Quantum Semiprime Factorization: Leveraging Grover's Algorithm for Efficient Prime Decomposition

Jacob Collins[1], Jaime Raigoza[1], Sam Siewert[1]

[1] California State University, Chico, United States

Abstract

This paper details some of the first steps that our research group has taken towards a practical demonstration of quantum advantage.

Grover's search algorithm can be leveraged to efficiently compute the inverse of many functions; this is addressed both from a general perspective, as well as specific applications, including inversion of multiplication, thereby finding the factors of semiprimes with greater efficiency than classical HPC methods and similar existing quantum algorithms.

RSA encryption relies on the fact that the multiplication of large prime numbers is a trap-door function- computationally trivial to compute the semiprime product, but classical methods of finding prime components is so computationally expensive that it can be considered nearly impossible at a large scale. This principle is the backbone of modern-day cybersecurity [1].

Key terms: *Semiprime, superposition, entanglement, Grover's algorithm, Shor's algorithm, quantum fourier transform, adjoint, RSA encryption, quantum advantage, RSA number, reversible function, trap-door function, quantum circuit, scaling, parallel, CUDA, CUDA-Q, sieve, time complexity.*

1. Introduction

The practicality of applied quantum computing is a topic of much debate, and there are not yet many examples of quantum computers being able to out-perform their classical counterpart, a feat known as quantum advantage.

Quantum algorithms have been in development at a theoretical level for decades, but the application of these methods is still in early stages. It is quite common to find code which implements a quantum algorithm like Grover's[2] or Shor's[3], but very often these programs are hard-coded, and only work for a small handful of specific values[4].

This paper describes the generalized use of Grover's algorithm as the inverse of some function $F(x, y) = z$, with demonstrated applications inverting the arithmetic functions addition and multiplication. Specifically, our implementation can generate and simulate a circuit to find the distinct prime factors of any given semiprime.

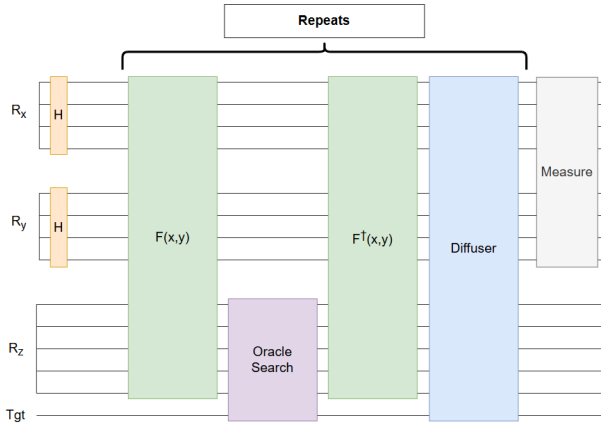

Figure 1. General circuit diagram for Grover's Algorithm, used to find the inputs x, y for which $f(x, y) = z$, for some known output z.

1.1. Goals

The greater objective of this research is to clearly demonstrate quantum advantage by showing clear speed-up to solve the same problem at the same scale.

While the semiprime factoring problem is just one algorithm where quantum circuits should have significant advantage over parallel factoring, the goal of this research is to concretely show that this particular problem is in fact a clear case of quantum advantage.

One major intention for this paper in particular is demonstration of the broad utility of Grover's algorithm. Furthermore, another long-term goal of this research is to identify the types of algorithms that likewise may have significant quantum advantage based upon reversible functions.

1.2. Motivation

Semiprime factoring is an ideal problem for demonstrating quantum advantage, as it is both challenging to compute classicaly, and has wide-reaching impact if efficient solutions are found.

Semiprime factoring is known to have a high time complexity due to the nature of prime number generation with a sieve (filters out all non-primes) and trial division search. This problem in particular becomes complex for large size semiprimes commonly used in encryption, which might be 4096 bits or more.

For example, the RSA number 250, which is 892-bits long, was only recently factored using a parallel computer with the best known SP factoring method known right now [5].

However, a quantum computer has a much lower time complexity if the number of qubits can be scaled to implement a quantum circuit that can factor a semiprime. Ongoing research at China State Key Laboratories suggests that they have been making significant progress in breaking RSA encryption, considering their 48-bit demo using 10 qubits in 2022 [6].

