

# Productionizing Quantum Mass Production

William J. Huggins,<sup>1,\*</sup> Tanuj Khattar,<sup>1</sup> and Nathan Wiebe<sup>2,3,4</sup>

<sup>1</sup>*Google Quantum AI, Mountain View, CA, USA*

<sup>2</sup>*Department of Physics, University of Toronto, Toronto, ON, Canada*

<sup>3</sup>*Department of Computer Science, University of Toronto, Toronto, ON, Canada*

<sup>4</sup>*Pacific Northwest National Laboratory, Richland, WA, USA*

(Dated: June 10, 2025)

For many practical applications of quantum computing, the most costly steps involve coherently accessing classical data. We help address this challenge by applying mass production techniques, which can reduce the cost of applying an operation multiple times in parallel [1–3]. We combine these techniques with modern approaches for classical data loading based on “quantum read-only memory” (QROM). We find that we can polynomially reduce the total number of gates required for data loading, but we find no advantage in cost models that only count the number of non-Clifford gates. Furthermore, for realistic cost models and problem sizes, we find that it is possible to reduce the cost of parallel data loading by an order of magnitude or more. We present several applications of quantum mass production, including a scheme that uses parallel phase estimation to asymptotically reduce the gate complexity of state-of-the-art algorithms for estimating eigenvalues of the quantum chemical Hamiltonian, including both Clifford and non-Clifford gates, from  $\tilde{O}(N_{orb}^2)$  to  $\tilde{O}(N_{orb}^{\log_2 3})$ , where  $N_{orb}$  denotes the number of orbitals. We also show that mass production can be used to reduce the cost of serial calls to the same data loading oracle by precomputing several copies of a novel QROM resource state.

## INTRODUCTION

The high cost of loading classical data poses an obstacle to the search for quantum advantage in machine learning, data analysis, and many other contexts. For example, some of the most promising quantum algorithms for the electronic structure problem make heavy use of classical preprocessing and data loading [4–7]. Even when the amount of data is modest by classical standards, data loading may be costly when we account for the slow speeds we expect from fault-tolerant quantum computers [8, 9]. We apply recently-developed “quantum mass production” techniques to help address this problem [3].

These mass production techniques build on the earlier work of Uhlig, who showed how to construct a classical circuit for evaluating an arbitrary  $n$ -bit boolean function in parallel  $r$  times with a circuit complexity of  $\mathcal{O}(2^n/n)$ , so long as  $r$  is not too large ( $r = 2^{o(n/\log n)}$ ) [1, 2]. Asymptotically, this matches the scaling required to implement a single arbitrary  $n$ -bit boolean function. Ref. 3 developed analogues of these results for preparing arbitrary quantum states or synthesizing arbitrary unitaries in parallel. In both cases, gate complexity of implementing  $r = 2^{o(n/\log n)}$  parallel operations is asymptotically proportional to the complexity of performing the basic task a single time.

We apply these mass production ideas to the task of querying a boolean function in superposition, sometimes referred to as “quantum read-only memory” (QROM) [10]. Given a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , the task is to implement the oracle

$$O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle. \quad (1)$$

There is a significant body of work on optimizing the implementation of such an  $O_f$  [11–17]. We review some of these developments in Appendix A, including the “SelectSwap” QROM that we use in this work [11, 12]. This approach reduced the number of non-Clifford T or Toffoli gates required, at the expense of requiring additional space. Historically, it has been believed that non-Clifford gates will be vastly more costly to implement for fault-tolerant quantum computing using the surface code, making this tradeoff appealing [18–20]. Continued progress on better techniques for implementing non-Clifford gates challenges this assumption [21, 22], motivating us to consider a cost model that does not neglect the cost of Clifford gates.

We outline our mass production protocol and the intuition behind it below, formalizing asymptotic scaling in Theorem 1. Informally, we show that the cost of implementing  $r$  queries to  $O_f$  in parallel is asymptotically proportional to the cost of implementing a single query, so long as  $r$  is not too large. We estimate the actual savings afforded by our construction for mass production using the Qualtran software package [23], finding that it is possible to achieve practical benefits at reasonable problem sizes. Interestingly, this advantage only appears when accounting for the cost of both Clifford and non-Clifford gates. We go on to discuss applications of our protocol, showing that we can reduce the gate complexity for certain applications of parallel phase estimation and amplitude amplification. We even find that mass production can be used to reduce the (amortized) cost of serial queries to the same data loading oracle.

## MASS-PRODUCED QROM QUERIES

Given a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , we aim to implement  $O_f^{\otimes r}$  for some  $r$  that is a power of two. As with Refs. 1–3, the technique for accomplishing this task most efficiently relies on a clever strategy for reducing the implementation of  $O_f^{\otimes r}$  to a collection of smaller data loading tasks. We begin our construction by describing the mass production protocol for implementing two parallel operations. We then construct the general protocol recursively. We give a high-level overview of the procedure here and refer the reader to Appendix B for more details.

We begin by defining a family of functions  $\{f_\ell\}$ , where  $\ell$  ranges from 0 to  $2^k - 1$  for a constant number of bits  $k < n$ . For each  $\ell$ , we let  $f_\ell$  denote the function  $f$  with the first  $k$  bits fixed to (the binary encoding of)  $\ell$ . For example, if  $k = 1$  then the first bit will be fixed to  $\ell$ . Then we define a family of functions,

$$\begin{aligned} g_0 &= f_0, \\ g_\ell &= f_{\ell-1} \oplus f_\ell \text{ for } 1 \leq \ell \leq 2^k - 1, \\ g_{2^k} &= f_{2^k-1}. \end{aligned} \quad (2)$$

Mass production takes advantage of the following equation to evaluate  $f$  on two inputs while only evaluating each  $g_\ell$  at most once,

$$f_\ell(z) = \bigoplus_{j=0}^{\ell} g_j(z) = \bigoplus_{j=\ell+1}^{2^k} g_j(z). \quad (3)$$

Our aim is to implement a circuit

$$\begin{aligned} C : |x_L\rangle |x_R\rangle |\alpha\rangle |y_L\rangle |y_R\rangle |\beta\rangle \\ \rightarrow |x_L\rangle |x_R\rangle |\alpha \oplus f(x)\rangle |y_L\rangle |y_R\rangle |\beta \oplus f(y)\rangle, \end{aligned} \quad (4)$$

where  $x_L$  ( $y_L$ ) denotes the first  $k$  bits of  $x$  ( $y$ ) and  $x_R$  ( $y_R$ ) denotes the remaining  $n - k$  bits. We begin by defining a data loading subroutine for each  $g_\ell$ ,

$$G_\ell : |z\rangle |\gamma\rangle \rightarrow |z\rangle |\gamma \oplus g_\ell(z)\rangle. \quad (5)$$

Using a series of arithmetic comparisons and controlled swap gates, we can control whether  $G_\ell$  is applied to the registers initially containing  $x_R$  and  $\alpha$  or  $y_R$  and  $\beta$ .

We work by applying each  $G_\ell$  in sequence. When  $x_L \leq \ell$ , we route the inputs and outputs so that the effect of  $G_\ell$  is to XOR  $g_\ell(x_R)$  into the  $\alpha$  register. Similarly, when  $y_L > \ell$ , we ensure that  $g_\ell(y_R)$  is XORed into the  $\beta$  register. After this process, the output register initially set to  $|\alpha\rangle$  contains the state  $|\alpha \oplus \bigoplus_{j=0}^{x_L} g_j(x_R)\rangle$  and the other output register contains  $|\beta \oplus \bigoplus_{j=y_L+1}^{2^k} g_j(y_R)\rangle$ . Applying the identities in Equation (3), we find that we have implemented Equation (4) as desired.

We defer analysis of the cost savings to Appendix B 2 and sketch the argument here. The dominant cost of

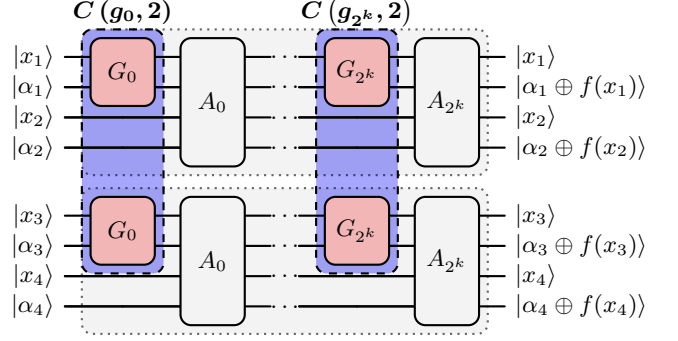


Figure 1: An illustration of how the  $r = 4$ -query mass production protocol is constructed recursively. Two copies of the 2-query protocol are laid out in parallel, each surrounded by dotted lines. We replace the parallel calls to each  $G_\ell$  (pale red) with calls to the 2-query mass production protocol for  $g_\ell$  (shaded blue and outlined with dashed lines).

the construction is the  $2^k + 1$  calls to the data loading oracles  $G_\ell$ . Working in a cost model where we account for Clifford gates, the cost of implementing each  $G_\ell$  is bounded by  $2^{n-k}mK$  for some constant  $K$ . Summing these costs, we have

$$\sum_{\ell=0}^{2^k} \text{Cost}(G_\ell) \leq 2^n m K \frac{2^k + 1}{2^k}. \quad (6)$$

As  $k$  grows, this cost approaches that of implementing a single call to  $O_f$ , which has a cost bounded by  $2^n m K$ . As we show in Appendix B 2, this conclusion breaks down if we neglect the cost of Clifford gates.

The insight that enables us to recursively construct the general protocol is that data loading is present in the two-copy case as a subroutine. Imagine implementing  $r/2$  copies of the two-query mass production protocol in parallel. Doing so would involve implementing  $r/2$  queries to each of the data loading oracles  $G_\ell$  in parallel. Instead of doing this, we implement the  $r/2$  parallel queries to each  $G_\ell$  using an  $r/2$ -query mass production protocol. We illustrate this for the  $r = 4$  case in Figure 1. Since the cost of each two-copy scheme is dominated by the  $G_\ell$ , this replacement will result in an  $r$ -query protocol with a cost comparable to the cost of a single  $r/2$ -query protocol. We formalize this below in Theorem 1, which we prove in Appendix B 3.

**Theorem 1** (Quantum Mass Production for quantum read-only memory). Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be arbitrary, and let  $O_f$  be a unitary operator that queries this function in superposition,  $O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle$ . Let  $\lambda$  be a function of  $n$  that returns a power of 2 such that  $\lambda = o\left(\frac{2^{n/2}}{\sqrt{n}}\right)$  and  $1 \leq \lambda \leq 2^{n/2}m^{-1/2}$ , and let  $m$  be a constant function of  $n$ . For any  $r$  which is a power

of two such that  $r = 2^{o(\frac{n-2 \log \lambda}{\log(n)})}$ , there exists a quantum circuit  $C$  that implements  $O_f^{\otimes r}$  and satisfies the following properties:

1.  $C$  is composed entirely of one- and two-qubit Clifford gates and Toffoli gates.
2. The number of one- and two-qubit Clifford gates in  $C$  is bounded by

$$\text{CLIFFORD}(C) \leq (K + o(1)) 2^n m \quad (7)$$

for some universal constant  $K$ .

3. The number of Toffoli gates in  $C$  is  $\text{TOFFOLI}(C) = (1 + o(1)) 2^n \lambda^{-1}$ .

We note that the  $\lambda$  parameter in the statement of this theorem arises from the usage of “SelectSwap” QROM as a subroutine, which allows for a reduced circuit depth and number of Toffoli gates at the expense of requiring  $\approx \lambda m$  ancilla qubits [11]. While we focus on computational complexity here, it might sometimes be desirable to limit the choice of  $\lambda$  to reduce the number of qubits used. Conversely, choosing a sufficiently large  $\lambda$  could make the additional space required for mass production negligible, leading to an overall spacetime cost that would be approximately proportional to the number of one- and two-qubit Clifford gates. Finally, we observe that the number of copies we can mass produce ( $r$ ) depends on  $\lambda$  in a way that is consistent with our claim that we see no benefit from mass production when we ignore the cost of Clifford gates (and maximize  $\lambda$  to reduce the number of Toffoli gates).

### CONSTANT FACTOR ANALYSIS

We use the Qualtran software package (See Ref. 23) to study the gate complexity of our mass production protocol in this section. We calculate the gate complexity of a circuit  $C$  using the formula

$$\text{COST}(C) = \text{CLIFFORD}(C) + \Xi \text{T}(C), \quad (8)$$

where  $\text{CLIFFORD}(C)$  denotes the number of one and two-qubit Clifford gates,  $\text{T}(C)$  denotes the number of non-Clifford T gates, and  $\Xi$  is a constant (see Appendix A 1 for a discussion of this model). To quantify the advantage provided by a circuit  $C_r$  that mass-produces  $r$  queries to a data loading oracle  $O$ , we calculate the “improvement factor”

$$\mathcal{I} = \frac{\text{COST}(O^{\otimes r})}{\text{COST}(C_r)} = \frac{r \text{COST}(O)}{\text{COST}(C_r)}. \quad (9)$$

For all of the data we present in this section, we minimize the costs of both  $C_r$  and  $O$  by varying the tunable

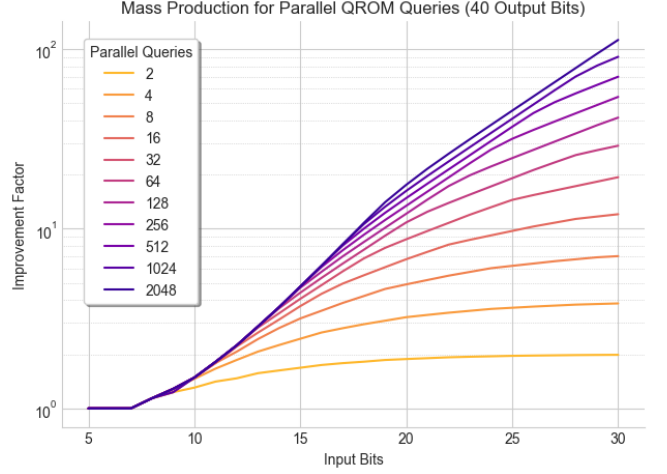


Figure 2: The “improvement factor”  $\mathcal{I}$  when using mass production to implement  $r$  parallel queries to a data loading oracle for an arbitrary function that takes  $n$  input bits and outputs 40 bits. When the number of parallel queries is small and the number of input bits is large, we observe that the improvement factor nearly attains the upper bound of  $r$ , implying that the entire mass production protocol is only slightly more expensive than a single query without mass production.

parameter  $\lambda$  (described in Appendix A 2) and the value of  $k$  chosen at each step in the recursive construction for mass production. See Appendix D for additional details on our numerical experiments.

We plot the improvement factor achievable for mass-producing queries to an arbitrary function of the form  $f : \{0, 1\}^n \rightarrow \{0, 1\}^{40}$  in Figure 2. We vary  $n$  and plot the improvement factor available when implementing  $O_f^{\otimes r}$  for several values of  $r$ , taking  $\Xi = 1$ . At a fixed value of  $r$ , the improvement factor converges towards its upper bound of  $r$  as the number of input bits ( $n$ ) increases. For the largest values of  $n$  and  $r$  we consider in this figure, we find an improvement factor of slightly more than  $10^2$ , which implies that we can perform 2048 parallel queries to a function  $f : \{0, 1\}^{30} \rightarrow \{0, 1\}^{40}$  with roughly the same cost as performing 20 using standard techniques. Specifying such an  $f$  requires more than 200 billion parameters, suggesting that implementations on this scale will have to wait for very large fault-tolerant quantum computers. However, we find that improvements by a factor of 10 or more are reasonable at smaller input sizes. For example, we analyze the sparse simulation method of Ref. 12 in Appendix E 1 and find that model systems with sizes ranging from 100 to 200 spin-orbitals require between 14 and 18 input qubits. The number of output qubits required for a desired level of accuracy can vary, but the original work considered output sizes of  $\approx 80$  for several systems [12].

In Figure 3, we study how the cost of the non-Clifford

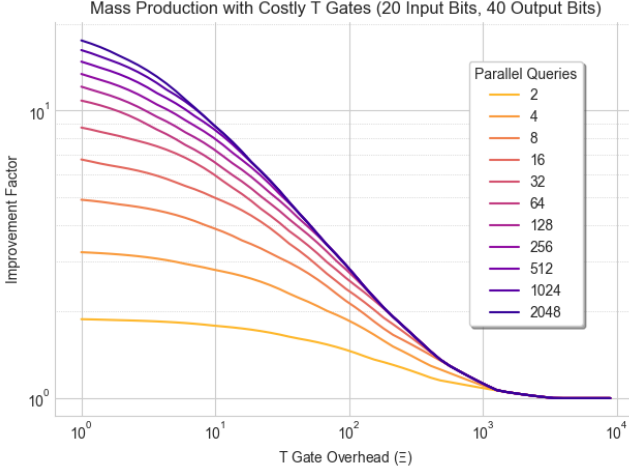


Figure 3: The improvement factor  $\mathcal{I}$  available when using mass production to implement  $r$  parallel queries to a data loading oracle for an arbitrary  $f$  with 20 input bits and 40 output bits, plotted as a function of the T gate overhead ( $\Xi$  in Equation (8)). When  $\Xi = 1$  and the cost of a T gate is taken to be the same as an arbitrary one- or two-qubit Clifford gate, mass production can offer reasonably large improvements. The potential benefit of using mass production is suppressed as  $\Xi$  increases.

T gate (relative to one- and two-qubit Clifford gates) impacts the effectiveness of mass production. Here we fix the number of input bits ( $n$ ) to 20 and the number of output bits ( $m$ ) to 40 and plot the improvement factor available as a function of the T gate overhead ( $\Xi$ , as defined in Equation (8)) for several values of  $r$ . We see that the potential benefit from mass production is strongly suppressed as  $\Xi$  increases. We discuss this behavior in Appendix B 2, where we see that it emerges because the non-Clifford gate complexity of data loading scales as  $2^{n/2}$  rather than  $2^n$  [11]. This suggests that recent work on reducing the cost of non-Clifford gates will be crucial to realizing the benefits of mass production for data loading. For example, Ref. 21 presents data that supports a value of  $\Xi$  roughly between 5 and 20 and the more recent Ref. 24 argues that it may be possible to achieve a value of  $\Xi$  close to 1.

## APPLICATIONS

We highlight a few promising applications of quantum mass production in this section, beginning with its application to parallel phase estimation. This approach involves sharing a GHZ state over  $r$  workers that each have a copy of a state  $|\psi\rangle$  such that  $U|\psi\rangle = e^{-iE}|\psi\rangle$  for some unitary  $U$ . Using phase kickback, each worker can apply the same controlled unitary to collectively ac-

cumulate the phase  $e^{-irE}$ . Ordinarily, this would simply reduce the circuit depth. However, as we explain in Appendix E 1 and Appendix E 2, classical data loading is the dominant cost in some of the state-of-the-art approaches for the quantum simulation of quantum chemistry [4, 5].

We can therefore use quantum mass production to reduce the overall gate complexity of eigenvalue estimation compared with a standard serial phase estimation approach [4, 5]. As examples, we examine the sparse simulation method of Ref. 12 and the tensor hypercontraction algorithm of Ref. 5. By reanalyzing the mass production protocol for a choice of  $r$  such that  $r \propto 2^n$  (see Appendix B 4), we are able to effectively reduce the cost of implementing the qubitized quantum walk operator by a factor that is polynomial in the system size,  $N_{orb}$ . Provided that the appropriate eigenstates are cheaply preparable, the combination of parallel phase estimation and mass production allows us to achieve the best known *total* gate complexity for these methods. We expect that the same improvement should be available for the recently introduced techniques based on spectrum amplification [7].

