QC Bootcamp Project 1: Quantum Calculator

Saket Shah

May 29, 2025

The project which this document describes in Qiskit code the following project:

Project 1. *Let $d \geq 1$ be any positive integer. Produce a quantum circuit QCalc on $3d+1$ qubits defined by on the computational basis by*

$$\mathrm{QCalc}|x\rangle_d |y\rangle_d |z\rangle_1 |0\rangle_d = \begin{cases} |x\rangle_d |y\rangle_d |z\rangle_1 |x+y \mod 2^d\rangle_d & \text{if } z = 0 \\ |x\rangle_d |y\rangle_d |z\rangle_1 |xy \mod 2^d\rangle_d & \text{if } z = 1, \end{cases}$$

where we interpret a computational basis element $|x\rangle_d = |x_{d-1}x_{d-2}\cdots x_0\rangle$ as the binary integer $\sum_{i=0}^{d-1} x_i 2^i$ in the usual way.

More generally, this implementation will behave on inputs as

$$\mathrm{QCalc}|x\rangle_d |y\rangle_d |z\rangle_1 |w\rangle_d = \begin{cases} |x\rangle_d |y\rangle_d |z\rangle_1 |w+x+y \mod 2^d\rangle_d & \text{if } z = 0 \\ |x\rangle_d |y\rangle_d |z\rangle_1 |w+xy \mod 2^d\rangle_d & \text{if } z = 1. \end{cases}$$

1 Implementation

The implementation is fundamentally based on Draper's algorithm [Dra00]. The idea is the following:

As in Draper addition, we apply a quantum Fourier transform to the d output qubits $|w\rangle_d$, and use the property that the transformation $|w\rangle \mapsto |w+k\rangle$ can be written as $|w\rangle \mapsto \mathrm{QFT}^\dagger \circ P_k \circ \mathrm{QFT}|w\rangle$, where $P_k|w\rangle = e^{2\pi i w k/2^d}|w\rangle$.

In this particular case, if we let $Q := \mathrm{id} \otimes \mathrm{QFT}_d$ denote the quantum Fourier transform taken on the output qubits, we observe that we can decompose QCalc as the composition

$$Q^\dagger \circ \Phi \circ Q,$$

where

$$\Phi|x\rangle_d|y\rangle_d|z\rangle_1|w\rangle_d = e^{2\pi i \cdot xyzw/2^d + 2\pi i\cdot(x+y)(1\oplus z)w/2^d}|x\rangle_d|y\rangle_d|z\rangle_1|w\rangle_d.$$

We break this down as a composition of two separate phase shifts (which will be further broken down): one by $e^{2\pi i\cdot xyzw/2^d}$ and the other by $e^{2\pi i\cdot(x+y)(1\oplus z)w/2^d}$.

For the first, we can write

$$x = \sum_{i=0}^{d-1} x_i 2^i,$$

and similarly for y and w; then the phase shift can be written as a product of phase shifts by $e^{2\pi i\cdot x_i y_j z w_k \cdot 2^{i+j+k}/2^d}$, which can each be independently implemented by a multi-controlled CCCPhase gate controlled on the ith x bit, jth y bit and z bit, scaling the kth w bit.

The second phase gate $e^{2\pi i\cdot(x+y)(1\oplus z)w/2^d}$ is similar; for simplicity we apply an X gate on the z bit which will be undone after the phase shift. Then we break this up as $e^{2\pi i\cdot x(1\oplus z)w/2^d}$ and $e^{2\pi i\cdot y(1\oplus z)w/2^d}$, which can

each be further broken up as a series of CCPhase gates controlled on the various x (or y) bits and the z bit, and scaling w bits. This is more directly given by the algorithm in Draper's paper, with an extra control on the z bit.

The CCPhase and CCCPhase implementations are given by decomposing Qiskit's own implementation in terms of controlled phase gates and CX gates, and use no ancillas. In broad strokes, the CX gates are used in the CCPhase and CCCPhase implementations to add in phase gates that run to cancel out the existing phases when not all bits are 1.

2 Analysis of complexity

In keeping with Draper's original algorithm, this algorithm makes use of no ancillas.

In the big-O sense, the main contribution to the algorithm's gate count and gate depth come from the multiplication step. The gate-count of the multiplication step has a gate count of $O(d^3)$ coming from the controlled phase gates running over the individual x_i, y_j, w_k qubits. The QFT and addition each have a gate count of $O(d^2)$. The gate depth of the multiplication step is $O(d^2)$; this can be seen by one of the x_i qubits and observing we still have $O(d)$ y_j and w_k qubits each. The gate depth of the addition step and the QFT are both $O(d)$.

In total, the algorithm has an $O(d^3)$ gate count and $O(d^2)$ gate depth.

References

[Dra00] Thomas G Draper. Addition on a quantum computer. *arXiv preprint quant-ph/0008033*, 2000.