Generating Formally Verified Quantum Fourier Transform Algorithms

Patrick Brinich✉ **Jeremy Johnson**
{pjb338, johnsojr}@drexel.edu

Drexel University

August 8, 2024

Overview

- Formally verified mapping from matrix specifications to quantum circuits
- Alternative circuits generated using verified rewrite rules corresponding to matrix factorizations
    - Matrix factorizations encoded and derived using SPIRAL methodology.
- Verification using the Coq Proof Assistant
    - Quantum circuits defined by SQIR programs
    - Matrix semantics supported by QuantumLib with extensions

Outline

1. Quantum Computing
2. The Discrete Fourier Transform and Quantum Fourier Transform
3. Generating Quantum Algorithms
4. Formal Verification
5. Future Work

Quantum Computing

Qubit

- Quantum State
  - 2-dimensional Complex Hilbert Space
  - $|0\rangle = e_0^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = e_1^2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- Superposition
  - $\alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ with $|\alpha|^2 + |\beta|^2 = 1$
- Measurement
  - States $|0\rangle$ and $|1\rangle$ with probability $|\alpha|^2$ and $|\beta|^2$

Quantum Computing

Multi-Qubit System

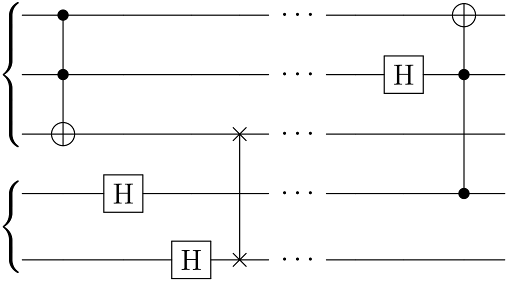
- Tensor Product
  - $e_i^m \otimes e_j^n = e_{in+j}^{mn}$
- $|00\rangle = |0\rangle \otimes |0\rangle = e_0^4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ldots, |11\rangle = |1\rangle \otimes |1\rangle = e_3^4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$
- Preserves norm condition
- Pure versus entangled

Quantum Computing

Quantum Circuits (without measurement)

- Unitary Operators
  - Must preserve norm
  - $UU^\dagger = \mathrm{I}_n$
- Implemented with quantum gates
- Not all Unitary operations have the same costs

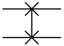
Quantum Circuit



Quantum Gates

- Hadamard Gate $-\boxed{\mathrm{H}}-$
  - $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- Phase Shift Gate $-\boxed{\mathrm{P}_{\mathsf{exp}(i\phi)}}-$
  - $\begin{bmatrix} 1 & \\ & \mathsf{exp}(i\phi) \end{bmatrix}$

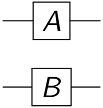
Quantum Gates

- SWAP gate 
    - $|q_0 q_1\rangle \mapsto |q_1 q_0\rangle$
    - $\begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix}$
- General $\mathrm{SWAP}_{m,n}$
    - $|q_0 \dots q_m \dots q_n \dots\rangle \mapsto |q_0 \dots q_n \dots q_m \dots\rangle$
    - Can be decomposed into multiple simple SWAPs.

Quantum Computing

Tensor (Kronecker) Product

- $(A \otimes B)(x \otimes y) = Ax \otimes By$
    - Associative (Not Commutative)
    - $(A \otimes B)(C \otimes D) = AC \otimes BD$
    - $(A \otimes B) = (A \otimes \mathrm{I})(\mathrm{I} \otimes B) = (\mathrm{I} \otimes B)(A \otimes \mathrm{I})$
    - $\mathrm{I}_m \otimes \mathrm{I}_n = \mathrm{I}_{mn}$
- Parallel Gate Application

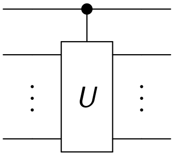


$$A^{2\times 2} \otimes B^{2\times 2} =$$

$$\begin{bmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{01}b_{00} & a_{01}b_{01} \\ a_{00}b_{10} & a_{00}b_{11} & a_{01}b_{10} & a_{01}b_{11} \\ a_{10}b_{00} & a_{10}b_{01} & a_{11}b_{00} & a_{11}b_{01} \\ a_{10}b_{10} & a_{10}b_{11} & a_{11}b_{10} & a_{11}b_{11} \end{bmatrix}$$

