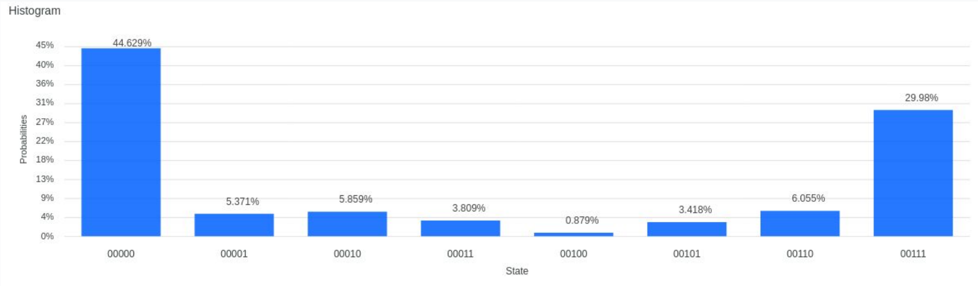
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

h q[0];
cx q[0],q[1];
cx q[1],q[2];

measure q -> c;
```

$$|\mathrm{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}$$





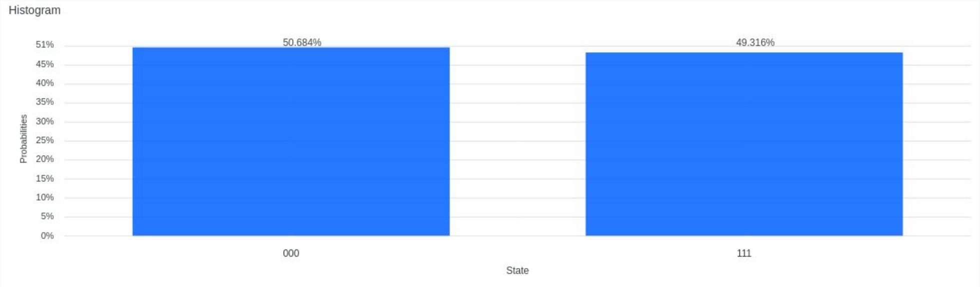
Simulator Results



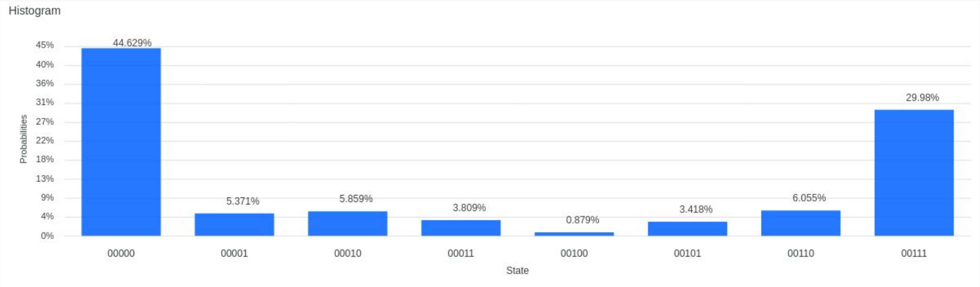
QC Results

Clearly, the distributions are vastly different

- Can the simulator generate random noise to mimic the QC results?



Simulator Results



QC Results

Now:

- We can simulate quantum computation on a classical machine
    - The resulting measurements are nearly perfect
    - How can we simulate error or noise?
        - Unfortunately, the circuit composer does not enable you to simulate noise when using the classical simulation.
- Fortunate for us, we can use IBM's qiskit
    - Qiskit has a programmatic way of introducing both error and noise...

Simulating Noise

Qiskit enables us to:

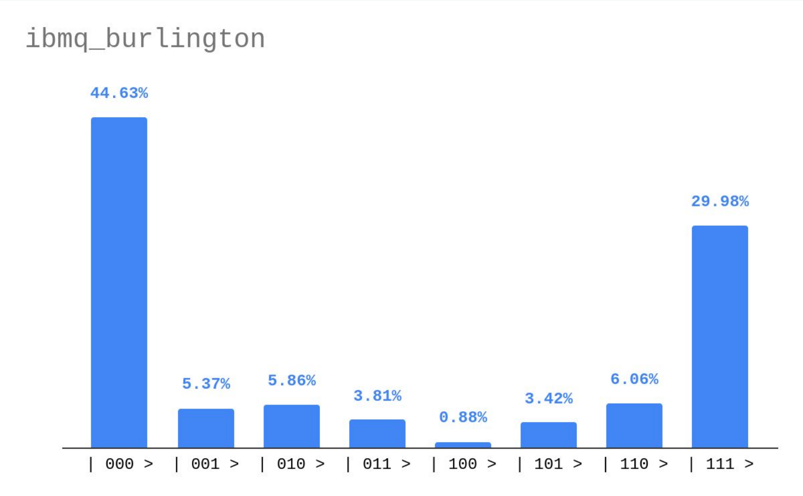
- Select gates to inject a so-called “noise model”
- Introduce error for specified gates
- Run the quantum circuit through a simulator on your local machine where the results exhibit error!

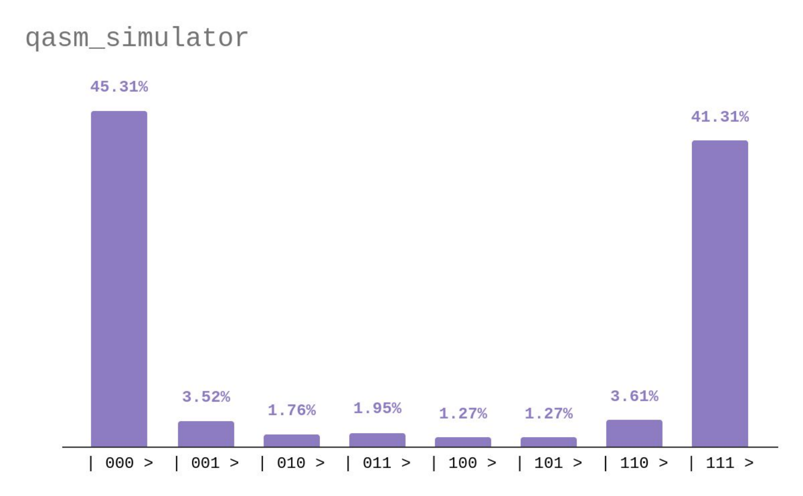
```
from qiskit import Aer, IBMQ, execute, QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.transpiler import CouplingMap
from qiskit.tools.monitor import job_monitor
from qiskit.providers.aer import noise
from qiskit.providers.aer.noise.errors import depolarizing_error

