Asymptotically Optimal Quantum Circuits for Comparators and Incrementers

Vivien Vandaele

Quantinuum, Terrington House, 13–15 Hills Road, Cambridge CB2 1NL, United Kingdom

Extended abstract

We present quantum circuits for comparison and increment operations that achieve an asymptotically optimal gate count of $\Theta(n)$ and depth of $\Theta(\log n)$ over the Clifford+Toffoli gate set, while using a provably minimal number of qubits. We extend these results to classical–quantum comparators, yielding an improved classical–quantum adder with an optimal qubit count. Given the ubiquity of these operations as algorithmic building blocks, our constructions translate directly into reduced circuit complexity for many quantum algorithms. As a notable example, they can be used to improve a space-efficient circuit for Shor's factoring algorithm, reducing circuit depth from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 \log^2 n)$ without increasing either the qubit count or the asymptotic gate complexity. Underpinning these results is a general theorem demonstrating how to trade ancilla qubits for control qubits with low overhead in both depth and gate count, providing a broadly applicable tool for quantum circuit design.

1 Introduction

Quantum arithmetic operators are fundamental building blocks of many quantum algorithms. Comparators and incrementers, in particular, appear in a wide variety of key applications, including quantum walk algorithms [1], Grover-based minimum finding on unsorted lists [2], state-of-the-art approximate rotation synthesis algorithms [3], and modular multiplication constructions for Shor's factoring algorithm [4, 5]. Consequently, improving the efficiency of these elementary arithmetic operators is crucial for reducing the cost of a broad range of quantum algorithms and bringing practical quantum computing closer to realization.

Three metrics primarily govern the cost of a quantum circuit: the gate count, the circuit depth, and the number of qubits. Quantum circuit design often involves trade-offs among these metrics; for instance, one can typically reduce circuit depth at the expense of additional ancilla qubits. Despite this, we demonstrate that it is possible to achieve simultaneous optimality in all three metrics when implementing a comparator or an incrementer. Specifically, we present quantum circuits for both operators that match the known lower bounds on gate count and circuit depth while using a provably minimal number of qubits. Moreover, our constructions use only classical reversible logic gates (Toffoli, CNOT, and NOT) and thus avoid the need for small-angle rotation gates, which typically incur significant rotation synthesis overhead in fault-tolerant quantum computing.

A comparison with prior work is provided in Table 1 for the various arithmetic operators considered in this paper. Our main contributions are summarized below.

- **Promise gates.** We introduce the notion of a *promise gate*, a unitary whose action on a target register is guaranteed only when a designated promise register satisfies a given condition. We demonstrate that promise gates provide a useful complementary framework for reasoning about conditionally clean ancilla qubits, introduced in [6] and used in several breakthrough results [6, 7, 8, 9].
- **Add controls, save ancillae.** Building on the formalism of promise gates, we prove a general theorem showing how to trade ancilla qubits for control qubits when implementing a unitary

Type	Reference	Ancillae	Depth	Gate Count	Optimal?
Incrementer	Gidney [10]	1 dirty	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Nie et al. [6]	1 clean	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n)$	✗
	Khattar et al. [8]	$\log_2^* n$ clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Remaud et al. [9]	1 dirty	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n \log n)$	✗
	This paper	1 dirty	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	✓
Quantum–Quantum Comparator	Cuccaro et al. [11]	1 clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Takahashi et al. [12]	0	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Remaud et al. [9]	0	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n \log n)$	✗
	This paper	0	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	✓
Classical–Quantum Comparator	Gidney [13]	2 clean	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$	✗
	Khattar et al. [8]	$\log_2^* n$ clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	This paper	1 dirty	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	✓
Classical–Quantum Adder	Häner et al. [5]	1 dirty	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	✗
	Remaud et al. [9]	1 dirty	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(n \log^2 n)$	✗
	Gidney [14]	3 clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	This paper	1 dirty	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n \log n)$	✗

Table 1: Comparison of prior work with our constructions in terms of ancilla requirements, circuit depth, and gate count. Only results using a constant number of ancillae and classical reversible gates are reported. Results marked with ✓ achieve a provably minimum number of ancillae and asymptotically optimal depth and gate count.

gate, with low overhead in both gate count and circuit depth. This theorem applies beyond simply adding controls that can be traded for ancilla qubits. In particular, when an operator decomposes into a multi-controlled unitary and a simpler one (as is the case for comparators and incrementers), the theorem can then be applied directly to the multi-controlled unitary, capturing its full benefit without having to add controls to the entire operator.

- **Optimal incrementers.** We present a quantum circuit for the n-bit increment operator with $\Theta(n)$ gates and $\Theta(\log n)$ depth, using a single dirty ancilla. This matches the lower bounds in all three metrics simultaneously.

- **Optimal comparators.** We present quantum circuits for the n-bit quantum–quantum comparator and classical–quantum comparator (comparing a quantum register to a classical constant), with $\Theta(n)$ gates and $\Theta(\log n)$ depth, using no ancilla qubits and a single dirty ancilla, respectively. Both constructions match the respective lower bounds in all three metrics.

- **Improved classical–quantum adder.** As a direct consequence of our optimal constructions for the increment and comparison operators, we obtain an improved circuit for adding a classical constant to a quantum register, with $\Theta(n \log n)$ gate count and $\Theta(\log^2 n)$ depth, using one dirty ancilla. When plugged into the construction of [5], this yields an improved quantum circuit for qubit-efficient implementations of Shor's algorithm, reducing the circuit depth from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 \log^2 n)$ without increasing either the number of qubits or the asymptotic gate count, as shown in Table 2.

Reference	Year	Qubits	Depth	Gate Count
Beckman et al. [15]	1996	$5n+1$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Vedral et al. [16]	1996	$4n+3$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Zalka [17]	1998	$3n+\mathcal{O}(1)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Beauregard [18]	2003	$2n+3$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3 \log^2 n)$
Takahashi et al. [19]	2006	$2n+2$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3 \log^2 n)$
Zalka [20]	2006	$1.5n+\mathcal{O}(1)$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3 \log^2 n)$
Häner et al. [5]	2016	$2n+2$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3 \log n)$
Gidney [13]	2017	$2n+1$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3 \log n)$
Gidney et al. [21]	2019	$3n+0.002n \log n$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n^3 \log n)$
Chevignard et al. [22]	2024	$n/2+o(n)$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^2 \log^4 n)$
This paper	2026	$2n+2$	$\mathcal{O}(n^2 \log^2 n)$	$\mathcal{O}(n^3 \log n)$

Table 2: Comparison of space-efficient quantum circuits for Shor's factoring algorithm applied to an n-bit RSA integer, in terms of qubit count, circuit depth, and gate count. Only constructions using $\mathcal{O}(n)$ qubits are reported.

References

[1] B. L. Douglas and J. B. Wang. "Efficient quantum circuit implementation of quantum walks". Physical Review A**79** (2009).

[2] Christoph Durr and Peter Hoyer. "A Quantum Algorithm for Finding the Minimum" (1999). arXiv:quant-ph/9607014.

[3] Christoffer Hindlycke and Jan-Åke Larsson. "Single-qubit rotation algorithm with logarithmic Toffoli count and gate depth". Physical Review Research**6** (2024).

[4] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In Proceedings 35th Annual Symposium on Foundations of Computer Science. Page 124–134. SFCS-94. IEEE Comput. Soc. Press (1994).

[5] Thomas Haner, Martin Roetteler, and Krysta M. Svore. "Factoring using 2n+2 qubits with Toffoli based modular multiplication". Quantum Information and Computation **17**, 673–684 (2017).

[6] Junhong Nie, Wei Zi, and Xiaoming Sun. "Quantum circuit for multi-qubit Toffoli gate with optimal resource" (2024). arXiv:2402.05053.

[7] Baptiste Claudon, Julien Zylberman, César Feniou, Fabrice Debbasch, Alberto Peruzzo, and Jean-Philip Piquemal. "Polylogarithmic-depth controlled-NOT gates without ancilla qubits". Nature Communications**15** (2024).

[8] Tanuj Khattar and Craig Gidney. "Rise of conditionally clean ancillae for efficient quantum circuit constructions". Quantum **9**, 1752 (2025).

[9] Maxime Remaud and Vivien Vandaele. "Ancilla-Free Quantum Adder with Sublinear Depth". Page 137–154. Springer Nature Switzerland. (2025).

[10] Craig Gidney. "Constructing Large Increment Gates". url: `https://algassert.com/circuits/2015/06/12/Constructing-Large-Increment-Gates.html`.

[11] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. "A new quantum ripple-carry addition circuit" (2004). arXiv:quant-ph/0410184.

[12] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. "Quantum addition circuits and unbounded fan-out". Quantum Information and Computation **10**, 872–890 (2010).

[13] Craig Gidney. "Factoring with n+2 clean qubits and n-1 dirty qubits" (2018). arXiv:1706.07884.

[14] Craig Gidney. "A Classical-Quantum Adder with Constant Workspace and Linear Gates" (2025). arXiv:2507.23079.

[15] David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. "Efficient networks for quantum factoring". Physical Review A **54**, 1034–1063 (1996).

[16] Vlatko Vedral, Adriano Barenco, and Artur Ekert. “Quantum networks for elementary arithmetic operations”. Physical Review A **54**, 147–153 (1996).
[17] Christof Zalka. “Fast versions of Shor’s quantum factoring algorithm” (1998). arXiv:quant-ph/9806084.
[18] S. Beauregard. “Circuit for Shor’s algorithm using 2n+3 qubits”. Quantum Information and Computation **3**, 175–185 (2003).
[19] Y. Takahashi and N. Kunihiro. “A quantum circuit for Shor’s factoring algorithm using 2n+2 qubits”. Quantum Information and Computation **6**, 184–192 (2006).
[20] Christof Zalka. “Shor’s algorithm with fewer (pure) qubits” (2006). arXiv:quant-ph/0601097.
[21] Craig Gidney and Martin Ekerå. “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits”. Quantum **5**, 433 (2021).
[22] Clémence Chevignard, Pierre-Alain Fouque, and André Schrottenloher. “Reducing the Number of Qubits in Quantum Factoring”. Page 384–415. Springer Nature Switzerland. (2025).

Asymptotically Optimal Quantum Circuits for Comparators and Incrementers

Vivien Vandaele

Quantinuum, Terrington House, 13–15 Hills Road, Cambridge CB2 1NL, United Kingdom

We present quantum circuits for comparison and increment operations that achieve an asymptotically optimal gate count of $\Theta(n)$ and depth of $\Theta(\log n)$ over the Clifford+Toffoli gate set, while using a provably minimal number of qubits. We extend these results to classical–quantum comparators, yielding an improved classical–quantum adder with an optimal qubit count. Given the ubiquity of these operations as algorithmic building blocks, our constructions translate directly into reduced circuit complexity for many quantum algorithms. As a notable example, they can be used to improve a space-efficient circuit for Shor's factoring algorithm, reducing circuit depth from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 \log^2 n)$ without increasing either the qubit count or the asymptotic gate complexity. Underpinning these results is a general theorem demonstrating how to trade ancilla qubits for control qubits with low overhead in both depth and gate count, providing a broadly applicable tool for quantum circuit design.

1 Introduction

Quantum arithmetic operators are fundamental building blocks of many quantum algorithms. Comparators and incrementers, in particular, appear in a wide variety of key applications, including quantum walk algorithms [1], Grover-based minimum finding on unsorted lists [2], state-of-the-art approximate rotation synthesis algorithms [3], and modular multiplication constructions for Shor's factoring algorithm [4, 5]. Consequently, improving the efficiency of these elementary arithmetic operators is crucial for reducing the cost of a broad range of quantum algorithms and bringing practical quantum computing closer to realization.

Three metrics primarily govern the cost of a quantum circuit: the gate count, the circuit depth, and the number of qubits. Quantum circuit design often involves trade-offs among these metrics; for instance, one can typically reduce circuit depth at the expense of additional ancilla qubits. Despite this, we demonstrate that it is possible to achieve simultaneous optimality in all three metrics when implementing a comparator or an incrementer. Specifically, we present quantum circuits for both operators that match the known lower bounds on gate count and circuit depth while using a provably minimal number of qubits. Moreover, our constructions use only classical reversible logic gates (Toffoli, CNOT, and NOT) and thus avoid the need for small-angle rotation gates, which typically incur significant rotation synthesis overhead in fault-tolerant quantum computing.

A comparison with prior work is provided in Table 1 for the various arithmetic operators considered in this paper. Our main contributions are summarized below.

- **Promise gates.** We introduce the notion of a *promise gate*, a unitary whose action on a target register is guaranteed only when a designated promise register satisfies a given condition. We demonstrate that promise gates provide a useful complementary framework for reasoning about conditionally clean ancilla qubits, introduced in [6] and used in several breakthrough results [6, 7, 8, 9].
- **Add controls, save ancillae.** Building on the formalism of promise gates, we prove a general theorem showing how to trade ancilla qubits for control qubits when implementing a unitary gate, with low overhead in both gate count and circuit depth. This theorem applies beyond

Type	Reference	Ancillae	Depth	Gate Count	Optimal?
Incrementer	Gidney [10]	1 dirty	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Nie et al. [6]	1 clean	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n)$	✗
	Khattar et al. [8]	$\log_2^* n$ clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Remaud et al. [9]	1 dirty	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n \log n)$	✗
	This paper	1 dirty	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	✓
Quantum–Quantum Comparator	Cuccaro et al. [11]	1 clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Takahashi et al. [12]	0	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	Remaud et al. [9]	0	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n \log n)$	✗
	This paper	0	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	✓
Classical–Quantum Comparator	Gidney [13]	2 clean	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$	✗
	Khattar et al. [8]	$\log_2^* n$ clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	This paper	1 dirty	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	✓
Classical–Quantum Adder	Häner et al. [5]	1 dirty	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	✗
	Remaud et al. [9]	1 dirty	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(n \log^2 n)$	✗
	Gidney [14]	3 clean	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✗
	This paper	1 dirty	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n \log n)$	✗

Table 1: Comparison of prior work with our constructions in terms of ancilla requirements, circuit depth, and gate count. Only results using a sublinear number of ancilla qubits and classical reversible gates are reported. Results marked with ✓ achieve a provably minimal number of ancilla qubits and asymptotically optimal depth and gate count.

simply adding controls that can be traded for ancilla qubits. In particular, when an operator decomposes into a multi-controlled unitary and a simpler one (as is the case for comparators and incrementers), the theorem can then be applied directly to the multi-controlled unitary, capturing its full benefit without having to add controls to the entire operator.

- **Optimal incrementers.** We present a quantum circuit for the n-bit increment operator with $\Theta(n)$ gates and $\Theta(\log n)$ depth, using a single dirty ancilla. This matches the lower bounds in all three metrics simultaneously.
- **Optimal comparators.** We present quantum circuits for the n-bit quantum–quantum comparator and classical–quantum comparator (comparing a quantum register to a classical constant), with $\Theta(n)$ gates and $\Theta(\log n)$ depth, using no ancilla qubits and a single dirty ancilla, respectively. Both constructions match the respective lower bounds in all three metrics.
- **Improved classical–quantum adder.** As a direct consequence of our optimal constructions for the increment and comparison operators, we obtain an improved circuit for adding a classical constant to a quantum register, with $\Theta(n \log n)$ gate count and $\Theta(\log^2 n)$ depth, using one dirty ancilla. When plugged into the construction of Häner et al. [5], this yields an improved quantum circuit for qubit-efficient implementations of Shor's algorithm.

Outline. The paper is organized as follows. Section 2 introduces preliminary notions used throughout the paper. In Section 3, we introduce promise gates and show how they can be used to design efficient quantum circuits. We then apply these ideas in Section 4 to construct optimal quantum–quantum and classical–quantum comparator circuits, and in Section 5 to construct optimal incrementer circuits. Finally, in Section 6, we use these results to derive an improved circuit for classical–quantum addition and apply it to the modular multiplication subroutine of Shor's algorithm.

2 Preliminaries

2.1 Multi-controlled X gates and lower bounds

The gates used in our constructions belong to a subset of the family of k-controlled X gates, denoted C^kX and defined as follows.

Definition 2.1. For a non-negative integer k, the C^kX gate acts as

$$C^kX\,|\boldsymbol{x}, t\rangle = |\boldsymbol{x}, t \oplus \bigwedge_{i=0}^{k-1} x_i\rangle, \tag{1}$$

where $\boldsymbol{x} \in \{0,1\}^k$ and $t \in \{0,1\}$.

Specifically, the elementary building blocks of our quantum circuits are the C^kX gates with $k \leq 2$:

- C^0X corresponds to the X gate, also known as the NOT gate;
- C^1X corresponds to the CX gate, also known as the CNOT gate;
- C^2X corresponds to the CCX gate, also known as the Toffoli or CCNOT gate.

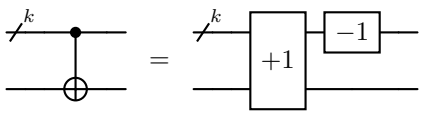
This gate set is a natural choice, as the CCX gate is well known to be the C^kX gate with the fewest controls that is universal for classical reversible computation [15]. While the constructions and optimality results in this paper target the $\{CCX, CX, X\}$ gate set, other gate sets not based on classical reversible logic are also of practical interest, as exemplified by QFT-based arithmetic [16]. A useful property of the $\{CCX, CX, X\}$ gate set is that any C^kX gate can be efficiently implemented over it using a single dirty ancilla qubit, as stated in the following lemma, proved in [6].

