浅谈量子计算与编程

OSDT 2017

邢明杰

2017-10-21

量子计算

“Changes occurring to a quantum state can be described using the languange of quantum computation.”

– Michael A. Nielsen and Issac L. Chuang
Quantum Computation and Quantum Information

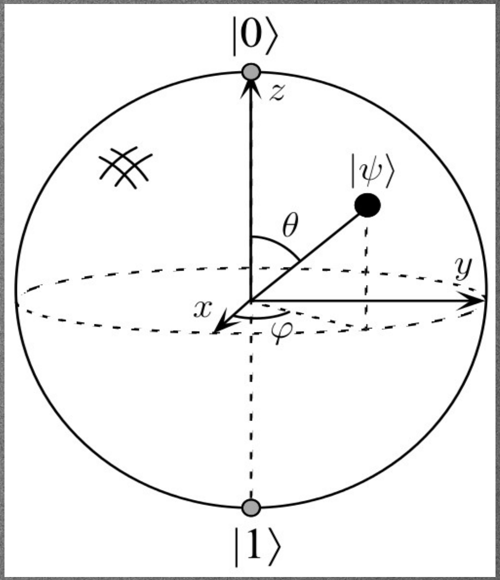
量子比特

- 1 个传统 bit 具有的状态为
  - 0 或 1
- 1 个 qubit 具有的状态为
  - 计算基态 $|0\rangle$ 和 $|1\rangle$ 的线性叠加

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

其中 α 和 β 为复数，且

$$|\alpha|^2 + |\beta|^2 = 1$$



布洛赫球面 (Bloch sphere)

量子比特

- 2 个 qubit
  - 有 4 个计算基态 $|00\rangle, |01\rangle, |10\rangle, |11\rangle$

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

- n 个 qubit
  - 有 2^n 个计算基态
  - 由 2^n 个系数组成一维状态向量

$$\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

测量

- 导致状态坍塌 不可逆操作
  - 量子力学的基本假设
- 结果是概率性的
  - 1 个 qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

    $|\alpha|^2$ 的可能性：测量结果为 0 ，测量后的状态坍塌成 $|0\rangle$
    $|\beta|^2$ 的可能性：测量结果为 1 ，测量后的状态坍塌成 $|1\rangle$
  - 2 个 qubit $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$

    测量第一个 qubit 的时候，$|\alpha_{00}|^2 + |\alpha_{01}|^2$ 的可能性：测量结果为 0 ，测量后的状态坍塌成：

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

量子门

- 量子计算机，逻辑上有量子电路和量子门组成
- 量子状态转换 <=> 酉变换：酉矩阵 × 状态向量
  - 酉矩阵 (unitary) ： $U^\dagger U = I$
  - 伴随矩阵 (adjoint) ：转置，然后复数共轭

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

量子门

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Pauli-X

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Pauli-Z

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Hadamard

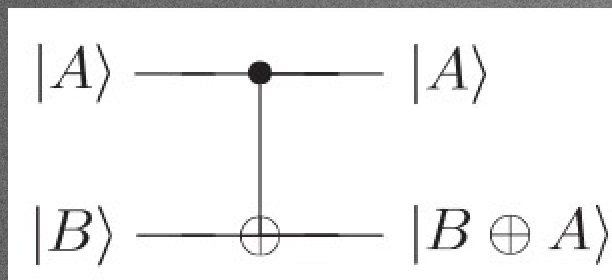
$$\alpha\left|0\right\rangle + \beta\left|1\right\rangle \;—\boxed{X}—\; \beta\left|0\right\rangle + \alpha\left|1\right\rangle$$

$$\alpha\left|0\right\rangle + \beta\left|1\right\rangle \;—\boxed{Z}—\; \alpha\left|0\right\rangle - \beta\left|1\right\rangle$$

$$\alpha\left|0\right\rangle + \beta\left|1\right\rangle \;—\boxed{H}—\; \alpha\,\frac{\left|0\right\rangle+\left|1\right\rangle}{\sqrt{2}} + \beta\,\frac{\left|0\right\rangle-\left|1\right\rangle}{\sqrt{2}}$$

