Programming Quantum Computers (Modules I: AL)

(**Subtrack of** Quantum Computing: An App-Oriented Approach)

Moez A. AbdelGawad

moez@{cs.rice.edu, alexu.edu.eg, srtacity.sci.eg}

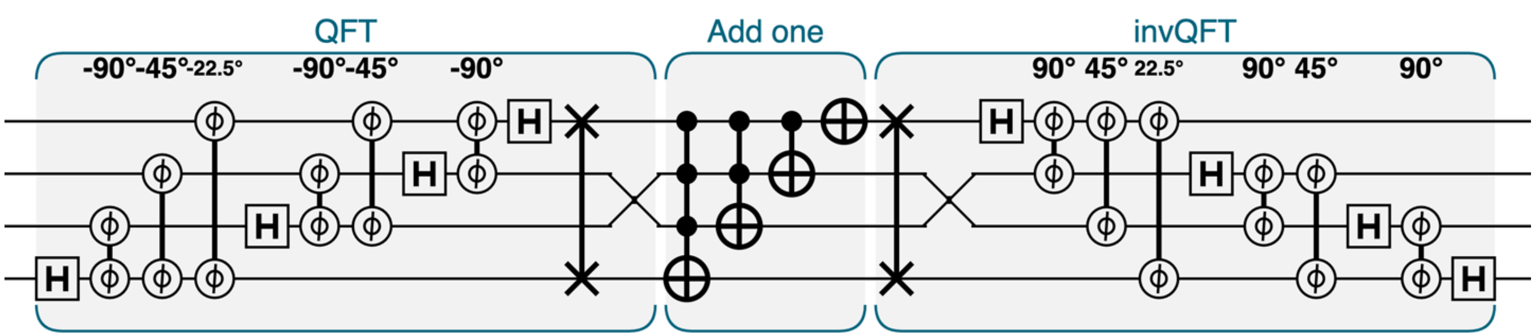
Sat., Nov. 16th, 2019

Quantum Computers are Real

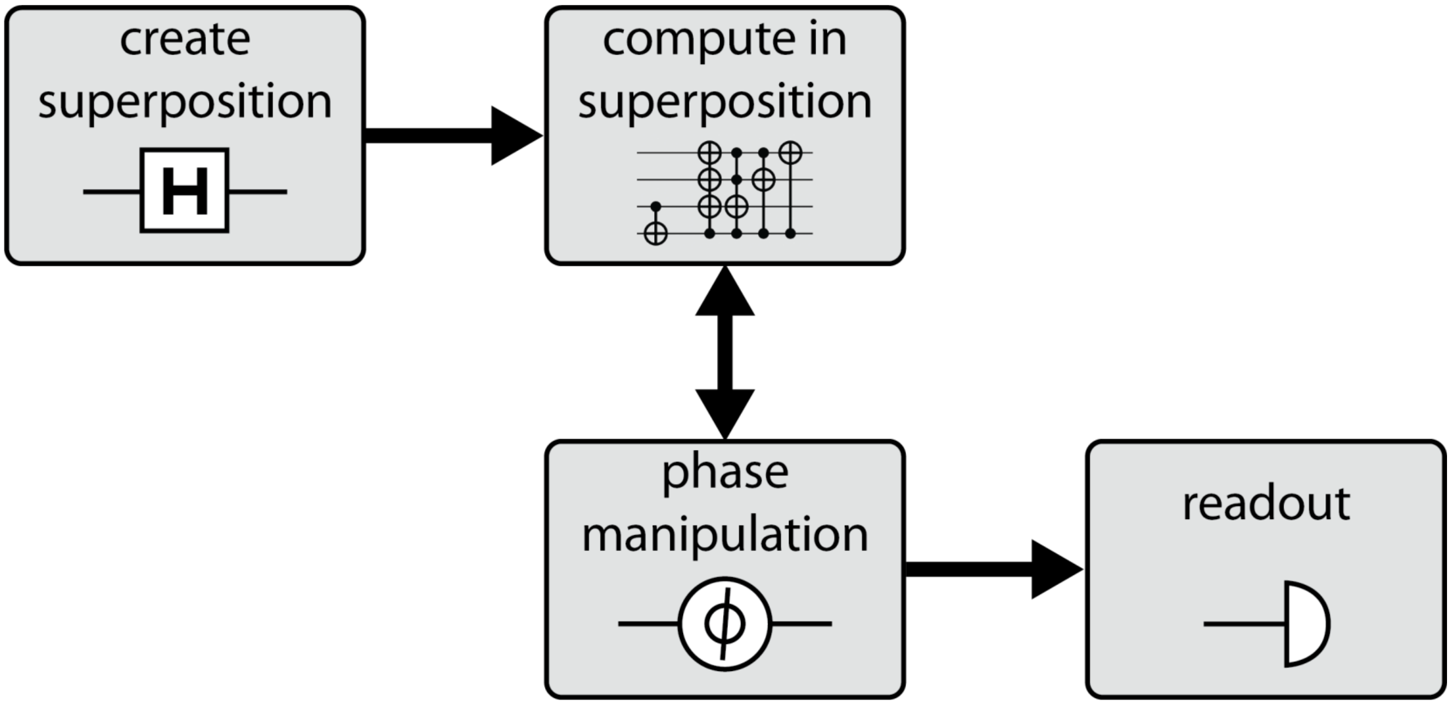
- What are they <u>useful</u> for?
  - Let's discover, by programming them!

- A hands-on approach to programming QCs/QPUs.
  - By doing; i.e., by writing code & building programs.
  - Using simulators, since real QCs are harder-to-access (so far).

- Goals: Read, understand, write, and *debug* quantum programs.
  - Ones like the following.

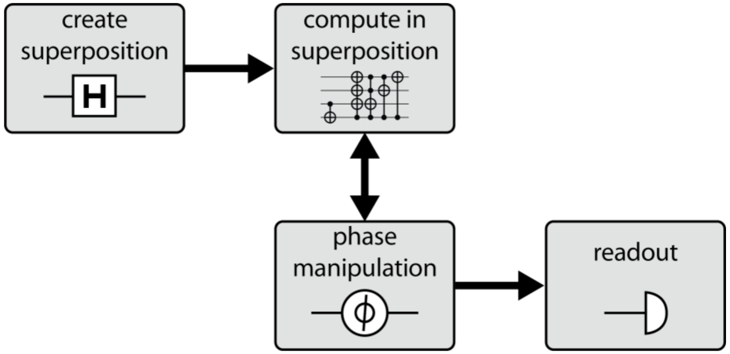


Structure of Quantum Apps



Structure of Quantum Apps

- Tendency to such structure, very roughly.
- Compute in superposition.
  - Implicit parallelism.
- Phase manipulation.
  - Practicality. Relative phase info directly inaccessible (unREADable).
- Modules are combined (*composed*) to define full quantum application.
  - Possibly in *iterations*.
  - Programming: A 'division problem'*.
- QP is an *art* (too).
  - "Quantum Knuth" … Anyone?



Quantum Modules Covered

Module	Type
Digital arithmetic and logic (AL)	Compute in superposition
Amplitude amplification (AA)	Phase manipulation
Quantum Fourier transform (QFT)	Phase manipulation
Phase estimation (PE)	Phase manipulation
Quantum data types (Sim)	Superposition creation

*See 'Conceptual Mathematics' by Lawvere and Schanuel, 2009. (An undergraduate-level introduction to category theory.)

QUANTUM ARITHMETIC AND LOGIC

Lecture Outline

- Quantum Arithmetic and Logic.
  - Some Special Properties of Quantum Programming.
  - Addition.
    - Adding Constants and Variables.
    - Negative Integers.
  - More Complex Arithmetic.
    - No Simple Multiplication Module.
    - Adding Squares.

Lecture Outline

- Quantum Arithmetic and Logic.
  - Quantum-Conditional Execution.
  - Phase-Encoded Results.
  - Reversibility and Scratch Qubits.
    - Uncomputing (“un-entangling” scratch qubits).
  - Basic Quantum Boolean Logic.