Amplitude amplification also can benefit from the use of mass production. Given an algorithm  $A$  that produces some desired “good” state  $|G\rangle$  with a small amplitude  $\sqrt{p}$ ,

$$A|0^{\otimes n}\rangle = \sqrt{p}|G\rangle|1\rangle + \sqrt{1-p}|B\rangle|0\rangle, \quad (10)$$

amplitude amplification lets us output  $|G\rangle$  using  $\approx \frac{1}{\sqrt{p}}$  calls to  $A$ . In Appendix E 4, we show that quantum mass production can allow us to reduce the gate complexity of outputting  $|G\rangle$  by a factor of  $\approx \sqrt{r}$ , provided that we can mass produce  $A^{\otimes r}$  at a cost that is comparable to  $A$ , e.g., when the cost of  $A$  is dominated by data loading. This is a promising direction because many quantum algorithms, including algorithms for linear algebra [25, 26], differential equations [27–29], ground state preparation [30], and machine learning [31, 32] involve heavy use of both amplitude amplification and data loading oracles that might be amenable to mass production.

Finally, we show how mass production can enable us to reduce the complexity of serial queries to the same data loading oracle. The idea, which we elaborate on in Appendix F, is simple. We can use our scheme for mass-producing QROM queries to prepare multiple copies of the QROM resource state

$$|\text{QROM}_f\rangle = O_f|+\rangle^{\otimes n}|0\rangle = 2^{-n/2} \sum_{y=0}^{N-1} |y\rangle |f(y)\rangle. \quad (11)$$

We show how to consume one of these resource states to perform a data lookup that is effectively a scrambled version of the desired data lookup,  $\bar{O}_f^{(b)} : |x\rangle|0\rangle \rightarrow |x\rangle|0 \oplus f(x \oplus b)\rangle$ , for some randomly determined bit-string  $b$ . We can correct this scrambling by performing



another data lookup for a function  $g(x) = f(x) \oplus f(x \oplus b)$ . Furthermore, we can take advantage of the fact that  $g(x)$  outputs the same value for the inputs  $x$  and  $x \oplus b$  to implement this correction using half the resources required to implement  $O_f$  directly. In many situations, this allows us to reduce the amortized cost of repeated queries to the same  $O_f$  by nearly a factor of two.

## DISCUSSION

In this paper, we have introduced a variant of quantum mass production that enables multiple parallel queries to a “quantum read-only memory” (QROM) to be performed at a cost that is comparable to the cost of a single query. We used the Qualtran software package to analyze the cost of implementing these parallel queries [23], combining mass production ideas with state-of-the-art techniques for reducing the total number of non-Clifford gates. In a naive cost model where we only account for the cost of non-Clifford gates, we find no benefit to mass-producing QROM queries. However, in a more realistic cost model that accounts for both Clifford and non-Clifford gates, we find that quantum mass production can offer a practical benefit at reasonable problem sizes, reducing the cost of parallel queries by an order of magnitude or more. This offers an alternative approach to benefiting from extra space that, unlike other advanced techniques for data loading [11, 12], has the potential to reduce the overall gate complexity rather than merely the non-Clifford gate complexity.

We proposed several applications of quantum mass production that go beyond the straightforward idea of parallelizing independent executions of the same algorithm. Focusing on the simulation of quantum chemistry as a concrete example, we discussed how mass production and parallel phase estimation might be combined to reduce the overall cost of eigenvalue estimation by parallelizing the PREPARE subroutine in a linear combination of unitaries framework. More generally, we show how mass production could be used in combination with amplitude amplification to reduce the cost for a wide variety of quantum algorithms, provided that their overall cost is dominated by the cost of classical data loading. These concrete examples we have discussed are only a small fraction of the possible applications of quantum mass production. For example, the initial state preparation step for first-quantized chemical simulation naturally involves parallel applications of the same arbitrary unitaries [33, 34], parallel calls to the same block encoding could be used to implement methods based on truncated Taylor series [35], and a host of quantum machine learning algorithms might benefit from cheaply encoding the same classical data multiple times in parallel [31, 36].

Additionally, we proposed a strategy that can reduce the amortized cost of serial queries to the same QROM

by using mass production to precompute several copies of a QROM resource state. We then consume these resource states to implement the desired query, up to a smaller correction. This protocol, which we present in Appendix F, can reduce the cost of repeated queries by almost a factor of two. It would be interesting to understand if our protocol can be improved, or if there is a fundamental reason that the cost reduction is limited to a factor of two. Outside of our general QROM resource state construction, there may be more specialized situations where mass production could be combined with precomputation to realize a speedup, perhaps using density matrix exponentiation as in Ref. 37.

## ACKNOWLEDGMENTS

We thank Craig Gidney for suggesting the resource state construction. We thank Stephen Jordan and Guang Hao Low for useful feedback on the manuscript. NW acknowledges support from the NSERC funded Quantum Software Consortium and also U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-design Center for Quantum Advantage (C2QA) under contract number DE-SC0012704 (PNNL FWP 76274). NW’s research is also supported by PNNL’s Quantum Algorithms and Architecture for Domain Science (QuAADS) Laboratory Directed Research and Development (LDRD) Initiative. The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under Contract DE-AC05-76RL01830.

## CODE AVAILABILITY

The code used to generate the data for this paper is available upon request.

---

\* [whuggins@google.com](mailto:whuggins@google.com)

- [1] D. Ulig, On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements, *Mathematical notes of the Academy of Sciences of the USSR* **15**, 558 (1974).
- [2] D. Uhlir, Networks computing boolean functions for multiple input values, in *Boolean Function Complexity* (Cambridge University Press, 1992) pp. 165–173.
- [3] W. Kretschmer, Quantum mass production theorems, [arXiv:2212.14399 \[quant-ph\]](https://arxiv.org/abs/2212.14399) (2022).
- [4] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, Quantum computing enhanced computational catalysis, *Phys. Rev. Res.* **3**, 033055 (2021).
- [5] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, Even more efficient

- quantum computations of chemistry through tensor hypercontraction, *PRX quantum* **2**, 030305 (2021).
- [6] A. Caesura, C. L. Cortes, W. Pol, S. Sim, M. Steudtner, G.-L. R. Anselmetti, M. Degroote, N. Moll, R. Santagati, M. Streif, and C. S. Tautermann, Faster quantum chemistry simulations on a quantum computer with improved tensor factorization and active volume compilation, [arXiv:2501.06165 \[quant-ph\]](#) (2025).
  - [7] G. H. Low, R. King, D. W. Berry, Q. Han, A. E. DePrince, III, A. White, R. Babbush, R. D. Somma, and N. C. Rubin, Fast quantum simulation of electronic structure by spectrum amplification, [arXiv:2502.15882 \[quant-ph\]](#) (2025).
  - [8] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, Focus beyond quadratic speedups for error-corrected quantum advantage, *PRX quantum* **2**, 010103 (2021).
  - [9] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoeffler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vashillo, Assessing requirements to scale to practical quantum advantage, [arXiv:2211.07629 \[quant-ph\]](#) (2022).
  - [10] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Encoding electronic spectra in quantum circuits with linear T complexity, *Phys. Rev. X* **8**, 041015 (2018).
  - [11] G. H. Low, V. Kliuchnikov, and L. Schaeffer, Trading T gates for dirty qubits in state preparation and unitary synthesis, [arXiv:1812.00954 \[quant-ph\]](#) (2018).
  - [12] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization, *Quantum* **3**, 208 (2019).
  - [13] S. Zhu, A. Sundaram, and G. H. Low, Unified architecture for a quantum lookup table, [arXiv:2406.18030 \[quant-ph\]](#) (2024).
  - [14] V. Giovannetti, S. Lloyd, and L. Maccone, Architectures for a quantum random access memory, [arXiv:0807.4994 \[quant-ph\]](#) (2008).
  - [15] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum random access memory, [arXiv:0708.1879 \[quant-ph\]](#) (2007).
  - [16] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, and P. V. Srinivasan, On the robustness of bucket brigade quantum RAM, *New J. Phys.* **17**, 123010 (2015).
  - [17] S. Jaques and A. G. Rattew, QRAM: A survey and critique, [arXiv:2305.10310 \[quant-ph\]](#) (2023).
  - [18] S. Bravyi and A. Kitaev, Universal quantum computation with ideal clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
  - [19] S. Bravyi and J. Haah, Magic-state distillation with low overhead, *Phys. Rev. A* **86**, 052329 (2012).
  - [20] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, [arXiv:1808.06709 \[quant-ph\]](#) (2018).
  - [21] D. Litinski, Magic state distillation: Not as costly as you think, *Quantum* **3**, 205 (2019).
  - [22] C. Gidney, N. Shutty, and C. Jones, Magic state cultivation: growing t states as cheap as cnot gates, [arXiv preprint arXiv:2409.17595](#) (2024).
  - [23] M. P. Harrigan, T. Khattar, C. Yuan, A. Peduri, N. Yosri, F. D. Malone, R. Babbush, and N. C. Rubin, Expressing and analyzing quantum algorithms with qualtran, [arXiv:2409.04643 \[quant-ph\]](#) (2024).
  - [24] G. Craig, S. Noah, and J. Cody, Magic state cultivation: growing T states as cheap as CNOT gates, [arXiv:2409.17595 \[quant-ph\]](#) (2024).
  - [25] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, *Phys. Rev. Lett.* **103**, 150502 (2009).
  - [26] A. M. Childs, R. Kothari, and R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, *SIAM J. Comput.* **46**, 1920 (2017).
  - [27] D. W. Berry, High-order quantum algorithm for solving linear differential equations, *J. Phys. A Math. Theor.* **47**, 105301 (2014).
  - [28] D. W. Berry, A. M. Childs, A. Ostrander, and G. Wang, Quantum algorithm for linear differential equations with exponentially improved dependence on precision, *Commun. Math. Phys.* **356**, 1057 (2017).
  - [29] J.-P. Liu, H. O. Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa, and A. M. Childs, Efficient quantum algorithm for dissipative nonlinear differential equations, *Proc. Natl. Acad. Sci. U. S. A.* **118**, e2026805118 (2021).
  - [30] L. Lin and Y. Tong, Near-optimal ground state preparation, *Quantum* **4**, 372 (2020).
  - [31] S. Lloyd, S. Garnerone, and P. Zanardi, Quantum algorithms for topological and geometric analysis of data, *Nat. Commun.* **7**, 10138 (2016).
  - [32] J. Liu, M. Liu, J.-P. Liu, Z. Ye, Y. Wang, Y. Alexeev, J. Eisert, and L. Jiang, Towards provably efficient quantum algorithms for large-scale machine-learning models, *Nat. Commun.* **15**, 434 (2024).
  - [33] Y. Su, D. W. Berry, N. Wiebe, N. Rubin, and R. Babbush, Fault-tolerant quantum simulations of chemistry in first quantization, *PRX Quantum* **2**, 040332 (2021).
  - [34] W. J. Huggins, O. Leimkuhler, T. F. Stetina, and K. B. Whaley, Efficient state preparation for the quantum simulation of molecules in first quantization, [arXiv preprint arXiv:2407.00249](#) (2024).
  - [35] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Simulating hamiltonian dynamics with a truncated taylor series, *Phys. Rev. Lett.* **114**, 090502 (2015).
  - [36] D. Gilboa, H. Michaeli, D. Soudry, and J. McClean, Exponential quantum communication advantage in distributed inference and learning, *Neural Inf Process Syst* **37**, 30425 (2023), 2310.07136.
  - [37] W. J. Huggins and J. R. McClean, Accelerating quantum algorithms with precomputation, [arXiv:2305.09638 \[quant-ph\]](#) (2023).
  - [38] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, Toward the first quantum simulation with quantum speedup, *Proc. Natl. Acad. Sci. U. S. A.* **115**, 9456 (2018).
  - [39] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, Elucidating reaction mechanisms on quantum computers, *Proceedings of the national academy of sciences* **114**, 7555 (2017).
  - [40] T. Häner, M. Roetteler, and K. M. Svore, Factoring using  $2n+2$  qubits with toffoli based modular multiplication, [arXiv preprint arXiv:1611.07995](#) (2016).
  - [41] Y. R. Sanders, D. W. Berry, P. C. Costa, L. W. Tessler, N. Wiebe, C. Gidney, H. Neven, and R. Babbush, Compilation of fault-tolerant quantum heuristics for combinatorial optimization, *PRX quantum* **1**, 020312 (2020).

- [42] J. O’Gorman and E. T. Campbell, Quantum computation with realistic magic-state factories, *Physical Review A* **10.1103/PhysRevA.95.032338** (2017).
- [43] C. Gidney and A. G. Fowler, Efficient magic state factories with a catalyzed  $|ccz\rangle$  to  $2|t\rangle$  transformation, *Quantum* **3**, 135 (2019), 1812.01238v3.
- [44] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara, H. Levine, G. Semeghini, M. Greiner, V. Vuletic, and M. D. Lukin, High-fidelity parallel entangling gates on a neutral atom quantum computer, [arXiv:2304.05420 \[quant-ph\]](https://arxiv.org/abs/2304.05420) (2023).
- [45] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, *Nat. Phys.* **20**, 1084 (2024).
- [46] L. Grover and T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, [arXiv:quant-ph/0208112 \[quant-ph\]](https://arxiv.org/abs/quant-ph/0208112) (2002).
- [47] D. Gosset, R. Kothari, and K. Wu, Quantum state preparation with optimal T-count, [arXiv:2411.04790 \[quant-ph\]](https://arxiv.org/abs/2411.04790) (2024).
- [48] N. J. Ross and P. Selinger, Optimal ancilla-free clifford+ t approximation of z-rotations, *arXiv preprint arXiv:1403.2975* (2014).
- [49] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019 (ACM, New York, NY, USA, 2019) pp. 193–204.
- [50] G. H. Low and I. L. Chuang, Hamiltonian simulation by qubitization, *Quantum* **3**, 163 (2019).
- [51] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, Grand unification of quantum algorithms, *PRX quantum* **2**, 040203 (2021).
- [52] J. R. McClean, R. Babbush, P. J. Love, and A. Aspuru-Guzik, Exploiting locality in quantum computation for quantum chemistry, [arXiv:1407.7863 \[quant-ph\]](https://arxiv.org/abs/1407.7863) (2014).
- [53] K. M. Svore, M. B. Hastings, and M. Freedman, Faster phase estimation, *arXiv preprint arXiv:1304.0741* (2013).
- [54] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, Matrix product state representations, *Quantum Inf. Comput.* **7**, 401 (2007).
- [55] I. V. Oseledets, Tensor-train decomposition, *SIAM J. Sci. Comput.* **33**, 2295 (2011).
- [56] S. Fomichev, K. Hejazi, M. S. Zini, M. Kiser, J. Fraxanet, P. A. M. Casares, A. Delgado, J. Huh, A.-C. Voigt, J. E. Mueller, *et al.*, Initial state preparation for quantum chemistry on quantum computers, *PRX Quantum* **5**, 040339 (2024).
- [57] D. W. Berry, Y. Tong, T. Khattar, A. White, T. I. Kim, S. Boixo, L. Lin, S. Lee, G. K.-L. Chan, R. Babbush, and N. C. Rubin, Rapid initial state preparation for the quantum simulation of strongly correlated molecules, [arXiv:2409.11748 \[quant-ph\]](https://arxiv.org/abs/2409.11748) (2024).
- [58] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, Quantum amplitude amplification and estimation, *arXiv preprint quant-ph/0005055* (2000).
- [59] C. Zalka, Grover’s quantum searching algorithm is optimal, *Phys. Rev. A* **60**, 2746 (1999).
- [60] T. J. Yoder, G. H. Low, and I. L. Chuang, Fixed-point quantum search with an optimal number of queries, *Phys. Rev. Lett.* **113**, 210501 (2014).

## Appendix A: Querying a classical function in superposition with quantum read-only memory (QROM)

Consider an arbitrary boolean function,

$$f : \{0, 1\}^n \rightarrow \{0, 1\}^m. \quad (\text{A1})$$

We often want to construct an oracle that allows us coherent access to such a function,

$$O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle. \quad (\text{A2})$$

In order to establish a framework for discussing asymptotic costs, we first begin with a short discussion of cost models for quantum algorithms in Appendix A 1. In Appendix A 2, we summarize prior work on instantiating such oracles [10–12, 38]. Throughout the paper, we use the phrase “quantum read-only memory,” or “QROM” to refer to the whole family of approaches to implementing such oracles, although specific variants often have other more technical names. Then, in Appendix A 3, we introduce a variant of QROM that will be particularly useful in constructing our mass production protocol.

### 1. A cost model for fault-tolerant algorithms

In order to discuss the asymptotic costs of various circuits, we need to specify a cost model. One possible strategy is to count the number of arbitrary one- and two-qubit gates, or the number of gates in some particular universal gate set. Motivated by the belief that non-Clifford gates will be dramatically more resource-intensive to implement using practical approaches to quantum error correction [18], many works on fault-tolerant quantum algorithms have focused exclusively on quantifying the number of non-Clifford, T or Toffoli gates [39–41]. However, continued progress in recent years suggests that this belief may be outdated [19, 21, 24, 39, 42, 43]. In particular, Ref. 24 argues that we may be able to “cultivate” magic states with sufficiently low error rates that we can implement the non-Clifford T gate in the surface code with only slightly more resources than a standard CNOT gate, although further work is needed to establish that this is true across a range of device parameters.

In order to model a range of plausible scenarios, we adopt the following cost model for a circuit  $C$  expressed in a Clifford + T gate set:

$$\text{COST}(C) = \text{CLIFFORD}(C) + \Xi \text{T}(C), \quad (\text{A3})$$

where  $\text{CLIFFORD}(C)$  denotes the number of one- and two-qubit Clifford gates in  $C$ ,  $\text{T}(C)$  denotes the number of T gates, and  $\Xi$  is a constant. Taking  $\Xi = 1$  recovers the model where we simply count the number of one- and two-qubit gates in a Clifford + T gate set, and taking  $\Xi \rightarrow \infty$  recovers the model where we discount the cost of Clifford gates entirely. By varying  $\Xi$ , we can explore regimes in between these extremes. While initial estimates for  $\Xi$  were in the thousands [18, 21], more modern approaches to magic state distillation have provided values in the tens or hundreds [21, 43], and Ref. 24 argues for a value of  $\Xi$  close to one.

Many of the constructions in this work, including the various implementations of QROM that we discuss in the next section, are naturally expressed using the non-Clifford Toffoli gate rather than T gates. However, for simplicity, in our constant factor cost analyses we ultimately report the cost in terms of T gates by using a standard decomposition for Toffoli gates into four T gates plus several one- and two-qubit Clifford gates.

While our cost model is more nuanced than merely counting the total number of gates or the total number of non-Clifford gates, it still only provides a rough estimate for the true cost. For example, some Clifford gates may be more difficult to implement than others, and the costs may vary based on the particular choice of quantum error correcting code or the underlying hardware platform [20, 44, 45]. Despite these subtleties, we expect that our cost model will provide useful estimates, particularly since we are focused on the relative costs of different approaches.

### 2. Prior work on implementations of QROM

There is a significant body of work on instantiating data loading oracles of the form given in Equation (A2). The quantum read-only memory of Ref. 10 provides a straightforward implementation whose gate complexity (in terms of the number of one- and two-qubit Clifford gates and T/Toffoli gates) is  $\Theta(2^n m)$ . The SelectSwap QROM of Ref. 11 demonstrated that a similar gate complexity could be obtained with a quadratically smaller number of non-Clifford



gates, at the expense of requiring additional space. Ref. 12, which refers to this construction as advanced quantum read-only memory (QROAM), developed additional tools to efficiently uncompute QROAM queries in certain contexts. Our constructions make use of the tools developed in these three works, but we note other improvements and variations have been proposed more recently [13].