1 个 qubit 的量子门（无限多种）

量子门



A 是 control qubit
B 是 target qubit

$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

CNOT

$$|00\rangle \rightarrow |00\rangle; \ |01\rangle \rightarrow |01\rangle; \ |10\rangle \rightarrow |11\rangle; \ |11\rangle \rightarrow |10\rangle$$

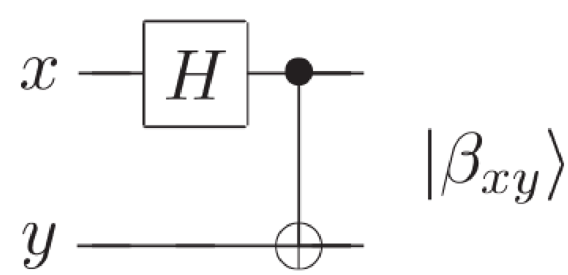
2 个 qubit 的量子门

（可用有限集合的量子门来生成任意个数 qubit 的量子计算）

量子纠缠

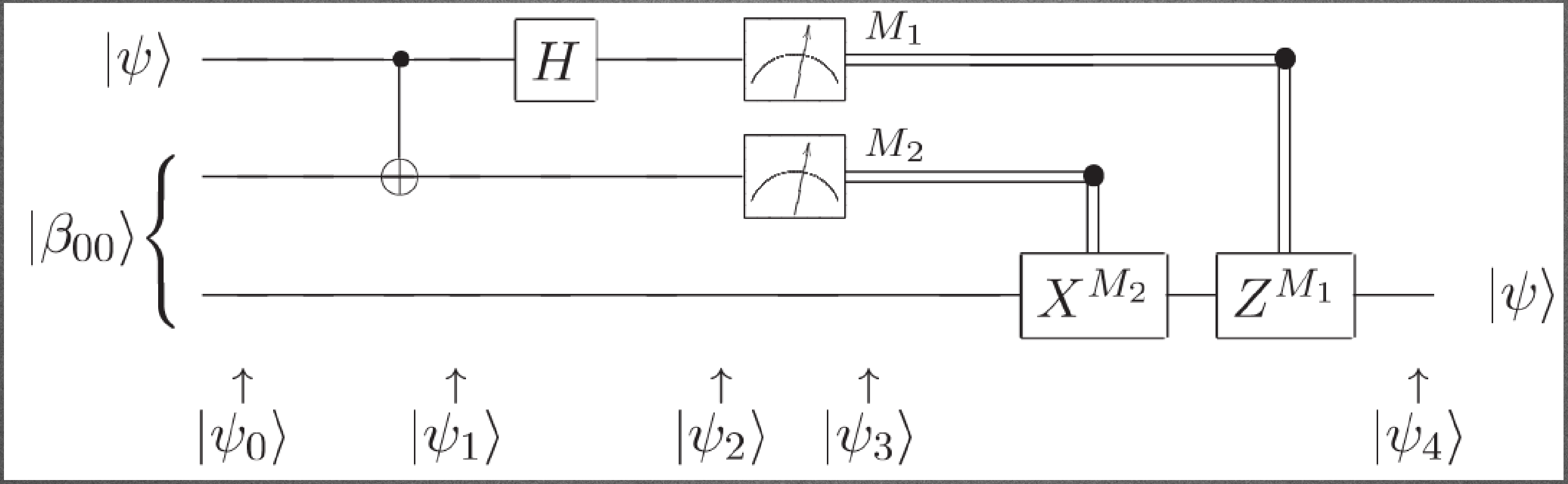
- Bell/EPR 状态
  - 随机性：对于 $|\beta_{00}\rangle$ 测量第 1 个 qubit ， 1/2 的可能性为 0 ，测量后的状态为 $|00\rangle$ ， 1/2 的可能性为 1 ，测量后的状态为 $|11\rangle$
  - 关联性：测量第 2 个 qubit 的状态，结果总是跟第 1 个相同