Lemma 1 (Nie et al. [6])**.** *The C^kX gate can be implemented over the $\{CCX, CX, X\}$ gate set with a gate count of $\Theta(k)$ and a circuit depth of $\Theta(\log k)$, using one dirty ancilla qubit.*

Throughout this paper, a *dirty ancilla qubit* refers to a qubit in an arbitrary unknown state that must be restored to that same state after use, whereas a *clean ancilla qubit* is one initialized in the $|0\rangle$ state and returned to $|0\rangle$ after use.

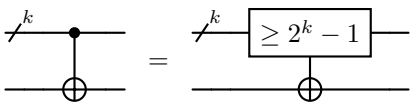
Lemma 1 is asymptotically optimal in both gate count and circuit depth and uses a minimal number of ancilla qubits. The gate count and circuit depth lower bounds of $\Omega(k)$ and $\Omega(\log k)$, respectively, hold for any implementation of the C^kX gate over any set of bounded-size gates, regardless of the number of ancilla qubits [17]. The ancilla lower bound follows from the fact that, for $k \geq 3$, the C^kX gate induces an odd permutation on $\{0,1\}^{k+1}$, whereas all generators in $\{CCX, CX, X\}$ induce even permutations [18].

The same lower bounds extend to k-bit incrementers and classical–quantum comparators. Indeed, a C^kX gate can be constructed from a constant number of incrementers as follows:



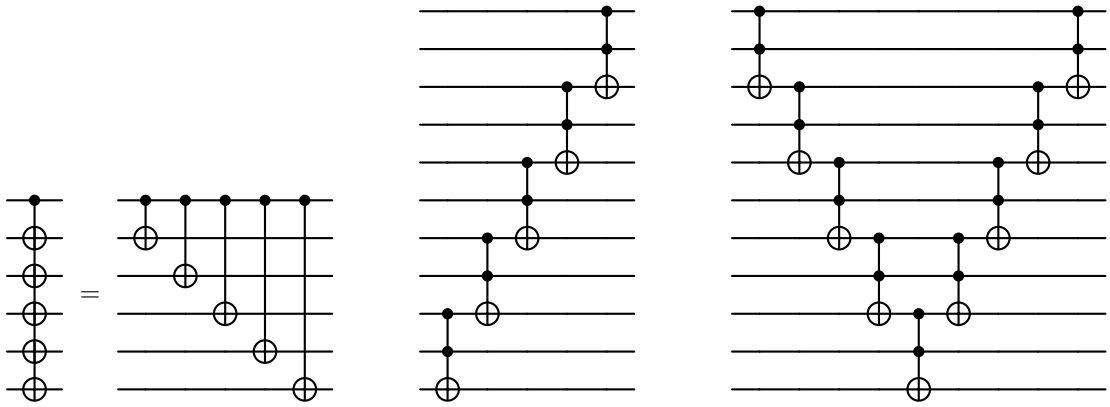
(2)

where the decrement gate can be obtained by conjugating the increment gate with X gates. Similarly, a C^kX gate can be constructed from a single classical–quantum comparator with the constant $2^k - 1$:



(3)

Therefore, if either of these operators could be implemented with no ancilla qubits, or with lower asymptotic gate count or depth complexities, then so could the C^kX gate, contradicting the established lower bounds. The quantum–quantum comparator, however, can be implemented



(a) Circuit implementing $\mathcal{F}_1^{(5)}$. (b) Circuit implementing $\mathcal{L}_2^{(5)}$. (c) Circuit implementing $\mathcal{V}_2^{(5)}$.

Figure 1: Naive implementations of the $\mathcal{F}_1^{(5)}$, $\mathcal{L}_2^{(5)}$, and $\mathcal{V}_2^{(5)}$ operators.

without ancilla qubits. Nevertheless, the gate count and circuit depth lower bounds of $\Omega(k)$ and $\Omega(\log k)$ still apply, since a quantum–quantum comparator can be used to construct a k-bit classical–quantum comparator using ancilla qubits, which in turn can be used to construct a C^kX gate, as shown by the circuit identity in Equation (3).

It is worth noting that the $\Omega(k)$ lower bound also applies specifically to the number of CCX gates, not merely to the total gate count. Indeed, Beverland et al. [19] showed that the CCX count required to implement the C^kX gate over the $\{CCX, CX, X\}$ gate set is $\Omega(k)$, regardless of the number of ancilla qubits. Together with the reductions in Equations (2) and (3), this implies an $\Omega(k)$ lower bound on the CCX count for any exact implementation of a k-bit incrementer or comparator. Moreover, this lower bound on the CCX count also holds for unitary circuits approximating the C^kX gate, or k-bit incrementer or comparator operators [20]. However, for ϵ-approximate implementation by mixed circuits, the C^kX gate requires only $\Theta(\log(1/\epsilon))$ CCX gates, whereas the lower bound for incrementers and comparators remains $\Omega(k)$ CCX gates [20].

In this work, we present quantum circuits for incrementers and comparators that match these lower bounds. Previously, these operators were considered more costly to implement than the C^kX gate, notably because a constant number of them suffice to construct the C^kX gate, as in Equations (2) and (3), whereas the converse does not hold. Despite this asymmetry, we show that these operators can in fact be implemented with the same asymptotically optimal gate count and circuit depth, using a minimal number of ancilla qubits.

2.2 Structural primitives

Many quantum arithmetic circuits rely on a few structural primitives that account for most of the gate count and depth complexity of the circuit. Here, we give the definitions and costs of the primitives used throughout the paper. Each of these operators admits a straightforward implementation with optimized gate count but suboptimal circuit depth. The lemmas below show that both gate count and circuit depth can often be simultaneously optimized, in some cases at the expense of ancilla qubits.

The first primitive is the k-th order fan-out operator: a collection of parallel C^kX gates all controlled by an additional single qubit.

Definition 2.2. The k-th order fan-out operator $\mathcal{F}_k^{(n)}$ acts on $n(k+1)+1$ qubits as

$$\mathcal{F}_k^{(n)}\left(|c\rangle \otimes \bigotimes_{i=1}^{n} |\boldsymbol{x}_i, t_i\rangle\right) = |c\rangle \otimes \bigotimes_{i=1}^{n} |\boldsymbol{x}_i, t_i \oplus c \bigwedge_{j=0}^{k-1} x_{i,j}\rangle, \tag{4}$$

where $c \in \{0,1\}$, $\boldsymbol{x}_i \in \{0,1\}^k$, and $t_i \in \{0,1\}$.

A quantum circuit representing the $\mathcal{F}_1^{(5)}$ operator is shown in Figure 1(a). We mainly use the first-order and second-order fan-out operators, $\mathcal{F}_1^{(n)}$ and $\mathcal{F}_2^{(n)}$. It is well known that these operators can be efficiently implemented, as established by the following lemma [17, 21, 9].

Lemma 2. *The $\mathcal{F}_1^{(n)}$ and $\mathcal{F}_2^{(n)}$ operators can both be implemented with $\mathcal{O}(n)$ $\{CCX, CX\}$ gates and $\mathcal{O}(\log n)$ circuit depth.*

Another recurring primitive, denoted $\mathcal{L}_k^{(n)}$, is the operator that can be implemented by a ladder of n consecutive C^kX gates.

Definition 2.3. The $\mathcal{L}_k^{(n)}$ operator acts on $kn+1$ qubits as

$$\mathcal{L}_k^{(n)}\left(\bigotimes_{i=0}^{kn}|x_i\rangle\right) = |x_0\rangle \otimes \bigotimes_{i=1}^{n}\left(\bigotimes_{j=1}^{k-1}|x_{k(i-1)+j}\rangle \otimes |x_{ki} \oplus \prod_{\ell=1}^{k} x_{ki-\ell}\rangle\right), \tag{5}$$

where $\boldsymbol{x} \in \{0,1\}^{kn+1}$.

A naive implementation of $\mathcal{L}_2^{(5)}$ is shown in Figure 1(b). The operator $\mathcal{L}_1^{(n)}$ can be implemented similarly but with a ladder of CX gates instead of CCX gates. It was proved in [9] that $\mathcal{L}_1^{(n)}$ can be implemented with the same gate count and depth as the $\mathcal{F}_1^{(n)}$ operator, up to a constant factor.

Lemma 3. *The $\mathcal{L}_1^{(n)}$ operator can be implemented with $\mathcal{O}(n)$ CX gates and $\mathcal{O}(\log n)$ circuit depth.*

By relying on a linear number of ancilla qubits, the $\mathcal{L}_2^{(n)}$ operator can be implemented with the same asymptotic gate count and circuit depth complexities as the $\mathcal{L}_1^{(n)}$ operator, as stated by the following lemma.

Lemma 4. *The $\mathcal{L}_2^{(n)}$ operator can be implemented with $\mathcal{O}(n)$ CCX gates and $\mathcal{O}(\log n)$ circuit depth, using n ancilla qubits.*

The proof of Lemma 4 is provided in Appendix A.1. The implementation of the $\mathcal{L}_2^{(n)}$ operator with $\mathcal{O}(n)$ gates, $\mathcal{O}(\log n)$ depth, and $\mathcal{O}(n)$ ancilla qubits was used in prior work on quantum addition circuits [22, 12]. The connection between the construction used in these works and the ancilla-free implementation of $\mathcal{L}_2^{(n)}$ was established in [23].

The following corollary follows from Lemmas 4 and 1:

Corollary 1. *The $\mathcal{L}_k^{(n)}$ operator can be implemented with $\mathcal{O}(kn)$ $\{CCX,\ CX,\ X\}$ gates and $\mathcal{O}(\log(kn))$ circuit depth, using n ancilla qubits.*

The proof of Corollary 1 is provided in Appendix A.2.

Finally, the last structural primitive used in this work, denoted $\mathcal{V}_k^{(n)}$, is defined in terms of $\mathcal{L}_k^{(n)}$ and its adjoint.

Definition 2.4. The $\mathcal{V}_k^{(n)}$ operator, acting on $kn+1$ qubits, is defined by

$$\mathcal{V}_k^{(n)} = \mathcal{L}_k^{(n)}\left(\mathcal{L}_k^{(n-1)^\dagger} \otimes I\right). \tag{6}$$

A naive construction of $\mathcal{V}_k^{(n)}$ is obtained by concatenating a ladder of C^kX gates implementing $\mathcal{L}_k^{(n-1)^\dagger}$ with a ladder of C^kX gates implementing $\mathcal{L}_k^{(n)}$, yielding a V-shaped circuit of C^kX gates. An example is shown in Figure 1(c) for $\mathcal{V}_2^{(5)}$. While the $\mathcal{V}_k^{(n)}$ operator and its implementation cost have not been explicitly studied in prior work, we show that it plays a central role in the implementation of the comparison operator.

3 Promise gates

We introduce the notion of a *promise gate*, a term inspired by the concept of a *promise problem* in computational complexity theory.[1] We first define promise gates in Subsection 3.1. Then, in Subsection 3.2, we present a theorem demonstrating how promise gates can be used to efficiently trade clean ancilla qubits for control qubits when implementing a unitary gate. We present a simple and concrete application of this theorem to controlled addition in Subsection 3.3.

3.1 Definition

A promise gate is a unitary acting on two designated registers: a *promise register* and a *target register*. It is only required to implement a target unitary U on the target register when the promise register satisfies a specified condition; its behavior when the condition is violated is left unspecified. We distinguish two variants: a *weak* promise gate may act arbitrarily on the promise register when the promise is violated, whereas a *strong* promise gate must preserve it for all inputs. In this work, we focus on the case in which the promise condition is that the promise register is in the $|0\rangle$ state, as formalized in the following definitions.

Definition 3.1. A *weak promise gate* with target unitary U is a unitary $\tilde{U}$ acting on a k-qubit promise register and an n-qubit target register such that

$$\tilde{U}(|0\rangle^{\otimes k} \otimes |\phi\rangle) = |0\rangle^{\otimes k} \otimes U|\phi\rangle$$

for every n-qubit state $|\phi\rangle$. The action of $\tilde{U}$ on inputs whose promise register is not in the state $|0\rangle^{\otimes k}$ is unconstrained.

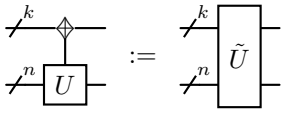
Definition 3.2. A *strong promise gate* with target unitary U is a unitary $\hat{U}$ acting on a k-qubit promise register and an n-qubit target register such that

$$\hat{U}(|\boldsymbol{j}\rangle \otimes |\phi\rangle) = |\boldsymbol{j}\rangle \otimes U_{\boldsymbol{j}}|\phi\rangle$$

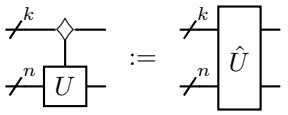
for every $\boldsymbol{j} \in \{0,1\}^k$ and every n-qubit state $|\phi\rangle$, where each $U_{\boldsymbol{j}}$ is a unitary on the target register with $U_{\mathbf{0}} = U$.[2] The unitaries $U_{\boldsymbol{j}}$ for $\boldsymbol{j} \neq \mathbf{0}$ are unconstrained.

Any strong promise gate with target unitary U is a weak promise gate with target unitary U: it satisfies the same guarantee on the promised subspace, but additionally preserves the promise register for all inputs. In contrast, a weak promise gate may act nontrivially on the promise register when the promise is violated.

We denote weak and strong promise gates by the following circuit notations, respectively:

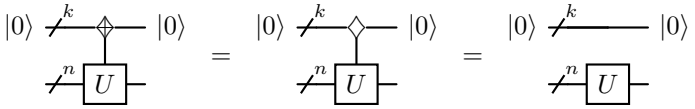


(weak promise gate) (7)



(strong promise gate) (8)

where ⟠ and ♢ mark the promise register. When the promise register is in the $|0\rangle^{\otimes k}$ state, the promise gate acts as U on the target register:

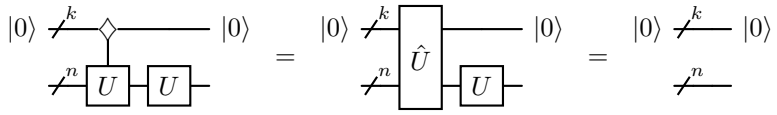


(9)

[1] In computational complexity theory, a *promise problem* is a generalization of a decision problem in which the input is promised to belong to a subset of all possible inputs. Any algorithm solving the promise problem is required to answer correctly only on promised inputs; the algorithm's behavior on inputs that violate the promise is left unspecified [24].

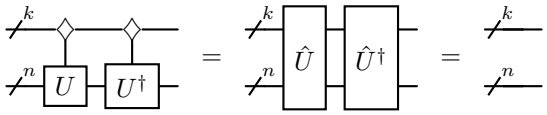
[2] A unitary $\hat{U}$ of this form is known as a *quantum multiplexor* (QMUX) [25]. A strong promise gate is thus a QMUX in which only $U_{\mathbf{0}}$ is specified.

The left-hand side of (7) and (8) uniquely identifies the promise and target registers together with the target unitary U, but it does not uniquely determine the promise gate. Indeed, distinct unitaries may satisfy Definition 3.1 or 3.2 while differing in their action outside the promised subspace. To ensure that circuit identities remain well defined, we only use equalities whose validity is independent of how the promise gate acts outside the promised subspace. For instance:



(10)

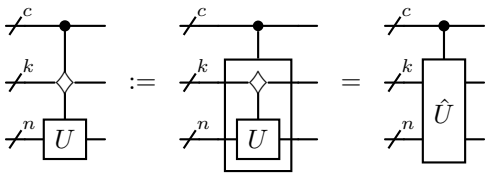
whenever $U^2 = I$, regardless of which promise gate $\hat{U}$ with target unitary U is chosen. Moreover, we adopt the convention that, within any given circuit, all promise gates sharing the same target unitary U and the same register sizes refer to the same underlying unitary $\tilde{U}$ or $\hat{U}$, and all promise gates sharing the same target unitary $U^\dagger$ and the same register sizes refer to the adjoint $\tilde{U}^\dagger$ or $\hat{U}^\dagger$, so that $\tilde{U}\tilde{U}^\dagger = I$ or $\hat{U}\hat{U}^\dagger = I$. This convention enables circuit simplifications such as:



(11)

This convention is particularly natural for the $\{CCX, CX, X\}$ gate set used in this paper, since each generator is self-adjoint. Consequently, constructing a promise gate $\tilde{U}$ or $\hat{U}$ for a target unitary U over this gate set automatically yields a construction for $\tilde{U}^\dagger$ or $\hat{U}^\dagger$, with the same gate count and circuit depth, simply by reversing the circuit.

