PHASE ACCUMULATION IN QRISP QUANTUM DICTIONARY SYNTHESIS

Raphael Seidel, **Sebastian Bock**, **Nikolay Tcholtchev**, **Manfred Hauswirth**, July 23, 2023

Fraunhofer

Wir machen
Städte schlau

1. Motivation

Fraunhofer

QUANTUM ALGORITHM DEVELOPMENT

- Quantum **hardware** (especially quantum volume) grew **faster than exponentially** last year.
- Quantum **algorithm discovery** and use-case identification need to **catch up**!

Fraunhofer

A CRITICAL PIECE OF THE PUZZLE

- Hoefler, Häner & Troyer in[1]:
  *"A large range of problem areas [...] such as many current **machine learning training** approaches, accelerating **drug design** and protein folding with Grover's algorithm, speeding up **Monte Carlo** simulations through quantum walks, as well as more traditional scientific computing simulations including the solution of many non-linear systems of equations, such as **fluid dynamics** in the turbulent regime, weather, and **climate simulations** will not achieve quantum advantage with **current quantum algorithms** in the foreseeable future.*

---

[1] Hoefler, Häner and Troyer. 2023. Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage.

A CRITICAL PIECE OF THE PUZZLE

Hoefler et al. conclude:

- Algorithms in many proposed cases of application are simply **not viable**.
- **Blackbox** approaches like Grover unlikely to yield practicality.
- Road to quartic (and higher) speedup lies in abusing **problem structure**.

⇒ Quantum developers need to be as **versatile**, **fast** and **specialized** (modular code!) as their classical equivalent!

QUANTUM ALGORITHM DEVELOPMENT

However: **Algorithm development via manual circuit construction is literally the slowest, least modular and most unstructured approach!**
⇒ Finding the right programming abstractions will be an important part in achieving quantum advantage for many fields of application.

QRISP

- Qrisp is a **fully compilable**, high-level programming framework[a].
- Central building block is the **QuantumVariable**.
- **Significantly enhances development aspects** like prototyping, code size, maintainability, bug-fixing/testing, modularity, readability, refactoring etc.

---

[a] www.qrisp.eu, Setup: `pip install qrisp`

QUANTUM PHASE ESTIMATION IN QRISP

Detailed introduction is **out of scope**. Instead we demonstrate a short quantum phase estimation implementation:

```
from qrisp import QuantumFloat, control, QFT, h
def QPE(psi, U, precision):
    res = QuantumFloat(precision, -precision)
    h(res)
    for i in range(precision):
        with control(res[i]):
            for j in range(2**i):
                U(psi)
    return QFT(res, inv = True)
```

Fraunhofer

2. QuantumDictionaries

Fraunhofer

QUANTUM DICTIONARIES

- The `QuantumDictionary` is a Qrisp data structure, which enables developers to load arbitrary **non-algorithmic data relations** in superposition

- Let qd be a **mapping/dictionary** of arbitrary (finite) sets

$$\mathrm{qd} : M \rightarrow N, x \rightarrow qd[x] \tag{1}$$

The **unitary** of the corresponding `QuantumDictionary` acts as

$$U_{\mathrm{qd}} |x\rangle |0\rangle = |x\rangle |\mathtt{qd[x]}\rangle \tag{2}$$

Fraunhofer

QUANTUM DICTIONARIES

- **Flexible** tool for algorithm design.
- Based on quantum logic synthesis $\Rightarrow$ **Scales rather bad** compared to more specific data-processing.
- Found **application in our TSP solution**[a] (x4 speed-up compared to QPE based, approach by Srinivasan et al.).

---

[a] www.qrisp.eu

QUANTUM DICTIONARIES IN SOLVING TSP

```
def calc_travel_distance(itinerary, precision, adjacency_matrix):
    from qrisp import QuantumFloat, QuantumDictionary
    res = QuantumFloat(precision, -precision)
    qd = QuantumDictionary(return_type = res)
    n = len(itinerary)
    for i in range(n):
        for j in range(n):
                qd[i, j] = adjacency_matrix[i, j]
    for i in range(n):
        trip_distance = qd[itinerary[i], itinerary[(i+1)%n]]
        res += trip_distance
        trip_distance.uncompute()
    return res
```

3. QuantumDictionary Compilation

QUANTUMDICTIONARY COMPILATION

- *QuantumDictionaries* are an inheritor of the regular Python dictionary and can be thought of as a **set of key/value pairs**.