Our summary will mostly follow the notation and language of Ref. 12. Let  $N = 2^n$  and let  $\lambda$  be a power of two such that  $1 < \lambda < N$ . The assumption that  $N$  is a power of two can be relaxed to cover the setting where we only care about the action of  $f$  on the first  $N$  non-negative integers (with  $n = \lceil \log N \rceil$ ). Since this only introduces a constant factor difference in the cost, we only consider the simpler case where  $N$  is a power of 2 for simplicity.

Using the techniques of Berry *et al.*, we can implement  $O_f$  using  $N\lambda^{-1} + m(\lambda - 1)$  Toffoli gates and  $(\lambda - 1)m + \log_2(N\lambda^{-1})$  clean ancilla (ancilla initialized in the  $|0\rangle$  state), together with  $\Theta(Nm)$  one- and two-qubit Clifford gates (for a generic  $f$ ). More specifically, the number of one- and two-qubit Clifford gates is bounded by  $NmK$  for some small constant  $K$ . Furthermore, the spacetime volume (the product of the depth of the circuit and the number of qubits it requires) scales as  $\tilde{O}(2^n m)$ , regardless of  $\lambda$ .

Technically their construction assumes that the output register is in the zero state, but the general case can be treated with only a small increase in cost. Potentially more seriously, their approach leads either i) an entangled junk register that must be uncomputed, or ii) random (but known at execution-time) phase errors that must be corrected. The cost of these corrections is often neglected because, when the output of the QROAM is used and then immediately uncomputed, the corrections can be applied at no additional cost during the uncomputation step. Since this will not necessarily be the case in our application of QROAM, we introduce a modified variant of the clean ancilla assisted QROAM in Figure 4 that performs the desired operation without requiring any additional corrections and without any constraints on  $\alpha$ .

An alternative construction makes use of dirty ancilla qubits (ancilla qubits borrowed and returned in an arbitrary state) rather than clean ancilla [11, 12]. Implementing QROAM this way uses  $2N\lambda^{-1} + 4m(\lambda - 1)$  Toffoli gates,  $m(\lambda - 1)$  dirty ancilla, and  $\log_2(N\lambda^{-1})$  clean ancilla, and  $\Theta(Nm)$  one- and two-qubit Clifford gates. There is no need to perform additional corrections when using this approach.

It is frequently critical to uncompute such a data lookup oracle. Berry *et al.* showed how this uncomputation can be done particularly efficiently using a measurement-based strategy. This approach uses  $N\lambda^{-1} + \lambda$  Toffoli gates and  $\lambda + \log_2(N\lambda^{-1})$  clean ancilla or  $2N\lambda^{-1} + 4\lambda$  Toffoli gates,  $\lambda - 1$  dirty ancilla, and  $\log_2(N\lambda^{-1}) + 1$  clean ancilla [12]. The number of one- and two-qubit Clifford gates for the uncomputation step scales as  $\Theta(N)$  (provided that we don't count the  $m$   $X$ -basis measurements as gates). This measurement-based uncomputation strategy therefore avoids the scaling with  $Nm$  that might be naively expected.

### 3. Modified clean ancilla-assisted QROAM that uncomputes the junk register

QROAM enables us to minimize the number of non-Clifford Toffoli gates required for coherently loading classical data by using additional space. As we discussed in Appendix A2, we can implement a data lookup oracle  $O_f$  for a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  following the methods of Ref. 12 (which build on Ref. 11). We can either do so using  $\lceil N/\lambda \rceil + m(\lambda - 1)$  Toffoli gates and  $(\lambda - 1)m + \lceil \log N/\lambda \rceil$  clean ancilla (ancilla initialized in the  $|0\rangle$  state), or using  $2\lceil N/\lambda \rceil + 4m(\lambda - 1)$  Toffoli gates,  $(\lambda - 1)m$  dirty ancilla (ancilla borrowed in an arbitrary state), and  $\lceil \log N/\lambda \rceil$  clean ancilla. In both cases, we define  $N = 2^n$  to parallel the notation of Ref. 12. Provided we have the clean ancilla available, using them results in the lowest Toffoli cost. Therefore, in this work we only consider the use of the clean ancilla variants of QROM, although we expect that our results could also be adapted to the dirty ancilla case.

However, the costs quoted for the clean ancilla version of QROM above do not exactly correspond to the costs for implementing the desired

$$O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle. \quad (\text{A4})$$

Rather than implementing the above operation, the clean-ancilla version of QROAM actually implements a unitary operation  $\tilde{O}_f$  that acts as below,

$$\tilde{O}_f : |x\rangle |0\rangle^{\otimes \lambda} \rightarrow |x\rangle |f(x)\rangle |j(x)\rangle, \quad (\text{A5})$$

where  $|j(x)\rangle$  is some “junk” computational basis state on  $m(\lambda - 1)$  qubits.<sup>1</sup> In practice, QROAM is often employed in a context where the output values will be used and then immediately uncomputed. If this is the case, then the

<sup>1</sup> The dirty ancilla variants of QROAM do not have this issue since

they explicitly restore the state of the borrowed qubits.

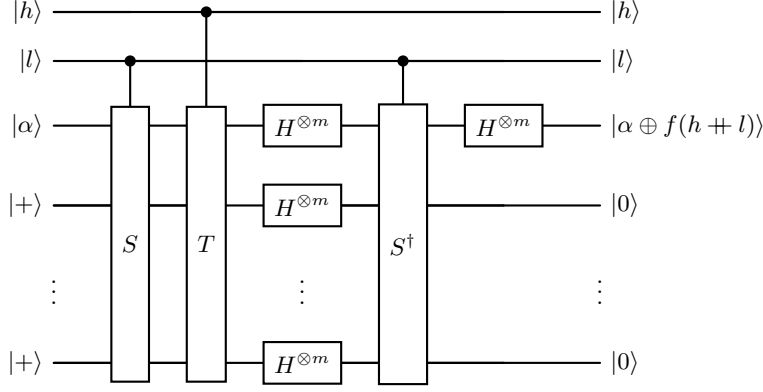


Figure 4: An alternative to the clean ancilla QROAM of Ref. 11 and Ref. 12 that does not require uncomputation on the ancilla. Instead, it directly implements the desired data loading operation, XORing the data into the first output register and returning the ancilla to the  $|0\rangle$  state. Here  $T$  and  $S$  refer to a QROM data lookup and a series of controlled swap gates, as in Ref. 12.  $S^\dagger$  can be implemented without using any non-Clifford gates using the measurement-based uncomputation approach shown in Figure 10 of Ref. 12.

uncomputation of the  $|j(x)\rangle$  register can be folded into the same step that uncomputes the  $|f(x)\rangle$  register at no additional cost, as described in Ref. 12.

However, in our case, we do not immediately use the output and then uncompute it, so it is not clear that the same kind of optimization is available to us. Naively, we might expect to have to pay an additional cost to uncompute the junk data before proceeding. Specifically, we could perform the usual data loading step, then  $m$  CNOT gates to XOR  $f(x)$  in the register containing  $\alpha$ , and then uncompute the registers containing  $f(x)$  and  $j(x)$ . However, the uncomputation itself requires a data loading, which would incur an additional cost of  $\lceil N\lambda^{-1} \rceil + \lambda$  Toffoli gates (using  $\lambda + \lceil \log N\lambda^{-1} \rceil$  clean ancilla) and  $\mathcal{O}(N)$  Clifford gates. In the worst case, this would nearly double the cost of data loading and would make it more difficult to benefit from mass production at all.

We therefore propose a modification to the standard clean ancilla QROAM that guarantees that the additional ancilla will be returned to the zero state. This modification requires zero additional non-Clifford gates, and only  $\Theta(m\lambda)$  additional one- and two-qubit Clifford gates. We show this alternative strategy in Figure 4.

This QROAM is a modification of the construction from Ref. 12, and we closely follow their notation for the remainder of this section. In our circuit diagram,  $S$  denotes a series of controlled swap gates that permutes the  $\lambda$  output registers so that the register that initially contains  $\alpha$  ends up in the  $l$ th position.  $T$  denotes the usual QROM data loading operation that outputs  $m\lambda$  bits so that the  $l$ th register contains the data for  $f(h + l)$ . Implementing  $T$  requires  $\lceil 2^n/\lambda \rceil$  Toffoli gates and  $\mathcal{O}(2^n m)$  Clifford gates. Our construction takes advantage of the fact that  $T$  will act as the identity on any output registers in the  $|+\rangle^{\otimes m}$  state. Initializing the additional clean qubits in the  $|+\rangle$  ensures that only the  $l$ th register is modified by the action of  $T$ .

Berry *et al.*'s construction for clean ancilla QROAM consists of exactly one controlled  $T$  gate and one controlled  $S$  gate. Our circuit differs from theirs by the addition of the Hadamard gates as shown in the figure, the  $X$ -basis initialization of the clean ancilla qubits, and the controlled  $S^\dagger$  operation. We could implement  $S^\dagger$  as a product of controlled swaps. In our particular circuit, we can take advantage of the fact that we know the outputs for all of the registers but the first one will be zero to simplify the circuit and entirely remove the need for non-Clifford gates, as in Figure 10 of Ref. 12. The circuit implementation of  $S$  contains exactly  $m(\lambda - 1)$  controlled swap gates (equivalently,  $2m(\lambda - 1)$  CNOT gates and  $m(\lambda - 1)$  Toffoli gates). The measurement-based implementation of  $S^\dagger$  contains  $m(\lambda - 1)$  each of CNOT gates, Hadamard gates, single-qubit measurements, and classically-controlled CZ gates.<sup>2</sup>

Overall then, we can say that our modified advanced QROM circuit requires  $\lceil N/\lambda \rceil + m(\lambda - 1)$  Toffoli gates and  $(\lambda - 1)m + \lceil \log N/\lambda \rceil$  clean ancilla, together with a number of Clifford gates bounded by  $KNm$  for some universal constant  $K$ . Furthermore, the number of Clifford gates required scales the same as the standard clean-ancilla advanced QROM, up to leading order.

<sup>2</sup> One could further simplify by combining the Hadamard gates of Figure 4 with the implementation of  $S^\dagger$  from Ref. 12 but we

neglect this optimization for now.

## Appendix B: Mass production for classical data loading

We detail our mass production protocol for classical data loading in this section. Following Uhlig and Kretschmer, we construct our mass production protocol recursively [1–3]. We first present the two-query protocol for implementing  $O_f^{\otimes 2}$ . Then we explain how to construct the protocol for  $O_f^{\otimes 2^r}$  given a protocol for  $O_f^{\otimes r}$ . We comment on the asymptotic costs as we present the construction, culminating in Appendix B3, where we prove Theorem 1 by induction. For an analysis of the constant factor costs, we refer the reader to the results presented in the main text and to our implementation of the protocol in the Qualtran software package [23].

Before we begin, it is helpful to introduce some convenient notation. We use the symbol  $++$  to denote the concatenation of two bitstrings, e.g.,  $11 ++ 00 = 1100$ . For integers  $a$  and  $b$  with  $a < b$ , we use the notation  $[a..b]$  to denote the sequence  $a, a+1, a+2, \dots, b$ . We frequently conflate non-negative integers and the bitstrings that encode them in a standard unsigned big-endian format.

Our protocol implements  $r$  queries to an oracle  $O_f$  for an arbitrary function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , where  $r = 2^t, t \in \mathbb{Z}^+$ . We construct our protocol with several free parameters that we set later in order to optimize the costs. We take  $\lambda$ , a parameter that governs a tradeoff involved in the QROAM data lookups, to be a power of 2 such that  $\lambda \leq 2^{n/2} m^{-1/2}$ . For the purposes of statements about asymptotic scaling, we consider  $m$  a constant and  $\lambda$  to be a function of  $n$ . Similarly, viewing  $t$  as a function of  $n$ , we require that  $t = o(\frac{n-2\log_2 \lambda}{\log_2 n})$  (although we will later relax this requirement for Proposition 3). We also introduce a parameter  $k \in \mathbb{Z}^+$ , subject to the requirement that  $n > kt$  and  $k = \mathcal{O}(\log n)$ . More specifically, we will set  $k = \lceil \log_2 n \rceil$  when we prove Theorem 1 in Appendix B3 and  $k = 1$  when we prove Proposition 3, although we keep it as a free parameter in the exposition since we will later optimize it to minimize the costs in our constant factor analysis.

Ultimately, we construct the circuit that implements  $O_f^{\otimes r}$  (which we denote by  $C_{f,n,m,\lambda,k,t}$ ) by induction on  $t$ . Since we require that  $t = o(\frac{n-2\log_2 \lambda}{\log_2 n})$  and we are interested in understanding the asymptotic scaling for non-zero values of  $t$ , we impose an additional technical condition on  $\lambda$ . Specifically, we demand that

$$\lim_{n \rightarrow \infty} \frac{n - 2\log_2 \lambda}{\log_2 n} \geq c \quad (\text{B1})$$

for all  $c > 0$ . Rearranging this expression, we find that we need

$$\lim_{n \rightarrow \infty} \lambda \leq 2^{-c} \frac{2^{n/2}}{\sqrt{n}}. \quad (\text{B2})$$

Since  $2^{-c}$  is an arbitrary positive number, this is equivalent to the demand that

$$\lambda = o\left(\frac{2^{n/2}}{\sqrt{n}}\right). \quad (\text{B3})$$

We note that this is a stricter demand (asymptotically) than the already-stated requirement that  $\lambda \leq 2^{n/2} m^{-1/2}$ .

### 1. The base case

We begin by constructing a circuit  $C_{f,n,m,\lambda,k,1}$ , which we abbreviate as  $C$  throughout this section, that evaluates  $f$  on two inputs. At a high level, we try to parallel Ref. 3 and Ref. 2 in the construction and notation of the mass production theorems and Ref. 12 in the construction and notation of the QROAM data lookups. The circuit  $C$  should act unitarily on  $2n + 2m$  qubits, applying  $O_f$  to a pair of input and output registers, i.e.,

$$C : |x\rangle |\alpha\rangle |y\rangle |\beta\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle |y\rangle |\beta \oplus f(y)\rangle. \quad (\text{B4})$$

For simplicity, we make the assumption for now that  $x \leq y$ . If we can implement  $C$  in this case, then we can address the more general situation by first checking  $x \leq y$  with a comparator and swapping the input and output registers when  $x > y$ . After implementing  $C$ , we can then swap the registers back and uncompute the comparator.

Following Ref. 2, we define two families of functions,  $\{f_\ell\}$  and  $\{g_\ell\}$ . Let  $k \in \{1, \dots, n\}$  be a constant that we will set later. Let  $\ell$  be a  $k$ -bit integer,  $\ell \in \{0, \dots, 2^k - 1\}$ , and  $f_\ell : \{0, 1\}^{n-k} \rightarrow \{0, 1\}^m$  denote the function  $f$  with the first  $k$  bits fixed to  $\ell$ , i.e.,

$$f_\ell(z) = f(\ell ++ z). \quad (\text{B5})$$

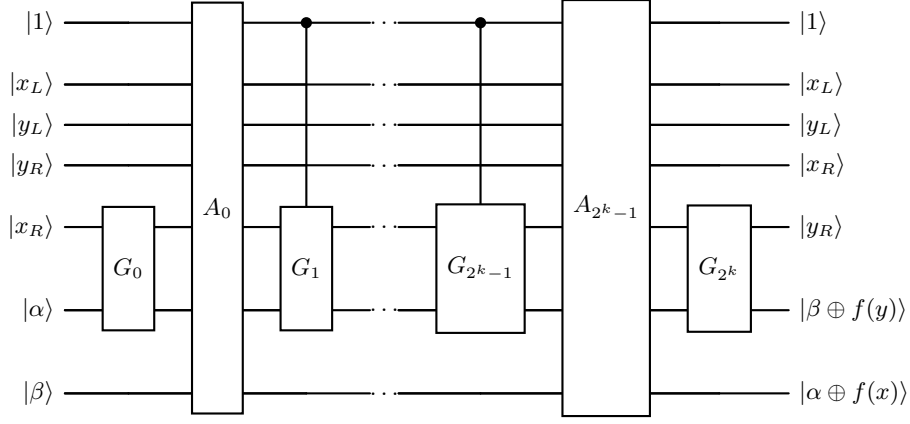


Figure 5: A quantum circuit diagram for  $C$ . For clarity, we omit some initial and final swaps required to make the registers correspond exactly to Equation (B4).  $G_\ell$  is a data lookup that implements the function  $g_\ell$ . The  $A_\ell$ , which are defined in the text and illustrated in Figure 6, handle the “control flow,” routing the inputs and outputs appropriately for each call to  $G_\ell$ .

Then, for  $\ell \in [0 \dots 2^k]$ , let

$$g_\ell = \begin{cases} f_0 & \ell = 0, \\ f_{\ell-1} \oplus f_\ell & 1 \leq \ell \leq 2^k - 1, \\ f_{2^k-1} & \ell = 2^k \end{cases} \quad (\text{B6})$$

Note that

$$f_\ell(z) = \bigoplus_{j=0}^{\ell} g_j(z) = \bigoplus_{j=\ell+1}^{2^k} g_j(z). \quad (\text{B7})$$

We will show that the key insight that enables mass production is that we can take advantage of Equation (B7) to evaluate  $f$  on two different inputs while only evaluating each  $g_\ell$  at most once. Because the  $g_\ell$  are “simpler” by a factor of  $2^k$ , then the overall cost of implementing  $2^k + 1$  of them is comparable to the cost of implementing a single call to  $f$ .

We explain the construction for  $C$  by describing its action on a computational basis state  $|x\rangle|\alpha\rangle|y\rangle|\beta\rangle$ . We treat the first  $k$  bits of  $x$  and  $y$  differently from the remaining  $n - k$ , so we adopt the notation that

$$x = x_L \uplus x_R \quad (\text{B8})$$

$$y = y_L \uplus y_R, \quad (\text{B9})$$

where  $x_L$  ( $y_L$ ) denotes the first  $k$  bits of  $x$  ( $y$ ) and  $x_R$  ( $y_R$ ) denotes the remaining bits. We implement  $C$  by alternating the application of two key primitives,  $G_\ell$  and  $A_\ell$ , as illustrated in Figure 5.

The first type of primitive,  $G_\ell$ , is a (usually controlled) call to a data lookup oracle for the corresponding function  $g_\ell$ ,

$$G_\ell : |c\rangle|z\rangle|\gamma\rangle \rightarrow |c\rangle \begin{cases} |z\rangle|\gamma\rangle & \text{if } c=0 \\ |z\rangle|\gamma \oplus g_\ell(z)\rangle & \text{if } c=1. \end{cases} \quad (\text{B10})$$

We implement this data lookup using the techniques described in Appendix A3, which is a slight variation of the usual QROAM construction from Ref. 12. As shown in Figure 5, we apply the  $G_\ell$  sequentially for  $\ell \in [0, \dots, 2^k]$ .