Note that a controlled promise gate is not the same as a promise gate whose target unitary is a controlled gate. A *controlled promise gate* ($C\tilde{U}$ or $C\hat{U}$) is obtained by adding a control qubit to a promise gate. When the control qubit is in the state $|0\rangle$, the gate acts as the identity on all registers; when it is in the state $|1\rangle$, the promise gate is applied. For a strong promise gate, this preserves the promise register regardless of the state of the control register; for a weak promise gate, the promise register may be modified when the control register is in the $|1\rangle$ state and the promise is violated. On the other hand, a promise gate ($\widetilde{CU}$ or $\widehat{CU}$) whose target unitary is the controlled-U gate CU, applies a controlled-U operation to the target register when the promise is satisfied, but its action on the target register when the promise is violated is left unspecified, even when the control qubit is in the state $|0\rangle$. In this work, we use the following quantum circuit notation to represent controlled promise gates:

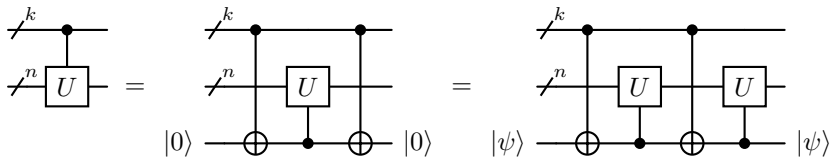


(12)

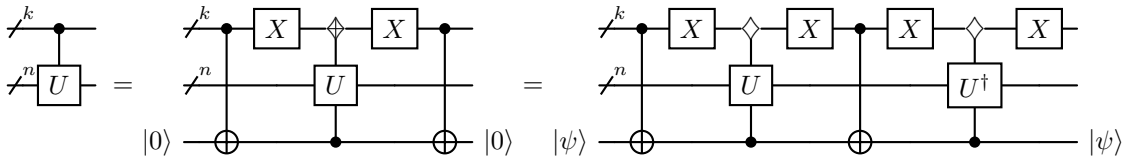
and similarly with $\Diamond$ and $\tilde{U}$ for the weak variant.

The notion of a promise gate is related to the concept of conditionally clean ancilla qubits, introduced in [6] and used in several breakthrough results [6, 7, 8, 9]. A conditionally clean ancilla is a qubit that is guaranteed to be in a known classical state whenever a certain predicate on other qubits holds. Conditionally clean ancilla qubits characterize the state of workspace qubits: they assert that a qubit is clean under a given condition. A promise gate captures a complementary behavioral abstraction: it prescribes what a gate is required to do on a subspace, leaving its action on the orthogonal complement unspecified (for weak promise gates) or constrained to preserve the promise register (for strong promise gates). Conditionally clean ancilla qubits can be viewed as the promise register of a promise gate: the predicate guaranteeing their cleanliness is precisely the condition under which the promise is satisfied.

This shift in perspective offers a key advantage: for weak promise gates, designing a quantum circuit $\mathcal{C}_U$ that implements a unitary U using m clean ancilla qubits is exactly the same problem as designing a quantum circuit $\mathcal{C}_{\tilde{U}}$ that implements a weak promise gate $\tilde{U}$ with target unitary U



(a) A C^kU gate can be decomposed into two C^kX gates and one controlled-U gate by using a clean ancilla. Whenever $U^2 = I$, the clean ancilla can be replaced by a dirty ancilla at the cost of an additional controlled-U gate.



(b) The controlled-U gate of (a) can be replaced by a controlled promise gate with target unitary U, where the top k-qubit register acts as the promise register. As in (a), replacing the clean ancilla with a dirty ancilla requires $U^2 = I$. In this case, strong promise gates must be used instead of weak promise gates to preserve the state of the k-qubit register. Note that the two strong promise gates associated with U and $U^\dagger$ on the right-hand side satisfy $\hat{U}\hat{U}^\dagger = I$, following the convention adopted in Section 3.1.

Figure 2: Decomposing a multi-controlled U gate to take advantage of promise gates.

and a promise register of size m. To see this, note that if $\mathcal{C}_{\tilde{U}}$ implements U whenever the promise register is in the state $|0\rangle$, then it also implements U when the promise register is replaced by clean ancilla qubits. Conversely, if $\mathcal{C}_U$ implements U with clean ancilla qubits, then $\mathcal{C}_U$ implements a weak promise gate $\tilde{U}$ when the clean ancilla register is reinterpreted as a promise register. This equivalence can simplify the design of some quantum circuits that exploit conditionally clean ancilla qubits, as constructing a circuit for a weak promise gate $\tilde{U}$ reduces to the familiar task of constructing a circuit for U with some clean ancilla qubits, a setting for which a large body of techniques already exists. This is precisely what makes promise gates powerful in practice: having access to clean ancilla qubits is generally very beneficial for reducing circuit depth or gate count, so the additional flexibility afforded by the promise register can yield implementations of $\tilde{U}$ that are significantly cheaper than an implementation of U without ancilla qubits.

For strong promise gates, the equivalence also holds, with the additional condition that the circuit $\mathcal{C}_U$ preserves the ancilla register for all inputs. This additional condition is naturally satisfied by many practical constructions. In particular, circuits over $\{CCX, CX, X\}$ that use clean ancilla qubits typically employ them in a compute-uncompute pattern. In such circuits, the restoration of the ancilla register does not depend on the ancilla qubits being initialized to $|0\rangle$: the uncomputation reverses the computation for every computational basis state of the ancilla register, so the register is preserved regardless of its initial state. For example, this is the case for the circuit construction for the $\mathcal{L}_2^{(n)}$ operator in Equation (55).

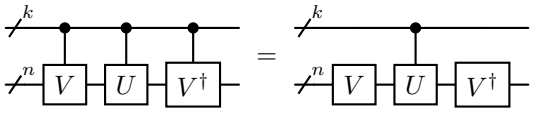
Moreover, promise gates compose naturally: if two promise gates $\tilde{U}$ and $\tilde{V}$ share the same promise register, their sequential composition $\tilde{V}\tilde{U}$ is itself a promise gate. More generally, the formalism makes it straightforward to reason modularly about large circuits that use promise gates, since each gate's correctness obligation is confined to the promised subspace.

Promise gates are one of the main ingredients for achieving the results presented in this paper. In the next subsection, we present a theorem that uses promise gates to trade clean ancilla qubits for controls when implementing a unitary. We then apply this theorem throughout the paper to construct asymptotically optimal quantum circuits for common arithmetic operators.

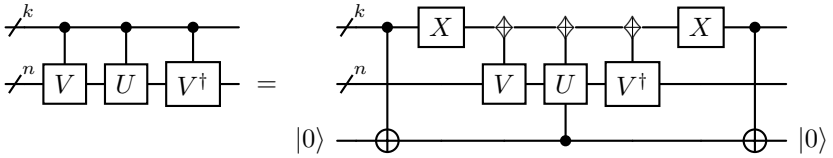
3.2 Add controls, save ancillae

To make effective use of promise gates, one must address the case in which the promise is not satisfied. Apart from trivial cases, such as when the promise register consists of clean ancilla qubits, we identify two main techniques for harnessing the power of promise gates. The circuit identities underlying these techniques are collected in Figures 2 and 3.

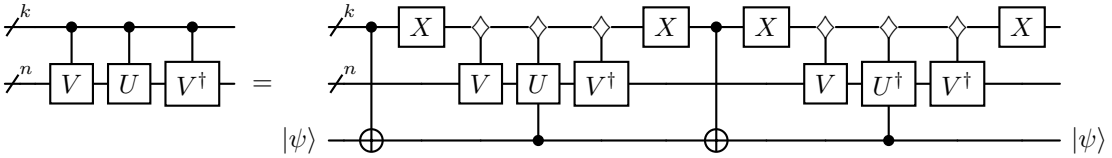
The first technique, well known from prior work on conditionally clean ancilla qubits, arises



(a) Controlled conjugation: when adding controls to $V^\dagger UV$, only the middle gate U needs to be controlled.



(b) Combining (a) with Figure 2(b): the unitaries V and $V^\dagger$ need not be controlled and are replaced by weak promise gates with target unitaries V and $V^\dagger$, respectively, using the top k-qubit register as the promise register.



(c) Dirty-ancilla variant of (b), obtained by combining (a) with the right-hand side of Figure 2(b). Requires $U^2 = I$.

Figure 3: Using controlled conjugation simplification with promise gates.

when the promise is satisfied if and only if the control qubit of a controlled promise gate is in the state $|1\rangle$. The first equality in Figure 2(a) shows that a C^kU gate can be implemented by introducing an ancilla qubit that detects whether the k-qubit control register is in the all-ones state. The controlled-U gate then acts nontrivially only when the top k-qubit register is in the $|1\rangle^{\otimes k}$ state, which the X gates map to $|0\rangle^{\otimes k}$. This means that the top register can serve as clean ancilla qubits for implementing U whenever the control qubit is in the $|1\rangle$ state, and can therefore be used as the promise register of a controlled weak promise gate with target unitary U, as shown in the first equality of Figure 2(b).

Moreover, whenever $U^2 = I$, the second equality in Figure 2(a) allows the clean ancilla to be replaced by a dirty ancilla. This leads to the second equality in Figure 2(b), where strong promise gates are required instead of weak promise gates, in order to preserve the state of the top k-qubit register.

Another effective use of promise gates arises when a promise gate is canceled by its adjoint whenever the promise is not satisfied. Consider the well-known circuit identity for controlled conjugations shown in Figure 3(a): when adding controls to $V^\dagger UV$, only U needs to be controlled, not V and $V^\dagger$. Combining this identity with the first equality in Figure 2(b) yields the circuit identity in Figure 3(b). Whenever the promise is not satisfied, the controlled promise gate with target unitary U acts as the identity, and the weak promise gates $\tilde{V}$ and $\tilde{V}^\dagger$ associated with V and $V^\dagger$ compose to the identity: $\tilde{V}^\dagger\tilde{V} = I$.

Figure 3(b) is therefore a valuable generalization of the controlled conjugation circuit identity of Figure 3(a): not only are the controls on V and $V^\dagger$ removed, but V and $V^\dagger$ are replaced by weak promise gates $\tilde{V}$ and $\tilde{V}^\dagger$ whose promise register can be used as if its qubits were clean ancilla qubits. This can lead to substantially cheaper implementations of $\tilde{V}$ and $\tilde{V}^\dagger$ compared with implementing V and $V^\dagger$ without ancilla qubits. The clean ancilla in Figure 3(b) can also be replaced by a dirty ancilla by using the second equality in Figure 2(b) instead of the first, yielding the circuit identity in Figure 3(c), which requires $U^2 = I$.

We now state the following theorem, which shows how to trade clean ancilla qubits for control qubits with little overhead in gate count and circuit depth.

Theorem 1. *Let $W = V^\dagger UV$ be an n-qubit unitary, where V and U are each implementable over a gate set $\mathcal{G}$ whose elements are all involutory (i.e., $G^2 = I$ for every $G \in \mathcal{G}$), using m clean ancilla qubits with gate counts c_V, c_U and circuit depths d_V, d_U, respectively. Then the k-controlled*

gate C^kW can be implemented over the gate set

$$\mathcal{G} \cup C(\mathcal{G}) \cup \{CCX, CX, X\},$$

where $C(\mathcal{G})$ denotes the set of singly controlled versions of gates in $\mathcal{G}$, with gate count $\mathcal{O}(c_V + c_U + d_U n + k)$, circuit depth $\mathcal{O}(d_V + d_U \log n + \log k)$, and using $\max(1, m - k + 1)$ clean ancilla qubits.

Moreover, if $U^2 = I$ and the implementations of V and U preserve the ancilla register for all inputs (so that they yield implementations of strong promise gates), then one clean ancilla can be replaced by one dirty ancilla with the same asymptotic gate count and circuit depth complexities.

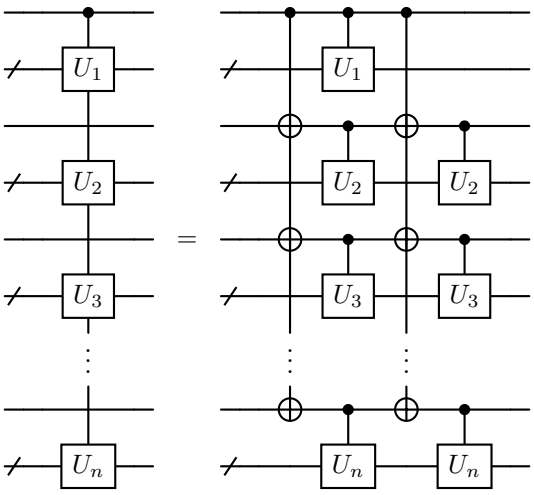
We rely on the following lemma in the proof of Theorem 1.

Lemma 5. *Let $\mathcal{G} = \{U_1, \ldots, U_n\}$ be a set of $n > 1$ unitary gates, each satisfying $U_i^2 = I$, and let $U = \bigotimes_{i=1}^n U_i$. Then the k-controlled gate C^kU can be implemented over the gate set*

$$C(\mathcal{G}) \cup \{CCX, CX, X\},$$

where $C(\mathcal{G})$ denotes the set of singly controlled versions of gates in $\mathcal{G}$, with gate count $\mathcal{O}(n + k)$ and circuit depth $\mathcal{O}(\log n + \log k)$, without using any ancilla qubits.

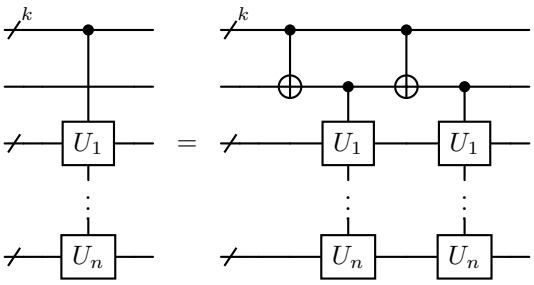
Proof. When $k = 1$, we repeatedly apply the circuit identity of Figure 2(a), each application using one dirty ancilla, to obtain the following circuit identity:



(13)

which uses $n - 1$ dirty ancilla qubits. To ensure that enough dirty ancilla qubits are available, we split the set $\mathcal{G}$ into two subsets of size $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$, respectively. Without loss of generality, we may assume that each unitary U_i acts on at least one qubit, so that the unitaries in each subset act on at least $\lfloor n/2 \rfloor$ qubits. Applying Equation (13) to each subset requires at most $\lceil n/2 \rceil - 1 \leq \lfloor n/2 \rfloor$ dirty ancilla qubits, so the qubits acted on by one subset can serve as dirty ancilla qubits for the other. The fan-out gate is then applied four times over at most $\lceil n/2 \rceil$ qubits each time, and each controlled U_i gate is applied at most twice. As stated in Lemma 2, the fan-out gate can be implemented with $\mathcal{O}(n)$ CX gates and circuit depth $\mathcal{O}(\log n)$. Thus, the total gate count is $\mathcal{O}(n)$ and the total circuit depth is $\mathcal{O}(\log n)$.

When $k > 1$, we use the following identity, also based on the circuit identity of Figure 2(a):



(14)

This reduces the k-controlled case to two C^kX gates and two instances of the singly controlled case. We apply this identity twice, once for each subset as described above, using the qubits of the other subset as dirty ancilla qubits. The two singly controlled instances in each application are then implemented using Equation (13), where one dirty ancilla can be taken from the k control qubits to compensate for the additional dirty ancilla required by Equation (14). The two C^kX gates can be implemented with $\mathcal{O}(k)$ gates and $\mathcal{O}(\log k)$ circuit depth using one dirty ancilla (Lemma 1). Thus, the total gate count is $\mathcal{O}(n+k)$ and the total circuit depth is $\mathcal{O}(\log n + \log k)$. □

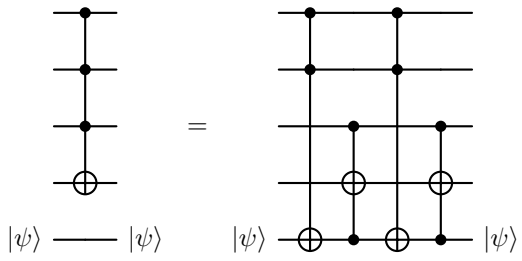
Using Lemma 5 together with the circuit identities presented in Figure 3, we can now prove Theorem 1.

Proof of Theorem 1. We begin by applying the circuit identity of Figure 3(b), which introduces one clean ancilla qubit and two C^kX gates, and replaces the k-controlled V, $V^\dagger$, and U gates with weak promise gates $\tilde{V}$ and $\tilde{V}^\dagger$ and a controlled weak promise gate $\tilde{U}$, all sharing a promise register of size k. Each C^kX gate can be implemented with $\mathcal{O}(k)$ gates and $\mathcal{O}(\log k)$ depth using one dirty ancilla from the n-qubit register (Lemma 1). The weak promise gates $\tilde{V}$ and $\tilde{V}^\dagger$ are not controlled, so we can use the implementations of V and $V^\dagger$ with c_V gates, circuit depth d_V, and m clean ancilla qubits, replacing k of the clean ancilla qubits with qubits from the promise register and thereby requiring only $\max(0, m-k)$ clean ancilla qubits.

It remains to implement the controlled weak promise gate $\tilde{U}$. As for $\tilde{V}$, we can implement the weak promise gate $\tilde{U}$ by using the construction for U with c_U gates, circuit depth d_U, and $\max(0, m-k)$ clean ancilla qubits, substituting k clean ancilla qubits with the promise register. We then add a control to each gate in this implementation and apply the construction of Lemma 5 to each of the d_U layers, for a total cost of $\mathcal{O}(d_U n)$ $C(\mathcal{G}) \cup \{CX\}$ gates and $\mathcal{O}(d_U \log n)$ circuit depth. The total gate count is therefore $\mathcal{O}(c_V + c_U + d_U n + k)$, and the total circuit depth is $\mathcal{O}(d_V + d_U \log n + \log k)$, using $\max(1, m-k+1)$ clean ancilla qubits.