Quantum Gates

- Controlled Gates
  - Applies U to *target* qubits when *source* qubit is $|1\rangle$.
  - Notation $\mathrm{C}U$ for gates controlled by the leading qubit
  - $\mathrm{C}U^{n\times n} = |0\rangle\langle 0| \otimes \mathrm{I}_n + |1\rangle\langle 1| \otimes U = \mathrm{I}_n \oplus U$
    - $\langle x| = |x\rangle^\dagger$



Quantum Fourier Transform

Discrete Fourier Transform

Let $\omega = \exp(\frac{2\pi i}{n})$ be a primitive n^{th} root of unity. The n-point Discrete Fourier Transform is the matrix vector product $y = \mathrm{DFT}_n x$ where

$$\mathrm{DFT}_n e_i^n = \sum_{j=0}^{n-1} \omega^{ij} e_j^n.$$

- $\mathrm{DFT}_n = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \dots & \omega^{(n-1)(n-1)} \end{bmatrix}$
- $\mathrm{DFT}_n^{-1} = \frac{1}{n}\mathrm{DFT}_n(\overline{\omega})$

Quantum Fourier Transform

- Normalized Discrete Fourier Transform
- QFT on n qubits is $\frac{1}{\sqrt{2^n}}\mathrm{DFT}_{2^n}$
- $\mathrm{QFT}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \mathrm{H}$

Generating Quantum Algorithms

Recursive Cooley-Tukey Factorization

$$\mathrm{DFT}_{rs} = (\mathrm{DFT}_r \otimes \mathrm{I}_s)\mathrm{T}^{rs}_s(\mathrm{I}_r \otimes \mathrm{DFT}_s)\mathrm{L}^{rs}_r$$

$$\mathrm{DFT}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & 1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & 1 & i \end{bmatrix} = \begin{bmatrix} 1 & & 1 & \\ & 1 & & 1 \\ 1 & & -1 & \\ & 1 & & -1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & i \end{bmatrix} \begin{bmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1 \\ & & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix}$$

$$\mathrm{DFT}_4 = (\mathrm{DFT}_2 \otimes \mathrm{I}_2)\mathrm{T}^4_2(\mathrm{I}_2 \otimes \mathrm{DFT}_2)\mathrm{L}^4_2$$

Generating Quantum Algorithms

Stride Permutations

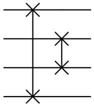
- $\mathrm{L}^{mn}_{n}(e^m_i \otimes e^n_j) = e^n_j \otimes e^m_i$
- Circular shift of qubits
- $\mathrm{L}^4_2 = \mathrm{SWAP}$

Generating Quantum Algorithms

$$
\begin{aligned}
&\mathrm{DFT}_{16} \\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}_8^{16}(\mathrm{I}_2 \otimes \mathrm{DFT}_8)\mathrm{L}_2^{16} \\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}_8^{16}(\mathrm{I}_2 \otimes (\mathrm{DFT}_2 \otimes \mathrm{I}_4)\mathrm{T}_4^{8}(\mathrm{I}_2 \otimes \mathrm{DFT}_4)\mathrm{L}_2^{8})\mathrm{L}_2^{16} \\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}_8^{16}(\mathrm{I}_2 \otimes (\mathrm{DFT}_2 \otimes \mathrm{I}_4)\mathrm{T}_4^{8}(\mathrm{I}_2 \otimes ((\mathrm{DFT}_2 \otimes \mathrm{I}_2)\mathrm{T}_2^{4}(\mathrm{I}_2 \otimes \mathrm{DFT}_2)\mathrm{L}_2^{4}))\mathrm{L}_2^{8})\mathrm{L}_2^{16} \\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}_8^{16}(\mathrm{I}_2 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{T}_4^{8})(\mathrm{I}_4 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{T}_2^{4})(\mathrm{I}_8 \otimes \mathrm{DFT}_2) \\
&\qquad \times (\mathrm{I}_4 \otimes \mathrm{L}_2^{4})(\mathrm{I}_2 \otimes \mathrm{L}_2^{8})\mathrm{L}_2^{16} \\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}_8^{16}(\mathrm{I}_2 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{T}_4^{8})(\mathrm{I}_4 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{T}_2^{4})(\mathrm{I}_8 \otimes \mathrm{DFT}_2)\mathrm{R}_{2^4}
\end{aligned}
$$

