IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 November 2025, accepted 22 December 2025, date of publication 31 December 2025,
date of current version 9 February 2026.

*Digital Object Identifier 10.1109/ACCESS.2025.3649805*

TOPICAL REVIEW

# Latency-Optimal Quantum Circuits for IoT Networks

**SAVO GLISIC[1], (Senior Member, IEEE), AND BEATRIZ LORENZO[2], (Senior Member, IEEE)**
[1]Department of Physics, Worcester Polytechnic Institute, Worcester, MA 01609, USA
[2]ECE Department, University of Massachusetts Amherst, Amherst, MA 01003, USA

Corresponding author: Savo Glisic (savo.glisic@ieee.org)

**ABSTRACT** Future IoT networks will demand increasingly lower latency for transmitting mission-critical information while simultaneously adopting quantum technologies to enhance security and accelerate information processing. This survey reviews the state-of-the-art in latency-optimal quantum circuit design, focusing on minimizing circuit latency without sacrificing essential functionality. The space is primarily allocated to the network level aspects, exploring balance between the network latency and the energy consumption, programmability and cost, while establishing a proper interface with the problems that should be delegated to the practical design carried on by circuit developers. First, we analyze how aggressive latency reduction constrains the use of *universal* and *reversible* gates—both of which tend to increase circuit depth (latency). Universal gates enable programmability, while reversible gates reduce energy dissipation, yet both can extend circuit latency. Aware of these trade-offs, we discuss latency-optimal design methodologies and their implementation aspects. Next, we discuss exact minimization of quantum circuits, presenting quantitative improvements. For instance, controlled-T gate implementations demonstrate reductions from 15 → 9 T gates (compression = 15/9), 16 → 12 CNOT gates (compression = 16/12), and 9 → 5 T-latency (compression = 9/5). Similarly, for a 1-bit full adder, reductions include 14 → 8 T gates (14/8), 12 → 10 CNOT gates (12/10), 4 → 2 H gates (2), and 8 → 2 T-latency (4).

**INDEX TERMS** Quantum circuits, latency minimization, IoT, next-generation networks.

## I. INTRODUCTION

Research and development in advanced IoT networks have devoted significant attention to latency reduction within both network protocols and circuit-level implementations of network control algorithms. As quantum information processing becomes increasingly relevant to next-generation IoT systems, understanding latency in quantum circuit design has become essential for network engineers and system architects [1], [2], [3]. Compared with classical technology, quantum circuits can be parallelized more effectively, enabling a reduction in circuit depth—and consequently, processing latency—at relatively lower cost. This improvement can be characterized by a *latency compression coefficient*, defined as the ratio (> 1) between the original and optimized circuit depths. Although the initial cost of quantum circuits may be high, their scalability and latency compression potential make them more cost-efficient as technology matures. The primary objective of this paper is to provide network designers with a comprehensive survey of techniques for reducing both latency and cost in quantum circuits. However, aggressively reducing circuit depth introduces limitations in the use of universal and reversible gates. The former provides programmability, while the latter reduces energy dissipation; both typically lead to greater depth (latency). The subsequent tables compare practical data for various quantum circuit configurations.

Table 1 compares the latency of universal and application-specific quantum circuits. While universal circuits support programmability, they typically exhibit an order-of-magnitude higher latency [1], [9]. Key references include: Nielsen and Chuang's foundational framework for quantum computing [1]; OpenQASM 3 for hybrid quantum-classical programming [2]; Google's Sycamore demonstration of quantum supremacy [3], [8]; IBM's Quantum System One [5]; and major algorithmic advances such as the Vari-

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Masini.

ational Quantum Eigensolver (VQE) [6] and optimized implementations of Shor's algorithm [7]. Reference [9] further reports superconducting qubit operations achieving fidelities sufficient for surface-code error correction—a critical milestone toward scalable, fault-tolerant systems.

Table 2 compares reversible and application-specific (irreversible) quantum circuits. While reversible circuits drastically reduce energy dissipation, they often entail significantly larger depth and higher gate counts—sometimes by several orders of magnitude [10], [17]. Foundational works include Frank [10] and Toffoli [11] on reversible logic; Landauer [12] and Bennett [13] on thermodynamic and logical reversibility; and more recent synthesis and optimization studies [14], [17].

Table 3 extends this comparison using additional studies such as [18] (BDD-based synthesis of reversible logic) and [19] (systematic methods for low-power reversible architectures). Collectively, these results demonstrate that aggressive latency minimization imposes trade-offs in programmability, manufacturability, and energy efficiency. Nevertheless, many IoT applications demand extremely low latency, prompting continued research into optimal circuit design [20], [33].

**TABLE 1. Latency comparison: universal vs. application-specific quantum circuits.**

| No. | Feature | Universal Quantum Circuit | Application-Specific Quantum Circuit |
|---|---|---|---|
| 1 | Purpose | General-purpose computation | Tailored to a specific algorithm or application |
| 2 | Flexibility | High flexibility to emulate any unitary operation [1] | Limited to a fixed function or narrow task [2] |
| 3 | Gate Count (Typical) | 10^4 – 10^6 gates for moderate-depth circuits [3] | Often < 10^3 gates depending on algorithm [4] |
| 4 | Qubit Count | 50–100+ qubits in current devices [3] | 10–50 qubits, depending on task-specific requirements [5] |
| 5 | Execution Speed | Slower due to generality and control overhead | Faster; optimized for fewer gates and shallow depth [6] |
| 6 | Latency (per run) | ~10–1000 µs (depends on gate count and error correction) [7] | ~1–100 µs (smaller circuits, lower overhead) [7][8] |
| 7 | Gate Latency | Single-qubit: ~10–100 ns; Two-qubit: ~100–500 ns [7][9] | Same, but fewer gates result in reduced total latency |
| 8 | Reusability | High; supports many algorithms | Low; designed for one function |
| 9 | Optimization | General optimization hard due to abstraction and variability | Highly optimized for specific algorithm or cost metric [6][9] |
| 10 | Example Systems | IBM Quantum (Qiskit), Google Sycamore [3][5] | QFT, Grover's, Shor's, VQE [4][6] |

In this survey, we review methods for controlled and optimized latency reduction in both classical and quantum circuits implementing diverse network functionalities.

Following introductory Section I, discusses latency-optimal design strategies and implementation aspects. Section III details exact circuit minimization, while Section IV addresses decomposition of continuous-variable (CV) operations into

**TABLE 2. Comparison - reversible vs. application-specific/Irreversible quantum circuits.**

| No. | Feature | Reversible Quantum Circuit | Application-Specific / Irreversible Quantum Circuit |
|---|---|---|---|
| 1 | Definition | Implements bijective (one-to-one) transformations — information is preserved [1] | Implements a specific task, may discard or overwrite data (logically irreversible) [10] |
| 2 | Gate Types | Toffoli, Fredkin, CNOT, and reversible quantum gates [1][11] | Often optimized using irreversible classical logic and measurement-based operations [6] |
| 3 | Flexibility | General-purpose, suitable for embedding any logic reversibly | Task-specific; may not be reversible end-to-end |
| 4 | Information Loss | None — conserves information (unitary operations) [1] | Possible — uses measurement or irreversible functions [10] |
| 5 | Energy Dissipation | Theoretically zero (per Landauer's principle) [12] | Non-zero due to erasure of information [12][13] |
| 6 | Latency (Gate Time) | Single-qubit: ~10–100 ns; Two/Three-qubit (e.g., Toffoli): ~200–1000 ns [14][15] | Often faster if classical/irreversible logic or fewer gate levels used [13][16] |
| 7 | Circuit Depth | Generally deeper due to ancilla and reversibility constraints | Shallower for specific, optimized tasks |
| 8 | Gate Count (Typical) | $10^3$–$10^6$ depending on the logic encoded reversibly [11][14] | Often < $10^3$ gates tailored to task [6][16] |
| 9 | Example Applications | Quantum arithmetic, Shor's algorithm, reversible computing [1][11] | Quantum classifiers, variational circuits, hybrid algorithms [6][17] |
| 10 | Example Gates / Circuits | Toffoli, Fredkin, Peres gates; reversible adders/multipliers [11] | Variational Quantum Circuits (VQC), Quantum Approximate Optimization Algorithm (QAOA) [17] |

universal gate libraries. Representative examples include controlled-T gate optimization (T gates: 15 → 9; CNOT: 16 → 12; T-latency: 9 → 5) and 1-bit full adder compression (T: 14 → 8; CNOT: 12 → 10; H: 4 → 2; T-latency: 8 → 2). Section IV concludes the paper and summarizes main results in the design of the circuits for low latency quantum networks.

## II. LATENCY-OPTIMAL QUANTUM CIRCUITS

We survey algorithms for computing an optimal quantum circuit that implements a given unitary transformation on $n$ qubits. These algorithms offer a significant advantage over

**TABLE 3. Detailed comparison: reversible vs. application-specific/irresesible quantum circuits.**

| No. | Feature | Reversible Quantum Circuit | Application-Specific / Irreversible Quantum Circuit |
|---|---|---|---|
| 1 | Core Principle | Implements reversible logic with unitary operations (no measurement) | Implements specific tasks, may include measurement or irreversible logic |
| 2 | Gate Types | Toffoli, Fredkin, Peres, CNOT, NOT (all reversible) [11][18] | U3, RX, RZ, CRY, CNOT, MEASURE, etc. (including irreversible components) [14] |
| 3 | Gate Count (Typical) | $10^3$–$10^6$ gates depending on algorithm and ancilla overhead [19][14] | $<10^3$ for many tasks (e.g., QAOA, VQE circuits) [14][17] |
| 4 | Circuit Depth | Typically $10^2$–$10^4$ layers depending on logic complexity [14] | Typically 10–$10^3$, optimized for minimal depth in variational/hybrid models [17][16] |
| 5 | Qubit Count (Typical) | 5–100+ including ancilla qubits (for garbage cleanup) [19][14] | 4–50 qubits depending on target application [6][16] |
| 6 | Latency (Total Execution) | 10–1000 µs (deep circuits, no measurement) [19][9] | 1–200 µs (shallower, faster due to measurements and hybrid use) [17][5] |
| 7 | Gate Latency | Single-qubit: 20–100 ns; Toffoli: ~300–1000 ns on current tech [19][9] | Single-qubit: 20–100 ns; CNOT: ~150–300 ns; Measurement: ~500 ns [17][5] |
| 8 | Power Efficiency | Theoretically zero energy loss per Landauer's limit (ideal) [11][12] | Non-zero due to state collapse and garbage output [12][10] |
| 9 | Fault Tolerance Requirement | High (error propagation in long-depth reversible logic) [14] | Lower (shorter depth and hybrid tolerance) [17][5] |
| 10 | Example Circuits | - Reversible full adder (3 qubits, 10–20 gates) [18]<br>- Reversible multiplier (6–10 qubits, ~100 gates)<br>- Shor's modular exponentiation (1000+ gates) [19] | - QAOA circuit (p=1: ~60 gates, 8 qubits) [16]<br>- VQE (UCCSD ansatz: 100–200 gates, 12 qubits) [6] |
| 11 | Applications | Arithmetic logic, cryptography, quantum simulation, Shor's algorithm [19][14] | Optimization (QAOA), chemistry (VQE), ML (quantum classifiers) [6][17][5] |

brute-force methods, achieving approximately a square-root speedup in runtime.

While primarily designed to find circuits that are optimal in terms of circuit latency, they can be optimized for other criteria as well. For example, [34] describes a variant of the algorithm that minimizes the number of sequential non-Clifford gates in a circuit. It is important to note, however, that since the brute force approach has exponential complexity, this improved algorithm is still exponential in nature. As a result, its practical applicability remains limited to relatively small circuits. Over the years, significant effort has gone into synthesizing optimal circuits for classical—specifically, reversible—Boolean functions. Shende et al. [19] investigated the synthesis of 3-bit reversible logic circuits using NOT, CNOT, and Toffoli gates by constructing circuit libraries and performing iterative searches through them. Building on this, Golubitsky and Maslov [35] extended the approach to 4-bit reversible circuits composed of NOT, CNOT, Toffoli, and 4-bit Toffoli gates. Their work demonstrated highly efficient performance in circuit synthesis.

Although the lookup speeds achieved in [35] are not replicated in the algorithm discussed in [34], the authors of [34] argue that synthesizing unitary quantum circuits is inherently more computationally demanding than synthesizing reversible classical circuits, complicating direct comparisons. Nonetheless, [34] successfully incorporates many of the search strategies from [35] to enhance performance and reduce synthesis time.

Hung et al. [36] addressed a problem more closely aligned with quantum circuit synthesis by developing a method for computing optimal-cost decompositions of reversible logic into NOT, CNOT, and the quantum controlled-$\sqrt{X}$ gate. Leveraging techniques from formal verification, they derived minimal-cost quantum implementations for various logic gates, including the Toffoli, Fredkin, and Peres gates. However, their approach operates within a restricted quantum circuit model, for instance, requiring control qubits to remain strictly Boolean, and is limited to describing only a finite subset of quantum circuits on $n$ qubits, using four-valued logic.

In contrast, the work presented in [34] optimizes over the full, continuous space of quantum circuits—infinitely many possibilities—and supports arbitrary gate sets. It generates circuits not only for Boolean operations but for general quantum gates and permits optimization with respect to a variety of cost functions.

Maslov and Miller [37] also explored synthesis of 3-bit circuits over this gate set, using a pruned breadth-first search approach akin to that in [19], rather than relying on formal verification techniques.

### A. OPTIMAL QUANTUM CIRCUIT SYNTHESIS

The problem of optimal quantum circuit synthesis remains relatively underexplored, with much of the existing work focused on approximate synthesis within small state spaces. In contrast, we emphasize finding exact decompositions for

various logical gates, though the algorithm can be naturally extended to produce approximate gate sequences as well.

Dawson and Nielsen [38] introduced an algorithm for computing $\varepsilon$-approximations of single-qubit gates in time $O\left(\log^{2.71}(1/\varepsilon)\right)$, and further generalized it to multi-qubit systems. Their method provides a constructive proof of the Solovay–Kitaev theorem [39], which guarantees that any unitary can be approximated to within error $\varepsilon$ using a sequence of gates of depth logarithmic in $1/\varepsilon$. However, the resulting circuits are often far from optimal in terms of depth or gate count [40], [41].

The Dawson–Nielsen algorithm operates by recursively approximating unitaries, with the base case relying on a lookup among previously generated gates for a coarse approximation. In contrast, the algorithm presented here is designed to find minimal-depth exact circuits and could also be employed to accelerate the base case lookup step in the Solovay–Kitaev algorithm by providing a more efficient method for identifying optimal base approximations.

More closely related to the approach presented here, Fowler [42] proposes an exponential-time algorithm for finding depth-optimal $\varepsilon$-approximations of single-qubit gates. His method leverages precomputed equivalences between gate subsequences to prune the search space, effectively discarding entire families of redundant sequences from consideration. While this technique is powerful, we argue that the algorithm presented in the following sections offers superior asymptotic behavior. Moreover, the methods shown in the sequel for reducing the search space are not only more general but also demonstrably more effective in practice, especially when scaling to larger systems or more complex cost metrics

### B. DEPTH-ONE CIRCUITS

More recently, Bocharov and Svore [43] introduced a depth-optimal canonical form for single-qubit circuits over the gate set $\{ H, T \}$. Leveraging this canonical form, they achieve a significant speedup over brute-force methods by searching through precomputed databases of canonical circuits to find depth-optimal $\varepsilon$-approximations. In this regard, their work shares some similarity with the approach presented here. However, their method is limited to single-qubit circuits over the specific $\{ H, T \}$ gate set, whereas approach presented here applies to arbitrary $n$-qubit circuits and supports any gate set. Although the method incurs slower search times, it requires significantly less memory, making it more scalable in constrained environments. Furthermore, this section focuses specifically on the synthesis of small, optimal multi-qubit circuits, a problem that single-qubit synthesis algorithms such as [45, 43] are not equipped to address.

In the circuit model of quantum computation, wires represent quantum bits (qubits), and gates act to transform their state. The state of an $n$-qubit system is typically described by a vector in a $2n$ -dimensional complex Hilbert space $H$, and quantum gates correspond to linear operators on $H$.

In this context, we focus exclusively on unitary operators, that is, operators $U$ satisfying $UU^\dagger = U^\dagger U = I$, where $U^\dagger$ is the adjoint (conjugate transpose) of $U$, and $I$ is the identity operator. The overall transformation implemented by a quantum circuit is the sequential composition of its constituent gates, and since the composition of unitary operators is also unitary, it follows directly that the linear operator represented by the entire circuit is itself unitary.

An individual quantum gate typically acts non-trivially on only a subset of the qubits in a system. To represent this formally, it is convenient to express the unitary operation of the gate as a tensor product: the non-trivial unitary acting on the targeted qubits, combined with the identity operator acting on the remaining qubits. This representation not only captures the gate's localized action but also highlights the parallelism inherent in quantum circuits.

For example, consider two gates $g_1$ and $g_2$ acting on disjoint subsets of qubits. If $g_1$ is represented by $(g_1 \otimes I)$, and $g_2$ by $(I \otimes g_2)$, then their sequential composition can be rewritten as the parallel operation $g_1 \otimes g_2$, illustrating how independent gates may be applied simultaneously in a quantum circuit.

The primary optimization criterion used in this work is the depth of a circuit, which we define as the length of the longest critical path through the circuit. When a quantum circuit is represented as a directed acyclic graph (DAG)—with nodes corresponding to gates and edges representing the flow of qubit inputs and outputs—a critical path is any path from an input node to an output node that has maximum length. The depth of the circuit is thus determined by the number of sequential operations along this path (see Fig. 1).

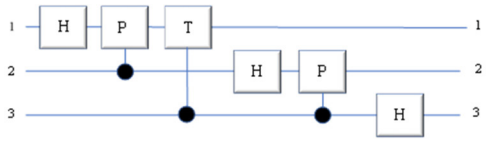



**FIGURE 1. A quantum circuit implementing the quantum Fourier transform (QFT), up to a permutation of the output qubits, is shown. This circuit has a depth of 5, with two critical paths extending from input 1 to output 3.**

The problem of quantum circuit synthesis involves constructing a circuit—composed solely of gates from a specified instruction set—that implements a given unitary transformation. This *instruction set*, denoted $\mathcal{G}$, is required to include the inverse of each of its gates to ensure completeness and reversibility. An $n$-qubit circuit over the instruction set $\mathcal{G}$ is defined as a composition of individual gates, each acting on a non-empty subset of the $n$ qubits, and tensored with the identity operator on the remaining qubits.

We can construct depth-one circuits over $n$ qubits by combining gates from a given instruction set $\mathcal{G}$, where each gate acts on a distinct subset of qubits. These depth-one circuits form a fundamental component of our synthesis algorithm. To formalize this, we define $V_{n,\mathcal{G}}$ as the set of all unitaries

corresponding to depth-one $n$ -qubit circuits constructed from gates in $\mathcal{G}$.

An $n$-qubit circuit $C$ over the instruction set $\mathcal{G}$ is said to have depth at most $m$ if it can be expressed as a sequence of unitaries: $C = U_1 U_2 \cdots U_m$,where each $U_i \in V_{n,\mathcal{G}}$. Furthermore, we say that circuit $C$ implements a unitary $U \in U\,(2^n)$ if: $U_1 U_2 \cdots U_m = U$.

In general, multiple distinct circuits can implement the same unitary transformation. While we often do not explicitly distinguish between a circuit and the unitary it realizes, it is important to note that the term "circuit" refers to a specific sequence of gates, rather than the resulting unitary operator itself.

Additionally, a key convention must be kept in mind: circuits are mathematically expressed using operator composition, meaning that unitaries are applied from right to left (i.e., the rightmost operator acts first). However, in standard circuit diagrams, gates are drawn and interpreted from left to right, with the leftmost gate acting first on the input state.

With these definitions in place, we can now state our main result. Specifically, we present an algorithm that, given an instruction set $\mathcal{G}$ and a unitary transformation $U \in U\,(2^n)$, determines whether $U$ can be implemented by a circuit over $\mathcal{G}$ of depth at most $l$. The algorithm runs in time $O\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2\rceil}\log\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2\rceil}\right)\right)$. Moreover, if such a circuit exists, the algorithm returns a circuit that implements $U$ with minimal depth over the instruction set $\mathcal{G}$.

