20th ANNUAL HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE

Quantum Circuits A Verified Optimizer for

Kesha Hietala, *Robert Rand**, Shih-Han Hung, Xiaodi Wu and Michael Hicks

***University of Chicago

Verified compiler stack

Verified compiler stack

Verified compiler stack

Small Quantum IR that two circuits are *equivalent up to a global phase*, written **Small Quantum IR**

• Intermediate goal: An IR suitable for representing and reasoning about quantum programs • Intermediate goal: An IR suitable for representing and
PARECIAE COAL: SURPLIER PIROGRAMS sqir with a *branching measurement* operation.

$$
U := U_1; U_2 | G q | G q_1 q_2
$$

$$
P := \text{skip} | P_1; P_2 | U | \text{meas } q P_1 P_2
$$

Verified Compiler Stack

• Superposition: Every quantum bit (or *qubit*) can be in a combination of 0 and 1 states simultaneously.

- *• Superposition*: Every quantum bit (or *qubit*) can be in a combination of 0 and 1 states simultaneously.
- *• Entanglement*: Qubits can be bound to one another in such a way that operating on qubit **x** influence qubit **y**.

- *• Superposition*: Every quantum bit (or *qubit*) can be in a combination of 0 and 1 states simultaneously.
- *• Entanglement*: Qubits can be bound to one another in such a way that operating on qubit **x** influence qubit **y**.
- *• Measurement*: Inspecting a qubit induces a *probabilistic transition* while perturbing the system.

- *• Superposition*: Every quantum bit (or *qubit*) can be in a combination of 0 and 1 states simultaneously.
- *• Entanglement*: Qubits can be bound to one another in such a way that operating on qubit **x** influence qubit **y**.
- *• Measurement*: Inspecting a qubit induces a *probabilistic transition* while perturbing the system.

- *• Superposition*: Every quantum bit (or *qubit*) can be in a combination of 0 and 1 states simultaneously.
- *• Entanglement*: Qubits can be bound to one another in such a way that operating on qubit **x** influence qubit **y**.
- *• Measurement*: Inspecting a qubit induces a *probabilistic transition* while perturbing the system. Deferred.

Why is quantum hard?
(operationally)

Why is quantum hard? (operationally)

• A n-qubit quantum state is represented as a length 2*n* vector of complex numbers.

Why is quantum hard? (operationally)

- A n-qubit quantum state is represented as a length 2*n* vector of complex numbers.
- Every operation on the quantum state can alter the entire vector.

Qubits

Qubits

Qubits

Bell pair

Bell pair

Linear Algebra

Linear Algebra

Linear Algebra

$SQLR$ (for ∈ R) represent the same physical state. We therefore say *^U*¹ [≅] *^U*2, when there exists a such that ⁿ*U*1o*^d* ⁼ *^ei* ⁿ*U*2o*^d* .

• Syntax

U ∶= U_1 ; U_2 | *G q* | *G q*₁ *q*₂ $P :=$ skip $| P_1; P_2 | U |$ meas *q* P_1 P_2

- Semantics assumes a **global register** of size *^d* emantics assumes a **giobal register**
	- A unitary program corresponds to a unitary matrix of size $2^d \times 2^d$
	- A non-unitary program corresponds to a function between density matrices of size $2^d \times 2^d$ | non-unitary program corresponds α non-unitary program corresponds to a finite space. recy

$SQLR$ (for ∈ R) represent the same physical state. We therefore say *^U*¹ [≅] *^U*2, when there exists a such that ⁿ*U*1o*^d* ⁼ *^ei* ⁿ*U*2o*^d* .