Furthermore, they claim that with the techniques they have been using, it would be possible to break RSA 2048 with a quantum circuit which needs only 372 qubits, far less than the 1,121 qubit machine currently available in the United States with IBM's Condor [7].

The two most significant limitations to applied quantum computing are qubit count and error rate. The algorithm displayed in this paper (and also by S. Whitlock, et al. [8]) demonstrates significant optimizations regarding the number of qubits required for semiprime factoring. It is important to note, however, that error rate increases dramatically in proportion to the number of gates applied, which may reduce the viability of these algorithms in practice. That being said, there are emerging quantum computing hardware solutions which claim to have far better error-control methods than have yet been observed, such as the Majorana I [9].

If it can be verified that SP factoring with Grover's algorithm uses significantly less qubits than Shor's, has less error, or has less overhead, then the application of quantum semiprime factoring at a meaningful scale may be feasible far sooner.

2. Literature Review

2.1. Grover's Algorithm

Grover's Algorithm is a quantum computing method that can be used to search a database of N values in $O(\sqrt{N})$ time rather than the naive classical time complexity of $O(N)$[2].

The exact number of iterations required varies on a case-by-case basis, but is typically expressed as follows: in a search for one matching input state, $\frac{\pi}{4}\sqrt{N}$ iterations are required, and in a search for 2 valid input states, $\lfloor \frac{\pi}{4} \cdot 2^{\frac{n_x+n_y}{2}} \rfloor$ iterations are required, where N is the size of the search domain, and n_x, n_y are the number of qubits needed to represent the prime factors x and y, respectively [8].

Given a function $f(x) = y$, where x is unknown (index, prime factors, sum components, etc.), and y is known (array value, semiprime/product, sum, etc.), Grover's algorithm effectively takes the role of $f'(y) = x$, allowing for a potential speedup in finding whatever input (s) to $f(x)$ will return y, provided that it is much faster to compute $f(x)$ than whatever classical methods might be used to otherwise solve for x given y.

This speedup comes from the fact that Grover's algorithm requires $O(\sqrt{y})$ iterations, each of which have a time complexity proportional to that of $f(x)$.

2.2. Shor's Algorithm

Shor's algorithm is the most well-known approach to prime factorization in quantum computing. While it is the fastest known quantum algorithm for factoring semiprimes, it requires (in most cases) $5n + 1$ qubits [10], which is too large for problems of a practical size [3, 8].

2.3. Quantum Factoring Algorithm with Grover Search

S. Whitlock et al. present a method of SP factoring with Grover's algorithm. The implementation in their paper is highly optimized, and modifies the target value in their circuit from some semiprime N, to M, a reduced number uniquely tied to N, which has unique factors p and q which can be used to calculate a, b, the prime factors of N [8].

While this approach is admirable, it introduces a level of complexity that may hinder a learner's understanding of the mechanics at work, so the implementation shown in section 3 forgoes this abstraction from N to M, and instead implements an oracle which searches directly for prime factors a and b of a semiprime N, without the additional pre-processing and post-processing steps.

3. Methodology

Quantum algorithms take advantage of superposition and entanglement, which enables many methods of computation which are otherwise impossible with a classical computer. For example, by placing a set of n qubits (otherwise known as a qubit **register**) in superposition, that register simultaneously represents every value which can be represented in n bits, until measured. Once measured, a

register in uniform superposition will collapse into one of these states with equal likelihood.

3.1. The Use of Grover's Algorithm for General Function Inversion

This use of Grover's algorithm involves three main parts: some function with one or multiple inputs ($F(x, y)$ and $F^{\dagger}(x, y)$), the oracle which filters for states in which the desired output was observed, and the diffuser, which increases the probability of observing inputs which match the previously marked state. An overview of this circuit is shown in Figure 1. The H operations on R_x and R_y are simply Hadamard operations, placing both registers in a uniform superposition.

The oracle in Grover's algorithm performs a controlled operation which is meant to differentiate states $-|x, y\rangle$ that do result in the wanted target z from states $|x, y\rangle$ that do not result in the target $|z\rangle$ by conditionally flipping the phase of the target qubit.

