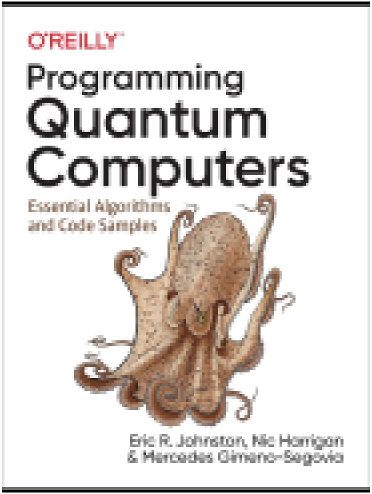
Informatique Quantique

Nicolas Boutry[1]

[1] Laboratoire de Recherche et Développement de l'EPITA (LRDE), France

Outline

Outline


O'REILLY
Programming
Quantum
Computers
Essential Algorithms
and Code Samples
Eric R. Johnston, Nic Harrigan
& Mercedes Gimeno-Segovia

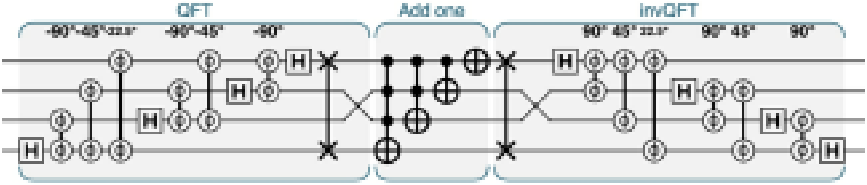


Figure P-1. Quantum programs can look a bit like sheet music

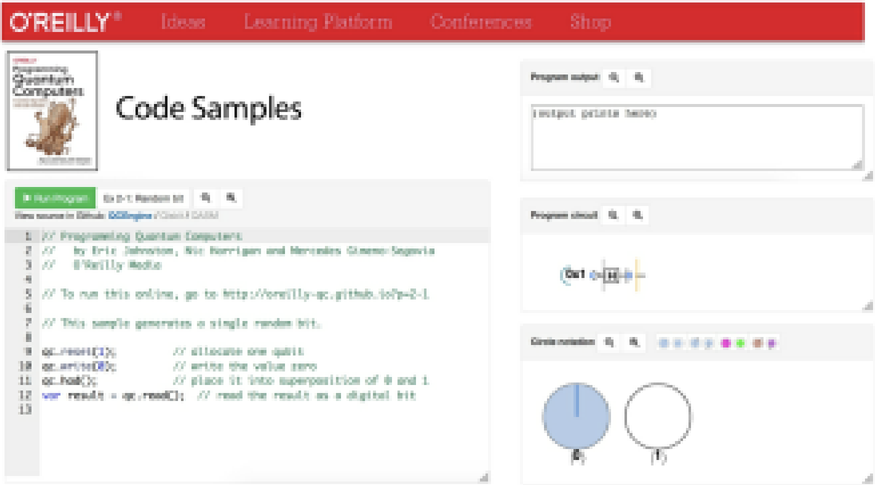


Figure 1-1. The QCEngine UI

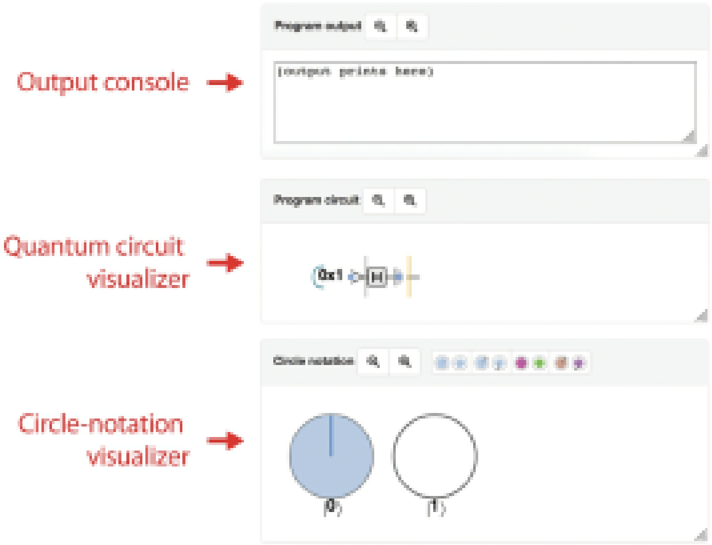


Figure 1-2. QCEngine UI elements for visualizing QPU results

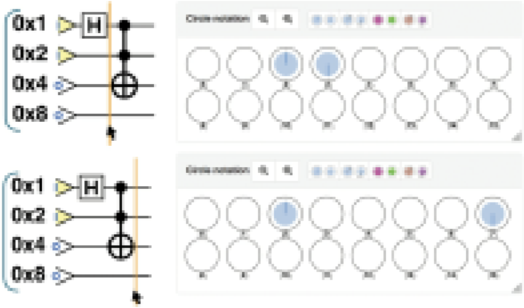
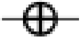
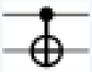
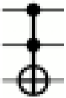
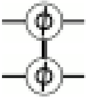
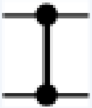
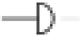
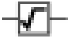