QP: Strangely Different

- Some Special Properties of Quantum Programming.
  - Computing in Superposition of States.
    - Implicit parallelism, causes speedup.
  - No-cloning: No Copying of State.
    - Conventional assignment operation (‘:=’ or ‘=’) is unavailable.
    - Workarounds are common (e.g., using entanglement).

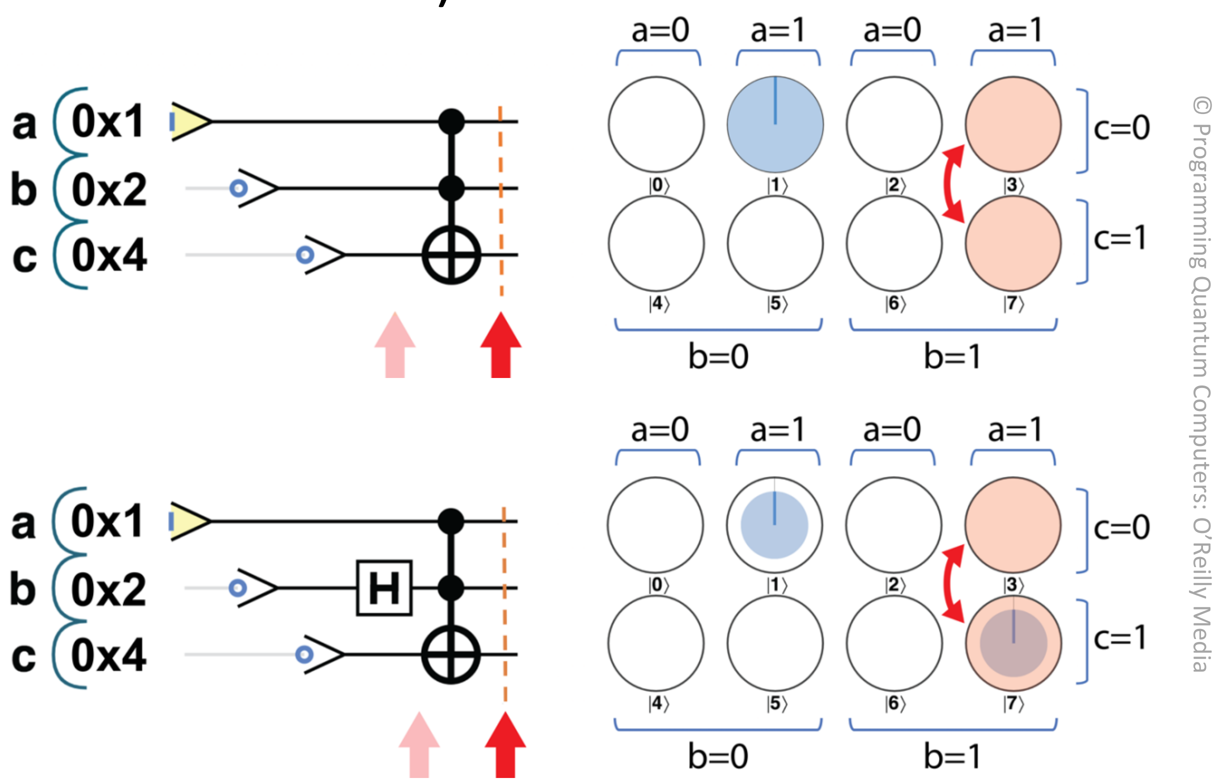
QP: Strangely Different

- Some Special Properties of Quantum Programming.
  - Reversibility of Primitive Ops and of Quantum Computation.
    - Except READ (and WRITE).
    - Due to laws of quantum mechanics.
      - Permutations, group theory, symmetry, and Rubik's cube*!
    - Often forces creative thinking when reproducing conventional operations.
    - Workarounds are common (e.g., scratch/ancilla qubits).
  - Entanglement of Qubits.
    - Causes controlled qubits to influence controlling qubits ("kickback").

*See 'Visual Group Theory' by Nathan Carter, 2009. (A highly-visual, intuitive and entertaining presentation of group theory.)

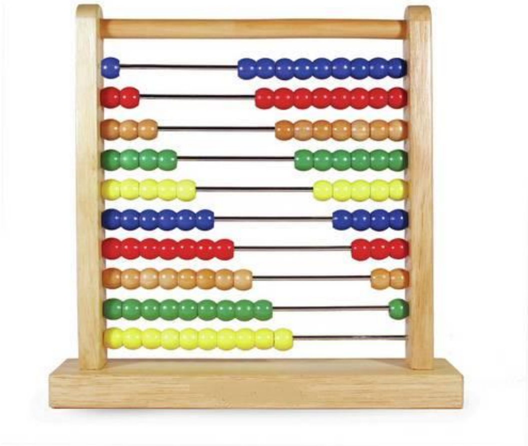
QP: Strangely Different

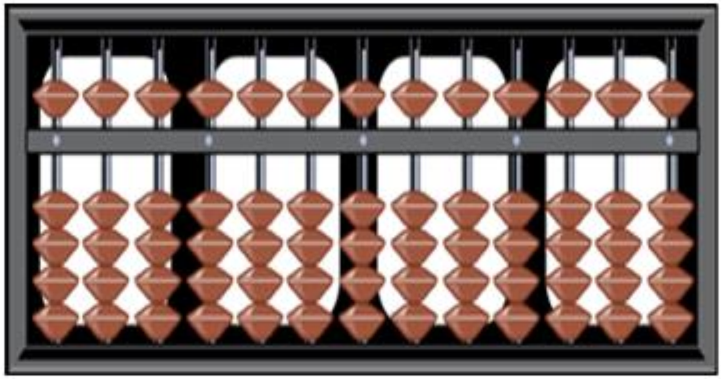
- Quantum Toffoli: Simultaneously, invert *and* not invert ("quantum flow control").



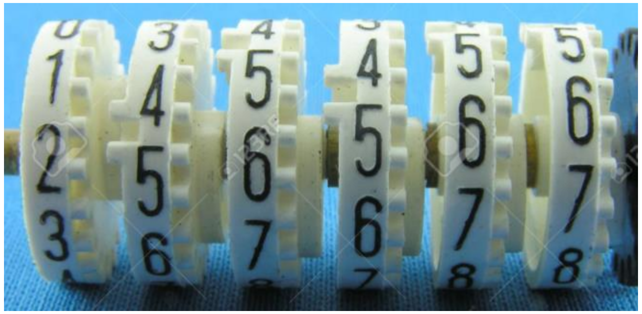
Adders and Counters: (Very) Classical Ones

- The Abacus: Starting point for learning math.



