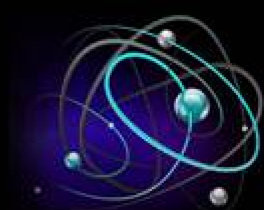
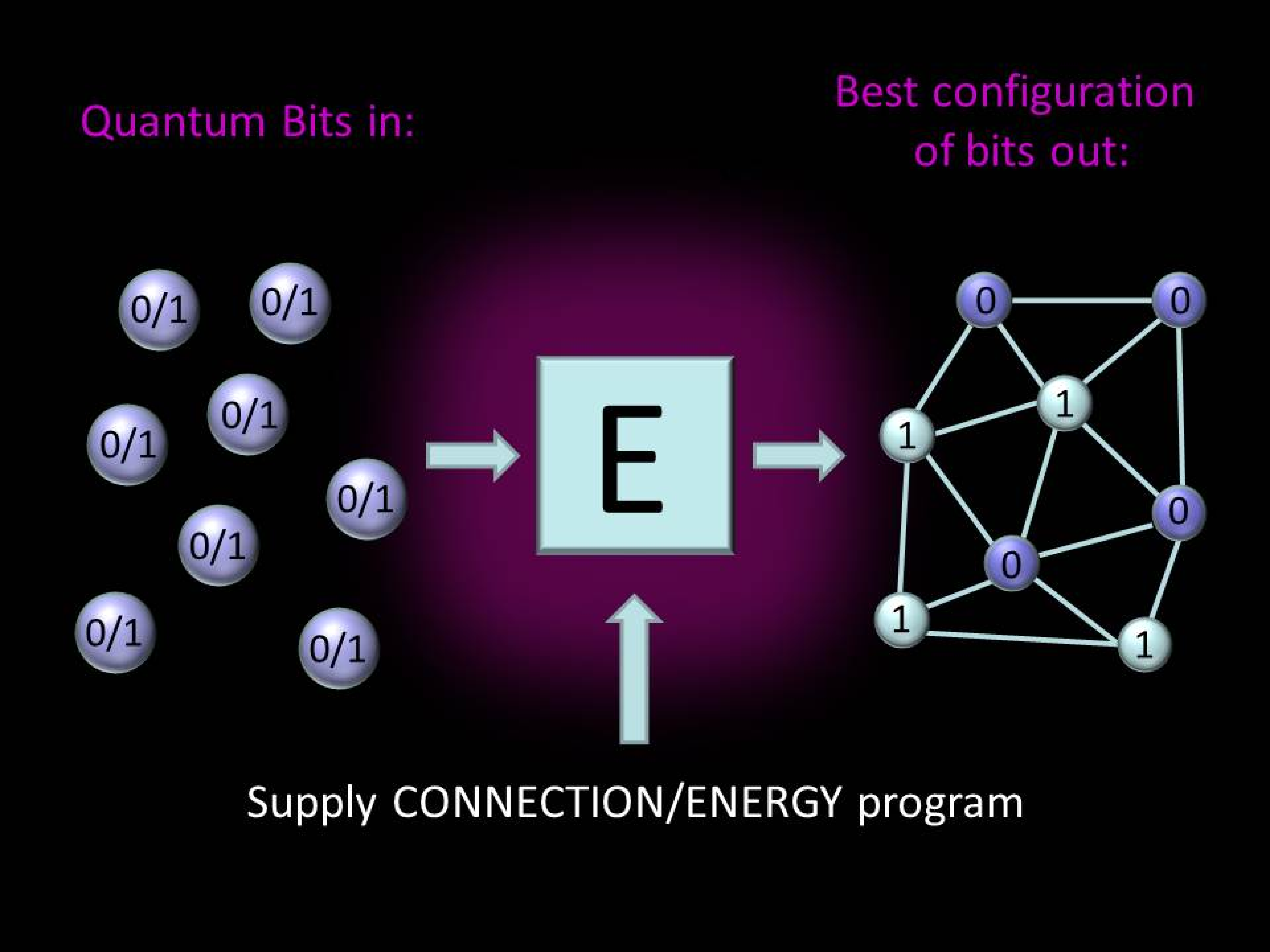
Hack the multiverse