Figure 1-3. Stepping through a QCEngine program using the circuit and circle-notation visualizers

Table 1-1. Essential QPU instruction set

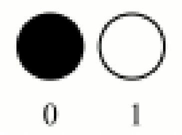
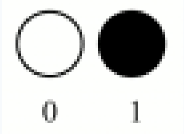
Symbol	Name	Usage	Description
	NOT (also X)	qc.not(t)	Logical bitwise NOT
	CNOT	qc.cnot(t,c)	Controlled NOT: if (c) then NOT(t)
	CCNOT (Toffoli)	qc.cnot(t,c1|c2)	if (c1 AND c2) then NOT(t)
	HAD (Hadamard)	qc.had(t)	Hadamard gate
	PHASE	qc.phase(angle,c)	Relative phase rotation

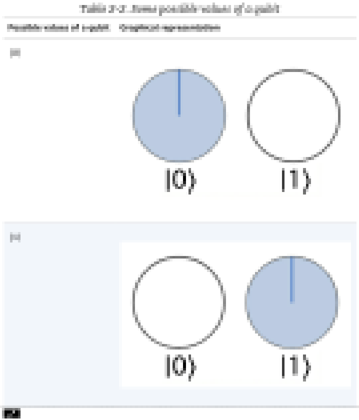
	[illegible]	[illegible]	Relative phase rotation by 180 °
	[illegible]	[illegible]	Relative phase rotation by 90 °
	[illegible]	[illegible]	Relative phase rotation by 45 °
	[illegible]	[illegible]	Conditional phase rotation

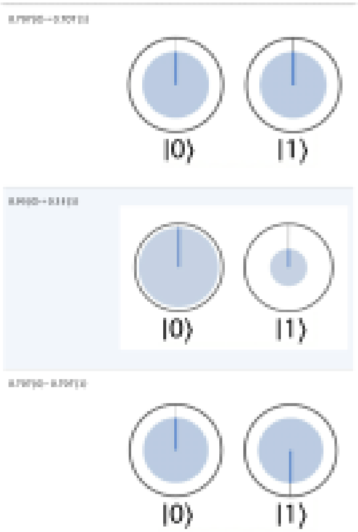
	[illegible]	[illegible]	Conditional phase rotation by 180°
	[illegible]	[illegible]	Read qubits, returning digital data
	[illegible]	[illegible]	Write conventional data to qubits
	[illegible]	[illegible]	Root-of-NOT operation

Outline

Table 2-1. Possible values of a conventional bit — a graphical representation

Possible values of a bit	Graphical representation
0	
1	


|0⟩
|1⟩
|0⟩
|1⟩


|0⟩
|1⟩
|0⟩
|1⟩
|0⟩
|1⟩

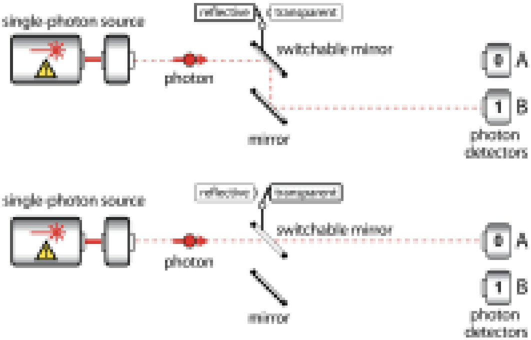


Figure 2-1. Using a photon as a conventional bit

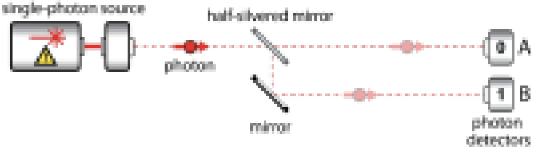


Figure 2-2. A simple implementation of one photonic qubit

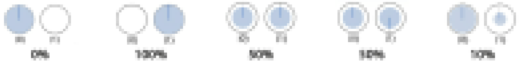


Figure 2-3. Probability of reading the value 1 for different superpositions represented in circle notation

Figure 2-4. Example relative phases in a single qubit

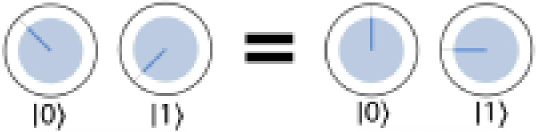
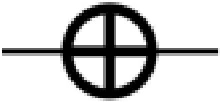


Figure 2-5. Only relative rotations matter in circle notation — these two states are equivalent because the relative phase of the two circles is the same in each case

Outline

QPU Instruction: NOT



NOT is the quantum equivalent of the eponymous conventional operation. Zero becomes one, and vice versa. However, unlike its traditional cousin, a QPU NOT operation can also operate on a qubit in superposition.

