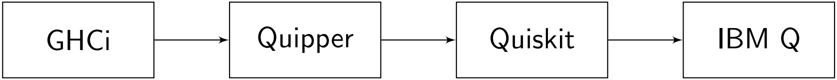
Towards Quantum Program Calculation

Ana Neri / José N. Oliveira

HASLAB / INESC TEC U. Minho

IBM QuantaLab Quantum Computing School, 2018

Tool-Chain



- **GHCi** - depending on the number of resources available in the target hardware, the monadic *quantamorphisms* are used to generate finite, *unitary matrices*;
- **Quipper** - this tool generates *quantum circuits* from the unitary matrices;
- **QISKit**[1] - the quantum circuit generated by Quipper is passed to this Python interface;
- **IBM-Q** - the actual code generated by *QISKit* runs on the actual, physical quantum device.

[1]Version < 6.0.

GHCi

Two kinds of quantamorphisms:

- For-loops (*qfor*) – controlled by natural numbers
- "Folds" (*qfold*) — controlled by finite lists

For quantum circuit generation we need to define a finite support for the matrix we want to generate and pass along to Quipper.

First example — a *qfor* with control qubits $\{00, 01, 10, 11\}$ — 3 cicles at maximum.

The matrix for qfor X and mqfor H is given by $f : \mathbb{B}_3 \times \mathbb{B} \to \mathbb{B}_3 \times \mathbb{B}$.

GHCi

The in a completely classic program the code implemented in Haskell is:

```
qfor ::  (b -> b) -> (Int, b) -> (Int, b)
qfor f (0,b) = (0,b)
qfor f (n+1,b) = let (m,b') = qfor f (n, f b) in (m+1,b')
```

When using a quantum gate it is important to use a monadic implementation:

```
mqfor ::  (Monad m) => (b -> m b) -> (Int, b) -> m (Int,
b)
mqfor f (0,b) = return (0,b)
mqfor f (n+1,b) = do  b' <- f b ; (m,b'') <- mqfor f (n,
b'); return (m+1,b'')
```

The **monadic encodings** of quantamorphisms given above are in one-to-one correspondence with **unitary matrices** describing quantum computations.
Running such monadic functions is a form of **simulating** such computations.

GHCi

Quantamorphism qfor X :

$$
\begin{array}{r|cccccccc}
 & \textit{(0,False)} & \textit{(0,True)} & \textit{(1,False)} & \textit{(1,True)} & \textit{(2,False)} & \textit{(2,True)} & \textit{(3,False)} & \textit{(3,True)} \\
\hline
\textit{(0,False)} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{(0,True)} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{(1,False)} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\textit{(1,True)} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\textit{(2,False)} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\textit{(2,True)} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\textit{(3,False)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\textit{(3,True)} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
\tag{1}
$$

Quantamorphism mqfor H :

$$
\begin{vmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}
\end{vmatrix}
\tag{2}
$$

Quipper

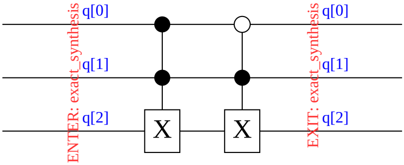


Figure: Quipper circuit resulting from matrix of the *quantamorphism* for X .

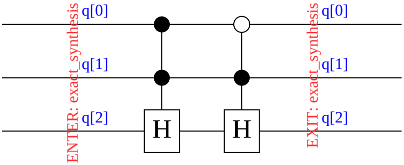


Figure: Quipper circuit resulting from matrix of the *monadic quantamorphism* for H .

Quipper

Decompostion & Translation "QuipperToQiskit"[2]

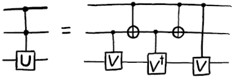
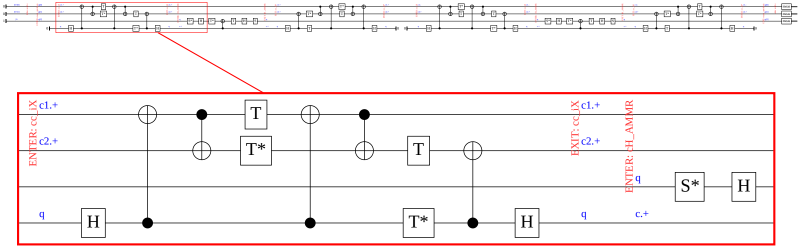


Figure: Decomposing our gates is not trivial, thus we choose let Quipper do the decomposition of the second case. (Figure from [tea18]).



