Programming Quantum Computers (Modules IV: PE)

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

Moez A. AbdelGawad

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

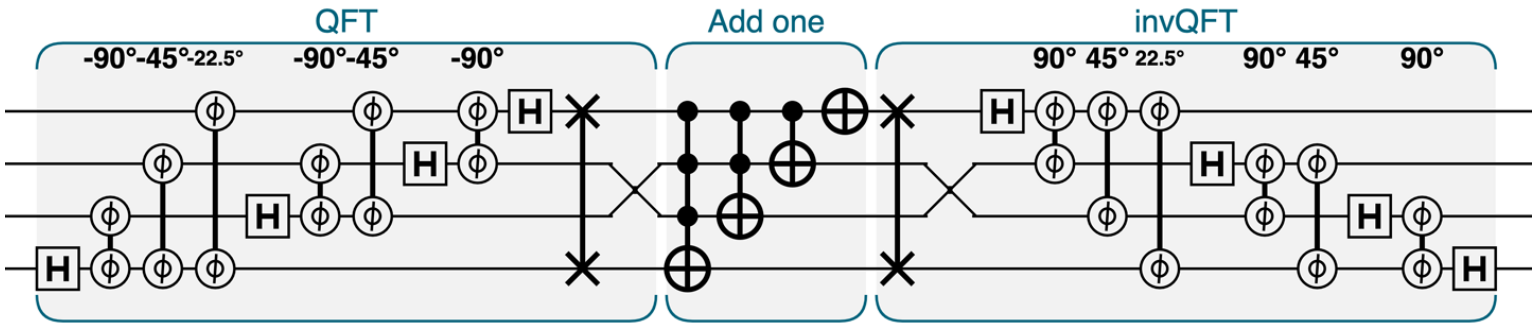
Sat., Dec. 7th, 2019

Quantum Computers are Real

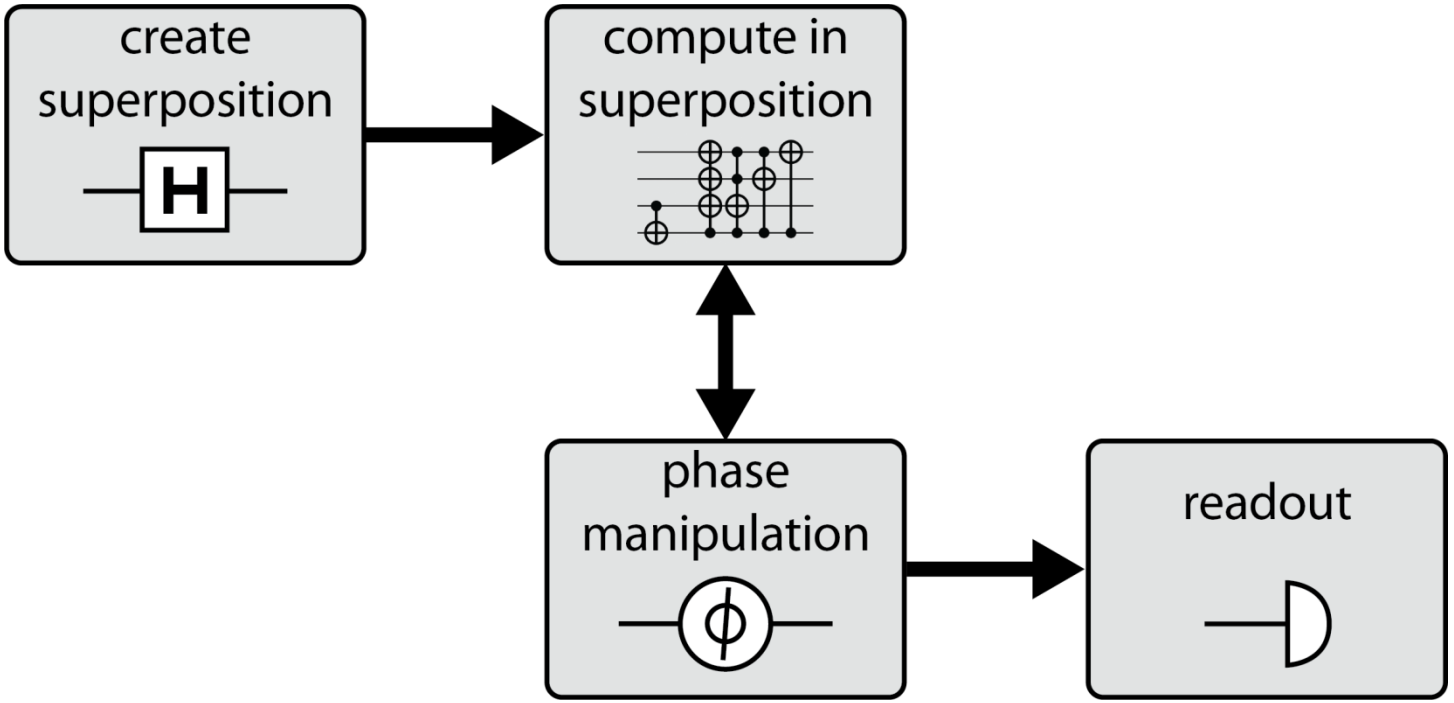
- What are they <u>useful</u> for?
  - Let's discover, by programming them!