Bit Reversal

- $\mathrm{R}_{2^n}(e^2_{i_0} \otimes e^2_{i_1} \otimes \cdots \otimes e^2_{i_{n-1}}) = (e^2_{i_{n-1}} \otimes e^2_{i_{n-2}} \otimes \cdots \otimes e^2_{i_0})$
- $\mathrm{R}_{2^n} = (\mathrm{I}_2 \otimes \mathrm{R}_{2^{n-1}})\mathrm{L}_2^{2^n}$
- Flips the order of qubits
- Can be implemented using telescoping SWAPs



Generating Quantum Algorithms

$$
\begin{aligned}
&\mathrm{DFT}_{16}\\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}^{16}_{8}(\mathrm{I}_2 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{T}^{8}_{4})(\mathrm{I}_4 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{T}^{4}_{2})(\mathrm{I}_8 \otimes \mathrm{DFT}_2)\\
&\qquad \times (\mathrm{I}_4 \otimes \mathrm{L}^{4}_{2})(\mathrm{I}_2 \otimes \mathrm{L}^{8}_{2})\mathrm{L}^{16}_{2}\\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}^{16}_{8}(\mathrm{I}_2 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{T}^{8}_{4})(\mathrm{I}_4 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{T}^{4}_{2})(\mathrm{I}_8 \otimes \mathrm{DFT}_2)\\
&\qquad \times (\mathrm{I}_2 \otimes (\mathrm{I}_2 \otimes \mathrm{L}^{4}_{2})\mathrm{L}^{8}_{2})\mathrm{L}^{16}_{2}\\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}^{16}_{8}(\mathrm{I}_2 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{T}^{8}_{4})(\mathrm{I}_4 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{T}^{4}_{2})(\mathrm{I}_8 \otimes \mathrm{DFT}_2)\\
&\qquad \times (\mathrm{I}_2 \otimes \mathrm{R}_{2^3})\mathrm{L}^{16}_{2}\\
&= (\mathrm{DFT}_2 \otimes \mathrm{I}_8)\mathrm{T}^{16}_{8}(\mathrm{I}_2 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{T}^{8}_{4})(\mathrm{I}_4 \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{T}^{4}_{2})(\mathrm{I}_8 \otimes \mathrm{DFT}_2)\mathrm{R}_{2^4}
\end{aligned}
$$

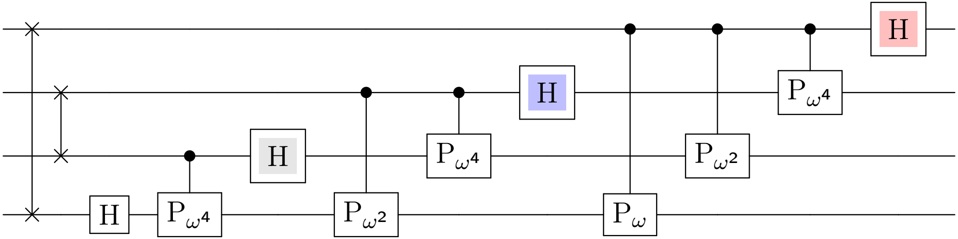
Twiddle Factors

- $$\mathrm{T}_n^{mn}(\omega) = \begin{bmatrix} \mathrm{I}_n & & & \\ & \Omega_n(\omega) & & \\ & & \ddots & \\ & & & \Omega^{m-1}(\omega) \end{bmatrix}$$
- $\Omega_n(\omega) = \mathrm{diag}(1, \omega, \omega^2, \ldots, \omega^{n-1})$
- Some properties
  - $\Omega_2(\omega) = \mathrm{P}_\omega$
  - $\mathrm{C}(U_1 U_2) = (\mathrm{C}U_1)(\mathrm{C}U_2)$
  - $\Omega_n^k(\omega) = \Omega_n(\omega^k)$
  - $\Omega_{2n}(\omega) = \Omega_2^n(\omega) \otimes \Omega_n(\omega)$