[2]public tool [NR18b].

QISKit

Implementation

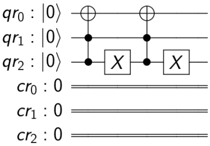


Figure: Circuit from qfor X .

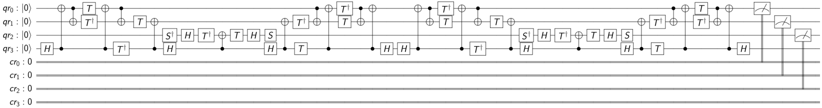


Figure: Circuit from mqfor H .

QISKit

Why swap q_0 with q_2 in qfor X ?

- Coupling map of ibmqx4.

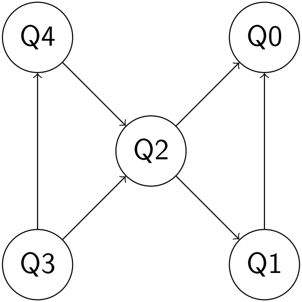


Figure: Coupling map of IBM Q 5 Tenerife V1.x.x (`ibmqx4`).

QISKit

Decomposition to QASM simulator - qfor X

The circuit from qfor X still needs decomposition to run a simulator:

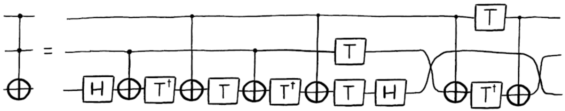


Figure: Decomposition of a Toffoli gate (Figure from [tea18])

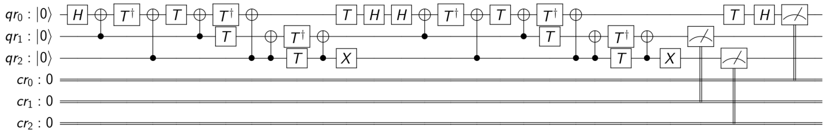


Figure: Circuit from qfor X .

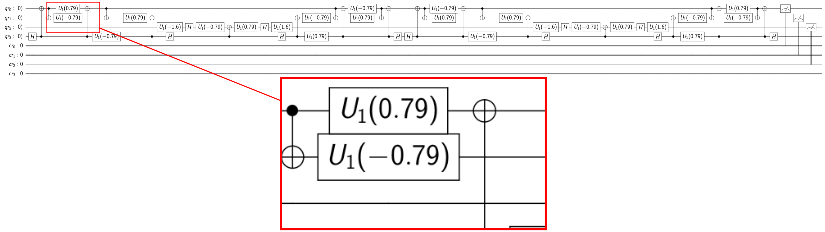
QISKit

Decomposition to QASM simulator - mqfor H

When running in IBM Q the gates T, $T^\dagger$, S, $S^\dagger$, Z and H are implemented with unitary gates:

$$U(\theta,\phi,\lambda) = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i\lambda+i\phi}\cos\frac{\theta}{2} \end{bmatrix} \tag{3}$$

IBM Q has $u_1(\lambda) = U(0,0,\lambda)$, $u_2(\phi,\lambda) = U(0,\phi,\lambda)$ and $u_3(\theta,\phi,\lambda) = U(\theta,\phi,\lambda)$.



QISKit

Expected results - QISKit simulations

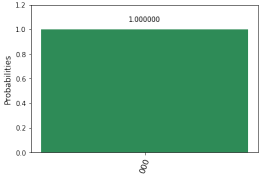


Figure: Circuit from qfor X with the initial state at $|000\rangle$.

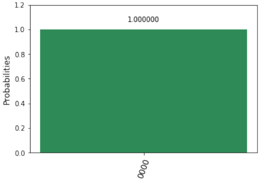


Figure: Circuit from mqfor H with the initial state at $|000\rangle$.