• Syntax

$$
U := U_1; U_2 | G q | G q_1 q_2
$$

$$
P := \text{skip} | P_1; P_2 | U | \text{meas } q P_1 P_2
$$

- Semantics assumes a **global register** of size *^d* emantics assumes a **giobal register**
	- A unitary program corresponds to a unitary matrix of size $2^d \times 2^d$
	- A non-unitary program corresponds to a function between density matrices of size $2^d \times 2^d$ | non unitary program corresponde , non-unitary program corresponds to
http://www.doncity.matrices.of.cize.⁰⁴ y et.
K

Semantics

Unitary program \rightarrow unitary matrix

$$
\llbracket X \rrbracket = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \llbracket H \rrbracket = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
$$

$$
\llbracket U_1; \ U_2 \rrbracket_d = \llbracket U_2 \rrbracket_d \times \llbracket U_1 \rrbracket_d
$$

$$
\llbracket G_1 \ q \rrbracket_d = \begin{cases} apply_1(G_1, q, d) \quad well-type d \\ 0_{2^d} & \text{otherwise} \end{cases}
$$

$$
\llbracket G_1 \ q_1 \ q_2 \rrbracket_d = \begin{cases} apply_2(G_1, q_1, q_2, d) \quad well-type d \\ 0_{2^d} & \text{otherwise} \end{cases}
$$

Semantics

Unitary program \rightarrow unitary matrix

$$
\llbracket X \rrbracket = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \llbracket H \rrbracket = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
$$

$$
\llbracket U_1; \ U_2 \rrbracket_d = \llbracket U_2 \rrbracket_d \times \llbracket U_1 \rrbracket_d
$$

$$
\llbracket H \ 1 \rrbracket_4 = \begin{cases} I_2 \otimes H \otimes I_4 & \text{well-typeed} \\ 0_{2^4} & \text{otherwise} \end{cases}
$$

$$
\llbracket G_1 \ q_1 \ q_2 \rrbracket_d = \begin{cases} \text{apply}_2(G_1, q_1, q_2, d) & \text{well-typeed} \\ 0_{2^d} & \text{otherwise} \end{cases}
$$

X/Z-Propagation

Lemma X_H_slide: X q; H q \equiv H q; Z q.

Lemma X_H_slide: X q; H q \equiv H q; Z q.

Lemma X_H_s lide: X q; H q \equiv H q; Z q.

Lemma Z_CNOT_slide: Z q; $\overline{C}NOT$ \overline{q} q' \equiv CNOT q q'; Z q

- Proving matrix equivalences in Coq is tedious Suppose we wish to prove the equivalence Suppose with the equivalence the extension of the ex proving that the two expressions are equal. Suppose we wish to prove the equivalence We use gridify to verify most of the equivalences used *X n*; *CNOT m n* [≡] *CNOT m n*; *X n*
- E.g. X *n*; *CNOT m* $n \equiv CNOT$ *m n*; X *n* **• E.g.** X *n*; *CNOT* m $n \equiv CNOT$ m n ; X n for arbitrary *n*,*m* and dimension *d*. Applying our de!nition \therefore X n of equivalence, the proving to proving the set of \mathcal{A}

 \sum *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) = $\begin{pmatrix} 1 \end{pmatrix}$ $\mathfrak{c}, \mathfrak{u}$): $apply_1(X, n, d) \times apply_2(CNOT, m, n, d) =$ \overline{a} $apply_2(CNOT, m, n, d) \times apply_1(X, n, d)$ *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) =

for arbitrary *n*,*m* and dimension *d*. Applying our de!nition

Verifying Matrix Equivalences rifying Matrix Fouiwalenc local rewrites. Proof that an optimization is correct thus relies \mathcal{F}_{max} 4.2 Proving Circuit Equivalences Weritving Matrix E local rewrites. Proof that an optimization is correct thus relies $\frac{1}{2}$ Valences \mathbf{b} is the equation of the equation (1) becomes of the equation (1) becomes of the equation (1) becomes \mathbf{b} All of voqc's optimizations use circuit equivalences to justify atrix Founvalonces. au in cyclocatorico \mathcal{L} of our circuit equivalence proofs have a common form, \mathcal{L} and \mathcal{L} which we interested by the partition of the basic set of the set of the set of the set of the basic set of the Suppose we wish to prove the equivalence proving the two expressions are equal. We use the extension of the equivalence of the equi \blacksquare Suppose the equivalence that \blacksquare **X LYUIVAITIUS**