This algorithm represents a significant improvement over the brute-force approach, which has a runtime of $O\left(|\mathcal{V}_{n,\mathcal{G}}|^{l}\right)$. In practice, with the aid of efficient data structures, it is possible to achieve running times close to $\Theta\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2\rceil}\right)$ yielding a roughly quadratic speed-up.

However, it is important to emphasize that the runtime remains exponential in $n$. This is due to the fact that for any instruction set $\mathcal{G}$ containing $k$ single-qubit gates, the size of $|\mathcal{V}_{n,\mathcal{G}}|$ grows at least as fast as $k^n$. As a result, the algorithm is only practical for synthesizing circuits over a small number of qubits. Motivated by results in fault tolerance, we use the instruction set (gate library) (*H, P, CNOT, T,* $P^\dagger$, $T^\dagger$) shown in Table 4.

The set of circuits composed from these gates forms a subset of $2^n \times 2^n$ unitary matrices over the ring $\mathbb{Z}\left[\frac{1}{\sqrt{2}}, i\right]$ defined as

$$\mathbb{Z}\left[\frac{1}{\sqrt{2}}, i\right] = \left\{\frac{a + be^{\frac{i\pi}{4}} + ce^{\frac{i\pi}{2}} + de^{\frac{i3\pi}{4}}}{\sqrt{2}^n};\ \begin{matrix} a, b, c, d, n \in \mathbb{Z} \\ n \geq 0 \end{matrix}\right\}$$

We also identify two important classes of matrices defined over this ring. The first is the Pauli group on $n$ qubits, denoted $\mathcal{P}_n$, which consists of all $n$-fold tensor products of the Pauli matrices (I, X, Y, Z), possibly multiplied by global phases $\pm 1$ or $\pm i$. Formally,

$$\mathcal{P}_n = \{e^{i\phi} \cdot \mathrm{P}_1 \otimes \mathrm{P}_2 \otimes \cdots \otimes \mathrm{P}_n | \mathrm{P_j}\} \in \{I, X, Y, Z\},$$
$$\phi \in \{0, \pi/2, \pi, 3\pi/2\}\}.$$

**TABLE 4. Gate library (*H, P, CNOT, T,* $P^\dagger$, $T^\dagger$).**

| |
|---|
| Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$, Phase gate $P = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$, |
| controlled-NOT $CNOT = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$, |
| and $T = \begin{pmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{pmatrix}$, along with $P^\dagger$ and $T^\dagger$. |

$$I = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}, X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, Y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix},$$
$$Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}, \tag{1}$$

The second important class is the Clifford group on $n$ qubits, denoted $\mathcal{C}_n$, which is defined as the normalizer of the Pauli group $\mathcal{P}_n$ in the unitary group $U\,(2^n)$. Formally,

$$2\mathcal{C}_\mathrm{n} = \left\{U \in U\left(2^\mathrm{n}\right) | U\mathcal{P}_\mathrm{n}U^{-1} \subseteq \mathcal{P}_\mathrm{n}\right\} \tag{2}$$

In other words, the Clifford group consists of those unitaries that map Pauli operators to Pauli operators under conjugation.

Circuits composed solely of the gates $X, Y, Z$, $H$(*Hadamard*), $P$ (phase), and CNOT produce unitaries in $\mathcal{C}_n$. Notably, the $T$ gate does not belong to the Clifford group.

A well-known result in quantum computation states that the Clifford group $\mathcal{C}_n$, when combined with any unitary $U \notin \mathcal{C}_n$, generates a dense subset of $U\,(2^n)$ [39]. Therefore, the gate set $\{H, P, P^\dagger, CNOT, \mathrm{T}, T^\dagger\}$ is universal for quantum computation—capable of approximating any unitary transformation to arbitrary precision, up to a global phase.

### 1) MEET-IN-THE-MIDDLE (MM) SEARCH ALGORITHM

We now provide a high-level overview of the algorithm used to compute optimal quantum circuits. Detailed descriptions of the unitary representations and their approximating circuits will be provided in the following section.

The key insight behind the algorithm lies in a simple yet powerful observation: to determine whether a unitary can be implemented with a circuit of depth $l$, it suffices to generate and analyze circuits of depth at most $\lceil l/2 \rceil$.

This crucial observation enables a significant reduction in the search space and is the foundation of what is referred to as the "meet-in-the-middle" (*mm*) algorithm [34].

*Lemma 1 ([34]):* Let $S_i \subset U\,(2^n)$ denote the set of all unitaries implementable with circuit depth $i$ over a gate set $\mathcal{G}$. Then, for a given unitary $U \in U\,(2^n)$, there exists a circuit over $\mathcal{G}$ of depth l implementing U if and only if: $S^\dagger_{\lfloor l/2 \rfloor} U \cap S_{\lceil l/2 \rceil} \neq \emptyset$.

Reference [34] applies this lemma to construct an efficient algorithm that determines whether a unitary $U$ can be implemented by a circuit over the gate set $\mathcal{G}$ with depth at

most $l$. If such a circuit exists, the algorithm also returns an implementation of U with minimum possible latency

**function** *mm*-FACTOR ($\mathcal{G}$, $U$, $l$)
  $S_0 := \{I\}$
  $i := 1$
  **for** $i \leq \lceil l/2 \rceil$ **do**
    $S_i := \mathcal{V}_{n,\mathcal{G}} S_{i-1}$
    **if** $S_{i-1}^\dagger U \cap S_i \neq \emptyset$ **then**
      **return** any circuit $VW$ s.t.
        $V \in S_{i-1}, W \in S_i, V^\dagger U = W$
    **else if** $S_i^\dagger U \cap S_i \neq \emptyset$ **then**
      **return** any circuit $VW$ s.t.
        $V, W \in S_i, V^\dagger U = W$
    **end if**
    $i := i + 1$
  **end for**
**end function**

Given an instruction set $\mathcal{G}$ and a target unitary $U$, the algorithm incrementally builds circuits of increasing depth and uses them to search for an implementation of $U$ with depth up to $2i$ (see Fig. 2). At each iteration, it generates the set $S_i$ of all circuits of depth $i$ by extending circuits from $S_{i-1}$ with an additional layer. It then computes the sets $S_{i-1}^\dagger U$ and $S_i^\dagger U$, and checks for intersections with $S_i$. According to Lemma 1, a circuit implementing $U$ exists with depth $2i-1$ or $2i$ if and only if $S_{i-1}^\dagger U \cap S_i \neq \emptyset$ or $S_i^\dagger U \cap S_i \neq \emptyset$, respectively. The algorithm halts at the smallest such depth $\leq l$ and returns a circuit implementing $U$ with minimal depth.

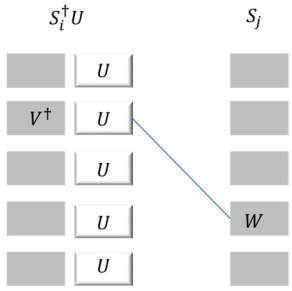



**FIGURE 2.** For each $V \in S_i$ we construct $W = V^\dagger U$ and perform a logarithmic-time search for W in $S_j$.

To get the stated runtime of $O\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2 \rceil} \log\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2 \rceil}\right)\right)$ we impose a strict lexicographic ordering on the set of unitaries — for example, by comparing matrices based on the first element at which they differ. This allows the set $S_i$ to be sorted in $O\left(|S_i| log\left(|S_i|\right)\right)$ time. Consequently, we can search $O\left(|S_{i-1}| log\left(|S_i|\right)\right)$ and $O\left(|S_i| log\left(|S_i|\right)\right)$ time, respectively. Since $|S_i| \leq |\mathcal{V}_{n,\mathcal{G}}|^i$, the total runtime for the $i$-th iteration is bounded by $2|\mathcal{V}_{n,\mathcal{G}}|^i \log\left(|\mathcal{V}_{n,\mathcal{G}}|^i\right)$ Thus, the overall runtime remains within the desired complexity bound. Since

$$\sum_{i=1}^{\lceil l/2 \rceil} |\mathcal{V}_{n,\mathcal{G}}|^i \log\left(|\mathcal{V}_{n,\mathcal{G}}|^i\right) \leq \sum_{i=1}^{\lceil l/2 \rceil} |\mathcal{V}_{n,\mathcal{G}}|^i \log\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2 \rceil}\right)$$

and

$$\sum_{i=1}^{\lceil l/2 \rceil} |\mathcal{V}_{n,\mathcal{G}}|^i \leq |\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2 \rceil} \left(1 + \frac{1}{|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2 \rceil} - 1}\right), \quad (3)$$

so we see that the algorithm runs in $O\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2 \rceil} \log\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil l/2 \rceil}\right)\right)$ time. It can also be noted that $\mathcal{V}_{n,\mathcal{G}} \in O\left(|\mathcal{G}|^n\right)$, so the runtime is in $O\left(|\mathcal{G}|^{\lceil n \cdot l/2 \rceil} \log\left(|\mathcal{G}|^{\lceil n \cdot l/2 \rceil}\right)\right)$.

### C. OPTIMIZING DIFFERENT COST FUNCTIONS

The *mm* algorithm can be extended to search for circuits optimized according to various cost functions, beyond just depth. For instance, one might define circuit cost as a weighted sum of gates, where each gate in the instruction set is assigned a specific positive weight. As long as all non-identity gates have strictly positive weights, the minimum achievable cost will increase strictly with circuit depth. This ensures that the cost of any discovered solution can serve as an upper bound on the depth that needs to be explored, keeping the runtime still primarily dependent on the depth of the optimal solution.

We concentrate on a particular family of cost functions that is increasingly relevant in fault-tolerant quantum computing. In many fault-tolerant models, gates from the Clifford group can be implemented efficiently, whereas non-Clifford gates are significantly more resource-intensive. For example, in Steane code–based schemes, non-Clifford operations are far more complex to realize [43]. More broadly, for all doubly even self-dual CSS codes—a category encompassing many widely used quantum error-correcting codes—all Clifford operations admit transversal implementations [44], making them relatively straightforward to perform.

In contrast, non-Clifford gates demand advanced methods such as ancilla state preparation and gate teleportation, greatly increasing implementation cost. This contrast is even more pronounced in surface code architectures, which offer high fault-tolerance thresholds but require particularly elaborate procedures to implement the T gate—the most common non-Clifford gate [45]. Consequently, the number of layers involving non-Clifford operations—known as the circuit's T-depth when T is the sole non-Clifford gate—often becomes the primary bottleneck in fault-tolerant quantum computation.

The *mm* algorithm naturally extends to optimize T-depth in quantum circuits. When the instruction set $G$ consists of generators for the Clifford group along with the T gate, one can first generate the complete set of Clifford unitaries. Then, by brute-force enumeration, circuits can be systematically explored in increasing T-depth. Applying the *mm* algorithm, we can efficiently search for a circuit implementing a given unitary by examining candidates with up to twice the target T-depth. This allows for effective synthesis of circuits optimized specifically for minimal T-depth, a critical metric in fault-tolerant quantum computing.

We define $\mathcal{C}_n$ as the Clifford group on $n$ qubits, implemented using gates from the instruction set $\mathcal{G}$, and let $\mathcal{T}_n$ be the set of tensor products of $I$ and $T$. To perform the mm search for minimal $T$-depth circuits, we initialize $S_0 = C_n$,

**function** *mm*-FACTOR $T$-DEPTH $(\mathcal{G}, U, l)$
  $S_0 := \{\mathcal{C}_n\}$
  $i := 1$
  **for** $i \leq l$ **do**
    $S_i := (\mathcal{C}_n \mathcal{T}_n \backslash \{I\})\, S_{i-1}$
    **if** $S_{i-1}^\dagger U \cap S_i \neq \emptyset$ **then**
      **return** any circuit $VW$ s.t.
        $V \in S_{i-1}, W \in S_i, V^\dagger U = W$
    **end if**
    **if** $S_i^\dagger U \cap S_i \neq \emptyset$ then
      **return** any circuit $VW$ s.t.
        $V \in S_i, W \in S_i, V^\dagger U = W$
    **end if**
    $i := i + 1$
  **end for**
**end function**

and recursively define $S_i = \mathcal{C}_n\,(\mathcal{T}_n \backslash \{I\})\, S_{i-1}$ so that each $S_i$ contains all circuits with $T$-depth $i$. The search proceeds by checking whether $S_{i-1}^\dagger U \cap S_i = \emptyset$ (for T-depth $2i-1$) or $S_i^\dagger U \cap S_i = \emptyset$ (for T-depth $2i$). This forms the basis for the full algorithm, summarized in the pseudocode above [42].

Searching in this manner becomes increasingly difficult as the dimensionality of the state space grows, due to the exponential growth of the Clifford group with the number of qubits. For instance, the Clifford group on three qubits already contains 92,897,280 elements (up to global phase) [46], making exhaustive search approaches computationally infeasible beyond just a few levels of circuit depth. For four qubits ($\mathcal{C}_4$), the size of the group is so large that it cannot be stored in memory on typical modern computers, rendering this method impractical without further optimization or pruning strategies

In practice, we can compute the sets $S_i$ with T-depth $\lceil i/2 \rceil$ by alternating between Clifford and T gate phases. This approach significantly reduces redundancy in the *mm* computation, as entire phases consisting solely of Clifford group operations can be omitted during the search. Given the exponential growth of the Clifford group with the number of qubits, this optimization yields substantial performance improvements over the more naive algorithm described earlier.

### D. CIRCUITS WITH ANCILLAS

Another valuable extension of the *mm* algorithm is its ability to search for circuits that utilize ancillary qubits—additional qubits initialized in the $|0\rangle$ state and required to return to $|0\rangle$ at the end of the computation. While in principle one could consider ancillas in arbitrary initial and final states, we restrict attention to the $|0\rangle$ state for simplicity. This restriction naturally includes any ancilla state that can be prepared and uncomputed using gates from the given instruction set.

More precisely, given a unitary $U \in U\,(2^n)$, we seek a unitary $U' \in U\left(2^{n+m}\right)$ such that:

$$U'\left(|0\rangle^{\otimes m} \otimes |\psi\rangle\right) = |0\rangle^{\otimes m} \otimes (U|\psi\rangle) \tag{4}$$

This guarantees that the ancilla qubits, initialized in the $|0\rangle^{\otimes m}$ state, are returned unchanged after the computation. To find such a U′, it suffices for U′ to match U on the subspace spanned by states of the form $|0\rangle^{\otimes m} \otimes \mid \psi\rangle$, i.e., for U′ to agree with U on the first $2^n$ rows and columns (those corresponding to input and output states where ancillas remain in $|0\rangle^{\otimes m}$).

However, a major challenge arises if we restrict our search to only those rows and columns. In doing so, we risk missing valid circuit decompositions. For instance, it could happen that: $V'W'\left(|0\rangle^{\otimes m}|\psi\rangle\right) = |0\rangle^{\otimes m}\,(U|\psi\rangle)$ but neither $V'$ nor $W'$ individually maps the ancillas cleanly back to $|0\rangle^{\otimes m}$ when acting alone. That is, $V'\left(|0\rangle^{\otimes m}|\psi\rangle\right) \neq |0\rangle^{\otimes m}\,(V|\psi\rangle)$ and similarly for $W'$. As a result, these intermediate steps might appear invalid when checking only partial information (i.e., the top-left submatrix), even though their composition yields the correct overall transformation. This limits the effectiveness of pruning based on subspace agreement and necessitates more sophisticated strategies to ensure such valid compositions are not discarded during the search.

Instead, we observe that since $|0\rangle^{\otimes m} \otimes (U|\psi\rangle) = (I \otimes U)\,|0\rangle^{\otimes m} \otimes |\psi\rangle$, our goal reduces to finding unitaries $V$ and $W$ such that $VW\left(|0\rangle^{\otimes m} \otimes |\psi\rangle\right) = (I \otimes U)\,|0\rangle^{\otimes m} \otimes |\psi\rangle$. This leads to the equivalent condition: $V^\dagger\,(I \otimes U)\,|0\rangle^{\otimes m} \otimes |\psi\rangle = W\left(|0\rangle^{\otimes m} \otimes |\psi\rangle\right)$.

Therefore, rather than comparing full unitaries, we can restrict our attention to how these operators act on inputs of the form $|0\rangle^{\otimes m} \otimes |\psi\rangle$. In practical terms, this means comparing only the first $2^n$ columns of the unitaries in the sets $S_{\{i-1,i\}}^\dagger\,(I \otimes U)$ and $S_i$, as these columns correspond to the action on the relevant input subspace.

However, this relaxation comes at a cost. Since we're no longer working with full unitaries, we lose the ability to eliminate redundant circuit permutations as efficiently as in the original *mm* algorithm. Consequently, more candidate circuits must be generated and compared, increasing both memory and computational overhead during the search. This trade-off is necessary to accommodate ancillary qubits but should be carefully managed in practical implementations.

### E. SEARCH TREE PRUNING

To further reduce the size of the search space, we apply search tree pruning, which in practice yields significant improvements in both memory usage and runtime. The search tree here refers to the conceptual structure in which each node represents a partial circuit, and each branch represents the application of an additional gate from the set $\mathcal{V}_{n,\mathcal{G}}$.

Each level $S_\lambda$ in this tree corresponds to all circuits of depth $\lambda$, and the *mm* algorithm generates this tree in a breadth-first manner. Without pruning, the size of the tree grows rapidly, as each circuit in $S_{\lambda-1}$ spawns many child circuits by appending a gate, leading to an exponential blowup.

Pruning works by identifying and eliminating circuits that are redundant or equivalent under some symmetry or canonical form. For instance, if two partial circuits yield the same resulting unitary (up to global phase or circuit equivalence), we can safely discard one without affecting the completeness of the search.

By keeping only the lexicographically minimal representative of each equivalence class (as determined by a strict ordering on unitaries), we dramatically shrink the number of unique paths that must be explored. This reduction not only decreases the total memory footprint, as fewer circuits must be stored, but also accelerates the search process by avoiding repeated computations on functionally identical subcircuits.

We introduce an equivalence relation "$\sim$" on $n$-qubit unitaries by declaring $U \sim V$ exactly when $V$ can be obtained from $U$ by permuting qubit labels, taking the inverse, or multiplying by an overall phase. Each unitary $U \in U(2^n)$ thus falls into a class $[U] = \{V \in U(2^n) \,|\, U \sim V\}$. Rather than storing every possible circuit, we select a single, depth-optimal implementation for one distinguished (canonical) member of each class. Whenever a new circuit is generated, we compute its canonical representative and check whether we have already recorded a minimal-depth realization for that representative—discarding duplicates and thereby pruning the search tree.

We impose a total order on $U(2^n)$ by comparing their matrix entries lexicographically, and call the smallest element in each equivalence class its canonical representative. Concretely, for any $n$-qubit unitary $U$ we generate all transforms arising from ($i$) permuting qubit labels—which corresponds to simultaneous row-and-column permutations of $U$'s matrix—and ($ii$) taking the inverse (i.e. the conjugate transpose). There are $2 \times n!$ such candidates, and by scanning them we pick the lexicographically minimal matrix in $O(n!)$ time. In practice, this extra cost per unitary is negligible for small $n$, since the time spent lex-ordering is dwarfed by the effort of querying the circuit database.

Selecting a canonical representative up to global phase requires one extra step. When our gate set is restricted to unitaries over the ring $\mathbb{Z}\left[1/\sqrt{2}, i\right]$, the only admissible overall phases are $e^{i\theta} \in \mathbb{Z}\left[1/\sqrt{2}, i\right] \iff \theta = k\pi/4, k \in \mathbb{Z}$ so there are exactly eight distinct phase factors [41]. Thus, for an $n$-qubit unitary U we first form all variants obtained by (1) permuting the qubit labels (simultaneous row-and-column permutations), (2) inverting via conjugate transpose, and (3) applying each of the eight allowed phases. In total there are $8 \times 2n!$ candidates, and we simply pick the lexicographically smallest matrix among them as the canonical representative.

However, explicitly generating all eight candidate phases is costly. Instead, we fix the global phase by choosing a single "reference" entry in the matrix and rotating the entire unitary so that this entry becomes real and positive. Concretely, we scan the $2^n \times 2^n$ matrix in row-major order to find its first nonzero element $u_{ij}$ then multiply U by $e^{-iarg(u_{ij})}$. This single normalization step uniquely removes the global phase without enumerating all eight possibilities.