- 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.

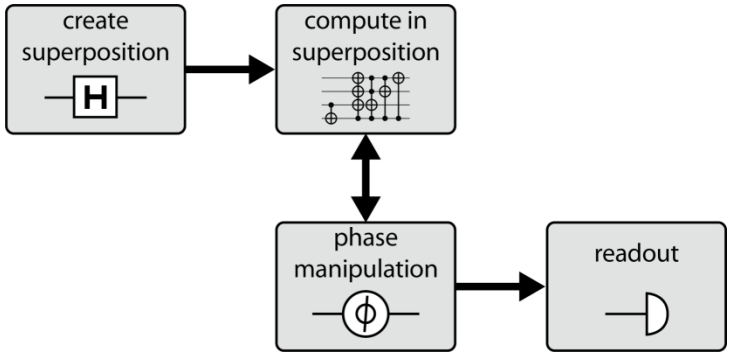


Structure of Quantum Apps



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*.
- Quantum programming is an art (too).



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

PHASE MANIPULATION MODULES

PHASE ESTIMATION

Lecture Outline

- What is (eigen)phase estimation (PE)?
- Eigenstates and eigenphases of quantum operations
  - E.g., of H, NOT and RotY/RotX.
- Using phase estimation.
- PE constraints and PE in practice.
- Inside the QPU.
  - PE: Intuition.
  - PE: Operation by operation.
- Note on Textbook (PQC): Superb very simple explanation of PE in Ch.8 (PE), but, compared to earlier chapters, Ch.8 has many *very annoying* mistakes and typos.
  - Checking the textbook errata is particularly useful while reading Ch.8.
  - Debugging, using the authors' own QCEngine, becomes a necessity, and is also very useful.

What is Phase Estimation?

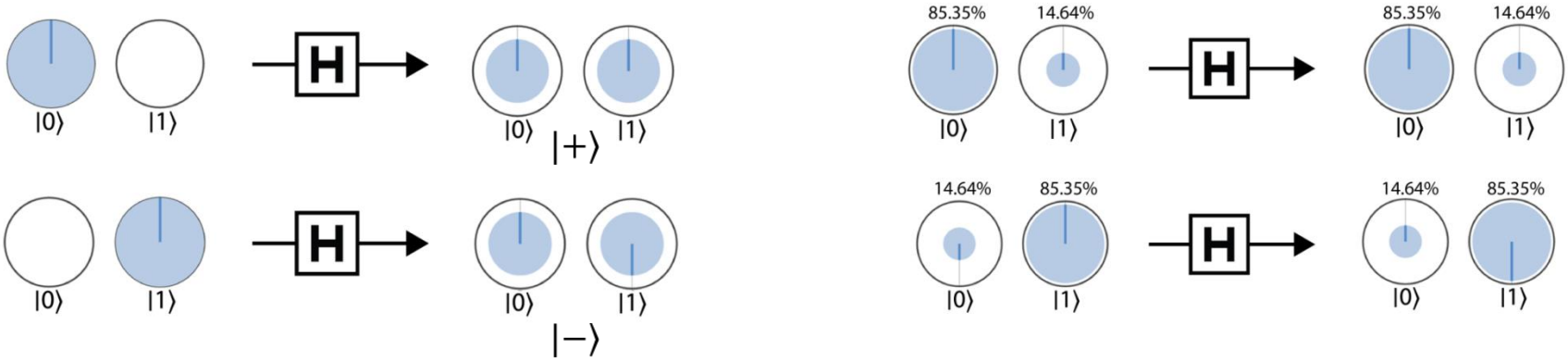
- Phase estimation (PE) tells an attribute of a quantum *operation*.
  - Unlike QFT and AA, PE does not produce a property of a quantum state .