In	Out
$\|00\rangle$	$(\|00\rangle + \|11\rangle)/\sqrt{2} \equiv \|\beta_{00}\rangle$
$\|01\rangle$	$(\|01\rangle + \|10\rangle)/\sqrt{2} \equiv \|\beta_{01}\rangle$
$\|10\rangle$	$(\|00\rangle - \|11\rangle)/\sqrt{2} \equiv \|\beta_{10}\rangle$
$\|11\rangle$	$(\|01\rangle - \|10\rangle)/\sqrt{2} \equiv \|\beta_{11}\rangle$



量子传输

- Alice 和 Bob 各有 1 个 Bell 状态的 qubit ，现在， Bob 隐藏了， Alice 的任务是，只通过发送传统信息的方式，将 |ψ⟩ 的状态传输给 Bob
- Bob 根据收到的信息，能够将自己的 qubit 修正为 |ψ⟩ 的状态



抽象的计算过程参见原书

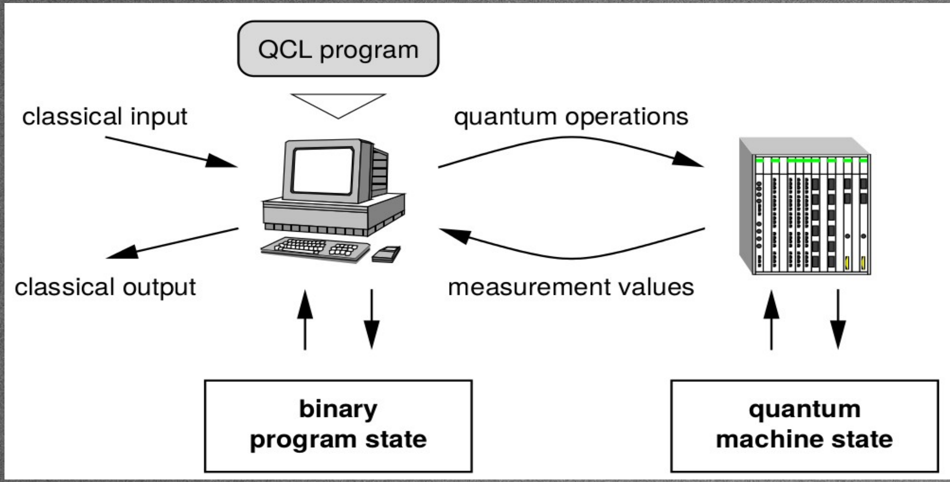
量子编程

QCL　QASM　LIQUi|⟩

QCL

- 早期代表性工作
  - http://tph.tuwien.ac.at/~oemer/qcl.html
  - B. Ömer. A procedural formalism for quantum computing. Master’sthesis, Theoretical University of Vienna, 1998
  - B. Ömer. Structured Quantum Programming. PhD thesis, TheoreticalUniversity of Vienna, 2003.
- 命令式语言 + 解析器（功能模拟）
- 开源 (GPL2) ！

QCL



QCL 体系结构模型

routine type	program state	machine state	recursion
procedure	all	all	yes
operator	none	unitary	no
qufunct	none	pseudo-classical	no
functions	none	none	yes

QCL 子程序类型，及对传统程序状态和量子机器状态的影响

QCL

```
// pseudo classic operator to swap bit order

cond qufunct flip(qureg q) {
  int i;                  // declare loop counter
  for i=0 to #q/2-1 {     // swap 2 symmetric bits
    Swap(q[i],q[#q-i-1]);
  }
}

// discrete Fourier transform (Coppersmith)

operator dft(qureg q) { // main operator
  const n=#q;             // set n to length of input
  int i; int j;           // declare loop counters
  for i=1 to n {
    for j=1 to i-1 {      // apply conditional phase gates
      V(pi/2^(i-j),q[n-i] & q[n-j]);
//      if q[n-i] and q[n-j] { Phase(pi/2^(i-j)); }
    }
    H(q[n-i]);            // qubit rotation
  }
  flip(q);                // swap bit order of the output
}
```