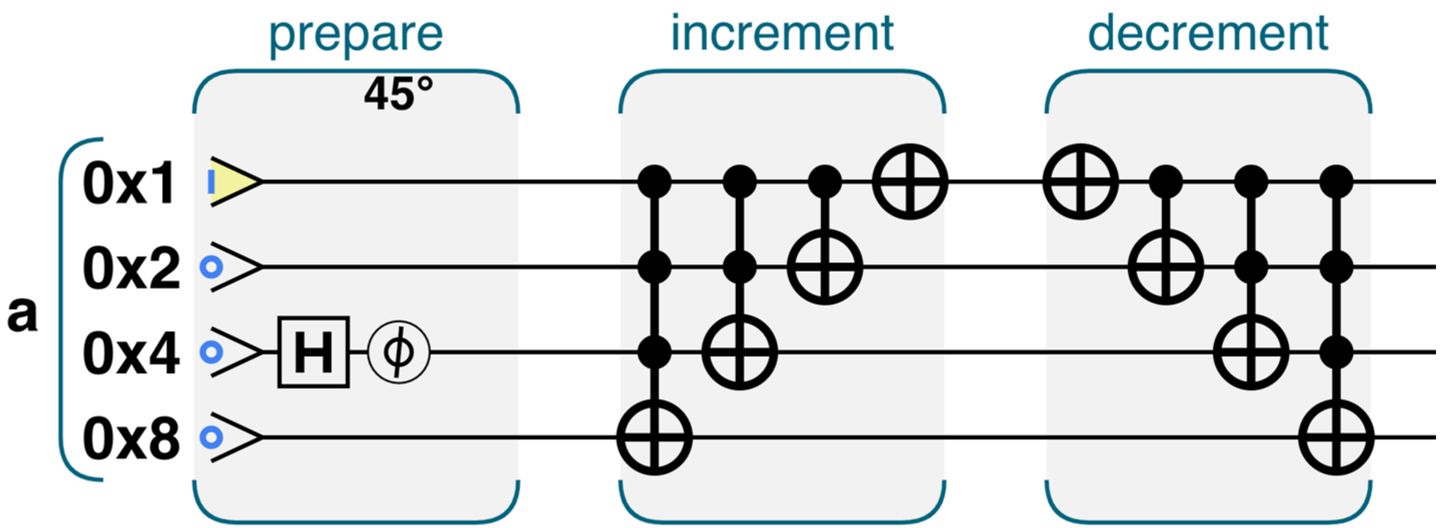


- And counters…



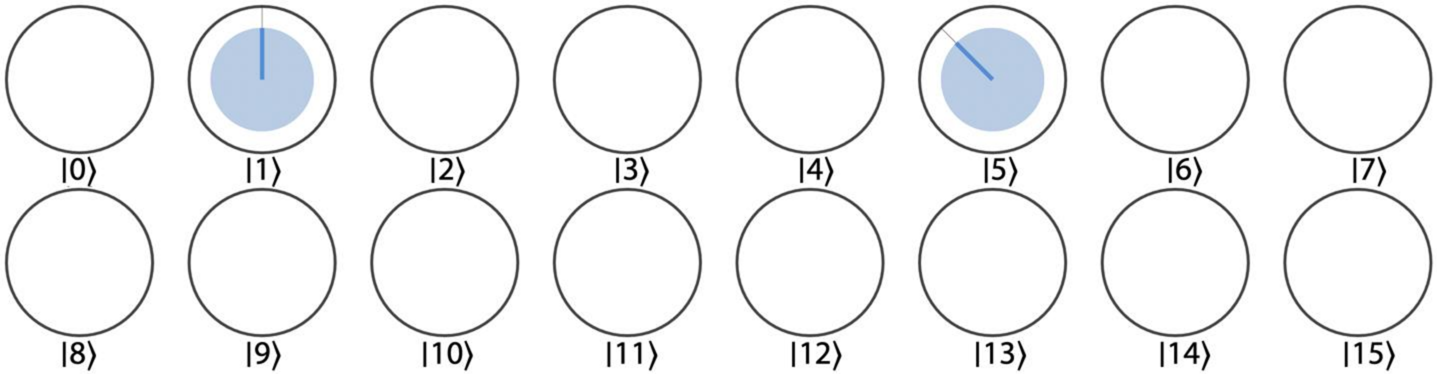
Quantum Adder (and Subtractor)

- Quantum Adder: Starting point for learning *computing in superposition*.

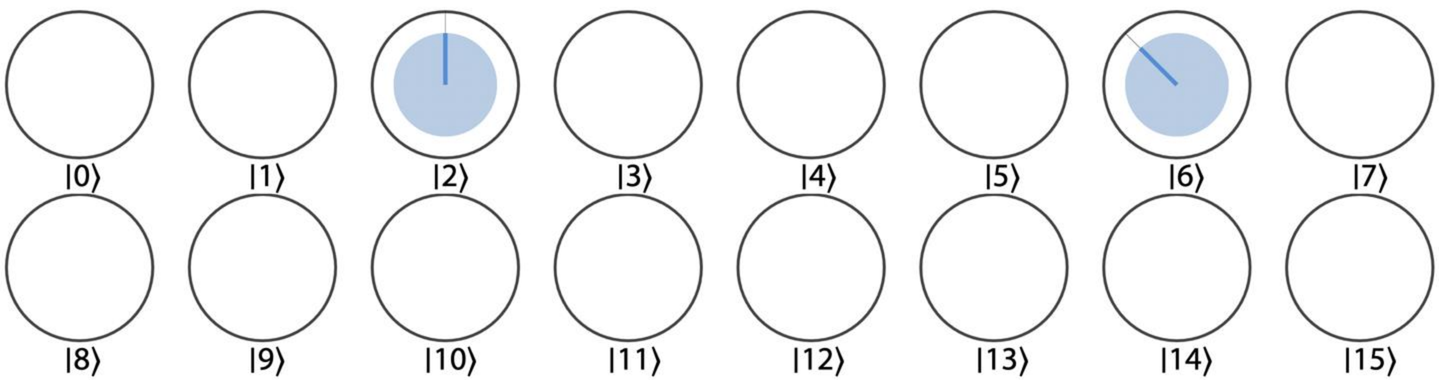


- Operation: ‘if *all* lower bits are 1, flip top bit’ (same as in previous slide).
  - Wrap-on-overflow addition (like in conventional programming).
  - Modular (a.k.a., “wall clock”) arithmetic (modulo 2^m).
- Reversible: Decrement is increment with constituent operations reversed.

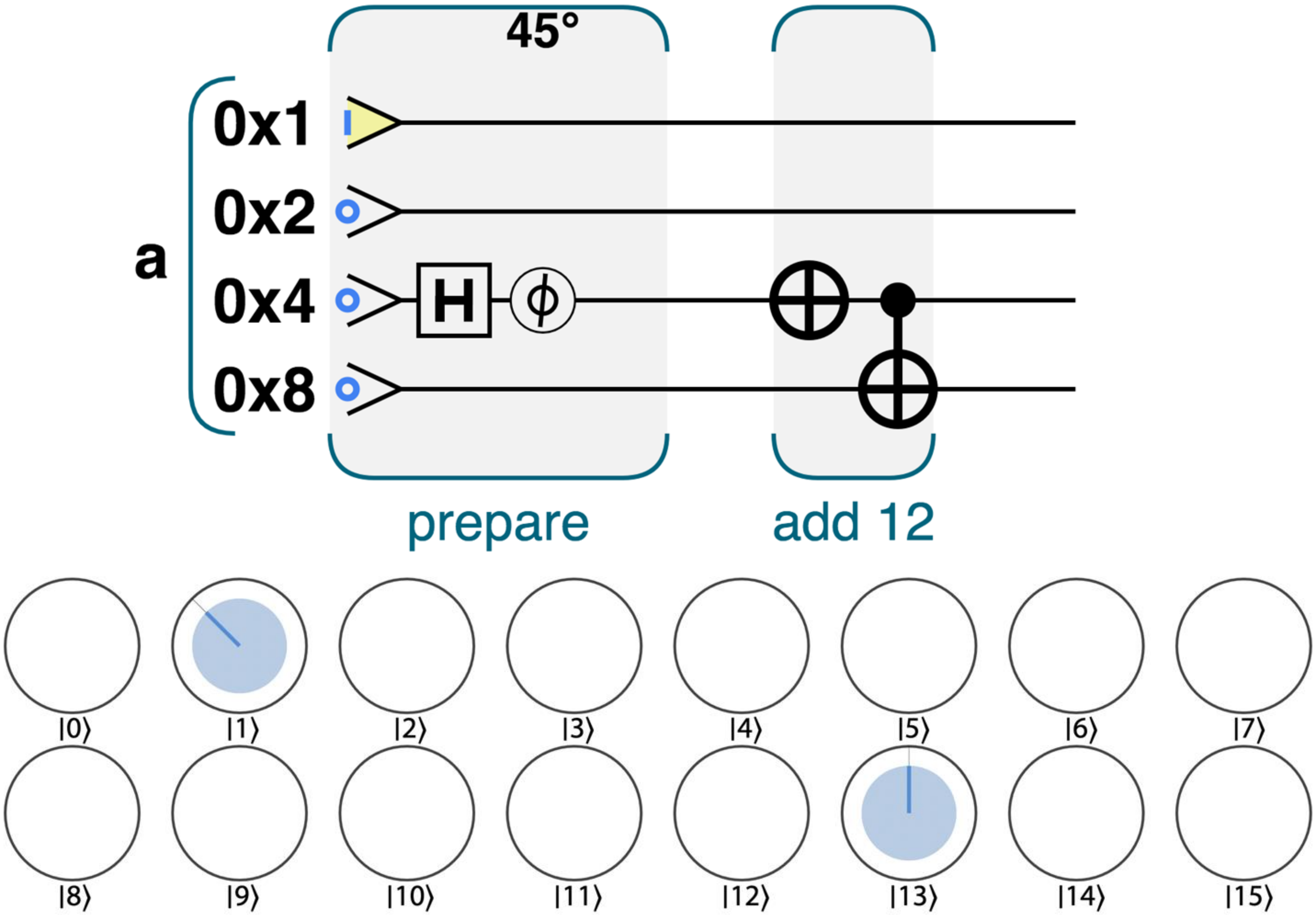
Quantum Adder: Computing in Superposition (*Quantum Parallelism*)



after adding one (<u>Hands-on</u>)