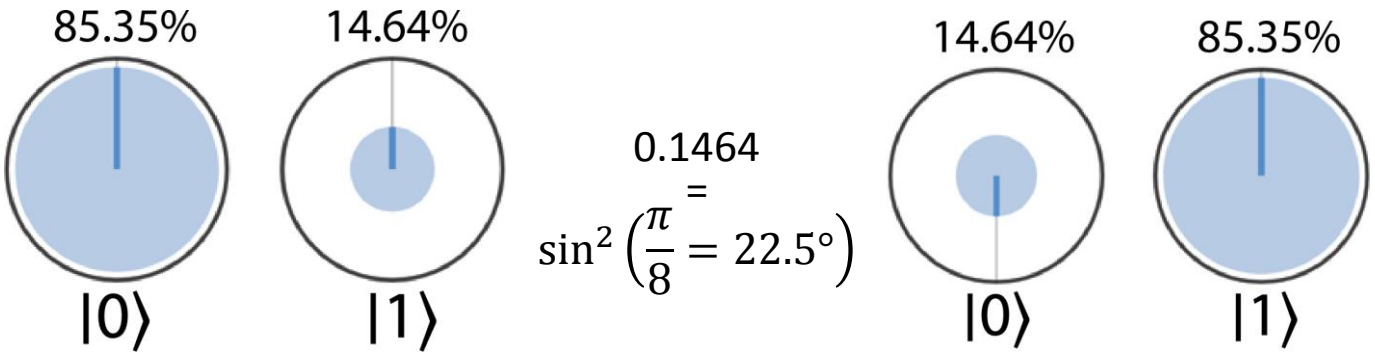
- Eigenstates and Eigenphases:
  - Matrices can represent quantum operations.
  - From linear algebra, matrices can have eigenvectors and eigenvalues.

$$(A - \lambda I)v = 0.$$

  - Eigenvectors correspond to *fixed* states, unchanged by the operation the matrix represents; they are called *eigenstates* of the quantum operation.
  - Eigenvalues correspond to global phases of the eigenstates; called *eigenphases*.
  - The set of eigenstates and eigenphases *uniquely characterizes* a quantum operation.

Eigenstates and Eigenphases … of H





(QQ: $|+\rangle$ and $|-\rangle$ are eigenstates — "fixed points" — of which operation?)

Phase Estimation

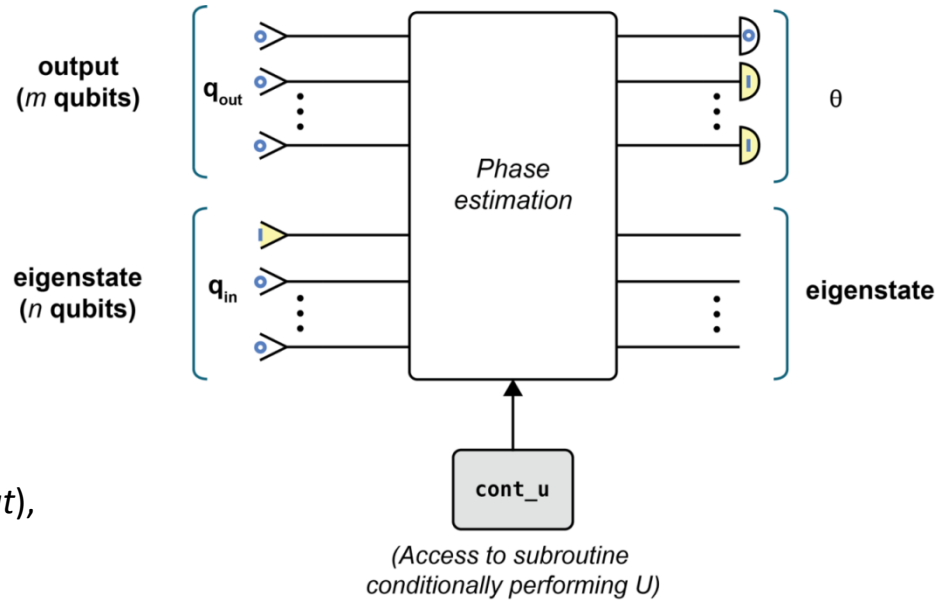
- Helps determine the eigenphases of eigenstates of a QPU operation, returning a superposition of all eigenphases.

- For a quantum operation *U*, phase estimation produces an estimate or an approximation of the eigenphase θ_j (or, a superposition of eigenphases θ_i) of *U* given the eigenstate u_j (or, a superposition of eigenstates u_i) of *U*.

- Despite being hugely important in many algorithms, PE initially *appears* to be useless and arbitrary.
  - Revealing its practical use needs resorting to some relatively advanced mathematics.
  - Precisely, eigenstates and eigenphases are the *eigenvectors* and *complex eigenphases* of the unitary matrices representing QPU operations (in the full mathematics of quantum computing).
  - Sounds important! … Used in QML applications (Ch.13).