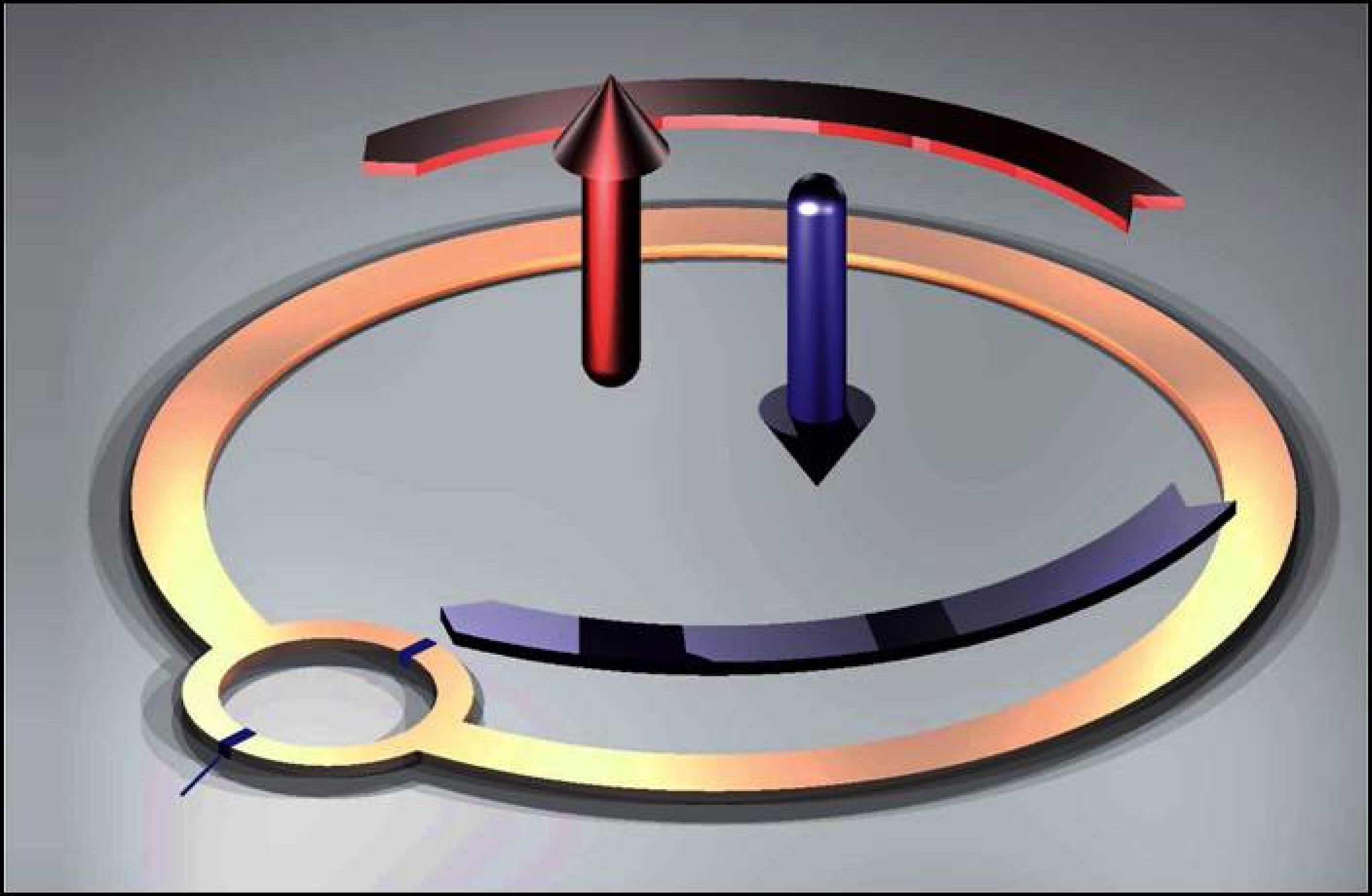


Dr. Suzanne Gildert



Quantum Bits in:
Best configuration of bits out:
0/1
0/1
0/1
0/1
0/1
0/1
0/1
0/1
E
0
0
1
1
0
0
1
1
Supply CONNECTION/ENERGY program

