CertiQ: A Push-button Verification Framework for the Qiskit Quantum Compiler

Abstract

Quantum computing is at the cusp of revolutionizing the society(to janet: or something similar, like "bringing transformative change to the scientific and commercial world". Want to emphasis that quantum computing was a theory on paper several years ago, but now it is really making impact to the world, then the next sentence "quantum cloud service is emerging" will support it). Recently, cloud-based quantum software stack are emerging, provided platforms for the first quantum applications and numerous new scientific discoveries. To guarantee the correctness of the quantum software platform is crucial but a grand challenge. This paper describes CertiQ — a push-button(keep?) verification framework for the Qiskit terra quantum compiler. Qisikit terra is the foundation of the Qiskit open-source framework for IBM Q-experience, the most complete and widely used quantum cloud-based service now. To overcome the state explosion problem with conventional simulation-based verificaiton, CertiQ adapts certified abstract layer(CAL), a novel verification techniques to decouple the underlying quantum complexity from higher level verification. CertiQ not only is deeply integrated with critical components exists in Qiskit terra, but also provides scalable, fully-automated ceritification to future code submission.

We present an extensive evaluation of CertiQ over existing and newly-implemented quantum compiler passes and demonstrate that it can effectively prove the correctness of real-world quantum compilation tools and discover critical bugs.

Introduction

This paper makes the following contributions:

- On the quantum/practical side, we provide the first automated verification framework for a complete quantum compiler, enabling us to:
  - greatly reduce the cost of code review process for the open source Qiskit terra compiler, which has more than 100 contributors in total and more than 50 contributors from third-party institutes.
  - provide guidelines for constructing large scale, industrial level quantum software, as quantum software engineering is a developing field in its early stage.
  - find existing bugs in the Qiskit terra compiler, which could potentially affect over tens of publications and millions simulations on the cloud every year.
- On the verification side, we largely extend previous work [3, 4] on push-button verification framework by introducing a combination of proof techniques:
  - we build our proof system with certified abstraction layers, encapsulating statically available quantum logic and exposing only useful lemmas to the layer above to decouple verification complexity.
  - We build our proof system upon existential theory of nonlinear real arithmetic in addition to linear arithmetic theory in previous work to verify the underlying quantum process.
  - By carefully designing the verification interface, we are able to adapt a rich variety of proof primitives such as loop invariants and Z3 DAG data structure while staying push-button, as opposed to basic theory of arrays used by previous work. These proof primitives further reduce verification complexity.

Overview

Qiskit

Certiq

Implementation

Certified quantum data structure

3.1.1. Equivalence of quantum circuit and quantum DAG

3.1.2. Validation of Bloch sphere representation in 1-qubit case Single qubit rotations are often depicted in the Bloch sphere picture [1]. The Bloch sphere description is a projection where the global phase of a qubit state $|\psi\rangle$ is omitted and $|\psi\rangle$ and $e^{i\phi}|\psi\rangle$ are mapped to the same qubit state(known as Hopf fibration [5]). This description is valid in 1-qubit case, however, is no longer applicable to 2-qubit case since it will lose the relative phase information.

Certified pass manager

Certified transpiler passes

Translational validator

Case studies

`optimize_1q_gate` pass

The `optimize_1q_gate` transpiler pass finds and merges consecutive unrolled single qubit gates (*i.e.* u_1, u_2, u_3 gates) to improve circuit latency. The examples in fig. 1 illustrated the effect of this pass on several basic unrolled circuits.

$$-\boxed{u_1(\lambda_1)}-\boxed{u_1(\lambda_2)}- \xrightarrow{\texttt{optimize\_1q\_gate}} -\boxed{u_1(\lambda_1+\lambda_2)}-$$

(a)

$$-\boxed{u_1(\lambda_1)}-\boxed{u_2(\frac{\pi}{2},\lambda_2)}- \xrightarrow{\texttt{optimize\_1q\_gate}} -\boxed{u_2(\lambda_1+\frac{\pi}{2},\lambda_2)}-$$

(b)

$$-\boxed{u_1(\lambda_1)}-\boxed{u_3(\theta_2,\phi_2,\lambda_2)}- \xrightarrow{\texttt{optimize\_1q\_gate}} -\boxed{u_3(\theta_2,\lambda_1+\phi_2,\lambda_2)}-$$

(c)

Figure 1: Example input/output of `optimize_1q_gate`

To efficiently combine the rotation along different axis, the `optimize_1q_gate` pass merges unrolled single qubit gates with Quaternion algebra [2]. With the following Quaternion-rotation mapping,

$$\vec{R} = \begin{bmatrix} 1-2(q_j^2+q_k^2) & 2(q_iq_j-q_kq_r) & 2(q_iq_k+q_jq_r) \\ 2(q_iq_j+q_kq_r) & 1-2(q_i^2+q_k^2) & 2(q_jq_k-q_iq_r) \\ 2(q_iq_k-q_jq_r) & 2(q_jq_k+q_iq_r) & 1-2(q_i^2+q_j^2) \end{bmatrix}$$

Conclusion

Acknowledgement

References

[1] F Bloch. Nuclear Induction. *Phys. Rev.*, 70(7-8):460–474, oct 1946.

[2] Philippe Gille and Tamas Szamuely. *Central Simple Algebras and Galois Cohomology.* 2009.

[3] Luke Nelson, Helgi Sigurbjarnarson, Kaiyuan Zhang, Dylan Johnson, James Bornholt, Emina Torlak, Xi Wang, and Dylan John-son. Hyperkernel: Push-Button Verification of an OS Kernel. page 18, 2017.

[4] Helgi Sigurbjarnarson, James Bornholt, Emina Torlak, and Xi Wang. Push-Button Verification of File Systems via Crash Refinement. In *the Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI*, 2016.

[5] H K Urbantke. The Hopf fibration-seven times in physics. 46:125–150, 2003.