For each  $G_\ell$ , we will be in one of three cases,

1.  $\ell \leq x_L$ ,
2.  $x_L < \ell \leq y_L$ , or



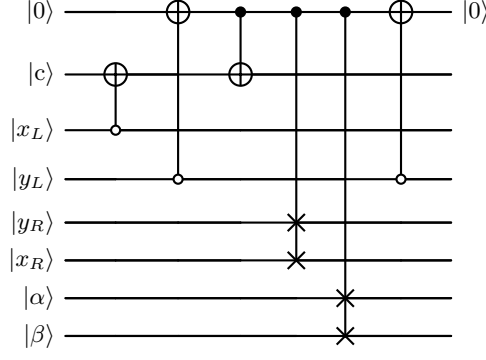


Figure 6: A quantum circuit diagram for a possible realization of  $A_\ell$ . In this figure, the controlled operations where the control qubit is marked with an open circle indicate operations that are controlled based on the condition that the control register encodes the value  $\ell$ . Our Qualtran implementation of this primitive uses machinery built into the library to compile the multi-qubit controls, which introduces  $\mathcal{O}(k)$  additional ancilla qubits not shown here.

### 3. $y_L < \ell$ .

In case 1, we would like to use  $G_\ell$  to evaluate  $g_\ell(x_R)$  and XOR the output into the  $\alpha$  register. In case 2, we should effectively not apply  $G_\ell$  at all. In case 3, we want to evaluate  $g_\ell(y_R)$  and XOR the output into the  $\beta$  register.

In order to implement the desired operations, the control qubit for the data lookups should be in the  $|1\rangle$  state whenever  $G_\ell$  is called and we are in case 1 or case 3, and in the  $|0\rangle$  state when we are in case 2. Furthermore, when we are in case 1,  $G_\ell$  should be called with the input register in the state  $|x_R\rangle$  and the output should be written to the register that is initially in the state  $|\alpha\rangle$ . When we are in case 3, the inputs and outputs should be swapped so that  $G_\ell$ 's input register is in the state  $|y_R\rangle$  and the output register is the one that initially contained the state  $|\beta\rangle$ .

By definition we are in case 1 when  $\ell = 0$ , so we can simply apply  $G_0$  without a control qubit as in the first step of Figure 5. After each call to  $G_\ell$  (except for the last), we can then apply  $A_\ell$  as defined in Figure 6 to flip the control qubit and swap the input and output registers as necessary. Inspecting Figure 6, we see that  $A_\ell$  will flip the control qubit for the  $G_\ell$  whenever  $\ell = x_L$ , transitioning us from the settings necessary for case 1 to the settings necessary for case 2. Likewise, when  $\ell = y_L$ , the control qubit is flipped again and we swap the two input and two output registers, as required for case 3. After applying  $G_{2^k}$ , the only remaining task is to swap the input and output registers back, since they were swapped exactly once by the action of the  $A_\ell$ s. Let us denote this final swapping operation by  $A_{2^k}$ .

Letting  $S$  denote the additional swap operations necessary to convert  $|x\rangle |\alpha\rangle |y\rangle |\beta\rangle$  to  $|x_L\rangle |y_L\rangle |y_R\rangle |x_R\rangle |\alpha\rangle |\beta\rangle$  (which are really an artifact of our presentation anyway), we have

$$S^\dagger A_{2^k} G_{2^k} \cdots A_1 G_1 A_0 G_0 S |1\rangle |x\rangle |\alpha\rangle |y\rangle |\beta\rangle = |1\rangle |x\rangle |\alpha'\rangle |y\rangle |\beta'\rangle, \quad (\text{B11})$$

where, taking advantage of Equation (B7), we have

$$\alpha' = \alpha \bigoplus_{j=0}^{x_L} g_j(x_R) = \alpha \oplus f_{x_L}(x_R) = \alpha \oplus f(x_L \oplus x_R) = \alpha \oplus f(x) \quad (\text{B12})$$

$$\beta' = \beta \bigoplus_{j=y_L}^{2^k} g_j(y_R) = \beta \oplus f_{y_L}(y_R) = \beta \oplus f(y_L \oplus y_R) = \beta \oplus f(y). \quad (\text{B13})$$

Therefore, the circuit presented in Figure 5 implements Equation (B4) as desired.

## 2. Cost analysis for the two copy protocol

The two-copy protocol for mass production forms the foundation for the  $r$ -copy protocol, so it is worth carefully analyzing its costs. Here we address the asymptotic scaling. There are contributions to the cost from four sources, which we order by their importance:

1. The QROM data lookups  $G_\ell$ .

2. The control flow circuits  $A_\ell$ .
3. The reduction to the case where  $x \leq y$ .
4. A few additional swap gates.

The dominant cost comes from the  $2^k + 1$  QROM data lookups. As we discussed in Appendix A 2, while it is possible to reduce the number of non-Clifford gates required for a data lookup to scale sublinearly in  $2^n m$  (using techniques known as SelectSwap QROM or advanced QROM), the number of Clifford gates can not be reduced. Therefore, in the cost model we are considering, the asymptotic scaling is driven by the number of Clifford gates. Nevertheless, we find it illuminating to discuss the Clifford and non-Clifford costs separately here.

Using the modified QROM construction presented in Appendix A 3, we can implement an oracle  $O_f$  with  $n$  input qubits and  $m$  output qubits using

$$\text{CLIFFORD}(O_f) \leq (K + o(1)) 2^n m \quad (\text{B14})$$

one- and two-qubit Clifford gates for some constant  $K$ , and

$$\text{TOFFOLI}(O_f) = 2^n \lambda^{-1} + \lambda m - m \quad (\text{B15})$$

Toffoli gates. Therefore,

$$\text{COST}(O_f) \leq (K + o(1)) 2^n m + \Xi(2^n \lambda^{-1} + \lambda m). \quad (\text{B16})$$

In our construction, we implement  $2^k + 1$  smaller data lookups, with  $n - k$  input bits and  $m$  output bits. The overall cost of implementing all of the  $G_\ell$  is given by

$$\text{COST}(\{G_\ell\}) \leq (1 + 2^k) ((K + o(1)) 2^{n-k} m + \Xi(2^{n-k} \lambda^{-1} + \lambda m)). \quad (\text{B17})$$

We claim that the other three costs are asymptotically smaller (as we increase  $n$ ), and so they can be neglected. We address them point by point. We implement  $2^k$  of the  $A_\ell$  circuits as described in Figure 6. Each of these involve some equality checks on  $k$  qubits and controlled swaps on  $n - k$  qubits and  $m$  qubits. The number of Clifford and non-Clifford gates required by this component of the algorithm therefore both scale as  $\mathcal{O}(2^k (n + m))$ . Using our assumption that  $k = \mathcal{O}(\log n)$ , then this scaling is  $\mathcal{O}(\text{Poly}(n))$ , which is dominated by  $2^{n-k} m$ . The reduction to the case where  $x \leq y$  involves inequality checks on registers of size  $n$  and controlled swaps on registers of size  $n$  and  $m$ . The Clifford and non-Clifford costs for this aspect of the algorithm both scale as  $\mathcal{O}(n + m)$ , which is also dominated by  $2^{n-k} m$ . The number of additional swap gates scales as  $\mathcal{O}(n + m)$  and is also negligible. Therefore, we have

$$\text{CLIFFORD}(C_{f,n,m,\lambda,k,1}) \leq (1 + 2^k) (K + o(1)) 2^{n-k} m \quad (\text{B18})$$

and

$$\text{TOFFOLI}(C_{f,n,m,\lambda,k,1}) = (1 + 2^k) (1 + o(1)) (2^{n-k} \lambda^{-1} + \lambda m - m). \quad (\text{B19})$$

Asymptotically then, it is simple to compare the cost of the entire circuit for the two-copy protocol (which we abbreviate as  $C$ ), with two separate QROAM queries ( $O_f^{\otimes 2}$ ). First, let us consider the limit where  $\lambda(n)$  is chosen such that the cost is dominated by the number of Clifford gates. If we assume that the bounds are saturated in both Equation (B16) and Equation (B18), we find that

$$\frac{\text{COST}(O_f^{\otimes 2})}{\text{COST}(C)} \sim \frac{2(2^n m)}{(1 + 2^k)(2^{n-k} m)} \sim \frac{2}{1 + 2^{-k}}. \quad (\text{B20})$$

Therefore, we save a factor of 2 asymptotically. Furthermore, we approach this level of savings quickly as  $k$  increases.

It is interesting to briefly consider what would happen if we focused on a cost model that ignores the cost of Clifford gates, as previous work on fault-tolerant quantum algorithms often has. When we choose  $\lambda$  to minimize the cost of QROAM in this model,<sup>3</sup> we find that implementing  $O_f$  for an  $n$  to  $m$  bit function  $f$  requires a number of Toffoli gates that scales as

$$\text{TOFFOLI}(O_f) \sim 2 \left( 2^{n/2} m^{1/2} \right). \quad (\text{B21})$$

---

<sup>3</sup> We relax the requirement that  $\lambda$  is a power of two for the pur-

poses of simplifying the analysis here.

We claim that the number of non-Clifford gates required to implement  $C$  is dominated by the cost of the  $G_\ell$  in this cost model as well. Taking this claim as true, the  $2^k + 1$  calls to  $G_\ell$  therefore require a number of Toffoli gates that scales as  $2(2^k + 1)2^{(n-k)/2}m^{1/2}$ . We can therefore calculate the cost ratio to find that

$$\frac{\text{TOFFOLI}(O_f^{\otimes 2})}{\text{TOFFOLI}(C)} \sim \frac{4(2^{n/2}m^{1/2})}{2(1+2^k)(2^{(n-k)/2}m^{1/2})} \sim \frac{2^{k/2+1}}{2^k+1}, \quad (\text{B22})$$

which is less than or equal to 1 for all positive  $k$  and asymptotically approaches  $2n^{-1/2}$  when we set  $k = \lceil \log_2 n \rceil$ .

In other words, it always requires fewer Toffoli gates to implement  $O_f^{\otimes 2}$  directly than to use mass production, and so there is no advantage to the two-copy mass production protocol in this cost model. Since the advantage for larger values of  $r$  depends on the two-copy version, there would be no advantage in those cases either. Fundamentally, this is because the strategy for mass production that we pursue relies on the smaller subproblems (the  $G_\ell$  in our case) becoming less costly at the same exponential rate that the number of such subproblems is increasing. This is not the case when we quantify the complexity by the number of non-Clifford gates required. It is an interesting open problem to determine whether or not it is possible to find an advantage in this cost model using a different construction for mass production.

### 3. The inductive step and the proof of Theorem 1

Now that we have discussed the two-copy protocol extensively, we can introduce the more general protocol. Following Uhlig, we construct the protocol for mass-producing  $r = 2^t$  copies recursively [2]. The key insight that enables this recursion is that the basic data loading task we are parallelizing is present in our construction as a subroutine. Specifically, when we use mass production to implement  $O_f^{\otimes 2}$ , we perform a series of calls to the data loading oracles  $G_\ell$ . Furthermore, as we saw, these calls to  $G_\ell$  are the dominant contribution to the overall cost.

Inspired by this observation, we can construct a protocol for mass-producing  $2r$  queries that uses an  $r$ -query mass production protocol (for some  $r = 2^t$ ) as a subroutine. To proceed, we begin by imagining that we implement  $r$  copies of the two-query mass production protocol in parallel. This would accomplish the desired task of implementing  $O_f^{\otimes 2r}$ , but it would not have the desired cost. If we were to proceed this way, then we would find ourselves implementing  $r$  queries to each of the smaller data loading oracles  $G_\ell$  in parallel. Instead of doing this, we simply implement the  $r$  parallel queries to each  $G_\ell$  using an  $r$ -query mass production protocol, as we illustrate in Figure 7 for  $r = 4$ .

Intuitively, if the  $r$ -query mass production scheme scales in the desired way (nearly independent of  $r$ ), then the  $2r$ -query one will as well. This is because, as we discussed, the cost of the two-query scheme is dominated by the cost of implementing the  $G_\ell$ . In order to prove Theorem 1, we will first prove the following lemma by induction:

**Lemma 2** (Quantum Mass Production for quantum read-only memory). Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be arbitrary, and let  $O_f$  be a unitary operator that queries this function in superposition,  $O_f : |x\rangle|\alpha\rangle \rightarrow |x\rangle|\alpha \oplus f(x)\rangle$ . Let  $\lambda$  be a function of  $n$  that returns a power of 2 such that  $\lambda = o\left(\frac{2^{n/2}}{\sqrt{n}}\right)$  and  $1 \leq \lambda \leq 2^{n/2}m^{-1/2}$ , and let  $m$  be a constant function of  $n$ . Let  $k$  be a function of  $n$  that returns a positive integer such that  $k = \mathcal{O}(\log n)$ . For any positive integer  $t$  such that  $kt < n$ , there exists a quantum circuit  $C_{f,n,m,\lambda,k,t}$  with the following properties:

1.  $C_{f,n,m,\lambda,k,t}$  implements  $O_f^{\otimes r}$ , where  $r = 2^t$ .
2.  $C_{f,n,m,\lambda,k,t}$  is composed entirely of one- and two-qubit Clifford gates and Toffoli gates.
3. The number of one- and two-qubit Clifford gates in  $C_{f,n,m,\lambda,k,t}$  is bounded by

$$\text{CLIFFORD}(C_{f,n,m,\lambda,k,t}) \leq (1 + 2^k)^t (K + o(1)) (2^{n-tk}m + \mathcal{O}(n)) \quad (\text{B23})$$

for some universal constant  $K$ .

4. The number of Toffoli gates in  $C_{f,n,m,\lambda,k,t}$  is

$$\text{TOFFOLI}(C_{f,n,m,\lambda,k,t}) = (1 + 2^k)^t (1 + o(1)) (2^{n-tk}\lambda^{-1} + m\lambda + \mathcal{O}(n)), \quad (\text{B24})$$

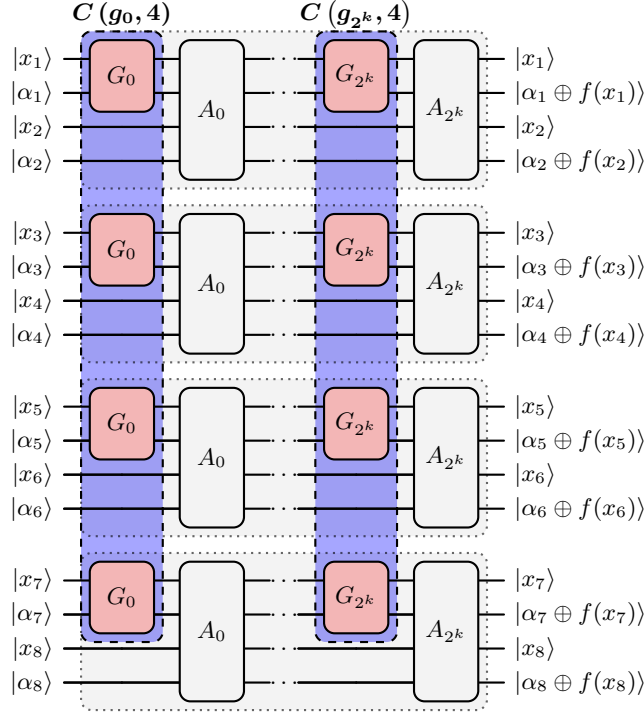


Figure 7: A diagram showing how the  $r = 8$ -query mass production protocol is constructed recursively.

*Proof.* We begin by making the strong inductive assumption that the lemma holds for any  $t' \leq t$ . The base case in our inductive argument follows from the observations made in Equation (B18) and Equation (B19) of Appendix B 2. Taken together, they show that the hypothesis holds for  $t = 1$ . Using this assumption, we will show that our assumption implies that the lemma holds for  $t + 1$  under the condition that  $k(t + 1) < n$ . By induction, this will prove that the lemma is true for all  $t$  such that  $kt < n$ .

Following the construction described above, we imagine implementing the two-copy mass production construction in parallel  $2^t$  times. Note that this involves  $2^t$  parallel calls to each  $G_\ell$  and that our assumption that  $k(t + 1) < n$  implies that  $kt < n - k$ . Therefore our inductive assumption tells us that there exists a circuit  $C_{g_\ell, n-k, m, \lambda, k, t}$  that implements  $G_\ell^{\otimes 2^t}$  for each  $g_\ell$  and that each of these circuits requires a number of Clifford gates that scales as

$$\text{CLIFFORD}(C_{g_\ell, n-k, m, \lambda, k, t}) \leq (1 + 2^k)^t (K + o(1)) (2^{n-k-tk} m + \mathcal{O}(n)). \quad (\text{B25})$$

and a number of Toffoli gates that scales as

$$\text{TOFFOLI}(C_{g_\ell, n-k, m, \lambda, k, t}) = (1 + 2^k)^t (1 + o(1)) (2^{n-k-tk} \lambda^{-1} + m\lambda + \mathcal{O}(n)). \quad (\text{B26})$$

We define the circuit  $C_{f, n, m, \lambda, k, t+1}$  by taking the circuit that implements the two-copy mass production in  $2^t$  times in parallel and replacing the parallel calls to each  $G_\ell$  with the corresponding  $C_{g_\ell, n-k, m, \lambda, k, t}$ . Now we need to show that the number of gates scales appropriately. We begin by considering the number of Clifford gates, finding that

$$\begin{aligned} \text{CLIFFORD}(C_{f, n, m, \lambda, k, t+1}) &\leq \underbrace{(1 + 2^k)^{t+1} (K + o(1)) (2^{n-(t+1)k} m + \mathcal{O}(n))}_{\text{Mass producing the } 2^k+1 \text{ calls to the } G_\ell^{\otimes 2^t}.} \\ &\quad + \underbrace{2^t \mathcal{O}(2^k (n + m))}_{\text{The } 2^t \text{ copies of the control flow circuitry.}} \end{aligned} \quad (\text{B27})$$

The first term comes from accounting for the cost of implementing a call to  $C_{g_\ell, n-k, m, \lambda, k, t}$  for each of the  $2^k + 1$  different data lookups  $G_\ell$ , i.e., we multiply Equation (B25) by  $2^k + 1$ . The second term follows from multiplying the complexity of the control flow circuitry for a single two-query mass production protocol by  $2^t$  (see Appendix B 2



for more details). Because  $2^t 2^k < (1 + 2^k)^{t+1}$  and  $n + m = \mathcal{O}(n)$ , the second term can be absorbed into the  $(1 + 2^k)^{t+1} K \mathcal{O}(n)$  component of the first term. As a result, we have

$$\text{CLIFFORD}(C_{f,n,m,\lambda,k,t+1}) \leq (1 + 2^k)^{t+1} (K + o(1)) (2^{n-(t+1)k} m + \mathcal{O}(n)). \quad (\text{B28})$$

This is the desired inequality for the number of Clifford gates in the  $t + 1$  case.

Now we turn our attention to the number of Toffoli gates. Performing the same calculation in this case, we have

$$\begin{aligned} \text{TOFFOLI}(C_{f,n,m,\lambda,k,t+1}) = & \underbrace{(1 + 2^k)^{t+1} (1 + o(1)) (2^{n-(t+1)k} \lambda^{-1} + \lambda m + \mathcal{O}(n))}_{\text{Mass producing the } 2^{k+1} \text{ calls to the } G_\ell^{\otimes 2^t}} \\ & + \underbrace{2^t \mathcal{O}(2^k (n + m))}_{\text{The } 2^t \text{ copies of the control flow circuitry}}, \end{aligned} \quad (\text{B29})$$

where the first term arises from mass-producing the calls to the  $G_\ell$  and the second term arises from having  $2^t$  copies of the control flow circuitry used by the two-copy protocol. Following the same steps as the Clifford case, we can absorb the second term, yielding

$$\text{TOFFOLI}(C_{f,n,m,\lambda,k,t+1}) = (1 + 2^k)^{t+1} (1 + o(1)) (2^{n-(t+1)k} \lambda^{-1} + \lambda m + \mathcal{O}(n)). \quad (\text{B30})$$

as desired, completing the proof of the lemma.  $\square$

Having proved Lemma 2, we are ready to prove Theorem 1, which we recall below. Our proof is straightforward, essentially using the assumptions in the statement of Theorem 1 and a particular choice for  $k$  to simplify the Clifford and Toffoli counts of Lemma 2 further.