Let $re^{i\theta}$ be the first nonzero entry of U. Naïvely, one might normalize by multiplying U by $e^{-i\theta}$, since then any equivalent $V = e^{i\varphi}U$ satisfies $e^{-i(\theta+\varphi)}V = e^{-i\theta}U$—but if $\theta$ is not a multiple of $\pi/4$, then $e^{-i\theta}$ falls outside $\mathbb{Z}\left[1/\sqrt{2}, i\right]$. To stay within the ring, we instead use the factor $re^{-i\theta}$, which always lies in $\mathbb{Z}\left[1/\sqrt{2}, i\right]$. Thus, we define the canonical representative of U to be $(re^{-i\theta})$.

Under this rule, any global-phase equivalent $V = e^{i\varphi}U$ has first entry $re^{i(\theta+\varphi)}$, and applying the same factor yields $(re^{-i(\theta+\varphi)})\ \mathrm{V} = (re^{-i\theta})\ \mathrm{U}$. Because $re^{-i\theta}$ lives in the ring, all comparisons can be carried out symbolically over $Z\left[1/\sqrt{2}, i\right]$, yielding both greater precision and faster performance.

Given a newly generated circuit $C$ of depth $i$ that implements a unitary $U$, the canonical representative of the equivalence class $[U]$ is first computed. The database of previously discovered circuits is then searched to check whether a circuit implementing this representative already exists. With appropriate data structures, each set $S_j$ of circuits of depth $j$, where $1 \le j \le i$, can be searched in $O\left(\log\left(|S_j|\right)\right)$ time. If no matching circuit is found, a new entry is stored for the circuit implementing the representative of $[U]$. Notably, given a circuit $C$ that implements $U$, one can obtain a circuit for the canonical representative by applying the corresponding permutation and/or inversion to $C$.

Permutations are applied by reassigning the qubits on which the individual gates in a circuit act. For inverses, we use the fact that $C^{-1} = C^\dagger$; thus, if $C = U_1 \cdots U_m$, then $C^{-1} = U_m^\dagger \cdots U_1^\dagger$. Since the instruction set is defined such that every gate has an inverse within the set, a circuit for $C^{-1}$ can be constructed by reversing the order of gates in $C$ and replacing each gate with its inverse. As a result, both permutations and inverses of a unitary can be implemented without increasing the circuit depth. This implies that all unitaries within same equivalence class share the same minimum circuit latency.

A subtle but important point is that if we search using only the representatives of equivalence classes at depths $i$ and $j$, we may fail to discover all equivalence classes at depth $i + j$. Consider a unitary $U = VW$, where $V$ is a circuit of depth $i$ and $W$ is of depth $j$. Let $V'$ and $W'$ denote the canonical representatives of $[V\ ]$ and $[W]$, respectively. In general, $(V')^\dagger VW \notin [W]$, meaning that searching only via class representatives does not guarantee discovery of $[U]$. However, since $V^\dagger \in \left[V'\right]$, it follows that $W \in \left[V'\right]VW$, and thus $[V']\ U = \left[W'\right]$.

In practice, this implies that any unitary $U = VW$ can be found by computing the canonical representative of $[[V]\,U]$. Therefore, even though intermediate class representatives may not suffice for all compositions, we can still discover all circuits of minimum depth by storing only the representatives of their equivalence classes.

In certain scenarios, an exact implementation of a unitary with the correct global phase is necessary—particularly when the circuit may be used as part of a controlled operation on another qubit. While one could define canonical representatives based solely on qubit relabeling and inversion, it is still possible to construct the correct global phase using any canonical implementation over the gate set $\mathcal{G} = \{H, P, P^\dagger, CNOT, T, T^\dagger\}$, provided the phase is implementable over $G$. As observed in [41], if a unitary $U$ is implementable by a circuit over $\mathcal{G}$, and a circuit $C$ over $G$ implements $e^{i\theta}U$, then $\theta = k\pi/4$ for some $k \in \mathbb{Z}$. Since $\left(HP^\dagger\right)^3 = e^{-i\pi/4}I$, we have $e^{i\theta}\left(HP^\dagger\right)^{3k}U = U$. Therefore, an exact circuit for $U$, with the correct global phase, can be obtained by composing $C$ with $\left(HP^\dagger\right)^{3k}$, yielding a valid implementation over $\mathcal{G}$.

### F. IMPLEMENTATION ASPECTS

As previously noted, the *mm* algorithm offers no advantage over the naive approach unless appropriate data structures are used. Without them, searching for collisions between the sets $S_i^\dagger U$ and $S_j$ would require $O\left(|S_i||S_j|\right)$ comparisons.

However, by imposing a lexicographic ordering on the generated circuits, searches can be performed in logarithmic time relative to the size of $S_j$. In the implementation, this ordering is realized by storing each $S_i$ as a red-black tree—a type of balanced binary search tree. Balanced trees are widely used in standard libraries and industrial databases for ordered sets and maps due to their reliable performance and scalability. Notably, since deletions—often the most computationally intensive operation in such trees [47]—are never required in the mm algorithm, red-black trees are a particularly well-suited choice for this context.

While hash tables could, in principle, offer performance improvements, an adaptation of the *mm* algorithm in [34] using the hash table implementation from *libstdc++* showed no measurable speedup. This held true even with a very low rate of key collisions—for example, among 1,316,882 distinct 3-qubit unitaries, no more than 52 were mapped to the same hash value. Despite this favorable collision rate, the hash-based approach did not outperform the red-black tree implementation in practice.

While storing unitaries directly enables efficient generation of new unitaries and rapid searching through circuit databases, the associated memory requirements make large-scale searches infeasible on machines with typical RAM capacities. For an instruction set $\mathcal{G}$ acting on $n$ qubits, the number of distinct unitaries $|\mathcal{V}_{n,\mathcal{G}}|$ grows at least as fast as $k^n$, where $k$ is the number of single-qubit gates in $\mathcal{G}$. Using the standard universal instruction set $\mathcal{G} = \{H, P, P^\dagger, CNOT, T, T^\dagger\}$ —which includes generators for the Clifford group along with the T gate—we find $|\mathcal{V}_{3,\mathcal{G}}| = 252$. Consequently, at depth 5, the number of possible circuits exceeds $10^{12}$.

If each 3-qubit unitary over $\mathbb{Z}\left[1/\sqrt{2}, i\right]$ is stored exactly, it requires $5 \times 64$ integers. Thus, storing all depth-5 circuits on 3 qubits would demand more than 1 petabyte of storage, rendering brute-force approaches impractical. In practice, storing only the representatives of equivalence classes significantly reduces memory usage. For instance, on 3 qubits, there are at most 36,042,958 unique equivalence classes with circuits of depth up to 5, as shown in the experiments reported in [34]. Additionally, the storage required for unitaries over $\mathbb{Z}\left[1/\sqrt{2}, i\right]$ can be further reduced through compression techniques. Nevertheless, even with these optimizations, storing the full unitary matrices becomes infeasible for deeper searches. As a result, a space-time trade-off is unavoidable, requiring careful balance between memory usage and computational cost.

To manage this space-time trade-off, one approach is to store the circuit itself rather than the generated unitary. Specifically, each circuit can be represented as a sequence of depth-1 layers, where each layer is encoded using $n$ bytes—one per qubit—indicating which gate is applied. This compact representation dramatically reduces memory usage. However, storing only the circuit structure introduces a new challenge: searching for a specific unitary becomes computationally expensive. Each comparison would require recomputing the unitary implemented by the circuit, significantly increasing the search time.

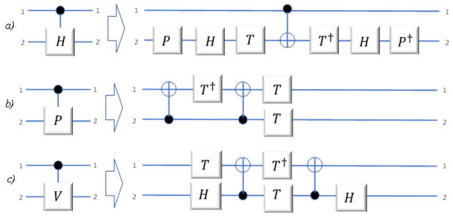



**FIGURE 3.** Logical gate of controlled unitaries without ancillas (a) Controlled-H (T-depth 2, total depth 7), (b) Controlled-P (T-depth 2, total depth 4), and (c) Controlled-$\sqrt{X}$ (T-depth 2, total depth 5).

As a compromise between storage efficiency and search performance, an m × m matrix M is stored as a key alongside each circuit. For a circuit $C$ implementing a unitary $U$, the matrix is defined by $M(i,j) = v_i^\dagger U v_j$, where the $\{v_i\}$ are vectors in $\mathbb{C}^{2^n}$ generated pseudo randomly. In practice, even using $m = 1$ suffices to search to meaningful depths for up to 4 qubits, with an extremely low rate of key collisions.

Since these keys are computed using floating-point arithmetic, it is crucial that all other operations—such as circuit evaluation—are performed symbolically. This ensures that unitaries which are mathematically equal produce identical keys despite any numerical error. Alternative experiments in [34] explored using random vectors over $\mathbb{Z}\left[1/\sqrt{2}, i\right]$ to eliminate floating-point computation entirely, but this approach resulted in an impractically high number of collisions. Ongoing work is focused on developing improved

methods for generating effective and collision-resistant unitary keys.

To further enhance performance during circuit searching, the implementation in [34] parallelizes the search process using the POSIX pthreads library in C. This allows concurrent traversal of balanced binary trees across multiple threads, significantly speeding up lookups. Additionally, generated circuit databases are serialized and stored to disk, eliminating the need to recompute them for each search and enabling efficient reuse across runs.

**TABLE 5. Performance for depth-optimal circuit search [19].**

| # qubits \ depth | | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 2 | database size (circuits) | 14 | 104 | 901 | 6,180 | 37,878 | 197,388 |
| | RAM (KB) | 2.092 | 16.686 | 146.701 | 1,013.358 | 6,249.708 | 32,766.246 |
| | generation time (s) | 0.001 | 0.015 | 0.155 | 1.354 | 10.761 | 75.301 |
| | search time (s) | 0.001 | 0.004 | 0.033 | 0.248 | 1.672 | 9.321 |
| 3 | database size (circuits) | 36 | 1,110 | 41,338 | 1,316,882 | 36,042,958 | - |
| | RAM (KB) | 5.633 | 179.657 | 6,737.931 | 215,968.485 | 7,738,582.749 | - |
| | generation time (s) | 0.012 | 1.059 | 40.619 | 1896.301 | 73,295.675 | - |
| | search time (s) | 0.015 | 0.350 | 12.619 | 414.722 | 11,759.390 | - |
| 4 | database size (circuits) | 84 | 9,984 | 1,755,677 | - | - | - |
| | RAM (KB) | 13.460 | 1,617.082 | 284,596.043 | - | - | - |
| | generation time (s) | 0.570 | 122.966 | 18,728.922 | - | - | - |
| | search time (s) | 0.603 | 71.420 | 12,853.887 | - | - | - |

*Depth-optimal implementations:* In [34], experiments were conducted to determine depth-optimal decompositions of various 2-, 3-, and 4-qubit logical gates using the gate set $\{H, P, P^\dagger, CNOT, T, T^\dagger\}$. The primary objective was to minimize circuit depth, with gate count serving as a secondary optimization criterion. Table 5 presents performance metrics for several representative gates based on this analysis.

Searches for minimal-depth implementations of various 2-, 3-, and 4-qubit logical operations were performed in [34] using precomputed circuit databases. These searches included depth-optimal decompositions of singly controlled versions of gates such as $P$ and $V = \sqrt{X} = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}$ (see Fig. 3). For completeness, depth-optimal implementations of the controlled-Z and controlled-Y gates were also computed and are shown in Fig. 4.

The unitary in Fig. 5 is also optimally decomposed and has found application in a notable quantum algorithm [48] that achieves exponential speedup via a quantum walk. Additional 3-qubit unitaries with minimal-depth implementations include several well-known gates: the Toffoli gate (controlled-controlled-NOT), the Fredkin gate (controlled-SWAP), the quantum OR gate—defined as the unitary mapping $|a\rangle|b\rangle|c\rangle \rightarrow |a\rangle|b\rangle|c \oplus a \vee b\rangle$)—and the Peres gate [21] (see Fig. 6). Notably, the mm algorithm reduces the total depth of the Toffoli gate from 12 [39] to 8, demonstrating a significant improvement in depth-optimal synthesis.

Searches were also conducted in [34] for each of the aforementioned n-qubit gates using up to 4−n ancilla qubits. These searches extended to the maximum circuit depth allowed for the total number of qubits, as summarized in Table 5. While none of the logical gates tested were found to admit decompositions with shorter overall depth or fewer T-gates, alternative circuits for the controlled-P and controlled-$\sqrt{X}$ gates were discovered with reduced T- *latency* (see Fig. 7). Moreover,

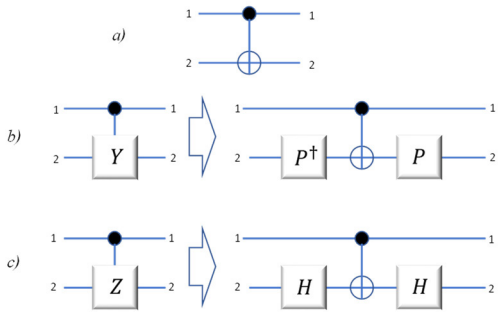



**FIGURE 4. Controlled Paulis (a) Controlled X (depth 1), (b) Controlled Y (depth 3), (c) Controlled Z (depth 3). The T-latency of all these circuits is equal to 0.**

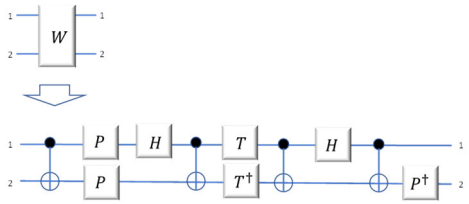



**FIGURE 5. W gate (T-depth 1, total latency 9).**

a specific unitary was found to achieve a lower minimal depth when decomposed using an ancilla (Fig. 8). Together, these results highlight the potential of ancilla-assisted decompositions to improve circuit execution time, particularly through reductions in T-depth and total *latency*.

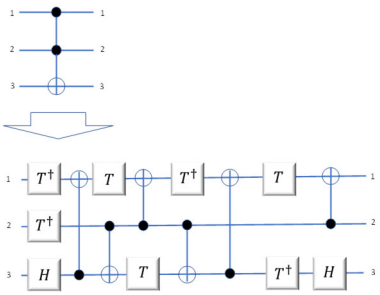
Additional gates were also investigated, including the 3-qubit quantum Fourier transform, which was proven to have no implementation within the instruction set at a depth of 10 or less. Similarly, no depth-optimal circuits of depth 6 or less were found for the 4-qubit Toffoli gate (controlled-Toffoli) or the 1-bit full adder. Furthermore, both the controlled-T and controlled-$\sqrt[4]{X}$ gates were shown to lack implementations of depth 10 or less and 6 or less, respectively, even when using one or two ancillas. These results underscore the limitations of the instruction set for certain complex operations, particularly at low depths or with minimal ancillary resources.

The authors of [34] also applied the *mm* algorithm to optimize known circuits, including implementations of the controlled-T gate and a 1-bit full adder. For the controlled-T gate, they constructed a circuit based on the decomposition $FREDKIN\ (I \otimes I \otimes T)\ FREDKIN$ while the full adder circuit was derived from the construction in [50], replacing the Peres gate with the more efficient version shown in Fig. 6d.

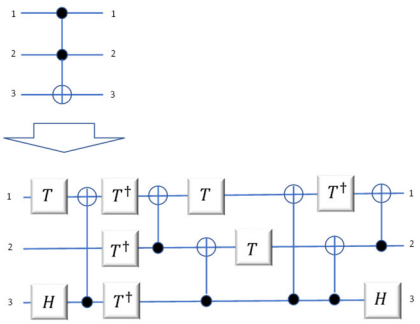
They then applied peephole optimization, a technique where small subcircuits are examined and replaced with more efficient versions synthesized using the *mm* algorithm. This approach yielded substantial improvements:

*Controlled-T gate (Fig. 9):* T gates reduced from 15 → 9, CNOT gates reduced from 16 → 12,T- *latency* reduced from 9 → 5

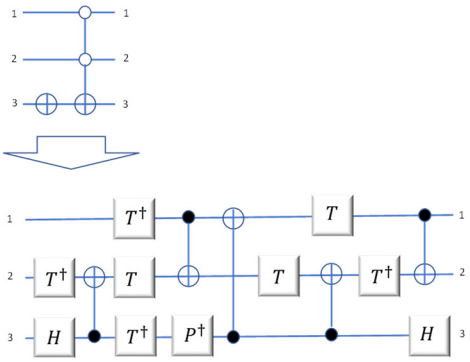
*1-bit full adder (Fig. 10):* T gates reduced from 14 → 8, CNOT gates reduced from 12 → 10, H gates reduced from 4 → 2, T- *latency* reduced from 8 → 2



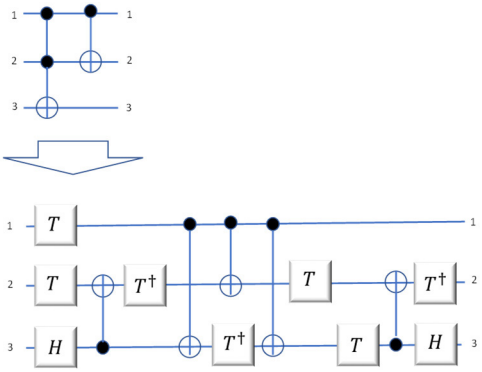
*(a) Toffoli gate (T-depth 4, total latency 8)*



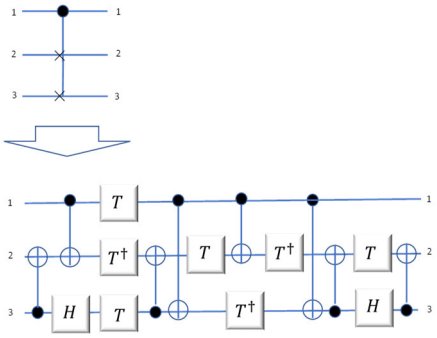
*(b) Toffoli gate, one negative control (T-latency 4, total latency 8)*



*(c) Quantum OR gate (T-depth 4, total latency 8)*



*(d) Peres gate (T-depth 4, total latency 8)*



*(e) Fredkin gate (T-depth 4, total latency 10)*

**FIGURE 6.** **3-qubit logical gates with no ancillas.**

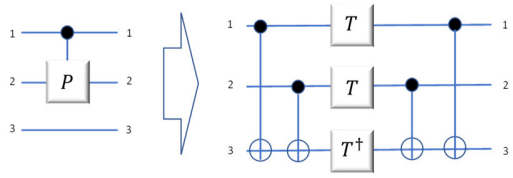
These results demonstrate the practical effectiveness of peephole resynthesis as a powerful tool for full-scale quantum circuit optimization.

For 2-qubit circuits, the Clifford group comprises 11,520 unique elements (up to global phase), which could be generated in approximately 1 second. Searching for a given unitary up to 1 T-stage, or 2 T-stages ending in a non-Clifford operation, required less than 2 seconds in total. This efficiency was sufficient to identify minimum T-depth implementations for the 2-qubit gates studied.

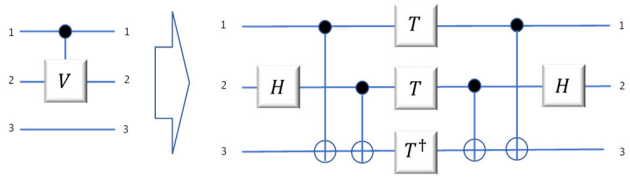
In stark contrast, generating the 92,897,280 unique 3-qubit Clifford group elements took nearly 4 days of computation, illustrating the exponential growth in complexity and the computational bottleneck this presents for scaling to higher qubit counts.

The minimal T-depth implementation of the controlled-H gate (Fig. 11) was computed in under one second following the generation of the Clifford group. For other 2-qubit logical gates, however, minimal T-depth circuits did not reduce the number of T-stages beyond those found in the corresponding minimal depth implementations. Consequently, the circuits for controlled-P, controlled-$\sqrt{X}$, and W are optimal with respect to both overall circuit depth and T-depth.

These findings highlight a key insight: using ancillas can strictly reduce the minimal T-depth required for certain




*(a) Controlled-P (T- latency 1, total latency 5)*




*(b) Controlled -$\sqrt{X}$ (T- latency 1, total latency 5).*

**FIGURE 7.** Reduced T- latency implementations utilizing ancillas. Note that qubit 3 is initialized in and returned to state $|0\rangle$.