库函数

```
procedure testdft() {
  qureg q[6];
  print "testing discrete Fourier transform";
  reset;
  dft(q);
  CPhase(pi/2,q[1]);
  !dft(q);
  print "expecting (1+i)/2 |000000> - i/2 |010000> + 1/2 |110000>";
  dump;
  reset;
}

testdft();
```

测试程序

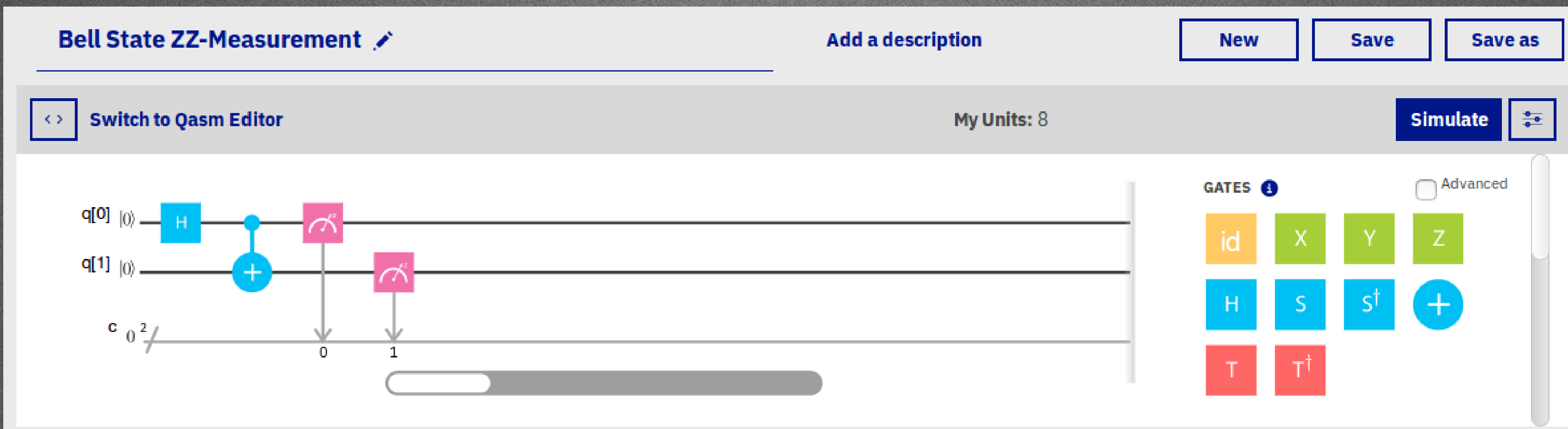
```
xmj@xmj-OptiPlex-9020:~/qc/qcl-0.6.4$ ./qcl lib/test.qcl
QCL Quantum Computation Language (64 qubits, seed 1508050089)
: testing discrete Fourier transform
: expecting (1+i)/2 |000000> - i/2 |010000> + 1/2 |110000>
: STATE: 6 / 64 qubits allocated, 58 / 64 qubits free
(0.5+0.5i) |0> - 0.5i |16> + 0.5 |48>
```