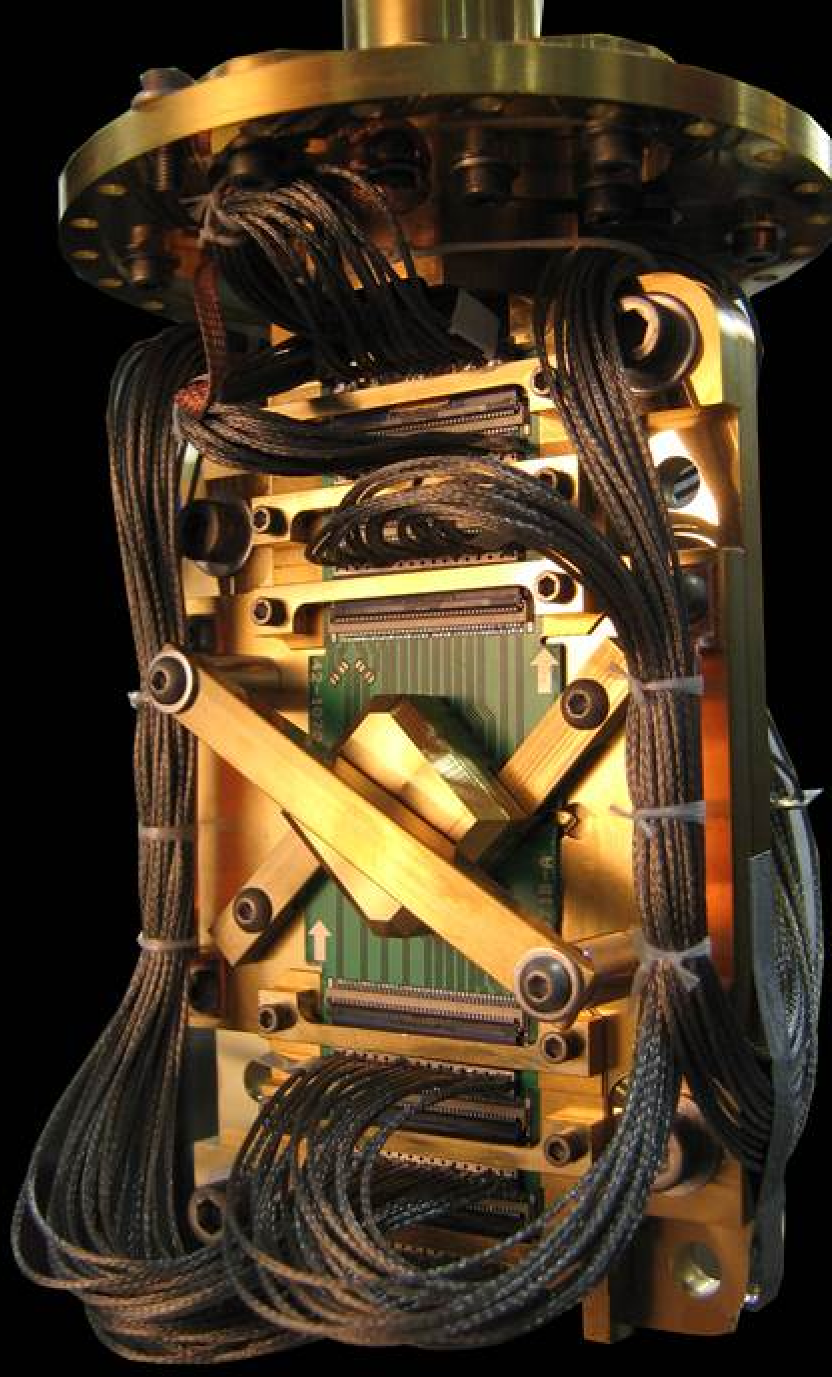
The building blocks of NQC