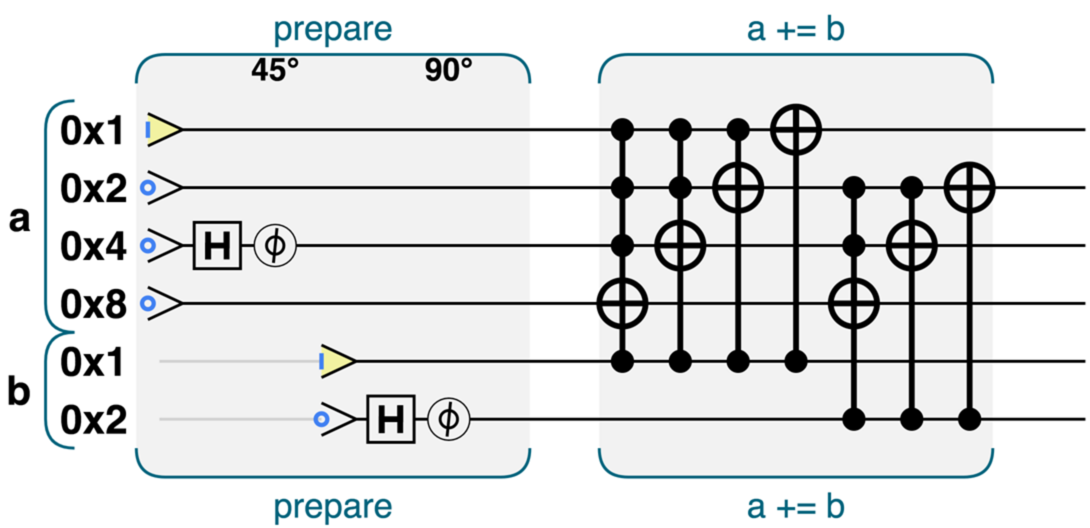


Constant Addition



Variable Addition

- Having ‘*c = a + b*’ violates reversibility (contents of *c* are lost) and violates ‘no cloning’ (*b = c - a* will effectively copy *a* to *b*).
- Instead, we have a quantum ‘*a += b*’.



- Both *a* and *b* can be in superposition. (Hands-on)
- Quantum ‘*a -= b*’ obtained by reversing order of gates. (Hands-on)
- Exercise: What if *b* has 3 qubits? Or 4 qubits? Or 5 qubits? (Hands-on)

Negative Integers

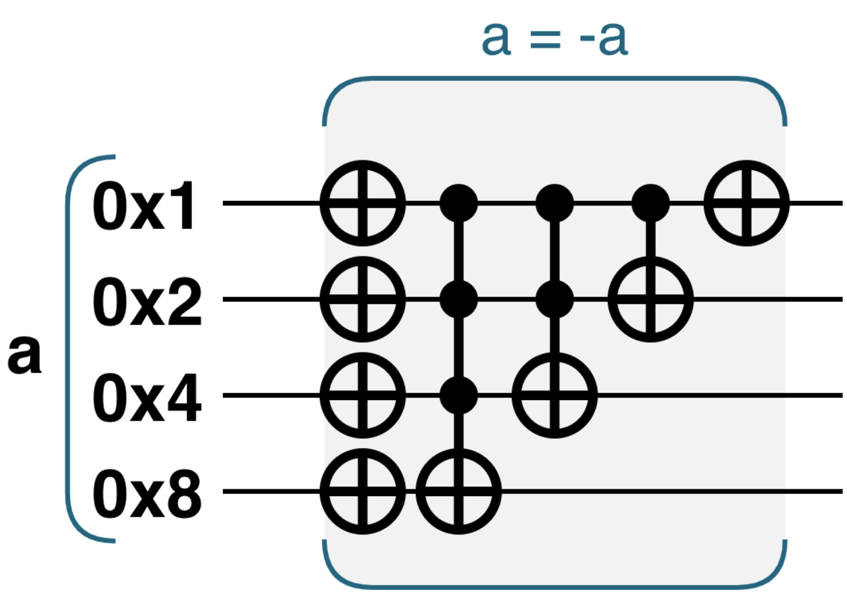
0	1	2	3	–4	–3	–2	–1
000	001	010	011	100	101	110	111

Two's Complement Encoding

- Values $-2^{m-1} \,..\, 2^{m-1} - 1$.
- Ops (e.g., addition) work out of the box!
- Highest-order bit indicates sign.
- Keep track of encoding (a "type system").

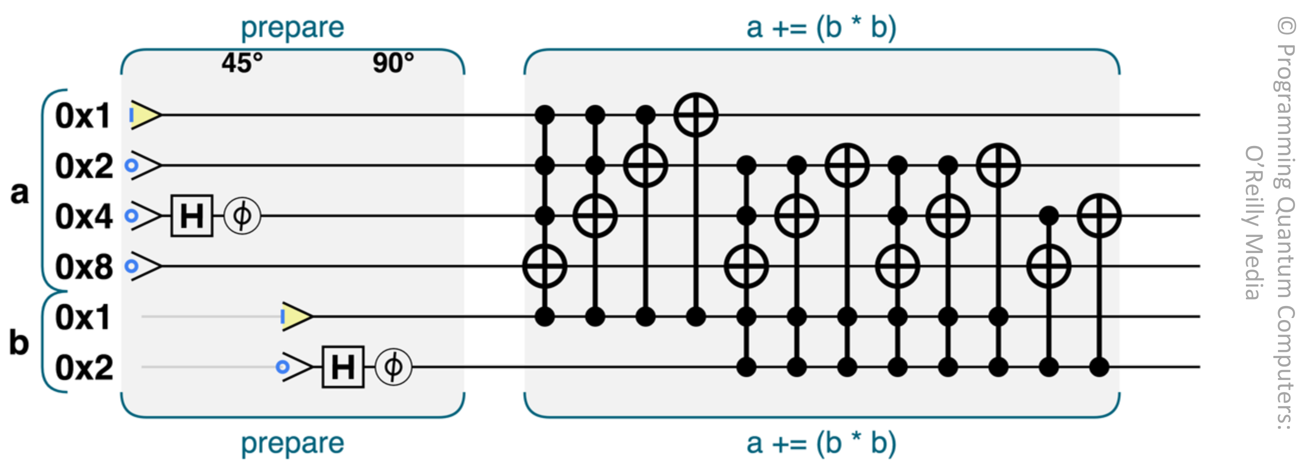
- *Negate* = Flip all qubits, then add 1.
  - Negation of -4 is -4.
  - QQ: Negation in one's complement?

- Exercise: What if gate order is reversed?
  - Hands-on.



No Multiplication. Add Squares.

- Multiplication is hard to perform reversibly.
- The related AddSquare (‘*a += b*b*’) is reversible.