运行结果

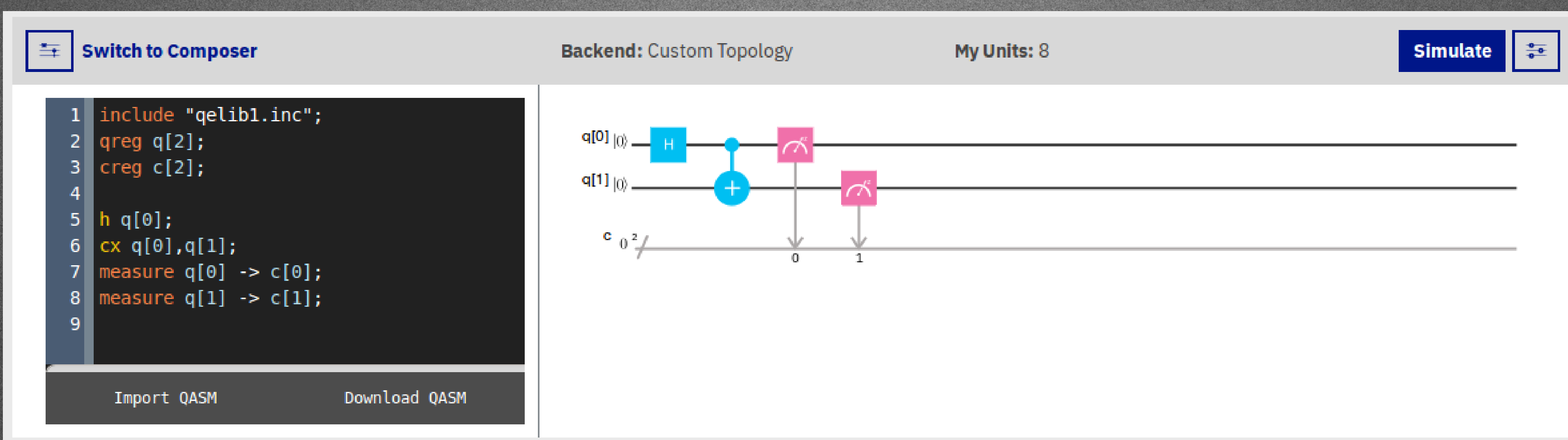
IBM Q

- 在线实验平台
  - https://www.research.ibm.com/ibm-q
  - 可在线编写量子程序
  - 可在模拟器或者量子计算机上运行，并图形化显示结果
- QASM 汇编语言规范
  - https://github.com/QISKit/openqasm
- Composer 图形化编辑器
- 使用教程
  - https://quantumexperience.ng.bluemix.net/qx/tutorial

IBM Q



Bell State ZZ-Measurement
Add a description
New
Save
Save as
Switch to Qasm Editor
My Units: 8
Simulate
q[0] |0⟩
q[1] |0⟩
c 0 2
0
1
GATES
Advanced
id
X
Y
Z
H
S
S†
T
T†



Switch to Composer
Backend: Custom Topology
My Units: 8
Simulate
include "qelib1.inc";
qreg q[2];
creg c[2];
h q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
Import QASM
Download QASM
q[0] |0⟩
q[1] |0⟩
c 0 2
0
1

QASM

Table 1: Open QASM language statements (version 2.0)