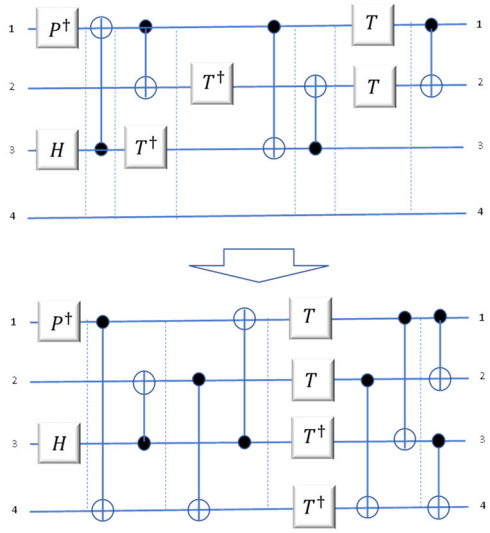



**FIGURE 8.** Addition of one ancilla (qubit 4), initialized and returned in state $|0i\rangle$, reduces the minimum circuit latency from 7 (left) to 6 (right).

unitaries. In particular, circuits for controlled-P and $\sqrt{X}$ that incorporate ancillas were found with lower T-depth than their ancilla-free counterparts, emphasizing the utility of ancillas in optimizing quantum circuits for execution time.

Although a provably minimal T-depth implementation of the Toffoli gate using zero ancillas has not yet been found, a circuit with T-depth 3 (Fig. 13) was discovered using the main *mm* algorithm. Given that the Toffoli gate is believed to require at least 7 T gates, it was conjectured that this implementation achieves minimal T-depth.

This circuit represents a significant improvement over the previously known implementation with T-depth 5 [39], offering an approximate 40% speed-up in fault-tolerant quantum architectures, where Clifford gates are considered inexpensive relative to T gates.

It's important to note that although these circuits are maximally parallelized, achieving a T-depth of $\lceil m/n \rceil$ for *mm* T-gates and $n$ qubits, not all circuits can be compressed to this

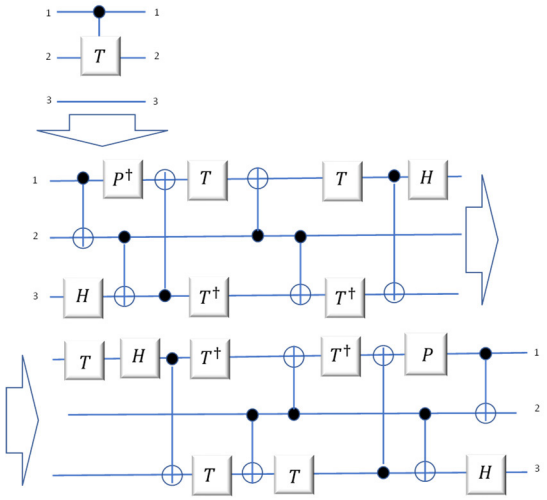



**FIGURE 9.** Circuit implementing a controlled-T gate ($T-\rangle$ latency 5, total latency 19). Note that qubit 3 is initialized in and returned to state $|0\rangle$.

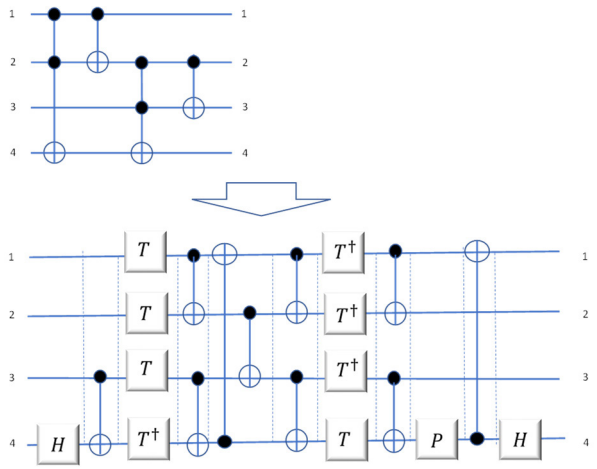



**FIGURE 10.** Circuit implementing a reversible 1-bit full adder.

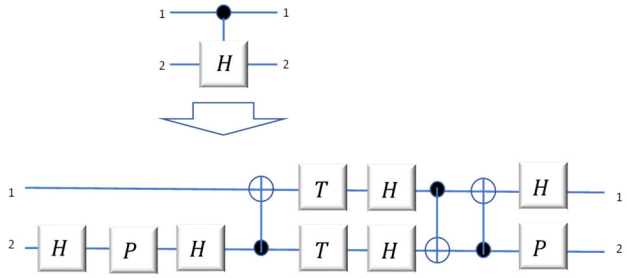



**FIGURE 11.** Circuit implementing a controlled-H gate (T- latency 1, total latency 9).

ideal. Some unitaries inherently resist such parallelization, underscoring the complexity of circuit optimization.

### G. EXACT DECOMPOSITION OF CONTROLLED UNITARIES

It is well known that a controlled version of any quantum circuit can be constructed by replacing each gate in the circuit with its controlled counterpart, using the same control qubit throughout [51]. The minimal-depth circuits derived in the previous section enable us to formalize the following result.

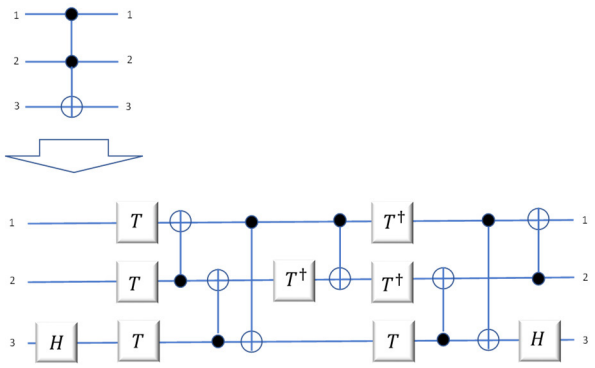



**FIGURE 12.** Circuit implementing a Toffoli gate (T- latency 3, total latency 9).

*Theorem 1 [51].* Let the gate cost of a quantum circuit be represented by the $\boldsymbol{x} = [x_H, x_P, x_C, x_T]^t$ where: $x_H$ is the number of Hadamard ($H$) gates, $x_P$ is the number of phase (P) or inverse phase ($P^\dagger$) gates, $x_C$ is the number of CNOT gates, and $x_T$ is the number of T or inverse T ($T^\dagger$) gates. Suppose a unitary operation $U$ can be implemented within error $\varepsilon \geq 0$ using a circuit over the gate set $G = \{H, P, P^\dagger, CNOT, T, T^\dagger\}$, with gate cost vector x. Then, the controlled version of U, denoted controlled- $U$, can be implemented to within error at most $\varepsilon$ using a circuit over the same gate set $G$, with gate cost given by $A$x, where A is a constant matrix.

$$A = \begin{bmatrix} 2 & 0 & 2 & 4 \\ 2 & 0 & 0 & 2 \\ 1 & 2 & 6 & 12 \\ 2 & 3 & 7 & 9 \end{bmatrix}$$

The controlled- $U$ circuit requires exactly one ancilla qubit if the decomposition of UU includes one or more $T$ - or $T^\dagger$-gates, and no ancilla qubits otherwise. Moreover, the controlled- $U$ can be implemented with a $T$ -depth of at most $x_H + 2x_P + 3x_C + 5x_T$

*T gate parallelization:* As previously noted, the three T gates in the controlled-P and controlled-$\sqrt{X}$ circuits (Figs. 3b and 3c) can be parallelized to achieve a T-depth of 1 using a single ancilla qubit (Fig. 6). Likewise, the seven T gates in the Toffoli gate decomposition (Fig. 12) can also be parallelized to T-depth 1 with the use of four ancilla qubits. In both cases, the parallelized T gates are interleaved with networks of CNOT gates. A general theorem relating the number of T gates in a {CNOT, T} circuit to the minimum achievable T-depth, given a specific number of ancilla qubits, was established in [10].

*Theorem 2 [10].* Any circuit on $n$ qubits over the gate set $\{CNOT, T\}$, containing $k$ T gates, can be implemented using a circuit over $\{CNOT, T\}$ on $n$ qubits and $m$ ancilla qubits (each initialized and returned to the $|0\rangle$ state), such that the T-depth of the resulting circuit is at most $\lceil \frac{k}{m+1} \rceil$.

We now proceed to prove *Theorem 1*, as originally presented in [34].

Suppose $U$ is implementable by a circuit over the gate set {CNOT,T} using $k$ T gates. Then, from *Theorem 1*, we can write $U|a_1 a_2 \ldots a_n\rangle = \omega^t |g(a_1, a_2, \ldots, a_n)\rangle$ where $t = f_1(a_1, \ldots, a_n) + f_2(a_1, \ldots, a_n) + \cdots + f_k(a_1, \ldots, a_n)$ with each $f_i$ a linear Boolean function, and $g$ a linear reversible function.

Assume $k \leq m$, and define a unitary operation F acting on n+m qubits by: $f|a_1 \cdots a_n\rangle|b_1 \cdots b_m\rangle = |a_1 \cdots a_n\rangle|c_1 \cdots c_k\rangle|b_{k+1} \cdots b_m\rangle$, where each $c_i = b_i \oplus f_i(a_1, a_2, \ldots, a_n)$

Since each $f_i$ is linear and the transformation only applies CNOT-like operations to copy $f_i$ into ancilla bits, the map F is linear and reversible. In fact, $F = F^{-1}$, so F is self-inverse and can be implemented using only CNOT gates.

Now, consider the unitary operator

$$V = I^{\otimes n} \otimes T^{\otimes k} \otimes I^{\otimes m-k}, \tag{5}$$

which applies the T gate to the first k ancilla qubits (those holding $c_1, \ldots, c_k$) and identity elsewhere.

$$f^{-1} V f |a_1 a_2 \cdots a_n\rangle|0\rangle^{\otimes m} = \omega^t |a_1 a_2 \cdots a_n\rangle|0\rangle^{\otimes m}. \tag{6}$$

Since $g$ is a linear reversible function, $U$ can thus be implemented by a circuit over $\{CNOT, T\}$ in $T$-depth $1 = \lceil \frac{k}{m+1} \rceil$.

Now suppose $k > m$. As before, there exists a linear reversible function $f$ implemented by a circuit over $\{CNOT\}$ such that

$$f|a_1 a_2 \cdots a_n\rangle|b_1 b_2 \cdots b_m\rangle = |a_1 a_2 \cdots a_n\rangle|c_1 c_2 \cdots c_m\rangle, \tag{7}$$

where $c_i = b_i \oplus f_i(a_1, a_2, \ldots, a_n)$. Additionally, $f_{m+1}$ is an output of some linear reversible function $h$, so the first $m+1$ factors of $\omega$ can be computed in $T$-depth 1 by implementing the unitary $f^{-1} h^{-1} V h f$, where $V$ is a tensor product of $I$ and $m+1T$ gates.

As a result,

$$\begin{aligned} &\left(U \otimes I^{\otimes m}\right) |a_1 a_2 \cdots a_n\rangle|0\rangle^{\otimes n} \\ &= \left(U' \otimes I^{\otimes m}\right) f^{-1} h^{-1} V h f |a_1 a_2 \cdots a_n\rangle|0\rangle^{\otimes n}, \end{aligned}$$

where $U'|a_1 a_2 \ldots a_n\rangle = \omega^{t'} |g(a_1, a_2, .., a_n)\rangle$ and $t' = f_{m+2}(a_1, \ldots, a_n) + \cdots + f_k(a_1 \ldots, a_n)$. By Lemma 2 $U'$ can be implemented by a circuit over $\{CNOT, T\}$ with $k - (m+1)$ $T$ gates, and thus $U$ can be implemented in $T$-depth at most $\lceil \frac{k}{m+1} \rceil$ by induction. □

In general, it is often possible to achieve a T-depth lower than $\lceil \frac{k}{m+1} \rceil$, since multiple $f_i$ functions can frequently be computed reversibly into data qubits simultaneously. Specifically, when there are $l$ linearly independent Boolean functions to be computed, it is possible to use $l$ data qubits in parallel.

The minimal achievable T-depth for a circuit with $n$ data qubits and $m$ ancilla qubits is determined by the smallest number of groups in a partition $\{S_1, S_2, \ldots, S_l\}$ of the set $\{f_1, f_2, \ldots, f_k\}$, such that each group $S_i$ satisfies $|S_i| \leq m + \dim(\text{ span } S_i)$ This condition ensures that all functions in each group can be computed simultaneously using the available qubits (data + ancilla), enabling more aggressive T-parallelization than the bound given by $\left\lceil \frac{k}{m+1} \right\rceil$.

As an example of T-parallelization, [34] demonstrates that the Toffoli gate decomposition, originally implemented with T-depth 3 (Fig. 12), can be rewritten to achieve T-depth 2 by introducing a single ancilla qubit (Fig. 14). Similarly, the implementation of the controlled-T gate (Fig. 9), which has T-depth 3, can be reduced to T-depth 1 with the use of one ancilla (Fig. 13). While the circuit is not explicitly shown, it is also noted that the 1-bit full adder circuit presented in Fig. 10 can be rewritten to T-depth 1 by employing 4 ancilla qubits.

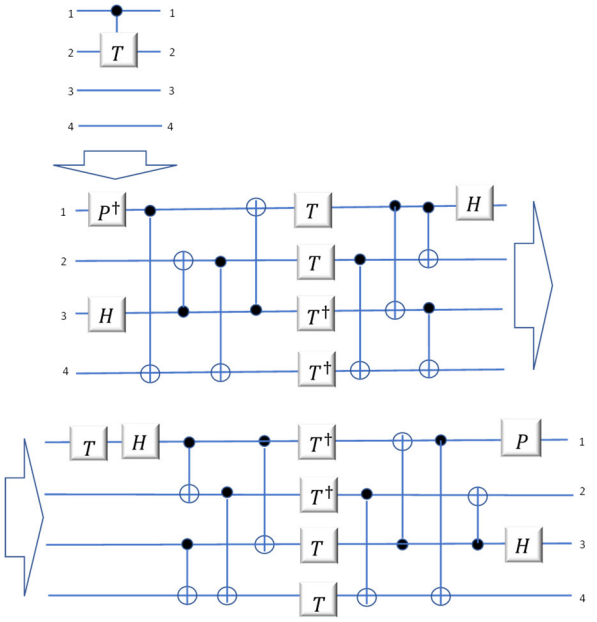


**FIGURE 13.** T- latency 3 implementation of the controlled-T gate.

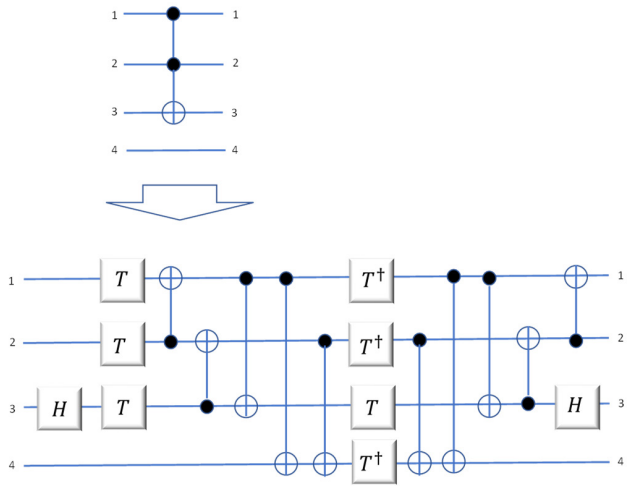


**FIGURE 14.** T - latency 2 implementation of the Toffoli gate.

### H. SUMMARY

We summarize the main results of the section and acknowledge the relevant aspects of the quantum circuit design that should be delegated to the hardware design.

First, this section develops the framework for designing latency-optimal quantum circuits and explores how implementation techniques can minimize the circuit depth—the principal determinant of latency in quantum operations. The section introduces the latency compression coefficient, defined as the ratio of a circuit's depth before and after optimization, as a normalized indicator of latency improvement. The literature survey emphasizes that latency minimization is achieved primarily through parallelization, gate-level compression, and reconfiguration of circuit topology while preserving logical equivalence. The section also presents several examples demonstrating significant reductions in both gate count and depth for common quantum operations such as controlled-T and full-adder circuits. These results provide quantitative evidence that intelligent restructuring of quantum operations can yield substantial latency improvements without necessarily increasing energy cost or qubit count.

*i) Critical Evaluation of Design Strategies*

*Gate-Level and Topological Optimization:* The section identifies the work on gate-level optimization, especially for T and CNOT gates, as a primary approach to reduce latency. This is supported by practical examples that show meaningful compression factors (up to 4× for T-latency). The methodology aligns with recent progress in quantum compilers, which exploit circuit commutativity and gate merging to achieve similar effects. *Effective Current Practices*: The explicit quantification of compression factors provides clear, reproducible metrics for performance improvement. The section grounds its claims in measurable parameters (T-count, T-depth), which are widely accepted benchmarks in quantum design. *Limitations and delegation to the hardware design:* Due to the targeted focus on networking aspects, and lack of the coverage in the literature, the discussion omits layout-aware constraints such as qubit connectivity graphs, SWAP overhead, and error propagation delays, all of which directly affect achievable latency in hardware implementations. No systematic analysis is given for how parallelism saturates when constrained by physical hardware topologies.

*ii) Latency Compression Coefficient as a Design Metric*

The introduction of the latency compression coefficient is an insightful contribution. It allows comparative evaluation of the work on circuit improvements independent of absolute gate numbers. *Effective Current Practices:* The metric formalizes latency improvement and allows normalization across circuit types and optimization levels. It provides a conceptual bridge between quantum circuit theory and IoT system requirements (where normalized latency improvement is more relevant than raw gate counts). *Limitations and delegation to the hardware design:* The coefficient ignores error-correction overheads, measurement delays, and classical feedback time, which are non-negligible in realistic quantum processors. There is no discussion of how latency compression interacts with fidelity degradation or decoherence constraints.

*iii) Implementation Aspects and Parallelism*

This section highlights the work where that latency can be minimized through parallel execution of commuting gates

and reorganization of control dependencies. This perspective aligns well with modular design strategies for near-term quantum devices (NISQ systems). *Effective Current Practices:* The section recognizes the critical role of parallelization and dependency analysis in latency optimization. It connects the notion of circuit restructuring to architectural cost models, albeit implicitly. *Limitations and delegation to the hardware design:* Due to the targeted focus on networking aspects, and lack of the coverage in the literature, the section does not include quantitative comparison between different quantum technologies (e.g., superconducting vs. trapped-ion vs. photonic systems), where the effectiveness of parallelization differs due to varied gate synchronization and communication latency. No experimental or simulation-based validation is presented to illustrate how these design principles translate to actual latency reduction on physical hardware.

*iv) Trade-Offs and Design Limitations*

The surveyed work in this section implicitly acknowledges that aggressive latency reduction can restrict the use of universal and reversible gates, thus impacting programmability and energy efficiency. However, the analysis remains qualitative. *Needed extensions: a*) Explicit models linking latency–energy and latency–fidelity trade-offs. *b*) Exploration of hybrid reversible–irreversible architectures, which may retain low dissipation while improving latency for control-intensive tasks. *c*) A clearer mapping between algorithmic compression and network-level latency improvements in IoT applications.

*v) Comparative Evaluation*

| Design Aspect | Strengths (Current Work) | Limitations / Missing Elements |
|---|---|---|
| **Gate Optimization** | Demonstrated measurable reductions in T and CNOT gate counts and depths. | No consideration of hardware constraints or inter-qubit communication overhead. |
| **Latency Compression Coefficient** | Introduces a normalized, generalizable latency metric. | Does not include system-level latency components (measurement, feedback). |
| **Parallelism & Implementation** | Recognizes circuit restructuring and gate concurrency as key techniques. | Lacks benchmarking across hardware platforms and gate synchronization models. |
| **Energy and Programmability Trade-Offs** | Highlights the inverse relationship between universality/reversibility and latency. | Needs quantitative energy–latency trade-off modeling. |
| **IoT Integration** | Conceptually aligns with low-latency IoT requirements. | Missing integration with network-level latency models and edge-device constraints. |

*vi) Recommendations for Future Work*

*1. Hardware-Aware Circuit Optimization:* Incorporate qubit connectivity, SWAP costs, and hardware timing models into latency calculations to achieve physically realistic results. 2. *Error-Correction-Inclusive Latency Models:* Extend the latency compression coefficient to account for T-gate distillation, syndrome extraction, and measurement delays. 3. *Cross-Layer Optimization for IoT:* Explore co-design frameworks linking quantum circuit latency to network and protocol latency in quantum-enabled IoT systems. 4. *Benchmarking and Comparative Evaluation:* Develop standardized benchmarks across superconducting, ion-trap, and photonic hardware to quantify achievable latency compression under real constraints. 5. *Hybrid Logic Architectures:* Investigate hybrid reversible–irreversible circuits to balance latency, energy efficiency, and programmability in embedded quantum IoT nodes.