Whenever U satisfies $U^2 = I$ and the implementations of V and U preserve the ancilla register for all inputs, we can use the circuit identity of Figure 3(c) instead of the one of Figure 3(b), which replaces the clean ancilla qubit with a dirty ancilla qubit at the cost of at most doubling the gate count and circuit depth. □

Note that when $\mathcal{G} = \{CCX, CX, X\}$, the circuit produced by Theorem 1 is over the $\{CCCX,$ $CCX,$ $CX,$ $X\}$ gate set. To return to the $\{CCX, CX, X\}$ gate set, we can decompose each C^3X gate into 4 CCX gates using one dirty ancilla via the following identity:



(15)

To ensure that each C^3X decomposition has access to a dirty ancilla, we use the same splitting technique as in the proof of Lemma 5: within each circuit layer, we partition the C^3X gates into two groups and use the qubits acted on by one group as dirty ancilla qubits for the other. This yields a circuit over the $\{CCX, CX, X\}$ gate set with the same asymptotic gate count and circuit depth as in Theorem 1. This decomposition introduces only a constant factor overhead in gate count and circuit depth, yielding the following corollary.

Corollary 2. *Under the hypotheses of Theorem 1 with $\mathcal{G} = \{CCX, CX, X\}$, the k-controlled gate C^kW can be implemented over the $\{CCX, CX, X\}$ gate set with gate count $\mathcal{O}(c_V + c_U + d_U n + k)$, circuit depth $\mathcal{O}(d_V + d_U \log n + \log k)$, and using $\max(1, m-k+1)$ clean ancilla qubits. Moreover, if $U^2 = I$ and the implementations of V and U preserve the ancilla register for all inputs, then one clean ancilla can be replaced by one dirty ancilla with the same asymptotic gate count and circuit depth complexities.*

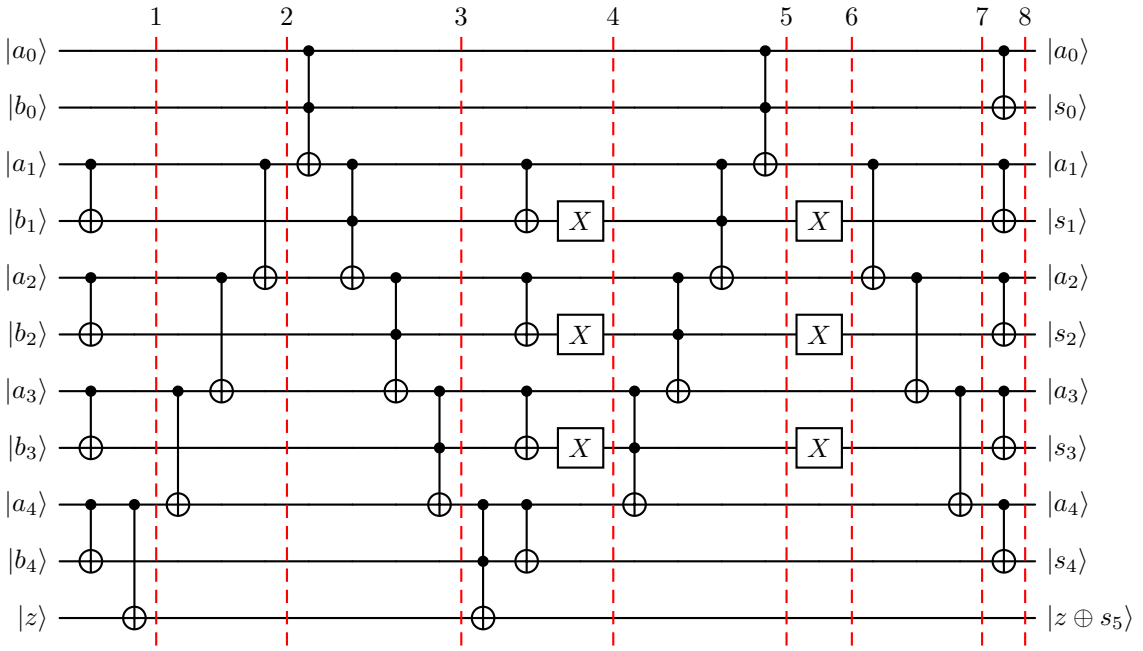


Figure 4: Ancilla-free circuit for quantum addition over 5-bit registers, where $s = a + b$ [12].

3.3 Application to controlled addition

We now demonstrate a straightforward application of Theorem 1: efficiently implementing a controlled adder by trading clean ancilla qubits for control qubits.

Corollary 3. *The k-controlled n-bit quantum adder can be implemented over the $\{CCX, CX, X\}$ gate set with $\mathcal{O}(n+k)$ gates, $\mathcal{O}(\log n + \log k)$ circuit depth, and $\max(1, n-k+1)$ clean ancilla qubits.*

Proof. We start from the ripple-carry adder of [12], an example of which is shown in Figure 4. Let U_i denote the unitary implemented by slice i in Figure 4. The ladders of CX gates in slices 2 and 7, corresponding to U_2 and $U_7 = U_2^\dagger$, can each be implemented with $\mathcal{O}(n)$ CX gates and circuit depth $\mathcal{O}(\log n)$, as stated in Lemma 3. The ladders of CCX gates in slices 3 and 5, corresponding to U_3 and $U_5 = U_3^\dagger$, can each be implemented with $\mathcal{O}(n)$ CCX gates and circuit depth $\mathcal{O}(\log n)$, using n clean ancilla qubits, as stated in Lemma 4.

Applying the controlled conjugation identity of Figure 3(a) twice, first to the outer $U_2/U_2^\dagger$ pair and then to the inner $U_3/U_3^\dagger$ pair, the k-controlled adder can be written as

$$C^k A = C^k(U_8) \cdot U_2^\dagger \cdot C^k(U_6) \cdot U_3^\dagger \cdot C^k(U_4) \cdot U_3 \cdot U_2 \cdot C^k(U_1). \tag{16}$$

We apply Theorem 1 to implement $U_3^\dagger \cdot C^k(U_4) \cdot U_3$ in Equation (16) with $\mathcal{O}(n+k)$ gates and $\mathcal{O}(\log n + \log k)$ circuit depth, using $\max(1, n-k+1)$ clean ancilla qubits. The single C^3X gate we obtain in slice 4 can be decomposed into four CCX gates using one available dirty ancilla, as in Equation (15). The remaining controlled unitaries $C^k(U_8)$, $C^k(U_6)$, and $C^k(U_1)$ can each be implemented over the $\{CCX, CX, X\}$ gate set with $\mathcal{O}(n+k)$ gates and $\mathcal{O}(\log n + \log k)$ circuit depth by Lemma 5. Thus, the resulting k-controlled adder has $\mathcal{O}(n+k)$ $\{CCX, CX, X\}$ gates, $\mathcal{O}(\log n + \log k)$ circuit depth, and uses $\max(1, n-k+1)$ clean ancilla qubits. □

Theorem 1 is not limited to simply adding controls to a unitary so that they can be traded for ancilla qubits. As we will see in the following sections, it is also applicable when an operator decomposes into a multi-controlled unitary and a smaller one, allowing its full benefit to be exploited without adding controls to the entire operator.

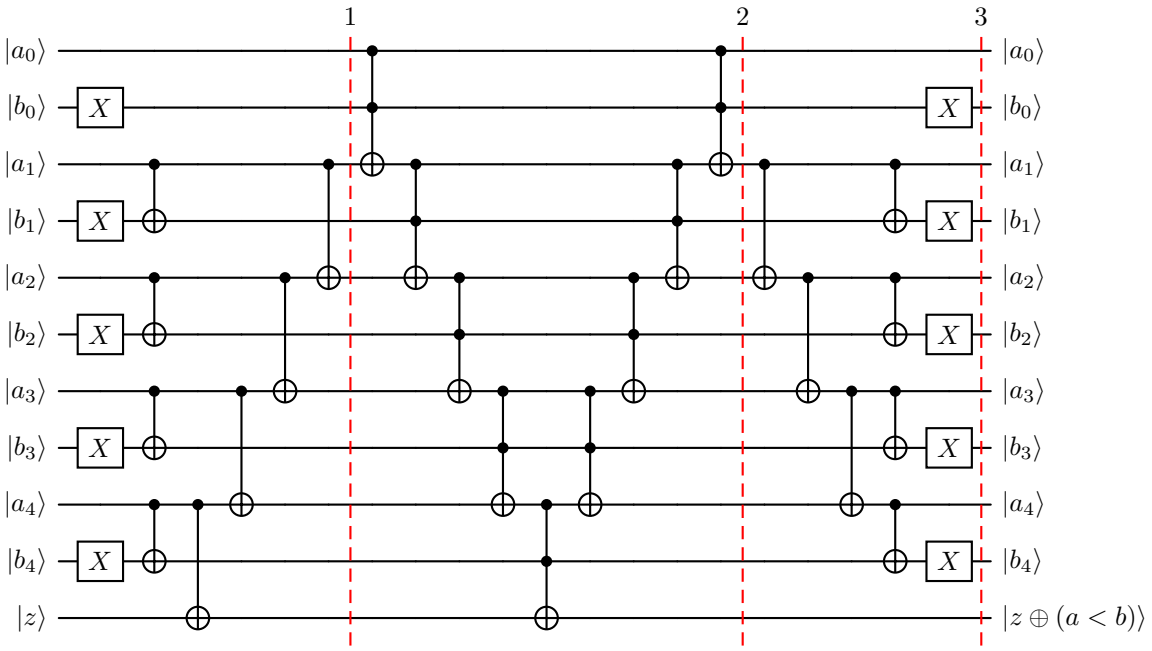


Figure 5: Ancilla-free circuit for the quantum–quantum comparator over 5-bit registers.

4 Quantum circuits for comparators

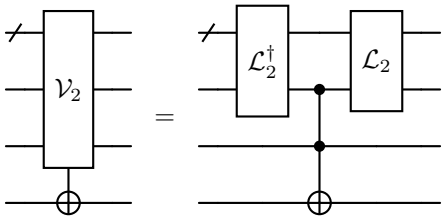
4.1 Quantum–quantum comparator

A quantum–quantum comparator over n-bit registers computes the following map:

$$|\boldsymbol{a}\rangle\,|\boldsymbol{b}\rangle\,|z\rangle \mapsto |\boldsymbol{a}\rangle\,|\boldsymbol{b}\rangle\,|z \oplus (a < b)\rangle\,, \tag{17}$$

where $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^n$ and $z \in \{0,1\}$. Note that such a comparator can also be used to compute $a \leq b$, since $a \leq b$ is equivalent to $\neg(b < a)$. The comparison operator can be implemented by computing the high bit of $a - b$, which is 1 if and only if $a < b$ [11]. The subtraction $a - b$ can be constructed from an adder by conjugating it with X gates on the register holding b. An example of a comparator obtained from this construction is shown in Figure 5, using the adder from Figure 4. Because this approach relies on a quantum adder, comparators have traditionally inherited the same circuit complexity as addition. In this section, we show that this need not be the case: comparison can be implemented with lower circuit complexity than the best-known adders.

A key component of the comparator is the $\mathcal{V}_2^{(n)}$ operator, as can be seen in slice 2 of Figure 5. This operator admits the following implementation, using the $\mathcal{L}_2^{(n-1)}$ operator and its adjoint together with a CCX gate in the middle:



(18)

This decomposition motivates the following corollary.

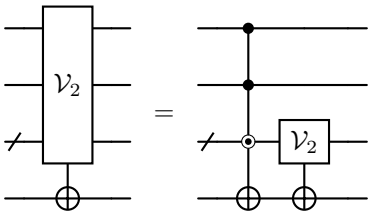
Corollary 4. *A strong promise gate with a promise register of size n whose target unitary is the $\mathcal{L}_k^{(n)}$ operator can be implemented with $\mathcal{O}(nk)$ $\{CCX, CX, X\}$ gates and $\mathcal{O}(\log(nk))$ circuit depth.*

This corollary follows straightforwardly from the construction used to prove Lemma 4; the proof is provided in Appendix A.3.

The following lemma provides a key construction used in the main result of this section.

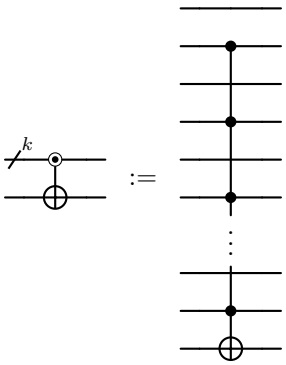
Lemma 6. *A controlled strong promise gate with a promise register of size* $2\lfloor\sqrt{n}\rfloor$ *whose target unitary is the* $\mathcal{V}_2^{(n)}$ *operator can be implemented with* $\mathcal{O}(n)$ $\{CCX, CX, X\}$ *gates and* $\mathcal{O}(\log n)$ *circuit depth.*

Proof. The $\mathcal{V}_2^{(n)}$ operator admits the following decomposition:



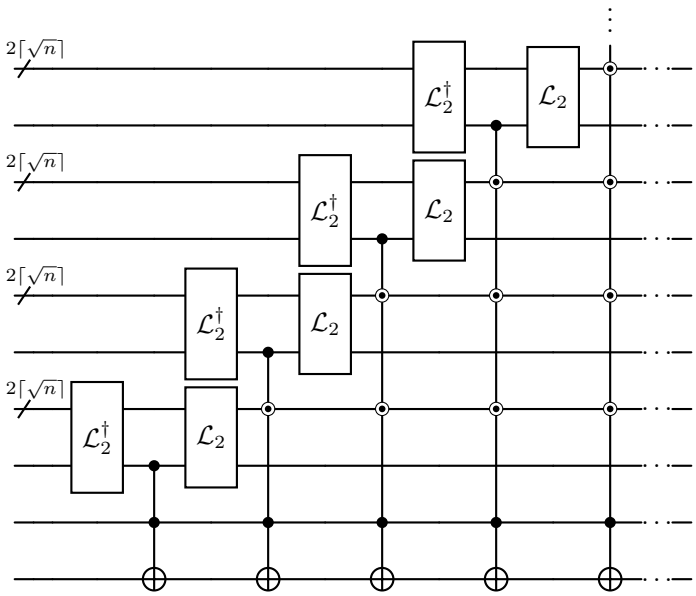
(19)

where the $\odot$ symbol indicates controls only on alternating qubits:



(20)

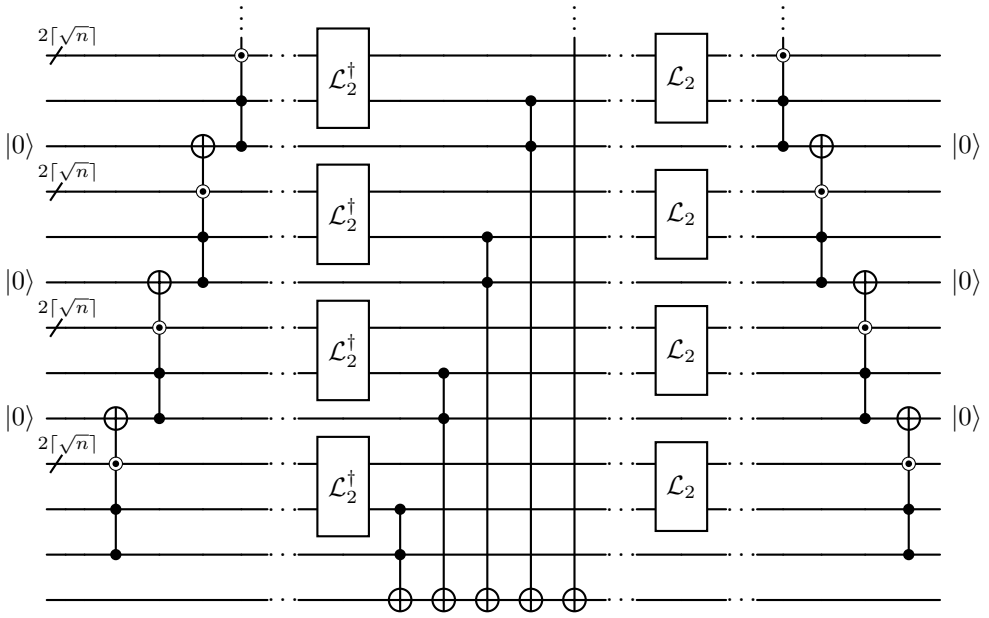
By Equations (18) and (19), we obtain the following implementation of the $\mathcal{V}_2^{(n)}$ operator:



(21)

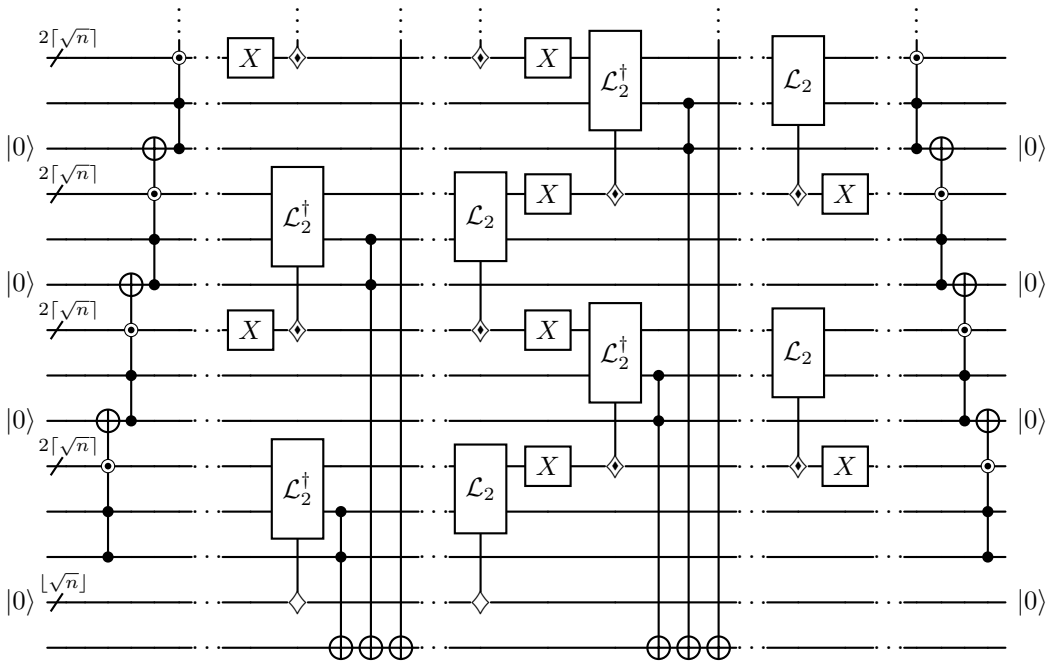
where the topmost register has size at most $2\lceil\sqrt{n}\rceil$.