**Theorem 1** (Quantum Mass Production for quantum read-only memory). Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be arbitrary, and let  $O_f$  be a unitary operator that queries this function in superposition,  $O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle$ . Let  $\lambda$  be a function of  $n$  that returns a power of 2 such that  $\lambda = o\left(\frac{2^{n/2}}{\sqrt{n}}\right)$  and  $1 \leq \lambda \leq 2^{n/2} m^{-1/2}$ , and let  $m$  be a constant function of  $n$ . For any  $r$  which is a power of two such that  $r = 2^{o\left(\frac{n-2\log_2 \lambda}{\log_2 n}\right)}$ , there exists a quantum circuit  $C$  that implements  $O_f^{\otimes r}$  and satisfies the following properties:

1.  $C$  is composed entirely of one- and two-qubit Clifford gates and Toffoli gates.
2. The number of one- and two-qubit Clifford gates in  $C$  is bounded by

$$\text{CLIFFORD}(C) \leq (K + o(1)) 2^n m \quad (7)$$

for some universal constant  $K$ .

3. The number of Toffoli gates in  $C$  is  $\text{TOFFOLI}(C) = (1 + o(1)) 2^n \lambda^{-1}$ .

*Proof.* We set  $k = \lceil \log_2 n \rceil$  and recall the assumption that  $r = 2^{o\left(\frac{n-2\log_2 \lambda}{\log_2 n}\right)}$ , or, equivalently,  $t = o\left(\frac{n-2\log_2 \lambda}{\log_2 n}\right)$ . Rearranging Equation (B23) by pulling out and cancelling a factor of  $2^{kt}$ , we find that

$$\text{CLIFFORD}(C_{f,n,m,\lambda,k,t}) \leq (1 + 2^{-k})^t (K + o(1)) (2^n m + 2^{kt} \mathcal{O}(n)). \quad (\text{B31})$$

Taking advantage of the fact that  $1 + x < e^x$  for  $x > 0$ , we have  $(1 + x)^t < e^{tx}$  for any  $x, t > 0$ , so  $(1 + 2^{-k})^t < e^{\frac{t}{2^k}} \leq e^{\frac{t}{n}}$ . The assumption that  $t = o\left(\frac{n-2\log_2 \lambda}{\log_2 n}\right)$  implies that  $\frac{t}{n} = o(\log_2^{-1} n)$ , so  $e^{\frac{t}{n}} = 1 + o(1)$ , and therefore  $(1 + 2^{-k})^t = 1 + o(1)$ . Asymptotically,  $2^{kt} \mathcal{O}(n)$  is bounded by  $2^{kt+c\log_2 n}$  for some constant  $c$ . Since  $kt = o(n - 2\log_2 \lambda) = o(n)$ , we also have  $kt + c\log_2 n = o(n)$  and therefore  $2^{kt} \mathcal{O}(n)$  is dominated by  $2^n m$ . Applying both of these simplifications yields

$$\text{CLIFFORD}(C_{f,n,m,\lambda,k,t}) \leq (K + o(1)) 2^n m. \quad (\text{B32})$$

Beginning a similar analysis of Equation (B24), we have

$$\text{TOFFOLI}(C_{f,n,m,\lambda,k,t}) = (1 + 2^{-k})^t (1 + o(1)) (2^n \lambda^{-1} + 2^{kt} \lambda m + 2^{kt} \mathcal{O}(n)). \quad (\text{B33})$$

Following the same reasoning as above, since  $(1 + 2^{-k})^t = 1 + o(1)$ , we have

$$\text{TOFFOLI}(C_{f,n,m,\lambda,k,t}) = (1 + o(1)) (2^n \lambda^{-1} + 2^{kt} \lambda m + 2^{kt} \mathcal{O}(n)). \quad (\text{B34})$$

In order to establish that the  $2^{kt} \mathcal{O}(n)$  term is superfluous, we note that  $kt = o(n)$  implies that  $2^{kt} \mathcal{O}(n)$  is dominated by  $2^{n/2} m$ . From our assumption that  $\lambda \leq 2^{n/2} m^{-1/2}$ , we have that  $2^n \lambda^{-1} \geq 2^{n/2}$ , and therefore the  $2^{kt} \mathcal{O}(n)$  term can be safely dropped. Finally, using our assumption on  $t$ , we have  $2^{kt} \lambda m = 2^{o(n - \log_2 \lambda)}$ , implying that the contribution from this term is dominated by  $2^n \lambda^{-1}$ . Therefore we can further simplify to yield

$$\text{TOFFOLI}(C_{f,n,m,\lambda,k,t}) = (1 + o(1)) 2^n \lambda^{-1}, \quad (\text{B35})$$

as desired, completing the proof of the theorem.  $\square$

#### 4. Maximizing the improvement factor

Taking  $\lambda$  to be a constant and assuming that Clifford costs saturate the inequalities in Equation (B14) and Equation (B32), Theorem 1 implies that we can implement  $O_f^{\otimes r}$  for a cost that is asymptotically equal to the cost of implementing  $O_f$  for any  $r$  such that  $r = 2^{o(n/\log n)}$ . Recall the definition of the improvement factor  $\mathcal{I}$ ,

$$\mathcal{I} = \frac{\text{COST}(O_f^{\otimes r})}{\text{COST}(C_{f,t})} = \frac{r \text{COST}(O_f)}{\text{COST}(C_{f,t})}, \quad (\text{B36})$$

where we drop the irrelevant subscripts and use  $C_{f,t}$  to denote the circuit that mass-produces  $O_f^{\otimes r}$  for some  $r = 2^t$ . When  $\text{COST}(O_f) \sim \text{COST}(C_{f,t})$ , we simply have  $\mathcal{I} \sim r$ . As a consequence of the restriction that  $r = 2^{o(n/\log n)}$ ,

$$\mathcal{I} = o(\text{Poly}(2^n)). \quad (\text{B37})$$

In other words, the improvement factor we obtain by applying Theorem 1 is subpolynomial in the total size of the data set represented by  $f$ .

Here we consider optimizing the mass production protocol to obtain a larger improvement factor. This is a different aim than the one that motivated Theorem 1 and prior works on mass production theorems [1–3]. These formulations of mass production focused on implementing a large number of parallel operations for a cost that is asymptotically proportional to the cost of implementing the same operation once. Here we relax the requirement that the mass production circuit has a cost that is asymptotically proportional to the cost of a single copy and instead aim to maximize the improvement factor. This optimization is captured in the following proposition:

**Proposition 3** (Mass production with a larger number of copies). Let  $n$  be a positive integer such that  $n > a$ , where  $a$  is a positive integer and a constant function of  $n$ . Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be arbitrary, with  $m$  a constant function of  $n$ . Let  $O_f$  be a unitary operator that queries this function in superposition,  $O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle$ . There exists a circuit  $C$  composed of one- and two-qubit Clifford gates and Toffoli gates implementing  $O_f^{\otimes r}$  for  $r = 2^{n-a}$  such that

$$\text{COST}(C) = \tilde{\Theta}(3^n), \quad (\text{B38})$$

where the tilde notations indicates that we neglect logarithmic factors.

*Proof.* Our proof proceeds by applying Lemma 2 with  $k = 1$ . This choice maximizes the number of times we can recursively apply mass production while still offering a reasonable reduction in cost. For simplicity, we will also choose  $\lambda$  such that the Clifford costs dominate by taking  $\lambda = 2^{n/4}$ . For any positive integer  $t < n$ , Lemma 2 establishes the existence of a circuit  $C$  with the following properties:

1.  $C$  implements  $O_f^{\otimes r}$  where  $r = 2^t$ .

2.  $C$  is composed of one- and two-qubit Clifford gates and Toffoli gates.
3. The cost of  $C$  is given by the expression

$$\text{COST}(C) \leq 3^t (K + o(1)) (2^{n-t} m + \mathcal{O}(n)). \quad (\text{B39})$$

Now we take  $t = n - a$ , yielding

$$\text{COST}(C) \leq 3^{n-a} (K + o(1)) (2^a m + \mathcal{O}(n)) = 3^n \mathcal{O}(n). \quad (\text{B40})$$

We make the conservative assumption that  $\text{COST}(C)$  saturates the inequality (note that we could always add more gates to  $C$  to make this true without loss of generality). We therefore achieve the desired scaling, completing the proof.  $\square$

When we take  $r = 2^{n-a}$  for some suitable constant  $a$  and apply Proposition 3, we find that the improvement factor is

$$\mathcal{I} = \frac{r \text{COST}(O_f)}{\text{COST}(C)} = \tilde{\Theta}\left(\frac{4^n}{3^n}\right) = \tilde{\Theta}\left(2^{n(2-\log_2 3)}\right). \quad (\text{B41})$$

Note that since  $a$  is a constant we can essentially omit it from the expression in Equation (B41). Using this approach, we are able to obtain an improvement factor that scales polynomially in  $2^n$ . A simple calculation reveals that the improvement factor has a negative derivative for all  $k > 1$ , under the assumption that  $t = (n - a)/k$ . This raises the question: is possible to obtain a polynomial improvement factor with an exponent larger than  $2 - \log_2(3) \approx .415$  by some other construction?

## 5. Mass production for QROM implies mass production for state preparation and unitary synthesis

Nearly-optimal methods for approximately preparing arbitrary states and approximately implementing arbitrary unitaries have been known for some time [11, 46]. In order to provide the best known asymptotic scaling, we can rely on the construction for state preparation recently introduced in Ref. 47. Ref. 47's results are focused on achieving the optimal asymptotic scaling with respect to the number of non-Clifford T gates. To accomplish this goal, they design a state preparation protocol that uses  $\mathcal{O}(1)$  QROM calls, using the SelectSwap techniques of Ref. 11 to implement these calls with the smallest possible T gate cost. While their results are framed in terms of the overall T-gate complexity, it is straightforward to extract the number of QROM calls and the complexities of the other steps from their exposition.

Ref. 47's main results on state preparation assume that one is preparing a state with real amplitudes, and they separately show how to efficiently implement an arbitrary diagonal unitary to address the general case. According to Lemma 3.5 and the proof of Theorem 1.1, the state preparation requires:

- $\mathcal{O}(1)$  reflections about the zero state on  $n + \log \log(1/\epsilon) + \mathcal{O}(1)$  qubits, which can be implemented using  $n + \log \log(1/\epsilon) + \mathcal{O}(1)$  one- and two-qubit Clifford gates and T gates;
- $\mathcal{O}(1)$  reflections about a smaller state;
- $\mathcal{O}(n)$  Hadamard gates;
- $\mathcal{O}(1)$  QROM reads with  $n + \log \log(1/\epsilon)$  input bits and one output bit;
- A layer of single-qubit gates that can be synthesized using  $\log(1/\epsilon)$  T gates altogether.

Implementing the diagonal unitary to the necessary precision requires (according to Theorem 1.2):

- $\mathcal{O}(\log(1/\epsilon))$  one- and two-qubit gates from a Clifford + T gateset.
- $\mathcal{O}(1)$  QROM reads with  $n$  input bits and  $\mathcal{O}(\log(1/\epsilon))$  output bits.

Now let us consider the cost of implementing  $r$  copies of this state preparation procedure in parallel for some  $r = 2^{o(n/\log n)}$  that is a power of two. We will use Theorem 1 to mass-produce the QROM calls, setting  $\lambda$  to be a constant for simplicity. The number of Clifford gates dominates the gate complexity of the QROM calls, and  $2^n m$

is  $2^n \log(1/\epsilon)$  for both types of call, so the overall cost from the QROM calls is  $\mathcal{O}(2^n \log(1/\epsilon))$  gates<sup>4</sup>. The non-QROM steps of each state preparation routine have a gate complexity that is bounded by  $n + \log(1/\epsilon)$ , so the overall cost from these non-mass-producible components of the  $r$  parallel implementations is  $\mathcal{O}(2^{o(n/\log n)}(n + \log(1/\epsilon)))$ , which is dominated by the complexity of the QROM calls. Therefore, the overall cost to implement  $r$  arbitrary state preparation routines in parallel is

$$\text{COST}(|\psi\rangle^{\otimes r}) = \mathcal{O}(2^n \log(1/\epsilon)). \quad (\text{B42})$$

There are a variety of ways to show a similar result for the parallelization of arbitrary unitary synthesis using the ability to mass-produce QROM reads. For example, Ref. 11 explains a technique for reducing unitary synthesis to ( $\approx 2^n$  calls to) arbitrary state preparation. However, another approach we can take is to follow Kretschmer, who explains in the proof of Theorem 2 in Ref. 3 how an arbitrary unitary on  $n$  qubits can be implemented as a product of  $2^n - 1$  controlled single-qubit rotations, each with  $n - 1$  controls. Using the same diagonal unitary implementation as above (from Theorem 1.2 of Ref. 47), we can implement each single-qubit rotation up to an error  $\epsilon$  in the spectral norm using  $\mathcal{O}(\log(1/\epsilon))$  one- and two-qubit Clifford gates and T gates, together with  $\mathcal{O}(1)$  QROM reads from  $n$  input bits to  $\mathcal{O}(\log(1/\epsilon))$  output bits. Taking  $r = 2^{o(n/\log n)}$  as above and letting  $\lambda$  be a constant, we can mass-produce the QROM reads and implement an arbitrary  $U^{\otimes r}$  to within a precision  $\epsilon'$  by setting the error for each single-qubit rotation to  $\epsilon = \epsilon'/2^n$  for a cost

$$\text{COST}(U^{\otimes r}) = \mathcal{O}(r2^n \log(2^n/\epsilon') + 4^n \log(2^n/\epsilon')) = \mathcal{O}(4^n(n + \log(1/\epsilon'))). \quad (\text{B43})$$

### Appendix C: The gate complexity of Kretschmer's mass production

The main body of our work examined the use of the classical mass production theorem of Uhlig [2] and in Appendix B5, we discussed how this result alone can be used to enable mass production of quantum states and unitaries through QROM constructions. However, there are cases where a direct application of the mass production theorem may be preferable due to memory considerations or other concerns. Furthermore, it is instructive to compare the costs achievable through a QROM-based approach with the costs achievable when using prior work directly. As we shall see, our mass production results for state preparation actually have a slightly lower cost than those of Ref. 3 due to the fact that we were able to build on the state preparation techniques of Ref. 47. Here we provide an extension of the analysis in Ref. 3 goes beyond counting the number of CNOT gates and specifically accounts for the number of gates in a discrete Clifford + T gate set. This is relevant especially for settings where CNOT operations do not dominate the overall cost such as in both magic state cultivation and traditional magic state distillation [22].

The main results of Ref. 3 are captured in the following theorems:

**Theorem 4.** Let  $|\psi\rangle$  be an  $n$ -qubit quantum state, and let  $r = 2^{o(n/\log n)}$ . Then there exists a quantum circuit with at most  $(1 + o(1))2^n$  CNOT gates that prepares  $|\psi\rangle^{\otimes r}$ .

**Theorem 5.** Let  $U$  be an  $n$ -qubit unitary transformation, and let  $r = 2^{o(n/\log n)}$ . Then there exists a quantum circuit with at most  $(5/2 + o(1))4^n$  CNOT gates that implements  $U^{\otimes r}$ .

By applying standard results for synthesizing single-qubit unitaries, we can obtain the following corollaries:

**Corollary 6.** Let  $|\psi\rangle$  be an  $n$ -qubit quantum state. There exists a quantum circuit using at most  $24(1 + o(1))2^n(n + \log_2(6/\epsilon))$  Hadamard, T, and CNOT gates that prepares the state  $|\bar{\psi}\rangle^{\otimes r}$ , where  $|\bar{\psi}\rangle$  is a state such that  $\| |\psi\rangle - |\bar{\psi}\rangle \| < \epsilon$  in the Euclidean norm. Similarly, for any  $n$ -qubit unitary  $U$ , there exists a quantum algorithm that uses  $60(1 + o(1))4^n(2n + \log_2(15/\epsilon))$  Hadamard, T, and CNOT gates to implement  $\bar{U}^{\otimes r}$ , where  $\|U - \bar{U}\| < \epsilon$  in the operator norm.

Note that the number of gates required for parallel state preparation using this approach is asymptotically larger than the result we obtain using QROM-based methods (see Equation (B42)).

<sup>4</sup> Note that even though  $m$  is assumed to be a constant in the statement of Theorem 1, we can apply the results here without a

loss of rigor by performing  $\log(1/\epsilon)$  QROM calls with a constant number of output bits.



*Proof.* The proof immediately follows from the following observation: Any quantum circuit on  $n$ -qubits composed of single-qubit gates and  $L$  CNOT gates can be written as a product of at most  $n + 2L$  arbitrary single-qubit gates and  $L$  CNOT gates. To see this, consider the following canonical form for a circuit composed of  $L$  CNOT gates:

$$\left( \prod_{j=1}^L (U_{x_j} \otimes U_{y_j}) \text{CNOT}_{x_j, y_j} \right) \left( \bigotimes_{i=1}^n U_i \right), \quad (\text{C1})$$

where we use the notation  $U_i$  to denote a single-qubit unitary acting on the  $i$ th qubit. Any circuit composed of single-qubit unitaries and CNOT gates can be put into this form by merging single-qubit unitaries together to the right until they are obstructed by a CNOT gate or the end of the circuit is reached.

We can make use of this canonical form to count the number of gates required in a discrete Clifford + T gate set. Recall that at most  $4 \log_2(1/\delta) + O(1)$  Hadamard and T gates are needed to approximate a single-qubit rotation about the Z axis to within a factor of  $\delta$  in the operator norm (which is also an induced Euclidean-norm) [48]. A single qubit rotation consists of at most 3 single qubit rotations using an Euler angle decomposition, which implies that  $12 \log_2(3/\delta) + O(1)$  Hadamard and T gates are sufficient to provide a  $\delta$ -accurate approximation to an arbitrary single-qubit rotation. The canonical circuit consists of  $n + 2L$  single qubit rotations and  $2L$  CNOT gates and therefore we can bound the number of gates required for an overall precision of  $\epsilon$  by setting  $\delta = \epsilon/(n + 2L)$ , yielding

$$(n + 2L)(12 \log_2(3(n + 2L)/\epsilon) + O(1)). \quad (\text{C2})$$

To calculate the total number of gates in a discrete gate set for Kretschmer's mass production, we set the  $n$  in Equation (C2) to  $rn$ , and  $L$  to  $(1 + o(1))2^n$ , finding that the overall number of gates required in a Clifford + T gate set is bounded by

$$24(1 + o(1))2^n (n + \log_2(6/\epsilon)). \quad (\text{C3})$$

For the case of unitary synthesis  $L = (5/2 + o(1))4^n$  [3], and so the number of gates needed is bounded by

$$60(1 + o(1))4^n (2n + \log_2(15/\epsilon)), \quad (\text{C4})$$

completing the proof.  $\square$

#### Appendix D: Additional supporting data and computational details

In the main text we presented some numerical data comparing the cost of our mass production protocol with the cost of a naive implementation of parallel data loading. In this section, we provide some additional details on our numerical implementations as well as some supplementary data in support of our conclusions.

In our asymptotic analysis we argued that the cost of the data loading subroutines  $G_\ell$  dominates the cost of mass production. Asymptotically this is true, but it would be useful to understand how true this is for implementations at practical sizes. To this end, we take our optimized circuits for mass production and plot the fraction of the costs not due to data loading in Figure 8. Specifically, we calculate the following quantity as a function of the number of input bits for several values of  $\Xi$  and  $r$ ,

$$1 - \frac{\text{COST}(\{G_\ell\})}{\text{COST}(C)}, \quad (\text{D1})$$

where  $\text{COST}(C)$  denotes the cost of the entire mass production protocol and  $\text{COST}(\{G_\ell\})$  denotes the cost of all of the data loading steps involved.