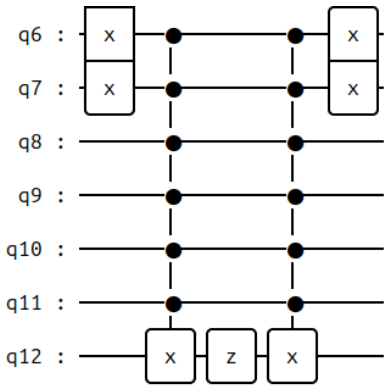

Figure 2. Example circuit diagram for Grover's oracle marking entangled states of the output register ($R_z = [q_6, q_{11}]$) via the target qubit (q_{12}), where $R_z = |15\rangle = |001111\rangle$.

Figure 2 shows an example of the oracle searching for the state $z = 15$, or rather, $R_z = |001111\rangle$. To implement a circuit which can selectively target $z = 15$, NOT gates are applied on $R_z[0]$ and $R_z[1]$ (seen in the figure as q_6, q_7), so that states where $R_z = |001111\rangle$ will then be in the state $R_z = |111111\rangle$, and the multi-qubit Toffoli gates[11] will apply a NOT operation to the target qubit only for the desired states where $z = 15$. The NOT gates are again applied to R_z, returning it to state $|001111\rangle$, while the target qubit retains the phase flip in those corresponding states.

To clarify, Toffoli gates have a target qubit (Seen as q_{12} in Figure 2, and 'Tgt' in Figure 1) and two control qubits which must both be in state $|1\rangle$ for a NOT operation to be applied to the target qubit. There are methods of combining multiple Toffoli gates to behave as a multi-qubit Toffoli gate, which behaves similarly, but can have any number of control qubits ($[q_6, q_{11}]$), instead of being limited to just two[11].

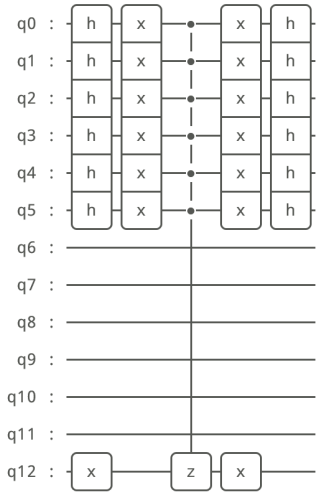

Figure 3. Example circuit diagram for a diffuser which amplifies the probability that measurement will result in a state which has been marked by the oracle. Input registers R_x and R_y ($[q_0, q_2]$ and $[q_3, q_5]$) are now more likely to be measured in a state such that $f(R_x, R_y)$ results in $R_z = |z\rangle$.

The diffuser in Grover's algorithm (Figure 3) can then increase the probability of observing $-|x, y\rangle$ (states which have been marked by the oracle) through a process known as inversion about the mean [2].

3.2. Arithmetic in the Quantum Fourier Domain

The quantum fourier transform, QFT, is a very useful operation, typically used to transform a qubit register from the computational basis (binary) to the fourier basis (phase). [12].

The inverse QFT operation, IQFT is used to return a qubit register R_x from the fourier basis to the computational basis, retaining the values which have been mapped and operations which have been applied to R_x.

To summarize, the QFT domain allows addition to be done through rotations in another basis, and these rotations can be scaled by some constant c, which in practice also scales the operand by c. Multiplication between values encoded in registers R_x and R_y can then be performed out-of-place into R_z by conditionally adding the full value encoded in R_x to R_z for each qubit q_i in R_y, such that each of these register-wide additions are performed as controlled operations which will only occur if $q_i = |1\rangle$, and that these additions are also scaled by $c = 2^i$.

The details of operations such as addition and multiplication are described at great lengths in the cited works, and interested readers are recommended to seek assistance there for further explanation [12].

3.3. Semiprime Factoring with Grover's Algorithm

After implementing the QFT arithmetic methods and the oracle and diffuser for grover's, it is trivial to implement a SP factoring circuit. Our implementation follows the diagram in Figure 1, while substituting $F(x, y)$ for the $\phi MULT(R_x, R_y)$ operation described at the end of section 3.2. Quantum computing operations can be represented as unitary matrices, so for any $MULT(R_x, R_y)$, the adjoint operation $MULT^{\dagger}(R_x, R_y)$ will return all effected registers to their original states. The only change remaining after applying the multiplication adjoint is that which was done by the oracle; the entangled state of the target qubit has a flipped phase only in those states which resulted in our desired semiprime was computed as an output- the states in which our prime factors reside.