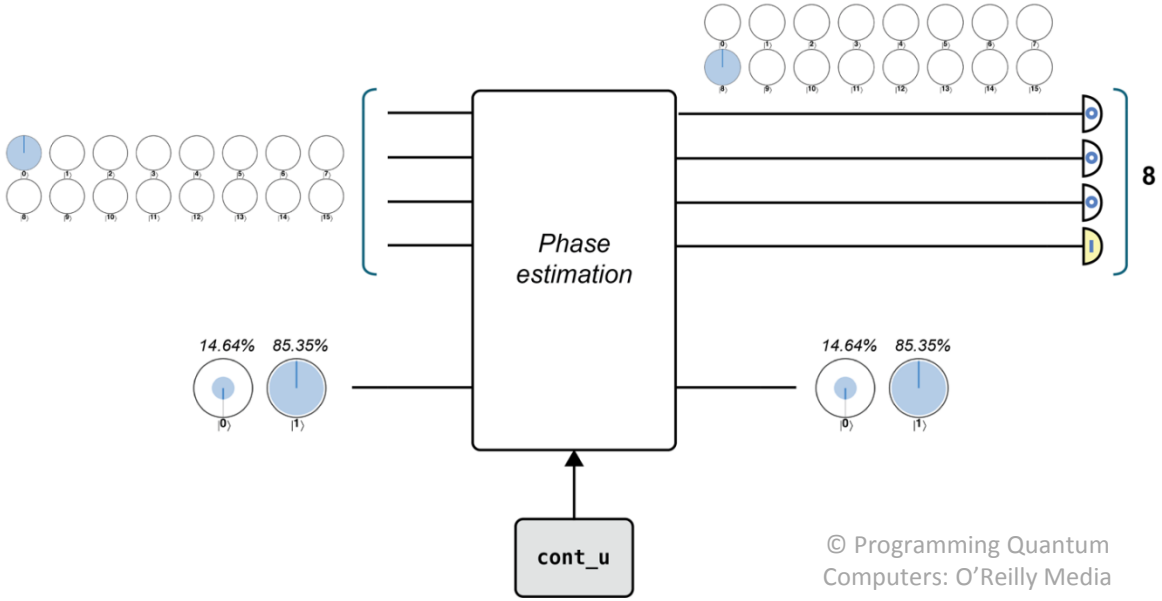
Using Phase Estimation

- PE signature:
  phase_est(q_{in}, q_{out}, *cont_u*)
  - q_{in} encodes u_j.
  - q_{out} is initialized to $|0\rangle$.
    - Will encode θ_j.
    - Larger q_{out} means greater precision in θ_j.
  - *cont_u is* a *controlled* version of *U*.
    - Provided in the form *cont_u*(*in*, *out*), where *in* is a single control qubit.



Using Phase Estimation: Concrete Example (H)

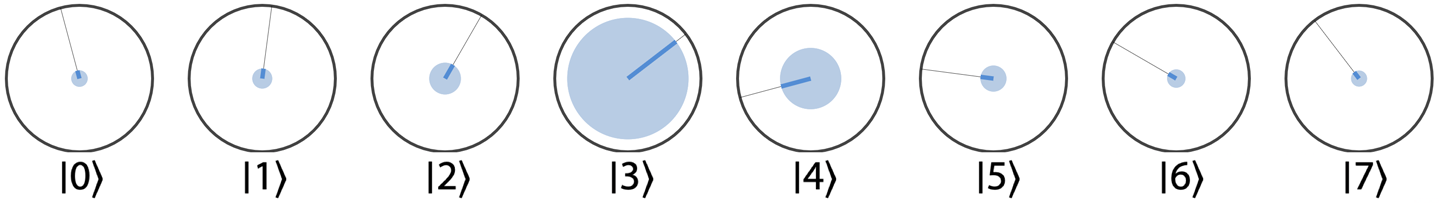
- *cont_u* is `chadamard`, q_{in} is the special eigenstate of H with eigenphase $180°$, and q_{out} is $|0\rangle$.
  - Expected output 180? … Output best read as a *fraction* (of 2^m).
  - Eigenphase: $\theta_j = \frac{q_{out}}{2^m} \times 360°$.
  - E.g., $180° = \frac{8}{16} \times 360°$.