• Proving matrix equivalences in Coq is tedious Suppose we wish to prove the equivalence Suppose with the equivalence the extension of the ex $t \circ d$ We use gridify to verify most of the equivalences used Suppose we wish to prove the equivalence for arbitrary *n*,*m* and dimension *d*. Applying our de!nition *X n*; *CNOT m n* [≡] *CNOT m n*; *X n* • Proving matrix equivalences in Coq is tedious alences used in *gate cancellation* and *Hadamard reduction* $\overline{}$

*appl*² as follows with *^p* ⁼ *ⁿ* [−]*^m* [−] ¹ and *^q* ⁼ *^d* [−] *ⁿ* [−] 1:

• E.g. X *n*; *CNOT m* $n \equiv CNOT$ *m n*; X *n* for arbitrary *n*,*m* and dimension *d*. Applying our de!nition \sum *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) = $\begin{pmatrix} 1 \end{pmatrix}$ $\mathfrak{c}, \mathfrak{u}$): • E.g. X n; CNOT m n \equiv CNOT m n; X n for arbitrary *n*,*m* and dimension *d*. Applying our de!nition $apply_1(X, n, d) \times apply_2(CNOT, m, n, d) =$ \overline{a} \therefore X n $apply_1(X, n, d) \times apply_2(CNOT, m, n, d) = apply_2(CNOT, m, n, d) \times apply_1(X, n, d).$ of equivalence, the proving to proving the set of \mathcal{A} *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) = *appl*2(*CNOT*,*m*,*n*,*d*) × *appl*1(*X*,*n*,*d*), $\overline{\imath}$; per Figure 3. Suppose both sides of the equation are well *^m* < *ⁿ* (the *ⁿ* < *^m* case is similar). We expand *appl*¹ and *appl*2(*CNOT*,*m*,*n*,*d*) × *appl*1(*X*,*n*,*d*), $p(x) \in \mathbb{N}$ is a suppose both side of the equation are well as $p(x)$ and $p(x)$ are well as $p(x)$ are $p(x)$ and $p(x)$ ar typed (*^m* < *^d* and *ⁿ* < *^d* and *^m* ⁼/ *ⁿ*), and consider the case $MOT \times n$ d = applus $(CMOT \times n$ d \times applus $(Y \times d)$ $\mathcal{L}NOI(m, n, d) = apply_2(CNOI, m, n, d) \times apply_1(X, n, d)$

*appl*2(*CNOT*,*m*,*n*,*d*) × *appl*1(*X*,*n*,*d*), $p p l y_1(X, n, a) = I_{2^n} \otimes \sigma_X \otimes I_{2^q}$ $NOT, m, n, d) = I_{2^m} \otimes |1\rangle\langle 1| \otimes I_{2^p} \otimes \sigma_x \otimes I_{2^q} + I_{2^m} \otimes |0\rangle\langle 0| \otimes I_{2^p}$ $\overline{}$ $9 \vert 1_2 q \vert$ per $\frac{1}{1 + \sum_{i=1}^{n} 1}$ $\text{supp} \mu y_1(\Lambda, n, u) - 12^n \otimes 0_X \otimes 12^q$ *apply*₂ (CNOT, m, n, d) = $I_{2^m} \otimes |1\rangle\langle 1| \otimes I_{2^p} \otimes \sigma_x$ *appl*² as follows with *^p* ⁼ *ⁿ* [−]*^m* [−] ¹ and *^q* ⁼ *^d* [−] *ⁿ* [−] 1: $\mathbf{1} \cdot \mathbf{3}$ optimization by Propagation and Cancellation and Ca $\begin{bmatrix} 1 & 1 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 2 & 0 \end{bmatrix}$ $\begin{bmatrix} 2 & 0 \end{bmatrix}$ typed (*^m* < *^d* and *ⁿ* < *^d* and *^m* ⁼/ *ⁿ*), and consider the case $\sigma_x \otimes I_{2q}$ $\langle 1 \rangle \langle 1 | \otimes I_{2P} \otimes \sigma_x \otimes I_{2q} + I_{2m} \otimes |0 \rangle \langle 0 | \otimes I_{2P} \otimes I_2 \otimes I_{2q}$ *appl*1(*X*,*n*,*d*) = *I*2*ⁿ* ⊗ *^x* ⊗ *I*2*^q* $apply_1(X, n, d) = I_{2^n} \otimes \sigma_x \otimes I_{2^q}$ $apply_2(CNOT, m, n, d) = I_{2m} \otimes |1\rangle\langle 1| \otimes I_{2p} \otimes \sigma_x \otimes I_{2q}$ rules. Then we apply a circuit equivalence to replace that set $\left|0\right\rangle$ (0 \otimes I_{α} \otimes I_{α} \otimes I_{α}) of code patterns, but one—*not propagation*—is di#erent, so *a*_{*a*} ⊗ *I*_{2*q*} $\overline{C_x}$ ⊗ *I*_{2*q*} $\overline{C_x}$ ∴ ∑ *i*² ∞ ∞ $+ I_{2}m \otimes |0\rangle\langle0| \otimes I_{2}p \otimes I_{2} \otimes I_{2}q$