Generating Quantum Algorithms

$$
\begin{aligned}
\mathrm{T}_8^{16}(\omega) &= \begin{bmatrix} \mathrm{I}_8 & \\ & \Omega_8(\omega) \end{bmatrix} \\
&= \mathrm{C}\Omega_8(\omega) \\
&= \mathrm{C}(\Omega_2^4(\omega) \otimes \Omega_4(\omega)) \\
&= \mathrm{C}(\Omega_2^4(\omega) \otimes (\Omega_2^2(\omega) \otimes \Omega_2(\omega))) \\
&= \mathrm{C}(\Omega_2(\omega^4) \otimes \Omega_2(\omega^2) \otimes \Omega_2(\omega)) \\
&= \mathrm{C}(\mathrm{P}_{\omega^4} \otimes \mathrm{P}_{\omega^2} \otimes \mathrm{P}_{\omega}) \\
&= \mathrm{C}((\mathrm{P}_{\omega^4} \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{P}_{\omega^2} \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes P_{\omega})) \\
&= \mathrm{C}(\mathrm{P}_{\omega^4} \otimes \mathrm{I}_4)\mathrm{C}(\mathrm{I}_2 \otimes \mathrm{P}_{\omega^2} \otimes \mathrm{I}_2)\mathrm{C}(\mathrm{I}_4 \otimes P_{\omega})
\end{aligned}
$$

Example breakdown: 4-qubit QFT



$$\begin{aligned}
&\mathrm{DFT}_{16}\\
&= (\mathrm{H} \otimes \mathrm{I}_8)(\mathrm{C}(\mathrm{P}_{\omega^4} \otimes \mathrm{I}_4))(\mathrm{C}(\mathrm{I}_2 \otimes \mathrm{P}_{\omega^2} \otimes \mathrm{I}_2))(\mathrm{C}(\mathrm{I}_4 \otimes \mathrm{P}_{\omega}))\\
&\times (\mathrm{I}_2 \otimes \mathrm{H} \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{C}(\mathrm{P}_{\omega^4} \otimes \mathrm{I}_2))(\mathrm{I}_2 \otimes \mathrm{C}(\mathrm{I}_2 \otimes \mathrm{P}_{\omega^2}))\\
&\times (\mathrm{I}_4 \otimes \mathrm{H} \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{CP}_{\omega^4})\\
&\times (\mathrm{I}_8 \otimes \mathrm{H})(\mathrm{I}_2 \otimes \mathrm{SWAP} \otimes \mathrm{I}_2)\mathrm{SWAP}_{0,3}
\end{aligned}$$

Formal Verification

$$
\begin{tikzcd}
\text{Specification in SPL} \arrow[r, "\text{denotation}"] \arrow[d, "\text{search}"] & \text{Unitary Matrix} \\
\text{Algorithms in SPL} \arrow[r, "\text{translation}"] \arrow[ur, "\text{denotation}"] & \mathrm{SQIR}\ \text{Programs} \arrow[u, "\text{denotation}"']
\end{tikzcd}
$$

Formal Verification

$$\mathrm{QFT}_4$$

Unitary Matrix Specification

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & 1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & 1 & i \end{bmatrix}$$

Matrix Factorization in SPL

$$(\mathrm{H} \otimes \mathrm{I}_2)\mathrm{CP}_i(\mathrm{I}_2 \otimes \mathrm{H})\mathrm{SWAP}$$

Algorithm mapped to SQIR

```
SWAP 0 1;
H 1;
control 0 (Rz i 1);
H 0
```

Formal Verification

Formal Verification done in the Coq Proof Assistant

- Encoding the abstract syntax of the necessary SPL subset as inductive data type (deep embedding)
- Unitary Matrix Semantics for SPL subset
- Rewrite Rules with self-contained proofs of correctness
  - Recursive Cooley-Tukey
  - Iterative Cooley-Tukey
- Verified simple search process
- Verified SPL to SQIR translation

SPL Embedding

```
Inductive spl : nat → Set :=
| spl_I : ∀ n, spl n
(* Non-terminal *)
| spl_F : ∀ n, spl n
| spl_T : ∀ r s, spl (r+s)
| spl_L : ∀ r s, spl (r+s)
| spl_Rev : ∀ n, spl n
(* Intermediary *)
| spl_T' : ∀ r s (k: nat), spl (r+s)
| spl_Ω : ∀ n (k: nat), spl n
(* Gates *)
| spl_H : spl 1
| spl_P : nat → spl 1
| spl_SWAP : spl 2
| spl_GSWP : ∀ (dim m n : nat), spl dim
...
(* Controlled Unitary *)
| spl_C : ∀ {n}, spl n → spl (1 + n)
(* s1 × s2 *)
| spl_comp : ∀ {n} (s1 : spl n) (s2 : spl n), spl n
(* s1 ⊗ s2 *)
| spl_kron : ∀ {n1 n2} (s1 : spl n1) (s2 : spl n2),
             spl (n1+n2)
(* Equal-size casts with proof *)
| spl_cast : ∀ {n1 n2} (Heq: n1 = n2) (s : spl n1),
             spl n2.
```

with notations (not shown)