© Programming Quantum Computers: O'Reilly Media

Using Phase Estimation: The Fine Print

- Choosing output size m (eigenphase precision).
  - Hands-on? Eigenphase $150°$ with 4-qubits, specified *imprecisely* in q_{out}.
    - Code not present in textbook (Ex. 8-2, as of Nov. 26th, 2019. Check textbook errata.)



$$m = p + \left\lceil \log\left(2 + \frac{1}{\epsilon}\right) \right\rceil$$

m: size of output register
p: required bits of accuracy
ϵ: max. probability of error
$(= 2\epsilon$ in "Mike&Ike")

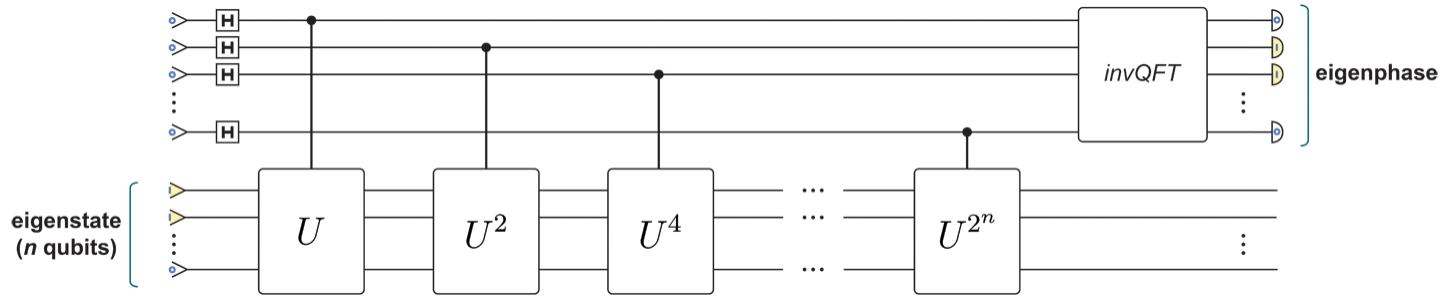
Using Phase Estimation: The Fine Print

- Complexity:
  - For more precision more operations are needed.
  - $O(m^2)$, due to dependency on *invQFT*.
  - Performance of *cont_u* matters significantly.
- Conditional *U*:
  - *cont_u* is called *multiple* times inside PE (more on this later).
  - For some *U* it is difficult to find *efficient* controlled versions.
- In practice:
  - Superposition of eigenstates produces superposition of eigenphases.
  - Fact: For *each* op, *any* qstate is a superposition of the op's eigenstates!
  - Useful in math apps involving linear algebra (superposition of eigenphases info raises app quantum parallelization possibility).

Phase Estimation: Implementation

- `phase_est` is very concise!

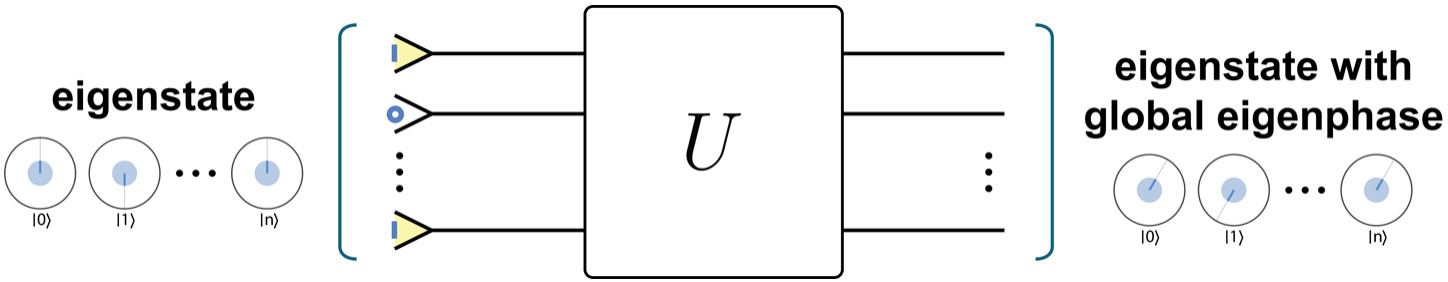
```
function phase_est(q_in, q_out, cont_u)
{
    q_out.had();

    // Apply conditional powers of u
    for (var j = 0; j < q_out.numBits; j++)
        cont_u(q_out, q_in, 1 << j); // 1 << j = 2^j

    q_out.invQFT();
}
```



© Programming Quantum Computers: O'Reilly Media

PE Implementation: Intuition

- Suggested implementation:

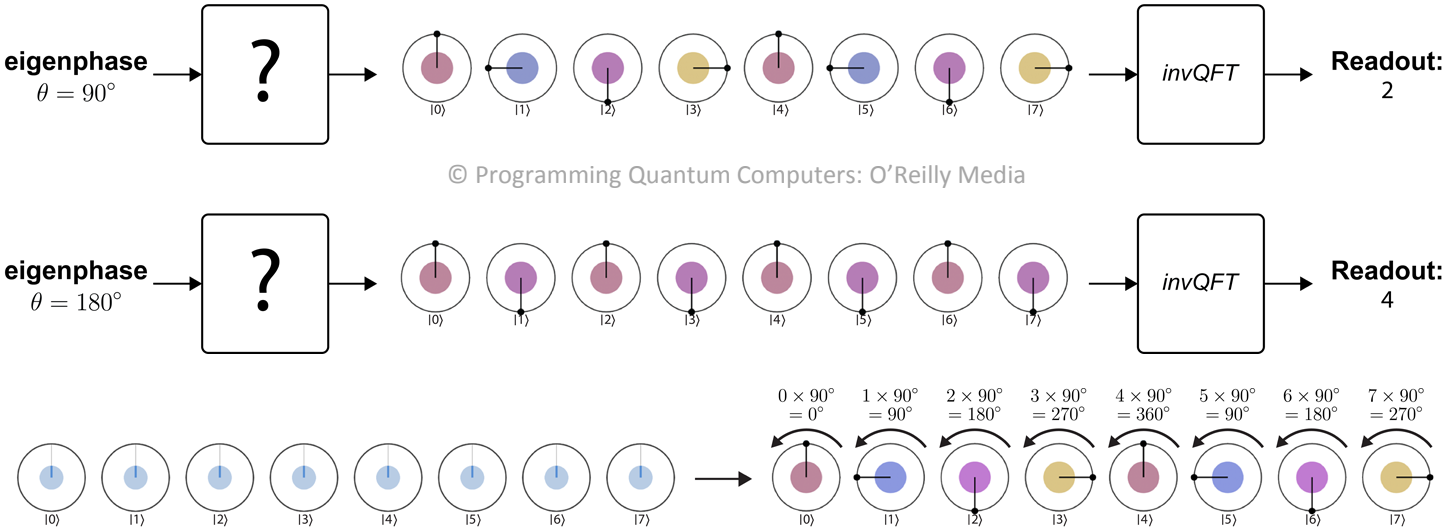


  - Output register is the eigenstate of *U* (i.e., the same as input), but with eigenphase applied as a global phase.

- Problem:
  - *Global* phase cannot be READ out (not by AA, not even by QFT/invQFT).
  - All-too-familiar problem in quantum programming: Info we want is trapped inside the (relative or global) phases of a QPU register.

PE Implementation: Intuition

- Towards a solution:
  - Desired behavior. What may replace the '?' below?



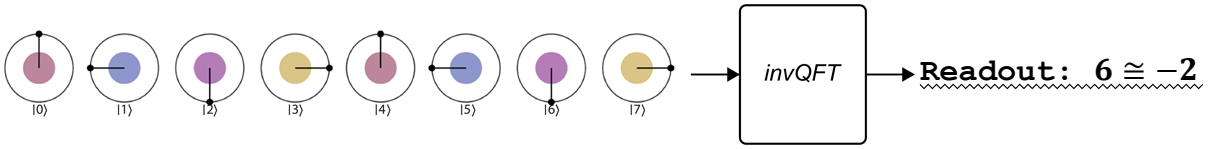
(Similar to idea behind QFT/invQFT)

- Distraction: A series of textbook mistakes starts here!
  - (Hands-on) Via debugging and other readings, I tried to fix (most of) those mistakes, in these slides and in the textbook's errata.

Book Erratum

- Code illustrating error in Fig. 8-8.

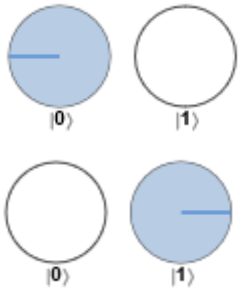
```
qc.reset(3);

r=qint.new(3,'reg');
r.write(2);

qc.label('gen signal')
   r.had(4);
   r.cphase(90,4,2);
   r.cphase(45,4,1);

   r.had(2);
   r.cphase(90,2,1);

   r.had(1);

   qc.swap(0x1|0x4);
qc.label('')

qc.nop()
qc.invQFT();
```

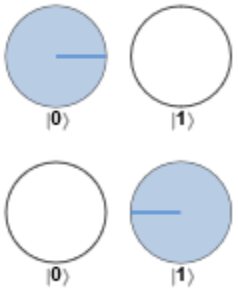


Via some educated guesses we can discover that:

```
// |0> is an eigenstate
// with eigenphase -90.
// |1> with 90.
qc.reset(1)
qc.write(0) // or 1
qc.roty(180)
qc.rotx(180)
```