# Construct quantum circuit
qr = QuantumRegister(3, 'qr')
cr = ClassicalRegister(3, 'cr')
circ = QuantumCircuit(qr, cr)
circ.h(qr[0])
circ.cx(qr[0], qr[1])
circ.cx(qr[1], qr[2])
circ.measure(qr, cr)

# Select the QasmSimulator from the Aer provider
simulator = Aer.get_backend('qasm_simulator')

# Execute and get counts
result = execute(circ, simulator).result()
counts = result.get_counts(circ)
print(counts);

noise_model = noise.NoiseModel(['h', 'cx'])


error2 = depolarizing_error(0.1, 2)
noise_model.add_all_qubit_quantum_error(error2, ['cx'])


# Get the basis gates for the noise model
basis_gates = noise_model.basis_gates

# Select the QasmSimulator from the Aer provider
simulator = Aer.get_backend('qasm_simulator')
cm = CouplingMap([[0,1], [0,2], [1,2]])
# Execute noisy simulation and get counts
result_noise = execute(circ, simulator,
                       noise_model=noise_model,
                       coupling_map=cm,
                       basis_gates=basis_gates).result()
counts_noise = result_noise.get_counts(circ)
print(counts_noise)
```

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

h q[0];
cx q[0],q[1];
cx q[1],q[2];

measure q -> c;
```

Results

- After simulating for noise / error the (local) qasm_simulator now shows the same number of resulting states.
- Even so, the simulator still seems to follow a slightly more uniform distribution.





Of course there is a limit to the number of qubits we can simulate classically

- The qiskit qasm_simulator simulates up to 32 qubits
    - Perhaps because 2^{32} is the largest value representable by a 32 bit unsigned integer?

- How to overcome this limitation?

“Classical simulations of quantum circuits are limited in both space and time when the qubit count is above 50, the realm where quantum supremacy reigns” [1].

- Can we effectively simulate the error exhibited in quantum computation?
    - Further, how can we probabilistically generate the error exhibited without simulating the actual quantum circuit?
        - Hint: Harness randomness

Future Research

- Heuristics to simulate error of larger quantum systems
- Provided that there is a way to determine the probabilistic outcomes of a quantum computation - formalize a way to generate this probability distribution classically
- More practical uses of programming with qiskit

Citations

[1] Qiskit. (2018, December 19). Introducing Qiskit Aer: A high performance simulator framework for quantum circuits. Retrieved from https://medium.com/qiskit/qiskit-aer-d09d0fac7759

[2] Gil Kalai: "The Argument against Quantum Computers", 2019; arXiv:1908.02499.