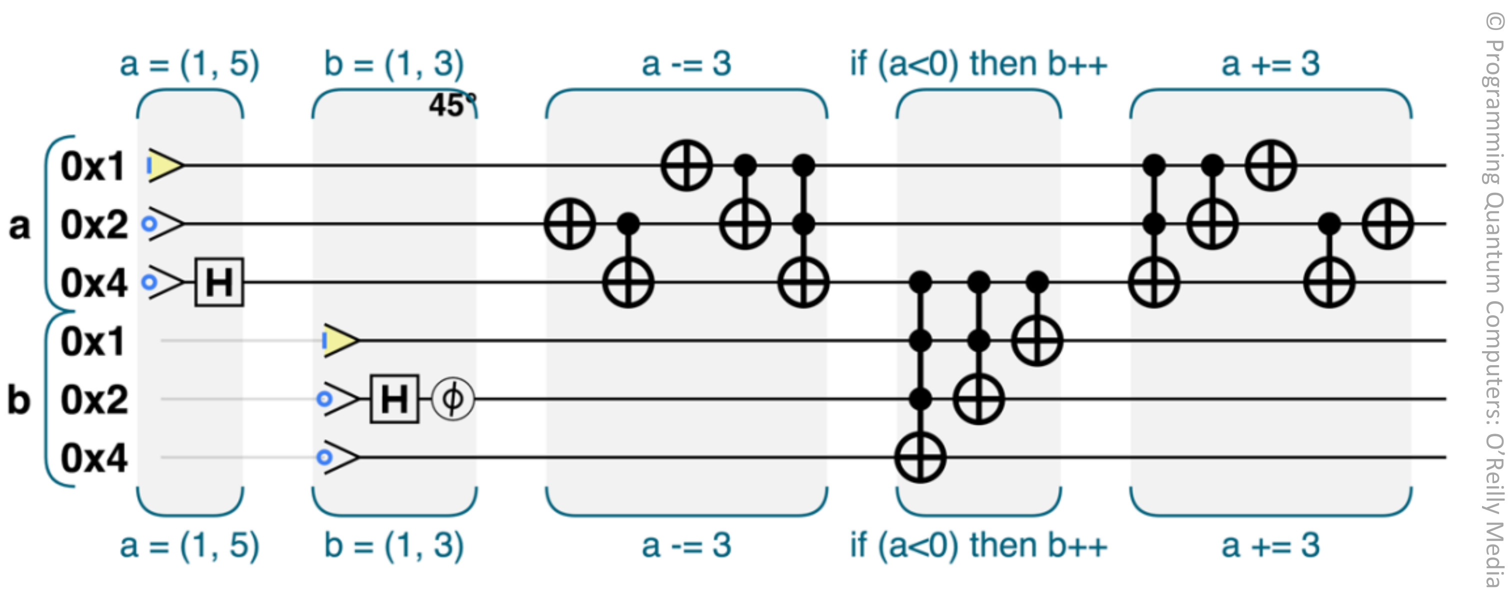


Multiplication by repeated addition, conditional on bits of *b*.

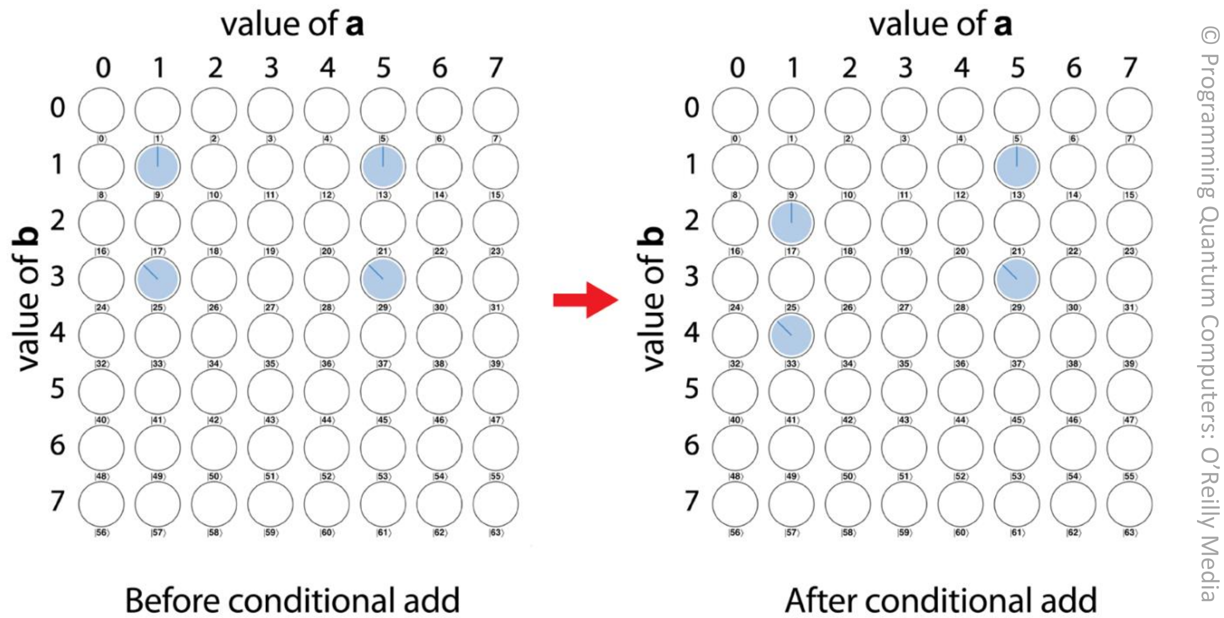
- Reversing order of gates gives ‘*a -= b*b*’.
- Exercise: What if *b* has 3, 4, or 5 qubits? (Hands-on)

Quantum-Conditional Execution

- Conditionally execute operations in superposition.
- E.g., increment *b* only for *a* values 0, 1, 2, and 7.



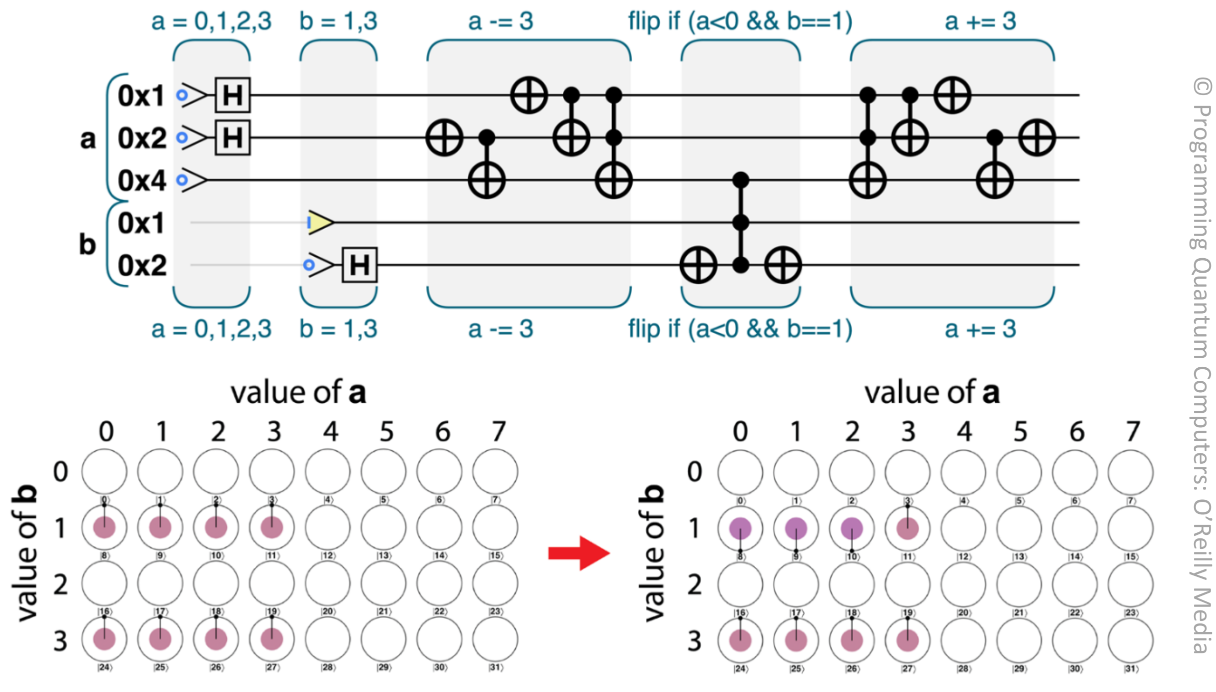
Quantum-Conditional Execution



- Exercise: What does value 7 of *a* stand for?
- <u>Hands-on</u>: Different values of *a*.
- Exercise: Having more control over condition (i.e., over permissible values of *a*)?