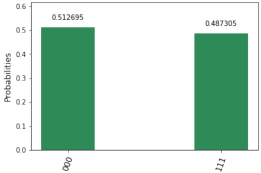


Figure: Circuit from qfor X where the initial state of the control qubits is bell state.

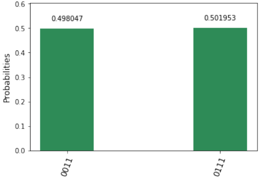


Figure: Circuit from mqfor H with the with the control qubits at $|11\rangle$.

QISKit

Decompostion to ibmqx4 device

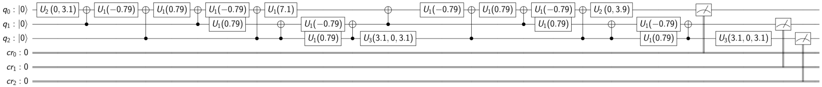


Figure: Circuit from qfor X

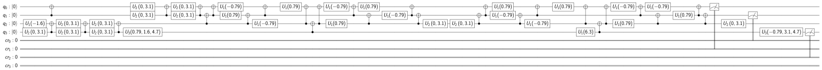


Figure: Circuit from mqfor H

QISKit

Results - execution in ibmqx4

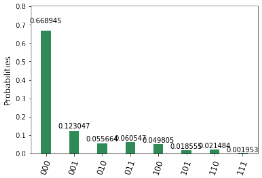


Figure: Circuit from qfor X with the initial state at $|000\rangle$.

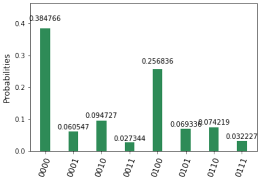


Figure: Circuit from mqfor H with the initial state at $|000\rangle$.

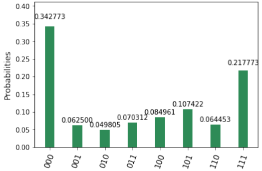


Figure: Circuit from qfor X where the initial state of the control qubits is bell state.

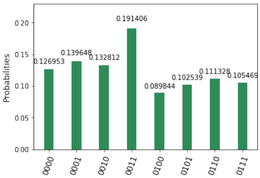


Figure: Circuit from mqfor H with the with the control qubits at $|11\rangle$.

QISKit

Why the errors?

- Number of gates;
- There may be some bugs in decomposing to a specific device.

The simulations are made to work in a very similar fashion to the quantum devices which leads to the conclusion that the theoretical work is correct, but the devices still have a lot to improve.[3]

[3] Details in [NR18a].

QISKit

Important notes

Fortunately, there is evidence that improvements are happening fast:

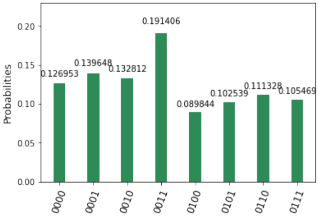


Figure: The output of mqfor H with control qubits in $|1\rangle$ in ibmqx4 device.

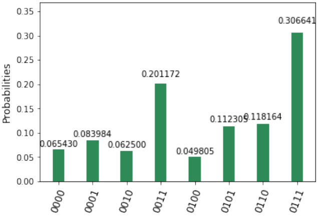


Figure: The output of mqfor H with control qubits in $|1\rangle$ in ibmq_20_tokyo device.

Conclusions

- The **tool-chain** is useful but can still be improved;
- Our programs are **correct by construction**;
- The error rate is high but has been decreasing a lot.

References I

A. Neri and A. Rodrigues, *Quantamorphisms*, 2018, GitHub project, https://github.com/arcalab/quantamorphisms. (Accessed on 02/10/2018).

______, *Quippertoqiskit tool*, 2018, GitHub project, https://github.com/arcalab/quantamorphisms/tree/master/Tool-QuipperToQiskit. (Accessed on 02/10/2018).

IBM Q team, *Ibm q experience*, https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=introduction, Out 2018, (Accessed on 10/23/2018).

Project finance by INESC TEC.