*vii) Overall Assessment*

The section presents a coherent and theoretically rigorous foundation for survey of the work on latency-optimal quantum circuit design. Its introduction of the latency compression coefficient and its systematic gate-level analysis provide valuable tools for researchers. Nonetheless, when surveying the existing work, the section's primary limitation lies in its lack of discussion of hardware and system-level integration, which is delegated to the hardware research—it remains algorithmic and theoretical rather than empirical or architectural. To advance this field, future studies should bridge the gap between quantum circuit theory and IoT-specific deployment realities, emphasizing end-to-end latency modeling that encompasses both computation and communication layers. When coupled with empirical validation and hardware benchmarking, such integration will transform the current conceptual framework into a practical design methodology for real-time quantum-enabled IoT systems.

## III. EXACT MINIMIZATION OF QUANTUM CIRCUITS

Since non-permutative quantum gates (NPQGs)—such as the controlled square-root-of-NOT gate ($CV = CV^{\dagger}$)—follow more complex transformation rules than permutative quantum gates (PQGs), synthesizing them is significantly more challenging. An $n$-letter permutation can be represented as an $n \times n$ binary permutation matrix, where each row and column contains exactly one entry equal to 1, with all other entries being 0. Among these, so-called ''magic'' permutation matrices are distinguished by having a 1 on the main diagonal, indicating that the identity component of the transformation is preserved. Some of the most well-known examples of permutation matrices correspond to familiar quantum gates, such as the Pauli gates.

$$X = \begin{pmatrix} 01 \\ 10 \end{pmatrix} \equiv (2,1)\,, I \otimes X \equiv (2,1)\,(4,3)\,,$$

$$\text{CNOT} = \begin{pmatrix} 1000 \\ 0100 \\ 0001 \\ 0010 \end{pmatrix} \equiv (1,2)\,(4,3)\,,$$

$$\text{CCNOT} \equiv (1,2,3,4,5,6)\,(8,7)\,,$$

These permutation gates may act on one, two, or even three qubits, depending on the specific operation. Moreover, permutation gates can also be generalized to qudits (quantum systems with $d>2$ levels). An example is the shift gate, which cyclically permutes the computational basis states of a qudit. Specifically, the shift gate maps $|j\rangle \mapsto |(j+1)\text{modd}\rangle$ acting as a modular increment operation on the state index.

$$X = \begin{pmatrix} 010 \\ 001 \\ 100 \end{pmatrix} \equiv (2,3,1)$$

In efficient synthesis algorithms, the direct use of non-permutative quantum gates (NPQGs) should generally be avoided. Instead, the key strategy is to construct new permutative quantum gates (PQGs) using quantum gates in such a way that NPQGs are effectively replaced by equivalent PQG-based structures. This approach relies on a quantum gate library composed of primitives optimized for minimal quantum cost.

Following the methodology outlined in [53], we begin by introducing a set of novel gates similar to the controlled square-root-of-NOT gate ($CV = CV^{\dagger}$), specifically the controlled k-th root of NOT gates for $k = 2,4,8,\ldots$ We provide the corresponding unitary matrices for each of these gates.

Furthermore, we present a general and efficient method for directly constructing an optimal quantum logic gate library using combinations of CNOT gates and these NPQGs. This technique also introduces new ways to derive PQGs that exhibit reduced quantum cost, enabling more efficient circuit synthesis overall.

As previously discussed, reversible logic synthesis plays a vital role in quantum information processing. A small number of elementary quantum gates, each with carefully optimized quantum cost, can be combined to construct arbitrary quantum logic circuits. Reducing the quantum cost of a circuit not only enhances its speed and efficiency but also minimizes the likelihood of errors—an essential consideration in practical quantum computation.

However, the synthesis of reversible (permutative quantum) circuits using quantum gates differs significantly from classical (non-reversible) logic synthesis. In addition to the concepts introduced in Section II, a number of researchers have explored methods for synthesizing quantum circuits with exact minimal cost, focusing on gate sets that are inexpensive to implement physically.

One notable example is the Peres gate, which is often preferred over the standard Toffoli gate due to its lower quantum cost and simpler realization. This illustrates the importance of selecting gate primitives not only for their functionality but also for their implementation efficiency in real quantum systems

Improvements in quantum logic gate design play a critical role in reducing both circuit complexity and execution time. At the heart of this effort lies the challenge of constructing the most cost-efficient equivalents of logic gates like the Peres gate, using only elementary quantum gates. These elementary gates include NOT, CNOT, controlled square-root-of-NOT (also called controlled-NOT$^{1/2}$ or controlled- V/$V^{\dagger}$), and even more fine-grained gates such as the controlled fourth-root-of-NOT (controlled-NOT$^{1/4}$ or controlled- $W$ /$W^{\dagger}$).

A foundational study by Barenco et al [54] provided the first comprehensive approach to realizing multiple-control Toffoli (MCT) gates using such elementary gates. This was further advanced by a method in [55] for systematically determining elementary gate realizations of MCT gates.

Subsequently, [56] introduced two general analytic expressions for simulating $n$-qubit controlled-U gates using standard single-qubit gates and CNOTs, with exponential and polynomial complexity, respectively. These formulations were accompanied by explicit circuit constructions and general decomposition strategies.

Further innovation came with the introduction of TISC (2)-interval symmetric controlled) quantum permutative gates in [57], and the gate library was extended in [58] to include fourth-root-of-NOT gates, broadening the scope for fine-tuned quantum circuit synthesis.

Due to the exponential growth in memory requirements and run-time complexity, only a limited number of existing methods [59], [63] are capable of optimally synthesizing 3-qubit circuits using the NCV quantum gate library, which includes NOT, CNOT, and controlled square-root-of-NOT (i.e., controlled-NOT$^{1/2}$) gates.

A key breakthrough in this area involves reducing NCV circuit synthesis to four-valued logic, which simplifies the process of identifying optimal gate sequences. Notably, [63] introduced an approach that constructs a new quantum gate library using only NCV gates yet is functionally equivalent to the original NCV library in terms of synthesizing all optimal 3-qubit circuits.

This reformulation allows the synthesis problem to be reduced from four-valued logic back to binary logic, which is significantly easier to implement and reason about. As a result, it opens up the possibility for more scalable and efficient synthesis of small-scale quantum circuits using elementary gate primitives.

In this work, we present an efficient 3-qubit synthesis algorithm based on a perfect hash function, capable of rapidly constructing all optimal 3-qubit circuits. Remarkably, the average synthesis speed for circuits with minimum quantum cost is approximately 127 times faster than the best previous method reported in [34].

As part of this approach, we first introduce a series of new gates similar to CV/CV$^{\dagger}$-namely, the controlled k-th root of NOT gates for k = 2,4,8,…—and provide the corresponding unitary matrices for each. These gates expand the

set of non-permutative quantum gates (NPQGs) available for synthesis.

More importantly, we present a novel and generic method for constructing an optimal quantum logic gate library using only CNOT gates and these newly introduced NPQGs. This method enables efficient synthesis by systematically exploring low-cost compositions.

Experiments conducted with this new gate set—as well as supporting results from [53]—demonstrate significant improvements in synthesis performance. Collectively, these findings introduce a new paradigm for discovering additional permutative quantum gates (PQGs) with lower quantum cost, paving the way for more efficient quantum circuit design.

### A. PRELIMINARIES

We already know that the operation of each gate in an $n$-line reversible or quantum circuit can be represented by a square matrix of dimension $2^n$. The matrix of the NOT gate is $N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, and $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ represents the identity circuit. From [58], we also get

$$N^{1/k} = \frac{1}{2}\begin{pmatrix} 1+i^{2/k} & 1-i^{2/k} \\ 1-i^{2/k} & 1+i^{2/k} \end{pmatrix}; \tag{8}$$

where $k$ is a power of 2. $N^{1/k}$ is a $k$-th root of $N$; thus $\left(N^{1/k}\right)^k = N$. As a preliminary to this section, we use a number of propositions suggested in [53], staring with (8).

Let $G_k = N^{1/k}; k = 2^n$, and $n \in \{0, 1, 2, 3...\}$, then $G_{2^n} = N^{2^{-n}}$, and $G_{2^n}^\dagger$ is the adjoint of the $G_{2^n}$ matrix.

*Proposition 1 ([53]):*

$$G_{2^n} = \frac{1}{2}\begin{pmatrix} 1+e^{i\pi/2^n} & 1-e^{i\pi/2^n} \\ 1-e^{i\pi/2^n} & 1+e^{i\pi/2^n} \end{pmatrix};$$

and its adjoint matrix is

$$G_{2^n}^\dagger = \frac{1}{2}\begin{pmatrix} 1+e^{-i\pi/2^n} & 1-e^{-i\pi/2^n} \\ 1-e^{-i\pi/2^n} & 1+e^{-i\pi/2^n} \end{pmatrix}$$

*Proposition 2 ([53]):*

$$\left(G_{2^{n+1}}\right)^2 = G_{2^n};$$
$$G_{2^n} G_{2^n}^\dagger = I$$
$$\left(G_{2^{n+1}}^\dagger\right)^2 = G_{2^n}^\dagger$$
$$(G_{2^n})^{2^n} = N;$$
$$\left(G_{2^n}^\dagger\right)^{2^n} = N$$

*Proposition 3 ([53]):* Let $G_{2^n,m} = (G_{2^n})^m$, where $m \in \{1, 3, 5..., 2^n - 1\}$; then

$$\left(G_{2^n,m}\right)^{2^n} = \left(G_{2^n,m}^\dagger\right)^{2^n} = N \tag{9}$$

*Proposition 4 ([53]):*

$$\left(G_{2^{n+1},k}\right)^2 = \begin{cases} G_{2^n,k} & k \in \{1, 3, 5, ..., 2^n - 1\} \\ N \cdot G_{2^n,k-2^n} & k \in \left\{2^n + 1, 2^n + 3, ..., 2^{n+1} - 1\right\} \end{cases} \tag{10}$$

*Definition 1:* The controlled square-root-of-NOT gate, also referred to as the controlled-NOT$^{1/2}$ gate, includes both the controlled- $V$ ($CV$) gate and the controlled-V$^\dagger$ (CV$^\dagger$) gate. Their corresponding unitary matrices are derived and illustrated in Fig. 15

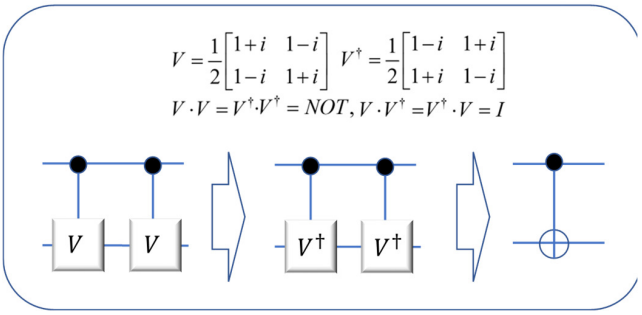



**FIGURE 15.** Basic quantum algebra rules for $CV = CV^\dagger$ gates.

*Definition 2:* The controlled fourth-root-of-NOT gate, also known as the controlled-NOT$^{1/4}$ gate, consists of the controlled- $W$ ($CW$) gate and its adjoint, the controlled-W$^\dagger$(C W$^\dagger$) gate. Here, the gates W and W$^\dagger$ are formally defined as the square root of the square root of the NOT gate, i.e., W = NOT$^{1/4}$ and W$^\dagger$ = ( NOT$^{1/4}$)$^\dagger$. As shown in Equation (b), the tautological transformations associated with these gates are illustrated in Fig. 16.

*Definition 3:* The controlled eighth-root-of-NOT gate, or controlled- NOT$^{1/8}$ gate, includes the controlled $J$ ($CJ$) gate and its adjoint, the controlled-$J^\dagger$ ($CJ^\dagger$) gate.

These gates are defined by (10), and their corresponding unitary transformations are derived accordingly.

$$G_8 = G_{8,1} = \frac{1}{2}\begin{pmatrix} 1+e^{i\pi/8} & 1-e^{i\pi/8} \\ 1-e^{i\pi/8} & 1+e^{i\pi/8} \end{pmatrix};$$
$$G_8^\dagger = G_{8,1}^\dagger = \frac{1}{2}\begin{pmatrix} 1+e^{-i\pi/8} & 1-e^{-i\pi/8} \\ 1-e^{-i\pi/8} & 1+e^{-i\pi/8} \end{pmatrix}; \tag{11}$$

from (10), we get $\left\{G_{8,1}, G_{8,3}, G_{8,5}, G_{8,7}, G_{8,1}^\dagger, G_{8,3}^\dagger, G_{8,5}^\dagger, G_{8,7}^\dagger\right\} \subseteq \left\{N^{1/8}\right\}$, from Eq. (11), we have

$$\left(G_{8,1}\right)^2 = G_{4,1}, \left(G_{8,3}\right)^2 = G_{4,3},$$
$$\left(G_{8,5}\right)^2 = N \cdot G_{4,1}, \left(G_{8,7}\right)^2 = N \cdot G_{4,3}$$

and from Eq. (10), we deduce that

$$\left(G_{8,1}\right)^4 = \left(G_{4,1}\right)^2 = V$$
$$\left(G_{8,3}\right)^4 = \left(G_{4,3}\right)^2 = N \cdot V = V^\dagger$$
$$\left(G_{8,5}\right)^4 = \left(N \cdot G_{4,1}\right)^2 = V$$
$$\left(G_{8,7}\right)^4 = \left(N \cdot G_{4,3}\right)^2 = N \cdot V = V^\dagger$$

so we define

$$J = G_{8,1}, J^\dagger = G_{8,1}^\dagger, or\ J = G_{8,3}^\dagger, J^\dagger = G_{8.3} \tag{12}$$

or $J = G_{8,5}, J^{\dagger} = G_{8,5}^{\dagger}$ or $J = G_{8,7}^{\dagger}, J^{\dagger} = G_{8,7}$ where $J^8 = \left(J^{\dagger}\right)^8 = \text{NOT}$; $J^4 = V$ and $\left(J^{\dagger}\right)^4 = V^{\dagger}$

In [53], two fundamental methods were proposed for synthesizing optimal reversible logic circuits using quantum gates: a) Direct synthesis using quantum gates to construct the circuit. b) A two-stage synthesis process, where: First, a set of new logic gates is designed using quantum gates. Then, these newly defined logic gates are used to synthesize the target circuit.

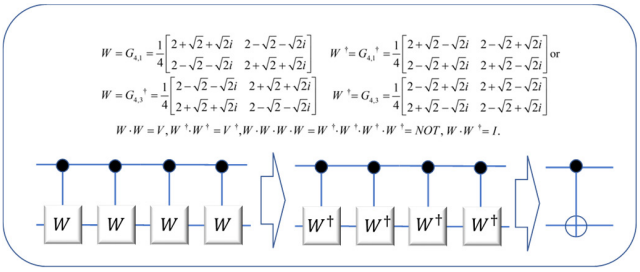



**FIGURE 16.** Basic quantum algebra rules for $CW = CW^{\dagger}$ gates.

Between the two, the second method is significantly more efficient from an algorithmic standpoint. This is because it reduces the problem to a binary logic synthesis task, which is computationally simpler. Additionally, the newly defined logic gates need to be synthesized only once, after which they can be reused across multiple circuits.

All of these new logic gates are systematically constructed from elementary quantum gates, ensuring that the resulting circuits remain compatible with standard quantum hardware primitives.

*Assumption 1:* The generated logic gate satisfies the following conditions:

(p1) It is a reversible Boolean logic gate, and all control lines are restricted to Boolean values.

(p2) It is indivisible; that is, it cannot be broken down into simpler logic gates. Specifically, if the gate consists of a sequence of elementary gates, the first and last gates on each qubit must not be logic gates.

(p3) The target level of any controlled gates must not include the target of a CNOT gate.

(p4) The quantum cost of the gate cannot be minimized through any form of equivalence transformation.

(p5) Control lines are strictly limited to Boolean values.

*Assumption 2:* The logic gate library satisfies the following criteria:

(p1) Optimality: Every gate in the library is optimal in terms of quantum cost.

(p2) Minimality: No gate in the library can be constructed from two or more other gates within the same library without increasing the quantum cost. Naturally, it also cannot be composed at a lower cost, since all gates are already optimal.

(p3) Equivalence: Using gates from the new library yields the same optimal circuit (with the same quantum cost) as using gates from the original library.

The approach exclusively utilizes CNOT gates and controlled- $\text{NOT}^{1/k}$ gates, where k = 2,4,8,... for constructing logic gates.

Let us now consider functions implemented by nn-qubit circuits composed solely of CNOT gates.

*Theorem* 3.1 [53]. For any Boolean set of input variables $x_1, x_2, \ldots, x_n$ applied to an n-qubit circuit composed exclusively of CNOT gates, the output of the circuit must be of the form: $x_1^{(j_1)} \oplus x_2^{(j_2)} \oplus \ldots \oplus x_n^{(j_n)}$, where $n \geq 2, j = (j_n \cdots j_2, j_1)_2$ is the binary representation of $j, 0 < j \leq 2^n - 1$, and

$$x_i^{(j_i)} = \begin{cases} 0 \ \textit{if} \ j_i = 0 \\ x_i \ \textit{if} \ j_i = 1 \end{cases}$$

This is called a linear circuit.

### B. PERMUTATIVE GATE LIBRARY

*Logic gate construction using CNOT, $C-NOT^{1/k}$ gates, k = 2, 4, 8, ...*: Let us denote arithmetic addition by + and Sigma and Boolean logic exclusive addition by $\oplus$.

*Theorem 3* [53]. For any set of Boolean variables $x_1, x_2, \ldots, x_n$, we have

$$m(x_1, x_2, \ldots, x_n) = \sum_{j=0}^{2^n-1} \left(x_1^{(j_1)} \oplus x_2^{(j_2)} \oplus \ldots \oplus x_n^{(j_n)}\right) = \begin{cases} 0 \ \textit{if} \ x_i = 0, i \in \{1, 2, \ldots, n\} \\ 2^{n-1} \ \textit{else} \end{cases}$$

where $n \geq 2, j = (j_n \cdots j_2, j_1)_2$ is the binary representation of $j$, and

$$x_i^{(j_i)} = \begin{cases} 0, \ \textit{if} \ j_i = 0 \\ x_i, \ \textit{if} \ j_i = 1 \end{cases}$$

*DESIGN EXAMPLE 1*: We aim to construct an $(n+1)$-qubit logic gate. The input Boolean variables are denoted as $x_0, X_n$, where $x_0$ is the input on the bottom line, and $X_n = \{x_1, x_2, \ldots, x_n\}$ are the control line inputs.

Suppose we use $2^n - 1$ controlled- $NOT^{1/2^{n-1}}$ gates, each with its target on the bottom line, and each controlled by a distinct value in the set:

$$\xi(X_n, J_n), = \{x_1, x_1 \oplus x_2, x_1 \oplus x_2 \oplus x_3, \ldots, x_1 \oplus x_2 \oplus \cdots \oplus x_n\},$$

where each control value corresponds to a binary index $(j_1 j_2 \cdots j_n)_2$ such that $1 \leq (j_1 j_2 \cdots j_n)_2 \leq 2^n - 1$.

According to *Theorem 2*, for any input $X_n$, the function $m(X_n)$, the number of active control signals (i.e., the number of control values evaluating to 1), must be either 0 or $2^n - 1$. This implies that:

a) If zero control signals are active, then none of the $C-NOT^{1/2^{n-1}}$ gates are applied, and the output on the bottom line is simply $x_0$; that is, the identity function.