In the main text, we presented some data that shows how the improvement factor available when using mass production depends on the number of input bits. Specifically, in Figure 2, we focused on the case where the number of output bits ( $m$ ) is 40 and the T gate overhead ( $\Xi$ ) is 1. We plotted the improvement factor as a function of the number of input bits ( $n$ ) for different choices of the total number of parallel queries ( $r$ ). In Figure 9, we show similar data for several different values of  $m$  and  $\Xi$ . Each subfigure of this  $4 \times 3$  collection contains a single plot in the same format as Figure 2, with the number of output bits and the T gate overhead specified in the subfigure title. As in our other calculations, we numerically optimized the choice of  $k$  at each step in the recursive construction for mass production as well as the choice of  $\lambda$  in the data lookup subroutines.

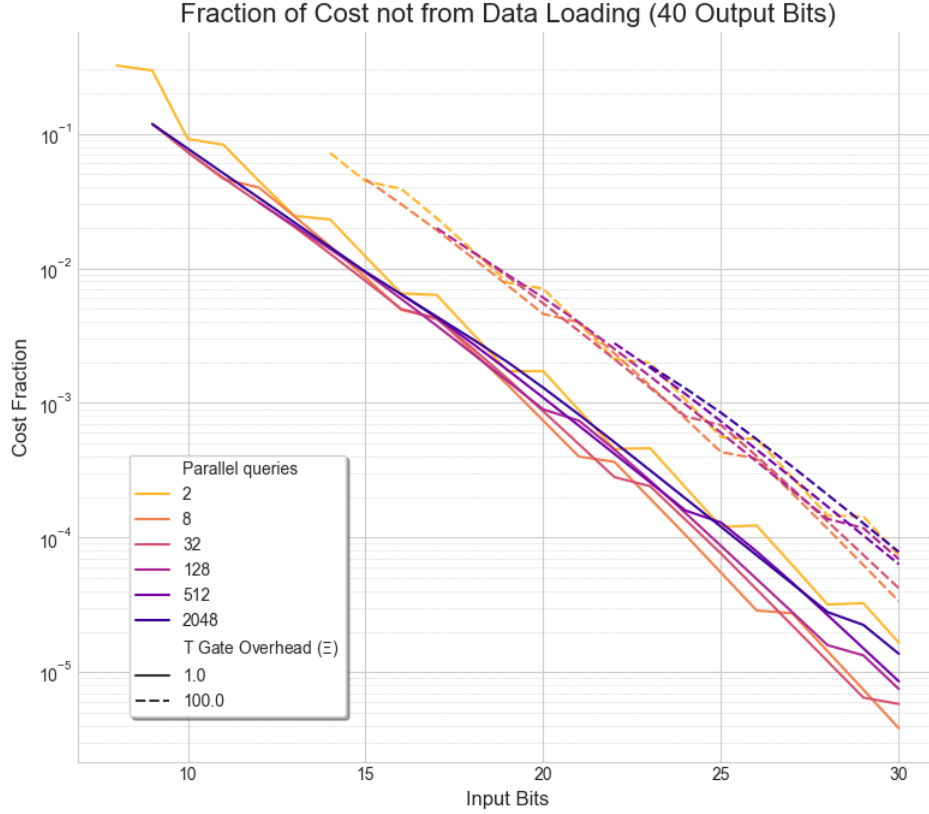


Figure 8: The fraction of costs not due to data loading in our optimized mass production circuits. We fix the number of output bits to 40 and vary the number of input bits ( $n$ ), the T gate overhead ( $\Xi$ ), and the number of parallel copies ( $r$ ).

Our cost analysis was carried out using the Qualtran software package [23], which contains implementations of many of the basic primitives we use, including variants of “SelectSwap” QROM used for the  $G_\ell$  (with minor modification, as detailed in Appendix A 3) and the multicontrolled gates used in the construction of the  $A_\ell$ . For completeness, we mention some details about these implementations. QROM and its variants include a data lookup step that repeatedly flips bits of the output register(s) conditioned on a single control qubit. These multitarget single-control gates can be expressed as a product of CNOT gates, one for each 1 in the corresponding output. When estimating the cost of this step without specific data, Qualtran upper bounds the number of CNOT gates by assuming that there is one for each output qubit.

In our construction of  $A_\ell$ , we repeatedly use multicontrolled gates to help us efficiently flag which of the three cases we are in. By default, Qualtran compiles these multicontrolled gates using a series of “AND” operations, which introduce additional ancilla. However, since these multicontrolled gates have  $k$  controls and we generally take  $k$  to be either 1 or  $\approx \log_2 n$ , this additional ancilla cost is negligible compared to the space required by the input and output registers ( $r(m+n)$ ) and the ancilla space used by the SelectSwap QROM. Furthermore, a slightly more sophisticated implementation could replace our naive use of multicontrolled gates with a strategy more like unary iteration, which would reduce the number of one- and two-qubit gates slightly at the expense of increasing the complexity of the presentation.

For additional details, we refer the reader to our Qualtran implementation of the mass production protocol.

## Appendix E: Applications

We discuss several potential practical applications of quantum mass production in this appendix. We highlight several situations where it is natural to implement some mass-producible operation multiple times in parallel. We focus the discussion around the mass production of QROM queries, although in some cases the applications that we

## Improvement Factor Available with Mass Production

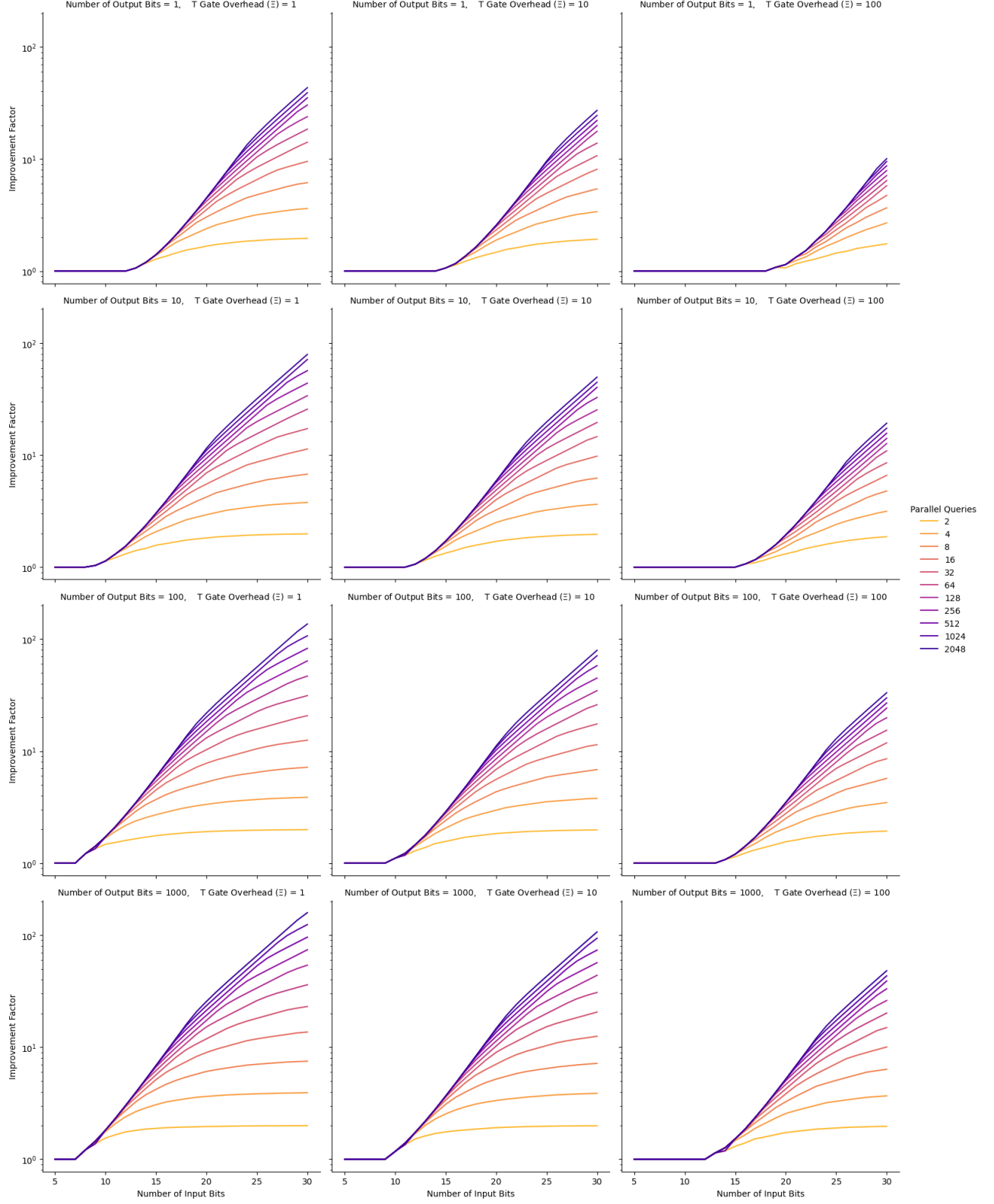


Figure 9: The “improvement factor”  $\mathcal{I}$  when using mass production to implement  $r$  parallel queries to a data loading oracle for an arbitrary function that takes  $n$  input bits and outputs  $m$  bits. We plot the improvement factor for various values of the number of output bits ( $m$ , different rows), the T gate overhead ( $\Xi$ , different columns).

consider could be framed in terms of mass production for state preparation or unitary synthesis. While we present some numerical evidence about the potential for benefitting from mass production in Appendix E 1, we leave the end-to-end analysis of particular applications to a future work.

QROM queries can be used to realize a wide range of quantum oracles. We begin in Appendix E 1 by focusing on a particular well-studied example, the **PREPARE** oracle in a simple formulation of the linear combination of unitaries (LCU) framework. By analyzing previously-published data using the Qualtran software package [5, 23], we provide an example of a natural situation where the cost of QROM queries dominates the overall cost of implementing an algorithm. Building on this, we explain in Appendix E 2 and Appendix E 3 how the combination of parallel phase estimation with quantum mass production can provide a substantial reduction to the overall cost of eigenvalue estimation. Along the way, we discuss the application of quantum mass production to the tensor hypercontraction simulation algorithm, providing a concrete example of a situation where the ability to mass-produce QROM queries appears more powerful than the ability to mass-produce state preparation or unitary synthesis.

We close by discussing a more general example in Appendix E 4, the combination of mass production with amplitude amplification. Specifically, we argue that quantum mass production will enable us to reduce the overall cost of algorithms that require amplifying some small success probability using amplitude amplification. We leave the analysis of particular applications to future work, but we note that there are many quantum algorithms that 1) rely heavily on amplitude amplification and 2) involve oracles whose implementation may require large QROM queries [25–32].

### 1. Constructing block encodings with a linear combination of unitaries

In this section, we briefly introduce the concepts of block encoding and linear combination of unitaries. We refer the reader to Ref. 49 for a more comprehensive introduction. Many variations on these ideas have been explored over the last few years, and we do not attempt to provide a comprehensive review. Instead, we provide enough background to discuss some particular examples that illustrate how these primitives can benefit from quantum mass production.

The block encoding framework is a general approach to representing and manipulating matrices using quantum algorithms. We say that a unitary  $U$  is a block-encoding of a square matrix  $A$  if

$$\left(\langle 0|^k \otimes \mathbb{I}\right) U \left(|0\rangle^k \otimes \mathbb{I}\right) = A/\lambda, \quad (\text{E1})$$

for some  $\lambda > 0$ <sup>5</sup>. Equivalently, we can say that the sub-normalized  $A/\lambda$  is the top left block of  $U$ ,

$$U = \begin{pmatrix} A/\lambda & \cdot \\ \cdot & \cdot \end{pmatrix}. \quad (\text{E2})$$

Access to a block encoding allows us to perform a variety of tasks, many of which can be cast as implementing polynomial functions of the original matrix [49–51].

Block encodings can be obtained in various ways, but one of the most commonly used approaches is to represent the matrix  $A$  as a linear combination of (efficiently implementable) unitaries. We begin with a decomposition of  $A$  as a linear combination of unitaries (LCU),

$$A/\lambda = \sum_{k=0}^{N-1} \alpha_k U_k, \quad (\text{E3})$$

where the  $\alpha_k$  are positive numbers such that  $\sum_{k=0}^{N-1} \alpha_k = 1$ . We define a unitary operator **PREPARE** by its action on the zero state,

$$\text{PREPARE } |0\rangle |0\rangle = \sum_{k=0}^{N-1} \sqrt{\alpha_k} |k\rangle |\text{junk}_k\rangle, \quad (\text{E4})$$

where  $|k\rangle$  is a computational basis state that encodes  $k$  (usually in binary), and  $|\text{junk}_k\rangle$  can be an arbitrary normalized state. Then we define an operator **SELECT**,

$$\text{SELECT} = \sum_k |k\rangle\langle k| \otimes \mathbb{I} \otimes U_k. \quad (\text{E5})$$

---

<sup>5</sup> Block encodings can also be generalized to non-square matrices.

It is straightforward to see that we can combine these two operators to block encode  $A$ ,

$$(|0\rangle\langle 0| \otimes \mathbb{I}) \text{PREPARE}^\dagger \cdot \text{SELECT} \cdot \text{PREPARE} (|0\rangle\langle 0| \otimes \mathbb{I}) = A/\lambda. \quad (\text{E6})$$

One standard way to implement **PREPARE** is to use a technique known as coherent alias sampling [10]. This technique allows one to efficiently approximate the unitary **PREPARE**, preparing a state of the form

$$|\phi\rangle = \sum_{k=0}^{N-1} \sqrt{\tilde{\alpha}_k} |k\rangle |\text{junk}_k\rangle, \quad (\text{E7})$$

where the  $\tilde{\alpha}_k$  are  $\mu$  bit approximations to the  $\alpha_k$  with  $|\alpha_k - \tilde{\alpha}_k| \leq \frac{1}{2^\mu N}$ . The procedure begins by preparing a uniform superposition,  $N^{-1/2} \sum_{k=0}^{N-1} |k\rangle$ , and then uses a single QROM query to coherently load bitstrings  $|\text{alt}_k\rangle$  and  $|\text{keep}_k\rangle$ , resulting in the state

$$N^{-1/2} \sum_{k=0}^{N-1} |k\rangle |\text{alt}_k\rangle |\text{keep}_k\rangle. \quad (\text{E8})$$

These registers are of size  $\lceil \log_2(N) \rceil$ ,  $\lceil \log_2(N) \rceil$ , and  $\mu$  respectively. We then adjoin a final register of size  $\mu$  initialized in a uniform superposition,  $2^{-\mu/2} \sum_{\sigma=0}^{2^\mu-1} |\sigma\rangle$ . To complete the procedure, we swap the first and second registers (containing  $|k\rangle$  and  $|\text{alt}_k\rangle$ ) if  $\text{keep}_k \leq \sigma$ .

As Ref. 10 explains, it is always possible to choose the bitstrings  $\text{alt}_k$  and  $\text{keep}_k$  such that the resulting state takes the form of  $|\phi\rangle$  in Equation (E7). Furthermore, this classical preprocessing can be done sequentially in time that scales roughly linearly with  $N$ . As we discussed in Appendix A 2, the cost of the QROM read is driven by the number of one- and two- qubit Clifford gates, which scales as  $\mathcal{O}(N(\log_2 N + \mu))$ . All other steps scale logarithmically in  $\mu$  and  $N$ , so the cost of coherent alias sampling is dominated by the cost of the QROM read.

The overall cost of implementing a block encoding depends on the cost of both **PREPARE** and **SELECT**. When we implement **PREPARE** using coherent alias sampling and the cost of **PREPARE** dominates the overall cost, then there is significant potential to benefit from applying quantum mass production. To provide a concrete example, we use Qualtran to analyze the fraction of the cost that comes from classical data loading in the sparse simulation algorithm proposed in Ref. 12. The implementation of the **PREPARE** subroutine in this algorithm is slightly more sophisticated than the coherent alias sampling that we described above, but at a high level it works similarly and relies heavily on classical data loading. The quantum chemical Hamiltonian is a two-body operator, so in a naive representation one might expect that a system of size  $N_{orb}$  (the number of spin-orbitals) would require loading  $\mathcal{O}(N_{orb}^4)$  parameters. In the sparse simulation approach of Ref. 12, the cost of the classical data loading is reduced if the Hamiltonian is sparse in a particular basis once very small terms have been dropped. Asymptotically, there are arguments that the number of non-zero terms should scale as  $\mathcal{O}(N_{orb}^2)$  for a sufficiently large system in a properly chosen basis [52]. Regardless of the sparsity, the other components of the algorithm scale nearly linearly in  $N_{orb}$ .

In Figure 10, we consider the task of implementing a block encoding of the quantum chemical Hamiltonian in second quantization using this sparse simulation approach. Specifically, we consider two examples reported in Ref. 5, a chain of Hydrogen atoms represented in an STO-6G basis where the system size was varied by changing the number of atoms, and a square configuration of four Hydrogen atoms where the system size was varied by changing the size of the single-particle basis set. We refer to the first example as the ‘‘Thermodynamic’’ scaling limit and the second as the ‘‘Continuum’’ scaling limit. In both cases, we use the data from Figure 14 of Ref. 5 to fit an equation of the form

$$c = aN_{orb}^b, \quad (\text{E9})$$

where  $c$  is the number of non-zero coefficients,  $N_{orb}$  is the number of spin-orbitals,  $a$  is a free parameter, and  $b$  is taken from the caption of the figure (1.78 and 3.83 for the two limits, respectively). Then, using the cost model described in Appendix A 1, we use the Qualtran software package to calculate the fraction of the cost that comes from the classical data loading subroutine as a function of the system size for various values of  $\Xi$  (the overhead of a T gate compared with a typical Clifford gate). We plot this data in the left panel. In the right panel, we show the number of input bits required for the classical data loading subroutine as a function of system size.

In both scaling limits, Figure 10 reveals that the cost of implementing the block encoding using this algorithm is dominated by the QROM queries at large system sizes and small values of  $\Xi$ . Furthermore, the number of input bits is large enough that we may be able to obtain a significant cost reduction from mass production, although reducing the cost by an order of magnitude or more appears challenging at these system sizes (see Figure 2).



