Quantum Circuit Optimization with SPIRAL: A First Look

Scott Mionis Franz Franchetti Jason Larkin

https://github.com/spiral-software/spiral-software

https://github.com/spiralgen/spiral-package-quantum

Carnegie Mellon University
Electrical & Computer ENGINEERING

SPIRAL and the Quantum Package detailed in this presentation can be found on GitHub

QC is Linear Algebra

Quantum State - Physics

- Superposition: qubit states are unit-norm complex vectors on the Bloch Sphere
- Coordinates with respect to a certain basis denote the "square root probability" that the value is read when measured in that basis

$q = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

measured in 0,1 basis

$(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ probability of reading "0"

$(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ probability of reading "1"

$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

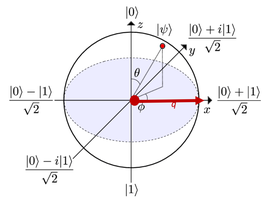
$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$

$q = 1|+\rangle + 0|-\rangle$

measured in +,- basis

100% probability of reading "+"

0% probability of reading "-"



Quantum State – Linear Algebra

- Qubit states are unit-norm, 2D complex vectors (4D space)
- Qubits can be "measured" in any orthogonal basis; Outcome probability is the magnitude squared of the projection onto measured basis

$q = \begin{bmatrix} a \\ b \end{bmatrix} \quad a, b \in \mathbb{C} \quad ||q|| = 1$

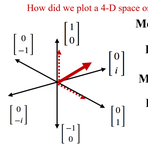
- Or -

$q = a\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + b\begin{bmatrix}0\\1\\0\\0\end{bmatrix} + c\begin{bmatrix}0\\0\\1\\0\end{bmatrix} + d\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

$a, b, c, d \in \mathbb{R} \quad ||q|| = 1$

"Measurement" is destructive: Qubit state collapses to measured state

How did we plot a 4-D space on 3 axes?



Measuring Φ in the x basis

$P(x): |x \cdot \Phi|^2$

Measuring $y\Phi$ in the x basis

$P(x): |x \cdot y\Phi|^2 = |x \cdot \Phi|^2$ If $|y| = 1$

Global Phase Shift

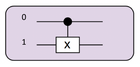
These two states are indistinguishable to a measurement device

Problem Definition

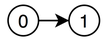
Qubit Connectivity

- Qubits must be physically adjacent to interact

$CNOT = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$

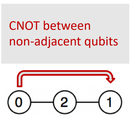


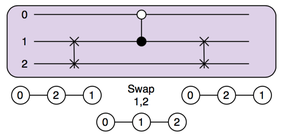
Requires connectivity



- For loosely-connected devices, must insert SWAPs

CNOT between non-adjacent qubits

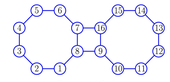




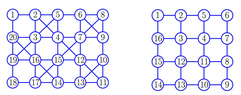
Non-trivial to find minimal swaps on large circuits!

Qubit Topology

- k-qubit operation on a n-qubit mesh has ~O(P_k^n) placements in the mesh $\frac{n!}{(n-k)!}$.
  But only those subgroups that meet connectivity requirements

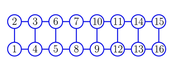


(a) Rigetti 16Q-Aspen



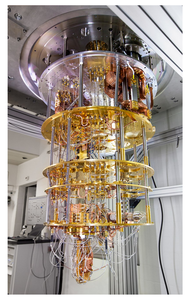
(d) IBM QX20 Tokyo

(c) 16q-Square



(b) IBM QX5

IBM Research, Zurich

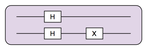


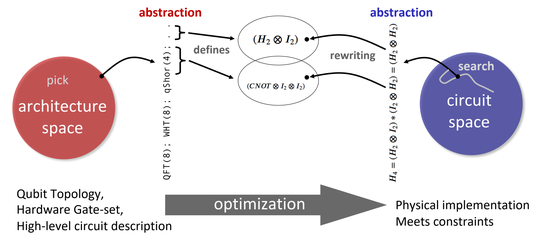
Our Approach: SPIRAL

For more about the classical SPIRAL compilation system, visit **http://spiral.net**

SPIRAL Quantum Compiler

Model: transform abstraction => Space of implementing circuits



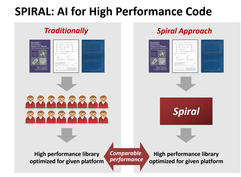


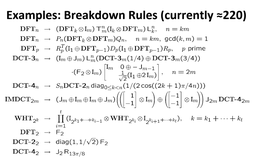
Qubit Topology, Hardware Gate-set, High-level circuit description