If exactly $2^{n-1}$ control signals are active, then $2^{n-1}$ gates are applied in sequence on the bottom line. Since each gate

has a rotation angle of $\pi/2^{n-1}$, cascading all of them results in a full NOT gate (i.e., a rotation of $\pi$), effectively flipping the value of $x_0$.

Therefore, depending on the control inputs, the gate either performs the identity function or a NOT operation on the bottom line. As a result, the entire configuration defines a valid logic gate.

Thus, the newly constructed gate behaves as a reversible Boolean logic gate with a conditional NOT operation controlled by $X_n$. So in the new gate, we have

$$k = 2^{n-1}, \text{ i.e., } n = \log_2 k + 1. \tag{13}$$

All circuits derived by Barenco [54] using the $V$ and $W$ gates can be seen as special cases under this new method. In this context, we define circuit types based on their gate composition:

A circuit constructed using $CV/CV^\dagger$ and CNOT, gates is referred to as a CV-type circuit.

A circuit constructed using $CW/CW^\dagger$, and CNOT gates is referred to as a CW-type circuit.

These classifications help generalize and unify various existing constructions within the framework of our proposed approach.

If we aim to construct a new logic gate using only CNOT and $CG_k/CG_k^\dagger$ gates, where $k = 2,4,8,\ldots$ we observe the following: the targets of the $CG_k/CG_k^\dagger$ gates are the $G_k/G_k^\dagger$ gates themselves, which are not logic gates individually. However, a sequence of $k$ such gates (either all $G_k$ or all $G_k^\dagger$) can be cascaded to form a valid logic gate.

To construct an (n+1)-bit gate under this framework, all targets of the $G_k/CG_k^\dagger$ gates must be placed on a single line, which we designate as the bottom line for descriptive convenience. The remaining $n$ lines are denoted as $Xn = \{x_1,x_2,\ldots,x_n\}$ and serve as the control lines.

Now, consider placing $2^n - 1 - nCG_k/CG_k^\dagger$ gates, each controlled by a unique Boolean function from the set:

$$\xi(X_n, J_n)|\forall j \in \left\{1, \ldots, 2^n - 1\right\},$$

where $k = 2^{n-1}$. According to *Theorem 3*, for any input configuration of $X_n$, the number of control values evaluating to 1 (i.e., the number of active control signals) must be either 0 or $2^{n-1}$.

As a result: If 0 control values are active, none of the $CG_k/CG_k^\dagger$ gates are applied, and the bottom line performs the identity function. If $2^{n-1}$ control values are active, then exactly $2^{n-1}G_k$ (or $G_k^\dagger$) gates are applied in sequence on the bottom line, together forming a valid logic gate. Therefore, this configuration ensures that the constructed gate behaves as a reversible logic gate, conditionally applying a transformation to the bottom line based on the collective input state of the control lines.

*Assumption 3:* For a gate to qualify as a logic gate, one of the following conditions must be satisfied for any input configuration:

(p1) All of the selected gates are $CG_k$ gates.

(p2) All of the selected gates are $CG_k^\dagger$ gates.

(p3) The selected gates include an equal number of $CG_k$ and $CG_k^\dagger$ gates.

The control values of the $2^n - 1 - nCG_k/CG_k^\dagger$ gates are given by the set:

$$X_n^* = x_1 \oplus x_2, x_1 \oplus x_2 \oplus x_3, \ldots, x_1 \oplus x_2 \oplus \ldots \oplus x_n$$

each representing a unique Boolean combination of the input variables. This portion of the circuit—responsible for generating these control values—can be constructed using a suitable configuration of CNOT gates. The remaining $n$ control values used by the other gates are the original input variables $X_n = \{x_1,x_2,\ldots,x_n\}$, which are directly sourced from the inputs of the new gate.

As in [53], we introduce a framework for constructing new $(n + 1)$- bit logic gates using CNOT and $CG_k/CG_k^\dagger$ gates, illustrated in Fig. 17. For clarity, we focus on $(n+1)$- bit gates rather than $n$-bit gates. In Fig. 17, the set of control values is defined as: $\{T_n\} = \{t_{n+1}, t_{n+2}, \ldots, t_{2^n-1}\} = \{X_n^*\}$ where $X_n^*$ represents a specific set of XOR combinations of the input variables $X_n$.

However, it is important to note that not all logic circuits can be realized using the configuration shown in Fig. 17. To address this limitation, the framework is generalized and extended in Fig. 18. The new framework in Fig. 18 provides a more comprehensive structure, with the framework in Fig. 17 now considered a special case of this more general model.

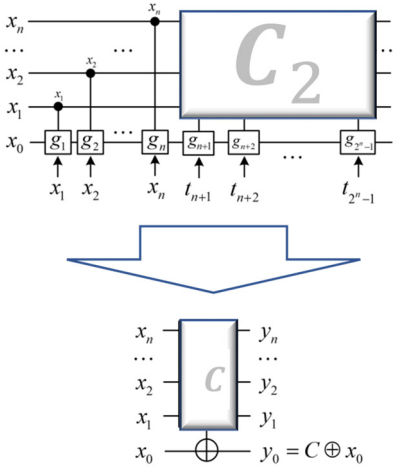



**FIGURE 17.** The framework of new $(n+1)$ -qubit logic gate implementation using $CG_k/CG_k^\dagger$ and CNOT gates.

*Theorem 4*. In Fig. 17, the new gate is defined with $T_n = X_n^*$. Then, it must hold that $t_1 \in \{X_n\}$.

*Proof.* Assume, for contradiction, that $t_1 \notin \{X_n\}$. Then it follows that $t_1 \in \left\{X_n^*\right\}$, meaning that $t_1$ is not a direct input variable but rather a Boolean function (specifically, an XOR combination) of the input variables in $X_n$. To obtain $t_1$, at least one CNOT gate must precede the gate $g_1$ in order to compute the required control value. This implies that the portion of

the circuit before $g_1$ (which generates $t_1$) constitutes a logic circuit. The part of the circuit after $g_1$, starting from $g_1$ itself, is also a logic circuit. However, according to property (p2) of *Assumption 2*, a logic gate cannot be composed of smaller logic circuits. This contradicts our assumption. Therefore, the assumption $t_1 \notin \{X_n\}$ must be false $\Rightarrow t_1 \in X_n$ □

In Fig. 18, there are multiple possible implementations of the subcircuit $C_2$, each of which can be synthesized exclusively using CNOT gates. We now present a specific method for constructing $C_2$ that generates the required $2^n - 1 - n$ control values used by the corresponding $2^n - 1 - nCG_k$ or $CG_k^\dagger$ gates. Clearly, since each of these control values corresponds to a unique Boolean XOR combination of the input variables $X_n$, the construction of $C_2$ requires at least $2^n - 1 - n$ CNOT gates. This raises a fundamental question: How can we synthesize the circuit $C_2$ using the minimum number of CNOT gates The challenge lies in finding an efficient configuration that produces all necessary XOR combinations without redundancy, thereby minimizing quantum cost and circuit latency.

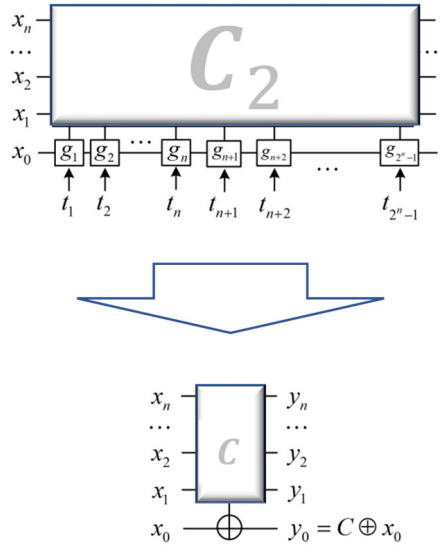



**FIGURE 18.** The generic framework for Fig. 17.

In Fig. 17, there are multiple possible implementations of the subcircuit $C_2$ , each of which can be synthesized exclusively using CNOT gates. We now present *a* specific method for constructing $C_2$ that generates the required $2^n - 1 - n$ control values used by the corresponding $2^n - 1 - nCG_k$ or $CG_k^\dagger$ gates.

Clearly, since each of these control values corresponds to a unique Boolean XOR combination of the input variables $X_n$, the construction of $C_2$ requires at least $2^n - 1 - n$ CNOT gates. This raises a fundamental question: How can we synthesize the circuit $C_2$ using the minimum number of CNOT gates The challenge lies in finding an efficient configuration that produces all necessary XOR combinations without redundancy, thereby minimizing quantum cost and circuit latency. This approach fundamentally differs from the conventional synthesis method, which relies solely on function permutations. In contrast, the current method involves generating $2^n - 1 - n$ control values using the operator $C_2$, corresponding to the same number of $CG_n$ *or* $CG_n^\dagger$ gates. As a result, the synthesis process appears more complex. However, we demonstrate that systematic rule exists for constructing $C_2$. Based on this rule, [53] presents examples of typical 4-bit logic gates implemented with $CG_4/CG_4^\dagger$ gates (denoted as $CW/CW^\dagger$), as well as 5-bit logic gates using $CG_8/CG_8^\dagger$ gates (denoted as $CJ/CJ^\dagger$).

*DESIGN EXAMPLE 1: Gates construction using CNOT,* $C - NOT^{1/4}(CW/CW^\dagger)$ *Library*

The 4-bit logic gates using $CG_4/CG_4^\dagger$ and CNOT gates are given in Figs. 19 and 20. From Eq. (13), $n = (\log_2 k + 1)\,|_{k=4} = 2 + 1 = 3$, so the new gate has $(n+1)\,|_{n=3} = 4$ lines.

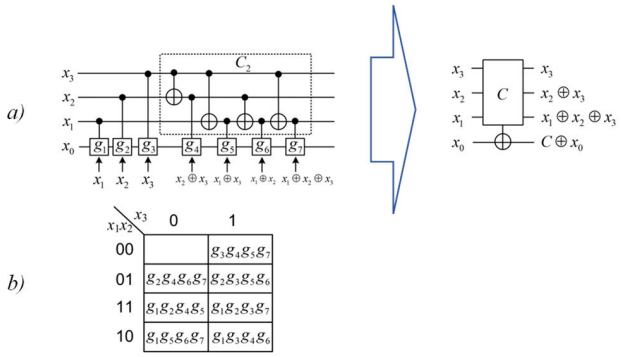



**FIGURE 19.** a) Framework for implementing a novel 4-qubit logic gate using $CG_4 = CG_4^\dagger$ and CNOT gates b) Symbolic graphical representation of the circuit from a) using a quantum Karnaugh map (QMap).

In the framework from Fig. 19 and Fig. 20, $g_i \in \left\{G_4, G_4^\dagger\right\}, i \in \{1, 2, 3, 4\}$, and there are $2^4$ combinations, but in Table 6, there are only two combinations satisfy the stringent conditions outlined in *Assumption* 2.

**TABLE 6.** New 4-bit logic gates in Fig 19.

| No. | $g_1$ | $g_2$ | $g_3$ | $g_4$ | $g_5$ | $g_6$ | $g_7$ | $x_1x_2x_3$ | | | | | | | | $C$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 | |
| 1 | $W$ | $W^\dagger$ | $W^\dagger$ | $W$ | $W^\dagger$ | $W^\dagger$ | $W$ | $I$ | $I$ | $I$ | $N$ | $I$ | $I$ | $I$ | $I$ | $\bar{x}_1x_2x_3$ |
| 2 | $W$ | $W$ | $W$ | $W^\dagger$ | $W^\dagger$ | $W^\dagger$ | $W$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $N$ | $x_1x_2x_3$ |

The structure of 5-bit logic gates using $CG_8/CG_8^\dagger$ and CNOT gates is given in Fig. 21. One $C_2$ circuit implementation using minimum CNOT gates (i.e., only $(2^n - 1 - n)|_{n=4} = 11$ CNOT gates) is given in Fig. 22.

*DESIGN EXAMPLE 2: Gate construction using CNOT and* $C\text{-}NOT^{1/8}$ $(CJ/CJ^\dagger)$

Let $J = G_8$ and $J^\dagger = G_8^\dagger$. Then in Eq. (12) we see that $J^8 = \left(J^\dagger\right)^8 = NOT$. In the framework from

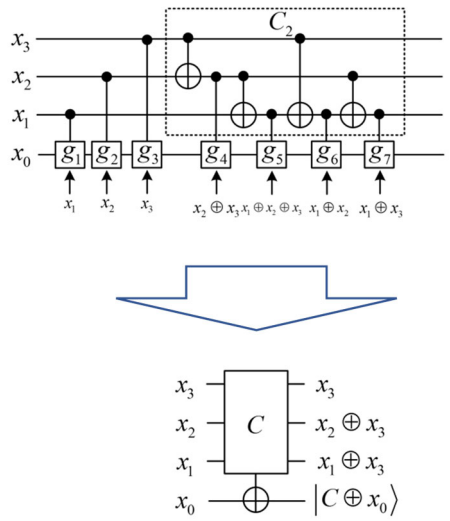



**FIGURE 20.** Another framework for the implementation of new 4-qubit logic gates using $CG_4/CG_4^\dagger$ and CNOT gates.

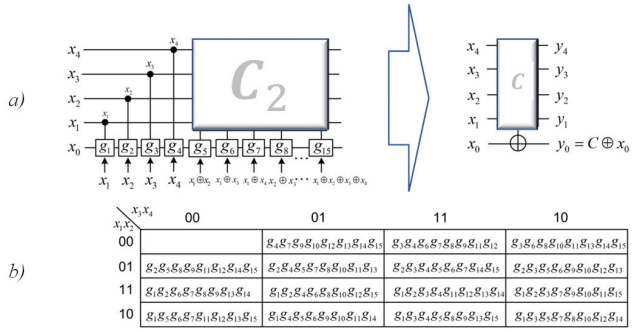



**FIGURE 21.** a) Framework for implementing a novel 5-qubit logic gate using $CG_8/CG_8^\dagger$ and CNOT gates. b) Symbolic graphical representation of the circuit from (a) in a QMap, utilizing controlled J and controlled $J^+$ gates.

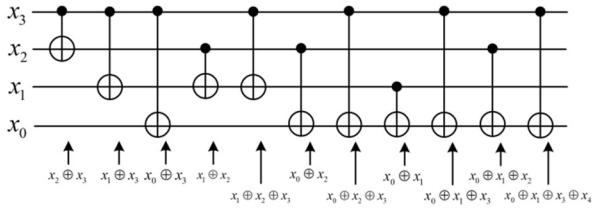



**FIGURE 22.** Circuit $C_2$ implementation, designed for inclusion in Fig. 21, optimized to use the minimum number of CNOT gates.

Fig. 21, $g_i \in \left\{J, J^\dagger\right\}, i \in 1, 2, , \ldots, 15$, and there are $2^{15}$ combinations, but in Table 7, there are only five combinations that meet the strict requirements of Assumption 2. From (13), $n = \left(\log_2 k + 1\right)|_{k=8} = 3 + 1 = 4$, so this new gate has $(n+1)|_{n=4} = 5$ lines.

*DESIGN EXAMPLE: Extension of a standard Toffoli gate*

**TABLE 7.** 5-bit logic gates in Fig 21.

| No. | $x_1x_2x_3\cdots x_{15}$ | $(x_1x_2x_3x_4)_2$ | | | | | | | | | | | | | | $C$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| 1 | $JJJ^\dagger J^\dagger J^\dagger J^\dagger JJ^\dagger J^\dagger JJJ$ | $I$ | $I$ | $I$ | $N$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $\bar{x}_1x_2\bar{x}_3\bar{x}_4$ |
| 2 | $JJ^\dagger JJ^\dagger JJ^\dagger JJ^\dagger J^\dagger JJ$ | $I$ | $I$ | $I$ | $I$ | $N$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $\bar{x}_1x_2\bar{x}_3x_4$ |
| 3 | $JJ^\dagger J^\dagger J^\dagger JJ^\dagger J^\dagger JJ^\dagger J^\dagger JJ$ | $I$ | $I$ | $I$ | $I$ | $I$ | $N$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $\bar{x}_1x_2x_3\bar{x}_4$ |
| 4 | $JJ^\dagger J^\dagger JJJJ^\dagger J^\dagger JJJJ^\dagger J$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $N$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $x_1\bar{x}_2\bar{x}_3\bar{x}_4$ |
| 5 | $JJ^\dagger JJJJ^\dagger J^\dagger J JJ^\dagger J^\dagger J^\dagger JJ^\dagger J$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $I$ | $N$ | $I$ | $I$ | $I$ | $I$ | $x_1\bar{x}_2x_3x_4$ |

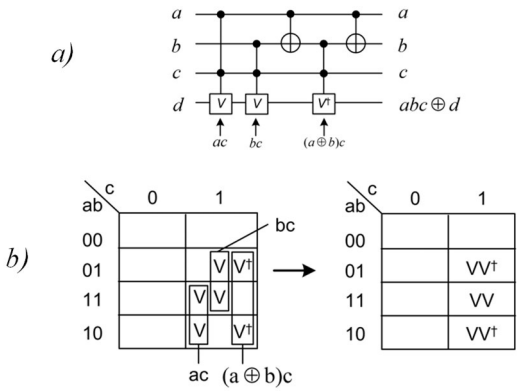



**FIGURE 23.** a) Extension of a standard Toffoli gate to a 4 × 4 Toffoli gate by multiplying with signal c b) Using QKMAP to analyze the circuit from Fig. a).

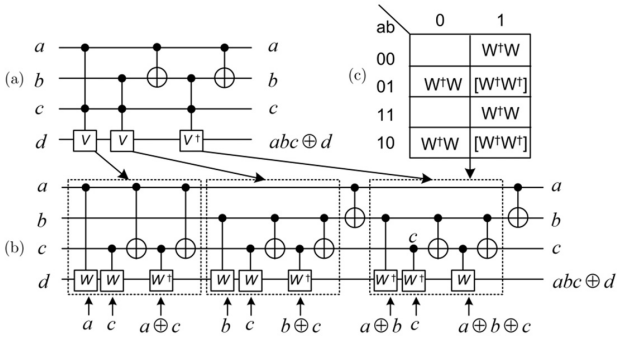



**FIGURE 24.** (a) Circuit from Fig. 23a repeated to help the reader in analysis of Fig. 23b, (b) Realization of the 4 × 4 Toffoli gate using $CW/CW^\dagger$ gates. Linear controls are explicitly indicated for all$W/W^\dagger$ gates, to facilitate analysis. The blocks indicate the gates of a 3 × 3 Toffoli gate with additional multiplication by c (from Fig. 23a), (c) QKMap-based analysis of circuit interrupt lines from the right of the circuit.

DESIGN EXAMPLE: Another *method for* 4*-bit gate construction using* $CW/CW^\dagger$ *and CNOT gates*

To implement the function $abc \oplus d$, each double-controlled $V/V^\dagger$ gate must be decomposed into $2 \times 2$ gates, as illustrated in Fig. 24. Similarly, controlled Hermitian gates (CCV) are constructed in Fig. 24 using $CW/CW^\dagger$ gates.

The circuit presented in Fig. 24 can be simplified into the circuit shown in Fig. 25 by applying quantum simplification rules. This transformation effectively reconstructs the CCC-NOT (triple-controlled NOT) circuit originally introduced by Barenco [54].

In Fig. 26, the $4 \times 4$ Toffoli gate is implemented using only seven $CW = CW^\dagger$ gates and six CNOT gates, whereas the final circuit in Fig. 26 requires seven $CW/CW^\dagger$ gates and eight CNOT gates. Despite this, the method shown in Fig. 26

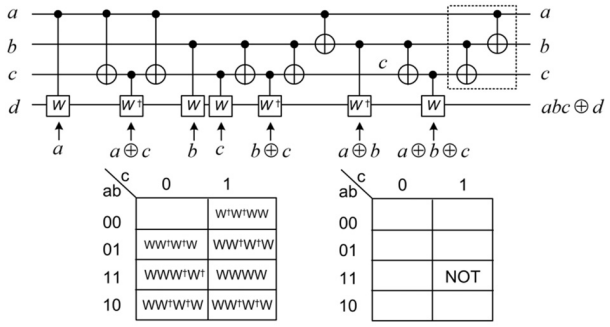



**FIGURE 25.** Simplified circuit derived from Figs. 47 and 48. The identity $WW^\dagger = I$ was applied to gates $W/W^\dagger$ controlled by c in Fig. 24, allowing the removal of two gates from the original configuration. The resulting circuit contains only six controlled $W/W^\dagger$ gates, each governed by a linear control function—demonstrating an affine root of a NOT gate. The final implementation includes seven $CW/CW^\dagger$ gates and eight CNOT gates. A Peres-like gate variant, shown in the top-right of this figure, can be realized with seven $CW/CW^\dagger$ gates and only six CNOT gates by omitting two CNOTs from the circuit.