The only fine-tuning necessary here was to add additional bits to the R_z register. Typically, only $\lceil \log_2(N) \rceil$ qubits are needed to represent some number N, and while that is true, we experienced something that might be considered quantum integer overflow before increasing the number of qubits in R_z. Prime factors a, b of N may be as little as one bit smaller than N, and due to the fact that R_x and R_y representing a, b are in superposition, that meant that all numbers up to 2^{n-1} were being included in the superposition for both a and b, resulting in products far larger than could be represented in R_z, which in turn led to false positives due to products that were some sum of powers of 2 greater than n.

The temporary solution we found was to simply decrease the size of R_x and R_y to $n_x = \lceil \log_2(\frac{N}{3}) \rceil$, and set the size of R_z to double that, to ensure no further overflow could occur. The size of R_x and R_y comes from the fact that 2 and 3 may be considered trivial factors, so no number $p \leq N/3$ will result as a factor. $\lceil \log_2(\frac{N}{3}) \rceil$ is a simple computation to find the minimum number of bits required to represent any number $\leq \frac{N}{3}$ in binary, as would be encoded in the qubit registers. The size of R_z being double the size of R_x or R_y is based on the fact that the largest resultant multiple from any value in R_x or R_y can be approximated as $(2^{n_x})^2 = 2^{2 \cdot n_x}$, which in turn would require $2 \cdot n_x$ bits to be safely represented.

4. Results and Discussion

Our implementation of Quantum Semiprime Factoring with Grover's Algorithm is a fully generalized circuit-building algorithm which can take any semiprime N and return unique prime factors p and q with accuracy asymptotically approaching 100% [13]. The program automatically calculates the required number of qubits for each register, and initializes a custom instance of Grover's oracle to check for the provided value N. The simulation uses statevectors to track the superposition of the circuit and apply gates as their corresponding matrices. The circuits are run numerous times, then correctness is checked across those runs to find the statistic for $\%$ acc, as seen in Figure 4.

```
Usage: ./simple_factor.x [N]
Finding prime factors of 143

VERIFIED INPUTS
N: 143 (000010001111)
Circuit requires 25 qubits.

SP Factoring finished in 4.03882s.

MEASURED RESULTS
13 * 11 = 143 (53/100 = 53.00%) ✓
  Full result: 001101_001011_000010001111
  (R1) x: 13 (001101)
  (R2) y: 11 (001011)
  (R3) N: 143 (000010001111)
11 * 13 = 143 (47/100 = 47.00%) ✓
  Full result: 001011_001101_000010001111
  (R1) x: 11 (001011)
  (R2) y: 13 (001101)
  (R3) N: 143 (000010001111)
100 / 100 Shots Correct. (100.00%)
```

Figure 4. Example program output for $N = 143$. The state of each qubit when measured is either $|1\rangle$ or $|0\rangle$, and registers are separated by underscores, as seen after 'Full result: ...'

To date, we have preliminary results based on CUDA-Q simulation of quantum circuits, which have been compared to classic parallel methods to generate primes with a sieve and search them for modulo zero factoring.

We first plan to compare the Grover quantum circuit to our best parallel sieve and modulo search with this scale-out with SDSC Expanse.

4.1. Accuracy and Limitations

On our CSU Chico A100 system, we have been able to scale this implementation up to a total of 32 qubits, used to find the prime factors $47 \times 13 = 611$, where 611 is a 10-bit semiprime. This performance is due to the sub-optimality of the current implementation, as we have not yet introduced the search-term reduction from N to M as is done by S. Whitlock et al.[8].

For context, our current implementation uses $4\lceil \log_2(\frac{N}{3}) \rceil$ qubits to find the prime factors of an n-bit semiprime, whereas the design by S. Whitlock et al.[8] uses up to $2n - 5$ qubits, which allowed them to simulate circuits

which found prime factors of up to 35-bit semiprimes, using up to a total of 65 qubits. The limiting factor for problem scaling in our simulations is memory, so it is expected that after implementing the optimizations from S. Whitlock et al., our simulations should be able to scale up to 18-bit semiprimes on the current equipment.