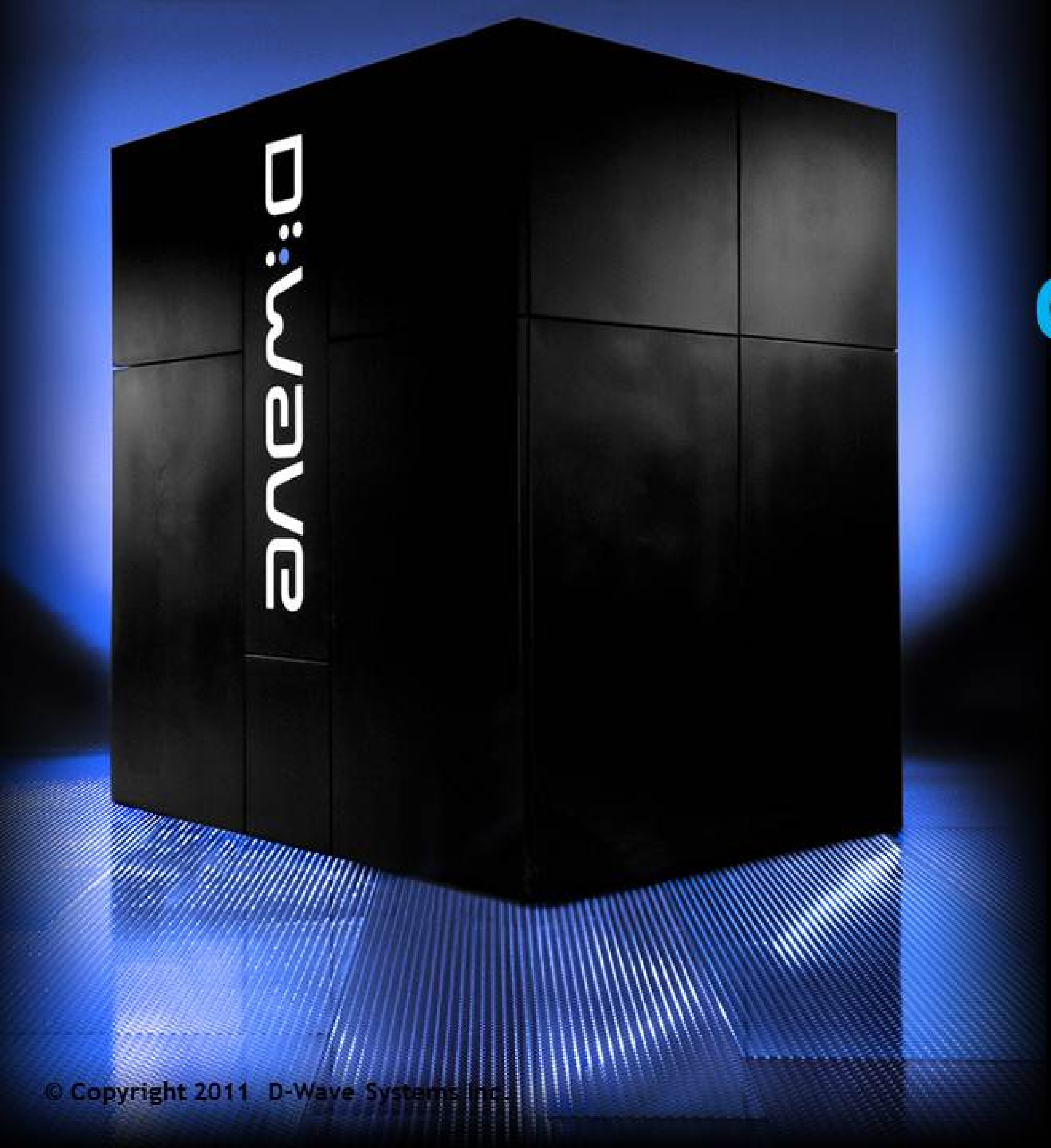
Meet the qubit – the quantum workhorse.