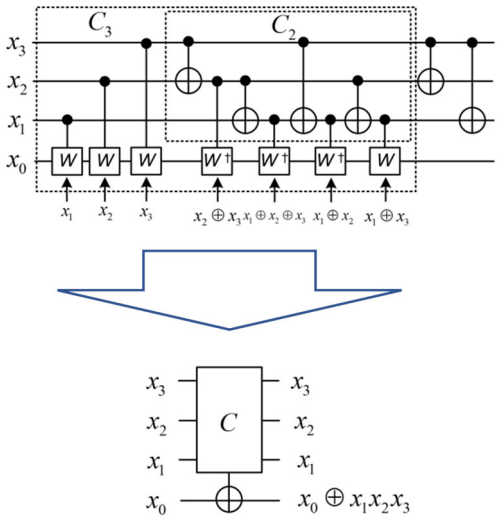



**FIGURE 26.** The new 4 × 4 Toffoli gate, as implemented in Fig. 20, utilizes $CW/CW^\dagger$ and CNOT gates.

is simpler than the one presented in Fig. 14 and enables more straightforward construction of new and larger quantum logic gates. For example, constructing a $C_3$ gate using a cascade of one Toffoli-4 gate and two CNOT gates results in a total cost of 13+1+1 = 15. In contrast, the method introduced in this section reduces the cost to just 11. Therefore, employing these new quantum logic gates can significantly lower the overall cost of many quantum circuits.

Fig. 27 demonstrates that strategic placement of CNOT gates can reduce their overall count, building upon the framework established in Fig. 28. However, incorporating the framework from Fig. 17 may result in a higher overall quantum cost. Examples of quantum IoT applications are shown in Table 8.

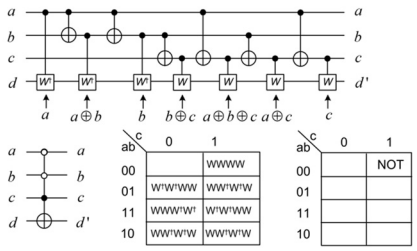



**FIGURE 27.** Realization of function $\sim a \sim bc \oplus d$ with seven CG gates and only six CNOT gates.

**TABLE 8.** Examples of quantum IoT applications.

| Use Case / Application | Quantum Technology Used | IoT Element | Purpose / Benefit | Example / Scenario | Ref |
|---|---|---|---|---|---|
| Secure Smart Grid Communication | Quantum Key Distribution (QKD) | Smart meters, sensors | Ensures unbreakable encryption between grid elements | QKD between power substations and control centers | [76], [77] |
| Quantum-enhanced Sensor Networks | Quantum Entanglement | Distributed sensors | Higher precision sensing and time synchronization | Seismic activity monitoring with entangled sensors | [78] |
| Quantum GPS for IoT Devices | Quantum Clocks, Quantum Interference | IoT-based navigation systems | Ultra-accurate geolocation without satellite reliance | Submarine or indoor navigation using quantum clocks | [79], [80] |
| Healthcare IoT (Q-IoMT) | Quantum Machine Learning (QML) | Wearables, implantables | Accelerated pattern recognition in biomedical data | Real-time ECG anomaly detection using quantum ML | [81], [82] |
| Smart City Infrastructure | QKD, Quantum Sensors | Traffic cameras, air quality sensors | Secure data sharing and enhanced environmental sensing | Pollution monitoring with quantum magnetometers | [83], [84] |
| Quantum Edge Computing for IoT | Quantum Annealing, QML | Edge IoT devices | Faster optimization and decision-making at the edge | Real-time delivery route optimization using QML | [85], [86] |
| Supply Chain and Logistics | Quantum Cryptography, Quantum Sensors | RFID, GPS trackers | Secure data transmission and tamper-evidence | Cold-chain pharma tracking with quantum sensors | [87] |
| Autonomous Vehicles | QML, Quantum Sensing | Lidar, radar, V2X modules | Better perception, secure vehicle-to-everything communication | Quantum-enhanced object detection in self-driving cars | [88], [89] |
| Quantum-Enabled Industrial IoT | Quantum Sensing, QML | Industrial machines, robots | High-precision process control and fault detection | Monitoring vibrations in turbines with quantum sensors | [90] |
| Agriculture and Environment | Quantum Sensing | Soil, water, climate sensors | Extremely sensitive detection of environmental changes | Soil nutrient detection using quantum spectroscopy | [91], [92] |

### C. SUMMARY

In summary, this section presents an in-depth exploration of the work on exact latency minimization techniques for quantum circuits, emphasizing the synthesis of reversible and permutative quantum gates with minimal quantum cost. The primary objective is to replace computationally expensive non-permutative quantum gates (NPQGs)—such as the controlled-$\sqrt{\text{NOT}}$ (CV = CV$^\dagger$)—with functionally equivalent configurations composed exclusively of permutative quantum gates (PQGs) built from CNOT and controlled-NOT$^{1/k}$ gates for ($k$ = 2, 4, 8,..).

The section begins by discussing the work distinguishing PQGs from NPQGs through their matrix representations, introducing "magic permutation matrices" that preserve the identity component along the main diagonal. It then builds a rigorous mathematical framework for defining and composing families of gates such as $G_{2^n}$, $G_{2^n}^\dagger$, and their controlled variants (CV, CW, CJ, etc.), each corresponding to successive roots of the NOT operation. Using this foundation, we discuss the work developing an optimal gate library that satisfies strict assumptions of *reversibility*, *indivisibility*, *minimality*, and *equivalence*—ensuring that every gate is both cost-optimal and re-usable across multiple circuit designs. Several theorems (e.g., Theorems 3 and 4) formalize

the synthesis rules governing the construction of n + 1–qubit gates from control lines and single-target configurations. The design examples (4-bit and 5-bit circuits using CW/CW$^\dagger$ or CJ/CJ$^\dagger$ gates) demonstrate that exact synthesis under this model can yield circuits with substantially reduced quantum cost compared to traditional decompositions of Toffoli or Peres gates. The section culminates with multiple design frameworks (Figs. 17–27) and shows that optimized control logic (the C2 subcircuit) can be implemented with the minimum number of CNOT gates, enabling systematic minimization of both gate count and cost.

*Critical Evaluation of Methods and Contributions*

*Theoretical Foundations and Generality:* The section surveys the work providing a rigorous mathematical treatment of quantum gate decomposition, notably through parametric gate families $G_{2^n}$ and their adjoints. This hierarchical structure unifies previously disparate gate definitions (V, W, J, etc.) into a single algebraic framework, offering a scalable path for constructing higher-order gates. *Strengths: a*) The surveyed work establishes a general recursive definition of quantum roots of NOT, extending prior work by Barenco et al. [54] into a formalized gate family. *b*) Clearly defines synthesis rules and logical constraints (Assumptions 1–3) ensuring the resulting gate libraries remain both reversible and physically realizable. *c*) Introduces a structured binary-to-quantum mapping that enables efficient reduction of NPQGs into PQG-based circuits. *Limitations: a*) The framework, while elegant, is theoretical and algebraic, with no empirical evaluation on hardware or simulation platforms which is delegated to the work on hardware. *b*) The formalism presumes idealized unitary operations and neglects decoherence, gate error rates, and timing mismatches that influence real quantum implementations. *c*) While the recursive $G_{2^n}$ approach is powerful, its computational scalability beyond small n (e.g., $> 5$ qubits) remains unquantified.

*Gate Library Optimization and Minimality Criteria*

The explicit criteria for optimality and minimality (Assumptions 1–2) represent a significant step toward a provably minimal synthesis methodology. Each gate in the proposed library is required to be both cost-optimal and indivisible, ensuring reuse and avoiding redundant decompositions. *Strengths: a*) The formal constraints yield a consistent and self-contained synthesis framework that can be algorithmically verified. *b*) The concept of indivisibility ensures that every gate functions as an atomic unit of computation, facilitating composability in larger systems. *Limitations:* a) The cost model focuses exclusively on gate count, overlooking execution latency, hardware parallelism, and energy–error trade-offs. b) Equivalence is defined algebraically rather than resource-equivalently, meaning some ‘‘optimal’’ gates may be optimal only under an abstract cost metric, not under hardware-specific metrics such as two-qubit gate fidelity or routing complexity. Most of these problems should be addressed in a hardware-aware cost model (e.g., latency, parallelism, fidelity, and routing constraints).

*Comparison with Previous Synthesis Approaches*

Compared to early methods by Barenco et al. [54], Shende et al. [56], and later NCV-based techniques [59], [60], [61], [62], [63], the approach discussed in this section shifts from generic decomposition to structural optimization through parameterized root gates.

| Method | Core Idea | Complexity / Scalability | Strengths | Limitations |
|---|---|---|---|---|
| Barenco et al. (1995) | Decomposition of MCT gates using V/W primitives | Polynomial | Foundational framework for controlled-U synthesis | High gate cost for large controls |
| Shende et al. (2004) | Analytic decomposition of n-controlled U using CNOTs and single-qubit gates | Exponential / polynomial forms | Systematic and general | Heavy resource use for large n |
| NCV Library Methods [61–65] | Minimization using NOT, CNOT, CV | Limited to 3-qubit circuits | Compact optimal synthesis | Restricted scalability |
| Proposed $G_{2^n}$ Framework | Recursive construction using controlled roots of NOT | Polynomial up to 5 qubits | Unified gate family, improved minimal cost | Lacks hardware-aware and fault-tolerant validation |

The new approach thus outperforms prior NCV-based methods in speed ($\approx 127\times$) and cost efficiency, particularly for 3–5 qubit circuits. However, it remains untested for larger systems, where memory and synthesis time scale exponentially.

*Design Frameworks and Practical Implications*

The illustrative frameworks (Figs. 17–27) effectively demonstrate stepwise logic gate construction using minimal CNOT structures. The design examples of 4-qubit (CW/CW$^\dagger$) and 5-qubit (CJ/CJ$^\dagger$) gates showcase measurable savings: for instance, a $4 \times 4$ Toffoli gate implemented using CW/CW$^\dagger$ and CNOT gates reduces the total cost from 15 to 11. *Strengths: a*) The surveyed work demonstrates how the same algebraic principles generalize across different control depths and qubit counts. *b*) Shows that circuit simplification via quantum Karnaugh maps (QMaps ) and gate-pair cancellation can reduce redundancy and optimize control signal usage. *Limitations: a*) No exploration of error propagation or noise sensitivity for deeper root gates (e.g., J, W). These gates may amplify phase errors due to fractional exponentiation of unitaries. *b*) Lack of comparative evaluation against modern compiler optimizations (e.g., t-parallelization or ZX-calculus simplifications). *c*) The synthesis remains isolated from fault-tolerant frameworks, especially relevant since small differences in decomposition can lead to large changes in T-gate overhead in practical fault-tolerant implementations.

*Comparative Assessment of Design Strategies*

| Design Dimension | Strengths (Proposed Work) | Limitations / Gaps |
|---|---|---|
| **Mathematical Formalism** | Rigorous, recursive definition of controlled roots; unifies multiple gates (V, W, J). | No scalability analysis or error modeling. |
| **Gate Library Optimality** | Enforces indivisibility and minimal cost; reusability across designs. | Ignores fidelity and hardware constraints. |
| **Framework Efficiency** | Systematic method to generate all control values with minimal CNOT use. | Complexity grows rapidly with n; lacks empirical validation. |
| **Comparison with Prior Art** | Outperforms NCV and MCT decompositions in synthesis speed and gate count. | Comparison limited to small ($\leq 5$ qubit) circuits. |
| **Implementation in Quantum IoT** | Conceptually supports efficient and compact control logic. | Missing latency–energy integration and IoT-level performance metrics. |

*Recommendations for Future Work*

*1. Hardware-Constrained Optimization:* Extend the theoretical synthesis to consider hardware coupling maps, SWAP networks, and calibration errors, enabling physically realizable minimal circuits. *2. Integration with Fault-Tolerant Models:* Evaluate how root-of-NOT decompositions translate into T-gate counts and magic-state distillation costs, ensuring practical deployability in error-corrected systems. 3. Automated Gate Library Synthesis: Develop a compiler-based library generator that automatically identifies indivisible gate sets under given hardware constraints or optimization goals (latency, fidelity, or cost). *4. Scalability Studies Beyond 5 Qubits:* Perform computational experiments for 6–8 qubit systems to determine scaling limits and potential memory optimization strategies in exact synthesis. *5. Quantum IoT System Integration:* Couple gate-level cost models with network-level latency and energy metrics, particularly relevant for edge-deployed quantum control units.

*Overall Assessment of the Section*

This section surveys the work on a theoretically robust and methodologically significant contribution to the field of exact quantum circuit minimization. The selected work advances prior NCV and Barenco-style approaches by introducing a recursive and unified gate family capable of producing provably minimal reversible circuits. The resulting framework not only achieves tangible quantum-cost reductions for small circuits but also offers a scalable conceptual model for gate synthesis. However, the work remains largely analytical, lacking experimental validation, hardware benchmarking, and cross-layer integration with quantum IoT architectures. Bridging these gaps—through simulation, compiler integration, and hardware-aware modeling—will be crucial for transforming the elegant algebraic formalism of this section into a practical synthesis methodology for next-generation quantum systems.

## IV. CONCLUSION

As a summary of the main results in the design of the circuits for low latency quantum networks, we should keep in mind:

*Lessons from conventional circuit design*:

*Energy Savings: Reversible Gates: NOT, SWAP, and CNOT:* One promising strategy to mitigate the thermal energy loss associated with irreversible gates is to transition towards reversible logic gates in chip design. In a reversible logic gate, each output is uniquely associated with a corresponding input, and vice versa. This ensures that no information is lost during the computation, making it possible to reverse the computation after obtaining a result.

*Universal Reversible Gates: FREDKIN and TOFFOLI:* Just as there are universal gates for classical irreversible computing, such as the NAND gate, there also exist universal gates for classical reversible computing. However, the smallest gates that are both reversible and universal require three inputs and three outputs. Two prominent examples of such gates are the FREDKIN gate (also known as the controlled-SWAP gate) and the TOFFOLI gate (also known as the controlled-CNOT gate).

*Universal Reversible Basis: NOT–CNOT–TOFFOLI:* In the study of reversible circuit synthesis, we commonly work over the NOT–CNOT–TOFFOLI gate set, which serves as a standard universal basis for reversible computation.

*Quantum Logic Gates Library*

Any classical computation can be decomposed into a series of logic gates, each operating on a small number of classical bits. Similarly, quantum computations can be constructed from a sequence of quantum gates, each acting on a limited number of qubits. The key distinction lies in their capabilities: classical gates handle definite bit values, either 0 or 1, while quantum gates manipulate complex quantum states, which may be superpositions of basic computational states and can also exhibit entanglement. As a result, the range and nature of quantum logic gates are significantly broader than those in classical systems.

Since a quantum gate must be realized through the physical evolution of an isolated quantum system, its operation is dictated by the Schrödinger equation: $i\hbar\partial|\psi\rangle/\partial t = H|\psi\rangle$ where $H$, the Hamiltonian, defines the system's internal forces and external fields. Consequently, the unitary operator representing a quantum gate corresponds to the system's time evolution and is given by: $U = exp\,(-iHt/\hbar)$. In this expression, the Hamiltonian $H$ characterizes the interactions responsible for driving the quantum transformation.

We have discussed the main characteristics and construction of:

*Latency optimal quantum circuit:*

*The gate set* $\{H, P, P^\dagger,$ *CNOT,* $T, T^\dagger\}$ *is universal for quantum computation, capable of approximating any unitary transformation to arbitrary precision, up to a global phase. 1)Meet-in-the-Middle Search Algorithm 2) Latency-optimal implementations: various 2, 3 and 4 qubit logical gates using the gate set* $\{H, P, P^\dagger,$ *CNOT,* $T, T^\dagger\}$.

*Optimization gains:*

*Controlled-T gate (Fig. 9):T gates achieve compression factor 15/9, CNOT gates 16 / 12, T-depth reduced 9/5.*

*1-bit full adder (Fig. 10):T gates achieve compression factor 14 / 8, CNOT gates 12/ 10, H gates 2, T-depth 4.*

*Exact minimization of quantum circuits:*

Two fundamental methods were proposed for synthesizing optimal reversible logic circuits using quantum gates: a) Direct synthesis using quantum gates to construct the circuit. b) A two-stage synthesis process, where: First, a set of new logic gates is designed using quantum gates. Then, these newly defined logic gates are used to synthesize the target circuit.

Between the two, the second method is significantly more efficient from an algorithmic standpoint. This is because it reduces the problem to a binary logic synthesis task, which is computationally simpler. Additionally, the newly defined logic gates need to be synthesized only once, after which they can be reused across multiple circuits.

We believe that the presented options for quantum circuit design will help the network designers when choosing the circuit for their project. The above topics have been covered partly in existing literature and to the best of our knowledge this is the first unifying survey that includes most of the relevant problems when it comes to planning and designing a quantum network with minimum latency which is of paramount importance for future IoT networks. The comparison of the contribution of our paper with the existing work is summarized in Table 9. The paper establishes a direct relation between the quality of the circuit and the overall network performance.

**TABLE 9.** Topic coverage in existing work.

| *REFERENCES* | QUANTUM GATES LIBRARY | LATENCY-OPTIMAL QUANTUM CIRCUITS | EXACT MINIMIZATION OF QUANTUM CIRCUITS | DECOMPOSING CV OPERATIONS INTO A UNIVERSAL GATE LIBRARY | NETWORK LATENCY | CIRCUIT/NETWORK LEVEL PERFORMANCE | COVERAGE OF THE RELEVANT TOPICS |
|---|---|---|---|---|---|---|---|
| [1-25] | | | | | # | | 15% |
| [26-33] [65-75] | # | | | | | | 15% |
| [34-52] | | # | | | | | 15% |
| [53-63] | | | # | | | | 15% |
| [64] | | | | # | | | 15% |
| our paper | # | # | # | # | # | # | 100% |

*Analytical Summary and Critical Evaluation*

Here we effectively consolidate the survey's core findings on latency-optimal quantum circuit design, linking reversible computation principles with contemporary quantum logic synthesis. It reaffirms that lessons from classical reversible logic—notably the conservation of information and energy through NOT, CNOT, and SWAP gates—serve as the foundational paradigm for quantum circuit minimalism. By drawing conceptual parallels between reversible classical gates (TOFFOLI, FREDKIN) and their quantum analogues within the universal gate set {H, P, $P^\dagger$, CNOT, T, $T^\dagger$}, the surveyed work highlights the continuity between low-energy computation and quantum coherence preservation.

*Strengths and Key Insights:* 1) The survey provides a coherent narrative linking energy-efficient reversible computing with quantum circuit design, reinforcing the thermodynamic rationale for reversibility. 2) The discussion on the gate universality basis (NOT–CNOT–TOFFOLI and its quantum extensions) underlines the essential building blocks for latency reduction in quantum networks. 3) By referencing specific optimization gains—such as T-gate compression ratios and T-depth reductions for Controlled-T and full adder circuits—the survey demonstrates tangible performance benefits that stem from algorithmic and structural refinements. 4) It clarifies the dual synthesis strategies (direct versus two-stage methods), offering a clear argument for why the two-stage approach is computationally superior in practice. 5) The conclusion also frames the contribution of the survey as a unifying treatment that bridges reversible logic, gate-level optimization, and network-level latency analysis, positioning the work as a reference point for future IoT-oriented quantum architectures.