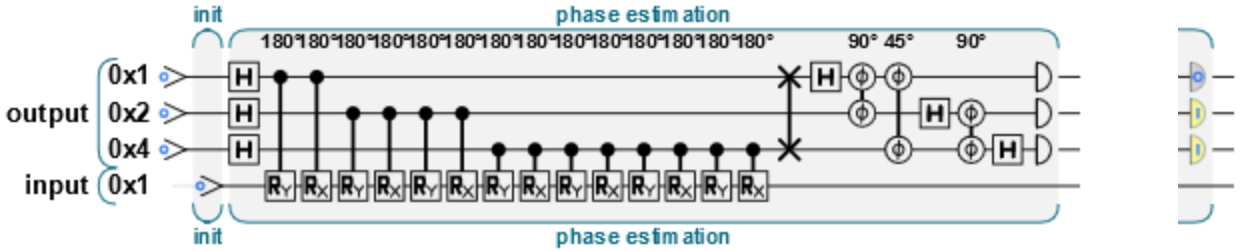
```
// |0> is an eigenstate
// with eigenphase 90
// |1> with -90.
qc.reset(1)
qc.write(0) // or 1
qc.rotx(180)
qc.roty(180)
```

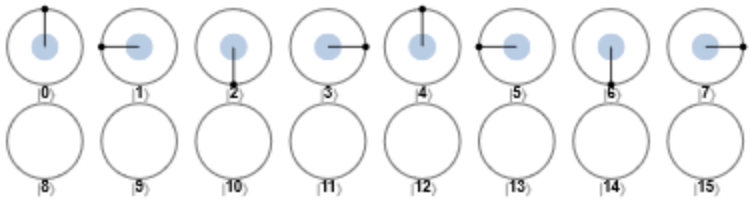


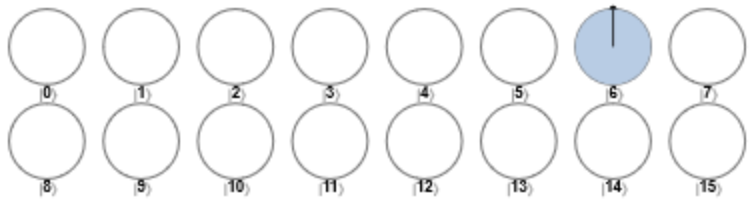


PE: Another Concrete Example (RotY+RotX)

```
// Initialize input register as eigenstate
qin.write(0);
// Define conditional unitary
function cont_u(qtarget, qcontrol, control_count) {
    //iter = control_count % 4
    for(i=0; i < control_count /*iter*/; i++){
        qc.roty(180, qtarget, qcontrol.bits(control_count));
        qc.rotx(180, qtarget, qcontrol.bits(control_count));
    }
}
```