Ultimate Physics

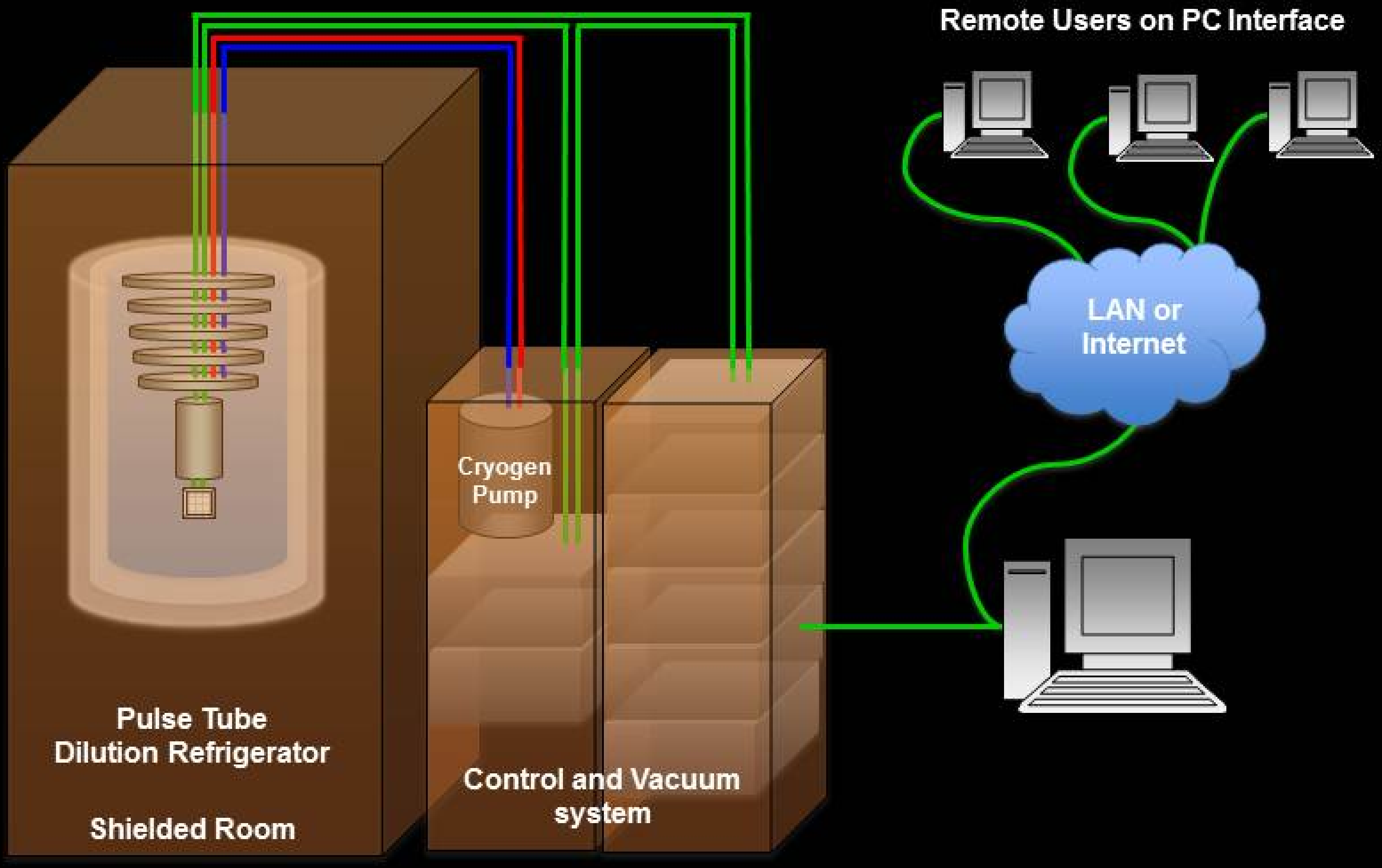
Computing close to absolute zero



The Quantum cloud

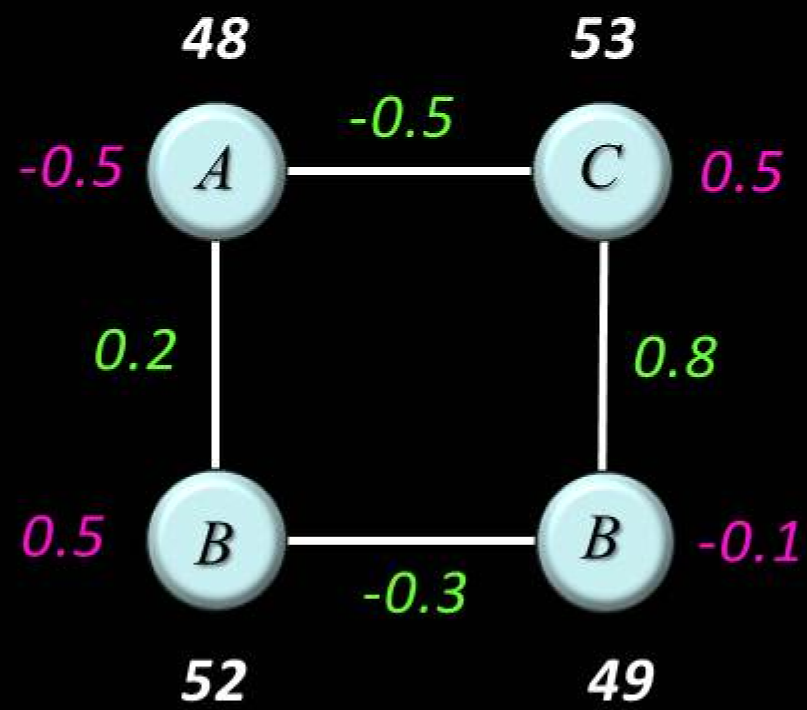


Multiverse Hacking



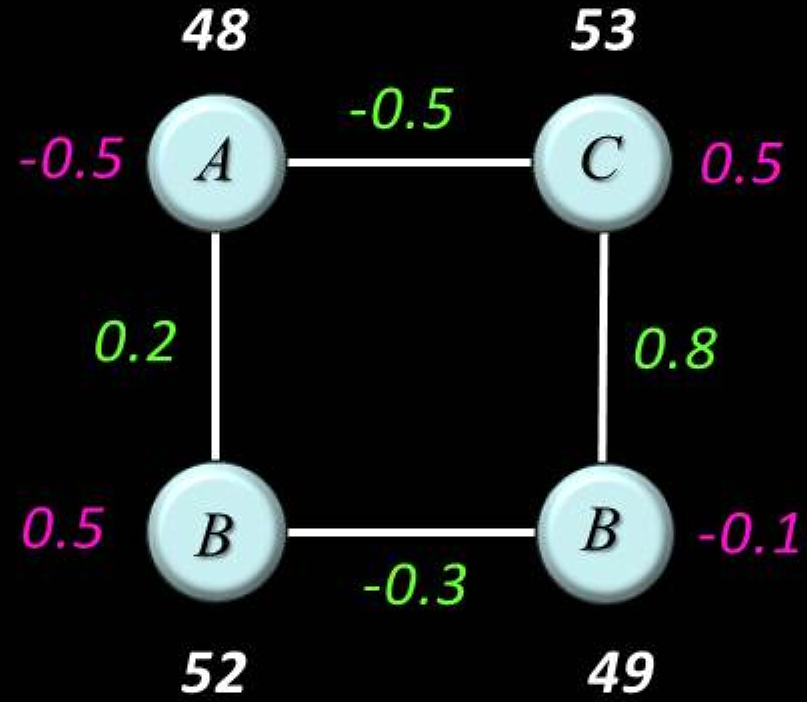
Hello quantum worlds!

```
import qupy
#D-Wave's Python API interface
qp = qupy.quantumprocessor('url',
'username', 'password')

#define the problem
h = [0]*128
h[48] = -0.5
h[52] = 0.5
h[49] = 0.1
h[53] = 0.5

j = {
    (48,52): 0.2,
    (49,52): -0.3,