*Critical Assessment and Gaps:* While the survey successfully integrates the paper's technical themes, several limitations remain evident. We argue that these limitations should be discussed within the hardware design umbrella and provide complementary/ additional solutions. These problems include: *1) Discussion on hardware and physical layer integration:* Although the survey identifies the theoretical gate-level optimizations, it does not yet translate these improvements into measurable latency savings on real quantum hardware (e.g., superconducting or photonic platforms). The link between minimized quantum cost and end-to-end latency in IoT network contexts is conceptually stated but not empirically validated. 2) Absence of fault-tolerance and noise considerations: Reversible gate synthesis and latency optimization are discussed at the abstract level. However, error propagation, decoherence, and gate fidelity trade-offs—which critically affect latency in fault-tolerant circuits—are not yet incorporated into the analysis. *3) Incomplete Cross-Layer Optimization:* The conclusion summarizes gate-level advances but does not extend the discussion to quantum network-level scheduling, routing, and entanglement distribution, where latency becomes systemically constrained by communication overhead and synchronization delays. *4) Need for quantitative benchmarking:* While examples like the Contr.-T and 1-bit adder show compression factors, broader benchmarking across circuit families (e.g., arithmetic, communication, or cryptographic circuits) would solidify claims of latency optimality.

Below we summarize achievements of the paper and remaining challenges.

| Aspect | Achievements in the Paper | Remaining Challenges / Opportunities |
|---|---|---|
| **Theoretical Foundation** | Establishes a unified reversible–quantum synthesis perspective. | Needs formal proof of latency–energy equivalence at system level. |
| **Optimization Strategy** | Demonstrates gate-level cost and depth reductions using standard universal gate sets. | Missing fault-tolerant metrics (T-count, Clifford+T overhead). |
| **Circuit Design Impact** | Introduces latency-optimal building blocks for arithmetic and control circuits. | Unclear scaling behavior for multi-qubit and multi-node networks. |
| **Network-Level Integration** | Conceptually connects circuit design to IoT latency constraints. | Lacks simulation or analytical latency model for quantum IoT. |

*Future Research Directions*

Building on the insights of this conclusion, future work should focus on:

*1. Hardware-Constrained Latency Modeling:* Integrate the proposed gate-level optimizations with hardware-specific characteristics (connectivity graphs, qubit coherence times, gate parallelism) to achieve practical latency predictions. 2. *Fault-Tolerant and Error-Corrected Design:* Extend reversible synthesis frameworks to fault-tolerant logical gates and evaluate latency trade-offs under error-corrected implementations. 3) *Cross-Layer Cooptimization:* Develop methodologies that couple gate synthesis, circuit routing, and quantum network scheduling, ensuring consistent latency minimization from gate to protocol level. 4. *Automated Toolchains:* Design compilers and synthesis tools capable of automatically generating latency-optimal circuits under specific IoT deployment constraints. 5) *Experimental Validation:* Implement representative circuits on existing quantum platforms (IBM Q, Rigetti, IonQ) to benchmark real-world latency gains.

*Overall Evaluation:* The surveyed work establishes a strong conceptual bridge between reversible computing principles and latency-optimal quantum circuit synthesis, reaffirming the importance of energy-efficient, reversible design paradigms for future quantum IoT systems. Its forward-looking value lies primarily in identifying research frontiers—the translation of algebraic circuit minimality into hardware-aware, latency-measurable, and fault-tolerant implementations. Addressing these dimensions will be pivotal for evolving the presented framework from a theoretical foundation into a practical blueprint for quantum network design. This part may be shared between circuit designers once a clear requests are set up by the network planners.

## REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[2] A. W. Cross, A. Javadi-Abhari, T. Alexander, N. de Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, "OpenQASM 3," 2023, *arXiv:2304.05219*.

[3] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. STOC*, 1996, pp. 212–219.

[5] IBM Quantum. *Quantum System One*. Accessed: Jan. 17, 2026. [Online]. Available: https://www.ibm.com/quantum

[6] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver," *Nature Commun.*, vol. 5, no. 4213, 2014, Art. no. 4213.

[7] C. Gidney and M. Ekerå, "How to factor 2048-bit RSA in 8 hours," *Quantum*, vol. 5, p. 433, Jun. 2021.

[8] Google Quantum AI. *The Sycamore Processor*. Accessed: Jan. 17, 2026. [Online]. Available: https://quantumai.google

[9] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, "Superconducting quantum circuits at the surface code threshold," *Nature*, vol. 508, no. 7497, pp. 500–503, 2014.

[10] M. Frank, "The future of computing depends on making it reversible," *IEEE Spectr.*, vol. 55, no. 7, pp. 35–41, Jul. 2018.

[11] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming*. Cham, Switzerland: Springer, 1980, pp. 632–644.

[12] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Develop.*, vol. 5, no. 3, pp. 183–191, Jul. 1961.

[13] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Develop.*, vol. 17, no. 6, pp. 525–532, Nov. 1973.

[14] D. Maslov. *Reversible Logic Synthesis Benchmarks Page*. Accessed: Jan. 17, 2026. [Online]. Available: https://web.archive.org/web/20201203143617/http://webhome.cs.uvic.ca/

[15] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 818–830, Jun. 2013.

[16] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, "Quantum chemistry in the age of quantum computing," *Chem. Rev.*, vol. 119, no. 19, pp. 10856–10915, 2019.

[17] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*.

[18] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Proc. DAC*, Sep. 2009, pp. 270–275.

[19] V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 710–722, Jun. 2003.

[20] A. S. Shafigh, B. Lorenzo, S. Glisic, and Y. Fang, "Low-latency robust computing vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 2, pp. 2130–2144, Feb. 2023.

[21] I. Sugathapala, S. Glisic, M. Juntti, A. S. Shafigh, and L.-N. Tran, "Queue aware resource optimization in latency constrained dynamic networks," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–6.

[22] I. Kovacevic, A. S. Shafigh, S. Glisic, B. Lorenzo, and E. Hossain, "Multi-domain network slicing with latency equalization," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2182–2196, Dec. 2020.

[23] I. Kovacevic, E. Harjula, S. Glisic, B. Lorenzo, and M. Ylianttila, "Cloud and edge computation offloading for latency limited services," *IEEE Access*, vol. 9, pp. 55764–55776, 2021.

[24] C. Li, J. Li, M. Peng, B. Rasti, P. Duan, X. Tang, and X. Ma, "Low-latency neural network for efficient hyperspectral image classification," *IEEE J. Select. Topics Appl. Earth Observ. Remote Sens.*, vol. 18, pp. 7374–7390, 2025.

[25] D. Zenati, T. Maimon, and K. Cohen, "RRO: A regularized routing optimization algorithm for enhanced throughput and low latency with efficient complexity," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 2, pp. 437–447, Feb. 2025.

[26] G. Le, V. T. Hoang, S. Ferdousi, A. Marotta, S. Xu, Y. Hirota, Y. Awaji, M. Tornatore, and B. Mukherjee, "Reliable provisioning of low-latency and high-bandwidth extended reality live streams," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 5, pp. 1755–1766, May 2025.

[27] D. Li, J. Li, D. Niyato, W. Feng, and W. Jiang, "Deep energy-efficient optimization network for URLLC over cell-free massive MIMO," *IEEE Internet Things J.*, vol. 12, no. 12, pp. 20973–20987, Jun. 2025.

[28] D. Townend, S. D. Walker, N. Parkin, and A. Tukmanov, "C-RAN and optical fronthaul latency in representative network topologies," *IEEE Open J. Commun. Soc.*, vol. 6, pp. 1438–1445, 2025.
[29] A. Belli, M. Esposito, S. Raggiunto, L. Palma, and P. Pierleoni, "Relaying mechanisms in BLE mesh networks: A method for improving latency and reliability," *IEEE Internet Things J.*, vol. 12, no. 12, pp. 22282–22297, Jun. 2025.
[30] P. Qin, Q. Li, and D. Xu, "Decentralized federated learning in LEO satellite-based IoE communications: Latency optimization under reliability constraints," *IEEE Internet Things J.*, vol. 13, no. 1, pp. 24–38, Jan. 2025.
[31] Z. Niu, H. Yang, Q. Yao, B. Wu, S. Yin, S. Shen, B. Wei, J. Zhang, and A. V. Vasilakos, "Reliable low-latency routing for VLEO satellite optical network: A multiagent reinforcement learning approach," *IEEE Internet Things J.*, vol. 12, no. 3, pp. 2309–2321, Feb. 2025.
[32] A. U. Haq, S. S. Sefati, S. J. Nawaz, A. Mihovska, and M. J. Beliatis, "Need of UAVs and physical layer security in next-generation non-terrestrial wireless networks: Potential challenges and open issues," *IEEE Open J. Veh. Technol.*, vol. 6, pp. 554–595, 2025.
[33] M. Polverini, A. Cianfrani, T. Caiazzi, and M. Scazzariello, "SRv6 meets DetNet: A new behavior for low latency and high reliability," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 2, pp. 448–458, Feb. 2025.
[34] M. Amy, D. Maslov, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," 2012, *arXiv:1206.0758*.
[35] O. Golubitsky and D. Maslov, "A study of optimal 4-bit reversible Toffoli circuits and their synthesis," *IEEE Trans. Comput.*, vol. 61, no. 9, pp. 1341–1353, Sep. 2012.
[36] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 9, pp. 1652–1663, Sep. 2006.
[37] D. Maslov and D. M. Miller, "Comparison of the cost metrics for reversible and quantum logic synthesis," *IET Comput. Digit. Techn.*, vol. 1, no. 2, pp. 98–104, 2005.
[38] C. M. Dawson and M. A. Nielsen, "The Solovay–Kitaev algorithm," *Quantum Inf. Comput.*, vol. 6, no. 1, pp. 81–95, Jan. 2006.
[39] M. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[40] A. Bocharov and K. M. Svore, "A depth-optimal canonical form for single-qubit quantum circuits," 2012, *arXiv:1206.3223*.
[41] V. Kliuchnikov, D. Maslov, and M. Mosca, "Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and t gates," 2012, *arXiv:1206.5236*.
[42] A. G. Fowler, "Constructing arbitrary steane code single logical qubit fault-tolerant gates," *Quantum Inf. Comput.*, vol. 11, no. 9, pp. 867–873, Sep. 2011.
[43] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 code," *Quantum Inf. Comput.*, vol. 6, no. 2, pp. 97–165, Mar. 2006.
[44] X. Zhou, D. W. Leung, and I. L. Chuang, "Methodology for quantum logic gate construction," *Phys. Rev. A, Gen. Phys.*, vol. 62, no. 5, Oct. 2000, Art. no. 052316.
[45] A. G. Fowler, A. M. Stephens, and P. Groszkowski, "High-threshold universal quantum computation on the surface code," *Phys. Rev. A, Gen. Phys.*, vol. 80, no. 5, Nov. 2009, Art. no. 052312.
[46] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A, Gen. Phys.*, vol. 70, no. 5, Nov. 2004, Art. no. 052328.
[47] S. Sen and R. E. Tarjan, "Deletion without rebalancing in balanced binary trees," in *Proc. 21st Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2010, pp. 1490–1499.
[48] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by a quantum walk," in *Proc. 35th ACM Symp. Theory Comput.*, 2003, pp. 59–68.
[49] A. Peres, "Reversible logic and quantum computers," *Phys. Rev. A, Gen. Phys.*, vol. 32, no. 6, pp. 3266–3276, Dec. 1985.
[50] R. P. Feynman, "Quantum mechanical computers," *Found. Phys.*, vol. 16, no. 6, pp. 507–531, 1986.
[51] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*. London, U.K.: Oxford Univ. Press, 2007.
[52] K. N. Patel, I. L. Markov, and J. P. Hayes, "Efficient synthesis of linear reversible circuits," 2003, *arXiv:quant-ph/0302002*.
[53] Z. Li, X. Song, M. Perkowski, and H.-W. Chen, "Realization of a new permutative gate library using controlled-kth-root-of-NOT quantum gates for exact minimization of quantum circuits," *Int. J. Quantum Inf.*, vol. 12, no. 5, Aug. 2014, Art. no. 1450034.
[54] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.
[55] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffoli gates," in *Proc. 41st IEEE Int. Symp. Multiple-Valued Log.*, May 2011, pp. 288–293.
[56] Y. Liu, G. L. Long, and Y. Sun, "Universal quantum circuit for multiqubit controlled gates," *Int. J. Quantum Inf.*, vol. 6, no. 3, pp. 447–457, 2008.
[57] E. Tsai and M. Perkowski, "Synthesis of permutative quantum circuits with Toffoli and TISC gates," in *Proc. IEEE 42nd Int. Symp. Multiple-Valued Log.*, May 2012, pp. 50–55.
[58] Z. Sasanian and D. M. Miller, "Transforming MCT circuits to NCVW circuits," in *Proc. Workshop Reversible Comput.*, Mar. 2012, pp. 77–88.
[59] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Synthesis of reversible logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 9, pp. 1652–1660, Sep. 2006.
[60] G. Yang, W. N. N. Hung, X. Song, and M. Perkowski, "Exact synthesis of 3-qubit quantum circuits from non-binary quantum gates using multiple-valued logic and group theory," in *Proc. Design, Autom. Test Eur.*, 2005, pp. 434–439.
[61] Z. Li, H. Chen, W. Liu, X. Xue, and F. Xiao, "Reversible logic circuit synthesis based on quantum cost," *Acta Electronica Sinica*, vol. 41, no. 4, pp. 690–695, 2013.
[62] D. Maslov and D. M. Miller, "Multiple-valued multiple-controlled Toffoli gates," *IET Comput. Digital Techn.*, vol. 1, no. 2, pp. 98–106, 2007.
[63] G. W. Yang, X. Song, M. Perkowski, W. N. N. Hung, J. Biamonte, and Z. Tang, "Reversible logic circuit synthesis using genetic algorithm," *IET Comput. Digit. Techn.*, vol. 1, no. 4, pp. 382–391, 2007.
[64] T. Kalajdzievski and J. M. Arrazola, "Exact gate decompositions for photonic quantum computing," 2018, *arXiv:1811.10651*.
[65] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*, 2nd ed., Reading, MA, USA: Addison-Wesley, 2011.
[66] W. Magnus, "On the exponential solution of differential equations for a linear operator," *Commun. Pure Appl. Math.*, vol. 7, no. 4, pp. 649–673, Nov. 1954.
[67] M. Suzuki, "Generalized Trotter's formula and systematic approximants of exponential operators," *Commun. Math. Phys.*, vol. 51, pp. 183–190, Feb. 1976.
[68] S. Sefi and P. van Loock, "Maximally entangled states from Gaussian squeezing and displacement," *Phys. Rev. Lett.*, vol. 107, no. 17, 2011, Art. no. 170501.
[69] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, "Toward the first quantum simulation with quantum speedup," 2017, *arXiv:1711.10980*.
[70] C. Sparrow, "Simulating the vibrational quantum dynamics of molecules using a photonic quantum processor," *Nature*, vol. 557, pp. 660–666, May 2018.
[71] H.-K. Lau, R. Pooser, G. Siopsis, and C. Weedbrook, "Quantum chirp parameter estimation with Gaussian states," *Phys. Rev. Lett.*, vol. 118, no. 8, 2017, Art. no. 080501.
[72] J. M. Arrazola, T. Kalajdzievski, C. Weedbrook, and S. Lloyd, "Quantum algorithm for non-homogeneous linear partial differential equations," 2018, *arXiv:1809.02622*.
[73] T. Kalajdzievski, C. Weedbrook, and P. Rebentrost, "Quantum computation with continuous-variable cluster states," *Phys. Rev. A, Gen. Phys.*, vol. 97, May 2018, Art. no. 062311.
[74] T. Sowiński, O. Dutta, P. Hauke, L. Tagliacozzo, and M. Lewenstein, "Dipolar molecules in optical lattices: Quantum simulating rotational degrees of freedom," *Phys. Rev. Lett.*, vol. 108, Mar. 2012, Art. no. 115301.
[75] P. Rebentrost, B. Gupt, and T. R. Bromley, "Photonic quantum algorithm for Monte Carlo integration," 2018, *arXiv:1809.02579*.
[76] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden, "Advances in quantum cryptography," *Adv. Opt. Photon.*, vol. 12, pp. 1012–1236, May 2020.
[77] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, Oct. 2018, Art. no. eaam9288.

[78] P. Komar, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sorensen, J. Ye, and M. D. Lukin, "A quantum network of clocks," *Nat Phys.*, vol. 10, pp. 582–587, Mar. 2014.
[79] Y. Jiang, J. A. Sherman, and C. W. Oates, "Making optical atomic clocks portable with fiber links," *Nat Photon.*, vol. 5, pp. 158–161, Jul. 2011.
[80] P. Kómár, E. M. Kessler, J. Ye, and M. D. Lukin, "Quantum network for clock synchronization," *Nat Commun.*, vol. 7, p. 12413, Apr. 2016.
[81] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemp Phys.*, vol. 56, no. 2, pp. 172–185, 2015.
[82] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: A review of recent progress," *Rep. Prog. Phys.*, vol. 81, no. 7, Jul. 2018, Art. no. 074001.
[83] H. K. Lo, M. Curty, and K. Tamaki, "Secure quantum key distribution," *Nat Photonics.*, vol. 8, no. 8, pp. 595–604, 2014.
[84] J. M. Taylor, P. Cappellaro, L. Childress, L. Jiang, D. Budker, P. R. Hemmer, A. Yacoby, R. Walsworth, and M. D. Lukin, "High-sensitivity diamond magnetometer with nanoscale resolution," *Nature Phys.*, vol. 4, no. 10, pp. 810–816, Oct. 2008.
[85] M. Benedetti, "A generative modeling approach for benchmarking and training shallow quantum circuits," *NPJ Quantum Inf.*, vol. 5, p. 45, Feb. 2019.
[86] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.
[87] X. Zou, L. Qian, and X. Zhang, "Quantum secure Internet of Things and its applications in smart cities," *Future Gener. Comput. Syst.*, vol. 100, pp. 14–25, May 2019.
[88] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.
[89] L. Gyongyosi and S. Imre, "A survey on quantum computing technology," *Comput. Sci. Rev.*, vol. 31, pp. 51–71, Feb. 2019.
[90] C. L. Degen, F. Reinhard, and P. Cappellaro, "Quantum sensing," *Rev. Mod. Phys.*, vol. 89, no. 3, 2017, Art. no. 035002.
[91] M. F. Riedel, P. Böhi, Y. Li, T. W. Hänsch, A. Sinatra, and P. Treutlein, "Atom-chip-based generation of entanglement for quantum metrology," *Nature*, vol. 464, no. 7292, pp. 1170–1173, Apr. 2010.
[92] B. L. Higgins, D. W. Berry, S. D. Bartlett, H. M. Wiseman, and G. J. Pryde, "Entanglement-free heisenberg-limited phase estimation," *Nature*, vol. 450, no. 7168, pp. 393–396, Nov. 2007.

**SAVO GLISIC** (Senior Member, IEEE) was a Visiting Scientist with Cranfield Institute of Technology, Cranfield, U.K., from 1976 to 1977, and University of California, San Diego, from 1986 to 1987. He is currently with WPI, Worchester, MA, USA, and was with Oulu University, and INS Institute for Networking Sciences/Globalcom Oy. He has been active in the field of wireless communications and has published a number of papers and books. He has also published the latest book *Artificial Intelligence and Quantum Computing for Wireless Networks* (John Wiley and Sons, 2021) covers the enabling technologies for the definition, design, and analysis of incoming 6G/7G systems. His research interests include network optimization theory, artificial intelligence, block chain technology, cloud/edge/fog computing, networks information theory, network sciences, quantum channel information theory, and quantum computing enabled communications. He has served as the Technical Program Chair for the Third IEEE ISSSTA'94, the Eighth IEEE PIMRC'97, and IEEE ICC'01. He was the Director of IEEE ComSoc MD Programs.

**BEATRIZ LORENZO** (Senior Member, IEEE) received the M.Sc. degree in telecommunications engineering from the University of Vigo, Vigo, Spain, in 2008, and the Ph.D. degree from the University of Oulu, Oulu, Finland, in 2012. She is currently an Associate Professor and the Director of the Network Science Laboratory, Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA, USA. She has published more than 70 papers and co-authored two books on advanced wireless networks. Her research interests include quantum computing, AI for wireless networks, B5G and 6G network architectures and protocol design, mobile computing, optimization, and network economics.

Dr. Lorenzo was a recipient of the Fulbright Visiting Scholar Fellowship with the University of Florida, from 2016 to 2017. She served as an Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and IEEE TRANSACTIONS ON MOBILE COMPUTING. She was the General Co-Chair of WiMob Conference, in 2019, and serves regularly in the TPC of top IEEE and ACM conferences.