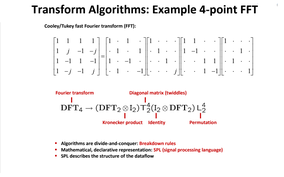
optimization

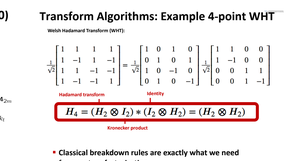
Physical implementation Meets constraints

SPIRAL System Overview



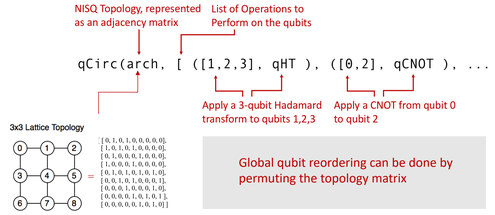






Defining a Circuit

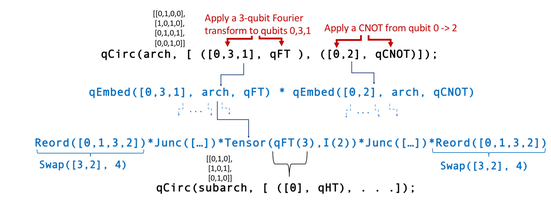
- A top-level circuit definition:



- **Next:** Recursive breakdown stages

The Embed Operation

- Architecture is Pruned in recursive calls
  Tree represents all possible mesh placements



- How do we find the globally-optimal embedding?

Single-Qubit Computation

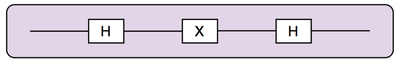
- We perform single-qubit operations with 2x2 unitary rotation matrices
- Hardware "ISA" codifies a subset of these as physically-realizable operations, or *Quantum Gates*

Single-Qubit gates: irreducible basic blocks

$\begin{bmatrix}\cos(\pi/4) & -e^{i0}\sin(\pi/4)\\ e^{i0}\sin(\pi/4) & e^{i0+i0}\cos(\pi/4)\end{bmatrix} * \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\sin(\pi/4)\\ \cos(\pi/4)\end{bmatrix}$

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

Single-qubit circuit:



$= HXH = \frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix} * \begin{bmatrix}0&1\\1&0\end{bmatrix} * \frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix} = \begin{bmatrix}1&0\\0&-1\end{bmatrix} = Z$

$q_0 = \begin{bmatrix}a\\b\end{bmatrix} \Longrightarrow$ Z $\Longrightarrow q_1 = \begin{bmatrix}1&0\\0&-1\end{bmatrix} * \begin{bmatrix}a\\b\end{bmatrix}$

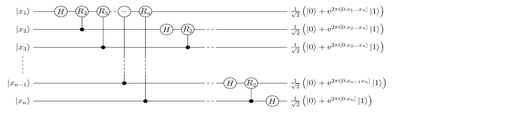
Quantum Fourier Transform (QFT)

- Quantum Fourier transform

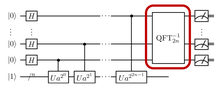
$\text{QFT}: |x\rangle \mapsto \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\omega_N^{xk}|k\rangle.$

$\text{QFT} = \frac{1}{\sqrt{N}}\begin{bmatrix}1&1&1&1&\cdots&1\\1&\omega&\omega^2&\omega^3&\cdots&\omega^{N-1}\\1&\omega^2&\omega^4&\omega^6&\cdots&\omega^{2(N-1)}\\\vdots&\vdots&\vdots&\vdots&&\vdots\\1&\omega^{N-1}&\omega^{2(N-1)}&\omega^{3(N-1)}&\cdots&\omega^{(N-1)(N-1)}\end{bmatrix}$

$\text{QFT}(|x_1 x_2 \ldots x_n\rangle) = \frac{1}{\sqrt{N}}\left(|0\rangle + e^{2\pi i\,[0.x_n]}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\,[0.x_{n-1}x_n]}|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle + e^{2\pi i\,[0.x_1 x_2 \ldots x_n]}|1\rangle\right)$