^m < *ⁿ* (the *ⁿ* < *^m* case is similar). We expand *appl*¹ and

⁺ *^I*2*^m* [⊗] [∣]0⟩⟨0[∣] [⊗] *^I*2*^p* [⊗] *^I*² [⊗] *^I*2*^q*

localize a set of gates by repeatedly applying commutation

- Proving matrix equivalences in Coq is tedious Suppose we wish to prove the equivalence Suppose with the equivalence the extension of the ex proving that the two expressions are equal. Suppose we wish to prove the equivalence We use gridify to verify most of the equivalences used *X n*; *CNOT m n* [≡] *CNOT m n*; *X n*
- E.g. X *n*; *CNOT m* $n \equiv CNOT$ *m n*; X *n* **• E.g.** X *n*; *CNOT* m $n \equiv CNOT$ m n ; X n for arbitrary *n*,*m* and dimension *d*. Applying our de!nition \therefore X n of equivalence, the proving to proving the set of \mathcal{A}

 \sum *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) = $\begin{pmatrix} 1 \end{pmatrix}$ $\mathfrak{c}, \mathfrak{u}$): $apply_1(X, n, d) \times apply_2(CNOT, m, n, d) =$ \overline{a} $apply_2(CNOT, m, n, d) \times apply_1(X, n, d)$ *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) =

for arbitrary *n*,*m* and dimension *d*. Applying our de!nition

- Proving matrix equivalences in Coq is tedious Suppose we wish to prove the equivalence Suppose with the equivalence the extension of the ex proving that the two expressions are equal. Suppose we wish to prove the equivalence We use gridify to verify most of the equivalences used *X n*; *CNOT m n* [≡] *CNOT m n*; *X n*
- \bullet E.g. X *n*; *CNOT m* $n \equiv CNOT$ *m n*; X *n* **• E.g.** X *n*; *CNOT* m $n \equiv CNOT$ m n ; X n for arbitrary *n*,*m* and dimension *d*. Applying our de!nition \therefore X n of equivalence, the proving to proving the set of \mathcal{A}

*appl*2(*CNOT*,*m*,*n*,*d*) × *appl*1(*X*,*n*,*d*),

normalize

per Figure 3. Suppose both sides of the equation are well

typed (*^m* < *^d* and *ⁿ* < *^d* and *^m* ⁼/ *ⁿ*), and consider the case

 \sum *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) = $\begin{pmatrix} 1 \end{pmatrix}$ $\mathfrak{c}, \mathfrak{u}$): $apply_1(X, n, d) \times apply_2(CNOT, m, n, d) =$ \overline{a} $apply_2(CNOT, m, n, d) \times apply_1(X, n, d)$ *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) =

per Figure 3. Suppose both sides of the equation are well as the equation are well as the equation are well as
The equation are well as the equatio