We then introduce $\lfloor \sqrt{n} \rfloor$ clean ancilla qubits to parallelize the $\mathcal{L}_2$ gates as follows:



(22)

We can then apply the circuit identity of Figure 3(b), where V and $V^\dagger$ correspond to the $\mathcal{L}_2^\dagger$ and $\mathcal{L}_2$ operators and U corresponds to the CCX gate in between. Each CCX gate in the middle of the circuit acts nontrivially only when the ancilla qubit on which it is controlled is in the $|1\rangle$ state. In turn, this ancilla is in the $|1\rangle$ state only when the register below it is in the $|1\rangle$ state. For each $\mathcal{L}_2$ gate, the register on which the associated CCX gate is conditioned can therefore serve as a promise register. Since each promise gate uses the register below it as its promise register, we cannot apply all promise gates simultaneously; instead, we proceed in two rounds. We obtain the following circuit:

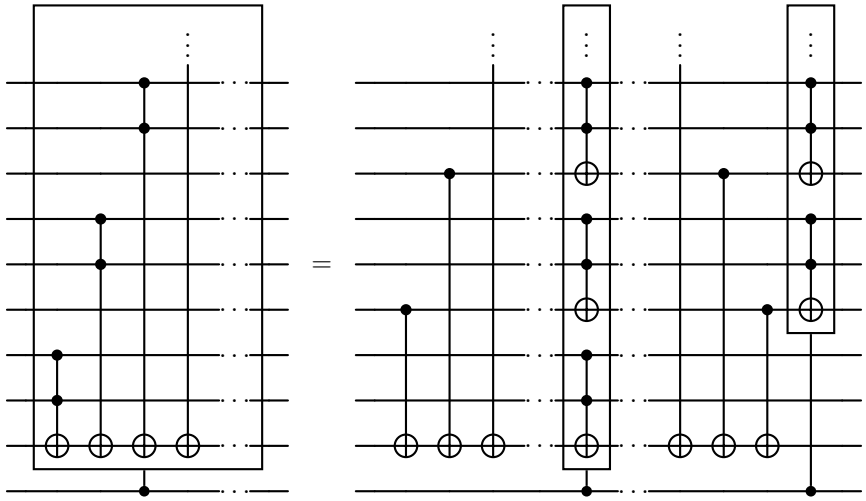


(23)

where $\lfloor \sqrt{n} \rfloor$ clean ancilla qubits were inserted at the bottom to serve as the promise register for the bottommost promise gates, and where $\diamondsuit$ denotes $\diamondsuit$ symbols on alternating qubits, analogously to $\odot$ in Equation (20).

We then replace the clean ancilla qubits with a promise register of size $2\lfloor\sqrt{n}\rfloor$ and add a control to each gate to obtain a controlled strong promise gate whose target unitary is the $\mathcal{V}_2$ operator. By Figure 3(a), only the CCX gates targeting the bottom qubit need to be controlled. We obtain the circuit presented in Figure 6.

We now analyze the cost of the resulting circuit. Using $\lfloor\sqrt{n}\rfloor$ qubits from the promise register, the first and last ladders of multi-controlled X gates can be implemented with $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth, as stated in Corollary 1. The CCX gates targeting the same qubit and controlled by the bottom qubits can be implemented as follows:



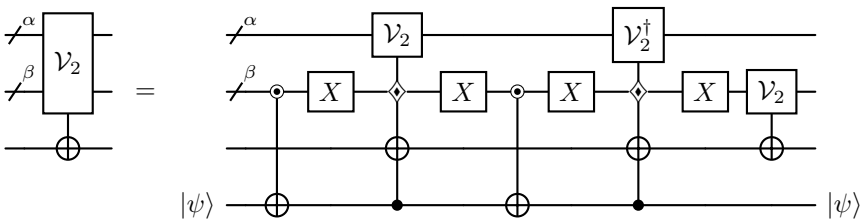
(24)

The two sets of CX gates in Equation (24) sharing the same target correspond to fan-in gates, and can be implemented with $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth (Lemma 2). Similarly, the two sets of CCX gates in Equation (24) all controlled by the same control qubit correspond to the second-order fan-out operator and can also be implemented with $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth (Lemma 2). Finally, by Corollary 4, each promise gate with target unitary $\mathcal{L}_2$ or $\mathcal{L}_2^\dagger$ can be implemented with $\mathcal{O}(\sqrt{n})$ gates and $\mathcal{O}(\log n)$ circuit depth. Thus, the total gate count and circuit depth of the circuit of Figure 6 are $\mathcal{O}(n)$ and $\mathcal{O}(\log n)$, respectively. □

Theorem 2. *The n-bit quantum–quantum comparator can be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n)$ gates and $\Theta(\log n)$ circuit depth, without any ancilla qubits.*

Proof. The two layers of parallel X and CX gates in slices 1 and 3 of Figure 5 contribute $\mathcal{O}(n)$ gates with a constant depth. The two ladders of CX gates in these slices can each be implemented with $\mathcal{O}(n)$ CX gates and $\mathcal{O}(\log n)$ circuit depth (Lemma 3). It remains to evaluate the cost of implementing the $\mathcal{V}_2^{(n)}$ operator in slice 2 of Figure 5.

Based on the circuit identity of Figure 3(c) and Equation (19), we have:



(25)

where the dirty ancilla can be taken from the register of size β, and where the ⊙ and ⧫ symbols represent • and ♢ on alternating qubits, as in Equation (20).

Let $\alpha = n - 2\lfloor\sqrt{n}\rfloor$ and $\beta = 2\lfloor\sqrt{n}\rfloor$. The two $C^{\lfloor\sqrt{n}\rfloor}X$ gates can be implemented with a total cost of $\mathcal{O}(\sqrt{n})$ gates and $\mathcal{O}(\log n)$ circuit depth (Lemma 1). By Lemma 6, the two controlled strong promise gates whose target unitary is the $\mathcal{V}_2$ operator can each be implemented with $\mathcal{O}(n)$

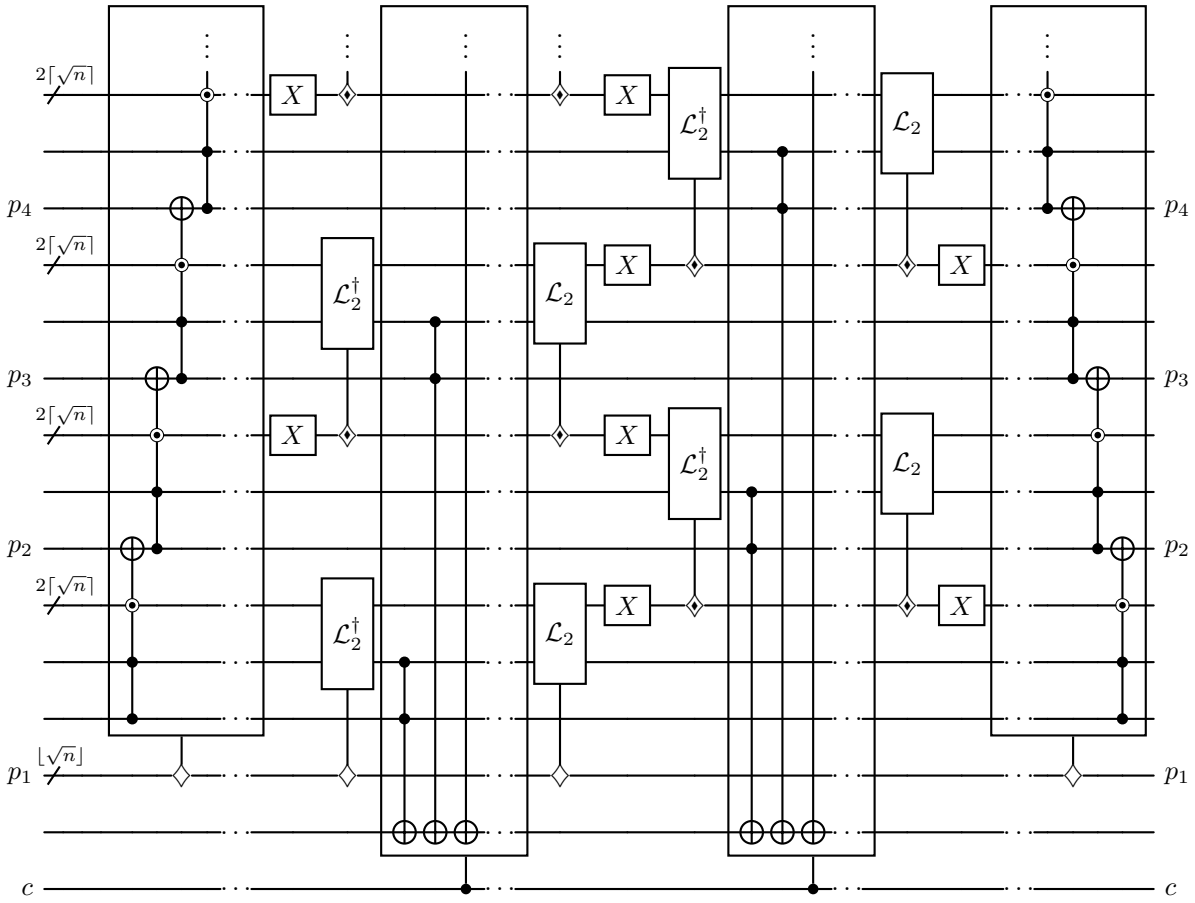


Figure 6: Quantum circuit for a controlled strong promise gate whose target unitary is the $\mathcal{V}_2^{(n)}$ operator. c denotes the control qubit and $\boldsymbol{p}$ denotes the promise register of size $2\lfloor\sqrt{n}\rfloor$. The $\odot$ and $\diamondsuit$ symbols represent $\bullet$ and $\diamondsuit$ on alternating qubits, as in Equation (20).

$\{CCX, CX, X\}$ gates and $\mathcal{O}(\log n)$ circuit depth. We then implement the $\mathcal{V}_2$ operator on the bottom β qubits recursively. The total gate count satisfies

$$C(n) = \Theta(n) + C(2\lfloor\sqrt{n}\rfloor) \tag{26}$$

and the total circuit depth satisfies

$$D(n) = \Theta(\log n) + D(2\lfloor\sqrt{n}\rfloor) \tag{27}$$

which yield $C(n) = \Theta(n)$ and $D(n) = \Theta(\log n)$, respectively. □

As a direct consequence of Theorem 2, we obtain an efficient construction for the k-controlled comparator.

Corollary 5. *The k-controlled n-bit (where $n > 1$) quantum–quantum comparator can be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n + k)$ gates and $\Theta(\log(kn))$ circuit depth, without any ancilla qubits.*

Proof. By the controlled conjugation identity of Figure 3(a), when adding k control qubits to the comparator circuit of Figure 5, only a single CX gate in slice 1 and the $\mathcal{V}_2^{(n)}$ operator in slice 2 need to be controlled.

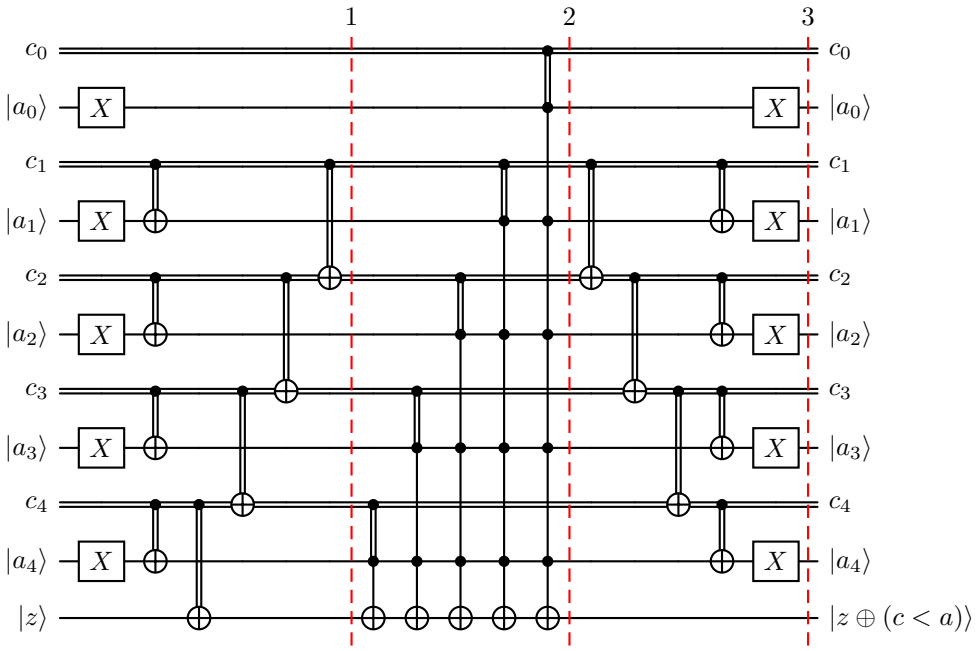
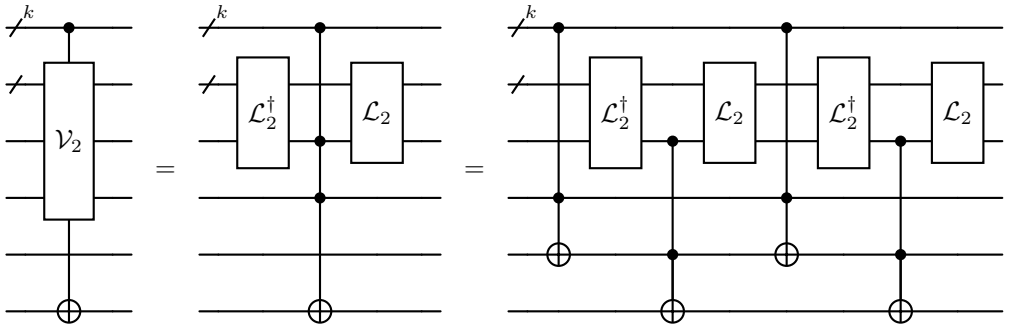


Figure 7: Circuit for the classical–quantum comparator over 5-bit registers.

The k-controlled $\mathcal{V}_2$ operator can be implemented as follows:



(28)

using one dirty ancilla qubit, which is available since $n > 1$. The $C^{k+1}X$ gates can be implemented with $\Theta(k)$ gates and $\Theta(\log k)$ circuit depth (Lemma 1). Each pair of $\mathcal{L}_2$ and $\mathcal{L}_2^\dagger$ operators with a CCX gate in the middle corresponds to a $\mathcal{V}_2$ operator, which can be implemented with $\Theta(n)$ gates and $\Theta(\log n)$ circuit depth by Theorem 2. □

4.2 Classical–quantum comparator

We now turn to the classical–quantum comparator and prove the analogous optimal result. An n-bit classical–quantum comparator, parameterized by a classical constant $c \in \{0,1\}^n$, computes the following map:

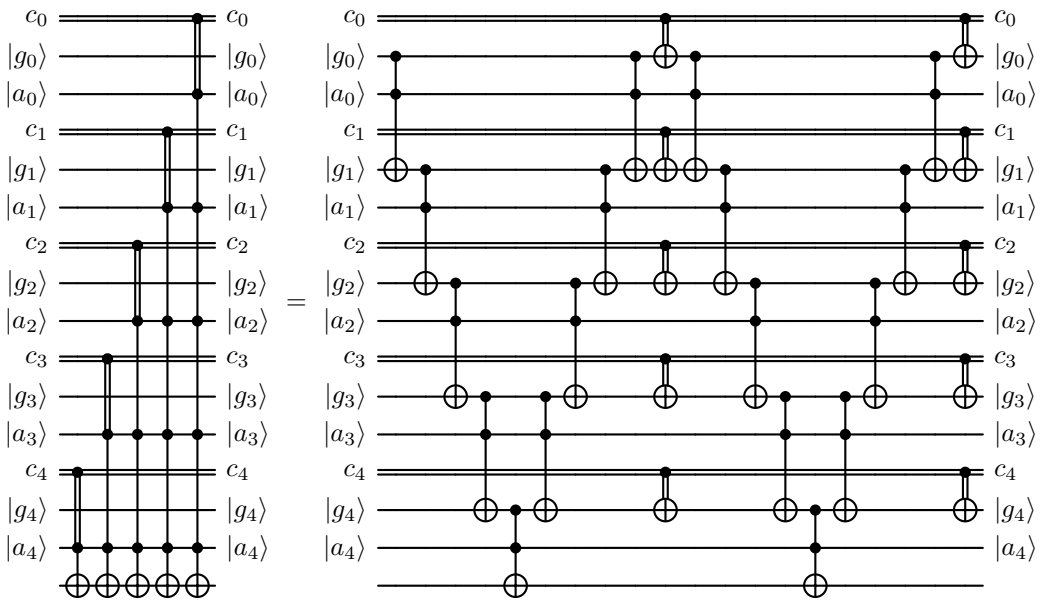
$$|\boldsymbol{a}\rangle\,|z\rangle \mapsto |\boldsymbol{a}\rangle\,|z \oplus (c < a)\rangle\,, \tag{29}$$

where $\boldsymbol{a} \in \{0,1\}^n$ and $z \in \{0,1\}$. An example of a classical–quantum comparator is shown in Figure 7. We show that this comparator can be implemented with the same asymptotically optimal gate count and circuit depth as the quantum–quantum comparator, using one dirty ancilla.