In circle notation this results, very simply, in the swapping of the $|0\rangle$ and $|1\rangle$ circles, as in Figure 2-6.

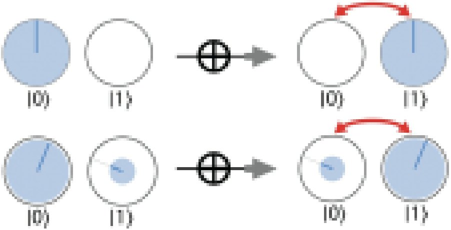
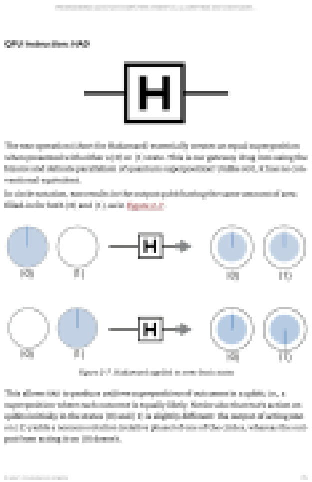
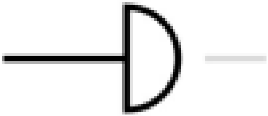


Figure 2-6. The NOT operation in circle notation

Reversibility: Just as in digital logic, the NOT operation is its own inverse; applying it twice returns a qubit to its original value.

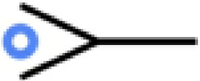


QPU Instruction: READ



The READ operation is the formal expression of the previously introduced readout process. READ is unique in being the only part of a QPU's instruction set that potentially returns a random result.

QPU Instruction: WRITE



The WRITE operation allows us to initialize a QPU register before we operate on it. This is a deterministic process.

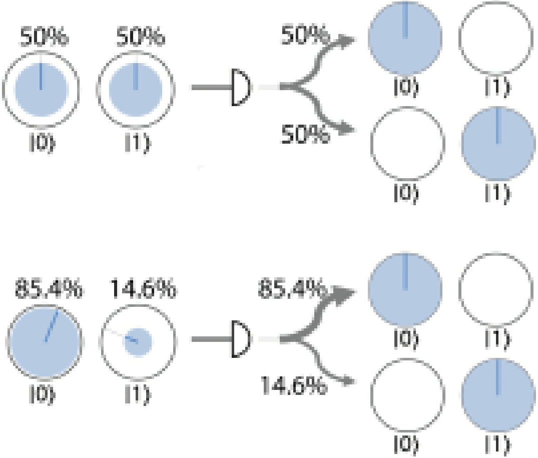


Figure 2-8. The READ operation produces random results

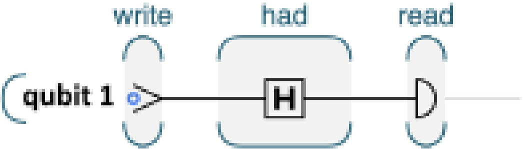


Figure 2-3. Generating a perfectly random bit with a QPU

SAMPLE CODE

Run this sample online at http://oreilly-qc.github.io?p=2-1.

Example 2-1. One random bit

```
qc.reset(1);          // allocate one qubit
qc.write(0);          // write the value zero
qc.had();             // place it into superposition of 0 and 1
var result = qc.read();  // read the result as a digital bit
```

NOTE

All of the code samples in this book can be found online at http://oreilly-qc.github.io, and can be run either on QPU simulators or on actual QPU hardware. Running these samples is an essential part of learning to program a QPU. For more information, see Chapter 1.