Phase-Encoded Results

- Same range of *a* as before, but operation changes *phases* not magnitudes.



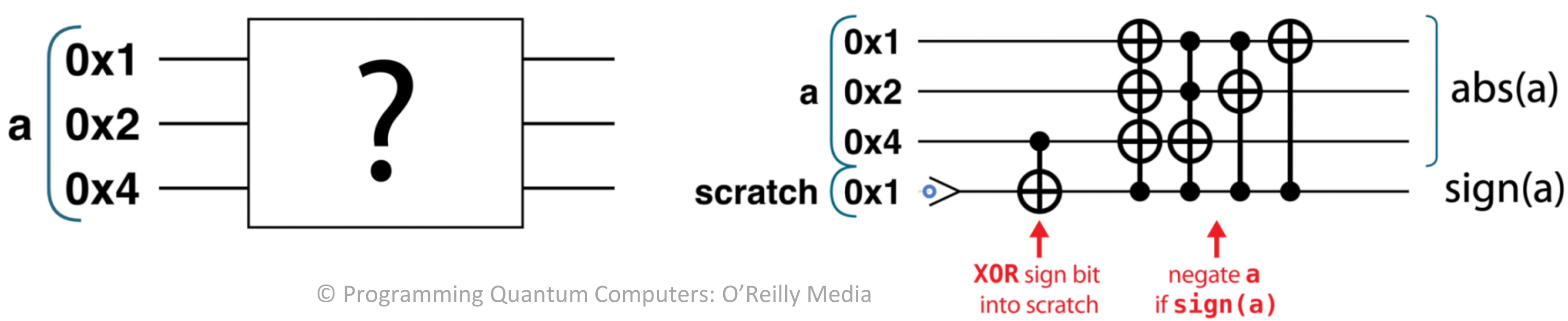
Before conditional phase

After conditional phase

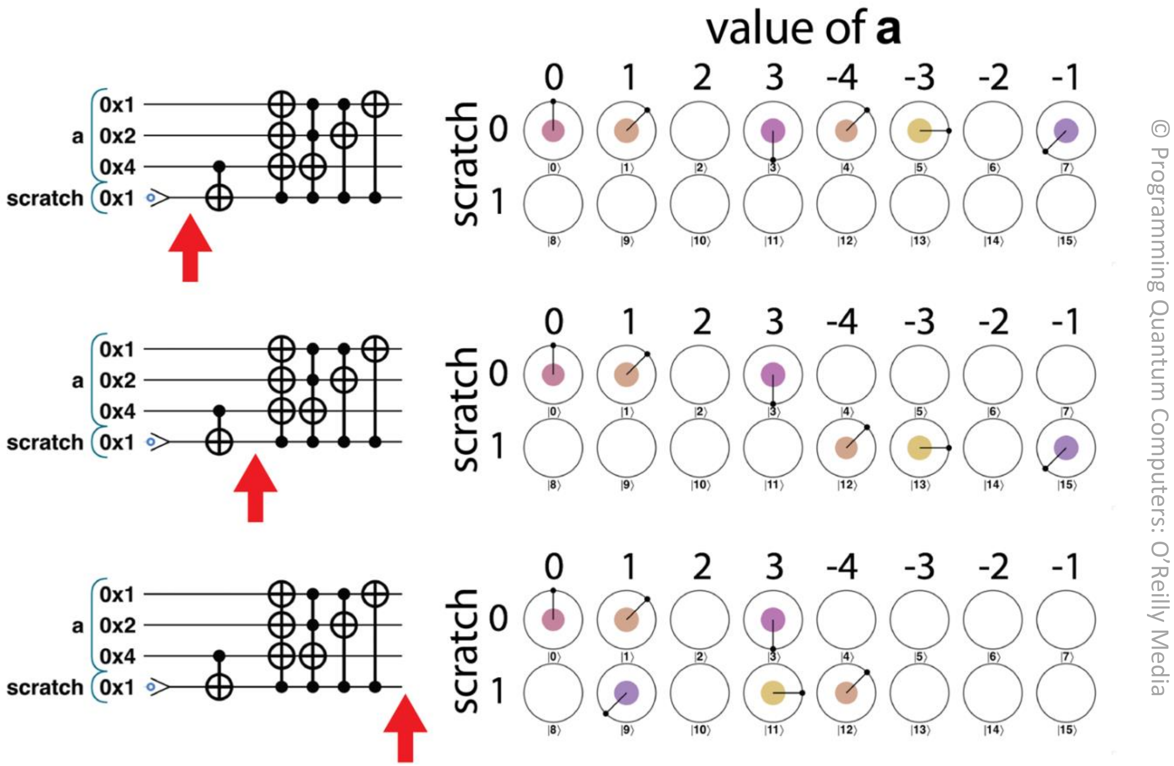
- Hands-on: Different values of *a*, *b* (e.g., with all values in superposition).

Reversibility and Scratch Qubits

- Ensuring operations are reversible.
  - Except for measuring, quantum ops preserve information, i.e., can always be reversed.
  - Not all computable operations are directly reversible.
    - No hard-and-fast way to convert irreversible to reversible.
  - Using scratch qubits, to preserve info into, is usually a helpful workaround.
- Without workarounds, *negate* is reversible while *abs* is irreversible.
  - *abs* (the absolute value function) destroys integer sign info.
  - With one scratch qubit, sign info can be preserved, and *abs* can be computed!
    - Sign info is *entangled* not copied. (Important distinction.)



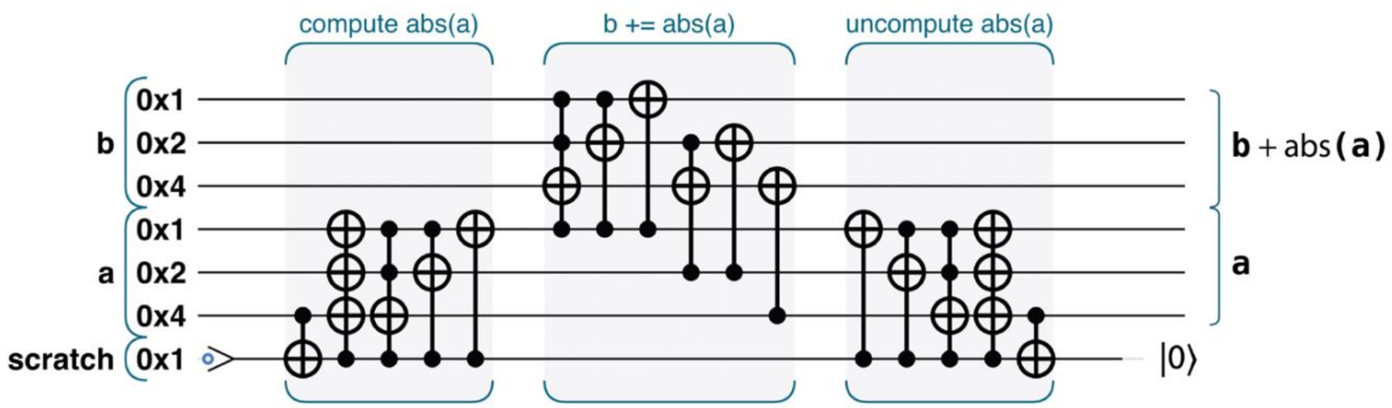
Quantum Absolute Value, Visually



- The scratch qubit gives “room” (an extra row, specifically) to move info into then across, leaving original state info untouched and recoverable.