Theorem 3. *The n-bit classical–quantum comparator can be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n)$ gates and $\Theta(\log n)$ depth, using one dirty ancilla qubit.*

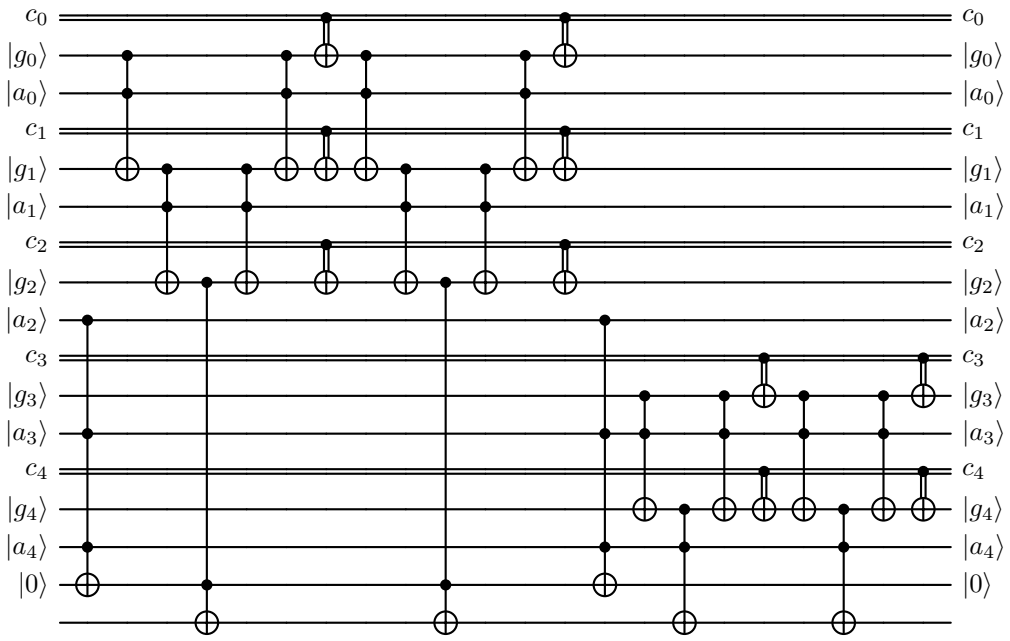
Proof. Consider the classical–quantum comparator circuit given in Figure 7. The subcircuit in

slice 2 can be implemented as follows with $\mathcal{V}_2^{(n)}$ operators and n dirty ancilla qubits:



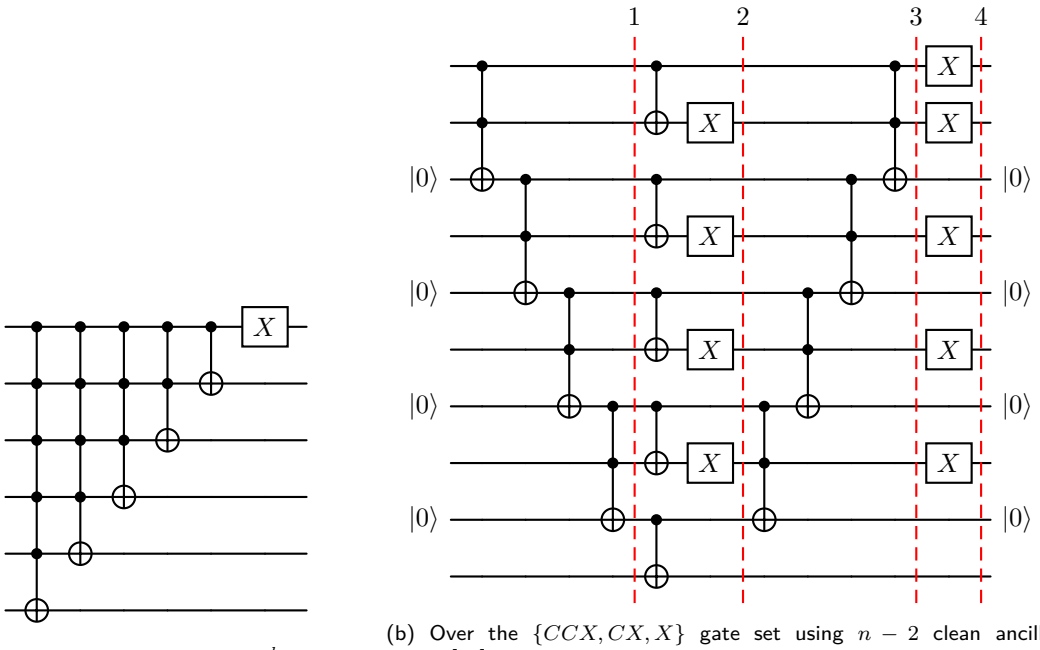
(30)

By introducing one clean ancilla qubit, this circuit can be split into two parts, with two $\mathcal{V}_2^{(\lceil \frac{n}{2} \rceil)}$ operators in the first part and two $\mathcal{V}_2^{(\lfloor \frac{n}{2} \rfloor)}$ operators in the second part:



(31)

The first two $\mathcal{V}_2^{(\lceil \frac{n}{2} \rceil)}$ operators can be implemented using the bottom $\lceil \frac{n}{2} \rceil$ input qubits as dirty ancilla qubits. Similarly, the last two $\mathcal{V}_2^{(\lfloor \frac{n}{2} \rfloor)}$ operators can be implemented using the top $\lceil \frac{n}{2} \rceil$ input qubits as dirty ancilla qubits. For instance, for the circuit of Equation (31), we can use a_2, a_3, and a_4 instead of g_0, g_1, and g_2, and we can use a_0 and a_1 instead of g_3 and g_4. We obtain the

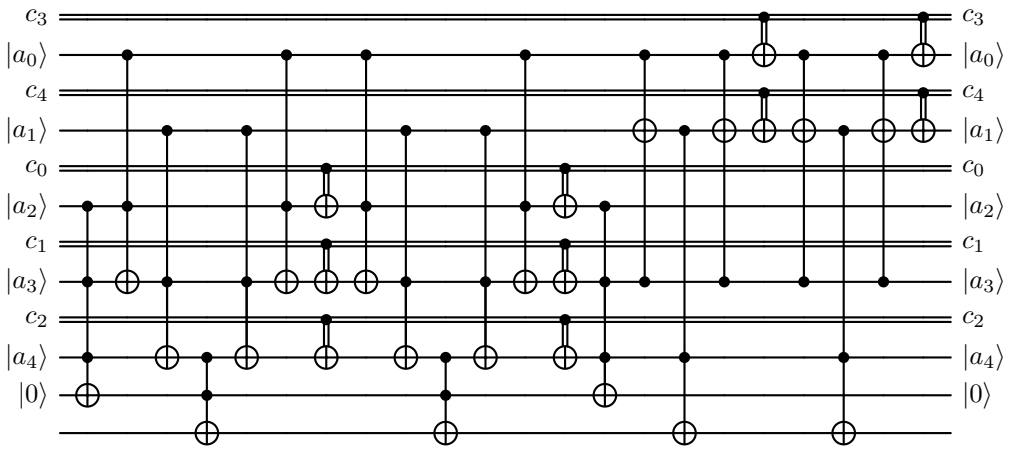


(a) Naive implementation using C^kX gates.

(b) Over the $\{CCX, CX, X\}$ gate set using $n-2$ clean ancilla qubits [10].

Figure 8: Quantum circuits for the 6-bit incrementer.

following circuit:



(32)

All $\mathcal{V}_2$ operators can then be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n)$ gates and $\Theta(\log n)$ circuit depth, as shown in Theorem 2. Finally, the clean ancilla can be replaced by a dirty ancilla using the circuit identity of Figure 2(a). □

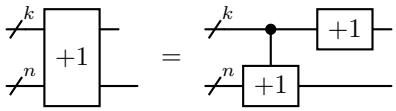
5 Quantum circuits for incrementers

An n-bit quantum incrementer computes the following map:

$$|\boldsymbol{x}\rangle \mapsto |(\boldsymbol{x}+1) \bmod 2^n\rangle\,, \tag{33}$$

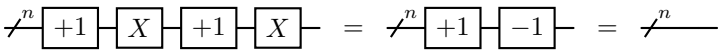
where $\boldsymbol{x} \in \{0,1\}^n$. An example of a naive implementation using C^kX gates is shown in Figure 8(a). A useful property of the increment operator is that it admits the following decomposition, which

can be applied recursively to obtain the naive implementation of Figure 8(a):



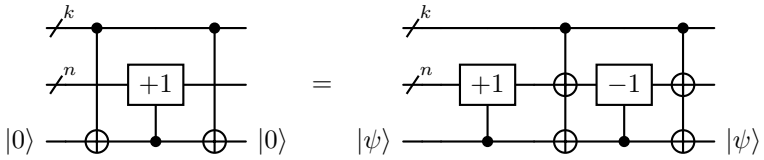
(34)

The increment operator U does not satisfy $U^2 = I$, which is problematic because we cannot directly apply the second circuit identity of Figure 2(a), a key tool when working with dirty ancilla qubits. However, the increment operator satisfies the following circuit identity:



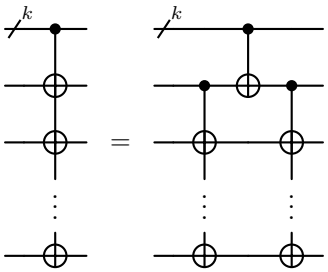
(35)

which means that instead of using Figure 2(a), we can rely on the following circuit identity [10]:



(36)

The C^kX gates convert the decrement operation into an increment operation whenever the control register is in the $|1\rangle^{\otimes k}$ state. When the control register is not in the $|1\rangle^{\otimes k}$ state, the C^kX gates act as the identity, and the increment and decrement gates cancel each other if the dirty ancilla is in the $|1\rangle$ state; otherwise, they both act as the identity when the dirty ancilla is in the $|0\rangle$ state. The k-controlled fan-out gates in Equation (36) can be implemented as follows:



(37)

using two fan-out gates, each implementable with $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth (Lemma 2), and one C^kX gate, implementable with $\mathcal{O}(k)$ gates and $\mathcal{O}(\log k)$ circuit depth (Lemma 1).

We first state a series of lemmas for constructions that will be used in our main result: a quantum circuit implementing the increment operator with $\Theta(n)$ $\{CCX, CX, X\}$ gates, $\Theta(\log n)$ circuit depth, and one dirty ancilla.

Lemma 7. *A k-controlled strong promise gate with a promise register of size $n-1$ whose target unitary is the n-bit incrementer can be implemented with $\mathcal{O}(n+k)$ $\{CCX, CX, X\}$ gates and $\mathcal{O}(\log n + \log k)$ circuit depth.*

Proof. We start from Gidney's circuit for implementing an n-bit incrementer with $\mathcal{O}(n)$ $\{CCX, CX, X\}$ gates, $\mathcal{O}(n)$ circuit depth, and $n-2$ clean ancilla qubits [10]. An example for $n = 6$ is given in Figure 8(b). Replacing the $n-2$ clean ancilla qubits with a promise register of the same size yields a strong promise gate whose target unitary is the increment operator.

We then add a control to each gate to obtain a singly controlled promise gate with the incrementer as target unitary. By Figure 3(a), the two ladders of CCX gates in slices 1 and 3 do not need to be controlled. For example, for $n = 6$ we obtain the circuit of Figure 9. All the other layers of gates controlled by the top qubit in slices 2 and 4 can be implemented by applying the construction of Lemma 5, with a total of $\mathcal{O}(n)$ $\{CCX, CX, X\}$ gates and circuit depth $\mathcal{O}(\log n)$.

We implement the two CCX ladders using the construction of Lemma 4, with $\mathcal{O}(n)$ CCX gates and circuit depth $\mathcal{O}(\log n)$, using $n - 2 - \mathcal{O}(\log n)$ ancilla qubits from the promise register.

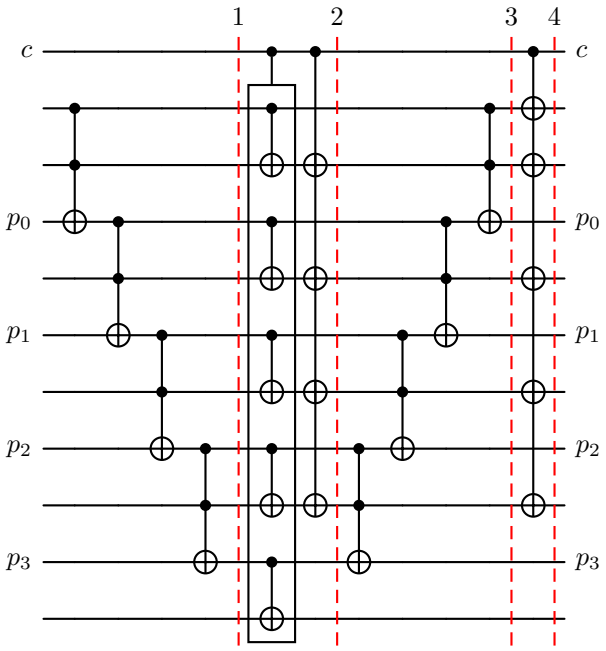
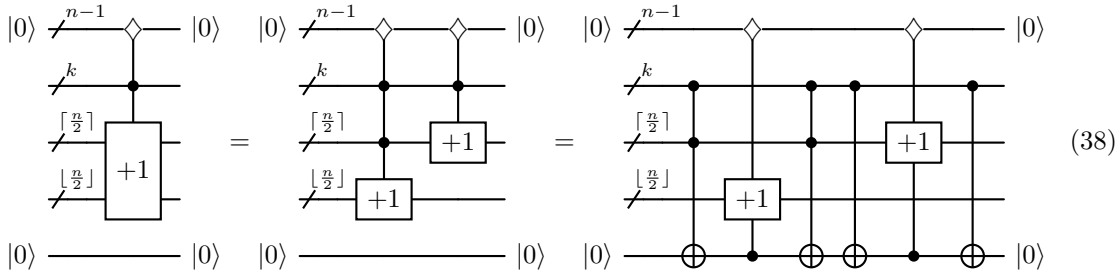


Figure 9: Quantum circuit for a controlled promise gate whose target unitary is the 6-bit incrementer. Here c denotes the control qubit and $\boldsymbol{p}$ denotes the promise register. The circuit is built from Figure 8(b) by replacing the clean ancilla qubits with a promise register $\boldsymbol{p}$ and adding a control to the gates in slices 2 and 4.

This would require a promise register of size at least $2n-4$. To use a promise register of smaller size, and to generalize to k control qubits, we use the following circuit identity:



(38)

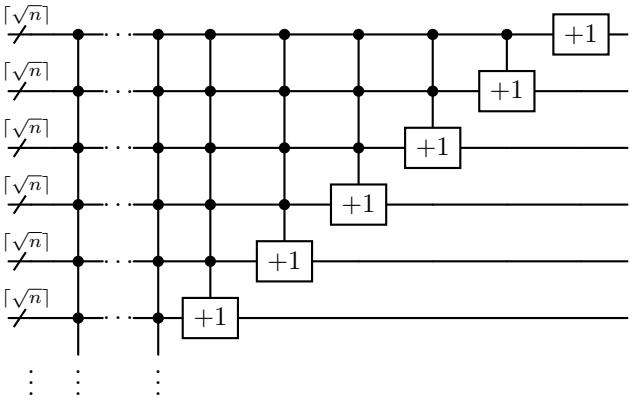
The two singly controlled promise gates whose target unitary is the incrementer can then be implemented as described above with a gate count of $\mathcal{O}(n)$ and circuit depth $\mathcal{O}(\log n)$, using $2\lceil \frac{n}{2} \rceil - 4 < n-2$ ancilla qubits from the promise register.

Finally, we exchange the clean ancilla qubit from Equation (38) for a dirty ancilla (which can be taken from the promise register) using Equation (36). Thus, the total gate count is $\mathcal{O}(n+k)$ and the circuit depth is $\mathcal{O}(\log n + \log k)$, using a promise register of size $n-1$. □

The following lemma relies on Lemma 7 to further reduce the size of the promise register.