To be blatantly clear, this Grover's-based implementation is more of a proof-of-concept than a truly preferrable design over Shor's. The longer time complexity of this implementation is expected to coincide with increased error if it were to be run on an actual quantum computer, and the implementation in it's current state has greater time complexity than current classical methods like GNSF [5, 8].

5. Conclusion

Grover's algorithm is highly versatile, and has been implemented in a generalized fashion, such that it is effectively plug-and-play, with the only necessary parameter being the value to search for. Everything else is determined by the function that is chosen for $F(x, \ldots)$.

Despite the large time complexity, even this sub-optimal implementation requires fewer qubits than Shor's algorithm, and constructing the program in such a generalized manner has made it trivial to use Grover's algorithm for nearly any other such function that can take as input registers that are in uniform superposition.

6. Future Work

The current priority regarding the quantum circuit is to implement the optimizations mentioned by S. Whitlock et al. to substitute the search term N with the smaller search term M, which should reduce the number of qubits needed from $4\lceil \log_2(\frac{N}{3}) \rceil$ to $2n - 5$ [8].

We have recently gained access to resources at the San Diego Supercomputing Center (SDSC) Expanse cluster via an NSF allocation grant. This equipment will allow us to simulate on far larger problem scales, and by tracking the number of operations (quantum gates) used at different problem scales and comparing those counts against the number of operations performed in our classical parallel solution, we should be able to simulate circuits at a scale much larger than our current limitation of ≈ 32 qubits.

Acknowledgments

Jacob Collins has been funded by Chico State Enterprises as an undergraduate research assistant through funding provided by the Department of Energy via a grant titled: *QIST in the CSU: Expanding Access to Quantum Information Science and Technology.*

References

[1] Ferguson, Niels and Schneier, Bruce. Practical Cryptography, volume 141. New York: Wiley, 2003.

[2] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996. URL https://arxiv.org/abs/quant-ph/9605043.

[3] Shor, Peter W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing October 1997; 26(5):1484–1509. ISSN 1095-7111. URL http://dx.doi.org/10.1137/S0097539795293172.

[4] Skosana, Unathi and Tame, Mark. Demonstration of Shor's factoring algorithm for N = 21 on IBM quantum processors. Scientific Reports August 2021;11(1). ISSN 2045-2322. URL http://dx.doi.org/10.1038/s41598-021-95973-w.

[5] Shi Bai and Pierrick Gaudry and Alexander Kruppa and Emmanuel Thomé and Paul Zimmermann. Factorisation of rsa-220 with cado-nfs. HAL preprint, 2016. URL https://inria.hal.science/hal-01315738/. Hal-01315738.

[6] Bao Yan et al. Factoring integers with sublinear resources on a superconducting quantum processor, 2022. URL https://arxiv.org/abs/2212.12372.

[7] AbuGhanem M. Ibm quantum computers: Evolution, performance, and future directions, 2024. URL https://arxiv.org/abs/2410.00916.

[8] S. Whitlock and T. D. Kieu. Quantum Factoring Algorithm using Grover Search, 2023. URL https://arxiv.org/abs/2312.10054.

[9] Quantum. MA, Aghaee M, Alcaraz Ramirez A, et al. Interferometric single-shot parity measurement in InAs–Al hybrid devices. Nature 2025;638:651–655.

[10] Beckman, David and Chari, Amalavoyal N. and Devabhaktuni, Srikrishna and Preskill, John. Efficient networks for quantum factoring. Physical Review A August 1996; 54(2):1034–1063. ISSN 1094-1622. URL http://dx.doi.org/10.1103/PhysRevA.54.1034.

[11] Junhong Nie and Wei Zi and Xiaoming Sun. Quantum circuit for multi-qubit Toffoli gate with optimal resource, 2024. URL https://arxiv.org/abs/2402.05053.

[12] Ruiz-Perez, Lidia and Garcia-Escartin, Juan Carlos. Quantum arithmetic with the quantum Fourier transform. Quantum Information Processing April 2017;16(6). ISSN 1573-1332. URL http://dx.doi.org/10.1007/s11128-017-1603-1.

[13] Collins, Jacob. Quantum Factoring Repository. GitHub, 2025. URL https://github.com/collinsjacob127/QuantumFactoring.

Address for correspondence:

Jacob Collins
1565 Filbert Avenue, Chico, CA 95926
jbcollins@csuchico.edu