*appl*2(*CNOT*,*m*,*n*,*d*) × *appl*1(*X*,*n*,*d*),

for arbitrary *n*,*m* and dimension *d*. Applying our de!nition

• Proving matrix equivalences in Coq is tedious Suppose we wish to prove the equivalence Suppose with the equivalence the extension of the ex proving that the two expressions are equal. Suppose we wish to prove the equivalence We use gridify to verify most of the equivalences used *X n*; *CNOT m n* [≡] *CNOT m n*; *X n*

per Figure 3. Suppose both sides of the equation are well

typed (*^m* < *^d* and *ⁿ* < *^d* and *^m* ⁼/ *ⁿ*), and consider the case

 \bullet E.g. X *n*; *CNOT m* $n \equiv CNOT$ *m n*; X *n* for arbitrary *n*,*m* and dimension *d*. Applying our de!nition \sum *a*_p
*a*_p *apprmalize* $\begin{pmatrix} 1 \end{pmatrix}$ $\mathfrak{c}, \mathfrak{u}$): **• E.g.** X *n*; *CNOT* m $n \equiv CNOT$ m n ; X n for arbitrary *n*,*m* and dimension *d*. Applying our de!nition $apply_1(X, n, d) \times apply_2(CNOT, m, n, d) =$ *appl*2(*CNOT*,*m*,*n*,*d*) × *appl*1(*X*,*n*,*d*), \overline{a} \therefore X n $apply_2(CNOT, m, n, d) \times apply_1(X, n, d)$ one used in *rotation merging*, we do not use gridify directly, of equivalence, the proving to proving the set of \mathcal{A} *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) = \mathcal{P}^{e} typed (*^m* < *^d* and *ⁿ* < *^d* and *^m* ⁼/ *ⁿ*), and consider the case normalize normalize

per Figure 3. Suppose both sides of the equation are well as the equation are well as the equation are well as
The equation are well as the equatio

^m < *ⁿ* (the *ⁿ* < *^m* case is similar). We expand *appl*¹ and

 \mathcal{A} . So the Propagation by Propagation and \mathcal{A}

• Proving matrix equivalences in Coq is tedious Suppose we wish to prove the equivalence Suppose with the equivalence the extension of the ex $t \circ d$ We use gridify to verify most of the equivalences used Suppose we wish to prove the equivalence ● (*A* ⊗ *B*) × (*C* ⊗ *D*) = (*A* ×*C*) ⊗ (*B* × *D*) *X n*; *CNOT m n* [≡] *CNOT m n*; *X n* OVING Matrix equivalences in Coq is tealous

• E.g. X *n*; *CNOT m* $n \equiv CNOT$ *m n*; X *n* for arbitrary *n*,*m* and dimension *d*. Applying our de!nition \sum *a*_p
*a*_p *apprmalize* per Figure 3. Suppose both sides of the equation are well as the equation are well as the equation are well as
The equation are well as the equatio $t_{2m} \otimes |1\rangle\langle 1| \otimes I_{2p} \otimes I_2 \otimes I_{2q} + I_{2m} \otimes |0\rangle\langle 0| \otimes I_{2p} \otimes \sigma_x \otimes I_{2q}$ $\begin{pmatrix} 1 \end{pmatrix}$ $\mathfrak{c}, \mathfrak{u}$): **• E.g.** X *n*; *CNOT* m $n \equiv CNOT$ m n ; X n for arbitrary *n*,*m* and dimension *d*. Applying our de!nition $apply_1(X, n, d) \times apply_2(CNOT, m, n, d) =$ *appl*2(*CNOT*,*m*,*n*,*d*) × *appl*1(*X*,*n*,*d*), \overline{a} per Figure 3. Suppose both sides of the equation are well $I \quad \otimes 111/100 \quad I \quad \otimes I \quad \otimes I \quad \vdots \quad I \quad \otimes$ $I_{2^m}\otimes|1\rangle\langle1|\otimes I_{2^p}\otimes I_2\otimes I_{2^q}+I_{2^m}\otimes|0\rangle\langle0|\otimes I_{2^p}\otimes\sigma_x\otimes I_{2^q}.$ \therefore X n $apply_2(CNOT, m, n, d) \times apply_1(X, n, d)$ one used in *rotation merging*, we do not use gridify directly, 4.3 Optimization by Propagation and Cancellation of equivalence, the proving to proving the set of \mathcal{A} *appl*1(*X*,*n*,*d*) × *appl*2(*CNOT*,*m*,*n*,*d*) = \mathcal{P}^{e} typed (*^m* < *^d* and *ⁿ* < *^d* and *^m* ⁼/ *ⁿ*), and consider the case *^m* < *ⁿ* (the *ⁿ* < *^m* case is similar). We expand *appl*¹ and $I_{2^m}\otimes|1\rangle\langle1|\otimes I_{2^p}\otimes I_2\otimes I_{2^q}+I_{2^m}\otimes|0\rangle\langle0|\otimes I_{2^p}\otimes\sigma_x\otimes I_{2^q}$ rules. (For instance, in the example above, *I*2*ⁿ* would be reg. X *n*; $CNOT$ *m* $n \equiv CNOT$ *m n*; X *n* $\left\{\left.\right. \right\}$ are in grid normal form, gridifying $\left\{\left.\right\}$ $\sum_{i=1}^{\infty}$ $m, u, u \rightarrow \alpha p p u g_2(\text{CNOT}, m, u, u) - \alpha p p u g_2(\text{CNOT}, m, u, u) \sim \alpha p p u g_1(x)$ norma, aliza $\frac{n^{\alpha}}{2}$ normalize normalize