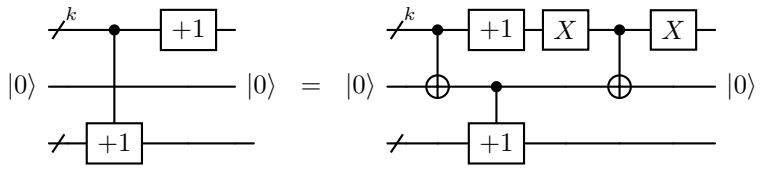
Lemma 8. *A controlled strong promise gate with a promise register of size $2\lfloor \sqrt{n} \rfloor$ whose target unitary is the n-bit increment operator can be implemented with $\mathcal{O}(n)$ $\{CCX, CX, X\}$ gates and $\mathcal{O}(\log n)$ circuit depth.*

Proof. Using Equation (34), an incrementer can be implemented as follows:



(39)

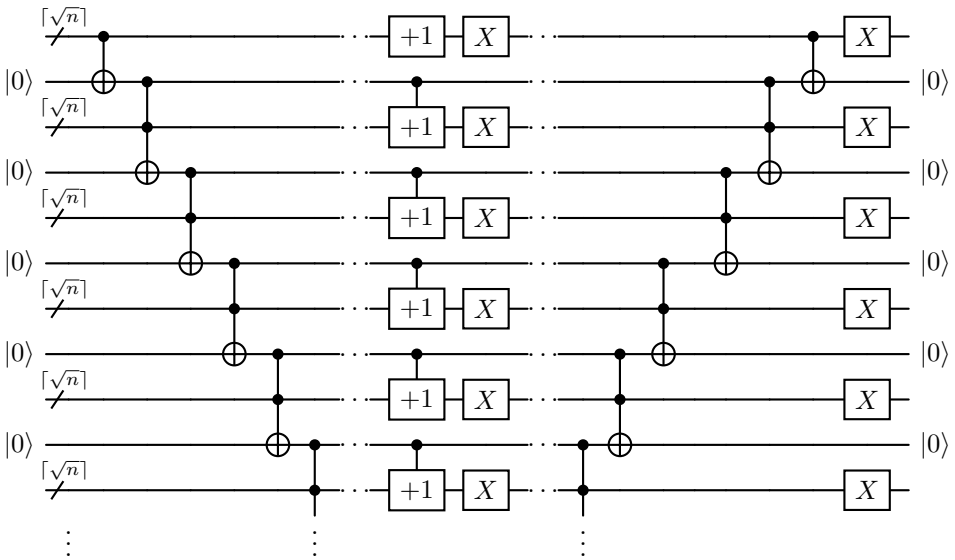
where the last register has size at most $\lceil \sqrt{n} \rceil$. We then use the following circuit identity:



(40)

which holds because if the top register is in the $|1\rangle^{\otimes k}$ state, the increment operation switches it to the $|0\rangle^{\otimes k}$ state. The X gates then switch the register back to the $|1\rangle^{\otimes k}$ state so that the two C^kX gates have the same effect on the ancilla.

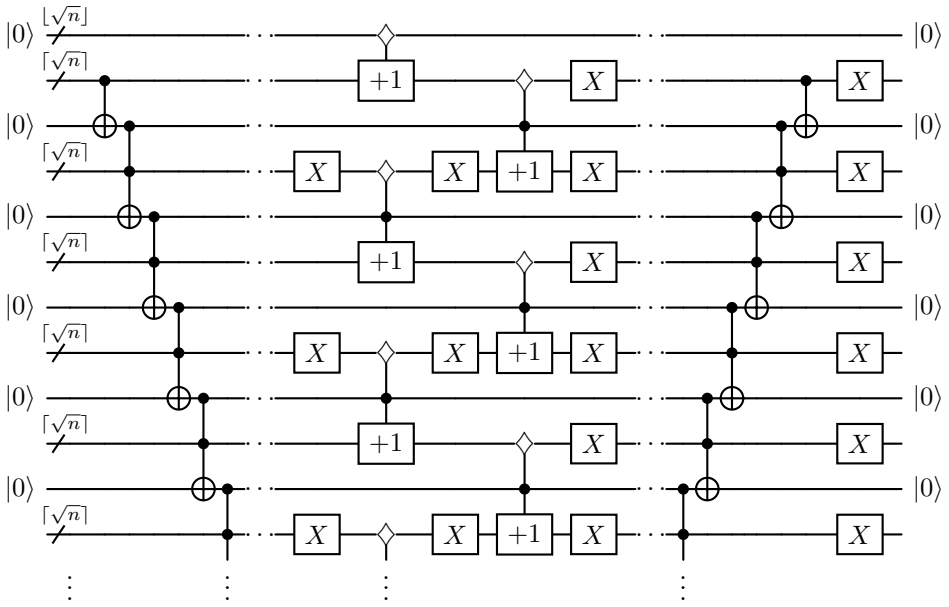
By Equations (39) and (40), we can then obtain the following implementation of an n-bit incrementer:



(41)

using $\lfloor \sqrt{n} \rfloor$ clean ancilla qubits. Each controlled increment on a register of size $\lceil \sqrt{n} \rceil$ is activated only when all qubits of the $\lceil \sqrt{n} \rceil$-qubit register above it are in the state $|1\rangle$. That register can therefore serve as a promise register for the controlled increment, as in Figure 2(b). Since each promise gate uses the register above it as its promise register, we cannot apply all promise gates simultaneously; instead, we proceed in two rounds. Moreover, if the increment on the register above has already been performed, there is no need to apply X gates to the promise register, as the increment operator maps the $|1\rangle^{\otimes k}$ state to the $|0\rangle^{\otimes k}$ state. Applying this to the circuit of

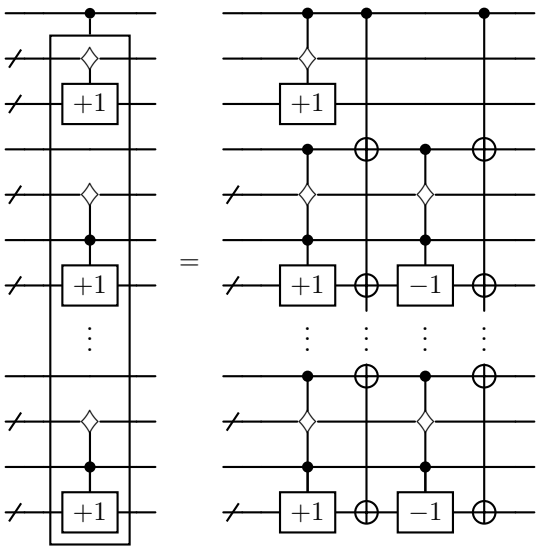
Equation (41), we obtain:



(42)

where $\lfloor\sqrt{n}\rfloor$ ancilla qubits were inserted at the top to serve as the promise register for the topmost promise gate.

We then replace the clean ancilla qubits with a promise register of size $2\lfloor\sqrt{n}\rfloor$ and add a control to each gate to obtain a controlled strong promise gate whose target unitary is the increment operator. By Figure 3(a), the two ladders of C^kX gates do not need to be controlled. Using $\lfloor\sqrt{n}\rfloor$ qubits from the promise register, these two ladders can be implemented with $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth, as stated in Corollary 1. Finally, using available dirty ancilla qubits from the promise register, we apply the following circuit identity based on Equation (36):



(43)

We obtain the circuit presented in Figure 10.

We now analyze the cost of the resulting circuit. The two ladders of C^kX gates contribute $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth, as already discussed. The fan-out gates can be implemented with $\mathcal{O}(n)$ CX gates and $\mathcal{O}(\log n)$ circuit depth (Lemma 2). The controlled strong promise gates, whose target unitary is the $\lceil\sqrt{n}\rceil$-bit increment or decrement operator, each require $\mathcal{O}(\sqrt{n})$ gates and $\mathcal{O}(\log n)$ circuit depth (Lemma 7). Since there are $\mathcal{O}(\sqrt{n})$ such gates arranged in a constant number of layers, their total cost is $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth. The overall complexity of the circuit presented in Figure 10 is therefore $\mathcal{O}(n)$ $\{CCX, CX, X\}$ gates and $\mathcal{O}(\log n)$ circuit depth. □

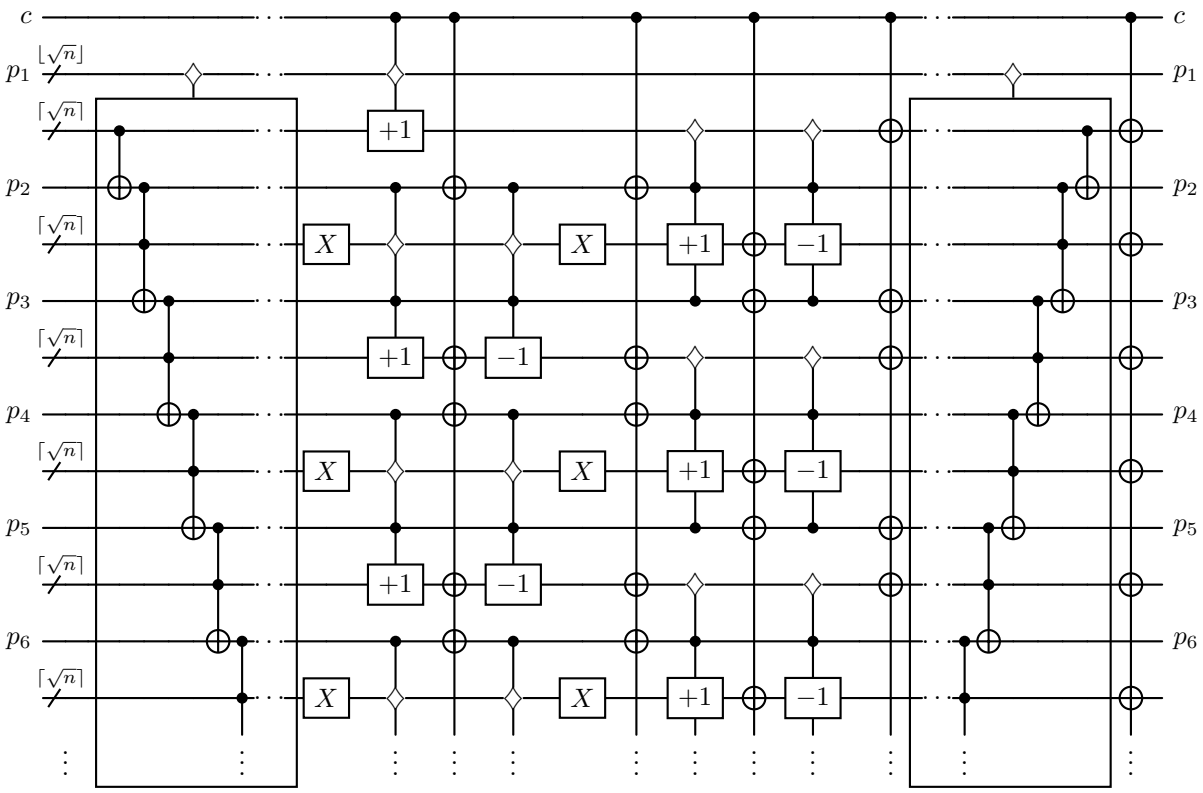
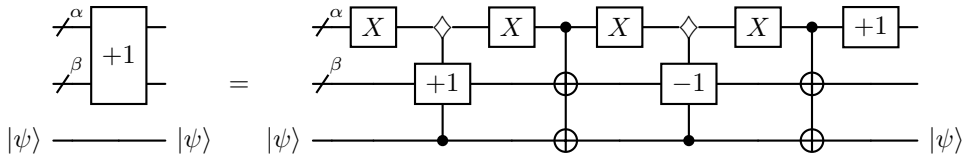


Figure 10: Quantum circuit for a controlled strong promise gate whose target unitary is the increment operator. c denotes the control qubit and $\boldsymbol{p}$ denotes the promise register of size $2\lfloor\sqrt{n}\rfloor$.

We now use Lemma 8 to prove the following theorem.

Theorem 4. *The n-bit increment operator can be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n)$ gates and $\Theta(\log n)$ circuit depth, using one dirty ancilla qubit.*

Proof. Based on Equation (34) and Equation (36) we have:



(44)

The top α-qubit register serves as the promise register for the two strong promise gates acting on the β-qubit register. The X gates conjugating each promise gate ensure that, whenever the top register is in the state $|1\rangle^{\otimes\alpha}$, the promise register is mapped to $|0\rangle^{\otimes\alpha}$, satisfying the promise. When the top register is not in the state $|1\rangle^{\otimes\alpha}$, the promise is not satisfied, but the C^kX gates act as the identity, so the two strong promise gates (for the increment and decrement operators) compose to the identity, as required.

Let $\alpha = 2\lfloor\sqrt{n}\rfloor$ and $\beta = n - 2\lfloor\sqrt{n}\rfloor$. The α-controlled fan-out gates can be implemented with a total cost of $\mathcal{O}(n)$ gates and $\mathcal{O}(\log n)$ circuit depth, using Equation (37). By Lemma 8, the two controlled strong promise gates whose target unitaries are the β-bit increment and decrement operators can be implemented with $\mathcal{O}(n)$ $\{CCX, CX, X\}$ gates and $\mathcal{O}(\log n)$ circuit depth. We then implement the increment operator on the top α qubits recursively. The total gate count satisfies

$$C(n) = \Theta(n) + C(2\lfloor\sqrt{n}\rfloor) \tag{45}$$

and the total circuit depth satisfies

$$D(n) = \Theta(\log n) + D(2\lfloor\sqrt{n}\rfloor) \tag{46}$$

which yield $C(n) = \Theta(n)$ and $D(n) = \Theta(\log n)$, respectively. □

6 Classical–quantum adder and modular multiplication

In this section, we show how the results of the previous sections can be combined to construct efficient classical–quantum adders, which in turn can serve as building blocks for modular multiplication circuits used in Shor's factoring algorithm [4].

6.1 Classical–quantum adder

An n-bit classical–quantum adder computes the following map:

$$|\boldsymbol{x}\rangle \mapsto |(\boldsymbol{x} + \boldsymbol{c}) \bmod 2^n\rangle\,, \tag{47}$$

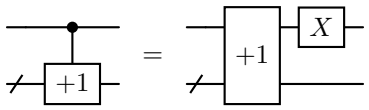
where $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{c} \in \{0,1\}^n$ is a classically known constant.

We now show how plugging our optimal constructions for incrementers and comparators into the approach of Häner et al. [5] yields improved asymptotic complexities for classical–quantum addition.

Theorem 5. *The n-bit classical–quantum adder can be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n \log n)$ gates and $\Theta(\log^2 n)$ circuit depth, using one dirty ancilla qubit.*

Proof. The classical–quantum adder can be implemented using the circuit of Figure 11, following the approach of Häner et al. [5]. The carry operator computes the highest bit of the addition of x_L and c_L. This bit is 1 if and only if $x_L + c_L \geq 2^{\lceil n/2 \rceil}$, which means the carry can be computed via the classical–quantum comparison $x_L \geq 2^{\lceil n/2 \rceil} - c_L$.

By Theorem 3, the carry operator can be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n)$ gates and $\Theta(\log n)$ circuit depth. The two fan-out gates can also be implemented with $\Theta(n)$ CX gates and $\Theta(\log n)$ circuit depth (Lemma 2). The two controlled incrementers can each be implemented using one incrementer and X gates via the following circuit identity:



(48)

And by Theorem 4, the increment operator can be implemented over the $\{CCX, CX, X\}$ gate set with $\Theta(n)$ gates and $\Theta(\log n)$ circuit depth.

The total gate count then satisfies

$$C(n) = \Theta(n) + 2C\left(\frac{n}{2}\right) \tag{49}$$

and the total circuit depth satisfies

$$D(n) = \Theta(\log n) + D\left(\frac{n}{2}\right) \tag{50}$$

which yield $C(n) = \Theta(n \log n)$ and $D(n) = \Theta(\log^2 n)$. □

Historically, and somewhat counterintuitively, classical–quantum addition has been harder to implement efficiently than quantum–quantum addition, due to the lack of available workspace qubits. Recently, the gap between the two settings has narrowed: Gidney [14] discovered a classical–quantum adder with a linear number of gates, linear depth, and a constant number of ancilla qubits, matching the costs of the quantum–quantum adder of Cuccaro et al. [11]. Our results further contribute to closing this gap: the classical–quantum adder presented here achieves the same linearithmic gate count and sublinear circuit depth as Remaud's quantum–quantum adder [9], which also uses a minimal number of qubits.

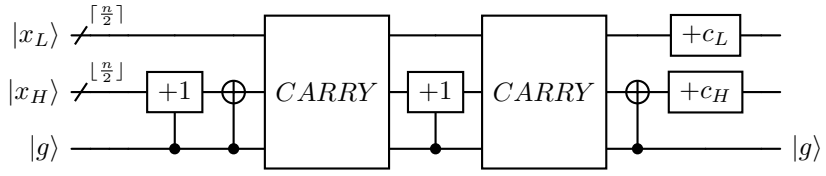


Figure 11: Circuit for classical–quantum addition [5], where $\boldsymbol{c}$ is the classical constant added to the quantum register $\boldsymbol{x}$, with x_H, x_L and c_H, c_L denoting the high- and low-bit halves of $\boldsymbol{x}$ and $\boldsymbol{c}$, respectively.

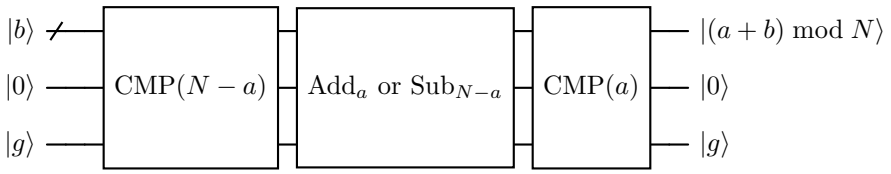


Figure 12: Construction of a modular adder from a non-modular adder [26]. The CMP gate computes the comparison $b < N - a$, and the result determines whether a or $a - N$ is added.

