# QUANTUM ARITHMETIC ROUTINES: IMPLEMENTATION, VERIFICATION, AND RESOURCE ESTIMATION

**STUDENTS:** KUSH AGGARWAL, THOMAS HUANG, CHRISTOPHER SHARP

## Motivation and Objective

- Quantum arithmetic routines provide a way to perform mathematical computation on inputs in superposition for e.g. Shor's algorithm. Our goal was to implement several arithmetic routines of different types into a Pythonic development framework, Workbench by PsiQuantum, as "Qubricks," then analyze their resource consumption to understand which performs better and in which contexts. We have done this for four addition and six multiplication algorithms.

## Project Background

- As quantum computing hardware is finalized, we must prepare for development of quantum software.
- Because quantum operations must be unitary, implementations must preserve enough information to be reversible; the destruction of information is fundamentally prohibited in quantum physics.
- Classical computing gates destroy information; for example, an AND gate returns 0 for bits 00, 01, and 10 so it is impossible to reconstruct the input.
- Quantum computing circuits consist of unitary quantum gates as primitives, and algorithm design prioritizes replacing classical gates with quantum gates, like the Toffoli (double-controlled NOT gate) gate for an AND gate.
- T-gates implement Toffoli gates; T-gates are very expensive because of their requirements for quantum error correction (Aliferis et al., 2005).
- Previous algorithms have been implemented and tested in other platforms like Q# but we would like to implement them in Workbench by PsiQuantum.
- Workbench allows Pythonic programming of quantum routines, with Qubricks implemented analogously to classes.
- At the same time, qubit count, gate cost, and circuit depth must be optimized.

## Methods – Algorithms

Adders are separated by whether they update an existing register (in-place) or store the result in a new sum register (out-of-place)

In-place adders

- TTK – Utilizes no additional qubits at the tradeoff of higher circuit complexity (Wang et al., 2016)

Multipliers

- Schoolbook (with TTK adder) – Multiplies one digit at a time and shifts the multiplication over one place at a time and adds them together, like is commonly taught in school.
- JHHA implements, instead of an add and shift multiplication, an add and rotate multiplication to improve efficiency (Jayashree et al., 2016).

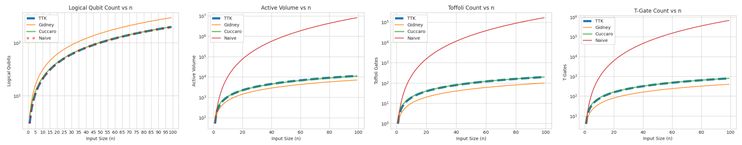
Out-of-place adders

- CT – Scales 1-qubit addition to *N*-qubit addition (Cheng, 2002)
- Wang – Designed specifically for minimizing complexity of the quantum circuit in exchange for increasing the amount of qubits used in memory (Wang et al., 2016)
- Gayathri – Designed to reduce the complexity of the quantum circuit by reducing the number of expensive Toffoli gates per qubits (Gayathri et al., 2021)

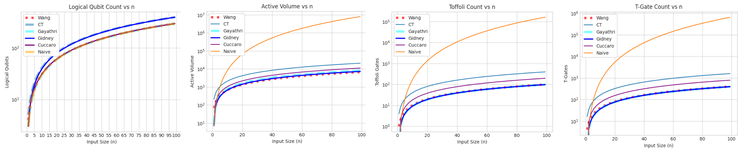
## Results and Analysis

We compare our algorithms – the adders TTK, CT, Wang, and Gayathri, and the multipliers JHHA and schoolbook with TTK – against exiting, pre-implemented arithmetic algorithms in Workbench: Gidney, Cuccaro, and "Naive". We measure number of qubits, active volume, Toffoli(-equivalent) count, and T-gate count.

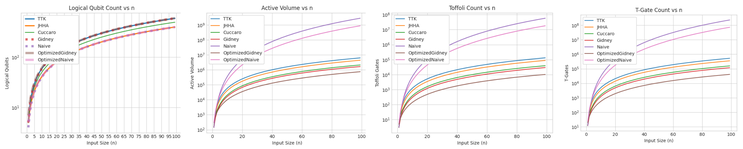
In-place adders




Out-of-place adders




Multipliers




Multiplication has much more complexity because we naively would expect quadratic growth input size, and more optimized algorithms try to improve on that. Accordingly, we see that "naive" arithmetic here has by far the greatest complexity of implementation, forcing our use of the logarithmic vertical axis scale to make the other algorithms visible. Among in-place adders, our TTK implementation performs comparably with pre-implemented Cuccaro adder. Among out-of-place adders, our CT implementation expectedly performs worse than other non-naive algorithms (being an early algorithm), while Gayathri and Wang perform comparably to the pre-implemented Cuccaro and Gidney adders. Among multipliers, JHHA performs somewhat better than TTK-based schoolbook, and both perform worse than pre-existing algorithms, although they are being compared in part to optimized multipliers.