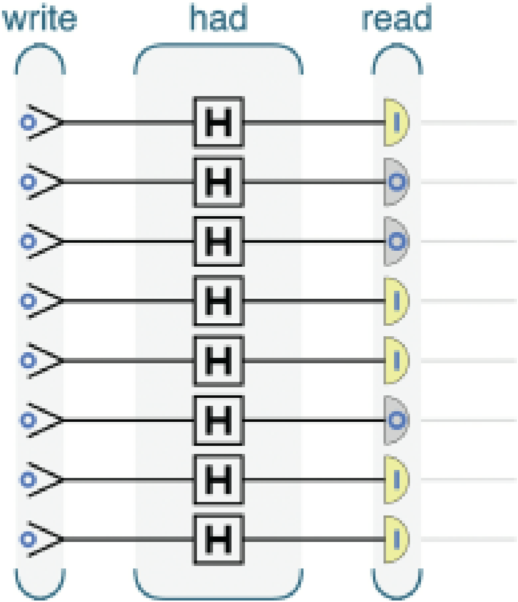
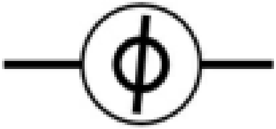


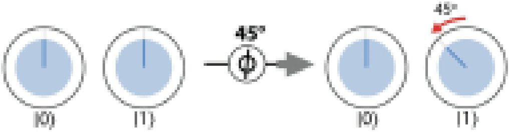
Figure 2-10. Generating one random byte

QPU Instruction: PHASE(θ)



The PHASE(θ) operation also has no conventional equivalent. This instruction allows us to directly manipulate the *relative phase* of a qubit, changing it by some specified angle. Consequently, as well as a qubit to operate on, the PHASE(θ) operation takes an additional (numerical) input parameter — the angle to rotate by. For example, PHASE(45) denotes a PHASE operation that performs a 45° rotation.

In circle notation, the effect of PHASE(θ) is to simply rotate the circle associated with $|1\rangle$ by the angle we specify. This is shown in Figure 2-11 for the case of PHASE(45).


45°
ϕ
45°
|0⟩
|1⟩
|0⟩
|1⟩

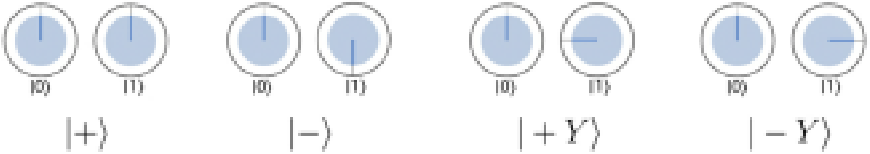


Figure 2-12. Four very commonly used single-qubit states

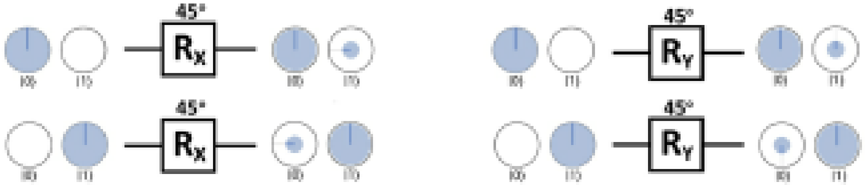


Figure 2-13. ROTX and ROTY actions on 0 and 1 input states

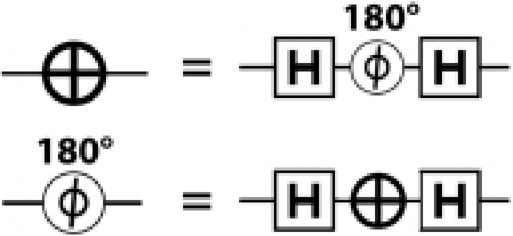


Figure 2-14. Building equivalent operations

Figure 2-13. An impossible operation for conventional bits

There's more than one way to construct this operation, but Figure 2-14 shows one simple implementation.

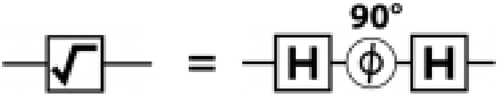
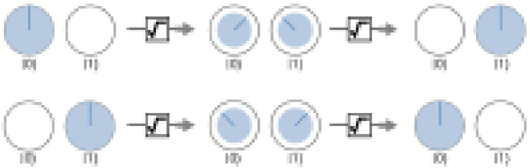


Figure 2-14. Recipe for ROOT-of-NOT


|0⟩
|1⟩
|0⟩
|1⟩
|0⟩
|1⟩
|0⟩
|1⟩
|0⟩
|1⟩
|0⟩
|1⟩

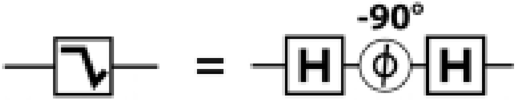


Figure 2-18. Inverse of RNOT

Outline

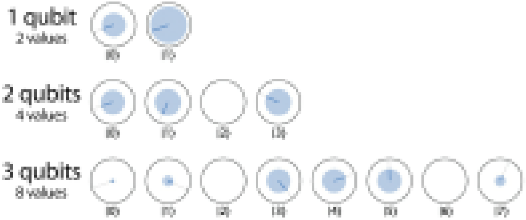


Figure 3-1. Circle notation for various numbers of qubits

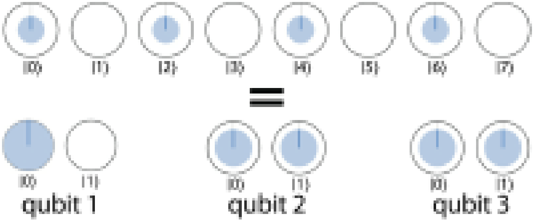


Figure 3-2. Some multi-qubit quantum states can be understood in terms of single-qubit states