Formal Verification

Rewrite Rules

- Dependent Records
  - Takes a left-hand-side SPL term
  - A number of degrees of freedom with constraints
  - The right-hand-side term
  - Proof that both sides are equal under constraints
  - A way to generate all possible right-hand-sides
  - Proofs that this generation is sound and complete
- Everything you need to know about a rule is in one place!
- Correctness proof of search is mechanical

Formal Verification

```
Record rule {n} (r_lhs : spl n) :=
  mkRule
    { r_dof : nat
    ; r_constraint : nary Prop r_dof (* r_dof-ary predicate *)
    ; r_rhs : rule_rhs n r_dof r_constraint
    ; r_gen : list (gen_pair r_dof) (* every possible combination of parameters *)
    ; r_correct : rule_correct _ r_constraint r_lhs r_rhs (* correct w.r.t constraint *)

(* Generation produces every correct set of parameters *)
    ; r_gen_sound : Forall (constraint_holds r_constraint) (r_gen)
    ; r_gen_complete : ∀ p, (constraint_holds r_constraint p) → In p r_gen
    }.
```

Formal Verification

Cooley-Tukey Rule

$$\mathrm{F}_{2^n} \equiv (\mathrm{F}_{2^r} \otimes \mathrm{I}_{2^s})\mathrm{T}^{2^r 2^s}_{2^s}(\mathrm{I}_{2^r} \otimes \mathrm{F}_{2^r})\mathrm{L}^{2^r 2^s}_{2^r}$$

- Two degrees of freedom: r and s
- Constraint: $n = r + s \wedge r \neq n \wedge s \neq n$
- Parameterized right-hand-side on r, s, and proof that constraint is satisfied
- Proof of equivalence for all such constrained r and s.
- A way to generate pairs (r, s)
- Proof that any such pair satisfies constraint
- Proof that all such pairs are generated

Formal Verification Effort

```
Program Definition cooley_tukey_rule n : rule F_{n} :=
  {| r_dof := 2
  ; r_constraint := (fun (r s: nat) ⇒ r+s = n ∧ r ≠ n ∧ s ≠ n)
  ; r_rhs := (fun r s (H : _ ) ⇒
                   (((F_{r} ⊗ I_{s})
                       × T_{r, s}
                       × (I_{r} ⊗ F_{s})
                       × L_{r, s}) : { (proj1 H) }))
  ; r_gen := sum_gen n (*  list (nat*nat)  *)
  |}.
Next Obligation.
(* Proving r_correct : rule_correct _ r_constraint
  i.e.  ∀ r s (H: r_constraint r s),
  F_{n} ≡ r_rhs r s H *)
...
Defined.
```

Formal Verification

Algorithm Generation

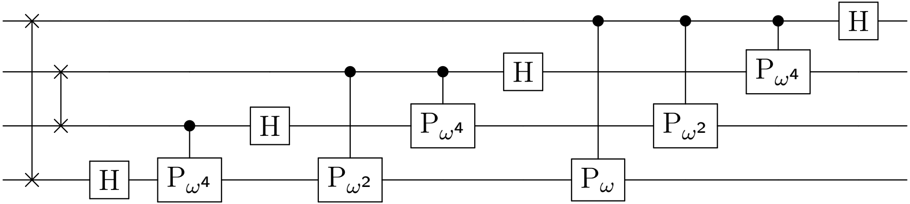
- Fueled search for all algorithms
- Single Step
  - Applying a rule to an applicable sub-expressions produces a list of results
  - Recursively rewrite subexpressions and concatenate results
- Parameterized rewrite rules produce alternative algorithms
- Fuel needed is roughly the number of qubits plus a small constant