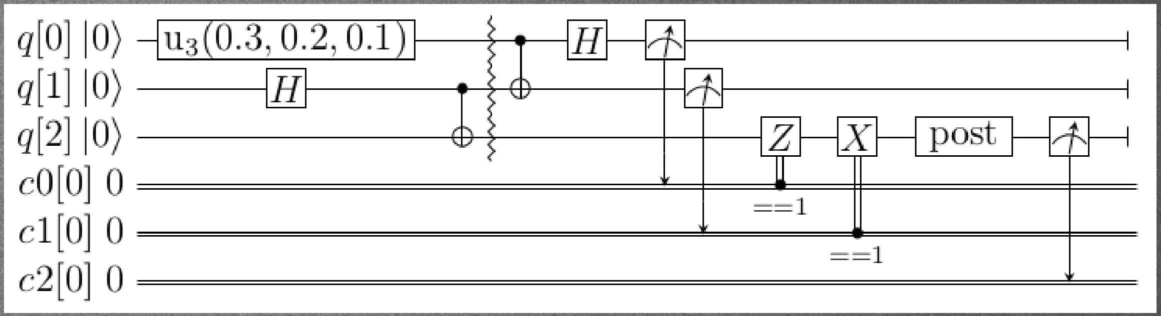
Statement	Description	Example
`OpenQASM 2.0;`	Denotes a file in Open QASM format[a]	`OpenQASM 2.0;`
`qreg name[size];`	Declare a named register of qubits	`qreg q[5];`
`creg name[size];`	Declare a named register of bits	`creg c[5];`
`include "filename";`	Open and parse another source file	`include "qelib1.inc";`
`gate name(params) qargs { body }`	Declare a unitary gate	(see text)
`opaque name(params) qargs;`	Declare an opaque gate	(see text)
`// comment text`	Comment a line of text	`// oops!`
`U(theta,phi,lambda) qubit|qreg;`	Apply built-in single qubit gate(s)[b]	`U(pi/2,2*pi/3,0) q[0];`
`CX qubit|qreg,qubit|qreg;`	Apply built-in CNOT gate(s)	`CX q[0],q[1];`
`measure qubit|qreg -> bit|creg;`	Make measurement(s) in Z basis	`measure q -> c;`
`reset qubit|qreg;`	Prepare qubit(s) in $\|0\rangle$	`reset q[0];`
`gatename(params) qargs;`	Apply a user-defined unitary gate	`crz(pi/2) q[1],q[0];`
`if(creg==int) qop;`	Conditionally apply quantum operation	`if(c==5) CX q[0],q[1];`
`barrier qargs;`	Prevent transformations across this source line	`barrier q[0],q[1];`

1-qubit 量子门：

$$U(\theta,\phi,\lambda) := R_z(\phi)R_y(\theta)R_z(\lambda) = \begin{pmatrix} e^{-i(\phi+\lambda)/2}\cos(\theta/2) & -e^{-i(\phi-\lambda)/2}\sin(\theta/2) \\ e^{i(\phi-\lambda)/2}\sin(\theta/2) & e^{i(\phi+\lambda)/2}\cos(\theta/2) \end{pmatrix}$$

QASM

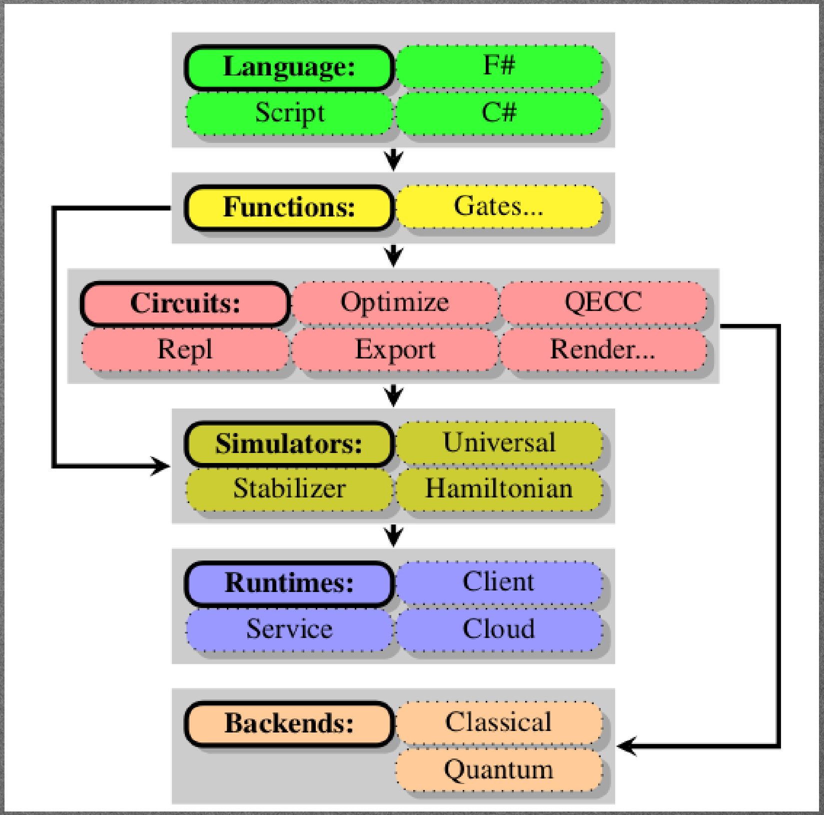
```
// quantum teleportation example
OPENQASM 2.0;
include "qelib1.inc";
qreg q[3];
creg c0[1];
creg c1[1];
creg c2[1];
// optional post-rotation for state tomography
gate post q { }
u3(0.3,0.2,0.1) q[0];
h q[1];
cx q[1],q[2];
barrier q;
cx q[0],q[1];
h q[0];
measure q[0] -> c0[0];
measure q[1] -> c1[0];
if(c0==1) z q[2];
if(c1==1) x q[2];
post q[2];
measure q[2] -> c2[0];
```