[3] Zhao-Yun Chen, Qi Zhou, Cheng Xue, Xia Yang, Guang-Can Guo: "64-Qubit Quantum Circuit Simulation", 2018, Science Bulletin, 2018, 63(15):964-971; arXiv:1802.06952. DOI: 10.1016/j.scib.2018.06.007.

[4] IBM Q Experience. (n.d.). Retrieved from https://quantum-computing.ibm.com/support/guides/introduction-to-quantum-circuits?section=5cae613866c1694be21df8cc

[5] Andrew W. Cross, Lev S. Bishop, John A. Smolin: "Open Quantum Assembly Language", 2017; arXiv:1707.03429.

[6] CouplingMap — Qiskit 0.13.0 documentation. (n.d.). Retrieved from https://qiskit.org/documentation/api/qiskit.transpiler.CouplingMap.html

[7] qiskit.providers.aer.noise.noise_model — Qiskit 0.13.0 documentation. (n.d.). Retrieved from https://qiskit.org/documentation/_modules/qiskit/providers/aer/noise/noise_model.html#NoiseModel.add_all_qubit_quantum_error

[8] NoiseModel — Qiskit 0.13.0 documentation. (n.d.). Retrieved from https://qiskit.org/documentation/api/qiskit.providers.aer.noise.NoiseModel.html#qiskit.providers.aer.noise.NoiseModel.basis_gates

[9] Atkin, S. (2018, December 21). Demystifying Superdense Coding. Retrieved from https://medium.com/qiskit/demystifying-superdense-coding-41d46401910e

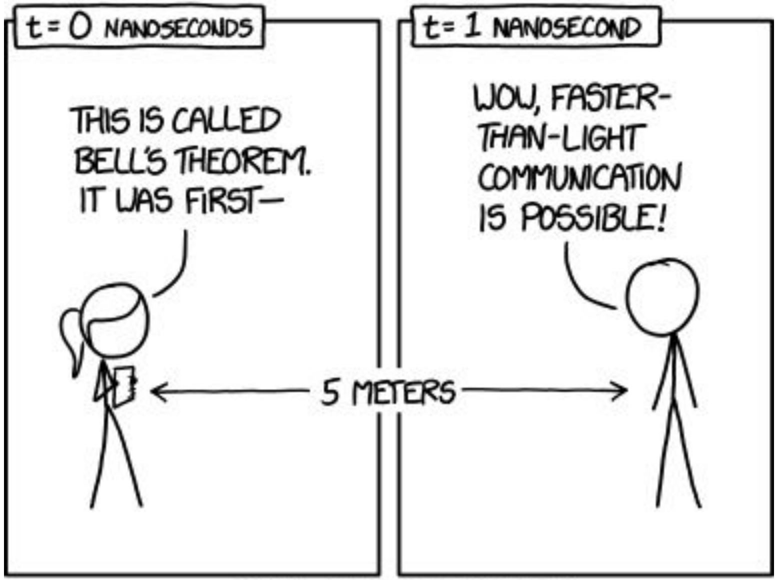
[10] Qiskit/qiskit-terra. (n.d.). Retrieved from https://github.com/Qiskit/qiskit-terra/blob/master/examples/python/qft.py

[11] Hakop Pashayan, Joel J. Wallman: "Estimating outcome probabilities of quantum circuits using quasiprobabilities", 2015, Phys. Rev. Lett. 115, 070501 (2015); arXiv:1503.07525. DOI: 10.1103/PhysRevLett.115.070501.

PARTICLE PROPERTIES IN PHYSICS

PROPERTY	TYPE/SCALE
ELECTRIC CHARGE	-1 0 +1
MASS	0 1kg 2kg
SPIN NUMBER	-1 -½ 0 ½ 1
FLAVOR	(MISC. QUANTUM NUMBERS)
COLOR CHARGE	R G B (QUARKS ONLY)
MOOD	
ALIGNMENT	GOOD–EVIL, LAWFUL–CHAOTIC
HIT POINTS	0
RATING	
STRING TYPE	BYTESTRING–CHARSTRING
BATTING AVERAGE	0% 100%
PROOF	0 200
HEAT	
STREET VALUE	$0 $100 $200
ENTROPY	(THIS ALREADY HAS LIKE 20 DIFFERENT CONFUSING MEANINGS, SO IT PROBABLY MEANS SOMETHING HERE, TOO.)

Source: https://www.xkcd.com/2217



BELL'S SECOND THEOREM:
MISUNDERSTANDINGS OF BELL'S THEOREM
HAPPEN SO FAST THAT THEY VIOLATE LOCALITY.