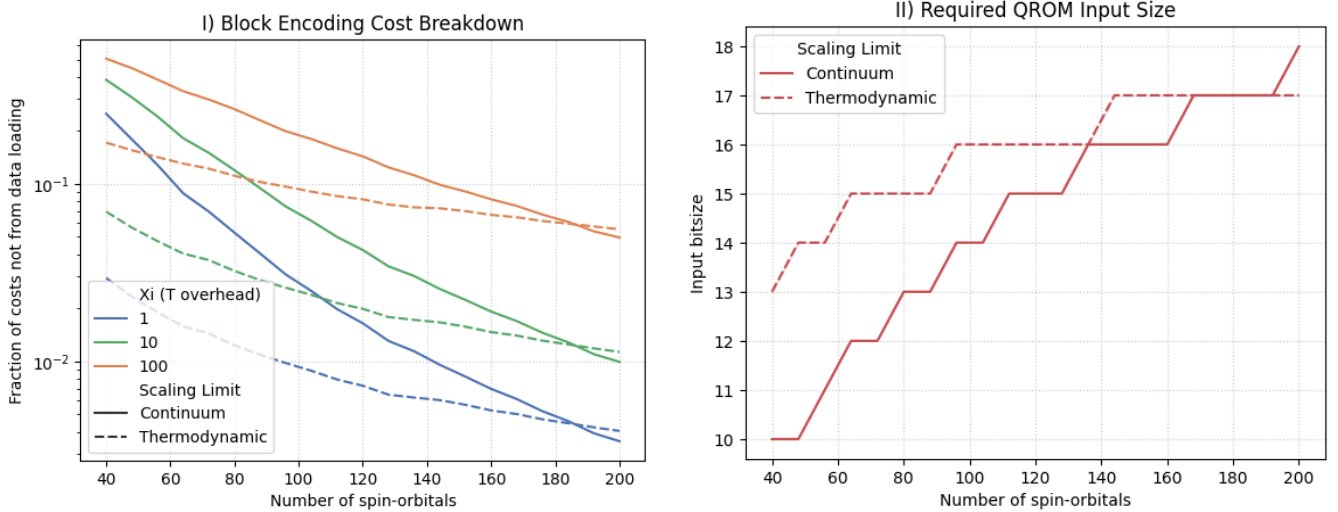


Figure 10: Left Panel (I): The fraction of the cost of block-encoding the quantum chemical Hamiltonian that is not due to data loading when using the sparse simulation algorithm of Ref. 12, for several values of  $\Xi$  (the T gate overhead in the cost model defined in Appendix A1). We vary the system size by adding Hydrogen atoms to a chain (Thermodynamic) or by adding additional diffuse basis functions to a simulation of four Hydrogen atoms in a square geometry (Continuum). In both cases, the number of non-zero matrix elements is obtained from Figure 14 of Ref. 5. Right Panel (II): The number of input bits required by the classical data loading subroutine used to block-encode the Hamiltonians. We find that classical data loading dominates the cost, particularly at larger system sizes and smaller values of  $\Xi$ . Furthermore, the number of input bits grows relatively large. Taken together, this suggests that it is possible to significantly benefit from mass production techniques.

While the linear combination of unitaries approach to constructing a block encoding is frequently used in state-of-the-art quantum algorithms, the actual implementation is often more complicated than the straightforward approach described above. In particular, a significant body of work is dedicated to more advanced techniques for efficiently block-encoding the quantum chemical Hamiltonian [4, 5, 10, 12]. By varying the form of the LCU decomposition as well as the strategies for implementing PREPARE and SELECT, the costs of the block encoding can be dramatically reduced (both in terms of the asymptotic scaling and the constant factors) [5]. Despite this variation, the cost of classical data loading dominates many of the most advanced algorithms for this application. In the most advanced algorithms, this classical data is used for tasks other than arbitrary state preparation and unitary synthesis [4, 5]. This observation highlights the benefit of our approach to quantum mass production, which focuses on parallelizing classical data loading rather than the more specific tasks of state preparation or unitary synthesis.

## 2. Parallel Phase Estimation

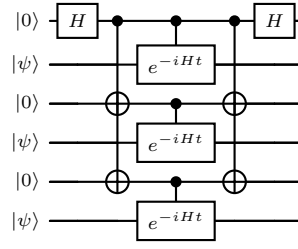
In this section, we discuss how quantum mass production can be combined with parallel phase estimation in order to reduce the gate complexity required for eigenvalue estimation. In its standard form, parallel phase estimation assumes that we have access to a unitary  $U$  and a state such that  $|\psi_0\rangle$  such that

$$U |\psi_0\rangle = e^{i\phi_0} |\psi_0\rangle. \quad (\text{E10})$$

The task is to estimate the phase  $\phi_0$ . Parallel phase estimation differs from standard phase estimation in that it uses  $r$  copies of the initial state to reduce the required depth. Parallel phase estimation does not generally reduce the query complexity or gate complexity of the phase estimation task, although it can enable multiple quantum processors to collaborate on solving the phase estimation task with very little communication overhead.

We illustrate a circuit that implements the parallel version of an iterative phase estimation protocol in Figure 11. This approach begins by first preparing the GHZ state tensored with  $r$  copies of the state  $|\psi_0\rangle$ ,

$$\frac{1}{\sqrt{2}} \left( |0\rangle^{\otimes r} + |1\rangle^{\otimes r} \right) |\psi_0\rangle^{\otimes r}. \quad (\text{E11})$$

Figure 11: Parallel QPE circuit for  $r = 3$  workers.

We then take each qubit in the GHZ state and perform a controlled unitary  $U$  onto the corresponding  $|\psi_0\rangle$  state. This yields the state

$$\prod_{j=1}^r \left( |0\rangle\langle 0|_j \otimes I_j + |1\rangle\langle 1|_j \otimes U_j \right) \frac{1}{\sqrt{2}} \left( |0\rangle^{\otimes r} + |1\rangle^{\otimes r} \right) |\psi_0\rangle^{\otimes r} = \frac{1}{\sqrt{2}} \left( |0\rangle^{\otimes r} + e^{ir\phi_0} |1\rangle^{\otimes r} \right) |\psi_0\rangle^{\otimes r}, \quad (\text{E12})$$

where we use the notation that  $[\cdot]_j$  is an operator acting on the  $j^{\text{th}}$  ancilla qubit or state  $|\psi_0\rangle$  as appropriate. By using  $r$  parallel applications of the controlled form of  $U$ , we apply a phase equal to  $r\phi_0$  to the GHZ state. Uncomputing the GHZ state effectively leaves us with a single control qubit in a state proportional to  $|0\rangle + e^{ir\phi_0} |1\rangle$ . An equivalent experiment performed with the standard iterative phase estimation circuit would require total evolution time  $rt$  instead of  $t$ .

There are many ways that we could perform such a simulation but we will focus on qubitization based approaches here. We assume that we have access to the Hamiltonian as a linear combination of  $N$  different unitaries, as reviewed in Appendix E1. Using the oracles **PREPARE** and **SELECT**, we can construct a “qubitized quantum walk operator”  $W$  of the form

$$W = -\text{SELECT} \left( I - 2 \text{PREPARE} |0\rangle\langle 0| \text{PREPARE}^\dagger \otimes I \right), \quad (\text{E13})$$

where  $|0\rangle$  denotes the combined zero state of both registers in Equation (E4).  $W$  has the property that if  $|\psi_j\rangle$  is an eigenvector of  $H$  with eigenvalue  $E_j$ , then there exist eigenstates  $|\phi_j^\pm\rangle \in \text{span}((\text{PREPARE} |0\rangle) |\psi_j\rangle, W(\text{PREPARE} |0\rangle) |\psi_j\rangle)$  with eigenvalues

$$W |\phi_j^\pm\rangle = e^{\pm i \arccos(E_j/\lambda)} |\phi_j^\pm\rangle, \quad (\text{E14})$$

where  $\lambda$  is the normalization factor of the block encoding. We can therefore directly perform phase estimation on  $W$  in order to determine properties about the spectrum of  $H$ <sup>6</sup>.

The simplest application of quantum mass production in this context is in the application of the **PREPARE** operation. This could be done either through the use of coherent alias sampling and a parallel QROM read, as we discussed in Appendix E1, or by directly using the techniques of Ref. 3 for arbitrary state preparation. Using the coherent alias sampling approach described above and setting  $\mu$  to a constant, then we have that  $\text{COST}(\text{PREPARE}) = \mathcal{O}(N \log N)$ . Assuming that we are working in a regime where we can fruitfully mass-produce  $r$  copies of the QROM read for coherent alias sampling by applying Theorem 1, we can implement  $\text{PREPARE}^{\otimes r}$  for a comparable cost. So the cost of implementing  $W^{\otimes r}$  with mass production is

$$\mathcal{O}(N \log(N) + r \text{COST}(\text{SELECT})), \quad (\text{E15})$$

whereas the cost without mass production would be

$$\mathcal{O}(rN \log(N) + r \text{COST}(\text{SELECT})). \quad (\text{E16})$$

This implies that there is an asymptotic advantage provided that

$$\text{COST}(\text{SELECT}) = o(N \log(N)). \quad (\text{E17})$$

<sup>6</sup> Note that the controlled variant of the walk operator can be found by controlling the **SELECT** operator, which can also be trivially constructed if atleast one of the unitaries has a known

+1 eigenvector through controlled swap operations and a single query.

As we discussed in Appendix E1, the sparse simulation method of Ref. 12 satisfies this requirement. Specifically, for this approach, COST(SELECT) scales as  $\mathcal{O}(N_{orb})$ , and the size of the QROM queries used to implement PREPARE scales as  $N \propto N_{orb}^b$ , where  $b$  is naively 4, but is perhaps 2 in some asymptotic limits. In two particular examples (a Hydrogen chain and a square arrangement of Hydrogen atoms), Ref. 12 numerically estimated  $b \approx 1.78$  and  $b \approx 3.83$  respectively. This allows us substantial room to benefit by taking  $r \gg 1$ .

Naively, one might hope to balance the costs in Equation (E15) by taking  $r \approx N_{orb}^{b-1}$ . However, Theorem 1 restricts us to values of  $r$  such that  $r = 2^{o(\frac{n - \log \lambda}{\log n})}$ . For any choice of  $a > 0$ , taking  $r = N^a$  implies  $r = 2^{n \log a}$ , which does not satisfy the requirement on  $r$ . Nevertheless, we show in Proposition 3 that we can obtain a polynomial advantage when taking  $r = N/2$ , reducing the cost of implementing all  $r$  data lookups from  $\Theta(N^2)$  to  $\Theta(N^{\log_2 3})$ . In other words, the amortized cost per data lookup goes as  $N^{\log_2(3/2)}$ . For  $N \propto N_{orb}^b$ , this means that Proposition 3 implies that we can implement  $W^{\otimes r}$  for  $r = N_{orb}^b/2$  for a cost that scales as

$$\mathcal{O}\left(N_{orb}^{b \log_2 3} \log(N_{orb})\right) + N_{orb}^{b+1}. \quad (\text{E18})$$

Using the sparse simulation method to construct the walk operator, mass production with an appropriate value of  $r$ , we can follow a parallel phase estimation approach to apply the transformation

$$|\phi_j^\pm\rangle^{\otimes r} \mapsto (W^t)^{\otimes r} |\phi_j^\pm\rangle^{\otimes r} = e^{\pm i r t \arccos(E_j/\lambda)} |\phi_j^\pm\rangle^{\otimes r}. \quad (\text{E19})$$

Requiring that the error from iterative phase estimation is  $O(\epsilon)$  with high probability, then the above circuit needs to be repeated a logarithmic number of times using a total evolution at each iteration that is on the order of  $rt \in \tilde{\mathcal{O}}(\lambda_{sparse}/\epsilon)$  [53]. Up to logarithmic factors, the overall cost is therefore

$$\text{COST(Parallel QPE with sparse simulation and mass production)} = \tilde{\mathcal{O}}\left(\frac{\lambda_{sparse}}{\epsilon} \left(N_{orb} + N_{orb}^{b \log_2(3/2)}\right)\right), \quad (\text{E20})$$

which is polynomially smaller than the cost without mass production,

$$\text{COST(Standard QPE with sparse simulation)} = \tilde{\mathcal{O}}\left(\frac{\lambda_{sparse}}{\epsilon} N_{orb}^b\right). \quad (\text{E21})$$

We now briefly explain how mass production may be useful in accelerating this THC-based approach to quantum chemistry simulation, although we refer the reader to Ref. 5 for a complete description of the algorithm. Starting with a standard second-quantized representation of the electronic structure Hamiltonian,

$$H = T + V = \sum_{pq} h_{pq} a_p^\dagger a_q + \sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s, \quad (\text{E22})$$

tensor hypercontraction gives a method of rewriting the tensor describing the Coulomb operator in an approximate form,

$$V \approx \sum_{pqrs} \sum_{\mu, \nu=1}^R \chi_p^{(\mu)} \chi_q^{(\mu)} \chi_r^{(\nu)} \chi_s^{(\nu)} \zeta_{\mu\nu} a_p^\dagger a_q^\dagger a_r a_s. \quad (\text{E23})$$

Empirically, it has been argued that

$$R = \tilde{\mathcal{O}}(N_{orb} \log(1/\epsilon_{\text{THC}})) \quad (\text{E24})$$

is sufficient, where  $\epsilon_{\text{THC}}$  is some error one allows in the THC decomposition.

We claim that the actual cost (including the Clifford gates) of this algorithm is dominated by a QROM read with input size  $\approx R^2$  (and a small number of output bits that scales logarithmically with the desired precision). Furthermore, the next most dominant cost is a QROM read with input size  $\approx R$  and output size  $\approx N_{orb}$ , and all remaining components of the algorithm scale as  $\tilde{\mathcal{O}}(N_{orb})$ . Notably, this second QROM read is not used for arbitrary state preparation (or a similar primitive), but instead it is used to coherently load parameters for a series of controlled rotations. This application offers an example that highlights the power of using mass production for data loading rather than the more specific tasks of state preparation or unitary synthesis.

Mass production can help us reduce the cost of parallel calls to the walk operator in this case as well. Applying Proposition 3 and performing a similar calculation to the one above reveals that we could reduce the amortized cost

of implementing a call to the walk operator in the THC algorithm (in the context of parallel phase estimation) from  $\tilde{\mathcal{O}}(R^2 + RN_{orb} + N_{orb})$  to  $\tilde{\mathcal{O}}(R^{\log_2(9/4)} + R^{\log_2 3/2} N_{orb} + N_{orb})$ . Assuming the empirical scaling for  $R$  in (E24) holds, the overall cost therefore scales as

$$\text{COST}(\text{Parallel QPE with THC and mass production}) = \tilde{\mathcal{O}}\left(\frac{\lambda_{THC}}{\epsilon} N_{orb}^{\log_2 3}\right), \quad (\text{E25})$$

where  $\lambda_{THC}$  is the rescaling factor of the THC block encoding. Note that the QROM read with the smaller input size and larger output size is the dominant cost after applying Proposition 3. Contrast this with the cost available in the standard THC approach,

$$\text{COST}(\text{Standard QPE with THC}) = \tilde{\mathcal{O}}\left(\frac{\lambda_{THC}}{\epsilon} N_{orb}^2\right). \quad (\text{E26})$$

The scaling in Equation (E25) represents a polynomial improvement in the gate complexity for this method when we account, as we do here, for the cost of Clifford gates as well as non-Clifford gates.

The parallel phase estimation considered above takes  $r$  copies of the appropriate eigenstate as an input to the simulation algorithm. This is not a free resource and must be considered in the overall complexity. The simplest way to address this is to use eigenstate filtering to prepare each of the individual eigenstates  $|\psi_j\rangle$ . Let us assume that we are promised that we know the eigenenergy within error  $\Delta$  and assume that the eigenvalue gap is at least  $2\Delta$ . In this case, the cost of performing this preparation within error  $\epsilon$  is given by [30]

$$\tilde{\mathcal{O}}\left(\frac{\lambda_{THC} N_{orb}^2 \log(1/\epsilon)}{\Delta}\right) \quad (\text{E27})$$

which is sub-dominant if we assume that the gap is larger than  $\epsilon$ . This is almost certainly the case if we are given an eigenstate of the Hamiltonian and merely use filtering to prepare a specific eigenstate of the walk operator. In the more general situation, we may or may not have a gap between  $\Delta$  and  $\epsilon$ . The proposed combination of mass production and parallel phase estimation is most likely to be practically useful for examples where  $\Delta \gg \epsilon$ . We could, of course, use mass production to help implement multiple filtering steps in parallel as a prelude to using parallel phase estimation to estimate an eigenvalue.

### 3. Matrix Product State Preparation

Parallel state preparation is a major obstacle in the application of parallel phase estimation. In this section, we consider the use of mass production to accelerate the parallel preparation of several matrix product states. This could serve as a complement or alternative to the phase estimation procedure discussed above, or it may be independently useful. For example, the state-of-the-art approach for initial state preparation for first-quantized quantum algorithms naturally involves parallel calls to the same matrix product state preparation circuits as a subroutine [34].

Matrix product states take the form

$$|\psi\rangle = \sum_{s \in \{0,1\}^n, \{\alpha\}} A_{\alpha_0}^{s_0} A_{\alpha_0 \alpha_1}^{s_1} \cdots A_{\alpha_{n-2}}^{s_{n-1}} |s_0 \dots s_{n-1}\rangle \quad (\text{E28})$$

for a set of two- and three-index tensors  $\{A_{\alpha_0}^{s_0} A_{\alpha_0 \alpha_1}^{s_1} \cdots A_{\alpha_{n-2}}^{s_{n-1}}\}$ . Matrix product states are particularly efficient at representing states of one-dimensional quantum systems [54], as well as amplitude-encoded functions of few variables [34, 55]. The individual dimensions of each of the  $\alpha$  indices are known as the “bond dimension,” which we denote as  $\chi_j$  for the index  $\alpha_j$ . For simplicity, we refer to the bond dimension of a matrix product state by a single number  $\chi$  such that  $\chi_j \leq \chi$  for all  $j$ . Matrix product states can exactly represent any state using an exponentially large bond dimension, and they are frequently used as ansatz for the ground state problem even when they are expected to scale exponentially.

It is also possible to approximately prepare a matrix product state using a quantum circuit whose cost scales polynomially with the bond dimension(s). Specifically, consider the task of preparing a state  $|\phi\rangle$  such that it closely approximates the target matrix product  $|\psi\rangle$ , i.e.,

$$\| |\psi\rangle - |\phi\rangle \| \leq \epsilon \quad (\text{E29})$$

for a chosen  $\epsilon > 0$ . A standard protocol for this task, discussed in Refs. [34, 56, 57] involves preparing an arbitrary state in a space of dimension  $\mathcal{O}(\chi)$  and then implementing a series of  $n - 1$  unitaries of dimension  $\mathcal{O}(\chi)$ . This requires a number of gates that scales as  $\tilde{\mathcal{O}}(n(\chi^2 + \chi \log(1/\epsilon)))$ . Note that optimizations to this process can reduce the number of non-Clifford Toffoli gates needed down to  $\tilde{\mathcal{O}}(n\chi^{3/2} \log(1/\epsilon))$  [34, 56, 57]. However, as argued previously this reduction may not be as useful in settings such as magic state cultivation where the costs of Clifford and non-Clifford gates are approximately equivalent.

Because this protocol for matrix product state preparation reduces the task to arbitrary state preparation and unitary synthesis, we can clearly accelerate it using mass production. The states and unitaries that we require act on spaces of dimension at most  $\chi$ , so we can benefit from mass-producing the most expensive such steps  $r$  times in parallel, provided that

$$r \in 2^{o(\log(\chi)/\log \log(\chi))}. \quad (\text{E30})$$

The procedures outlined in Ref. [56], Ref. [34], and Ref. [57] already use QROM for the state preparation and unitary synthesis steps, so we could either apply our theorem to parallelize those QROM calls, or apply the results of Ref. [3] directly. In either case, we expect that a total of  $\tilde{\mathcal{O}}(\chi^2 \log(1/\epsilon))$  gates will be required.

Recent work has found that using matrix product states with very high bond dimensions as initial states for quantum phase estimation is a promising route towards practical quantum advantage for quantum chemical simulation [56, 57]. Ref. [57]’s findings suggest that the cost of matrix product state preparation is moderately smaller than the cost of phase estimation for some of the smallest molecular systems that may be beyond the reach of classical methods. For larger systems, the cost of matrix product state preparation is expected to grow exponentially, so it will be useful to be able to reduce its cost using mass production techniques. This could be useful in the context of parallel phase estimation, or when it is necessary to perform several separate phase estimation experiments (on separate initial states) in order to have a high probability of obtaining the ground state energy.