Uncomputing

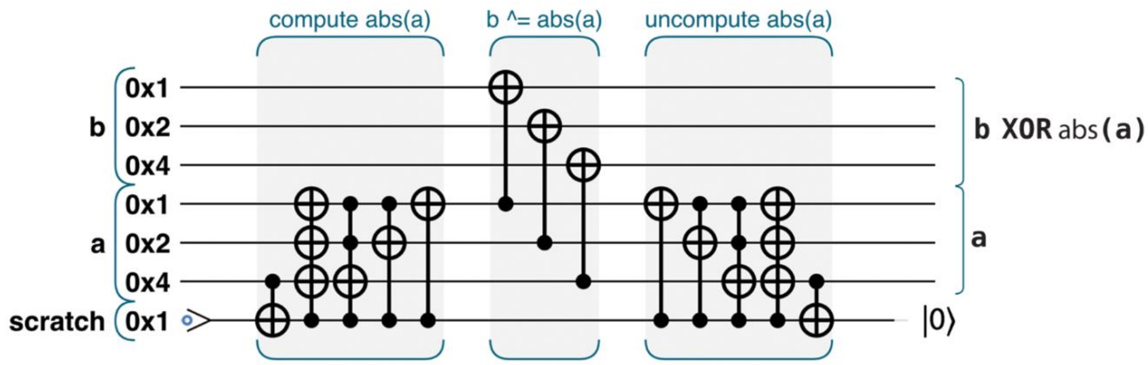
- Reuse of Scratch Qubits.
  - Scratch qubits usually get entangled with result.
    - Directly uncomputing the (irreversible) operation destroys result!
  - Solution: Use result in further computation, then uncompute.



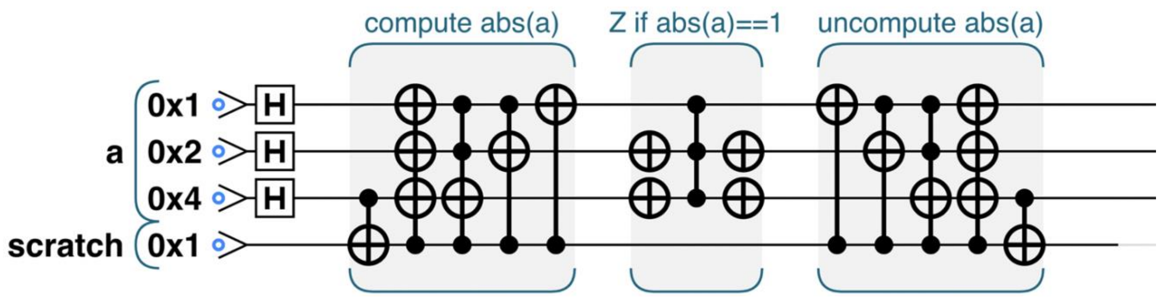
© Programming Quantum Computers: O'Reilly Media

Uncomputing

– Or… perform XOR/CNOT with result.
  - If $b = 0$ then we have a no-cloning workaround ("Copy")!

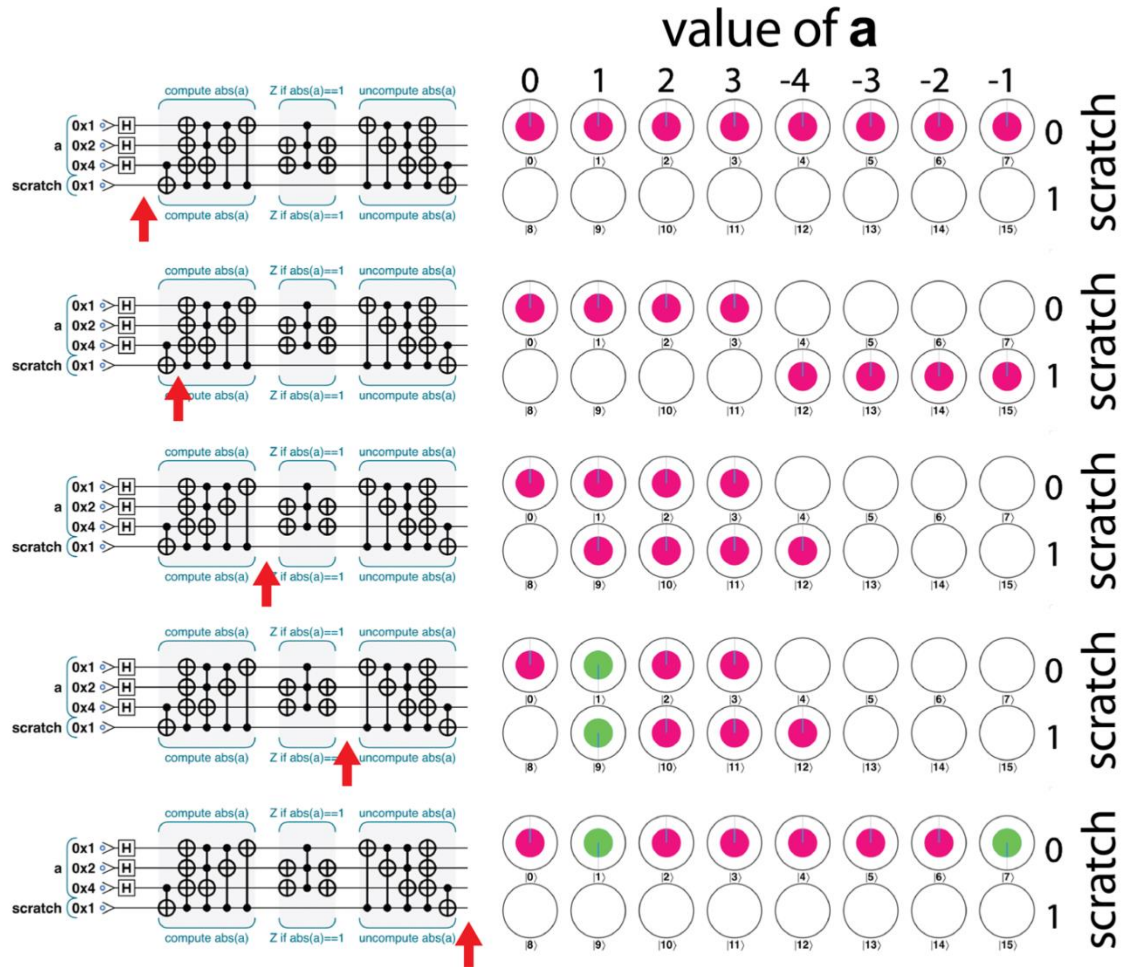


– Or… store result in phase.



Uncomputing (Hands-on)

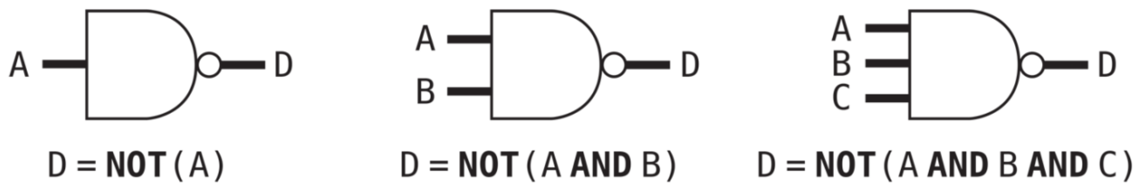
```
qc_options.color_by_phase = true

qc.reset(4)
var a = qint.new(3,'a')
var s = qint.new(1,'s')
qc.write(0)
a.had()

qc.nop()

qc.cnot(8,4)
qc.cnot(7,8)
a.add(1,8)

qc.nop()

a.not(~1)
a.cphase(180)
a.not(~1)

qc.nop()

a.subtract(1,8)
qc.cnot(7,8)
qc.cnot(8,4)

// code above not from textbook
```



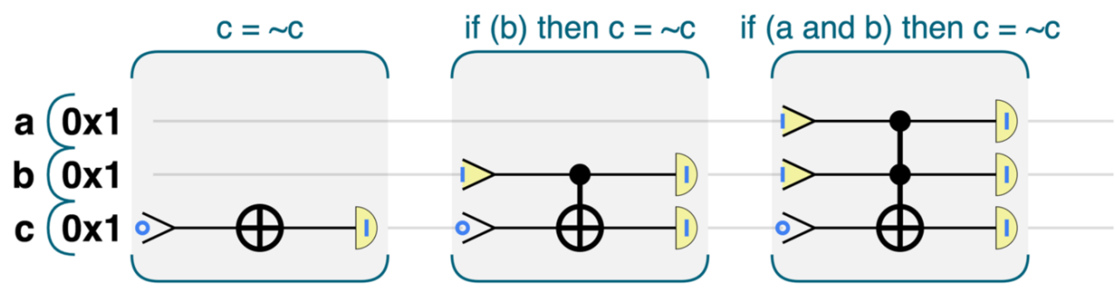
© Programming Quantum Computers: O'Reilly Media

Basic Quantum Boolean Logic

- In classical Boolean logic, NAND is universal.