    (48,53): -0.5,
    (49,53): 0.8 }

#send the problem to hardware
answer = qp.solve(h,j,1000)
print '48 = ', answer['solutions'][0][48]
print '49 = ', answer['solutions'][0][52]
print '52 = ', answer['solutions'][0][53]
print '53 = ', answer['solutions'][0][53]
```



Hello quantum worlds!

- ***48 = [0]***
- ***49 = [0]***
- ***52 = [1]***
- ***53 = [1]***

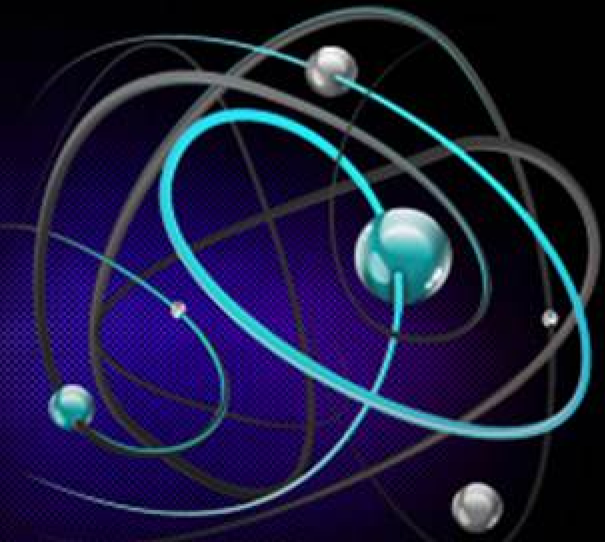


And if you like the multiverse…

…you've just created 2^N universes on-chip.