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

[Nam, Ross, Su, Childs, Maslov, 2018]

Performance

• Our *verified* optimizer vs. existing *unverified* optimizers

- 1<https://qiskit.org/>
- 2<https://cqcl.github.io/pytket/build/html/index.html>
- 3<https://arxiv.org/pdf/1710.07345.pdf>
- 4<https://arxiv.org/pdf/1303.2042.pdf>
- 5<https://github.com/Quantomatic/pyzx>

Lemma Z_meas: Z q ; meas q then c1 else c2 \equiv meas q then c1 else c2

Lemma X-meas:
\n
$$
X q
$$
; means q then c1 else c2 \equiv meas q then X q; c2 else X q; c1

Lemma X-meas:
\nX q ; meas q then c1 else c2
$$
\equiv
$$

\nmeas q then X q; c2 else X q; c1

Lemma meas
$$
\begin{bmatrix}\n\text{CMO} \\
\text{meas} \\
\text{meas} \\
\text{meas} \\
\text{theo} \\
\text
$$

Circuit Mapping

- Given an input program & description of machine connectivity, mapping produces a program that meets connectivity constraints
	- \triangleright E.g. CNOT 0 2 0 \rightarrow 1 \rightarrow 2 \rightarrow SWAP 0 1; CNOT 1 2
- We prove that the output program is equivalent to the original, up to permutation of indices
	- Above, $\left[\text{CNOT } 0\ 2\right]_3 = P \times \left[\text{SWAP } 0\ 1\right]$; CNOT $1\ 2\right]_3$ where P implements the permutation $\{0 \rightarrow 1, 1 \rightarrow 0, 2 \rightarrow 2\}$

Boolean Oracle Compilation

- Many quantum programs rely on *classical oracles*, classical functions evaluated on quantum data
- We have verified the compilation of Boolean formulas into quantum circuits with X, CNOT, and Toffoli gates

General Verification

- Scale invariant correctness proofs of variety of quantum algorithms:
	- GHZ state preparation
	- Deutsch-Jozsa algorithm
	- Simon's algorithm
	- Quantum phase estimation

In Progress

- Full proof of correctness for Shor's algorithm
- Adding Python bindings to improve VOQC usability
- Extending oracle compilation to include arithmetic functions
- Implementing additional optimizations (e.g. those used in Qiskit)

Future Directions

- Verify approximate algorithms and optimizations
- Compile from high-level languages (Silq, Q#) to SQIR
- Compile from SQIR to low-level pulses
- Verify other parts of the software stack (e.g. resource estimation)

Future Directions

- Verify approximate algorithms and optimizations
- Compile from high-level languages (Silq, Q#) to SQIR
- Compile from SQIR to low-level pulses
- Verify other parts of the software stack (e.g. resource estimation)

Paper: http://people.cs.uchicago.edu/~rand/voqc_draft.pdf

Future Directions

- Verify approximate algorithms and optimizations
- Compile from high-level languages (Silq, Q#) to SQIR
- Compile from SQIR to low-level pulses
- Verify other parts of the software stack (e.g. resource estimation)

Paper: http://people.cs.uchicago.edu/~rand/voqc_draft.pdf Code: <https://github.com/inQWIRE/SQIR>