量子传输

LIQUi|⟩

- 内嵌函数式语言，基于 F#
- 编译器
- 电路数据结构
- 辅助性工具
- 模拟器
- 运行时
- 后端



整体架构

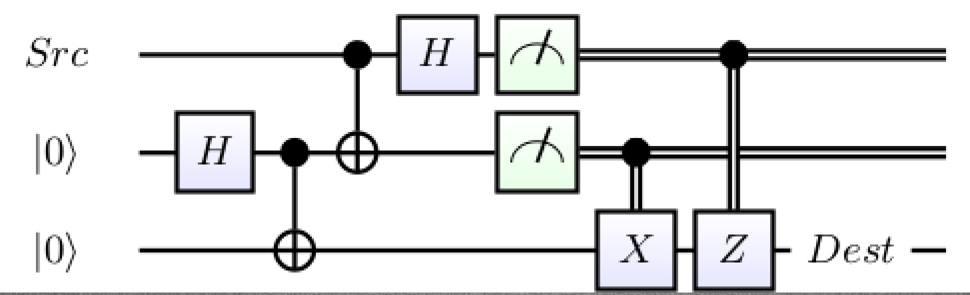
LIQUi|⟩

- 数据类型
  - Bit ，传统值
  - Qubit ，量子值
  - Ket ，系统中所有 Qubit 的状态向量
  - Gate ，量子门，数据结构
  - Operation ，量子操作， F# 中的函数
  - Circuit ，量子电路

LIQUi|⟩

- 函数 (例子相关部分)
  - 测量， M qs
  - H 门和 CNOT 门
  - 二进制控制门， BC X qs
- 电路操作 (例子相关部分)
  - 运行，执行电路
  - Dump ，打印电路中间表示
  - Fold ，优化，去除 identity gate
  - GrowGates ，优化，将连续的酉门合并成单个大操作

LIQUi|⟩



量子传输

```
// Define an EPR function
let EPR (qs:Qubits) = H qs; CNOT qs

// Teleport qubit 0 to qubit 2
let teleport (qs:Qubits) =
  let qs' = qs.Tail

  LabelL >!< // Give names to the qubits
    (["Src";"\\ket{0}";"\\ket{0}"],qs)

  EPR qs'; CNOT qs; H qs
  M qs'; BC X qs'          // Maybe apply X
  M qs; BC Z !!(qs,0,2)    // Maybe apply Z
  LabelR "Dest" !!(qs,2)   // Label output
```

函数定义

```
let ket = Ket(3)       // Create state
let qs  = ket.Qubits
teleport qs            // Run Teleport
let circ  =            // Compile to circuit
    Circuit.Compile teleport qs
circ.Run qs            // Run circuit
circ.Dump()            // Dump gates to log

circ.Fold()            // Fold the circuit
    .RenderHT(``Teleport'') // Draw HTML and TeX
let circ2 =            // Grow Unitaries together
    circ.GrowGates ket
circ2.Run qs           // Run the optimized circuit
```

基于电路的各种操作

问题？

发送邮件到：
mingjie.xing@gmail.com