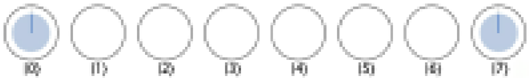


Figure 3-3. Quantum relationships between multiple qubits

This represents a state of three qubits in equal superposition of $|0\rangle$ and $|7\rangle$. Can we visualize this in terms of what each individual qubit is doing like we could in Figure 3-2? Since 0 and 7 are 000 and 111 in binary, we have a superposition of the three qubits being in the states $|0\rangle|0\rangle|0\rangle$ and $|1\rangle|1\rangle|1\rangle$. Surprisingly, in this case, there is no way to write down circle representations for the individual qubits! Notice that reading out the three qubits always results in us finding them to have the same values (with 50% probability that the value will be 0 and 50% probability it will be 1). So clearly there must be some kind of link between the three qubits, ensuring that their outcomes are the same.

This link is the new and powerful *entanglement* phenomenon. Entangled multi-qubit states cannot be described in terms of individual descriptions of what the constituent qubits are doing, although you're welcome to try! This entanglement link is only describable in the configuration of the *whole* multi-qubit register. It also turns out to be impossible to produce entangled states from only *single-qubit* operations. To explore entanglement in more detail, we'll need to introduce multi-qubit operations.

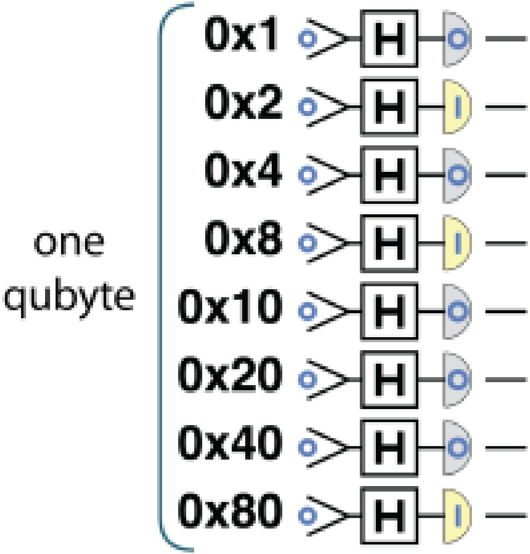


Figure 3-4. Labeling qubits in a qubyte

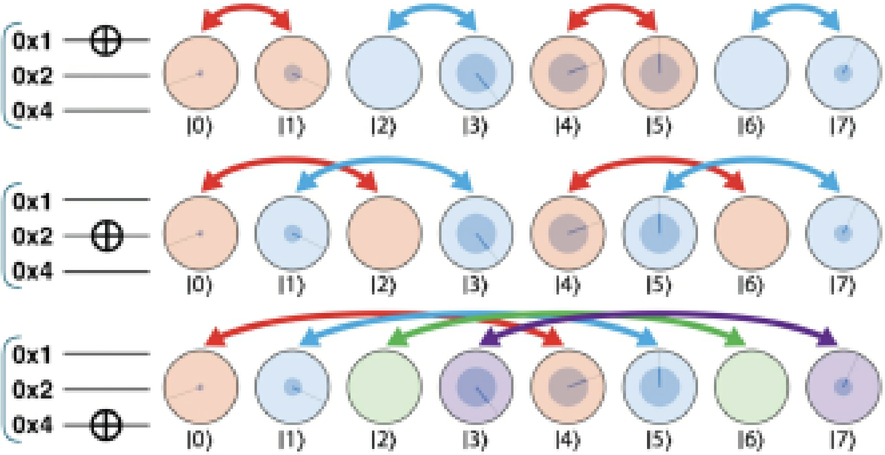


Figure 3-5. The NOT operation swaps values in each of the qubit's operator pairs; here, its action is shown on an example multi-qubit superposition