6.2 Modular multiplication

The classical–quantum adder presented in the previous subsection can be used to implement Shor's factoring algorithm [4], following the approach of previous works [34, 26, 5]. Factoring an n-bit integer requires computing a modular exponentiation, which decomposes into $\mathcal{O}(n)$ controlled modular multiplications [30]. Each controlled modular multiplication is in turn implemented via $\mathcal{O}(n)$ controlled modular additions, and each modular addition reduces to a constant number of classical–quantum additions and comparisons, as shown in Figure 12. This yields a total of $\mathcal{O}(n^2)$ classical–quantum additions, each with a gate count of $\mathcal{O}(n \log n)$ and a circuit depth of $\mathcal{O}(\log^2 n)$. The overall implementation of Shor's algorithm therefore has a total gate count of $\mathcal{O}(n^3 \log n)$ and a circuit depth of $\mathcal{O}(n^2 \log^2 n)$, using $2n + 2$ qubits.

A comparison with other space-efficient implementations of Shor's algorithm is given in Table 2.

7 Conclusion

We presented Clifford+Toffoli quantum circuits that match the established lower bounds in both gate count and circuit depth for comparators and incrementers, while using a minimal number of qubits. Our optimal results are analogous to those of Nie et al. [6], who showed how to implement the multi-controlled X gate with asymptotically optimal gate count and depth using a minimal number of ancilla qubits. This raises a natural question: can the family of operators implementable with linear gate count, logarithmic depth, and a minimal number of qubits be further extended? In particular, this question remains open for quantum addition, for which a gap persists between the known lower bounds and the best-known constructions.

While the proposed circuits are asymptotically optimal in gate count and circuit depth, the constant factors have not been optimized, and numerous avenues remain for improving these constructions, making them more practical, and exploring various resource trade-offs. These optimizations, as well as detailed resource estimation and cost comparisons with other approaches, are left as future work.

A central ingredient of our constructions is the notion of promise gates, which provides a framework for reasoning about conditionally clean ancilla qubits and enabled us to discover new ways of making effective use of them in quantum circuit design. In particular, we established a systematic method for trading clean ancilla qubits for control qubits with little overhead in gate count and circuit depth. We expect this technique to find applications beyond the operators studied in this paper.

Reference	Year	Qubits	Depth	Gate Count
Beckman et al. [27]	1996	$5n + 1$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Vedral et al. [28]	1996	$4n + 3$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Zalka [29]	1998	$3n + \mathcal{O}(1)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Beauregard [30]	2003	$2n + 3$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3 \log^2 n)$
Takahashi et al. [26]	2006	$2n + 2$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3 \log^2 n)$
Zalka [31]	2006	$1.5n + \mathcal{O}(1)$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3 \log^2 n)$
Häner et al. [5]	2016	$2n + 2$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3 \log n)$
Gidney [13]	2017	$2n + 1$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3 \log n)$
Gidney et al. [32]	2019	$3n + 0.002n \log n$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n^3 \log n)$
Chevignard et al. [33]	2024	$n/2 + o(n)$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^2 \log^4 n)$
This paper	2026	$2n + 2$	$\mathcal{O}(n^2 \log^2 n)$	$\mathcal{O}(n^3 \log n)$

Table 2: Comparison of space-efficient quantum circuits for Shor's factoring algorithm applied to an n-bit RSA integer, in terms of qubit count, circuit depth, and gate count.

Acknowledgements

This work is dedicated to the memory of Maxime Remaud, with whom I shared many valuable discussions and a deep enthusiasm for this line of research.
The author thanks Silas Dilkes and Yuka Kikuchi for reviewing this paper.

References

[1] B. L. Douglas and J. B. Wang. "Efficient quantum circuit implementation of quantum walks". Physical Review A**79** (2009).
[2] Christoph Durr and Peter Hoyer. "A Quantum Algorithm for Finding the Minimum" (1999). arXiv:quant-ph/9607014.
[3] Christoffer Hindlycke and Jan-Åke Larsson. "Single-qubit rotation algorithm with logarithmic Toffoli count and gate depth". Physical Review Research**6** (2024).
[4] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In Proceedings 35th Annual Symposium on Foundations of Computer Science. Page 124–134. SFCS-94. IEEE Comput. Soc. Press (1994).
[5] Thomas Haner, Martin Roetteler, and Krysta M. Svore. "Factoring using 2n+2 qubits with Toffoli based modular multiplication". Quantum Information and Computation **17**, 673–684 (2017).
[6] Junhong Nie, Wei Zi, and Xiaoming Sun. "Quantum circuit for multi-qubit Toffoli gate with optimal resource" (2024). arXiv:2402.05053.
[7] Baptiste Claudon, Julien Zylberman, César Feniou, Fabrice Debbasch, Alberto Peruzzo, and Jean-Philip Piquemal. "Polylogarithmic-depth controlled-NOT gates without ancilla qubits". Nature Communications**15** (2024).
[8] Tanuj Khattar and Craig Gidney. "Rise of conditionally clean ancillae for efficient quantum circuit constructions". Quantum **9**, 1752 (2025).
[9] Maxime Remaud and Vivien Vandaele. "Ancilla-Free Quantum Adder with Sublinear Depth". Page 137–154. Springer Nature Switzerland. (2025).
[10] Craig Gidney. "Constructing Large Increment Gates". url: `https://algassert.com/circuits/2015/06/12/Constructing-Large-Increment-Gates.html`.
[11] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. "A new quantum ripple-carry addition circuit" (2004). arXiv:quant-ph/0410184.
[12] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. "Quantum addition circuits and unbounded fan-out". Quantum Information and Computation **10**, 872–890 (2010).

[13] Craig Gidney. "Factoring with n+2 clean qubits and n-1 dirty qubits" (2018). arXiv:1706.07884.
[14] Craig Gidney. "A Classical-Quantum Adder with Constant Workspace and Linear Gates" (2025). arXiv:2507.23079.
[15] Tommaso Toffoli. "Reversible computing". Page 632–644. Springer Berlin Heidelberg. (1980).
[16] Thomas G. Draper. "Addition on a Quantum Computer" (2000). arXiv:quant-ph/0008033.
[17] M. Fang, S. Fenner, F. Green, S. Homer, and Y. Zhang. "Quantum lower bounds for fanout". Quantum Information and Computation **6**, 46–57 (2006).
[18] V.V. Shende, A.K. Prasad, I.L. Markov, and J.P. Hayes. "Synthesis of reversible logic circuits". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**, 710–722 (2003).
[19] Michael Beverland, Earl Campbell, Mark Howard, and Vadym Kliuchnikov. "Lower bounds on the non-Clifford resources for quantum computations". Quantum Science and Technology **5**, 035009 (2020).
[20] David Gosset, Robin Kothari, and Chenyi Zhang. "Multi-qubit Toffoli with exponentially fewer T gates" (2025). arXiv:2510.07223.
[21] Anne Broadbent and Elham Kashefi. "Parallelizing quantum circuits". Theoretical Computer Science **410**, 2489–2510 (2009).
[22] T.G. Draper, S.A. Kutin, E.M. Rains, and K.M. Svore. "A logarithmic-depth quantum carry-lookahead adder". Quantum Information and Computation **6**, 351–369 (2006).
[23] Maxime Remaud. "Quantum adders: on the structural link between the ripple-carry and carry-lookahead techniques" (2025). arXiv:2510.00840.
[24] Oded Goldreich. "On Promise Problems: A Survey". Page 254–290. Springer Berlin Heidelberg. (2006).
[25] V.V. Shende, S.S. Bullock, and I.L. Markov. "Synthesis of quantum-logic circuits". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**, 1000–1010 (2006).
[26] Y. Takahashi and N. Kunihiro. "A quantum circuit for Shor's factoring algorithm using 2n+2 qubits". Quantum Information and Computation **6**, 184–192 (2006).
[27] David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. "Efficient networks for quantum factoring". Physical Review A **54**, 1034–1063 (1996).
[28] Vlatko Vedral, Adriano Barenco, and Artur Ekert. "Quantum networks for elementary arithmetic operations". Physical Review A **54**, 147–153 (1996).
[29] Christof Zalka. "Fast versions of Shor's quantum factoring algorithm" (1998). arXiv:quant-ph/9806084.
[30] S. Beauregard. "Circuit for Shor's algorithm using 2n+3 qubits". Quantum Information and Computation **3**, 175–185 (2003).
[31] Christof Zalka. "Shor's algorithm with fewer (pure) qubits" (2006). arXiv:quant-ph/0601097.
[32] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". Quantum **5**, 433 (2021).
[33] Clémence Chevignard, Pierre-Alain Fouque, and André Schrottenloher. "Reducing the Number of Qubits in Quantum Factoring". Page 384–415. Springer Nature Switzerland. (2025).
[34] Rodney Van Meter and Kohei M. Itoh. "Fast quantum modular exponentiation". Physical Review A**71** (2005).

A Proofs

A.1 Proof of Lemma 4

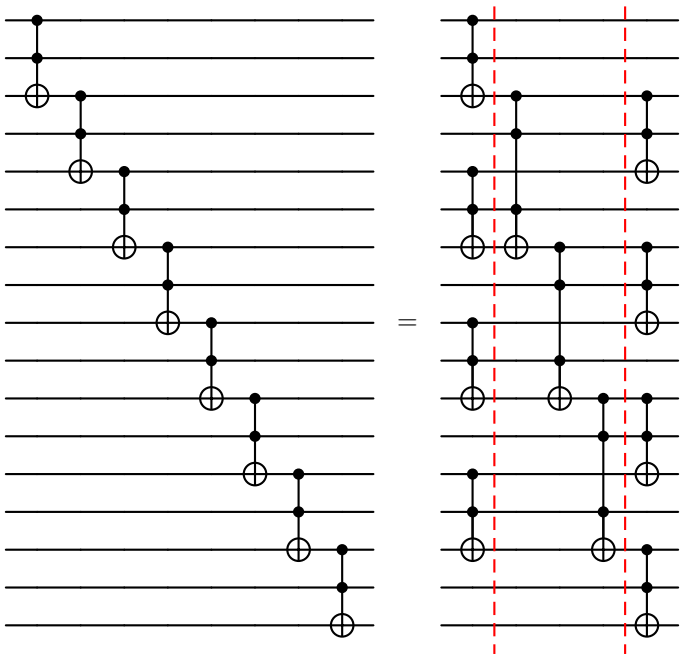
Proof. It was shown in [9] that the $\mathcal{L}_2^{(n-1)}$ operator can be implemented with

$$C(n) = 2n - 2 - \lfloor \log_2 n \rfloor - \left\lfloor \log_2 \frac{2n}{3} \right\rfloor \tag{51}$$

C^kX gates and a circuit depth of

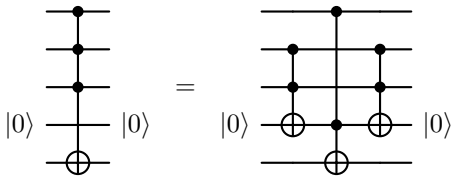
$$D(n) = \lfloor \log_2 n \rfloor + \left\lfloor \log_2 \frac{2n}{3} \right\rfloor, \tag{52}$$

without ancilla qubits. The recursive algorithm presented in [9] achieves this by inserting a layer of C^kX gates at the beginning and end of the circuit, then recursing on a smaller C^kX ladder in the middle. For example, one iteration of the algorithm for $\mathcal{L}_2^{(8)}$ yields the circuit on the right-hand side of the following identity:



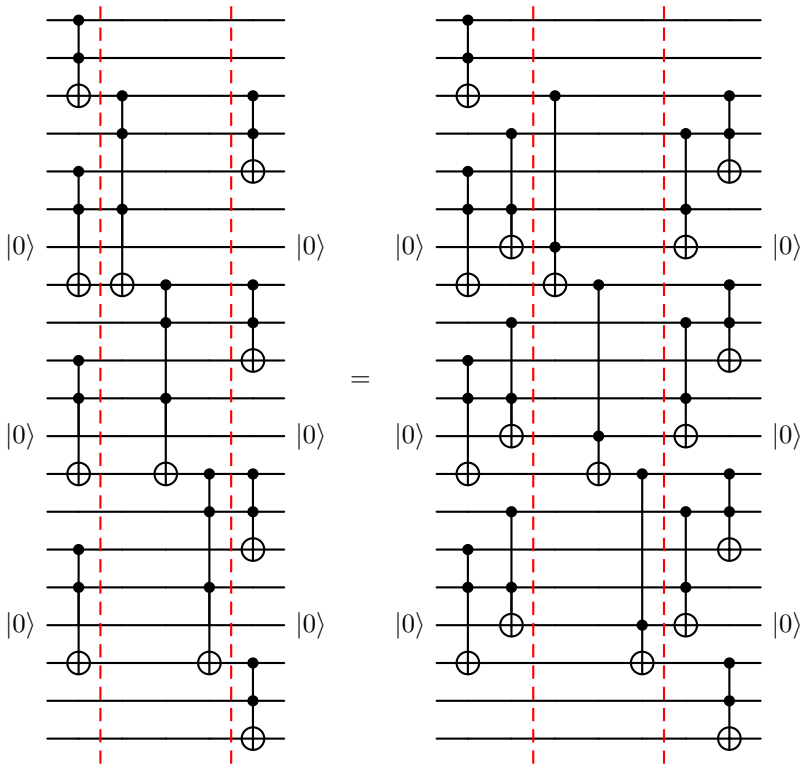
(53)

Before recursing on the C^kX ladder in the middle slice, we can convert it into a ladder of CCX gates by introducing ancilla qubits, using the following circuit identity:



(54)

For example, the first iteration of the algorithm for $\mathcal{L}_2^{(8)}$ yields:



(55)

The identity in Equation (54) introduces two additional gates for each gate in the middle C^kX ladder. As a result, the total gate count is at most triple that of the circuit constructed by the algorithm of [9], minus the $n-1$ gates in the first and last layers. This yields a CCX count of at most

$$3(C(n) - n + 1) = 3n - 3 - 3\lfloor \log_2 n \rfloor - 3\left\lfloor \log_2 \frac{2n}{3} \right\rfloor. \tag{56}$$

At each recursive call, the two slices of gates inserted around the middle ladder have a depth of 2 instead of 1, which at most doubles the circuit depth:

$$2D(n) = 2\lfloor \log_2 n \rfloor + 2\left\lfloor \log_2 \frac{2n}{3} \right\rfloor. \tag{57}$$

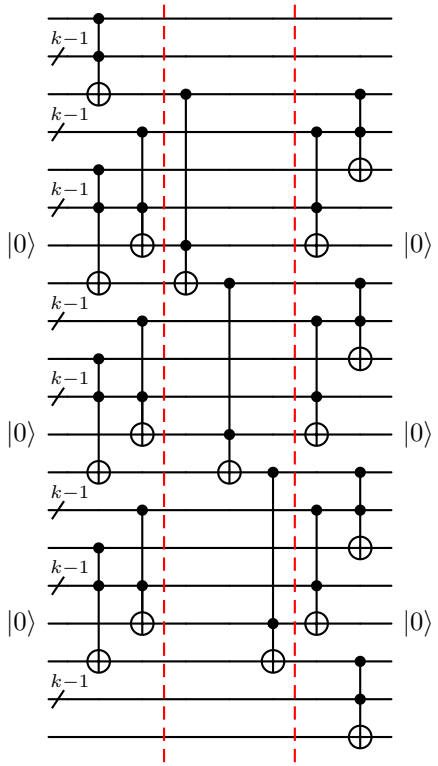
Finally, at each recursive call, the number of ancilla qubits introduced equals the number of gates in the middle C^kX ladder. Therefore, the total number of ancilla qubits in the final circuit is at most $C(n)$ minus the $n-1$ gates in the first and last layers:

$$C(n) - n + 1 = n - 1 - \lfloor \log_2 n \rfloor - \left\lfloor \log_2 \frac{2n}{3} \right\rfloor. \tag{58}$$

□

A.2 Proof of Corollary 1

Proof. We apply the same algorithm as the one described in Appendix A.2 for the proof of Lemma 4. For example, one iteration of the algorithm for $\mathcal{L}_k^{(8)}$ yields the following circuit:



(59)

The recursive call on the middle ladder of CCX gates yields a total of $\mathcal{O}(n)$ CCX gates and a circuit depth of $\mathcal{O}(\log n)$, by Lemma 4. The resulting circuit uses the same number of ancilla qubits as in Lemma 4. Finally, all the C^kX gates in the first two and last two layers can be implemented with a total gate count of $\mathcal{O}(nk)$ and a circuit depth of $\mathcal{O}(\log(nk))$, by Lemma 1. □

A.3 Proof of Corollary 4

Proof. We use the same construction as in the proof of Lemma 4. As shown in that proof, the $\mathcal{L}_2^{(n)}$ operator can be implemented with $\mathcal{O}(n)$ CCX gates and $\mathcal{O}(\log n)$ circuit depth, using n clean ancilla qubits. Each ancilla is introduced via the identity in Equation (54), in which the ancilla participates in a compute-uncompute pattern: it is computed by a CCX gate, used as a control for another CCX gate, and then uncomputed by a second CCX gate with the same controls. This uncomputation does not depend on the ancilla being initialized to $|0\rangle$, so the ancilla register is preserved regardless of its initial state. Replacing the clean ancilla qubits with a promise register of size n therefore yields a strong promise gate whose target unitary is the $\mathcal{L}_2^{(n)}$ operator, implemented with $\mathcal{O}(n)$ $\{CCX, CX, X\}$ gates and $\mathcal{O}(\log n)$ circuit depth.

The generalization to $\mathcal{L}_k^{(n)}$ follows by applying the same argument to the construction in the proof of Corollary 1, where the C^kX gates in the outer layers are likewise used in compute-uncompute pairs. □