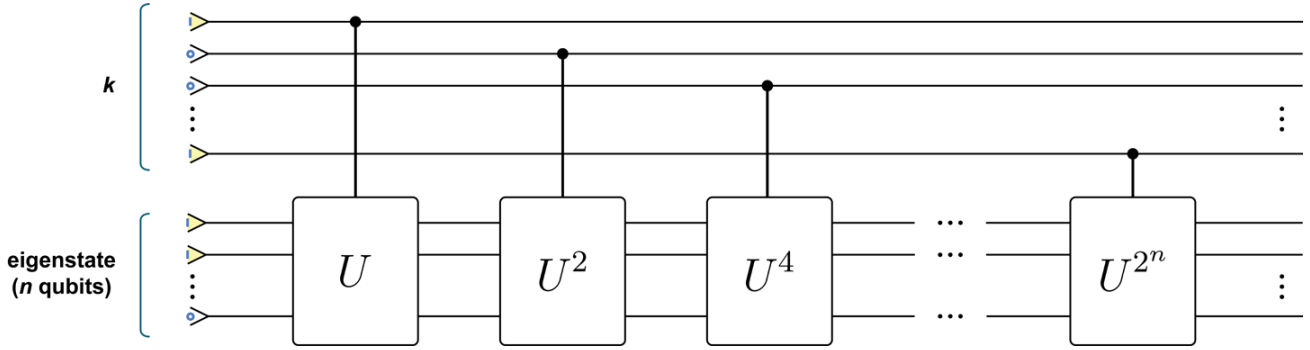




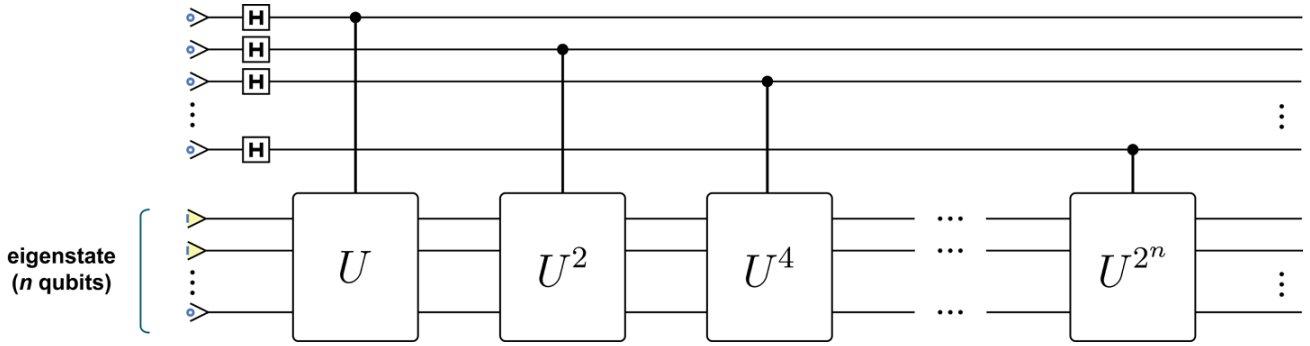


PE Implementation: Intuition

- Phase kickback, for a particular value *k*.



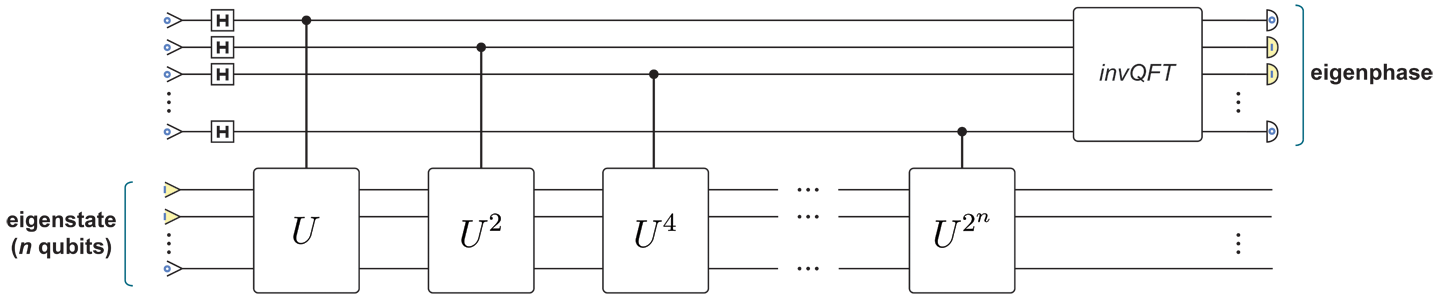
- For *all* values, in *uniform superposition*.



© Programming Quantum Computers: O'Reilly Media

PE Implementation: Intuition

- Phase kickback pushes *global* phase rotations back into (negative) *relative* phases of control qubits.
  - Use *invQFT* to read frequency info!
  - Why *invQFT* not *QFT*??
    - Guess: Because $QFT(-s) = QFT^{-1}(s)$.
    - Textbook dodges question by using an example operation with eigenphases $0°$ and $180°$.
      - Since, effectively, $180° \cong -180°$, and, in 2's complement, -(-4) ≅ -4 ≅ 4. Same for $0°$ and 0.



- PE: Very smart!
  - Development of ideas, from initial suggested implementation to final one.
  - Clear why *controlled U* is needed? and how size of output limits precision?
- Efficiency of computing U^{2^j} significantly affects the efficiency of PE.

Textbook Errata

- Many errata in Ch.8. For example:
  - Missing code.
    - For eigenphase 150° example.
  - Mistakes in figures.
    - Inaccurate output in Fig. 8-8.
  - Mistakes in text.
    - Using ‘state’ instead of ‘value’, right before Fig. 8-9.
  - Mistakes and lack of clarity in explanation of intuition behind PE implementation.
    - Why is *invQFT* used rather than *QFT*.
    - Implicit or non-crystal-clear references to QPU registers, before Figs. 8-11 and 8-12.
    - Signature of `cont_u` used in code is different from the one specified in text.
    - Operation of `chadamard` and meaning of its first two parameters are not explained anywhere.

Discussion

Q & A

Research Question

- Now that we know precisely how PE works, can we suggest another operation-attributes revealing module?
  - i.e., a module that reveals the same (or different) attributes using, e.g., different operations, and/or that produces better or just different results.
  - If you think it is possible, suggest how.
  - If you think it is not possible, explain why not.

Homework

- Implement PE on QX, Q#, and Cirq.

Next Lecture Appetizer

- In next lecture (isA):
  - Real quantum data, and QRAM.
    - More complex data structures.
    - Representing vectors.
    - Representing matrices (*quantum simulation*).

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 useful material.

Thank You