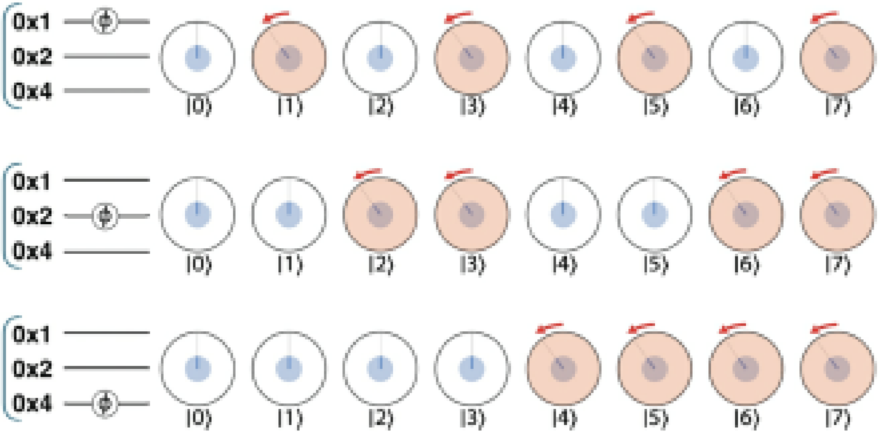


Figure 3-6. Single-qubit phase in a multi-qubit register

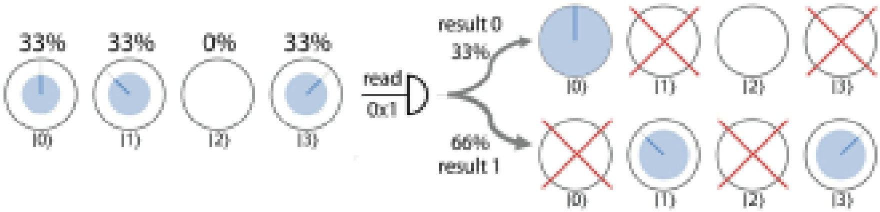


Figure 3-7. Reading one qubit in a multi-qubit register

Quand on lit le bit des unités :

- soit on obtient 0, càd 00 $= |0\rangle$ et 10 $= |2\rangle$ comme possibilités,
- soit on obtient 1, càd 01 $= |1\rangle$ et 11 $= |3\rangle$ comme possibilités.

Les énergies sont ensuite renormalisées telles que leur somme vaut 1.

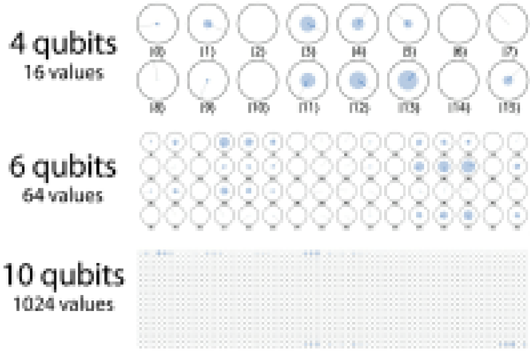
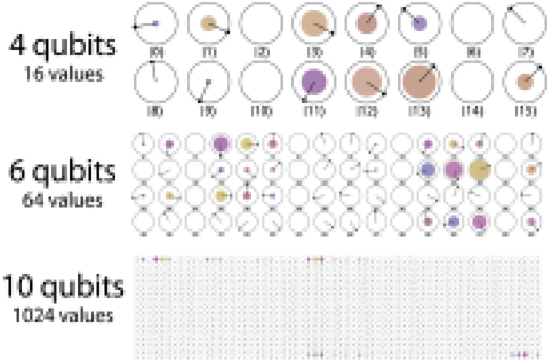
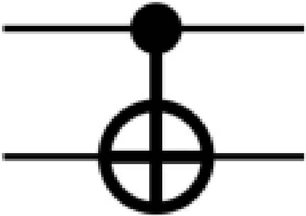


Figure 3-8. Circle notation for larger qubit counts


4 qubits
16 values
6 qubits
64 values
10 qubits
1024 values

QPU Instruction: CNOT



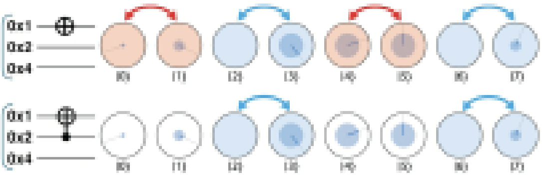


Figure 3-10. NOT versus CNOT in operation

Bell pair:

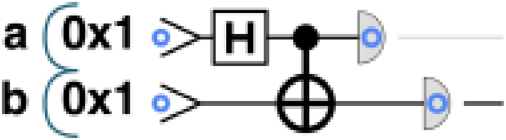


Figure 3-11. CNOT with a control qubit in superposition

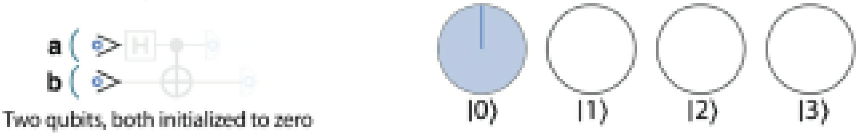


Figure 3-12. Bell pair step 1

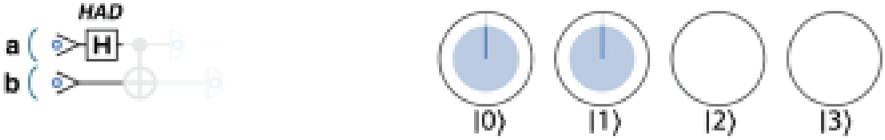


Figure 3-13. Bell pair step 2

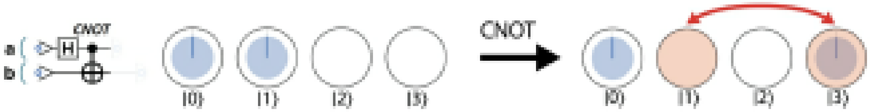


Figure 3-14. Bell pair step 3

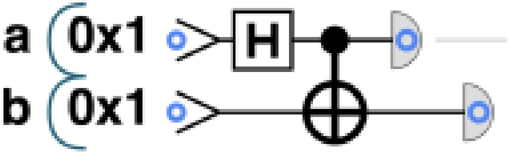
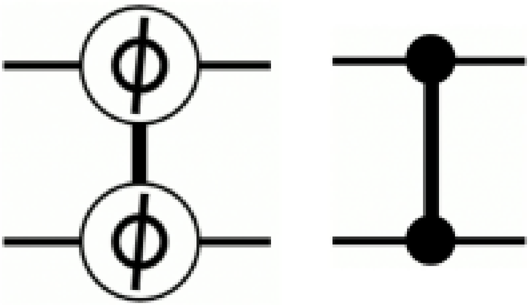


Figure 3-15. Bell pair circuit

QPU Instructions: CPHASE and CZ



Another very common two-qubit operation is CPHASE(θ). Like the CNOT operation, CPHASE employs a kind of entanglement-generating conditional logic. Recall from Figure 3-6 that the single-qubit PHASE(θ) operation acts on a register to rotate (by angle θ) the $|1\rangle$ values in that qubit's operator pairs. As CNOT did for NOT, CPHASE restricts this action on some target qubit to occur only when another control qubit assumes the value $|1\rangle$. Note that CPHASE only acts when its control qubit is $|1\rangle$, and when it does act, it only affects target qubit states having value $|1\rangle$. This means that a CPHASE(θ) applied to, say, qubits 0x1 and 0x4 results in the rotation (by θ) of all circles for which both these two qubits have a value of $|1\rangle$. Because of this particular property, CPHASE has a symmetry between its inputs not shared by CNOT. Unlike with most other controlled operations, it's irrelevant which qubit we consider to be the target and which we consider to be the control for CPHASE.

Attention au bogue du $|1\rangle$ qui tourne aussi à la 1ère ligne ci-dessous ...

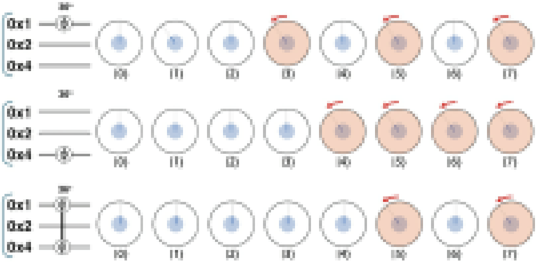


Figure 3-16. Applying CPHASE in circle notation

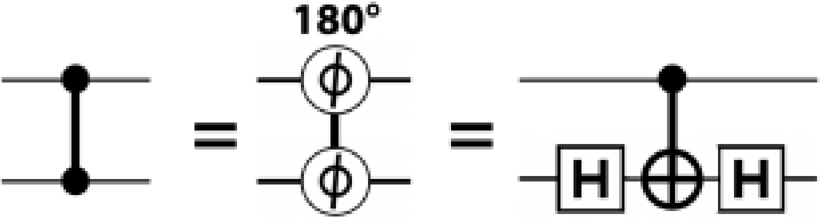


Figure 3-17. Three representations of CPHASE(180)

QPU Trick: Phase Kickback

Once we start thinking about altering the phase of one QPU register *conditioned* on the values of qubits in some other register, we can produce a surprising and useful effect known as *phase kickback*. Take a look at the circuit in Figure 3-18.