- To **compile the loading procedure** from a *QuantumDictionary*, we follow the following protocol:
    1. Pick an integer **labeling function** for the elements of the key/value set.
    2. For each **key**, identify the label in binary. Do the same for each **value**.
    3. Stack the previously identified bit-strings to form a **truth table**.
    4. Load the truth table values using **quantum logic synthesis**.

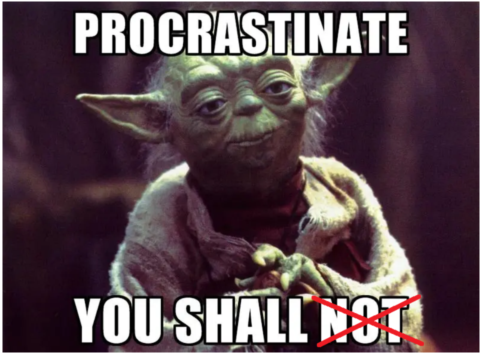
EXAMPLE TRUTH TABLE

x	$qd[x]$	$I_M(x)$	$I_N(qd[x])$	$I_M(x)_i$			$I_N(qd[x])_i$				
0	c	0	2	0	0	0	0	0	0	1	0
0.5	g	1	6	0	0	1	0	0	1	1	0
1	f	2	5	0	1	0	0	0	1	0	1
1.5	u	3	20	0	1	1	1	0	1	0	0
-2	k	4	10	1	0	0	0	1	0	1	0
-1.5	a	5	0	1	0	1	0	0	0	0	0
-1	c	6	2	1	1	0	0	0	0	1	0
-0.5	u	7	20	1	1	1	1	0	1	0	0

4. Phase accumulation

PHASE ACCUMULATION

- **Phase accumulation** is a novel technique for quantum logic synthesis of **truth tables with many columns**.
- Based on **Gray-code traversal**.
- Idea: Delay **phase clean-up** of each truth-table column until the end and perform **accumulated correction** instead.



INTERLUDE: GRAY CODE TRAVERSAL

- Gray code traversal is an algorithm that goes through a given set of **parity operators** (eg. $x_0 \oplus x_1 \oplus x_{42}$) and applies an **RZ-gate** to each of those.

- For an **arbitrary phase function** ϕ Gray code traversal returns a quantum circuit acting as

$$U_{gray}(\phi)\,|x\rangle = \exp(i\phi(x))\,|x\rangle \tag{3}$$

- Can be used to synthesize an **arbitrary single column, n-bit truth-table** T by wrapping the target qubit in H-gates and choosing

$$\phi(j) = \begin{cases} \pi & T(j_0, j_1..j_{n-1}) \wedge j_n \\ 0 & \text{else} \end{cases} \tag{4}$$

Fraunhofer

INTERLUDE: PERMEABILITY

Previous work of ours:

Definition.
An n-qubit unitary $U \in SU(2^n)$ is permeable on qubit i if and only if

$$Z_i U = U Z_i. \tag{5}$$

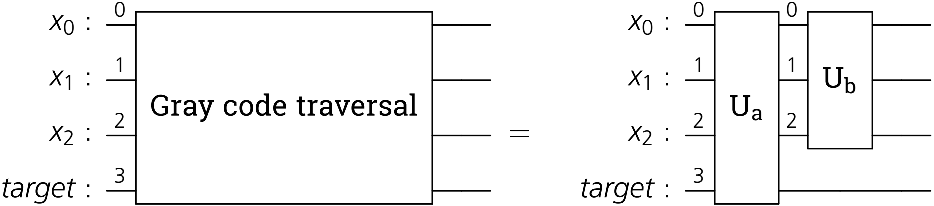
Where Z_i is the Pauli Z Operator of qubit i.

Important implication: Two unitaries U, V **commute** if they only **intersect on permeable qubits**. Enables DAG representation of quantum circuits abstracting non-trivial commutation relations.