Formal Verification

Coq Development metrics

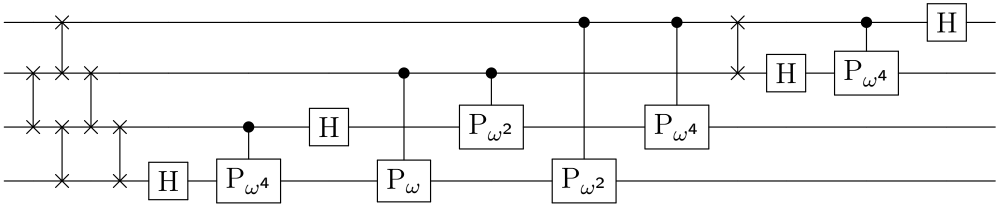
- 9167 loc
- 112 Definitions, mostly non-recursive
- 541 lemmas
- Roughly half of the actual proving dedicated to digit-permutations and additional supporting linear algebra

Example breakdown: 4-qubit QFT using the iterative radix-2 factorization



$$
\begin{aligned}
&\mathrm{QFT}_{16}\\
&= (\mathrm{H} \otimes \mathrm{I}_8)(\mathrm{C}(\mathrm{P}_{\omega^4} \otimes \mathrm{I}_4))(\mathrm{C}(\mathrm{I}_2 \otimes \mathrm{P}_{\omega^2} \otimes \mathrm{I}_2))(\mathrm{C}(\mathrm{I}_4 \otimes \mathrm{P}_{\omega}))\\
&\times (\mathrm{I}_2 \otimes \mathrm{H} \otimes \mathrm{I}_4)(\mathrm{I}_2 \otimes \mathrm{C}(\mathrm{P}_{\omega^4} \otimes \mathrm{I}_2))(\mathrm{I}_2 \otimes \mathrm{C}(\mathrm{I}_2 \otimes \mathrm{P}_{\omega^2}))\\
&\times (\mathrm{I}_4 \otimes \mathrm{H} \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{CP}_{\omega^4})\\
&\times (\mathrm{I}_8 \otimes \mathrm{H})(\mathrm{I}_2 \otimes \mathrm{SWAP} \otimes \mathrm{I}_2)\mathrm{SWAP}_{0,3}
\end{aligned}
$$

Example breakdown: 4-qubit QFT using the recursive radix-4 factorization



$$
\begin{aligned}
&\mathrm{QFT}_{16} \\
&= (\mathrm{H} \otimes \mathrm{I}_8)(\mathrm{CP}_{\omega^4} \otimes \mathrm{I}_4))(\mathrm{I}_2 \otimes \mathrm{H} \otimes \mathrm{I}_4)(\mathrm{SWAP} \otimes \mathrm{I}_4) \\
&\times (\mathrm{C}(\mathrm{I}_2 \otimes \mathrm{P}_{\omega^4} \otimes \mathrm{I}_2))(\mathrm{C}(\mathrm{I}_4 \otimes \mathrm{P}_{\omega^2})(\mathrm{I}_2 \otimes \mathrm{C}(\mathrm{P}_{\omega^2} \otimes \mathrm{I}_2))(\mathrm{I}_2 \otimes \mathrm{C}(\mathrm{I}_2 \otimes \mathrm{P}_{\omega})) \\
&\times (\mathrm{I}_4 \otimes \mathrm{H} \otimes \mathrm{I}_2)(\mathrm{I}_4 \otimes \mathrm{CP}_{\omega^4})(\mathrm{I}_8 \otimes \mathrm{H})(\mathrm{I}_4 \otimes \mathrm{SWAP}) \\
&\times (\mathrm{I}_2 \otimes \mathrm{SWAP} \otimes \mathrm{I}_2)(\mathrm{SWAP} \otimes \mathrm{SWAP})(\mathrm{I}_2 \otimes \mathrm{SWAP} \otimes \mathrm{I}_2)
\end{aligned}
$$

Summary and Future Work

Summary

- Formally verified (in Coq)...
  - derivation of quantum algorithms (SPL)
  - and the resulting circuits (SQIR)
  - using matrix semantics
- Generated alternative algorithms for future exploration and optimization

Future Work

- Gate optimization
- Quantum Measurements
- Additional Quantum Algorithms

References

- *The Coq proof assistant reference manual*. The Coq development team. 2022.
- "A Verified Optimizer for Quantum Circuits". Kesha Hietala et al. 2021.
- *QuantumLib*. QWIRE Team. 2022.
- "SPIRAL: Extreme Performance Portability". F. Franchetti et al. 2018.
- *Optimized Quantum Circuit Generation with SPIRAL*. Scott Mionis. 2021.