#### 4. Parallel execution with amplitude amplification of success

In this section, we consider the combination of our mass production protocol with amplitude amplification. We consider an  $n$ -qubit unitary  $A$  such that

$$A |0^{\otimes n}\rangle = \sqrt{p} |G\rangle |1\rangle + \sqrt{1-p} |B\rangle |0\rangle, \quad (\text{E31})$$

where  $|G\rangle$  is some arbitrary “good” state and  $|B\rangle$  is some arbitrary “bad” state. The usual task of amplitude amplification is to use queries to  $A$  and  $A^\dagger$  to prepare  $|G\rangle$  with some constant success probability greater than, say,  $1/2$ . This can be done using  $\approx \frac{1}{\sqrt{p}}$  queries to  $A$  and  $A^\dagger$ . We focus on the simplest case, where  $p$  is known ahead of time, although the more general case where  $p$  is unknown can be handled with a constant factor increase in cost [58]. Before we explain how to speed up amplitude amplification using quantum mass production, we note that query lower bounds for related problems, such as Grover’s search algorithm [58, 59], will not apply in the models that we consider. This is because implementing quantum mass production necessarily makes use of “white-box” access to the function being mass-produced.

We begin by making the assumption that we can fruitfully mass-produce up to  $r_{\max}$  queries to  $A$ . Concretely, we assume that, there is a mass production protocol  $C(A, r)$  such that  $C(A, r) = A^{\otimes r}$  and

$$\text{COST}(C(A, r)) \approx \text{COST}(A) \quad (\text{E32})$$

for any  $r \leq r_{\max}$ , where  $r$  and  $r_{\max}$  are both powers of two. Similarly, we assume that we can mass-produce queries to  $A^\dagger$  under the same conditions. Furthermore, we make the assumption that  $pr_{\max} \ll 1$  in order to focus on the most interesting case.

Consider the action of  $A^{\otimes r}$  on the appropriate zero state,

$$A^{\otimes r} |0^{\otimes rn}\rangle = \bigotimes_{i=1}^r \left( \sqrt{p} |G\rangle |1\rangle + \sqrt{1-p} |B\rangle |0\rangle \right). \quad (\text{E33})$$

Let  $|\mathcal{G}\rangle$  denote the “good” state obtained by expanding this tensor product, collecting all of the terms where at least one of the copies is in the state  $|G\rangle |1\rangle$ , and normalizing. Likewise, let  $|\mathcal{B}\rangle$  denote the “bad” state  $|\mathcal{B}\rangle = \bigotimes_{i=1}^r |B\rangle |0\rangle$ . Finally, let  $\mathcal{A}_r$  denote the unitary that applies  $A^{\otimes r}$  and then uses a multi-qubit OR gate to flip an ancilla qubit if any

of the  $A$ s have successfully prepared a  $|G\rangle$  state. We assume that the cost of this OR operation is negligible compared to the cost of implementing  $A$ . Then we can write

$$\mathcal{A}_r |0^{\otimes nr+1}\rangle = \sqrt{p_r} |\mathcal{G}\rangle |1\rangle + \sqrt{1-p_r} |\mathcal{B}\rangle |0\rangle, \quad (\text{E34})$$

where  $p_r$  is implicitly defined by the expression  $\sqrt{1-p_r} = (\sqrt{1-p})^r$ . Solving for  $p_r$ , we have

$$p_r = \sum_{i=1}^r (-1)^{i+1} \binom{r}{i} p^i = rp + \mathcal{O}(r^2 p^2). \quad (\text{E35})$$

In order to combine mass production with amplitude amplification, we simply apply amplitude amplification to  $\mathcal{A}_r$  to amplify the probability of obtaining  $\mathcal{G}$ . Once we have obtained  $\mathcal{G}$ , we can either measure the individual ancilla qubits to determine which register is in the state  $|G\rangle$ , or sort the registers according to their ancilla values (if we wish to avoid non-unitary operations). Producing at least one copy of  $|G\rangle$  with high probability using this approach requires  $\approx \frac{1}{\sqrt{rp}}$  queries to  $\mathcal{A}_r$ . By assumption, the cost for implementing  $\mathcal{A}_r$  is comparable to the cost of implementing  $A$  for any  $r \leq r_{\max}$ . Therefore, mass production allows us to reduce the costs for preparing  $|G\rangle$  by a factor of  $\sqrt{r_{\max}}$ .

**Corollary 7.** Let  $A$  be a quantum algorithm that makes no measurements, implemented using  $q$  queries to an oracle  $O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle$  for some known  $f : \{0,1\}^n \rightarrow \{0,1\}^m$ , together with at most  $c \geq 1$  additional Clifford and Toffoli gates. Furthermore, let  $A|0\rangle = \sqrt{p} |\mathcal{G}\rangle |1\rangle + \sqrt{1-p} |\mathcal{B}\rangle |0\rangle$  for some  $p < 1$ , and assume that  $A$  uses at most  $n'$  qubits. For any  $r$  that satisfies  $r = 2^{o(n/\log n)}$  and  $rp \in o(1)$ , there exists an algorithm that prepares a copy of the state  $|G\rangle$  with probability  $1 - \delta$  using  $\mathcal{O}\left(\frac{\log(1/\delta)}{\sqrt{rp}} (2^n mq + rc)\right)$  Clifford and Toffoli gates. This algorithm uses at most  $rn' + 1$  qubits.

*Proof.* We can implement the transformation in Equation (E34) by combining  $r$  parallel queries to  $A$  with an OR operation that flags the success of any of the individual calls to  $A$ . This requires exactly  $rn' + 1$  qubits and the OR operation requires a number of gates that scales linearly in  $r$ . Rather than implementing each call to  $O_f^{\otimes r}$  directly, we can apply Theorem 1 to do so with a number of gates that scales as  $\mathcal{O}(2^n m)$  (taking  $\lambda$  to be a constant). Since  $q$  queries need to be made to  $O_f$ , the cost of  $\mathcal{A}_r$  is  $q$  times the cost of querying  $O_f^{\otimes r}$  followed by at most  $rc$  additional Clifford and Toffoli gates where  $c \geq 1$  is the additional Clifford and Toffoli gates needed by  $A$  in addition to query operations. The overall number of gates required to implement  $\mathcal{A}_r$  is therefore

$$\text{COST}(\mathcal{A}_r) = \mathcal{O}(2^n mq + rc) \quad (\text{E36})$$

We can use standard fixed-point amplitude amplification techniques to prepare  $|\mathcal{G}\rangle$  using  $\mathcal{O}\left(\frac{\log(1/\delta)}{\sqrt{p_r}}\right)$  invocations of  $\mathcal{A}_r$  and its inverse [60] together with  $\mathcal{O}\left(\frac{\log(1/\delta)}{\sqrt{p_r}}\right)$  additional gates. By assumption,  $rp \in o(1)$  so we have from (E35)  $p_r \sim rp$ , implying that  $\mathcal{O}\left(\frac{\log(1/\delta)}{\sqrt{rp}}\right)$  calls to  $\mathcal{A}_r$  are sufficient. Multiplying the number of calls to  $\mathcal{A}_r$  by the number of gates required to implement it yields the claimed complexity.  $\square$

## Appendix F: Mass production for serial operations

Quantum mass production is most useful when it is natural to parallelize calls to the same data loading oracle. However, we show in this section that it is also possible to obtain some benefit when serial queries are required. The basic idea is to use mass production to cheaply prepare several copies of the following resource state:

$$|\text{QROM}_f\rangle = O_f |+\rangle^{\otimes n} |0\rangle = 2^{-n/2} \sum_{y=0}^{N-1} |y\rangle |f(y)\rangle. \quad (\text{F1})$$

As we explain below, we can consume this resource state to implement the operation

$$\bar{O}_f^{(b)} : |x\rangle |0\rangle \rightarrow |x\rangle |0 \oplus f(x \oplus b)\rangle, \quad (\text{F2})$$

where  $b$  is a bitstring that is determined randomly when the resource state is consumed and the bar above  $O_f^{(b)}$  indicates that the output register must be in the zero state (we discuss the implications of this requirement after introducing the protocol). We can then recover the operation

$$\bar{O}_f : |x\rangle |0\rangle \rightarrow |x\rangle |f(x)\rangle \quad (\text{F3})$$



by performing an additional QROM query,

$$O_g : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus g(x)\rangle, \quad (\text{F4})$$

where we define

$$g(x) = f(x) \oplus f(x \oplus b). \quad (\text{F5})$$

Because  $g(x) = g(x \oplus b)$  for all  $x$ , we can specify  $g(x)$  using half as much information as  $f(x)$ . We show below how this allows us to implement  $O_g$  more cheaply than  $O_f$  or  $\bar{O}_f$ .

Provided that the cost of storing several copies of the resource state is negligible, these observations allow us to nearly halve the cost of repeated serial queries to  $\bar{O}_f$ . In order to implement  $c$  serial queries to  $\bar{O}_f$ , we first use mass production to prepare  $|\text{QROM}_f\rangle^{\otimes c}$  by replacing  $c$  parallel calls to  $O_f$  with a single call to a circuit  $C$  that mass-produces  $O_f^{\otimes c}$ . The cost to consume these resource states is dominated by implementing the  $O_g$ , which, as we show below, is approximately half the cost of implementing  $O_f$ . Assuming that we are in the regime where  $\text{COST}(C) \approx \text{COST}(O_f)$ , the overall cost is

$$\underbrace{\text{COST}(\bar{O}_f \cdots \bar{O}_f \cdots \bar{O}_f)}_{c \text{ serial queries}} \approx \text{COST}(O_f) + c \cdot \text{COST}(O_g), \quad (\text{F6})$$

which approaches  $\frac{c}{2} \cdot \text{COST}(O_f)$  as  $c$  increases.

The process for consuming the resource states is simple. Let  $|\psi\rangle = \sum_{x=0}^{N-1} c_x |x\rangle$  be an arbitrary state of an input register. To obtain  $\bar{O}_f |\psi\rangle |0\rangle$ , we begin with the input state  $|\psi\rangle$  and one copy of the resource state  $|\text{QROM}_f\rangle$  and perform the following steps:

$$|\psi\rangle |\text{QROM}_f\rangle \quad (\text{F7})$$

$$= |\psi\rangle \left( 2^{-n/2} \sum_{y \in \{0,1\}^n} |y\rangle |f(y)\rangle \right) \quad (\text{F8})$$

$$= 2^{-n/2} \sum_{x,y \in \{0,1\}^n} c_x |x\rangle |y\rangle |f(y)\rangle \quad (\text{F9})$$

$$\xrightarrow{\text{Apply CNOTs: } |x\rangle|y\rangle \rightarrow |x\rangle|x \oplus y\rangle} 2^{-n/2} \sum_{x,y \in \{0,1\}^n} c_x |x\rangle |x \oplus y\rangle |f(y)\rangle \quad (\text{F10})$$

$$\xrightarrow{\text{Measure 2nd register, outcome } b \in \{0,1\}^n} \sum_{x \in \{0,1\}^n} c_x |x\rangle |b\rangle |f(x \oplus b)\rangle \quad (\text{F11})$$

$$\xrightarrow{\text{Discard 2nd register (containing } |b\rangle)} \sum_{x \in \{0,1\}^n} c_x |x\rangle |f(x \oplus b)\rangle. \quad (\text{F12})$$

Note that each value of  $b$  is equally likely and that the post-measurement state in the above equations is normalized. After discarding the unentangled register, the final state is equivalent to  $\bar{O}_f^{(b)} |\psi\rangle |0\rangle$ , where  $\bar{O}_f^{(b)}$  is an operator that computes  $f(x \oplus b)$  into the output register. To obtain the desired  $f(x)$ , we apply a correction using a QROM read  $O_g$  for a function  $g$  that can be specified using only  $2^{n-1}$  values. The specific construction of  $O_g$  depends on the measurement outcome  $b$ , as detailed below. Let  $x'$  denote an  $(n-1)$ -bit string representing the last  $n-1$  bits of an  $n$ -bit string  $x$ , such that  $x = 0 \oplus x'$  or  $x = 1 \oplus x'$ . There are four cases to consider.

**Case 1:**  $b = 0^n$ : No correction is needed, as  $f(x \oplus 0^n) = f(x)$ . The process is complete.

**Case 2:**  $b = 1 \oplus 0^{n-1}$  (i.e.,  $b = 100 \dots 0$ ): The state after resource consumption is effectively

$$\sum_{x' \in \{0,1\}^{n-1}} (c_{0 \oplus x'} |0 \oplus x'\rangle |f(1 \oplus x')\rangle + c_{1 \oplus x'} |1 \oplus x'\rangle |f(0 \oplus x')\rangle).$$

We define a correction function  $g(z) = f(0 \oplus z) \oplus f(1 \oplus z)$  for  $z \in \{0, 1\}^{n-1}$ . Performing a QROM read  $O_g$  (where the input bits are the last  $n - 1$  bits of the full input register) yields:

$$\sum_{x' \in \{0,1\}^{n-1}} (c_{0 \oplus x'} |0 \oplus x'\rangle |f(1 \oplus x') \oplus g(x')\rangle + c_{1 \oplus x'} |1 \oplus x'\rangle |f(0 \oplus x') \oplus g(x')\rangle) \quad (\text{F13})$$

$$= \sum_{x' \in \{0,1\}^{n-1}} (c_{0 \oplus x'} |0 \oplus x'\rangle |f(0 \oplus x')\rangle + c_{1 \oplus x'} |1 \oplus x'\rangle |f(1 \oplus x')\rangle) \quad (\text{F14})$$

$$= \sum_{x \in \{0,1\}^n} c_x |x\rangle |f(x)\rangle. \quad (\text{F15})$$

**Case 3:**  $b = 1 \oplus b'$ , where  $b' \in \{0, 1\}^{n-1}$  and  $b' \neq 0^{n-1}$ : This generalizes Case 2. The state after resource consumption is:

$$|\phi_{\text{init}}\rangle = \sum_{x' \in \{0,1\}^{n-1}} (c_{0 \oplus x'} |0 \oplus x'\rangle |f(1 \oplus (x' \oplus b'))\rangle + c_{1 \oplus x'} |1 \oplus x'\rangle |f(0 \oplus (x' \oplus b'))\rangle). \quad (\text{F16})$$

The correction function is  $g(z) = f(0 \oplus z) \oplus f(1 \oplus (z \oplus b'))$  for  $z \in \{0, 1\}^{n-1}$ . The correction proceeds in steps:

$$\begin{aligned} |\phi_{\text{init}}\rangle &\xrightarrow{\text{Step 1}} \sum_{x' \in \{0,1\}^{n-1}} \left( c_{0 \oplus x'} |0 \oplus x'\rangle |f(1 \oplus (x' \oplus b'))\rangle \right. \\ &\quad \left. + c_{1 \oplus x'} |1 \oplus (x' \oplus b')\rangle |f(0 \oplus (x' \oplus b'))\rangle \right) \\ &\xrightarrow{\text{Step 2}} \sum_{x' \in \{0,1\}^{n-1}} \left( c_{0 \oplus x'} |0 \oplus x'\rangle |f(1 \oplus (x' \oplus b')) \oplus g(x')\rangle \right. \\ &\quad \left. + c_{1 \oplus x'} |1 \oplus (x' \oplus b')\rangle |f(0 \oplus (x' \oplus b')) \oplus g(x' \oplus b')\rangle \right) \\ &\xrightarrow{\text{Step 3}} \sum_{x' \in \{0,1\}^{n-1}} (c_{0 \oplus x'} |0 \oplus x'\rangle |f(0 \oplus x')\rangle + c_{1 \oplus x'} |1 \oplus x'\rangle |f(1 \oplus x')\rangle) \\ &= \sum_{x \in \{0,1\}^n} c_x |x\rangle |f(x)\rangle. \end{aligned}$$

Step 1 conditionally applies CNOTs to XOR  $b'$  into the last  $n - 1$  bits of the input register if the first bit is 1. Step 2 applies the QROM  $O_g$ , controlled by the last  $n - 1$  bits of the (potentially modified) input register. Step 3 substitutes the definition of  $g$  and uncomputes the conditional XOR from Step 1, restoring the input register to its original state.

**Case 4:**  $b = 0 \oplus b'$ , where  $b' \in \{0, 1\}^{n-1}$  and  $b' \neq 0^{n-1}$ : If  $b' = 0^{n-1}$ , this reduces to Case 1. Otherwise, we find any index  $i \in \{2, \dots, n\}$  such that the  $i$ -th bit of  $b$  is 1. We can then relabel the qubits of the input register so that the  $i$ th qubit becomes the first qubit. This permutation is also applied to the classical definition of  $f$  (to define  $f_{\text{perm}}$ ) and to  $b$  (to get  $b_{\text{perm}}$ ). This transformation ensures that the first bit of  $b_{\text{perm}}$  is 1. This relabeling is a classical pre-processing step that determines which  $2^{n-1}$ -entry QROM  $O_g$  to synthesize for  $f_{\text{perm}}$ ; it does not require any additional quantum operations during the correction phase itself. The problem then reduces to Case 3, using  $f_{\text{perm}}$  and  $b_{\text{perm}}$ .

The cost of consuming the resource state and applying the correction is dominated by the cost of implementing  $O_g$ , a QROM read with exactly half as many inputs as  $O_f$  (and therefore half the cost).

This strategy allows us to benefit when we are repeatedly implementing  $\bar{O}_f$ , which queries  $f$  in superposition, subject to the requirement that the output register is in the  $|0\rangle$  state. Operationally, we could define the action of this protocol on a more general state by resetting the output register to the zero state before proceeding. Letting

$$|\Psi\rangle = \sum_{i,j} c_{ij} |x_i\rangle |\alpha_j\rangle, \quad (\text{F17})$$

our protocol would implement the quantum channel

$$|\Psi\rangle\langle\Psi| \rightarrow O_f (\text{tr}_{\text{output}} [|\Psi\rangle\langle\Psi|] \otimes |0\rangle\langle 0|) O_f^\dagger. \quad (\text{F18})$$

This would not be sufficient for applications that require us to implement the map  $|x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle$  for arbitrary  $\alpha$ , nor would it be sufficient for applications that require us to act unitarily.<sup>7</sup>

<sup>7</sup> In some applications of QROM, the data loading oracle is only specified by its action on input states of the form  $|x\rangle |0\rangle$ . In many of these cases though, there is also a (sometimes implicit) require-

ment that the data loading oracle act unitarily on an arbitrary input.

However, it is frequently possible to use  $\tilde{O}_f$  to implement the more general

$$O_f : |x\rangle |\alpha\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle \quad (\text{F19})$$

with little to no additional overhead. This is because we can always perform the lookup to an ancilla output register initialized in the  $|0\rangle$  state before XORing the result into the real output register, implementing the operation

$$\tilde{O}_f : |x\rangle |\alpha\rangle |0\rangle \rightarrow |x\rangle |\alpha \oplus f(x)\rangle |f(x)\rangle. \quad (\text{F20})$$

As we briefly discuss in Appendix A 2, we can use the measurement-based uncomputation techniques of Ref. 12 to uncompute the ancilla register with a cost that scales as  $\mathcal{O}(2^n)$  rather than  $\mathcal{O}(2^n m)$ . When  $m$  is large, this additional overhead is negligible. In other cases, when we intend to temporarily use the value output by the QROM and then uncompute it anyway, it may be possible to fold the cost of uncomputing the junk register into the subsequent uncomputation step with no additional cost at all. See, e.g., the use of QROM to implement a block encoding using a linear combination of unitaries approach, as we review in Appendix E 1.

In the future, it would be interesting to investigate other applications for our resource state construction, and to understand whether or not it is possible to generalize it to reduce the cost of serial queries by more than a factor of  $\approx 2$ .