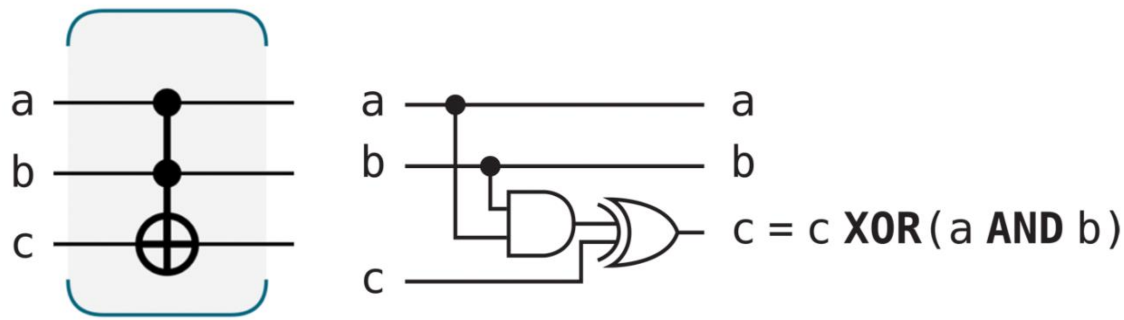


- In quantum Boolean logic, we use Toffoli gate.



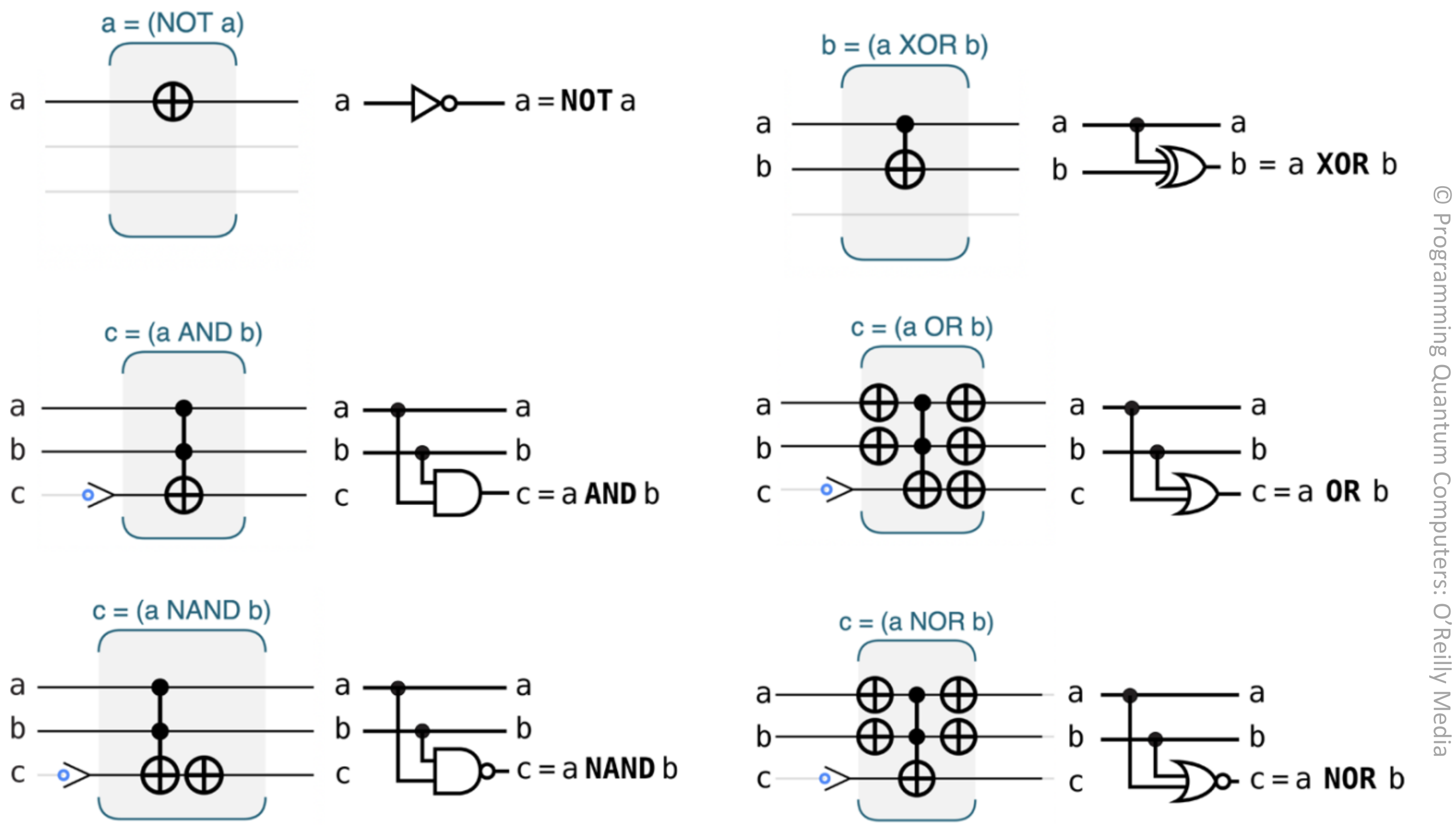
Basic Quantum Boolean Logic

- Classical equivalent of Toffoli (multi-qubit CNOT) gate.



- RM (Reed-Müller) classical circuits
  - Use NOT, AND, and *XOR* gates.

Basic Quantum Boolean Logic



(Note the use of scratch qubits in the last four circuits)

Discussion

Q & A

Lab Assignment

- Open book. 15 minutes.
- Write (or draw) a QCEngine quantum program (or circuit) for each of the following tasks:
  a. Adding 7 to a 4-qubit positive integer variable, say *a*, that's in a superposition of any four non-consecutive integers.
  b. Subtracting -7, encoded in 2's complement, from a 4-qubit integer variable, say *b*, that is in a superposition of four values, encoded in 2's complement.
     - Hint: Don't spend much time on this one (make use of your answer to part (a)).
  c. Adding four 3-qubit quantum variables to the first of them (i.e., computes *a += b + c + d*).
- Homework: Translate programs into QX, Q#, and Cirq programs.
  – Submit code and report (code explanation, in plain English) by email.

Extra Exercise

- Some Group Theory.

$$(abc)^{-1} = c^{-1}b^{-1}a^{-1}$$

  - Without hints, can someone explain the intuitive meaning of this algebraic equation? … and its relation to quantum programming?
  - Does equation hold for conventional programming? Why, or Why Not?
  - What about the following equations: What do they mean? And when may they hold?

$$(abc)^{-1} = cba$$
$$(abc)^{-1} = abc$$
$$(abc)^{-1} = a^{-1}b^{-1}c^{-1}.$$

Next Lecture Appetizer!

- In next lecture:
  - Amplitude Amplification (AA).
    - Converting between phase (invisible info) and magnitude (visible info).
    - The `mirror` module, the core of Grover’s quantum search algorithm: How it works, and why (“Slingshotting!”).

Course Webpage

http://eng.staff.alexu.edu.eg/staff/moez/teaching/pqc-f19

- Where you can:
  - Download lecture slides (incl. exercises and homework).
  - Check links to other relevant material.

Thank You