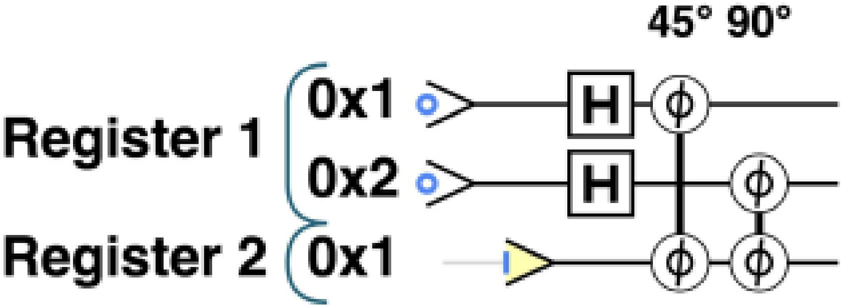
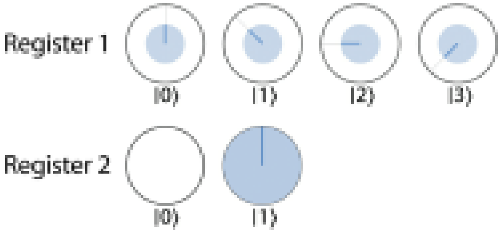


Figure 3-18. Circuit for demonstrating phase-kickback trick


Register 1
|0⟩
|1⟩
|2⟩
|3⟩
Register 2
|0⟩
|1⟩

SAMPLE CODE

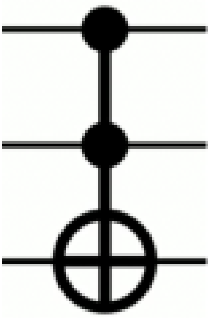
Run this sample online at http://oreilly-qc.github.io?p=3-3.

Example 3-3. Phase kickback

```
qc.reset(3);
// Create two registers
var reg1 = qint.new(2, 'Register 1');
var reg2 = qint.new(1, 'Register 2');
reg1.write(0);
reg2.write(1);
// Place the first register in superposition
reg1.had();
// Perform phase rotations on second register,
// conditioned on qubits from the first
qc.phase(45, 0x4, 0x1);
qc.phase(90, 0x4, 0x2);
```

Phase kickback will be of great use in Chapter 8 to understand the inner workings of the *quantum phase estimation* QPU primitive, and again in Chapter 13 to explain how a QPU can help us solve systems of linear equations.

QPU Instruction: CCNOT (Toffoli)



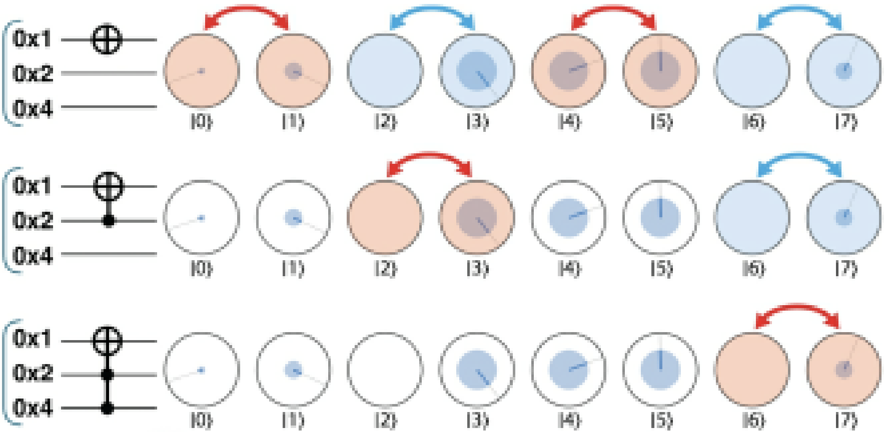
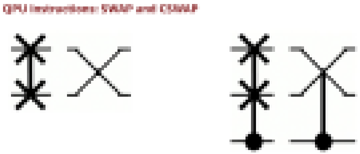
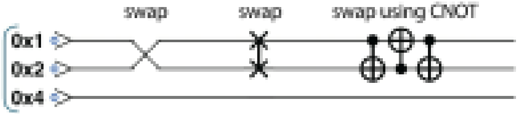
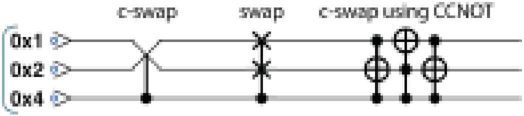


Figure 3-20. Adding conditions makes NOT operations more selective


QPU instructions: SWAP and CSWAP


swap
swap
swap using CNOT
0x1
0x2
0x4


c-swap
swap
c-swap using CCNOT
0x1
0x2
0x4

$\rightarrow$ voir exercice Déphasage conditionnel