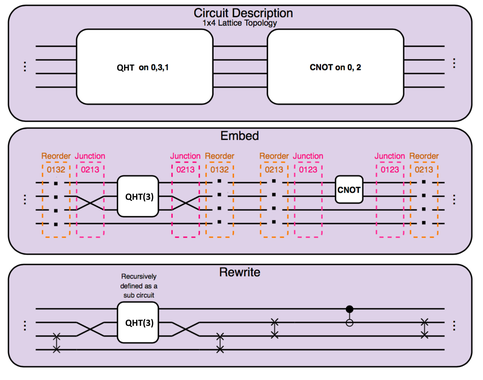


- Shor's Algorithm…



Every logical circuit is a matrix, decomposed into gate blocks

Embedding a Transform



Rewrite Rules

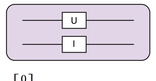
- Perform direct or conditional substitutions to collapse gates and simplify circuit description

```
Rewriting Rules:

# Flatten Tensors
• ex) Tensor(Tensor(H2, I2), Tensor(X2, Y2)) => Tensor(H2, I2, X2, Y2)

# Combine Tensors
• ex) Tensor(H2, I2) * Tensor(I2, H2) => Tensor(H2, H2)

# Combine Reorder
• ex) Reorder([0,3,2,1]) * Reorder([0,3,1,2]) => Reorder([0,1,3,2])

# Combine CNOT
• ex) CNOT(1->0) * CNOT(1->0) => I(2)

. . .
```

The "Best" embedding has adjacent Reorder steps that reduce

Multi-Qubit Computation

- Parallel gate applications are tensor products



Hadamard to qubit 0, Identity to qubit 1

$U = \begin{bmatrix}p&r\\q&s\end{bmatrix}$

$01 = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}0\\1\end{bmatrix} \Rightarrow U * \begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}p\\q\end{bmatrix}\otimes\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}p\\q\\0\\0\end{bmatrix}$

$10 = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}1\\0\end{bmatrix} \Rightarrow U * \begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}r\\s\end{bmatrix}\otimes\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}r\\s\\0\\0\end{bmatrix}$

Transform basis vectors, leveraging separation principle

Extends to EPR pair

• • • Continue for 4 bases

Transformation Matrix

$Circuit = \begin{bmatrix}p&r&0&0\\q&s&0&0\\0&0&p&r\\0&0&q&s\end{bmatrix} = U \otimes I$

Quantum Algorithm Search

- Many existing solvers are Peephole Optimizers
  Circuit space is exponentially large
- **The true problem: Search over the circuit space for a given matrix**
  Every implementation has a unique mathematical expression in a limited language

$circuit_{opt}(Mat) = \underset{circ(Mat)}{\arg\min}\ \text{Cost}(circ(Mat)) \quad \text{where} \quad circ(Mat) = \text{Factorize}(Mat)$

Mat: Characteristic Circuit Matrix

Factorize: Factor Mat into BNF form

```
<basic_gate>_{1} ::= H
                   | X
                   | Z
                   | I
                   ...
<cnot>_{2}       ::= CNOT_c1   # qubit 1 controls qubit 0
                   | CNOT_c0   # qubit 0 controls qubit 1

<circuit>_{n}    ::= <circuit>_{n} * <circuit>_{n}
                   | <circuit>_{k} tensor <circuit>_{m}  where n = mk
                   | <cnot>_{2}
                   | <basic_gate>_{1}
```

Can we automatically solve this problem?

- **A logical quantum circuit is a 2^n x 2^n matrix that can be expressed in a language of gates, matrix products, and tensor products**
  Matrix Factorization problem, Backus-Naur Form

Optimization

$rt_{opt}(arch) = \underset{rt,arch}{\arg\min}\ \text{Cost}(\text{Rewrite}(\text{Breakdown}(rt, circ, arch)))$

Breakdown(rt, circ) : applies breakdown rule sequence *rt* to transform *circ*

Rewrite(c) : applies rewrite rules to simplify expression *c*

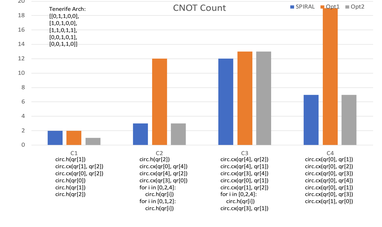
Cost(t) : returns the cost of gate expression *t*

arch: The qubit topology of the architecture, as an adjacency matrix

- Solve via Dynamic Programming or Genetic Algorithms
- Unparse the circuit as a QASM program
- Actually a factorization of subgroups of the permutation group

First Results

- We tested SPIRAL against IBM's Quiskit optimizer with Cost(t) = #(CNOT)(t)
- Executed final QASM code on IBM's Tenerife and Bogota devices



Test circuit, written in quiskit syntax

- Next steps are to add breakdown-rule heuristics, additional rewriting rules, and QFT support (Leveraging SPIRAL's FFT algorithms).