## References

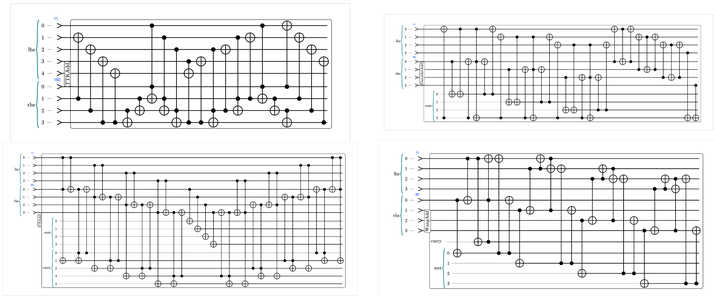

- Aliferis, P., Gottesman, D., and Preskill, J. "Quantum accuracy threshold for concatenated distance-3 codes." 2005.
- Caesura, A., Glemmestad, E., Gover, L., Greenaway, S., Gui, K., Harvey, C., Heavey, S., Jałowiecki, K., Johnston, E., Lemieux, J., Litteken, A., Morley-Short, S., Mykhailova, M., Okrut, O., Pallister, S., Pol, W., Russo, V., Sim, S., Simon, W., Stęchły, M., Uchehara, G., & Vincent, T. (2026). PsiQDK (Version 2.0.0) [Computer software]. https://github.com/PsiQ/psiqdk/
- Cheng, K. and Tseng, C. "Quantum full adder and subtractor." November 2002. Electronics Letters. 38(22):1343 - 1344.
- Gayathri, S. S., Kumar, R., Dhanalakshmi, S., Dooly, G., & Duraibabu, D. B. (2021). T-count optimized quantum circuit designs for single-precision floating-point division. Electronics, 10(6), 703. https://doi.org
- Jayashree H. V. & Thapliyal, Himanshu & Arabnia, Hamid & Agrawal, V.. (2016). Ancilla-Input and Garbage-Output Optimized Design of a Reversible Quantum Integer Multiplier. The Journal of Supercomputing. 72. 10.1007/s11227-016-1676-0.
- Takahashi, Y., Tani, S., and Kunihiro, N. "Quantum addition circuits and unbounded fan-out." 14 October 2009.
- Wang, F., Luo, M., Li, H., Qu, Z., and Wang, X. "Improved quantum ripple-carry addition circuit." 2016. Sci. China Inf. Sci. 59, 042406


## Methods – Implementation and Circuits

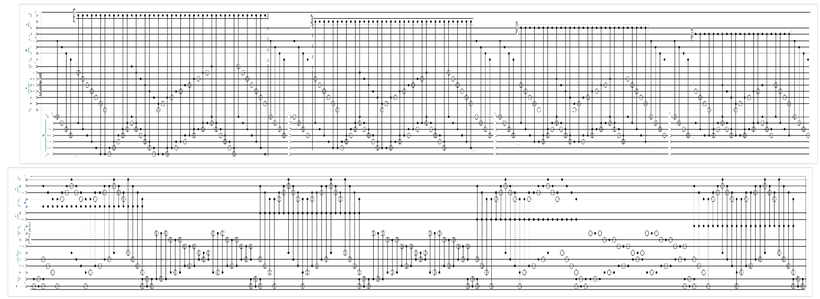
We implemented our arithmetic algorithms as Qubricks in Workbench, taking advantage of the platform's arithmetic-friendly "QUInt" data types, which are qubit registers that can be interacted with as unsigned integers. We integrated a CI testing pipeline. In particular, we had to make sure quantum registers returned to their original state after computation. We then ran Workbench's resource estimator to analyze performance.

**Qubrick API** | **QUInt registers** | **Unit tests** | **Uncompute** | **Estimator**

The platform supports rendering circuit diagrams for the algorithms, shared below.



Above are the circuit diagrams for our adders. Clockwise from top-left: TTK, Gayathri, Wang, and CT.



Above are the circuit diagrams for our multipliers. Top is Schoolbook with TTK Adder; bottom is JHHA.

## Future Work

- Implement new algorithms of existing arithmetic types like more in-place adders and Karatsuba multiplication, and new arithmetic types like division and modular multiplication
- Explore optimization techniques using the optimized Gidney and Naive multipliers as potential templates
- Streamline and standardize Qubricks such as through subtraction flags
- Contribute routines and analysis to enable more efficient quantum computation

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

**ADVISERS: MARIIA MYKHAILOVA**

**SPONSOR: PSIQUANTUM**

**INSTRUCTOR: SARA MOURADIAN**