GRAY-CODE-TRAVERSAL DECOMPOSITION

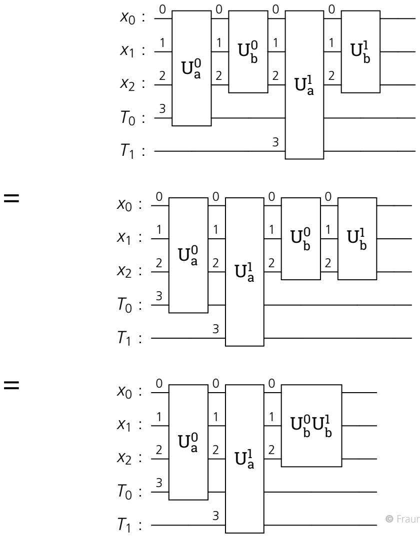
- In another previous work, we demonstrated that any n-qubit Gray code traversal circuit can be **decomposed into two operators** $U_a \in SU(2^n)$ and $U_b \in SU(2^{n-1})$.

- Both U_a and U_b are **permeable** on all inputs.



MULTI COLUMN TRUTH TABLE SYNTHESIS

- Naive approach to multi-column truth table synthesis: Synthesize single column truth-table **sequentially**.
- Our work: Permute U_b operators to the **end of the circuit** and perform **accumulated phase correction** using Gray-code traversal.



PERFORMANCE - GATE COUNT

- Consider a multi-column truth table

$$T : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m, x \rightarrow T(x)$$

- **Worst case** CNOT/RZ gate count:

$$\#\text{CNOT}(\text{Gray synthesis}(T)) = m2^{n+1} - (m-1)2^n$$

PERFORMANCE - CIRCUIT DEPTH

- Circuit depth could be estimated similarly, BUT:
  High amount of **idle time** for control qubits.

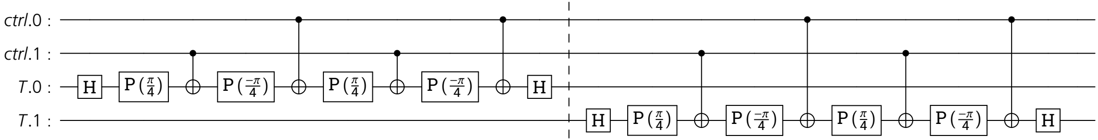
- Qrisp compiler performs **automatic parallelization** of different U_a operators.

- Based on **steered linearization** of the previously mentioned **permeability DAG**.
  This technique is not restricted to the use-case at hand. More details will be published soon.

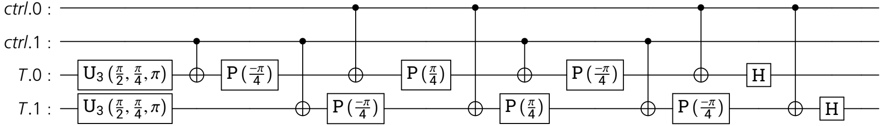
- For $m \ll n$: **Small depth overhead** for additional truth-table columns.

EXAMPLE: U_A PARALLELIZATION

Without parallelism: **Depth 15**

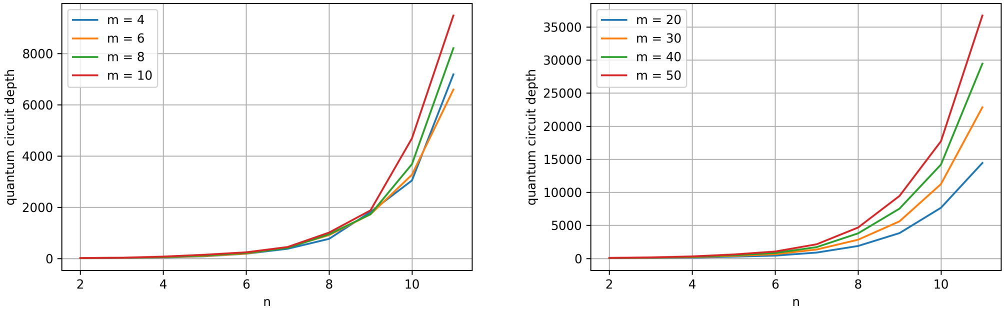


With parallelism: **Depth 10**



Fraunhofer

PERFORMANCE - CIRCUIT DEPTH



5. Summary

SUMMARY

- **Abstract programming** will play an important role in achieving **quantum advantage** outside factoring and quantum chemistry.
- Qrisp is a **high-level programming language**, which contains **state of the art compilation routines**, yet provides an accessible user interface.
- **QuantumDictionaries** are a Qrisp data-structure, that permit developers to include **real-world data** in their quantum algorithms.
- Their compilation is based on **quantum logic synthesis**, for which we presented an effective technique for **resource optimization**.

QUESTIONS?

grisp.eu