The first thing to understand when programming a QC: Think in terms of ENERGY.

So whatever problem you have, you HAVE to map it into a form where minimizing an energy gives you the correct answer.



What can we *energy program* from scratch in 10 minutes?

Let's choose a simple problem:
Simulating a NAND gate.

Logical NAND gate

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

Energy NAND gate

A	*B*	*C*	***ENERGY FUNCTION:*** $AB - 2(A + B)(1 - C) - 3C$
0	0	0	0
0	**0**	**1**	**-3**
0	1	0	-2
0	**1**	**1**	**-3**
1	0	0	-2
1	**0**	**1**	**-3**
1	**1**	**0**	**-3**
1	1	1	-2

Energy NAND gate

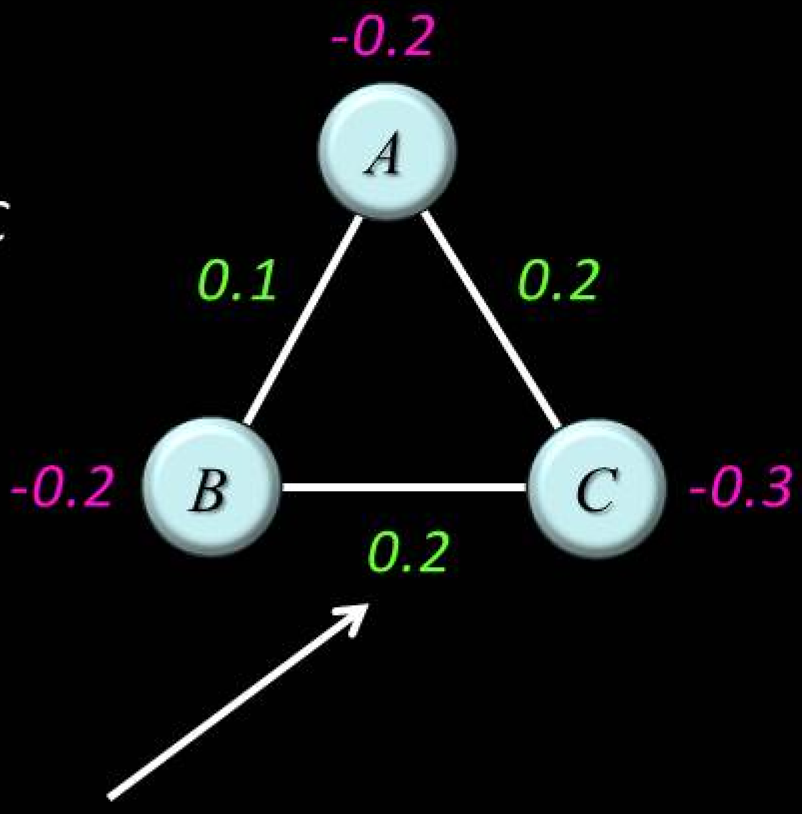
ENERGY FUNCTION:

AB - 2(A + B)(1 - C) - 3C

= 1AB – 2A – 2B +2AC + 2BC - 3C

So:

A = -2	AB = 1
B = -2	AC = 2
C = -3	BC = 2



Scale values to be <1

Let's program it!

```
import qupy #D-Wave's Python API
interface
qp = qupy.quantumprocessor('url',
'username', 'password')

#define the problem
h = [0]*128 # Set all non-used qubits to
zero
h[48] = -0.2
h[52] = -0.2
h[49] = -0.2
h[53] = -0.3
J[48-52] = 0.1
J[52-49] = -1
h[49-53] = 0.2
h[48-53] = 0.2

#send the problem to hardware
answer = qp.solve(h,J,1000)
print '48 = ', answer['solutions'][0][48]
print '52 = ', answer['solutions'][0][52]
print '53 = ', answer['solutions'][0][53]
```